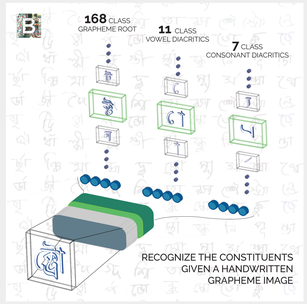
# Bengali.AI Handwritten Grapheme Classification

Gary Ong (1002758)
Peng ShanShan (1002974)
Nashita Abd Guntaguli (1003045)
Lu Jiankun (1002959)
Joseph Chan (1002948)
Hong Pengfei (1002949)

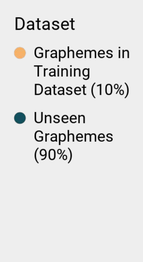
## Problem Statement

Bengali is spoken by hundreds of millions of people, making it the 5th most spoken language in the world. As such, achieving reliable optical handwritten character recognition algorithm can result in advancements in business and education that are extremely beneficial for this population. However, character recognition has proven to be very challenging as the language comprises of nearly 13000 possible graphemes, compared to the 250 graphemes of English.

***Based on a given image of a handwritten Bengali grapheme, to separately classify three constituent elements in the image: grapheme root, vowel diacritics, and consonant diacritics.***




## DataSet

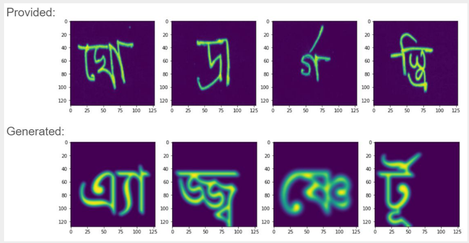



To gauge the performance of our model during training, we need to be able to assess it on both seen (part of the 1295 unique graphemes) and unseen (not part of the 1295 unique graphemes) graphemes. Thus, we decided to remove a small sample of combinations from the training set and use them to simulate unseen data. The remaining data is split into
training and validation set.
The final split is as follows:

Training set: 77%
Validation seen set: 20%
Validation unseen set: 3%

## Data Generation

We have also generated new training images that are utilized in some experiments. This has the potential to help the model learn features of graphemes that would otherwise be unseen. We expanded our data set to slightly more than 3000 unique graphemes, up from the initial 1295. For each unique grapheme, we generated text images using different fonts. We applied gaussian blur to reduce the sharpness of the edges and improve generalisation, so that the model is able to expose itself to unseen graphemes and learn to recognize overall topology of each grapheme.

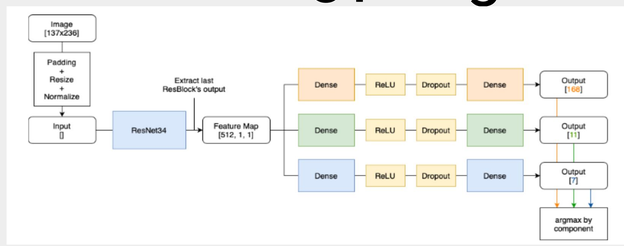



## Baseline Model

Our baseline model uses resnet18 pretrained from imagenet.The last fully connected layer has 186 neurons in total, which are splitted into three parts, 168 for grapheme root classification, 11 for vowel diacritics classification and the rest 7 for consonant diacritics classification. We selected resnet18 as our baseline as it performs the best amongst all other models..

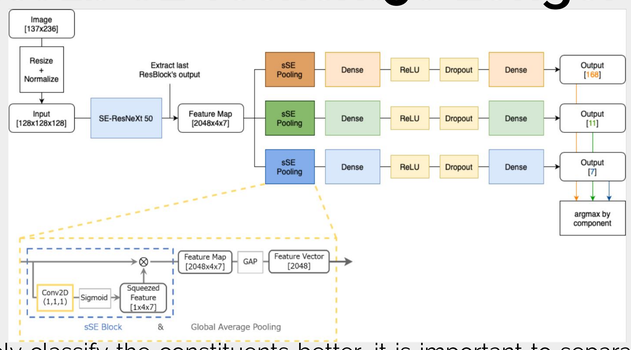
| Model | Seen Accuracy | Unseen Accuracy |
|---|---|---|
| Resnet18 | 0.9465 | 0.8391 |
| GoogleNet | 0.9442 | 0.8285 |
| MobileNetV2 | 0.934 | 0.812 |
| VGG16 | 0.9327 | 0.8492 |
| AlexNet | 0.8284 | 0.7205 |
| Resnet18 /w Gen Data | 0.9433 | 0.8247 |

## Model 1a Resnet34 Longer Tails



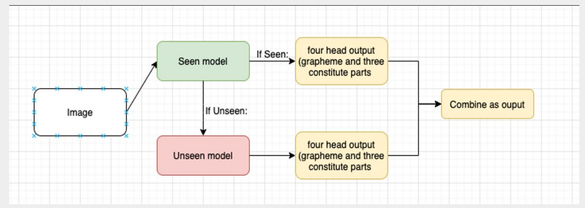

The baseline model only splits the tails at the last layer. It may be too short to effectively learn how to separately classify the three constituents with only one layer of splitted tails. Hence, we added more Dropout and Dense layers after the split.

## Model 1b SE-resnext50 Longer tails



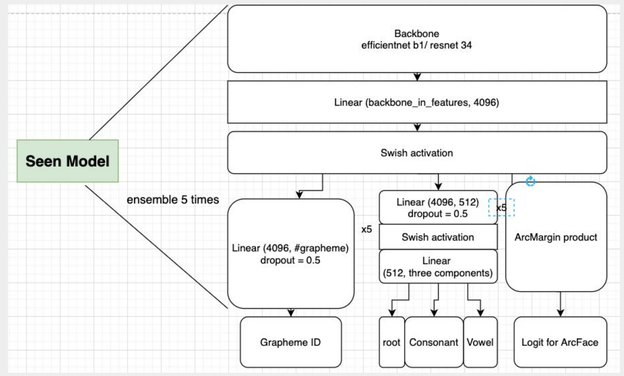

To separately classify the constituents better, it is important to separate not only the classification tails, but also the feature extraction part. Hence, before passing the CNN feature maps into classification tails, instead of doing global average pooling, we pass the last feature map (2048 4 4) to 3 sSE blocks.

## Model2: Seen and Unseen



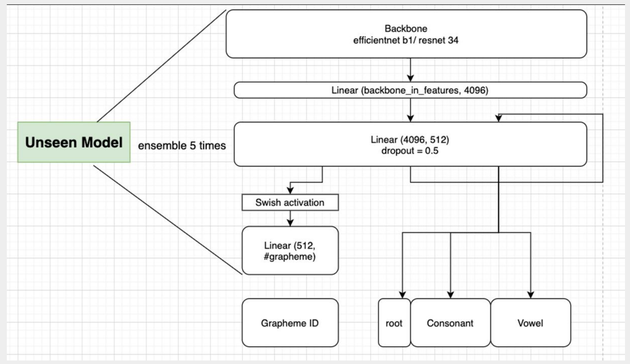

This model adapts a 2-stage prediction process using two models, one model is used to predict seen images and one model is used to predict unseen images. Seen model will output the result for the grapheme as well as its ArcFace Loss which can be used to classify whether the grapheme has been seen before. If the Arcface loss classify the grapheme as unseen before by the model, we will feed it to the unseen model and use its prediction as final prediction result. The intuition behind this model is that due to a lot of the graphemes in the test dataset have not appeared in the training

## Seen Model




We used efficient-net as our backbone for the model, and used Swish activation for more versatile activation after the last linear layer. Then we ensembled the results of dropout by taking the average of the results each time after the dropout then adding another activation and dropout. We put more layers in the head than unseen models because this way will allow us to enable the model to balance between different kinds of features. Although grapheme id is not needed as a final result, since we have the label, we used it to co-train the model hoping that it can train better feature extractor.

## Unseen Model




Grapheme is output as a combination of its three components rather than previously in Seen model we consider a complex interaction by the output features of the 'body' efficient net in our case. The intuition behind this is that we would like to put our focus on predicting the correct component of the grapheme rather than grapheme itself, therefore unlike the seen model, we don't model grapheme separately using its own MultiLayer perceptrons. We also added cutmix augmentation which allows the model to learn better localizable features.

## Results

| Model | Private | Public |
|---|---|---|
| ResNet18 Baseline | 0.8926 | 0.9529 |
| ResNet34 LongerTail 34 epoch | 0.9043 | 0.9605 |
| ResNet34 LongerTail 69 epoch | 0.9027 | 0.9636 |
| ResNet34 LongerTail 104 epoch | 0.9023 | 0.9641 |
| ResNet34 LongerTail SnapshotEnsemble | 0.9084 | 0.9661 |
| Se-resnext LongerTail 30 epoch | 0.9176 | 0.9743 |
| Se-resnext LongerTail SnapshotEnsembe | 0.9247 | 0.9737 |
| Seen and Unseen Model | 0.9350 | 0.9814 |