Bengali.AI Handwritten Grapheme Classification

Abraham Jose
CAP 6412: Advanced Computer Vision
University of Central Florida
abraham@knights.ucf.edu

Abstract

*Bengali is considered as one of the most spoken language in the whole world. Handwritten character recognition in this language is particularly difficult because it uses very complex structures and it is difficult to distinguish diacritics from its grapheme roots and there can be close to 13,000 grapheme combinations in the script compared to 250 graphemic variations in English. There are only 1292 grapheme variations that the training dataset has which are frequently used in the language while the rest are either not being used in training dataset or obsolete. This is a classification problem and it requires each grapheme variation to be classified into one in each of 168 grapheme roots, 11 vowels and 7 consonant diacritic groups. To tackle the problem, the state-of-the art version of ResNet is being used along with certain modifications in the network to acheive performance boost. In-order to achieve high accuracy and robust performance in real world tasks, certain augmentations seasoned for this particular problem is used for training the model. The final model SE-ResNeXt-50 was able to perform with a recall score of **97.1%** and accuracy of **96%**, **99%** and **98.8%** for grapheme roots, vowels and consonants respectively.*

1. Introduction

Bengali AI OCR recognition is a competition introduced by Kaggle and is the 5th most spoken language in the world. This challenge was introduced in Kaggle and had its final submission deadline on March 16, 2020 starting from March 9th. This competetion was hosted by Apurba and Intelligent Machines Limited towards their Bengali NLP research for accelerating the field and research in the domain for advanced solutions and AI research. Bengali or Bangala is an Indo-Aryan language primarily spoken in South Asia with 228 million speakers and the complexity of the script is relatively high considering many variations of the basic grapheme units. The language has about 168 grapheme roots, 11 vowels and 7 consonants which makes 12936 combinations of grapheme variation. However, in the spoken and written language, there are many combinations that is not being used as many words are lost or not being used frequently as opposed to some other grapheme units. There are 1292 grapheme variations in the training dataset and there could be unseen grapheme variations in the testing set. Each of this characters in the grapheme root will be accompanied with the vowel marks and consonant marks and it is difficult to distinguish between each components in these many combinations of characters. This makes the problem interesting from traditional OCR problems. For example in English language, the OCR detection is fairly easy where there are only 27 characters to be detected with no complex structures in the scripts and the detection is spanned in just this 27 characters and so is many other languages. Bengali is an Indo-Aryian language rooted in Sanskrit and has many other South Asian languages like Hindi, Tibetan, Tirhuta, Assamese etc.. having same scripting complexities and efficiently solving this problem means the same techniques can be applied for the other languages as well. The OCR enables the

2. Background and Related work

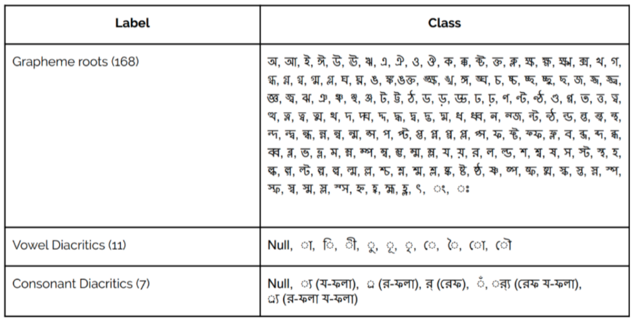
2.1. Optical Character Recognition

Optical Character Recognition(OCR) is considered as one of the oldest computer vision/ image processing challenge. It requires electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text. It started of as a pattern recognition problem in 1914 addressed by Emanuel Goldberg who invented a mechanical machine that read characters and converted them into standard telegraph code. It was in 1974 when Kurzweil Computer Products developed multi-font OCR(electronic conversion) into production. These are based on statistical based image processing with the technology available at the time and OCR became a pattern and character recognition problem. It was in 1989 when the first artificial learning model called LeNet5 was applied to this problem by Yann LeCun at Bell Labs introducing the back propagation to the world. It was a single-layer neural network applied to the hand written recognition problem to tackle real-world practical problems using OCR. The learning capabilities of neural network was explored extensively since then using large data and computation capabilities that was exponentially growing. When convolutional neural networks were introduced for OCR, the performance increased and LeCun was able to achieve a model that performs with only 1% error rate and 9% rejection rate in the dataset that he was using.

The recent works in the field of OCR includes the STN-OCR[2], which is capable of detecting the text components and recognize them using a single model. It uses a localization network followed by a grid generator and then the recognition network to recognize each text using a single model. There has not been many works in OCR as such since the state-of-the art performance is at its best and is used widely in many domains extensively. The other works in the field includes [3]Online and offline handwritten Chinese character recognition, which has around 3036(ETL9B dataset)-3755(GB2312-80 dataset) Japanese character classes including 2965 Kanji character classes which has fairly complex characters to be recognized. The method they employed has 4 types of classifiers modified quadratic discriminant function, nearest prototype classifier(NPC), NPC with discriminative feature extraction and discriminative quadratic discriminant function using statistical methods. The other recent work is [4] which uses an attention mechanism for training the model based on ResNet which uses an attention mechanism called FAN to automatically draw back the drifted attention. The method has an attention network and a focusing network to attain state of the art performance in the natural image datasets such IIIT5k, SVT and ICDAR.

2.2. Bengali OCR Dataset

The Bengali OCR dataset has 168 grapheme roots, 11 vowels and 7 consonants. In the training dataset provided from Kaggle, there are 200840 images in total and the csv containing the classification for their corresponding grapheme root, vowel and consonant. The images are stored in parquet format in 4 different parquet files with each file containing 50210 flattened images, each saved in 32332 size array of values. Inorder to read the image, we need to resize the image by (137, 236) to read the single channel images from the dataset. As opposed to the training dataset, the testing dataset only has 11 images for testing the model with no ground truth as these images are meant for submission in the particular competition. So the work is constrained to the training set alone which will be divided to training and validation sets respectively. The scripts in Bengali looks like the following and the diacritics are the appendages to the grapheme roots which will give us the variation of a particular grapheme root.



Label	Class
Grapheme roots (168)	অ, আ, ই, ঈ, উ, ঊ, ঋ, এ, ঐ, ও, ঔ, ক, ক্ক, ক্ট, ক্ত, ক্ল, ক্ষ, ক্ষ্ণ, ক্ষ্ম, ক্স, খ, গ, গ্ধ, গ্ন, গ্ব, গ্ম, গ্ল, ঘ, ঘ্ন, ঙ, ঙ্ক, ঙ্ক্ত, ঙ্ক্ষ, ঙ্খ, ঙ্গ, ঙ্ঘ, চ, চ্চ, চ্ছ, চ্ছ্ব, ছ, জ, জ্জ, জ্জ্ব, জ্ঞ, জ্ব, ঝ, ঞ, ঞ্চ, ঞ্ছ, ঞ্জ, ট, ট্ট, ঠ, ড, ড়, ড্ড, ঢ, ঢ়, ণ, ণ্ট, ণ্ঠ, ণ্ড, ণ্ণ, ত, ত্ত, ত্ত্ব, ত্থ, ত্ন, ত্ব, ত্ম, থ, দ, দ্ঘ, দ্দ, দ্ধ, দ্ব, দ্ভ, দ্ম, ধ, ধ্ব, ন, ন্জ, ন্ট, ন্ঠ, ন্ড, ন্ত, ন্ত্ব, ন্থ, ন্দ, ন্দ্ব, ন্ধ, ন্ন, ন্ব, ন্ম, ন্স, প, প্ট, প্ত, প্ন, প্প, প্ল, প্স, ফ, ফ্ট, ফ্ফ, ফ্ল, ব, ব্জ, ব্দ, ব্ধ, ব্ব, ব্ল, ভ, ভ্ল, ম, ম্ন, ম্প, ম্ব, ম্ভ, ম্ম, ম্ল, য, য়, র, ল, ল্ড, শ, শ্ব, ষ, স, স্ট, স্থ, হ, ল্ক, ল্গ, ল্ট, ল্প, ল্ব, ল্ম, ল্ল, শ্চ, শ্ন, শ্ম, শ্ল, ষ্ক, ষ্ট, ষ্ঠ, ষ্ণ, ষ্প, ষ্ফ, ষ্ম, স্ক, স্ত, স্ন, স্প, স্ফ, স্ব, স্ম, স্ল, স্স, হ্ন, হ্ব, হ্ম, হ্ল, ৎ, ং, ঃ
Vowel Diacritics (11)	Null, া, ি, ী, ু, ূ, ৃ, ে, ৈ, ো, ৌ
Consonant Diacritics (7)	Null, ্য (য-ফলা), ্র (র-ফলা), র্ (রেফ), ঁ, র্্য (রেফ য-ফলা), ্র্য (র-ফলা য-ফলা)

Figure 1: Vowel Diacritics v/s Consonant Diacritics

Not all the grapheme roots takes all the variations in the vowels and consonants and there is a huge frequency gap between the available combinations of grapheme root variations in the training dataset.

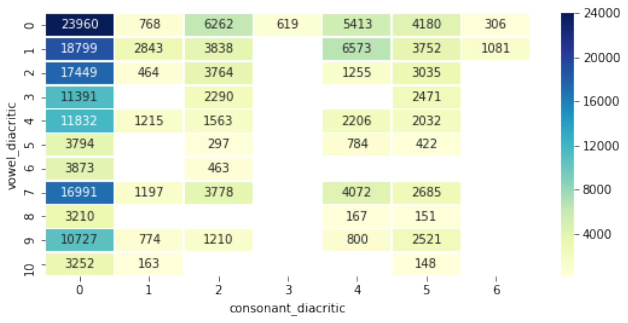


Figure 2: Vowel Diacritics v/s Consonant Diacritics

From figure 2. we can see that there is a lot of variations in the dataset distribution in the consonant and vowel diacritics categories from the 200840 image samples. The grapheme root vs vowels/consonants diacritics contains a much sparse matrix from the scripts used in current commonly language. Hence there is lot of data distribution imbalance in the dataset as we can see and not all the combinations are available adn there is a huge frequency gap between many variations of grapheme roots which makes it a challenging problem.

3. Method

3.1. Data Pre-Processing

Since the dataset as we have seen has substantial number of data imbalance and the inputs can easily get over fitted to the categories with high frequency of data and still give us better accuracy. But what we are trying to achieve is to forcefully teach the model how to train the model and to make sure that the model was able to learn the specific complex structure in the script separately. Inorder to make the model learn the representation properly and to make the model robust to noisy input, we could use hand-crafted augmentations that could produce real-scenario samples of the image which will make the output robust and let the network learn the representation.

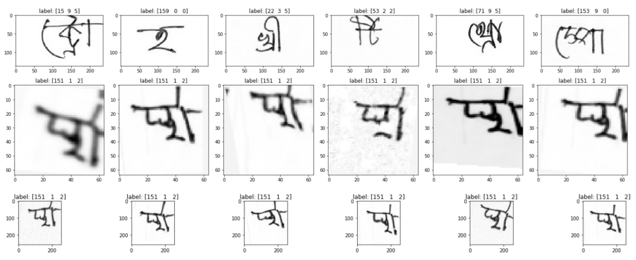


Figure 3: Data samples [1]: Handwritten Bengali graphemes [2]: After affine transformation and resizing [3]: After all agumentations in the dataset.

Affine Transformations : Affine transformation is a key augmentation technique that has been used in this project to produce better results and affine transforms of the image which ensures that many versions of same images panned in many different angles and directions are accounted. These affine transforms are important in OCR models as there are lot of instances in real life where the images have been captured from different perspectives and angles. Inorder to achieve this, I have used library from skimage for affine transforms and wrapping.

Other Augmentations : Other augmentations includes a myriad of transformations and functions. These augmentations are required and are relevant because these helps the model to learn better representation for the grapheme variations. The basic operations that I have done is resizing and cropping as the model input size was limited to 128x128 and since the dataset's inherent size is 137x236. The augmentations that has been included are gaussian blur, random noise, random brightness and contrast, random shift scale rotation, grid distortions in the image and normalization to finally normalize the image between 0 and 1 as a tensor data.

The training dataset available from the Kaggle Bengali AI competition is divided into training and validation dataset in ratio 9:1 and the training dataset has 180756 images and the validation dataset has 20084 images respectively. The dataset is divided into 5 folds(4folds for training and 4 folds for testing) to check the uniformity of the data and the performance in that dataset as discussed in Section 4.

3.2. Model : SE-ResNeXt-50

The choice of the training model was SE-ResNeXt, a variation of ResNet architecture that is used for quick learning outcomes with better learning capacity and adaptability as opposed to vanilla ResNet architecture. Specifically I am using the SE-ResNeXt-50 model which has 50 layers because after training multiple ResNet model with 34,50 and 101 layers, the best model available was the one with 50 layers in terms of trade-off between the learning and the computation cost. The learning was inherently at its capacity with available training dataset. The specifics are available in the Section 4.

ResNet-50 : ResNet[1] was introduced by Microsoft Research to ease the training of deep neural network. ResNet-50 is a 50 layered residual network model that has stacks of deep residual networks in each layers. These residual connections ensures identity mapping and this makes the model easy to optimize even with a deep network and helps to provide a smooth training loss landscape which facilitates better and quick learning when set to learn. Following is the model architecture of ResNet-50.

Table 1: Resnet-50 Architecture

conv1	conv2_x	conv3_x
(7,7),64F stride-2	$\begin{bmatrix} 1\text{x}1, 64 \\ 3\text{x}3, 64 \\ 1\text{x}1, 256 \end{bmatrix}$ x3	$\begin{bmatrix} 1\text{x}1, 128 \\ 3\text{x}3, 128 \\ 1\text{x}1, 512 \end{bmatrix}$ x4

conv4_x	conv5_x	FC	FLOPs
$\begin{bmatrix} 1\text{x}1, 256 \\ 3\text{x}3, 256 \\ 1\text{x}1, 1024 \end{bmatrix}$ x6	$\begin{bmatrix} 1\text{x}1, 512 \\ 3\text{x}3, 512 \\ 1\text{x}1, 2048 \end{bmatrix}$ x3	2048x168, 2048x11, 2048x7	$3.8\text{x}10^9$

ResNeXt : The ResNeXt architecture[5] comes with a homogeneous and multi-branch architecture that has significantly less number of parameters to tune and it is a highly modularized network architecture that has better performance compared to traditional network architectures. This model introduces cardinality, the size of the set of transformations as per the author which is another essential parameter as depth and width for a neural network. Increasing cardinality is as effective as increasing the width or the depth of a neural network, while maintaining the complexity of the network. The basic building block of the ResNeXt is in Figure 4 with cardinality of 32. In our ResNeXt-50, we introduced a cardinality of 32 and bottleneck width as 4 to our ResNet-50 model. The ResNeXt also has stronger representations and hence the model is capable of driving to a much lower training error.ResNeXt-50 has FLOPs of

4.2×10^9 and 25.5×10^6 parameters while for ResNet-50 it is 3.8×10^9 and 25.0×10^6 parameters.

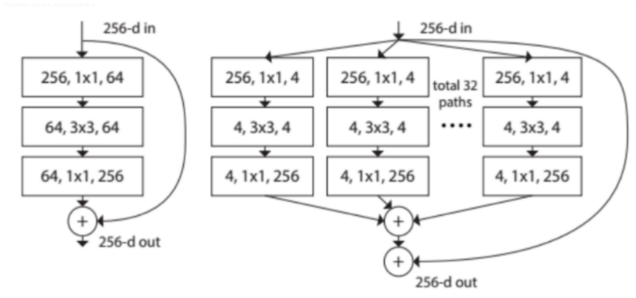


Figure 4: Basic building block for ResNet and ResNeXt architecture

SE-ResNet : Squeeze and Excitation block[6] is a novel architectural unit developed to boost to the representational power of a neural network. Using the SE block, it adaptively recalibrates to channel-wise feature responses by explicitly modelling inter dependencies between channels. This architectural unit boosts the generalization capability of the model extremely well in challenging datasets. SENets are effective because of their ability in modelling channel-wise feature dependencies which boost the performance in all tasks requiring strong discriminative features.

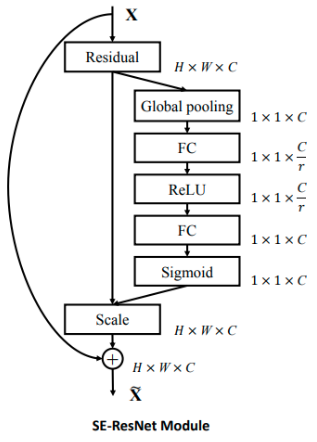


Figure 5: Basic building block for Squeeze and Excitation unit

3.3. Final Model Architecture

The final model architecture is the ResNet model with 50 layers and with a cardinality of 32 and bottleneck of depth 4 with SE blocks attached. This design choice was based on the performance of the model and the to improve the representation ability of the model. As discussed in the previous subsections this particular design choice to include cardinality helps to improve the stronger representational ability and thus achieving lower training error. Squeeze and Excitation(SE) helps with dynamic channel-wise feature recalibration and hence provide us with a better discriminative feature. After the features extraction using SE-ResNeXt-50, the model will flattened and then three branches of fully connected layers, each for grapheme, vowel and consonant detection are derived from the flattened 2048 tensor as shown in the Figure 6. The loss function employed is Cross-Entropy for grapheme root, vowel and consonant softmax output. These losses are added up to get the final loss. Inorder to achieve best performance the model used was pretrained on the ImageNet dataset. The final dully connected layers are removed and modified to achieve the network as described in the Figure 6.

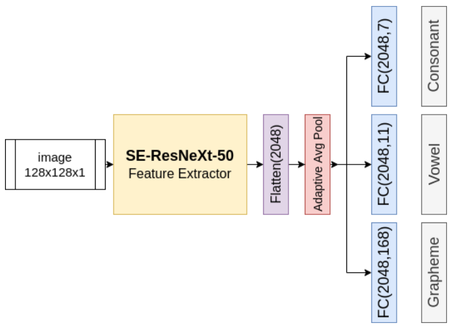


Figure 6: SE-ResNeXt-50 Architecture

3.4. Metric : Weighted Recall

The metric that I am using for this assignment is recall as opposed to accuracy as recall gives us the idea about how many components of each script/ image are correctly identified. Thus by using recall we can measure how many relevant grapheme roots, vowels and consonants are picked correctly from the original label. Let the recall for grapheme roots be $recall_{graph}$, recall for vowel and consonants are $recall_{vow}$ and $recall_{cons}$ respectively. The weighted average recall is given by,
$Score = Recall_{WT} = \frac{2 \times recall_{graph} + recall_{vow} + recall_{cons}}{4}$

As we can see, in this weighted recall metric hereafter refereed as score will be calculated weighing $recall_{graph}$ more as there are 168 classes and retrieval in the grapheme root category is important as opposed to $recall_{vow}$ and $recall_{cons}$. This metric was used by Kaggle for evaluating the submissions.

4. Experiments

4.1. Ablation Study : Different Folds

I have trained the model in different folds to ensure that the model will be able to reproduce similar results and the model will be able to generate better generalization overall. There are only 1292 grapheme variations in the training dataset and there can be unseen data in validation dataset during validation and performance could be worse in such case. So it is important to ensure that our training data is somewhat equally distributed. Since the actual testing data from Kaggle has minimal number of datapoints(11), the valdiations on various folds ensure that the data is almost homogeneous, and it has comparable distribution of data. As described early, the data is divided into 5 folds and trained on 4 random folds and tested on the remaining fold. The model was able to deliver similar results in all 4 folds. Figure 7 is the training and validation score and loss in different folds.

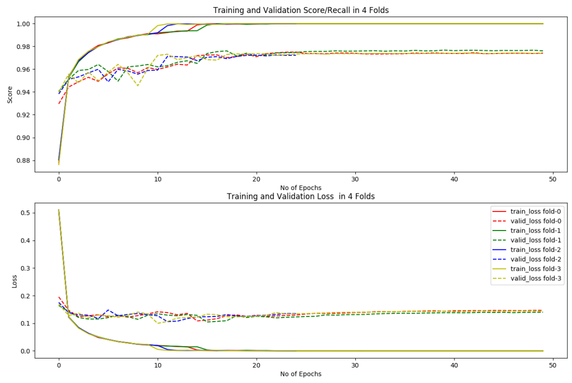


Figure 7: ResNet-50 Architecture: Ablation study on different Folds

4.2. Ablation Study : Size of ResNet

ResNet was easily the choice for the network as it is used as state-of-the-art feature extractor that works well in many cases. It is possible that by using ResNet, we can train the model quickly(even in the transfer learning, which is employed in this implementation) and the representation capacity of ResNet is best. I have chose 3 versions of ResNet for ablation study, ResNet-34, ResNet-50 and ResNet-101 respectively. Figure 8 shows the Loss and Recall Score of these various ResNet networks, the dashed line means the result on validation dataset. Clearly from the data, we can see that the performance of ResNet-50 on the training dataset is superior and it is dominant in the validation dataset. ResNet-50 has significantly lower validation loss and the best validation loss respectively. It should be noted that the ResNet-34, ResNet-50 and ResNet-101 has similar training scores and loss profiles once the model is saturated. ResNet-34 performs worst once it saturates because it does not have the capacity to retain the diversity in information and ResNet-101 has bigger capacity which might result in underexposed neurons in the network and the data is not complex enough to hold the network resulting in subpar loss optimization during training process which results in weaker performance during validation. Figure 9 shows the Accuracy profiles

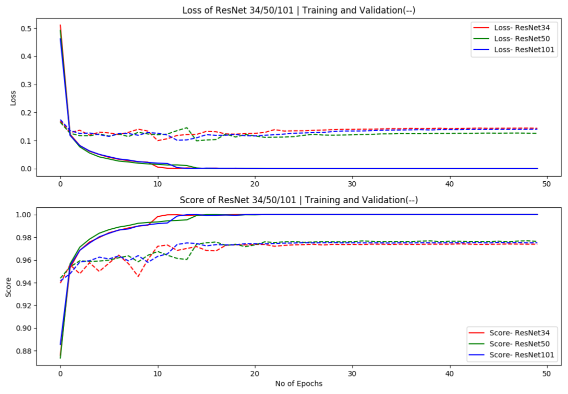


Figure 8: ResNet-34/50/101 Architecture: Loss and Score reports

for various versions of ResNets for grapheme roots, vowels and consonants. As expected, ResNet-50 was able to reach learn quickly when compared to other version of ResNet as expected. ResNet-34 has a slow learning profile, however for ResNet-50 and ResNet-101, we can find a sudden spike in the performance and I am not certain about this effect. The training accuracy of each model is significantly close to each other, but the recall of ResNet-50 is better compared to any ResNet-34 and ResNet-101 which makes ResNet-50 the ideal size for the model to learn grapheme.

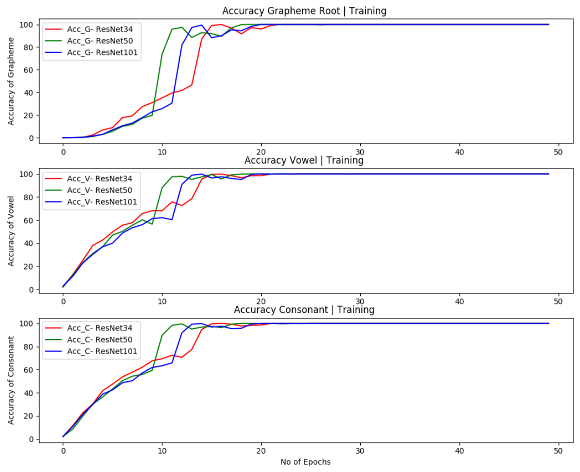


Figure 9: ResNet-34/50/101 Architecture: Ablation study on Accuracy in grapheme root, vowel and consonant.

4.3. Final Results

The model was trained for 80 epochs until the model has been trained to its saturation and started to getting diverge from the results. Among the accuracy that I was able to achieve, the accuracy of grapheme root is substantially less and the performance is subpar compared to vowels and diacritics. This can a be a result of having many classes in the grapheme root, as much as 16 times, compared to the number of classes in vowels(11) and consonants(7) categories.

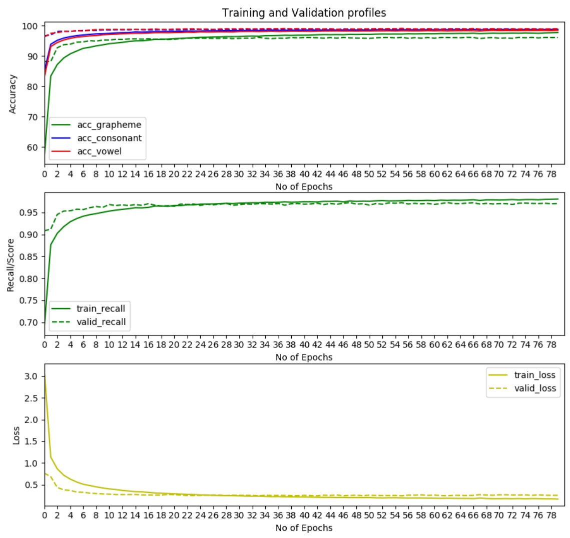


Figure 10: SE-ResNeXt-50 : Training and Validation over Epochs

The best performance achieved at 55th epoch using the model is given below in table 2. We can see that the loss with the grapheme root during validation is twice as much as the training, which means the model was not able to learn the complexity of the grapheme roots or the model there are some unseen datasets in the validation dataset which reduces the model performance drastically. The gap in the performance of the model is visible in the loss gap and the accuracy gap between the training and the validation scores which are 0.06190 and 1.094% respectively and these are directly affecting our final score. Weighted recall/Score is 97.646% for the training dataset and 97.198% for the validation dataset which is satisfactory.

Table 2: Model Performance @

Metric	Training	Validation
Score	97.646	97.198
Total Loss	0.19259	0.24163
Loss Grapheme	0.09303	0.15493
Loss Vowel	0.05531	0.04258
Loss Consonant	0.04424	0.04412
Accuracy Grapheme	97.190	96.095
Accuracy Vowel	98.271	99.064
Accuracy Consonant	98.599	98.840

5. Discussion

The model is performing really well with our current choice model architecture and the learning of the model is remarkably good, as learning saturates and reached peak performance before 50 epochs as we are using pretrained ImageNet model for training the network. Transfer learning, introducing cardinality and SE blocks helps us to increase and boost the learning capacity and performance of the model. However, we there is still room for improvement, especially in the performance in the classification of grapheme roots. There is a significant gap between the accuracy obtained from training and validation data which suggests that there is room for improvement especially if unseen variations are available. A possible solution will be custom loss function for this class. Using Additive Angular Margin Loss known as ArcFace[7] can boost the performance of the network.

Another method is to use the dataset and divide it into Seen and Unseen categories. As discussed early, there are 1292 combinations in training dataset. It can be splited to 1162 for Seen dataset and 130 for Unseen dataset respectively and use EfficientNet for distinguishing whether the input image is a Seen class or an Unseen class. This helps with out of distribution data. Two EfficientNet models were used for detecting unseen data and predicting classes if it is in seen dataset. If data is from unseen distribution, prediction from cycleGAN trained with cycle consistency loss will be used.

6. Conclusion

The SE-ResNeXt-50 model has been able to learn the model very well.Performance boost from the Squeeze and Excitation block and by introducing cardinality helped to achieve best performance using ResNet architecture. There is still room for improvement in this model especially to boost the classification of grapheme roots. Through this project I was able to gain skills to tweak and modify the network architecture with relevant modules and blocks to boost the performance and to push the results to state-of-the-art in the competitions like Kaggle. The first place solution is Cyclegan Based Zero Shot Learning which is novel in many aspects.

7. Reference

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; "Deep Residual Learning for Image Recognition" CVPR, 2015
[2] Christian Bartz, Haojin Yang, Christoph Meinel; "STN-OCR: A single Neural Network for Text Detection and Text Recognition", 2017
[3] Cheng-Lin Liu, Fei Yin, Da-Han Wang, Qiu-Feng Wang; "Online and offline handwritten Chinese character recognition: Benchmarking on new databases" 2013

[4] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng; "Focusing Attention: Towards Accurate Text Recognition in Natural Images", ICCV, 2017
[5] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He; "Aggregated Residual Transformations for Deep Neural Networks", CVPR, 2017
[6] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu; "Squeeze-and-Excitation Networks", CVPR, 2019
[7] Jiankang Deng, Jia Guo, Niannan Xue, Stefanos Zafeiriou; "ArcFace: Additive Angular Margin Loss for Deep Face Recognition", CVPR, 2019
[8] https://www.kaggle.com/c/bengaliai-cv19

Appendix

All relevant codes used for this experiment to reproduce the result is attached as zip file.