Bengali Handwritten Grapheme Classification on Bengali.AI dataset

Rafin Abrar Rono
Dept. of CS
American International University - Bangladesh
22-47226-1@student.aiub.edu

Md. Tahsin Hasib
Dept. of CS
American International University - Bangladesh
22-46026-1@student.aiub.edu

Umme Sumiya Rashid Dristi
Dept. of CS
American International University - Bangladesh
22-46841-1@student.aiub.edu

Khushbu Alam Rahi
Dept. of CS
American International University - Bangladesh
22-46947-1@student.aiub.edu

***Abstract*—To classify Bengali graphemes along with vowel and consonant diacritics, 2 pretrained models (DenseNet121 & Xception) were trained and a convolutional neural network (CNN) was constructed and trained on 50,000 images from the large multi-target dataset from Bengali.AI handwritten grapheme classification contest. For each training image there were 3 distinct labels: grapheme root, vowel diacritic & consonant diacritic. The neural network consisting of 684,000 parameters consists of 10 separable convolutional layers, followed by a batch normalization and a max pooling layer. After convolution, there is a global average pooling layer to flatten the outputs. Finally, there are 3 separate output dense layers each with softmax activations and neuron counts corresponding to respective classes in each component. Dropout layers were added to reduce overfitting. On the Bengali.AI dataset, the CNN achieved grapheme root validation accuracy of 74.23%, vowel diacritic validation accuracy of 90.33% and consonant diacritic validation accuracy of 90.17%.**

***Keywords*—Bangla grapheme, CNN, DenseNet121, Xception, Deep learning, Transferlearning.**

I. Introduction

Compared to English, the Bengali script is more complex containing a total of 50 base characters comprised of 11 vowels and 39 consonants. These can be combined with 18 diacritics to form thousands of unique grapheme combinations. To classify different aspects of combinations accurately is a challenging task when considering various of these combinations are so similar is shape that they're distinguished using a dot or a single line. These small but significant distinct features can be captured using modern deep learning convolutional models. But even that will require a significantly large dataset for these models to train on. On this topic, Bengali.AI from Bangladesh has crafted a large multi-target dataset containing publicly sourced handwritten grapheme symbols [1]. This large pool of thousands of grapheme combinations can be used to train and extract features for complex classification tasks related to Bengali symbols.

The recent developments in deep learning has given rise to various techniques of image classification. Deep convolutional neural networks (DCNN), residual networks, dense networks are just a few of the commonly practiced architectures for classification, detection, pattern recognition problems. Pooling, batch normalizations added both diversion and stability in the learning process and resulted in deeper networks being trained significantly faster. Dropout layers can slow down the learning but result in reduced overfitting as the model will slow down its memorization while increase its learning. These additions to a model can help it to classify the Bengali grapheme classification problem more smoothly.

In order to classify the 3 components (root, vowel diacritic & consonant diacritic) of handwritten grapheme images, 2 pretrained models DenseNet121 & Xception were fine-tuned to the training dataset at first. After that, to decrease the total size and parameters, a convolutional neural network (CNN) was constructed and trained on a subset of the Bengali.AI multi-target dataset.

II. Related Studies

Several studies have used deep learning models with varying techniques to recognize patterns and classify Bengali graphemes. Roy et al. (2017) [2] presented a novel deep learning technique which was tested on CMATERdb 3.1.3.3 dataset. The Deep CNN (DCNN) used RMSProp algorithm to achieve faster convergence and outperformed SVMs with an error rate of 9.67% and accuracy of 90.33%. This model also represented a 10% improvement over other popular DCNNs of the time.

Similarly, Chowdhury et al. [3] constructed a CNN and trained it on BanglaLekha-Isolated dataset to recognize and convert Bengali handwritten symbols to electronic format. This model achieved a 91.81% accuracy on classifying the 50 base Bengali characters and the accuracy increased to 95.25% when images were augmented to expand the dataset size.

For multi-target classification, Roy et al. (2021) tested Multi-layer Perceptron (MLP) model as well as ResNet50 on the Bengali.AI grapheme classification Kaggle competition dataset [3, 4]. The competition proposed the challenge of classifying 3 elements of a Bengali grapheme: root, vowel & consonant

diacritics) from each image. To improve the accuracy and fit the solution within the time constraint, they proposed their own CNN which obtained 95.32% validation root accuracy, 98.61% vowel accuracy & 98.76% consonant accuracy. Additionally, they proposed the usage of a Regional Proposal Network, which detects the edges surrounding the graphemes to estimate each component separately.

III. Dataset

For this study, the Bengali.AI Kaggle competition dataset was used. Instead of procuring each of the 200,540 training images, the data was supplied in 4 parquet files with a csv file containing the necessary mapping to and their corresponding labels. As this is a multi-target classification problem, each training image consisted of 3 labels (grapheme root, vowel diacritic & consonant diacritic). The parquet files contained the color values [0-255] for each pixel of each 137x236 training image. A class map file was also provided which contained the mapping of which grapheme/vowel/consonant corresponded to which index of their respective class. Fig-1 visualizes the class mapping structure as well as how each handwritten grapheme image contained the 3 components.

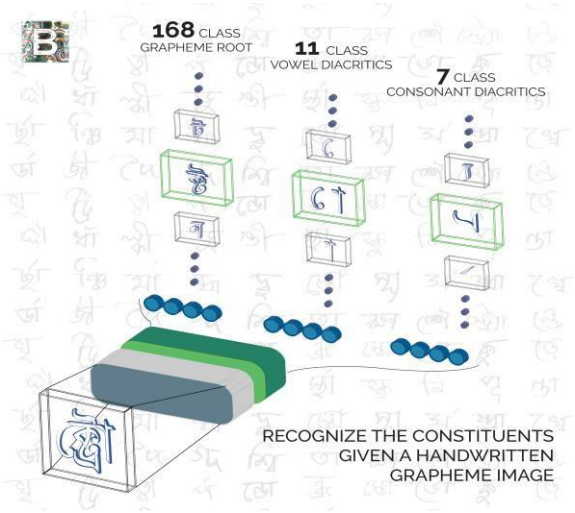


Fig. 1: Class map of a grapheme image [4].

Additionally, the following graphs showcase the frequency of each of the class elements for all components. The count is limited to the first 50,000 images as we are performing training on a subset only.

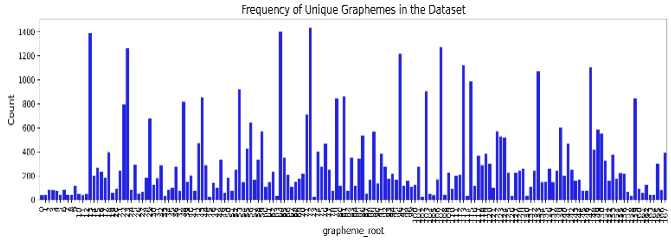


Fig. 2: Occurrences of unique 168 grapheme roots.

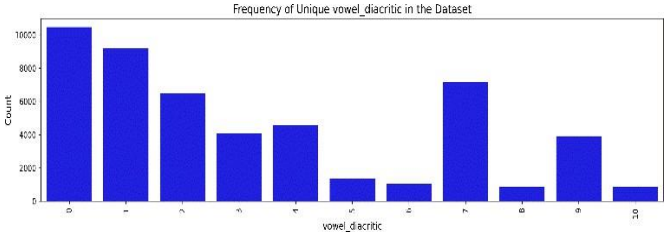


Fig. 3: Occurrences of unique 11 vowel diacritics.

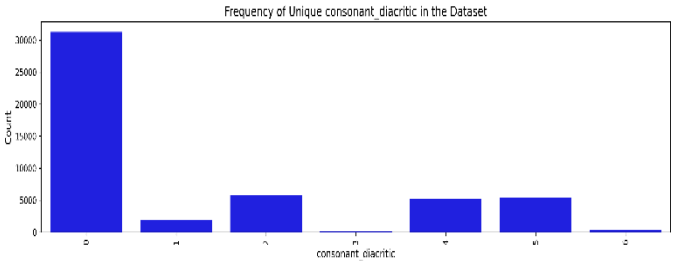


Fig. 4: Occurrences of unique 7 consonant diacritics.

IV. Methods

The primary objective is to construct a model that can accurately identify the 3 components a given Bengali grapheme image. But firstly, 2 pretrained deep learning models, DenseNet121 and Xception were finetuned to observed their performance on the training data.

A. DenseNet121 & Xception

These 2 models were used as the backbone separately. Both were trained on the first 50,000 training images out of the available 200,840 to save time. For each of them the output of the backbone model were passed global average poling, followed by dropout and dense layers to finally be connected to 3 separate output layers to classify 3 different components.

B. Proposed Method

A CNN model was crafted to classify grapheme components: roots, vowels & consonant diacritics.

1) Preprocessing:

In the image preprocessing pipeline, each training image was converted to grayscale for faster computations. After that binary thresholding was placed to crop out the writing from the image and leave out as few empty spaces around the writing as possible. To maintain consistency of input shapes square padding was used for unevenly cropped images. The pixel values of the images were also divided by 255 to normalize them between the [0-1] range.

2) Data Augmentation:

By observing the original data, it can be noticed that different images have writings on different places of the image. Some images have writings in a slightly curved way, some were slightly rotated and size of the writings themselves also varied. To enable the model to recognize patterns even in writings of different sizes, rotations, Keras's image data

generator was used to augment the training images. The augmentations being: random rotation, height & width shifts and random shearing.

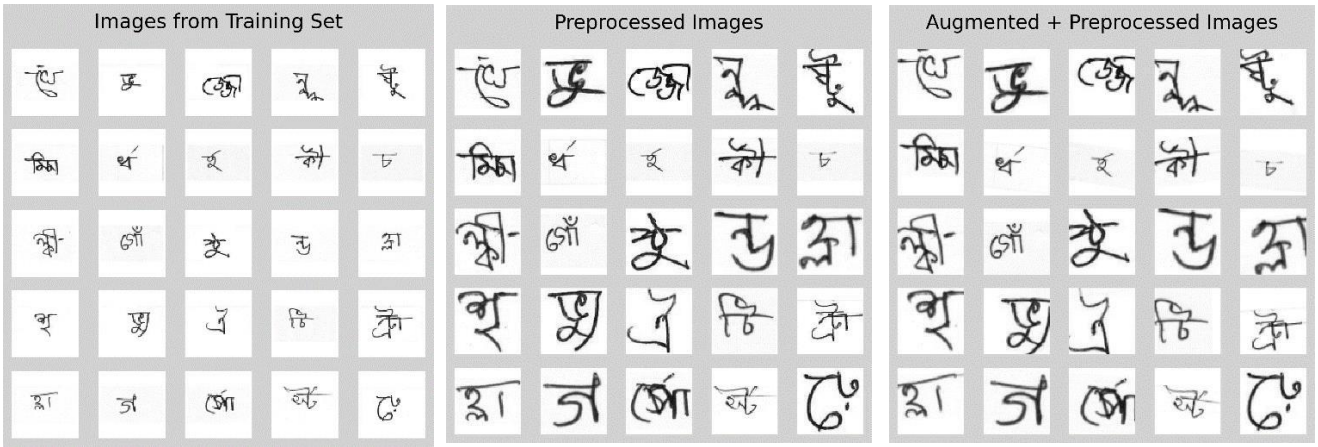
3) Model:

The model (fig. 6) takes input images in 64x64 shape while grayscale formatted. The model uses 10 separable convolutional layers, each of which has 'same' padding and *relu* as the activation function. These layers perform similarly to regular convolutional layers but in a 2-step way resulting in less parameters for nearly similar convolution.

In the model architecture, 2 convolutional layers are stacked on after another, followed by 1 batch normalization layer with 0.5 momentum to normalize outputs and speed up the training process. To help the model learn images from a different perspective, the output tensor is passed to a max pooling layer with a pool size of (2, 2). Every time a tensor is passed to a max pool layer, the output dimensions get halved which ultimately results in a very small tensor dimension. These 4 layers make up 1 of the 5 convolution blocks used throughout the model. Between 2 of these blocks a dropout of 0.2 was used, resulting in 20% of the weights being reduced to 0 for reduce overfitting. As the model goes deeper, the filter size for convolution blocks also increase, with starting from 32 filters in the first block and reaching 256 filters in the final convolution block. After these 5 blocks, a global average pooling layer was used to take the average value from each of the dimensions, which flattens the input tensor in a one-dimensional area. Due to the global average process, this layer results in fewer parameters compared to when using a flatten layer to get similar results. After this, the output is passed to a dense layer comprised of 1024 neurons with *relu* activation function, which is then connected to a normalization layer.

Finally, the normalization layer is connected to 3 separate dense layers corresponding to the 3 components aimed to be detected by this model. These 3 layers use *softmax* as the activation function to get a probabilistic value on each of the possible classes and the neuron count in each of them correspond to the values for each type of grapheme component. When compiling the model for training, *Adam* optimizer was used with an initial learning rate of 0.001. To calculate the loss, each of the output layers incorporated *sparse categorical cross entropy.* The final model had a total of 684,00 parameters with a size of just 2.62 Mbytes. This makes the model a very light CNN with enough dropout and pooling to account for the various formations of grapheme structures seen throughout the training images.

The Grapheme structures after preprocessing and augmentation is shown in fig. 5.



(a) Original images (b) Preprocessed images (c) Augmented Images

Fig. 5: A view of the original images (a), preprocess images (b) & augmented images after preprocessing (c).

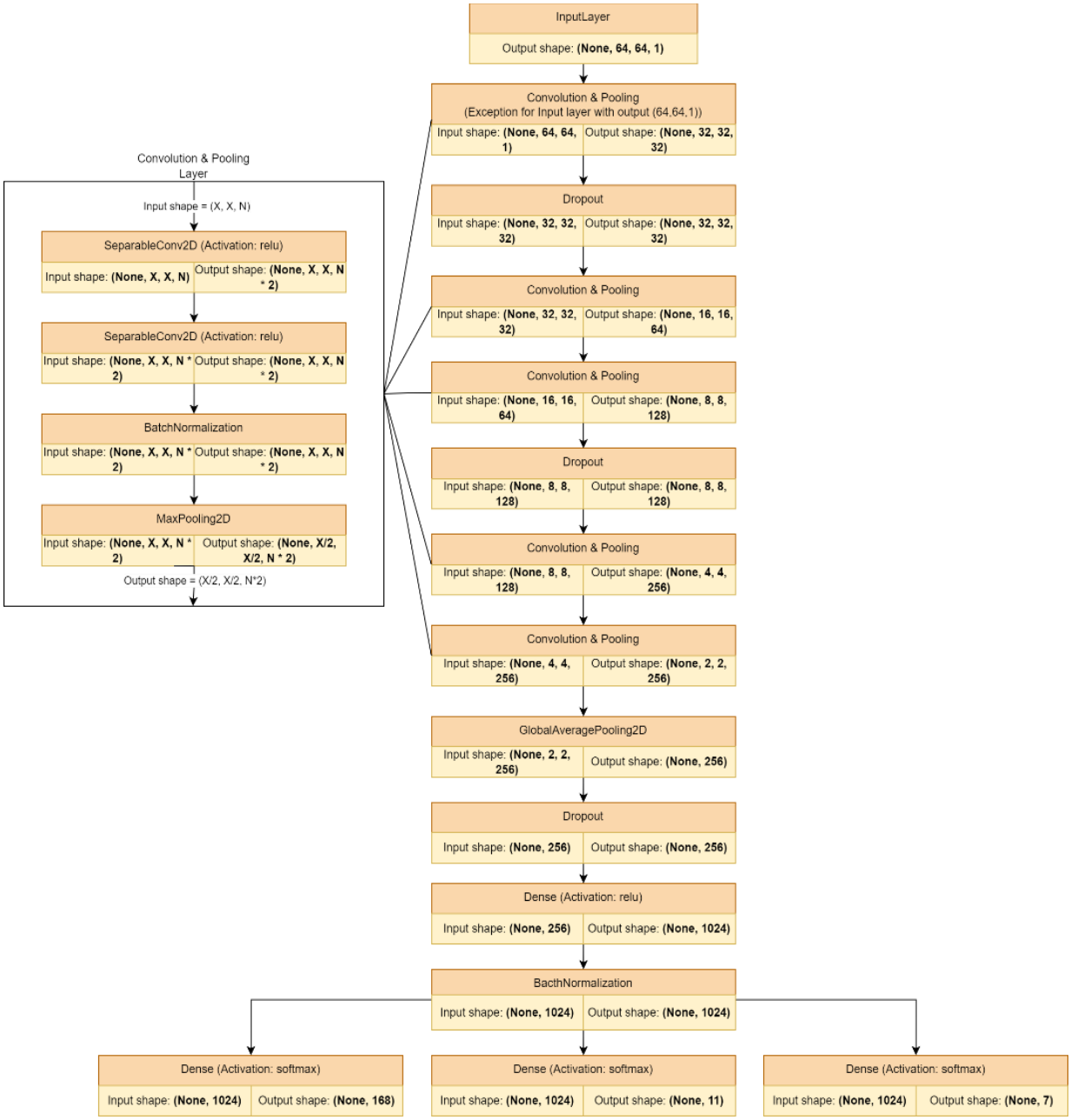


Fig. 6: Architecture of proposed CNN model.

V. Results and Discussion

A. *DenseNet121 & Xception*

Model accuracy and losses of both the DenseNet121 and Xception for first 50,000 training images is shown below. Batch size was 128 while 75% of the images were used for training while the remainder was used for validation. Even though the models were designated to train for 50 epochs but using Keras's early stopping callback with a threshold of 0.001 and patience of 5, DenseNet121 stopped at 28 epochs and Xception stopped at 32. The trends for each of the models is displayed below.

For DenseNet121:

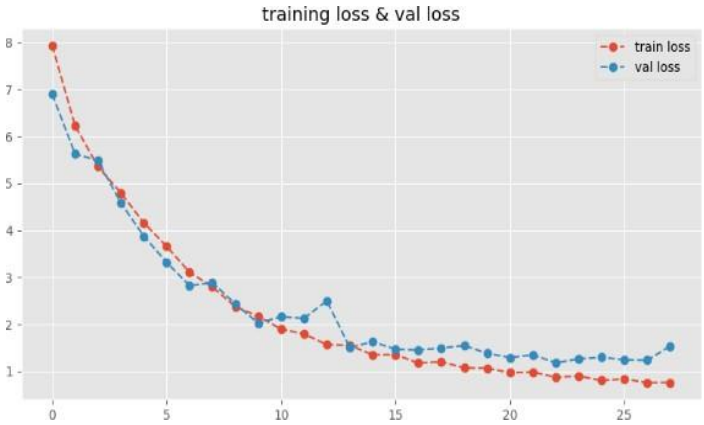


Fig. 7: Train loss and Validation loss for densenet121.

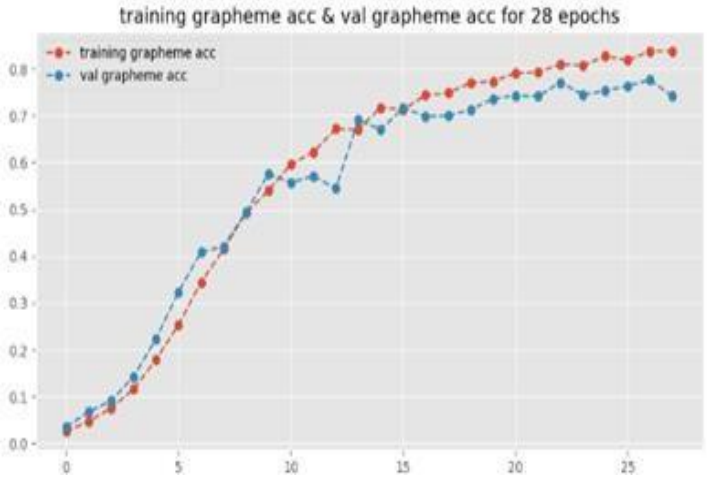


Fig. 8: Grapheme root accuracy for densenet121.

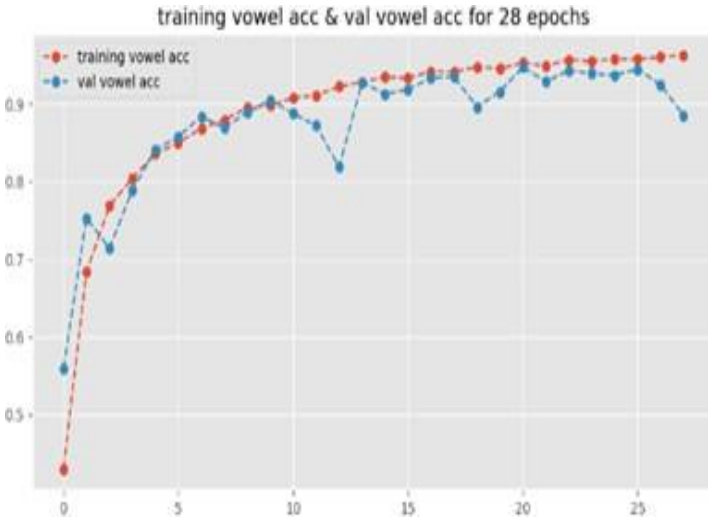


Fig. 9: Vowel diacriticsaccuracy for densenet121.

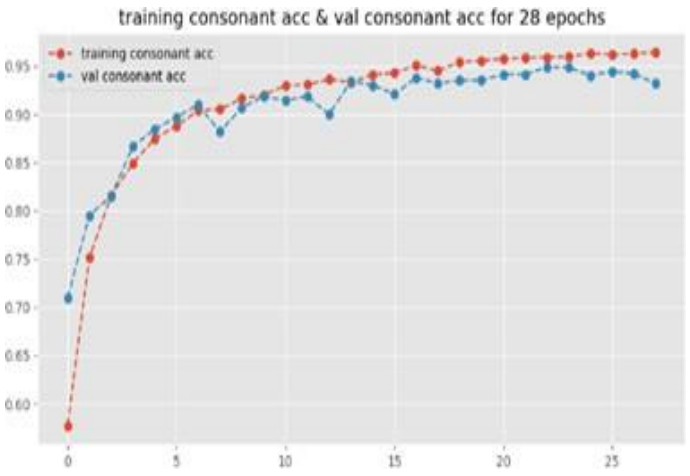


Fig. 10: Consonant diacritics accuracy for densenet121.

For Xception:

Fig. 11: Train loss and Validation loss for Xception.

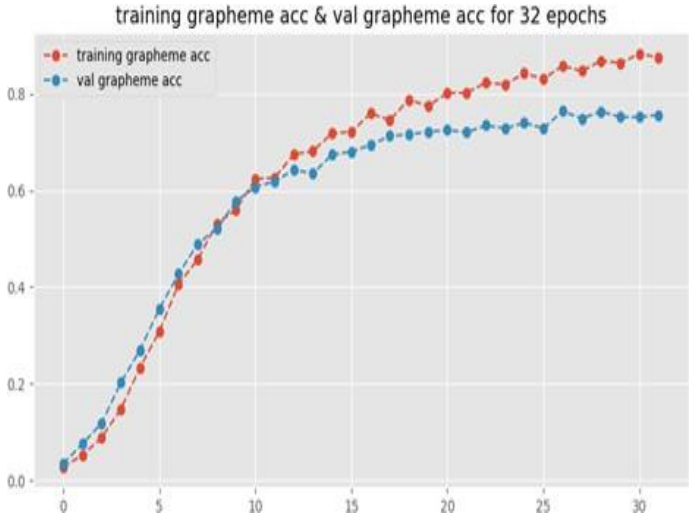


Fig. 12: Grapheme roots accuracy for Xception.

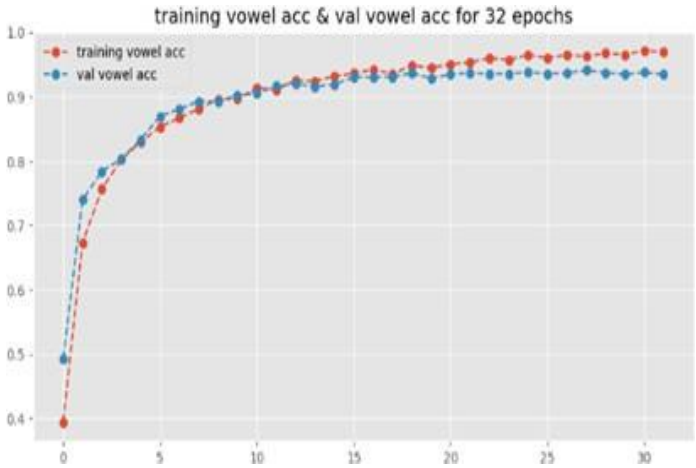


Fig. 13: Vowel diacritics accuracy for Xception.

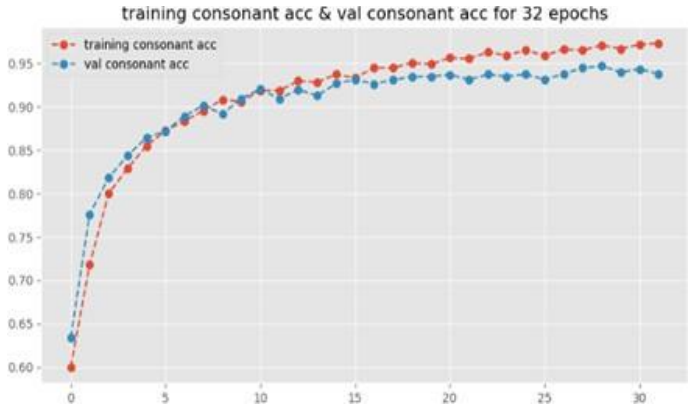


Fig. 14: Consonant diacritics accuracy for Xception.

B. *Proposed Model*

The finetuned models worked great, but the models themselves were very deep in layers and the saved models resulted in sized around 400 Mbytes. So, the custom CNN was tested on the same subset of training images as the before 2. Epoch was set to 50 with a batch size of 128. For callbacks, early stopping was used based on validation loss with the same parameters as the first 2 models, but unlike DenseNet121 or Xception, the training continued all the way, meaning despite the slower learning rate, the model had a continued reduction in its overall validation loss.

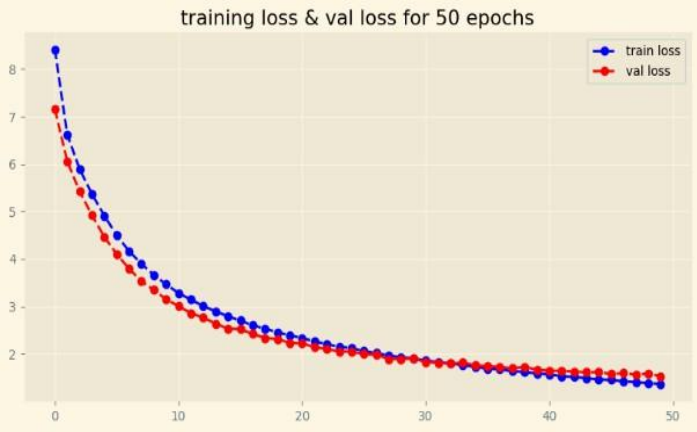


Fig. 15: Train loss and Validation loss for CNN.

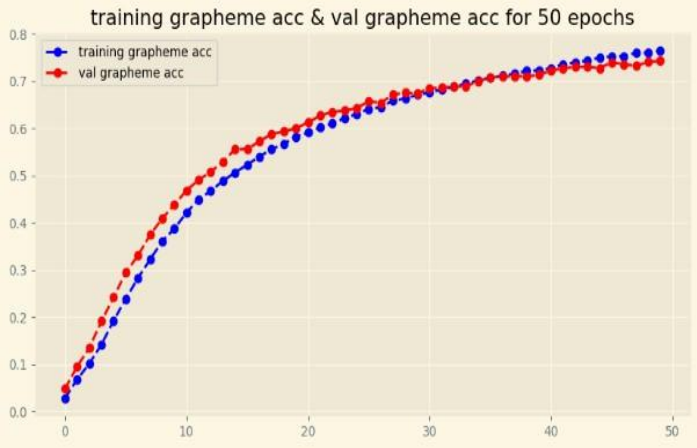


Fig. 16: Grapheme accuracy for CNN.

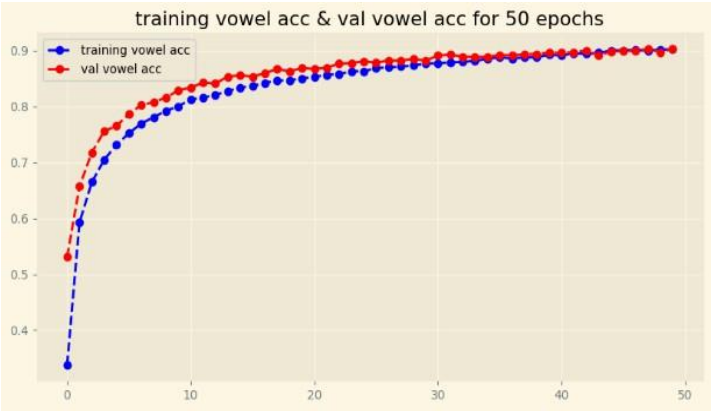


Fig. 16: Vowel diacritics accuracy for CNN.

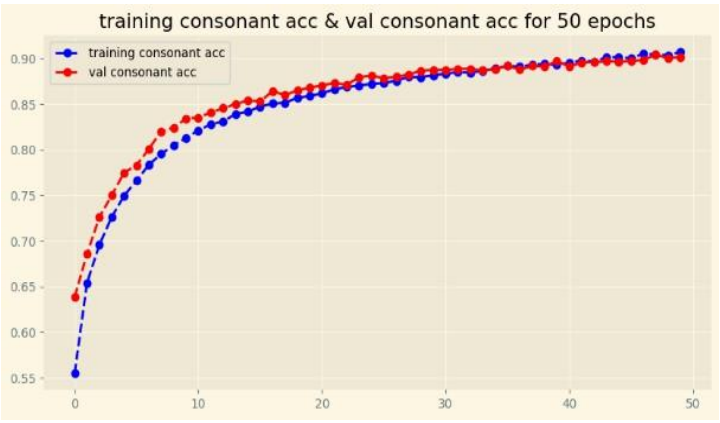


Fig. 17: Consonant diacritics accuracy for CNN.

VI. Conclusion

Though CNNs are used for various purposes including pattern recognition and classification problems, large models often lead to huge training time and disk space when used in systems. For Bengali grapheme multi-target classification tasks where there can be a huge number of unique combinations, the classification process may lead to huge training & inference time. The performances (train/validation) of 3 models discussed in this study is shown below:

Model	Grapheme accuracy	Vowel accuracy	Consonant accuracy
DenseNet121	83.28% / 74.11%	96.34% / 88.56%	96.31% / 93.17%
Xception	87.68% / 75.64%	96.86% /93.*48*%	97.25% / 93.78%
Proposed CNN	76.27% / 74.23%	90.12% / 90.33%	90.89% / 90.17%

References

[1] S. Alam, T. Reasat, A. S. Sushmit, S. M. Siddiquee, F. Rahman, M. Hasan, and A. I. Humayun, "A large multi-target dataset of common bengali handwritten graphemes," 2021.

[2] M. K. Saikat Roy, Nibaran Das and M. Nasipuri, "Handwritten isolated bangla compound character recognition:a new benchmark using a novel deep learning approach," *CoRR*, vol. abs/1802.00671, 2018. [Online]. Available: https://arxiv.org/abs/1802.00671

[3] T. Roy, H. Hasam, K.Hossai & M. A. Rumi. "Bengali Handwritten Grapheme Classification: Deep Learning Approach", *CORR* vol. abs/2111.08249, 2021 [Online]
Available: https://arxiv.org/abs/2111.08249

[4] "Bengali.AI Handwritten Grapheme Classification", https://www.kaggle.com/bengaliai-cv19