Big Data and Information Analytics

http://www.aimspress.com/journal/BDIA

BDIA, 6: 41–55.
DOI:10.3934/bdia.2021004
Received: 18 February 2021
Accepted: 09 June 2021
Published: 16 June 2021

Research article

A convolution neural network with encoder-decoder applied to the study of Bengali letters classification

Sayed Mohsin Reza[1,*], **Md Al Masum Bhuiyan**[2] **and Nishat Tasnim**[3]

[1] Department of Computer Science, University of Texas at El Paso, Texas, USA

[2] Department of Mathematics & Statistics, Austin Peay State University, USA

[3] Department of Computer Science and Engineering, Daffodil International University, Bangladesh

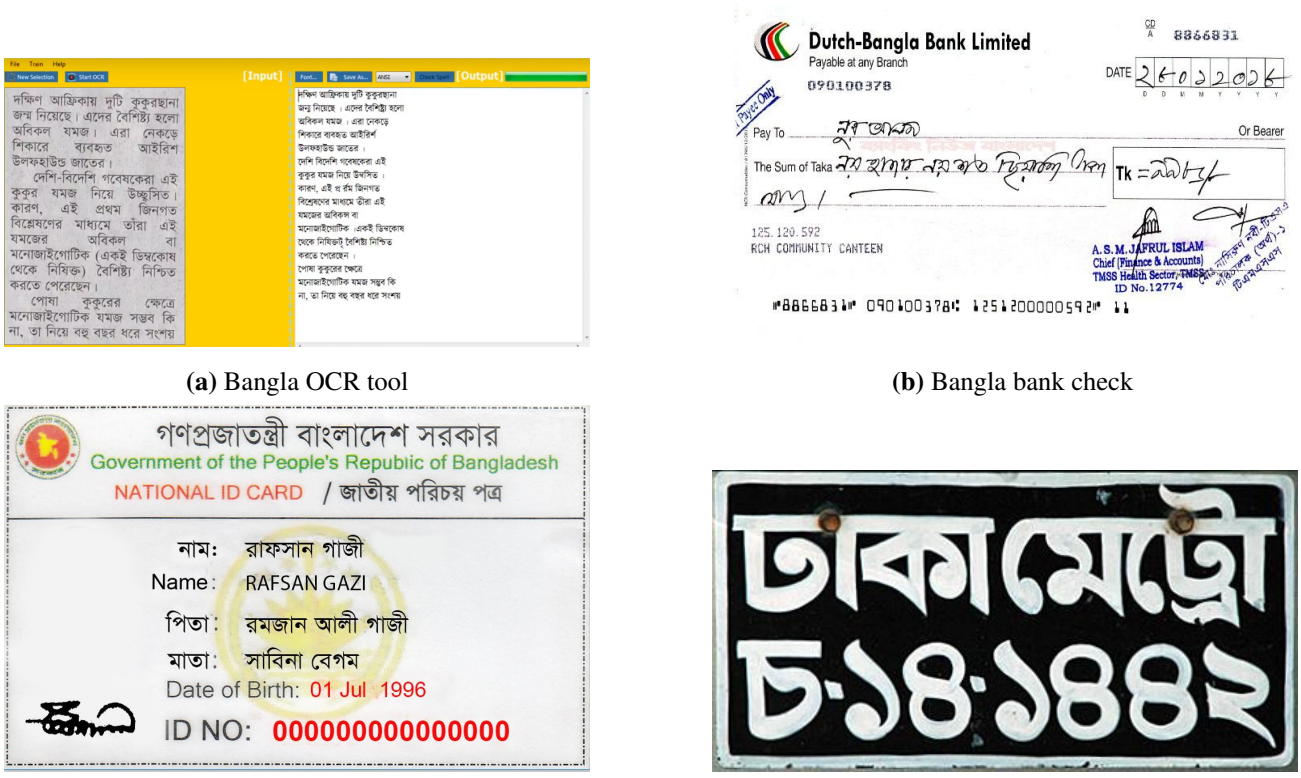
* **Correspondence:** Email: sreza3@miners.utep.edu; Tel:+17202727236.

Abstract: Handwritten grapheme recognition is popular research in computer vision and now widespread in the commercial industry due to its large number of applications in document analysis and recognition. Bengali grapheme classification is a complex task as it has 49 letters and 18 potential diacritics with almost 13,000 possible variations. Bengali is now the fifth most spoken native language and the seventh most spoken language by the total number of speakers in the world. Having set a bigger scope, this paper deals with the recognition of Bengali handwritten script letters. A class of deep Convolutional Neural Networks (CNNs) with encoder-decoder is used to classify handwritten letters. We use several serial non-linear layers as the encoders and a corresponding set of decoders that work as a pixel-wise classifier for letter recognition. The key idea is to encode images by convolution and decode them by deconvolution so that the max-pooling and up-sampling layers can correctly identify grapheme pixel-by-pixel. In this study, almost 200,840 grapheme images were analyzed, including root, vowels, and consonants. A large number of variations make additional complexity in recognition and may lead the model into over-fitting or under-fitting. We introduce regularization techniques to reduce the over-fitting in the fully connected layers. The results suggest that CNN with encoder-decoder can recognize complex grapheme characters with higher precision than traditional CNN. Experimental results show that the images' augmentation helps the model train better and improves its accuracy and loss.

Keywords: Bengali grapheme; deep Learning; convolution neural network (CNN); document analysis

1. Introduction

Printed script letter classification has a tremendous commercial and pedagogical importance for book publishers, online Optical Character Recognition (OCR) tools, bank officers, postal officers, video makers, and so on [1–3]. Postal mail sorting according to zip code, the verification of signatures, and check processing are usually done with the application of grapheme classification [4–6]. Some sample images of its application are shown in Figure 1.



(a) Bangla OCR tool

(b) Bangla bank check

(c) Bangla national ID card

(d) Bangla vehicle registration plate

Figure 1. Example application areas of Bangla handwritten letter classification in real-life.

Statistically, the importance of printed script letter classification is enormous when a large population uses a specific language. For example, with nearly 230 million native speakers, Bengali (also called Bangla) ranks as the fifth most spoken language in the world [7]. It is the official language of Bangladesh and the second most spoken language in India [8] after Hindi.

Handwritten character classification or recognition is particularly challenging for the Bengali language, as it has 49 letters and 18 potential diacritics (or accents). Moreover, this language supports complex letter structures created from its basic letters and diacritics. In total, the Bangla letter variations are estimated to be around 13,000, which is 52 times more than the English letter variations [9]. Although several language grapheme classifications have received much attention [10–12], Bangla is still a relatively unexplored field, with most works done in the detection of vowels and consonants. The limited progress in exploring the Bengali language has motivated us to classify the Bengali handwritten letters into three constituent elements: root, vowel, and constant.

Previously, several machine learning models have been used for different language grapheme recognition [13]. Several research using the deep convolutional neural network has been successful in the detection of handwritten characters in Latin, Chinese, and English language [14, 15]. In other words, successful feature extraction became possible using different types of layers in neural networks. The convolutional neural network with the augmentation of the images of handwriting can generate a better model through training in deep learning [16, 17]. Previous research in this area faced fewer challenges regarding variations, massive data usage, and model creation that deep learning needs most regarding the high number of label classification [19, 20].

This paper proposes a deep convolutional neural network with an encoder-decoder to facilitate the accurate classification of Bangla handwritten letters from images. We use 200,840 handwritten images to train and test the proposed deep learning model. We train in 4 steps with four subsets of images containing 50,210 images each. In doing so, we create three different labels (roots, vowels, and consonants) from each handwritten image. The result shows that the proposed model can classify handwritten Bangla letters as follows: roots-93%, vowels-96%, and consonants-97%, which is much better than previous works done on Bangla grapheme variations and dataset size.

The paper is organized as follows: Section 2 discusses the brief background of existing research on this research area. Section 3 is devoted to present Bangla handwritten letter dataset details. This section discusses the dataset structure and format and augments the dataset to create many variations. Section 4 discusses the architecture of the models for Bangla handwritten grapheme classification, and in section 5, we discuss the experimental process, tools, and used programming language. Section 6 shows the detailed results of the training process and validation. Finally, section 7 concludes the research work by discussing the contributions and applicability in the classification of Bangla handwritten letters.

2. Related work

Optical Character Recognition is one of the favorite research topics in computer graphics and linguistic research. This section briefly discusses two areas first, how deep learning is used in character recognition, second, how deep learning is used in Bangla handwritten digit and letter classification.

In optical character recognition, many research works have been proposed. Théodore et al. showed that learning features with CNN is much better for handwritten word recognition [21]. However, the model needs a longer processing time when classifying a word compared to a letter. Zhuravlev et al. mentioned this issue in their research and experimented with a differential classification technique on grapheme images using multiple neural networks [22, 23]. However, the model works better with that small dataset and will fail to detect when augmented images will be provided. Jiayue et al. discussed this issue and proved that proper augmentation before feeding into CNN could be more efficient in grapheme classification [24].

Regarding Bangla character and digit recognition, there are few kinds of research available. A study by Shopon et al. presented unsupervised learning using an auto-encoder with deep ConvNet to recognize Bangla digits [25]. A similar study by Akhand et al. proposed a simple CNN for Bangla handwritten numeral recognition [26, 27]. These methods achieved 99% accuracy in detecting Bangla digits and faced fewer challenges in classifying only ten labels than more character recognition labels.

A study on Bangla character recognition by Rabby et al. used CMATERdb and BanglaLekha as datasets and CNN to recognize handwritten characters. The resulted accuracy of CNN was 98%, and

95.71% for each dataset [28]. Another study by Alif et al. showed ResNet-18 architecture is giving similar accuracy on CMATERdb3, which is a relatively large dataset than previous [29, 30]. The limitation of these two research is their image variations and dataset size.

This paper addresses the limitations of previous research in image augmentation, dataset size, proper model creation, and a high number of label classification.

3. Dataset preparation

This section describes raw data and data augmentation to ensure better data preparation for our proposed model.

3.1. Data background

This study uses the dataset from the Bengali.AI community that works as an open-source dataset for research. The dataset was prepared from handwritten Bangla letters by a group of participants. The images of these letters are provided in parquet format. Each image is 137×236 pixels. Some sample handwritten images are shown in Figure 2.

Figure 2. Sample handwritten Bangla grapheme images.

The Bengali language has 49 letters with 11 vowels, 38 consonants, and 18 potential diacritics. The handwritten Bengali characters consist of three components: *grapheme-root*, *vowel-diacritic*, and

consonant-diacritic. To simplify the data for ML, we organize 168 *grapheme-root*, 10 *vowel-diacritic*, and 6 *consonant-diacritic* as unique labels. All the labels are then introduced as a unique integer. Table 1 summarizes the metadata information of the training dataset.

Table 1. Metadata information of training dataset.

Features	Description
image_id	Sample ID number for each handwritten image
grapheme_root	Unique number of vowels, consonants, or conjuncts
vowel_diacritic	Nasalization of vowels, and suppression of the inherent vowels
consonant_diacritic	Nasalization of consonants, and suppression of the inherent consonants
grapheme	Target variable

The raw training dataset has a total of 200,840 observations with almost 10,000 possible handwritten image variations. The raw testing dataset is created separately to distinguish it from the training dataset. Table 2 summarizes the metadata information of the test data.

Table 2. Metadata information of the testing dataset.

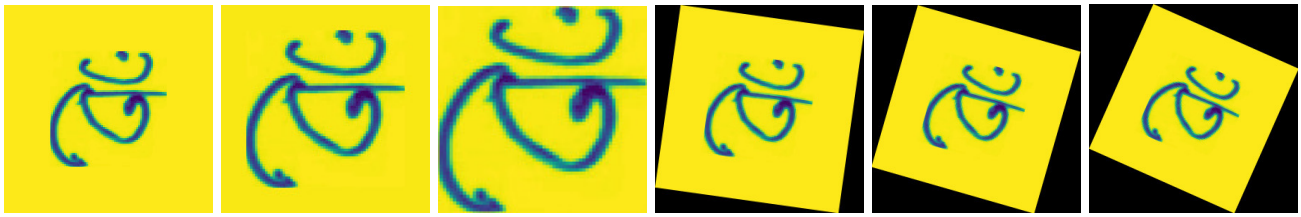
Features	Description
image_id	An unique image ID for each testing image
row_id	Test id of grapheme root, consonant diacritic, vowel diacritic
component	Grapheme root, consonant diacritic, vowel diacritic

3.2. Dataset augmentation

The raw dataset is a set of images in the parquet format, as discussed in the previous section. The images are generally created from a possible set of grapheme writing, but it does not cover all the aspects of writing variations. To create more variations (more than 10,000), dataset augmentation becomes a necessary step. In reality, it has around 13,000 possible letter variations that make the problem harder than any other language grapheme classification. Therefore, a pre-processing of the dataset is done to increase the more number of grapheme variations.

We apply the following data augmentation techniques: (1) shifting, (2) rotating, (3) changing brightness, and (4) applying zoom. In all cases, some precautions are taken so that augmented handwritten images are well generated. For example, too much shifting or too much brightness can lose the image pixels [31]. Applying random values to those operations is also prohibited during our pre-processing of the dataset.

In terms of shifting, we apply the following image augmentation: *width shift*, and *height shift* on our images. In rotation, an image was rotated to (1) 8 degree, (2) 16 degree, and (3) 24 degree in both positive and negative direction. In case of zoom, we apply (1) 0.15%, and (2) 0.3% zoom in. Some sample output of Bangla handwritten letter's augmentation is shown in Figure 3.



(a) Raw handwritten image **(b)** Zoom image (0.15%) **(c)** Zoom image (0.3%) **(d)** Rotated image (8 degree) **(e)** Rotated image (16 degree) **(f)** Rotated image (24 degree)

Figure 3. Bengali handwritten letter augmentation samples.

The augmenting was minimized to only four options due to minimizing the risk of false image creation. As our dataset is related to characters, we need to verify that augmentation may add false image augmentation. For example, there is a *horizontal flip* option for image augmentation. If we apply that to our dataset, some non-Bangla handwritten images and the proposed model may learn wrongly to classify the handwritten letters.

4. Methodology

This section describes the architecture of the models that we build for Bangla grapheme image classification. We also discuss the neural networks and their necessary layers useful for fitting data into the model.

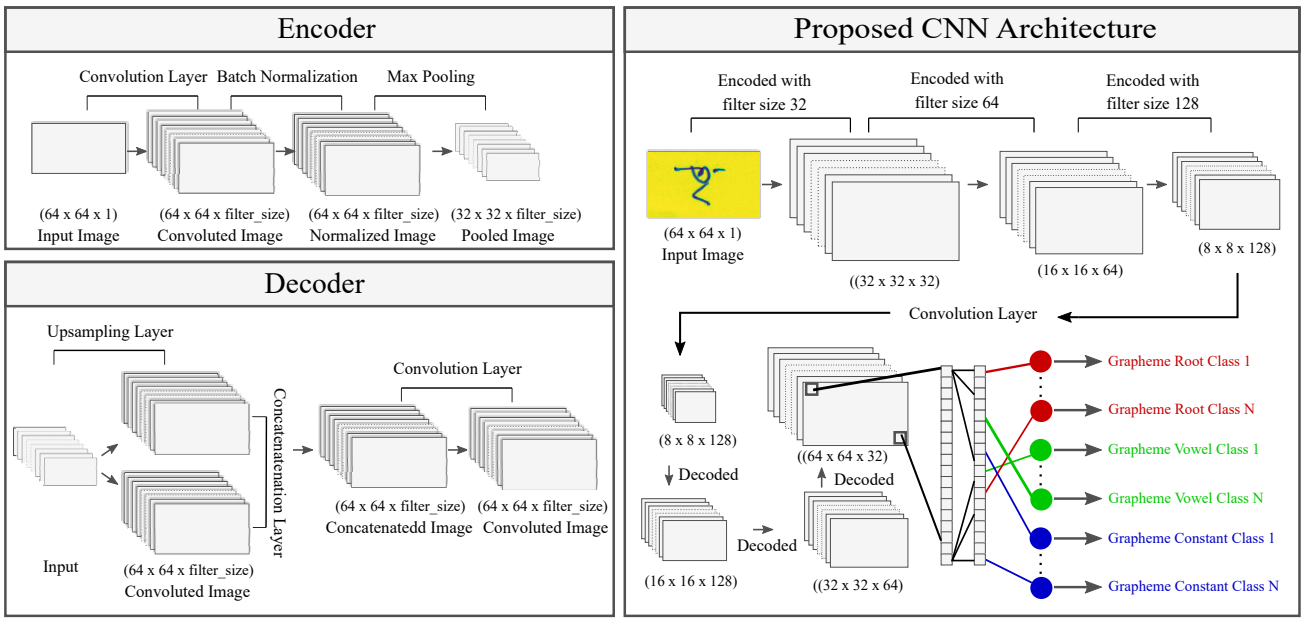


Figure 4. The proposed architecture of neural networks for Bangla grapheme classification.

4.1. Neural network

A neural network is a series of algorithms that process the data through several layers that mimic how the human brain operates. A neural network for any input x can be expressed as:

$$y = f(\sum_j w_j x_j + b) \tag{4.1}$$

where, w_j is the network weights, b is a bias term, and f is a specified activation function. For a better approximation, multiple neurons are used in the form of hidden layers as follows:

$$y_i = f(\sum_j w_{i,j} x_j + b_i) \tag{4.2}$$

To obtain the final output with higher accuracy, multiple hidden layers are used. The final output can be obtained as:

$$z = f(\sum_k w_k^{(z)} y_k + b^{(z)}) \tag{4.3}$$

For image processing, we flat the image and feed it through neural networks. A vital characteristic of the images is that images have high dimensional vectors and take many parameters to characterize the network. To solve this issue, convolutional neural networks were used to reduce the number of parameters and adapt the network architecture specifically to vision tasks. CNN's are usually composed of a set of layers that can be grouped by their functionalities.

In this study, we implemented the CNN architecture with an encoder-decoder system. The encoder-decoder classifier works at the pixel level to extract the features. A recent work by Jong et al. shows that how encode-decoder networks using convolutional neural networks work as an optimizer for complex image data sets [33]. The encoder and decoder are developed using convolution, activation, batch normalization, and pooling layers. The detailed picture of these layers is described in the following section and shown in Figure 4.

4.2. Convolution layers

The convolution layer extracts feature maps from the input or previous layer. Each output layer is connected to some nodes from the previous layer [34,35] and valuable in the classification process. The convolutional layer is used to the sliding window through the image and convolves to create several sub-image, increasing the volume in terms of depth. The implementation view of the convolutional layer exists in both encoder and decoder, shown in Figure 4.

4.3. Activation layers

This section describes the activation function used in neural networks for proposed model training. We include the rectified Linear Unit (ReLU) function to add the non-linearity in the proposed network. The function is defined as

$$f(x) = \max(0, x) \tag{4.4}$$

where it returns 0 for negative input and x for positive input.

There is some other nonlinearity function named as *sigmoid* and *tanh*. The *sigmoid* non-linearity function can also be declared as follows

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.5}$$

which maps a real number x to a value between [0, 1]. Another nonlinear function *tanh* maps the values into the interval [−1, 1].

As the Bangla grapheme list contains more labels to detect than other language graphemes, we implement several non-linear layers as an encoder and a corresponding set of decoders to improve the recognition performance. However, we use ReLu due to its linear form, and this function improves the performance of classification compared to *sigmoid*, and *tanh* functions [36, 37]. An essential feature of such ReLu function is that it shows non-linearity around 0 because its slope is not constant around 0. Therefore, when the CNN model includes bias terms, the nodes can change the slope at different values for our input using the ReLu function [37].

In our model, we introduce such ReLu functionality after each non-pooling layer in the encoder to map the value of each neuron to an actual number. However, in some cases, such function can die during training. To solve this issue, leaky Relu has been added so that if there is any negative value, it will add a slight negative slope [38].

4.4. Normalization & dropout layers

The batch normalization layer normalizes each input channel across a mini-batch [39, 40]. Our study adjusts and scales the previously filtered sub-images and normalizes the output by subtracting the batch mean and diving by the standard deviation of the batch. Then, it shifts the image input by a learnable offset. Generally, using such a layer after a convolution layer and before non-linear layers is useful in speeding up the training and reducing sensitivity to network initialization [39].

We also implement a dropout layer during the final classification step. We use this layer to reduce the labels by setting zero, which has less probability in classification [41, 42].

4.5. Pooling layers

The pooling layer is mainly used to reduce the resolution of the feature maps. To be specific, this layer downsamples the volume along the image dimensions to reduce the size of representation and number of parameters to prevent overfitting. Our model uses a max-pooling layer in each encoder block, which downsamples the volume by taking max from each block.

5. Experiment

The proposed model is trained with a configuration of the epoch, loss function, and batch size in the experiment. Also, the model contains trainable and non-trainable parameters. The trainable parameter size for the proposed model is 136,048,154, whereas the non-trainable parameter size is 448. There are thirteen convolution layers, three pooling layers, three normalization layers, and four dropout layers are used in our model. The experiment is implemented using the Python programming language with Keras [43], and Theano [44] library.

In terms of configuration, the proposed model uses 25 epochs with a batch training size of 128. To calculate the loss function, we use categorical cross-entropy for root, vowel, and constant classification. Mean Squared Error (MSE) is another metric to calculate loss function but categorical cross-entropy performance is better in classification tasks [45].

After developing the model in Python, we run it on an Intel (R) Core (TM) i7-7500U CPU @ 2.70 GHz machine with 16GB RAM. For both validation and training, the same batch size and epochs were used. The experimental results performance is calculated using accuracy and model performance evaluation metric. We also use a loss function to evaluate how well our deep learning model trains the given data. Both of these metrics are popular in classification tasks using deep learning [46–48].

6. Results & discussion

In this section, we present the outcome of the model in terms of evaluation metrics. The proposed CNN method is applied in four different phases on four subsets of the grapheme dataset and produces the results. This way, we test how more Bangla handwritten letter images are helpful to produce better deep neural network models. However, conducting this research with more subsets of images will have computational and complexity challenges. In evaluation, the accuracy and loss of each epoch of training and validation are used.

6.1. Phase 1 results

In the first phase, a subset of 50,210 images is sent to training with 30 epoch. The results show that accuracy in detecting the root is less than the vowel and constant in both training and validation. The training and validation root accuracy are 85% and 88% respectively, where vowel accuracy is 92% and 95%, constant accuracy is 95% and 96%. This is because many root characters are needed to identify than the number of vowels, and constants are needed to identify. Figures 5 and 6 show the train and validation accuracy and cross-entropy loss over epochs. The results show that the model seems to have good converged behavior. It implies the model is well configured and no sign of over or underfitting.

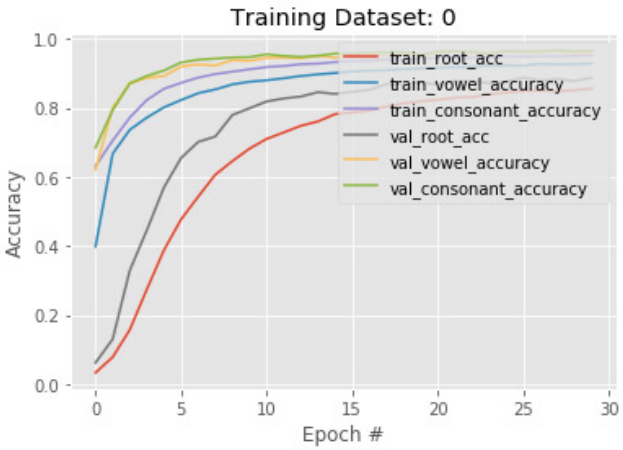


Figure 5. Accuracy of proposed model (Phase 1).

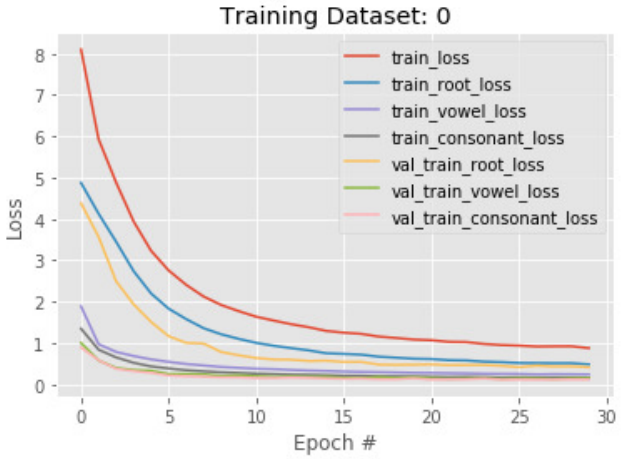


Figure 6. Loss of proposed model (Phase 1).

We also test how a CNN model with an encoder-decoder is compared to a traditional CNN that does

not have an encoder and decoder concept. There are six convolution layers, three pooling layers, five normalization layers, and four dropout layers are used in the traditional CNN model. Figures 7 and 8 visualize the accuracy and loss of the simple CNN model for 30 epochs. The results we see from the figure that the loss and accuracy are fluctuating. The reason behind this, the simple CNN model is sensitive to noise and produces random classification results. This problem is also known as overfitting. The results show a better performance with an encoder and decoder concept than a traditional one.

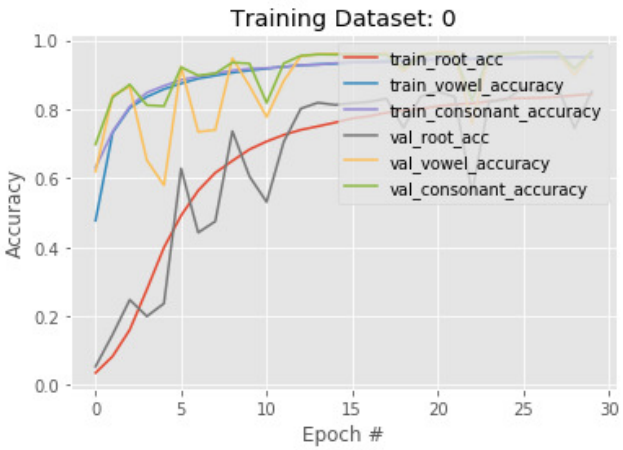


Figure 7. Accuracy of traditional CNN model.

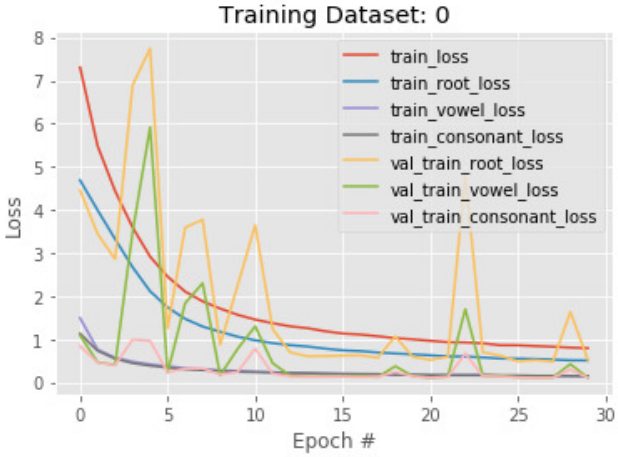


Figure 8. Loss of traditional CNN model.

6.2. Phase 2 results

After getting a training accuracy of 85% in the first phase, we train another subset of images with the existing model. The hypothesis is that the more variations of images are trained, the more the model is learning when we have many root variations. We take a different 50,210 images and train with the existing model with 30 epochs in this phase. Figures 9 and 10 visualize the accuracy and loss of the model in phase 2 respectively.

At the beginning of the training stage, we observe that train root accuracy drops from 85% to 80%. Not only train root accuracy, but all other categories' accuracy also drops after adding a new subset of images. The opposite behavior is observed in terms of the loss function. In epoch 0, loss functions of every category are increased. However, the final result of training 2 ends up with better train root, vowel, and constant accuracy of 92%, 95%, and 96%, respectively. In terms of the loss function, we observe the decrements over epochs in every case. These results imply that the model is appropriately converged and trained well due to more subset images used in the training process.

6.3. Phase 3 results

As a good result is maintained in our previous phases, we introduce another set of images with more variations in learning by our model. However, this time we observe little changes happen after the training. Figures 11 and 12 show the accuracy and loss of the model in phase 3. After another 30 epochs, training root, vowel, and constant accuracy are 94%, 96%, and 97%. The model root accuracy is increased by 2% and vowel, and constant accuracy are increased by 1%. The same behavior is

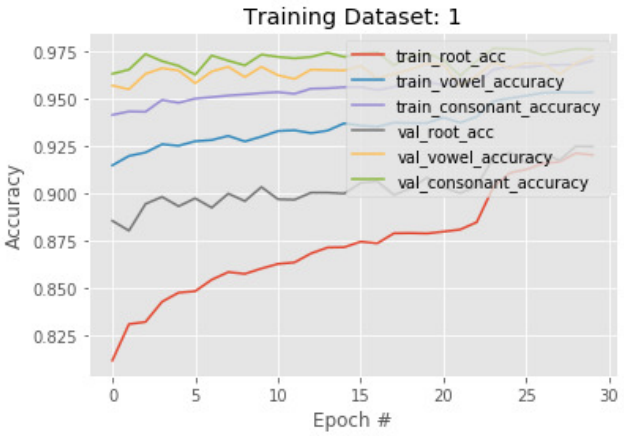


Figure 9. Accuracy of proposed model (Phase 2).

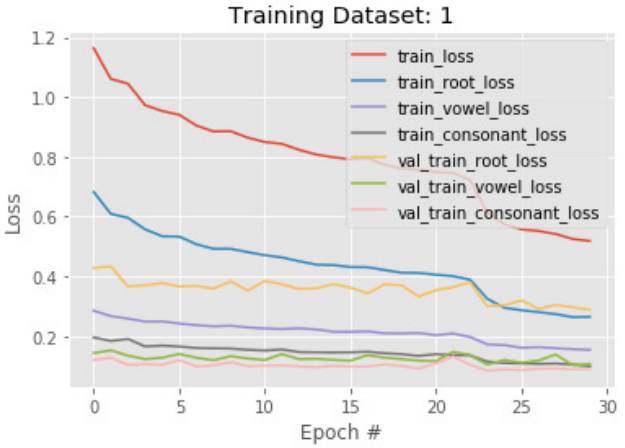


Figure 10. Loss of proposed model (Phase 2).

found on the validation data also. It implies the model has converged very well and can be finalized by another training.

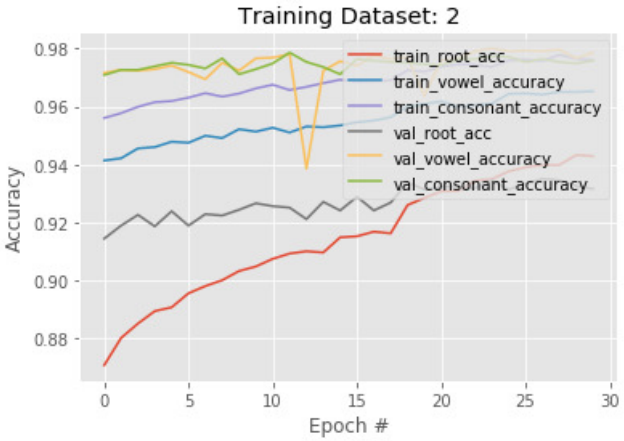


Figure 11. Accuracy of proposed model (Phase 3).

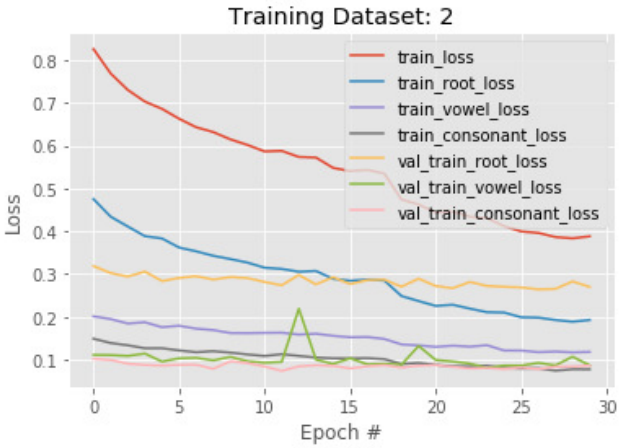


Figure 12. Loss of proposed model (Phase 3).

6.4. Final phase results

This final phase verifies the converge of accuracy and loss function by just doing another final training. Another set of 50,210 images of Bangla handwritten letters are introduced. Figures 13 and 14 show the accuracy and loss function of the model in the final phase. The results show that root accuracy drops from 94% to 93% in the training stage, and the accuracy drops from 98% to 97% in the validation stage. Nevertheless, in all other cases, it seems improvement or no change. Also, the results start to create bumpy behavior in the accuracy metric, and loss functions are also converged. All the validation loss functions are 3% or below. These all results imply, our final model is converged and ready to report the final accuracy and loss.

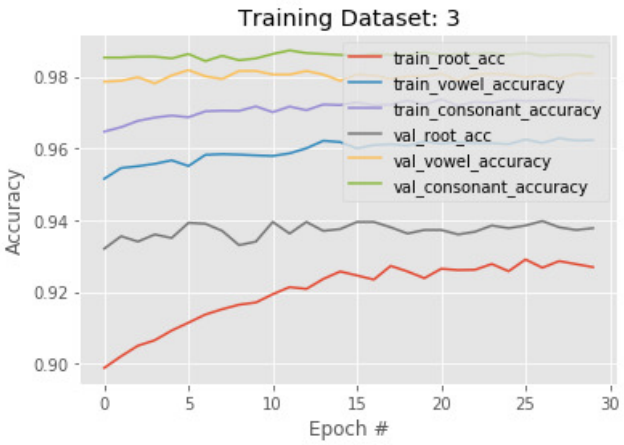


Figure 13. Accuracy of proposed model (Final Phase).

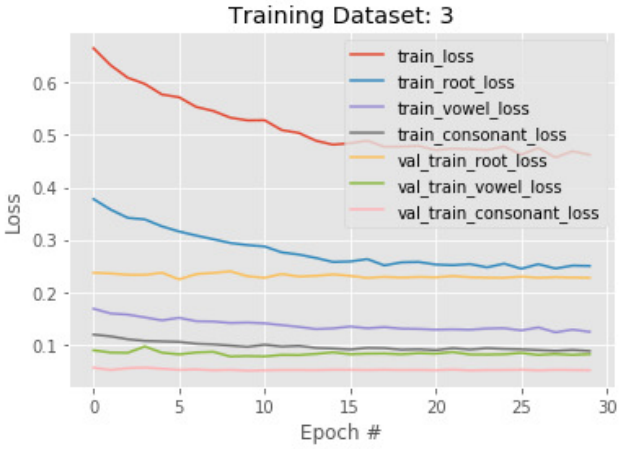


Figure 14. Loss of proposed model (Final Phase).

7. Conclusions

Despite the advances in the classification of grapheme images in computer vision, Bengali grapheme classification has mainly remained unsolved due to confusing characters and many variations. Moreover, Bangla is one of the most spoken languages in Asia, and its grapheme classification has not been made, as there is no application as yet. However, many industries like banks, post offices, book publishers, and many more industries need the Bangla handwritten letters recognition.

In this paper, we implement a CNN architecture with encoder-decoder, classifying 168 grapheme roots, 11 vowel diacritics (or accents), and 7 consonant diacritics from handwritten Bangla letters. One of the challenges is to deal with 13,000 grapheme variations, which are way more than English or Arabic grapheme variations. The performance results show that the proposed model achieves root accuracy of 93%, vowel accuracy of 96%, and consonant accuracy of 97%, which are significantly better in Bangla grapheme classification than in previous research. Finally, we report the detailed loss and accuracy in 4 phases of training and validation to show how our proposed model learns over time.

To illustrate the model performance, we compared our model with a traditional CNN applied to the same dataset. The results show that the accuracy and loss function fluctuate over time in the traditional CNN model, which means an over-fitted model. In comparison, we see that the proposed CNN model with encoder-decoder does much better in classifying Bangla handwritten grapheme images.

References

1. Drobac S, Lindén K, (2020) Optical character recognition with neural networks and post-correction with finite state methods. *Int J Doc Anal Recognit* 23:279–295.
2. Rahman A F R, Fairhurst M C, (2003) Multiple classifier decision combination strategies for character recognition: A review. *Doc Anal Recognit* 5:166–194.
3. Paul M C, Sarkar S, Rahman M, Reza S M, Kaiser M S, (2016) Low cost and portable patient monitoring system for e-Health services in Bangladesh. *2016 IEEE Int Confer Computer Communication and Informatics* :1–4.

4. Matan O, Baird H S, Bromley J, et al. (1992) Reading handwritten digits: A zip code recognition system. *Computer* 25: 59–63.

5. Sazal M M R, Biswas S K, Amin M F, et al. (2014) Bangla handwritten character recognition using deep belief network. *2013 Int Confer Electron Inf Commun Technol*: 1–5.

6. Reza S M, Rahman M, Parvez M H, Kaiser M S, Al Mamun S, (2015) Innovative approach in web application effort & cost estimation using functional measurement type. *2015 IEEE Int Confer Elect Eng. Infor Commun. Technology* :1–7.

7. Karim M R, Chakravarthi B R, McCrae J P, et al. (2020) Classification benchmarks for under-resourced bengali language based on multichannel convolutional-lstm network. *2020 IEEE 7th Int Confer Data Sci Adv Anal*: 390–399.

8. M. G. Kibria, Imtiaz A, (2012) Bengali optical character recognition using self organizing map. *2012 Int Confer Inf Electron Vision*: 764–769.

9. Akhand M A H, Ahmed M, Rahman M, (2016) Convolutional neural network based handwritten bengali and bengali-english mixed numeral recognition. *Int J Image Graph Signal Proc* 8: 40.

10. Plamondon R, Srihari S N, (2000) Online and off-line handwriting recognition: a comprehensive survey. *IEEE Trans Pattern Anal Mach Intell* 22:63–84.

11. Wong P K, Chan C, (1998) Off-line handwritten chinese character recognition as a compound bayes decision problem. *IEEE Trans Pattern Anal Mach Intell* 20:1016–1023.

12. Rahman S, Sharma T, Reza S M, Rahman M M, Kaiser M S, (2016) PSO-NF based vertical handoff decision for ubiquitous heterogeneous wireless network (UHWN). *Int Workshop Computational Intelligence* :153–158.

13. Ashiquzzaman A, Tushar A K, (2017) Handwritten arabic numeral recognition using deep learning neural networks. *2017 IEEE Int Confer Imaging Vision Pattern Recognit*: 1–4.

14. Kim I J, Xie X H, (2015) Handwritten hangul recognition using deep convolutional neural networks. *Int J Doc Anal Recognit* 18: 1–13.

15. Cireşan D C, Meier U, Schmidhuber J, (2012) Transfer learning for latin and chinese characters with deep neural networks. *2012 Int Joint Confer Neural Networks*: 1–6.

16. Zhang J L, Guo M T, Fan J P, (2019) A novel cnn structure for fine-grained classification of chinese calligraphy styles. *Int J Doc Anal Recognit* 22: 177–188.

17. Reza S M, Rahman M M, Mamun S A, (2014) A new approach for road networks - a vehicle xml device collaboration with big data. *2014 IEEE Int Conf Elec Eng and Infor & Commun Tech* : 1–5.

18. Reza S M, Rahman M M, Mahmud M M, Mamun S A, (2014) A New Approach of Big Data collaboration for Road Traffic Networks considering Path Loss Analysis in context of Bangladesh. *JU Journal of Information Technology* 3: 1–5

19. Tagougui N, Kherallah M, Alimi A M, (2013) Online arabic handwriting recognition: a survey. *Int J Doc Anal Recognit* 16: 209–226.

20. Reza S M, Badreddin O, Rahad K, (2020) Modelmine: a tool to facilitate mining models from open source repositories. *2020 ACM/IEEE Conf Model Driven Eng Lang Sys* : 1–5.

21. Bluche T, Ney H, Kermorvant C, (2013) Feature extraction with convolutional neural networks for handwritten word recognition. *2013 12th Int Confer Doc Anal Recognit*: 285–289.
22. Alekseevich Z A, Rybkin V, Vladimirovich A K, (2020) Differential classification using multiple neural networks. U. S. Patent No. 10,565,478.
23. Transue S, Reza S M, Halbower A C, Choi M, (2018) Behavioral analysis of turbulent exhale flows. *2018 IEEE EMBS Int Conf Biome & Health Infor*: 42–45.
24. Yang J Y, (2020) Gridmask based data augmentation for bengali handwritten grapheme classification. *Proc 2020 2nd Int Confer Intell Med Image Proc*: 98–102.
25. Shopon M, Mohammed N, Abedin M A, (2016) Bangla handwritten digit recognition using autoencoder and deep convolutional neural network. *2016 Int Workshop Comput Intell*: 64–68.
26. Akhand M A H, Ahmed M, Rahman M M H, (2016) Convolutional neural network training with artificial pattern for bangla handwritten numeral recognition. *2016 5th Int Confer Inf Electron Vision*: 625–630.
27. Rahad K, Badreddin O, Reza S M, (2021) The human in model-driven engineering loop: A case study on integrating handwritten code in model-driven engineering repositories. *2021 Journ Softw Pract Exper* 51: 1308–1321.
28. Rabby A S A, Haque S, Islam M S, et al. (2018) Bornonet: Bangla handwritten characters recognition using convolutional neural network. *Proc Comput Sci* 143: 528–535.
29. Alif M A R, Ahmed S, Hasan M A, (2017) Isolated bangla handwritten character recognition with convolutional neural network. *2017 20th Int Confer Comput Inf Technol*: 1–6.
30. Rahad K, Badreddin O, Reza S M, (2021) Characterization of Software Design and Collaborative Modeling in Open Source Projects. *9th Int Conf Model-Driven Eng and Soft Dev*: 254–261.
31. Mikołajczyk A, Grochowski M, (2018) Data augmentation for improving deep learning in image classification problem. *2018 Int Interdiscip PhD Workshop*: 117–122.
32. Reza S M, Badreddin O, Rahad K, Mahmud S U, (2021) Software code quality and source code metrics dataset. *Mendeley Data* DOI: 10.17632/77p6rzb73n.
33. Ye J C, Sung W K, (2019) Understanding geometry of encoder-decoder cnns. *Int Confer Mach Learning*: 7064–7073.
34. Albawi S, Mohammed T A, Al-Zawi S, (2017) Understanding of a convolutional neural network. *2017 Int Confer Eng Technol*: 1–6.
35. Reza S M, Rahman M M, Parvez H, Badreddin O, Mamun S A, (2020) Performance Analysis of Machine Learning Approaches in Software Complexity Prediction. *Int Conf Trends in Computa and Cogni Eng*: 27–39.
36. Nwankpa C, Ijomah W, Gachagan A, et al. (2018) Activation functions: Comparison of trends in practice and research for deep learning. arXiv:1811.03378
37. Qiumei Z, Dan T, Fenghua W, (2019) Improved convolutional neural network based on fast exponentially linear unit activation function. *IEEE Access* 7: 151359–151367.
38. Singh P, Varshney M, Namboodiri V, (2020) Cooperative initialization based deep neural network training. *IEEE Winter Confer Appl Comput Vision*: 1141–1150.

39. Bjorck N, Gomes C P, Selman B, Weinberger K Q, (2018) Understanding batch normalization. *Adv Neural Inf Proc Syst*: 7694–7705.

40. Ioffe S, (2017) Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *Adv Neural Inf Proc Syst*: 1945–1953.

41. Helmbold D P, Long P M, (2017) Surprising properties of dropout in deep networks. *J Mach Learning Res* 18: 7284–7311.

42. Kubo Y, Tucker G, Wiesler S, (2016) Compacting neural network classifiers via dropout training. arXiv:1611.06148

43. Arora S, Bhatia M P S, (2018) Handwriting recognition using deep learning in keras. *2018 Int Confer Adv Comput Commun Control Networking*: 142–145.

44. Ma H, Mao F, Taylor G W, (2016) Theano-mpi: a theano-based distributed training framework. *Eur Confer Parallel Proc* : 800–813.

45. Rusiecki A,. Trimmed categorical cross-entropy for deep learning with label noise. *Electron Lett*, 55:319–320.

46. Tang D Y, Wei F R, Qin B, et al. (2014) Coooolll: A deep learning system for twitter sentiment classification. *Proc 8th Inte Workshop Semantic Eval*: 208–212.

47. Singh A, Principe J C, (2010) A loss function for classification based on a robust similarity metric. *2010 Int Joint Confer Neural Networks*: 1–6.

48. Cheng D, Gong Y H, Zhou S P, et al. (2016) Person re-identification by multi-channel parts-based cnn with improved triplet loss function. *Proc IEEE Confer Computer Vision Pattern Recognit*: 1335–1344.

AIMS Press