Optical Character Recognition System for Nepali Language Using ConvNet

Bidhan Bhattarai[1], *Manish K. Sharma*[1],

bhattaraibidhan@gmail.com, manishksharma50@gmail.com

Abstract

This paper describes the implementation of CNN (Convolution Neural Network) based Optical Character Recognition System for Nepali Language, a commonly spoken language in Nepal. The system has been developed in python using Keras[1] library on top of Theano[2] and numpy[3]. The system has been trained using a set of real world[4] and synthesized data sets considering various noise conditions. The tests have also been carried out in a similar setup. The system is further enhanced by using nGram based language models. This paper details the experiment by discussing the concept, implementation details and overall interpretation of the system.

Keywords: Optical character recognition, CNN, ANN, deep learning, Nepali language, parallel image processing, CUDA, GPU, line segmentation, word segmentation, character segmentation, artificial data synthesis

1. Introduction

Optical Character Recognition is a system that translates images of handwritten or printed texts to machine-editable electronic texts which can then be used for tasks like assisting visually impaired users, making printed literary works searchable, extracting information from business documents among many other tasks. This system is composed of line segmentation, word segmentation, character segmentation, classifier, and language model.

There has already been significant progress in the field of OCR for English texts. Some systems are even capable of accurately extracting texts from pictures[5]. But, there has been no noteworthy attempt in doing so for Nepali texts. This project aims to make an attempt in contributing to the field by using advanced learning algorithms on GPUs.

Building an Optical Character Recognition for Nepali texts is a challenging task mainly for two reasons. First, there are much more characters in Nepali than in English – including all attachments and combinations, there are more than 400 distinct characters that need to be recognized and classified correctly. Second, the characters are joined. To tackle these challenges, this project employs deep learning technologies, advanced character segmentation scheme, parallel image processing, etc.

The presented system uses histogram based approach for accurately segmenting lines, words, and characters. A deep convolutional neural network has been added to the pipeline to perform the main character classification task. The system has been designed in such a way that a language model can be integrated to fine tune the results.

The paper is divided into five sections. Section 1 gives Introduction, Section 2 describes the literature Survey, Section 3 focuses on the Nepali Character Set, Section 4 deals with Methodology along with Section 5 discussing Result and Analysis followed by Section 6 with the Conclusions and 7 with References.

2. Literature Survey

Nepali Optical Character Recognition is a field where significant research hasn't been carried out. The situation exacerbates when we consider the deep learning approach to the problem. However, we laid our hands on a few papers that have tried to shed some light on the problem.

The paper "Deep Learning Based Large Scale Handwritten Devanagari Character Recognition" by Acharya et al.[4] uses a relatively small dataset of handwritten character set to classify 46 characters. This paper introduces the use of Convolutional Neural Network for the classification. However, this project doesn't address the characters obtained from the combination of consonants and vowels which are more prevalent in the Nepali Language. The paper also does not address the process of character localization or segmentation from words in a document.

Another related paper is "Optical Character Recognition for Nepali, English Character and Simple Sketch Using Neural Network" by Shakya et al. [6] This paper suggests the use of Artificial Neural Network with back propagation with two hidden layers for classification of Nepali string with adaptive learning rate.

In our research, we have used Deep Learning approach using Convolutional Neural Network as this approach has proved to be promising in the field of Optical Character Recognition. We have used real world data set as well as synthesized data set to obtain a large amount of data to minimize the chances of overfitting. Also, our system is capable of recognizing not only the vowels or consonants but the combination of both in the form of consonant and 'matras'.

3. Devnagari(Nepali) Character Set

Nepali Language is written in Devanagari Script. There are 12 vowels in general with 36 consonants in this language. The combination of vowels with consonants gives rise to a large number of characters that combine to give words. But, unlike the English Language, a combination of a consonant with a vowel give rise to a new symbol.(क + आ = का) . Such added symbol is commonly referred to as *Matra*. The vowels and their matra are shown in Table 1 . Similarly, the consonants are shown in Table

Table 1: Vowels and Corresponding Matras Used in Nepali Language

Vowel	अ	आ	इ	ई	उ	ऊ	ए	ऐ	ओ	औ	अं	अः
Matra		ा	ि	ी	ु	ू	े	ै	ो	ौ	ं	ः

2.

Table 2: Consonants of Nepali Language

क	ख	ग	घ	ङ
च	छ	ज	झ	ञ
ट	ठ	ड	ढ	ण
त	थ	द	ध	न
प	फ	ब	भ	म
य	र	ल	व	श
स	ष	ह	क्ष	त्र
ज्ञ				

4. Methodology

4.1. Dataset Preparation

In Machine Learning Applications, data set can be obtained by getting samples that best represent the problem domain from the real world or by generating the samples having a close resemblance to the real world data. Thus, in this project we have created our dataset in the following ways:

4.1.1. Real-World Data Samples

Real-world dataset was obtained from Devanagari Handwritten Character Dataset [4]. The public image dataset consists of 92 thousand images of 36 characters and 10 digits of Devnagari script. Each image is 32x32 pixels.

4.1.2. Artificial Data Synthesis

The dataset generation process was carried out using the following approach:

1. Get the character set. For instance, the character set contains vowels (◌ा, ,◌ि ◌ी ...), consonants (क, ख, ग ...) and special symbols (◌ँ, ◌ं,...).
2. Get all possible combinations from the obtained character set.
3. From the set of images containing samples of background noise, select one randomly.
4. Generate random noise profile to overlay upon the image.
5. Take a character from the obtained set after combination and superimpose it upon the background image. Add the random noise profile after carrying out a variety of transformations (scale, skew, rotate, warp) combinations with the generated noise profile over a number of randomly selected images.
6. Store the data with appropriate labels

The first phase of the project was carried out using 30% real world data and 70% Artificially Synthesized Data.

4.2. Character Segmentation

The expected inputs to the system are documents scanned from various sources with inherent distortion, print imperfections and scanning limitations. Thus, a character segmentation method based on histograms has been implemented in this project. A binary thresholding system has been kept in place for improving the Character Segmentation system accuracy.

The system works as follows:

1. Based on a preselected threshold value, the pixels having intensity less than the threshold are assumed to be black and those greater or equal to the threshold value are assumed to be white.
2. Scan each row of the image and split the image based on the fact that each consecutive lines are separated by at least one row is present without any black pixels. After this step, we get a number of images containing multiple words.
3. Using similar principle but scanning vertically, word segmentation is performed. After completion of this step, we obtain the words in each line.
4. The 'dika'(The line,touching the upper portion of a word, used to connect the word) is then removed, and then letters are extracted by the similar process of blank column detection.

The obtained characters are then passed on to the ConvNet based classifier.

4.3. Feature Extraction

For the sake of simplicity, we have used grayscale values as our feature. Each pixel corresponds to a value from 0 to 255 representing pitch black to bright white and various shades of gray in between.

Each training image is scaled to a size of 30x30 cropping out anything that exceeds the value. The grayscale result from this phase is a 30x30 matrix, which we consider as the required feature matrix.

4.4. Classifier

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self-driving cars particularly because ConvNets handle translation of images pretty well. i.e. it will still make a correct prediction even if we move an image sideways[7].

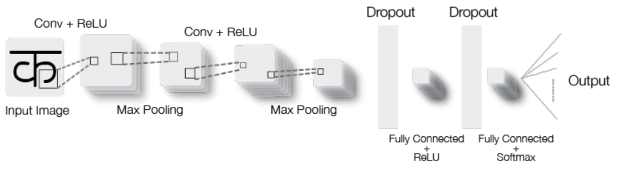

Figure 1: Architecture of Convolutional Neural Networks Used

The main problem of using a multi-layered net is that it is harder to train.

We used two convolutional layers with 32 convolution filters of size 3x3 and rectifier activation as the first layer. ReLU is a non-linear pixel-wise operation carried out on the feature map. Basically, it replaces all the negative values in the map by zero. ReLU is particularly used to introduce non-linearity in the system as convolution is a linear operation and is not quite sufficient to handle the non-linearities prevalent in the real-world data[8].

Then, we added a pooling layer with a pool size of 2x2. Pooling layers are generally added right after convolutional layer in order to simplify the output of the convolutional layer. There are a number of approaches to pooling such as Max Pooling, Average Pooling, L2 Pooling, etc. In our project, we have implemented Max Pooling. Max Pooling is a simple operation in which the maximum value of a feature map is returned. This step ensures that our feature map has dimensions that are manageable and possible overfitting due to a large number of parameters is avoided. Here, the pooling layer summarizes every 2x2 region of the convolution layer which is immediately followed by a dropout layer with a probability of 25%.

The dropout layer randomly selects a subset of neurons for each iteration and assumes that remaining other neurons don't exist. Although it might seem to be strange at first, using a dropout layer helps in minimizing over-fitting because it's like using a different neural network in every iteration. The net effect is balanced out[7].

5. Result and Analysis

5.1. Approach

A variety of architectures were tested by varying the number, types of layers and the activation function. For the purpose of training, 80% of the dataset has been used and for the purpose of testing 20% of the dataset has been used. The tests were carried out for a number of epochs and the convergence patterns were studied.

5.2. Result

Upon running the system for 15 epochs with kernel size of 3x3 the accuracy of the system was found to be 97.01% for previously untrained data or previously unseen data and 81.44% for previously trained data. When the kernel size was increased to 5x5, and dropout layer removed, an accuracy of 94.82% of accuracy was obtained for previously untrained data and 96.06% was obtained for previously trained data.

Upon running the system for 32 Epochs, the system resulted in an accuracy of 99.31% for previously untrained data and 96.5% for data that has been used for training previously

5.3. Analysis

The overall accuracy for a system consisting of the proposed architecture run for 32 Epochs is 99.31% on previously unseen data and 96.5% on previously seen data. The fact that the accuracy is less for test set data that has been previously trained can be best attributed to the presence of dropout layer. It can be best seen from the plot for the accuracy of the system on testing with previously trained data, where the highest accuracy reached is

Table 3: Confusion Matrix for Major Errors

	ड.	ट	ड	त	थ	द	ध	य	र	व	ह	त्र
घ					1		3	4		9		
ड.			28									
ड	100	1										
ढ		4									5	
ध					47			8				
य							1			5		1
ल				8					1			
व	1	1					1	1				44
स				12					6			
ह	1	3	3	2		8			3			1

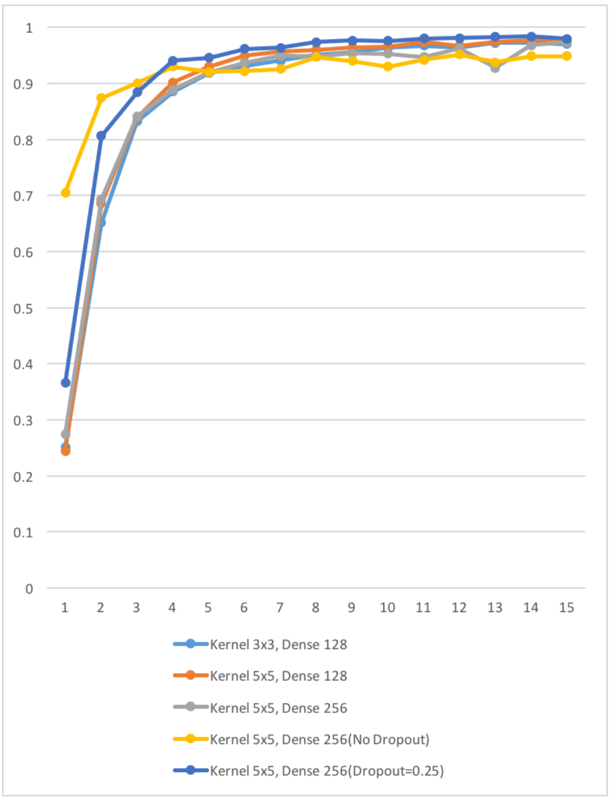

Figure 2: Accuracy of System for Various Cases on Previously Trained Data

96.06% present in the case where dropout layers are absent. When the small value of dropout i.e. 0.25 is used, it gives rise to the accuracy of the value next to the largest value. Hence, it can be inferred that, in the absence of the dropout layer, the system tends to memorize instead of learning the data. Furthermore, it can be seen that in the absence of dropout the architecture tested on previously unseen or untrained data, the accuracy is the least i.e. 94.82%. Also, it can be seen that using the kernel size of 3x3 results in a comparatively lower accuracy than that obtained using the kernel size of 5x5.

From the confusion matrix in Table 3, it can be seen that the system makes an error in some of the cases. Most of the errors occur in cases where two characters are visually similar. For instance the character 'ङ' and 'ड.' are similar to a large extent, hence accounting for the

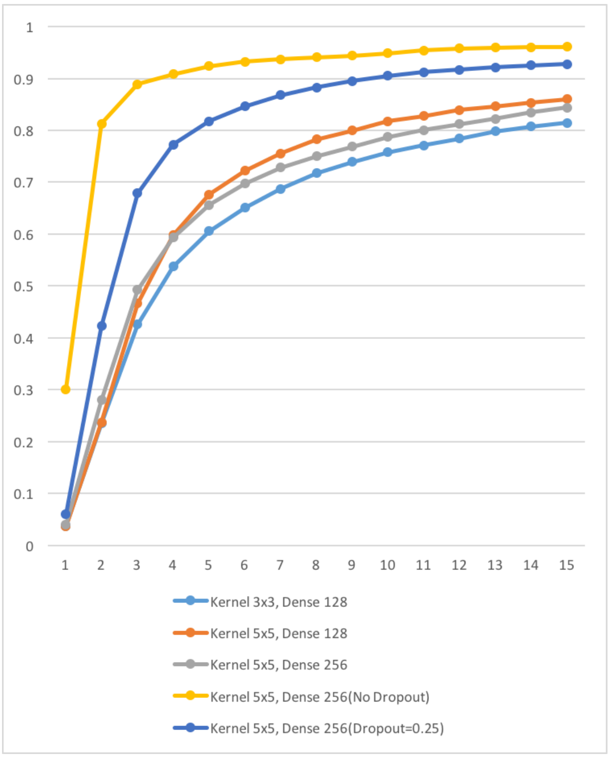

Figure 3: Accuracy of System for Various Cases on Previously Untrained Data

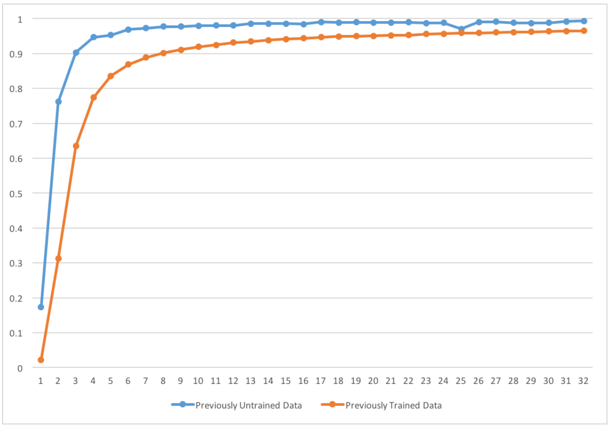

Figure 4: Accuracy of System with Proposed Architecture Run for 32 Epochs

largest number of errors in classification.

6. Future work and Conclusions

Hence, an Optical Character Recognition System has been implemented using Convolutional Neural Networks using Keras[1] and Theano[2]. The system has given encouraging results for a limited domain. However, The work can be extended to a much larger domain.

For further improving results from the system, the architecture of the classifier can be changed to Recurrent Neural Network instead of just CNN. More advanced feature extraction could also boost the accuracy of the system in various test conditions. Furthermore, improvement in the character localization process can result in the higher accuracy of the whole system.

7. References

[1] F. Chollet, "Keras: Deep Learning library for Theano and TensorFlow," www.keras.io, [Online].

[2] "Theano," www.github.com/Theano, [Online].

[3] T. Oliphant, "Numerical Python Library," www.numpy.org, 2014, [Online].

[4] A. K. Pant, P. K. Gyawali, and S. Acharya, "Automatic nepali number plate recognition with support vector machines," *In Proceedings of the 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pp. 92–99, 2015.

[5] A. C. et. al., "Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning," 2011.

[6] S. Shakya, A. Basnet, S. Sharma, and A. Gurung, "Optical character recognition for nepali, english character and simple sketch using neural network," *Recent Advances in Information and Communication Technology*, pp. 45–53, 2016.

[7] M. Nielsen, "Neural Networks and Deep Learning," http://neuralnetworksanddeeplearning.com/, [Online].

[8] A. K. et. al., "ImageNet Classification with Deep Convolutional Neural Networks," 2012.