Khulna University Studies Special Issue ICSTEM4IR: 863-877, 2022

ORIGINAL ARTICLE **SCIENCE & ENGINEERING**

BANGLA HANDWRITTEN NUMERAL RECOGNITION USING DEEP CONVOLUTIONAL NEURAL NETWORK

Md. Hadiuzzaman Bappy*, Md. Siamul Haq, Kamrul Hasan Talukder

Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh-9208

KUS: ICSTEM4IR-22/0166

Manuscript submitted: July 28, 2022 Accepted: September 28, 2022

Abstract

The recognition of Bangla handwritten numerals (BHNR) has recently emerged as a very interesting area for machine learning and pattern recognition research. Recently, the technology for character or object recognition has also advanced. Bangla handwritten number recognition can serve as a foundation for creating an Optical Character Recognition (OCR) in the Bangla language. However, the lack of a sizable and accurate dataset makes Bangla's handwritten numeral recognition study insufficient in contrast to that of other well-known languages. Similar to MNIST for English digits, NumtaDB is by far the largest dataset collection for handwritten digits in the Bangla language. The most used datasets for the recognition of Bangla handwritten numerals in the past were NumtaDB, CMATERdb, and ISI. The majority of approaches now in use rely on feature extraction and a few outdated machine learning algorithms. Although some approaches operate quickly enough to meet practical demands, they are not always accurate. Even while certain techniques work quite well for languages other than Bangla, they still require improvement. Convolutional Neural Networks (CNN), in particular, are demonstrating excellent achievements in this discipline with higher accuracy. In this work, we've used custom CNN architectures to build our model to recognize digits using all the existing datasets with a high degree of accuracy. Our CNN model shows an average of 98% accuracy recognizing Bangla numeric in respect of above datasets. We have cross verified our model with mixed datasets and the result is also promising.

Keywords: Bangla Handwritten, Numeral Recognition, Optical Character Recognition (OCR), Deep CNN, Computer Vision, Machine Learning

Introduction

Background

Everywhere image processing and classification issues arise, deep learning is applied. It is a subset of machine learning techniques built on artificial neural networks, where the model uses hidden algorithms and optimizations to identify accurate representations, frequently at several layers and with high-level characteristics, when an input is provided. Convolutional neural networks have become an element for

*Corresponding author: <hbappy79@gmail.com>
DOI: https://doi.org/10.53808/KUS.2022.ICSTEM4IR.0166-se

computer vision problems that are used for classification problems. Over the years, these models have been proved very powerful. But in the case of mobile devices, it is still a challenge because of high computational power.

The identification of Bangla handwritten numerals (BHNR) is a well-known issue in the field of computer science vision. This system has several real-world applications, including OCR, postal code recognition, bank check recognition, etc. [Chaudhuri, 1998]. The ability to identify Bangla numerals in documents has grown in significance (Pal, 1994). 10 is the amount of digits in Bangla, hence there will be a total of ten classes. Since each person has their own unique writing style, the challenge of detecting different handwritten digits from provided data sets is far more challenging than it first appears. Obtaining the robust performance and maximal accuracy for a sizable, unprocessed, unbiased, and heavily enhanced NumtaDB dataset [Alam, 2018] is a challenging challenge. But it was much easier for CMATERDB and ISI datasets as they were much cleaner than the NumtaDB dataset. Total six datasets were merged in NumtaDB from various sources and at various times. It contains highly augmented data e.g., noise, blurring, translation, rotation, zooming, shear, contrast, brightness, height/width shift, superimposition, and occlusions. The primary issue with a work like this involving visual recognition is how the features are collected and which algorithm is employed. Support vector machines (SVM), capsule networks (CapsNet), convolutional neural networks (CNN), logistic regression, decision trees, and K-Nearest Neighbor (KNN) algorithms are a few examples of techniques. In the end, our suggested deep CNN model performs admirably with these data sets. For a greater accuracy rate, a sophisticated two-step preprocessing technique (image preprocessing and augmentation) is applied. We next used our deep convolutional neural network model to classify the transformed photos.

Motivation

Extracting Bangla has numerous potentials. The main two of them are described here:

1. Automation of Data Input: In a previous time when we needed to digitized data from a document that had many input fields, then we input those data one by one through a person. But it is so much time-consuming work. It will be solved if we get any way to make it automatically. Motivated from here we want to combine a way to do that and it will be for our mother tongue Bangla. Sometimes we want to store data in a digitized form so that we can use it for further purposes.
2. Analyzing Digital Data: When we will extract data, then we can use those data for further operation. Such as after extracting data from the research questionnaire we can store this as dataset information. We can use it for analysis in the future.

Background Study

Optical character recognition

Because it employs optical methods to collect the characters, character recognition is sometimes referred to as optical character recognition. Using a collection of image processing methods called optical character recognition, it is possible to automatically identify individual characters from a scanned page. The majority of these OCR methods depend on scripts. The translation of any handwritten document into structured text form, automated number plate identification, reading assistance for the blind, bank checks, etc. are just a few examples of the many uses for optical character recognition, or OCR. Characters are utilized to create various linguistic patterns and are regarded as the fundamental building blocks of all languages (Kader, 2012). Character recognition is a procedure that gives items (letters, symbols, and numbers) drawn on an image a symbolic meaning (Kakkar, 2014). OCR significantly aids in the automation process and enhances the interaction between humans and machines in a variety of applications (Prasad, 2013).

An OCR system typically consists of a number of parts. An optical scanner scans the input paper to create a gray-level or binary bit-mapped picture. OCR software frequently transform analog or multilayer

images into black and white in order to conserve memory and computing resources. Thresholding is the name of this method (Alcorn, 1969).

Artificial Neural Network

A computational network that mimics the human brain is called an artificial neural network. The input and output neurons of a simple neural network are connected by weighted synapses. Training or learning is the process of altering the weights. The dataset's quantity and variety affect the learning process. Additionally, the larger and more varied the data collection, the more the neural network will learn over time and become more adept at forecasting outcomes. An ANN receives inputs, computes the weighted sum of the inputs, and then adds bias. A Transfer function is used to express this computation in order to get the desired outcome.

$$\sum_{i=1}^{n} w_i x_i + b$$

Layers

The three main layers of an artificial neural network are as follows:

1. Input layers: The neural network receives a pattern from the outside world. CNN's input layer interfaces with that setting. It just addresses the inputs. The hidden layers will then get this input. The neural network training situation is represented by the input layer.
2. Hidden layers: Between the input layer and the output layer lies a layer called the hidden layer. The activation function is being applied to the group of neurons. Its primary duty is to process the inputs that its preceding layer got. The relevant properties or features must thus be extracted from the incoming data via this hidden layer. There have been several attempts to estimate the number of neurons in the hidden layer, but none of them have been able to provide a reliable figure. A neural network can have several hidden layers added to it. The hidden layer serves no purpose if data can be divided into linear subsets. However, depending on the complexity of the problem or the level of precision necessary, we can employ a few hidden layers up to hundreds of hidden layers when dealing with situations involving complicated judgments.
3. Output layers: The neural network's output layer, the last layer, accumulates prior results and communicates information in accordance with how it was intended to do so. The output layer's exhibited pattern can be used to trace its origins back to the input layer. The amount of neurons in the output layer, also known as the final layer, ought to be proportional to the kind of work the neural network was doing. An illustration of a neural network's several layers (Figure 1).

Activation function

The threshold, also known as categorization or partition, is an activation function. It determines whether or not to fire a neuron by computing the weighted total and then incorporating bias into it. The output layer is only accessible to those that are fired. In a deep learning setting, an activation function's job is to make sure that the representation in the input space is translated into a different space in the output. A neural network behaves exactly like a linear regression model if it lacks an activation function. Depending on the type of jobs, several activation functions, such as ReLu (Rectified Linear Unit), Softmax, etc., are accessible. An artificial neural network's activation function is seen in Figure 2.

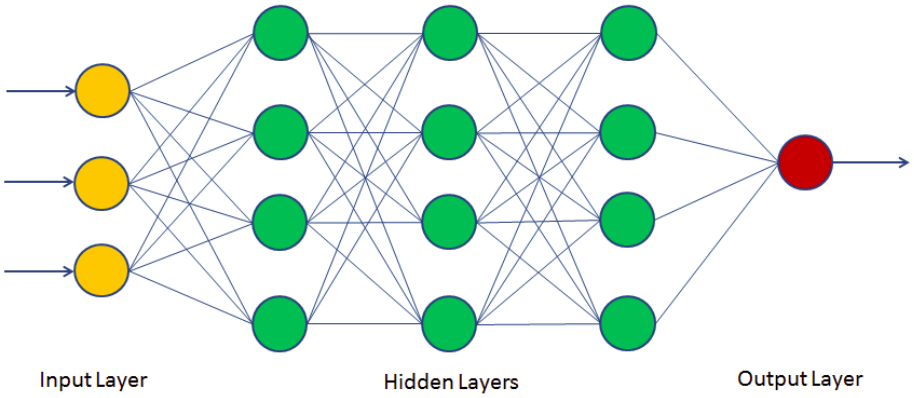


Figure 1. The different layers of a neural network [8].

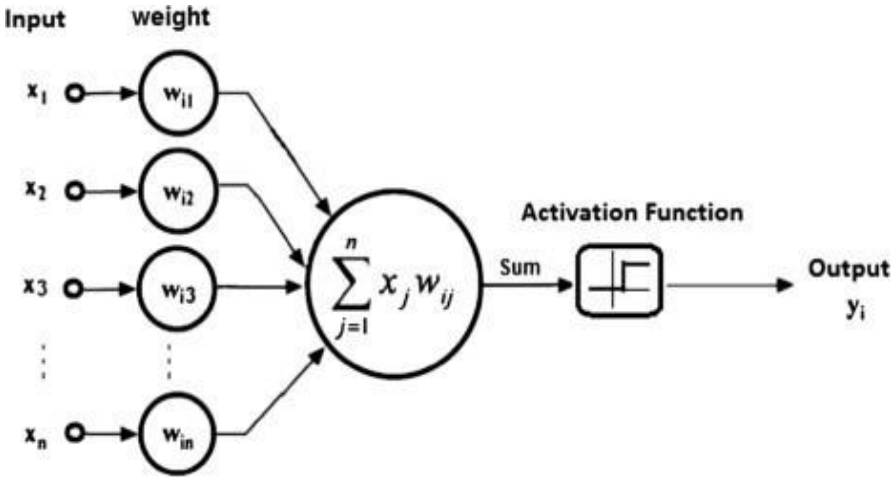


Figure 2. The activation function of ANN.

Convolutional Neural Network

A kind of feed-forward neural network is the convolutional neural network (CNN). Feature learning and classification are its two components. Convolutional and pooling layers make up the first section, while the fully linked layer makes up the second. The second portion conducts non-linear transformations of the acquired features and serves as the classifier, while the first part serves as a feature extractor. A convolutional layer's neurons' primary task is to search for certain characteristics. They generate a high activation if they can identify the precise traits. One convolutional layer can be utilized for this with one or more filters. Most often, the pooling layer is applied just after the convolutional layer. Reducing the spatial size is the pooling layer's primary task (only width and height, not depth). This lowers the amount of parameters, which lowers computation.

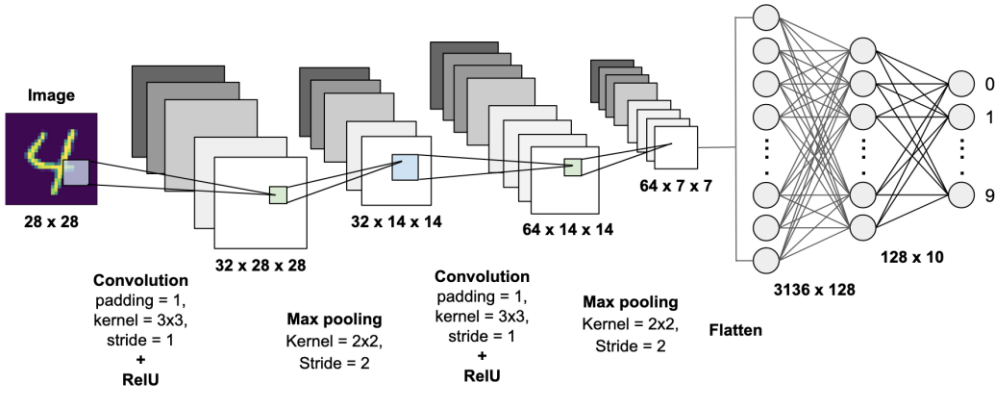


Figure 3. The different layers of a convolutional neural network [9].

Related works

For the recognition of Bangla handwritten numbers, many strategies have been published with good recognition accuracy. Several academics have noted the detection of Arabic text with Arabic numbers in previous years. Several researchers have shown the recognition of Persian (Farsi) handwritten numerals (Mahmoud, 2009). However, showed recognition accuracies for Arabic/Farsi numerals require aggressive signs of progress to be natural.

Huang and Suen (Huang, 1993) showed a proper technique named Behavior-Knowledge Space (BKS) technique is provided to aggregate the disposition gained from single classifiers and obtain the highest disposition from the statistical hint of inspection. The technique is granted for recognition of handwritten Roman digits.

Wen et al. (Wen, 2007) standardized the genuine picture into a 16*16 and 28*28-pixel space, respectively, and showed the pure handwritten Bangla digit without computing any characteristics. In their studies, 16,000 and 10,677 records of handwritten digits were employed, respectively. With the help of the support vector machine (SVM) classifier, they were able to achieve an 86.1% recognition rate. They offered a positive way of numerical identification for handwritten Bangla. The usefulness of the suggested techniques was supported by experimental findings. In the proposed system, they addressed two recognition algorithms based on PCA. For the system to perform better at recognition, they used the integration technique. The combined system's recognition result was more reliable than that of a single technique. The recognition rate was efficient compared to previous related work but the current recognition rate is more dependable than their rate.

In Arabic script writing, Nabawi et al. (Nabawi, 2000) published an absolute approach for the recognition of offline handwritten Indian numerals (0,1... 9). Utilized were gradient, structural, and concavity (GSC) characteristics. The low-level gradient direction frequency was captured by the gradient features. Middle-level geometric properties, such as the existence of corners and lines pointing in different directions, were represented by the structural features. High-level topological and geometrical characteristics, such as the direction of bays, the presence of holes, and big vertical and horizontal strokes, are applied to the concavity features. In addition, they employed the k-nearest neighbor technique together with the SVM (Support Vector Machine) and HMM (Hidden Markov Method) classifiers. The points of the SVM parameters that provide the maximum recognition accuracies were calculated by applying an exhaustive two-tier parameter estimation method (coarse- and fine-search). The outcomes of SVM recognition were compared with those of the HMM classifier. The outcomes of the two classifiers were discussed rapidly. Their study was conducted on a database

of 44 authors, 48 all-digit patterns, and a total of 21120 patterns. 75% of the data were used to train the SVM and HMM technique classifiers, while the remaining data were used to test them. Other portions of the data used for training and testing were computed, and the results were unmatched. When the SVM and HMM classifiers were used, the average recognition accuracy was 99.83% and 99.00%, respectively. It was determined that SVM recognition accuracy was higher for all numbers. Comparison at the writer level (Writers 34 to 44) presupposed that for all seasoned writers, SVM accuracy beat HMM accuracy. The SVM classifier's classification misinterpretation was examined. The technique utilised a larger collection of characteristics, and the SVM classifier was shown to be more effective than the HMM classifier in identifying Arabic (Indian) digits written by independent authors. However, the recognition rate is still not greater than that in (Liu, 2009).

Proposed method

Different methods have estimated Handwritten Bangla digit in different ways. They have provided some optimal solutions for recognizing Bangla numerals. In this research, a concentration on developing a new approach of recognizing Bangla handwritten numerals for more accuracy and less time consumption is given. In this chapter, proposed methods and steps of the experiment with necessary block diagrams and the expected result are shown. Images of the datasets have first been imported and prepped for this investigation. Then, to boost the quantity of training data, we have expanded our dataset.

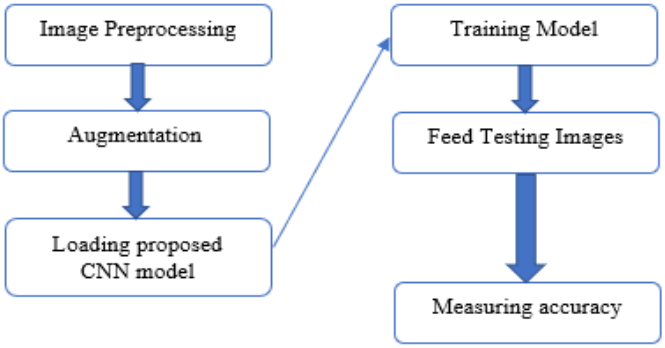


Figure 4. Block Diagram.

Image preprocessing

It includes the process before using the datasets for further processing. We will do some operations which are Noise Removing, Minimum Cropping, Resizing Images, and One Hot Encoding.

Figure 5. Image Preprocessing Steps.

Noise removing

There were several noises in the photos of our Dataset. We employed the Median Filter Blurring approach, which is the best at eradicating salt and pepper noises from photos, to get rid of such noises.

Minimum cropping

For minimum cropping of an image, we used a python library named OpenCV. It provides a more optimal and robust way of image operation in image processing and it supports the deep learning frameworks TensorFlow (OpenCV, 2021).

Sharpening

For this purpose, we used a kernel that will roam over the image and take the central input more precisely and bigger than before. That's why it makes the images more accurate in view and detect for further operation.
kernel = np.array([[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]])

Gray-scale image

By separating the luminance (Y) and chrominance (Cb and Cr) components, the RGB picture input is transformed into a grayscale image. The initial picture utilized for further processing is the gray level (Y) image.

Image Binarization

A grayscale image is used as input in the image binarization process, and the result is a black and white image. By selecting a threshold between 128 and 255 intensity levels and using the parameter supplied by the threshold picture, we binarize the photos. The inverse value is then obtained by subtracting the binarized picture from 255.

Finding max-contour

First, we detect all the contours from the previous output image and store it. Then we calculate the max contour from the contour list. This allows us to detect the minimum region where the number lies.

Cropping from median-filtered image

From the max contour, we take the contour initial position, height, and width. Then we make padding of 5 pixels from each side if it is available to pad. After resizing the width and height of the max contour, we crop this portion from the sharpened image that we generated and stored.

Resize image

For our Convolutional Neural Network training and testing, we need to resize every image in a certain dimension size that will be fed to our model nodes. We take a fixed dimension size which is 64 pixels.

One hot encoding

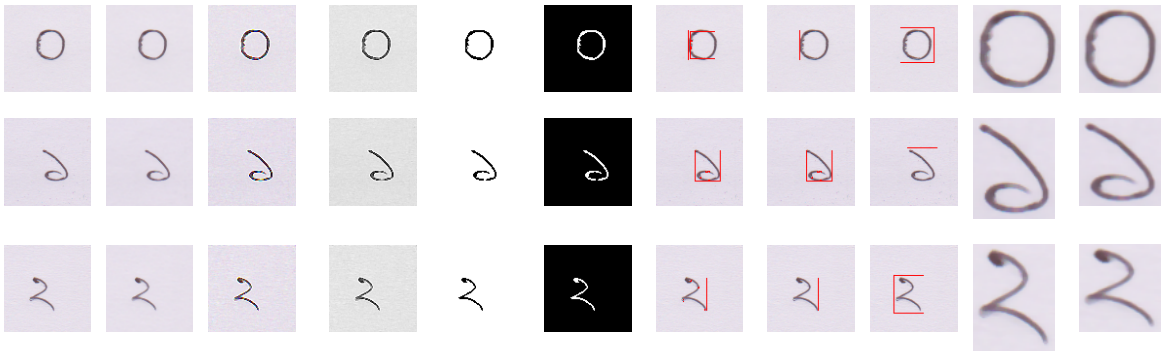
The output layer of our model has ten nodes. The output labels must be transformed into one hot vector as required by a neural network. A binary vector representation of categorical data is known as a hot vector. Each categorical value must be converted to an integer value in order to do this. A binary vector is then used to represent each integer value, with the exception of the integer numbered index, which has a value of 1. Our output labels in this work are already formatted as integers. The number range was defined to be from 0 to 9. Then, for each label, a one-hot vector of size 10 is created, with the value at the index of the number being 1 and the remaining values being 0s.

Augmentation

A method of generating new data with altered orientations, forms, inversions, cropping, padding, and shearing is known as data augmentation. It may be beneficial to add more pertinent data to the dataset. Additionally, it avoids overfitting, which lowers the rate of random mistake. This has to do with how neural networks pick

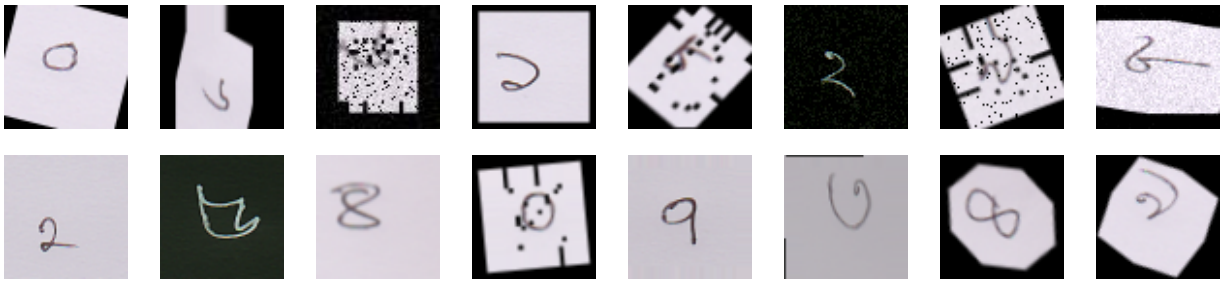
Table 1. The output images after different Image preprocessing. We focused on preprocessing much stronger than any other operation regarding to get more accuracy because when input will be more precise than before then result will also get advantage from those processed images

Original Image	Median Filtered Image	Sharpened Image	Gray-scale Image	Binarized Image	Inverse Binarized Image	Contours in Image	Max Contour in Image	Max Contour with Padding in Image	Cropped Image	Resized Image



things up. We have enhanced the photographs by flipping, rotating, zooming, distorting, erasing part of them, and changing their color, contrast, and brightness. Table 2 displays a few examples of enhanced image output.

Table 2. Highly augmented training images



Training

In this research, we used a custom CNN model with 5 pairs of a 2-dimensional convolutional layer (Conv2D) with 3 Dense fully connected (FC) layer that derives the model to the output result in total. Since the sequential model API works well for creating deep learning models in the majority of circumstances, we specified our model as a sequential model in the first stage. Next, we added two Conv2Ds, each with 32 nodes, a 5x5 kernel size, the Rectified Linear Unit (ReLU) as the activation function, the same padding value, and the glorot uniform algorithm with a seed of 0. In the first layer of this initial pair layer, we contributed. Input shape will be the size and dimension of our training image. In our thesis, we used 64x64x3 (Width x Height x Depth) resized images for both training and testing. We added two Conv2D layers with 64 nodes each, a 5x5

kernel size, the Rectified Linear Unit (ReLU) as the activation function, the same padding value, and glorot uniform with a seed of 0. We added a pair of Conv2D with 64 nodes each, a 5x5 kernel with Rectified Linear Unit (ReLU) as the activation function, the same padding value, and glorot uniform with a 0 seed to the third pair of layers. We added a pair of Conv2D layers with 128 nodes each, a 3x3 kernel size, the Rectified Linear Unit (ReLU) as the activation function, the same padding value, and glorot uniform with a 0 seed to the fourth pair of layers. A pair of Conv2D layers with 128 nodes each, a 3x3 kernel size, the Rectified Linear Unit (ReLU) as the activation function, and a padding value were added to the fifth pair of layers. identical and applied glorot uniform with seed 0.

In every pair of Conv2D layers, we added a Batch Normalization layer, MaxPooling2D layer, and a dropout layer. As we are utilizing a 3-dimensional RGB picture, the batch normalization layer with axis parameter 3 normalizes the activations of the preceding layer at each batch. The MaxPooling2D layer size of 2x2 and dropout with a rate of 0.2, or 20% of input nodes will be randomly eliminated, assist our model avoid overfitting by calculating the maximum or biggest value as a result in every patch of every feature map that emphasizes the most present feature in the patch.

After adding all Conv2D layers, we flatten input layer data perceived from previous layer output from 3 dimensions to a 1-dimensional array for inputting it to the next layer. It is connected to the next classification model, which is called a Dense FC layer. We used a 3 Dense FC layer to calculate the output. The First Dense FC layer has 1024 nodes and ReLU as an activation function. Then we added a dropout function with a rate of 0.2. In the second Dense FC layer, we added 512 nodes and ReLU as an activation function. Then in the third layer, we added nodes with the number of predicted output class which is 10 in our case as our output differs from 0 to 9 and SOFTMAX as the activation function. As a result, the dense layer's nodes' output when added together equals 1.

Then, we built our model using Categorical Cross Entropy as the loss function, Accuracy metrics to assess the model's performance, and Adam optimizer to identify the network node that caused the loss.

The value is 1 in the actual number's index for the one-hot vector of true value, which is represented by the symbol y. As a consequence, the value is anticipated to be close to 1 in the index of the number for the one-hot vector of the projected value, which is y. The greater the network loss, the farther y deviates from this forecast. When there is more than one node in the output layer, categorical cross-entropy uses the predicted value and the real value to calculate the loss.

For better training, we adjusted our model's learning rate to 0.001 and batch size to 64. We saved our model for a better result than previous learned models. We reduce the learning rate with patience 4, factor 0.5, and minimum learning rate 0.00001 values. We split training data 85% for a train set and the rest of the data is used for the validation set. Figure 6 illustrates proposed model.

Testing

After pre-processing and training all the images we run in total 30, 40, 60 epochs to compare results based on the dataset size as we are using 3 datasets to get the resultant accuracy and testing accuracy. The most effective model kept during the training phase was employed for testing. The number of each input number picture from the test set is predicted by the model. We calculated the training and testing accuracy using the expected and actual numbers.

Experimental Results and Comparisons

Datasets

A database of 114988 samples of handwritten Bangla digits is utilized to do tests using the methods previously stated. The database is created by randomly picking samples from a bigger database of 85596 for NumtaDB, 23392 for ISI [16], and 6000 for CMATERdb[17] for each of the 10 digit classes. The largest dataset collection

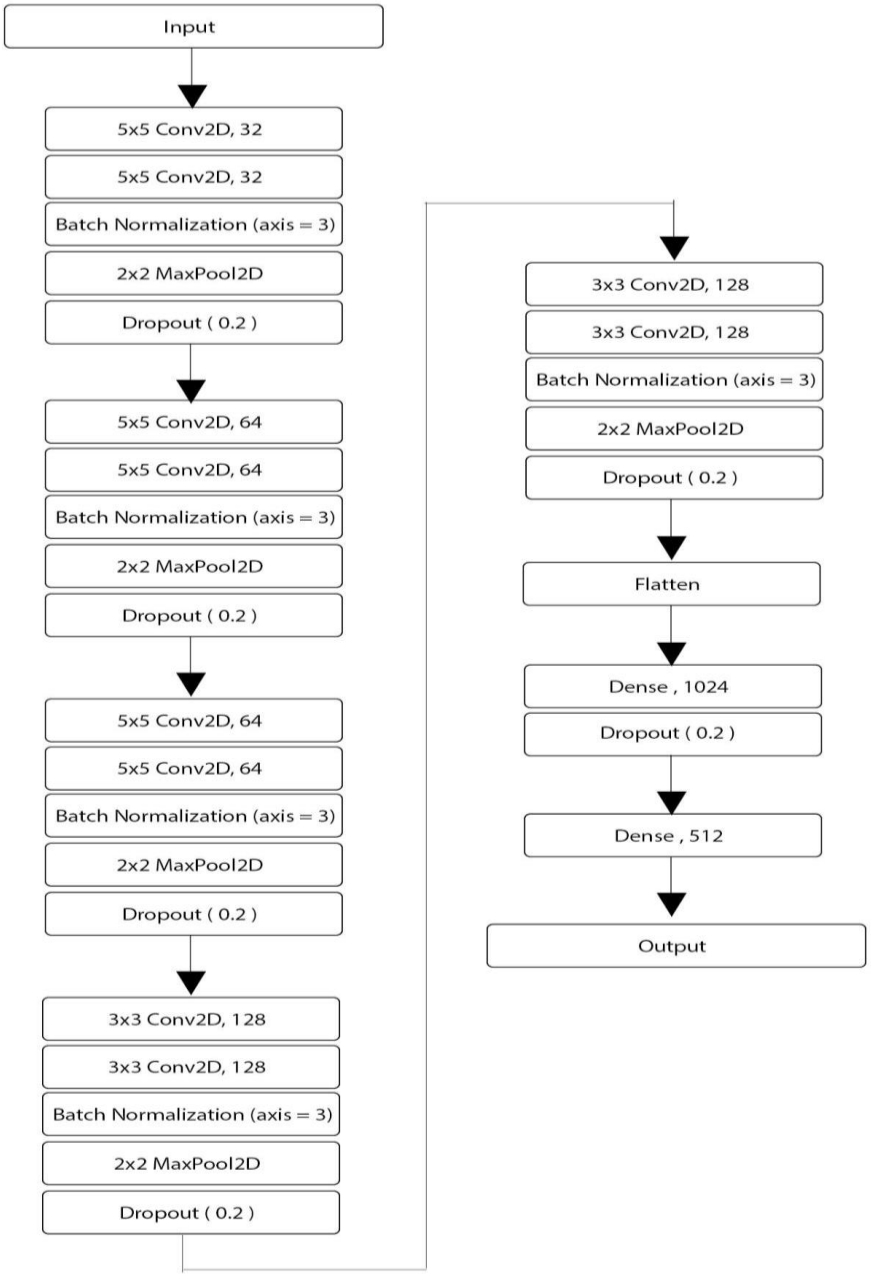


Figure 6. Proposed CNN Model.

for handwritten digits in Bangla is called NumtaDB, and it consists of six sizable datasets gathered from various sources. The samples that made up the datasets were compiled on a pre-defined database sheet from persons of various ages, sexes, and educational levels. As previously indicated, we employed three datasets for

the training set and test set, with the NumtaDB dataset having 72044 samples and 13552 samples, the ISI dataset having 19337 samples and 3986 samples, and the CMATERdb dataset having 4000 samples and 2000 samples, respectively.

Confusion matrix

In this section, Table 4, 5, 6 shows the Confusion matrix corresponding to CMATERdb, ISI, and CMATERdb, ISI mix datasets.

Table 4. For the CMATERdb Dataset, a confusion matrix

Predicted / Actual	0	1	2	3	4	5	6	7	8	9
0	100	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	1	1
2	0	0	100	0	0	0	0	0	0	0
3	0	0	0	100	0	0	0	0	0	0
4	0	0	0	0	98	0	0	1	0	1
5	0	0	0	0	0	100	0	0	0	0
6	0	0	0	0	0	0	100	0	0	0
7	0	1	1	0	0	0	0	98	0	0
8	0	0	0	0	0	0	0	0	100	0
9	0	1	1	0	0	0	1	0	0	97

Table 5. Confusion matrix for ISI Dataset

Predicted / Actual	0	1	2	3	4	5	6	7	8	9
0	192	5	0	1	0	0	0	2	0	0
1	0	194	0	0	0	0	0	0	0	5
2	0	4	189	2	3	0	0	0	0	0
3	0	2	0	188	0	2	7	0	0	0
4	2	2	0	0	190	0	0	0	3	0
5	1	0	0	1	0	191	3	0	0	0
6	0	0	0	0	0	2	197	0	0	0
7	3	0	0	3	0	0	0	193	0	0
8	1	0	0	0	1	0	3	0	195	0
9	0	5	0	1	0	0	0	0	0	193

Experimental results

A significant amount of change may be detected by looking at the accuracy of different machine learning models. Without making any modifications or adjustments to existing models, the accuracy of the majority of models as compared to our model is much lower. Support Vector Machine (SVM) models had a boost inaccuracy of about 84 percent, which was the highest of any model. It is a modest modification for certain models, but it is a significant change for other models. Each iteration of CNN and CapsNet used 80% of the training data and 20% of the testing data. On other models, each iteration used 85% of the training data and 15% of the testing data. With a batch size of 64, 30 epochs have been completed in CNN. Two convolutional

Table 6. Confusion matrix for Train: ISI and Test: CMATERdb Dataset

Predicted / Actual	0	1	2	3	4	5	6	7	8	9
0	100	0	0	0	0	0	0	0	0	0
1	0	99	0	0	0	0	0	0	0	1
2	0	0	99	0	1	0	0	0	0	0
3	0	0	0	100	0	0	0	0	0	0
4	0	0	0	0	100	0	0	0	0	0
5	0	0	0	0	0	100	0	0	0	0
6	0	0	0	0	0	0	100	0	0	0
7	0	0	0	0	0	0	0	100	0	0
8	0	0	0	0	0	0	0	0	100	0
9	0	0	0	0	0	0	0	0	0	100

layers with a dropout of 0.5 and a learning rate of 0.003 were employed in CNN. putting the first filter at 32 and the last filter at 64. The neuron was activated using SoftMax (Paul, 2018).

Table 7. Comparing accuracy for different models on NumtaDB

Training Model Name	Accuracy
Convolutional Neural Network (CNN)	0.913
Capsule Network (CapsNet)	0.871
K-Nearest Neighbor (KNN)	0.779
K-Nearest Neighbor with Principle Component Analysis (KNN with PCA)	0.798
Support vector machine (SVM)	0.845
Support vector machine with Principle Component Analysis (SVM with PCA)	0.742
Logistic Regression	0.742
Logistic Regression with Principle Component Analysis	0.738
A Deep CNN (Islam, 2018)	.961
Proposed Model	0.962

Four layers have been employed to implement CapsNet. The first layer is a convolution layer with a 256-bit filter and stride 1, the second layer is a convolutional layer with squash activation and stride 2, the third layer is a capsule layer on top of which the routing algorithm operates, and the last layer is an auxiliary layer. With a batch size of 32, CapsNet only performs 10 epochs, with a dropout rate of 0.5 and a learning rate of 0.001. Twenty iterations were completed for all the KNN-containing models, and three neighbors were chosen. The number of components was set at 25 for models that included PCA (Paul, 2018).

The percentage of accurately predicted samples to the total number of input samples is known as classification accuracy.

$$Accuracy = \frac{number\ of\ correctly\ predicted\ samples}{total\ number\ of\ samples} \quad (1)$$

The proposed method achieved 96.02% recognition accuracy for NumtaDB, 99.10% recognition accuracy for CMATERdb, and 98.44% recognition accuracy for the ISI dataset respectively.

Table 8. Comparing accuracy for different models on CMATERdb and ISI

Training Model	Datasets	Accuracy
SRC (Khan, 2014)	CMATERdb	0.9400
KNN (Hassan, 2015)		0.9670
SVM+GA (Das, 2012)		0.9770
SVM+NSGA (Sarkhel, 2016)		0.9780
CNN+DBN (Alom, 2017)		0.9878
SVM+CRO (Boni, 2018)		0.9896
Proposed Model		0.9910
SVM+QTS (Roy, 2012)	ISI	0.9735
PCA+SVM (Wen, 2007)		0.9505
SVM+NSGA (Sarkhel, 2016)		0.9823
Proposed Model		0.9844
ACMA (Shopon, 2016)	Train: ISI Test: CMATERdb	0.9950
Proposed Model		0.9980

Conclusion

Bangla is the first language of more than 25 crore people worldwide, and that figure is rising quickly every day. Therefore, creating an OCR in this language is imperative. A technique called optical character recognition (OCR) converts handwritten, printed, or typewritten text into digital data that a computer can quickly understand. Compared to the original image file, these converted electronic formatted files are smaller. The user who had to deal with a large number of papers work, can now easily convert the papers by using OCR. In this era of technical progress, this will surely be a new milestone.

We have presented a well-known optimization problem named the handwritten digit recognition problem which has been solved by many existing supervised learning and meta-heuristics algorithms. The background and motivation describe the necessity of the problem to solve. Some of the existing methodologies are described in the literature review section.

Limitations and Future direction

There is a limitation of the proposed work. The approach can't give 100% accuracy on any given dataset also the execution time is too high. It can only give the best possible result. However, most previous works showed that digit recognition accuracy can be improved day by day and time complexity has also improved. In the future, we may work with the Bangla character dataset since we have not considered the Bangla character in the proposed work.

References

Chaudhuri, B. B., & Pal, U. (1998). A complete printed Bangla OCR system. *Pattern recognition*, *31*(5), 531-549.

Pal, U., & Chaudhuri, B. B. (1994). OCR in Bangla: an Indo-Bangladeshi language. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, 2, 269-273.

Alam, S., Reasat, T., Doha, R. M., & Humayun, A. I. (2018). Numtadb-assembled bengali handwritten digits. *arXiv preprint arXiv:1806.02452*.

Kader, M. F., & Deb, K. (2012). Neural network-based English Alphanumeric character recognition. *International Journal of Computer Science, Engineering and Applications*, *2*(*4*), 1.

Kakkar, P., & Dutta, U. (2014). A novel Approach to Recognition of English Characters Using Artificial Neural Networks. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, *3*(*6*).

Prasad, K., Nigam, D. C., Lakhotiya, A., & Umre, D. (2013). Character recognition using matlab's neural network toolbox. *International Journal of u-and e-Service, Science and Technology*, *6*(*1*), 13-20.

Alcorn, T. M., & Hoggar, C. W. (1969). Pre-processing of data for character recognition. *Marconi Review*, *32*(*172*), 61.

[Online]. Available: https://res.cloudinary.com/dyd911kmh/image/upload/f_auto,q_auto:best/v1547672259/3_qwv5gr.png [Accessed June 12, 2019].

[Online]. Available: https://miro.medium.com/max/3744/1*SGPGG7oeSvVlV5sOSQ2iZw.png [Accessed June 17, 2019].

Mahmoud, S. A., & Olatunji, S. O. (2009). Automatic recognition of off-line handwritten Arabic (Indian) numerals using support vector and extreme learning machines. *International Journal of Imaging*, *2*(*A09*), 34-53.

Huang, Y. S., & Suen, C. Y. (1993). The behavior-knowledge space method for combination of multiple classifiers. In *IEEE computer society conference on computer vision and pattern recognition*, 347-347.

Wen, Y., Lu, Y., & Shi, P. (2007). Handwritten Bangla numeral recognition system and its application to postal automation. *Pattern recognition*, *40*(*1*), 99-107.

Nabawi, A. A. F., & Mahmoud, S. A. (2000). Arabic optical text recognition: a classified bibliography. *ERJ. Engineering Research Journal*, *23*(*1*), 79-131.

Liu, C. L., & Suen, C. Y. (2009). A new benchmark on the recognition of handwritten Bangla and Farsi numeral characters. *Pattern Recognition*, *42*(*12*), 3287-3295.

"OpenCV" [Online]. Available: https://en.wikipedia.org/wiki/OpenCV [Accessed Jan. 7, 2021].

"Off-Line Handwritten Bangla Numeral Dataset." [Online]. Available: http://www.isical.ac.in/~ujjwal/ [Accessed August 7, 2019].

"CMATERdb 3.1.1: Handwritten Numeral Database." [Online]. Available: http:// code. google.com/p/cmaterdb/ [Accessed August 13, 2019]

Paul, O. (2018). Image pre-processing on NumtaDB for Bengali handwritten digit recognition. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, 1-6.

Islam, K. M., Noor, R., Saha, C., & Rahimi, J. (2018). A Deep Convolutional Neural Network for Bangla Handwritten Numeral Recognition. In *2018 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, 45-50.

Khan, H. A., Al Helal, A., & Ahmed, K. I. (2014). Handwritten bangla digit recognition using sparse representation classifier. In *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*, 1-6.

Hassan, T., & Khan, H. A. (2015). Handwritten bangla numeral recognition using local binary pattern. In *2015 international conference on electrical engineering and information communication technology (ICEEICT)*, 1-4.

Das, N., Sarkar, R., Basu, S., Kundu, M., Nasipuri, M., & Basu, D. K. (2012). A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application. *Applied Soft Computing*, *12*(*5*), 1592-1606.

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.

Alom, M. Z., Sidike, P., Taha, T. M., & Asari, V. K. (2017). Handwritten bangla digit recognition using deep learning. *arXiv preprint arXiv:1705.02680*.

Boni, P. K., Abir, B. S., Hasan, H. M., & Islam, M. R. (2018). Handwritten bangla digit recognition using chemical reaction optimization. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 1-7.

Roy, A., Mazumder, N., Das, N., Sarkar, R., Basu, S., & Nasipuri, M. (2012). A new quad tree based feature set for recognition of handwritten bangla numerals. In *2012 IEEE international conference on engineering education: Innovative practices and future trends (AICERA)*, 1-6.

Shopon, M., Mohammed, N., & Abedin, M. A. (2016). Bangla handwritten digit recognition using autoencoder and deep convolutional neural network. In *2016 International Workshop on Computational Intelligence (IWCI)*, 64-68.