The Open Automation and Control Systems Journal

An Anomaly Detection Method Based On Deep Learning

Deng Hongli*, Yang Tao and Gao Jiangjin

The Open Automation and Control Systems Journal, Volume 7, 2015

ISSN: 1874-4443/15
DOI: 10.2174/1874444320150610E005

Article Type: Review Article

Received: January 16, 2015
Revised: March 23, 2015
Accepted: May 31, 2015

Provisional PDF Publication Date: June 16, 2015

An Anomaly Detection Method Based On Deep Learning

Deng Hongli*, Yang Tao and Gao Jiangjin

China West Normal University, Nanchong, Sichuan, 637002, P.R. China

Abstract: In order to improve the accuracy of anomaly detection and solve the problem of feature extraction, this paper proposed an anomaly detection method based on deep learning. It built a deep neural network model with multiple hidden layers to learn features of data. In this model, a real zero-value sparse auto-encoder (RZSAE) is proposed to achieve the pre-training of network. The learned features fully characterize the inherent information of the original data and improve data discrimination, so that the accuracy of anomaly detection is improved. This method does not require manual extraction of features, and the learning process is unsupervised, avoiding the difficulty of acquiring the labeled data. Experimental results show that this method can effectively learn the essential characteristics of the data, and anomaly detection methods with the learned features could significantly improve the detection rate.

Keywords: deep learning; anomaly detection; feature representation.

1. INTRODUCTION

Intrusion detection technology [1] is to prevent or reduce the threat of cyber-attacks, under the condition that the network performance is not affected. It can be divided into misuse detection and anomaly detection technology. The anomaly detection can detect unknown attacks, so the research of that gets more attention compared to the other. At present, most anomaly detection methods can be summarized as the following: specific instances are represented as data samples according to the features (attributes) of instances, and then use the classic pattern classification algorithms such as neural network, decision tree, cluster analysis and Bayesian theory, support vector machine (SVM), K nearest neighbor algorithm (KNN) to classify sample points [2-4]. Features are the raw materials of the classification system. Good features play a very key role for the accuracy of the anomaly detection algorithm.

In 2006, Geoffrey Hinton, professor of university of Toronto, and a leader in the field of machine learning, published a paper in science [5], which points out that the deep learning model has outstanding advantages on learning features. The features learn by this model can represent data with more rich information, and then improve the classification performance. Deep learning model [6] is to stack multiple nonlinear transformation functions, through combining low-level features to form a more abstract and more useful high-level features. It uses layered characteristics transform to map the samples from the original space to a new feature space which can facilitate the classification. So, in order to improve the accuracy of classification, we can use deep model to learn better and more complete features of data as the input of anomaly detection system.

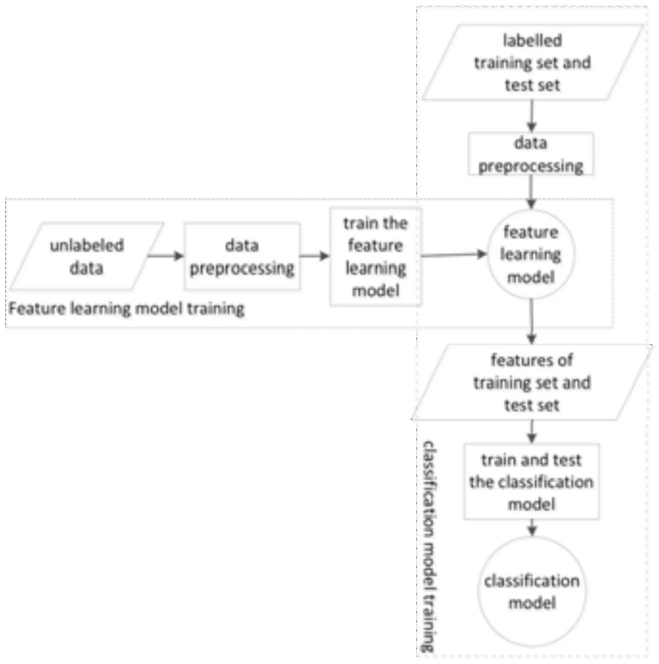
This paper proposes an anomaly detection method based on deep learning. By building deep neural network model with more hidden layers, the features of original data are extracted step by step from the bottom to the top. And then these features are used as the input to the current anomaly detection algorithm, in order to improve detection accuracy with good expression of original data. The whole learning process does not need any artificial feature extraction work, using the original data as input of the learning algorithm. And the learning process is unsupervised, so as to overcome the difficulty of acquiring labeled data. In this paper, two anomaly detection algorithms based on this method were given to prove the effectiveness of our method. Experimental results show that the features of data learning by our deep model can significantly improve the detection rate of original anomaly detection algorithm.

2. ANOMALY DETECTION PROCESS BASED ON DEEP LEARNING

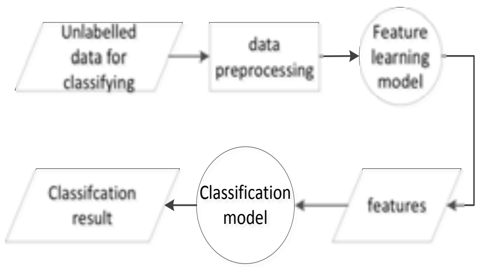
The anomaly detection method in this paper proposed a feature learning algorithm based on deep learning model, which learns the feature of the input data by unsupervised learning, and then the feature is input into the classification model to detect abnormal behavior. As shown in Fig. (1), this anomaly detection method includes two stages. One is the training phase for model, as shown in Fig. (1) (A). It includes the training for feature learning model and classification model. Data preprocessing in this phase is standardization and normalization of input data, and data type conversion, etc. The training process of feature learning model, with unlabeled data after preprocessing as input, was carried out by unsupervised method. Trained feature learning model can be directly applied to learn feature of new data, mapping the sample in the original space to a new feature space. The training process of classification model use the labeled training set to execute supervised training. The classification model here can be any anomaly detection algorithm that can be used for the classification, such as KNN, neural network and SVM, and so on [7].

Another phase is anomaly detection shown in Fig.(1) (B). The first step in this phase is data preprocessing. And then we learn the feature of data for classifying, where the trained feature learning model in first stage is used. Finally, the classification is carried out by trained classification model with the learned data feature as input.

*Address correspondence to this author at Experiment Center ,China West Normal University, Nanchong, Sichuan, 637002, P.R. China; Tel: +86 18090574679;
E-mail: yangtao_cwnu@163.com


labelled training set and test set
data preprocessing
unlabeled data
data preprocessing
train the feature learning model
feature learning model
Feature learning model training
features of training set and test set
train and test the classification model
classification model
classification model training

(A) Model training phase



(B) Anomaly detection phase

Fig. (1). Anomaly detection process based on deep learning.

3. RESEARCH METHOD

It is well known that, the training of deep neural network is very difficult before 2006. The objective function, which has many local optimal values [5], is very hard to optimize. The initial parameter is a pivotal role for whether the network can achieve the optimal solution. If the initial value is not good, neural network is easy to fall into local optimum. Until 2006, Geoffrey Hinton put forward that the difficulty of training neural network can be effectively overcome by "layer-wise training" [4]. Initialize network one layer after one layer, and finally form the initial value of the entire network. This initialization will help the gradient descent begin at a better initial search point, so as to converge to better local optima.

3.1 TRAINING PROCESS

The basic rule for deep neural network training is as follows: first pre-train network by an unsupervised training method (i.e., layer-wise training for initialization); then stack multiple layers that has been initialized to form a deep network; finally fine-tune this pre-trained deep network to get the feature learning model. This paper follows the same principle. First, using unlabeled data (labeled data can also be ok), all kinds of feature learning method can be applied to parameters pre-training of every network layer to obtain the initial parameters of the whole deep network. Pre-training process is as follows:

Algorithm 1:

```
Starting from the layer 2, where i = 2
(1) Use the feature value in layer i - 1 as the input of present layer i (the value of layer 1 is the original data) to train layer i, so as to learn the initial encoding parameters $W^{(i)}$, $b^{(i)}$ of layer i. And then these parameters are applied to get the features in the layer i ($h^{(i)}$).
(2) Input $h^{(i)}$ to the next layer, followed by the training of the next layer.
Repeat (1) and (2) to train every layer, until the last layer.
```

Layer 1 is the input of entire deep network, namely the original data. The last layer features is the output of entire feature learning model, and is the input of classifier. The $W^{(i)}$ and $b^{(i)}$ represent weights and bias values of layer i respectively.

Pre-training is critical to the success of deep neural network training. The network without pre-training is easy to fall into bad local extreme value with the increasing number of layers. Pre-training can help the network to find a good local extreme value point. Next, we will detail our pre-training method.

3.2 AUTO-ENCODER

The existing pre-training methods include auto-encoder (AE), restricted Boltzmann machine (RBM), sparse coding and deep belief network (DBN). The pre-training method in this paper proposed based on the auto-encoder. In order to state our method well, we will briefly describe the auto-encoder here. Auto-encoder [8-10] consists of two parts: encoder and decoder.

Encoder: nonlinear mapping function f maps input data ($x \in R^n$) to the representation in hidden layer ($h \in R^m$). The map is represented as follows:

$$h = f(x) = s_f(Wx + b) \qquad (1)$$

The parameters contain a weight matrix of encoder with size $m \times n$ (W), and a bias vector ($b \in R^m$).

Decoder: nonlinear mapping function g reconstructs input data from the representation in hidden layer ($h \in R^m$) to form $r \in R^n$.

$$r = g(h) = s_g(W'h + b') \qquad (2)$$

Where W' is the weight matrix of decoder with size $m \times n$, and $b' \in R^n$ is its bias vector.

The s_f, s_g are nonlinear activation functions, which usually adopt the form $\text{sigmoid}(z) = \frac{1}{1+e^{-z}}$.

The training of encoder is to search the parameters of network ($\theta = \{W, b, W', b'\}$). The training objective is to minimize the reconstruction error on the training set (D_N). The objective function is represented as:

$$J_{\text{AE}}(\theta) = \left(\sum_{x \in D_N} L\big(x, \mathrm{g}(f(x))\big)\right) + \frac{\lambda}{2}\sum_{ij} W_{ij}^2 \qquad (3)$$

The first item is reconstruction error, which usually is squared-error cost function:

$$L(x,r) = \|x-r\|^2 \qquad (4)$$

or the cross entropy cost function

$$L(x,r) = -\sum_{j=1}^{n} x_j \log(r_j) + (1-x_j)\log(1-r_j) \qquad (5)$$

The second item is weighted decay, where λ is weight decay parameter used to control the relative importance of the two items.

When the feature in first layer is obtained by auto-encoder, it becomes the input signal of the second layer. Through minimizing the loss function, we will get the parameter in second layer, and then get corresponding feature in this layer, which is the second expression of the original input information. Through repeating the above steps, we can form a multi-layer network, where each layer is a different feature expression of raw input data

3.3 THE PROPOSED PRE-TRAINING METHOD

In this paper, a real zero-value sparse auto-encoder neural network (RZSAE) with three layers is proposed to realize pre-training. Each RZSAE will learn one feature expression of input data. We train this neural network to minimize the error between input and output, so as to realize the minimum error after data transformation.

(1) Network structure

Suppose that we have an unlabeled training set $X = \left\{ x^{(1)}, x^{(2)}, \ldots x^{(N)} \right\}$ of N examples, $x^{(k)} \in R^n$ represents a sample.

Usually, auto-encoder neural network [8-10] is trained by unsupervised learning algorithm, which uses the back propagation algorithm to make the output (reconstruction of input) approach the input. The training of RZSAE in this paper still adopts this learning algorithm. The objective of training is setting as $r_\theta(x^{(i)}) \approx x^{(i)}$. Fig. (2) gives the structure of RZSAE which based on that of auto-encoder neural network.

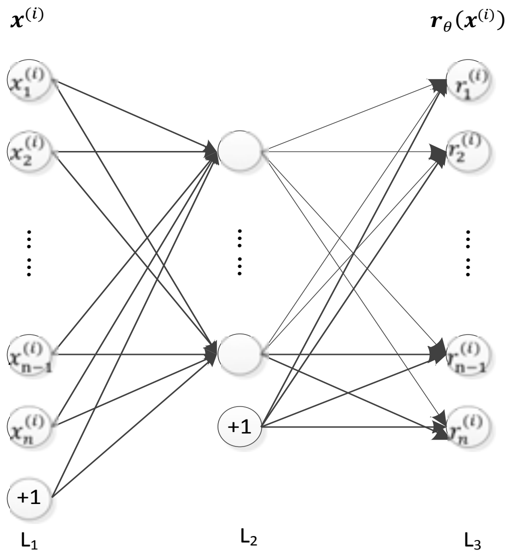


Fig. (2), Network structure.

Here, circle represents neurons. The neurons with "+1" is bias unit. There are three layers in the network. They are the input layer on the left, the output layer on the right and the hidden layer in the middle. Let n_l denotes the number of layers. For example, here n_l =3. We label layer l as L_l , so layer L_1 is the input layer, and layer L_{n_l} is the output layer. The network has parameters $\theta = (W^{(2)}, b^{(2)}, W^{(3)}, b^{(3)})$, where $W_{ij}^{(l)}$ denotes the weight associated with the connection between unit j in layer l and unit i in layer $l+1$. Also, $b_i^{(l)}$ is the bias associated with unit i in layer $l+1$.

And $r_\theta(x^{(i)})$ is the activation value of the output layer when $\mathrm{x}^{(\mathrm{i})}$ is given as input. In other words, it tries to learn an approximation to the identity function, so as to output $r_\theta(x^{(i)})$ that is similar to input $x^{(i)}$. The mapping from input of network to the value in hidden layer is encoding input data, namely learning feature of that. The value in hidden layer is then transformed to the output of network is decoding process, namely the reconstruction process. Our goal is to make the output after reconstruction approach the original input with the reconstruction error is minimal. The encoding result (activation value of hidden layer) is the feature expression of the input data.

(2) Cost function

In this paper, squared-error cost function is used. In detail, for a single training example $x^{(i)}$, we define the cost function with respect to that single example to be:

$$L\left(\theta; x^{(i)}\right) = \frac{1}{2}\left\| r_\theta(x^{(i)}) - x^{(i)} \right\|^2 \qquad (6)$$

This is a one-half squared-error cost function. Given a training set of m examples, we then define the overall cost function to be:

$$\begin{aligned} J(\theta;X) &= \frac{1}{N}\sum_{k=1}^{N}\mathrm{L}\left(\theta;x^{(k)}\right)+\frac{\lambda}{2}\sum_{l=2}^{n_l}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{l})^2 \\ &= \frac{1}{N}\sum_{k=1}^{N}\frac{1}{2}\left\|r_\theta\left(x^{(k)}\right)-x^{(k)}\right\|^2+\frac{\lambda}{2}\sum_{l=2}^{n_l}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{l})^2 \end{aligned} \tag{7}$$

The first item in the definition of $J(\theta;X)$ is an average sum-of-squares error. θ is the parameters of network, including the weights and biases in all the layers. The second item is weight decay to decrease the magnitude of the weights, so as to help prevent over fitting.

(3) Real zero-value sparsity

This article adds the real zero-value sparsity to hidden layer neurons of traditional sparse auto-encoder neural network to form RZSAE. It makes most of hidden layer neurons are real zero-value, which is different from the approximate zero-value in traditional sparse auto-encoder. That makes some of hidden neurons are activated, but the activation of other hidden neurons equals zero, so as to make the average activation of hidden neurons in a small range. In this paper, the activation function of neurons is the sigmoid function.

We use $h_j^{(2)}(x^{(i)})$ to denote the activation of hidden neuron j given the input sample $x^{(i)}$. Further, the average activation of hidden neuron j over all the samples in training set is defined to be:

$$\hat{\rho}_j = \frac{1}{N}\sum_{i=1}^{N}h_j^{(2)}(x^{(i)}) \tag{8}$$

Then, the sparsity penalty item which will be added into the overall cost function to be:

$$\sum_{j=1}^{s_2}\left[\rho\log\frac{\rho}{\hat{\rho}_j}+(1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}\right] \tag{9}$$

Where ρ is a sparsity parameter, usually a small value close to zero. The sparsity penalty item makes $\hat{\rho}_j$ close to ρ, so as to make the average activation of hidden neurons small enough.

So the overall cost function after adding sparsity penalty item is as follows:

$$J_{\text{sparse}}(\theta; X) = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{2}\left\| r_\theta\left(x^{(k)}\right) - x^{(k)}\right\|^2 + \frac{\lambda}{2}\sum_{l=2}^{n_l}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{l})^2 + \beta\sum_{j=1}^{s_2}\left[\rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}\right] \qquad (10)$$

The result of using this sparsity penalty item is that the activations of part of hidden neurons approximate zero-value, but still more than zero. In order to achieve real sparsity in hidden neurons, we set a threshold s to the result $h_j^{(2)}\left(x^{(i)}\right)$ after training with sparsity penalty item. It forces the hidden neurons whose activation approximates zero-value equals to real zero-value.

We define a threshold function to be:

$$thred(h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right)) = \begin{cases} 0, h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right) < s \\ 1, h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right) \geq s \end{cases} \qquad (11)$$

Then, the final activation of hidden neurons is:

$$h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right) = h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right) \times thred(h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right)) \qquad (12)$$

(4) Learning algorithm

The learning algorithm is trying to find the final activation of hidden neurons (one feature expression of input) through searching the minimal $J_{sparse}(\theta; X)$. In this paper, we adopt the batch gradient descent algorithm as follows:

Algorithm 2:

1) *Compute the overall cost function* $J_{sparse}(\theta; X)$

Step 1: forward propagation. Compute the activations of every layer

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^l \qquad (13)$$

$$a^{(l)} = f\left(z^{(l)}\right) \qquad (14)$$

$a^{(l)}$ *denotes the activation of layer* l. $z^{(l)}$ *is the net input of layer* l.

Step 2: compute the average activation of every hidden neuron using formula (8)

Step 3: apply formula (10) to compute the overall cost function $J_{sparse}(\theta; X)$

2) Compute the gradient of $J_{sparse}(\theta; X)$ *with respect to every parameters* $\nabla_{W^{(l)}} J_{sparse}(\theta; X)$ *and* $\nabla_{b^{(l)}} J_{sparse}(\theta; X)$
3) Update all the parameters

$$W^{(l)} = W^{(l)} - \alpha \nabla_{W^{(l)}} J_{sparse}(\theta) \qquad (15)$$

$$b^{(l)} = b^{(l)} - \alpha \nabla_{b^{(l)}} J_{sparse}(\theta) \qquad (16)$$

where α *is learning rate.*
4) Repeat 1) - 3) until our cost function $J_{sparse}(\theta; X)$ *is small enough.*

5) Compute the initial activations of hidden neurons through forward propagation with the trained parameters of network
6) Compute the final activation of hidden neurons according formula (12)

Using this algorithm, we can obtain one feature expression of input. Then we use the algorithm 1 and algorithm 2 to form the feature network, which is used to get the last feature expression of input.

4. EXPERIMENT

4.1 DATASET

In order to verify the validity of this method, this article use BCW dataset from the UCI database to do experiment. The dataset contains 699 samples, and every sample is 9 dimension data. There are two kinds of samples in the dataset, containing 458 normal data samples and 241 abnormal data. We set the first category as self-set (normal data), and the second category is nonself-set (abnormal data). This paper focuses on the detection rates of abnormal data (nonself-set). So we use the whole self-set and 100 nonself samples as the training sets, and the remaining 141 nonself samples as test set.

4.2 THE EXPERIMENTAL SETUP

In our experiment, the feature learning network in this paper contains two feature learning layers, one input layer and one output layer. The parameters of each feature learning layer were pre-trained by sparse auto-encoder neural network. The number of sparse auto-encoder neural network is decided according to feature learning layers' number. So in our experiment two sparse coding neural networks are trained to initialize the feature learning network. The first layer of the first sparse coding neural networks is input layer, whose neurons number s_1 is equal to the dimension of the input data, in this case $s_1 = 9$.The neurons number of second layer (hidden layer) $s_2 = 20$. Finally for the output layer, the number of neurons in output layer is equal to the number of input layer neurons $s_3 = s_1 = 9$. For the second sparse auto-encoder network, the input is the feature learned from the first sparse auto-encoder neural network (namely the value in hidden layer of the first network), so the number of input layer of the second network $s_1^{'} = s_2 = 20$. We set the hidden layer and output layer neurons are 20. Other parameters are set as λ = 0.003, $\beta = 3$, $\rho = 0.5$, and the threshold s=0.001. This set of parameters is chosen because they worked well in our experiments.

This features learned by learning model are applied to other regular anomaly detection algorithm. Here we test the result on KNN algorithm and multi-layer neural network (MLP) algorithm. Set the neighbor set size K = 10 for KNN algorithm. MLP network includes three layers, the input layer, one hidden layer and one output layer. The number of input layer neurons is equal to the dimension data that is 9. There are 20 hidden layer neurons, and the number of output layer neurons is decided by the classification number, which equals to 2 here.

4.3 THE EXPERIMENTAL RESULTS ANALYSIS

All experimental results are the average of 10 times repeated tests. We compared the result of KNN algorithm with that of DL_KNN (KNN based on DL) method. The results are shown in Fig.(3), which show that, under different K values, almost all the detection rates of DL_ KNN method are higher than that of KNN algorithm, only slightly lower when K = 3 (KNN is 83.69%, DL_KNN is 83.69%). Among them, the best detection rate of KNN method under various K values is 90.17 when K equals 16, while that of DL_KNN algorithm is up to 97.16% (K = 10). In addition, we found that when K approaches 200, the detection rate declined quickly. Because in training set the self-sample number is 100, when the training set size close to 200, nonself samples close to the self-samples in the collection of K neighbors. So the detection rate decline. When K = 200, KNN detection rate was 2.13%, almost to zero, while the DL + KNN detection rate was 13.48%. Because the KNN algorithm is based on K nearest neighbor to determine their own categories, so when K > 200, self-samples always more than nonself samples, which results that nonself samples always judge as self-samples, and accordingly the detection rate is zero.

Fig.(4) shows the results of MLP and DL_MLP method. Through the results, we can see that, in a variety of hidden layer numbers, the detection rates of DL_MLP method are all higher than that of MLP method.
All the experimental results show that, the detection rates of various detection algorithms based on deep learning have improved significantly.

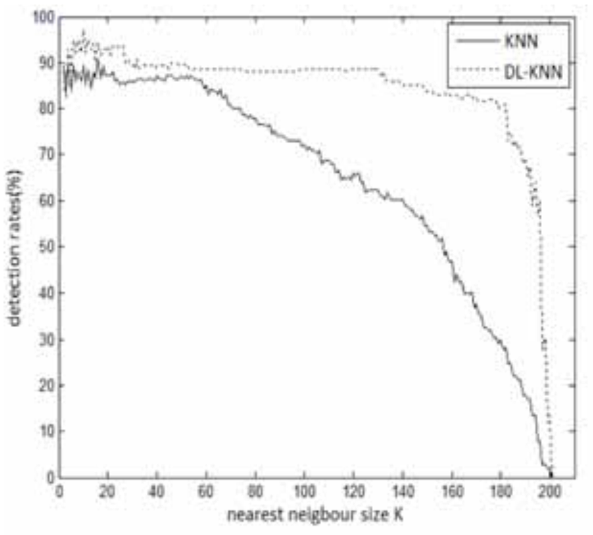


Fig. (3). Comparison between KNN and DL_KNN method.

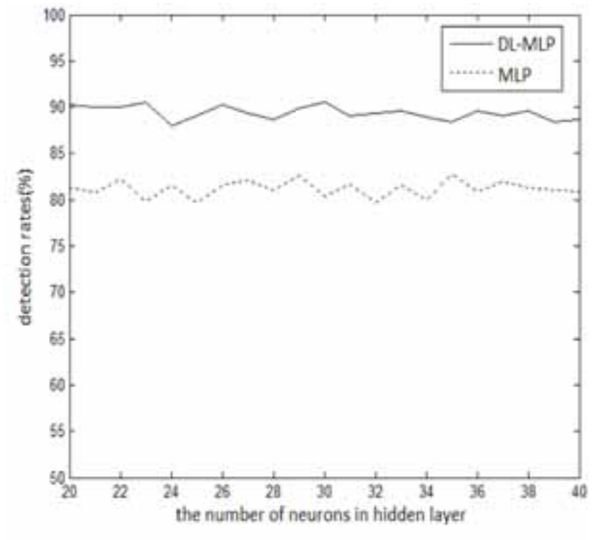


Fig. (4). Comparison between MLP and DL_MLP method.

Through the study we found that the deep learning method is first applied to extract the features of anomaly detection data. This method proposed in this paper overcomes some limitations of the classical anomaly detection method. The main advantage of this method is that: (1) it do not need artificial feature extraction. Artificial neural network with much more hidden layers has outstanding advantages in terms of feature learning. It can automatically learn with full using of big data to get useful features; (2) the deep model transforms data from one expression to another, which implements the mapping of the samples from the original data space to the new feature space. This transformation keeps sufficient information of the original data, at the same time improve the ability to distinguish different samples, and accordingly improve the accuracy of anomaly detection; (3) the feature learning model is completely unsupervised training process, which make full use of huge amounts of unlabeled data that are easy to access, overcoming the difficulty to obtain labelled data.

5. CONCLUSIONS

Aiming at the problems of feature learning and detection accuracy in anomaly detection algorithms, this paper proposed a new anomaly detection algorithm which uses a feature learning model to improve the accuracy of anomaly detection. This method applied the deep feature learning model based on sparse auto-encoder neural network to learn the useful features, which fully express the data information. And these features are combined with the traditional anomaly detection algorithm to obtain better detection accuracy. Experimental results show that the proposed method effectively improves the detection accuracy.

CONFLICT OF INTEREST

This article content has no conflict of interest

ACKNOWLEDGEMENTS

This study was supported by the project grant from the Science and Technology Department of SiChuan Province, China. No. 15ZB0147 and No. 15ZB0142.

REFERENCES

[1] Z. Zhang, R. Zhang, X. Jiang, and B. Zeng, “Anomaly detection method based on feature selection and support vector machine,” *Computer engineering and design*, vol. 34, no.9, pp. 3046-3049, 3162, 2013.

[2] X. Guo, F. Li, and W. Wang, “Local outlier detection algorithm of multivariate time series based on k-nearest neighbor,” *Journal of Jiangsu University of Science and Technology*, vol. 26, no.5, pp. 505-513, 2012.

[3] D. Hou, Y. Chen, H. Zhao, P. Huang, and G. Zhang, “Water quality anomaly detection method based on RBF neural network and wavelet analysis,” *Transducer and Microsystem Technologies,* vol. 32, no.2, pp. 138-141, 2013.

[4] L. Quan, and W. Wu, “Anomaly detection model based on support vector machine and Bayesian classification,” *Journal of Computer Applications*, vol. 32, no. 6, pp. 1632-1635,1639, 2012.

[5] G. Hinton, and R. Salakhutdinov, “Reducing the dimensionality of data with neural network,” *Science*, vol.313, no. 5786, pp. 504-507, 2006.

[6] K. Yu, L. Jia, Y. Chen, and W. Xu, “Deep Learning: Yesterday, Today, and Tomorrow,” Baidu, Beijing 100085, vol.50, no.9, pp.1799-1804, 2013.

[7] Y. Bengio, “Learning deep architectures for AI,” *Foundations and trends® in Machine Learning*, vol.2, no.1, pp. 1-127, 2009.

[8] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, “Contractive auto-encoders: Explicit invariance during feature extraction,” In:*Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 833-840, 2011.

[9] Y. Bengio, “Deep learning of representations: Looking forward,” *Statistical Language and Speech Processing*, Springer Berlin Heidelberg, pp. 1-37, 2013.

[10] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol.35, no.8, pp. 1798 – 1828, 2013.