# Bangla Handwritten Digit Recognition Using an Improved Deep Convolutional Neural Network Architecture

Chandrika Saha, Rahat Hossain Faisal and Md. Mostafijur Rahman
*Dept. of Computer Science and Engineering*
*University of Barisal*
Barisal, Bangladesh
chandrika.cse1.bu@gmail.com, rhfaisal@gmail.com, mostafij.csebu@gmail.com

***Abstract*—Deep Convolutional Neural Network has recently gained popularity because of its improved performance over the typical machine learning algorithms. However, it has been very rarely used on recognition of Bangla handwritten digit. This paper proposes a Deep Convolutional Neural Network (DCNN) based Bangla handwritten digits recognition scheme. The proposed method applies a seven layered D-CNN containing three convolution layers, three average pool layers and one fully connected layer for recognizing Bangla handwritten digits. Rigorous experimentation on a relatively large Bangla digit dataset namely, CMATERdb 3.1.1 provides considerable recognition accuracies.**

***Keywords*—Optical Character Recognition (OCR), Handwritten Character Recognition (HCR), Bangla Digit Recognition, Deep Learning, Deep Convolutional Neural Network.**

## I. Introduction

Optical Character Recognition (OCR) uses images of characters and, converts it into digital writing which is a far better way of storing data as images take more space to store and again data retrieval is much harder from raw image. OCR is used exigently in number plate recognition, data digitizing, automatic key information extraction, smart education, helping visually disabled people, computer vision and many other sectors [1]. Handwritten Character Recognition (HCR) is another form of OCR that extract text data from images containing handwritten characters. The use of HCR is seen extensively in various sectors including handwritten office documents storing, robotics vision, digitizing handwritten text based data, helping visually disabled people and a lot of other sectors. Being the mother tongue of Bangladeshi people and officially used language of some Indian states, Bangla is the fifth largest spoken language around the world [2]. The character set of Bangla language consists of 50 basic characters, 10 digits and 171 compound characters.

For building a HCR we need to design a classifier that will take as input the image of an alphabet and give as output the identification of the alphabet of the image. There are a variety of classifiers that can be used for classifying Bangla alphabet. Some classification techniques that have been used in Bangla HCR are Multi-Layer-Perceptron based classifier [3]–[5], Support Vector Machine [5]–[7], N-fold cross validation based schemes [7], [8] etc. Now-a-days, deep learning is gaining much popularity for giving significant performance gain over the state of art machine learning algorithms. The main reason for this performance gain is that with increased data, the state of art machine learning algorithms does not improve its performance whereas the deep learning model gets better and better with increased data. Deep Convolutional Neural Network (D-CNN) is a certain type of deep learning network that is proven to work well on image data. However, there has been relatively lower number of works on Bangla handwritten basic, compound characters and digits recognition based on D-CNN. Applying D-CNN on relatively large dataset can increase the recognition accuracy for Bangla alphabet. The main objective of this paper is to classify Bangla handwritten digits using a seven layer D-CNN. Therefore, the main contribution of the paper is as follows

- A seven-layered D-CNN architecture is developed
- The architecture is applied to recognize Bangla handwritten digits

The paper consists of six Sections. These Sections are arranged as: Section II gives the literature review, Section III provides discussion on D-CNN, Section IV gives description of the proposed 7 layered architecture, Section V provides the experimental evaluation with comparisons and finally, Section VI concludes the paper with future guidance.

## II. Literature Review

The research works on Bangla OCR began at 1980s [6]. However, most of the works since then was printed document based. Works on Bangla HCR are relatively few. In this Section some remarkable works on Bangla handwritten digit recognition are discussed.

An automatic offline Bangla handwritten numeral recognition method was proposed by Pal et al. [9]. A model based on features obtained from the concept of water overflow was used in this study. The region of the character is imagined to be a water reservoir where the direction of water overflow from the reservoir, position of the reservoir according to the character bounding box, height of water level, shape of reservoir were used as recognition scheme. On collected

978-1-5386-9111-3/19/$31.00 

data from varying individuals they obtained 91.98% recognition accuracy. Pal et al. further extended their studies on Bangla handwritten numerals using the same types of features obtained before [10]. On a dataset of 12000 images they obtained 92.8% accuracy. Khan et al. proposed a scheme for Bangla handwritten character recognition that uses a sparse representation classifier based method [11]. They obtained 94% accuracy on CMATERdb3.1.1 dataset [12], [13]. They used zone density as feature extraction method. Another study was proposed by Hassan et al. [14] where they used Local Binary Pattern (LBP) [15] as feature descriptor and K-Nearest-Neighbor (KNN) algorithm for classification. They obtained 96.7% accuracy on CMATERdb 3.1.1 dataset. Aziz et al. introduced a method where they used local gradient direction pattern based feature descriptor [16]. For classification they used KNN and SVM. On CMATERdb3.1.1 dataset 95.62% accuracy was obtained. Another study [7] suggested a Bangla handwritten character recognition scheme using Local Binary Pattern (LBP) [15] based feature descriptors [17]–[20] and Support Vector Machine based classification for Bangla basic characters, compound characters and digits.

Very recently, alom et al. has developed a five layered D-CNN based architecture for Bangla digit recognition [21]. They evaluated the performance of CMATERdb 3.1.1 dataset using D-CNN with various filters and dropouts.

From the aforementioned discussion we see that Deep Convolutional Neural network based schemes are rarely used for recognizing Bangla digits despite of its effectiveness. Therefore, further studies need to be conducted on Bangla handwritten numeral recognition to address the performance of D-CNN based models on it.

## III. Deep Convolutional Neural Network

The basic architecture of Deep Convolutional Neural Network (D-CNN) consists of some steps which may appear many times in an orderly way in a particular network. The main three layers of a D-CNN are convolution layer, pooling layer and fully connected layer. An particular arrangement of these layers starts with input layer and ends with output layer. These layers are described briefly in the following subsections.

### A. Input Layer

This is the first layer in every D-CNN. Here, typically input is the pixel values obtained from any image. Sometimes raw pixel values are feed to the network, but also some form of pre-processing can be done for better results. For gray scale images the shape of the input would be (height of the image) x (width of the image) x (Number of Channels = 1). In the case of RGB or true color images the input shape will be (height of the image) x (width of the image) x (Number of Channels = 3). However, D-CNN can also take 1D data or 3D data volume and process these with little modification.

### B. Convolutional Layer

This layer is the core part of D-CNN. Convolution operation is the building block of this layer. Fig. 1 shows an example of convolution operation. For convolution operation we need to have some filters or kernels that can find specific structures from a given image data.

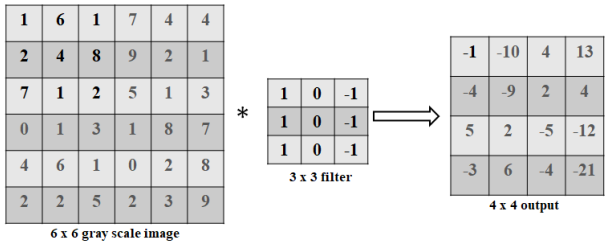


Fig. 1. Convolution operation

Another terminology used in D-CNN is stride which refers to number of cells to be shifted. The way filters work is that, the filter is placed at the top most corner of the input image then an element wise production is performed between the values of the filter and corresponding values of the image pixels after that, the produced results are summed to get a value. The sum gained is then placed at one cell of the output (Fig. 1 depicts the whole process). After that, the filter is moved to the right one stride and same calculations are repeated. Many filters can be used on the same layer. While using more than one filter, the convolution operations are done with all these filters separately and the outputs are then stacked together which are used as input to other layers. A notable

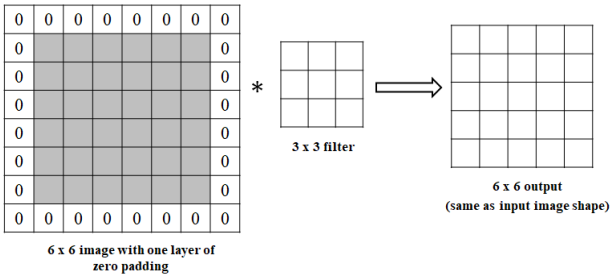


Fig. 2. Zero padding before convolution

point here is that, the depth of the filter in a layer must be equal to the depth of layer channel. That is why, for RGB images filter of shape f x f x 3 is to be used for the first convolution layer as the input image has the shape height x width x 3 (here, f is the size of filter).

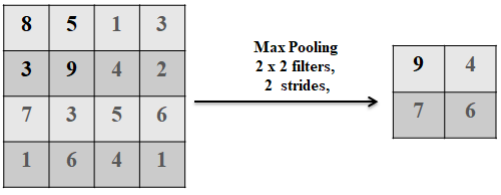


Fig. 3. Max pooling operation

Convolution operation can be of two types, namely, valid convolution and same convolution. Fig. 1 is an example of

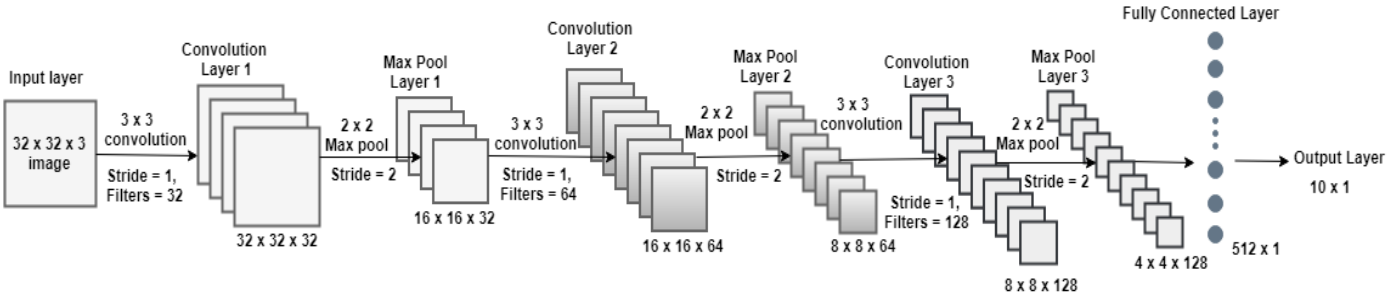


Fig. 4. Overall architecture of the proposed system

valid convolution where the output shape has been decreased. In the case of same convolution operation the input image is zero padded so that the output has the same height and width as the input (Fig. 2). The previously given example shows a predefined filter being used. However, typically the filters to be used are initialized with random numbers. Using feed-forward and backward propagation, these randomly chosen numbers are optimized to fit for any particular problem. One iteration of feed-forward and backward propagation is called an Epoch. We need to have an activation function to add non-linearity for gaining a generalized solution. The output of the convolution layer is added with a bias value (element wise) and then it is passed through an activation function. Different types of activation functions can be used like sigmoid, tanh, ReLU, leaky ReLU function etc. For this study, ReLU activation function, which is used popularly is applied.

### C. Pooling Layer

Pooling layer extracts principle features from the input and reduces the height and width of the input. This layer is typically used after one or more convolution layers. By reducing the dimension this layer increases computational performance also by reducing number of parameters this layers decreases the chance of over fitting. Pooling can be done using several methods, namely, max pooling (where the maximum value is taken from a particular shape of filter), average pooling (the average value from a certain shape of filter) etc. Fig. 3 depict a max pooling operation.

### D. Fully Connected Layers

After passing the input data through some convolution and pooling layers, the output from the last layer is flattened (i.e., reshaped into a linear array) and given as input to the nodes of a fully connected layer. After bringing the outputs of all previous layers into a single linear array a matrix multiplication with the weights is done followed by an addition with a bias value.

### E. Output Layer

After fully connected layer, there is output layer. These last two layers can be compared with last layers of typical neural networks. However, a softmax output unit is generally used to get the output. A softmax unit gives a probability to each of the output classes which adds up to 1. The class containing the greatest value will be the correct class.

## IV. Proposed 7 Layered D-CNN architecture

The D-CNN architecture used for this paper is inspired by LeNet-5 architecture [22]. This architecture consists of 7 layers. The input images are feed to the network taking 50 images as a batch at a time. All images are re sized to have 32 x 32 dimensions and all these images are true color or RGB images. The overall architecture is depicted on Fig. 4. For optimization adam (Adaptive Moment Estimation) optimizer is used. Adam optimizer takes into account the previous values of parameter for making the learning smooth. Update rule for adam is given in (1).

$$\begin{aligned} v_t &= \beta_1 v_{(t-1)} + (1-\beta_1)dw, \\ s_t &= \beta_2 s_{(t-1)} + (1-\beta_2)dw^2, \\ W &= W - \alpha \frac{v_t}{\sqrt{s_t}+\epsilon} \end{aligned} \tag{1}$$

Here, $\beta_1$ and $\beta_2$ are two hyper parameter. $\alpha$ is also a hyper parameter which indicates the learning rate. dw is the gradient of learning parameter and $\epsilon$ is a constant value added with the denominator to avoid division by zero. For implementation $\beta_1 = 0.9$, $\beta_2 = 0.99$and $\alpha = 0.001$ were used which are used conventionally. The parameter and hyper parameter (size of filter, stride, padding number etc.) settings for each layer are described in the following subsection.

*1) Convolution Layer 1:* As can be seen from Fig. 4, this layer takes as input the 32 x 32 x 3 RGB images (where 3 is the number of color channels) and convolves it with 32 different randomly initialized filters. Dimension of each filter is 3 x 3 x 3 and stride of 1 is used to shift these filters. As same convolution is used with zero padding, the output shape is 32 x 32 x 32 of this layer. A bias value is added with this output and then reLU activation function is applied. This final result is then passed to the next layer. The bias value is also randomly initialized.

*2) Max Pool Layer 1:* This layer takes as input the 32 x 32 x 32 output from the previous layer and performs max pooling. For max pooling 2 x 2 filters and a stride of 2 is used. After applying max pooling, the shape of output becomes 16 x 16 x 32.

*3) Convolution Layer 2:* This layer takes as input the previous layers output and run convolution operation with 64 filters. Filters have 3 x 3 x 32 dimensions and stride of 1 is used. Same convolution is used in this layer. The output shape is 16 x 16 x 64. After adding a bias value with this convolution output, reLU activation is applied to add non-linearity.

*4) Max Pool Layer 2:* This layer takes as input the output from previous layer and applies a 2 x 2 max pool with stride of 2. So, the output shape is 8 x 8 x 64.

*5) Convolution Layer 3:* This layer takes as input the previous layers output and run convolution operation with 128 filters. Filters have 3 x 3 x 64 dimensions and stride of 1 is used. Same convolution is used in this layer. The output shape is 8 x 8 x 128. With this output the randomly initialized bias value is added. Then, a reLU activation is applied.

*6) Max Pool Layer 3:* This layer takes as input the output from previous layer and applies a 2 x 2 max pool with stride of 2. So, the output shape is 4 x 4 x 128.

*7) Fully Connected Layer:* The output from previous layer is flattened to be a linear array of 2048 values and feed to the fully connected layer. This fully connected layer consists of 512 nodes. Output from previous layer is densely connected with these nodes.

After the fully connected layer a softmax output unit predicts the categories. All 10 output classes are assigned a probability value that sums up to 1. The class containing highest probability is the chosen prediction.

For training the model, first all the parameters, i.e., the values of the filters and bias values for each layers are randomly initialized. After one pass of feed forward, the adam optimizer uses the derivative of the loss function with respect to parameters to optimize all the parameters to fit the model using (1). Here, loss function is a measurement of how good the model is doing which is calculated by taking a look at the actual class and the predicted class. Many types of loss functions can be used. For this paper, categorical cross entropy is used as loss function. This function is defined using (2).

$$H_p = -\sum_i p_i log(y_i) \tag{2}$$

Here, p is the predicted class and y is the true class. $H_p$ denotes the loss function of p with respect to y.

Adam optimizer finds a parameter set that makes the value of this loss function smallest. Test data used for this model are used to validate the model as well, so that the model is generalized properly.

## V. Experimental Evaluation

This experiment is conducted on a Windows 7 machine containing Intel core i5-3470 CPU with 8 GB RAM and 4GB NVIDIA GeForce 1050 Ti graphics card. The proposed system is implemented with Keras framework [23] with Tensorflow [24] as backend. The dataset used for experiment and obtained accuracy is given in the following subsections.

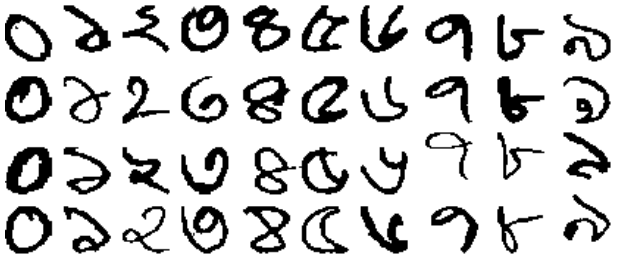

Fig. 5. Example images from CMATERdb 3.1.1

### A. CMATERdb 3.1.1 Dataset

Bangla digits have 10 principle categories like many other languages. For finding out the effectiveness of the current architecture, CMATERdb 3.1.1 dataset is used [12], [13]. Some example images from this dataset are shown in Fig. 5. It is a relatively large dataset of Bangla handwritten digits containing total 6000 images. Each of the 10 categories contains 600 images. Among these 6000 images 4000 is used for training and 2000 is used for testing. Two separate folders are created to store the train and test images. On each folder 10 sub-folders are created to store the images of each class. In train folder each sub-folder contains 400 images and 200 images for each category are used as test images.

### B. Result and Discussion

The dataset was first trained and tested on a LeNet-5 like five-layered architecture which provides 96.80% accuracy. However, to train the dataset on the proposed seven layered architecture, all the images of train folder are divided into 80 batches all containing 50 images. This is done for memory constrains.

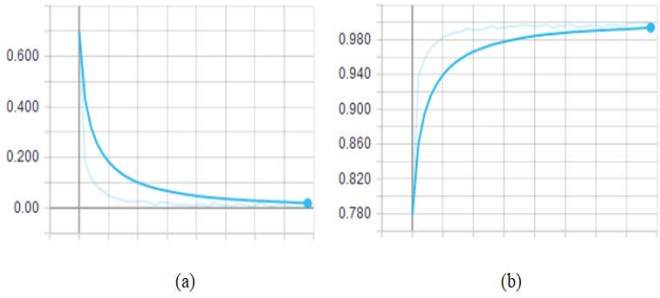


Fig. 6. (a) Training loss and (b) Training accuracy (horizontal axis: No. of epochs)

Total 40 epochs were used to train the model. Fig. 6 gives the graphs of loss values and accuracies for training data with respect to number of epochs. As can be seen from the figure, the training accuracy starts from about 0.78 or 78% and with each epoch as the model is trained, the accuracy keeps increasing. The highest training accuracy obtained is 0.9990 or 99.9%. The model was tested on 2000 image. Fig. 7 shows the test loss and accuracies with respect to number of epochs. As can be seen, the testing accuracy obtained at the end is 0.9760 or 97.6%.

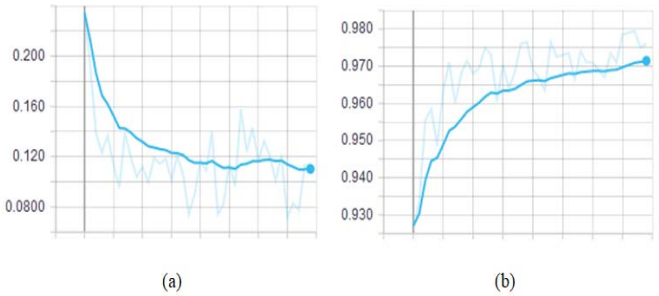


Fig. 7. (a) Test loss and (b) Test accuracy (horizontal axis: No. of epochs)

Table I shows a comparison of the accuracy obtained by the proposed method with four previously used methods on this dataset. Also Fig. 8 shows the comparison graphically. From Table I we can see that proposed 7 layerd D-CNN model performed superior than others.

TABLE I
COMPARATIVE ANALYSIS

| Method Name | Accuracy Obtained |
|---|---|
| Haider Adnan Khan et al. [11] | 94% |
| Basu et al. [25] | 95.1% |
| Aziz et al [16] | 95.62% |
| Hassan et al [14] | 96.7% |
| Five- layered architecture | 96.8% |
| Proposed Seven-layered architecture | **97.6%** |

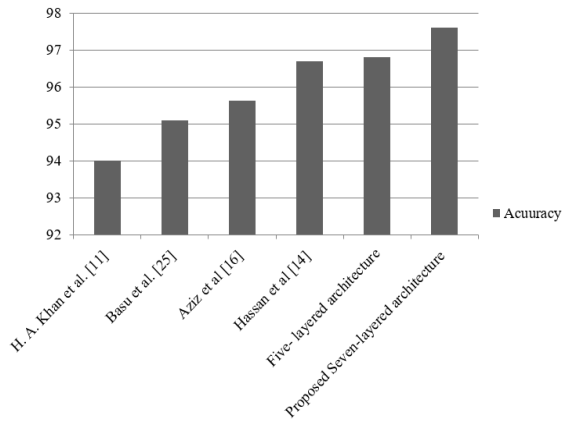


Fig. 8. Comparative analysis

## VI. CONCLUSION

A seven layered D-CNN model is proposed in this paper for Bangla handwritten isolated digits, which provides up to 99.9% accuracy on training data and 97.6% accuracy on test data. However, using data augmentation to increase the amount of data and a deeper network may increase the performance significantly which will be addressed in future works.

## REFERENCES

[1] V. Govindan and A. Shivaprasad, "Character recognitiona review," *Pattern recognition*, vol. 23, no. 7, pp. 671–683, 1990.

[2] G. F. Simons, *Ethnologue: Languages of the world*. sil International, 2017.

[3] 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.

[4] 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.

[5] N. Das, B. Das, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "Handwritten bangla basic and compound character recognition using mlp and svm classifier," *arXiv preprint arXiv:1002.4040*, 2010.

[6] M. T. Pervin, S. Afroge, and A. Huq, "A feature fusion based optical character recognition of bangla characters using support vector machine," in *Electrical Information and Communication Technology (EICT), 2017 3rd International Conference on*. IEEE, 2017, pp. 1–6.

[7] C. Saha, R. H. Faisal, and M. M. Rahman, "Bangla handwritten character recognition using local binary pattern and its variants." In press, 2018.

[8] U. Pal, T. Wakabayashi, and F. Kimura, "Handwritten bangla compound character recognition using gradient feature," in *Information Technology,(ICIT 2007). 10th International Conference on*. IEEE, 2007, pp. 208–213.

[9] U. Pal and B. Chaudhuri, "Automatic recognition of unconstrained off-line bangla handwritten numerals," in *Advances in Multimodal InterfacesICMI 2000*. Springer, 2000, pp. 371–378.

[10] U. Pal, B. Chaudhuri, and A. Belaïd, "A complete system for bangla handwritten numeral recognition," *IETE journal of research*, vol. 52, no. 1, pp. 27–34, 2006.

[11] H. A. Khan, A. Al Helal, and K. I. Ahmed, "Handwritten bangla digit recognition using sparse representation classifier," in *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*. IEEE, 2014, pp. 1–6.

[12] 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.

[13] N. Das, J. M. Reddy, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "A statistical–topological feature combination for recognition of handwritten numerals," *Applied Soft Computing*, vol. 12, no. 8, pp. 2486–2495, 2012.

[14] 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.

[15] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[16] T. I. Aziz, A. S. Rubel, M. S. Salekin, and R. Kushol, "Bangla handwritten numeral character recognition using directional pattern," in *Computer and Information Technology (ICCIT), 2017 20th International Conference of*. IEEE, 2017, pp. 1–5.

[17] M. M. Rahman, S. Rahman, M. Kamal, M. Abdullah-Al-Wadud, E. K. Dey, and M. Shoyaib, "Noise adaptive binary pattern for face image analysis," in *Computer and Information Technology (ICCIT), 2015 18th International Conference on*. IEEE, 2015, pp. 390–395.

[18] M. M. Rahman, S. Rahman, R. Rahman, B. M. Hossain, and M. Shoyaib, "Dtcth: a discriminative local pattern descriptor for image classification," *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, p. 30, 2017.

[19] M. M. Rahman, S. Rahman, and M. Shoyaib, "Mcct: a multi-channel complementary census transform for image classification," *Signal, Image and Video Processing*, vol. 12, no. 2, pp. 281–289, 2018.

[20] J. Wu and J. M. Rehg, "Centrist: A visual descriptor for scene categorization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1489–1501, 2011.

[21] M. Z. Alom, P. Sidike, T. M. Taha, and V. K. Asari, "Handwritten bangla digit recognition using deep learning," *arXiv preprint arXiv:1705.02680*, 2017.

[22] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, p. 20, 2015.

[23] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[25] 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.