An Empirical Study on Unsupervised Network Anomaly Detection using Generative Adversarial Networks

Tram Truong-Huu
I^{2}R, A*STAR, Singapore
truonght@i2r.a-star.edu.sg

Nidhya Dheenadhayalan
I^{2}R, A*STAR, Singapore
nidhyad@i2r.a-star.edu.sg

Partha Pratim Kundu
I^{2}R, A*STAR, Singapore
kundupp@i2r.a-star.edu.sg

Vasudha Ramnath
I^{2}R, A*STAR, Singapore
vasu@i2r.a-star.edu.sg

Jingyi Liao
I^{2}R, A*STAR, Singapore
liaoj@i2r.a-star.edu.sg

Sin G. Teo
I^{2}R, A*STAR, Singapore
teosg@i2r.a-star.edu.sg

Sai Praveen Kadiyala
I^{2}R, A*STAR, Singapore
saipk@i2r.a-star.edu.sg

ABSTRACT

Network anomalies can arise due to various causes such as abnormal behaviors from users, malfunctioning network devices, malicious activities performed by attackers, malicious software or botnets. With the emergence of machine learning and especially deep learning, many works in the literature developed learning models that are able to detect network anomalies. However, these models require massive amounts of labeled data for model training and may not be able to detect unknown anomalous traffic or zero-day attacks. Unsupervised learning techniques such as autoencoder and its variants do not require labeled data but their performance is still poor. Generative adversarial networks (GANs) have successfully demonstrated their capability of implicitly learning data distributions of arbitrarily complex dimensions. This motivates us to carry out an empirical study on the capability of GANs in network anomaly detection. We adopt two existing GAN models and develop new neural networks for their components, i.e., generator and discriminator. We carry out extensive experiments to evaluate the performance of GANs and compare with existing unsupervised detection techniques. We use multiple datasets that include both realistic traffic captures (PCAP) and synthetic traffic generated by simulation platforms. We develop a traffic aggregation technique to extract statistical features that are useful for the models to learn traffic behaviors. The experimental results show that GANs outperform the existing techniques with a significant improvement in different performance metrics.

KEYWORDS

Deep learning, generative adversarial networks, network security, unsupervised anomaly detection

ACM Reference Format:
Tram Truong-Huu, Nidhya Dheenadhayalan, Partha Pratim Kundu, Vasudha Ramnath, Jingyi Liao, Sin G. Teo, and Sai Praveen Kadiyala. 2020. An Empirical Study on Unsupervised Network Anomaly Detection using Generative Adversarial Networks. In *Proceedings of the 1st Security and Privacy on Artificial Intelligent Workshop (SPAI '20), October 6, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3385003.3410924

1 INTRODUCTION

Communication networks have significantly evolved in velocity and traffic behaviors due to a large number of Internet-connected devices and online applications. By the year 2025, the number of Internet-connected devices is expected to exceed 75.44 billion [25] and the traffic generated only by the Internet vehicles is expected to reach 300000 Exabyte [31]. Securing such complex and high-speed networks become challenging as conventional solutions are no longer scalable and effective. On one hand, host-based anomaly detection approaches are not scalable as many network-connected devices such as IoT devices are resource-constrained. On the other hand, centralized powerful servers using signature-based or rule-based detection are not so effective in detecting unknown anomalous behaviors or zero-day attacks.

Network-centric anomaly detection has been studied to overcome the scalability issue. It captures traffic features and makes the decision based on analysis outcomes. With the development of networking technology, Internet routers today have the capability of capturing traffic features and export metadata such as NetFlow records [3], which has significantly reduced the amount of information captured from the network compared to raw traffic. However, the volume of metadata is still very large due to the large size and high velocity of the network. For instance, an Internet router with 24×10 Gbps ports can produce a few GB of traffic records in 5 minutes of collections. Thus, analyzing such a large amount of traffic is a challenging task. On one hand, it requires domain-specific expertise to determine which records are the anomalous events that will be further investigated. The higher the false positive rate, the higher the upfront investigation cost that will be lost. On the other hand, anomaly detection should also operate in an online and real-time manner that incurs a low computational overhead so as to not delay the traffic forwarding to the destinations.

To overcome the above challenges, many works in the literature have adopted statistical models and deep learning algorithms, which have attracted significant attention in various domains as they have the capability of dealing with complex problems [30]. Network anomaly detection is considered as a binary classification problem in which a data sample (i.e., a set of values of traffic features captured at a time instant or within a time period) is classified as either normal or anomalous. However, existing works [11, 22, 29] present several drawbacks and suffer performance degradation when deployed in practice. First, they use only a small set of predetermined features such as traffic volume, flow rates and entropy to characterize the traffic behavior even though network and security threats are dynamically evolving. The set of appropriate features differs across anomalous traffic or attack types, and there is not "one-size that fits all". This process of feature selection referred to as *feature engineering* requires domain-specific knowledge and a deep understanding of every feature as well as its impact on the performance of the detection algorithm. Second, supervised learning approaches require a training dataset with a large number of data samples associated with their ground truth label –normal or anomalous– to achieve high accuracy. Labeling data, however, is a time-consuming process that requires huge laborious efforts as well as deep domain-specific knowledge. Furthermore, labeling data could be impossible given the large number of traffic records captured on a daily basis [24] while anomalies are typically rarely occurring in practice. Yet, supervised learning might not be able to detect unknown anomalous traffic and zero-day attacks.

Generative adversarial networks (GANs) [7] have recently become one of the popular techniques in machine learning with the capability of implicitly learning data distributions of arbitrarily complex dimensions. Standard GANs consist of two adversarial components: a *generator* and a *discriminator* that are both neural networks. By mapping latent variables drawn from a prior distribution to samples in the training set, the generator aims at learning the data distribution of the training set so as to be able to generate new samples that are similar to real ones. On the other hand, the discriminator serves as an adversary to the generator by distinguishing between samples that are originally from the training set and the samples that are produced by the generator. GANs have empirically demonstrated their success in many applications such as natural images [4, 18], speech [12] and medical images [19].

In this work, we aim at applying GANs to the security domain for network anomaly detection. We demonstrate that GANs can detect whether a data sample is anomalous or not using a reconstruction-based approach. We argue that if the generator has successfully learned the data distribution (i.e., normal network activities), it is reasonable to assume that, the generator would be able to generate the samples that are similar to those in the training set from points in the latent space, resulting in small reconstruction errors between real samples and generated ones. In contrast, given an anomalous data sample, there might not exist any point in the latent space that can result in a generated sample close to the anomalous one. By comparing the reconstruction error between the real sample and the generated sample against a predefined threshold, we then can decide whether the real data sample is anomalous or not. Last but not least, the high-dimension deep learning models implemented in the generator and discriminator relieve the need of employing domain-specific knowledge to determine the appropriate set of features. GANs also do not need data labels during the training phase, thus enabling unsupervised anomaly detection.

We adopt two GAN architectures. The first one is AnoGAN [19] that is based on the standard GAN architecture. The second one is ALAD [32] built upon bi-directional GANs with multiple improvements that stabilize the GAN training process and enable a fast inference (detection). We develop new neural networks with a large number of hidden layers and nodes for the components in the two GAN architectures so that they can learn complex distributions of network traffic. We develop a traffic aggregation technique that performs packet analysis, aggregates the packets into flows and then the flows into sessions. As a result, multiple statistical traffic features can be extracted, thus improving the performance of anomaly detection models. We demonstrate the effectiveness of GANs using conventional machine learning metrics such as precision, recall, F1 score, area under the receiver operating characteristic curve (AUROC) and precision-recall curve (AUPRC). We use multiple datasets including UNSW-NB15 [15], CICIDS2017 [20] and Stratosphere IPS [26]. While UNSW-NB15 and CICIDS2017 provide extracted traffic features, Stratosphere IPS only provides raw traffic captures (i.e., PCAP files). We use the developed aggregation technique to extract features. These datasets represent various traffic distributions ranging from real Internet traffic to synthetically generated traffic. We compare the performance of GANs against several anomaly detection methods including deep structured energy-based models (DSEBM) [33], deep autoencoding Gaussian mixture model (DAGMM) [35] and autoencoder [16].

The rest of the paper is organized as follows. In Section 2, we present the background of GANs and their adoption to the network anomaly detection problem. In Section 3, we present our traffic feature aggregation technique for statistical feature extraction. In Section 4, we present the datasets used in our work. In Section 5, we present the performance study. We analyze several related works in Section 6 before we conclude the paper in Section 7.

2 GENERATIVE ADVERSARIAL NETWORKS FOR NETWORK ANOMALY DETECTION

2.1 Standard Generative Adversarial Networks

In standard architecture, GANs are made up of two adversarial components: a generator and a discriminator that are neural networks.

Definition 1 (Generator and Discriminator - Standard GANs). The generator and discriminator are defined as follows:

$$\begin{aligned} G_{\boldsymbol{\theta}} &: \mathcal{Z} \to \mathcal{X} & D_{\boldsymbol{\psi}} &: \mathcal{X} \to [0,1] \\ \boldsymbol{z} &\mapsto \boldsymbol{x} & \boldsymbol{x} &\mapsto y \end{aligned} \tag{1}$$

We define $G_{\boldsymbol{\theta}}$ and $D_{\boldsymbol{\psi}}$ to be the generator and discriminator networks, where $\mathcal{X} \subseteq \mathbb{R}^d$ is the *data* space and $\mathcal{Z} \subseteq \mathbb{R}^L$ is the *latent* space. The generator and discriminator mappings depend on the trainable parameters $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$. In standard GANs, the choice of parameterization for both of these mappings is given by the use of fully connected neural networks trained using gradient descent via backpropagation. By mapping latent variables $\boldsymbol{z} \in \mathcal{Z}$ to samples $\boldsymbol{x} \in \mathcal{X}$, the generator $G_{\boldsymbol{\theta}}$ induces a conditional distribution $p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x}|\boldsymbol{z}) = \delta(\boldsymbol{x} - G_{\boldsymbol{\theta}}(\boldsymbol{z}))$ and the distribution of the generative model is given by $p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z} \sim p_{\mathcal{Z}}(\boldsymbol{z})}[p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x}|\boldsymbol{z})]$, where the prior distribution $p_{\mathcal{Z}}(z)$ of the latent variable is usually chosen to be a standard Gaussian. By training the parameters $\boldsymbol{\theta}$ of the generator network, the objective of $G_{\boldsymbol{\theta}}$ is to induce $p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x}) \approx p_{\mathcal{X}}(\boldsymbol{x})$. On the other hand, the discriminator $D_{\boldsymbol{\psi}}$ serves as an adversary to $G_{\boldsymbol{\theta}}$ by distinguishing between sample $\boldsymbol{x}$ that is originally from $\mathcal{D}$ and sample $G_{\boldsymbol{\theta}}(z)$ that is produced by the generator. Ideally, a perfect $D_{\boldsymbol{\psi}}$ gets an input $\boldsymbol{x} \in \mathcal{X}$ and outputs a probability y where $y = 0$ if $\boldsymbol{x}$ is a generated sample, i.e., $\tilde{\boldsymbol{x}} := G_{\boldsymbol{\theta}}(z)$ for $z \sim p_{\mathcal{Z}}(z)$ and $y = 1$ if $\boldsymbol{x}$ is a real sample, i.e., $\boldsymbol{x} \sim p_{\mathcal{X}}(\boldsymbol{x})$. In standard GANs, the discriminator $D_{\boldsymbol{\psi}}$ is defined as $D_{\boldsymbol{\psi}} := \sigma(C_{\boldsymbol{\psi}})$ where σ is the sigmoid activation function and $C_{\boldsymbol{\psi}}$ is a real-valued function known as the *critic* that represents the pre-activation logits of the discriminator network.

Definition 2 (Objective Function of GANs)**.** The objective function of GANs is a saddle point problem defined as

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\psi}} \mathcal{L}_{\mathrm{GAN}}(\boldsymbol{\theta}, \boldsymbol{\psi}) = \mathbb{E}_{\boldsymbol{x} \sim p_{\mathcal{X}}(\boldsymbol{x})}[\log D_{\boldsymbol{\psi}}(\boldsymbol{x})] + \mathbb{E}_{z \sim p_{\mathcal{Z}}(z)}[\log(1 - D_{\boldsymbol{\psi}}(G_{\boldsymbol{\theta}}(z)))] \quad (2)$$

This is equivalent to a two-player minimax game between the generator and the discriminator. It is shown in [7] that given sufficient capacity, a unique solution corresponding to a Nash equilibrium exists. In addition, given an optimal $D^*_{\boldsymbol{\psi}}$, minimizing the objective of $G_{\boldsymbol{\theta}}$ is equivalent to minimizing the Jensen-Shannon divergence between $p_{\mathcal{X}}(\boldsymbol{x})$ and $p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x})$. Moreover, this is achieved if and only if $p_{\mathcal{X}}(\boldsymbol{x}) = p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x})$, where the distribution of the generative model successfully replicates the underlying data distribution.

2.2 Enabling GANs for Anomaly Detection

Given a data sample to be examined, the trained GANs aim to generate a new sample that is closest to the given sample and compute the reconstruction error between the real sample and the generated one. If the error is larger than a pre-defined threshold, the real sample is detected as anomalous. We present two techniques for sample reconstruction for different GAN models.

2.2.1 Likelihood Estimation with Sampling. Standard GANs only yield a one-way mapping from the latent space $\mathcal{Z}$ to the data space $\mathcal{X}$ through the generator network. GANs do not have a mechanism for learning an inverse mapping from the data space to the latent space, preventing them from performing inference to compute a posterior distribution $\pi(z|\boldsymbol{x})$ conditioned on a given data sample $\boldsymbol{x}$. Thus, to reconstruct a sample with standard GANs, Schlegl *et al.* presented in AnoGAN [19] a sampling technique to find the best latent representation z of the given data sample $\boldsymbol{x}$. The sampling technique starts with a randomly sampling z_1 from the latent space $\mathcal{Z}$ and pass it through the trained generator network $G_{\boldsymbol{\theta}}$ to obtain a generated sample $G_{\boldsymbol{\theta}}(z_1)$. Based on $G_{\boldsymbol{\theta}}(z_1)$, a loss function is applied to provide gradients that are used to update z_1, resulting in an updated latent representation z_2. Given a pre-defined number of iterations T, the best latent representation of $\boldsymbol{x}$ is obtained at the last iteration through T backpropagation steps. At iteration $t \leqslant T$, the loss function is defined as

$$\mathcal{L}(z_t) = (1 - \lambda) \sum |\boldsymbol{x} - G_{\boldsymbol{\theta}}(z_t)| + \lambda \sum |\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(G_{\boldsymbol{\theta}}(z_t))| \quad (3)$$

where $\boldsymbol{f}(\cdot)$ is the output of an intermediate layer of the discriminator network $D_{\boldsymbol{\psi}}$. The drawback of AnoGAN is the computational overhead as it requires a large number of backpropagation steps to achieve an accurate estimation of likelihoods, i.e., to determine the best latent representation. Thus, applying this technique to network anomaly detection may not be practical especially when network velocity is high and it is required to provide fast detection.

2.2.2 Adversarially Learned Inference. Motivated by the drawbacks of standard GANs, bidirectional models such as ALI [6] have since been studied to incorporate methods, which can simultaneously learn mutually coherent and two-way bijective mappings in order to generate high-quality samples in both the latent and data spaces. The ALI framework learns a bijection between the domains $\mathcal{X}$ and $\mathcal{Z}$ by introducing an inference mechanism in the form of an encoder mapping, $E_{\boldsymbol{\phi}}$, from $\mathcal{X}$ to $\mathcal{Z}$.

Definition 3 (Generator, Encoder - ALI)**.**

$$\begin{aligned} G_{\boldsymbol{\theta}} &: \mathcal{Z} \to \mathcal{X} & \qquad E_{\boldsymbol{\phi}} &: \mathcal{X} \to \mathcal{Z} \\ z &\mapsto \tilde{\boldsymbol{x}} & \boldsymbol{x} &\mapsto \tilde{z} \end{aligned} \quad (4)$$

Given the individual distributions of the latent and data spaces, one domain can be inferred based on the other through conditional distributions induced by $G_{\boldsymbol{\theta}}$ and $E_{\boldsymbol{\phi}}$. The generator $G_{\boldsymbol{\theta}}$ induces a conditional distribution $p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x}|z)$ and a joint distribution $p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x}, z)$ by mapping latent variables $z \in \mathcal{Z}$ to $\tilde{\boldsymbol{x}} \in \mathcal{X}$. On the other hand, the encoder $E_{\boldsymbol{\phi}}$ induces a conditional distribution $p_{E_{\boldsymbol{\phi}}}(z|\boldsymbol{x})$ and a joint distribution $p_{E_{\boldsymbol{\phi}}}(\boldsymbol{x}, z)$ by mapping data samples $\boldsymbol{x} \in \mathcal{X}$ to $\tilde{z} \in \mathcal{Z}$.

Definition 4 (Discriminator - ALI)**.**

$$\begin{aligned} D_{xz_{\boldsymbol{\psi}}} &: \mathcal{X} \times \mathcal{Z} \to [0, 1] \\ (\boldsymbol{x}, z) &\mapsto y \end{aligned} \quad (5)$$

Unlike the discriminator in standard GANs, $D_{xz_{\boldsymbol{\psi}}}$ is trained to discriminate not only samples in the data space ($\boldsymbol{x}$ versus $G_{\boldsymbol{\theta}}(z)$), but also samples in the joint data and latent space ($(\boldsymbol{x}, \tilde{z}) \sim p_{E_{\boldsymbol{\phi}}}(\boldsymbol{x}, z)$ versus $(\tilde{\boldsymbol{x}}, z) \sim p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x}, z)$). The output for a perfect $D_{xz_{\boldsymbol{\psi}}}$ is the probability y where $y = 0$ if $\boldsymbol{x}$ is a generated sample, i.e., $\tilde{\boldsymbol{x}} := G_{\boldsymbol{\theta}}(z)$ for $z \sim p_{\mathcal{Z}}(z)$ and $y = 1$ if $\boldsymbol{x}$ is a real sample, i.e., $\boldsymbol{x} \sim p_{\mathcal{X}}(\boldsymbol{x})$. In actual implementation, the input for $D_{xz_{\boldsymbol{\psi}}}$ is in the form of a concatenation of the data sample $\boldsymbol{x}$ and the latent variable z.

Definition 5 (Objective Function of ALI)**.** The objective function of ALI is a saddle point problem defined as

$$\min_{\boldsymbol{\theta}, \boldsymbol{\phi}} \max_{\boldsymbol{\psi}} \mathcal{L}_{\mathrm{ALI}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \mathbb{E}_{\boldsymbol{x} \sim p_{\mathcal{X}}(\boldsymbol{x}), \tilde{z} \sim p_{E_{\boldsymbol{\phi}}}(z|\boldsymbol{x})}[\log D_{xz_{\boldsymbol{\psi}}}(\boldsymbol{x}, \tilde{z})] + \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x}|z), z \sim p_{\mathcal{Z}}(z)}[\log(1 - D_{xz_{\boldsymbol{\psi}}}(\tilde{\boldsymbol{x}}, z))] \quad (6)$$

While the optimal discriminator could be achieved theoretically by solving the above optimization problem, it is not the case in practice since the training does not necessarily converge. Consequently, it can be happen that $G_{\boldsymbol{\theta}}(E_{\boldsymbol{\phi}}(\boldsymbol{x})) \neq \boldsymbol{x}$, leading to cycle-consistency violation as discussed in [13]. Such a violation could create issues for network anomaly detection using the reconstruction-based approach. Yet, training of GANs is empirically well-known for being highly unstable and sensitive. The loss functions of both the discriminator and generator with respect to their parameters tend to oscillate wildly during training, for theoretical reasons investigated in [1]. GANs have also been observed to display signs of mode collapse, which occurs when the generator finds only a limited variety of data samples, which "work well" against the discriminator and repeatedly generates similar copies of data samples.

To address the cycle-consistency issue, Li *et al.* recently developed the ALICE framework [13] that regularizes the conditional distributions with a conditional entropy under the joint distribution over $\boldsymbol{x}$ and $\boldsymbol{z}$. The regularization imposed on the generator network $G_{\boldsymbol{\theta}}$ and encoder network $E_{\boldsymbol{\phi}}$ is achieved by adding an additional discriminator $D_{\boldsymbol{xx}_{\boldsymbol{\eta}}}(\boldsymbol{x}, \hat{\boldsymbol{x}})$, which is adversarially trained with parameter $\boldsymbol{\eta}$ to discriminate $\boldsymbol{x}$ versus $\hat{\boldsymbol{x}}$ where $\hat{\boldsymbol{x}}$ is the reconstruction of $\boldsymbol{x}$, i.e., $\hat{\boldsymbol{x}} = G_{\boldsymbol{\theta}}(E_{\boldsymbol{\phi}}(\boldsymbol{x}))$. Based on the ALICE framework, in [32], Zenati *et al.* developed ALAD framework that further stabilizes the training of GANs by applying another conditional entropy regularization and spectral normalization [14]. In addition to discriminator $D_{\boldsymbol{xx}_{\boldsymbol{\eta}}}$, the authors added a new discriminator $D_{zz_{\omega}}$ that is adversarially trained with parameter $\boldsymbol{\omega}$ to discriminate $\boldsymbol{z}$ versus $\hat{\boldsymbol{z}}$ where $\hat{\boldsymbol{z}}$ is the latent representation of the sample generated from $\boldsymbol{z}$, i.e., $\hat{\boldsymbol{z}} = E_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))$. In summary, the discriminators in the ALAD framework are define as follows:

Definition 6 (Discriminators - ALAD)**.**

$$\begin{aligned} D_{xz_{\psi}} &: \mathcal{X} \times \mathcal{Z} \to [0,1] & D_{xx_{\eta}} &: \mathcal{X} \times \mathcal{X} \to [0,1] \\ &(\boldsymbol{x}, \boldsymbol{z}) \mapsto y & &(\boldsymbol{x}, \hat{\boldsymbol{x}}) \mapsto y \\ D_{zz_{\omega}} &: \mathcal{Z} \times \mathcal{Z} \to [0,1] \\ &(\boldsymbol{z}, \hat{\boldsymbol{z}}) \mapsto y \end{aligned} \tag{7}$$

In actual implementation, the input for $D_{xx_{\eta}}$ is in the form of a concatenation of the data sample $\boldsymbol{x}$ and its reconstruction $\hat{\boldsymbol{x}}$. Similarly, the input for $D_{zz_{\omega}}$ is a concatenation of $\boldsymbol{z}$ and $\hat{\boldsymbol{z}}$.

Definition 7 (Objective Function of ALAD)**.**

$$\begin{aligned} \min_{\boldsymbol{\theta},\boldsymbol{\phi}} \max_{\boldsymbol{\psi},\boldsymbol{\eta},\boldsymbol{\omega}} & \mathcal{L}_{\text{ALAD}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\eta}, \boldsymbol{\omega}) = \\ & \mathbb{E}_{\boldsymbol{x} \sim p_X(\boldsymbol{x}), \tilde{z} \sim p_{E_{\boldsymbol{\phi}}}(z|\boldsymbol{x})}[\log D_{xz_{\psi}}(\boldsymbol{x}, \tilde{\boldsymbol{z}})] + \\ & \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x}|z), z \sim p_Z(z)}[\log(1 - D_{xz_{\psi}}(\tilde{\boldsymbol{x}}, \boldsymbol{z}))] + \\ & \mathbb{E}_{\boldsymbol{x} \sim p_X(\boldsymbol{x})}[\log D_{xx_{\eta}}(\boldsymbol{x}, \boldsymbol{x})] + \\ & \mathbb{E}_{\hat{\boldsymbol{x}} \sim p_{G_{\boldsymbol{\theta}}}(\hat{\boldsymbol{x}}|z), z \sim p_{E_{\boldsymbol{\phi}}}(z|\boldsymbol{x})}[\log(1 - D_{xx_{\eta}}(\boldsymbol{x}, \hat{\boldsymbol{x}}))] + \\ & \mathbb{E}_{z \sim p_Z(z)}[\log D_{zz_{\omega}}(\boldsymbol{z}, \boldsymbol{z})] + \\ & \mathbb{E}_{\hat{z} \sim p_{E_{\boldsymbol{\phi}}}(\hat{z}|\boldsymbol{x}), \boldsymbol{x} \sim p_{G_{\boldsymbol{\theta}}}(\boldsymbol{x}|z)}[\log(1 - D_{zz_{\omega}}(\boldsymbol{z}, \hat{\boldsymbol{z}}))] \end{aligned} \tag{8}$$

The ALAD framework has empirically demonstrated the ability to provide greater stability in training and to better reconstruct data samples as compared to other GAN models.

2.2.3 Computation of Reconstruction Errors. Given that GANs have been trained and their parameters are optimized, we now present the method to compute the reconstruction error for data sample $\boldsymbol{x}$. Generally, we first need compute the latent representation $\boldsymbol{z}$ of sample $\boldsymbol{x}$ and then pass $\boldsymbol{z}$ through the generator network $G_{\boldsymbol{\theta}}$ to obtain $\hat{\boldsymbol{x}}$. Depending on the GAN model whether it is AnoGAN or ALAD, the suitable approach will be used to compute the latent representation as discussed in the previous section, i.e., AnoGAN needs to run an iteration process of likelihood estimation while ALAD just passes the sample $\boldsymbol{x}$ through the encoder network $E_{\boldsymbol{\phi}}$.

Given $\boldsymbol{x}$ and its reconstruction $\hat{\boldsymbol{x}}$, there exist different methods to compute the distance (reconstruction error) between them such as Euclidean distance, mean squared errors, etc. AnoGAN adopts its own loss function defined in Eq. (3) to compute the reconstruction error where the latent representation used to reconstruct sample $\boldsymbol{x}$ is obtained at the last iteration of the likelihood estimation process. The reconstruction error is defined as

$$A(\boldsymbol{x}, \hat{\boldsymbol{x}}) = (1 - \lambda) \sum |\boldsymbol{x} - \hat{\boldsymbol{x}}| + \lambda \sum |\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\hat{\boldsymbol{x}})| \tag{9}$$

where $\boldsymbol{f}(\cdot)$ is the output of an intermediate layer of the discriminator network $D_{\boldsymbol{\psi}}$. Similar to AnoGAN, ALAD also uses an intermediate feature representation of the cycle-consistency discriminator network $D_{xx_{\eta}}$ to compute the reconstruction error between sample $\boldsymbol{x}$ and its reconstruction $\hat{\boldsymbol{x}}$. Specifically,

$$A(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}, \hat{\boldsymbol{x}})\|_1 \tag{10}$$

where $\boldsymbol{f}(\cdot, \cdot)$ is the activation values of the layer before the logits in the cycle-consistency discriminator network $D_{xx_{\eta}}$.

Since GANs have been trained on the training set with only normal samples, they should be able to accurately encode and reconstruct normal data samples with small reconstruction error. Samples with a large value of reconstruction error are considered to be anomalous as they are poorly reconstructed. There are several ways to choose the threshold of the anomaly score to determine whether a sample is anomalous or not. If the training set contains only normal data samples, one may use the highest reconstruction error among the samples in the training set as the threshold. In practice, the training set may contain a small percentage (e.g., 5%) of the data as anomalous samples. One can also use this percentage as the threshold of the anomaly score. The smaller the threshold, the higher the false positive rate. If the labeled data is available, we can make use of it to determine the threshold so as to achieve a high detection rate while having a small false positive rate.

3 TRAFFIC AGGREGATION AND STATISTICAL FEATURE EXTRACTION

Regardless of the capability of deep learning models in learning high-level features, it is not practical to apply them directly to raw traffic due to the high velocity of networks. This requires a technique that performs flow recording and aggregates multiple records that have common fields (e.g., having the same source IP address) into sliding windows referred to as *traffic sessions* so as to provide a richer set of statistical features for the anomaly detection models. Such aggregation allows us to identify the offending IP addresses and also the traffic sessions that are anomalous.

Definition 8 (Traffic Flow)**.** A traffic flow is a sequence of packets that have the same 5-tuple (source IP, destination IP, source port, destination port, protocol) such that the inter-arrival time between two consecutive packets is less than a threshold, e.g., 5 seconds.

For each flow, different features can be captured such as the total number of packets, total number of bytes, packet rate and byte rate. We also note that with the definition above, traffic flows have an inherent notion of direction. This results in application-level requests or commands to be part of one flow, while the response packets are part of a separate flow. We define a traffic session as the next level of aggregation where statistical features of corresponding request and response flows are computed. Most NetFlow or IPFIX implementations are unidirectional, meaning that TCP connections between two hosts result in two flows (i.e., A to B and B to A), which can be stitched back into one bidirectional flow. To capture more statistical features that could be useful for the detection of different attacks, not only two unidirectional flows should be aggregated but also any other two flows that have common basic features can be aggregated. For instance, if a malicious host carries out a scan attack on different victims, the source IP, source port and protocol should be the same for all the generated flows. Those flows should be then aggregated in one session.

Definition 9 (Traffic Session)**.** A traffic session is a set of flows that have the same 5-tuple (source IP, destination IP, source port, destination port, protocol) such that the inter-arrival time between two consecutive flows is less than a threshold, e.g., 3 minutes.

Given the above definitions of traffic flows and sessions, the duration of flows and sessions could be arbitrarily large. To avoid this, a timer and/or an accounting function is needed to trigger the aggregation process after receiving a number of flows. In other words, the sliding window defines an upper bound of flow duration and session duration. If a flow that lasts longer than this sliding window will be split into multiple flows, each having the duration as that of the sliding window except the last flow and belonging to different sessions. This also ensures that the maximum duration of a session will be bounded by the sliding window. Determining the length of such a sliding window needs to consider both practicality and quality of aggregated statistics. If the sliding window is too short, the aggregated statistics are not significant while using a long time will delay the detection if anomalies occur. Consequently, there is a total of 39 features that can be collected from traffic flows and sessions, which are classified into four categories:

Aggregated features. These are computed as sum, total count, max or min value of a simple feature over a session. These include the total number of bytes, total number of flows, total number of packets. If the aggregation is performed with bidirectional flows, we may have aggregated features for forward traffic (i.e., flows from the host represented by source IP to the host represented by destination IP) and backward traffic (i.e., flows from the host represented by destination IP to the host represented by source IP).

Temporal features. These are the sum, mean and standard deviation of the duration of the flows in a session. If the flows are bidirectional, we may have other temporal features of forward and backward traffic: the sum, mean and standard deviation of the duration of forward and backward flows.

Statistical feature. These are the mean and standard deviation of a simple feature over all the flows in a session including the number of packets and number of bytes. If flows are bidirectional, we can also have the mean and standard deviation of the number of packets and number of bytes of forward and backward flows.

Connection-context features. These features are useful in detecting coordinated (multi-flow) attacks with co-related patterns. Considering a set of n previous connections before the last occurrence of a packet in the current flow under processing, these features include (i) the number of connections originating from the source IP; (ii) the number of connections to the destination IP; (iii) the number of connections originating from the source IP and bound for the specific port at any destination IPs; (iv) the number of connections ending at the destination IP and from a specific port of any source IPs; and (v) the number of connections originating from the source IP and bound for the destination IP.

4 DATASETS

We describe synthetic and realistic datasets used to demonstrate the performance of the developed GAN models.

4.1 UNSW-NB15

This dataset was recently collected by Moustafa *et al.* [15]. The authors used an attack automatic generation machine called IXIA PerfectStorm. They collected `tcpdump` traces of the network traffic for a total of 31 hours at the beginning of 2015, resulting in around 2M flow records, each having 49 features. In Fig. 1a, we present a 2-dimensional projection of 10000 randomly-selected data samples from the dataset using t-SNE [28]. We have carried out an extensive parameter tuning in terms of the number of intermediate features produced by PCA and the perplexity value of the t-SNE algorithm. However, we observed that there is not a clear separation between normal traffic and anomalous traffic in this projection. This is due to the non-linearity of the data distribution that creates a great impact on the performance of t-SNE, which actually uses linear dimensionality reduction for its purpose. On the other hand, this could also be due to the similarity of behavior between normal traffic and malicious traffic.

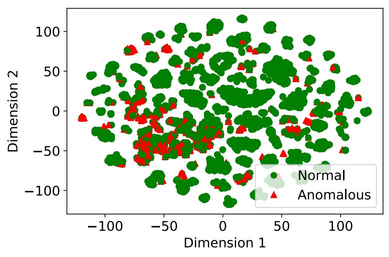
4.2 CICIDS2017

The second dataset is CICIDS2017 [20]. To build this dataset, the authors designed and developed two networks, namely attacking network and victim network. The authors used the B-Profile system developed in their previous work to generate background and normal traffic. To generate attacking traffic, the authors created 6 attack profiles and executed them on the attacking network by using related tools and codes publicly available. The authors run the testbed infrastructure for 5 days from Monday, July 3rd to Friday, July 7th. Attacks were subsequently executed during this period. In Fig. 1b, we present the t-SNE plot of 10000 data point randomly selected from the CICIDS2017 dataset.

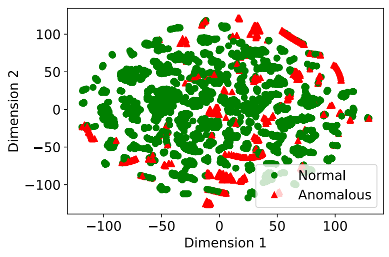
We note that both UNSW-NB15 and CICIDS2017 do not provide the statistical features of traffic sessions. The flows are bidirectional and identified by the 5-tuple (source IP, destination IP, source port, destination port, protocol). A TCP flow will be recorded if it terminates with a FIN packet while a UDP flow is terminated by a timeout, e.g., 600 seconds. However, this may lead to a fact that an attacking flow with a long duration can only be detected after its completion. We address this by incorporating the concept of traffic sessions and define a threshold of the maximum duration of flows and sessions, thus enabling a fast detection.

4.3 Stratosphere IPS

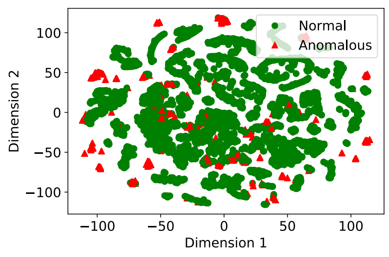
The last dataset used in our work is Stratosphere IPS [26]. Unlike UNSW-NB15 and CICIDS2017 that provide traffic features, Stratosphere IPS provides realistic network traffic captures in the format of PCAP files. We downloaded all the normal traffic captures and malware captures and then applied the traffic aggregation technique presented above to extract features. We made use of all the captures of normal traffic available on the Stratosphere website. For malware captures, we downloaded one capture for each of the following malware families: Andro, Barys, Emotel, Geodo, Htbot, Miuref, Necurse, Sality, Vawtrak, Yakes and Zeus. Depending on the parameter setting of the traffic aggregation technique (i.e., the inter-arrival time between two packets, inter-arrival time between two



(a) UNSW-NB15.



(b) CICIDS2017.



(c) Stratosphere IPS.

Figure 1: 2-dimensional projection of 10000 normal (in green) and anomalous (in red) samples of different datasets.

Table 1: Datasets and normal/anomalous traffic distribution

Name of dataset	Number of data points	Percentage of anomalies (%)
UNSW-NB15	2540047	12.65
CICIDS2017	2813797	19.67
Stratosphere IPS	463347	9.81

flows, maximum session duration), there will be different numbers of the traffic sessions recorded. For instance, with the inter-arrival time between two packets set to 5 seconds, inter-arrival time between two flows and maximum session duration set to 180 seconds, we obtained 418247 sessions, corresponding to 418247 data points for normal traffic. To maintain a low percentage (say 10%) of anomalous traffic in the entire dataset, we took only the first 4100 sessions of each malware capture. In Fig. 1c, we present a 2-dimensional projection of 10000 randomly-selected data points from the dataset.

In Table 1, we summarize the number of data points and percentage of anomalous data points for each dataset. We can see that all the datasets are imbalanced, i.e., there is a large difference between the number of normal data points and the number of anomalous data points. This reflects the realistic scenario that anomalies seldomly occur in production networks. However, this could create a critical issue if we use supervised learning algorithms to detect anomalies. As we discussed earlier, while using an unsupervised learning approach for anomaly detection, the label of flow records is not used during the learning but only in the testing phase. The percentage of anomalous data points is used to determine the threshold of anomaly scores for each dataset. For training the detection models, we conducted a train-test split of approximately 80% – 20% from the datasets using a random seeded shuffle. We discarded anomalous samples from the training set, thus setting up a novelty detection task, also known as "zero-positive" anomaly detection.

5 EXPERIMENTS

5.1 Comparison and Performance Metrics

We run the experiments with AnoGAN, ALAD, and the following baseline algorithms.

Deep Structured Energy-Based Models (DSEBM). DSEBM [33] used an energy-based method for image anomaly detection. Similar to denoising autoencoders, DSEBM computes the accumulated energy across layers in the neural networks. To detect an anomaly, the authors used both reconstruction error and energy error. In our experiments, we used both criteria denoted as DSEBM-r (for reconstruction error) and DSEBM-e (for energy error).

Deep Autoencoding Gaussian Mixture Model (DAGMM). It is an autoencoder-based approach for anomaly detection. DAGMM consists of an autoencoder and an estimator. While the autoencoder is trained to generate a sensible latent space and reconstruction features, the estimator is trained to output parameters of a Gaussian mixture model (GMM) that models the lower-dimensional latent space. At inference time, the learned GMM computes the likelihood of a sample based on its latent and reconstruction features. The computed likelihood is used as the anomaly score to determine whether the sample is an anomaly or not.

Autoencoder (AE). This is a standard autoencoder whose encoder and decoder are deep fully-connected neural networks. Similar to other baseline algorithms, the autoencoder also detects anomalies using the reconstruction-based approach.

In Appendix A, we present the details of architecture and parameters of neural networks used in AnoGAN and ALAD, respectively. We note that the number of units in the last layer of the generator depends on the number of total features after encoding the categorical features in each dataset. This also applies to the first layer (i.e., input layer) of the discriminators in both AnoGAN and ALAD. For a fair comparison among the models, the components with the same functionality in the models share the same architecture. For instance, the encoder in DSEBM, DAGMM, AE and ALAD share the same architecture. This also applies to the decoder of DSEBM, DAGMM and AE or the generator of ALAD.

To evaluate the performance of GAN models and baseline models, we made use of labels of data samples to compute conventional performance metrics of machine learning including precision, recall and F1 score. We also compute the area under the receiver operating characteristic curve (AUROC) and area under the precision-recall curve (AUPRC) to provide more insightful analysis.

5.2 Analysis of Results

5.2.1 Overall Performance. In Table 2, we present the overall performance of all the models including precision, recall, F1 score, AUROC and AUPRC on different datasets. Overall, the experimental results show the trends that ALAD achieves the highest performance in all the metrics for all the datasets, AE is the second best followed by DAGMM. DSEBM and AnoGAN are interchangeably the worst. Compared to the second best model (AE), ALAD significantly improves the performance, especially on UNSW-NB15. In contrast to ALAD, AnoGAN does not perform well on all the datasets, especially on CICIDS2017. This is explained by the fact that AnoGAN uses the likelihood estimation with sampling to determine the best latent representation for a data sample, which is

Table 2: Precision, Recall, F1 score, AUROC and AUPRC

Model	Prec.	Recall	F1 score	AUROC	AUPRC
UNSW-NB15					
DSEBM-r	0.4505	0.4564	0.4534	0.8756	0.4112
DSEBM-e	0.4300	0.4356	0.4328	0.8214	0.3656
DAGMM	0.5351	0.5272	0.5311	0.9170	0.5051
AE	0.7738	0.7825	0.7781	0.9709	0.7237
AnoGAN	0.4345	0.4394	0.4369	0.8765	0.3864
ALAD	**0.8473**	**0.8583**	**0.8527**	**0.9882**	**0.8831**
CICIDS2017					
DSEBM-r	0.2646	0.2649	0.2647	0.8958	0.3520
DSEBM-e	0.2430	0.2432	0.2431	0.8765	0.2793
DAGMM	0.7641	0.7649	0.7651	0.9512	0.7855
AE	0.8084	0.8092	0.8088	**0.9743**	0.8158
AnoGAN	0.1439	0.1440	0.1439	0.7492	0.1041
ALAD	**0.8260**	**0.8268**	**0.8264**	0.9529	**0.8271**
Stratosphere IPS					
DSEBM-r	0.4249	0.4250	0.4249	0.7997	0.4740
DSEBM-e	0.4372	0.4374	0.4373	0.7902	0.4702
DAGMM	0.6521	0.6524	0.6523	0.8603	0.6940
AE	0.6535	0.6538	0.6536	**0.9179**	0.7232
AnoGAN	0.4676	0.4678	0.46.77	0.8493	0.4102
ALAD	**0.6993**	**0.6996**	**0.6995**	0.9053	**0.7501**

not effective compared to training an encoder as done in ALAD. Interestingly, AE performs much better compared to DAGMM even though both of them use an autoencoder for reconstructing data samples. We note that the autoencoder component in DAGMM has the same architecture and training parameters as AE. The difference between them is the computation of the anomaly scores.

Comparing the performance of ALAD among the datasets, we observe that ALAD has a similar performance on UNSW-NB15 and CICIDS2017. However, there is a significant drop in the performance on Stratosphere IPS. The reason for this performance degradation could be the lack of training data since deep learning models are well-known to be data-hungry and require millions of data samples for training to achieve the desired performance. Even though we made use of all the normal captures (i.e., PCAP files) available on the Stratosphere IPS website we could produce only about 400K data samples, which is 5× fewer compared to UNSW-NB15 and 6× fewer compared to CICIDS2017.

In Fig. 2, we present the distribution of the anomaly scores (reconstruction errors) for the test data of the three datasets. We used ground truth labels to differentiate between normal traffic and anomalous traffic. This also allows us to analyze the behavior of anomalous traffic compared with that of normal traffic. Generally, we observe that there is some overlap between normal and anomalous traffic, meaning that some anomalous traffic sessions behave almost similarly to normal traffic. However, we can determine a cut off point to differentiate them.

5.2.2 *ROC and PRC for Anomaly Detection.*

We evaluate the performance of the models by analyzing Receiver Operating Characteristic (ROC) curves and Precision-Recall Curves (PRC). While ROC curves represent the trade-off between the true positive rate and false positive rate for the models using different anomaly thresholds, Precision-Recall curves represent the trade-off between the true positive rate and positive predictive value for the models using different anomaly thresholds. Based on ROC curves, a model has a high performance if the true positive rate is high while the false positive rate is low. However, as the datasets used in our work are imbalanced and anomaly detection focuses more on the anomalies (i.e., the minority class), Precision-Recall curves are more appropriate to assess the performance of the models. Indeed, as presented in Table 2, we observe that several models could achieve high AUROC's value but AUPRC's value is low (e.g., DAGMM and AnoGAN). In Fig. 3, we present the ROC curves and Precision-Recall curves achieved by ALAD for the three datasets. The experimental results show that ALAD can better generalize the detection of anomalies based on statistics obtained from the normal traffic in the zero-positive training datasets. This makes ALAD outperform the remaining models by achieving high performance in AUPRC. For the case of the CICIDS2017 and Stratosphere IPS datasets, AE has a slightly better performance in AUROC compared to ALAD.

Table 3: Training Time of Models (in seconds)

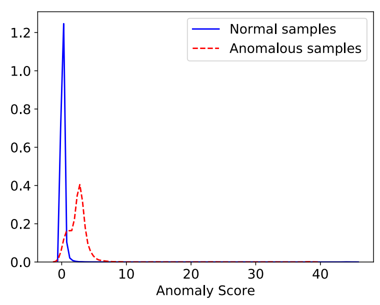
Name of dataset	AE		ALAD	
	Time	No. Epochs	Time	No. Epochs
UNSW-NB15	33	1000	329	120
CICIDS2017	37	1000	220	170
Stratosphere IPS	68	500	500	90

5.2.3 *Training Time of Models.*

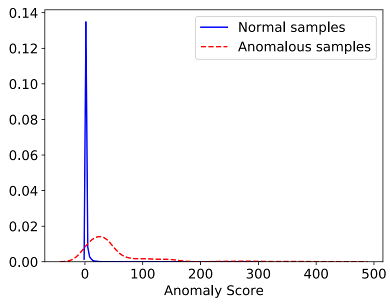
We note that the training time of the models highly depends on the complexity of the models, their training parameters and computational capacities. Nevertheless, we present a relative comparison of training time between the two best models (AE and ALAD) to achieve the performance stated in Table 2. In Table 3, we present the average time of a training epoch of AE and ALAD trained on the same computer with the same training parameters (i.e., batch size, learning rate and decay). We observe that ALAD requires a significantly longer time (at least 6× longer) to train the model for an epoch. This is reasonable as ALAD's architecture is much more complex compared to AE with 3 more neural networks for the discriminators. However, to achieve the stated performance, AE requires a much higher number of training epochs compared to ALAD. This leads to the fact that AE and ALAD finally require the same amount of time for training but achieving a different level of performance as analyzed above.

6 RELATED WORK

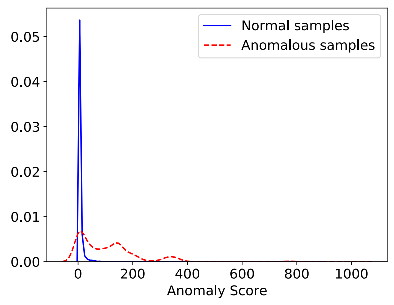
Different detection approaches have been developed such as rule-based detection [5], statistical models [23] and data analytics approaches [2, 16, 17]. The rule-based detection systems [5] could not react to new types of anomalies that can be caused by different factors such as user behavior and software updates. The statistical approaches first model normal traffic and then use a theoretical framework to detect the deviation from the normal data [23]. Such approaches require a large amount of labeled data for the model to continuously adapt over time. In [2], the authors used a Random Forest (RF) classifier to detect command-and-control (C&C) servers by analyzing the NetFlow records. In [17], the authors developed an anomaly detector using an unsupervised deep learning technique. The authors used variational autoencoders to learn the latent representation of network traffic and detect anomaly using reconstruction error. In [16], the authors developed anomaly detection models



(a) UNSW-NB15 Test Set.

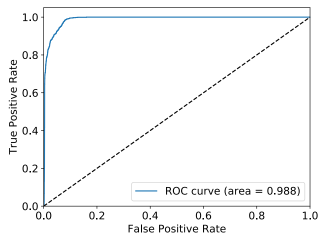


(b) CICIDS2017 Test Set.

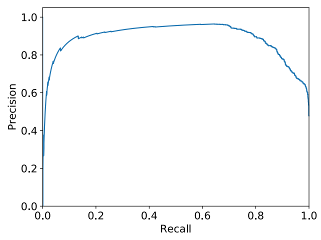


(c) Stratosphere IPS Test Set.

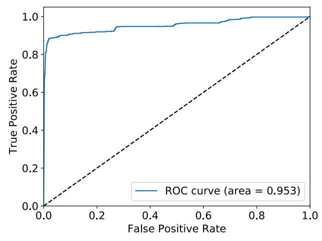
Figure 2: Distribution of Anomaly Scores on the Test Sets with ALAD.



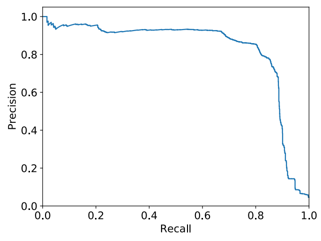
(a) ROC for UNSW-NB15.



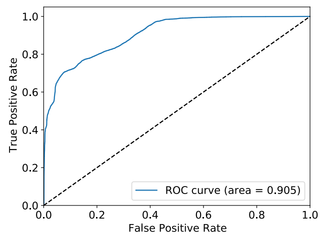
(b) PRC for UNSW-NB15.



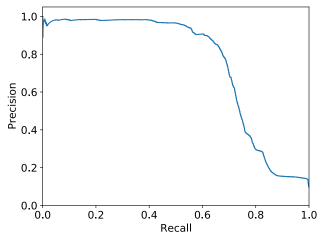
(c) ROC for CICIDS2017.



(d) PRC for CICIDS2017.



(e) ROC for Stratosphere IPS.



(f) PRC for Stratosphere IPS.

Figure 3: ROC and PRC on the Test Sets with ALAD.

based on different deep neural network structures including convolutional neural networks (CNN), autoencoders and recurrent neural networks. The experimental results on the NSLKDD dataset [27] showed that the recurrent neural network model (LSTM) delivers the best performance with 89% of accuracy. Also using CNN, the work presented in [10] used one-dimensional convolutional layers to detect anomaly in networks. Nevertheless, except autoencoders, CNN and LSTM need labeled data for training purposes. In [34], the authors developed a feed-forward neural network model and CNN for network anomaly detection. The authors validated their proposed models using the NSLKDD dataset and compared the performance with conventional machine learning algorithms such as J48, RF, Naive Bayes, Random Tree, and support vector machines.

There are also approaches that combine different learning models to perform network anomaly detection. In [8], the authors first used an unsupervised anomaly detection with a shallow autoencoder and then used a custom nearest-neighbor classifier to filter false positives. In [9], the authors combined CNN and LSTM and developed a model called C-LSTM for anomaly detection in web traffic. In [21], the authors developed a nonsymmetric deep autoencoder (NDAE) for unsupervised feature learning. The output features will be the input of an RF model that detects anomalies and classifies the different attack types. Different from the above works, in this paper, we adopted GANs for network anomaly detection with an unsupervised learning approach. Furthermore, most of the works used KDDCup99 dataset and its variants for performance evaluation. Such datasets are outdated and no longer reflect the dynamics of the current networks. We carried out experiments with recent datasets, which include the datasets generated by enterprise network simulators and datasets collected from realistic traffic.

7 CONCLUSION

In this paper, we studied the use of generative adversarial networks in network anomaly detection. We presented different GAN architectures and their adoption to develop an unsupervised network anomaly detection framework. We developed a traffic aggregation technique in which we introduced the concept of traffic sessions to extract more useful statistical features to train the detection models. We carried out extensive experiments to evaluate the performance of GAN-based network anomaly detection using different publicly available datasets including both synthetic and realistic network traffic. We compared the performance of GANs against existing deep learning models using conventional performance metrics such as precision, recall, F1 score, AUROC and AUPRC. The experimental results show that GAN-based network anomaly detection is highly competitive with state-of-the-art anomaly detection approaches without using label information during the training. While our work addressed the first step of anomaly detection, an anomaly can be caused by a security attack. Thus, a potential future work is to perform anomaly attribution to determine the type of anomalies such as attack types.

ACKNOWLEDGMENTS

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its RIE2020 AME Core Funds (SERC Grant No. A1916g2047).

REFERENCES

[1] Martin Arjovsky and Léon Bottou. 2017. Towards Principled Methods for Training Generative Adversarial Networks. In *Proc. ICLR 2017.* Toulon, France.

[2] L. Bilge et al. 2012. Disclosure: Detecting Botnet Command and Control Servers Through Large-scale NetFlow Analysis. In *Proc. ACSAC'12.* Orlando, USA.

[3] B. Claise et al. 2013. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information.* Technical Report RFC 7011. Cisco Systems.

[4] A. Creswell et al. 2018. Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine* 35, 1 (Jan. 2018), 53–65.

[5] N. Duffield et al. 2009. Rule-Based Anomaly Detection on IP Flows. In *IEEE INFOCOM 2009.* Rio de Janeiro, Brazil, 424–432.

[6] Vincent Dumoulin et al. 2017. Adversarially Learned Inference. In *Proc. ICLR 2017.* Toulon, France.

[7] Ian Goodfellow et al. 2014. Generative Adversarial Nets. In *Proc. NIPS 2014.* Montréal, Canada.

[8] G. Kathareios et al. 2017. Catch It If You Can: Real-Time Network Anomaly Detection with Low False Alarm Rates. In *Proc. 16th IEEE International Conference on Machine Learning and Applications.* Cancun, Mexico, 924–929.

[9] Tae-Young Kim and Sung-Bae Cho. 2018. Web traffic anomaly detection using C-LSTM neural networks. *Expert Systems with Applications* 106 (Sept. 2018).

[10] D. Kwon et al. 2018. An Empirical Study on Network Anomaly Detection Using Convolutional Neural Networks. In *Proc. IEEE CDCS 2018.* Vienna, Austria.

[11] Anukool Lakhina et al. 2004. Characterization of Network-wide Anomalies in Traffic Flows. In *Proc. ACM SIGCOMM IMC'04.* Taormina, Sicily, Italy, 201–206.

[12] Hung-yi Lee and Yu Tsao. 2018. Generative Adversarial Network and its Applications to Speech Signal and Natural Language Processing. http://sigport.org/2863

[13] Chunyuan Li et al. 2017. ALICE: Towards Understanding Adversarial Learning for Joint Distribution Matching. In *Proc. NIPS 2017.* Long Beach, USA.

[14] Takeru Miyato et al. 2018. Spectral Normalization for Generative Adversarial Networks. In *Proc. ICLR 2018.* Vancouver, Canada.

[15] N. Moustafa and J. Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *Proc. 2015 Military Communications and Information Systems Conference.* Canberra, Australia.

[16] S. Naseer et al. 2018. Enhanced Network Anomaly Detection Based on Deep Neural Networks. *IEEE Access* 6 (Aug. 2018), 48231–48246.

[17] Q. P. Nguyen et al. 2019. GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection. In *Proc. IEEE CNS 2019.* Washington.

[18] Alec Radford et al. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *Proc. International Conference on Learning Representation, Workshop Track.* Caribe Hilton, San Juan, Puerto Rico.

[19] Thomas Schlegl et al. 2017. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In *Proc. International Conference on Information Processing in Medical Imaging.* Boone, USA, 146–157.

[20] I. Sharafaldin et al. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proc. ICISSP 2018.* Canberra, Australia.

[21] N. Shone et al. 2018. A Deep Learning Approach to Network Intrusion Detection. *IEEE Trans. Emerg. Topics Comput. Intell.* 2, 1 (Feb. 2018), 41–50.

[22] Fernando Silveira et al. 2010. ASTUTE: Detecting a Different Class of Traffic Anomalies. In *Proc. ACM SIGCOMM 2010.* New Delhi, India, 267–278.

[23] F. Simmross-Wattenberg et al. 2011. Anomaly Detection in Network Traffic Based on Statistical Inference and α-Stable Modeling. *IEEE Trans. Depend. Sec. Comput.* 8, 4 (July 2011), 494–509.

[24] R. Sommer et al. 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *Proc. IEEE S&P 2010.* Oakland, CA, USA.

[25] Statista Research Department. 2016. Internet of Things - number of connected devices worldwide 2015-2025. White Paper.

[26] Stratosphere. 2015. Stratosphere Laboratory Datasets. https://www.stratosphereips.org/datasets-overview Retrieved March 13, 2020.

[27] Mahbod Tavallaee et al. 2009. A Detailed Analysis of the KDD CUP 99 Data Set. In *Proc. IEEE CISDA'09.* Ottawa, Ontario, Canada, 53–58.

[28] L.J.P. van der Maaten and G.E. Hinton. 2008. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research* 9 (Nov. 2008), 2579–2605.

[29] J. Wang et al. 2015. Statistical Traffic Anomaly Detection in Time-Varying Communication Networks. *IEEE Trans. Control Netw. Syst.* 2, 2 (June 2015).

[30] M. Wang et al. 2018. Machine Learning for Networking: Workflow, Advances and Opportunities. *IEEE Network* 32, 2 (Mar. 2018), 92–99.

[31] W. Xu et al. 2018. Internet of Vehicles in Big Data Era. *IEEE/CAA Journal of Automatica Sinica* 5, 1 (Jan. 2018), 19–35.

[32] Houssam Zenati et al. 2018. Adversarially Learned Anomaly Detection. In *Proc. IEEE ICDM 2018.* Singapore, 727–736.

[33] Shuangfei Zhai et al. 2016. Deep Structured Energy Based Models for Anomaly Detection. In *Proc. ICML 2016.* New York, NY, USA, 1100–1109.

[34] Mingyi Zhu et al. 2018. Network Anomaly Detection and Identification Based on Deep Learning Methods. In *CLOUD 2018.* Seattle, WA, USA, 219–234.

[35] Bo Zong et al. 2018. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In *Proc. ICLR 2018.* Vancouver, BC, Canada.

A ARCHITECTURE OF MODELS

Table 4: ALAD design and parameters

Operation	Units	Non Linearity	Batch Norm.	Drop.
$E(x)$				
Dense	1536	LReLU	×	0.0
Dense	1024	LReLU	×	0.0
Dense	512	LReLU	×	0.0
Dense	384	LReLU	×	0.0
Dense	256	LReLU	×	0.0
Dense	128	LReLU	×	0.0
Dense	64	LReLU	×	0.0
Dense	32	LReLU	×	0.0
Dense	16	LReLU	×	0.0
Dense	8	None	×	0.0
$G(z)$				
Dense	16	ReLU	×	0.0
Dense	32	ReLU	×	0.0
Dense	64	ReLU	×	0.0
Dense	128	ReLU	×	0.0
Dense	256	ReLU	×	0.0
Dense	384	ReLU	×	0.0
Dense	512	ReLU	×	0.0
Dense	1024	ReLU	×	0.0
Dense	1536	ReLU	×	0.0
Dense	#Feats	Non	×	0.0
$D_{xz}(x, z)$				
Only on x				
Dense	128	LReLU	✓	0.0
Only on z				
Dense	128	LReLU	×	0.5
Concatenate outputs				
Dense	128	LReLU	×	0.5
Dense	1	Sigmoid	×	0.0
$D_{xx}(x, \hat{x})$				
Concatenate x and $\hat{x}$				
Dense	128	LReLU	×	0.2
Dense	64	LReLU	×	0.2
Dense	1	Sigmoid	×	0.0
$D_{zz}(z, \hat{z})$				
Concatenate z and $\hat{z}$				
Dense	64	LReLU	×	0.2
Dense	32	LReLU	×	0.2
Dense	1	Sigmoid	×	0.0
Optimizer	Adam ($\alpha = 10^{-5}, \beta_1 = 0.5$)			
Batch size	100 for CICIDS2017 and UNSW-NB15 25 for Stratosphere IPS			
Latent dimension	8			
LReLU slope	0.2			
Weight, bias init.	Xavier initializer, Constant (0)			

Table 5: AnoGAN design and parameters

Operation	**Units**	**Non Linearity**	**Dropout**
$G(z)$			
Dense	16	ReLU	0.0
Dense	32	ReLU	0.0
Dense	64	ReLU	0.0
Dense	128	ReLU	0.0
Dense	256	ReLU	0.0
Dense	384	ReLU	0.0
Dense	512	ReLU	0.0
Dense	1024	ReLU	0.0
Dense	1536	ReLU	0.0
Dense	#Features	Non	0.0
$D(x)$			
Dense	128	LReLU	0.2
Dense	64	LReLU	0.2
Dense	1	Sigmoid	0.0
Optimizer	Adam ($\alpha = 10^{-5}, \beta_1 = 0.5$)		
Batch size	100 for CICIDS2017 and UNSW-NB15 25 for Stratosphere IPS		
Latent dimension	8		
LReLU slope	0.2		
Weight, bias init.	Xavier initializer, Constant (0)		