Proceedings of the 6th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/69/20241511

IoT traffic classification and anomaly detection method based on deep autoencoders

Qi Xin[1a,*], Zeqiu Xu[1b], Lingfeng Guo[2], Fanyi Zhao[3], Binbin Wu[4]

[1a]Management Information Systems, University of Pittsburgh, Pittsburgh, PA, USA
[1b]Computer Science, Carnegie Mellon University, CA, USA
[2]Business Analytics, Trine University, AZ, USA
[3]Computer Science, Stevens Institute of Technology, NJ, USA
[4]Heating Ventilation and Air Conditioning Engineering,Tsinghua University, Beijing China

*Corresponding author E-mail: QIX29@pitt.edu

Abstract. This study investigates anomaly detection of IoT device traffic using Convolutional Neural Networks (CNN) and Variational Autoencoders (VAE) to enhance the detection capability of security threats in IoT environments. A series of hardware configurations, software environments, and hyperparameters were utilized to optimize the training and testing processes of the models. The CNN model demonstrates robust classification performance, achieving an accuracy rate of 95.85% on the test dataset, effectively distinguishing between different types of IoT device traffic. Meanwhile, the VAE model exhibits proficient anomaly detection capabilities by effectively capturing abnormal patterns in the data using reconstruction loss and KL divergence. The combined use of CNN and VAE models offers a comprehensive solution to cybersecurity challenges in IoT environments. Future research directions include exploring diverse IoT traffic data, practical deployment for validation, and further optimization of model structures and parameters to improve performance and applicability.

Keywords: IoT, Convolutional Neural Network, CNN, Variational Autoencoder, VAE, Anomaly Detection.

1. Introduction

Network traffic anomaly detection is an important means to detect network attacks. With the increase in traffic feature dimensions and noise data, the accuracy and robustness of traditional machine learning methods in traffic anomaly detection are challenged, which affects the performance of traffic attack detection. To address these problems, anomaly detection methods based on deep learning have gradually become a research hotspot.

There are three main anomaly detection methods based on deep learning:

1. Anomaly detection method based on deep Boltzmann machine: [1] This method extracts the essential characteristics of high-dimensional traffic data through learning, thereby improving the detection rate of traffic attacks. However, this method is less robust when extracting features, and the detection performance will be significantly degraded when the input data contains noise.

DOI: 10.54254/2755-2721/69/20241511

2.Anomaly detection method based on Stacked Autoencoders (SAE): This method learns traffic data layer by layer to extract high-accuracy traffic features. However, the robustness of feature extraction is also poor, and the detection accuracy will be reduced when the measured data is damaged.

3. Anomaly detection method based on convolutional neural network [2] (CNN): The traffic features extracted by this method have strong robustness and high attack detection performance. However, this method needs to convert network traffic into images first, which increases the burden of data processing and does not fully consider the impact of network structure information on the accuracy of feature extraction.

This paper proposes a deep autoencoder-based traffic classification and anomaly detection method for the Internet of Things (IoT).[3] In this method, the particle swarm optimization (PSO) algorithm is used to optimize the structure of the deep autoencoder (SDA) in two stages. The traffic features are extracted using deep learning, and an anomaly detection classifier is constructed to effectively detect traffic attacks.

2. Related work

2.1. Attack traffic detection technology based on machine learning

Network traffic attack detection technology based on machine learning is one of the hot research spots at present. Its main idea is to analyze the network situation by combining the characteristics of network traffic, and determine whether there is an attack behavior traffic that is different from the normal traffic through machine learning algorithm. Among them, machine learning algorithms mainly include naive Bayes, decision trees, support vector machines, etc., which carry out anomaly detection through classification and clustering of network traffic data. In this kind of recognition algorithms, the most studied methods are statistics-based and behavior-based [4].

Casas uses a simple detection model based on decision tower under large-scale network traffic to study the performance of machine learning methods on abnormal traffic detection, and finds that the model still has good detection performance in the face of a large number of traffic data.

However, some researchers have found that the model can maintain a high recognition performance mainly depends on a large number of feature engineering, whose quality will directly affect the classification performance of the network, which often requires a lot of work [5].

2.2. Attack traffic detection technology based on deep learning

As an important branch of machine learning algorithms, deep learning-based network traffic attack detection technology has also attracted wide attention. Compared with traditional machine learning methods, deep learning-based methods can automatically extract advanced features in network traffic to detect network attack behaviors more accurately, addressing the limitations of traditional machine learning methods in feature engineering.Common deep learning-based network attack detection algorithms include Deep Neural Networks (DNN)[6], Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory networks [7](LSTM), and Autoencoders. These algorithms can extract deep and high-level features from raw network traffic data, have strong adaptability and generalization ability, and can effectively detect various types of network attacks.

This method constructs a convolutional neural network to train on traffic data and detect abnormal traffic. The key idea is to use the CNN to extract and learn features from traffic data, enabling more accurate classification and detection of traffic. Lotfollahi extracts the spatiotemporal features of data packets through multi-layer convolution and pooling operations, and then performs packet detection and identification through the fully connected layer and softmax layer.In summary, existing attack traffic detection technology has been widely used in complex network environments and has successfully addressed many security problems. However, we cannot ignore the limitations related to network traffic and the models themselves: machine learning models have high data requirements, so data imbalance issues can affect the accuracy of these models in detecting attack traffic. Therefore, it is necessary to consider these problems and propose new attack traffic detection methods.

DOI: 10.54254/2755-2721/69/20241511

2.3. Variational Autoencoder (VAE)

Variations, or variations. What should we know about functionals before we talk about variations? To review the functions we have learned since the beginning, it is to take a given input value x, through a series of changes f(x), to get the output value y. Notice here we're putting a number in, and we're putting a number out. Is there a case where our argument is a function instead of a number? The classic question is, given two fixed points A and B, we can take any path from point A to point B, and find out in what path the time from point A to point B is shortest? By this point most people have the answer - the shortest line between two points. A function where the input variable is a function and the output variable is a numerical value is called a functional. The popular understanding of a functional is a function of a function.

Usually, we feed the input image into the NN Encoder and get a latent code, usually the dimension of this latent code is much smaller than the dimension of the input object, which is a compact representation of the input object. Next, we feed this latent code into [8] NN Decoder for decoding and output the reconstructed original object.

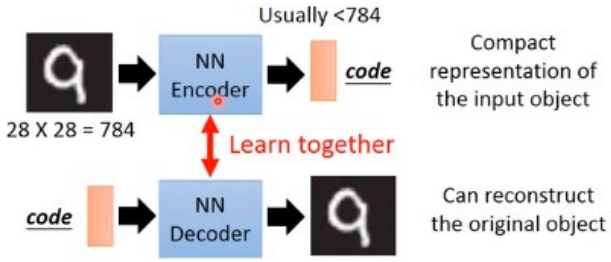

Figure 1. Auto-Encoder architecture diagram

Auto-Encoder was proposed by Rumelhart in 1986 and can be used for processing high-dimensional complex data, which promotes the development of neural networks. A self-coding neural network is an unsupervised learning algorithm (training examples are not labeled) that uses the BP backpropagation algorithm and strives to make the output as close to the input as possible.

AE networks generally have two characteristics:

1.dim(Hidden layer) << dim(Input layer) : the hidden layer dimension should be much smaller than the input dimension.

2.The Output of the decoding layer is used for Reconstruction Input, so we should minimizer(Reconstruction error(Input, Output)), that is, minimize the reconstruction error between input and output.

AE algorithm description:

1.Encoder is responsible for compressing the input data, and compressing the N-dimensional input data into M-dimensional data (m << n) through the Hidden layer. In other words, encoder learns a set of parameters to obtain a latent code;

2.Decoder is responsible for restoring data and restoring original data as much as possible with the least loss when needed.

3. Methodology

3.1. Data Set

The network traffic dataset utilized in this study encompasses diverse types of IoT devices and network attack traffic. Prior to analysis, the dataset underwent preprocessing steps including data cleaning and normalization, ensuring data quality and enhancing the efficacy of model training. The dataset, as described previously, comprises images categorized into 10 classes, representing various objects such as pedestrians, cars, vans, buses, and others, contributing to a comprehensive understanding of real-world scenarios in the domain of object detection and classification.

DOI: 10.54254/2755-2721/69/20241511

3.2. Variational Autoencoder (AVE) Model

VAE is to add appropriate noise to the code on the original AE structure. First we input the input into the NN Encoder and calculate two sets of encodings: one is encoded as the mean encoding $m = (m1, m2, m3)$, and the other is encoded as the variance encoding $\sigma = (\sigma1, \sigma2, \sigma3)$.that controls the noise interference degree. [9] The variance code σ is mainly used to assign weight to noise code. In the figure, a layer of exponential operation is applied to variance code $z = (e1, w2, e3)$.before the weight is assigned to z, as long as the reason is that the weight value learned by NN is positive and negative, so this is to ensure that the weight assigned is positive. Finally, we overlay the original code m with the weighted noise code to obtain a new latent code, which is then fed into the NN Decoder. Observing the figure above, it can be seen that in addition to the reconstruction error of traditional AE, the loss function has the following additional item:

In this experiment, the functions and advantages of VAE model can be further elaborated as follows:

1. Data potential representation learning: The VAE model maps the input data to the distribution in the potential space through the encoder part, and reconstructs the representation in the potential space into the original data through the decoder part. This learning process enables the model to capture the underlying structure and features in the data, helping to improve the understanding and analysis of the data.

2. Data reconstruction and generation: Through the decoder part, VAE model can generate new samples similar to the input data, so as to achieve data reconstruction and generation. This ability can be used to enhance, expand or generate new data samples in the experiment, which helps to improve the generalization ability and performance of the model.

3. Continuity and interpolation of potential space: Because VAE model learns and transforms potential space, there is continuity and interpolability between different representations in potential space. This means that in the potential space, similar data samples correspond to similar potential representations, so that data samples can be interpolated and generated in the potential space, which is conducive to the exploratory analysis and visualization of the model.

4. Optimization of loss function [10]: In this experiment, the VAE model is optimized using mean square error (MSE) and KL divergence as loss functions. The selection of the loss function helps the model to learn the reconstruction error of the data and the structure information of the potential space, thus improving the training stability and performance of the model.

Through these methods, VAE can effectively capture potential structures in high-dimensional data and take advantage of reconstruction errors for anomaly detection. This method can detect not only significant anomalies, but also those small but potentially important anomalies, providing a powerful support for further data analysis and processing.

4. Experiment

4.1. Experimental design

The purpose of this experiment is to investigate the anomaly detection of IoT devices using convolutional neural networks (CNNS) and variational autoencoders (VAE). The experiments were conducted in the following hardware configuration and software environment, and a series of hyperparameters were set to optimize the training and testing process of the model.

(1) Hardware Configuration

Component	Details
CPU	Intel Core i7
GPU	NVIDIA GTX 1080 Ti
RAM	16 GB
Storage	500 GB SSD

DOI: 10.54254/2755-2721/69/20241511

(2) Software Environment

Component	Details
Operating System	Ubuntu 20.04
Programming Language	Python 3.8
Deep Learning Framework	PyTorch 1.10.0
Other Libraries	pandas, numpy, matplotlib

(3) Dataset

Dataset Type	Details
Training Set	Stored in the train folder, includes traffic data from multiple IoT devices
Test Set	Stored in the test folder, used to evaluate model performance

4.2. Model Architecture and Hyperparameters

(1) Convolutional Neural Network (CNN)

Component	Details
Convolutional Layers	Conv1: Input 1, Output 32, Kernel size 3, Padding 1 Conv2: Input 32, Output 64, Kernel size 3, Padding 1 Conv3: Input 64, Output 128, Kernel size 3, Padding 1 Conv4: Input 128, Output 128, Kernel size 3, Padding 1
Pooling Layers	Max Pooling, Kernel size 2, Stride 2
Fully Connected Layers	FC1: Input 128 * 7, Output 128 FC2: Input 128, Output 9
Activation Function	ReLU
Optimizer	Adam, Learning rate 0.01
Loss Function	CrossEntropyLoss
Batch Size	32
Training Epochs	10

(2) Variational Autoencoder (VAE)

Component	Details
Encoder	Linear Layers: Input dimension input_dim, Output 512 Output 256 Output 128
Activation Function	ReLU
Latent Variables	Mean Layer: Input 128, Output latent_dim Log-Variance Layer: Input 128, Output latent_dim
Decoder	Linear Layers: Input latent_dim, Output 128 Output 256 Output 512 Output input_dim, Activation Function Sigmoid
Optimizer	Adam, Learning rate 0.01
Loss Function	Reconstruction Loss (MSE Loss) and KL Divergence
Batch Size	128
Training Epochs	50
Latent Dimension	20

4.3. Experimental Procedure

The training process for both the Convolutional Neural Network (CNN) and the Variational Autoencoder (VAE) involves several key steps. Firstly, the data is loaded and preprocessed, with features extracted and normalized as necessary. Subsequently, forward propagation occurs through the respective models, where input data passes through layers of neurons, applying activation functions and

DOI: 10.54254/2755-2721/69/20241511

transformations. During backward propagation, the loss is computed, typically using functions like CrossEntropyLoss for CNN and a combination of reconstruction loss and KL divergence for VAE. Gradients of the loss with respect to model parameters are then calculated, allowing the optimizer (such as Adam) to update the model parameters iteratively. [11] This process repeats for each batch of data for a specified number of epochs, gradually optimizing the model parameters towards better performance.

Following the training phase, the CNN model's performance is evaluated on the test dataset. The model's accuracy, visualized in "Accuracy.png," illustrates its ability to correctly classify IoT device data. Specifically, the CNN model achieved an impressive accuracy of 95.85% on the test set. This high accuracy underscores the effectiveness of the CNN architecture in accurately classifying different types of IoT device data, showcasing its potential utility in real-world applications for anomaly detection and classification tasks.

4.4. Experimental result

To succinctly represent the detection results for different types of traffic, we will select three images that effectively capture the detection outcomes across diverse IoT device categories:

The detection results for various types of IoT device traffic are represented through three key images. Firstly, "gafgyt_danmini_doorbell(a)" illustrates the detection outcomes for traffic associated with the Gafgyt malware targeting Danmini Doorbell devices, showcasing anomalies in this commonly targeted IoT device category. Secondly, "gafgyt_ecobee_thermostat(b)" displays the detection results for traffic related to the Gafgyt malware affecting Ecobee Thermostat devices, shedding light on anomalies in another significant IoT device category often exploited by malware. Lastly, "mirai_provision_security_camera(c)" demonstrates the detection outcomes for traffic associated with the Mirai malware targeting Provision Security Camera devices, providing insights into the detection results for a distinct type of IoT device affected by a different malware variant.

4.5. Experimental discussion

The CNN model demonstrates strong classification performance, achieving a high accuracy rate of 95.85% on the test dataset. This indicates that the CNN model effectively classifies different types of IoT device traffic, showcasing its capability in accurately identifying and categorizing various patterns within the data. The classification outcomes of the CNN model provide valuable insights into the effectiveness of the model in distinguishing between different classes of IoT device traffic, thereby aiding in the identification of potential anomalies or malicious activities.

In summary, while the CNN model excels in classifying different types of IoT device traffic with high accuracy, the VAE model effectively detects anomalies and irregularities within the data, contributing to a comprehensive approach for enhancing cybersecurity in IoT environments.CNN-VAE approach offers significant advantages in IoT cybersecurity, addressing limitations and exploring potential improvements are essential for further enhancing its effectiveness in detecting and mitigating security threats in IoT environments.

5. Conclusion

This study explores the application of CNN and VAE models to analyze IoT device traffic data and presents several significant findings. Experimental verification revealed the CNN model's proficiency in classifying various types of IoT device traffic, providing us with a robust classification ability to distinguish between different traffic patterns effectively. At the same time, VAE models have made significant progress in anomaly detection, which can capture the potential distribution of data and identify abnormal patterns, providing us with an effective means of anomaly detection. Finally, we can further optimize the model structure and parameters, improve the overall performance and applicability, and make greater contributions to building a more secure and reliable IoT environment. Through continuous research and exploration, we can continue to improve the level of IoT security technology and make greater contributions to building a secure and reliable IoT ecosystem.

DOI: 10.54254/2755-2721/69/20241511

However, we should also be aware of some limitations of this study. First of all, there may be an imbalance in the experimental data. The small number of data samples in some categories may affect the training and performance of the model. Second, the model may lack the ability to generalize to previously unseen or emerging IoT device traffic patterns, which may limit the model's applicability in real-world applications. In addition, the performance of the model may be affected by the selection and adjustment of hyperparameters, which requires further optimization and adjustment to obtain better performance. Therefore, we need to continuously improve and perfect the model in further research and practice to improve its effect and performance in practical applications.

References

[1] Haras, Maciej, and Thomas Skotnicki. "Thermoelectricity for IoT–A review." Nano Energy 54 (2018): 461-476.

[2] Laghari, A. A., Wu, K., Laghari, R. A., Ali, M., & Khan, A. A. (2021). A review and state of art of Internet of Things (IoT). Archives of Computational Methods in Engineering, 1-19.

[3] Lee, I., & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business horizons, 58(4), 431-440.

[4] Khan, S., Muhammad, K., Mumtaz, S., Baik, S. W., & de Albuquerque, V. H. C. (2019). Energy-efficient deep CNN for smoke detection in foggy IoT environment. IEEE Internet of Things Journal, 6(6), 9237-9245.

[5] Khan, Salman, et al. "Energy-efficient deep CNN for smoke detection in foggy IoT environment." IEEE Internet of Things Journal 6.6 (2019): 9237-9245.

[6] Shin, M., Paik, W., Kim, B., & Hwang, S. (2019). An IoT platform with monitoring robot applying CNN-based context-aware learning. Sensors, 19(11), 2525.

[7] Hussain, F., Hussain, R., Hassan, S. A., & Hossain, E. (2020). Machine learning in IoT security: Current solutions and future challenges. IEEE Communications Surveys & Tutorials, 22(3), 1686-1721.

[8] Tahsien, S. M., Karimipour, H., & Spachos, P. (2020). Machine learning based solutions for security of Internet of Things (IoT): A survey. Journal of Network and Computer Applications, 161, 102630.

[9] Aldahiri, Amani, Bashair Alrashed, and Walayat Hussain. "Trends in using IoT with machine learning in health prediction system." Forecasting 3.1 (2021): 181-206.

[10] Kusner, Matt J., Brooks Paige, and José Miguel Hernández-Lobato. "Grammar variational autoencoder." International conference on machine learning. PMLR, 2017.

[11] Vahdat, Arash, and Jan Kautz. "NVAE: A deep hierarchical variational autoencoder." Advances in neural information processing systems 33 (2020): 19667-19679.