An Automated System for Recognizing Isolated Handwritten Bangla Characters using Deep Convolutional Neural Network

Md. Nahid Hasan
Department of Computer Science and Engineering
Varendra University
Rajshahi, Bangladesh
nahid12cse@gmail.com

Rafi Ibn Sultan
Department of Computer Science and Engineering
Varendra University
Rajshahi, Bangladesh
rafi.ruet13@gmail.com

Mohammad Kasedullah
Department of Computer Science and Engineering
Varendra University
Rajshahi, Bangladesh
kasid.raj@gmail.com

***Abstract*—Among various handwritten character recognition of different languages, Bangla stands as one of the most challenging tasks. Because of its unique texture and morphologically complex structure often classification models do not provide the expected result as one hopes to have. In this research, a deep novel Convolutional Neural Network (CNN) of 11 layers is proposed. Among the layers, 6 are convolutional layers. This deep CNN model was trained to classify Bangla basic isolated characters of 50 classes (each representing a character). The research utilized a publicly available dataset, CMATERdb 3.1.2, for its classification purpose. The model performed relatively better than the current research on this field and wielded a 98.03% accuracy on the test dataset. This improvement leads us to believe that more accurate results can be achieved in the future by working with other such kinds of datasets or by tweaking the existing model.**

Keywords—Bangla character recognition; HCR; Classification; CNN; Deep Convolution Neural Network; OCR; Pre-processing;

I. INTRODUCTION

Character recognition is the process of associating a symbolic identity to the image of a character. This recognition can be distinguished into two categories. One of them is the characters from printed text and the other one is the hand characters written by people. Recently, handwritten character recognition has gradually become one of the most prolific computer vision thesis topics among researchers worldwide [1]. Intending to create an automated system that will produce an effective enough result, which in turn will substitute the human eyes in recognition, researchers are exploiting different models and algorithms.

Although there have been many works done on major languages such as English and other popular languages, the success rate in recognizing Bangla handwritten characters is still not up to the mark [2]. It is mostly due to the unique characteristics such as strokes, styles, and structure of the characters varying widely among each language [3]. Especially handwritten character recognition (HCR) is more difficult compared to printed forms of characters [4]. Furthermore, different people's characters are unique to each person and vary in different aspects such as size, shape, and style. In the case of Bangla HCR, the complexity increases ten folds than in some specific languages, mostly because of the morphologically complex nature of Bangla characters. Bangla contains many similar shaped characters that can look the same to even some human eyes if they are new to the language. In many cases, one character differs from another with a single dot or mark. Numerous variations of Bangla language users' writing style can be observed, and the nature of various sets of characters can often be quite similar. This scenario has dramatically influenced the research and development of automated systems that try to recognize handwritten characters.

There is a total of 50 characters in the Bangla alphabet that are depicted in figure 1. Recognition of handwritten characters involves two significant steps: extracting features from the character set and then employing classification or learning tool(s) to classify individual characters. In any visual recognition task, the main challenge is feature extraction from the images. The more a model can overcome this challenge, the better the model performs, i.e., can classify more accurately. Convolutional Neural Network (CNN) is the first-ever algorithm to perform well on digit recognition tasks [4] and it has been used in this field ever since. CNN has been performing well in such image recognition challenges mostly because of its insensitiveness to translation variance and scale variance of the features that are extracted from the images [2].

In this paper, Bangla handwritten character recognition is investigated based on a deep convolution neural network (DCNN) as the image classifier. The proposed DCNN model works around to recognize 50 basics isolated (a single image will have only one character) Bangla handwritten characters as accurately as possible. The aforementioned 50 characters are depicted in figure 1, with a symbolic representation denoting the class names of every character.

অ (A)	আ (AA)	ই (I)	ঈ (II)	উ (U)	ঊ (UU)
ঋ (R)	এ (E)	ঐ (AI)	ও (O)	ঔ (AU)	
ক (KA)	খ (KHA)	গ (GA)	ঘ (GHA)	ঙ (NGA)	চ (CA)
ছ (CHA)	জ (JA)	ঝ (JHA)	ঞ (NYA)	ট (TTA)	ঠ (THA)
ড (DA)	ঢ (DHA)	ণ (NNA)	ত (TA)	থ (THA)	দ (DA)

ধ (DHA)	ন (NA)	প (PA)	ফ (PHA)	ব (BA)	ভ (BHA)
ম (MA)	য (YY)	র (RA)	ল (LA)	শ (SHA)	ষ (SSA)
স (SA)	হ (HA)	ড় (DDHA)	ঢ় (DHRA)	য় (YYA)	□ (KHAND)
ং (ANUS)	ঃ (VISARG)	ঁ (BINDU)			

Fig 1. Bangla basic characters

II. Related Works

This section provides a summary of related research work carried out so far in the area of Bangla handwritten character recognition. Purkaystha et al. [2] contributed to this field by using a variation of CNN to recognize Bangla HCR. The proposed model was a typical CNN model achieving an accuracy of 91.23% on basic alphabets. Chowdhury et al. [5] introduced data augmentation in Bangla HCR in addition to their CNN model. This greatly increased their model's accuracy comparing to their early model with no augmentation. The accuracy of the final model was found to be at 95.25%. Rabby et al. [6] introduced a multiclass CNN model for recognizing Bangla HCR with the help of graphemes. The trained model, called "Borno," performed remarkably on a dataset of more than a million characters achieving a 92.61% accuracy. Abir et al. [7] claimed to have a novel approach that combines CNN followed by an inception module and a fully connected neural network (NN). It resulted in an accuracy of 91.1% in recognizing the 50 basic Bangla characters. Hakim et al. [8] also contributed to this field by using a deep CNN Model. Their 9-layer sequential CNN model classified with a 96.64% accuracy. Finally, Saha et al. [9] also explored this field with a deep CNN model to classify isolated characters. The proposed model named BBCNet-15 achieved a 96.40% validation accuracy on the CMATERdb 3.1.2 dataset.

III. Proposed Methodology

The basic phases of our proposed method for Bangla Handwritten Character Recognition are given below.

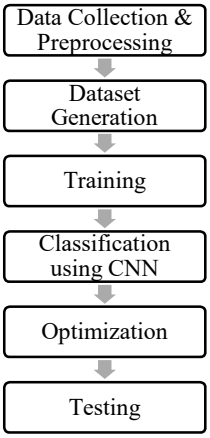

Fig 2. Proposed Methodology

A. Data Collection & Preprocessing

a) Sample Collection

In this work, we used the public dataset CMATERdb 3.1.2 [10] [11] which was created by the Center for Microprocessor Applications for Training Education and Research (CMATER), a research laboratory of Jadavpur University in Kolkata. This dataset contains 50 different Bangla isolated handwritten characters. Among the 50 basic characters, there are 39 constants and 11 vowels. The dataset consists of a Train folder and a Test folder denoting the training and testing datasets. Both of these folders have 50 subfolders mapping the 50 different characters (each folder maps to a particular character). Both in the training and testing subfolders, each contains an equal number of sample images (240 and 50 images, respectively). This balancing ensures that the class imbalance problem will not occur for the model while training. Counting all the samples of the dataset in total, there are 12,000 and 3,000 images respectively for the training and testing dataset. To see the dataset's distribution, randomly 1000 image labels were selected and are illustrated in figure 3. The labels of the images are put in the Y-Axis where each number denotes their corresponding class that they belong to. The numerical numbers denote the alphabets' classes chronically.

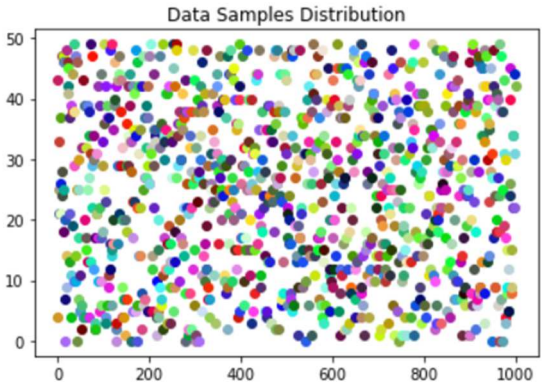

Fig 3. Distribution of the Dataset (Randomly Taken 1000 Image Labels)

The dataset contains handwritten characters that were collected from a wide range of people of different ages and sex. Therefore, it ensures that a variety of numerous forms are available for every character. Figure 4 shows several samples of images that contain the first Bangla consonant character "ক" in handwritten form.

Fig 4. Raw images of character "ক"

One sample image of every character from the dataset is also illustrated in Figures 5 and 6 to understand of the dataset.

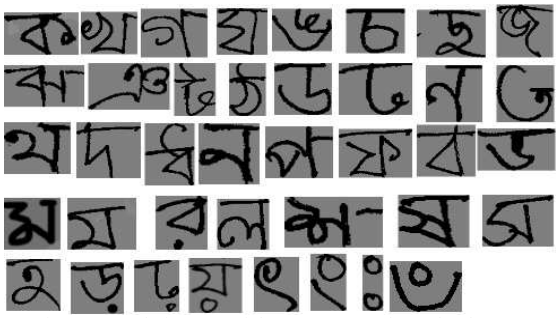

Fig 5. Sample images of the consonants from the dataset, extracted from [9]

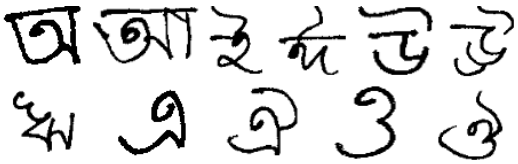

Fig 6. Sample images of the vowels from the dataset, extracted from [9]

b) *Preprocessing*

Before feeding the images into the model, some preprocessing was done to extract features from them relatively easily. At first, the raw images were converted into grayscale images. In a grayscale image, each pixel value is a single integer number ranging from 0 to 255, representing a pixel's brightness. Generally, 0 represents a black pixel, and 255 represents a white pixel. As the sizes of the collected images of the characters were different in the first place, all of them were resized into a 28×28 dimension to maintain an appropriate and equal input shape. The figure below shows some resized grayscale images of character "ক" that were used as the input of Convolution Neural Network (CNN)

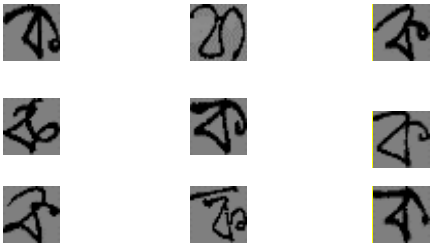

Fig 7. 28x28 resized images of character "ক"

B. *Dataset Generation*

After converting to grayscale images, we created the matrix of the images. To reduce the computational cost, we converted the pixel values to a value between 0 and 1 by dividing each value by the maximum of 255. We separated the row and the column to make many one-dimensional arrays of images. Each row of images contained a classifier number by which we could identify each character. From the dataset, there were four arrays created in the first place. The first two arrays (x_train and y_train) were prepared for training the machine and another two arrays (x_test & y_test) were made for testing. The training dataset was further split into two groups: training and validation set. From the original training dataset, 10% of the set of images was selected to create the validation dataset (x_val & y_val), which was used to validate while training the model. The testing dataset of 3000 images was reserved for the final evaluation of the proposed model. Using separate validation and test dataset, it had been made sure that no intentional bias would be imposed in the final tuned model's earned accuracy. The summary of the dataset is shown in figure 8.

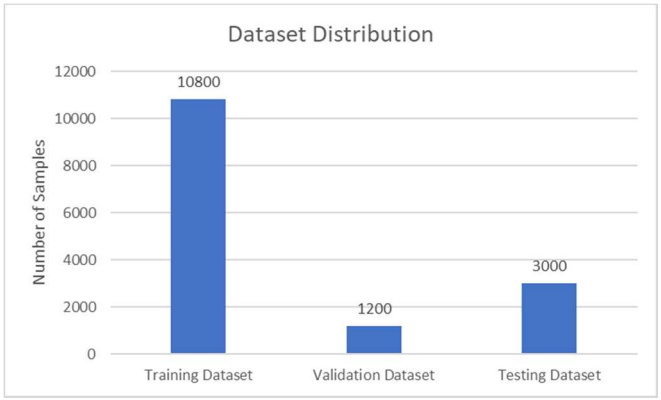

Fig 8. A Summary of the Dataset Used

C. *Classification using CNN*

As discussed earlier, this paper implements a novel deep CNN based model for the classification task. CNN works remarkably well in any computer vision classification task, making it one of the primary reasons for being used very frequently in such tasks [12]. CNN is a famous deep learning architecture that can learn features from input images without much preprocessing. Typically, a CNN architecture consists of multiple nonlinear transformations in various stages followed by a supervisor classifier at the end, which has the classifying responsibility. Sample images are fed into a CNN model and the model is trained by learning features of these images. The final output layer matches the images with the provided labels of those images. CNN works so successfully because of the concept of backpropagation. In the forward pass, the gradient generated from the mismatched errors after predicting the image labels is fed back from the output label. The parameters in different layers of the model are tuned with respect to the gradients generated in a way that errors are minimized. The model repeats this process a considerable number of times until it reaches a saturation point where the model's final accuracy can be determined.

The CNN model that was designed is depicted in figure 9. The model has 6 convolution layers where a max-pooling layer is used after two consecutive convolutional layers. There is a fully connected layer before the final output layer generating the output scores. Each layer of the proposed CNN model is briefly described below:

1. **Convolutional Layer 1:** Input data, i.e., training images of shape (28x28x1) (Here, the digit 1 identifies that the images are of greyscale) are fed through this layer. A total

of 64 filters of (3x3) size are used to convolute the images for extracting features from the provided images. The ReLU activation function is used in this layer for introducing the nonlinear properties to the network. This activation function is responsible for converting the input signal of each node of the layer to an outgoing output signal for the next layer. This activation function is appropriate for the model because of its enormous success in real life-practice of similar cases [13] as it is faster and more efficient than other similar types of activation functions. The output result's shape of the first convolution layer is 28x28x64 (Here, 64 identifies that the images are convoluted with 64 filters). Since this layer uses the "same padding" technique, the height and weight of the output shape images remain intact.

2. **Convolutional Layer 2:** The second convolutional layer has the same setup as the first convolutional with the same number and shape of the filters, the same activation function, and the same initializer.
3. **Max Pooling Layer 1:** The outputs generated by the previous layer's ReLU are now passed into this max-pooling layer (2x2). This layer is used mainly to help overcome overfitting while training the model. Such pooling layers commonly occur after two consecutive convolutional layers. This layer reduces the shape to exactly half (14x14x64). A dropout of the score (0.5) [15] is also applied to the output of the layer, which also operates to overcome the model's overfitting.
4. **Convolutional Layer 3:** This convolutional layer has the same setup as before, except it uses 128 filters (3x3). This layer converts the current size to 14x14x128 and passes it to the next convolutional layer.
5. **Convolutional Layer 4:** Just as before, this convolutional layer has the same setup as the previous convolutional layer before it.
6. **Max Pooling Layer 2:** This max-pooling layer uses the same configuration and the same dropout technique as the first max pooling. This operation again reduces the shape to exactly half (7x7x128).
7. **Convolutional Layer 5:** Like the other convolutional layers of the model, this layer also follows the same setup, except it again increases its filter numbers to 256 (3x3 size). The current shape is then converted into a 7x7x256 shape.
8. **Convolutional Layer 6:** This layer is a carbon copy of the convolutional layer preceding this.
9. **Max Pooling Layer 3:** Following the previous two max-pooling layers' footsteps, this does the same thing by reducing the current shape to exactly half. (3x3x256).
10. **Fully Connected Layer:** After the last pooling operation, the results are flattened and fed to this fully connected layer of 128 neurons.
11. **Output Layer:** The Softmax activation unit is applied to every node in this layer. Every node will generate a probability value assigned to the 50 output nodes, which will determine the image label. The highest probability score from the nodes will then be classified as the corresponding label of that image.

The total learnable parameters of the model are 1,445,746. The summary of the model is given in table 1.

TABLE I. SUMMARY OF EACH LAYER OF THE PROPOSED MODEL

Layer	Output Shape	No. of Parameters
Convolution 1	(None, 28, 28, 64)	640
Convolution 2	(None, 28, 28, 64)	36928
Max Pool 1	(None, 14, 14, 64)	0
Convolution 3	(None, 14, 14, 128)	73856
Convolution 4	(None, 14, 14, 128)	147584
Max Pool 2	(None, 7, 7, 128)	0
Convolution 5	(None, 7, 7, 256)	295168
Convolution 6	(None, 7, 7, 256)	590080
Max Pool 3	(None, 3, 3, 256)	0
Fully Connected Layer	(None, 128)	295040
Output	(None, 50)	640

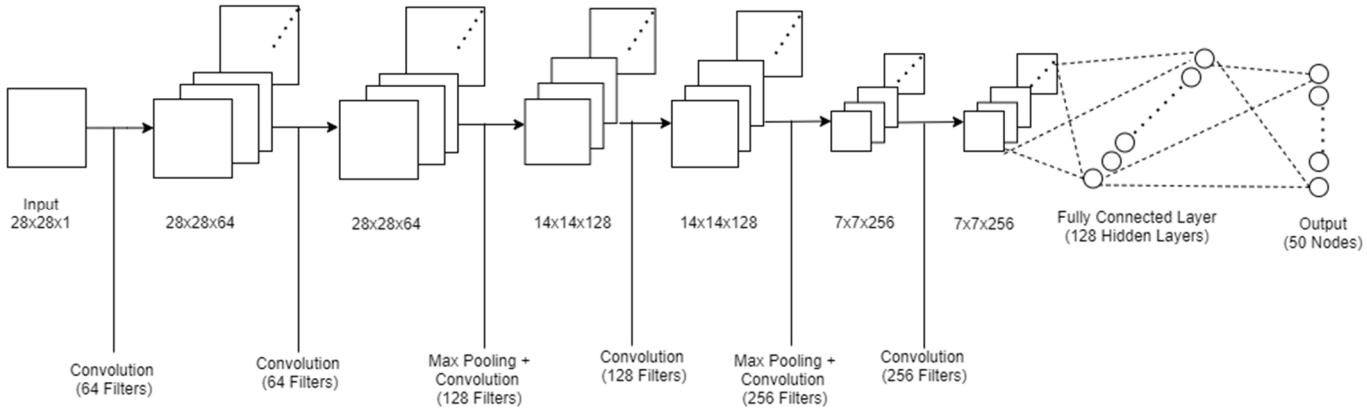

Fig 9. An illustration of the Structure of our Proposed CNN Architecture

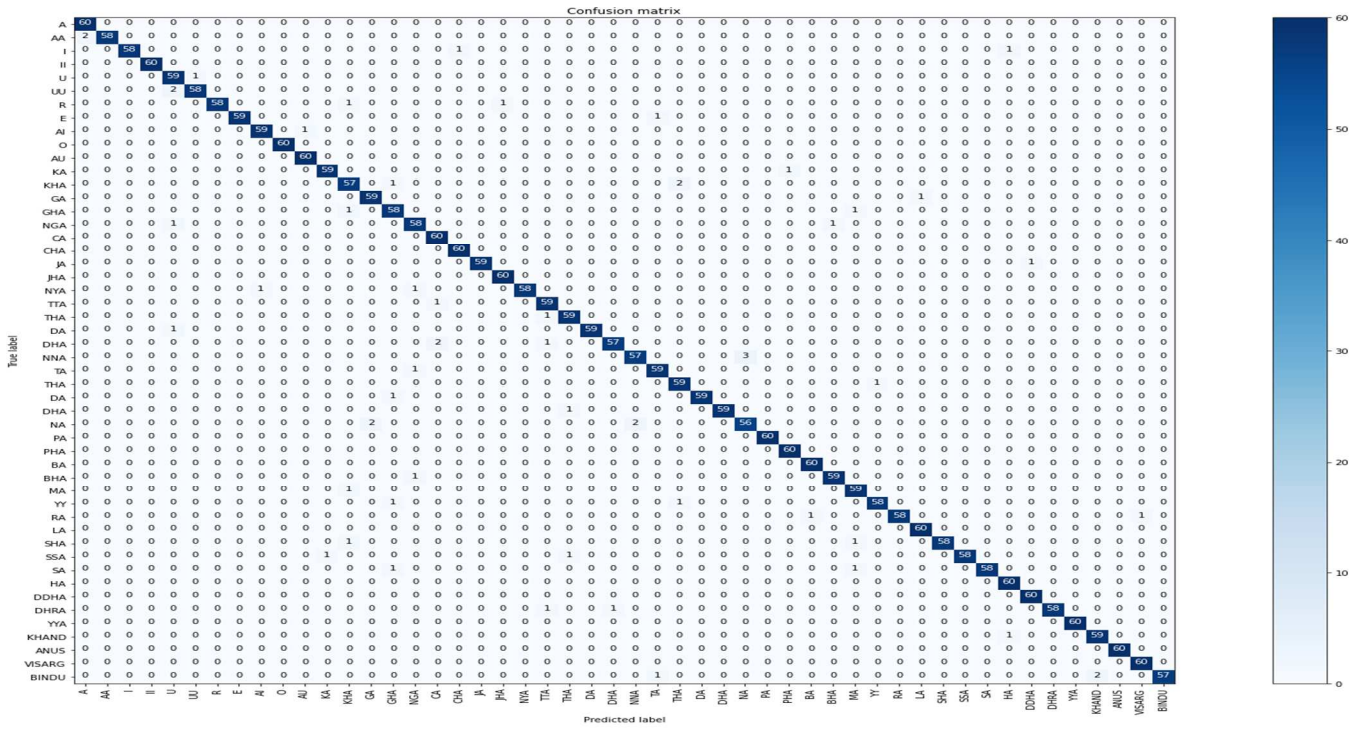

Fig 10. Confusion Matrix Produced for Test Sample

D. *Optimization*

For the training purpose, instead of having a fixed learning rate globally and equally for all parameters, the model used an adaptive learning method called Adadelta Optimizer [16]. This method provides a continuous change in the learning rate while training a model, ensuring the most effective learning rate to be used along the learning process. Adadelta Optimizer works to reduce the current learning rate when weights tend to receive high gradients while updating weights. On the other hand, if the weights receive very small or infrequent updates in the learning process, the effective learning rate will increase. The proposed CNN model was trained using the default values of Adadelta Optimizer (learning rate = 1.0, decay rate = 0.95). With opting to use the Adadelta Optimizer, the tedious work of manual tuning of the learning rate became obsolete.

E. *Training the Model*

For measuring this classification model's performance, cross-entropy loss [17] (between the labels and predictions) was applied. For initializing the weights, Xavier initialization [14] method was used for providing appropriate weight values. This method made sure that the neurons in the layers didn't start training in saturation and are kept in a reasonable range of values through different layers while initializing. The model was trained using mini-batches of size 32. We also implemented the ReduceLROnPlateau class [18], which can reduce the learning rate when the validation loss is not improving. In this model, the learning rate was decayed after every consecutive three epochs where there were no improvements. We used a total of 100 epochs to train the model.

IV. Experimental Analysis

The CNN model was trained on an NVIDIA GeForce GTX 1650 (8GB GDDR6 memory), system RAM of 8 GB. It implemented the Keras API on top of Tensorflow (CUDA toolkit 10.1.243, cuDNN v7.6.5, and Python 3.6.12). The training accuracy and validation accuracy for increasing epoch numbers are depicted in figure 11.

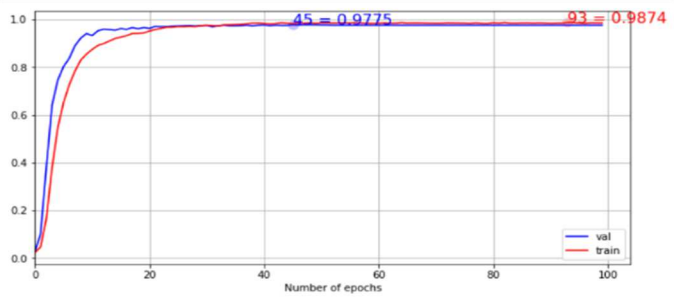

Fig 11. Training Accuracy and Validation Accuracy for increasing epochs

Figure 12 illustrates another performance measurement of the model by plotting the training loss and validation loss concerning epochs.

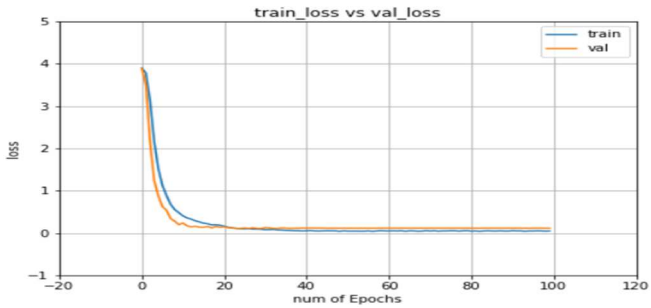

Fig 12. Training Loss vs. Validation Loss

The final training and validation accuracy of the model after completing training is 98.74% and 97.75%. The model is then evaluated on the test dataset, which resulted in a 98.03% test accuracy. The performance of the model is summarized in table 2.

TABLE 2. PERFORMANCE OF THE MODEL

Dataset	Total Images	Accuracy
Training dataset	10800	98.74%
Validation dataset	1200	97.75%
Test dataset	3000	98.03%

Figure 10 delineates the confusion matrix of the proposed CNN model in further performance analysis of the model. The confusion matrix is a 50x50 matrix where the predicted labels and true labels of the output classes (symbolic representations of the characters, as mentioned in figure 2) are mapped from the test data samples. As each output class had exactly 60 images for the testing purpose, the matrix was evaluated on this fact. From the confusion matrix, we can see that NA (ন) was mislabeled the most. From the 60 test samples belonging to NA, the model correctly labeled 56 images as NA while 4 images were labeled as NNA (ণ) and GA (গ), respectively. The main reason behind this error can be the structural similarities between these three characters. Finally, summarizing the confusion matrix, it holds a result of 0.98 in Precision and 0.98 in Recall and an overall **f1-score of 0.98**.

A summary of accuracy in comparison to other related works in Bangla HCR is depicted in table 3.

TABLE 3. COMPARING THE MODEL PERFORMANCE WITH OTHERS

Related Work	Related Work Accuracy
Purkaystha et al. [2]	91.23%
Chowdhury et al. [5]	95.25%
Rabby et al. [6]	92.61%
Abir et al. [7]	91.1%
Saha et al. [9]	96.64%
Proposed CNN Model	**98.03%**

Looking at table 3, we can say with certainty that our proposed model works better than other related work.

V. CONCLUSION

Considering our proposed model's accuracy compared to the accuracy obtained by the other models described in related work, we can conclude that the proposed model performs better than previously envisioned techniques. This approach could result in advancement in the pursuit of digitization of all Bangla texts and literature. Our model only recognizes characters at the moment. It does not have full-text recognition capabilities yet. However, its performance might suggest that further modifications to this model might enlighten how to create a complete Bangla Handwritten Text recognition system in the future. We also have plans to explore different Neural Network models and work with various datasets to create a more robust and effective automated system.

REFERENCES

[1] M. Y. W. Teow, "Understanding convolutional neural networks using a minimal model for handwritten digit recognition," *2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, Kota Kinabalu, 2017, pp. 167-172, doi: 10.1109/I2CACIS.2017.8239052.

[2] Purkaystha, Bishwajit, Tapos Datta, and Md Saiful Islam. "Bengali handwritten character recognition using deep convolutional neural network." *2017 20th International Conference of Computer and Information Technology (ICCIT)*. IEEE, 2017.

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

[4] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a backpropagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[5] Chowdhury, Rumman Rashid, et al. "Bangla handwritten character recognition using convolutional neural network with data augmentation." *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*. IEEE, 2019.

[6] Rabby, AKM Shahariar Azad, et al. "Borno: Bangla Handwritten Character Recognition Using a Multiclass Convolutional Neural Network." *Proceedings of the Future Technologies Conference*. Springer, Cham, 2020.

[7] Abir, B. M., et al. "Bangla handwritten character recognition with multilayer convolutional neural network." *Advances in Data and Information Sciences*. Springer, Singapore, 2019. 155-165.

[8] S. M. Azizul Hakim and Asaduzzaman, "Handwritten Bangla Numeral and Basic Character Recognition Using Deep Convolutional Neural Network," *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, Cox's Bazar, Bangladesh, 2019, pp. 1-6, doi: 10.1109/ECACE.2019.8679243.

[9] Saha, Chandrika, Rahat Hossain Faisal, and Md Mostafijur Rahman. "Bangla handwritten basic character recognition using deep convolutional neural network." *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*. IEEE, 2019.

[10] S. Basu, N. Das, R. Sarkar, M. Kundu, M. Nasipuri, and D. K. Basu, "Handwritten bangla alphabet recognition using an mlp based classifier," *arXiv preprint arXiv:1203.0882*, 2012.

[11] N. Das, S. Basu, R. Sarkar, M. Kundu, M. Nasipuri *et al.*, "An improved feature descriptor for recognition of handwritten bangla alphabet," *arXiv preprint arXiv:1501.05497*, 2015.

[12] LeCun, Yann, Koray Kavukcuoglu, and Clément Farabet. "Convolutional networks and applications in vision." *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on. IEEE*, 2010

[13] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

[14] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.

[15] Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research 15.1 (2014): 1929-1958*.

[16] Zeiler, Matthew D. "ADADELTA: an adaptive learning rate method." arXiv *preprint arXiv:1212.5701* (2012).

[17] Zhang, Zhilu, and Mert Sabuncu. "Generalized cross entropy loss for training deep neural networks with noisy labels." *Advances in neural information processing systems*. 2018.

[18] Zaheer, Manzil, et al. "Adaptive methods for nonconvex optimization." *Advances in neural information processing systems*. 2018.