# Handwritten Bangla Numeral and Basic Character Recognition Using Deep Convolutional Neural Network

S M Azizul Hakim
*Dept. of Computer Science and Engineering*
*Chittagong University of Engineering and Technology*
Chittagong, Bangladesh
azizulraihan19@gmail.com

Asaduzzaman
*Dept. of Computer Science and Engineering*
*Chittagong University of Engineering and Technology*
Chittagong, Bangladesh
Asad@cuet.ac.bd

***Abstract*—In this paper, the problem of recognizing handwritten Bangla characters is addressed. Handwritten character recognition is one of the most practiced tasks in computer vision. Over the past few years Convolutional Neural Network has produced the best results in case of English handwritten character recognition. Although Bangla the official language of Bangladesh and several Indian states with over 200 million native speakers Bangla handwritten character recognition is quite far behind.We present a 9 layer sequential Convolutional Neural Network model to recognize 60 (10 numerals+ 50 basic characters) Bangla handwritten characters. BanglaLekha-Isolated dataset is used as train-validation set. A new dataset of 6000 images is created for cross validation. Our proposed model trained to recognize 60 characters achieves state-of-the-art 99.44% accuracy on BanglaLekha-Isolated dataset and 95.16% accuracy on prepared test set. Experiments on recognizing Bangla numerals separately also show state-of-the-art performance.**

***Index Terms*—handwritten character recognition, image processing, deep learning, convolutional neural networks.**

## I. Introduction

Recently Convolutional Neural Network has shown great success in the field of pattern recognition and image classification. CNN is proven to be superior than other pattern recognition techniques. Handwritten character recognition is one of the most practiced tasks in pattern recognition. Several applications including postal system automation, mail sorting, automatic bank cheque processing, number plate recognition require handwriting character recognition systems. Different machine learning methods have been applied successfully for recognizing English handwritten characters. Although Bangla the official language of Bangladesh and several Indian states [1] with over 200 million native speakers Bangla handwritten character recognition is quite far behind. Two crucial reasons behind that are lack of large dataset and the complexity of Bangla characters. Recently a dataset named BanglaLekha-Isolated [2] is released which is the largest dataset on Bangla handwritten characters. The complex nature of Bangla characters makes it challenging for recognition task. Similar patterns occur in many characters which sometimes makes it difficult to recognize Bangla characters even for human eyes. Bangla basic characters are already complex in nature. When we combine numerals with basic characters, the complexity of the recognition task increases even further. There are many reoccurring patterns in Bangla characters. Also, the existence of 'matra' (horizontal line over the characters) creates ambiguity between many numerals and basic characters like. Another ambiguity occurs between characters with or without 'fota' (a dot below the characters).

CNN based techniques show state-of-the-art accuracy on ImageNet Large Scale Visual Recognition Challenge (ILSVRC) which is a benchmark in the object category classification and detection on hundreds of object categories and millions of images [3]. Starting from AlexNet [4] through the evolution of deep learning architecture, ResNet [5] have achieved better than human accuracy [3] on the ImageNet classification challenge. We took interest on VGGNet [6], the winner of ILSVRC 2014. They indicated that using 3X3 convolution kernel, deeper networks can perform better than the shallower ones with a relatively large kernel size.

Inspired from VGG-16, we propose a sequential CNN model consists of 6 convolutional layers, 2 full connection layers and an output layer. Additionally, a dropout layer of 0.2 is add after every convolution and full connected layers. For train-validation BanglaLekha-Isolated dataset is used at 0.80 : 0.20 split. Different data augmentation techniques are used on train set to increase the variety of data [7]. It helps to train a more generalized model to achieve better result on new data. We develop a test set of 6000 images for cross validation. After collection images are scanned and have gone through several preprocessing steps before emerging as the usable test set. After choosing all the proper hyper-parameters proposed model is trained twice. First to recognize Bangla numerals and basic characters all together. Additionally, we test our model for recognizing Bangla numerals separately.

## II. Related Works

Several attempts were taken in the field of Bangla handwritten character recognition. In this section we intend to mention some of them. K. Ray et al. [8] designed a nearest-neighbor classifier for recognizing hand-printed Bengali alphabetic characters. It is one of the earliest works in Bangla character recognition. They extracted a set of simple features using a string connectivity criterion. A. Dutta et al. [9] suggested a mechanism for recognition of Bangla alpha-numeric characters drawn from various fonts of print and different handwriting styles. Chowdhury, Ahmed Asif, et al. [10] employed a complete Optical Character Recognition system for the printed Bangla text. They described efficient ways involving line and word detection, zoning, character separations and character recognition. Wen, Y et al. [11] proposed two approaches for recognizing handwritten Bangla numerals. One is the image reconstruction recognition approach, and the other is the direction feature extraction approach combined with PCA and SVM. Das, Nibaran, et al. [12] identified a new variation of feature set in Bangla alphabet recognition. They used a feature set including visual modified shadow features, octant and centroid features, distance based features, quad tree based longest run features. T Hassan et al. [13] classified Bangla digits from their local binary pattern histogram. They compared the performance of three different variations of LBP – the basic LBP, the uniform LBP and the simplified LBP. Das, Nibaran, et al. [14] proposed a two pass soft-computing approach for Bangla HCR. The first pass classifies an unknown sample pattern in a group of pattern classes. In the second pass, it makes a finer classification of the sample by determining its membership to a class within the group. Sharif, S. M. A., et al. [15] combined hand crafted feature extraction based approaches with the automatically learnt features of Convolutional Neural networks. Alif, M. A. R. et al. [16] proposed a modified ResNet architecture and applied on BanglaLekha-Isolated and CMATERdb dataset

## III. DATA PREPROCESSING

For train-test we use BanglaLekha-Isolated. Another small dataset of 6000 samples (100 per character) is collected for cross validation. Table-I shows statistical information about the datasets.

TABLE I
A BRIEF NUMERIC DESCRIPTION OF DATASETS BEING USED

| | BanglaLekha-Isolated | Prepared dataset |
|---|---|---|
| Numerals | 17,645 | 1000 |
| Basic characters | 83,458 | 5000 |
| Total | 101,103 | 6000 |

### *A. Train-validation set*

The dataset is checked thoroughly. Some of the images are removed due to ambiguous handwriting. Images without proper ‘matra’ and ‘fota’ are also removed. After removing all the ambiguities and unclear images we have a total of 17,645 Numeral characters and 83,458 basic characters. A label file is created mapping all the samples to numeric values from 0 to 59 according to the character classes. Labeling them with character or string is not possible. The Only way is to assign numerical values to each class. We assigned 0-9 to the numerals and 10-59 to the basic characters.

### *B. Test set*

Validation dataset is collected from 25 students of Chittagong University of Engineering and Technology. They are informed about the ambiguities we faced and are instructed to be careful about that.

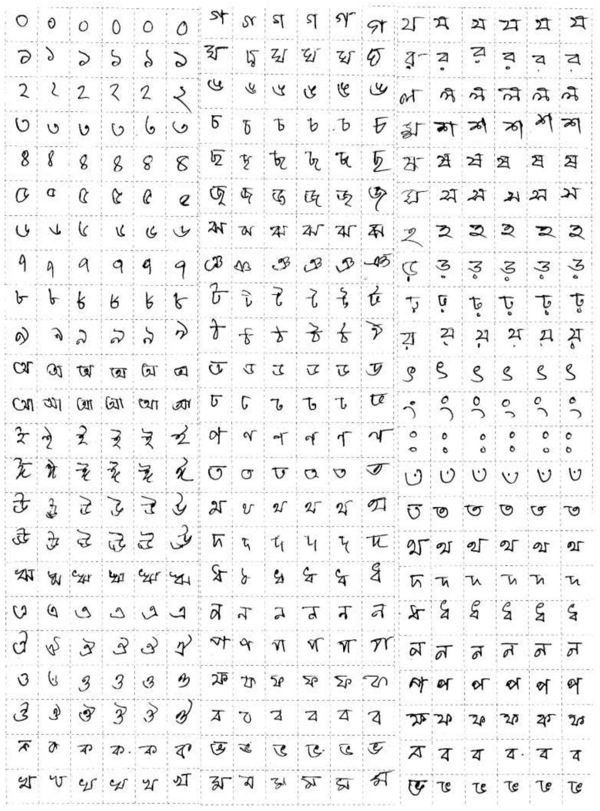
Fig. 1. A portion of data collection forms of for preparing the cross- validation dataset.

Data is collected in standard A4 papers with square grids drawn in thin dotted lines. Collected hard copies are then scanned at 600dpi. After acquiring the digital copy of the dataset, a series of preprocessing steps are carried out.

- All the images were loaded in grayscale mode and converted into binary image using the Otsu method. Converting the images into binary reduced a lot of noise and made the image more clear.
- As background is the largest part of our images, keeping the background pixel value to zero reduces a lot of computation by the CNN model. So, we inverted the images to get black (pixel value 0) background and white foreground (pixel value 1). Therefore, this inversion increases the efficiency of our model.
- Gaussian filter reduces noise and makes the edges blur. It is very fast noise reduction method.
- Images were cut by the grid line to get the individual character image. Segmented character images were then divided in folders by image classes. Each folder contained images of a single class creating 60 folders.

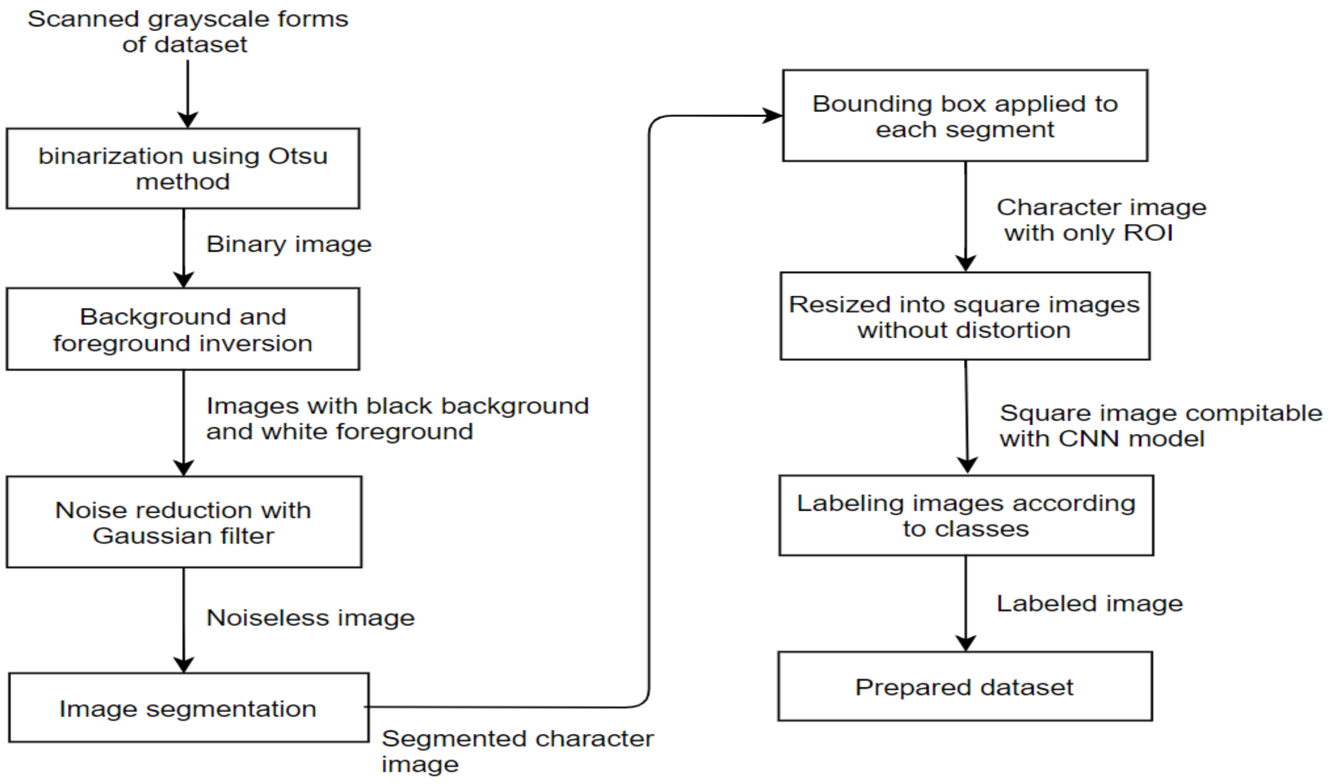
Scanned grayscale forms of dataset
binarization using Otsu method
Binary image
Background and foreground inversion
Images with black background and white foreground
Noise reduction with Gaussian filter
Noiseless image
Image segmentation
Segmented character image
Bounding box applied to each segment
Character image with only ROI
Resized into square images without distortion
Square image compitable with CNN model
Labeling images according to classes
Labeled image
Prepared dataset

Fig. 2. Preprocessing steps for the preperation of cross-validation dataset.

- Bounding box method is applied to extract only the character area of each image. This helped us to reduce the size of the dataset also kept the focus on only the character.
- CNN accepts square images only. If we feed an image with uneven edges then the model collapses the image to make it square. So, prior to that we resized all the images into square images. Required padding is added to the smaller edge area to avoid distortion of the image.
- All the processed images were then labeled according to the image class. Labeling them by character or string is not possible. The Only way is to assign numerical values to each class. We assigned 0-9 to the numerals and 10-59 to the basic characters.

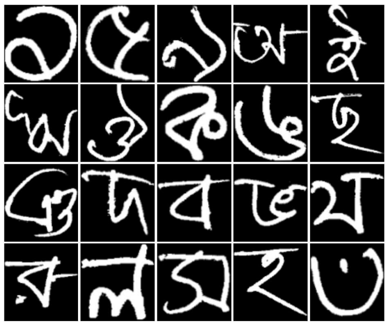

Fig. 3. A small portion of the prepared cross-validation dataset.

Figure 3 shows some of the samples of our prepared dataset after preprocessing is done. Although small in size, our dataset has enough variation to be used for cross validation.

### C. Data augmentation

Although we have the largest dataset on the Bangla characters, the classification task of 60 classes needs more data. Our proposed CNN model is very deep with 8,987,964 learnable parameters. So, the more data we can feed into the model the more features to learn from the data which results in better classification result. Data augmentation [17] is a very good way to increase the size of the dataset. Data augmentation is done by applying one or more augmentation techniques to the dataset. Augmented data is added to the original dataset hereby increasing the size of the dataset. Also, the variation in dataset due to different augmentation techniques is useful to prevent overfitting. Augmentation techniques we used are: Rotation: 10, Width shift: 0.1, Height shift: 0.1 and Shear: 0.1 We applied these data augmentation techniques only to the train set.

## IV. PROPOSED CNN MODEL

VGG-16 is a very deep model with 134,000,000 parameters. It is used for classifying ImageNet [3] dataset which consists of 1000 classes. Our problem of recognizing 60 characters is rather simple compared to that. So, we focus on training a simpler version of VGG-16 as efficiently as possible. Our proposed 9 layer sequential model consists of 6 convolutional

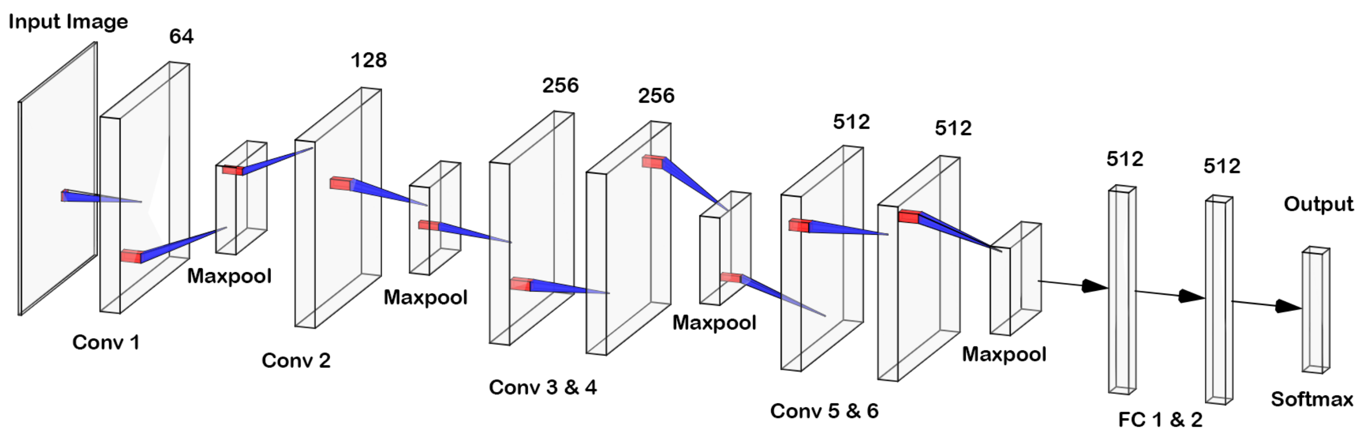

Fig. 4. Architecture of the proposed CNN model. Here Conv means convolutional layer and FC means full connection layer.

layers, 2 full connection layers and an output layer. Number of parameters is estimated to be 8,987,964 which is much less than the original VGG-16 architecture.

Kernel size is kept 3X3 for each convolution layer and applied with a stride of 1. Required padding is added after each convolutional layer to keep the image size same. All the Max pool layers had 2X2 kernels. As per the original paper of VGGNet we use SGD optimizer, momentum value is set to 0.9. Learning rate is initially set to 1e-2 with 1e-6 decay. To improve the performance of our model we tried different regularization techniques. Focused on mainly to avoid overfitting, these regularizers also increase the performance of our model.

### A. *Dropout*

While working with deep networks overfitting is a common problem. Because of large computation time combining many deep models to deal with overfitting is difficult. Dropout [18] is very helpful in this scenario. Dropout basically removes some of the weight connections between two layers. The weights to be removed is selected randomly. It is helpful when the network is getting overdependent on few features. This dropout of weights only occurs during training. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. In the original paper of VGGNet dropout is used only after the full connection layers. Due to smaller dataset compared to the number of parameters in our model, we use dropout after every convolutional and full connected layers.

### B. *Weight Decay*

While training neural networks weight decay is a common regularizer [7] to use. Here weights are multiplied by a regularizing factor less than 1 which prevents the weights from growing exponentially. We introduce a weight decay of 10-6 to each layer except the output layer.

## V. EXPERIMENTAL RESULTS

All the codes are run on Google Colaboratory. Colaboratory is a research tool for machine learning education and research. It's a Jupyter notebook environment which runs on dedicated servers. Colab VM comes with 2-core Xeon 2.2GHz processor, 13GB system memory, 33GB space and an NVDIA Tesla K80 gpu. We use Keras on the Tensorflow background to run the code.

Due to computational constrains, the largest image size we could use was 80X80 pixels. So, we experimented between 3 image sizes- 32X32, 64X64 and 80X80.

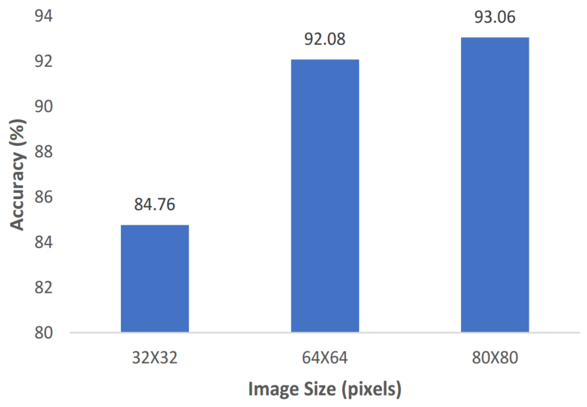

Fig. 5. Model's performance trained with images different size.

With image size 80X80 pixels we got better performance. The result is expected because larger image size means more information and more features to extract.

### A. *Hyperparameter tuning*

Starting with the optimizer, we search for optimized values for all the hyperparameters. After optimizing a hyperparameter we change its value to the optimized one and moved to next hyperparameter. Every time we train the model for 50 epochs. Initially train-validation split was set to 80:20 and batch size to 64. For evaluating the hyperparameter values each time, we used the model's performance on our prepared dataset instead of validation set. This helps our model to be more generalized.

VGGNet like sequential models performs well with SGD optimizer. As suggested in the original VGGNet paper we set momentum value to 0.9 and 1e-6 learning rate decay is used. Keeping these values constant we try 3 different learning rates. As shown in figure-5 SGD with 0.01 learning rate yield the best result.

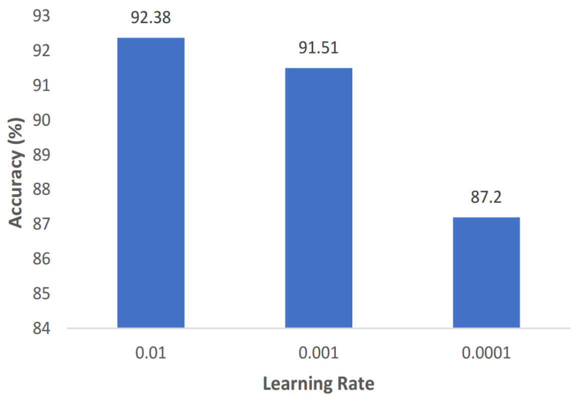

Fig. 6. Performance analysis for different learning rates.

We trained our model for four different batch sizes 32, 64,128 and 256. Batch of 256 shows the best result as shown in figure-6. But with larger batch sizes, time required to train per epoch increased significantly.

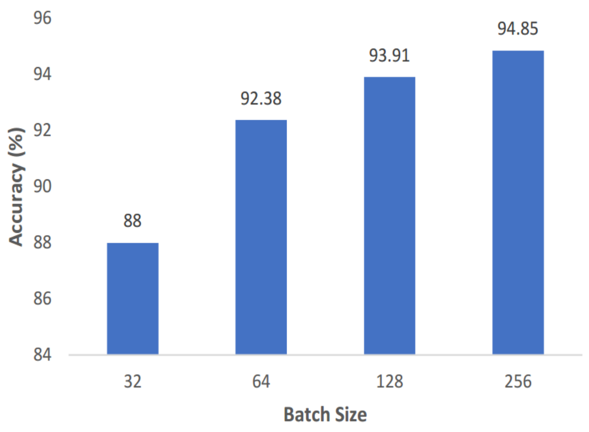

Fig. 7. Performance analysis for different batch sizes.

Using the optimized batch size and learning rate values we searched for optimal train-validation split. Among 5 different splits, 80:20 performed best which was set initially as shown in figure-8

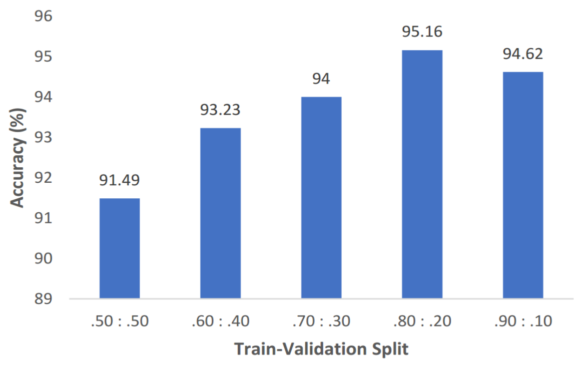

Fig. 8. Performance analysis for different train-validation splits.

In case of number of epoch we checked for 40, 50, 60, 70 and 80 epochs. With little difference 50 epochs proven to be best suited for our CNN model.

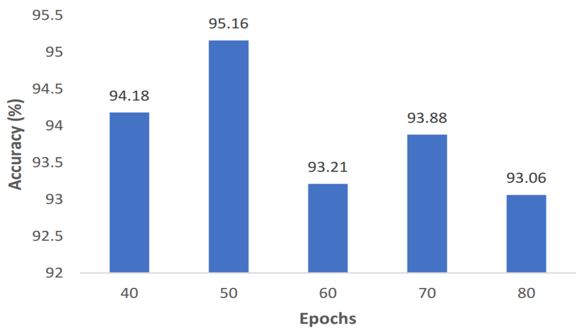

Fig. 9. Model.s performance trained on different number of ephocs.

## B. Result analysis

With optimized hyperparameter and parameter values we train our proposed model to recognize 60 Bangla characters which achieves 99.44% accuracy on the BanglaLekha-Isolated dataset. We then evaluated our CNN model on our prepared dataset yielding 95.25% accuracy. Large performance difference between two datasets suggests overfitting problem. Another thing to consider is that BanglaLekha-Isolated dataset was collected from people of different age, gender, orientation and occupation. On the other hand, our prepared dataset was collected from 25 male student of almost same age. So, various handwriting styles are also responsible for this difference.

TABLE II
FINAL RESULT OF TRAINED MODELS ON BANGLALEKHA-ISOLATED AND PREPARED DATASET

| | BanglaLekha-Isolated | Prepared dataset |
|---|---|---|
| 60 characters | 99.44% | 95.16% |
| 10 numerals | 99.82% | 99.60% |

We again train our model separately only for 10 numerals achieving 99.6% accuracy on prepared dataset and 99.82% accuracy on BanglaLekha-Isolated dataset.

BanglaLekha-Isolated is released recently and not many work is done on this dataset. So for comparison we combined the best performances of both BanglaLekha-Isolated and CMATERdb dataset.As we can see from table III, our model outperforms other previous works on BanglaLekha-Isolated dataset.

TABLE III
COMPARISON OF PROPOSED MODEL WITH PREVIOUS WORKS FOR BANGLA CHARACTERS

| Work reference | Dataset used | Methodology | Accuracy |
|---|---|---|---|
| Sarkhel et al. [19] | CMATERdb | SVM | 87.28% |
| Roy et al. [20] | CMATERdb | DCNN | 90.33% |
| Das et al. [14] | CMATERdb | SVM | 86.96% |
| Alif et al. [16] | Ban-Iso | ResNet-18 | 95.10% |
| Proposed | Ban-Iso | CNN | 99.44% |

Our model also shows state-of-the-art performance on Bangla numerals. Table IV gives the comparison of previous Bangla handwritten numerals recognition works with our proposed model on different datasets .

TABLE IV
COMPARATIVE ANALYSIS OF PROPOSED MODEL FOR BANGLA NUMERALS

| Work reference | Dataset used | Methodology | Accuracy |
|---|---|---|---|
| Shopon et al. [21] | CMATERdb | CNN | 99.50% |
| Liu et al. [22] | ISI | Gradient feature | 99.40% |
| Das et al. [23] |  | SVM | 97.0% |
| Basu et al. [24] |  | MLP | 95.1% |
| Proposed | Ban-Iso | CNN | 99.82% |

## VI. CONCLUSION

In this paper, we propose a 9 layer CNN model which shows state-of-the-art performance on BanglaLekha-Isolated dataset for both 60 Bangla characters and 10 Bangla numerals. But our cross validations performance on prepared dataset shows scope for improvement. Expending cross validation dataset is necessary which includes a wide range of various data. More research can be done in data augmentation techniques. Another way to tackle the problem is by varying the number of layers and number of neurons in each layer. As we saw earlier with the increase of image size the performance our model increases. So, in future larger image size and a deeper model can be used to increase generalization performance of our model.

## REFERENCES

[1] A. Grant, "The cambridge handbook of areal linguistics," 2017.

[2] M. Biswas, R. Islam, G. K. Shom, M. Shopon, N. Mohammed, S. Momen, and M. A. Abedin, "Banglalekha-isolated: A comprehensive bangla handwritten character dataset," *arXiv preprint arXiv:1703.10661*, 2017.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[7] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[8] A. K. Ray and B. Chatterjee, "Design of a nearest neighbour classifier system for bengali character recognition," *IETE Journal of Research*, vol. 30, no. 6, pp. 226–229, 1984.

[9] A. Dutta and S. Chaudhury, "Bengali alpha-numeric character recognition using curvature features," 1993.

[10] A. A. Chowdhury, E. Ahmed, S. Ahmed, S. Hossain, and C. M. Rahman, "Optical character recognition of bangla characters using neural network: A better approach," in *2nd ICEE*, 2002.

[11] Y. Wen, Y. Lu, and P. Shi, "Handwritten bangla numeral recognition system and its application to postal automation," *Pattern recognition*, vol. 40, no. 1, pp. 99–107, 2007.

[12] 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.

[13] T. Hassan and H. A. Khan, "Handwritten bangla numeral recognition using local binary pattern," in *Electrical Engineering and Information Communication Technology (ICEEICT), 2015 International Conference on*. IEEE, 2015, pp. 1–4.

[14] N. Das, R. Sarkar, S. Basu, P. K. Saha, M. Kundu, and M. Nasipuri, "Handwritten bangla character recognition using a soft computing paradigm embedded in two pass approach," *Pattern Recognition*, vol. 48, no. 6, pp. 2054–2071, 2015.

[15] S. Sharif, N. Mohammed, N. Mansoor, and S. Momen, "A hybrid deep model with hog features for bangla handwritten numeral classification," in *Electrical and Computer Engineering (ICECE), 2016 9th International Conference on*. IEEE, 2016, pp. 463–466.

[16] M. A. R. Alif, S. Ahmed, and M. A. Hasan, "Isolated bangla handwritten character recognition with convolutional neural network," in *Computer and Information Technology (ICCIT), 2017 20th International Conference of*. IEEE, 2017, pp. 1–6.

[17] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[19] R. Sarkhel, N. Das, A. K. Saha, and M. Nasipuri, "A multi-objective approach towards cost effective isolated handwritten bangla character and digit recognition," *Pattern Recognition*, vol. 58, pp. 172–189, 2016.

[20] S. Roy, N. Das, M. Kundu, and M. Nasipuri, "Handwritten isolated bangla compound character recognition: A new benchmark using a novel deep learning approach," *Pattern Recognition Letters*, vol. 90, pp. 15–21, 2017.

[21] M. Shopon, N. Mohammed, and M. A. Abedin, "Bangla handwritten digit recognition using autoencoder and deep convolutional neural network," in *Computational Intelligence (IWCI), International Workshop on*. IEEE, 2016, pp. 64–68.

[22] C.-L. Liu and C. Y. Suen, "A new benchmark on the recognition of handwritten bangla and farsi numeral characters," *Pattern Recognition*, vol. 42, no. 12, pp. 3287–3295, 2009.

[23] N. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application," *Applied Soft Computing*, vol. 12, no. 5, pp. 1592–1606, 2012.

[24] S. Basu, R. Sarkar, N. Das, M. Kundu, M. Nasipuri, and D. K. Basu, "Handwritten bangla digit recognition using classifier combination through ds technique," in *International Conference on Pattern Recognition and Machine Intelligence*. Springer, 2005, pp. 236–241.