Handwritten recognition of Tamil vowels using deep learning

IOP Conf. Series: Materials Science and Engineering **263** (2017) 052035 doi:10.1088/1757-899X/263/5/052035

Handwritten recognition of Tamil vowels using deep learning

N Ram Prashanth, Siddarth B, Anirudh Ganesh and Vaegae Naveen Kumar

School of Electronics Engineering, VIT University, Vellore 632014, Tamil Nadu, India

E-mail: vegenaveen@vit.ac.in

Abstract. We come across a large volume of handwritten texts in our daily lives and handwritten character recognition has long been an important area of research in pattern recognition. The complexity of the task varies among different languages and it so happens largely due to the similarity between characters, distinct shapes and number of characters which are all language-specific properties. There have been numerous works on character recognition of English alphabets and with laudable success, but regional languages have not been dealt with very frequently and with similar accuracies. In this paper, we explored the performance of Deep Belief Networks in the classification of Handwritten Tamil vowels, and conclusively compared the results obtained. The proposed method has shown satisfactory recognition accuracy in light of difficulties faced with regional languages such as similarity between characters and minute nuances that differentiate them. We can further extend this to all the Tamil characters.

1. Introduction

Tamil is one of the oldest known languages in the world and is the native language of India and Sri Lanka. Its roots dates back to 300 BC and has evolved from the old Tamil found in scriptures to the language we know today. This major change in literature and pronunciation is the consequence of the use of European-style punctuation and the use of consonant clusters that were previously unused. This paper takes into account of the newly evolved alphabets that is used in the modern world. Figure 1 shows the twelve Tamil vowels. Deep Belief Networks (DBN) [7] has become the de facto standard for complex pattern recognition problems [4], [6]. DBN based character recognition systems have produced very low error rates on many common datasets such as MNIST [7]. In this paper, we have explored the use of DBNs to recognize handwritten Tamil vowels.

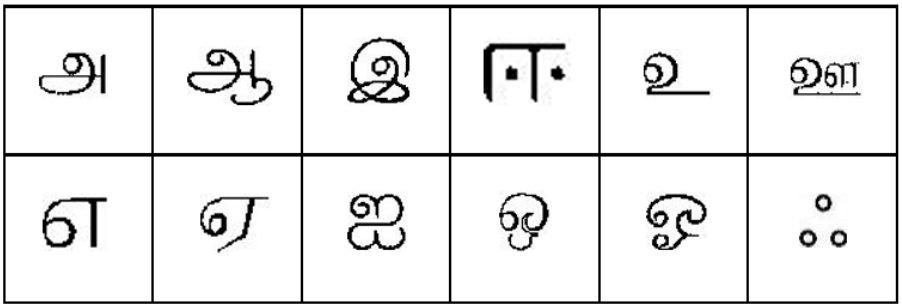

Figure 1. Tamil vowels.

2. Morphological erosion and dilation

Morphology is a huge set of image processing operations that process images based on shapes. Morphological operations apply a structuring element to an input image, creating an output image of the same size. In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbours. By deciding and selecting the size and shape of the neighbourhood, you can make a morphological operation that is affected to certain shapes in the input image. The morphological operations we used here were dilation and erosion. In Dilation pixels are added to the boundaries of objects in an image, while erosion subtracts the pixels on object boundaries. The number of pixels to be added or subtracted is decided based on the size and shape we used for structuring the image to be processed. In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbours in the input image. This rule used for processing the pixels expresses the operation as erosion or dilation.

2.1. Image pre-processing

The images are converted into a standard format by pre-processing in order to feed into the classifier. The images in our dataset are handwritten characters for the purpose of this study. We have processed 12 of those classes. We have RGB images of different sizes so these images are first converted to gray-scale where each pixel value is an integer between 0 and 255. These values correspond to the intensity of the pixel where 255 represent the white pixels and 0 represents the black pixels. The sample image size varies from person to person for every character and the images are therefore the input images have been resized to constant size of 28 × 28. Finally, the 28 × 28 gray-scale images are normalized to have intensity values in the range of 0 to 1. A set of such arrays is then given as input to the DBN network.

2.2. Data augmentation

Data Augmentation is an important step after data extraction which would improve our output in accuracy, reliability and thereby creating a data set with wide range of inputs [2]. The different techniques used by us for the augmentation are image scaling, image rotation, and image shearing. The images have been scaled using factors of 0.45, 0.625, 1.2 and 1.6. The various number of rotation we used without distorting the image, are performed at angles 10, -10, 15 and -15 degrees. Additionally, we then added Gaussian noise to all the images as it enhances the training of neural network. Besides using each of these techniques independently, we have also applied compound transformations consisting of any two of the above stated techniques. To the resultant 9000 samples that were obtained, Gaussian noise with a mean of 0 and variance 0.01 As a result, we obtained 18000 samples for 12 classes of numerals, averaging around 1800 image samples per numeral class.

3. Deep belief networks based approach

3.1. Deep belief neural networks

A Deep Belief Network is usually viewed as a number of Restricted Boltzmann Machines (RBM) stacked together, such that each RBM is connected with previous and next layers. A Restricted Boltzmann Machine is a neural network that comprises of a layer of visible neurons, a layer of hidden neurons and a bias unit. Each hidden neuron receives an input from each of the visible neurons and the bias is connected to all hidden neurons as well as visible neurons [5]. Figure 2 shows a deep belief network that is formed by stacking of RBMs and training them in a greedy manner.

Deep Belief Networks contain multiple layers of latent variables and are probabilistic generative models [3]. These latent variables are also known as feature detectors or hidden units. Undirected symmetric connections between the top two layers form an associative memory and directed connections are formed by the lower layers. The latent variable in the previous layer is used to train the next layer. The first hidden layers h_k and vector x are modeled by their joint distribution:

$$P(x,h^1,\ldots,h^l)=\left(\prod_{k=0}^{l-2}P(h^k|h^{k+1})\right)P(h_{l-1},h_l) \quad (1)$$

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. In DBNs, the weights, W learned by an RBM defines both p (v∨ h, W) and the prior distribution over hidden vectors p (h|w) , so the probability of generating a visible vector v can be written as:

$$p(V)=\sum_h p(h/W)p(V\vee h,W) \quad (2)$$

After learning the weights, we keep p (v∨h, w) but we replace p (h∨w) by an improved model of the combined posterior distribution over the vectors that are hidden. We then treat the hidden activity vectors obtained from the training data as the input for the subsequent learning module in order to learn a better model. There is a significant increase a variationally lower bound on the composite model's probability of training the data if executed right [5].

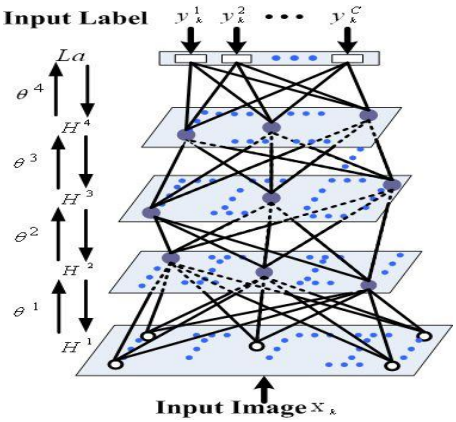

Figure 2. Deep belief neural network topology.

3.2. Classification settings

We have implemented DBNs in Theano and have used a separate class for RBM and used this class to create a stack of RBM classes for our deep belief network. Our network consists of two stages: a layer-wise pre-training stage and a fine-tuning stage. In the pre-training stage, the input for each of the RBMs in the level for each layer of the network is determined. In the fine-tuning stage, the cost function using negative log likelihood through statistic gradient descent with mini-batches is minimized.

3.3. Implementation

The MLP is formed by storing feed forward graphs that is obtained by using an n sigmoid. This n layered RBM stack is the restricted Boltzmann machine stack that is used to learn higher complexity features. The method that generates the training functions yields the RBM stack as a list which implements one step of RBM at each layer. Then, the last layer is the logistic regression layer which is used for providing classification. The dataset is then split up into a training set (60%), a cross validation set (20%) and a test set (20%). Our cost function is minimized by the use of a negative log-likelihood (NLL) function as our cost function. This is because operations on small numbers in the logarithmic scale yields larger negative value which a machine working on a floating point (high precision) can handle better. It converts sums into multiplications which is more stable numerically.

The NLL acts as an alternate for a 0-1 classifier because it scales better with a larger dataset and more classes [1]. We take maximum advantage of the GPU for faster calculations as all of these computations are intensive parallel computation applications. In order to do this, we use Theano's CUDA bindings, which allow us to improve our calculations, convolutions and large-element wise operations from 5-50x by using the CUDA cores on the GPU for massive parallelization [9]. This is a significant increase in the rate of processing when compared to a CPU [8].

4. Results and Discussions

DBN takes a long time to converge but it achieves a good accuracy. The error percentage decreases gradually and reaches its minimum after around 30 epochs. The performance analysis of DBN is shown in figure 3, 4 and 5. As shown in the table 1, the error percentage is found to decrease as the number of epochs increases and it eventually flattens out to reach minima showing that the algorithm has converged.

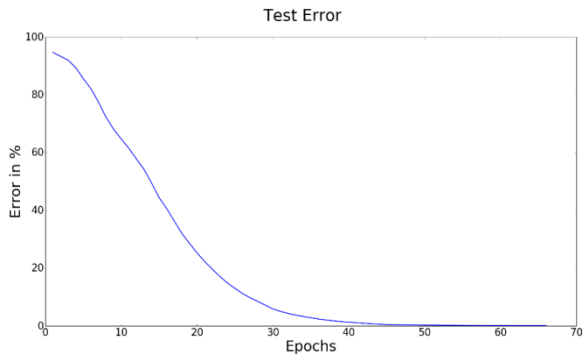

Figure 3. Test error of DBN.

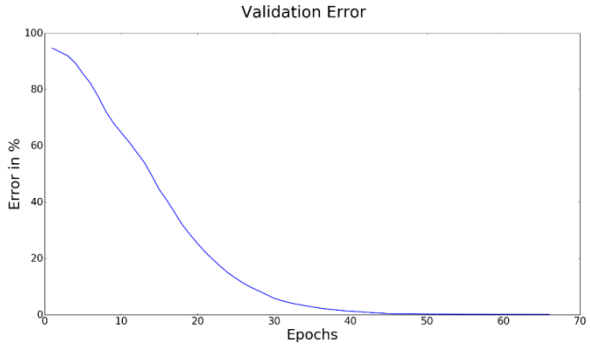

Figure 4. Validation error of DBN.

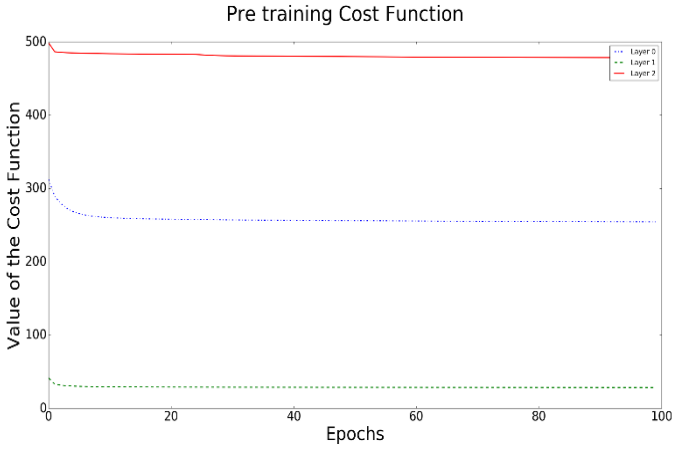

Figure 5. The smooth convergence of cost function of different pre-training layers of the DBN.

Table 1. Performance of DBN

Epoch	Deep Belief Network Test Error	Epoch	Deep Belief Network Test Error
1	94.7	30	5.89
5	85.6	35	2.79
10	64.7	40	1.30
15	44.4	45	0.43
20	25.2	50	0.27
25	13.0	66	0.11

5. Conclusion

In this paper, the performance of the DBN network is analysed for the recognition of handwritten Tamil vowels. The accuracy level has been found to be 99.89 percent. It seems to converge fast and is easy to train. DBNs are quite feasible and they converge at a fast rate. Further, we can extend the approach to all classes of Tamil characters.

References

[1] Alex K, Ilya S and Geoffrey E. H 2012 ImageNet Classification with Deep Convolutional Neural Networks
[2] Bengio Y and Glorot X 2010 Understanding the difficulty of training deep forward neural networks AISTATS
[3] Bengio Y, Lambli P, Popovici D and Larochelle H 2007 Greedy Layer-Wise Training of Deep Networks *In. Advances in Neural Information Processing Systems 19 NIPS'06* (*MIT Press*)
[4] Bishop C. M 1995 Neural Networks for Pattern Recognition Clarendon
[5] Hinton G. E and Salakhutdinov R. R 2006 Reducing the Dimensionality of Data with Neural Networks *Science* 2006 vol 313 (5786)
[6] LeCun Y, Bottou L, Bengio Y, and Haffner P 1998 Gradient-based learning applied to document recognition *Proceedings of the IEEE*
[7] Mohammad A. K, Mohammad M. H 2014 A brief survey on deep belief networks and introducing a new object oriented toolbox DeeBNet.arXiv:1408.3264
[8] Ritesh S, Nibaran D, Amit K S and Mita N 2016 A multi-objective approach towards cost effective isolated handwritten Bangla character and digit recognition
[9] Theano's CUDA bindings: http://bit.ly/2gHY6oE