**Research Article**

**Distributed Data Mining Based on Deep Neural Network for Wireless Sensor Network**

**Chunlin Li, Xiaofu Xie, Yuejiang Huang, Hong Wang, and Changxi Niu**

*Southwest Communication Institute, China Electronics Technology Group Corporation, Chengdu 610041, China*

Correspondence should be addressed to Chunlin Li; li.chl@foxmail.com

Received 16 July 2014; Accepted 25 September 2014

Academic Editor: Haigang Gong

Copyright © 2015 Chunlin Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As the sample data of wireless sensor network (WSN) has increased rapidly with more and more sensors, a centralized data mining solution in a fusion center has encountered the challenges of reducing the fusion center’s calculating load and saving the WSN's transmitting power consumption. Rising to these challenges, this paper proposes a distributed data mining method based on deep neural network (DNN), by dividing the deep neural network into different layers and putting them into sensors. By the proposed solution, the distributed data mining calculating units in WSN share much of fusion center's calculating burden. And the power consumption of transmitting the data processed by DNN is much less than transmitting the raw data. Also, a fault detection scenario is built to verify the validity of this method. Results show that the detection rate is 99%, and WSN shares 64.06% of the data mining calculating task with 58.31% reduction of power consumption.

**1. Introduction**

With the developing of wireless sensor network technology, a variety of applications based WSN appear, such as land cover classification [1], SCR node detection in vehicular network [2], fault detection [3, 4], and groundwater quality estimation [5]. Traditionally, these applications analyze sample data in a fusion center [6]. However, when a large scale of WSN contains thousands of sensors, the performance for processing the sampling data is limited by the fusion center’s hardware, which is too expensive to be updated frequently. Moreover, the network transmitting consumes a large amount of power, especially for wireless relaying nodes.

Data mining techniques, which have been developed to extract useful information from massive data for years, are considered to be an effective tool for analyzing massive data. In the 1990s, shallow data mining models like support vector machine (SVM), boosting, and logistic regression are proposed. And they have successfully been used in massive data analysis since 2000 [7]. Using a shallow data mining algorithm could improve the fusion center's analysis performance, but the power consumption problem is still unsolved. Or we can execute these algorithms in the sensors to reduce the transmitting data amounts, but these algorithms are usually too complex to be executed in the wireless sensors.

In 2006, Professor Hinton [8] proposed a deep data mining model called deep neural network, which could be used to extract the internal representation and reduce the data dimensionality. It has helped researchers achieve the state-of-the-art results on voice recognizing, image recognizing, and semantics analysis [9–11]. Moreover, DNN employs a layered structure, which can be divided by layers and executed in different hierarchies of the WSN.

In order to improve the fusion center’s data mining performance and save the transmitting power consumption, this paper proposes a distributed data mining method based on DNN for WSN. Section 2 briefly introduces the DNN and points out the problems needed to be solved in detail. Section 3 presents the principle of the distributed data mining method based on DNN. Section 4 proposes the training method of the DNN, as well as the tradeoff between calculating and transmitting power consumption. Simulation is presented in Section 5 to verify the validity of proposed method. A conclusion is given in the last section.

**2. Preliminaries and Problem Formulation**

**2.1. Introduction of DNN.** Although there is no exact definition of DNN, we can depict it with some typical features, like
self-taught learning ability, internal representation exacting ability [12], and building multilayer perceptron (MLP) with more than one hidden layer [13]. Actually, the self-taught learning ability and internal representation exacting ability of DNN are developed based on a basic neural network called autoencoder (AE). An AE network can be trained without any predefined labels, which saves a large amount of manual work.

In this subsection, we introduce the principle of AE with a three-layer network (3-2-3) shown in Figure 1. Vector $X$ represents the input data, and each neural layer's output is $a^{(i)}$, where $i$ is the number of the neural layers. And $a^{(1)} = X$. When training the artificial neural network (ANN), we need to give corresponding output to training inputs. Usually, these outputs are set manually. However, if we make the outputs equal the inputs ($a^{(3)} = X$), such manual work is not needed anymore. Then, the question of what the outputs of the second layer represent appears. The size of $a^{(2)}$ especially is less than the size of $X$. D. Yu and L. Deng [13] figured out that the $a^{(2)}$ can be internal representation of the inputs, which means that we can represent the input space with a lower dimensionality of space. If the training of the three-layer neural network is finished, the network of the first two layers would gain the ability to extract internal representations of input data, which are composed of the simplest AE and also the basic unit of DNN.

Now we can build deep neural networks like Figure 2. The main idea is training these layers one by one, which is called greedy layer-wise training (GLT) [14]. If we want to train layer $n + 1$, where $n > 0$, we select layers $n$, $n + 1$, and $n + 2$. And these three layers are trained as the simplest AE. Based on such training method, we can train the whole network from the second layer to the last layer one by one. And Le [15] used a DNN network trained by GLT to analyze images. The results show that the internal representations of different layers are just like the representations of V1 and V2 zones in brain of a human's visual process system.

2.2. Advantages of Applying Distributed DNN Data Mining to WSN. In this subsection, we discuss what advantages can be brought by using a distributed data mining based on DNN. Besides the advantages of DNN introduced in Section 2.1 using a distributed structure can bring more advantages especially for WSN. In conclusion, we can gain at least four advantages.

(a) There is no need to label amounts of training data manually for different applications, and the training can be finished automatically.

(b) The internal representations can be combined with other data mining algorithms, improving these algorithms to achieve better results.

(c) The dimensionality ability of DNN can reduce the transmitting data via WSN and save the WSN's power.

(d) The distributed calculating reduces the calculating burden of the fusion center, which can save a lot of money for updating hardware.

2.3. Challenge of Applying Distributed DNN Data Mining to WSN. Before we use a distributed DNN data mining structure for WSN, there are two challenges needed to be overcome.

One challenge is training the distributed layers of DNN. When using a distributed data mining structure, some of the nodes in WSN need to take the data mining task. And such a node is called a calculating unit in this paper. Obviously, we need to ensure consistency of the data processed by these distributed calculating units. This means that the DNN layers in each distributed calculating unit have the same parameters. Generally, training these distributed DNN layers in each calculating unit separately may lead to different parameters. A solution is training the DNN in the fusion center and sending the corresponding parameter to every calculating unit. However, training in the fusion needs number of samples, which are transmitted via the WSN. And transmitting these
samples consumes lots of the WSN’s power. Thus, the training problem becomes the first challenge.

The other challenge is the tradeoff between calculating power consumption and transmitting power consumption. When the calculating units join the data mining process, extra power consumption is needed to support the calculating. And this may counterbalance the saving power by reducing transmitting data. Pottie and Kaiser [16] pointed out that the power consumption of transmitting a bit to 100 meters away equals the power consumption of executing about 3000 instructions. Such a relationship between calculating and transmitting power consumption infers that the design of distributed data mining should trade off these power consumptions. And this is also a challenge.

3. Principle of Distributed Data Mining Based on DNN

In this section, we introduce the principle of distributed data mining based on DNN proposed in this paper. Consider that there is a WSN with a fusion center aggregated by three levels (Figure 3(a)) and a 3-layer DNN (Figure 3(b)). We can note that the topology of WSN and the structure of DNN are similar in hierarchy. A feasible solution is dividing the DNN into layers and putting them into different levels of the WSN. Figure 3(c) gives an example of dividing the DNN into two parts and putting them in the fusion center and all sensors.

Generally, assume that a WSN is aggregated by \( m \) levels, and a DNN has \( n \) layers. If we divide the \( n \) layers into \( k \) parts (\( k \leq m, n \)), and each part is executed in the calculating units in the corresponding level of \( h_i \) in WSN, then the principle of D-DMBDD (Distributed Data Mining Based on DNN) can be depicted as the following steps.

Step 1: let \( i = 1 \). The sensors sample the raw data, and these data are processed in the calculating units in level \( h_1 \) by the first part of DNN; send the result to the calculating units in level \( h_1 + 1 \), and \( i = i + 1 \).

Step 2: calculating the inputs from the calculating units of former level, if \( i \geq k \), go to Step 4.
Step 3: if $h_i \geq m$, go to Step 5. Else, go to Step 2 and send the result to the calculating units in level $h_i + 1$, and $i = i + 1$.

Step 4: send data to the fusion center.

Step 5: data mining is finished.

4. Training and Design Methods

In this section, we propose the solutions for the challenges referred to in Section 2.3. In Section 4.1, a random source data selection method is used to solve the training problem, and the tradeoff between processing and transmitting power consumption is discussed in Section 4.2.

4.1. Training the Distributed DNN. Before applying DNN to data mining, we need to train the DNN in the fusion center first. As shown in Figure 4, the training data are sampled from all the WSN sensors, and the trained DNN parameters are sent to the DNN layers distributed in different calculating units.

Although a wireless sensor network can supply a mass of training data, these data also consume a lot of the network’s power. Actually, a sensor’s sample data do not change in a short time. Thus we can choose one of them to train the DNN. The problem is that we do not know when the data change. A random data selection method [17] has been proved useful in solving this problem, and a digital recognition research [18] showed that a random selection of 10% training can achieve a good result. So, a random selection method can effectively reduce lots of redundant data to be transmitted.

Then, we give the training flow with a random selection method as follows.

Step 1: the fusion center randomly generates a sensor’s ID and sends a request to the sensor.

Step 2: the selected sensor gets the request and sends the sample data.

Step 3: fusion center receives the training data from the selected sensor and sends the data to the GLT algorithm.

Step 4: the GLT algorithm checks whether the training result achieves the stop condition. If YES, go to step 5. Else, go to Step 1.

Step 5: the fusion center sends each part of the DNN’s configuration data to the corresponding calculating unit.

4.2. Tradeoff between Calculating and Transmitting Power Consumption. The distribution hierarchy of DNN depends on its application. However, any distribution hierarchy should be constrained by power consumption. In this subsection, we discuss the rule of designing the distribution hierarchy based on the tradeoff between calculating and transmitting power consumption.

Assume that there is a calculating unit, and it executes $c$ instructions to finish its data mining task. Each instruction consumes $E_i$ power. Moreover, the calculating unit consumes $E_t$ power sending a bit to the target node without any disturbing and attenuation. And all the disturbing and attenuation effects lead to more power consumption of $E_o$. Then we assert that a calculating unit can accept the DNN part if the following formula is satisfied:

$$cE_i \leq (b_i - b_o)(E_t + E_o), \quad (1)$$

where $b_i$ is the size of the calculating unit’s input in bit and $b_o$ is the size of the calculating unit’s output in bit, $b_i \geq b$. If $E_o$ is set to 0, then we have

$$c \leq \frac{b_i - b_o}{E_t} \quad (2)$$

Obviously, if formula (2) is satisfied, formula (1) must be satisfied too. Actually, formula (2) is a conservative constraint. It determines the upper limit calculating task which a calculating unit can take.

5. Simulation

5.1. Simulation Description. To verify the distributed data mining method, we create an application scenario of fault detection in Matlab 2010a. The DNN’s structure contains two parts (shown in Figure 5), the data representation analysis part and the classifying part. The former part uses a 2-layer AE network to extract the internal representations of the sample data. And the other part uses a Softmax regression algorithm. Both parts adopt Sigmoid function as the activation function.

The simulated WSN has three levels, one fusion center, ten transmitting relays, and two hundred wireless sensor nodes. Every sensor is a calculating unit with an ARM9 CPU. And the mean distance between each sensor is 100 meters. The source of the simulation sample data is KDD99 database, which have 41 fields. Each sensor samples 15,000 raw data. 300,000 sample data are labeled manually with 23 types. 1/3
of them are used to train the Softmax algorithm and the remaining data are used to test the algorithm.

This paper uses three criterions to verify the proposed method, calculating share rate, fault detection rate, and power consumption rate.

(a) Assume that the calculating task taken by WSN needs executing $C_{\text{DNN-WSN}}$ instructions. And the proportion of the total data mining calculating shared by the WSN is $CR_{\text{WSN}}$. $C_{\text{DNN}}$ represents the instructions executed by AE, and $C_{\text{Softmax}}$ represents the instructions executed by Softmax regression. Then the calculating share rate is defined as follows:

$$CR_{\text{WSN}} = \frac{C_{\text{DNN-WSN}}}{C_{\text{DNN}} - C_{\text{DNN-WSN}} + C_{\text{Softmax}}}.$$  \hspace{1cm} (3)

(b) Fault detection rate is defined as the correctly detected fault counts divided by the total faults.

(c) Power consumption rate is defined as the calculating units’ power consumption of executing DNN divided by the power consumption without executing DNN.

5.2. Design the Distributed DNN. This subsection mainly discusses the design of the distributed DNN based on formula (2).

In ARM9, a multiple instruction needs seven execution cycles \[23\], which equals seven add instructions. However, Sigmoid function is more complicated, which is given in the following formula:

$$a = \frac{1}{1 + e^{-z}}.$$ \hspace{1cm} (4)

According to Taylor expansion, if we keep the accuracy of two decimal places, $e^{-z}$ can be translated into a series of calculations with for multiple operations and four add operations. In the simulation network, we have

There are two power control strategies. One is calculating $a^{(2)}$ in the calculating units, and the other is calculating $a^{(2)}$ in the fusion center. Considering both cases, Table 1 lists the parameters of formula (2).

According to Table 1, if calculating unit calculates $a^{(2)}$, the output ($b_{o}$) is constrained to have at most 38 bytes. Otherwise the output ($b_{o}$) is constrained to have at most 40 bytes.

5.3. Simulation Result. The calculating share rate is the first checking criterions, which can be directly calculated based on the data given by the simulation assumption. In Table 1, $C_{\text{DNN}}$ is 1640 bytes, and $C_{\text{Softmax}}$ is 920 bytes. When the calculating units calculate $z^{(1)}$, $C_{\text{DNN-WSN}}$ is 328 bytes, and then we get $CR_{\text{WSN}}$ 12.81% according to formula (3). When the calculating units calculate $a^{(2)}$, $CR_{\text{WSN}}$ is 64.06%.

Then the simulation checks the effect on fault detection rate with different hidden layer size. Result in Figure 6 shows that when the hidden layer has more than 15 neurons, the detection rate becomes stable. And if the hidden layer has less than 12 neurons, the detection rate decreases rapidly. This simulation infers that the fault detection rate does not increase linealy with the hidden layer size. And this verifies
that the raw data has lots of redundant information, and the DNN can effectively extract the internal representations to help improving the data mining.

Moreover, Table 2 lists the state-of-the-art results of four different data mining algorithms. Compared to Figure 6, when the hidden layer size is bigger than 16, half of detection rates are better than these rates listed in the table. Then, we can assert that the training method is effective, and the distributed data mining method based on DNN improves the data mining’s performance.

To check power consumption rate of the two strategies referred to in Section 5.2, we run another simulation. Figure 7 gives the simulation results. As shown in the figure, both ratios increase as the hidden layer size increases. The difference between the two cases is quite small.

Combining the result in Figures 6 and 7, setting the hidden layer size 16 is quite reasonable. Then we get the calculating sharing rate is 64.06%, and the power consumption rate is 41.169%.

In conclusion, the above simulations verify the four advantages declared in Section 2.2. Then we can assert that the D-DMBDD method achieves its goal. Moreover, the training and design methods are also proved valid.

6. Conclusion

In this paper, we have presented a distributed data mining method for WSN based on DNN by solving two challenges, which are training the distributed layers of DNN and tradeoff between calculating power consumption and transmitting power consumption. The proposed solution can learn internal representations from unlabeled data collected by distributed sensors. And these representations improve data mining results. Additionally, a distributed DNN solution saves both power consumption of WSN and costs of updating hardware for mass data processing. An application simulation verifies the validity of this method. The results show that performance of data mining for WSN has been improved. The distributed calculating mode is especially suitable for large scale WSN. As a future work, we are planning more researches for additional improvements with sample data noise filtering and data mining with deeper DNN layers.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] B. Gong, J. Im, and G. Mountrakis, “An artificial immune network approach to multi-sensor land use/land cover classification,” Remote Sensing of Environment, vol. 115, no. 2, pp. 600–614, 2011.
[2] H. Gong, L. Yu, and X. Zhang, “Social contribution-based routing protocol for vehicular network with selfish nodes,” International Journal of Distributed Sensor Networks, vol. 2014, Article ID 753024, 12 pages, 2014.
[3] M. Yu, H. Mokhtar, and M. Merabti, “Fault management in wireless sensor networks,” IEEE Wireless Communications, vol. 14, no. 6, pp. 13–19, 2007.
[4] N. Boudriga, “On a controlled random deployment WSN-based monitoring system allowing fault detection and replacement,” International Journal of Distributed Sensor Networks, vol. 2014, Article ID 101496, 13 pages, 2014.
[5] Y. Kühçan, G. Tuna, G. Gezer, K. Gulez, O. Arkoc, and S. M. Potirakis, “ANN-based estimation of groundwater quality using a wireless water quality network,” International Journal of Distributed Sensor Networks, vol. 2014, Article ID 458329, 8 pages, 2014.
[6] L. Li, A. Scaglione, and J. H. Manton, “Distributed principal subspace estimation in wireless sensor networks,” IEEE Journal on Selected Topics in Signal Processing, vol. 5, no. 4, pp. 725–738, 2011.
[7] K. Yu, L. Jia, Y. Chen, and W. Xu, “Deep learning: yesterday, today, and tomorrow,” Journal of computer Research and Development, vol. 50, no. 9, pp. 1799–1804, 2013.
[8] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, 2006.
[9] N. Boulinger-Lewandowski, Y. Bengio, and P. Vincent, “Modeling temporal dependencies in high-dimensional sequences: application to polyphonic music generation and transcription,” in Proceedings of the 29th International Conference on Machine Learning (ICML’12), pp. 1159–1166, July 2012.
[10] F. Seide, G. Li, and D. Yu, “Conversational speech transcription using Context-Dependent Deep Neural Networks,” in Proceedings of the 12th Annual Conference of the International Speech
[11] Q. V. Le, M. Ranzato, R. Monga et al, “Building high-level features using large scale unsupervised learning,” in Proceedings of the 29th International Conference on Machine Learning, pp. 81–88, Edinburgh, UK, July 2012.

[12] I. Arel, D. C. Rose, and T. P. Karnowski, “Deep machine learning—a new frontier in artificial intelligence research,” IEEE Computational Intelligence Magazine, vol. 5, no. 4, pp. 13–18, 2010.

[13] D. Yu and L. Deng, “Deep learning and its applications to signal and information processing,” IEEE Signal Processing Magazine, vol. 28, no. 4, pp. 145–154, 2011.

[14] Y. Bengio, “Learning deep architectures for AI,” Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1–27, 2009.

[15] Q. V. Le, “Building high-level features using large scale unsupervised learning,” in Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP ’13), pp. 8595–8598, May 2013.

[16] G. J. Pottie and W. J. Kaiser, “Wireless integrated network sensors,” Communications of the ACM, vol. 43, no. 5, pp. 51–58, 2000.

[17] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” Journal of Machine Learning Research, vol. 13, pp. 281–305, 2012.

[18] A. Ng, J. Ngiam, C. Y. Foo et al., “Deep Learning,” 2014, http://deeplearning.stanford.edu/wiki/index.php/.

[19] H. Liu, Y. Jian, and S. Liu, “A new intelligent intrusion detection method based on attribute reduction and parameters optimization of SVM,” in Proceedings of the 2nd International Workshop on Education Technology and Computer Science (ETCS ’10), pp. 202–205, March 2010.

[20] S. Mabu, C. Chen, N. Lu, K. Shimada, and K. Hirasawa, “Intrusion-detection model based on fuzzy class-association-rule mining using genetic programming net-work,” IEEE Transactions on Systems, Man and Cybernetics C: Applications and Reviews, vol. 4, no. 1, pp. 130–139, 2011.

[21] S.-Y. Wu and X.-G. Tian, “Method for anomaly detection of user behaviors based on hidden Markov models,” Journal on Communications, vol. 28, no. 4, pp. 38–43, 2007.

[22] L. Zhang, Z.-Y. Bai, S.-S. Luo, K. Xie, G.-N. Cui, and M.-H. Sun, “Integrated intrusion detection model based on rough set and artificial immune,” Journal on Communications, vol. 34, no. 9, pp. 166–176, 2013.

[23] ARM, ARM920T Product Overview, ARM Ltd., 2003.
Submit your manuscripts at http://www.hindawi.com