Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2021.016264

ARTICLE

Multi-Layer Reconstruction Errors Autoencoding and Density Estimate for Network Anomaly Detection

Ruikun Li[1,*], Yun Li[2], Wen He[1,3], Lirong Chen[1] and Jianchao Luo[1]

[1]University of Electronic Science and Technology of China, Chengdu, 611730, China

[2]Guangdong Weichen Information Technology Co., Ltd., Guangzhou, 510000, China

[3]Chongqing Changan Automobile Corporation, Chongqing, 400000, China

[*]Corresponding Author: Ruikun Li. Email: dayslrk@163.com

Received: 21 February 2021 Accepted: 12 March 2021

ABSTRACT

Anomaly detection is an important method for intrusion detection. In recent years, unsupervised methods have been widely researched because they do not require labeling. For example, a nonlinear autoencoder can use reconstruction errors to attain the discrimination threshold. This method is not effective when the model complexity is high or the data contains noise. The method for detecting the density of compressed features in a hidden layer can be used to reduce the influence of noise on the selection of the threshold because the density of abnormal data in hidden layers is smaller than normal data. However, compressed features may lose some of the high-dimensional distribution information of the original data. In this paper, we present an efficient anomaly detection framework for unsupervised anomaly detection, which includes network data capturing, processing, feature extraction, and anomaly detection. We employ a deep autoencoder to obtain compressed features and multi-layer reconstruction errors, and feeds them the same to the Gaussian mixture model to estimate the density. The proposed approach is trained and tested on multiple current intrusion detection datasets and real network scenes, and performance indicators, namely accuracy, recall, and F1-score, are better than other autoencoder models.

KEYWORDS

Anomaly detection; deep autoencoder; density estimate

1 Introduction

In the last few years, numerous network attacks have emerged. An intrusion detection system acts as a barrier for a computer system, and can detect network anomalies. A typical intrusion detection method includes anomaly detection and misuse detection. The latter monitors network traffic or system activities for known misuse behaviors and can add known simple attacks to the model; however, it cannot detect unknown attacks. Anomaly detection identifies anomalies by establishing rules for normal behavior. With the extensive application of deep learning, unsupervised anomaly detection method is gaining increasing prominence [1,2] because it can train samples without labels and detect unknown attacks.

Normal behavior rules can be established without labeling by using unsupervised anomaly detection, and an appropriate value can be set as a detection threshold. Supervised learning has a number of shortcomings, which can be enumerated as follows: 1) it requires manual labeling, which is time-consuming and costly; 2) manually labeled data may be misclassified, which can potentially affect the effect of training; 3) training data for supervised model classification cannot cover huge types of attacks, which makes it challenging to identify new types of attack methods [3,4].

Although unsupervised models can detect unknown attack behavior, it is difficult to select the threshold to detect anomalies with low accuracy and high false alarm rate. Such models therefore cannot attain the requisite effect and need improvement.

Owing to the stated limitations, we propose a lightweight and network-based anomaly detection framework MEAEDE, which comprises a deep autoencoder (DAE) and density estimate network (DEN). To estimate the density of data for anomaly detection, we derive multi-layer reconstruction errors (MRE), which is the correlation coefficient of input and output from DAE. As the data in the latent layer has a low dimension to compute density, we can use normal data in the latent layer to form a standard feature space, while abnormal data in the latent can form another feature space, and thus it can be easily distinguished. The modules of our framework include network data capturing, managing, feature extraction, and anomaly detector. DAE is deployed in the detector for data compression and reconstruction, and then a Gaussian mixture model (GMM) is used to calculate the energy of features from DAE to obtain the detection threshold. Our framework has the following characteristics:

1) **Packet capture:** We use network cards to capture packets to obtain real-time data; to avoid data accumulation in memory, we quickly process and discard the processed data.
2) **Unsupervised model:** Compressed features, MRE, and correlation coefficients between input and reconstruction of DAE are first gathered, and then fed to GMM for energy detection; data can be then classified as anomalous if energy exceeds a threshold. Our experimental results prove that MRE can compensate for the loss of information for a latent layer, which can in turn enhance the performance of anomaly detection.

In DAE, abnormal data cannot represent compressed features effectively; therefore, reconstruction error is generally used to detect the anomaly. However, samples used for model training may contain noise, which could introduce problems during threshold acquisition. Density estimation in a hidden layer can decelerate this impact and also cause loss of pertinent sample information. With the aim of feature compensation in hidden layer of autoencoder for density estimation in GMM, we gather compressed features, MRE, and the correlation coefficient between the raw input and reconstruction features, feed them to GMM for density detection, and finally construct an appropriate objective function to jointly optimize sample loss. Furthermore, we adopt the acquired energy value as the standard for obtaining a discrimination threshold. The experimental results show that there is a strong correlation between these factors, which can effectively classify normal and abnormal data.

In brief, the contributions of this paper can be enumerated as follows:

1. We propose a lightweight network-based anomaly detection framework MEAEDE, which includes data capturing, efficient data processing, feature extraction, and anomaly detection. First, packets in the network are captured and parsed to bidirectional flows by a Packet Capturer module; next, time-related features are extracted by a Feature Extractor module,

which are further sent to an Anomaly Detector module to detect whether the data is normal. During this entire procedure, a Data Manager module is responsible for continuously capturing packets and removing processed packets.
2. Our framework can efficiently process and manage data including rapid feature extraction and prompt discarding of processed data; the experiment results show that MEAEDE has a high speed in training the model and easy deployment of the network.
3. MRE and other features from DAE are employed to form new features c, which are further fed into GMM to calculate energy and acquire the detection threshold. Our experimental results show that these features can greatly improve the performance of network anomaly detectors. We will discuss how MRE can improve the detection results of the autoencoder in Section 3.

The rest of this paper is organized as follows: Section 2 discusses related work in the domain of unsupervised anomaly detection. Section 3 describes the architecture of MEAEDE and details of its components. Section 4 presents the evaluation and includes experimental settings and test results. Section 5 outlines the conclusion and future work.

2 Related Work

There is two mainstream intrusion detection that is based on supervised and unsupervised methods. In the past, supervised learning methods have been widely researched because of their high accuracy, but they also have obvious shortcomings. First, the training data used to feed needs to be manually labeled; it is difficult to define abnormal and normal data and differentiate between them. Moreover, they have a low performance for the detection of unseen attacks, and thus unsupervised methods have been extensively researched in the past few years.

2.1 Conventional Anomaly Detection

Previous studies based on unsupervised anomaly detection have various approaches, which can be briefly categorized into distance-based, reconstruction-based and probabilistic-based methods [2]. Probabilistic-based methods such as GMM [5] and Kernal density estimate (KDE) [6] are used to estimate the density of normal points; a point will be classified as an anomaly if it has a lower density than a predefined value. Reconstruction-based method, such as principal component analysis (PCA) and autoencoder methods, assumes that the anomalies cannot effectively reconstruct from a compressed representation in a low dimension; therefore, higher reconstruction error may be detected as anomalies. PCA is used to reduce the dimension of input data when the sample has a high dimension [7,8]. This method can reduce the cost when data has a high complexity. However, a drawback of PCA is that the new features obtained after dimension reduction do not acquire all information from the original space, and it's hard to determine the dimension of subspace and detection threshold. Distance-based methods assume that abnormal data is far from a clustered set, while normal data is gathered within an acceptable cluster. In [9], the authors use an unsupervised clustering method to establish a classification center for data and calculates the distance of samples from the center to detect anomalies. However, the feature vector with a higher dimension has a high computational complexity, and it is not advisable to represent features simply by computing distance.

2.2 Autoencoder-Based Approaches

The state-of-the-art machine learning methods are widely used in anomaly detection [10]. With developments in deep learning, a few unsupervised methods, such as generative adversarial networks (GANs) and autoencoders, have been used in anomaly detection. In [11], the authors

apply a GAN model for anomaly detection and detect anomalies on a trained generator. In [12], anomaly detection is accomplished using a variational autoencoder predicated on a gradient-based fingerprinting technique. In [13], the authors use layers of autoencoders to calculate the mean square error of sample and reconstructed features in order to capture the data of real environment and obtain advantages in an effective way.

In the last few years, methods for obtaining reconstruction error from DAE have achieved effective results [14,15]. In these methods, normal data is first trained and its reconstruction is obtained, and then test data is used to get the deviation from normal data of a trained model; if the deviation exceeds the predetermined threshold, the data is classified as anomaly. However, these methods rely only upon reconstruction errors, and the effect is not ideal when the training sample contains noise or the model has a high complexity. The reconstruction error of anomalies may be hidden within the established threshold range because of the presence of noise. Therefore, some scholars have proposed a hidden layer density estimate method because the density of abnormal data in the hidden layer is lower than normal data. In [16], the authors combine autoencoder and density estimate for anomaly detection; their approach is based on the feature density of the hidden layer in the autoencoding network. First, the DAE network is utilized to acquire the subspace of original data; next, a Gaussian kernel function is used to obtain the likelihood of data points. This method can efficiently estimate the density of the sample in the hidden layer in a low dimension.

2.3 Mixed Approaches Combine Autoencoder and Density Estimate

The aforementioned methods either consider only the density of the hidden layer or the reconstruction error and fail to realize the effect of their inter-relation on the prediction of the sample density, and thus the resultant effect is not ideal when the training sample contains noises. In [17], the authors propose a deep autoencoding Gaussian mixture model (DAGMM) by combining DAE with GMM. The reconstruction error and compressed features in the hidden layer are adopted to construct new features, energy is acquired after feeding new features into GMM, and desirable results are obtained. However, the threshold of discrimination needs to be set in accordance with the prior knowledge of the dataset, and thereby the approach lacks generalization ability; In addition density estimation on compressed layer loses some high-level information, and thus the method needs to be improved in the actual scenario.

Inspired by the density estimation method, we adopt DAE and gather compressed features, MRE, and the correlation coefficient of input and reconstruction as new features, and then construct a suitable energy detection network to calculate the energy of the new features. Because normal data has lower energy (i.e., higher density) during training, the energy value is considered a part of the loss function, so the threshold for judging anomalies is obtained through joint optimization. One can infer from Section 3 why we adopt MRE, coefficient correlation and compressed features. In addition, we propose a complete and efficient network-based anomaly detection framework, optimized in data capturing, package management, feature extraction and anomaly detection. Precision, Recall, and F1-score are used as indicators for evaluating the proposed model's performance. The obtained experimental results confirm the advantages of our model.

3 Proposed Model

The framework of anomaly detection in this paper includes several modules of Data Capturer, Data Manager, Feature Extractor and Anomaly Detector and is depicted in Fig. 1.

3.1 Data Capturer

Data Capturer can continuously capture data from real network. The captured data can reflect information such as the source and purpose of the user. For example, the attacker who wants to attack the server using a denial-of-service (DoS) attack or a distributed-of-service (DDoS) attack will create a fake IP address, and then send a TCP SYN packet to the server. The server then sends an SYN as a response but fails. Receipt of subsequent confirmation packets causes the server to wait, consume resources, and evently achieve the purpose of DoS.

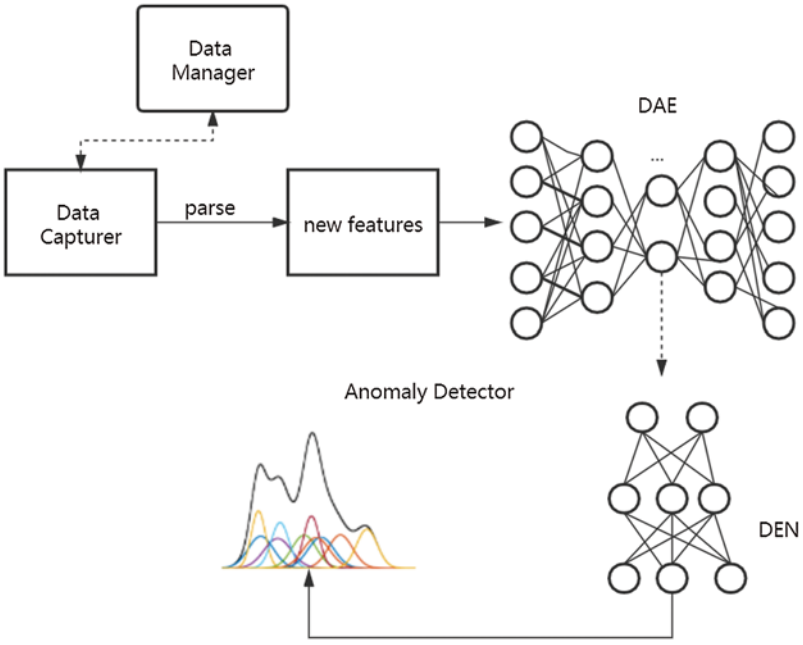

Figure 1: The framework and components of MEAEDE

The characteristics of the package can reflect its purpose to a certain extent, so it is possible to analyze whether the data is abnormal by analyzing the package. In this paper, libpcap is used to capture the original binary package, and then tcpdump can call the interface of libpcap and parse acquired raw packet into a standard package format for use by Data Manager.

3.2 Data Manager

The captured packet is saved in the ".pcap" files after parsing. In order to improve the efficiency of memory management and feature extraction, we revise the tcpdump source code so that it not only can capture packets continuously but also discard the processed data in time after feature extraction. In our experimental results in Section 4, average extraction time of 14s for every 100 k ".pcap" file is obtained.

3.3 Feature Extractor

Changes in applications and services pose challenges to traffic analysis, and thereby make it difficult to handle encrypted information such as in VPN. There are various protocols or services used for different applications. Time-related feature extraction is well adapted to the changes of network services [18] and can be used to distinguish between different applications. We extract bidirectional flows from the parsed packets, which are defined as the same packet sequence

(source IP, destination IP, source port, destination port, and protocol) and then estimate the time-related information from the flows as features. In a bidirectional flow, the first packet determines the forward (source to destination) and backward (destination to source) directions, so statistical analysis based on time-related features can be calculated separately. Since TCP and UDP flows end upon receiving a FIN packet and with flow timeout, the timeout of flow is unified to 15s in this paper.

The collective procedure of Data Capturer and Feature Extractor is illustrated in Fig. 2, where $t_1 \sim t_n$ is the start time of a flow, and the packet vector is a ".pcap" file. If the time interval exceeds 15s or a FIN flag is received, the flow will be sent to Feature Extractor for further analysis.

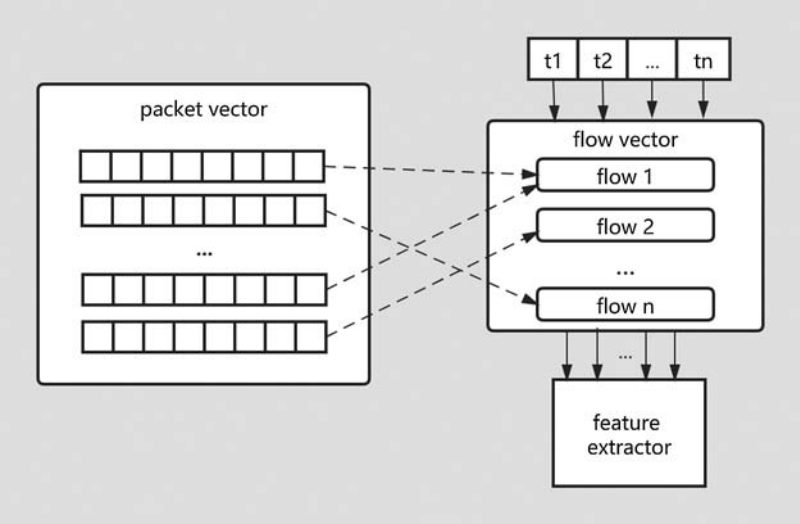

Figure 2: The procedure of data capturer and feature extractor

At the beginning, we initialize n flows in the flow vector $F = (f_1, f_2, \ldots, f_n)$ and then use revised tcpdump to capture packets continuously; if the size of packet vector exceeds 10 k, the packet file is moved, and a new vector is created with the same name, while packet is sent to its own flow. A time vector is also created to record the previous time of a flow; if the current timeout exceeds 15 s or a FIN flag of a TCP packet is received, then the flow is sent to Feature Extractor for further analysis.

Finally, we extract the features including time-related statistical information (mean and variance of packet length), packet number, packet length, and TCP flag bit from the flows to form input features of the anomaly detector. Finally, we acquire 49 features showed in Tab. 1, and the extracted features are normalized to 0-1 for training and testing of anomaly detector.

3.4 Anomaly Detector

Owing to the merits of the unsupervised model summarized in Section 1, DAE is used in this paper. However, the method of acquiring the threshold on the basis of reconstruction error is sensitive to the selection of parameters and can be easily affected by noise. Moreover, compressed features are not taken into account, which in turn hinders the detection effect. Therefore, MRE, compressed features in the hidden layer, and correlation coefficients of input and reconstruction are gathered to form new features. GMM is further used to calculate sample energy, and we design

appropriate loss functions, and use the random gradient descent method (SGD) to jointly optimize the loss of each component. Details will be introduced in the following sections.

Table 1: Description and value of features in a bidirectional flow

Features	Value
Duration (time duration of a flow)	–
sdt (src to dst time of a flow)	Mean, max, min, std
dst (dst to src time of a flow)	Mean, max, min, std
a2in (packet num from active to idle)	Mean, max, min, std
i2an (packet num from idle to active)	Mean, max, min, std
tfbn (tcp flag bit num)	SYN, FIN, RST, PSH, ACK, URG
lph (length of packet head)	Mean, max, min, std
lpd (length of packet data)	Mean, max, min, std
ps (packet size in a flow)	Mean, max, min, std
idt, act (idle time and active time)	Mean, max, min, std
noss2d (number of sequence from src to dst)	Mean, max, min, std
nosd2s (number of sequence from dst to src)	Mean, max, min, std
fps (flow packet number/s)	–
fbs (flow byte number/s)	–

The procedure of acquiring MRE from autoencoder is illustrated in Fig. 3, wherein the green neurons denote hidden layer neurons, yellow neurons are compressed features, and blue neurons are input and output data points. In Figs. 4 and 5, MRE distribution of KDDCUP99 and NSL-KDD datasets in a seven-layer autoencoder (including four layers of the encoder and three layers of the decoder) shows that symmetry reconstruction errors can clearly divide normal points and anomalies. In Figs. 4 and 5, cost1 represents the error of the first hidden layer of the encoder and the second layer of the decoder, cost2 represents the error of the second hidden layer of the encoder and the first hidden layer, and cost3 is the error of input data and output data of the decoder. Therefore, in our experiment, described in Section 4, we derive MRE and other features to acquire a new feature vector for density estimation of GMM in the last epoch of the training procedure, and the results reveal that MRE is a powerful feature in our network-based anomaly detection framework.

3.4.1 Loss of DAE

Generally, DAE consists of an encoder and a decoder; the encoder can compress original data into latent representation, while the decoder tries to reconstruct original data from latent layer. Let mark x_i be an original data point; we can then get a latent representation h_i by encoder function $\text{enc}(x_i, \theta)$, where θ denotes parameters of the encoder. Furthermore, the reconstruction vector can be expressed as $y_i = dec(\gamma; h_i)$, where γ is the parameters of the decoder. We define the objective function of DAE as

$$L_1(\theta, \gamma) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|x_i - y_i\|^2\right) \tag{1}$$

where m represents the count of input data.

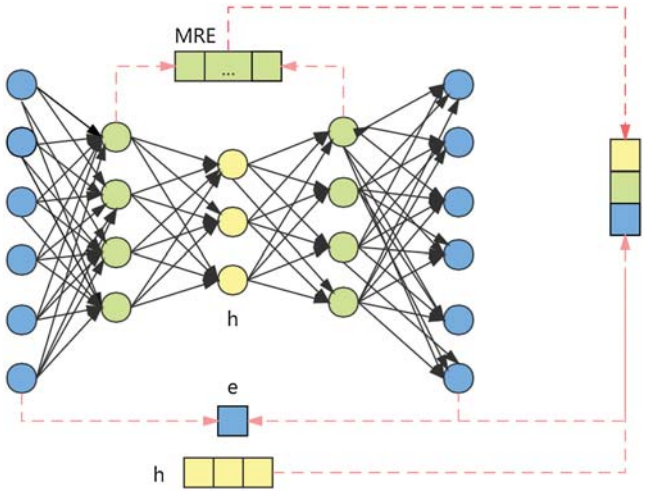

Figure 3: The procedure of MRE, where green points denote hidden layer neuros, yellow points are compressed features, and blue points are input and output data point

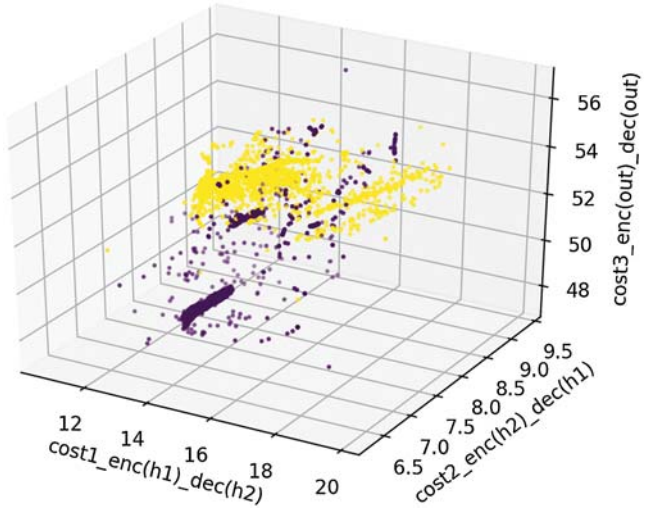

Figure 4: Reconstruction error distribution under KDDCUP99 with two hidden layers of the encoder and the decoder

3.4.2 New Features

To represent the reconstruction error, we use the root mean square error (RMSE), which represents the arithmetic square root of the difference between input and reconstruction and is defined as

$$e_i = \sqrt{\frac{\sum_{j=1}^{d}(x_i^{(j)} - y_i^{(j)})^2}{d}} \tag{2}$$

where d denotes the dimension of input, i is the i-th data point, and j is the j-th dimension of a layer. Pearson's correlation coefficient is obtained by dividing the covariance between input and

reconstruction by the product of the standard deviations between variables and is expressed as

$$\rho_i = \frac{cov(x_i, y_i)}{\sigma_{x_i}.\sigma_{y_i}} \tag{3}$$

where cov(.) is the covariance of x_i and y_i, σ is the standard deviation of a vector. Further, we can get MRE using Algorithm 1.

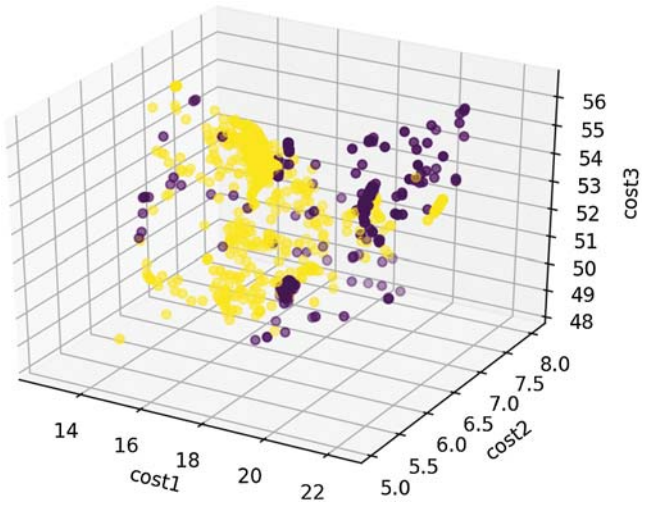

Figure 5: Reconstruction error distribution under NSL-KDD with two hidden layers of the encoder and the decoder

We can then acquire $c_i = [h_i, e_i, \rho_i, m_i]$, where h_i denotes compressed feature, and m_i is MRE.

We build a DEN to calculate the density of sample, which can receive a new feature c from DAE, and generate a K-dimensional distribution named $\hat{\pi}$. K denotes the number of components of GMM; note that to calculate the energy value, we feed $\hat{\pi}$ into the GMM. In addition, we adopt GELU [19] as the activation function for DAE and DEN; it can be defined as

$$\mathrm{GELU}(x) = x\mathrm{P}(X \leq x) = x\Phi(x) \tag{4}$$

As x decreases, the probability of it being classified becomes 0. As x increases, the activation value also increases. This type of activation method not only retains the probability, but also retains the dependence on the input.

3.4.3 GMM

GMM is a linear combination of multiple Gaussian functions. Assuming the density of anomalies to be lower than that of normal data, we train the energy threshold of normal data, and the input data will be classified as abnormal if energy exceeds the threshold. Fig. 6 shows energy distribution between normal data and anomalies under DDoS.

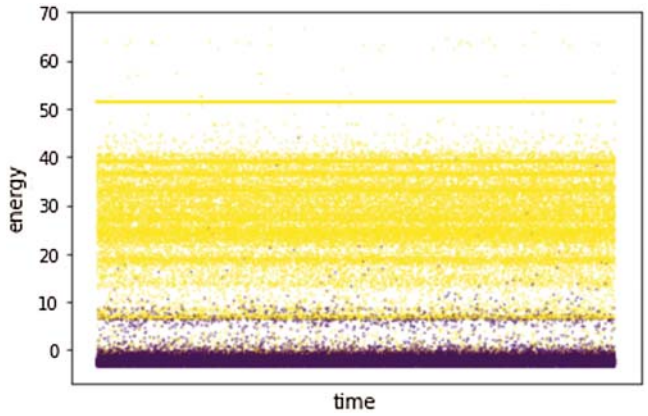

Figure 6: Energy distribution of normal data and anomalies under DDoS attack, where yellow points are anomalies, while purple points denote normal data

For input c, the probability prediction of components generated by GMM is expressed as

$$\hat{\pi} = softmax(DEN(c; \omega)) \tag{5}$$

where ω is the parameter of DEN. Thus, we define the probability of c as

$$p(c) = \sum_{k=1}^{K} \hat{\varphi}_k \cdot N(c \mid \mu_k, \Sigma_k) \tag{6}$$

where $N(c \mid \mu_k, \Sigma_k)$ is the k-th component of GMM, $\hat{\varphi}_k$ is mixture coefficient with limitation

$$\sum_{k=1}^{K} \hat{\varphi}_k = 1 \tag{7}$$

Given m samples from DAE, we can get the average probability, mean value and covariance of each component using DEN and GMM as

$$\hat{\varphi}_k = \sum_{i=1}^{m} \frac{\hat{\pi}_{ik}}{m}, \quad \hat{\mu}_k = \frac{\sum_{i=1}^{m} \hat{\pi}_{ik} \cdot c_i}{\hat{\pi}_{ik}}, \quad \hat{\Sigma}_k = \frac{\sum_{i=1}^{m} \hat{\pi}_{ik}(c_i - \hat{\mu}_k)(c_i - \hat{\mu}_k)^T}{\sum_{i=1}^{m} \hat{\pi}_{ik}} \tag{8}$$

Furthermore, we can define sample energy as

$$\mathrm{E}(z) = -\log\left(\sum_{k=1}^{K} \hat{\varphi}_k \frac{\exp(-\frac{1}{2}(c - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1}(c - \hat{\mu}_k))}{\sqrt{|2\pi\, \hat{\Sigma}_k|}}\right) \tag{9}$$

Finally, we get the objective function as

$$\mathrm{L}(\theta, \gamma, \omega, \hat{\mu}, \hat{\varphi}, \hat{\Sigma}) = L_1 + \frac{\lambda_1}{m} \sum_{i=1}^{m} E(z_i) + \lambda_2 P(\hat{\Sigma}) \tag{10}$$

where θ, γ, and ω denote parameters of DAE, DEN, and GMM, $P\left(\hat{\Sigma}\right) = \sum_{k=1}^{K} \sum_{i=1}^{d} \hat{\Sigma}_{kjj}^{-1}$, and d is the dimension of compressed features. Finally, the training object is to be minimized using Eq. (10).

In order to select a suitable threshold, we adopt φ as an energy parameter with the initial value $-\text{Inf}$, which represents the max energy during the process of training. We then set the adjustment factor $\epsilon \in (0, \infty)$. Hence, the threshold $T = \epsilon\varphi$. The training procedure of the anomaly detector is presented in Algorithm 2, and the testing process is shown in Algorithm 3.

Algorithm 1: The computing procedure of MRE

Input: Training data set S, epochs N, layer of DAE L, θ_i, γ_i for encoder and decoder of DAE
Output: MRE set M
Set $M = \varnothing$
for epoch from 1 to N
  if epoch == N:
  for $x_i \in$ S **do**
    for j from 2 to L
      $x_{ij} = enc(x_{i(j-1)};\ \theta_i),\ y_{ij} = dec(x_{i[\frac{L-1}{2}-j]};\ \gamma_i)$
      $e_i = \sqrt{\frac{\sum_{j=1}^{d}(x_{ij}, y_{ij})^2}{d}}$
      Append e_i into M

4 Evaluation

In this section, we evaluate the proposed MEAEDE using three classical intrusion detection datasets: KDDCUP99 10% [20], NSL-KDD [21], and CIC-IDS-2017 [22]. At the same time, our network model is compared with other autoencoder-based approaches, including DAE and DAGMM. In our experiment, reconstruction error is used only in DAE to derive threshold.

4.1 Datasets

1) KDDCUP99 contains 41 dimensions, of which 34 are continuous and 7 are categorical attributes. Each data point in KDDCUP is labeled as either normal or a specific attack group for big categories, including DoS, user to local (U2L), remote to local (R2L) and probe.

2) NSL-KDD has the same features as KDDCUP99, except that redundant data is filtered. The dataset consists of two files: KDDTrain+ and KDDTest+.

3) CIC-IDS-2017 was generated by CIC Laboratory of Canada, and the types of attacks collected are very extensive. From Monday to Friday, the network data was captured for five days, with nearly 80 dimensions. The dataset consists of eight files, including Monday benign sample, Tuesday BForce, SFTP and SSH sample, Wednesday DoS and Heartbleed Attacks, Slowloris, Slowhttptest, Hulk and GoldenEye, Thursday Web and Infiltration Attacks, Web BForce, XSS and Sql Inject, Infiltration Dropbox Download and Cool disk, Friday DDoS LOIT, Botnet ARES, and PortScans (sS, sT, sF, sX, sN, sP, sV, sU, sO, sA, sW, sR, sL and B) sample.

4.2 Metrics

To evaluate our model, we use the following three common information retrieval evaluation metrics:

1) **Precision (Pr):** It is the ratio of correctly classified attack flows (TP) to all the classified flows (TP + FP).

2) **Recall (Rc):** It is the ratio of correctly classified attack flows (TP) to all generated flows (TP + FN).

3) **F-Measure (F1):** It is a harmonic combination of the P_r and R_c into a single measure. These metrics can be defined as

$$P_r = \frac{TP}{TP + FP} \tag{11}$$

$$R_c = \frac{TP}{TP + FN} \tag{12}$$

$$\mathrm{F1} = \frac{2}{\frac{1}{P_r} + \frac{1}{R_c}} \tag{13}$$

Algorithm 2: The procedure for training the anomaly detector

Input: Training data set S, epochs N
Output: MEAEDE model, φ
θ, γ ← parameters of encoder and decoder of *DAE*
ω ← parameters of *DEN*
for epoch from 1 to N
for $x_i \in$ S **do** Minimize L(θ, γ, ω, $\hat{\mu}$, $\hat{\varphi}$, $\hat{\Sigma}$)
$h_i = enc(x_i; \theta)$, $y_i = dec(h_i; \gamma)$
$\rho_i = \frac{cov(x_i, y_i)}{\sigma_{x_i}.\sigma_{y_i}}$, $e_i = \sqrt{\frac{\sum_{j=1}^{d}(x_i^{(j)} - y_i^{(j)})^2}{d}}$
compute m_i as Algorithm 1
$c_i \leftarrow [h_i, e_i, \rho_i, m_i]$
$\hat{\pi} = DEN(c_i; \omega)$
$\hat{\mu}$, $\hat{\varphi}$, $\hat{\Sigma} \leftarrow$ update parameters for GMM as Eq. (8)
if $e_i > \varphi$: $\varphi \leftarrow e_i$
θ, γ, ω ← update parameters using SGD

4.3 Parameter Discussion

In order to optimally fit these datasets, we use slightly different parameters for data of different dimensions. Owing to the existence of categorical values, KDDCUP99 10% and NSL-KDD use unique one-hot encoding, and their inputs have 118 and 122 dimensions, respectively. After discarding the "flow packet number/s" and "flow byte number/s" features, 77 features are selected from CIC-IDS-2017. We find in the exploration that with four hidden layers in the encoder, the DAE yields the expected result when the number of layer increases, and the effect

does not rise significantly or even decreases; thus, the parameters of different networks set as follows:

KDDCUP99(NSL-KDD):

DAE:FC (118(122), 60, GELU)-FC (60, 30, GELU)-FC (30, 10, GELU)-FC (10, 3, none)-FC (3, 10, GELU)-FC (10, 30, GELU)-FC (30, 60, GELU)-FC (60, 118(122), none).

DEN: FC (5,10, GELU)-Drop (0.5)-FC (10, 5, softmax).

CIC-IDS-2017:

DAE:FC (77, 40, GELU)-FC (40, 20, GELU)-FC (20, 10, GELU)-FC (10, 3, none)-FC (3, 10, GELU)-FC (10, 20, GELU)-FC (20, 40, GELU)-FC (40, 77, none).

DEN: FC (5,10, GELU)-Drop (0.5)-FC (10, 5, softmax).

We uniformly set $\lambda_1 = 0.1$, and $\lambda_2 = 0.0001$, and the learning rate as 0.0001 in every model. As can be seen in Eq. (10), the objective loss function of MEAEDE has three components: L_1 represents reconstruction error from the DAE, the second component represents energy function, and the third represents the penalty function for covariance matrices. A large value of λ_1 may impact L_1 and make Eq. (10) getting unimportant, thus the expected representation of input data cannot be derived. When λ_1 has a small value, the GMM will not be trained well for the small weight of the estimated network. When λ_2 has a large value, the covariance of the GMM has a large value, which is not desirable as most samples have large values. When λ_2 has a small value, the regularization is weak in countering the singularity effect. In our experiment, we find that setting the values of the said hyper-parameters as 1, 0.1, and 0.0001 leads to a remarkable performance across the three datasets.

Algorithm 3: The procedure of executing anomaly detector

Input: Testing data set S, ϵ
Output: Anomalies
Set $T = \epsilon\varphi$
for x_i **in** data set S **do**
$h_i = enc(x_i; \theta)$, $y_i = dec(h_i; \gamma)$
$c_i \leftarrow [h_i, e_i, \rho_i, m_i]$
$\hat{\pi} = DEN(c_i; \omega)$
$e_i = E\left(\boldsymbol{c_i}; \hat{\mu}, \hat{\varphi}, \hat{\Sigma}\right)$
If $e_i > T$:
Alert

4.4 Detection Results

For KDDCUP99 and NSL-KDD, we calculated the average of each indicator. The experimental results are encapsulated in Tabs. 2 and 3. The results show that DEAEME outperforms other autoencoder-based models. With a simple threshold selection, it is hard for the DAE model to be compressed with other two models because of information loss in the hidden layer. The DAGMM uses cosine similarity and Euclidean distance of input and output data of DAE, and thus it outperforms DAE. While MEAEDE derives MRE, coefficient correlation and compressed features, it has a sharp increase in comparison with DAGMM, so MRE has a powerful effect to compensate for the loss of information in the hidden layer.

It is worth mentioning that the threshold of each model is set according to their optimal results. For DAE, we set threshold T as maximum reconstruction error multiplied by a value between 0 and 1, and for DAGMM and MEAEDE, we select threshold parameters according to different datasets to adapt to optimal results.

Table 2: Detection results on KDDCUP99 10% dataset

	DAE	DAGMM	MEAEDE
Precision	0.770	0.929	0.993
Recall	0.840	0.944	0.996
F1-score	0.770	0.936	0.995

Table 3: Detection results on NSL-KDD dataset

	DAE	DAGMM	MEAEDE
Precision	0.846	0.898	0.939
Recall	0.833	0.895	0.935
F1-score	0.839	0.896	0.937

In CIC-IDS-2017, we select six files from the entire datasets. A 1:1 training and testing ratio was adopted as data is imbalanced in the other two files. Unfortunately, due to the existence of infinite data, we remove the "flow bytes/s" and "flow packets/s" attributes, and finally 77 features of each data point are selected in the samples, and the results are shown in Tabs. 4–6. Upon encountering Friday's DDoS attack, MEAEDE and DAGMM perform equally well, while DAGMM has an inadequate performance when encountering Friday-PortScan and web-attacks. On the contrary, MEAEDE performs more satisfactorily most of the time. It is worth noting that DAE is stable for all datasets, and the average value of the three metrics is better than DAGMM, but the recognition accuracy is not commensurate to the expectations.

Table 4: MEAEDE detection results on CIC-IDS2017 dataset

	Precision	Recall	F1-score
Tuesday	0.977	0.958	0.967
Thursday-WebAttacks	0.987	0.990	0.988
Thursday-Infiltration	1.000	0.990	0.995
Wednesday	0.999	0.472	0.641
Friday-PortScan	0.643	0.882	0.744
Friday-DDoS	0.991	0.963	0.977

To evaluate the training time of MEAEDE, we compute the average time consumed during training of 50 epochs. Tab. 7 presents details of each dataset. Evidently, MEAEDE has an average training time of about 200 s in every dataset, which can be deployed and reset efficiently in a number of applications.

Table 5: DAGMM detection results on CIC-IDS2017 dataset

	Precision	Recall	F1-score
Tuesday	0.980	0.040	0.078
Thursday-WebAttacks	0.476	0.484	0.478
Thursday-Infiltration	0.983	0.700	0.818
Wednesday	0.527	0.582	0.554
Friday-PortScan	0.198	0.170	0.183
Friday-DDoS	0.999	0.901	0.948

Table 6: DAE detection results on CIC-IDS2017 dataset

	Precision	Recall	F1-score
Tuesday	0.736	0.648	0.705
Thursday-WebAttacks	0.744	0.797	0.769
Thursday-Infiltration	0.714	0.994	0.831
Wednesday	0.539	0.408	0.458
Friday-PortScan	0.363	0.423	0.388
Friday-DDoS	0.745	0.852	0.795

Table 7: Average time consumed in training datasets

	Training time (s)	Dimension
KDDCUP99 10%	104.806	41
NSL-KDD	201.727	41
CIC-IDS-2017	172.808	77

We also evaluate our model under actual network attacks using our feature extractor module and employ several types of attacks such as DDoS, port scan, and injection in five LAN hosts using eight different computers to launch attacks (including 30% of normal data and 70% of attacks). We then capture packets in the network card with an average feature extraction time of 14 s for every 100 k ".pcap" file. In order to correctly label the captured data, we use specific hosts to only implement attacks or normal requests and record them. Tab. 8 presents the results depicting the performance of our model. Unfortunately, the performance is not satisfactory on CIC-IDS-2017 dataset. In this dataset, the data ratio we selected is 1:1, while in the data we collected, the ratio is obviously different, which is the reason why the detection efficiency is not satisfactory. So the ratio of normal data and anomalies can disturb the accuracy of MEAEDE.

Table 8: Detection results in actual network environment with $\in$ of 0.8

	DDoS	Port scan	Injection
Precision	0.957	0.937	0.836
Recall	0.948	0.939	0.827
F1-score	0.952	0.938	0.832

5 Conclusion

In this paper, we propose a lightweight anomaly detection framework, which includes network packet capturing, efficient processing, and a feature extraction module that has a good expressive ability to derive network packets and flows. We then transform the extracted features to time-related features for Anomaly Detector module. This module combines MRE, the correlation coefficient of input and output, and compressed features from DAE to form a new feature vector, which is fed into a DEN to transform to a K-component vector and then fed to GMM to get an energy value. The system detects anomalies if the computed value exceeds a pre determined threshold. We train and test our model under several intrusion detection datasets and compare three common information retrieval evaluation metrics with DAE and DAGMM. With improvement in feature extraction, the proposed model outperforms other autoencoder-based models and can further improve the performance of anomaly detection. Furthermore, after training only with normal data, it can identify various unknown attacks such as DDoS, web-attack, and port-scan with desirable performance.

Notwithstanding its adequate performance, the input data distribution has a few limitations; it is only suitable to network-based packets and flows. In addition, the ratio of normal data and anomalies can disturb the performance of MEAEDE. In the future, it is necessary to improve the feature extraction ability and make the model more robust. It is also imperative to study better methods for threshold selection of GMM, which can efficiently identify more attack types to adapt to the complex network environment.

Acknowledgement: We would like to thank TopEdit (www.topeditsci.com) for the English language editing of this manuscript.

Funding Statement: This work is supported by the Introducing Program of Dongguan for Leading Talents in Innovation and Entrepreneur (Dongren Han [2018], No. 738).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Chalapathy, R., Chawla, S. (2019). Deep learning for anomaly detection: A survey. *Computing Research Repository*, 1–50. arXiv: 1901.03407.
2. Wang, X., Du, Y., Lin, S., Cui, P., Shen, Y. et al. (2020). AdVAE: A self-adversarial variational autoencoder with Gaussian anomaly prior knowledge for anomaly detection. *Knowledge-Based Systems, 190(17),* 105187. DOI 10.1016/j.knosys.2019.105187.
3. Wang, L. (2017). Big data in intrusion detection systems and intrusion prevention systems. *Journal of Computer Networks, 4(1),* 48–55. DOI 10.12691/jcn-4-1-5.
4. Petersen, R. (2015). *Data mining for network intrusion detection: A comparison of data mining algorithms and an analysis of relevant features for detecting cyber-attacks* (Dissertation). http://urn.kb.se/resolve?urn=urn:nbn:se:miun:diva-28002.
5. Ilonen, J., Paalanen, P., Kamarainen, J., Kalviainen, H. (2006). Gaussian mixture pdf in one-class classification: Computing and utilizing confidence values. *18th International Conference on Pattern Recognition*, vol. 2, pp. 577–580. Hong Kong, China. DOI 10.1109/ICPR.2006.595.
6. Yeung, D., Chow, C. (2002). Parzen-window network intrusion detectors. *16th International Conference on IEEE*, vol. 4, pp. 385–388. Quebec City, QC, Canada, IEEE. DOI 10.1109/ICPR.2002.1047476.
7. Lakhina, A., Crovella, M., Diot, C. (2004). Diagnosing network-wide traffic anomalies. *ACM SIGCOMM Computer Communication Review, 34(4),* 219–230. DOI 10.1145/1030194.1015492.

8. Ringberg, H., Soule, A., Rexford, J., Diot, C. (2007). Sensitivity of PCA for traffic anomaly detection. *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pp. 109–120. New York, NY, USA.
9. Yang, B., Fu, X., Sidiropoulos, N. D., Hong, M. (2017). Towards k-means-friendly spaces: Simultaneous deep learning and clustering. *The 34th International Conference on Machine Learning*, vol. 10, pp. 1–14. Sydney, Australia. arXiv: 1610.04794v2.
10. Seo, E., Song, H. M., Kim, H. K. (2018). Gids: Gan based intrusion detection system for in-vehicle network. *16th Annual Conference on Privacy, Security and Trust*, vol. 7, pp. 1–6. Belfast, Northern Ireland, UK. DOI 10.1109/PST.2018.8514157.
11. Di Mattia, F., Galeone, P., De Simoni, M., Ghelfi, E. (2019). A survey on GANs for anomaly detection. *Computing Research Repository, 6(1),* 1–16. arXiv: 1906.11632v1.
12. Nguyen, Q. P., Lim, K. W., Divakaran, D. M., Low, K. H., Chan, M. C. (2019). Gee: A gradient-based explainable variational autoencoder for network anomaly detection. *2019 IEEE Conference on Communications and Network Security*, vol. 6, pp. 91–99. Washington DC, DC, USA. DOI 10.1109/CNS.2019.8802833.
13. Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. *Network and Distributed Systems Security Symposium*, vol. 1, pp. 1–15. San Diego, CA, USA. DOI 10.14722/ndss.2018.23204.
14. Zhou, C., Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 665–674. DOI 10.1145/3097983.
15. Zhai, S., Cheng, Y., Lu, W., Zhang, Z. (2016). Deep structured energy based models for anomaly detection. *International Conference on Machine Learning,* vol. 48, pp. 1100–1109. New York, USA. arXiv: 1605.07717v2.
16. Van Loi, C., Miguel, N., James, M., Julia, H., Emma, H. et al. (2016). A hybrid autoencoder and density estimation model for anomaly detection. *International Conference on Parallel Problem Solving from Nature*, *8(1),* 717–726. Edinburgh, Scotland, UK, Springer. DOI 10.1007/978-3-319-45823-6_67.
17. Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C. et al. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. *International Conference on Learning Representations*, pp. 97–106. New York, NY, USA. DOI 10.1145/3269206.3271698.
18. Drapper Gil, G., Lashkari, A. H., Mamun, M., Ghorbani, A. A. (2016). Characterization of encrypted and VPN traffic using time-related features. *The Proceedings of the 2nd International Conference on Information Systems Security and Privacy*, vol. 2, pp. 407–414. Rome, Italy. DOI 10.5220/0005740704070414.
19. Hendrycks, D., Gimpel, K. (2016). Gaussian Error Linear Units (GELUS). arXiv: 1606.08415v3.
20. Siddiqui, M. K., Naahid, S. (2013). Analysis of KDD CUP 99 dataset using clustering based data mining. *International Journal of Database Theory and Application, 6(5),* 23–34. DOI 10.14257/ijdta.2013.6.5.03.
21. Dhanabal, L., Shantharajah, S. P. (2015). A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *International Journal of Advanced Research in Computer and Communication Engineering, 4(6),* 446–452. DOI 10.17148/IJARCCE.2015.4696.
22. Sharafaldin, I., Lashkari, A., Ghorbani, A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *International Conference on Information Systems Security & Privacy*, vol. 1, pp. 108–166. Funchal, Portugal. DOI 10.5220/0006639801080116.