# Stacked Sparse Autoencoder-Based Binary Tree Ensemble Intrusion Detection Method (Post-print)

**Authors:** Liu Yi, Yin Ziran, Hongzhou

**Date:** 2019-04-01T00:00:00+00:00

## Abstract

To date, numerous machine learning methods have been proposed; however, traditional machine learning methods cannot effectively solve the classification problem of large-scale intrusion data. To address this issue, a lightGBM (light gridient boosting model) binary tree algorithm based on stacked sparse autoencoders is proposed. First, the class labels are divided into five categories and constructed into a binary tree structure. Then, the imbalance problem in data distribution is resolved through upsampling methods. The aforementioned processing can decompose the large-scale data for subsequent separate training. Subsequently, a sparse autoencoder network is employed for feature dimensionality reduction. This dimensionality reduction method ensures that deeper features are extracted from the original data while saving dimensionality reduction time. Finally, classification is performed using the lightGBM ensemble algorithm. Moreover, compared to other models, the lightGBM model can save training time while ensuring classification performance. Experiments conducted on the NSL-KDD dataset measured the proposed method' s accuracy, precision, recall, and comprehensive evaluation metric F1, which on average reached 87.42%, 98.20%, and 91.31% respectively for the five-class classification, outperforming the comparison algorithms and significantly reducing computational time.

## Full Text

## Preamble

### Binary Tree Ensemble Intrusion Detection Method Based on Stacked Sparse Autoencoder

Liu Yi[1], Yin Ziran[1], Hong Zhou[2]
(1. School of Computer Science & Technology, Guangdong University of Tech-

nology, Guangzhou 510006, China;
2. Office of Academic Research, Guangzhou City Polytechnic, Guangzhou 510405, China)

**Abstract:** Numerous machine learning methods have been proposed to date, yet traditional approaches cannot effectively address the classification challenges posed by large-scale intrusion data. To tackle this problem, this paper proposes a LightGBM (Light Gradient Boosting Model) binary tree algorithm based on stacked sparse autoencoders. First, category labels are divided into five classes and constructed into a binary tree structure. Then, upsampling methods are employed to resolve data distribution imbalance. This decomposition enables large-scale data to be processed and trained separately. Subsequently, a sparse autoencoder network performs feature dimensionality reduction, which ensures that deeper features are extracted from the original data while saving reduction time. Finally, classification is performed using the LightGBM ensemble algorithm, which offers reduced training time compared to other models while maintaining classification performance. Experiments on the NSL-KDD dataset demonstrate that the proposed method achieves average accuracy, precision, recall, and comprehensive F1-score of 87.42%, 98.20%, and 91.31% across the five classes, respectively, outperforming comparison algorithms while significantly reducing computation time.

**Key words:** intrusion detection; stacked sparse autoencoder network; LightGBM algorithm; imbalanced data; NSL-KDD dataset

## 0 Introduction

Intrusion detection constitutes a critical component of information security, as proper detection is prerequisite for subsequent response and recovery. Intrusion detection is categorized into misuse detection and anomaly detection. Misuse detection identifies intrusions by modeling and leveraging distinctive attack signatures, achieving high detection rates for known attacks but failing to detect novel threats. Anomaly detection constructs models of normal behavior, flagging any deviation as anomalous [1]. However, since comprehensively modeling all normal behavior is difficult, anomaly detection often misclassifies legitimate activities as attacks.

With the continuous evolution of mobile internet and IoT, cyberattacks have become increasingly intelligent and sophisticated, complicating malicious intrusion detection. To meet these challenges, machine learning methods—including decision trees, naive Bayes, random forests, K-means clustering, and support vector machines—have been widely applied in intrusion detection. Traditional shallow-structure machine learning methods exhibit limited capacity for expressing complex functions and weak generalization, rendering them inadequate for complex classification tasks. In recent years, deep learning has emerged as a prominent topic in machine learning, demonstrating widespread success in facial recognition, speech recognition, and image classification, while also being

adopted for intrusion detection.

Reference [2] proposed an intrusion detection method using PCA for dimensionality reduction and KNN as classifier, showing that minority class detection rates remain notably low for multi-class problems, underscoring the importance of minority class handling in large-scale data scenarios. Reference [3] introduced a semi-supervised constrained Boltzmann machine (DRBM) model capable of detecting unknown intrusion events, achieving 96% accuracy in network anomaly detection, but lacking noise mitigation strategies and suffering from significant noise impact. Reference [4] combined DBN with SVM, using DBN for dimensionality reduction and SVM for classification, achieving promising results but neglecting class imbalance issues that critically impair detection performance for large-scale, diverse intrusion data. Reference [5] demonstrated that stacked denoising autoencoders effectively distinguish malicious from non-malicious software using a three-hidden-layer deep neural network with only six basic features readily available in SDN environments; however, this limited feature consideration risks information loss and proves unsuitable for large-scale, complex intrusion detection scenarios. Reference [6] employed a single-hidden-layer RBM for unsupervised feature dimensionality reduction, with weights transferred to another RBM to form a DBN. Pre-trained weights were then passed to a fine-tuning layer comprising a logistic regression classifier (trained with 10 iterations) and a softmax layer. Evaluated on KDD CUP99 data, this approach claimed a 97.90% detection rate with 2.47% false negative rate, improving upon comparable methods, though its stochastic gradient descent-based weight optimization scales poorly with data volume, necessitating further algorithmic updates. Reference [7] proposed a deep learning approach called self-taught learning (STL) combining sparse autoencoders with softmax regression, evaluated on the benchmark NSL-KDD dataset, showing improved classification accuracy for both binary and five-class scenarios, though weight optimization remained improvable. Reference [8] combined stacked sparse autoencoder networks (SSAE) with XGBoost ensemble algorithms, using SSAE for dimensionality reduction and XGBoost for classification, achieving an average F1-score of 91.97% for five-class classification; however, the five-layer sparse autoencoder' s dimensionality reduction efficiency and training time could be further improved for large-scale intrusion data.

Building upon these limitations, this paper proposes an enhanced method addressing large-scale intrusion data challenges, aiming to optimize computational time and mitigate class imbalance to improve classification performance. Using the NSL-KDD dataset [9] and a sparse autoencoder network with Adam optimizer for dimensionality reduction, extensive experiments determined appropriate pre-training iterations and hidden layer configurations. Finally, classification employs a LightGBM algorithm (LGB) [10] constructed binary tree structure. Contributions include: (1) using a three-layer stacked sparse autoencoder with Adam optimizer for dimensionality reduction, which improves minority class classification accuracy while reducing network training time; (2) employing LightGBM as classifier to further reduce training time and enhance

classification effectiveness; and (3) adopting a binary tree structure with 1:1 upsampling to address the low detection rate for minority classes prevalent in most algorithms.

## 1 Stacked Sparse Autoencoder Network

The stacked sparse autoencoder network learns deep features from intrusion detection data in an unsupervised manner. The sparsity constraint enhances the network' s generalization capability. Experimental results demonstrate that the proposed SSAE-LGB method can extract deep sparse features from high-dimensional intrusion data.

### 1.1 Single-Layer Sparse Autoencoder (SAE)

As shown in [Figure 1: see original paper], a single-layer SAE comprises an input layer, hidden layer, and output layer, providing a compressed representation of input vectors since the number of hidden nodes is smaller than the input vector length. The training process ensures the output vector approximates the input vector, enabling hidden nodes to capture effective feature representations of the dataset. Given an input vector $X$, the activation of all hidden nodes is computed as:

$$a(x) = f(W^{(1)}x + b_x)$$

where $W^{(1)}$ is the weight matrix connecting the input and hidden layers, $b_x$ is the bias vector of the input layer, and $f$ is the activation function containing $s$ hidden nodes. The output vector $\hat{x}$ is calculated as:

$$\hat{x} = f(W^{(1)T}a(x) + b_h^{(1)})$$

where $W^{(1)T}$ is the weight matrix connecting the hidden and output layers, and $b_h^{(1)}$ is the bias vector of the hidden layer.

The cost function $J_{SAE}$ includes error between all input and output data, a weight decay term, and a sparsity penalty term. Specifically, the cost function is defined as:

$$J_{SAE} = J_{AE} + \beta \sum_{j=1}^{s} KL(\rho||\hat{\rho}_j)$$

where $J_{AE}$ is the cost function without sparsity consideration, and the second term is the sparsity penalty. More specifically, $\beta$ controls the weight of the sparsity penalty term, $KL(\rho||\hat{\rho}_j)$ is the Kullback-Leibler divergence between $\rho$ (the desired sparsity parameter) and $\hat{\rho}_j$ (the average activation of hidden node

$H_j$ across all input data), where $n_k$ is the number of nodes in hidden layer $k$. The KL divergence is computed as:

$$KL(\rho||\hat{\rho}_j) = \rho \log\left(\frac{\rho}{\hat{\rho}_j}\right) + (1-\rho)\log\left(\frac{1-\rho}{1-\hat{\rho}_j}\right)$$

Notably, most hidden node activations are constrained to values near zero. The first term $J_{AE}$ is calculated as:

$$J_{AE} = \frac{1}{2n}\sum_{i=1}^{n}||x_i - \hat{x}_i||^2 + \frac{\lambda}{2}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ij}^{(l)})^2$$

where the first term measures total error between input and output data with $n$ being input data dimensionality; the second term is a weight decay term controlling weight magnitude to prevent overfitting, where $\lambda$ is the regularization parameter, $L$ is the number of layers, $l$ is the current layer index, $s_l$ and $s_{l+1}$ are node counts in adjacent hidden layers, and $W_{ij}^{(l)}$ is the weight matrix connecting adjacent layers.

Optimization proceeds via the Adam algorithm [11] to optimize weights and biases in the cost function, completing SAE training. Adam combines Momentum and RMSprop. After sufficient iterations, activations are reduced to minimal values, enabling automatic feature extraction.

Using Adam to update weights $W$ and layer biases $b$, the specific steps are:

First, compute momentum exponential weighted averages:

$$V_{\Delta w} = \beta_1 V_{\Delta w} + (1-\beta_1)\frac{\partial w}{\partial t}$$

$$V_{\Delta b} = \beta_1 V_{\Delta b} + (1-\beta_1)\frac{\partial b}{\partial t}$$

Then update using RMSprop:

$$S_{\Delta w} = \beta_2 S_{\Delta w} + (1-\beta_2)\left(\frac{\partial w}{\partial t}\right)^2$$

$$S_{\Delta b} = \beta_2 S_{\Delta b} + (1-\beta_2)\left(\frac{\partial b}{\partial t}\right)^2$$

Apply bias correction to equations (5)-(8):

$$\hat{V}_{\Delta w} = \frac{V_{\Delta w}}{1-\beta_1^t}, \quad \hat{V}_{\Delta b} = \frac{V_{\Delta b}}{1-\beta_1^t}$$

$$\hat{S}_{\Delta w} = \frac{S_{\Delta w}}{1-\beta_2^t}, \quad \hat{S}_{\Delta b} = \frac{S_{\Delta b}}{1-\beta_2^t}$$

The final parameter update functions are:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{S}_{\Delta w}} + \epsilon} \hat{V}_{\Delta w}$$

$$b_{t+1} = b_t - \frac{\alpha}{\sqrt{\hat{S}_{\Delta b}} + \epsilon} \hat{V}_{\Delta b}$$

where $\alpha$ is the learning rate, $V$ denotes moving averages, $S$ denotes squared gradients, $\beta_1$ and $\beta_2$ are exponential decay rates, $\epsilon$ is a set step size, $t$ denotes a time step, and subscripts indicate corresponding values at time $t$.

### 1.2 Stacked Sparse Autoencoder (SSAE)

As the name suggests, SSAE is a hierarchical encoding structure where single-layer SAEs are stacked. Each hidden layer aims to learn more abstract feature representations from the previous layer [12]. The training process for each SSAE layer mirrors SAE training: minimizing the cost function to obtain optimal weights and biases per layer. After proper training of all layers, SSAE can learn more complex and abstract feature representations from input training data.

[Figure 2: see original paper] illustrates the SSAE training architecture. SSAE construction involves individually training each SAE, where each layer' s input is the previous layer's hidden representation, finally connecting all hidden layers sequentially to form the complete SSAE. The first layer comprises $x$, $h_2$, and $X$, using equation (4) for unsupervised feature representation learning. Equations (13)-(14) then yield weights and biases $W_1$, $b_1$. The second layer consists of $h_1$, $h_2$, and $H_2$, undergoing identical training to obtain $W_2$, $b_2$. Repeating these steps yields the entire network' s parameters.

## 2 SSAE-lightGBM Binary Tree Structure Algorithm

### 2.1 Data Processing Flow

As shown in [Figure 3: see original paper], the process begins by importing the NSL-KDD dataset. Since data serves as neural network input, preprocessing is required. Continuous features require normalization to balance per-dimension influence, while categorical features need one-hot encoding. The standardized data then feeds into the SSAE-lightGBM binary tree. Finally, the model predicts experimental data and comparative analysis is performed.

### 2.2 SSAE-lightGBM Binary Tree Structure Algorithm (SSAE-lgb-BT) Framework

Class imbalance frequently occurs in intrusion detection, impairing prediction performance for low-volume categories. Common solutions include upsampling, downsampling, and cost-sensitive learning. By referencing decision tree classification processes and analyzing intrusion dataset distributions, we introduce a binary tree structure to address intrusion detection. Binary trees decompose multi-classification into binary classification, relatively balancing the original class imbalance and reducing subsequent ensemble computation.

Binary classification may still suffer from imbalance. At the data level, oversampling and undersampling are most representative. Oversampling creates synthetic samples for minority classes, adding them to the training set but requiring substantial training time. Undersampling reduces majority class samples to balance categories but risks losing important information. Based on this analysis, we propose a hybrid EUS-SMOTE method combining EUS [13] and SMOTE [14]. EUS-SMOTE first uses EUS to separate majority classes from minority classes with multiple categories, then applies SMOTE to minority classes, finally combining majority class samples with oversampled minority samples as the training set. This process repeats for minority classes with multiple categories. The oversampling ratio for imbalanced classes uses a 1:1 proportion. Finally, the SSAE-lightGBM ensemble algorithm trains and classifies on each separated dataset layer. The algorithmic framework is shown in [Figure 4: see original paper].

[Figure 4: see original paper] depicts the prediction phase. Intrusion data first enters the first-layer classifier A. After processing through the SSAE-lightGBM ensemble model, data is classified as class 0 or 1. Class 0 outputs the result and converts it to the actual label; class 1 passes the original intrusion data to the second-layer classifier B. This process repeats until completion.

[Figure 6: see original paper] details the SSAE-lightGBM model' s intrusion data prediction process. The first half is an encoder composed of three trained hidden layers (training process shown in [Figure 2: see original paper]). As data passes through layers 1-3, original feature dimensions are determined by each subsequent layer' s neuron count until the final layer, where the encoder automatically extracts meaningful features from high-dimensional data. These features then serve as input to the LightGBM classifier for prediction. The encoder in [Figure 6: see original paper] comprises four encoders generated during the training process shown in [Figure 5: see original paper], each produced via unsupervised learning from different datasets.

The SSAE-lightGBM ensemble algorithm' s training process is illustrated in [Figure 5: see original paper]. The final classifiers trained in (a)-(d) correspond to classifiers A, B, C, and D in [Figure 4: see original paper]. In [FIGURE:5(a)], the NSL-KDD training set is first divided into two major categories: Normal and (DoS, Probe, U2R, R2L), labeled 0 and 1 respectively. EUS-SMOTE addresses

the class balance between categories 0 and 1. SSAE then extracts deep features $T$, which are fed into the LightGBM model for training using bagging-style 5-fold cross-validation to fuse predictions, yielding classifier A. Identical steps produce classifiers B, C, and D.

## 3 Experiments and Results

To validate the proposed algorithm' s effectiveness, we compare the SSAE-lightGBM binary tree structure algorithm against SSAE-XGBoost binary tree structure and PCA-XGBoost binary tree structure algorithms on the public NSL-KDD dataset. All methods run on an Intel(R) CoreTM i5-3210M CPU @ 2.5Hz processor with 4 GB RAM, Windows 7 64-bit OS, and Pycharm 2017.

### 3.1 Data Description

Experiments utilize the complete NSL-KDD dataset, derived from the KDD-cup99 dataset by eliminating redundancy for enhanced practicality. NSL-KDD contains 125,973 training samples and 22,544 test samples, including four attack types: Denial of Service (DoS), Remote-to-Local (R2L), User-to-Root (U2R), and Probe attacks that attempt to gather target host information. [Figure 7: see original paper] shows the training and test data distributions, revealing significant inter-class imbalance—DoS instances vastly outnumber U2R instances. The dataset comprises 41 features and one class label: 38 continuous features and 3 categorical features.

### 3.2 Evaluation Metrics

We employ precision, recall, and F1-score for comparative evaluation:

$$Precision = \frac{TP}{TP + FP} \times 100\%$$

$$Recall = \frac{TP}{TP + FN} \times 100\%$$

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN} \times 100\%$$

where TP (True Positive) denotes correctly classified normal instances, TN (True Negative) correctly classified attack instances, FN (False Negative) normal instances misclassified as attacks, and FP (False Positive) attacks misclassified as normal. Precision indicates the correctness of returned normal categories, while recall measures misclassified attacks. The F1-score is the harmonic mean of precision and recall.

presents performance comparisons. The proposed SSAE-lightGBM binary tree ensemble (SSAE-lgb-BT), SSAE-XGBoost binary tree ensemble (SSAE-xgb-BT), and PCA-LightGBM binary tree (PCA-lgb-BT) algorithms are denoted SLB, SXB, and PLB respectively. F1-scores reflect detection effectiveness. SSAE-xgb-BT outperforms PCA-xgb-BT (which lacks deep feature extraction) while requiring only one-quarter of the runtime, demonstrating SSAE' s superior feature extraction over PCA. Comparing SSAE-lgb-BT and SSAE-xgb-BT, the former achieves 97.09% F1-score on Probe class—9 percentage points higher—while showing improved recall for minority class U2R despite lower precision on minority class R2L. Critically, SSAE-lgb-BT requires only one-fifth of SSAE-xgb-BT' s computation time, establishing its overall superiority.

compares SSAE-lgb-BT with DNN(1) [7] and DNN(2) [15], using experimental results reported in the original papers. SSAE-lgb-BT outperforms both DNN algorithms, confirming that stacked sparse autoencoders learn deeper features from high-dimensional data. The F1-score comparison shows SSAE-lgb-BT better handles data imbalance, yielding superior classification performance.

### 3.3 Selection of Sparse Autoencoder Iterations and Hidden Layers

For SSAE-lightGBM components, pre-training iteration count and hidden layer number are crucial. Insufficient iterations fail to reduce loss adequately, while excessive iterations waste computational resources. Too many hidden layers cause overfitting; too few yield poor detection performance. Experiments were conducted with batch size 64, epochs 80, and a three-layer structure of 100-80-60 neurons.

[Figure 8: see original paper] shows the relationship between loss and pre-training iterations across different hidden layers. As iterations increase, loss decreases variably across layers. After 40 iterations, losses for layers 1, 2, and 3 stabilize. Based on this analysis, we select a network structure with 3 hidden layers and 40 pre-training iterations.

### 3.4 Experimental Performance Evaluation

presents the performance of the stacked sparse autoencoder LightGBM binary tree structure algorithm across different categories, showing precision, recall, F1-score, and computation time.

## 4 Conclusion

This work utilizes stacked sparse autoencoder networks to learn deep features from intrusion detection data in an unsupervised manner. The sparsity constraint enhances generalization capability. Experimental results demonstrate that the proposed SSAE-LGB method extracts deep sparse features from high-dimensional intrusion data. Compared with linear PCA dimensionality reduction, SSAE-lgb-BT significantly improves detection performance. Relative to

SSAE-xgb-BT using five-layer stacked sparse autoencoders with stochastic gradient boosting optimization, SSAE-lgb-BT further enhances accuracy while saving substantial computation time. Although precision on minority class R2L remains modest, recall for minority class U2R improves markedly over SSAE-xgb-BT, yielding superior overall performance with an average F1-score of 91.31%. The method effectively handles class imbalance, improving F1-scores for minority categories, thus providing a novel research approach for network intrusion detection. Future work will further enhance algorithmic performance by clustering training data (with 5 cluster centers) and classifying test sets against these centers (potentially using KNN) to further improve SSAE-lgb binary tree ensemble accuracy and computation time.

## References

[1] Sarasamma S T, Zhu Qiuming A, Huff J. Hierarchical Kohonen net for anomaly detection in network security [J]. IEEE Trans on Systems Man & Cybernetics Part B Cybernetics, 2005, 35(2): 302.

[2] Khalid C, Zyad E, Mohammed B. Network intrusion detection system using L1-norm PCA [C]//Proc of the 11th International Conference on Information Assurance & Security. 2016.

[3] Fiore U, Palmieri F, Castiglione A, et al. Network anomaly detection with the restricted Boltzmann machine [J]. Neurocomputing, 2013, 122: 313.

[4] Salama M A, Eid H F, Ramadan R A, et al. Hybrid Intelligent Intrusion Detection Scheme [M]//Soft Computing in Industrial Applications, Volume 96 of the Advances in Intelligent and Soft Computing Book Series. Berlin: Springer-Verlag, 2011: 293-303.

[5] Wang Yao, Cai Wandong, Wei Pengcheng. A deep learning approach for detecting malicious JavaScript code [J]. Security & Communication Networks, 2016, 9(11): 1520-1534.

[6] Alrawashdeh K, Purdy C. Toward an online anomaly intrusion detection system based on deep learning [C]//Proc of the 15th IEEE International Conference on Machine Learning & Applications. Piscataway, NJ: IEEE Press, 2017.

[7] Javaid A Y, Niyaz Q, Sun Weqing, et al. A deep learning approach for network intrusion detection system [C]//Proc of the 9th EAI International Conference on Bio-inspired Information Communications Technologies. New York: ACM Press, 2015: 21-26.

[8] Zhang Baoan, Yu Yanhua, Li Jie. Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method [C]//Proc of IEEE International Conference on Communications Workshops. Piscataway, NJ: IEEE Press, 2018: 20-2.

[9] Tavallaee M, Bagheri E, Lu Wei, et al. A detailed analysis of the KDD CUP 99 data set [C]//Proc of the 2nd IEEE International Conference on Computational Intelligence for Security & Defense Applications. Piscataway, NJ: IEEE Press, 2009: 53-58.

[10] Ke Guolin, Meng Qi, Thomas F. LightGBM: a highly efficient gradient boosting decision tree [C]//Proc of the 31st Conference on Neural Information Processing Systems. Long Beach, CA: NIPS Press, 2017.

[11] Kingma D P, Ba J. Adam: a method for stochastic optimization [C]//Proc of the 3rd International Conference for Learning Representations.

[12] Lee H, Grosse R, Ranganath R, et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations [C]//Proc of the 26th Annual International Conference on Machine Learning. New York: ACM Press, 2009: 609–616.

[13] Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: synthetic minority over-sampling technique [J]. Journal of Artificial Intelligence Research, 2002, 16(1): 321-357.

[14] Garcia S, Herrera F. Evolutionary undersampling for classification with imbalanced datasets: proposals and taxonomy [J]. Evolutionary Computation, 2014, 17(3): 275-306.

[15] Tang T A, Mhamdi L, McLernon D, et al. Deep learning approach for network intrusion detection in software defined networking [C]//Proc of International Conference on Wireless Networks & Mobile Communications. Piscataway, NJ: IEEE Press, 2016.

*Note: Figure translations are in progress. See original paper for figures.*