Bengali.AI Handwritten Grapheme Classification

Kimmi
Computer Science & Engineering
NIT Delhi
Delhi, India
17210034@nitdelhi.ac.in

Mahendra Soni
Computer Science & Engineering
NIT Delhi
Delhi, India
171210036@nitdelhi.ac.in

Pranshu Sharma
Computer Science & Engineering
NIT Delhi
Delhi, India
17210043@nitdelhi.ac.in

Ramanpreet Kaur
Computer Science & Engineering
NIT Delhi
Delhi, India
171210047@nitdelhi.ac.in

***Abstract*---Optical character recognition is particularly challenging for Bengali. While Bengali has 49 letters in its alphabet, there are also 18 potential diacritics, or accents. This means that there are many more graphemes, or the smallest units in a written language. Bangladesh-based non-profit Bengali. AI is focused on helping to solve this problem. They build and release crowdsourced, metadata-rich datasets and open source them through research competitions. Through this work, Bengali.AI hopes to democratize and accelerate research in Bengali language technologies and to promote machine learning education. We are given the image of a handwritten Bengali grapheme and are challenged to separately classify three constituent elements in the image: grapheme root, vowel diacritics, and consonant diacritics. Motive of this project is to accelerate Bengali handwritten optical character recognition research and help enable the digitalization of educational resources.**

1.Introduction

Bengali is the 5th most spoken language in the world with hundreds of million of speakers. It's the official language of Bangladesh and the second most spoken language in India. Considering its reach, there's significant business and educational interest in developing AI that can optically recognize images of the language handwritten. This model hopes to improve on approaches to Bengali recognition. Bengali's alphabet is made up of 11 vowels, 7 consonants, and 168 grapheme roots. This results in ~13,000 different character variations; compared English's 250 characters variations.

Image Classification has become one of the most important problems nowadays. As computers are getting better at understanding images due to advances in computer vision, solving of image classification problems using Deep Learning becoming increasingly realistic.

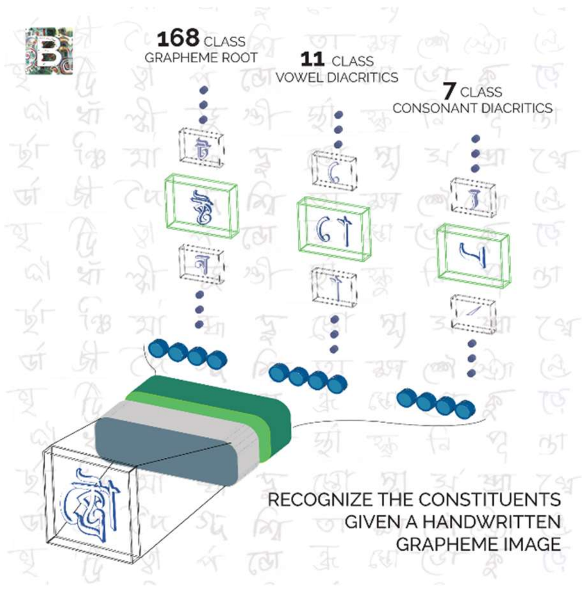

Figure 1. Bengali's alphabet is made up of 11 vowels, 7 consonants, and 168 grapheme roots

Deep learning comprises numerous layers of non-linear information processing. This allows learning architectures that implement function as repeated compositions of simpler tasks, thereby producing learning layers of abstraction with better generalization and representation capacity. A widespread algorithm for deep learning is the Convolutional Neural Network (CNN). CNNs are characterized as the innovative approach for image classification. Consequently, image recognition tasks have become prominent and essential for which convolutional neural networks have proven very effective. Deep CNNs have begun as leading substitutes to analyze immense data, presenting inventive results in image classification. Moreover, deep CNNs are at the core of most of the modern computer vision solutions extensive taxonomic tasks particularly for image classification also known for local connectivity and their weight sharing features.

In this project, we use Convolutional Neural Networks (CNN) to build a model that correctly identifies the properties of any given graphene based on the vowels, consonants and the root of the grapheme. Moreover, training a deep neural network is not that easy since it demands numerous inputs for a superior output. It also requires graphical processing units (GPUs) to process huge data inputs and intensified outputs. Further, the training process requires continual modifications of parameters to guarantee a balanced learning of all layers.

2.Dataset

Source of dataset is Kaggle. The dataset contains images of individual hand-written Bengali characters (graphemes) which are composed of three components: a grapheme_root, vowel_diacritic, and consonant_diacritic. There are 168 classes of consonant diacritic roots, 11 classes of vowel diacritics, and 7 classes of consonant diacritics.

The input dataset consists of 137x236x1 GRAYSCALE images with a total of 20,0840 training data points and 40168 test data points. Each training point is composed of an image representing a Bengali character and its corresponding components and each test point contains an image with a handwritten character and its corresponding components need to be predicted.

3.Preprocessing

Input images are clustered into 4 groups with each group containing 5,0210 images. Each group is stored as a parquet file. Each row in the parquet files contains an image_id column, and the flattened image.

4.Methodology

4.1 Baseline Model

Our baseline model training aims to ascertain that our data structure and initial model are compatible. The convolutional neural network was chosen as the baseline model for this training.

Convolutional neural networks are used since CNN is one of the algorithms which classify images with high accuracy and effectively extracts image features by reducing the number of parameters. The methodology used in the project is modifying the parameters, filters, number and position of convolutional layers, pooling layers, dropout layers in order to get the optimized parameters which provides a good accuracy while also taking into account the amount of time taken to train and test the model.

By investigating public online notebooks, we first built a CNN consisting of 20 convolutional layers with batch normalization and dropouts after max-pooling layers. The simplified architecture plot is shown below. The output layer is designed to be three parallel layers with 168 nodes, 11 nodes, and 7 nodes, respectively.

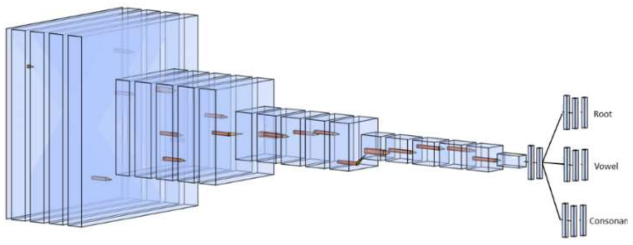

Figure 2

4.2 Modifying Structures

4.2.1 More Convolutional Layers

The first modification we tried is on the convolutional layers. The layer's parameters consist of a set of learnable filters. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. We wanted to add more Conv2D layers with fewer filters in the beginning to improve the model's interpretation of the characters. But after 40 epochs, the accuracy of three targets saw no increase.

4.2.2 Pooling

Pooling layer is used to reduce the dimensions of a layer. Pooling may compute a max, average or sum. Max pooling uses the maximum value, average pooling uses the average value and sum pooling computes the sum of each of a cluster of neurons.

4.2.3 ReLU layer

It effectively removes negative values from an activation map by setting them to zero while positive values remain unaffected.

4.2.4 Dropout

A fully connected layer is prone to overfitting. Dropout is one method to reduce overfitting. At each training stage, nodes are dropped out of the net with some probability.

5. Architecture

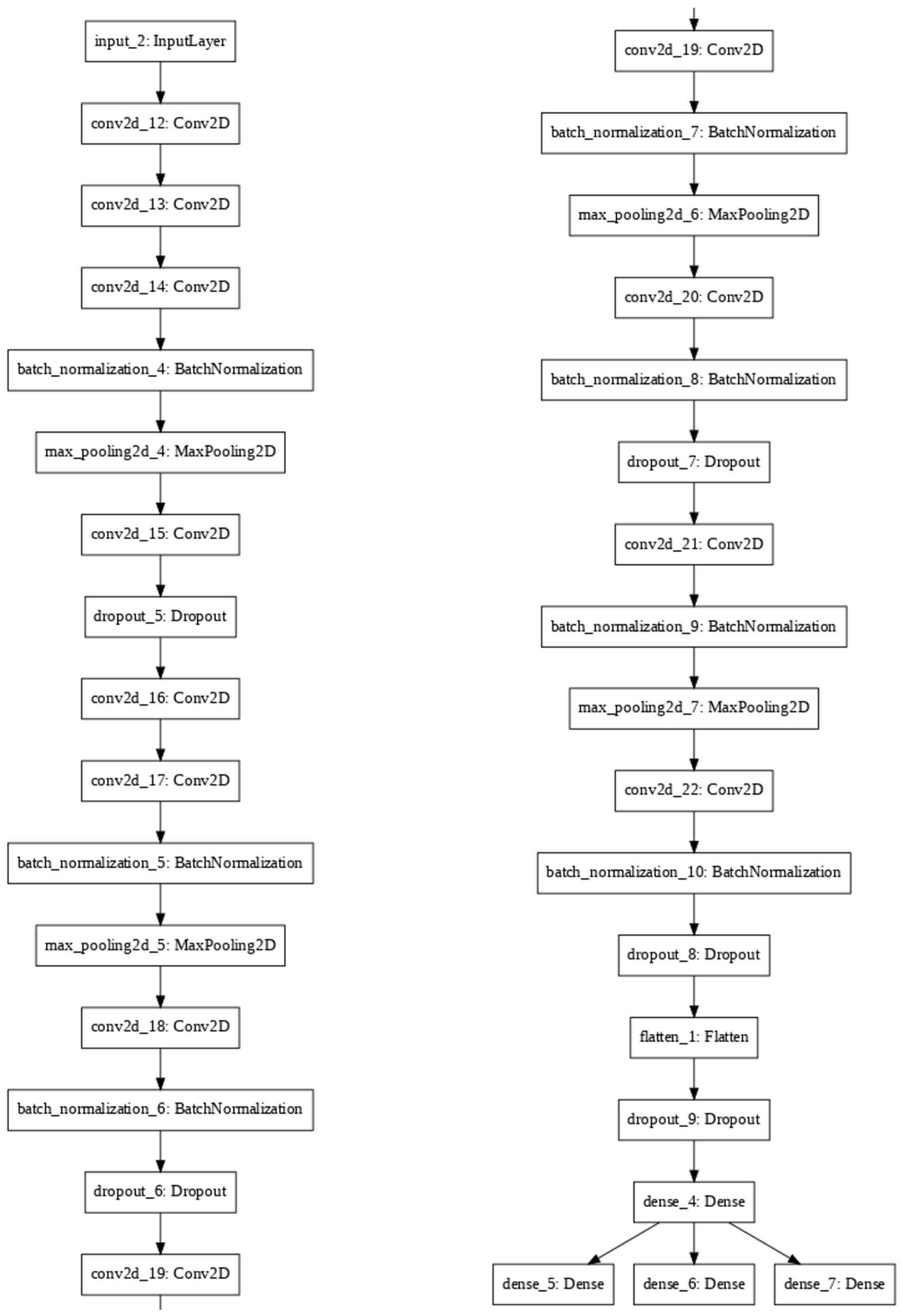
input_2: InputLayer
conv2d_12: Conv2D
conv2d_13: Conv2D
conv2d_14: Conv2D
batch_normalization_4: BatchNormalization
max_pooling2d_4: MaxPooling2D
conv2d_15: Conv2D
dropout_5: Dropout
conv2d_16: Conv2D
conv2d_17: Conv2D
batch_normalization_5: BatchNormalization
max_pooling2d_5: MaxPooling2D
conv2d_18: Conv2D
batch_normalization_6: BatchNormalization
dropout_6: Dropout
conv2d_19: Conv2D
conv2d_19: Conv2D
batch_normalization_7: BatchNormalization
max_pooling2d_6: MaxPooling2D
conv2d_20: Conv2D
batch_normalization_8: BatchNormalization
dropout_7: Dropout
conv2d_21: Conv2D
batch_normalization_9: BatchNormalization
max_pooling2d_7: MaxPooling2D
conv2d_22: Conv2D
batch_normalization_10: BatchNormalization
dropout_8: Dropout
flatten_1: Flatten
dropout_9: Dropout
dense_4: Dense
dense_5: Dense
dense_6: Dense
dense_7: Dense

Figure 3. Architecture of our Model

6.Training the Classifier Model

There are roughly 10,000 possible graphemes, of which roughly 1,000 are represented in the training set. The test set includes some graphemes that do not exist in train but has no new grapheme components.
The training process follows the network architecture. Before training, the dataset of images is split into 60% training, 20% validation and 20% testing. Then followed by a processing scheme including feature extraction, image resize and classification. The whole training took about 90 minutes just for one training without augmentation.

7. Result and Discussion

We trained all models using all four datasets and found the model with multiple separate dense layers after convolutional. The results are shown in the table below.

Table 1. Accuracy and loss of Model

	Training	Validation
Main loss	0.5789	0.16
Root loss	0.8861	0.4447
Vowels loss	0.1052	0.1113
Consonant Loss	0.0872	0.1056
Root accuracy	87.49%	87.68%
Vowel accuracy	96.80%	96.37%
Consonant accuracy	97.18%	96.87%

The accuracy plot shows that both training and validation accuracy generally keeps increasing across epochs.

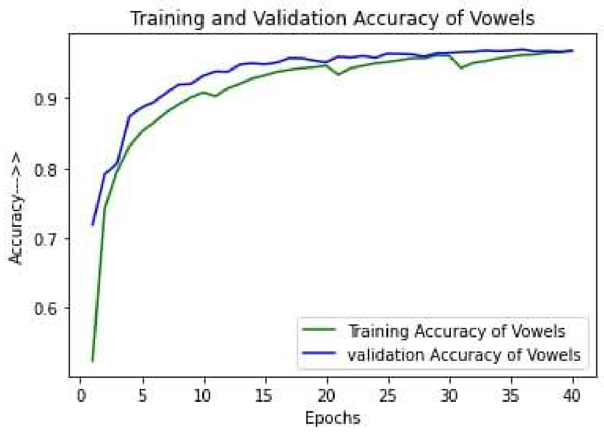

Figure 4. Training and Validation Accuracy of Vowels

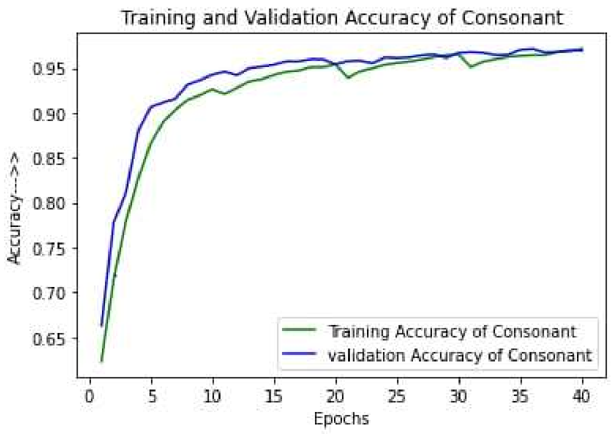

Figure 5. Training and Validation Accuracy of Consonant

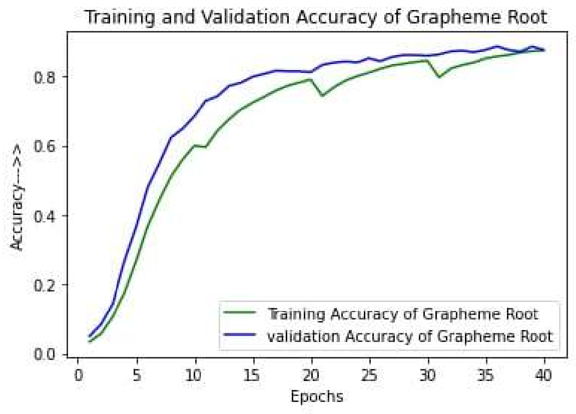

Figure 6. Training and Validation Accuracy of Grapheme Root

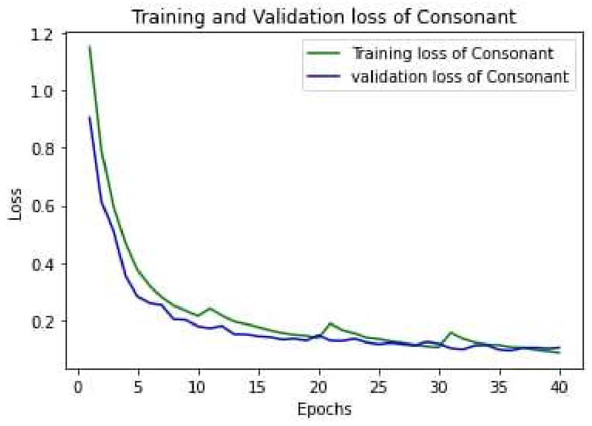

Figure 8. Training and Validation Loss of Consonant

The loss plot shows that the loss function generally decreases, indicating there is nearly no overfitting problem for the current model.

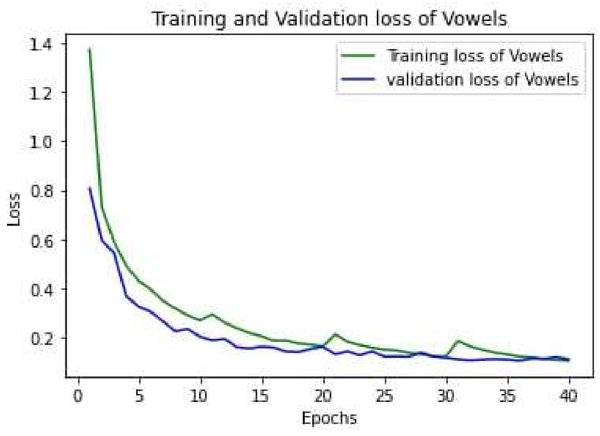

Figure 7. Training and Validation Loss of Vowels

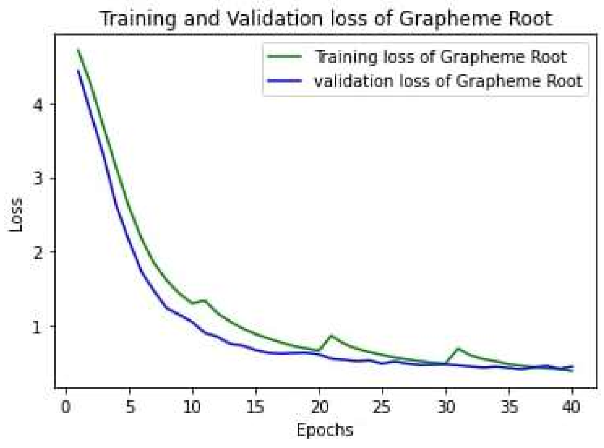

Figure 9. Training and Validation Loss of Grapheme Root

8. Future Works

Even though we have achieved a good result in this Bengali grapheme classification task, there are still some further improvement solutions we can apply. First, a more scientific hyperparameter selection process might result in better model performance. This means to consider more possible hyperparameters and choose a wider range of possible values for each hyperparameter. This procedure usually takes a lot of time and effort. In this project, we have limited time and resources. However, it is one of the possible and feasible ways to improve the result. Besides hyperparameters that take on specific values, other possible future solutions may include choosing a different initialization method, activation function, and so on. In addition, models can be improved through further changing the architectures. Different architectures are suitable for different contexts.

Other interesting aspects of this relatively difficult classification task is the test speed performance and number of weights pruning besides test accuracy performance. Even though nowadays hardware and computers are highly capable of loading large and deep neural networks, we still expect to utilize the network in light hardware, such as mobile phones. In this way, people can encapsulate the neural networks as an application and use their phones to classify the Bengali.

References

[1] https://www.kaggle.com/c/bengaliai-cv19

[2]https://en.wikipedia.org/wiki/Convolutional_neural_network

[3]https://www.google.com/url?sa=t&source=web&rct=j&url=https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff&ved=2ahUKEwjLw8mPvbXpAhXx4jgGHUebDwUQjjgwGXoECAEQAQ&usg=AOvVaw0Z8HDatbJDUYuPByuf622z

[4]https://www.pyimagesearch.com/2018/05/07/multi-label-classification-with-keras/

[5] https://www.tensorflow.org/api_docs/python/tf/keras/Model