International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 21 (2018) pp. 15255-15260

Classification with MapReduce Based Deep Belief Network (MDBN)

Azhagammal Algarsamy*
Department of Computer Science, Research Scholar, Bharathiyar University, Tamilnadu, India.

Ruba Soundar Kathavarayan
Department of Computer Science Engineering, P.S.R Engineering College, Sivakasi-626140, Tamilnadu, India.

*(*Assistant Professor, Department of Computer Science, Ayya Nadar Janaki Ammal College, sivakasi)*

Abstract

Deep Belief Network is an algorithm among deep learning. It is an effective method of solving the problems from neural network with deep layers, such as low velocity and the over fitting phenomenon in learning. The learning of a DBN starts with pre-training a series of the RBMs followed by ne-tuning the whole net using backpropagation. Generally, the sequential implementation of both RBMs and backpropagation algorithm takes signicant amount of computational time to process massive data sets. The emerging big data learning requires distributed computing for the DBNs. In this paper, we present a distributed learning paradigm for the RBMs and the backpropagation algorithm using MapReduce, a popular parallel programming model. Thus, the DBNs can be trained in a distributed way by stacking a series of distributed RBMs for pretraining and a distributed backpropagation for tuning. Through validation on the benchmark data sets of various practical problems, the experimental results demonstrate that the distributed RBMs and DBNs are amenable to large-scale data with a good performance in terms of accuracy and efficiency.

Keywords: Map Reduce, RBM, Deep Belief Network

INTRODUCTION

Deep learning is also named as deep structured learning or hierarchal learning. It is one of the machine learning methods based on learning data representation. Learning can be either supervised or semi-supervised or unsupervised. In Deep learning methods there four models they are: stacked auto-encoder, deep belief network, convolution neural network and recurrent neural network, which are also the most widely used for big data feature learning methods. In this paper we review the research work on Deep Belief Network for Big data feature learning.

Big Data is a term used to describe collection of data that is huge in size and yet growing exponentially with time. In short, *s*uch a data is so large and complex that none of the traditional data management tools are able to store it or process it efficiently.

Recently, the cyber-physical-social systems, together with the sensor networks and communication technologies, have made a great progress, enabling the collection of big data [1,2]. Big data can be defined by its four characteristics, i.e., large volume, large variety, large velocity and large veracity, which is usually called 4V's model [3–5].

Deep learning is playing an important role in big data solutions since it can harvest valuable knowledge from complex systems [6]. Specially, deep learning has become one of the most active research points in the machine learning community since it was presented in 2006 [7-9]. Actually, deep learning can track back to the 1940s. However, traditional training strategies for multi-layer neural networks always result in a locally optimal solution or cannot guarantee the convergence. Therefore, the multi-layer neural networks have not re-ceived wide applications even though it was realized that the multi-layer neural networks could achieve the better performance for feature and representation learning. In 2006, Hinton et al. [10] proposed a two-stage strategy, pre-training and fine-tuning, for training deep learning effectively, causing the first back-through of deep learning. In addition, the increase of computing power and data size also contributes to the popularity of deep learning. As the era of big data comes, a large number of samples can be collected to train the parameters of deep learning models. Meanwhile, training a large-scale deep learning model requires high-performance computing systems. Take the large-scale deep belief network with more than 100 million free parameters and millions of training samples developed by Raina et al. [11] for example. With a GPU-based framework, the training time for such the model is reduced from several weeks to about one day. Typically, deep learning models use an unsupervised pre-training and a supervised fine-tuning strategy to learn hierarchical features and representations of big data in deep architectures for the tasks of classification and recognition [12]. In this paper section 2 introduces the mapreduce framework,proposed methodology explained in section 3, experimental results are listed in section 4, finally the novel work is concluded in section 5.

REVIEW OF MAPREDUCE

MapReduce provides a programming paradigm for perform-ing distributed computation on computer clusters. Figure 1 gives an overview of the MapReduce framework. In a MapRe-duce system such as hadoop, the user program forks a

Master controller process and a series of Map tasks (*Mappers*) and Reduce tasks (*Reducer*).

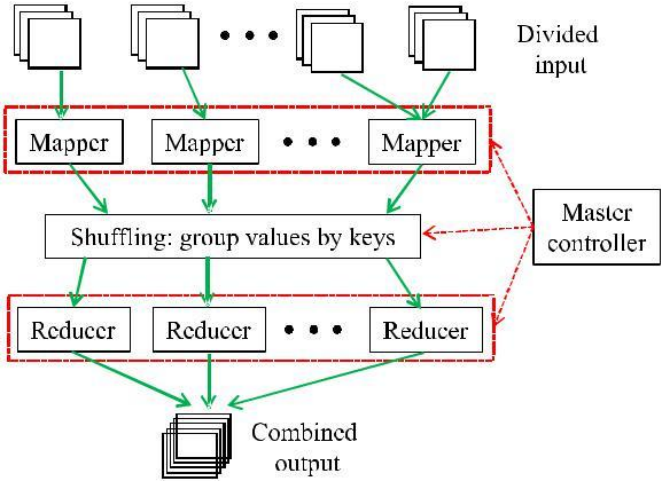

Figure 1: MapReduce Framework

Reduce tasks (*Reducers*) at different computers (nodes of a cluster). The responsibilities of the Master involve creating some number of Mappers and Reducers and keeping track of the status of each Mapper and Reducer (executing, complete or idle).

The computation in one MapReduce job consists of two phases, i.e., a map phase and a reduce phase. In the Map phase, the input dataset (stored in a distributed le system, e.g., HDFS) is divided into a number of disjoint subsets which are assigned to mappers in terms of <key, value> pairs. In parallel, each Mapper applies the user-speci ed map func-tion to each input <key, value> pair and outputs a set of inter-mediate <key, value> pairs which are written to local disks of the map computers. The underlying system pass the locations of these intermediate pairs to the master who is responsible to notify the reducers about these locations. In the Reduce phase, when the reducers have remotely read all intermediate pairs, they sort and group them by the intermediate keys. Each Reducer literately invokes a user-speci ed reduce function to process all the values for each unique key and generate a new value for each key. The resulting <key, value> pairs from all of the Reducers are collected as nal results which are then written to an output file.

In the MapReduce system, all the map tasks (and reduce tasks) are executed in a fully parallel way. Therefore, high-level parallelism can be achieved for data processing through the use of the MapReduce model. In recent years, there have been some parallel learning algorithms [13] [14] using the MapReduce framework for efficient implementation.

PROPOSED METHODOLOGY:

The first deep learning model that is successfully trained is the deep belief network [15,16]. Different from the stacked auto-encoder, the deep belief network is stacked by several restricted Boltzmann machines. The restricted Boltzmann machine consists of two layers, i.e., visible layer v and hidden layer h, as presented in Figure. 2 [17,18]

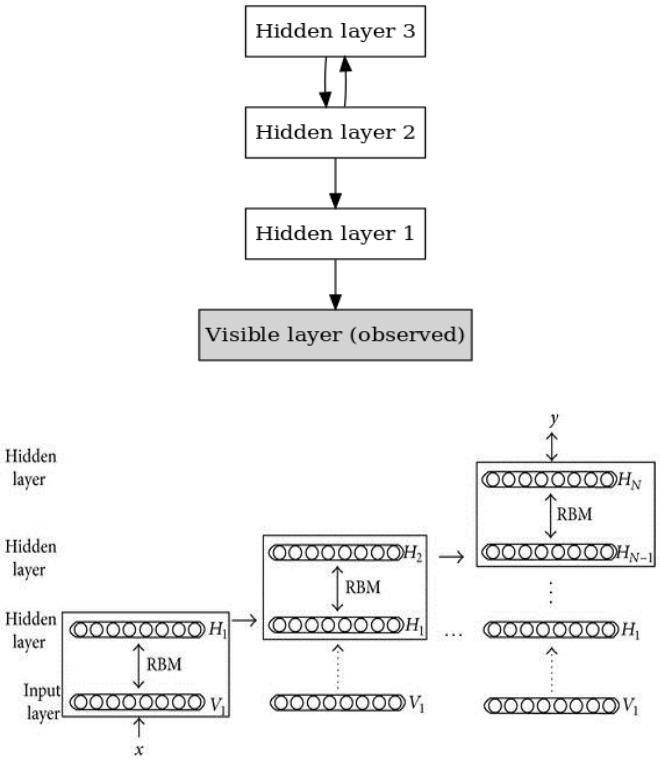

Figure 2: RBM Structure

A typical restricted Boltzmann machine uses the Gibbs sampling to train the parameters. Specially, the restricted Boltzmann machine uses the conditional probability P(h|v) to calculate the value of each unit in the hidden layer and then uses the conditional probability p(h|v) to calculate the value of each unit in the visible layer. This process is performed repeatedly until convergence.

Let $v_i(0\leq i\leq N)$ be a binary variable of input neurons, N is the number of input neurons, $h_i(0\leq j\leq M)$ be a binary variable of hidden neuron, where M is the number of hidden neurons. The energy function $E(i,j)$ for input vector $i \in \{0,1\}^N$ and hidden vector $j \in \{0,1\}^M$ [19] is represented as

$$\boldsymbol{E(v,h) = \sum_i b_i v_i - \sum_i c_j h_j - \sum_i \sum_j v_i W_{ij} h_j} \quad \textbf{(1)}$$

The joint probability distribution of v and h is represented as

$$\boldsymbol{p(v,h) = \frac{1}{Z}\exp(-E(v,h)),\ Z=\sum_v \sum_h \exp(-E(v,h))} \quad \textbf{(2)}$$

Where

b_i *and* c_i are the coefficients for v_i and h_j respectively,

Weight w_{ij} is the parameter between v_i and h_j,

Z is the partition function,

P(v,h) is a probability function, calculates sum over all possible pairs of visible and hidden vectors.

DBN copy the joint distribution between input vector v and the h hidden layers h^k as follows

$$\boldsymbol{P(a, h^1,\ldots,h^h)= (\pi_{k=0}^{h-2}\ p(h^k(h^{k+1}))\ P(1^{h-1},\ldots,h^h)} \quad \textbf{(3)}$$

Where $a = h^0$,

$$\boldsymbol{P(h^{k-1}|\ h^k)} \quad \textbf{(4)}$$

is a conditional distribution for the visible units conditioned on the hidden units of the RBM at level k, and $\boldsymbol{P(h^{h-1},h^b)}$ is the visible hidden joint distribution in the RBM[20]. Several restricted Boltzmann machines can be stacked into a deep learning model, called deep belief network, as presented in Fig. 2 [21].

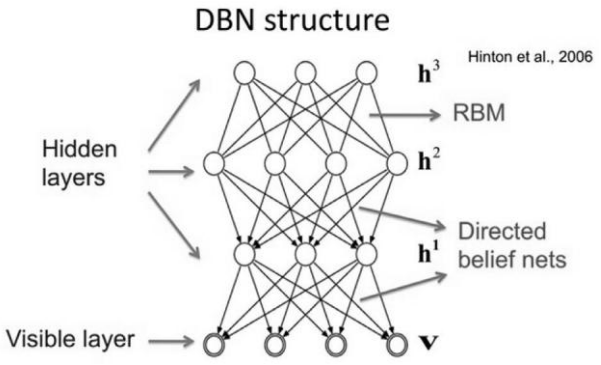

Figure 3: DBN Structure

Similar to the stacked auto-encoder, the deep belief network is also trained by a two-stage strategy. The pre-training stage is used to train the initial parameters in a greedy layer-wise unsupervised manner while the fine-tuning stage uses the supervised strategy to fine-tune the parameters with regard to the labeled samples by adding a softmax layer on the top layer. Deep belief networks have a wide range applications in image classification [22,23] and acoustic modeling [24] and so on [25].

Procedure 1 mrjob_RBM

1: Initialize the variables

2: **for** each epoch **do**

3: *Map phase*

Input: <mapID, valuelist>

Take the values and perform Gibbs sampling to compute the approximate gradients of W, b and c

Output: <key, valuelist>

4: *Reduce phase*

Input: <key, valuelist>

Sum up the approximate gradients to get the incre-ments of W, b and c, and then update them

Output: <mapID, valuelist>

5: **end for**

Output: the learned W, b and c

3.I) MAP PHASE

For each mapper, the corresponding mapper ID (a number) is as the input *key* and the input *value* is a list of values. Each of the values has two elements: the rst is a string (e.g., ${}^0W\ {}^0$) identifying the type of this value, the second is the corresponding data (e.g., it can be an $M\ N$ matrix if the rst element is ${}^0W\ {}^0$). In every epoch (except the rst one), the *value* is the output of the reducer in the previous epoch, which is the updated W, b and c and their accumulated approximate gradients.

The input dataset D is divided into a number of disjoint subsets which are stored as a sequence of les (blocks) on Hadoop Distributed File System (HDFS). After reading all of the key-value pairs, each mapper loads one subset from the HDFS into memory. Given the information, each mapper can compute the approximate gradients of the weight and biases by going through all the mini-batches of the subset of the training dataset. Each mapper will emit three types of intermediate *key*s: *delta*_W, *delta*_b and *delta*_c which represent the increments of W, b and c, respectively, and the intermediate *value*s have three elements: the value of *delta*_W, *delta*_b or *delta*_c, the corresponding increment and the current epoch index.

3.II) REDUCE PHASE

For the training of RBM, there are three reducers in ideal case. Each reducer reads as input one type (i.e., *delta*_W, *delta*_b or *delta*_c) of the intermediate key-value pairs, and applies the reduce function to rst calculate the increments and then update parameter. The reducer takes the mapper ID as the output *key*, and the resulting increment and the updated parameter as the output *value*.

EXPERIMENTAL RESULTS:

The proposed method has been implemented by R tool. This section will demonstrate the performance of the mapreduced DBN on two datasets namely pest formation in crops and medical dataset. Pest Formation in crops are presented by agricultural department which is used to classify the percentage of crops affected by the pest, the medical datasets are generated by diabetology hospital which is used to classify

the readmission rate of the diabetic patients. Above two datasets are divided into three test cases. In the first test case, the training process includes approximately 60% of instances and testing process includes 40% of an instance. In the second and third case, the training includes 70 %, 80%, and training includes 30%,20% of instance respectively. The test cases are listed in the Table 1.

Pest Formation in Crop dataset: This dataset used to classify the crops infected by pest based on color. It classifies crops color into three categories Green(normal),Yellow(mildly affected), white(fully affected). It has 50000 instances and 7 attributes. The accuracy for three test case are discussed on the Table 2. Fig.4 demonstrates the accuracy chart for Pest formation in crops dataset with three test cases.

Readmission rate of diabetic patients in medical dataset: This dataset used to classify the diabetic patients readmission rate in the hospital. It classifies the patients readmission as NO,>30 and <30. It includes 200000 instance and 51 attributes. The accuracy for three test case are discussed on the Table 5. Fig.5 demonstrates the accuracy chart for medical dataset with three test cases.

Table 1: Test Case Details

Dataset	Test Case 1		Test Case 2		Test Case 3	
	Training (instance)	Testing (instance)	Training (instance)	Testing (instance)	Training (instance)	Testing (instance)
Pest Formation in Crops	3000	2000	3500	1500	4000	1000
Medical Dataset	120000	80000	140000	60000	160000	40000

Table 2: Accuracy for PEST Formation dataset

Number of depth	TEST CASE 1			TEST CASE 2			TEST CASE 3		
	Accuracy in DBN	Accuracy in MapreduceDBN	Training Time	Accuracy in DBN	Accuracy in Mapreduce DBN	Training Time	Accuracy in DBN	Accuracy in MapreduceDBN	Training Time
1	62.44	65.34	817.4	64.57	65.90	819.5	66.7	69.27	821.4
2	63.2	67.54	823.15	65.71	66.61	824.80	68.861	71.76	825.74
3	64.25	69.64	825	70.99	71.49	826.1	73.49	73.58	830.95
4	66.128	70.83	829.1	71.84	73.82	831.71	75.4	76.63	832.4
5	68.41	71.45	832.48	72.963	74.980	834.5	76.73	78.76	834.889
6	70.54	73.36	836.57	75.64	76.94	836.69	79.8	80.69	837.5

Table 3: Accuracy for medical dataset

Number of depth	TEST CASE 1			TEST CASE 2			TEST CASE 3		
	Accuracy in DBN	Accuracy in Mapreduce DBN	Training Time	Accuracy in DBN	Accuracy in Mapreduce DBN	Training Time	Accuracy in DBN	Accuracy in Mapreduce DBN	Training Time
1	57.12	62.04	1007.8	64.09	66.5	1019.7	68.7	70.81	1027.5
2	63.72	64.89	1021.61	66.57	67.69	1024.1	70.98	72.47	1031.7
3	65.87	66.964	1029.4	68.71	69.89	1028.4	74.36	76.58	1039.5

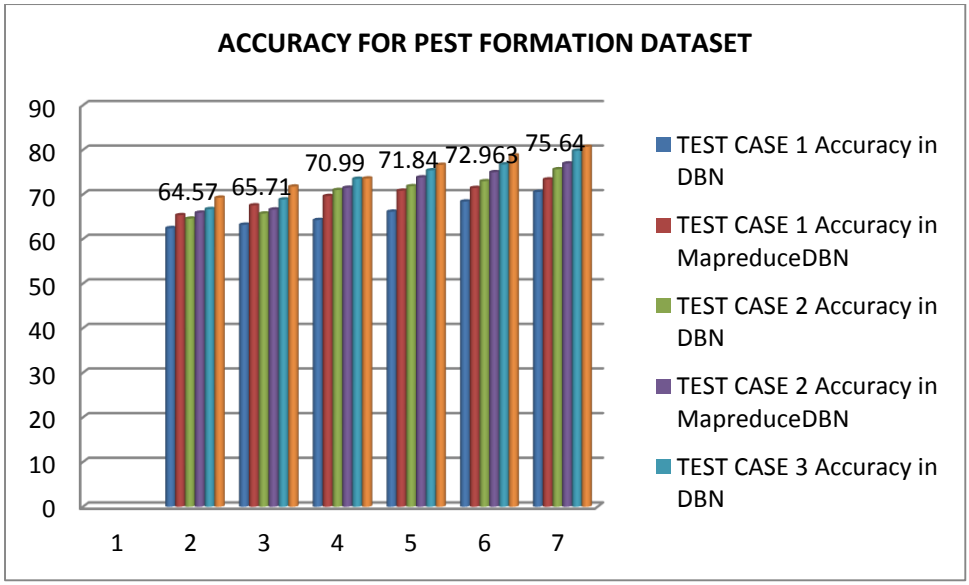

Figure 4: Accuracy for pest formation dataset

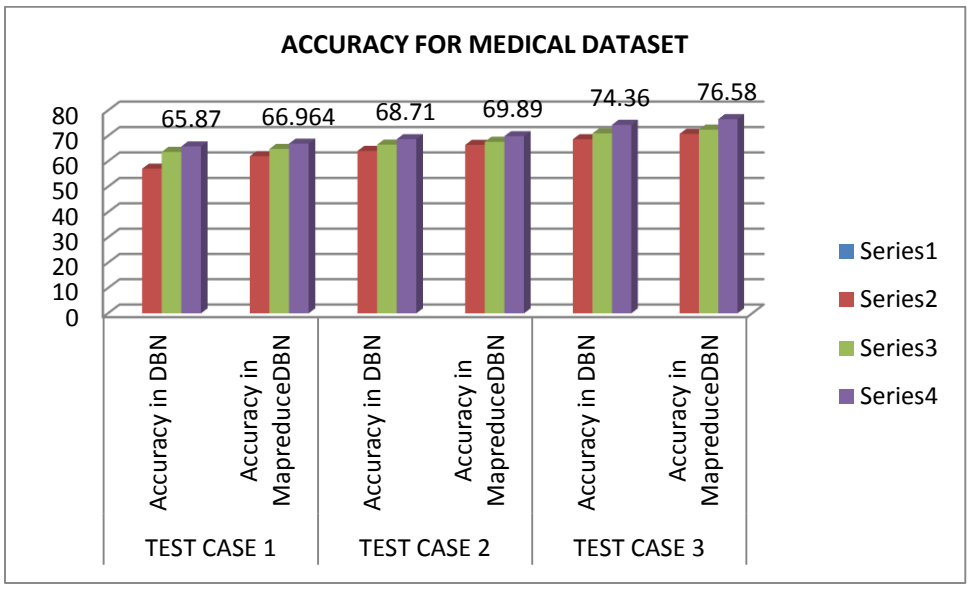

Figure 5: Accuracy for medical dataset

CONCLUSION

Above experimental results demonstrate the performance of MapReduce Deep Belief Network on classification accuracy. In the future work we modify the Deep Belief Network weight updating process by adding learning rate decay which is used to reduce the training time of the network and also increase the classification accuracy.

REFERENCE

[1] L. Kuang, L.T. Yang, Y. Liao, An integration framework on cloud for cyber physical social systems big data, IEEE Trans. Cloud Comput. (2015). 10.1109/ TCC.2015.2511766

[2] Y. Li, X. Qi, M. Keally, Z. Ren, G. Zhou, D. Xiao, S. Deng, Communication energy modeling and through joint packet size analysis of BSN and wifi networks, IEEE Trans. Parallel Distrib. Syst. 24 (9) (2013) 1741–1751.

[3] C.L.P. Chen, C. Zhang, Data-intensive applications, challenges, techniques and technologies: a survey on big data, Inf. Sci. 275 (2014) 314–347. 2014

[4] X. Wu, X. Zhu, G.-Q. Wu, W. Ding, Data mining with big data, IEEE Trans. Knowl. Data Eng. 26 (1) (2014) 97–107.

[5] G.B. Orgaz, J.J. Jung, D. Camacho, Social big data: recent achievements and new challenges, Inf. Fusion 28 (2016) 45–59.

[6] X. Chen, X. Lin, Big data deep learning: challenges and perspectives, IEEE Access 2 (2014) 514–525.

[7] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.

[8] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521(7553) 436–444.

[9] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61(2015).

[10] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[11] R. Raina, A. Madhavan, A. Ng, Large-scale deep unsupervised learning using graphics processors, Proceedings of International Conference on Machine Learning, (2009), p. 873 C880. ACM

[12] V. Sze, Y. Chen, T. Yang, J. Emer, Efficient processing of deep neual networks: a tutoria and survey, 2017, arXiv:1703.09039.

[13] C. Chu et al., ``Map-reduce for machine learning on multicore," in Neural Information Processing Systems, vol. 19. Cambridge, MA, USA: MIT Press, 2007, p. 281.

[14] K. Zhai, J. Boyd-Graber, N. Asadi, and M. L. Alkhouja, ``Mr. LDA: A exible large scale topic modeling package using variational infer-ence in MapReduce," in Proc. 21st Int. Conf. World Wide Web, 2012.

[15] G. W. Taylor, G. E. Hinton, and S. T. Roweis, ``Modeling human motion using binary latent variables," in Advances in Neural Infor-mation Processing Systems. Cambridge, MA, USA: MIT Press, 2006, pp. 1345 1352.

[16] B. Panda, J. S. Herbach, S. Basu, and R. J. Bayardo, ``PLANET: Massively parallel learning of tree ensembles with MapReduce," Proc. VLDB Endow-ment, vol. 2, no. 2, pp. 1426 1437, 2009.

[17] T. Sun, C. Shu, F. Li, H. Yu, L. Ma, and Y. Fang, ``An ef cient hierarchical clustering method for large datasets with map-reduce," in Proc. Int. Conf. Parallel and Distributed Comput., Appl. Technol., 2009, pp. 494 499.

[18] W. Zhao, H. Ma, and Q. He, ``Parallel K -means clustering based on MapReduce," in Proc. Int. Conf. Cloud Comput., 2009, pp. 674 679.

[19] Hinton. G. E.: A Practical Guide to Training Restricted Boltzmann Machines. *Neural Networks Tricks of the Trade Lecture Notes in Computer Science*. vol. 7700,.pp. 599-619, 2012.

[20] T. Li, J. Zhang, Y. Zhang, Classification of hyperspectral image based on deep belief networks, Proceedings of IEEE International Conference on Image Processing, (2014), pp. 5132–5136. IEEE

[21] J. Niu, X. Bu, Z. Li, Y. Wang, An improved bilinear deep belief network algorithm for image classification, Proceedings of International Conference on Computational Intelligence and Security, (2014), pp. 189–192. IEEE

[22] A. Mohamed, G.E. Dahl, G. Hinton, Acoustic modeling using deep belief networks, IEEE Trans. Audio Speech Lang. Process. 20 (1) (2012) 14–22.

[23] X. Zhang, J. Wu, Deep belief networks based voice activity detection, IEEE Trans. Audio Speech Lang. Process. 21 (4) (2013) 697–710.

[24] W. Huang, G. Song, H. Hong, K. Xie, Deep architecture for traffic flow prediction: deep belief networks with multitask learning, IEEE Trans. Intell. Transp. Syst. 15a. (2014) 2191–2201.

[25] R. Sarikaya, G.E. Hinton, A. Deoras, Application of deep belief networks for nat-ural language understanding, IEEE/ACM Trans. Audio Speech Lang. Process. 22 (2014) 778–784.