Article

IoT Traffic Classification and Anomaly Detection Method based on Deep Autoencoders

Qi Xin * , Zeqiu Xu , Linfeng Guo , Fanyi Zhao , Binbin Wu

Posted Date: 8 July 2024

doi: 10.20944/preprints202407.0530.v1

Keywords: IoT; convolutional neural network, CNN; variational autoencoder, VAE; anomaly detection

Article

IoT Traffic Classification and Anomaly Detection Method Based on Deep Autoencoders

Qi Xin [1,*], Zeqiu Xu [1,*], Linfeng Guo [2], Fanyi Zhao [3] and Binbin Wu [4]

[1] Management Information Systems, University of Pittsburgh, Pittsburgh, PA, 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
[*] Correspondence: lubyliuu45@gmail.com (Q.X.); zeqiu12@gmail.com (Z.X.)
[†] Computer Science, Carnegie Mellon University, CA, USA.

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.

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 (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). 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[2]. Both directions follow the idea of traditional machine learning methods: design a set of traffic feature sets, build a machine learning model according to actual needs, input the feature data of known labels into the model to complete parameter training, and test the recognition performance of the model through the traffic data of unknown labels. Compared with the rule-based method, this method avoids the research process of port and traffic keywords, and can adapt to encrypted traffic and many complex traffic modes, so the computational complexity is relatively low, so it has attracted more and more attention in the academic circles in recent years.

Chan used clustering methods to compare with existing rule methods and found that machine learning methods have high performance in identifying unlabeled attack traffic data.

Omar explored the performance of different machine learning methods, supervised and unsupervised, in terms of anomaly detection accuracy, and finally found that the unsupervised method performed better against unknown attacks, but the supervised method had higher detection accuracy against known attacks.

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[3]. Because machine learning algorithms require explicit input features, feature extraction is crucial for detection effectiveness. However, the amount of information in network traffic is large and attackers will take various covert means to hide the attack traffic, so feature extraction is a difficult task, and different feature extraction methods will lead to different performance. Since the training data of machine learning algorithm is historical data, it lacks generalization ability for emerging attack behavior. Attackers will constantly try new attack methods to evade detection systems, so algorithms need to be constantly updated and optimized to adapt to new attack methods.

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)[4], Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory networks [5](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.

1.Autoencoder-based Network Attack Traffic Detection Method

This method involves building an Autoencoder to train on normal traffic data and using it to detect abnormal traffic. The main idea is that normal traffic exhibits certain regularities and characteristics, which can be learned and extracted by the autoencoder. Abnormal traffic, having different characteristics from normal traffic, can be detected by comparing the reconstruction error or anomaly score. Xu [8] successfully used an autoencoder to represent features and achieve anomaly detection within financial enterprises.

2.Convolutional Neural Network-based Network Attack Traffic Detection Method

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 [9] 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.

3.Recurrent Neural Network-based Network Attack Traffic Detection Methods

Recurrent neural network models, such as LSTM or gated recurrent units [6](GRU), are used to model and analyze network traffic data. This approach treats network traffic data as a sequence of data and uses the recurrent neural network model to achieve modeling and analysis, thus enabling the detection and identification of network attack traffic.

Although deep learning models excel at handling highly complex operations and data, and can efficiently learn temporal and spatial features of traffic, their interpretability is poor. Due to the lack of clear feature definitions, the credibility of model training results can be questioned.

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.

2.3. Variational Autoencoder (VAE)[7]

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.

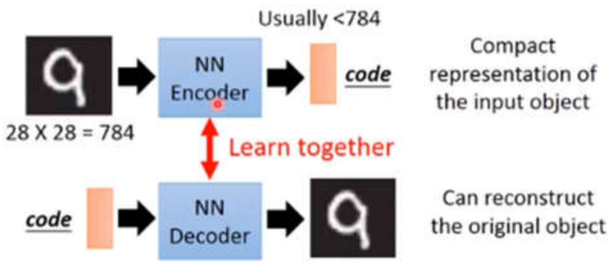

Usually <784
NN Encoder
code
Compact representation of the input object
28 X 28 = 784
Learn together
code
NN Decoder
Can reconstruct the original object

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.

AE can be applied to data dimensionality reduction, feature extraction and data visualization analysis in machine learning, and can also be extended and applied to generative models.

Auto-Encoder for a particular generation model, it should generally satisfy the following two points:

1. Encoders and decoders can be separated independently[9] (similar to GAN Generator and Discriminator).

2. Any code sampled in a fixed dimension should be able to produce a clear and true picture through the decoder.

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 $\sigma = (\sigma 1, \sigma 2, \sigma 3)$.that controls the noise interference degree. 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:

$$c = (c1, c2, c3) = \sum_{i=1}^{3}(e^{\sigma_i} - (1 + \sigma_i) + (m_i)^2) \quad (1)$$

The principle of VAE is an unsupervised generative model, which is based on Gaussian mixture model.

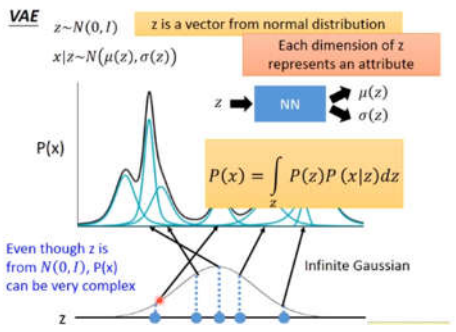

VAE
$z \sim N(0, I)$
$x|z \sim N(\mu(z), \sigma(z))$
z is a vector from normal distribution
Each dimension of z represents an attribute
z
NN
$\mu(z)$
$\sigma(z)$
P(x)
$P(x) = \int_z P(z)P(x|z)dz$
Even though z is from $N(0, I)$, P(x) can be very complex
Infinite Gaussian
z

Figure 2. Flow chart of VAE algorithm.

From the model structure of VAE[10], we can see that the noise code z is a vector generated by a standard normal distribution, which we randomly sample m points, where m follows a polynomial distribution p(x). Every time we sample a point m, we map it to a Gaussian distribution$N(\mu^m, \sigma^m)$, so a polynomial distribution using the mixture model can be represented as:

$$p(x) = \sum_m p(m)p(x \mid m) = \int z\, p(z)p(x \mid z)d_z \tag{2}$$

The $m \sim p(m), x \mid m \sim N(\mu^m, \sigma^m)$. After the above operation, we can convert the original discrete encoding with a large number of distorted areas into a continuous and effective encoding.

Therefore, variational autoencoders (VAE) -based anomaly detection methods identify anomalies by learning potential representations of the data. VAE is a generative model that compresses input data into a potential space via an encoder and then reconstructs the data via a decoder. The normal data is reconstructed efficiently by VAE, while the abnormal data has a large reconstruction error due to the different distribution from the training data. For example, in network traffic monitoring, VAE can detect abnormal traffic by analyzing normal traffic patterns, and when a significant reconstruction error is detected, it can be flagged as a potential network attack or failure.

The advantage of VAE lies in its high productivity and flexibility[11,12]. It is not only able to handle high-dimensional data, but also to capture complex patterns in the data, which makes it excellent in anomaly detection. VAE is more adaptable to new and unseen anomalies than traditional statistical or rule-based methods. For example, in the predictive maintenance of industrial equipment, VAE can be used to model the normal operation data of the equipment, and when the equipment exhibits abnormal behavior, VAE can quickly identify and issue alerts, thereby avoiding potential failures and downtime.

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.

Table 1. Dataset Details.

Dataset Path	/root/yolov8-main/dataset/data2/images
Training Set	train
Validation Set	val
Test Set	test
Number of Classes	10
Class Names	
0: pedestrian	Pedestrian
1: person	Person
2: car	Car

3: van	Van
4: bus	Bus
5: truck	Truck
6: motor	Motor
7: bicycle	Bicycle
8: awning-tricycle	Awning-tricycle
9: tricycle	Tricycle

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.

3.2. CNN Model

Convolutional layer is one of the core components in convolutional neural network (CNN), which is mainly responsible for extracting the features of input data (usually images). The convolution layer performs convolution operations by sliding small filters (also known as convolution cores or filters) over the input data, thereby extracting local features. These local features usually include visual information such as edges, textures, and shapes, which are crucial for subsequent tasks such as image classification and object detection.

The operation process of the convolution layer is shown below, using a convolution kernel to scan the entire image.

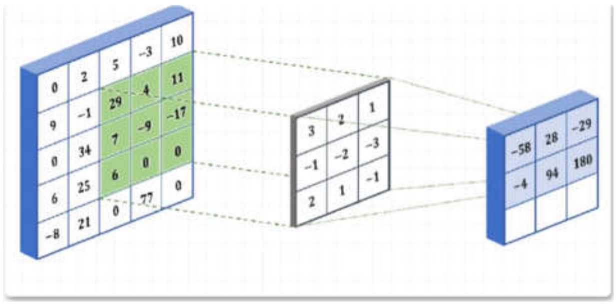


Figure 3. CNN model architecture.

In this paper, the use of CNN model mainly lies in the effective classification and identification of network traffic. Through the four convolutional layers and pooling layers in the model, the network can learn the features in the data and map them to a higher-dimensional representation space, so as to achieve feature extraction and dimensionality reduction of network traffic. This process of feature extraction and dimensionality reduction helps to reduce the complexity of data and improve the training efficiency and classification accuracy of the model.

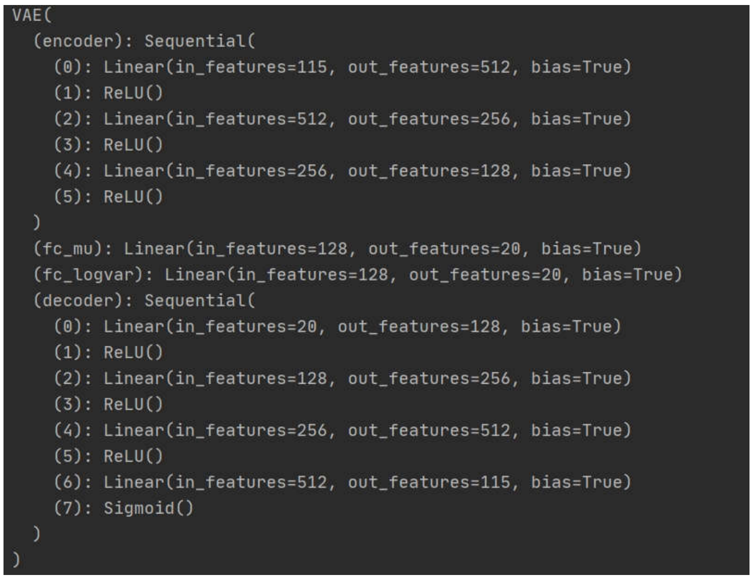
By training with the cross-entropy loss function and the Adam optimizer[13], the model can effectively learn the patterns and laws in the network traffic data and make accurate classification. The cross-entropy loss function is widely used in multi-class classification problems, which can measure the difference between the model prediction results and the actual labels, so as to guide the model optimization parameters to improve the classification accuracy. Adam optimizer is an adaptive learning rate optimization algorithm, which can automatically adjust the learning rate according to the gradient of parameters, accelerate the convergence speed of the model, and avoid falling into the local optimal solution.

Therefore, in this experiment, network traffic can be classified and identified efficiently through the application of CNN model, which provides strong support and guarantee for threat detection and defense in the field of network security.

doi:10.20944/preprints202407.0530.v1

3.3. Variational Autoencoder (AVE) Model

In this experiment, VAE (Variational Autoencoder) model plays an important role and has significant advantages. [14]The VAE model is composed of encoders and decoders, and its main purpose is to learn the potential representation of the data and realize the transformation of the potential space through the re-parameterization technique. In this experiment, the functions and advantages of VAE model can be further elaborated as follows:



```
VAE(
  (encoder): Sequential(
    (0): Linear(in_features=115, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=256, bias=True)
    (3): ReLU()
    (4): Linear(in_features=256, out_features=128, bias=True)
    (5): ReLU()
  )
  (fc_mu): Linear(in_features=128, out_features=20, bias=True)
  (fc_logvar): Linear(in_features=128, out_features=20, bias=True)
  (decoder): Sequential(
    (0): Linear(in_features=20, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=256, bias=True)
    (3): ReLU()
    (4): Linear(in_features=256, out_features=512, bias=True)
    (5): ReLU()
    (6): Linear(in_features=512, out_features=115, bias=True)
    (7): Sigmoid()
  )
)
```

Figure 4. Variational Autoencoder (AVE) Model.

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[15]: 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.

To sum up, VAE model plays an important role in this experiment, such as learning data potential representation, data reconstruction and generation, exploratory analysis of potential space, and has significant advantages in loss function optimization, data enhancement and model generalization.

3.4. Anomaly Detection Method

To effectively detect anomalies in the test data, Variational Autoencoder (VAE) is used for data reconstruction. The specific steps are as follows:

1. We use training data to train VAE models. During training, VAE learns how to compress input data into a low-dimensional latent space and reconstruct it back to its original space via a decoder. In this way, VAE is able to grasp the distribution characteristics of the input data, thus minimizing errors during reconstruction.

2. Apply the trained VAE model to the test data. For each test sample, VAE generates its reconstructed version and calculates the error between the original data and the reconstructed data. This Error is usually expressed as Mean Squared Error (MSE), i.e. :

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \check{x}_i)^2 \tag{3}$$

Where x_i is the original data, $\check{x}_i$ is the reconstructed data, and n is the dimension of the data.

In order to determine an effective anomaly detection threshold, we need to analyze the reconstruction error distribution of the training data. The specific method is to calculate the reconstruction error of all training samples and find the maximum value, which is denoted as max_mse,train. This maximum reconstruction error represents the maximum range of errors that can be accepted in the training data.

In the actual detection process, if the reconstruction error of a test sample exceeds

max_mse,train, we mark the sample as abnormal data. This is because under normal circumstances, the distribution of the test data should be similar to the training data, and if the reconstruction error of a test sample is too large, it may contain features or noise not seen in the VAE, and thus be considered abnormal.

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

(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)

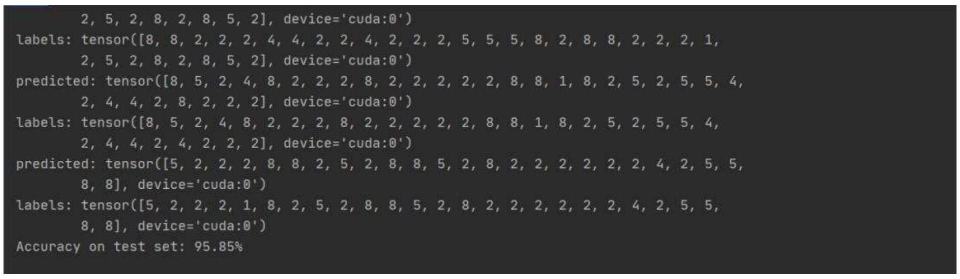
Component	Details
Convolutional Layers	Conv1: Input 1, Output 32, Kernel size 3, Padding 1 Conv2: Input 32, Output 64, Kernel size 3, Padding 1 Conv3: Input 64, Output 128, Kernel size 3, Padding 1 Conv4: Input 128, Output 128, Kernel size 3, Padding 1
Pooling Layers	Max Pooling, Kernel size 2, Stride 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 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. This process repeats for each batch of data for a specified number of epochs, gradually optimizing the model parameters towards better performance.

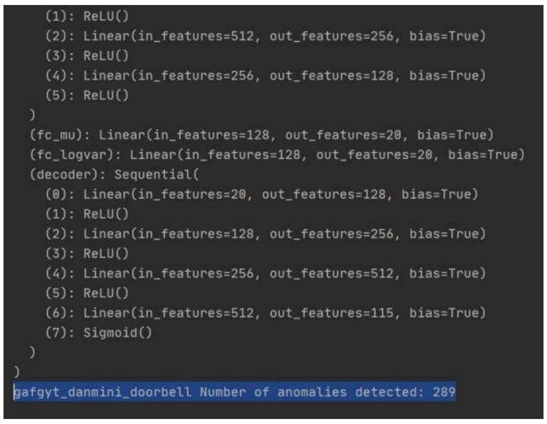


```
        2, 5, 2, 8, 2, 8, 5, 2], device='cuda:0')
labels: tensor([8, 8, 2, 2, 2, 4, 4, 2, 2, 4, 2, 2, 2, 5, 5, 5, 8, 2, 8, 8, 2, 2, 2, 1,
        2, 5, 2, 8, 2, 8, 5, 2], device='cuda:0')
predicted: tensor([8, 5, 2, 4, 8, 2, 2, 2, 8, 2, 2, 2, 2, 2, 8, 8, 1, 8, 2, 5, 2, 5, 5, 4,
        2, 4, 4, 2, 8, 2, 2, 2], device='cuda:0')
labels: tensor([8, 5, 2, 4, 8, 2, 2, 2, 8, 2, 2, 2, 2, 2, 8, 8, 1, 8, 2, 5, 2, 5, 5, 4,
        2, 4, 4, 2, 4, 2, 2, 2], device='cuda:0')
predicted: tensor([5, 2, 2, 2, 8, 8, 2, 5, 2, 8, 8, 5, 2, 8, 2, 2, 2, 2, 2, 2, 4, 2, 5, 5,
        8, 8], device='cuda:0')
labels: tensor([5, 2, 2, 2, 1, 8, 2, 5, 2, 8, 8, 5, 2, 8, 2, 2, 2, 2, 2, 2, 4, 2, 5, 5,
        8, 8], device='cuda:0')
Accuracy on test set: 95.85%
```

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.

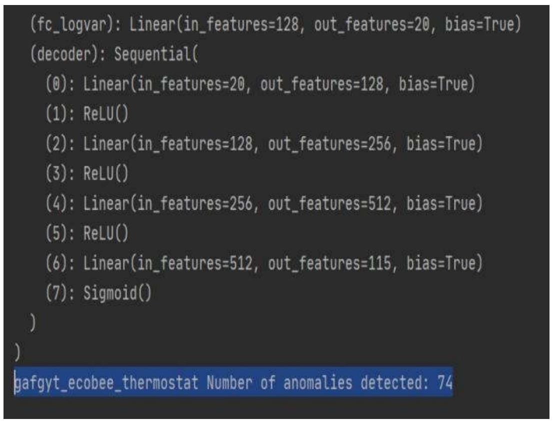
4.4. Experimental Result

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:



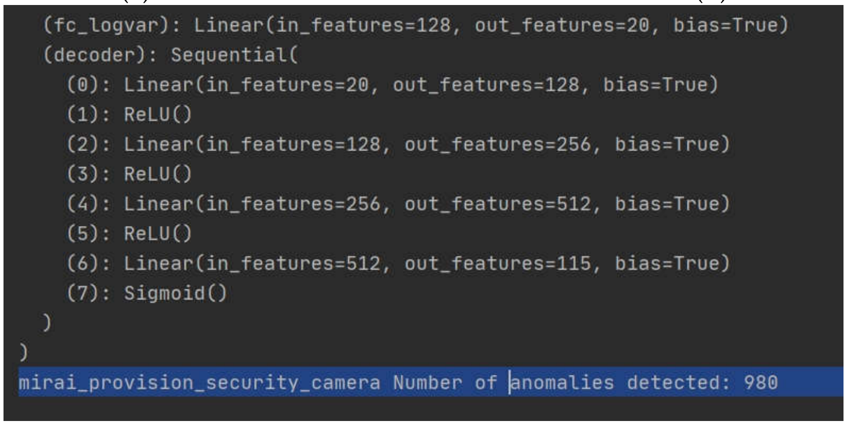
```
    (1): ReLU()
    (2): Linear(in_features=512, out_features=256, bias=True)
    (3): ReLU()
    (4): Linear(in_features=256, out_features=128, bias=True)
    (5): ReLU()
  )
  (fc_mu): Linear(in_features=128, out_features=20, bias=True)
  (fc_logvar): Linear(in_features=128, out_features=20, bias=True)
  (decoder): Sequential(
    (0): Linear(in_features=20, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=256, bias=True)
    (3): ReLU()
    (4): Linear(in_features=256, out_features=512, bias=True)
    (5): ReLU()
    (6): Linear(in_features=512, out_features=115, bias=True)
    (7): Sigmoid()
  )
)
gafgyt_danmini_doorbell Number of anomalies detected: 289
```

(**a**)



```
  (fc_logvar): Linear(in_features=128, out_features=20, bias=True)
  (decoder): Sequential(
    (0): Linear(in_features=20, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=256, bias=True)
    (3): ReLU()
    (4): Linear(in_features=256, out_features=512, bias=True)
    (5): ReLU()
    (6): Linear(in_features=512, out_features=115, bias=True)
    (7): Sigmoid()
  )
)
gafgyt_ecobee_thermostat Number of anomalies detected: 74
```

(**b**)



```
  (fc_logvar): Linear(in_features=128, out_features=20, bias=True)
  (decoder): Sequential(
    (0): Linear(in_features=20, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=256, bias=True)
    (3): ReLU()
    (4): Linear(in_features=256, out_features=512, bias=True)
    (5): ReLU()
    (6): Linear(in_features=512, out_features=115, bias=True)
    (7): Sigmoid()
  )
)
mirai_provision_security_camera Number of anomalies detected: 980
```

(**c**)

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. Together, these images offer a comprehensive overview of the detection performance across diverse IoT device categories and malware types, highlighting the effectiveness of the anomaly detection system in identifying and flagging suspicious traffic patterns.

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.

On the other hand, the VAE model exhibits proficient anomaly detection capabilities. By leveraging the reconstruction loss and KL divergence, the VAE model effectively captures the underlying data distribution and detects deviations from normal patterns. This enables the VAE model to identify anomalies or irregularities within the IoT device traffic data, even in the absence of explicit labels. The anomaly detection performance of the VAE model enhances the overall security posture by flagging potentially malicious activities or unexpected behaviors, thereby aiding in the timely detection and mitigation of cybersecurity threats targeting IoT devices.

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. By combining the strengths of both models, organizations can better safeguard their IoT ecosystems against a wide range of security threats and vulnerabilities.

4.6. Improvement Strategy

The combined use of CNN and VAE models offers significant advantages in enhancing the detection and classification of IoT device traffic. By integrating CNN for classification and VAE for anomaly detection, this approach provides a comprehensive solution to cybersecurity challenges in IoT environments.

The CNN model demonstrates impressive classification performance, achieving a high accuracy rate of 95.85% on the test dataset. Its strength lies in accurately categorizing different types of IoT device traffic, showcasing its ability to identify and distinguish various patterns within the data. This classification capability is crucial for distinguishing normal traffic from potentially malicious activities, thereby aiding in the identification of security threats.

The VAE model complements the CNN by providing proficient anomaly detection capabilities. By leveraging reconstruction loss and KL divergence, the VAE effectively captures the underlying data distribution and detects deviations from normal patterns. This enables the model to identify anomalies or irregularities within the data, even in the absence of explicit labels. The anomaly detection performance of the VAE enhances the overall security posture by flagging potentially malicious activities or unexpected behaviors, thereby contributing to timely threat detection and mitigation.

Despite these strengths, there are limitations to consider in the experimental results and the combined CNN-VAE approach. Data imbalance, where certain classes of IoT device traffic are underrepresented, may lead to biased classification outcomes and reduced accuracy. Additionally, the models may exhibit limited generalization to unseen or novel traffic patterns, impacting their ability to adapt to dynamic environments.Data augmentation and balancing techniques can mitigate data imbalance issues, while transfer learning and ensemble learning approaches can enhance model generalization and robustness. Moreover, implementing mechanisms for dynamic adaptation based on real-time feedback can improve the model's adaptability to changing traffic patterns and emerging threats.

In conclusion, while the combined [16,17]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[18]. By combining these two models, we achieve comprehensive monitoring and prevention of security threats in the IoT environment, providing strong technical support for building a more secure IoT ecosystem.

In future research, we can further expand and improve the results of this research, and propose the following possible research directions and improvement points. First, we can explore more complex and diverse IoT device traffic data and try to apply more advanced deep learning models to address increasingly complex and changing cybersecurity threats. Secondly, we can combine deployment and application in actual scenarios to carry out more in-depth experiments and verification, verify the practicality and effectiveness of the model, and provide more powerful support and guarantee for practical applications in the field of IoT security. 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.

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 model's performance may be affected by the selection and adjustment of hyperparameters, which requires further optimization and adjustment to obtain better performance. Therefore, we must continuously improve and perfect the model in further research and practice to improve its effect and performance in practical applications.

References

1. Zhan, X., Shi, C., Li, L., Xu, K., & Zheng, H. (2024). Aspect category sentiment analysis based on multiple attention mechanisms and pre-trained models. Applied and Computational Engineering, 71, 21-26.
2. Wu, B., Xu, J., Zhang, Y., Liu, B., Gong, Y., & Huang, J. (2024). Integration of computer networks and artificial neural networks for an AI-based network operator. Applied and Computational Engineering, 64, 115-120.
3. Liang, P., Song, B., Zhan, X., Chen, Z., & Yuan, J. (2024). Automating the training and deployment of models in MLOps by integrating systems with machine learning. Applied and Computational Engineering, 67, 1-7.
4. Li, A., Yang, T., Zhan, X., Shi, Y., & Li, H. (2024). Utilizing Data Science and AI for Customer Churn Prediction in Marketing. Journal of Theory and Practice of Engineering Science, 4(05), 72-79.
5. Wu, B., Gong, Y., Zheng, H., Zhang, Y., Huang, J., & Xu, J. (2024). Enterprise cloud resource optimization and management based on cloud operations. Applied and Computational Engineering, 67, 8-14.
6. Xu, J., Wu, B., Huang, J., Gong, Y., Zhang, Y., & Liu, B. (2024). Practical applications of advanced cloud services and generative AI systems in medical image analysis. Applied and Computational Engineering, 64, 82-87.
7. Huang, J., Zhang, Y., Xu, J., Wu, B., Liu, B., & Gong, Y. Implementation of Seamless Assistance with Google Assistant Leveraging Cloud Computing.
8. Zhang, Y., Liu, B., Gong, Y., Huang, J., Xu, J., & Wan, W. (2024). Application of machine learning optimization in cloud computing resource scheduling and management. Applied and Computational Engineering, 64, 9-14.

9. Shi, Y., Li, L., Li, H., Li, A., & Lin, Y. (2024). Aspect-Level Sentiment Analysis of Customer Reviews Based on Neural Multi-task Learning. Journal of Theory and Practice of Engineering Science, 4(04), 1-8.
10. Shi, Y., Yuan, J., Yang, P., Wang, Y., & Chen, Z. Implementing Intelligent Predictive Models for Patient Disease Risk in Cloud Data Warehousing.
11. Zhan, T., Shi, C., Shi, Y., Li, H., & Lin, Y. (2024). Optimization Techniques for Sentiment Analysis Based on LLM (GPT-3). arXiv preprint arXiv:2405.09770.
12. Li, Huixiang, et al. "AI Face Recognition and Processing Technology Based on GPU Computing." Journal of Theory and Practice of Engineering Science 4.05 (2024): 9-16.
13. Yuan, J., Lin, Y., Shi, Y., Yang, T., & Li, A. (2024). Applications of Artificial Intelligence Generative Adversarial Techniques in the Financial Sector. Academic Journal of Sociology and Management, 2(3), 59-66.
14. Lin, Y., Li, A., Li, H., Shi, Y., & Zhan, X. (2024). GPU-Optimized Image Processing and Generation Based on Deep Learning and Computer Vision. Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023, 5(1), 39-49.
15. Chen, Zhou, et al. "Application of Cloud-Driven Intelligent Medical Imaging Analysis in Disease Detection." Journal of Theory and Practice of Engineering Science 4.05 (2024): 64-71.
16. Wang, B., Lei, H., Shui, Z., Chen, Z., & Yang, P. (2024). Current State of Autonomous Driving Applications Based on Distributed Perception and Decision-Making.
17. Jiang, W., Qian, K., Fan, C., Ding, W., & Li, Z. (2024). Applications of generative AI-based financial robot advisors as investment consultants. Applied and Computational Engineering, 67, 28-33.
18. Ding, W., Zhou, H., Tan, H., Li, Z., & Fan, C. (2024). Automated Compatibility Testing Method for Distributed Software Systems in Cloud Computing.
19. Fan, C., Li, Z., Ding, W., Zhou, H., & Qian, K. Integrating Artificial Intelligence with SLAM Technology for Robotic Navigation and Localization in Unknown Environments.
20. Guo, L., Li, Z., Qian, K., Ding, W., & Chen, Z. (2024). Bank Credit Risk Early Warning Model Based on Machine Learning Decision Trees. Journal of Economic Theory and Business Management, 1(3), 24-30.
21. Li, Zihan, et al. "Robot Navigation and Map Construction Based on SLAM Technology." (2024).
22. Fan, C., Ding, W., Qian, K., Tan, H., & Li, Z. (2024). Cueing Flight Object Trajectory and Safety Prediction Based on SLAM Technology. Journal of Theory and Practice of Engineering Science, 4(05), 1-8.
23. Srivastava, S., Huang, C., Fan, W., & Yao, Z. (2023). Instance Needs More Care: Rewriting Prompts for Instances Yields Better Zero-Shot Performance. arXiv preprint arXiv:2310.02107.
24. Xiao, J., Wang, J., Bao, W., Deng, T. and Bi, S., Application progress of natural language processing technology in financial research.
25.
26. Ding, W., Tan, H., Zhou, H., Li, Z., & Fan, C. Immediate Traffic Flow Monitoring and Management Based on Multimodal Data in Cloud Computing.
27. Qian, K., Fan, C., Li, Z., Zhou, H., & Ding, W. (2024). Implementation of Artificial Intelligence in Investment Decision-making in the Chinese A-share Market. Journal of Economic Theory and Business Management, 1(2), 36-42.
28. Tian, J., Li, H., Qi, Y., Wang, X., & Feng, Y. (2024). Intelligent Medical Detection and Diagnosis Assisted by Deep Learning. Applied and Computational Engineering, 64, 121-126.
29. Zhou, Y., Zhan, T., Wu, Y., Song, B., & Shi, C. (2024). RNA Secondary Structure Prediction Using Transformer-Based Deep Learning Models. arXiv preprint arXiv:2405.06655.
30. Liu, B., Cai, G., Ling, Z., Qian, J., & Zhang, Q. (2024). Precise Positioning and Prediction System for Autonomous Driving Based on Generative Artificial Intelligence. Applied and Computational Engineering, 64, 42-49.
31. Cui, Z., Lin, L., Zong, Y., Chen, Y., & Wang, S. (2024). Precision Gene Editing Using Deep Learning: A Case Study of the CRISPR-Cas9 Editor. Applied and Computational Engineering, 64, 134-141.
32. Yu, D., Xie, Y., An, W., Li, Z., & Yao, Y. (2023, December). Joint Coordinate Regression and Association For Multi-Person Pose Estimation, A Pure Neural Network Approach. In Proceedings of the 5th ACM International Conference on Multimedia in Asia (pp. 1-8).
33. Wang, B., He, Y., Shui, Z., Xin, Q., & Lei, H. (2024). Predictive Optimization of DDoS Attack Mitigation in Distributed Systems using Machine Learning. Applied and Computational Engineering, 64, 95-100.
34. He, Zheng, et al. "Application of K-means clustering based on artificial intelligence in gene statistics of biological information engineering."