Research Article

Handwritten Bengali Alphabets, Compound Characters and Numerals Recognition Using CNN-based Approach

Md Asraful, Md. Anwar Hossain* and Ebrahim Hossen

Pabna University of Science and Technology, Pabna, Bangladesh
mdasrafulm333@gmail.com; manwar.ice@pust.ac.bd; ebrahim165577@gmail.com
*Correspondence: manwar.ice@pust.ac.bd

Received: 13th May 2023; Accepted: 19th June 2023; Published: 1st July 2023

Abstract: Accurately classifying user-independent handwritten Bengali characters and numerals presents a formidable challenge in their recognition. This task becomes more complicated due to the inclusion of numerous complex-shaped compound characters and the fact that different authors employ diverse writing styles. Researchers have recently conducted significant researches using individual approaches to recognize handwritten Bangla digits, alphabets, and slightly compound characters. To address this, we propose a straightforward and lightweight convolutional neural network (CNN) framework to accurately categorize handwritten Bangla simple characters, compound characters, and numerals. The suggested approach exhibits outperformance in terms of performance when compared too many previously developed procedures, with faster execution times and requiring fewer epochs. Furthermore, this model applies to more than three datasets. Our proposed CNN-based model has achieved impressive validation accuracies on three datasets. Specifically, for the BanglaLekha isolated dataset, which includes 84-character classes, the validation accuracy was 92.48%. On the Ekush dataset, which includes 60-character classes, the model achieved a validation accuracy of 97.24%, while on the customized dataset, which includes 50-character classes, the validation accuracy was 97.03%. Our model has demonstrated high accuracy and outperformed several prominent existing frameworks.

Keywords: ***Bangla Handwritten Recognition; Convolutional Neural Network; Deep Learning; Image Classification; Pattern Recognition***

1. Introduction

Bengali is an official language in Bangladesh and in the Republic of India, many Indians with approximately 300 million individuals worldwide communicate in Bengali as their primary language. The identification of handwritten Bangla characters and numerals has gained tremendous popularity as a research field in recent years, primarily due to its numerous practical applications [1]. The vast application areas of Bengali characters and numeral identification encompass a wide range of domains, including but not limited to bank cheques, postal addresses, optical image recognition, product expiration dates, post office automation, number plate reading, image-to-speech conversion with text recognition, Bengali identity card authentication, and real-time vehicle tracking [2]. Additionally, Bengali character and numeral identification also find their use in document analysis and recognition, particularly in digitizing handwritten documents such as historical records, manuscripts, and old texts in the Bengali language. In the healthcare industry, people also use it for handwritten text transcription, signature verification and authentication, and processing of handwritten medical records, prescriptions, and patient forms.
Furthermore, with the increasing trend of e-commerce and online transactions, Bengali characters and numeral identification can help verify customer information, addresses, and payment details. We can use

Md Asraful, Md. Anwar Hossain and Ebrahim Hossen, "Handwritten Bengali Alphabets, Compound Characters and Numerals Recognition Using a CNN-based Approach", *Annals of Emerging Technologies in Computing (AETiC)*, Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp. 60-77, Vol. 7, No. 3, 1st July 2023, Published by International Association for Educators and Researchers (IAER), DOI: 10.33166/AETiC.2023.03.003, Available: http://aetic.theiaer.org/archive/v7/v7n3/p3.html.

it to recognize handwritten answers and exam sheets for assessment and grading purposes. The entertainment industry also benefits from this technology by enabling the creation of Bengali language content, such as comics and novels, in digital form. It also has applications in linguistics research for analysing handwritten samples for research purposes. Moreover, we can achieve personalization by recognizing an individual handwriting style for personalized communication and authentication. Lastly, integrating Bengali character and numeral identification for text recognition and image processing features in mobile and web applications is becoming increasingly popular. Identifying Bengali characters and numerals is a critical concern that requires considerable attention to progress other active applications. The size and shape of different individual handwriting vary, making the task of character recognition challenging due to the numerous variations in each character's writing style [3]. The challenges of identifying Bengali characters far outweigh those of their English counterparts due to the inclusion of conjunct consonants in the Bengali alphabet, which are more complex to classify because they consist of two distinctive Bengali characters. The identification of such letters by a learning algorithm can prove problematic, as they may be identified as either one or the other, leading to inaccuracies.

Furthermore, as many Bengali characters are equivalent to one another, differentiating between them can be daunting. For instance, the initial "অ" and the second "আ" vowels in a single vertical line separate the Bengali alphabet, which presents a significant obstacle in character identification [4]. Although we have progressed, accurately recognizing complexly formed compound handwritten characters remains a significant challenge. The deep learning methodology of CNN has made remarkable progress in distinguishing Bangla handwritten characters by training on vast amounts of raw data to identify discriminative features [5]. To overcome the challenges in recognizing Bengali characters and numerals our study addresses the challenge of accurately identifying handwritten Bengali characters and numerals using a CNN-based model. We compared our proposed model with existing models commonly used in the literature, including CNN, MQDF, and ResNet-18, and found that our model outperforms these previous models with an accuracy rate of 92.48%, 97.24%, and 97.03% for respective datasets. Our proposed model also showed faster execution times and required fewer epochs than previous models. The main contribution of our research is introducing a comparatively simple and lightweight CNN-based model composed of 12 sequential layers, including five convolution layers, three pooling layers, and four dense layers. Despite its uncomplicated structure, our model accurately identified handwritten Bengali characters and numerals across 84, 60, and 50 classes. This approach provides a valuable alternative to previous, more complex models and offers improved performance for this task.

2. Related Works

In this section, we describe previously completed tasks related to Bangla handwritten recognition research area. Prior research in Bangla handwritten character classification has primarily focused on recognizing handwritten Bangla simple characters and numerals, with fewer studies focusing on identifying handwritten compound characters in Bengali. Despite that, the utmost of the exploration has been done on this task independently.

Rahman *et al*. [3] suggested that a CNN model can recognize only simple Bangla characters (50 classes) to achieve a testing accuracy of 85.36%. This model performs better with higher iterations closer to 300. Their model consists set of convolutional and other layers may achieve comparatively lower accuracy. A significant limitation of the model is its lower testing accuracy (85.36%) compared to its training accuracy (93.93%). So, the model overfitting to the training data. Our suggested method outperformed a well-known existing method in recognizing a greater number of classes (84 and 60) with fewer iterations (50 epochs). Our method achieved a training accuracy of 97% and a testing accuracy of 96%. The achieved accuracy is less distinguishable accuracy compared to the training accuracy.

Pal *et al.* [6] used the Modified Quadratic Discriminant Function (MQDF) to recognize Bangla compound characters and achieved 85.90 percent accuracy. MQDF is a cutting-edge classifier for handwriting recognition. Despite fitting the training data well, MQDF generalization performance is poor, as shown by a significant gap between training and test accuracy. We propose a CNN-based model to address the limitations of the suggested model and better performance with higher accuracy and

robustness. Our proposed model also performed combined characters such as simple and compound character sets else numerals.

Purkaystha *et al*. [7] designed a deep convolutional neural network (DCNN) approach for recognizing Bengali characters. The model achieves an accuracy of 91.23% for alphabet recognition (50-character categories) and 89.93% for recognition of almost all Bengali characters (80-character categories). They used a comparatively complex approach with more layers in their method. In addition, we proposed a comparatively simple and lightweight CNN approach that achieves accuracy rates of 97.03% for alphabets (50-character classes) and achieves accuracy rates of 92.48% for almost all Bengali characters (84 - character classes). Our method surpasses the previously suggested approach.

Khandokar *et al.* [5] developed a CNN model that trains on 1000 Bengali handwritten characters and tests on 200 handwritten characters. Its achieved accuracy rate is 92.91 percent. Working with a less amount of data results in a limited variety of characters. The model overfits when increasing the number of iterations. So, it is a significant limitation for any good deep learning model. However, our proposed model overcomes all the previously mentioned limitations, performs well on a comparatively large dataset, and achieves good accuracy.

Hossain *et al*. [8] proposed a model that performs on 60 classes of Bangla handwritten characters. It achieves an accuracy of 93.2 percent. This level of accuracy is proportionately good from other existing previously mentioned models. Since it accomplishes numerals and Bangla simple handwritten characters recognition, this model's execution time is a prominent point for model fit. However, the model keeps overfit and underfit. That is a significant limitation. However, this model is insufficient for better performance. Our deep learning model gets the better result than the previous model's mentioned limitations and achieves an accuracy of 97.24%. Our model training and validation accuracy learning graph indicate that this model is the best fit for the deep learning approach compared to the previously mentioned model.

Alif *et al.* [9] suggested a modified ResNet-18 architecture for recognizing isolated handwritten Bangla characters belonging to 84 different classes. The model achieves an accuracy of 95.99 percent. ResNet-18 is a pre-trained deep learning model that uses 72 layers and 18 deep layers, designed for a higher proportion of feature extraction layers to operate efficiently. However, integrating multiple deep layers into a network can lead to deterioration of output quality, and the model complexity is high with a large number of weights. Our proposed approach overcomes these shortcomings and achieves higher accuracy than ResNet-18 for this research domain.

Rabby *et al.* [10] investigated a lightweight CNN prototype for identifying handwritten Bengali numerals. The prototype surpasses all previously used methods with a higher accuracy (99.74%) and a faster execution time achieved in fewer epochs. The existing deep learning methods have shown impressive results in recognizing Bengali handwritten numeral character sets with 10 classes. However, our utilized technique surpasses these methods by accurately recognizing almost all character sets separately for 84 classes, 60 classes, and 50 classes.

Reddy and Raju [11] suggested model works on numerals getting accuracy gradually 99.74%, 97.07%, and 99.9%. It is a massive performance for numerals recognition through all provided models that perform only 10 classes. In this issue, our model performs more than 10 classes. Chowdhury *et al.* [12] proposed that CNN-based method accomplishes 50 classes character set and achieves an accuracy of 95.25% executed for 70 epochs. In this scenario, our suggested model produced a faster execution time with fewer epochs and achieved 97.03% accuracy. Our method recognizes areas for the significant achievement of the deep learning approach. Saha *et al.* [13] stated that a deep learning approach for recognizing Bangla handwritten simple characters of 50 classes achieved an accuracy of 96.40%. The recommended approach consists of six layers of convolution, six layers of pooling, and two dense layers. This approach has a total of 14 layers. However, our proposed model has only 12 layers, yet it achieves higher accuracy than the model mentioned above.

Roy [4] constructed a deep learning model for works on 84 classes of Bangla handwritten characters, numerals, and compound characters, with an accuracy of 96.40%. It is an outstanding performance of works on isolated character sets. It is the best out of 84 classes of isolated characters on work. Nevertheless, the current implementation of the model entails twelve convolutional, four pooling, and five fully connected layers. This framework is comparatively more complex and uses more hyperparameters, thus requiring

high-capacity devices and longer processing time. We have designed our proposed CNN model intending to achieve a faster execution by reducing the number of layers to 12, making it less complex and more superficial. We expect that our model will attain greater accuracy with more epochs. Consequently, our model is superior to the existing deep cognition models and significantly contributes to this research area.

Table 1 summarizes the research gap and critical findings for each study analyzed in this paper. The table includes the name of the study, its focus, approach, number of classes, achieved accuracy, and the research gaps addressed. The research gap column indicates the limitations or gaps addressed by each study. The accuracy column reports the level of accuracy achieved by each proposed model for recognizing handwritten Bengali numeral character sets. The approach column describes the proposed deep learning approach, while the number of classes column indicates the number of distinct character sets recognized by the model. This table offers an overview of the contributions of each study and the gap in the literature they addressed.

Table 1. Summary of existing Handwritten Bengali Character Recognition models and research gap

Study	Focus	Approach	Classes	Accuracy	Research Gap
Rahman *et al.* [3]	Simple characters	CNN	50	85.36%	Recognizes only simple characters
Pal *et al.* [6]	Compound characters	MQDF	20,543 characters	85.90%	Limited generalization performance
Purkaystha *et al.* [7]	Bengali character sets	Deep CNN	80	89.93%	Comparatively complex approach
Khandokar *et al.* [5]	Bengali handwritten characters	CNN	1000 characters (training) and 200 characters (testing)	92.91%	Overfitting when the model iteration increases
Hossain *et al.* [8]	Simple and numeral characters	CNN	60	93.2%	Overfitting and underfitting
Al Rabbani Alif *et al.* [9]	Isolated handwritten characters	ResNet-18	84	95.99%	Complexity and limited output quality
Rabby *et al.* [10]	Handwritten Bengali numeral recognition	Lightweight CNN prototype	84, 60, 50, 10	99.74%	Outperforms existing methods with more classes
Reddy and Raju [11]	Handwritten numeral recognition	LR, SVM, KNN, CNN	10, 50, 60, 84	97.07-99.9%	Performs well in more than 10 classes
Chowdhury *et al.* [12]	Handwritten character recognition	CNN-based method	50	97.03%	Faster execution time and higher accuracy than the existing method
Saha *et al.* [13]	Handwritten Bangla simple character recognition	Deep learning approach with 14 layers	50	96.40%	96.40% Proposed model achieves higher accuracy with fewer layers
Roy [4]	Handwritten Bangla character, numeral, and compound recognition	Deep learning model with 12 convolutional,4 pooling, and 5 fully connected layers	84	96.40%	The proposed model is less complex with faster execution time and can achieve greater accuracy with more epochs

3. Background Study

3.1. Convolutional Neural Network

Convolutional Neural Networks are extensively used and it's dominated machine learning, especially in classifying images and patterns. These networks process images as objects in three-dimensional space, incorporating the notion of three-dimensional objects. The influence of CNN on computer vision has undergone a revolutionary transformation [14]. Handwriting recognition is essential in machine learning and has many applications. Various techniques exist to transform written text on physical documents into a format for machines. Character recognition technologies can support the move toward a paperless world by facilitating the digitalization and processing of paper-based documents. CNNs are noteworthy as they

mimic neural networks in the human brain and can detect and map spatial relationships in an image to recognize and classify its content [5]. CNN uses a powerful fitting mechanism to identify unique features in an image. During training, the network adjusts parameters, costs, and biases to convert input images to feature vectors, leading to a deeper understanding of image characteristics. García-Ordás *et al.* [15] suggested that the CNN process, shown in Figure 1, includes components such as convolution, pooling, fully connected neural networks, and layers to tackle overfitting/underfitting.

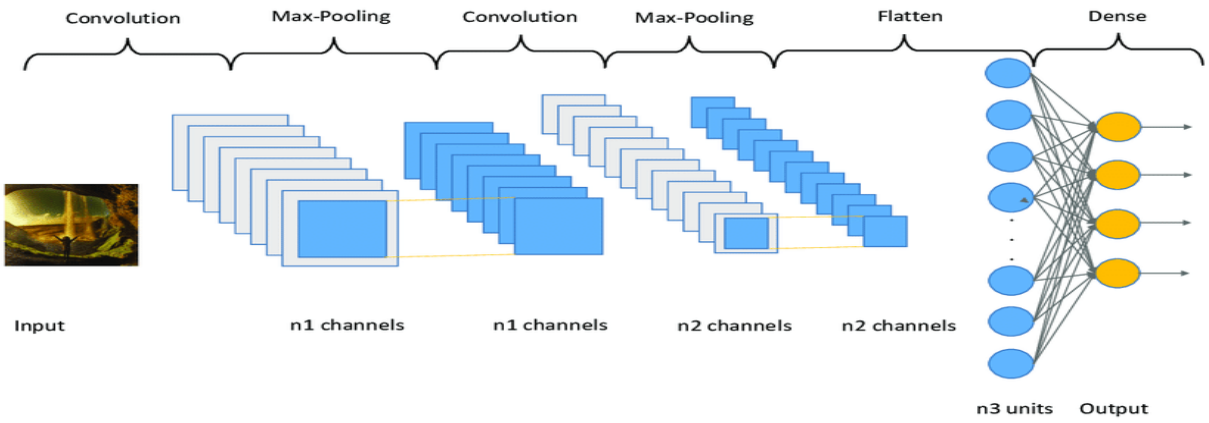


Figure 1. Basic CNN architecture [15]

3.2. Convolution Layer

This section uses memorization kernels like Gaussian or Gabor to perform convolution on feature maps from the previous layer, then pass the results through linear or non-linear activation functions like sigmoid, softmax, and relu to create output maps [16]. Convolution is a technique used for image processing to enhance, augment, or sharpen images. It involves passing a matrix of numerical values, called a kernel, across an image to reconstruct it. The kernel multiplies with the top-left pixels element-wise, and we sum the resultant values to generate the first element in the output matrix. We repeat this process throughout the image and move the filter toward the right. In RGB images, the input shape is (height x width x 3), and the filter shape is Fr x Fc x 3. We refer to the number of cells the filter shifts as the "stride." We convolve every filter segment with its corresponding section of the input image, and the resulting three values are then summed and considered as the output. We can utilize multiple filters in either scenario and when using more than one filter, we concatenate the output matrices. The resulting shape for both of these scenarios is provided as:

$$(\mathrm{H} - \mathrm{Fr} + 1) \times (\mathrm{H} - \mathrm{Fc} + 1) \times \mathrm{N_c} \quad (1)$$

In convolution operation, the variables 'H' and 'Fr, Fc' denote the height and width of the input image and filter, respectively. Meanwhile, the variable 'N_c' signifies the number of filters in the process. Padding is added to the input image to preserve its shape during the convolution operation, which may otherwise cause image shrinking or loss of crucial information. Two types of convolutions are there: Valid convolution with no padding and the other same convolution that preserves the shape of the input. The second type is the same convolution, which uses padding to preserve the shape of the input. To calculate the output shape when using padding, one can utilize the following formula:

$$(\mathrm{H} + 2\mathrm{PL} - \mathrm{Fr} + 1) \times (\mathrm{H} + 2\mathrm{PL} - \mathrm{Fc} + 1) \times \mathrm{N_c} \quad (2)$$

PL represents the number of padding layers, and the other variables are the same as defined previously. CNNs use randomly initialized filters as learning parameters, followed by bias addition and ReLU activation to introduce non-linearity [13]. ReLU prevents identical behaviour regardless of layer number. This paper uses ReLU. The Rectified Linear Unit (ReLU) activation function express as:

$$\mathrm{Out} = \max(0, \mathrm{Bias\_v}) \quad (3)$$

Here, Out represents the layer output, and $Bias\_v$ represents the value derived by integrating the bias value with the convolution output. In this triggering feature, convert the input to a positive number. The

reshaping of the features using this technique leads to a decrease in dimensionality compared to the original input.

3.3. Pooling Layer

Convolutional neural networks rely on pooling, which can perform in various ways. Max-pooling is commonly used, down sampling output and selecting features. Pooling has no learning parameters, only two hyperparameters, filter shape, and stride. Pooling reduces height and width, but filters stay constant. The most popular filter shape is 2x2 with stride 2, making the network more efficient. The formula determines the resulting shape of the output after pooling.

$$\frac{\text{Height} - \text{F\_S}}{\text{Strd}} + 1 \times \frac{\text{Width} - \text{F\_S}}{\text{Strd}} + 1 \times \text{N}_\text{F} \quad (4)$$

In the above formula, the height, width, and number of filters are denoted by Height, Width, and N_F, respectively. Additionally, 'Strd' represents the number of strides while 'F_S' denotes the shape of the filter. In this particular study, max pooling has been utilized [13].

Our approach includes the process of max pooling, which involves selecting the maximum value from a filter of a given size. The research paper explains that when using a 2x2 max pool filter with a stride of 2, we begin by processing the top-left 2x2 segment. We determine the maximum value of that segment and place it in the output. After this calculation, we move the filter two cells to the right and repeat the process. Max-pooling retains the essential features of data, and it is famous for extracting high values from sub-regions, thus retaining more information, which results in faster convergence and better model performance [17].

3.4. Dense Layer

A Convolutional Neural Network (CNN) usually includes a fully connected neural network in its last section, often referred to as a dense layer. These networks are essentially artificial neural networks that possess complete connectivity. Throughout the training phase, the system generates the weights. The dense system receives the output of the convolution action and identifies the label that most accurately signifies the image. The central purpose of the dense network is to generate an association between the feature vector of an image and its respective class. The strengths of the network, which represent the linking paths, multiply the output of the convolution maneuverer. The final output undergoes processing through an activation function.

3.5. Dropout

Connecting all the features to the FC layer often leads to overfitting in the training dataset. Overfitting transpires when a model acts exceptionally better on the training data, yet when applied to new and previously unexplored data; it adversely influences the model's performance. During the training process, the dropout layer disables a small number of neurons to tackle this problem. It's leading to a smaller and more efficient model. Upon reaching a dropout value of 0.25, 25% of the nodes in the neural network were disabled at random. Dropout is an effective technique for enhancing the activity of a deep learning model by mitigating overfitting through the simplification of the network. During training, the neural network removes neurons using a dropout layer [18].

3.6. Batch Normalization

Batch normalization is a useful method for faster and more efficient network training by reducing the internal covariate shift in data. It has become a crucial part of many advanced network topologies. This research highlights the importance of batch normalization in successful training and improved performance. The normalization layer behaved differently during training and inference, with mean and variance calculated separately for each batch and applied to all images during inference. The study shows that omitting batch normalization layers can negatively impact performance and demonstrates through experiments that it is critical for successful network training. The cutting-edge deep learning architectures

known for their performance on ImageNet have found similar results: batch normalization is necessary for training intense networks [19]. We have two alternatives for normalizing our data. The first method is

$$\text{Normpoint}_{(\text{normalized})} = \frac{\text{Normpoint}-\text{Mean}}{\text{Normpoint}_{\max}-\text{Normpoint}_{\min}} \quad (5)$$

Equation 5 for Normpoint is the normalized data point, Mean is the data set's mean, Normpoint_{max} $ax$is the maximum value, and Normpoint_{min} is the minimum value. Data inputs often employ this strategy. For large-scale data, use standard deviation. The second method is

$$\text{Normpoint}_{(\text{normalized})} = \frac{\text{Normpoint}-\text{Mean}}{\text{std}} \quad (6)$$

Equation 6 for Normpoint is the normalized data point, Mean is the data set's mean, and std is the set standard deviation. Each data point now resembles a typical normal distribution. None of the features on this scale will be biased, so our models will learn better.

4. Methodology and System Architecture

4.1. Data Preprocessing

The proposed model utilized three datasets for the experiment: the Ekush Bangla handwritten character dataset [20], the BanglaLekha dataset [21], and a custom dataset. The three datasets used in the experiment include handwritten hand-drawn basic Bengali letters, numerals, and complex characters. The Ekush dataset has 122 classes with 367,018 images, including gender and age classification. However, this research focused on Bangla handwritten basic characters and numerals, resulting in 60 classes and 183,164 images. We split the data into two sets using the split folder python module - 80% for training and 20% for validation. The BanglaLekha-Isolated dataset contains over 84 classes of basic Bengali letters, numerals, and complex characters and 162,540 images. We segregated our data model into three distinct groups: train, validation, and test sets. The train set contained 1,375 images per class, totaling 115,500 images. The validation set contained 295 images per class, totaling 24,780; the test set had 265 images per class, totaling 22,260. We ensured that each set was balanced. The study used a custom dataset of 50 unique categories with 15,000 images. The train set had 12,000 images, and the validation set had 3,000. We used the TensorFlow module 'image_dataset_from_directory' to read the images from directory names. We renamed each folder according to its class to be able to use the 'image_dataset_from_directory' module. There were 50 folders containing basic Bangla characters, 10 containing Bangla numerals, and 24 containing Bangla compound characters, with class numbers ranging from 1 to 84. Table 2 lists the classes of 84 Bangla basic letters, numerals, and compound characters.

Table 2. Bengali Character classes picked for recognition

Bangla basic characters									
অ [1]	আ [2]	ই [3]	ঈ [4]	উ [5]	ঊ [6]	ঋ [7]	এ [8]	ঐ [9]	ও [10]
ঔ [11]	ক [12]	খ [13]	গ [14]	ঘ [15]	ঙ [16]	চ [17]	ছ [18]	জ [19]	ঝ [20]
ঞ [21]	ট [22]	ঠ [23]	ড [24]	ঢ [25]	ণ [26]	ত [27]	থ [28]	দ [29]	ধ [30]
ন [31]	প [32]	ফ [33]	ব [34]	ভ [35]	ম [36]	য [37]	র [38]	ল [39]	শ [40]
ষ [41]	স [42]	হ [43]	ড় [44]	ঢ় [45]	য় [46]	ৎ [47]	ং [48]	ঃ [49]	ঁ [50]
Bangla numerals									
০ [51]	১ [52]	২ [53]	৩ [54]	৪ [55]	৫ [56]	৬ [57]	৭ [58]	৮ [59]	৯ [60]
Bangla compound characters									
ক্ষ [61]	ব্দ [62]	ঙ্গ [63]	ক্ক [64]	স্ফ [65]	স্থ [66]	চ্ছ [67]	ক্ত [68]	ন্ন [69]	ঞ্চ [70]
স্প [71]	ষ্ম [72]	প্ত [73]	ষ্ব [74]	ন্ত [75]	ড্ড [76]	থ্ব [77]	ষ্ঠ [78]	ল্ল [79]	ম্প [80]
ন্দ [81]	ন্ধ [82]	ষ্ম [83]	ষ্ঠ [84]						

4.2. Data Augmentation

Data augmentation is a method that involves generating modified versions of existing datasets to raise the size of the train set. This process usually involves minor alterations to the data or creating new data points using a deep neural network. The proposed tactic involves artificially expanding the handwritten dataset, which prevents overfitting, improves model accuracy, and reduces the costs associated with labelling and cleaning the raw dataset. Various methods were chosen for data augmentation, as this data manipulation could introduce variation that reflects how digits are written differently [4,22].

- Featurewise_center=False: Using feature-wise normalization this parameter sets the input mean to zero over the entire dataset.
- Samplewise_center=False: The sample-wise centre parameter sets each sample's mean to false per sample.
- Featurewise_std_normalization=False: Transform the input by dividing each feature by the dataset's standard deviation in a feature-wise manner.
- Samplewise_std_normalization=False: For each sample, divide the inputs by the standard deviation of the dataset using sample-wise.
- Zca_whitening=False: Apply ZCA whitening using zca_whitening.
- Rotation_range: We will rotate our training images by 15 degrees.
- Zoom_range: Randomly apply a 1% zoom to the training image.
- Height_shift_range and width_shift_range: We introduced variation in the dataset by randomly shifting the images in both height and width by 10%.
- Horizontal_flip and vertical_flip: We have set the random flipping of images for both vertical and horizontal orientations to false.

4.3. System Architecture and Proposed Model Algorithm

4.3.1. System Architecture

A CNN composes distinct trio categories of layers, namely convolutional, max-pooling, and dense layers, also known as connected. Arranging these layers in a specific order creates the architecture of a CNN. Our proposed deep learning model based on CNN architecture consists of twelve sequentially connected layers. It incorporates five convolutional, three max-pooling, and four dense layers. The CNN framework's initial layer is the input layer, which includes various parameters such as input shape, filter size, kernel size, and activation function. Within our model, we have established filtered scope of 64 and kernel dimensions of 3 by 3. We have also specified the input shape to be (64, 64, 3), where the first two values in the input data represent the image's width and height, respectively, while the third value indicates the number of color channels. In this case, the color channels are represented by 3 for an RGB image. Additionally, we have equipped our model with ReLU as its activation function. Next, we incorporate a two-dimensional convolutional layer into our model with a filter size of 64 and a relu as its activation utility. Following this, we introduce a max-pooling layer that reduces the magnitude of the convoluted chin map, ultimately conserving computational resources. It contains two distinct types of layers: max-pooling and average pooling layers. Specifically, we have implemented a pooling layer with pool dimensions of 2 by 2. In addition, we have introduced two additional layers to our model: batch normalization and dropout layers. These are critical in mitigating the risk of overfitting and underfitting. Batch normalization operates by normalizing its mean and standard deviation for a given batch, while dropout facilitates the removal of specific neurons from a given layer. Within our model, we have set a dropout value of 0.25, indicating that the dropout layer will disable 25% of the neurons. Following this, we have incorporated two consecutive convolutional layers, each utilizing a filter size of 128, with all other parameters identical to those employed in the prior convolutional layer. Subsequently, we integrate another CNN first layer utilizing a filter size of 256. We then familiarize a pooling side with a pool scope of 2 by 2, surveyed by including a normalization layer. We have implemented a dropout layer with a threshold rate of 0.25. In order to transition from a convolutional layer to a dense layer, a flattened layer is often incorporated. This layer is answerable for converting the two-dimensional input into a one-dimensional format. Its primary usage occurs at the intersection of the convolutional and dense layers. We introduce a dense layer to establish connectivity between neurons across layers. This layer, consisting of weights and biases, is employed to finalize the architecture of a CNN. Frequently positioned prior to the output layer, dense layers typically constitute the concluding segments of a CNN model. The current model implements three dense layers before the last layer. The initial dense section has a unit size of 512 and utilizes a relu activation function. A hyperparameter known as kernel regularizer is employed to prevent overfitting grounded on its threshold rate. The unit size of the second dense layer is 256, and it possesses the same parameters as the previous layer. The third dense layer has a unit size of 128, with its remaining values set identically to the preceding

layer. Additionally, we implemented a dropout layer in the model. Subsequently, the final dense layer functions as the output for the model, providing the results of the classification process. The unit size of this layer is the total number of classes to be classified, and softmax defines its activation function. In our research, the suggested deep learning framework comprises a total of 9121300 parameters, with 9120404 of them being trainable. The remaining 896 parameters are non-trainable. Table 3 enumerates all the parameters required to establish our proposed model for classifying Handwritten Bengali alphabets, numerals, and compound characters. The total parameters of the model are 9121300, with 9120404 being trainable and 896 being non-trainable.

Table 3. Proposed CNN model for internal parameter

Type of layer	Output Shape	Parameters
Convolution 2D layer	(None, height=64, width=64, filter size=64)	1792
Convolution 2D layer	(None, height=64, width=64, filter size =64)	36928
MaxPooling 2D layer	(None, height=32, width=32, filter size =64)	0
Batch normalization layer	(None, height=32, width=32, filter size =64)	256
Dropout layer	(None, height=32, width=32, filter size =64)	0
Convolution 2D layer	(None, height=32, width=32, filter size =128)	73856
Convolution 2D layer	(None, height=32, width=32, filter size =128)	147584
MaxPooling 2D layer	(None, height=16, width=16, filter size =128)	0
Batch normalization	(None, height=16, width=16, filter size =128)	512
Dropout layer	(None, height=16, width=16, filter size =128)	0
Convolution 2D layer	(None, height=16, width=16, filter size =256)	295168
MaxPooling layer	(None, height=8, width=8, filter size =256)	0
Batch normalization	(None, height=8, width=8, filter size =256)	1024
Dropout layer	(None, height=8, width=8, filter size =256)	0
Flatten layer	(None, 16384)	0
Fully connected layer	(None, filter size =512)	8389120
Fully connected layer	(None, filter size =256)	131328
Fully connected layer	(None, filter size =128)	32896
Dropout layer	(None, filter size =128)	0
Fully connected layer	(None, filter size =84)	10836

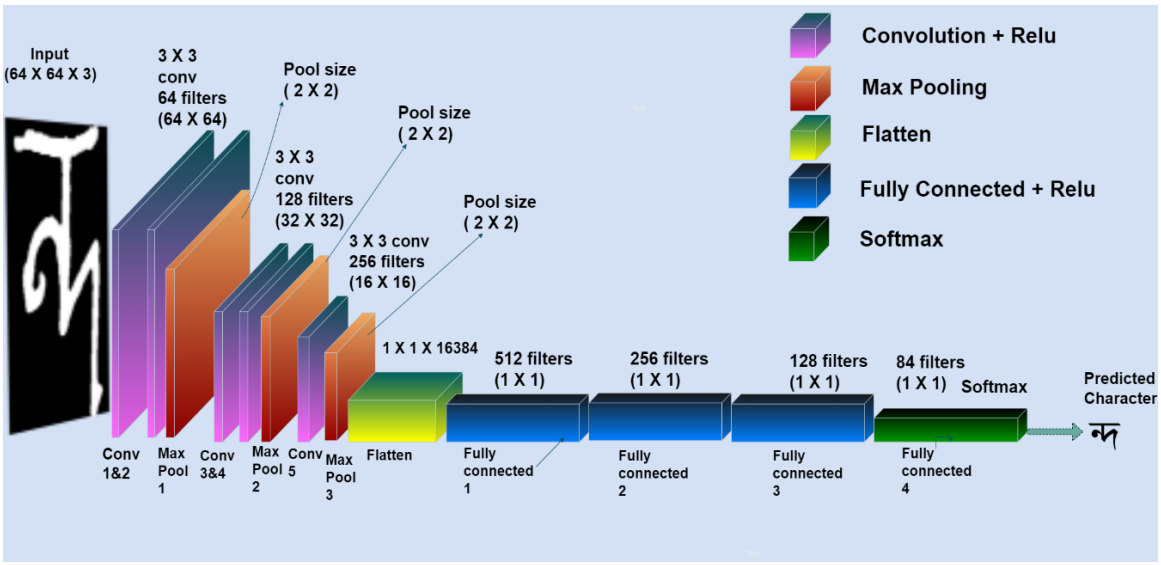


Figure 2. CNN architecture for Bangla handwritten character recognition. Here conv stands for Convolutional layer

Figure 2 depicts the architecture of our stated CNN model. It takes in an RGB image with dimensions of 64x64x3 as input. The initial two layers employ 64 filters of a 3x3 kernel size with the same padding, resulting in a volume of 64x64x64. Then we employed a pooling layer with a max-pooling technique using a pool size of 2x2. This operation resulted in a reduction of the height and width of the volume from 64x64x64 to 32x32x64. Subsequently, two additional convolutional layers with 128 filters each were applied, resulting in a new volume dimension of 32x32x128. Following the utilization of the pooling layer, the volume underwent a reduction, resulting in a new dimension of 16x16x128. An additional convolutional

layer was incorporated, consisting of 256 filters, resulting in a new dimension of 16x16x256. After this, a final pooling layer was applied, resulting in a new feature map dimension of 8x8x256. We flattened the output, resulting in a feature vector with dimensions of (1x1x16384). The model utilized four fully connected layers. The first layer received input from the final feature vector and produced an output vector with dimensions of (1x1x512). The second and third layers generated output vectors with dimensions of (1x1x256) and (1x1x128), respectively. Finally, the fourth layer produced an output of 84 channels to correspond to the 84 classes. The model utilized a fourth fully connected layer, which implemented the softmax function to classify the 84 classes. The activation function used for all the hidden layers was relu. This architecture is commonly used for image classification tasks and has achieved high accuracy on various datasets.

4.3.2. Proposed Model Algorithm

Algorithm 1. Bangla handwritten character recognition for CNN algorithm

	Initialize model		Model = Sequential ()
1	Convolution2D	→	Conv2D (filter, kernel dimensions, activation, padding, input shape)
2	Convolution2D	→	Conv2D (filter, kernel dimension, activation, padding, input shape)
3	Max pooling	→	MaxPooling2D (pool dimensions)
4			Batch Normalization ()
5			Dropout (rate)
6	Convolution2D	→	Conv2D (filter, kernel dimensions, activation, padding)
7	Convolution2D	→	Conv2D (filter, kernel dimensions, activation, padding)
8	Max pooling	→	MaxPooling2D (pool dimensions)
9			Batch Normalization ()
10			Dropout (rate)
11	Convolution2D	→	Conv2D (filter, kernel dimensions, activation, padding)
12	Max pooling	→	MaxPooling2D (pool dimensions)
13			Batch Normalization ()
14			Dropout (rate)
15			Flatten ()
16	Dense	→	Dense (Units, activation, regularizer= regularizers. l2(learning rate))
17	Dense	→	Dense (Units, activation, regularizer= regularizers. l2(learning rate))
18	Dense	→	Dense (Units, activation, regularizer= regularizers. l2(learning rate))
19			Dropout (rate)
20	Dense	→	Dense (total classification classes, activation="softmax"))
	End		

4.3.3. Model Compilation

We compiled the proposed CNN-based model using an ADAM optimizer, with a learning rate of 0.00004, and categorical_crossentropy as its regression function. We trained our model in a Kaggle notebook utilizing GPU services.

4.4. Hyperparameters Tuning

Hyperparameters refer to the settings that regulate the learning process and influence the values of the model parameters learned by an algorithm. Now delve into some crucial hyperparameters.

4.4.1. Activation Functions

The activation function is a crucial component in the CNN model as it learns and estimates complex relationships between network variables. It brings non-linearity into the network and decides which information to propagate and which to disregard. Commonly used activation functions in CNN models are softmax, tanh, relu, and sigmoid, each serving a distinct purpose. Researchers often choose between the softmax and sigmoid functions for binary classification, while softmax is a common choice for multi-class classification. Activation functions determine if input data is relevant enough to undergo mathematical operations and play a significant role in deciding whether a neuron should be activated [23].

4.4.2. Optimizer

Optimizers refer to algorithms designed to minimize error functions or maximize production efficiency. They are mathematical functions influenced by the model's learnable parameters, such as weights and biases. Neural networks employ optimizers to adjust the weights and learning rate to reduce

the loss function. In training deep CNN models, the optimizer plays a vital role in performing iterative updates on the parameters of all layers required to optimize the neural network's performance. This research's crucial emphasis is evaluating the impact of Adam, RMSProp, and stochastic gradient descent optimizers on our dataset [24].

4.4.3. Learning Rate

The learning rate is a significant hyper-parameter in a neural network that controls weight adjustments based on loss gradient and determines how often the network updates learned perceptions. Selecting an appropriate value is difficult, as too small a value can led to prolonged training, while too large a value can cause unstable training or rapid learning of low weights. It is an essential hyperparameter to consider during configuration, and tools like Adam and RMSprop can manually select the learning rate schedule in each learning session [21, 25]. We have established the learning rate at 0.00004 and set the kernel regularization rate to 0.001 in this model.

5. State-of-the-art Architectures Analysis

In the State-of-the-Art analysis section of the research paper, the performance and accuracy of three popular image classification models, namely ResNet50, DenseNet121, and MobileNetV3-Large, were evaluated by training and testing them on the selected dataset. The main objective was to identify the most effective model for the given task by comparing their performance on various metrics. The analysis aimed to determine the state-of-the-art accuracy achieved by these models.

ResNet50 is a widely-used convolutional neural network model in deep learning research. It has 50 layers and utilizes connections to address the vanishing gradient problem, which can occur in deep neural networks. By allowing gradients to flow through the network more efficiently, ResNet50 can learn more effectively and achieve better performance on image classification tasks, such as the ImageNet dataset. Its success has made it a popular starting point for developing new deep-learning models. Overall, ResNet50 is a powerful and vital tool for computer vision research.

Densenet121 is a deep learning model that uses dense connections to address the problem of vanishing gradients in deep neural networks. It comprises 121 layers of dense blocks and transition layers to facilitate information flow. Densenet121 has achieved state-of-the-art performance on image classification tasks and has been applied to other computer vision tasks, significantly impacting computer vision research.

MobileNetV3-Large is a neural network created for mobile and embedded devices with limited computational resources. It has 17 building blocks that use a bottleneck layer, hard-swish activation function, and expansion layer to maintain accuracy while decreasing computational cost. The model also incorporates a squeeze-and-excitation module that recalibrates feature maps for improved accuracy. MobileNetV3-Large has 5.5 million parameters but still achieves state-of-the-art accuracy on image classification benchmarks, making it a lightweight and efficient choice for deployment on mobile and embedded devices.

Table 4 compares the performance of different models, including ResNet50, DenseNet121, MobileNetV3-Large, and a proposed model, in terms of their accuracy on training, validation, and testing sets of two specific datasets: BanglaLekha isolated and Ekush. This comparison aims to evaluate the proposed model's effectiveness and identify its strengths and weaknesses compared to existing models.

Table 4. Comparison of the proposed model with pre-trained models

Model	BanglaLekha isolated					Ekush dataset				
	Classes	Train Acc.	Val Acc.	Test Acc.	Parameter size	Classes	Train Acc.	Val Acc.	Test Acc.	Parameter size
ResNet50	84	0.9290	0.9285	0.9141	23759828	60	0.9709	0.9719	0.9672	23710652
DenseNet121	84	0.9344	0.9258	0.9063	32771220	60	0.9745	0.9723	0.9721	32758908
MobileNet-V3	84	0.9317	0.9214	0.9203	4066516	60	0.9697	0.9643	0.9612	4041916
Proposed model	84	0.9278	0.9248	0.9213	9121300	60	0.9773	0.9724	0.9712	9118204

6. Performance Evaluation and Mathematical Analysis

We evaluate the performance of our method using Precision, Recall, F-scores, Exact Match, and Modified Accuracy.

6.1. Precision

Precision is the ratio of true positives (TP) to the total number of predicted positives (TP + false positives (FP)). In other words, precision is the proportion of true positive predictions out of all positive predictions made by the model. A high precision indicates that the model is good at identifying positive cases and has a low false positive rate. The following formulas are to determine precision.

$$\text{Precision} = \frac{\sum_{k=1}^{m} |G_k \cup E_k|}{\sum_{k=1}^{m} |E_i|} \quad (7)$$

6.2. Recall

Recall, also known as sensitivity or true positive rate, is the ratio of true positives (TP) to the total number of actual positives (TP + false negatives (FN)). In other words, recall is the proportion of positive cases the model correctly identified as positive. A high recall indicates that the model is good at identifying all positive cases, including those that are hard to detect. We use the following mathematical expressions to compute recall for a classification model.

$$\text{Recall} = \frac{\sum_{k=1}^{m} |G_k \cup E_k|}{\sum_{k=1}^{m} |G_k|} \quad (8)$$

Where G_k and E_k denote gold-standard targets and model's predicted levels for k^{th} word such that $W_k \in W$. The intersection of gold-standard targets and the model's predicted levels for a given word $M_k \in M$ is considered as,

$$G_k \cup E_k =: \{E \in E_k \mid \exists G \in G_k, \text{Match}(G, E)\} \quad (9)$$

6.3. F∝ score

The F∝ score is a weighted harmonic mean of precision and recall, where beta is a positive constant that controls the trade-off between precision and recall. When ∝ set to 1, the F∝ score becomes the harmonic mean of precision and recall, commonly known as the F1 score. The following formula using calculate F∝ score:

$$F\propto = \frac{(1+\propto^2) \times P \times R}{(\propto^2 \times P)+R} \quad (10)$$

Where P denotes the Precision and R denotes the Recall. The F∝ score is a useful metric for evaluating a classification model's overall performance, as it considers both precision and recall and allows for a customizable balance between the two.

6.4. Match Value

The effectiveness of the model in all categories, including accuracy, by calculating it using the formula: (True Positive + True Negative) / (True Positive + False Positive + True Negative + False Negative). In this formula, True Positive, True Negative, False Positive, and False Negative refer to different scenarios where the model's output (p(x)) is either correct or incorrect based on whether the prediction (Est (y)) matches the actual label (y). If the prediction is accurate, it is deemed correct.

$$F(x) = \begin{cases} 1, & \text{if holds Est (y)} = p(x) = y \\ 0, & \text{Otherwise} \end{cases} \quad (11)$$

The Match Value metric calculates the ratio of accurate predictions to the total number of instances. A higher Match Value score indicates better model performance. To compute the Match Value score, use the following formula.

$$Match\ Value = \frac{\sum_{1}^{INST} F(x)}{INST} \quad (12)$$

The sum of F(x) from 1 to INST represents the count of accurate predictions, where INST is the number of instances.

6.5. Update Accuracy

We determine the Update Accuracy by measuring the model's accuracy within the top-L predictions. Unlike Accuracy, which measures overall correctness, the Match Value metric evaluates a model's

effectiveness over a corpus. We consider a prediction positive if top-L predictions match any outcome in the desired corpus. To determine whether a prediction is positive, use the following formula.

$$G(x) = \begin{cases} 1, & \text{if holds Est } (y) = p(x) = L \in W \\ 0, & \text{Otherwise} \end{cases} \quad (13)$$

Similar to the Match Value metric, the Update Accuracy measures the ratio of total positive predictions to the instances in the corpus. A higher Update Accuracy score indicates better model performance. To compute the Update Accuracy score, use the following formula.

$$Update\ Score = \frac{\sum_1^{PPN} G(x)}{PPN} \quad (14)$$

The sum of G(x) from 1 to PPN represents the count of accurate positive predictions, where PPN is the number of instances in the corpus.

6.6. Softmax Loss

The Softmax loss is a loss function that combines multinomial logistic loss and softmax. When evaluating a training set, let $A^{(i)}$ and $B^{(i)}$ represent the i-th input image patch and its corresponding target class label among P classes, respectively, where $i \in \{1, ..., M\}$ and $B^{(i)} \in \{1, ..., P\}$. We transform the prediction for the j-th class of the i-th input using the softmax function as follows:

$$M_j^{(i)} = \frac{e^{P_j^{(i)}}}{\sum_{l=1}^{P} e^{P_l^{(i)}}} \quad (15)$$

Where $P_j^{(i)}$ are usually the activations of a fully connected layer, so $P_j^{(i)}$. That expressed as

$$P_j^{(i)} = Weight_j^N C^{(i)} + Q_j \quad (16)$$

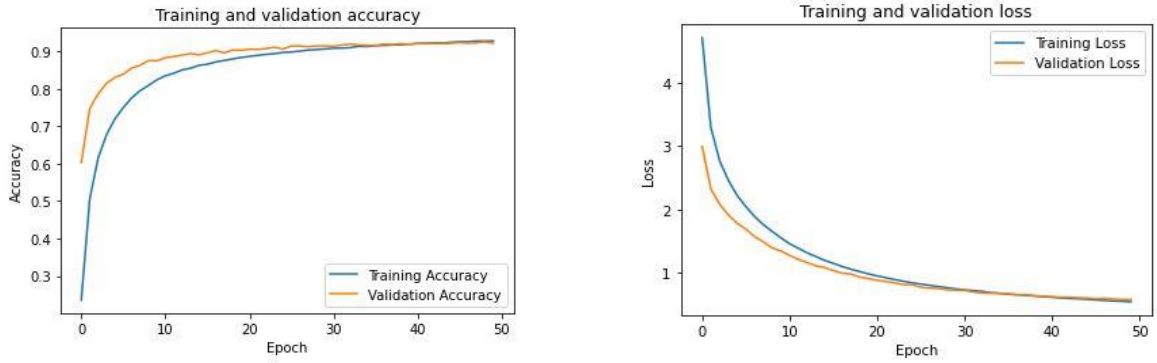
The softmax function turns raw prediction scores into a probability distribution over classes by making them non-negative and normalizing them. We then use this distribution to compute the softmax loss, which measures the difference between predicted and actual label distributions. The goal is to minimize this difference and improve the classifier's accuracy:

$$LOSS = -\frac{1}{PPN}\left[\sum_{k=1}^{ppn}\sum_{l=1}^{M}\{y^{(k)} = j\}\log M_l^{(k)}\right] \quad (17)$$

7. Result and Discussion

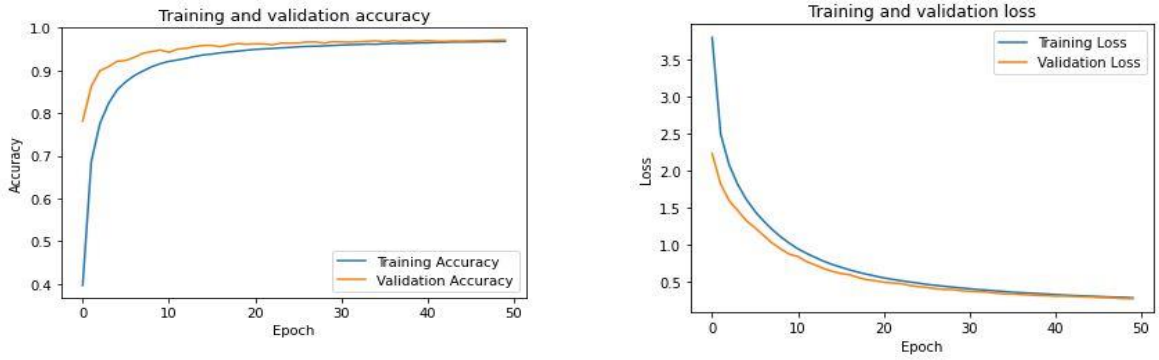
In this study, we proposed a model for the isolated BanglaLekha dataset with 84-character classes and achieved a training accuracy of 94%, a validation accuracy of 93%, and a testing accuracy of 93% after 50 epochs. We trained the model using a learning rate of 0.00004, the Adam optimizer, a dropout rate of 25%, and a regularization rate of 0.001. The accuracy and loss trends for training and validation sets over 50 epochs are plotted in Figures 3(a) and 3(b) to analyse the results. The model achieved high accuracy early in the training process and stabilized, indicating that it effectively learned the patterns in the dataset. The model's regularization and dropout techniques helped prevent overfitting and improve its generalization ability. The regularization technique reduced the complexity of the model, and the dropout technique prevented the model from relying too heavily on any single node. Overall, the proposed model was effective in learning the patterns in the BanglaLekha dataset and generalizing them to new data, and we can perform experimentation to improve its accuracy. We can infer that the proposed model achieved excellent accuracy on the Ekush dataset, as evidenced by its training accuracy of 98%, validation accuracy of 97%, and testing accuracy of 97%. Figures 4(a) and 4(b) depict training and validation accuracy and loss trends. We obtained the results using a learning rate of 0.00004, an Adam optimizer, a dropout rate of 25%, and a regularization rate of 0.001. However, the section lacks analysis and justification of the obtained results, which could be addressed by providing insights into the performance of the model and how it compares to other existing models in terms of accuracy, computational efficiency, and generalization ability. Based on the information provided, figures 5(a) and 5(b) show the results obtained from the created model for the 50-character classes dataset. These results indicate a 97% training accuracy, 96% validation accuracy, and 96% testing accuracy after 50 epochs. Using a learning rate of 0.00004 and an Adam optimizer with a 25% dropout rate and a regularization rate of 0.001 contributed to the results. In addition to the graphical representation of the accuracy and loss trends, we can further analyse and justify the obtained results by comparing them to other

state-of-the-art models and examining any potential limitations or challenges encountered during the model creation and training process.



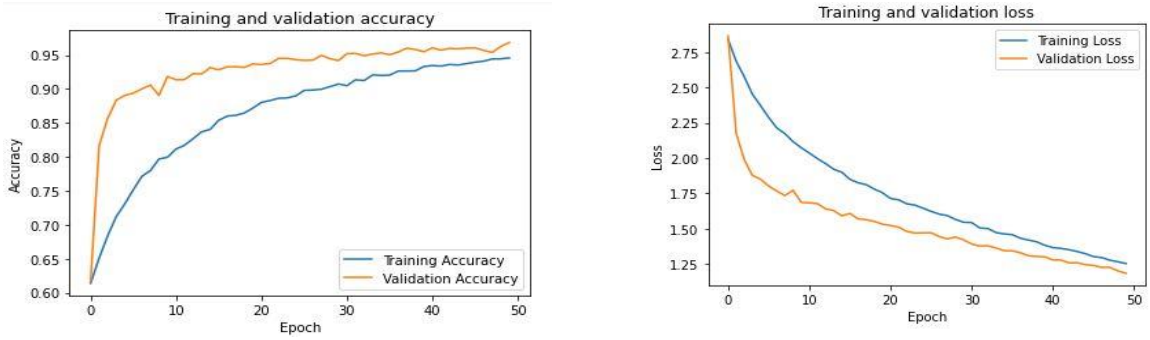
(a) (b)

Figure 3. (a) Training and Validation accuracy for BanglaLekha dataset (b) Training and Validation loss for BanglaLekha dataset



(a) (b)

Figure 4. (a)Training and Validation accuracy for Ekush dataset (b)Training and Validation loss for Ekush dataset



(a) (b)

Figure 5. (a) Training and Validation accuracy for custom dataset (b) Training and Validation loss for custom dataset

8. Model Evaluation Performance Comparison on Different Datasets

We evaluated the suggested framework over numerous datasets, and the outcomes indicate favorable performance across the training, testing, and validation sets. Table 5 presents an immediate of these findings.

Table 5. Model evaluation performance

Dataset	Train Error	Test Error	Val. Error	Train Accuracy	Test Accuracy	Val. Accuracy	Classes
BanglaLekha	0.49	0.55	0.58	0.94	0.93	0.93	84
Ekush Bangla	0.24	0.25	0.26	0.98	0.97	0.97	60
Custom	1.16	1.18	1.18	0.97	0.96	0.96	50

In the following comparison, Table 6 presents an evaluation of various deep learning models utilized for Bangla handwritten character recognition. The study assesses models including CNN, MQDF, and ResNet-18 that other researchers have previously proposed. In addition, the proposed model is also evaluated and compared to these models in terms of accuracy and limitations.

Our proposed model outperforms previous models with accuracies of 92.48%, 97.24%, and 97.03%, respectively, and shows faster execution times and fewer required epochs. Limitations of previous models include low accuracy, restrictions to certain types of characters, overfitting, underfitting, high complexity, and large numbers of weights. The comparison table comprehensively evaluates various deep-learning models for Bangla handwritten character recognition.

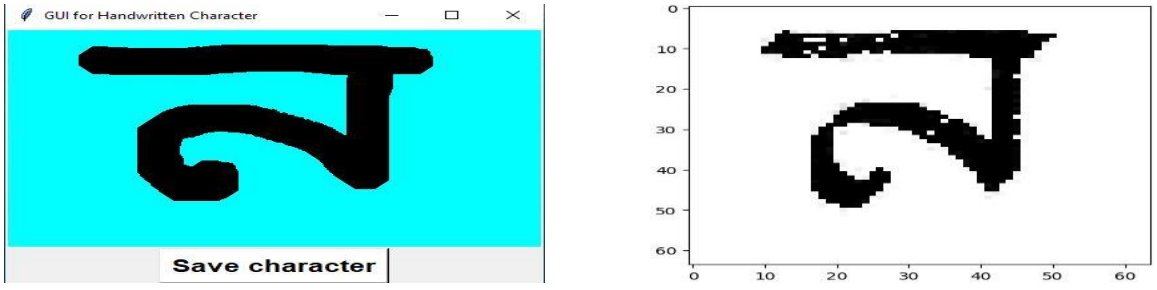
Table 6. Comparison of the proposed model with other existing different deep learning models

Research paper	Proposed model	Classes	Validation Accuracy	Necessary comments
Rahman et al. [3]	CNN	50	85.36%	Low accuracy, requires high iterations
Pal et al. [6]	MQDF	20,543 characters	85.90%	Limited to compound characters, generalization performance not encouraging
Khandokar et al. [5]	CNN	1200 characters	92.91%	Overfitting when iterations increase
Hossain et al. [8]	CNN	60	93.2%	Overfitting and underfitting
Al Rabbani Alif et al. [9]	ResNet-18	84	95.99%	High complexity, a large number of weights, not suitable for this specific research domain
Saha et al. [13]	Deep learning	50	96.40%	Six layers of convolution, six layers of pooling, and two dense layers; more experimental results needed
Roy [4]	Deep learning	84	96.40%	Twelve convolutional layers, four pooling layers, and five fully connected layers; require high-capacity devices and longer processing time
Our Proposed Model	CNN	84, 60, 50	92.48%, 97.24%, 97.03%	A lightweight model with 12 layers; outperforms previous approaches in terms of performance with faster execution times and fewer epochs required

9. Graphical User Interface

We developed of a self-contained graphical user interface (GUI) for retrieving Bengali handwritten characters from users. We constructed this system using the Python tkinter module, a well-known GUI library. Tkinter allowed us to quickly and efficiently develop a graphical user interface. The GUI allows the user to draw a Bengali character, our trained model, then processes it after being converted into a 64x64 image. The GUI is displayed on the screen, as shown in Figure 6. Upon opening the GUI, a new window contains a blank drawing area with dimensions of 400x256 pixels. Users can draw the desired character within this area using the mouse. We set the color of the drawing area to cyan and the drawing tool to black with a width of 30 pixels. Once the user has finished drawing the character, they can save it by clicking the "Save character" button. Then save the image to a specified location on the computer. This process allows users to create custom Bengali characters and use them for testing or other purposes.

In summary, the GUI provides a simple and efficient way for users to interact with our model and input custom Bengali characters for testing. It allows for a more intuitive and user-friendly experience than manually creating and inputting images and enhances the overall usability of our model.



(a) (b)

Figure 6. (a) Graphical user interface built in python (b) Model read from interface image

10. Conclusion and Future Work

The investigation presented here describes a deep learning technique utilizing a convolutional deep learning network to identify hand-drawn Bengali letters, including simple characters, numerals, and compound characters. We evaluate the performance of the model on both balanced, noise-free datasets and imbalanced, noisy datasets, and we achieve reasonable accuracy in both cases. The model exhibits good performance, as evident from the accuracy achieved in the validation stage. The validation accuracy is 92.48% for the 84-character class, 97.24% for the 60-character class, and 97.03% for the 50-character class. Compared to other standalone Bangla recognition of hand-drawn characters frameworks currently available, our model is less complex, lightweight, and yet achieves high accuracy. We are expecting that our framework will perform exceptionally well on datasets in other languages regarding similar character sets and style of writing to Bengali, such as Hindi, Arabic, and Telugu. We believe that our strategy has the ability to provide an essential contribution to the area of language analysis and recognition in numerous types of languages. In the future, we aim to continuously improve our algorithm to achieve higher levels of accuracy in classifying numerous character categories.

Acknowledgment

We thank the Department of Information and Communication Engineering, Pabna University of Science and Technology for assisting with this research.

References

[1] Nishatul Majid and Elisa H. Barney Smith, "Introducing the Boise State Bangla Handwriting Dataset and an Efficient Offline Recognizer of Isolated Bangla Characters", in *Proceedings of the 16th IEEE International Conference on Frontiers in Handwriting Recognition 2018 (ICFHR '18)*, 05-08 August 2018, Niagara Falls, NY, USA, E-ISBN:978-1-5386-5875-8, Print on Demand(PoD) ISBN: 978-1-5386-5876-5, DOI: 10.1109/ICFHR-2018.2018.00073, pp. 380-385, Published by IEEE, Available: https://ieeexplore.ieee.org/abstract/document/8583791.

[2] Asfi Fardous and Shyla Afroge, "Handwritten Isolated Bangla Compound Character Recognition", in *Proceedings of the IEEE International Conference on Electrical, Computer and Communication Engineering 2019 (ECCE '19)*, 07-09 February 2019, Cox's Bazar, Bangladesh, E-ISBN:978-1-5386-9111-3, Print on Demand(PoD) ISBN: 978-1-5386-9112-0, DOI: 10.1109/ECACE.2019.8679258, pp. 01-05, Published by IEEE, Available: https://ieeexplore.ieee.org/abstract/document/8679258.

[3] Md. Mahbubar Rahman, M. A. H. Akhand, Shahidul Islam, Pintu Chandra Shill and M. M. Hafizur Rahman, "Bangla Handwritten Character Recognition using Convolutional Neural Network", *International Journal of Image, Graphics and Signal Processing*, Print ISSN: 2074-9074, Online ISSN: 2074-9082, pp. 42–49, Vol. 7, No. 8, 8th July 2015, Published by MECS Press, DOI: 10.5815/ijigsp.2015.08.05, Available: https://www.scinapse.io/papers/755956977.

[4] Akash Roy, "AKHCRNet: Bengali handwritten character recognition using deep learning", *Computing Research Repository*, Online ISSN: 2331-8422, Vol. 2008.12995, 23rd January 2021, DOI: 10.48550/arXiv.2008.12995, Available: https://dblp.org/rec/journals/corr/abs-2008-12995.

[5] I Khandokar, Md M Hasan, F Ernawan, Md S Islam and M N Kabir, "Handwritten character recognition using convolutional neural network", in *Proceedings of the 7th International Conference on Mathematics, Science, and Education 2020 (ICMSE '20)*, 06 October 2020, Semarang, Indonesia, vol. 1918, no. 4, p. 042152, DOI: 10.1088/1742-6596/1918/4/042152, Published by IOP, Available: https://iopscience.iop.org/article/10.1088/1742-6596/1918/4/042152/meta.

[6] U. Pal, T. Wakabayashi and F. Kimura, "Handwritten Bangla Compound Character Recognition Using Gradient Feature", in *Proceedings of the 10th IEEE International Conference on Information Technology 2007 (ICIT '07)*, 17-20 December 2007, Rourkela, India, Print ISBN: 0-7695-3068-0, DOI: 10.1109/ICIT.2007.62, pp. 208-213, Published by IEEE, Available: https://ieeexplore.ieee.org/abstract/document/4418297.

[7] Bishwajit Purkaystha, Tapos Datta and Md Saiful Islam, "Bengali handwritten character recognition using deep convolutional neural network", in *Proceedings of the 20th IEEE International Conference of Computer and Information Technology 2017 (ICCIT '17)*, 22-24 December 2017, Dhaka, Bangladesh, E-ISBN:978-1-5386-1150-0, Print on Demand(PoD) ISBN: 978-1-5386-1151-7, USB ISBN: 978-1-5386-1149-4, DOI:10.1109/ICCITECHN.2017.8281853, pp. 01-05, Published by IEEE, Available: https://ieeexplore.ieee.org/abstract/document/8281853.

[8] Md. Anwar Hossain, Mirza A. F. M. Rashidul Hasan, A. F. M. Zainul Abadin and Nafiul Fatta, "Bangla Handwritten Characters Recognition Using Convolutional Neural Network", *Australian Journal of Engineering and*

Innovative Technology, Print ISSN: 2663-7790, Online ISSN: 2663-7804, pp. 27–31, Vol. 4, No. 2, 31th March 2022, Published by UniversePG, DOI:10.34104/ajeit.022.027031, Available: https://universepg.com/journal-details/317.

[9] Mujadded Al Rabbani Alif, Sabbir Ahmed and Muhammad Abul Hasan, "Isolated Bangla handwritten character recognition with convolutional neural network", in *Proceedings of the 20th IEEE International Conference of Computer and Information Technology 2017 (ICCIT '17)*, 22-24 December 2017, Dhaka, Bangladesh, E-ISBN:978-1-5386-1150-0, Print on Demand(PoD) ISBN: 978-1-5386-1151-7, DOI:10.1109/ICCITECHN.2017.8281823, pp. 01-06, Published by IEEE, Available: https://ieeexplore.ieee.org/abstract/document/8281823.

[10] AKM Shahariar Azad Rabby, Sheikh Abujar, Sadeka Haque and Syed Akhter Hossain, "Bangla Handwritten Digit Recognition Using Convolutional Neural Network", In *Advances in Intelligent Systems and Computing: Emerging Technologies in Data Mining and Information Security*, Singapore: Springer Nature, 2018, Vol. 755, pp 111–122, Print ISBN: 978-981-13-1950-1, Online ISBN: 978-981-13-1951-8, DOI: 10.1007/978-981-13-1951-8_11, Available: https://link.springer.com/chapter/10.1007/978-981-13-1951-8_11.

[11] R. Pradeep Kumar Reddy and C .Naga Raju, "Comparative Analysis of Handwritten Digit Recognition Using Logistic Regression, SVM, KNN and CNN Algorithms", *Journal of Science and Technology*, ISSN: 2456-5660, pp. 94-102, Vol. 6, No. 6, November-December 2021, Published by Longman Publishers, DOI: doi.org/10.46243/jst.2021.v6.i06.pp94-102, Available: https://jst.org.in/previous-issue.php?id=40.

[12] Rumman Rashid Chowdhury, Mohammad Shahadat Hossain, Raihan ul Islam, Karl Andersson and Sazzad Hossain, "Bangla Handwritten Character Recognition using Convolutional Neural Network with Data Augmentation", in *Proceedings of the 8th IEEE International Conference on Informatics, Electronics & Vision 2019 (ICIEV '19) and 3rd International Conference on Imaging, Vision & Pattern Recognition 2019 (icIVPR '19)*, 30 May 2019 - 02 June 2019, Spokane, WA, USA, E-ISBN:978-1-7281-0788-2, Print on Demand(PoD) ISBN: 978-1-7281-0789-9, Print ISBN: 978-1-7281-0786-8, DOI: 10.1109/ICIEV.2019.8858545, pp. 318-323, Published by IEEE, Available: https://ieeexplore.ieee.org/abstract/document/8858545.

[13] Chandrika Saha, Rahat Hossain Faisal and Md. Mostafijur Rahman, "Bangla Handwritten Basic Character Recognition Using Deep Convolutional Neural Network", in *Proceedings of the 8th IEEE International Conference on Informatics, Electronics & Vision 2019 (ICIEV '19) and 3rd International Conference on Imaging, Vision & Pattern Recognition 2019 (icIVPR '19)*, 30 May 2019 - 02 June 2019, Spokane, WA, USA, E-ISBN:978-1-7281-0788-2, Print on Demand(PoD) ISBN: 978-1-7281-0789-9, Print ISBN: 978-1-7281-0786-8, DOI: 10.1109/ICIEV.2019.8858575, pp. 190-195, Published by IEEE, Available: https://ieeexplore.ieee.org/abstract/document/8858575.

[14] Tanuja Kumari, Yatharth Vardan, Prashant Giridhar Shambharkar and Yash Gandhi, "Comparative Study on Handwritten Digit Recognition Classifier Using CNN and Machine Learning Algorithms", in *Proceedings of the 6th IEEE International Conference on Computing Methodologies and Communication 2022 (ICCMC '22)*, 29-31 March 2022, Erode, India, E-ISBN:978-1-6654-1028-1, Print on Demand(PoD) ISBN: 978-1-6654-1029-8, DVD ISBN: 978-1-6654-1027-4, DOI: 10.1109/ICCMC53470.2022.9753756, pp. 882-888, Published by IEEE, Available: https://ieeexplore.ieee.org/abstract/document/9753756.

[15] María Teresa García-Ordás, José Alberto Benítez-Andrades, Isaías García-Rodríguez, Carmen Benavides and Héctor Alaiz-Moretón, "Detecting Respiratory Pathologies Using Convolutional Neural Networks and Variational Autoencoders for Unbalancing Data", *Sensors*, ISSN: 1424-8220, p. 1214, Vol. 20, No. 4, 22th February 2020, Published by MDPI, DOI: 10.3390/s20041214, Available: https://www.mdpi.com/1424-8220/20/4/1214.

[16] Md Zahangir Alom, Paheding Sidike, Tarek M. Taha and Vijayan K. Asari, "Handwritten Bangla Character Recognition Using the State-of-the-Art Deep Convolutional Neural Networks", *Computational Intelligence and Neuroscience*, Print ISSN: 1687-5265, Online ISSN: 1687-5273, pp. 1-13, Vol. 2018, 27th August 2018, Published by Hindawi, DOI: 10.1155/2018/6747098, Available: https://doi.org/10.1155/2018/6747098.

[17] Dominik Scherer, Andreas Müller and Sven Behnke, "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition", in *Lecture Notes in Computer Science (LNTCS)*, vol. 6354, Online ISBN: 978-3-642-15825-4, Print ISBN: 978-3-642-15824-7, Series Print ISSN: 0302-9743, Series Online ISSN: 1611-3349, DOI: 10.1007/978-3-642-15825-4_10, pp. 92–101, 2010, Published by Springer-Verlag, Available: https://link.springer.com/chapter/10.1007/978-3-642-15825-4_10.

[18] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy *et al.*, "Recent advances in convolutional neural networks", *Pattern Recognition*, pp. 354–377, Vol. 77, 1st May 2018, DOI: 10.1016/j.patcog.2017.10.013, Available: https://www.sciencedirect.com/science/article/abs/pii/S0031320317304120.

[19] Vignesh Thakkar, Suman Tewary and Chandan Chakraborty, "Batch Normalization in Convolutional Neural Networks — A comparative study with CIFAR-10 data", in *Proceedings of the 5th IEEE International Conference on Emerging Applications of Information Technology 2018 (EAIT '18)*, 12-13 January 2018, Kolkata, India, E-ISBN:978-1-5386-3719-7, Print on Demand(PoD) ISBN: 978-1-5386-3720-3, DOI: 10.1109/EAIT.2018.8470438, pp. 01-05, Published by IEEE, Available: https://ieeexplore.ieee.org/abstract/document/8470438.

[20] AKM Shahariar Azad Rabby, Sadeka Haque, Md. Sanzidul Islam, Sheikh Abujar and Syed Akhter Hossain, "Ekush: A Multipurpose and Multitype Comprehensive Database for Online Off-Line Bangla Handwritten

Characters", In *Communications in Computer and Information Science: Recent Trends in Image Processing and Pattern Recognition*, Singapore: Springer Nature, 17th July 2019, Vol. 1037, ch. 14, pp 149–158, Print ISBN: 978-981-13-9186-6, Online ISBN: 978-981-13-9187-3, DOI: 10.1007/978-981-13-9187-3_14, Available: https://link.springer.com/chapter/10.1007/978-981-13-9187-3_14.

[21] Mithun Biswas, Rafiqul Islam, Gautam Kumar Shom, Md Shopon, Nabeel Mohammed *et al.*, "Banglalekha-isolated: A Multi-purpose comprehensive dataset of Handwritten Bangla Isolated Characters", *Data in Brief*, ISSN: 2352-3409, pp. 103-107, Vol. 12, 29th March 2017, Published by Elsevier, DOI: 10.1016/j.dib.2017.03.035, Available: https://doi.org/10.1016/j.dib.2017.03.035.

[22] Luis Perez and Jason Wang, "The effectiveness of data augmentation in image classification using deep learning", *Computing Research Repository*, Online ISSN: 2331-8422, Vol. 1712.04621, 13th August 2018, DOI: 10.48550/arXiv.1712.04621, Available: https://dblp.org/rec/journals/corr/abs-1712-04621.

[23] Savita Ahlawat, Amit Choudhary, Anand Nayyar, Saurabh Singh and Byungun Yoon, "Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)", *Sensors*, ISSN: 1424-8220, p. 3344, Vol. 20, No. 12, 12th June 2020, Published by MDPI, DOI: 10.3390/s20123344, Available: https://www.mdpi.com/1424-8220/20/12/3344.

[24] Md. Rajibul Islam, Md. Asif Mahmod Tusher Siddique, Md Amiruzzaman, M. AbdullahAl-Wadud, Shah Murtaza Rashid Al Masud *et al.*, "An Efficient Technique for Recognizing Tomato Leaf Disease Based on the Most Effective Deep CNN Hyperparameters", *Annals of Emerging Technologies in Computing (AETiC)*, Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp. 1–14, Vol. 7, No. 1, 1st January 2023, Published by International Association for Educators and Researchers (IAER), DOI: 10.33166/aetic.2023.01.001, Available: http://aetic.theiaer.org/archive/v7/v7n1/p1.html.

[25] S M Azizul Hakim and Asaduzzaman, "Handwritten Bangla Numeral and Basic Character Recognition Using Deep Convolutional Neural Network", in *Proceedings of the IEEE International Conference on Electrical, Computer and Communication Engineering 2019 (ECCE '19)*, 07-09 February 2019, Cox's Bazar, Bangladesh, E-ISBN: 978-1-5386-9111-3, Print on Demand(PoD) ISBN: 978-1-5386-9112-0, DOI: 10.1109/ECACE.2019.8679243, pp. 01-06, Published by IEEE, Available: https://ieeexplore.ieee.org/abstract/document/8679243.