Nepali Character Recognition Using Deep Belief Nets

Aadesh Neupane
aadeshnpn@byu.edu
Brigham Young University

Abstract

Various deep learning architectures such as Deep Neural Network (DNN), Convolution Neural Network (CNN) and Long short term memory (LSTM) Recurrent Neural Nets (RNN) have been applied to fields such as natural language processing, speech recognition and computer vision where they have been shown to produce state-of-the-art results. But these advance models have not yet been applied in the context of Nepali language. Trivial problems like automatic recognition of a small piece of a handwritten document is still a challenging task in Nepali. So, we want to apply state-of-the-art deep learning models to recognize handwritten Nepali characters. This might convince researchers to explore these deep models which in turn will help to expand research on Nepali language. After careful analysis of different hyper parameters, we choose four layers deep network with 1024 hidden units in each layer. We achieved 90.18% accuracy in a test set against 72.3% accuracy obtained using simple Multi Layer Perceptron (MLP). This result shows a significant improvement above all other methods reported. This trained model can be used for any general purpose application which involves recognizing Nepali characters effectively.

Introduction

Nepali is the official language and de facto langua franca of Nepal. It is also spoken in many parts of India, and the total number of speakers is around 16 million. Nepali was developed in proximity of a number of Indo-Aryan languages showing Sanskrit influences. Nepali language is very different than the more known languages such as English and Spanish. Nepali has 36 consonant characters, 12 vowel characters and each vowel character can modify each consonant character. In this regard, there can be a total of 446 characters including numeric characters. This complication might be one of the factors why so little research have been done on this language. Another factor is that we had no datasets available publicly to carry out research. Recently (Neupane 2014) made Nepali Character dataset publicly available, and research has been accelerating since then.

(Khanale and Chitnis 2011) showed that simple MLP can be used for optical character recognition for Nepali. But it lacks to provide the standard dataset that they worked on, doesn't describe the architecture completely and lacks in evaluation strategy. (Subarna Shakya et al. 2016) also used MLP to solve the character recognition problem. They also used similar techniques used by (Khanale and Chitnis 2011) but those techniques couldn't be applied for general purpose application. (Shanti Shakya et al. 2009) showed some promising architecture to solve recognition problem. (Nirajan Pant 2016) followed similar architecture as of (Shanti Shakya et al. 2009). (Pant, Panday, and Joshi 2012) showed that Radial Basis function outperformed the MLP based recognition system.

We are the first one to try to solve the Nepali character recognition problem using DNNs. DNNs has been used extensively in diverse fields (Deng 2014). The rest of this paper is organized as follows. Section 2 describes the model that we have employed, section 3 will describe the results and section 4 will provide conclusion.

Deep Belief Nets

Recently, Deep Neural Network (DNN) are being used to solve a plethora of problems. We are interested to explore Deep Belief Network (DBN) to solve Nepali character recognition task. We choose DBN in place of CNN as (Hinton, Osindero, and Teh 2006) showed that DBNs outperformed CNNs in MINIST dataset.

Deep Belief Networks are basically Restricted Boltzmann Machine (RBM) stacked together and each layer of the RBMs trained in a greedy approach (Hinton and Salakhutdinov 2006) (Bengio et al. 2007). DBNs are good in extracting a deep hierarchical representation of the training data. When trained on a set of examples in an unsupervised way, a DBN can learn to probabilistically reconstruct its inputs. The layers then act as feature detectors on inputs. After this learning step, a DBN can be further trained in a supervised way to perform classification. They model the joint distribution between observed vector x and the l hidden layers h^k as follows:

$$P(x, h^1, ..., h^l) = (\sum_{k=0}^{l-2} P(h^k|h^{k+1}))P(h^{l-1}, h^l)$$

where $x = h^0, P(h^{k-1}|h^k)$ is a conditional distribution for the visible units conditioned on the hidden units of the RBM at level k, and $P(h^{l-1}, h^l)$ is the visible-hidden joint distribution in the top-level RBM.

Dataset

We are using the sub-set of this (Neupane 2014) dataset containing only numeral characters and has 10 class labels similar to the MINIST dataset. So, we thought DBNs would be good choice to experiment with. The reason to use the sub-set of the dataset was because the machine didn't had the capacity to process full dataset. The full dataset contains fifty-one class labels with a total of 223,999 images. Each image size is 32*32 in gray scale. The sub-set of the dataset that we are using just contains ten class labels which fall into the numeral category. It has 44,688 images as this size can be easily processed using our machine. The dataset was randomly divide into a training set(80%), test set (10%)and validation set (10%). Figure 1 shows a few example characters from the dataset.

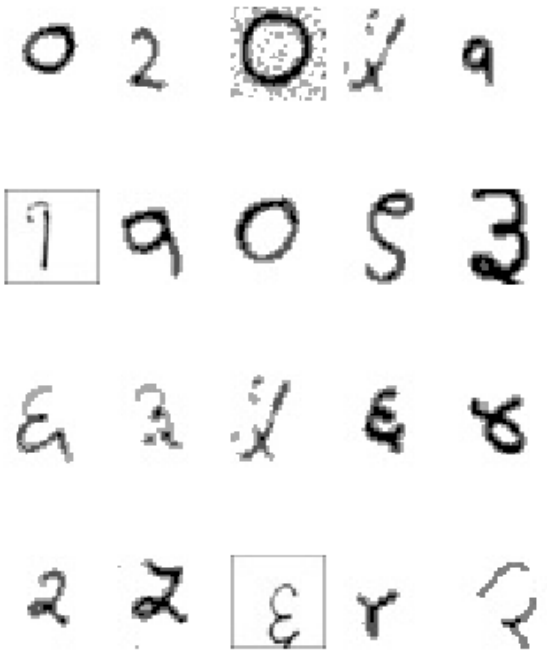

Figure 1: Nepali Numeral Characters

Model

The model takes in the 32*32 gray scale image as input. We use 1024 nodes in the input layer, one for each pixel in the image. The inputs to the model are normalized. Let X be the matrix representing the images. Then the normalization technique used is $X = X - \overline{X}/\sqrt{X}$. Next, we use four hidden layers of RBM that are trained using unsupervised greedy layer approach. After training each hidden layer in unsupervised fashion, a logistic output layer fine-tunes the network parameters using the original labels provided with the dataset. At first a simple MLP with softmax and entropy loss function was implemented. Then, a RBM using contrastive divergent algorithm was implemented. After testing both of them independently, they were combined to build a DBN. Figure 2 show the high level view of our model.

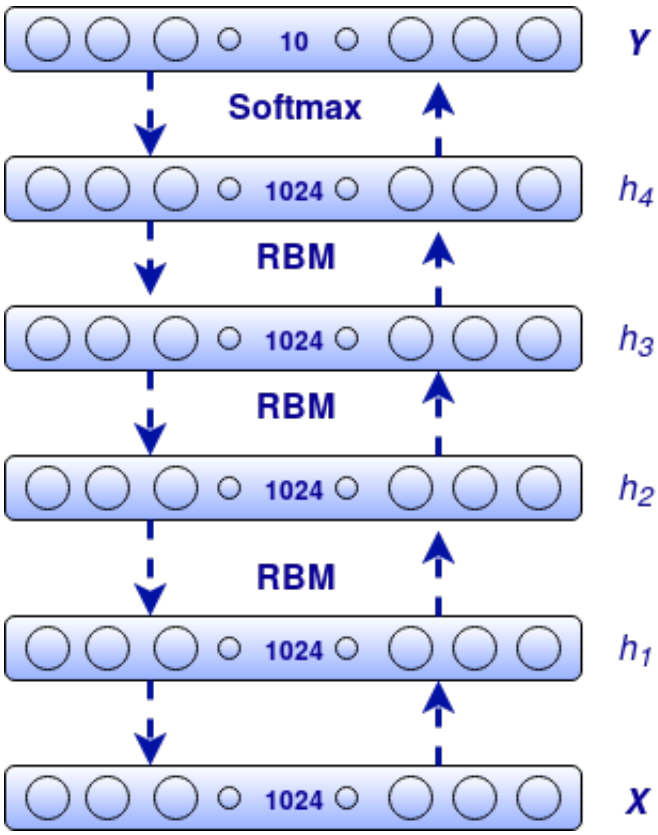

Figure 2: DBNs model for Character Recognition

Problems

Here, we discuss some of the challenges that we faced in solving our problem using DBNs. Vanishing or exploding gradients is the difficulty faced in training neural networks which uses gradient based methods where the weights seems to increase or decrease exponentially. Gradient exploding was the biggest problem faced during the implementation phase. At first we implemented MLP with stochastic gradient to update weight for each input element without vectorization. This implementation worked fine for simple datasets. But after vectorization of the implemented algorithms, the gradient exploding problem was observed in simple MLP as well. Since, this problem was not experienced with simple datasets like Iris, Wine and others, the primary reason should be the large size of input and hidden nodes. To make matter worse, vectorized implementation involved multiplying the input matrix of size 44688*1024 with

weight matrix of size 1024*1024. Then it is obvious to observe exploding gradients resulting the weights updates in *Nan*. Significant time was spent fixing these issues. Based on observations, we found that the gradient exploding or vanishing problem not only exists in DNN but it could arise in MLP with large feature space as well.

To minimize this issue, instead of using complete input data, we started using a minibatch size of 200 image each. Then we adjusted the softmax function using normalization technique to prevent overflow or underflow in final the layer. This gave us insights why the minibatch is useful in training and why all ML practitioners use this technique. After applying these techniques, we were finally able to run MLP on this dataset.

It was hard to implement the paper since it lacked guidance on choosing/setting initial parameters. Instead of randomly initializing weights, we need to draw the weights from normal distribution for convergence. Finally, after all the pieces were combined to build DBNs , still the model was not able to generalize, and the accuracy obtained was less than 30%. It took rigorous tinkering to figure out that without batch normalization of weights, DNNs doesn't work as expected (Ioffe and Szegedy 2015). We came to know the significance of batch normalization only when DDN continuously failed to converge.

Results

We evaluated the datasets with three different models MLP, CNN and DBNs. We first ran the dataset using MLP and got 72.30% as baseline accuracy. The MLP had only one hidden layer with 1024 hidden units. Logistic regression with softmax function was used on the output layer to give probabilistic output. Then we evaluated the dataset using CNN with 3 convolution layers and 2 fully connected final layers which led us to 89.17% accuracy. The CNN model had three convolution layers and two fully connected layers. Using DBNs we obtained best accuracy of 90.18% with 4 hidden layers with 200 epochs for unsupervised RBM weight updates and 100 epochs of supervised fine-tuning of weights. This result is the most promising result recorded on this dataset. We applied our model to MNIST as well and got 96.5% accuracy. Table 1 shows accuracy for different settings with 20 epochs for pre-training and 10 epochs for fine-tuning for Nepali Character dataset. Figure 3 and 4 show performance of MLP and DBN respectively.

Conclusion

In this project, we implemented DBN to solve Nepali character recognition problem. It showed promising results than some other models that we evaluated. Through implementation of DBN, we gained insights on the importance of minibatch and batch normalization techniques, and the role the play to effectively train DNNs. Our trained model can be used for any general purpose Nepali character recognition task.

Layers	Hidden Units	Accuracy (%)
2	1024	90.1
2	256	89.97
3	1024	88.36
3	256	89.97
4	1024	89.97
4	256	89.97
5	1024	87.88
5	256	89.97
6	1024	88.36
6	256	89.97
7	1024	90.18
7	256	89.97

Table 1: Accuracy of DBNs with variations of number of hidden layers and hidden units

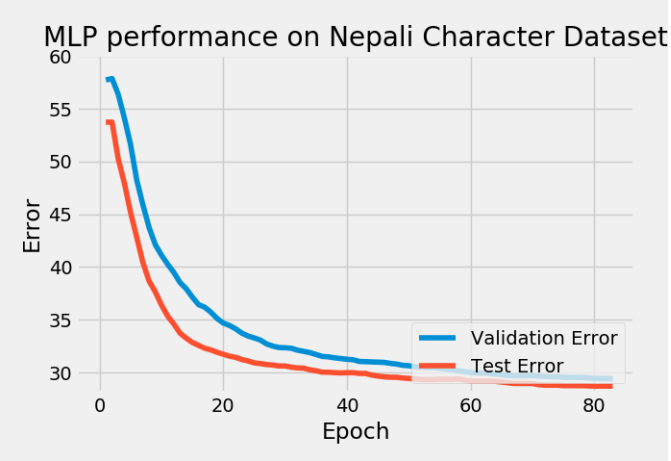

Figure 3: MLP Performance on Nepali Character Dataset

References

Bengio, Yoshua et al. (2007). "Greedy layer-wise training of deep networks". In: *Advances in neural information processing systems* 19, p. 153.

Deng, Li (2014). "A tutorial survey of architectures, algorithms, and applications for deep learning". In: *APSIPA Transactions on Signal and Information Processing* 3, e2.

Hinton, Geoffrey E, Simon Osindero, and Yee-Whye Teh (2006). "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7, pp. 1527–1554.

Hinton, Geoffrey E and Ruslan R Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks". In: *science* 313.5786, pp. 504–507.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167*.

Khanale, PB and SD Chitnis (2011). "Handwritten devanagari character recognition using artificial neural network". In: *Journal of Artificial Intelligence* 4.1, pp. 55–62.

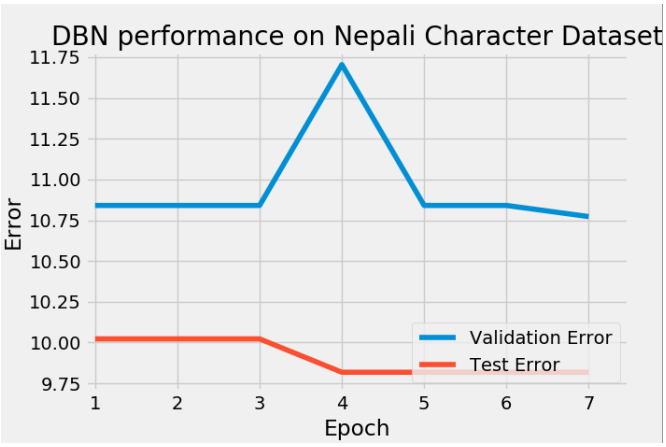

Figure 4: DBN performance on Nepali Character Dataset

Neupane, Aadesh (2014). "Article: Development of Nepali Character Database for Character Recognition based on Clustering". In: *International Journal of Computer Applications* 107.11. Full text available, pp. 42–46.

Nirajan Pant, Bal Krishna Bal (2016). "Nepali Optical Character Recognition - A hybrid Approach". In: *None*.

Pant, A. K., S. P. Panday, and S. R. Joshi (2012). "Offline Nepali handwritten character recognition using Multilayer Perceptron and Radial Basis Function neural networks". In: pp. 1–5.

Shakya, Shanti et al. (2009). "Interim Report on Nepali OCR". In: *Madan Puraskar Pustakalaya*.

Shakya, Subarna et al. (2016). "Optical Character Recognition for Nepali, English Character and Simple Sketch Using Neural Network". In: ed. by Phayung Meesad, Sirapat Boonkrong, and Herwig Unger. Springer International Publishing.