Offline Handwritten Character Recognition Including Compound Character from Scanned Document

Abstract

Recognizing the handwritten characters and converting them into machine-editable text is very tedious due to the diversity of writing styles and character patterns. Extracting data from images and identifying the characters becomes more complicated when a language consists of compound structures and characters, such as Bengali. There has been a lack of programs for recognizing Bengali scripted basic and complex numeric signs and letters with high accuracy. This paper develops a novel approach to extracting and identifying Bengali handwritten primary characters, digits, and primarily used compound characters. In this proposed model, an image containing Bengali handwritten text takes as input and processed. Then processed images are segmented into lines and characters. The features are extracted from segmented characters and recognized using a Convolutional Neural Network (CNN). The CNN obtains 98.23% accuracy in the training dataset and 96.02% in the validation dataset. Apart from that, the proposed model has gained 89.6% precision and 92.6% recall scores on scanned image data.

Keywords: Character Recognition; Computer vision; Image Processing; Artificial Intelligence; Deep Learning.

1. Introduction

OCR (optical character recognition) is a technique that solves the difficulty of recognizing a wide range of characters. Handwritten and printed characters can be detected and translated into a digital data format that machines can read. It is a method of converting photographs, photos of letters, typewritten material, or vehicle number plates into information that a computer or other device can easily understand [1]. For processing typewritten text, OCR is used. OCR takes the typewritten text data and processes it to make it computer-readable. It mainly recognizes individual characters. To acknowledge a whole typewritten text word optical word recognition technique is used. This OCR is primarily used for digital dictionaries, identifying words and making suggestions. Intelligent Character Recognition is mainly used for handwritten documents. It recognizes individual characters from a scanned document. From scanned papers, intelligent word recognition finds handwritten words[2]. Handwriting character recognition is a subset of OCR that focuses on a computer's ability to accept and interpret comprehensible handwritten input from various sources, including paper documents, pictures, touch screens, and other devices.

The primary task for recognizing handwritten characters is to segment the surface and then classify those characters. Character segmentation is a process that attempts to break down a picture of a series of characters into individual symbol sub-images[3]. There are several segmentation techniques, and they can be broadly classified into three categories [4]. Histogram grounded projection analysis in the explicit segmentation category is a well-known approach for segmenting lines and characters from images. It may also be employed during the detection stage[5]. Convolutional Neural Networks (CNN) and other deep learning approaches have significantly advanced state-of-the-art voice recognition, visual object identification, object detection, and many other discplines[6]. Optical character recognition (OCR) is critical for quickly converting handwritten materials into digital text files.

There, one can find many handwritten character recognition models or systems for English, but it is hard to find a model with reasonable accuracy for the Bengali language. Even though Bengali is one of the major spoken languages in the world (around 228 million native speakers) and it is recognized as the 7^{th} largest native language, little work exists on OCR. This is because of the longhand nature and complex structure of Bengali characters. In the Bengali language, there are 11 vowels, 49 consonants, and they also contain compound alphabets (Juktakkhors). This paper develops a system to recognize handwritten Bengali characters, including compound characters. This system is necessary because it can be used to detect characters by reviewing historical writings, bank cheques, Certificate verification, and many day-to-day work-related documents. This has been done by using several Computer vision methods and Deep learning algorithms such as Convolutional Neural Network (CNN). It provides better performance on object detection and recognition. A better neural network architecture offers better accuracy and efficiency.

The remaining part of the paper is organized as follows: Section 2 offers a brief review of related work in the field of character recognition using neural networks. Section 3 sets out in detail the examined methodology and different techniques. Section 4 shows the experimental results and error analysis; section 5 consists of the conclusion and future improvements.

2. Related Work

Contributions Robust methods for character segmentation and recognition for multilingual (Latin and Devanagari) Indian document images have been developed previously by Parul Sahare & Sanjay B. Dhok, 2019 [7]. Kumar, Jindal, and Sharma proposed an effective offline handwritten Gurmukhi letter identification system based on diagonal features and transitional characteristics using a k-NN classifier[8]. The distribution of points on a character's bitmap image determined the character's diagonal and transitional properties. Using diagonal features and a k-NN classifier, the suggested system achieved maximum recognition accuracy of 94.12%. The idea of utilizing k nearest neighbor for Bengali handwriting is devised from this paper.

Ms. Snehal Pachpande and Prof. Anagha Chaudhari, 2017 [1] presented a segmentation technique of words from the Devanagari script. The image was scanned using a flatbed scanner and binarized to convert to a binary representation of 0s and 1s. On a binarized picture, some distortions were removed using a median filter. The study found that the histogram methodology is the most accurate segmentation method, with 98.1 percent accuracy for word segmentation.

An offline handwriting recognition system for Bangla script using sequential detection of characters and diacritics with a Faster R-CNN was presented by Nishatul Majid and Elisa H. Barney Smith [9]. In a fully segmentation-free technique, the characters and related diacritics were recognized individually using various networks termed C-Net and D-Net. Both networks were prepared with transfer learning from VGG-16. They employed essay scripts from the Boise State Bangla Handwriting Dataset and traditional data augmentation approaches for training and testing. The F1 scores for the C-Net and D-Net networks are 89.6% and 93.2%, respectively. These detection modules were combined into a word recognition unit with a CER of 11.2% and a WER of 24.4. A spell checker reduced the number of mistakes to 8.9% and 21.5 %, respectively.

This approach is likely to work on numerous additional Abugida scripts related to Bangla.Sadia Chowdhury et al.[10] suggested a system that accepts a scanned picture of Bengali handwritten text as input and outputs an editable version of that text after processing. The system is divided into many parts, the most important of which are image processing, machine learning via neural network training, and finally, detection of Bengali letters. The significant lack of this research is that it cannot identify the compound Bengali alphabet (Juktakkhors).

A Convolutional Neural Net architecture model called “EkushNet” was proposed by Azad Rabby et al.,2018 [11], which can recognize fundamental Bengali letters, modifiers, digits, and familiar complex characters written in Bengali handwriting. It achieves recognition accuracy of 97.73% on the Ekush dataset and 95.01% cross-validation accuracy on the CMATERdb dataset.

In this research, the proposed model is divided into different modules, and each module is further divided into smaller portions for programming benefit. The modular approach helps to detect programming errors easily. Our research is divided into three main modules: image preprocessing, Segmentation and Training of the Neural Network model & character recognition. By considering the result of each module, we move forward to the next step, and those main modules are also divided into some other submodules.

3.1 Image Preprocessing

Preprocessing is one of the most critical aspects of OCR since the accuracy of the OCR system is strongly dependent on the quality of Preprocessing and Segmentation.

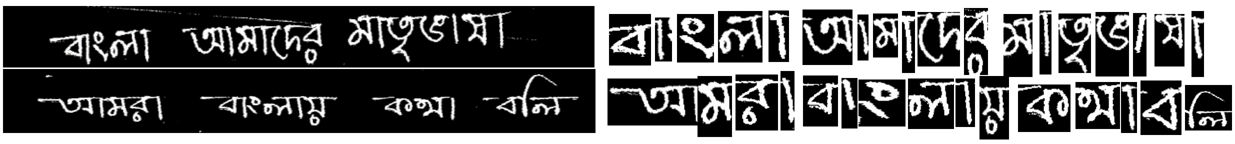
The first step, Binarization, is converting a colored image into an image consisting only of black and white pixels. It helps identify the letter from the background and makes the processing more accessible and time complexity lesser. Sometimes a scanned document might be slightly skewed, which means the image is aligned at a certain angle with horizontal. In the Noise reduction stage, we smoothen the picture by eliminating smaller dots/patches with higher intensity than the rest. Because various writers have distinct writing styles and hence varying stroke widths, dilation is used to adjust the consistency of stroke width.

(a) (b)

Fig. 1. Preprocessed image (a) Original scanned image, (b) image after preprocessing.

3.2 Segmentation

The ability to recognize characters is highly dependent on the accurate segmentation of a document. Our research uses a process to segment the scanned copy in lines and characters using conventional horizontal and vertical projection profile algorithms. Character segmentation is quicker than the traditional segmenting of all characters from a text using linked component processing. The process of segmentation includes the following steps:



(a) (b)

Fig. 2. (a) Line level, (b) Character level segmentation of the preprocessed image

After the segmentation, we save these segmented characters into separate folders for further processing and later feed these images to our neural network model to recognize the handwritten character.

3.3 Training of Neural Network Model & Character Recognition

Training neural network models and character recognition is our system's last and most crucial stage. It can be treated as the brain of the whole system. It acts like a human brain to learn the pattern by accounting for many examples of the same thing; after correct learning, it can recognize those patterns. Thus, how neural network exhibits intelligence. The Neural Network Model and character recognition module training consist of several essential submodules. They play an indispensable role in recognizing characters with a high accuracy rate.

3.3.1 Dataset Preparing

To train a Neural network model, we need to prepare a dataset. The preparation of the dataset reduces the computational complexity and allows for better performance in less time. This research uses the publicly available Ekush [11] dataset to train and test our model. We also provide our own customize data for cross-validation. This dataset is vast and has numerous raw sample images for each Bengali alphabet. Ekush data set contain ten modifiers, 50 primary characters, 52 compound characters, and ten digits. So, in total, we use 122 classes for training and testing the neural network. As mentioned earlier, binarization makes processing more accessible and less time complex. We apply binarization to the datasets, which helps to identify the letter from the background. The binarization results in the image's background being black, and the characters are white.

Fig. 3. Training data sample

Resizing is another crucial aspect of dataset preparation as the images of the collected dataset are different in size. We have resized these images into 32×32 pixels. In the final stage of dataset preparation, we split the dataset into training and testing sets. For our research, we use 36604 image files belonging to 122 classes. We split our data set 80% for training and 20% for testing. So, we have 29284 images for training and 7320 images for testing/ validation.

```
Found 36604 files belonging to 122 classes.
Using 29284 files for training.
Found 36604 files belonging to 122 classes.
Using 7320 files for validation.
```

3.3.2 Train the Model

Our used model is a supervised deep learning neural network model. That means it learns for all weights and biases from labeled data. This way, our system learns characters' patterns by extracting the features from given images. The primary computational part of our system is done in this section.

I. Neural Network Architecture

We use a multilayer Convolutional Neural network for training our neural network model and classify Bangla handwritten characters into 122 classes. Here we use the convolution layer, max-pooling layer, batch normalization, and fully connected dense layer. In our CNN architecture, we define our model as a sequential classifier. Layer 1 and layer 2 of the classifier is a 2D convolutional layer with a filter size of 64, kernel size of (3,3), and ReLu is used as an activation function. Following these two layers, we use a MaxPooling layer with pool size (2,2) and a dropout layer with about 15% drop in layer three and layer 4. This dropout layer is used to prevent the overfitting of data into training.

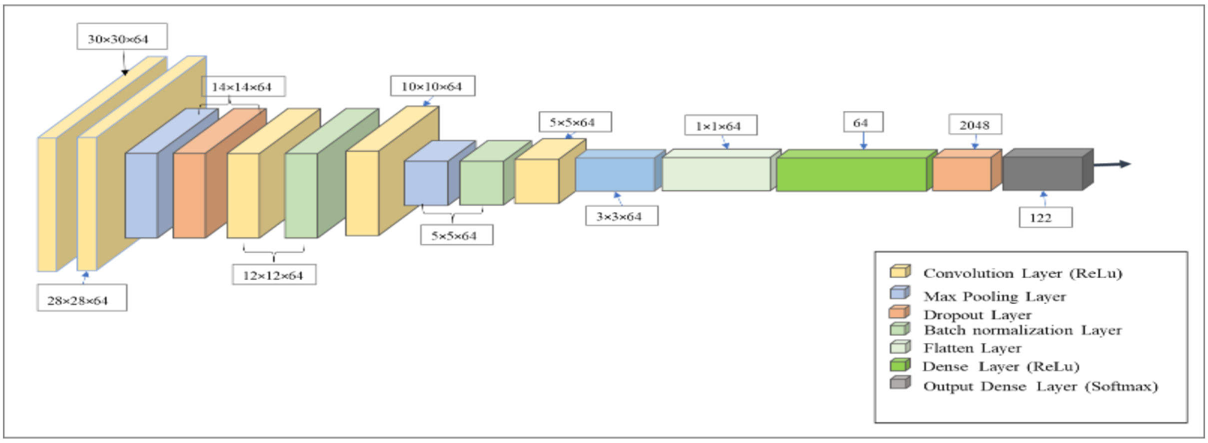


Fig. 4. CNN Architecture

We use a convolutional layer with a filter size of 64, kernel size (3,3), and ReLu activation function in layer five. We apply batch normalization to regularize the training data in layer six. Next, layer 7 uses a convolutional layer with the same configuration as layer 5. Layers 8 and 9 use the same structure as layers three and 6. The same thing is repeated in layers 10 and 11 as in layers 7 and 8.

After these 11 operations, we transform the output into a flattened array by applying it to the flattened layer, which is layer 12. Then it is passed through a fully connected dense layer with 2048 hidden units and ReLu as an activation function. After this, we applied 15% dropout. In the final layer, we apply a fully connected dense layer with 122 units as we have 122 classes and use softmax as an activation function for the multiclass classifier.

The softmax function is defined as below:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Where k = number of classes, z_i = input vector;

II. Optimization

We use optimizer algorithms to optimize our CNN model, which eventually helps reduce the error rate. The Adam optimizer updates its weights using the following equation,.

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where w = model weights

η = Step size (depend on iteration),

III. Loss Calculation

We use categorical cross-entropy to calculate the loss of our multiclass classification model. These are problems in which an example can only fit into one of several potential categories, and the model must choose one. Categorical cross-entropy finds the loss using the following equation;

$$CE = -\sum_{i}^{c} t_i \log\left(f(s_i)\right)$$

where t_i and s_i = ground truth

IV. Training the Model

We train our model on the dataset with a batch size of 32. While splitting the dataset, we keep the shuffle “True” with a seed 128. The optimization algorithm helps to reduce the error and converge faster by decreasing the learning rate. After running 70 epochs, we obtain a good accuracy with our training dataset.

3.4 Testing the Model

In the final part, we evaluate our model with various testing data. First, we test our model with the testing data spliced from the primary data set. Then we try our model with the data taken as a scanned document and segmented into characters. We measure how accurately our model predicts these custom data that belong to a specific class.

3.5 Input data into The System

We take a photo of the document with Bangla handwritten characters in our system. The document contains all modifiers, essential characters, digits, and compound Bangla characters. We also take a photo of the document containing two Bangla sentences in two lines. These images go through image preprocessing, segmentation, and finally into our trained model to predict the character. Then we analyze the result with different metrics.

4. Experimental Result & Error Analysis

The result analysis is divided into two-section, one for the result of the segmentation process and another is convolutional neural network model performance.

4.1 Segmentation

Our used method for segmentation shows excellent performance on given input data. Our model offers almost 100% accuracy in segmenting the data from the given document example where the overlapping character is avoided. To find the accuracy, we use the following equation;

$$\text{Segmentation Accuracy} = \frac{\text{Number of correctly segmented character}}{\text{Total character in document}}$$

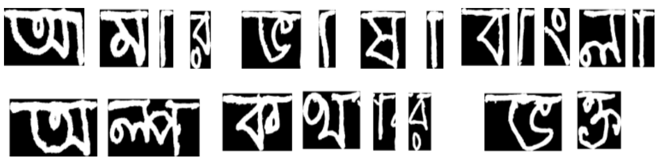


Fig. 5. The segmentation output for the second document.

4.2 Performance of Neural Network

The second and most important analysis of our system is measuring the performance of the CNN model. We use different measurement factors for the classification technique to evaluate our neural network model.

4.2.1 Measurement Factor of Classification Technique

Some evaluation factors are used to analyze the classification problem in deep learning. Some evaluation factors include accuracy, sensitivity, specificity, precision, f1 measure, recall, etc. The confusion matrix determines the exhibition evaluation variables. The evaluation variables are **True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).**

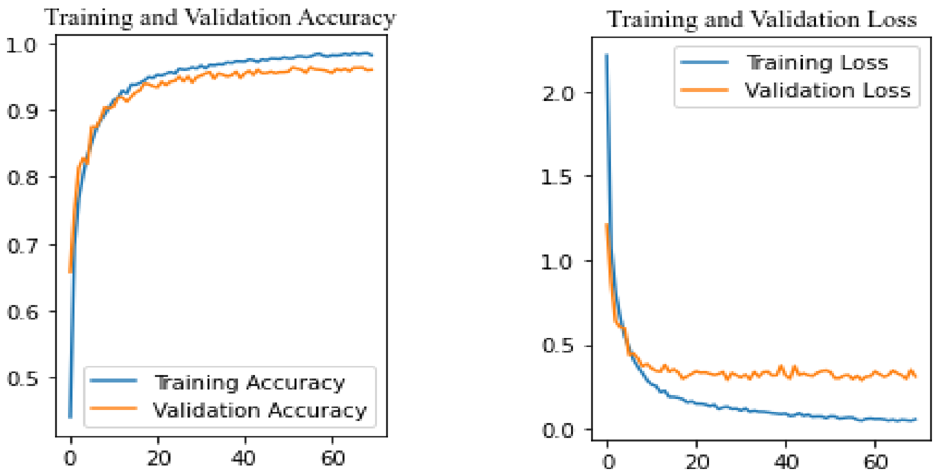
4.2.2 Accuracy

Accuracy is the ratio of correct prediction and the total number of classes. The basic equation for calculating the accuracy equation is as follows;

$$\textbf{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

After evaluating our model, it shows our model has obtained 98.23% accuracy on the training set and 96.02% on the validation set, which we get after running the classifier model 70 epochs. We can visualize the training process from the figure given below;

Figure 6.(a) shows that the validation accuracy was higher than the training accuracy. But in the overall scenario, they both have poor accuracy in the initial stage, as our model trains on more input data features, the training and validation accuracy increases. But the accuracy of validation is less than the accuracy of training, which is a good sign that our model doesn't overfit on training data.



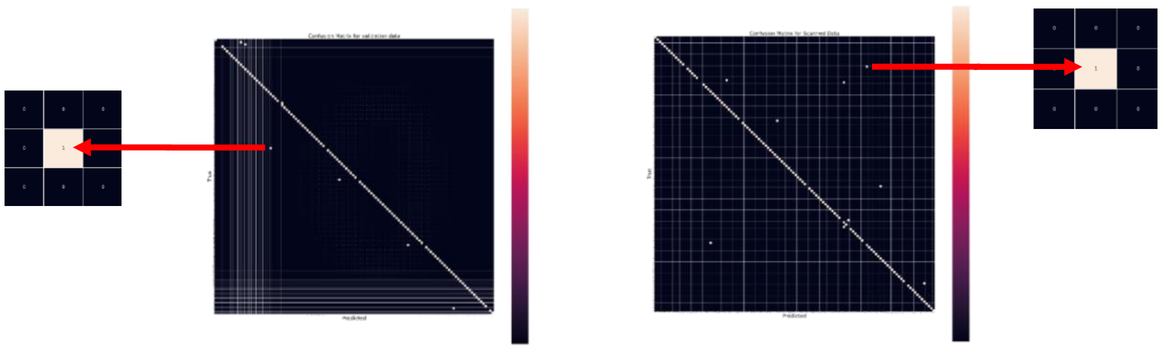
(a) (b)

Fig. 6. (a) Comparison between training and validation accuracy. (b) Training and validation loss.

Figure 6.(b) represent the comparison between training loss and validation loss. It shows that the training loss and the validation loss were much higher in the initial stage; with every epoch, both the validation and training loss decreased. Another thing that can be defined from the graph is that the validation loss is still higher than the training loss. It is because our training data doesn't overfit.

4.2.3 Confusion Matrix

The confusion matrix table briefly describes the predicted outcome for the classification problem. We construct two confusion matrixes; one is for the data we split for validation during the train test split, and another is for the data we extracted from the scanned document with individual Bangla characters.



(a) (b)

Fig. 7. (a)Confusion Matrix for validation data. (b) Scanned data.

The first confusion matrix shows that it predicts 93% of data correctly, and 7% of data was miss labeled in the validation data set. The image in figure 7.(b) shows that it predicts 91% of data correctly, and 9% of data was miss labeled in the data set of the input image.

4.2.4 Precision

Precision gives a positive predicted value. That means it shows the ability of a model to predict correctly out of all all-positive predictions. The equation to calculate the precision score is as follows;

$$\boldsymbol{Precision} = \frac{True\ Positive}{True\ positive + False\ Positive}$$

We calculated the precision score for both the validation data and the data from the scanned input image. It gives the score as below;

Table 1 Precision Score	Precision score
Ekush (validation set)	91.4%
Scanned document data	89.6%

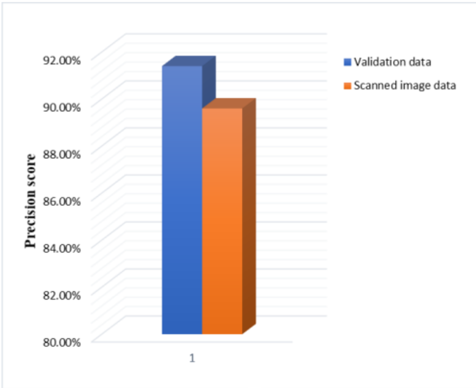


Fig. 8. Comparison of a precision score of validation and scanned image data.

Figure (4.6) compares validation data with a slightly better precision score than our scanned image data. but it is still a good precision score for the input scanned image data. It implies data our model actually can detect 89.6% of data correctly out of all positive predicted data.

4.2.5 Recall

Recall score gives a value that shows the ability of a model to predict positive classes out of actual positive classes. The equation for calculating the recall score is as below;

$$\textbf{Recall} = \frac{True\ Positive}{True\ positive + False\ Negative}$$

We also calculated the recall sore for both the validation data and the data from the scanned input image using the above-described equation and methods. It gives the score as below;

Table 2 Recall Score	Recall score
Ekush (validation set)	94.3%
Scanned document data	92.6%

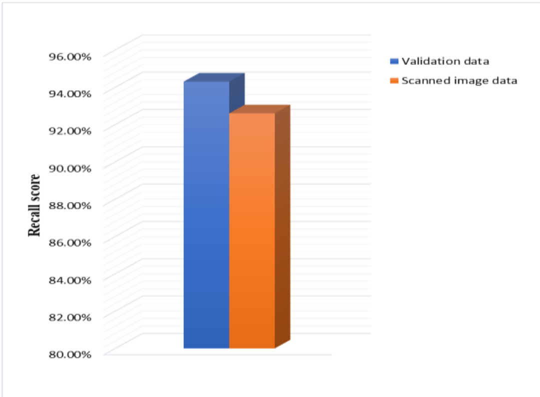


Fig. 9. Comparison of a Recall score of validation and scanned image data.

Figure (4.7) compares validation data with a slightly better recall score than our scanned image data. But it is still a good precision score for the input scanned image data. It implies data our model actually can detect 92.6% of data correctly as Positive to the total number of Positive samples.

4.2.6 F1 Score

F1 measures the precision of the model by a blend of precision and recall. That means it takes false positive and false negative to calculate the F1 score. Often f1 score is calculated to tune the precision and recall value. By the following equation, we can calculate the F1 score;

$$\textbf{F1 Score} = \frac{2 * (Recallonecision)}{(Recall + Precision)}$$

To test our model, we calculated the F1 Score for both the validation dataset and the scanned image dataset. We already have calculated the recall and precision score for both values; We use these values to calculate the F1 score for both dataset and has the score as below:

Table 3 F1 Score	F1 score
Ekush (validation set)	92.3%
Scanned document data	90.6%

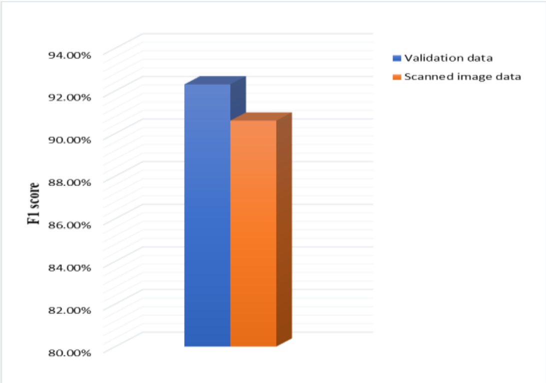


Fig. 10. Comparison of F1 score of validation and scanned image data.

4.3 Result Comparison in different Model.

There is no exactly existing system like our proposed system which take scanned document containing all types of Bengali handwritten characters and recognize them. In the following table, we present some comparisons on accuracy between some previous proposed work on different Bangla characters datasets.

Table 4 Result Comparison.

Previous Work	Classifier	Accuracy
Handwritten Bangla Compound Character Recognition Using Gradient Feature[12]	MQDF	80.90%
Bangla Handwritten Character Recognition using Convolutional Neural Network [13]	BHCR-CNN	85.96%
Proposed Method	CNN	96.02%

Here we also provide a comparison (Table: 5) of the accuracy of our model with different models evaluating with the same dataset. In the MLP model (13,5) hidden layer and for SVM nonlinear kernel ‘rbf’ were used. The comparison table shows that our proposed model out-performs other traditional machine learning model in terms of accuracy.

Table 5: Comparison with other Models.

Model	Accuracy
Multilayer perceptron (MLP)	62.54%
Support vector machine (SVM)	63.11%
Linear Discriminant Analysis (LDA)	64.7%
Random Forest	66.22%
sProposed CNN Method	96.02%

4.4 Error Analysis

Though our model gives a good performance measurement, there are some errors. In the segmentation process, our algorithm sometimes cannot segment the character where one character overlaps with another character.

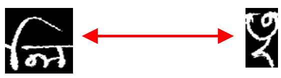


Fig. 11. Overlapped character segmentation error.

This is because while our algorithm counts the pixel values of the foreground of the image, it sometimes is unable to distinguish these different pixel pixels from another character

5. Conclusion

In this research work, we have developed a model and analyzed the model using different algorithms and metrics. We mentioned some errors in result analysis where our segmentation technique is not performing well in some scenarios. This research work also analyzes the character recognition problem using different algorithms in different modules. These algorithms give different results and help to achieve the optimal result. From those scores, we can conclude that our proposed model has a state-of-the-art performance in related fields. Our neural network algorithm is still time-consuming and slows down the whole system. In future work, we aim to optimize our neural network algorithm and comprehensively build a segmentation technique to get a more optimal result.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by the Department of Information and Communication Engineering at Noakhali Science and Technology University, Bangladesh. The authors are immensely grateful to the university for providing the research facilities and learning environment.

References

[1] Ms. S. Pachpande and A. Prof. Chaudhari, Implementation of Devanagri Character Recognition System Through Pattern Recognition Techniques,2017 International Conference on Trends in Electronics and Informatics (ICEI). IEEE, 2017.

[2] C. C. Tappert, C. Y. Suen, and T. Wakahara, “The State of the Art in On-Line Handwriting Recognition,” 1990.

[3] A. Rehman and T. Saba, “Off-line cursive script recognition: Current advances, comparisons and remaining problems,” Artificial Intelligence Review, vol. 37, no. 4, pp. 261–288, Apr. 2012, doi: 10.1007/s10462-011-9229-7.

[4] A. Choudhary, “A Review of Various Character Segmentation Techniques for Cursive Handwritten Words Recognition,” 2014. [Online]. Available: http://www.irphouse.com

[5] B. Hossain, F. Naznin, Y. A. Joarder, Z. Islam, and J. Uddin, Recognition and Solution for Handwritten Equation Using Convolutional Neural Network.

[6] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, May 27, 2015. doi: 10.1038/nature14539.

[7] P. Sahare and S. B. Dhok, “Robust Character Segmentation and Recognition Schemes for Multilingual Indian Document Images,” IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India), vol. 36, no. 2, pp. 209–222, Mar. 2019, doi: 10.1080/02564602.2018.1450649.

[8] M. Kumar, M. K. Jindal, and R. K. Sharma, “k -Nearest Neighbor Based Offline Handwritten Gurmukhi Character RecognitionPradesh, India.,” 2011.

[9] N. Majid and E. H. B. Smith, “Segmentation-free bangla offline handwriting recognition using sequential detection of characters and diacritics with a faster R-CNN,” in Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, Sep. 2019, pp. 228–233. doi: 10.1109/ICDAR.2019.00045.

[10] S. Chowdhury, F. R. Wasee, M. S. Islam, and H. U. Zaman, “Bengali Handwriting Recognition and Conversion to Editable Text.”

[11] A. K. M. S. Azad Rabby, S. Haque, S. Abujar, and S. A. Hossain, “Ekushnet: Using convolutional neural network for Bangla handwritten recognition,” in Procedia Computer Science, 2018, vol. 143, pp. 603–610. doi: 10.1016/j.procs.2018.10.437.

[12] U. Pal, T. Wakabayashi, and F. Kimura, “Handwritten Bangla Compound Character Recognition Using Gradient Feature,” Apr. 2008, pp. 208–213. doi: 10.1109/icit.2007.62.

[13] Md. M. Rahman, M. A. H. Akhand, S. Islam, P. Chandra Shill, and M. M. Hafizur Rahman, “Bangla Handwritten Character Recognition using Convolutional Neural Network,” International Journal of Image, Graphics and Signal Processing, vol. 7, no. 8, pp. 42–49, Jul. 2015, doi: 10.5815/ijigsp.201