

Devanagari Handwritten Character Recognition with Convolutional Neural Network

Asaf Gazit, Xin Li

Introduction	
Motivation: Handwritten character recognition has been a popular research area for many applications. India is a multi-lingual country and Devanagari is the most popular script [1,2]. It is valuable to build a model to read Devanagari handwritten characters efficiently so it may benefit society and businesses alike.	Model: Convolutional neural network is a favourable method on handwritten character recognition and it already has many commercial applications [4]. It is inspired by the visual cortex (neuroscience) and usually consists of convolutional and pooling layers as the main architecture and a few dense fully connected layers (a classifier) at the end. The patches of units in the previous feature maps feed into the convolutional layer which passes the result to the next layer. All units in the convolutional layer shows the same weight thus tolerance to variants in location to a degree. Pooling layers merge similar features into one. Two or three convolutional and pooling layers are stacked and followed by fully connected layers for classification. Backpropagation is used to adjust all the weights. [5]
Dataset: The dataset contains 36 consonants and 10 numeral characters, 46 Devanagari characters in total. 2000 handwritten images for each character. 92000 instances in total split into training (85%, 78200) and test sets (15%, 23000) [3].	Model Pros: <ul style="list-style-type: none"><li>High accuracy in practice</li><li>No need for feature extraction</li><li>Tolerance to position variant</li></ul> Model Cons: <ul style="list-style-type: none"><li>Overfitting may occur when training dataset is small</li><li>Can be computational expensive</li><li>Difficult to comprehend or interpret</li></ul>

Grid Search Parameters

Image resolution (32×32 vs. 28×28) (implemented but not in final results), image binary threshold (black&white 95 vs. Grayscale), Batch size (2000 vs. 4000), Max epochs (10 vs. 20), and training size (78200 vs. 19550) are iteratively adjusted for optimal configuration. Learning rate is 0.4 and learning decay is 0.0002. The original image size is 32×32 pixels and [1] is rescaled to 28×28 to compare both sizes. The image is white character with black background. The grayscale of some character edge pixels is of value 95. We have applied a binary threshold of 95 (to transform the image to black and white) and compare that to the original grayscale. The minimum batch size is set at 2000 to enable each character is passed through in each batch. Article [1] shows different accuracy level with epochs from 10 to 50. However, there is only marginal gain after 20 and thus 10 and 20 is compared in this study. The size of training size is also varied. Figure 1 shows the architecture of the convolutional neural network. Stochastic gradient descent (SGD) is employed consistent with [1] providing [2] presented that [1] achieved one of the highest accuracy among five studies on this domain using different learning algorithms.

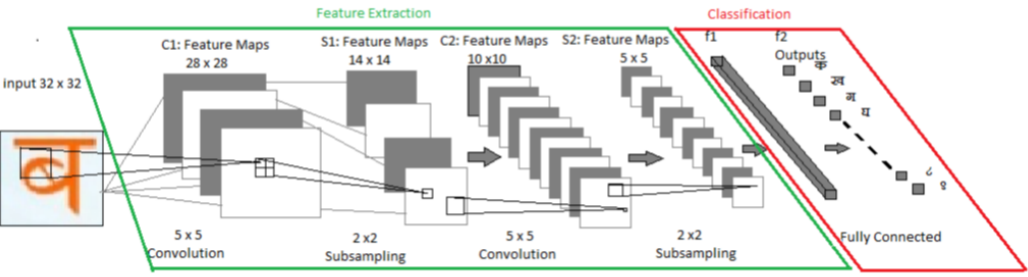


Figure 1. An example of ‘LeNet-5’ Convolutional Neural Network for character classification, from [1]

Notes, Challenges & Issues	
This coursework was implemented on IBM's DSX as it requires the OpenCV and BigDL packages that are not available on City University's servers. During this implementation, we have encountered two issues:	together included casting to the BigDL Classifier objects (DLEstimator and DLClassifier). Those are designed for wrapping BigDL's classifiers for the use of a pipeline. However, our attempts showed that they cannot be used with a neural network, at least not at moment. We have tried numerous versions of BigDL (current one in 0.5.0) as well as altering our datasets, changing our image loading methods (such as BigDL's DistributedImageFrame) to try and fit the “Fit()” function. We have tried to create our own Transform object to make our process fit the pipeline. Unfortunately, all of our attempts to use the “Fit()” functions have failed. We have resorted to implementing our own grid search to look for the best parameters for a BigDL Neural Network.
1) Spark Clash / Optimiser Failure - This issue that has surfaced in Lab 8 has prevented the optimising of the neural network. The generic error message does not provide clear direction for investigation (TypeError: 'JavaPackage' object is not callable). But with the help of IBM's support we managed to locate the source of the error. The error originates as one of BigDL's prerequisites is PySpark. Upon installing BigDL, it had also installed PySpark. As DSX projects are associated with the Spark engine service of DSX, this created a clash of services. A snippet of code that shows whether this issue consists in the DSX project associated, including handling uninstalling and reinstalling a "dependence free" BigDL is included in the coursework notebook. This solution was also tested on Lab8 (MNIST) solution and was successful.	3) Run Time, Computational Resources Constraints & Unstable DSX – DSX runtime was inconsistent. This aspect became especially obvious when executing long computations. During the period allocated for testing, the DSX platform suffered outages rendering the service unusable. Slowdowns of the runtime, sometimes to a complete halt, partially may be also attributed to the limit of 6GB RAM on DSX's workers. Due to the above and our time restrictions, in the final run we reduced the training size to 15640 and 31280 (20% and 40% of training set respectively). Also, we rescaled images resolution to 28×28 without comparison.
2) Spark Pipeline & BigDL Optimiser - In common packages (SKlearn, Keras and even most classifiers of BigDL) training, or fitting, a classifier usually calls a “Fit()” function. However, we would like to note that the Sequential model object that BigDL deploys uses “Optimize()”. Those functions do not seem to overlap. Our attempts to make those work	

Results

The performance of the model is measured by the top1 accuracy rate (i.e. the most probable predicted label vs. ground truth). The performance of the 16 scenarios is presented below in Figure 2.

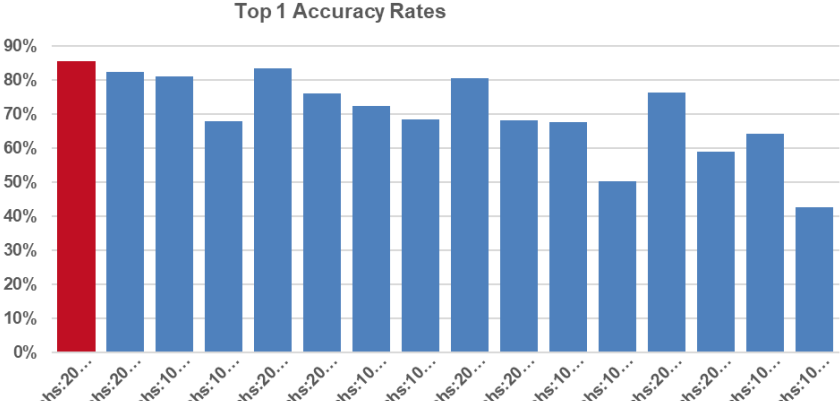


Figure 2. Top 1 Accuracy Rate Comparison among 16 Experiments

The top 1 accuracy rate curve of the best model is shown below. Y-axis represents accuracy rate and X-axis represent the number iterations. In addition, the 46×46 Confusion Matrix of the best model is presented below [Figure 4]. A larger version is included in the notebook of this coursework.

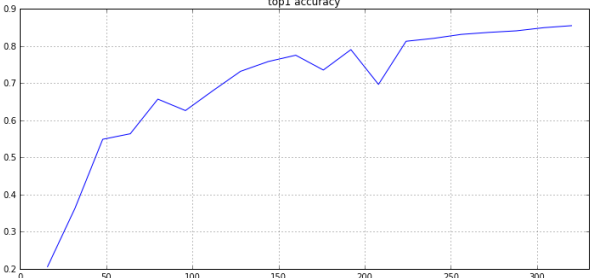


Figure 3. Top 1 Accuracy Curve of the Best Model

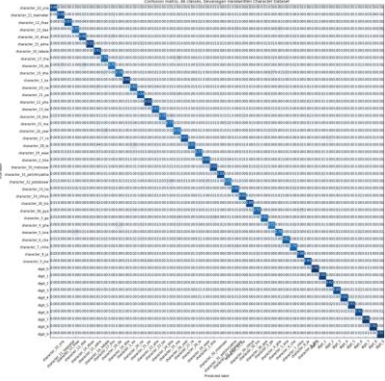


Figure 4. Confusion Matrix

The weights extracting of the first and second convolutional layers are shown below:

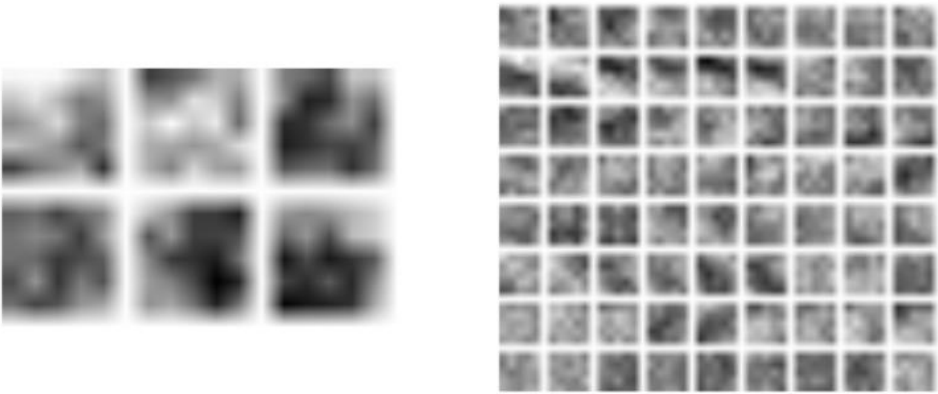



Figure 5. Extracting of Weights: the first (left) and the second (right) convolutional layers

Figure 2 shows that CCN with 31280 training samples, 28×28 resolution, grayscale, 250 batch and 20 epochs gives the highest top 1 accuracy rate of 86%. And a few examples of prediction and ground truth are shown on the right.

Ground Truth labels:  
19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0  
Predicted labels:  
19, 32, 14, 19, 19, 19, 19, 19



Conclusions		References
Based on the grid search of this study, the impact of different parameters is summarised into two categories. And the saturation represents their scale of impact on accuracy rate.		<div>[1] Acharya, S., Pant, A.K. and Gyawali, P.K., 2015, December. Deep learning based large scale handwritten Devanagari character recognition. <i>Software, Knowledge, Information Management and Applications (SKIMA), 2015 9th International Conference</i> (pp. 1-6). IEEE. [2] Pal, U., Wakabayashi, T. and Kimura, F., 2009, July. Comparative study of Devnagari handwritten character recognition using different feature and classifiers. <i>Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference</i> (pp. 1111-1115). IEEE. [3] Archive.ics.uci.edu. (2018). UCI Machine Learning Repository: default of credit card clients Data Set. [online] Available at: <a href="https://archive.ics.uci.edu/ml/datasets/Devanagari+Handwritten+Character+Dataset">https://archive.ics.uci.edu/ml/datasets/Devanagari+Handwritten+Character+Dataset</a> [Accessed 21 Apr. 2018]. [4] LeCun, Y. and Bengio, Y., 1995. Convolutional networks for images, speech, and time series. <i>The handbook of brain theory and neural networks</i>, 3361(10), p.1995. [5] LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. <i>Nature</i>, 521(7553), p.436.</div>
Parameters have <b>positive</b> impact on accuracy: <ul style="list-style-type: none"><li>Increase in training set sample size</li><li>Increase in number of epochs</li><li>Convert image to grayscale</li><li>Keep original resolution (32×32)</li></ul>	Parameters have <b>negative</b> impact on accuracy: <ul style="list-style-type: none"><li>Increase in batch size</li><li>Convert image to black &amp; white</li><li>Change resolution (28×28)</li></ul>	