Twin Variational Auto-Encoder for Representation Learning in IoT Intrusion Detection

Phai Vu Dinh[1,2], Nguyen Quang Uy[3], Diep N. Nguyen[1], Dinh Thai Hoang[1], Son Pham Bao[1,2], Eryk Dutkiewicz[1]
[1] School of Electrical and Data Engineering, University of Technology Sydney, Australia
[2] JTIRC, VNU University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam
[3] Le Quy Don Technical University, Hanoi, Vietnam

***Abstract*—Intrusion detection systems (IDSs) play a pivotal role in defending IoT systems. However, developing a robust and efficient IDS is challenging due to the rapid and continuing evolving of various forms of cyber-attacks as well as a massive number of low-end IoT devices. In this paper, we introduce a novel deep learning architecture based on auto-encoders that allows to develop a robust intrusion detection system. Specifically, we propose a novel neural network architecture called Twin Variational Auto-Encoder (TVAE) for representation learning. TVAE includes a variational Auto-Encoder (VAE) and an Auto-Encoder (AE) that share a common stage where the decoder of the VAE is used as the encoder of the AE. The TVAE is trained in an unsupervised manner to effectively transform the original representation of data at the input of the VAE into a new representation at the output of the AE. In the new representation space, the difference between normal and attack data is more distinguishable. A variant of TVAE, namely Twin Sparse Variational Auto-Encoder (TSVAE) is also introduced by imposing a sparsity constraint on the representation units. The effectiveness of TVAE and TSVAE is evaluated using popular IDS and IoT botnet datasets. The simulation results show that the accuracy of TVAE and TSVAE can achieve the best results on six datasets, which is higher than those of state-of-the-art AE and VAE variants. We also investigate various characteristics of TVAE in the latent space as well as in the data extraction process. Besides applications on the IoT IDS, TVAE can also be applicable to all conventional network IDSs.**

***Index Terms*—Deep Learning, representation learning, TVAE, TSVAE, VAE, AE, IoT, IDS.**

I. Introduction

The Intrusions Detection Systems (IDSs) can be categorized into host-based IDS and network-based IDS [1]. The former monitors and analyzes the data, collected from a specific host. The data collection may consist of a vector of system calls, unexpected users' activities, and system logs [2]. The latter focuses on the incoming network traffic to detect malicious connections which many include DoS attacks, port scans, malicious emails, phishing scams [3]. However, developing a robust and efficient IDS is challenging due to the rapid and continuing evolving of various forms of cyber-attacks as well as a massive number of low-end IoT devices. Moreover, IoT devices are often limited in computing and communications, energy resources. That makes more vulnerable to cyber-threats, especially for low-end devices (e.g., sensors and actuators).

Over the last decade, deep neural networks or deep learning (DL) have been emerging as promising engines in IDSs. Amongst DL techniques, auto-encoders (AEs) are widely used for representation learning or feature learning [4]. AEs aim to learn the best parameters required to reconstruct its input at the output. The bottleneck layer of AEs is often called as the latent vector or the latent space. The latent vector is very helpful in IDSs since it transforms the original representation into new representation where attack data is often more distinguishable from the normal data. Variational auto-encoder (VAE), a variant of AE, is one of the most prevalent generative model frameworks [5]. However, when applying VAE to representation learning, the effectiveness of VAE is often not as good as that of AE [6]. The reason is that the latent representation of VAE is sarcastically sampled from a Gaussian distribution thus this representation is not stable to be used as the input to a classifier. In this paper, we propose a novel deep learning architecture based on AE and VAE, referred to as Twin Variational Auto-Encoder (TVAE), for representation learning. TVAE includes a VAE and an AE that share a common stage where the decoder of the VAE is used as the encoder of the AE. The objective of the AE in TAVE is to reconstruct the latent representation of the VAE at its output. Thus, the output of the AE in TAVE can be used as a new representation for the original data. The new representation is then used as the input to classification algorithms to detect potential attacks. Moreover, a variant of TVAE, namely Twin Sparse Variational Auto-Encoder (TSVAE) is introduced by imposing a sparsity constraint on the representation units. The experiments performed on IDS datasets named NSLKDD [7] and five IoT botnet datasets [8] show the superior performance of the TVAE and TSVAE over state-of-art AE and VAE variants in IDSs. Our main contributions are summarized as follows:

- We propose a novel neural network architecture, i.e., Twin Variational Auto-Encoder (TVAE) and a variant of TVAE called Twin Sparse Variational Auto-Encoder (TSVAE) for IDSs. TVAE and TSVAE provide an effective technique for representation learning that makes attack data more distinguishable from normal data (in the latent domain). This is thanks to stochastic model in TVAE and TSVAE in the training process while they use a non-probabilistic model in the testing process.
- We conduct intensive experiments using IDS datasets and five IoT botnet datasets to evaluate the performance of TVAE and TSVAE. The experimental results show that the TVAE and TSVAE models can support the classifiers to perform classification tasks much better than those of VAEs and AE. For example, using the representation of the TVAE and TSVAE, the tested classifiers can achieve the best results on six datasets.
- We numerically study various characteristics of the latent representation of TVAE and TSVAE in the latent space as well as in the data extraction process to justify the superior performance of TVAE and TSVAE over the AE

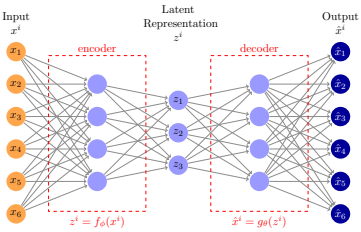

Fig. 1: Autoencoder

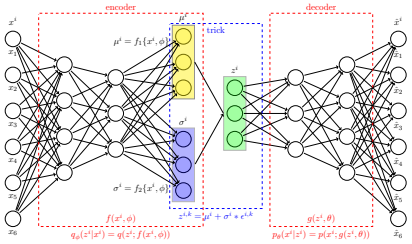

Fig. 2: Variational Autoencoder

and VAE variants.

II. Related Work

This section brief reviews the previous research on representation learning for IDSs based on AEs and VAEs.

AE is a well-known DL architecture for representation learning in recent years. The authors in [9], [10] used AE as a non-linear transformation to discover unknown data structure compared to those of Principal Component Analysis (PCA). Shone et al in [11] proposed non-symmetric deep-auto encoder which only uses an encoder for both encoded and decoded tasks. This architecture can extract data from the latent space to enhance the performance of Random Forest (RF). Recently, the authors in [12] proposed Convolutional Sparse Auto-Encoder (CSAE), which leverages the structure of the convolutional AE and incorporates the max-pooling to heuristically sparsify the feature maps for representation learning. VAE is as common as Generative Adversarial Networks (GAN) when commonly using as a generative model [13]. The VAE model generates adversarial examples which are used to improve performance of deep-learning models in handling imbalanced datasets [14]. However, VAE is hardly used for representation learning as its latent representation is stochastically sampled from a Gaussian distribution, and thus it is unstable when using as the input to classification algorithms [15]. To facilitate the robust learning of disentangled representations in VAE, the authors in [16] and [17] added an extra hyperparameter β to the VAE and a quantization term in the bottleneck of VAE, respectively.

Note that the latent space of AE can be extremely irregular. The reason is that the close points in latent space of AE can be mixed together. VAE is seldom used for representation learning since it is unstable due to sampling process to generate the latent space. In this paper, we propose a novel deep learning architecture using unsupervised learning based on the architectures of VAE and AE, called Twin Variational Auto-Encoder (TVAE) for representation learning. The TVAE allows to reconstruct the latent representation and uses this reconstructed representation as a new representation for classifiers. The TVAE is then trained with a probabilistic function, whilst its extraction process for testing is non-probabilistic. For that, TVAE is more stable than those of the previous stochastic models, i.e., VAE, β-VAE and VQVAE. In addition, to discover interesting structure in the representation, a variant of TVAE, namely Twin Sparse Variational Auto-Encoder (TSVAE) is also introduced by imposing a sparsity constraint on the representation units.

III. Background

This section presents AE and VAE in details. They are the two key components of the proposed models in the next section.

A. Auto-Encoder (AE)

An Auto-Encoder is a neural network trained by unsupervised learning to learn the best parameters required to reconstruct its output as close to its input as possible [4]. An AE has two parts, an encoder and a decoder, as illustrated in Fig. 1. Let W, b, $W^{'}$, and $b^{'}$ be the weight matrix and biases of the encoder and decoder, respectively. $\phi = (W, b)$ is the parameter sets of the encoder function $z = f_\phi(x)$, while $\theta = (W^{'}, b^{'})$ is the parameter sets of the decoder function $\hat{x} = g_\theta(z)$. We use a dataset $x = \{x^1, x^2, .., x^n\}$ to train the AE model which has its loss function as follows:

$$\ell_{\text{AE}}(x, \phi, \theta) = \frac{1}{n}\sum_{i=1}^{n}\left(x^i - \hat{x}^i\right)^2, \tag{1}$$

where $\hat{x}^i$ is the output of the AE corresponding with the input x^i. The latent representation of the encoder is usually referred to as a bottleneck which is used as an input for classifiers.

B. Variational Auto-Encoder (VAE)

A Variational Auto-Encoder (Fig. 2) is another version of AE [18]. The difference between a VAE and an AE is that the latent representation (z) of the VAE is sampled from a Gaussian distribution parameterized by mean (μ) and standard devidation (σ). We define the encoder by $q_\phi(z^i|x^i)$ and the decoder by $p_\theta(x^i|z^i)$, then the loss function of a VAE for a datapoint x^i includes two terms as follows:

$$\ell_{\text{VAE}} = D_{\text{KL}}\big(q_\phi(z^i|x^i)|p_\theta(z^i)\big) - E_{q_\phi(z^i|x^i)}[\log p_\theta(x^i|z^i)]. \tag{2}$$

The first term in (2) is the KL divergence between the approximation posterior ($q_\phi(z^i|x^i)$) and the prior distribution ($p_\theta(z^i)$). This divergence measures how close the posterior is to the prior. The second term $-E_{q_\phi(z^i|x^i)}[\log p_\theta(x^i|z^i)]$ is reconstruction error (RE) of the VAE. This term forces the decoder to learn to reconstruct the input data.

Since, it is tricky to generate samples z^i from $q_\phi(z^i|x^i)$, a reparameterization trick in (3) is used to overcome the high variance problem when applying the Monte Carlo method [19]. In particular, instead of using a random variable from the original distribution, the reparameterization trick uses a random variable z^i from a standard normal distribution as follows:

$$z^{i,k} = \mu^i + \sigma^i * \epsilon^{i,k}; \quad \epsilon^{i,k} \approx N(0, I), \tag{3}$$

where μ^i and σ^i are mean and standard deviation of the Gaussian distribution of an individual latent variable z^i, respectively. The values of μ^i and σ^i are obtained via the encoder by using functions $\mu^i = f_1(x^i, \phi)$ and $\sigma^i = f_2(x^i, \phi)$ as illustrated in Fig. 2, respectively.

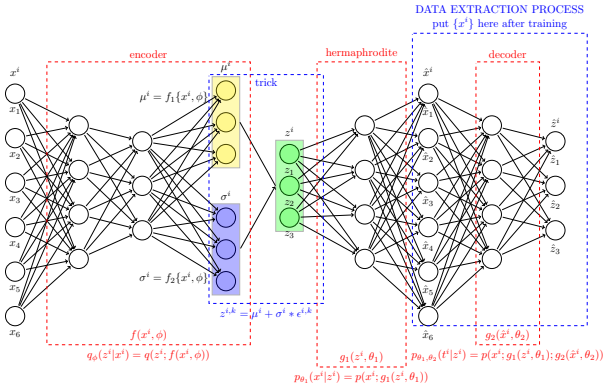

Fig. 3: Twin Variational Autoencoder Architecture

IV. TWIN VARIATIONAL NEURAL NETWORK

In this section, we present our novel deep learning neural network architecture: Twin Variational Auto-Encoder (TVAE).

A. Architecture of TVAE

The architecture of TVAE is shown in Fig. 3. The TVAE has three components: an encoder, a hermaphrodite and a decoder. First, the encoder attempts to map an input sample x^i to two hidden vectors μ^i and σ^i by using functions $f_1(x^i, \phi)$ and $f_2(x^i, \phi)$, respectively. A feasible reparameterization trick is applied by using a deterministic function $z^{i,k} = \mu^i + \sigma^i * \epsilon^{i,k}$, which helps to transfer the input x^i into a new latent space z^i. Second, the hermaphrodite performed by function $g_1(z^i, \theta_1)$ has two roles. On the one hand, the hermaphrodite is the decoder of the VAE in TVAE. On the other hand, the hermaphrodite is the encoder of the AE. The third component in TVAE is the decoder fed by $\hat{x}^i$ that aims to reproduce z^i by performing the function $g_2(\hat{x}^i, \theta_2)$. After training the TVAE model with dataset $\{x^i\}_{i=1}^n$, we can put a data point x^i into the decoder to obtain a new representation $\hat{z}^i$. That has the same dimensionality as that of the latent variable z^i.

B. TVAE Loss Function

The loss function ℓ_{TVAE} includes three terms in 4.

$$\begin{aligned} \ell_{\text{TVAE}} &= D_{\text{KL}}[q(z|x)||p(z)] \\ &- E_{q(z|x)} \log p(x|z) - E_{q(z|t)} \log p(t|z). \end{aligned} \tag{4}$$

The first term in 4 is the KL divergence between the approximation posterior $q(z|x)$ and the prior distribution $p(z)$. The second term is the reconstruction error of the VAE and the third term is the reconstruction error of the AE. Assume that both target variables $\hat{x}^i$ and $t^i = \hat{z}^i$ are given by deterministic functions with additive Gaussian noise. The values of $-E_{q(z|x)} \log p(x|z)$ and $-E_{q(z|t)} \log p(t|z)$ are thus scaled with mean-squared-error function, as follows:

$$\begin{aligned} -E_{q(z|x)} \log p(x|z) &= \frac{1}{n} \sum_{i=1}^{n} (x^i - \hat{x}^i)^2, \\ -E_{q(z|t)} \log p(t|z) &= \frac{1}{n} \sum_{i=1}^{n} (z^i - \hat{z}^i)^2. \end{aligned} \tag{5}$$

The KL divergence $D_{\text{KL}}[q(z|x)||p(z)]$ between the approximation posterior $q(z^i|x^i)$ and the prior distribution $p(z^i)$ of the latent variable z^i can be measured by assuming both of them being normal Gaussian distribution. Thus, the ℓ_{TVAE} can be written as:

$$\begin{aligned} \ell_{\text{TVAE}} = \frac{1}{n} \sum_{i=1}^{n} \big[&(x^i - \hat{x}^i)^2 + \beta_1 (z^i - \hat{z}^i)^2 \\ &+ \beta_2 \sum_{j=1}^{J} \big(-1 - \log(\sigma_j^i)^2 + (\mu_j^i)^2 + (\sigma_j^i)^2 \big) \big], \end{aligned} \tag{6}$$

where β_1 and β_2 are hyper-parameter settings to control the trade-off amongst three terms in (6), J denotes the dimension of z^i.

Algorithm 1: Training the TVAE model

Input: Training dataset $x = \{x^i\}_{i=1}^n$, R_0, ϕ, θ_1, θ_2, n_epoch
Output: TVAE model is trained.
for $epoch = 1$ *to* n_epoch **do**
  foreach x^i *in* x **do**
    $z^i = \mu^i + \sigma^i \epsilon^{i,k}$
    $\hat{x}^i = g_1(z^i, \theta_1)$
    if $epoch \ \% \ R_0 == 0$ **then**
      $g_2(x^i, \theta_2)$
    end
    else
      $g_2(\hat{x}^i, \theta_2)$
    end
    Update $(\phi, \theta_1, \theta_2)$ by utilizing gradient descent.
  end
end

C. TVAE Training Process

The objective of the decoder in TVAE is to transform the input data (x^i) to a new representation $\hat{z}^i$. However, in the architecture of TVAE, the value of $\hat{z}^i$ is calculated with the input is $\hat{x}^i$. Thus, to construct a robust representation $\hat{z}^i$, we propose the training process of TVAE as in Algorithm 1 in which both x^i and $\hat{x}^i$ are used to construct both x^i and $\hat{x}^i$ with a ratio R_0.

V. TWIN SPARSE VARIATIONAL AUTO-ENCODER (TSVAE)

TSVAE aims to improve the representation of TVAE by imposing a sparsity constraint on its representation units like [12]. Let $a_j^f(\hat{z})$ be the activation of this representation unit and $\rho_j = \frac{1}{n} \sum_{i=1}^{n} (a_j^f(\hat{z}_j^i))$ be the average activation of representation unit j^{th} over the training datasets. To enforce the constraint, we set $\rho_j = \rho$ where ρ is a sparsity parameter. The ρ is set at small value close to 0, i.e., $\rho = 0.05$. The difference between TSVAE and TVAE is a penalty term

$\sum_{j=1}^{|\hat{z}|} KL(\rho||\rho_j)$ added to the loss function, as follows:

$$\ell_{\text{TSVAE}} = \frac{1}{n}\sum_{i=1}^{n}\big[(x^i-\hat{x}^i)^2 + \beta_1(z^i-\hat{z}^i)^2 + \beta_2\sum_{j=1}^{J}\big(-1-\log(\sigma_j^i)^2+(\mu_j^i)^2+(\sigma_j^i)^2\big)\big] + \beta_3\sum_{j=1}^{|\hat{z}|}KL(\rho||\rho_j), \quad (7)$$

where $|\hat{z}|$ is the number of neurons in the representation. $KL(\rho||\rho_j)$ is the Kullback-Leibler (KL) divergence between a Bernoulli (Gaussian) random variable with mean ρ and a Bernoulli (Gaussian) random variable with mean ρ_j.

VI. Experiment Results

A. Performance Metrics

We use two popular metrics, i.e., *Accuracy* and *F-score* and False Alarm Rate (*FAR*) [20], to evaluate the performance the performance of TVAE and TSVAE in IoT IDSs. The first metrics is $\textit{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$ where *TP*, *TN*, *FP*, *FN* are True Positive, True Negative, False Positive and False Negative, respectively. The second metric is the *F-score* that is the harmonic mean of the *Precision* and the *Recall*. *Precision* is the proportion of positive prediction that was actually correct, and *Recall* is the proportion of actual positives was identified correctly. Based on that, *F-score* is calculated by $\textit{F-score} = 2 \times \frac{Precision \times Recall}{Precision+Recall}$ and it is an effective measurement for imbalanced datasets. In addition, $\textit{FAR} = \frac{FP}{FP+\text{TN}}$ is measured by the proportion of benign events incorrectly classified as malicious.

B. Datasets

In order to perform the evaluations, we use a popular IDS datasets, namely NSLKDD, and four IoT botnet datasets, i.e., Danmini Doorbell (Dan), Ecobee Thermostat (Eco), Philips B120N10 Baby Monitor (Phi), Provision PT 838 Security Camera (838) [8]. The NSLKDD dataset was proposed in [7] to surpass the inherent issues of the KDD Cup 99 dataset. The NSLKDD has data from the normal traffic and four attack types, i.e., DoS, Probe, R2L and U2L. The data points have 41 features, as shown in the Table II. The IoT botnet datasets (Table III) also form into anomaly (binary) and multi-class problems. For anomaly datasets, we randomly select 70% and 30% of normal traffic for training and testing sets, respectively. For the attack data, the Gafgyt botnet is selected for training, whilst the Mirai botnet is for testing. For multi-class IoT botnet datasets, we select the normal traffic and five types of Gafgyt attacks. For each scenario, 70% samples are for training and 30% are for testing. The number of features in the IoT botnet datasets is 115.

C. Experimental Setting

The experiments are implemented in Python using two frameworks Tensorflow and Scikit-learn [21]. Four classifiers including SVM, LR, DT, and RF are applied on the new representation. We conduct a grid search to tune the hyper-parameters of four classifiers [21], as shown in Table IV. The ADAM optimization algorithm [22] is used to train the neural networks. The learning rate α is initially set at 10^{-4} for all configurations. The weights are initialized using the methods proposed in [23] to facilitate the convergence. The number $epochs$ is set at 2000 and the parameter $batch\_size$ is set at 100. The number of hidden layers of all representation learning models, i.e., VAE, β-VAE, VQVAE, AE, TVAE, and TSVAE are shown in Table V. To apply CSAE to IDS, we implemented this model based on 1D convolution. The same architecture in [24] is built using 1D convolution in Tensorflow. The $pool\_size$ is 2, and the parameters $kernel\_size$ and $feature\_maps$ are 3, and an array of 16, 1, respectively. Finally, for the proposed models, we set all parameters for trading-off the loss function, i.e., β_1, β_2, and β_3 to be 1. We also used the $reduce\_mean$ function for each batch training instead of the $reduce\_sum$ function, making the TVAE and TSVAE become more effectively. This is because the decoders of TVAE and TSVAE are trained with both $\hat{x}$ and x, however the data $\hat{z}$ from the extraction process only uses x. Therefore, using of the $reduce\_mean$ function likely generalizes both x and $\hat{x}$ better than using the $reduce\_sum$ function. R_0 is set at 2 for TVAE and TSVAE during training process.

D. Result Analysis

In this subsection, we will evaluate the performances of TVAE and TSVAE on both anomaly (binary) datasets and multi-label datasets. The results are compared with the current state-of-the-art methods in the same fields. We also investigate various characteristics of TVAE and TSVAE in the latent space as well as in the new representation. Table VI shows the performance of TVAE and TSVAE compared to the other methods including VAE [5], β-VAE [16], Vector Quantized-Variational Auto-Encoder (VQVAE) [17], AE and Convolutional Sparse Auto-Encoder (CSAE) [12] using two metrics, i.e., $Accuracy$ and *F-score*. In this table, the best results on each dataset are printed boldface.

First, we compare TVAE and TSVAE with VAE and β-VAE. It is clear that the performance of TVAE and TSVAE are much greater than those of VAE and β-VAE on multi-label datasets. For instance, on NSLKDD dataset, TVAE and TSVAE achieve accuracies obtained by RF classifier of 84.9% and 85.9%, respectively, whilst those values of VAE and β-VAE are 50.1% and 50.4%. For both anomaly datasets, i.e., DanG-2 and EcoG-2, the results of TVAE and TSVAE are slightly higher than those of VAE variants. The reason could be that, these are two easy binary datasets and these four methods achieve nearly 100% correct classification. This result shows the benefit of using TVAE and TSVAE to represent the new representation by using non-probabilistic function compared to those of VAE and β-VAE using probabilistic function.

Second, comparing with representation learning methods, i.e., AE, CSAE, and VQVAE. Table VI also shows the better performance of TVAE and TSVAE. For example, TVAE and TSVAE achieve the best results on all six datasets, while AE achieves only one time on DanG-2 dataset. For the results obtained by CSAE and VQVAE, $Accruacy$ obtained by TVAE and TSVAE are mostly greater than those of CSAE and VQVAE. More specifically, on 838G-6 dataset, the accuracies obtained by TVAE and TSVAE using RF classifier are 71.6% and 72.7%, respectively, while the figures for CSAE and VQVAE are 70.9% and 71.5%, respectively. This results show the better representation of TVAE and TSVAE compared to the representation of SCAE and VQVAE in classifying the attack data from the normal data in IDS datasets. In addition, the results of AE, SCAE, TVAE, TSVAE are more stable since

Fig. 4: Illustration of data representation on NSLKDD datasets utilizing TVAE.

TABLE II: NSLKDD DATASET

Attack Types	Normal	DoS	Probe	R2L	U2R	Total
No. Train	67343	45927	11656	995	52	125973
No.Test	9711	5741	1106	2199	37	18794

TABLE III: IoT DATASETS

Labels	**Dan**	**Eco**	**Phi**	**838**
Benign	49548	13113	175240	98514
Gafgyt	316650	310630	312723	309040
Mirai	652100	512133	610714	429337

TABLE IV: Values of parameters used in the grid search for classifiers.

LR	Default or $solver$={'lbfgs', 'liblinear'}; C={0.1, 0.5, 1.0, 5.0, 10.0}
SVM	LinearSVC: default or C={0.1, 0.2, 0.5, 1.0, 5.0, 10.0}
DT	Default or $criterion$={'gini', 'entropy'}; max_depth={5, 10, 20, 50, 100}
RF	Default or $n\_estimators$={5, 10, 20, 50, 100, 150}

TABLE V: Neural network architecture settings.

Datasets	Input	$h1$	μ, σ, z	$h2$	$\hat{x}$	$h3$	$\hat{z}$
NSLKDD	41	20	10	20	41	20	10
IoT Family	115	50	10	50	115	50	10

they are used the non-probabilistic function to transfer the original representation into new representations.

To investigate/explain the domination of the TVAE compared to the traditional methods, after training, we put data into the encoder and the decoder of the TVAE and TSVAE models to extract z and $\hat{z}$, respectively. If the $\hat{z}$ dataset is easier to classify than z dataset, it implies that the TVAE model is better than the VAE and the β-VAE models. This is because if the reconstruction error between z and $\hat{z}$ in (6) is removed, the TVAE has the same loss function as that of the VAE and the β-VAE models. In fact, the accuracies of DT classifier using $\hat{z}$ are always by far greater than those using z in Table VII. For example, $Accuracy$ obtained by TVAE on the NSLKDD dataset utilizing $\hat{z}$ achieve 84.6% compared to 47.3% of z. The similar results are observed by TSVAE.

Fig. 4 shows the change in distribution of the training data at different stages under TVAE. First, after the training process, the original training dataset in the Fig. 4a is pushed into the TVAE through the encoder. Then, the latent space z shown in the Fig. 4b is extracted. The Fig. 4c depicts data $\hat{z}$ obtained via data extraction process. It is apparent that the original datasets are more difficult to classify since data points stacked up. Thanks to the assumption that posterior distribution $q(z|x)$ being Gaussian, data points are separated in latent space z by Gaussian distributions. As can be seen from the output layer $\hat{z}$ in Fig. 4c, it manages to copy the distribution of z as similar as possible. After the training process, the data obtained via the data extraction process, to some extent, retains as much information of the Gaussian distribution like that of z. Thus, Fig. 4 is an outstanding justification for why the performance of TVAE is better than those of AE and variant VAE. On the other hand, the Fig. 4 gives an example of experiments to explain the worse results of data given by the z. This is because the latent space z in Fig. 4b is mixed together in a Gaussian, making classifiers to get difficult to classify. Table VIII illustrates the performance of TVAE and TSVAE compared to others using False Alarm Rate (FAE). It is clear that FAE obtained by TVAE and TSVAE are lower than those of AE variants and VAE variants. For instance, on 838G-6 dataset, TSVAE achieve 8.5% in terms of FAE while those of AE and CSAE are similar at 8.7%.

Overall, the results in this subsection show the superior performance of two proposed models, i.e., TVAE and TSVAE. Both models are trained in an unsupervised manner and their performances are often higher than those of the state-of-the-art representation learning models including VQVAE and CSAE. TVAE and TSVAE also achieve the superior performance compared to VAE and β-VAE that using the probabilistic function.

VII. Conclusions

In this paper, we have introduced novel deep neural networks named TVAE and TSVAE for repesentation learning in IoT intrusion detection. The TVAE and TSVAE are trained with a probabilistic function, whilst their extraction process for testing are non-probabilistic. For that TVAE and TSVAE are much more stable than those of the previous stochastic models like VAE and β-VAE. The TVAE's architecture projects data into a new latent space where attack and normal data are more separable. Intensive experiments have been conducted on common datasets, i.e., NSLKDD and five IoT botnet datasets with four common classifiers LR, SVM, DT and RF. The results show that the accuracy of TVAE and TSVAE can achieve the best results on six datasets, which is higher than those of state-of-the-art AE and VAE variants.

References

[1] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, Apr. 2017.

[2] D. Wagner and P. Soto, "Mimicry attacks on host-based intrusion detection systems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, NY, United States, 2002, pp. 255–264.

TABLE VI: Performance of TVAE and TSVAE compared to others.

Datasets	Cfs	Accuracy (%)							F-score (%)						
		VAE	β-VAE	VQVAE	AE	CSAE	TVAE	TSVAE	VAE	β-VAE	VQVAE	AE	CSAE	TVAE	TSVAE
NSLKDD	LR	51.7	51.7	81.1	80.9	75.2	83.7	83.3	35.2	35.2	76.1	76.0	67.8	78.7	78.4
	SVM	51.7	51.7	80.6	78.8	75.6	83.5	84.1	35.2	35.2	75.6	73.9	68.2	78.5	79.1
	DT	51.6	51.6	84.3	84.0	84.4	84.6	**85.9**	35.3	35.3	81.0	79.9	80.0	80.6	**82.9**
	RF	50.1	50.4	84.3	84.4	85.1	84.9	**85.9**	38.0	38.3	80.1	79.7	80.2	80.9	82.3
DanG-6	LR	28.9	28.9	28.9	66.8	63.7	53.3	65.4	13.0	13.0	13.0	54.9	52.4	41.7	54.0
	SVM	28.9	28.9	28.9	66.7	66.8	62.6	66.6	13.0	13.0	13.0	54.8	55.2	51.0	54.7
	DT	28.9	28.8	28.9	67.2	61.7	66.7	**67.9**	13.0	14.0	13.0	55.7	52.3	54.8	**57.2**
	RF	27.0	27.1	28.9	67.3	66.9	66.8	67.4	19.6	19.8	13.0	55.9	55.5	55.2	56.0
PhiG-6	LR	35.9	35.9	35.9	75.1	69.5	58.8	74.0	19.0	19.0	19.0	66.2	58.3	45.0	65.3
	SVM	35.9	35.9	35.9	75.0	73.2	75.1	74.9	19.0	19.0	19.0	66.1	64.5	66.2	66.0
	DT	35.9	35.8	35.9	75.5	75.2	71.6	**76.1**	19.0	19.2	19.0	67.1	66.3	62.9	**68.1**
	RF	34.8	34.8	35.9	72.9	75.2	72.1	75.8	21.0	21.1	19.0	64.2	66.3	63.5	67.5
838G-6	LR	25.7	25.7	69.2	70.7	64.4	48.8	68.7	10.5	10.5	59.0	60.2	51.8	34.3	58.5
	SVM	25.7	25.7	64.5	70.7	67.8	59.3	69.8	10.5	10.5	52.5	60.2	57.3	47.3	59.5
	DT	25.6	25.6	71.6	71.6	71.0	71.9	72.6	12.4	11.1	62.0	62.0	60.7	62.8	**63.8**
	RF	24.1	24.2	71.5	71.5	70.9	71.6	**72.7**	20.6	20.8	61.8	61.6	60.4	62.0	**63.8**
DanG-2	LR	97.8	97.8	**100**	**100**	90.9	97.8	**100**	96.7	96.7	**100**	**100**	93.1	96.7	**100**
	SVM	97.8	97.8	**100**	**100**	97.8	97.8	**100**	96.7	96.7	**100**	**100**	96.7	96.7	**100**
	DT	97.8	97.7	97.5	**100**	73.1	96.5	**100**	96.7	96.7	97.9	**100**	82.5	97.2	**100**
	RF	97.8	97.8	99.3	**100**	99.9	**100**	**100**	96.7	96.7	99.3	**100**	99.9	**100**	**100**
EcoG-2	LR	99.2	99.2	92.9	93.7	99.4	99.2	**100**	98.9	98.9	95.6	96.2	99.3	98.9	**100**
	SVM	99.2	99.2	99.2	93.8	99.2	99.2	99.9	98.9	98.9	98.9	96.2	98.9	98.9	99.9
	DT	99.2	99.0	81.8	99.2	99.4	94.8	99.6	98.8	98.7	89.3	99.4	99.5	96.7	99.6
	RF	99.2	99.2	68.0	99.9	99.8	99.5	99.7	98.9	98.9	80.2	99.9	99.8	99.6	99.7

TABLE VII: Comparison of data extracted by z and $\hat{z}$ obtained by DT classifier (Accuracy %).

Datasets	TVAE		TSVAE	
	z	$\hat{z}$	z	$\hat{z}$
NSLKDD	47.3	**84.6**	84.0	**85.9**
DanG-6	21.0	**66.7**	20.3	**67.9**
PhiG-6	23.3	**71.6**	23.2	**76.1**
838G-6	21.5	**71.9**	21.6	**72.6**
DanG-2	82.4	**96.5**	94.7	**100.0**
EcoG-2	92.9	**94.8**	91.9	**99.6**

TABLE VIII: False Alarm Rate (%) of TVAE and TSVAE compared to others obtained by DT classifier.

Datasets	False Alarm Rate						
	AE	CSAE	VAE	β-VAE	VQVAE	TVAE	TSVAE
NSLKDD	15.2	**13.6**	51.6	51.6	13.7	14.3	14.1
DanG-6	11.7	12.2	28.9	28.8	28.9	11.8	**11.6**
PhiG-6	**6.0**	6.1	35.9	35.8	35.9	8.3	6.7
838G-6	8.7	8.7	25.6	25.7	8.7	8.6	**8.5**
DanG-2	0.5	0.3	97.8	97.7	0.2	**0.1**	**0.1**
EcoG-2	0.4	5.2	99.2	99.0	13.0	**0.2**	0.7

[3] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, 2013.

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

[5] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014*, Banff, AB, Canada, 2014.

[6] J. M. Tomczak and M. Welling, "VAE with a vampprior," *CoRR*, vol. abs/1705.07120, 2017.

[7] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, 2009, pp. 1–6.

[8] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, Mar. 2018.

[9] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[11] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[12] W. Luo, J. Li, J. Yang, W. Xu, and J. Zhang, "Convolutional sparse autoencoders for image classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 7, pp. 3289–3294, Jul. 2018.

[13] J. Kos, I. Fischer, and D. Song, "Adversarial examples for generative models," in *2018 IEEE Security and Privacy Workshops (SPW)*, San Francisco, 2018, pp. 36–42.

[14] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, "Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic," *IEEE Sensors Letters*, vol. 3, no. 1, pp. 1–4, Nov. 2018.

[15] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, Toulon, France, 2017, pp. 1–12.

[16] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in beta-vae," *arXiv preprint arXiv:1804.03599*, 2018.

[17] H. Wu and M. Flierl, "Vector quantization-based regularization for autoencoders," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, New York, USA, 2020, pp. 6380–6387.

[18] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, p. 307–392, Nov. 2019.

[19] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, Dec. 2015.

[20] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation," in *Australasian joint conference on artificial intelligence*. Berlin, Heidelberg: Springer, 2006, pp. 1015–1021.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Feb. 2011.

[22] K. DP and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of the 3rd International Conference for Learning Representations (ICLR)*, San Diego, California, US, 2015, pp. 1–15.

[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, Chia Laguna Resort, Sardinia, Italy, 2010, pp. 249–256.

[24] J. Park, J. Lee, and D. Sim, "Low-complexity cnn with 1d and 2d filters for super-resolution," *Journal of Real-Time Image Processing*, vol. 17, no. 6, pp. 2065–2076, Jun. 2020.