Borno: Bangla Handwritten Character Recognition Using a Multiclass Convolutional Neural Network

A K M Shahariar Azad Rabby [](https://orcid.org/0000-0003-3994-3105)1, Md. Majedul Islam1, Nazmul Hasan, Jebun Nahar1 and Fuad Rahman2

1Apurba Technologies, Dhaka, Bangladesh.

2Apurba Technologies, Sunnyvale, CA, USA

{rabby,majed,nazmul,jebunnahar,fuad}@apurbatech.com

**Abstract.** Handwriting recognition is still not a solved problem. With the advancements in artificial intelligence and machine learning, the construction of Optical Character Recognition systems (OCRs) has become more effective. However, there is still no serious commercially available OCRs for many low-resource languages, such as Bangla. Bangla presents additional challenges, since oftentimes, the vowels and consonants in the middle of the words are abbreviated and replaced with notations called *diacritics*, and multiple letters can be combined to build shorthand representations, called *compound* characters. Furthermore, the compound characters can have diacritics as well, making the recognition task extremely complex. This means that a successful commercial OCR should not only model individual characters but also model these diacritics and combined characters, leading us to propose a grapheme-based holistic recognition approach. Borno is the first multiclass convolutional neural network-based deep learning model that can recognize Bangla handwritten characters with graphemes. The proposed model has been trained on a dataset of 1,069,132 images, with 50 basic characters, 10 numerals, 146 compound characters, 10 modifiers, and 6 consonant diacritics classes. The trained Borno model achieves a 92.61% average character recognition accuracy in the validation set.

**Keywords:** Bangla handwritten, Grapheme, Document Image Analysis, Pattern Recognition, Optical Character Recognition, Convolutional Neural Network Deep Learning.

1. Introduction

Bangla is the fifth most-spoken native language in the world, with approximately 228 million people speaking it as their first language [1][2][3][4]. Bangla is in a unique position as far as languages go. In essence, it is not only one of the most commonly spoken languages, but it also has a vibrant literature and tradition. Rabindranath Tagore, who wrote in Bangla, was the first non-European to win the Nobel Prize in Literature in 1913. Ironically, the Nobel Committee commented: "…his profoundly sensitive, fresh and beautiful verse, by which, with consummate skill, he has made his poetic thought, expressed in his own *English* words, a part of the literature of the West" (emphasis ours) [5]. Unfortunately, Bangla, although so rich and well-studied, has proven consistently to be poor in adapting to computerization and automation of the language in terms of automated language recognition, machine translation, and character recognition.

But all that has changed recently, with the Government of Bangladesh committing itself to “Digital Bangladesh,” a philosophy that vows to ensure people’s democracy and rights, transparency, accountability, establishing justice and ensuring delivery of government services to the doorsteps of the citizens through the use of technology. A major part of that strategy has been the establishment of a project titled “Enhancement of Bangla Language in ICT through Research & Development,” allocating funds to develop the essential resources for rapid Bangla computerization: building a corpora for written and spoken Bangla, developing interoperability, building OCR, speech-to-text and text-to-speech capabilities, sentiment analysis and so on. This has reinvigorated the research and commercial community in Bangladesh to focus on solving these problems.

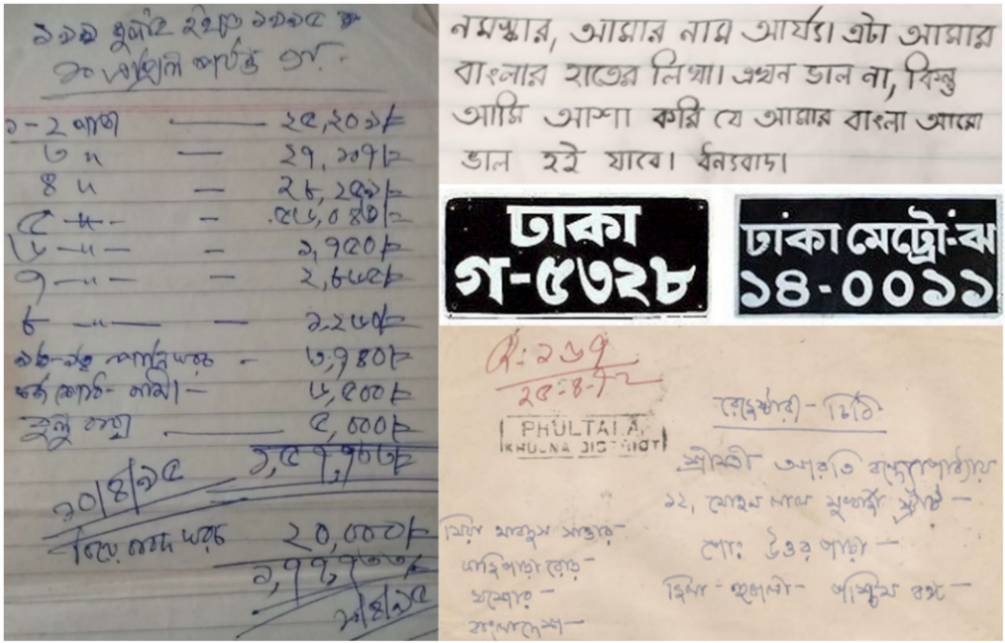
There are inherent difficulties in Bangla character recognition, such as diacritics (vowels and consonants changing, shorthand notations, and multiple positions for diacritics around the main consonant), compound characters (two or more characters combining to form a new character), and the interconnections of the neighboring characters, to name a few. Due to these complications, most research in Bangla has focused on isolated characters or digits alone. There exists no published research or commercial product that presents a holistic solution.

We proposed a holistic OCR solution to address the aforementioned difficulties for Bangla in this paper. We also have aggregated, normalized, and standardized character image data from five different public datasets with a total of 1,069,132 images. Fig 1 shows some examples of Bangla documents.

1. Literature Review

As indicated in the last Section, there has not been a lot of work on Bangla handwritten character recognition. What has been reported so far has mostly focused on isolated character recognition. This Section will present an overview of state of the art.

Rabby et al. [6] introduced a CNN model named EkushNet that was applied on their dataset called "Ekush" [7], which can recognize Bangla written by hand: 50 fundamental characters, 10 digits, 10 modifiers, and 50 compound characters. In this work, they achieved 97.73% accuracy on their dataset named Ekush, and for the CMATERdb, it was 95.01% for cross-validation. Saha et al. [8], in their paper "Bangla Handwritten Basic Character Recognition Using Deep Convolutional Neural Network," implemented a Deep CNN model named BBCNet-15, and they evaluated their model with CMATERdb dataset and reported a model accuracy of 96.40% on that dataset. In BBCNet-15, they used six convolution layers, six max-pooling layers, two



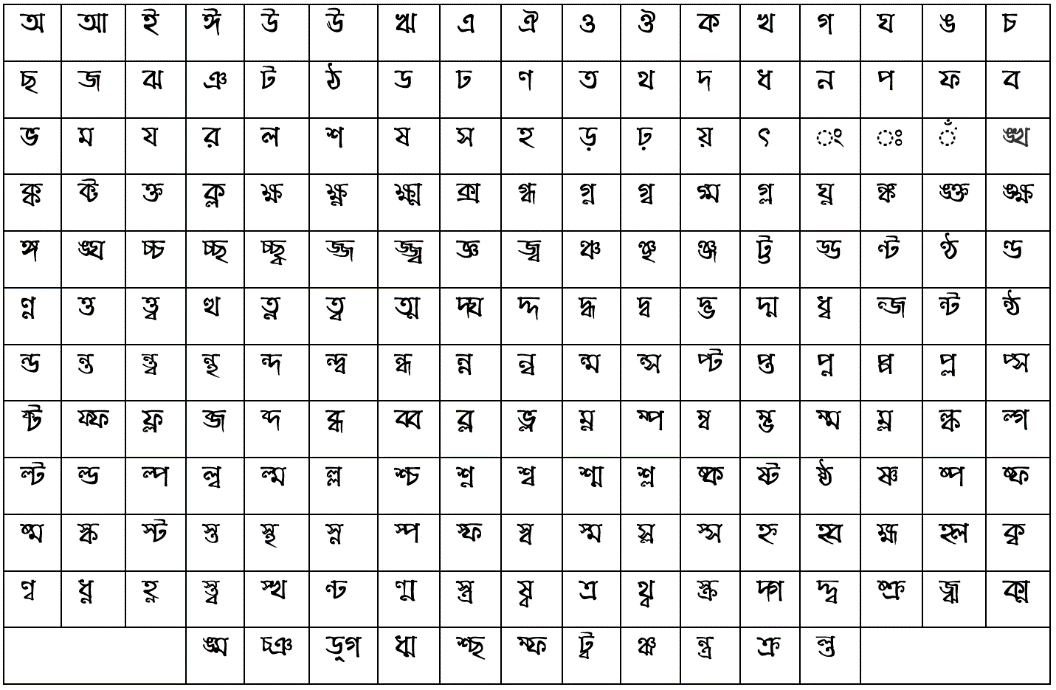
**Fig. 1.** Examples of Bangla handwritten documents

fully connected layers followed by the SoftMax output unit. Das et al. [9] proposed a feature extraction method and used an MLP based classifier to classify characters with 85.40% accuracy. In another work, Rahman et al. [10] proposed a Deep CNN based solution for classifying Bangla's fundamental characters. In their network, they use two convo layers, one fully connected layer, two pooling, and one output layer on a 20,000-image dataset. Sarkhel et al. [11] discussed a perspective of multi-objective-based locale choice issues where the most useful districts of character tests were utilized; later compound character databases used the conventional function engineering approach in conjunction with a classifier. In particular, an SVM classifier used a function descriptor composed of convex hull and quad tree-based features. Pal et al. [12] worked for Indian postal computerization utilizing the water store plot. They accomplished the precision of 94.13% and 93% for the transcribed Bangla and English numerals, individually, without determining the reaction time and the unwavering acknowledgment quality. Sharif et al. [13] proposed a crossover model consolidating profound CNN with HOG highlights to arrange the handwritten Bangla numerals. They achieved a recognition rate of 99.02% for the ISI dataset, and 99.17% precision for CAMTERDB dataset, which comprises of 6,000 pictures (600 pictures for every digit) of unconstrained transcribed Bangla digits, with 4,000 images used for training and 2,000 images were used for validation. Be that as it may, incorporating this half and half model with different high-quality highlights needs further research. Shopon et al. [14] utilized profound CNN with a solo pre-prepared autoencoder for handwritten Bangla digit acknowledgment. They increased the precision of 99.50% for CMATERDB; however, their approach is unfit to pre-train bigger datasets for better results. Alom et al. [15] accomplished 98.78% accuracy applying CNN with Gabor highlighting utilizing the CMATERDB dataset. They likewise chipped away at handwritten Bangla character recognition using diverse, profound learning models. They demonstrated that Dense Net gave the most noteworthy acknowledgment precision of 98.31% for letters in order, 99.13% for digits, and 98.18% for unique characters. They additionally gave a detailed examination of various cutting-edge profound models with multiple learning systems. They utilized the AlexNet and GoogLeNet models to characterize the ImageNet dataset. Rahman et al. [16] presented a synthetic example-based CNN strategy for better-managed learning and exactness, which gave a precision of 98.98% utilizing the ISI dataset. Shaha and Shaha [17] got an exactness of 97.21% subsequent to applying another profound CNN model over the Bangla handwritten disconnected character dataset. They put uncommon consideration on ordering essential and compound Bangla handwritten characters and numerals.

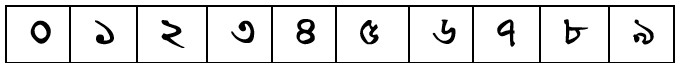
1. Datasets

Machine learning approaches heavily depend on data. If the size and variation of the dataset are inadequate, it is difficult to train a model effectively. We have used three different datasets: the training set, the testing set, and the validation set.

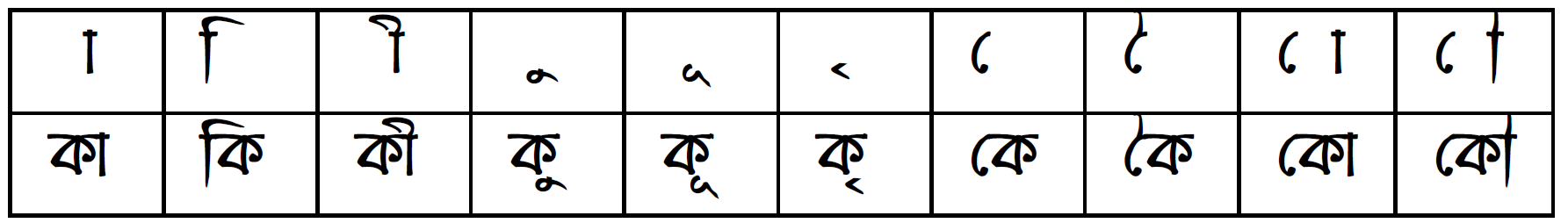
In our proposed system, we built a large dataset of 1,069,132 unique images. This dataset is assembled from five separate publicly available datasets: CMATERdb [18], BanglaLekha-Isolated [19], Ekush [7], MatriVasha [20], and the Bengali.AI [21] datasets. Fig 2 shows the character classes used in this paper.



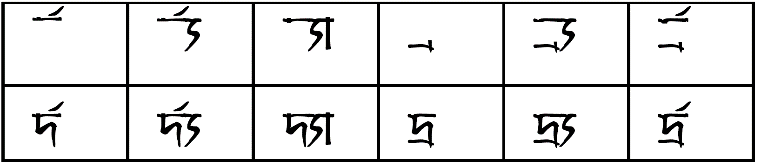
1. Bangla Basic and Compound Characters



(b) Bangla Numerals



(c) Bangla Modifiers



(d) Bangla Consonant Diacritics

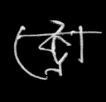
**Fig. 2.** Bangla Character List

* 1. Image Preparation and Preprocessing

We collected data from five different open source repositories. All this data came in various forms and output formats. CMATERDB, BanglaLekha-Isolated, Ekush, MatriVasha came as image and one output label, where the Bengali.ai came with parquet format with three outputs, as it is a grapheme dataset. The CMATERdb dataset has 63,351 images. BanglaLekha-Isolated has 166,105 images, Ekush has 367,018 images, MatriVasha has 306,464 images, and Bengali.ai has 200,840 data in 3 output labels. So, we needed to convert all these data in five different formats into a standard format. Fig 3 shows examples from each dataset before processing.



**(a)** CMATERDB **(b)** Ekush



**(c)** BanglaLekha-Isolated **(d)** Bengali.AI



**(e)** MatriVasha

**Fig. 3.** Examples from each dataset before processing

First, we converted all the parquet files into image format. All five datasets have different image formats and sizes. To make them in one form and size, we used three methods: Gray, Invert, and Resize. Usually, the information contained in the grayscale image is enough for classification but using a color image will increase the number of inputs three times, corresponding to R, G, B color values. This has two issues. Firstly, the so-called ‘curse of dimensionality.’ This essentially means that large dimensional vectors are almost equally spaced and hence are difficult to separate by a classifier. Secondly, a lot more parameters are now needed due to a more extensive network, forming more significant input vectors. Training a larger number of parameters is challenging and requires additional training data and a number of iterations. We decided to convert all images into one channel: grayscale. Previous work shows that inverted images (black as background and white as the character) work better than the typical image and reduce lots of computation [6][22]. Thus, we inverted all images. As all the images were in different sizes, we resized all images to 28x28 pixels and saved all the final images into JPG format. Fig 4 shows examples from each dataset after processing.



**(a)** CMATERDB **(b)** Ekush



**(c)** BanglaLekha-Isolated **(d)** Bengali.AI



**(e)** MatriVasha

**Fig. 4.** Examples from each dataset after processing

Normalization is a technique that is sometimes used as part of machine learning data preparation [23][24]. All images are normalized using a Min-Max normalizer. This normalizer maps the image value between 0 to 1, which reduces the effect of light and noise.

Data augmentation helps to artificially produce more data [25]. Data augmentation plays an essential role in handwriting recognition, as there can be an infinite number of variations for a character, seeing as each person has a unique style of writing. Also, one single person writes a character in different ways, both intentionally and unintentionally. One never draws a character quite the same twice. As a result, we chose to use four augmentation techniques.

* Rotation: Images were rotated 8 degrees randomly.
* Zoom: Images were 15% zoomed in and zoomed out.
* Width shift: Randomly shift images 15% horizontally.
* Height shift: Randomly shift images 15% vertically.

Fig 5 shows some data augmentation examples.



1. Rotation image (b) Zoom in/out

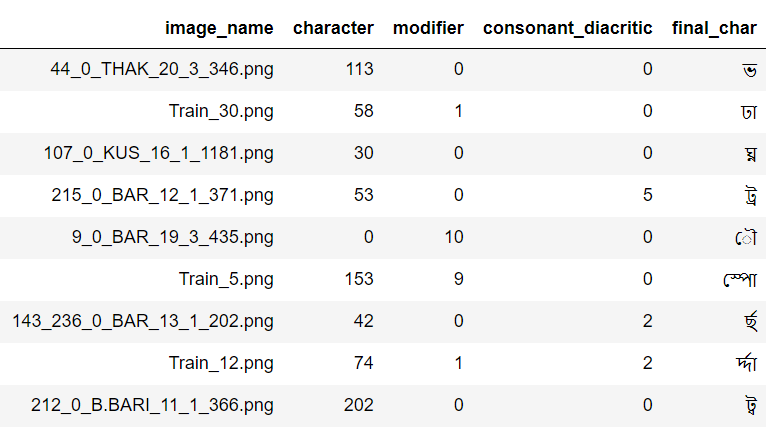


**(**c**)** Width shift (d) Height shift

**Fig. 5.** Example of data augmentation

* 1. Output level preparation

CMATERDB, BanglaLekha-Isolated, Ekush, and MatriVasha are in the same format. Every class image is placed in a folder, and the name of the folder is the class name. Bengali.ai is a grapheme dataset—the image name of this dataset is a map with a class in a separate CSV file. It has 5 outputs: ‘image\_id,’ ‘grapheme\_root,’ ‘vowel\_diacritic,’ ‘consonant\_diacritic,’ and ‘grapheme.’ The following are the meanings of each of the outputs: ‘image\_id’ contains the name of the image; ‘grapheme\_root’ is the label for basic and compound characters; ‘vowel\_diacritic’ indicates the vowel modifier; ‘consonant\_diacritic’ represents the consonant diacritic, and the last column, ‘grapheme,’ represents the final output characters. We converted all the datasets into a common grapheme format. Some classes in Ekush and MatriVasha are in grapheme, and we prepare the multilevel output for those image classes. All images are mapped in a CSV file, with an image name which has five columns: ‘image\_name’ contains the name of the image; ‘character’ is the class number of the basic and compound characters; ‘modifier’ is the label of Bangla modifier; ‘consonant\_diacritic’ is the label for consonant diacritic; ‘final\_char’ is the output for the last Bangla char; and finally, ‘class’ is the label output for that character.  Fig 6 shows an example of the output data format.



**Fig. 6.** Output label data

1. Proposed Method
   1. Model Preparation and Tuning

Convolution is the basic building block of Convolutional Neural Network, or CNN [22], which is a mathematical combination of two functions that merges two sets of information to produce a third function. The convolution is performed on the input image by using a multiple filter and kernel to create the feature map. CNN can minimize the number of parameters in order to solve complex image recognition tasks.

The pooling layer mainly uses down-sampling over the input dimension for reducing the complexity for further layers, which is also called resolution reduction in image processing. It minimizes the training time and controls the over-fitting. The most popular pooling method is Max-pooling, which divides the image into sub-region rectangles, picking the maximum value among other values inside of that sub-region.

The proposed model has a multiclass Convolutional Neural Network for the classification of handwritten Bangla characters, which, in turn, has three outputs. The convolution part of this model used the “Convolutional Layer.” “Max-Pool Layer” is later connected with the “Dense Layer.” We used two regularization techniques like “Batch-Normalization” (1) [26] and “Dropout” [27] with complicated architecture to reduce the overfitting. In batch normalization, we use a parameter momentum. For faster training, gradient momentum gives better results as it omits the noise in the gradient update term.

(1)

Model architecture has the same four blocks. Every block has four convolutional layers with kernel size 3 for all blocks, of 32, 64, 128, and 256, respectively. In every block’s 5th layer Batch Normalization layer, the momentum is set to 15%, which is then connected with the max-pooling layer and uses as input in convolutional layer with a kernel size of 5, again with filter sizes of 32, 64, 128, and 256, respectively, using ReLU activation with the same padding followed by 30% dropouts layer. The output is then flattened and transformed into an array to feed into a dense layer that has 1,024 neurons and is regularized with a 30% dropout and passing through a fully connected dense layer of 512 nodes with ReLU (2) activation.

(2)

After that, three output layers for models that are fully connected dense layers are initiated accordingly with the 207, 11, and 7 nodes with SoftMax (3) activation.

(3)

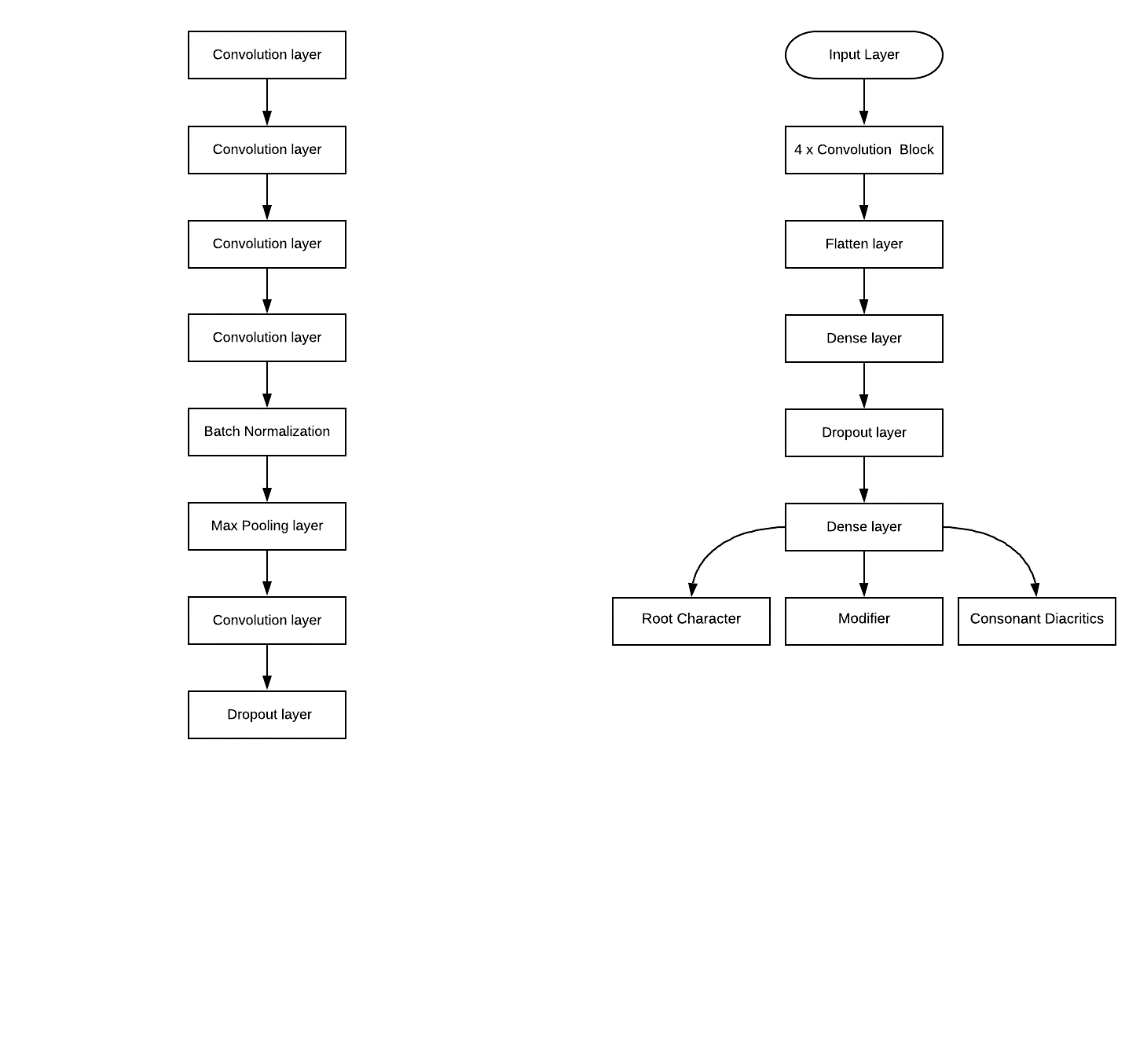
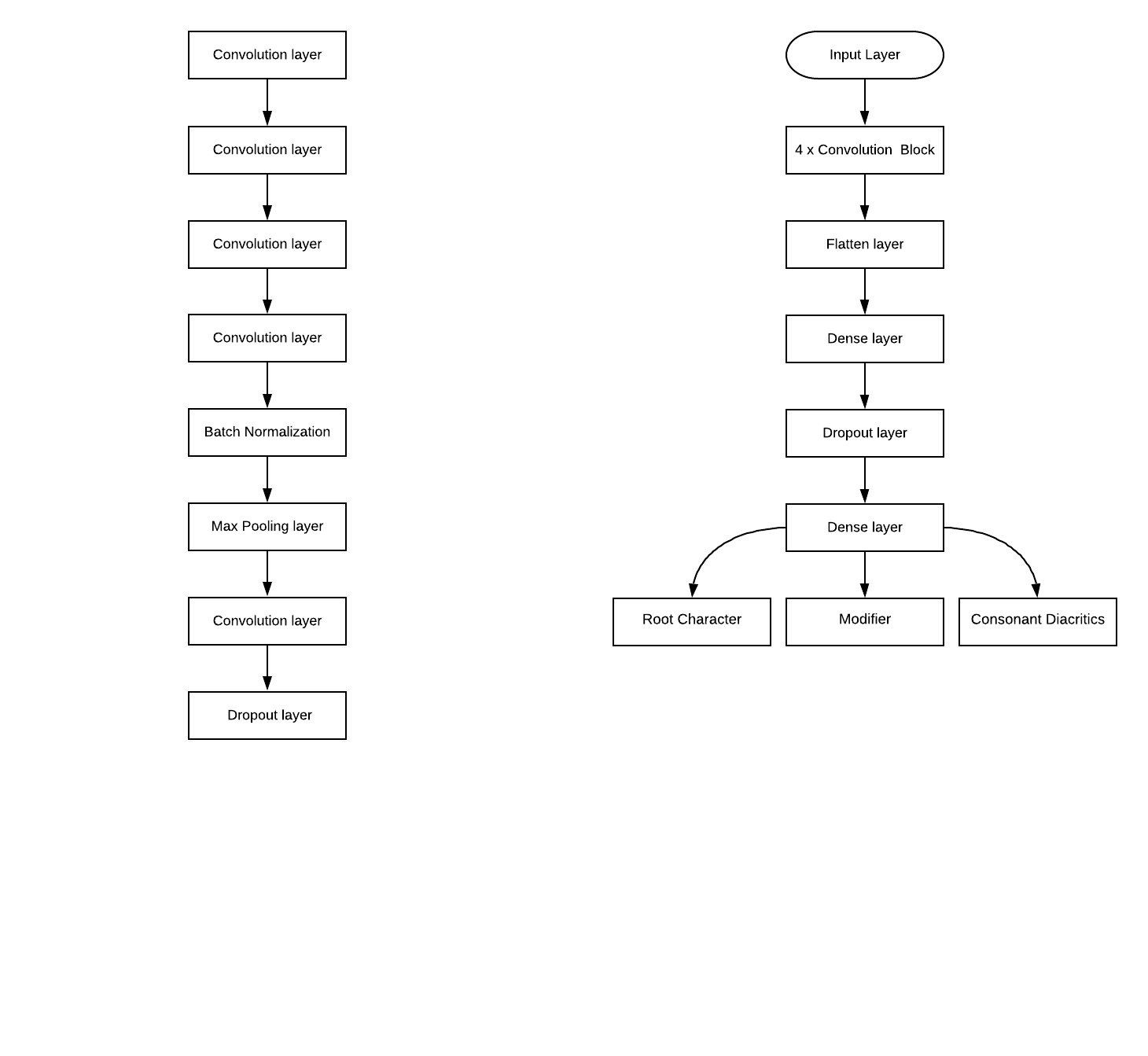
To optimize the error of the convolutional algorithms, optimization algorithms are widely used. The proposed Borno uses a learning rate of 0.001 for Adam (4) optimizer. Adam is widely used for better results in computer vision science.

(4)

To measure the error of the Adam, a categorical cross-entropy (5) function is used. Some previous works [28][29] shows that cross-entropy can perform better than other error function like Classification Error or Means Squared Error.

(5)

Hyperparameter tuning is one of many complicated parts of training a CNN, and the learning rate is one of them. We may get better accuracy, but the model will take much more time to reach the global optima if we set the learning rate lower. For a higher learning rate, the model accuracy will not converge, and may even diverge in certain situations. Therefore, it is essential to select the best value for the learning rate. We are using an adaptive learning rate to conquer the obstacle. The higher learning rate of 0.001 is set for faster computation, which is reduced atomically by measuring validation accuracy. Fig 7a shows the convolutional Layer, and Fig 7b shows the full model architecture.

**(**a) Convolution Block (b) Borno Architecture

**Fig. 7.** Model Architecture

* 1. Output Processing

The model results in three outputs: one for the character class, one for the modifier class, and the final one for consonant diacritic. We need to combine these three classes in order to identify the output characters. In Bangla, basic and compound characters can be represented independently, but modifiers and consonant diacritics need basic or compound characters. After getting the probability of all three labels, we chose the best one.

In Bangla, there are rules to put the characters, modifiers, and consonant diacritics together. As a consequence, some combinations are invalid. For example:

* The modifier cannot work without a basic or compound character.
* A consonant diacritic cannot be used independently without a basic character.
* Basic characters like **ঁ** cannot work independently. It needs another character to attach to.
* **ঁ** can sit with basic characters and modifiers together.
* Modifiers and consonant diacritics cannot sit together with basic characters like **ৎ, ং, ঃ, ঁ**.

In general, rules for basic and compound characters need to be added with consonant diacritic and then added with modifiers. But there are some exceptions:

* The consonant diacritic র্ should be added before the basic and compound characters, and then with a modifier.
* The basic character **ঁ** should be added after adding the basic and compound characters with a modifier.
* The consonant diacritic র্্য has two parts: র্ and ্য. The first part needs to be added before the basic and compound characters.

We took all the unique characters, modifiers, and consonant diacritics from our dataset that we built. There are some characters in Bangla which can be grouped with modifiers or consonant diacritics, such as Bangla numerals and characters like ৎংঃ, so we remove all those characters. Then, we make all possible combinations of all characters, modifiers, and consonant diacritics. We found 13,908 possible combinations of characters. However, not all of these combinations are valid.

To find all the possible Bangla characters, we built a corpus of Bangla words from Wikipedia containing 3 million words. Then we searched all 13,908 unique characters against a 3M word corpus and found 3,236 unique character combinations out of 13,908 characters combinations.

We implemented all these language rules in our system. After detecting the character from CNN, we check the output against the rules and generate the final output character. If the generated character is a grapheme, then we search the generated character in the 13,908 possible grapheme combinations. If it matches, we know that we generated a correct grapheme combination.

1. Performance

The proposed Borno model gives us satisfactory performance on our train and validation set. Below is an extended discussion of the relevant performance metrics.

* 1. Train Test Split

Our combined dataset had 1,069,132 images. We split the dataset into two parts, one for training and another one for validation. We reserved 10% of the data for validation and used 90% for training. After the split, we have 9,62,218 images in the training set, and 1,06,914 images in the validation set.

* 1. Model Performance

We fit all the data into our model. After 30 epochs, our proposed model achieves satisfactory accuracy and loss for recognition of the basic characters, compound characters, modifier, and consonant diacritics. We saved the best model from these 30 epochs and retrained for 50 more epochs and improved our accuracy and loss. The result of both training and validation is shown in Table 1.

**Table 1.** Accuracy and Loss of the proposed model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 30 Epoch Train | 30 Epoch Val | 50 Epoch Train | 50 Epoch Val |
| Root Char Acc | 0.9077 | 0.8666 | 0.9274 | 0.8609 |
| Modifier Acc | 0.9898 | 0.9545 | 0.9927 | 0.9556 |
| Diacritic Acc | 0.9879 | 0.9571 | 0.9909 | 0.9399 |
| Average Acc | 0.9618 | 0.9261 | 0.9703 | 0.9188 |
| Root Char Loss | 0.3635 | 0.6193 | 0.2762 | 0.6575 |
| Modifier Loss | 0.0396 | 0.2735 | 0.0273 | 0.2880 |
| Diacritic Loss | 0.0418 | 0.1858 | 0.0305 | 0.2672 |
| Total Loss | 0.4449 | 1.0780 | 0.3340 | 1.2135 |

Visual representation of the performance of our proposed model is shown in Fig 8.

(a) Loss for 30 epochs (b) Accuracy for 30 epochs

(c) Loss for 50 epochs (d) Accuracy for 50 epochs

**Fig. 8.** Loss and accuracy graph for the proposed model

* 1. Result Comparison

There is no existing published research for Bangla characters that includes classifiers for all possible classes, including all the possible graphemes. So, a direct comparison of our work with the published work is impractical. Table 2 lists the leaders in reported works in perspective. As is clearly seen, most of these approaches were applied to a subset of the problem and focused on specific parts of the overall character set. We have, for the first time, attempted to model and implement a holistic classifier that handles all the possible characters in Bangla.

**Table 2.** Comparison with previous work

|  |  |  |
| --- | --- | --- |
| Work | Dataset Discerption | Accuracy |
| Recognition of Handwritten Bangla Characters Using Gabor Filter and Artificial Neural Network [30] | 1,045 Vowel in 11 classes collected from 95 people. | 79.40% |
| Recognition of Bangla handwritten basic characters and digits using convex hull-based feature set [31] | 10,000 Alphabet in 50 class collected from 200 people.  12,000 digits in 10 classes. | 76.86% |
| Handwritten Bangla Character Recognition Using Neural Network [32] | 6,000-digit images in 10 class from CMATERdb dataset  15,000 Alphabet in 50 classes from CMATERdb dataset | 84.00% |
| Bangla Handwritten Character Recognition using Convolutional Neural Network [16] | 20,000 Alphabet in 50 classes. | 85.36% |
| BornoNet: Bangla Handwritten Characters Recognition Using Convolutional Neural Network. [33] | * 15,000 images from CMATERdb dataset * 37,858 images from ISI Dataset in 50 alphabet class * 3.98,950 images from BanglaLekha-Isolated in 50 alphabet classes. | Isolated: 95.71%  CMATERdb: 98%  ISI: 96.81% |
| HMM-Based Online Handwritten Bangla Character Recognition using Dirichlet Distributions [34] | Collected 38,567 images in 50-character class. | 91.85% |
| Bangla Hand-Written Character Recognition Using Support Vector Machine [35] | Collected 7,500 Image in 50-character class | 93.43% |
| Bengali handwritten character recognition using Modified syntactic method [36] | 50-character class | 95.00% |
| EkushNet: Using Convolutional Neural Network for Bangla Handwritten Recognition [6] | 368,776 images in 50 alphabets, 10-digit, 52 compound character classes | Ekush: 97.73%  CMATERdb:95.01% |
| Proposed Borno Model | 1,069,132 images in 207 root characters 10 modifiers and 6 consonant diacritic classes. | Root: 86.09%  Modifier: 95.56%  Diacritic: 93.99 %  Average: 91.88% |

1. Future Work

There are several areas that need further work. A primary source of errors in our model is the fact that a lot of characters and graphemes in Bangla are very similar, as seen in Fig. 9. Unfortunately, our current model often failed to differentiate between these difficult classes. Our next step is to build a confusion matrix to identify clusters of similar classes using this model as the first step of a multiple step solution. The idea is to build additional models that are specifically trained to solve these clusters of confusing classes. If the first model (our current model) identifies a class as belonging to any of these suspect classes, they will be reclassified using a specialized classifier that is trained specifically on those classes.



**Fig. 9.** Visually similar Bangla character combinations

The only source for our grapheme samples is from Bengali.ai, and it does not even cover all the Bangla compound characters. We plan to correct this shortcoming by building a corpus that will have ample samples for these very important classes. This project is already underway.

We have not explored using Bangla language models in improving our solution. In a parallel work, we are also looking at building a stable Bangla language model. We are determined to explore how we can use that in designing better OCRs.

1. Conclusion

In this paper, we proposed, implemented, and discussed the first-ever fully comprehensive and holistic Bangla handwritten character recognition solution. We have presented detailed descriptions of our proposed model and have demonstrated that it has produced very robust recognition rates. However, although this has produced a stable solution, this is only the beginning and not the end. As discussed in the previous section, we have already identified and are getting ready to explore multiple possible avenues to enhance the performance of our solution.

Acknowledgement

The authors would like to acknowledge the encouragement and funding from the “Enhancement of Bangla Language in ICT through Research & Development (EBLICT)” project, under the Ministry of ICT, the Government of Bangladesh.

**References**

1. "The World Factbook." www.cia.gov. Central Intelligence Agency. Archived from the original on 13 February 2008. Retrieved 21 February 2018.
2. "Summary by language size." Ethnologue. 3 October 2018. Archived from the original on 11 September 2013. Retrieved 21 February 2019.
3. Bengali | Ethnologue. A language profile for Bengali. from population to dialects and usage. Retrieved April 28, 2020, from https://www.ethnologue.com/language/ben
4. Hays, Jeffrey. "BENGALIS – Facts and Details". factsanddetails.com. Archived from the original on 30 July 2017. Retrieved 4 July 2018.
5. The Nobel Prize in Literature 1913. NobelPrize.org. Nobel Media AB 2020. Sat. 25 Apr 2020. <https://www.nobelprize.org/prizes/literature/1913/summary>.
6. Rabby, A. S. A., Haque, S., Abujar, S., & Hossain, S. A. (2018). EkushNet: Using Convolutional Neural Network for Bangla Handwritten Recognition. Procedia computer science, 143, 603-610.
7. Rabby, A. S. A., Haque, S., Islam, M. S., Abujar, S., & Hossain, S. A. (2018, December). Ekush: A Multipurpose and Multitype Comprehensive Database for Online Off-Line Bangla Handwritten Characters. In International Conference on Recent Trends in Image Processing and Pattern Recognition (pp. 149-158). Springer, Singapore.
8. Saha, C., Faisal, R. H., & Rahman, M. M. (2019, May). Bangla handwritten basic character recognition using deep convolutional neural network. In 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR) (pp. 190-195). IEEE.
9. N. Das 1, B. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, “Handwritten Bangla Basic and Compound character recognition using MLP and SVM classifier,” Journal of Computing, vol. 2, Feb. 2010.
10. Md. M. Rahman, M. A. H. Akhand, S. Islam, P. C. Shill, “Bangla Handwritten Character Recognition using Convolutional Neural Network,” I. J. Image, Graphics and Signal Processing, vol. 8, pp. 42-49, July 15, 2015.
11. Sarkhel, R., Das, N., Saha, A. K., & Nasipuri, M. (2016). A multi-objective approach towards cost effective isolated handwritten Bangla character and digit recognition. Pattern Recognition, 58, 172-189.
12. U. Pal, A. Belad, Ch. Choisy, “Touching numeral segmentation using water reservoir concept,” Pattern Recognition Lett. vol. 24, pp. 261-272, 2003.
13. Sharif S.M.A., Mohammed N., Momen S., Mansoor N. (2018) Classification of Bangla Compound Characters Using a HOG-CNN Hybrid Model. In: Mandal J., Saha G., Kandar D., Maji A. (eds) Proceedings of the International Conference on Computing and Communication Systems. Lecture Notes in Networks and Systems, vol 24. Springer, Singapore
14. Shopon, M., Mohammed, N., & Abedin, M. A. (2016, December). Bangla handwritten digit recognition using autoencoder and deep convolutional neural network. In 2016 International Workshop on Computational Intelligence (IWCI) (pp. 64-68). IEEE.
15. Alom, M. Z., Sidike, P., Taha, T. M., & Asari, V. K. (2017). Handwritten bangla digit recognition using deep learning. arXiv preprint arXiv:1705.02680.
16. Rahman, M. M., Akhand, M. A. H., Islam, S., Shill, P. C., & Rahman, M. H. (2015). Bangla handwritten character recognition using convolutional neural network. International Journal of Image, Graphics and Signal Processing (IJIGSP), 7(8), 42-49.
17. S. Saha and N. Saha (2018). A Lightning fast approach to classify Bangla Handwritten Characters and Numerals using newly structured Deep Neural Network. Procedia Computer Science, 132, 1760-1770.
18. Sarkar, R., Das, N., Basu, S., Kundu, M., Nasipuri, M., & Basu, D. K. (2012). CMATERdb1: a database of unconstrained handwritten Bangla and Bangla–English mixed script document image. International Journal on Document Analysis and Recognition (IJDAR), 15(1), 71-83.
19. Biswas, M., Islam, R., Shom, G. K., Shopon, M., Mohammed, N., Momen, S., & Abedin, A. (2017). Banglalekha-isolated: A multi-purpose comprehensive dataset of handwritten bangla isolated characters. Data in brief, 12, 103-107.
20. Ferdous, J., Karmaker, S., Rabby, A. S. A., and Hossain, S. A. (2021). MatriVasha: A multipurpose comprehensive database for Bangla handwritten compound characters. In: Hassanien A. E., Bhattacharyya S., Chakrabati S., Bhattacharya A., Dutta S. (eds) Emerging Technologies in Data Mining and Information Security. Advances in Intelligent Systems and Computing. Springer, Singapore.
21. Bengali.AI Handwritten Grapheme Classification | Kaggle. Kaggle.com. (2020). Retrieved 28 April 2020, from <https://www.kaggle.com/c/bengaliai-cv19/data>.
22. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
23. Images "Normalization". Deep Learning Course Forums. (2017). Retrieved 28 April 2020, from https://forums.fast.ai/t/images-normalization/4058/11.
24. Jaitley, U. (2018). Why Data Normalization is necessary for Machine Learning models. Medium. Retrieved 28 April 2020, from <https://link.medium.com/EXbwW8iP25>.
25. Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0
26. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.
27. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.
28. Golik, P., Doetsch, P., & Ney, H. (2013, August). Cross-entropy vs. squared error training: a theoretical and experimental comparison. In Interspeech (Vol. 13, pp. 1756-1760).
29. Mannor, S., Peleg, D., & Rubinstein, R. (2005, August). The cross-entropy method for classification. In Proceedings of the 22nd international conference on Machine learning (pp. 561-568).
30. Begum, H., & Islam, M. M. Recognition of Handwritten Bangla Characters using Gabor Filter and Artificial Neural Network. International Journal of Computer Technology & Applications, 8(5), 618-621.
31. Das, N., Pramanik, S., Basu, S., Saha, P. K., Sarkar, R., Kundu, M., & Nasipuri, M. (2014). Recognition of handwritten Bangla basic characters and digits using convex hull-based feature set. arXiv preprint arXiv:1410.0478.
32. Badsha, M. A., Ali, M. A., Deb, D. K., & Bhuiyan, M. N. (2012). Handwritten Bangla Character Recognition Using Neural Network. International Journal of Advanced Research in Computer Science and Software Engineering, 2(11), 307-312.
33. Rabby, A. S. A., Haque, S., Islam, S., Abujar, S., & Hossain, S. A. (2018). Bornonet: Bangla handwritten characters recognition using convolutional neural network. Procedia computer science, 143, 528-535.
34. Biswas, C., Bhattacharya, U., & Parui, S. K. (2012, September). HMM based online handwritten Bangla character recognition using Dirichlet distributions. In 2012 International Conference on Frontiers in Handwriting Recognition (pp. 600-605). IEEE.
35. Azim, R., Rahman, W., & Karim, M. F. (2016). Bangla Hand-Written Character Recognition Using Support Vector Machine. International Journal of Engineering Works, 3(6), 36-46.
36. Islam, M. B., Azadi, M. M. B., Rahman, M., & Hashem, M. M. A. (2005, February). Bengali handwritten character recognition using modified syntactic method. In Proceedings of 2nd National Conference on Computer Processing of Bangla (NCCPB-2005). Independent University, Bangladesh.