A Performance Comparison of HTM and DBN Models Applied to Visual Object Classification

[1,2]K. Valentín, [1]I. Bajla, [3,1]S. Štolc

[1]Institute of Measurement Science, Department of Theoretical Methods, Slovak Academy of Sciences, Bratislava, Slovakia
[2]Faculty of Mathematics, Physics, and Informatics, Department of Applied Informatics, Comenius University in Bratislava, Slovakia
[3]Austrian Institute of Technology, GmbH, Seibersdorf, Austria
Email: kristian.valentin@fmph.uniba.sk

Abstract. *In the paper we compare the performance of two Deep architecture models – Hierarchical Temporal memory (HTM) and Deep Belief Network (DBN) in the task of visual object classification. Both models utilize k-Nearest Neighbor classifier for the final object classification. We conduct two experiments in order to study invariance properties of the models with respect to object's rotation and scaling using image database of tree leaves containing 11 categories. Obtained results show that the HTM model outperforms the DBN model in both types of experiments.*

Keywords: Hierarchical Temporal Memory (HTM), Deep Belief Network (DBN), Visual Object Classification, Rotational and Scaling Invariance

1. Introduction

Both Hierarchical Temporal Memory (HTM) and Deep Belief Network (DBN) models can be framed into Deep architectures [1,2] which attracted attention of the computational neuroscience in the recent years. They are inspired by basic principles of information processing in the neocortex, but their structure, learning and inference algorithms differ in several aspects. They both aspire to solve problems of visual object classification, however, no comparative analysis of their performance in this area has been published yet. Therefore, the goal of this paper is to evaluate the performance of the HTM and DBN models on appropriate benchmark experiments. In particular, we are interested in how they tackle problems of the visual object classification invariant against scale and rotation geometrical transformations.

2. Description of Deep Architecture Models

Hierarchical Temporal Memory

Structurally, HTM [3,4] is a multi-layered hierarchical network. The hierarchy has a tree-shaped structure consisting of a set of basic computational units, called nodes. The nodes have receptive fields that are reflected in a way that nodes in the given layer are connected only to a subset of nodes in the previous layer. By ascending the hierarchy, the nodes "see" gradually growing portion of the network's input. Each node uses the very same learning and inference algorithm. Unlike many other methods, HTM combines spatial and temporal learning. For the temporal learning, a sequence of input data is needed. The sequence can be either natural (e.g., video stream) or it must be generated from a still image. The first step in learning a layer of nodes is spatial learning which produces a lossy compression in the spatial domain using vector quantization. Then the temporal learning takes the result of the spatial learning and by

exploiting the learned sequences it creates a spatio-temporal representation of the input data. The representation is stored in a hierarchy that is organized in such a way that the higher levels of the hierarchy cover larger parts of the input space and they correspond to longer durations of time. This all is done in semi-unsupervised manner. The only "teacher" is time that is based on assumption that patterns occurring close in time are mutually related. The inference is based on Bayesian belief propagation.

Deep Belief Network

Deep Belief Network (DBN) is a graphical probabilistic generative model composed of multiple layers of stochastic, latent variables [5]. The latent variables, often called hidden units or feature detectors, have usually binary values. The structure of DBN can be seen as a stack of restricted type of Boltzmann machines (RBM) [6]. Each RBM contains a layer of visible units that represent the data and a layer of hidden units that learn to represent features which capture higher-order correlations in data. The two layers are connected by a matrix of symmetrically weighted connections. There are no lateral connections between units in a layer. When two RBMs are to form a DBN, the layer with hidden units of the first RBM is linked with the layer with visible units of the second RBM. The top two layers form RBM. DBN trains one layer at a time starting from the bottom one. The values of the hidden units in one layer, when they are being inferred from data, are treated as the data for training the next layer. The training is based on the stochastic gradient descent method and uses contrastive divergence algorithm to approximate the maximum log-likelihood. The common way of how to use DBN for image classification tasks is a combination of unsupervised pre-training and subsequent supervised fine-tuning. This discriminative fine-tuning can be performed by adding a final layer of variables that represent the desired outputs and back-propagating error derivatives.

3. Comparison of HTM and DBN in a classification task of visual objects

In order to create a model of input data, HTM uses the "time as a teacher" concept, whereby the time is not inherently used in DBN, even if it can be adapted to learn sequences. This leads to a different learning procedure for HTM, where a learning sequence has to be generated from still input images, e.g., by moving an object smoothly within the field of view of the network. Designing an optimal HTM architecture with optimal values of the parameters is a non-trivial task, although, there is graceful performance degradation when parameters are set suboptimally. In the HTM networks, the node-sharing mode during learning with a combination of pooling algorithms, supports certain invariance to position, rotation and scale, whereas in the DBN networks, there is no such architectural or learning predetermination.

4. Experiments

Data

Based on preliminary tests carried out separately with DBN on a dataset of tree leaves, we have decided to use the same dataset for our comparative analysis. This dataset consists of 11 categories of five gray-scale images of size 64×64 pixels. Within each image there is always one object – a leaf – centered on the black background. The most important information for classification of the tree leaves is contained in their shape. In order to extract such a type of information regardless from the object's absolute brightness, we decided for using a well-known edge detection method – Canny edge detector (lower threshold=100, upper threshold=200). In the first step, we applied the edge detector to all original images, afterwards, the resulting contour images were used as equivalent inputs for both the HTM and DBN models.

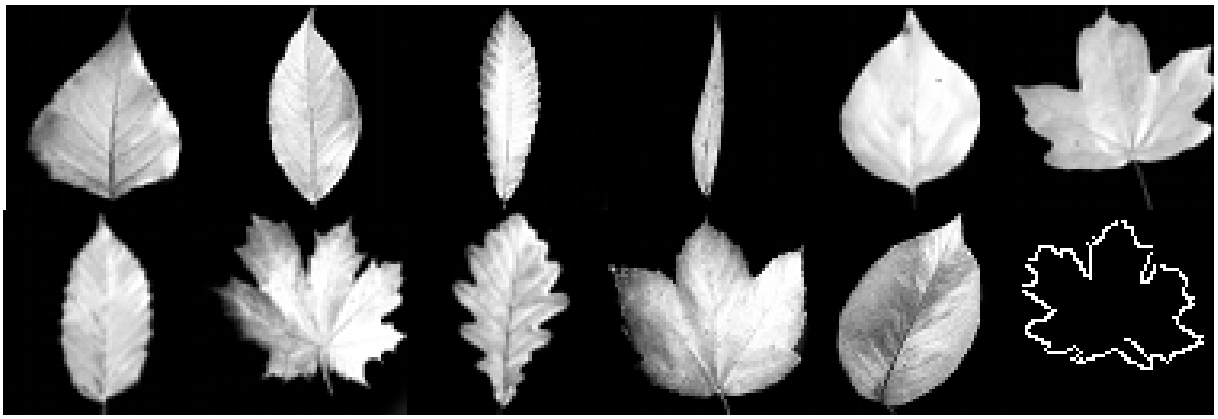

Fig. 1. Examples of 11 tree categories form the database leaves used. The bottom-right image shows the actual network's input obtained from the gray-scale image by application of the Canny edge detector.

To test the both network models in the task of invariant object classification, we conducted two types of experiments. The aim of Experiment 1 was to test the rotational invariance. The original images were rotated 12 times by 30 degrees which resulted altogether in 660 images. The aim of Experiment 2 was to test the invariance to scale of the objects within images. Leaves in original images were scaled randomly 5 times with scaling factor ranging between 0.5 and 1. The size of the dataset for this experiment was 275 images.

Set-up

In order to make the training conditions for both networks equivalent, we decided to require the same number of image pixels to be used for training. To satisfy this requirement, we adjusted the number of epochs in the case of DBN and the length of training sequences in the case of HTM. The other parameters were tuned empirically for the best results for these particular classification tasks and data. Moreover, the same classifier was used for both models, namely k-Nearest Neighbor (k-NN) with k=1.

The DBN network consisted of two hidden layers having 512 and 200 neurons. The learning rate for the training was set to 0.1, the batch size was set to 10, and the number of training epochs was set to 600.

The HTM network contained two levels of nodes. The nodes in 1st level were arranged to cover 8×8 pixels image regions. At 2nd level, the nodes formed 2×2 regions. The training sequence was generated using the random sweep explorer with 5000 iterations. The node-sharing mode was chosen for both levels. The values for 3 parameters of the spatial pooler and 2 parameters of the temporal pooler are listed in Table 1.

Table 1. *Parameters of the HTM network.*

Level	Spatial pooler			Temporal pooler	
	maxDistance	*sigma*	*maxCoincidenceCount*	*transitionMemory*	*maxTemporalGroupCount*
1	0.01	1.5	400	6	64
2	0.20	1.5	500	12	128

5. Results

Overall results of the two experiments are summarized in Table 2. The performance has been analyzed with respect to the perceptual split between data used for training and testing (i.e., 20:80, 30:70, 40:60, and 50:50). That gave us a clearer picture about inherent capabilities of each tested model regarding generalization and learned invariances. As a baseline, we provide

also performance of the k-NN (k=1) classifier applied to the original input data (edge images). Each experiment has been performed 20 times for each split factor to allow for calculating averaged classification accuracy values and associated standard deviations.

Table 2. *The average classification accuracy values in % and associated standard deviations obtained by the tested models. Experiments 1 and 2 refer to the rotational and scale invariance, respectively.*

Train set vs. test set in %	Classification accuracy in % (stdev)					
	Experiment 1			Experiment 2		
	k-NN	*DBN & k-NN*	*HTM & k-NN*	*k-NN*	*DBN & k-NN*	*HTM & k-NN*
20:80	34.54 (1.52)	56.07 (2.41)	63.53 (1.91)	17.75 (2.13)	39.43 (5.51)	57.31 (3.81)
30:70	37.75 (1.30)	60.92 (3.04)	69.90 (1.99)	17.95 (1.19)	50.67 (4.85)	61.34 (3.53)
40:60	38.33 (1.25)	64.10 (2.32)	73.35 (2.90)	19.79 (2.19)	64.18 (2.14)	65.39 (3.87)
50:50	38.82 (1.09)	66.76 (2.91)	75.36 (2.46)	20.10 (1.77)	58.80 (3.48)	66.44 (3.48)

6. Conclusions

In this paper we provide a comparative study on the performance of the Deep Belief Network (DBN) and Hierarchical Temporal Memory (HTM) models in the task of invariant object classification of tree leaves selected from a benchmark image database. For both compared network models, we have considered the k-NN classifier. As a baseline, we applied k-NN in the input (edge feature) space too.

The following conclusions can be drawn from the obtained results:

- both network models yield classification accuracy (CA) values significantly greater than those achieved in the original data space,
- the improvement of CA when increasing the ratio of training images is apparent; in the case of Experiment 1, CA of DBN improved by 10.69% while CA of HTM improved even more by 11.83%,
- finally, for both the rotational and scale invariance experiments, HTM significantly outperforms the results achieved by DBN.

Acknowledgements

This work has been supported by the Slovak Grant Agency for Science (project No. 2/0043/13).

References

[1] Bengio, Y. (2009). Learning Deep Architectures for AI. Foundations and Trends in Machine Learning, 2(1), 1-127.

[2] Arel, I., Rose, D., and Karnowski T. (2010). Deep machine learning - a new frontier in artificial intelligence research. IEEE Computational Intelligence Magazine, 5:13-18.

[3] Hawkins, J. and Blakeslee, S. (2004). On intelligence. Henry Holt and Company, New York.

[4] George, D. and Hawkins, J. (2009). Towards a mathematical theory of cortical micro-circuits. PLoS Computational Biology 5(10). DOI 10.1371/journal.pcbi.1000532.

[5] Hinton, GE. (2009). Deep belief networks. Scholarpedia, 4(4):5947.

[6] Hinton, GE. and Sejnowski, TJ. (1983). Optimal Perceptual Inference. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 448-453, Washington DC.