Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0455.v1

Recognition of Brahmi words by Using Deep Convolutional Neural Network

Neha Gautam [1*], Soo See Chai [1], Jais Jose[2]

[1]Faculty of Computer Science and Information Technology, University Malaysia Sarawak, Kuching, Malaysia
[2]Amity University, Noida, India
[*]nehagautam1208@gmail.com

Abstract: Significant progress has made in pattern recognition technology. However, one obstacle that has not yet overcome is the recognition of words in the Brahmi script, specifically the identification of characters, compound characters, and word. This study proposes the use of the deep convolutional neural network with dropout to recognize the Brahmi words. This study also proposed a DCNN for Brahmi word recognition and a series of experiments are performed on standard Brahmi dataset. The practical operation of this method was systematically tested on accessible Brahmi image database, achieving 92.47% recognition rate by CNN with dropout respectively which is among the best while comparing with the ones reported in the literature for the same task.

Keywords: pattern recognition; deep convolutional neural network; Brahmi script; CNN

1. Introduction

A recent problem in pattern recognition is automatic word recognition, which can be recognized by an optical character recognition (OCR) system[1]. OCR is a system that automatically supports the conversion of several kinds of documents like scanned paper documents, various types of written document in format or pictures were taken into editable and searchable data [1]. OCR widely used as a tool for information extraction from printed paper such as passport, invoice, bank statement, and digital receipt. Automatic word recognition has become a research area of pattern recognition for many years because of its various applications and needs [2, 3]. However, very few works have completed on the South and Central Asian scripts [4-7].

Brahmi script is the modern name given to one of the ancient writing scripts used in South, Southeast, and East Asia. A study identified 198 different current scripts like Gurmukhi, Gujarati, and Oriya, which originated from the Brahmi script [8, 9]. Brahmi script is a source of Theravada Buddhism, which is known as “pillars of Asoka” because pillars were written in the Brahmi script [9]. Pillars, caves, stones, etc. was the way to spread the Buddhist culture in Asia, mostly in South, Southeast and East Asia [10].

In the work of Brahmi scripts recognition, the suitable features of the Brahmi scripts are first extracted and next, using the suitable classifier, the extracted features will be classified. For instance, geometric features [9] of the Brahmi scripts had been used as the suitable features for Brahmi scripts recognition, zoning method [4, 11] was applied to obtain the features of the Brahmi scripts and template matching [4] and Artificial Neural Network (ANN) [11] were used as classifier to classify the extracted features.

This study focusses on increasing the accuracy of existing Brahmi word recognition system, and deep learning has performed well to recognize the Bangla script and Devanagari script, which are derived from the Brahmi script. So, this study also uses deep learning to identify Brahmi words.

Characteristic and properties of the Brahmi script is explained in section II. Earlier studies based on the character, word, text recognition is given in part III. The complete recognition procedure illustrates in article IV, Testing, and the result of this study discusses in section V and closing observations give in chapter VI.

2. Brahmi Script

Brahmi script is an ancient script and the period of Brahmi script was 3rd BCE to 6th centuries (CE) [12, 13]. This script was used to write various religious books (Buddhist and Jainism) [14, 15]. At one time, Brahmi script was referred as the "pin-man" script that is "stick figure" script in English [15, 16]. In other words, Brahmi characters have geometric features like lines, curves, corners, and dots [9] and these features can help to recognize the Brahmi script from the various scripts (figure 1).

2.1. Properties of Brahmi script

2.1.1 Compound character are the modified characters in shape, which is the combination of consonant and vowel. The modifications in shape can see on the left, right, top, or bottom of the consonant, it depends on the vowel [17]. Some time, two vowels follow a consonant and become a new character that called complex compound characters. Both properties follow by Brahmi script like Devanagari script, Bangla script.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0455.v1

2.1.2 Brahmi script has 368 total numbers of characters including the number of consonants are 33, vowels are 10, and the rest (325) are compound characters [18, 19].

2.1.3 Left side to right follow to write text in Brahmi script.

2.2. Characteristic

Brahmi consonants are followed by different vowels (Figure 2) to make compound characters (Figure 3). The features are added in the consonants to create compound characters called "Matra". Mostly, these "Matra" can be seen on the outer edge of the consonants (but not always because it also depends on the shape of the consonants) or using a dot feature (.) that is added after the consonant.

Figure 1: Character and vowels of Brahmi script [19]

The first character is the consonant, and the rest are the compound characters, as per Figure 3 which are created by using all vowels of Figure 2, respectively.

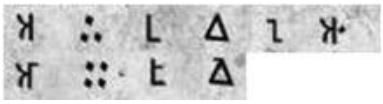


Figure 错误!文档中没有指定样式的文字。: Vowels of the Brahmi script [19]

Nine types of features ("Marta") are used to make compound characters.

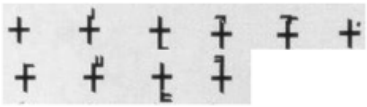


Figure 3: Example of consonant and compound characters [19]

3. Literature review

Of the previous works that have developed Brahmi script recognition systems, only a few focused on Brahmi character recognition and even fewer reported working on Brahmi word recognition. The history of Brahmi script recognition can be traced back to 1983. In this year, Siromoney, et al. [18] developed a system to recognise all Brahmi characters, including all 368 standard vowels, consonants, and compound characters. Their work, however, ignored all complex compound characters. A coded run method was used for the recognition of machine-printed characters of the Brahmi script, with each Brahmi character manually changed into a rectangular binary array. However, the proposed method was a general method, so it can be applied to recognise other scripts. Soumya and Kumar [20] focused on recognising the ancient script in the time of King Ashoka (Brahmi script) using Gabor features and zone-based feature extraction methods to extract the features of these characters and ANN for the classification of the extracted features. The accuracy of this system was 80.2%. Next, Vellingiriraj, et al. [11] focused on developing a Brahmi word recognition system in which they used zonal density to extract the features and ANN to classify the extracted features. Three types of samples: vowels, consonants, and consonantal vowels (compound characters), were used to measure the performance of the system, achieving accuracies of 93.45%, 93.45%, and 90.24%, respectively. Apart from Brahmi word recognition, in 2016, Gautam, et al. [4] focused on Brahmi character recognition. The zone method was used to extract the features of the Brahmi characters, and the template matching method (coefficient correlation) was used for the classification of all extracted features, achieving an accuracy of 88.83%. However, this method could not recognise non-connected characters. Another work also developed a system to recognise Brahmi characters [9], focusing on the geometric elements in each zone (corner point, bifurcation point, ending point) and the geometric features (corner point, bifurcation point, ending point, intersect point, circle, and semi-circle) to extract the features. In addition, some sets of rules were created for the classification of all these features, yielding a final accuracy of 94.10%. Apart form that the translation of Brahmi script into Pali language is also suggested for understanding the Brahmi words [21].

Various studies have been performed to recognise several scripts. These days, CNN and DCNN are very popular classifiers for classifying characters, digits, words, etc. Both classifiers have been investigated for various script recognition systems involving the English script, the Chinese script, and other scripts under the Abugida system, returning very good performances. The Abugida system is part of the world writing system to which the Brahmi script belongs. Therefore, this review only focused on the script in the Abugida system.

Adak, et al. [22] suggested a method for offline cursive Bengali word recognition. The authors used a hybrid model of CNN and a recurrent neural network (RNN). The overall word recognition accuracy obtained by the study was 86.96% using a NewISIdb database, whereas the proposed architecture was segmentation free. Various studies focused on word recognition without segmentation. However, a dictionary or a book might be required to predict the word and obtain the information for all possible combinations of characters, which will create a meaningful word. This study did not find any Brahmi dictionary or book in the literature. Therefore, a further literature review focusing only on those studies that used isolated characters to train and test the system was conducted.

Recently, deep learning-based techniques have attracted growing attention for character recognition [23,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0455.v1

24]. Bangla numbers and digits have been classified using CNN, achieving 77.01% [25] and 98.37% [26] accuracy, respectively, whereas Cecotti and Vajda [25] used 10,000 training datasets for 10 classes, i.e. 1,000 (average) samples for each type. Maitra, et al. [26] used 13,392 datasets for 10 classes, resulting in 1,339 (standard) samples for each category. Bangla characters were classified with CNN and achieved 95.6% accuracy, where 25,000 samples were used to train 50 levels [26], i.e. 500 (average) examples for each category. Similarly, Nair, et al. [27] used two lack samples to train Bangla characters, where the Bangla script had 50 types of characters, including vowels and consonants. So, it can be said that the author used around 4,000 samples (average) for each class to train Bangla characters. Previous authors [27] suggested a method and mentioned that CNN could prove to be a state-of-the-art method for other script and hence could provide the possibility to also produce a higher recognition rate for Malayalam characters. DCNN was used to train Bangla compound characters and achieved 90.33% accuracy [28]. The authors used the CMATERdb 3.1.3.3 database, and the dataset had 34,439 individual samples for training the 171 unique classes of Bangla compound characters. That is, on average, 201 samples (average) were used to train each Bangla compound characters in the study.

Maitra, et al. [26] used 18,794 samples to train ten classes of Devanagari numbers using CNN and achieved 98.54% accuracy. That is, around 1,879 examples (average) were used for each category. Similarly, 92,000 samples were used to train 46 different classes [29], and 33,120 examples for the training of 42 levels [30] were used to classify Devanagari characters using the CNN classifier, achieving 98.47% and 98.19% accuracy, respectively. DCNN was also used to classify Devanagari characters, where 49,000 training samples of 49 classes [31] and 36,172 characters of 47 levels [32] were used to classify the Devanagari characters, obtaining 90.58%, and 96.02% accuracy, respectively.

Apart from the Bangla script and the Devanagari script, CNN was also used to classify the Telugu script, the Tamil script, and the Oriya script. CNN was used to train 2,000 samples of 10 classes of Telugu numbers [26] and 47,428 examples of Telugu characters including 36 consonant classes and 15 vowel modifier classes [33], and achieved 96.5% and 92.26% accuracy, respectively. Similarly, 18,535 samples were used to train 35 levels of Tamil characters using CNN and reached 94.4% accuracy [34]. Four thousand nine hundred seventy samples were used to train ten classes of Oriya numbers using CNN and achieved 97.2% accuracy [26].

Deep learning has a limitation in that it requires many samples of each class to train the system. Apart from this issue, two main points also emerge: computation time and over-fitting. To minimize these issues, GPUs are able to increase the computation time meaningfully [35]. So, it can be said that the time issue depends on the performance of the GPU. Various regularisation methods have already been announced to solve the issue of over-fitting. Dropout is a current introduced regularisation technique, which is newly worked in deep learning. It is related to bagging [36], in which a set of models are trained on different subsets of the same training data. Vijayaraghavan and Sra [34] used CNN and dropout together to recognise the characters in Tamil script. They achieved an accuracy of 94.4% using the WFHR-10 Tamil character dataset from the HP Labs India website [34]. Similarly, Alom, et al. [37] tested various combinations of SVM, DBN, CNN, Gaussian filter, Gabor filter, and dropout on an official dataset consisting of Bangla digits to check the performance and noted that the combination of CNN, Gabor filter, and dropout achieved the highest accuracy of 98.78%.

In conclusion, CNN with Gabor filter performed well in recognising Devanagari characters. In addition, the combination of CNN, Gabor filter, and dropout achieved the highest accuracy in identifying Bangla digits. Therefore, this study will also apply this same combination to determine the accuracy of the proposed Brahmi word recognition system.

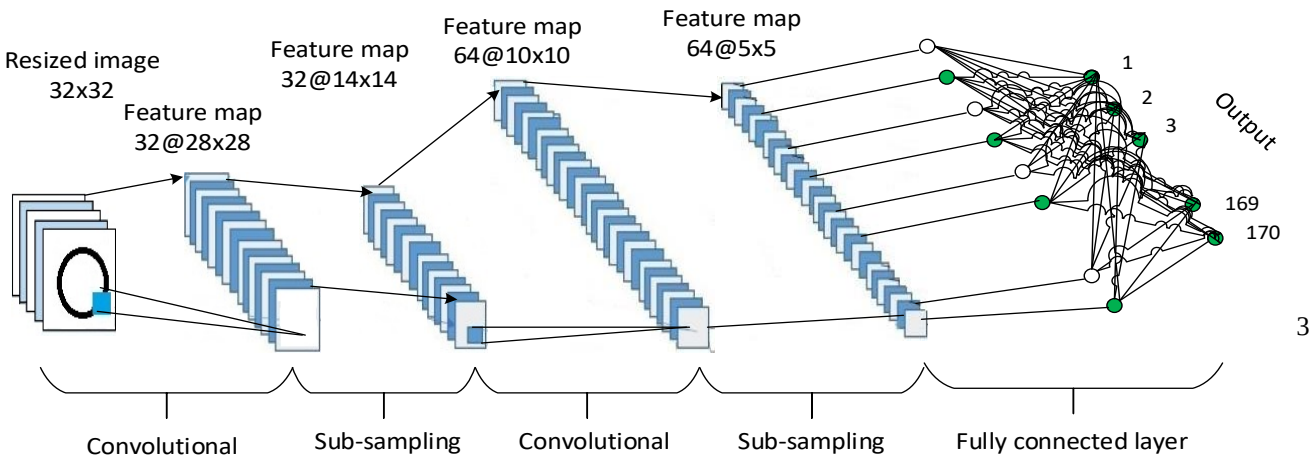
4. Methodology

The recognition of a character image usually undergoes segmentation (line and character detection), pre-processing (grayscale, binarization, and size normalisation), feature extraction, and classification. However, all characters have already been isolated in this study, so segmentation was not required. Pre-processing is used to increase the quality of the samples. After that, feature extraction and classifier step the decision taking parts.

4.1. Input image

Primary job of the word recognition system is input the samples. In this study, the image of Brahmi words in JPG format was obtained and treated as the dataset in the first step. The size and resolution of the pictures were not fixed, so a pre-processing step was applied to enhance the quality of the samples and to convert it into useful content.

4.2. Pre-processing



Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0455.v1

Pre-processing is an essential step for enhancing the visibility and readability of input. The pre-processing steps employed by this study are thresholding and size normalisation (resizing), further elaborated below.

5.2.1 Binarization: Grayscale image can be converted into binary image by Binarization (thresholding). The main motive of thresholding techniques is to enhance the visibility of edge, in which case, the shape of the region is more important compare to the intensities of pixels. Two approaches: global threshold, local threshold can use under binarization. A single threshold value is used for the whole image according to the background of the image in the global threshold. Hence, this method is useful if background of the image is simples and uniform. Although, various values threshold for each pixel is chosen based on the data of local area in local threshold. Usually, this approach is used for real-life documents because, sometimes, these documents are designed deliberately with stylistic, colourful, and complex backgrounds.

The background of all images in the Brahmi dataset was white and therefore not complicated [38]. Hence, a Global threshold approach was used to perform the thresholding process in this study.

5.2.2 Resized: Normalisation was applied to obtain all characters of uniform size. The characters varied in size, so their size should be made uniform to enable comparison. For example, the input of ANN and SVM should be in vector form of a fixed size. Normalisation should reduce or increase the original size of the image however, shape of the image should not be changed.

The size of the image depends on the feature extraction and classifier method. Hence, the input images were resized to 32 × 32 pixels for further work. After that, CNN and CNN with dropout were used to recognise the Brahmi words.

4.3. Convolutional Neural Network

Kunihiko Fukushima [39] announced the Convolutional Neural Network (CNN) in 1980. However, its complicated training algorithm made it hard to use at the time. In the 1990s, promising outcomes were obtained when LeCun, et al. [40] used a gradient-based learning algorithm with CNN. Following this achievement, CNN research improved significantly, achieving better pattern recognition rates. CNN's design allows it to mimic human visual processing, and its structures can be optimised towards the feature extraction and abstraction of 2D features. CNN's max-pooling layer is particularly functional for the absorption of shape variations. Further on, compare to that type of network which are fully connected and similar in size, parameters play more important role. Ultimately, CNN enables trainability through the gradient-based algorithm.

The gradient-based algorithm inculcates an entire network with a straightforward minimisation of the error criterion. Therefore, CNN can output significantly optimised weights. In Figure 4, the comprehensive architecture of CNN can be shown to comprise of two stages: feature extraction and classification. Looking at the feature extraction layers, the input of each layer of the network is the output from its immediate previous layer, as the input of next layer use as output of the previous layer. Convolution, max-pooling, and classification layers are used in the suggested architecture where, low-middle level of the network build by convolutional and max-pooling layers.

Convolution works through even-numbered layers, while max-pooling works through odd-numbered layers. Output nodes from convolution and max-pooling are grouped into a 2D plane, a process known as feature mapping. Every plane of a layer commonly incorporates a unity of one or more than one planes from the past layers, and a small region of each connected planes of the previous layer is associated the node of the plane. The convolution operation is performed on input nodes, where feature extraction process is conducted in each node of the convolution layer. Abstracting features through average or propagating operation is performed by the max-pooling layer also on input nodes.

Spread features of lower level are used to get the features of upper level. During propagation to the highest level, measurements of the features are reduced relative to

Figure 4: The overall architecture of DCNN applied in this study, which contains an input layer and multiple

the dimensions of the convolutional and max-pooling masks. However, for better recognition performance of input images, feature mapping numbers usually increase. The fully connected network, known as classification layers, gets its input from the last feature map outputs of CNN. In this study, feed-forward neural networks were employed for the classification, as it has provided higher performance in contrast of previous studies [37].

Feature selection techniques determine the required number of features in the classification layer with regard to the dimensions of the weight matrix of the final ANN. The extracted features are used for the classification, where prediction of the input sample is computed. After it, the classification process is conducted by using the extracted features for the calculation the assurance for the input samples. According to maximum assurance, the classification process suggested a group where the input images belong to. The next section discusses the mathematical characteristics of various layers of CNN.

5.1.1. Convolution Layer: Kernels such as Gabor filter, which has high learnability, are united to the feature maps of current layers to further procced the convolution layer. Moreover, for the developing of output feature map, Sigmoid, Hyperbolic Tangent, Softmax, Rectified Linear, and Identity Functions types of linear or non-linear functions to introduce the kernels. The mathematical model of this process is given by Equation 1:

$$x_j^i = f\left(\sum_{i\epsilon M_j} x_i^{i-1} k_{ij}^i + b_j^i\right) \quad (1)$$

Where x_j^i, x_i^{i-1}, k_{ij}^i, b_j^i, and M_j represents the output of the current layer, output of the previous layer,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0455.v1

kernel of the current layer, bias for the present layer, and selection of input maps, respectively. For each output map, there is an additive bias b. Nevertheless, input maps and different kernels are combined to introduce the corresponding output maps.

5.1.2. Sub-sampling Layer: In this layer, a down-sampling operation is performed on the input maps. The input and output maps remain as they are in the sub-sampling layer, for instance, let there be number of input maps will be equal to the number of output maps. As a result of down-sampling, reduction in sizes of output maps can be done, which are dependent on the size of the down-sampling mask. In the research carried out here, a 2 × 2 down-sampling mask was applied. The mathematical formula for this operation is described by Equation 2:

$$x_j^i = f\left(\beta_j^i down\left(x_j^{i-1}\right) + b_j^i\right) \quad (2)$$

Sub-sampling function is represented by *down* in Equation 2, which commonly performs a sum over an n × n block of maps from the preceding layers and chooses the average or maximum values of this block. According to this, regarding to the sizes of feature map, n times size is decreased of the output map. After that, the output maps are set under the linear or non-linear activation functions.

5.1.3. Classification Layer: This layer is used the extracted features by previous convolutional layer to calculate the predication value of each class of each sample. It is a fully connected layer. For the experiment, this study considers the dimensions of the feature map to be 5 × 5, and the classification layer as a feed-forward neural net. As recommended by most studies in the literature, the Sigmoid function was used as the activation function.

5.1.4. Back-Propagation: Convolutional procedure is performed between the convolutional layer and sub-sampling layer, and the meanwhile, filters are modified. Accordingly, the weight matrix is computed for apiece layer.

4.4. Convolutional Neural Network with Dropout

The reduction of test errors can effectively be achieved by combination of the predictions of various models [37]. However, it is a computationally costly process for large ANN and processing time can be various days. Alternatively, here, also another effective method for combining models, known as “dropout” [41]. This method sets the outputs of hidden layer neurons to zero for a probability of less than or equal to a present value, 0.5, for instance. The “dropped out” neurons do not have a direct effect on BP.

The dropout technique somewhat successfully simplifies the network, since it provides the co-adaptation of neurons, where one set of neurons does not rely on the presence of another set of neurons. Therefore, it is forced to learn robust features that are useful in aggregation with many different random subsets of the other neurons. Dropout requires more iterations before reaching the needed convergence level and it is a disadvantage of Dropout method. In this study, the dropout method was employed in the starting two fully connected layers.

5. Experimental results and discussion

5.1. Structure of the DCNN and parameters

Six layers are applied to introduce the deep convolutional neural networks (DCNNs) architecture to recognise Brahmi words in this study. Two-two layers were allocated for convolution and pooling, while the last two layers were allocated for classification. The first convolution layer and the second convolutional layer had a 32-output and 64-output mapping, respectively.

Further discuss steps are followed to compute the parameters of convolutional network: the input image size was set to 32 × 32 pixels. 28 × 28 with 32 feature maps as the output of the first layer (convolutional layer). A 5 × 5 Filter mask size was used for both convolution layers. Therefore, a total (5 × 5 + 1) × 32 = 832 parameters were used to learn and 28 × 28 × (5 × 5 + 1) × 32 = 652,288 represented the total number of connections. Zero parameters were used for the first subsampling layer in which the output of this layer was 14 × 14 with 32 feature maps. Similarly, the parameter of the next two convolutional and subsampling layers was calculated. For example, ((5 × 5 + 1) × 32) × 64 = 53, 248 represents the learning parameters of the second convolution layer, and 0 represents the parameter of the sub-sampling layers. The next layers were the fully connected layers so 312 × 64 × (5 × 5 + 1) = 519168 were the parameters of the first fully connected layer, while 170 × (312 + 1) = 53210 were the parameters of the final layer. Hence, the total number of parameters used in this study was 626,453 (Table 1).

5.2. Performance evaluation

In order to compute the accuracy of the suggested architecture (Figure 4), this study used CNN to classify Brahmi words. In order to classify the Brahmi words, 3/4 of the 6,475 data was selected as training data, 1/4 as validation data and 536 as test data.

While training CNN having four convolutional layers for the experiments, this study considered the learning rate, the number of hidden neurons, and the batch size as parameters. It was observed from the results that CNN worked efficiently by increasing the network scale with one major drawback of the problem of over-fitting due to a longer time incurred while training. So, to resolve the overfitting issue, dropout techniques used the

Layer	Operation of each Layer	Number of feature maps	Size of feature maps	Size of window	Number of parameters
C1	Convolution layer	32	28×28	5×5	832
S1	Max-pooling layer	32	14×14	2×2	0
C2	Convolution layer	64	10×10	5×5	53243
S2	Max-pooling layer	64	5×5	2×2	0
F2	Fully connected layer	170	1×1	NA	53210

Table 1: Parameter setup for CNN

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0455.v1

value of dropout parameter was 0.5.

The experiment was conducted using two approaches: 1) CNN with the Gabor filter and 2) CNN with dropout and the Gabor filter, to recognise the Brahmi words. This study considered a total of ten iterations for training and testing. The testing accuracy using CNN with the Gabor filter achieved 91.65% while CNN with dropout and the Gabor filter provided 92.47% accuracy. In conclusion, the performance of CNN with the Gabor filter and dropout was better for Brahmi word recognition (Table 2).

Table 2: Performance of the unsupervised techniques in recognising Brahmi words

Method	Accuracy
CNN with Gabor filter	91.65%
CNN with dropout and Gabor filter	92.47%

This study is considered the learning rate, the number of hidden neurons, and the batch size as parameters to train the CNN which have four convolutional layers. According to the observation of the results, DCNN worked efficiently according to the network scale. However, over-fitting problem appeared during the training because of the long duration. Moreover, batch size can help to reach the optimum level of the system. However, if the batch size is enhanced to some certain value so, the rule of thumb is the suggested method cannot be trained [42]. Moreover, the batch size was also dependent on the available memory. Achieved accuracy of system is presented in Figure 5 while values of the batch size were changed.

To minimize the over-fitting issue, this study trained the model in a controlled environment using a momentum value of 0.1 and it assisted to obtain the optimum performance. This study increased the number of convolutional cores gradually to reach the optimum state of the network because of the overfitting issue. In the case of batch size, relatively large numbers were required to reach the global gradient. This study preferred the 0.0025 and 42 value of the learning rate and a batch size, respectively, and an average accuracy rate of 92.03%. As can see that in performance graphs, number of epochs is directly proportional to the number of hidden neurons to maximize the performance. Although, hidden neurons and number of classed are two different entities here. In practice, the number of hidden neurons in each new hidden layer equals the number of connections to be made. It is also mentioned in the structure of CNN that the internal layer may be comprised of different numbers of hidden neurons. These neurons help in choosing the features of the input image as deeply as possible. It is pertinent to mention that adding to the number of neurons may increase the complexity, but it helps achieve a higher accuracy rate. The same set of experiments was performed with the standard dataset of Brahmi words [38]. The same set of parameters with different values was applied for this set of experiments. We chose a learning rate of 0.08 with a starting batch size of 50, the final test accuracy was around 91.65% for Brahmi word recognition. Efficiency graphs using the different number of hidden neurons for the Brahmi words is shown in Figure 6 where achieved accuracy was 91.54%, 90.65%, and 91.65% with 10, 30 and 50 hidden neurons respectively.

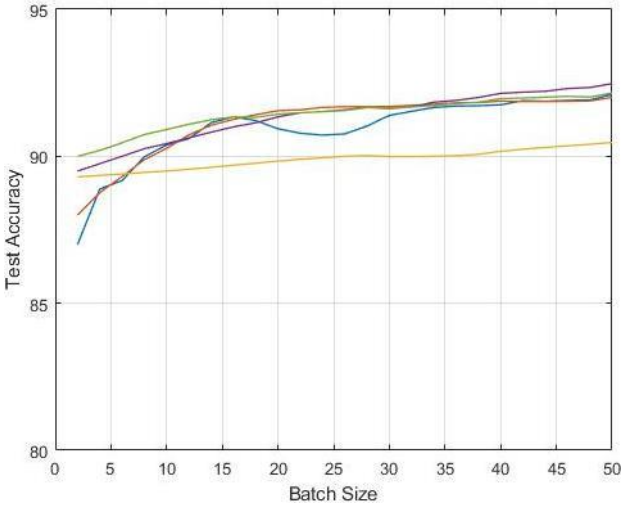
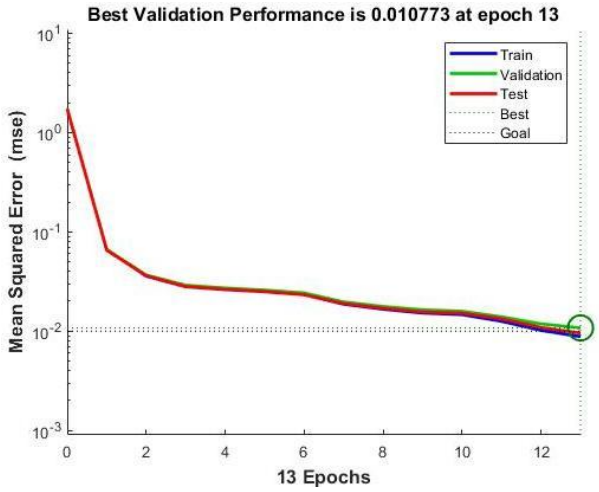
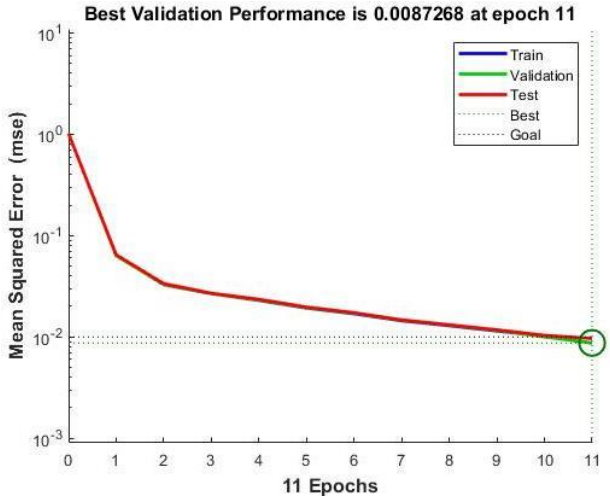


Figure 5: Effect of batch size and on accuracy

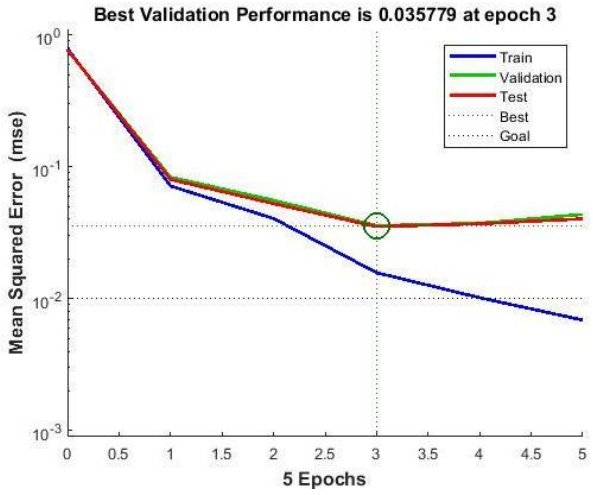
The experiments were also performed using variations of the n-fold cross-validation approach in order to avoid any confusion regarding the ratio of training and testing data. The average accuracy of the Brahmi word recognition was 89.34%, 90.84% using 10-fold and 8-fold cross validation, respectively.



a



b



c

Figure 6: Respective performance graphs of the accuracy in Brahmi words classification. (a) best validation performance with 13 hidden neurons; (b) best validation performance with 11 hidden neurons; (c) best validation performance with 3 hidden neurons.

Apart from the upper discussed approach to avoid the over-fitting approach, dropout is also useful to avoid this issue. This study is used dropout method along with CNN and achieved 92.47% accuracy where the sets the outputs of hidden layer neurons to zero for a probability of less than or equal to a present value, 0.5, for instance.

This study also compared the deep learning method (CNN+ Gabor filter, CNN+ Gabor + Dropout) with other studies that used zonal density with ANN [11], and Gabor filter+ zonal structural features [20].

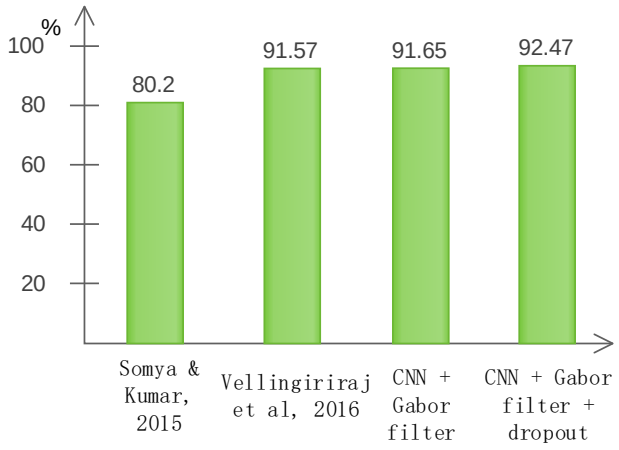


Figure 7: Comparison of performance of unsupervised techniques with previous studies

The achieved accuracy was 91.65% and 92.47% for the CNN with Gabor filter, and the combination of CNN with dropout and Gabor filter, respectively. In contrast, the achieved accuracy using the zonal density with ANN [11], and Gabor filter + zonal structural features [20] was 80.20% and 91.57%, respectively. Hence, CNN+ Gabor + Dropout had the highest accuracy compared to the previous studies (Figure 7).

Overall, it was observed that our proposed model of CNN showed better predictive accuracy compared with other classification models.

6. Conclusion

This study focused on the DCNN (deep convolutional neural network) to recognize the Brahmi word. While performing experiments on the standard dataset using DCNN, this study compared the results of different approaches in order to propose recommendations based on parameter tuning. Furthermore, there is a lack of standard data resource in the Brahmi domain in order to generate benchmark results.

In the field of machine learning, DCNNs come with a revolutionary change by providing quite efficient results in comparison with conventional approaches. However, there are also some inherent questionable issues like there is a lack of knowledge of how to determine the number of levels and hidden neurons in each layer. Furthermore, a standard dataset is required to check the validity and efficiency of deep network models. Therefore, in the experiment, this study had to train the DCNN by using samples of standard dataset [38]. In addition, finding a set of optimal parameters to generate error-free results is also a research issue. Similarly, some complex future tasks like character recognition of rotated, mirror-text, and noisy images by extracting novel features could benefit.

According to Table 8, the proposed system was significantly better than the approaches used in the related literature regarding to the number of parameters and the amount of calculation and the proposed system was quite efficient (in terms of accuracy) and effective at performing the recognition and classification since it provided better accuracy as compared with the others.

References

[1] N. Gautam and S. S. Chai, "Zig-zag diagonal and ANN for English character recognition," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 1.4, pp. 57-62, 2019.

[2] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar, "Towards Accurate Handwritten Word Recognition for Hindi and Bangla," in *Computer Vision, Pattern Recognition, Image Processing, and Graphics: 6th National Conference, NCVPRIPG 2017, Mandi, India, December 16-19, 2017, Revised Selected Papers 6*, 2018: Springer, pp. 470-480.

[3] N. Bhattacharya, P. P. Roy, U. Pal, and S. K. Setua, "Online Bangla Handwritten Word Recognition," *Malaysian Journal of Computer Science*, vol. 31, no. 4, pp. 300-310, 2018.

[4] N. Gautam, R. S. Sharma, and G. Hazrati, "Handwriting Recognition of Brahmi Script (an Artefact): Base of PALI Language," in *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems*, vol. 2: Springer, Cham, 2016, pp. 519-527.

[5] N. Gautam, R. Sharma, and G. Hazrati, "Akkhara-Muni: An instance for classifying PALI characters," in *International Conference on Computational Intelligence and Communication Networks*, Jabalpur, India, 2015: IEEE, pp. 252-253.

[6] N. Gautam, R. Sharma, and G. Hazrati, "Ariyaka: A PALI Alphabet Recognition Script," in *International Conference on Computational Intelligence and Communication Networks*, Jabalpur, India, 2015: IEEE, pp. 293-295.

[7] N. Gautam, R. Sharma, and G. Hazrati, "Eastern Arabic Numerals: A Stand out from Other Jargons," in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, 2015: IEEE, pp. 337-338.

[8] E. R. Acharya, "Evidences of Hierarchy of Brahmi Numeral System," *Journal of the Institute of Engineering*, vol. 14, no. 1, pp. 136-142, 2018.

[9] N. Gautam and S. S. Chai, "Optical Character Recognition for Brahmi Script Using Geometric Method," *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, no. 3-11, pp. 131-136, 2017.

[10] J. Igunma, *Buddhism Illuminated: Manuscript Art from South-East Asia*. University of Washington Press, 2018.

[11] E. Vellingiriraj, M. Balamurugan, and P. Balasubramanie, "Text Analysis and Information Retrieval of Historical Tamil Ancient Documents Using Machine Translation in Image Zoning," *International Journal ofLanguages Literature and Linguistics*, vol. 2, no. 4, pp. 164-168, 2016, doi: 10.18178/IJLLL.2016.2.4.88.

[12] E. H. Seland, "Archaeology of trade in the western Indian Ocean, 300 BC–AD 700," *Journal of Archaeological Research*, vol. 22, no. 4, pp. 367-402, 2014.

[13] U. Singh, *A History of Ancient and Early Medieval India: From the Stone Age to the 12th Century*. India: Pearson Education, 2008.

[14] J. R. Subba, *Evaluation Of Man And The Modern Society In Sikkim*. Gyan Publishing House, 2008.

[15] R. Salomon, *Indian epigraphy: A guide to the study of inscriptions in Sanskrit, Prakrit, and the other Indo-Aryan languages*. Oxford University Press, 1998.

[16] J. Keay, *India: A History*. Paw Prints, 2008.

[17] U. Pal and B. Chaudhuri, "Indian script character recognition: a survey," *pattern Recognition*, vol. 37, no. 9, pp. 1887-1899, 2004, doi: 10.1016/j.patcog.2004.02.003.

[18] G. Siromoney, R. Chandrasekaran, and M. Chandrasekaran, "Machine recognition of Brahmi script," *IEEE Transactions on Systems Man and Cybernetics*, vol. SMC-13, no. 4, pp. 648-654, 1983, doi: 10.1109/TSMC.1983.6313155.

[19] A. Roy and M. Mandal, *Brahmi: Rediscovering the Lost Script*. 2016.

[20] A. Soumya and G. H. Kumar, "Recognition of Historical Records using Gabor and Zonal Features," *Signal and Image Processing: An International Journal*, vol. 6, no. 4, pp. 57-69, 2015, doi: 10.5121/sipij.2015.6405.

[21] N. Gautam, S. S. Chai, and M. Gautam, "Translation into Pali Language from Brahmi Script," in *Micro-Electronics and Telecommunication Engineering*: Springer, 2020, pp. 117-124.

[22] C. Adak, B. B. Chaudhuri, and M. Blumenstein, "Offline cursive Bengali word recognition using CNNs with a recurrent model," in *15th International Conference on Frontiers in Handwriting Recognition*, Shenzhen, China, 2016: IEEE, pp. 429-434.

[23] I. J. Kim and X. Xie, "Handwritten Hangul recognition using deep convolutional neural networks," *International Journal on Document Analysis and Recognition*, vol. 18, no. 1, pp. 1-13, 2015, doi: 10.1007/s10032-014-0229-4.

[24] M. M. Rahman, M. Akhand, S. Islam, P. C. Shill, and M. H. Rahman, "Bangla handwritten character recognition using convolutional neural network," *International Journal of Image, Graphics and Signal Processing*, vol. 7, no. 8, pp. 42-49, 2015, doi: 10.5815/ijigsp.201 5.08.05.

[25] H. Cecotti and S. Vajda, "A radial neural convolutional layer for multi-oriented character recognition," in *12th International Conference on Document Analysis and Recognition*, Washington, DC, USA, 2013: IEEE, pp. 668-672, doi: 10.1109/ICDAR.2013.137.

[26] D. S. Maitra, U. Bhattacharya, and S. K. Parui, "CNN based common approach to handwritten character recognition of multiple scripts,"

presented at the 13th International Conference on Document Analysis and Recognition, Tunis, Tunisia, 2015.

[27] P. P. Nair, A. James, and C. Saravanan, "Malayalam handwritten character recognition using convolutional neural network," in *International Conference on Inventive Communication and Computational Technologies*, Coimbatore, India, 2017: IEEE, pp. 278-281, doi: 10.1109/ICICCT.2017.7975203.

[28] 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, doi: 10.1016/j.patrec.2017.03.004.

[29] S. Acharya, A. K. Pant, and P. K. Gyawali, "Deep learning based large scale handwritten Devanagari character recognition," presented at the 9th International Conference on Software, Knowledge, Information Management and Applications, Kathmandu, Nepal 2015.

[30] K. Mehrotra, S. Jetley, A. Deshmukh, and S. Belhe, "Unconstrained handwritten Devanagari character recognition using convolutional neural networks," in *4th International Workshop on Multilingual OCR*, Washington, D.C., USA, 2013: ACM, p. 15, doi: 10.1145/2505377.2505386.

[31] P. Singh, A. Verma, and N. S. Chaudhari, "Deep convolutional neural network classifier for handwritten Devanagari character recognition," in *Information Systems Design and Intelligent Applications*. New Delhi, India: Springer, 2016, pp. 551-561.

[32] M. Jangid and S. Srivastava, "Handwritten Devanagari Character Recognition Using Layer-Wise Training of Deep Convolutional Neural Networks and Adaptive Gradient Methods," *Journal of Imaging*, vol. 4, no. 2, pp. 1-14, 2018, doi: 10.3390/jimaging4020041.

[33] S. T. Soman, A. Nandigam, and V. S. Chakravarthy, "An efficient multiclassifier system based on convolutional neural network for offline handwritten Telugu character recognition," in *National Conference on Communications*, New Delhi, India, 2013: IEEE, pp. 1-5, doi: 10.1109/NCC.2013.6488008.

[34] P. Vijayaraghavan and M. Sra, "Handwritten Tamil recognition using a convolutional neural network," in *MIT Media Lab*, 2014, doi: 10.1109/PACC.2011.5978989.

[35] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Neural Networks*, vol. 71, pp. 1-10, 2015, doi: 10.1016/j.neunet.2015.07.007.

[36] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123-140, 1996, doi: 10.1023/A:1018054314350.

[37] M. Z. Alom, P. Sidike, T. M. Taha, and V. K. Asari, "Handwritten Bangla digit recognition using deep learning," presented at the arXiv preprint arXiv:1705.02680, 2017.

[38] N. Gautam, S. S. Chai, and M. Gautam, "The Dataset for Printed Brahmi Word Recognition," in *Micro-Electronics and Telecommunication Engineering, Lecture Notes in Networks and Systems*, vol. 106: Springer, Singapore, 2020, pp. 125-133.

[39] K. Fukushima, "Neural network model for a mechanism of pattern recognition unaffected by shift in position-Neocognitron," *Biological cybernetics*, vol. 36, no. 4, pp. 193-202, 1980.

[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the Institute of Electrical and Electronics Engineers*, vol. 86, no. 11, pp. 2278-2324, 1998, doi: 10.1109/5.726791.

[41] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[42] M. Husnain *et al.*, "Recognition of Urdu Handwritten Characters Using Convolutional Neural Network," *Applied Sciences*, vol. 9, no. 13, p. 2758, 2019.