Bengali.AI Kaggle Competition

Luuk Hesselink
Joeran Bosma
Anonymous

March 20, 2020

Introduction

This report contains an explorative analysis of several properties of convolutional neural networks (CNNs) applied to decipher Bengali graphemes. The source code that is used to study the properties of the CNNs, is available at `https://github.com/joeranbosma/Bengali`.

1 Classification task

The task at hand is to classify handwritten Bengali graphemes as hosted by Kaggle[1]. Each grapheme can be decomposed to its root (of which there are 11 vowels and 38 consonants), and possible diacritics. As the combination of these features results in ∼13,000 different grapheme variations (compared to English's 250 grapheme units), this classification task is challenging.
In this competition, this diversity is tackled by converting the detection of these graphemes to the detection of its constituents. More specifically, the competition requires the classification of the root (168 possibilities), the vowel diacritic (11 possibilities) and the consonant diacritic (7 possibilities).

2 Baseline model

To accelerate meaningful analysis of the problem at hand, a baseline model was taken from the competition's open-source notebooks[1]. This model and accompanying data preprocessing and augmentation setup achieved a leaderboard score of 0.9506. The score is the average of the macro recalls, where the root class weighs double.

3 Implementations

Overview of different implemented strategies, results and discussion. The models are trained on 7/8th of the train data and validated on the same 1/8th, unless stated otherwise. We used the online leaderboard score as test score.

3.1 Data preprocessing and augmentation

To allow for easier extraction of useful features from the input images, a new data preprocessing pipeline was implemented. The preprocessing consists of automatic selection of the region of interest, which can be performed since the data consists of uniform backgrounds from which the drawn symbol stands out. Additionally, the images are normalised using z-score normalisation and padding with a width of 5 pixels is applied to prevent too much loss of data during augmentation and convolutions.
To help the model generalise better to unseen data, the training images are augmented with random rotations and shearing. The shearing and rotation render some parts of the characters outside the padding, but tests showed that this does not negatively influence the validation score, as can be seen in fig. 4 of Appendix C. A more detailed overview of all pre-processing settings is explicitly included in table 1 of Appendix A.

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

3.2 Cyclical Learning Rate with momentum

Following two papers by L.N. Smith[2, 3], we implemented cyclical learning rates for training our convolutional neural network. A cyclical learning rate scheme implements a learning rate varying between two bounds, first linearly increasing to the upper bound and subsequently decreasing to the lower bound, as shown in fig. 1 (left). This scheme is set up to initially follow the large gradients without blowing up, and increase the learning rate gradually as the gradients decrease in magnitude. After the fast convergence due to the large learning rate, the learning rate is decreased to find a good local minimum, in a similar manner as simulated annealing. This example also shows an additional decreasing phase to 10% of the lower bound, to ensure convergence.
The papers claim that this approach also has a regularising effect and practically eliminates the need to tune the learning rate, while achieving near-optimal classification accuracy. Due to the fast convergence, less epochs are required and overall training time can be reduced.
Figure 1 (right) shows a comparison between the global validation accuracy of the base network with Adam optimiser and the same network with SGD optimiser with cyclical learning rates and Nesterov momentum of 0.75. The increase in performance after 50 epochs is 0.34%, with the model trained with cyclical learning rates performing best (97.66%).

Figure 1: (left) Example of a cyclic learning rate scheme. First, the learning rate increases linearly from a lower bound to an upper bound. Then, the learning rate decreases back again from the upper bound to the lower bound. Afterwards, there is a small extra decrease in learning rate to about 10% of the lower bound in order to ensure convergence to a local minimum. (right) Performance of training the CNN with either the Adam optimiser or employing a cyclical learning rate. The latter uses the mini-batch stochastic gradient descent (SGD) optimiser with Nesterov momentum of 0.75.

3.3 Ensemble Classifier and Region of Interest size

To improve upon the performance of a single model, predictions from multiple models can be combined in an ensemble. Combining multiple models can lead to the cancellation of bias of the individual models, similar to cancellation of random errors. To achieve this, the networks need to be different from each other, for example by training the models on different cross-validation folds. Testing the performance of the ensemble was combined with investigation of the Region of Interest (ROI) size.
To this end, the baseline architecture (OrigNet in fig. 2) and the EfficientNetB0 architecture were trained with image input sizes of 42x42, 74x74 and 106x106 pixels (ROI 32-64-96 and padding 5-5-5).

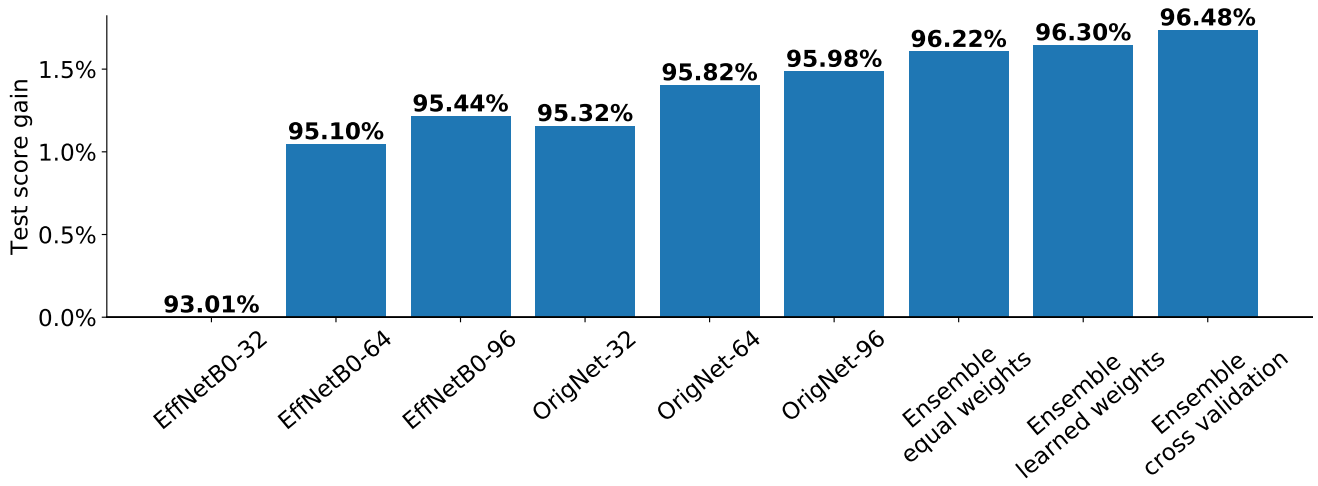

Figure 2: Performance of baseline architecture (OrigNet) and EfficientNetB0 for several ROI sizes, all with an additional padding of 5 pixels. The ensembles with equal and learned weights are combinations of the six aforementioned models. The final, rightmost ensemble consists of eight models trained on the different cross-validation folds. Shown is the public leaderboard score for the ensemble models and OrigNet with ROI size of 96. The score of the leftmost five models is calculated as the test score of OrigNet-96 minus the difference in validation accuracy.

3.4 Feature Selection Analysis

The models were observed to perform better with images of higher resolution. However, since the number of connections with the dense nodes also grows with the image size, a test setup was created to determine whether the increase in performance could be explained by the increase in capacity of the dense part, or due to the feature selection functioning better with higher-resolution input images. To this end, two tests were performed. The CNN performance with input image size 32x32 was compared to the CNN performance with input image size 64x64, where the bigger image has an additional 2x2 max-pooling layer between the feature selection and dense layers. This results in the same number of model parameters. The experiment was repeated by comparing image input resolutions of 64x64 with 128x128. The coincidental effect on performance of adding the max-pooling layer was not isolated in this test, yet this change is expected to be small. Future studies should investigate this effect further. Figure 3 shows that the feature extraction performs better when the resolution is increased from 32x32 to 64x64. Performance when increasing the input size to 128x128 did not increase. The underlying reason has not been investigated further.

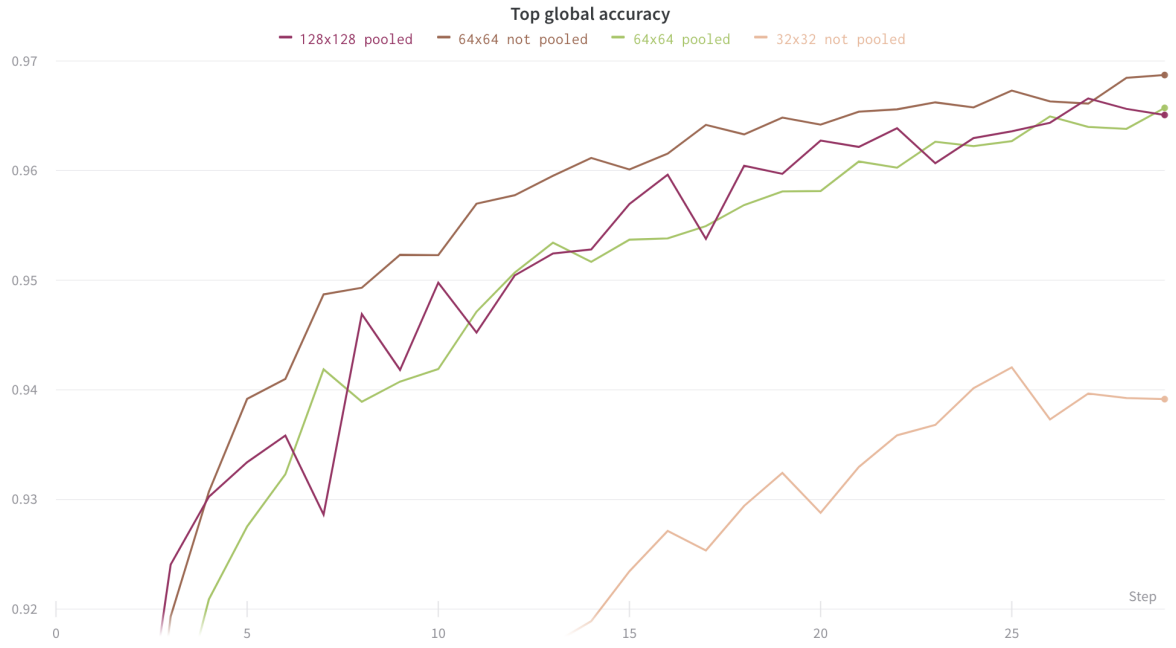

Figure 3: Comparison of feature selection performance of image input size: 32x32 with 64x64, and 64x64 with 128x128.

4 Conclusion & Recommendations

Starting from a baseline model from the Kaggle competition, an improved model is developed to classify handwritten Bengali graphemes. A new preprocessing pipeline is introduced, featuring automatic selection of the region of interest, leading to better model performance. Furthermore, data augmentation using rotation and shearing is implemented, which improves generalisation of the models. Implementation of a cyclical learning rate scheme improved performance significantly. Analysis into the effect of feature selection showed that increasing image resolution improved model performance, while keeping the number of model parameters the same. Additionally, an ensemble classifier of models trained on different cross-validation folds (each using 7/8th of the complete dataset) showed the best overall performance, with a Kaggle test score of 0.9648, yielding a leaderboard position of 1161/2046 participating teams. After the final evaluation of the private leaderboard on 46% new test data, the submission received a test score of 0.9096 with a position of 1325/2059 participants. Possible explanations for the difference could be poor generalisation due to overfitting on the training data.
Future research should investigate whether the performance of the ensemble can be improved by adapting the confidence in the predictive capabilities of a model based on the model performance on a specific class. These improvements are intended to be implemented in the second Kaggle competition of this course.

References

[1] Baseline open source Jupyter notebook in Kaggle Bengali.AI Handwritten Grapheme Classification competition. `https://www.kaggle.com/kaushal2896/bengali-graphemes-starter-eda-multi-output-cnn`. Accessed: 2020-02-20.

[2] Leslie N. Smith. Cyclical learning rates for training neural networks, 2015.

[3] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates, 2017.

Appendices

A Data augmentation settings

All models mentioned in this report are trained with data augmentation settings listed in table 1.

Table 1: Data augmentation settings used for training all models, disabled during testing

Variable	Value	Meaning (if true)
rotation_range	8	Max degrees of random image rotation
zoom_range	0.15	Max random zoom of image w.r.t. height, width
width_shift_range	0.15	Max random shift horizontally w.r.t. width
height_shift_range	0.15	Max random shift vertically w.r.t. height
horizontal_flip	False	Randomly flip images over horizontal center axis
vertical_flip	False	Randomly flip images over vertical center axis
shear_range	8	Max degrees of shearing intensity (counter clockwise)

B Details ensemble models on all cross-validation folds

Training models on all of the eight different 7/8th cross-validation folds was combined with small exploratory experiments to seek further improvements. This resulted in many different settings for these models. The default settings are described in table 2, deviations for the individual models are described table 3.

Table 2: Default settings for the models in the ensemble with all cross-validation folds

Variable	Value	Description
Epochs	55	Number of complete pass-throughs of training data
ROI size	96	Region of Interest size, in pixels
Padding	5	Padding around the ROI, in pixels
Cross validation parts	8	Number of cross-validation splits. One part validation, rest train
Optimiser	SGD+CLR	Mini-batch Stochastic Gradient Descent with Cyclical Learning Rate
Momentum	0.75	Nesterov momentum for the SGD optimiser
Learning rate min	0.1	Lower bound of learning rate
Learning rate max	1.221	Upper bound of learning rate
Converge epochs	7	Number of epochs toward 10% of lower bound learning rate
Dropout rate	40%	Dropout rate of baseline model (OrigNet)
Pooling	MaxPooling	Type of pooling layers in the baseline model (OrigNet)

Table 3: Deviations from default settings described in table 2

Cross-validation fold	Variable	New value
0	N/A	No deviation
1	Architecture	EfficientNetB0
	Epochs	60
	Converge epochs	12
	Learning rate min	0.03
2	Learning rate min	0.03
	Learning rate max	1.4
	Dropout rate	40%
3	Learning rate max	1.3
4	ROI size	64
	Learning rate min	0.03
5	ROI size	64
	Pooling	AvgPooling
6	ROI size	64
	Learning rate min	0.03
	Converge epochs	12
7	ROI size	64
	Learning rate min	0.03
	Learning rate max	1.4

C Additional Figures

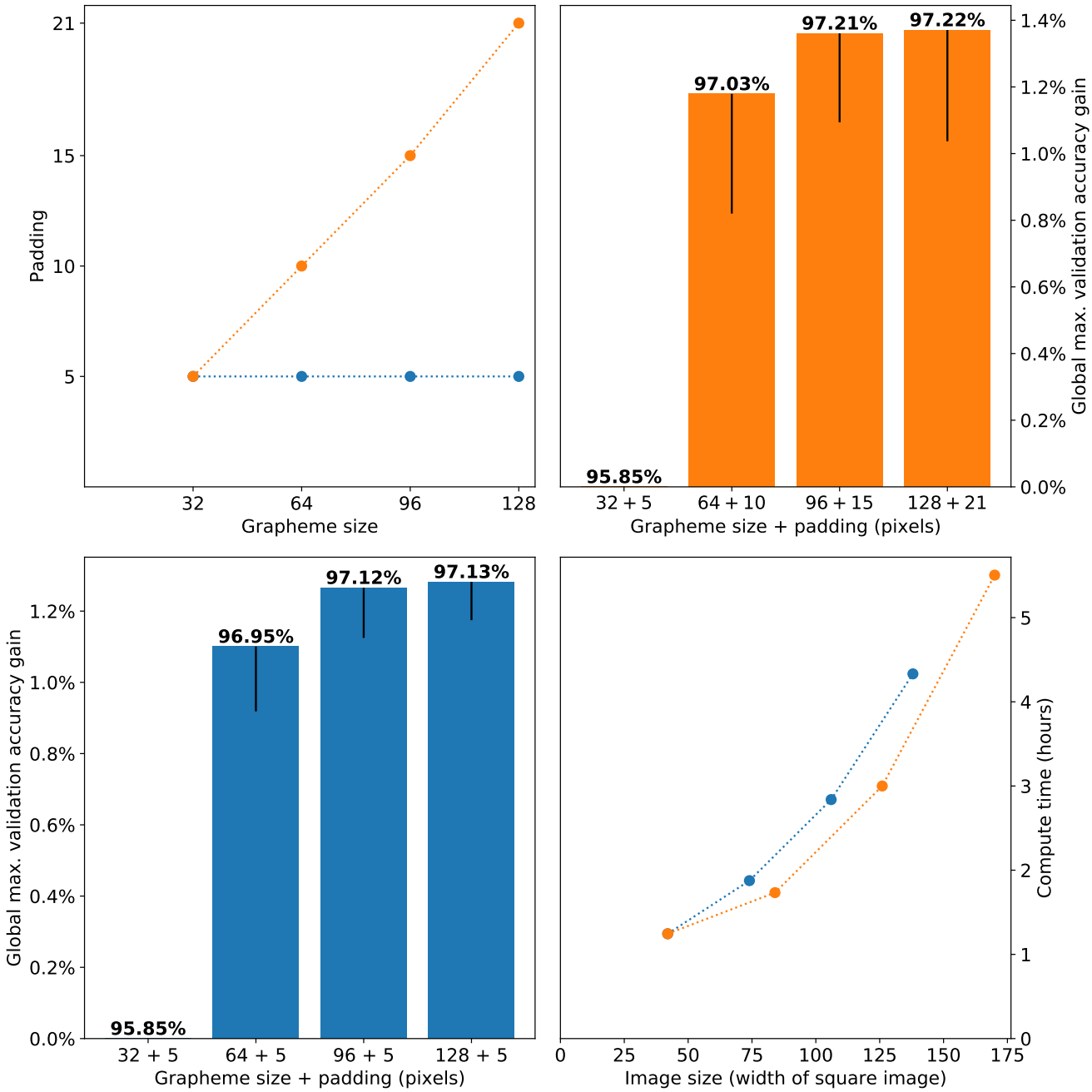

Figure 4: Maximal global validation accuracy achieved for several Region of Interest (ROI) sizes and padding widths. The bars show the maximal global validation accuracy between epoch 25 and 30 for each setting. The y-error shows the range between the minimal and maximal global validation accuracy between 25 and 30. As can be seen in the figure, the maximal global validation accuracy increases both for larger ROI size and larger padding width. However, the mean global validation accuracy of epoch 25-30 is not significantly different for larger padding width, but the deviation is smaller for the models trained with less padding (not shown). The trade-off with required compute time is shown in the lower right panel.