

Bengali Handwritten Character Recognition

using convolutional neural network

Final submission

Amit Kumar Biswas (011 171 117)

Mebin Rahman Fateha (011 171 124)

Abir Hossain Farabi (011 171 125)

Pattern Recognition Lab (CSI 416) | Section: A

05.06.2020

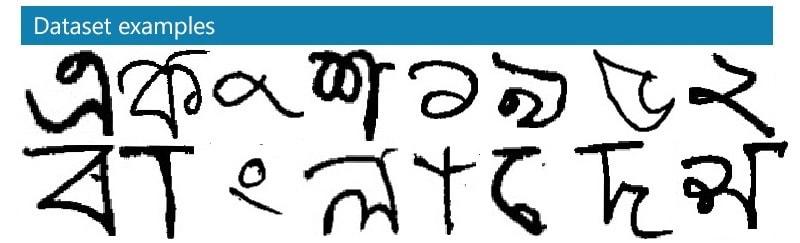
# Introduction

Handwritten character recognition is one of the most interesting issues in the present time due to its variant application and helps in making old forms and information digitalization reliable. It’s a big challenge to make a reliable model on this as every person has a different style of writing and the difference in shapes and sizes. Bengali is the 4th most popular language in the world. In the context of our mother tongue, this model helps to recognize Bengali handwritten 50 basic characters, 10 digits, 10 modifiers, and 52 mostly used compound characters. The proposed model is trained and validated with Ekush(1) dataset. It has shown satisfactory recognition accuracy of 93% for this dataset.

# Methodology

The proposed model used a Convolutional Neural Network to recognize handwritten characters. The steps are described below:

1. ***Dataset:*** The dataset that has been used here is the Ekush (1) Dataset. It has a total of 368,776 images where 155,570 alphabets, 151,607 compound characters, 30830 digits, and 30769 modifiers. Ekush(1) datasets image resolution depends on character size. Most of the images have padding with a black background while the characters are white. Some examples are,



1. ***Dataset Preparation:*** As the images are already in grayscale, it didn’t need any prep color-wise. But the different height and width have been generalized by resizing all of the images into 28 x 28 pixels. Then the images are stored in CSV files for better IO timing and turned into 785 columned rows where 784 columns represent the images and the last column is for the label. During training, the pixel values have been normalized using Minimax (2) Normalizer to compress the values of the pixels in a range of (0, 1). This helps CNN perform better. Lastly, all the data labels are converted into one hot encoding (3).
2. ***CNN Architecture:*** This model used a multilayer CNN for classifying the character which utilizes Convolutional layers(4), Max-pooling layers(5), Fully-connected dense layer(6), and some regularization methods like Batch Normalization(7) and Dropout(8). The architecture of the model is given below,

Layer (type) Output Shape Param# Connected to

===================================================================================

input\_2 (Input Layer) (None, 28, 28, 1) 0 conv2d\_8 (Conv2D) (None, 28, 28, 32) 320 input\_2[0][0]

conv2d\_9 (Conv2D) (None, 28, 28, 32) 25632 conv2d\_8[0] [0]

max\_pooling2d\_5 (None, 14, 14, 32) 0 conv2d\_9[0][0]

dropout\_6 (Dropout) (None, 14, 14, 32) 0 max\_pooling2d\_5[0][0]

conv2d\_10 (Conv2D) (None, 14, 14, 64) 51264 dropout\_6[0][0]

batch\_normalization\_3 (None, 14, 14, 64) 256 conv2d\_10[0][0]

conv2d\_11 (Conv2D) (None, 14, 14, 64) 102464 batch\_normalization\_3[0][0]

conv2d\_12 (Conv2D) (None, 14, 14, 64) 18496 dropout\_6[0][0]

batch\_normalization\_4 (None, 14, 14, 64) 256 conv2d\_11[0][0]

conv2d\_13 (Conv2D) (None, 14, 14, 64) 36928 conv2d\_12[0][0]

max\_pooling2d\_6 (None, 7, 7, 64) 0 batch\_normalization\_4[0][0]

max\_pooling2d\_7 (None, 7, 7, 64) 0 conv2d\_13[0][0]

dropout\_7 (Dropout) (None, 7, 7, 64) 0 max\_pooling2d\_6[0][0]

dropout\_8 (Dropout) (None, 7, 7, 64) 0 max\_pooling2d\_7[0][0]

concatenate\_2 (Concatenate) (None, 7, 7, 128) 0 dropout\_7[0][0]

dropout\_8[0][0]

conv2d\_14 (Conv2D) (None, 7, 7, 64) 73792 concatenate\_2[0][0]

max\_pooling2d\_8 (None, 3, 3, 64) 0 conv2d\_14[0][0]

dropout\_9 (Dropout) (None, 3, 3, 64) 0 max\_pooling2d\_8[0][0]

flatten\_2 (Flatten) (None, 576) 0 dropout\_9[0][0]

dense\_3 (Dense) (None, 2048) 1181696 flatten\_2[0][0]

dropout\_10 (Dropout) (None, 2048) 0 dense\_3[0][0]

dense\_4 (Dense) (None, 122) 249978 dropout\_10[0][0]

==================================================================================================

Total params: 1,741,082

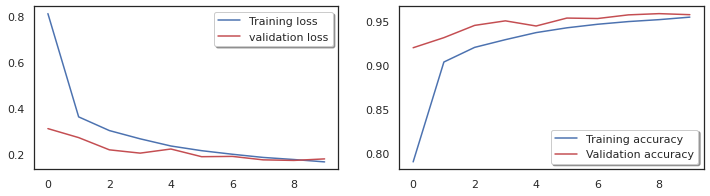
Trainable params: 1,740,826

Non-trainable params: 256

1. ***Optimizers and Learning Rate:*** Optimization helps CNNs to minimize the error. The optimizer that has been used here is the ADAM optimizer (9). It had a learning rate of 0.001, To Calculate the optimizing algorithm, categorical\_crossentropy (10) function has been used.

# Model Performance

After only 10 epochs, the model got **95.41%** accuracy on the training set and **95.69%** accuracy on the validation set. After the training, it was tested with some handwritten images and correctly identified the characters with only 2 exceptions. By analyzing the history graph, we can see the model performed well in classifying the characters.



# Conclusion

To consider the model we understand that Convolutional Neural Network can achieve better performance to classify and recognize Bangla handwritten characters, digits, and compound characters. If there were more support from a high configuration computer, GPU then the accuracy would have been better as this model gave the result with only 10 epochs. In future with more resources and bigger CNN architecture, researchers can achieve a better result and improve the state of the art scale for Bangla handwritten basic characters and digit and all compound characters recognition

# References

1. <https://shahariarrabby.github.io/ekush/#home>
2. <http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>
3. <https://keras.io/api/utils/python_utils/#to_categorical-function>
4. <https://cs231n.github.io/convolutional-networks/#conv>
5. <https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/>
6. <https://medium.com/datathings/dense-layers-explained-in-a-simple-way-62fe1db0ed75>
7. <https://keras.io/api/layers/normalization_layers/batch_normalization/>
8. <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5>
9. <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:~:text=Adam%20is%20a%20replacement%20optimization,sparse%20gradients%20on%20noisy%20problems.>
10. <https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html>