Semi-Supervised Outlier Detection Using a Generative and Adversary Framework

Jindong Gu, Matthias Schubert, Volker Tresp

Abstract**—**In many outlier detection tasks, only training data belonging to one class, i.e., the positive class, is available. The task is then to predict a new data point as belonging either to the positive class or to the negative class, in which case the data point is considered an outlier. For this task, we propose a novel corrupted Generative Adversarial Network (CorGAN). In the adversarial process of training CorGAN, the Generator generates outlier samples for the negative class, and the Discriminator is trained to distinguish the positive training data from the generated negative data. The proposed framework is evaluated using an image dataset and a real-world network intrusion dataset. Our outlier-detection method achieves state-of-the-art performance on both tasks.

Keywords**—**Outlier detection, generative adversary networks, semi-supervised learning.

I. Introduction

THREE main approaches for outlier detection have been discussed in literature [1]. The first one is based on supervised classification and requires labeled data from both the positive class and the negative class. The second one is based on unsupervised learning such as clustering, and models data from one or both classes. The third one uses semi-supervised learning to detects outliers using only data from the positive class. Semi-supervised learning has gained increasing attention in recent years, with One-class classification (OCC) being a popular example. The term ”One-class classification” was introduced in [2]. Other authors use the term Outlier Detection [3], Novelty Detection [4] or Concept Learning [5]. These terms are used interchangeably in this paper, even though they have specific meanings in other works. OCC can be used not only in machine monitoring tasks but also in many others, e.g., Text mining [6], Sentiment Analysis [7] and IT security [8].

OCC is relevant for many industrial applications. Consider image-based monitoring systems for product fault detection where a classifier should detect when the manufacturing process starts to behave abnormally. Images from the positive class are easy to obtain by measuring the normal operations of the machine. However, data with negative labels are typically very limited, or even totally unavailable and a classifier needs to be built only on positive training data.

Although many solutions for OCC have been proposed in the last several years, conducting robust outlier detection in high-dimensional spaces with high performance remains a challenging task. Neural Networks with deep architecture are well known for THEIR ability to manipulate high-dimensional data. They achieve state-of-art results in speech recognition, visual object recognition, object detection as well as many other application domains [9]. OCC solutions based on the strong modeling capacity of deep networks have been proposed in several recent works [10]-[12]. Reference [11] learns a mapping from the data space to a lower-dimensional feature space where the complex distribution is difficult to model accurately. Unfortunately, the restricted density estimation model used in [11] has shown suboptimal performance, if the actual distribution is complex.

A Generative Adversary Networks (GANs) [13] is based on two functional modules. The Discriminator is trained to distinguish the real data from generated data, and the Generator generates data to fool the Discriminator. The Generator thereof is cable of accurately modeling complex distributions. Recently, the adversary process is also explored in outlier detection tasks [14]-[16]. Although GANs have high modeling capacity, usually, the convergence of the training process of GANs cannot be guaranteed in practice. Our proposed model requires no strict convergence of the training process since the Generator is used to generate only outliers instead of high-quality samples from the real data distribution.

The core contribution of this work is to investigate the leverage of the generated negative data from GANs. The basic idea is to stop the training of the GAN modules before convergence, to ensure that the Generator keeps generating samples of the negative class (outliers). Furthermore, we propose a new objective function for training the GAN. With the new objective function, the Generator is able to keep generating various negative samples near the distribution boundary of the positive class.

The next section reviews related work. In Section III, we describe our models and provide a justification for our design choice. We first introduce and analyze an intuitive Early-Stopping based solution which motivates a solution that provides better performance. Section IV presents experiments, analyzes the results and compares the performance with state-of-the-art methods. The last section concludes our paper and describes future work.

II. Related Work

The main approaches for outlier detection can be categorized into five classes [17]. Probabilistic approaches estimate the generative probability density function (pdf) of the data from the positive class. The boundaries of normality in

Jindong Gu is with the Department of Computer Science, The University of Munich and Siemens AG, Germany (e-mail: jindong.gu@siemens.com).

Matthias Schubert is with the Department of Computer Science, The University of Munich, Germany.

Volker Tresp is with the Department of Computer Science, The University of Munich and Siemens AG, Germany.

the data space are defined by the resultant distribution together with a specified threshold, and an unseen sample is tested whether it comes from the same distribution or not. Popular choices for modelling the pdf are Gaussian Mixture Models (GMMs) [18], [19] and Kernel Density Estimators [20]-[22]. Probabilistic approaches require so solve complete density estimation problems in the feature space. If the data in feature space are high dimensional, huge amounts of data are required to fit the model, due to the curse of dimensionality. Another well-known approach, the Reconstruction-based approach, first trains a model to minimize the reconstruction error of training data with positive labels. Then, the trained model assigns an outlier score, the distance between the input representation vector and the output of the model, for each test example. References [23], [24] reviews most of the neural network-based methods. Additionally, a principal component analysis (PCA) can also be used detect outliers [25].

The Distance-based approach, e.g., Nearest neighbour-based methods [26], [27] and Clustering-based methods [28], [29], avoids estimating the pdf explicitly, but it requires a well-defined distance/similarity measure, which is especially difficult to define in high-dimensional space. Another approach is domain-based, which creates the boundary based on the structure of normal data without considering the density of the positive class. One-class SVM [30] and Support vector data description (SVDD) [31] are two basic examples. However, the choice of an appropriate kernel function is not easy, and the search for n appropriate one determines the computational cost. Moreover, the hyperparameters that control the tightness of the boundary are also difficult to select. Lastly, Information-theoretic approaches try to distinguish normal data from outliers by computing the information content of the dataset using an information measure. Similarly, the selection of appropriate information-theoretic measure is challenging.

The approaches described above learn from available positive samples only. Approaches that learn from both target samples and artificial outliers are have also been investigated. Reference [32], [33] generate outliers subject to a predefined distribution. The strong assumptions about the outlier data distribution in these approaches may be violated in real datasets [34]. Reference [35] proposes a method for generating artificial outliers, uniformly distributed in a hypersphere. However, in high-dimensional data space, their proposed technique is not feasible anymore because it is though to get a confident estimate of the target volume due to the large difference in volume of the target and outlier class. Reference [36] extends the given dataset by generating outlier examples distributed around the positive class. The approach first finds boundary points explicitly using SVM, which is computationally expensive. Then it generates negative examples only around positive class data using a distance measure, which is infeasible in high-dimensional spaces.

The recently proposed generative adversary networks (GANs) are able to model complex distributions, and are thus suitable for complex OCC. AnoGAN [14] models the normal data distribution using the Generator and identify the outliers by finding the minimal reconstruction error from the latent space. The search in latent space is computationally expensive. To avoid the costly search process, [16] computes the corresponding latent representation directly using the jointly trained encoder. Reference [15] proposes to generate both positive samples and negative sample for open-category classification. The generation and categorization process is based on a derivative-free optimization. Our proposed CorGan keeps generating negative examples around the positive class. Moreover, the model requires no strict convergence of training process.

III. Generating Outliers Using GANs

An overview of our framework is shown in Fig. 1. The outlier detection process consists of three steps, namely, training the GAN, training the classifier and training the detection process. In this work, we focus on adapting the Generator in GANs to generate desired outlier samples. We first describe three desired properties of the generated samples:

P1) *The generated samples belong to the negative class.*

P2) *The generated samples distribute around the subspace of the positive class (i.e. near the boundary between the positive and the negative classes).*

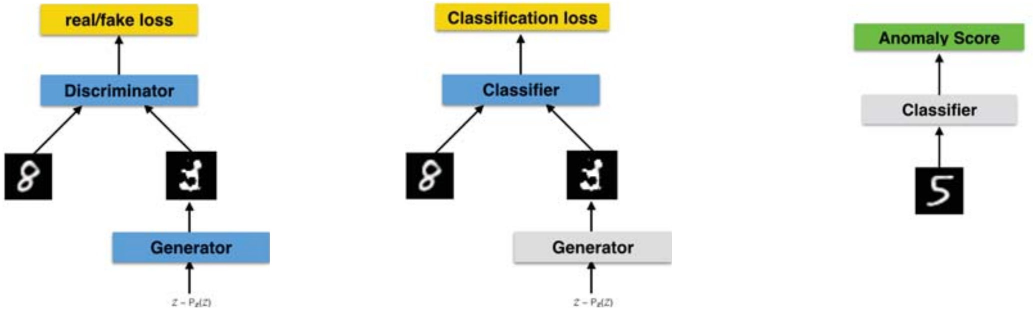
P3) *The generated samples are mutually separated in data space.*

The original Generative Adversary Network (GAN) is a framework for training generative models via an adversarial process [13]. The framework consists of two components, a generative model (Generator G) and a discriminative model (Discriminator D). The framework has been analyzed from various viewpoints, based on its Energy function, Class probability estimation, Divergence Minimisation, Ratio Matching and Moment Matching [37]-[40]. All annotations used in this paper are listed in Appendix V-A.

From the perspective of Divergence Minimisation, the Discriminator in GANs provides information about the relationship between the generative model distribution $q_\theta(x)$ and the data distribution $p(x)$, e.g., divergence or the density ratio. The Generator minimizes the divergence of the two distributions. The divergence of G reaches the minimum when density ratio is 1, namely $q_\theta(x) = p(x)$.

From the perspective of Class probability estimation, the D estimates the probability that a sample came from the training data rather than the Generator. The D is trained to distinguish samples in training data from generated samples by assigning a high probability to the former and a low probability to the latter. Contrarily, the objective of G is to maximize the probability of D making a mistake. Thus, after convergence, the found solution can be interpreted as a Nash-equilibrium.

In case of the convergence, the G is capable of generating samples from data distribution $p(x)$. The D always outputs the same probability values for both training samples and generated samples, and it cannot distinguish them anymore. Note that, after complete convergence, the generated samples of this original GAN do not satisfy the *property 1* because they belong to the positive class (i.e. subject to $p(x)$). The idea pursued in this paper is to stop training GAN earlier before convergence.



(a) Step1: Training the Generator and the Discriminator of a GAN

(b) Step2: Training a Classifier on the real positve data and the generated negative data

(c) Step3: Detecting outliers using the trained Classifier

Fig. 1 Overview of our approach: A framework to detect unseen outliers is shown, which is composed of three steps. Blue blocks mean that the parameters of the model will be updated, and grey blocks mean that the model is only used for inferences without updating their parameters. Yellow blocks are loss functions of the training process. The green block indicates the anomaly scores of the test sample

A. *Early Stopping in GANs*

In early training epochs (i.e., before convergence), the G is not able to generate data that follows the data distribution $p(x)$. In other words, the distributions generated by from G are different from the distribution of training datasets. After stopping the training process of GAN, we generate negative samples using the G and build another traditional binary classifier on the training data with positive labels and the generated data with negative labels. Since $q_\theta(x) \neq p(x)$, the classifier can separate them well.

We illustrate and analyze this solution on the MNIST dataset. The task setting follows [16], where 10 different datasets are generated from MNIST by successively making each digit class an anomaly and treating the remaining 9 digits as positive examples. The training set consists of 80% of the normal data, and the test set consists of the remaining 20% of normal data and all of the anomalous data. The model is trained only with normal data and tested with both normal and anomalous data. The performance of the model is evaluated with the measure of area under the precision-recall curve (AUPRC) since the dataset is imbalanced. The classifier is trained for 50 epochs. The architectures and hyperparameters of GAN and the classifier are listed in Appendix V-B.

The performance of this solution is shown in Fig. 2. In Fig. 2a, the x-axis indicates the stopped epoch. The ten horizontal rows correspond to the 10 aforementioned datasets, and the digit in the right side of each row means the anomaly class. In each row, the heights of the bars are the AUPRC scores. As shown in all ten datasets, the later we stop training the GAN, the more the performance decreases. In Fig. 2b, the generated images from different training epochs are shown. In the later epochs, some generated images can be categorized as positive samples. After several training epochs of the GAN, parts of the generated samples are from data distribution. As a result for the classifier in step 2, the number of false negative samples increase on the test dataset. The performance drops if training is terminated later.

The performance of the Early-Stopping based solution strongly depends on the stopped epoch. Since no negative sample is available in the task, it is difficult to define a meaningful stopping criterion. In the following subsection, we introduce a novel CorGAN, which is able to keep generating samples during the training process that satisfy the three desired properties.

B. *CorGANs Model*

In this subsection, we introduce the new objective of CorGAN and justify the design choice. We will show how the new objective updates the Generator so that it can keep generating samples with the desired properties. The two component of CorGAN are described as follows:

As in the original GAN, the Discriminator D in CorGAN maps the input (i.e. the training samples or the generated samples) to a single scalar, which represents the probability that the input came from the training datasets instead of the Generator G. The target value of the D is 1 for the samples from the training dataset and 0 for samples generated by the G. The D is trained to minimize the cost V(D):

$$\begin{aligned}\max_D V(D) = \mathbb{E}_{x\sim p(x)}[-\log(D(x))] \\ + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[-\log(1 - D(G(\boldsymbol{z}, \theta)))]\end{aligned} \tag{1}$$

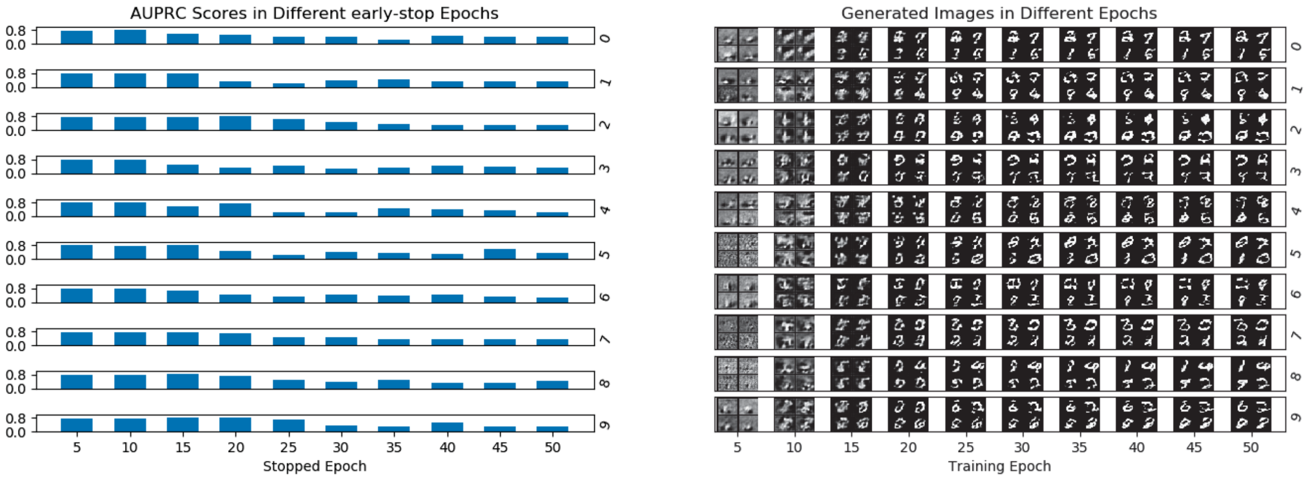
The objective of the Discriminator specified of the CorGAN is the same as that in the original GAN. For the fixed G, the optimal Discriminator [13] is

$$D_G^* = \frac{p(x)}{p(x) + q_\theta(x)} \tag{2}$$

The objective of the Generator in the CorGAN aims to the minimize the following function:

$$\begin{aligned}&\min_G V(G) = \mathbb{E}_{x\sim p(x)}[\log(D(x))] \\ &+ \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}, \theta)))] - \\ &\lambda \log(\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[D(G(\boldsymbol{z}, \theta)) - \mathbb{E}_{\boldsymbol{z'}\sim p_z(\boldsymbol{z})}[D(G(\boldsymbol{z'}, \theta))]])\end{aligned} \tag{3}$$

where the first two terms are the same as in the original GAN, the third term is a penalty term that specifies A SECOND goal for the Generator (maximizing the distance between two



(a) The performance in different stopped epochs of GAN

(b) The generated images in different training epochs of GAN

Fig. 2 The performance of the ealry-stopping solution and the images generated by the GAN model

generated examples), and λ is a hyperparameter. The penalty term aims only to push generated samples away from each other. The hyper-parameter will not have a great impact on the final detection performance. Next, we analyze how the three aforementioned properties are satisfied by the generated sample with the specified objective function.

In case of the optimal Discriminator, the objective function of the Generator can be reformulated as follows, by substituting (2) into (3):

$$\begin{aligned}
\min_G V(G) &= \mathbb{E}_{x\sim p}[\log(D^*_G(x))] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1-D^*_G(z))] \\
&\quad - \lambda \log(\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[D^*_G(z) - \mathbb{E}_{\boldsymbol{z'}\sim p_z(\boldsymbol{z})}[D^*_G(z')]]) \\
&= \mathbb{E}_{x\sim p}[\log(D^*_G(x))] + \mathbb{E}_{\boldsymbol{x}\sim q_\theta}[\log(1-D^*_G(x))] \\
&\quad - \lambda \log(\mathbb{E}_{\boldsymbol{x}\sim q_\theta}[D^*_G(x) - \mathbb{E}_{\boldsymbol{x'}\sim q_\theta}[D^*_G(x')]]) \\
&= \mathbb{E}_{x\sim p}[\log(\frac{p(x)}{p(x)+q_\theta(x)})] + \mathbb{E}_{\boldsymbol{x}\sim q_\theta}[\log(1-\frac{p(x)}{p(x)+q_\theta(x)})] \\
&\quad - \lambda \log(\mathbb{E}_{\boldsymbol{x}\sim q_\theta}[\frac{p(x)}{p(x)+q_\theta(x)} - \mathbb{E}_{\boldsymbol{x'}\sim q_\theta}[\frac{p(x)}{p(x)+q_\theta(x)}]]) \\
&= -\log(4) + KL(p||\frac{p+q_\theta}{2}) + KL(q_\theta||\frac{p+q_\theta}{2}) \\
&\quad - \lambda \log(\mathbb{E}_{\boldsymbol{x}\sim q_\theta}[\frac{p(x)}{p(x)+q_\theta(x)} - \mathbb{E}_{\boldsymbol{x'}\sim q_\theta}[\frac{p(x)}{p(x)+q_\theta(x)}]]) \\
&= -\log(4) + 2JSD(p||q_\theta) \\
&\quad - \lambda \log(\mathbb{E}_{\boldsymbol{x}\sim q_\theta}[\frac{p(x)}{p(x)+q_\theta(x)} - \mathbb{E}_{\boldsymbol{x'}\sim q_\theta}[\frac{p(x)}{p(x)+q_\theta(x)}]])
\end{aligned} \tag{4}$$

where p is the real data distribution $p(x)$, q_θ is the generated data distribution $q_\theta(x)$. When $p(x) = q_\theta(x)$, the loss function $V(G)$ described above does not reach the minimum. In other words, with the new objective, the CorGAN remains at the corrupted status, which ensures that the training process is not able to converge completely. In the case of unconvergence, all the generated samples belong to the negative class, which is the desired *property 1*.

The first two terms, reformulated as $-\log(4) + 2JSD(p||q_\theta)$, minimize the Jensen–Shannon divergence distance between the real data distribution and the generative model distribution. The error in the loss function pushes the generated samples close to the positive class. Since the generated samples stay out of the real data distribution, they all ”live” near the boundary between the positive class and negative class, such that the *property 2* is satisfied.

The third term describes the variance of the discrimination outputs for the generated samples. The loss function $V(G)$ minimizes the negative of the variance value, and thus maximizes the variance value. To reach such a goal, the generator is updated to generate various samples, even from similar vectors in latent space. The generated samples stay far away from each other, which satisfies the *property 3*. There is a tradeoff between the *property 2* and the *property 3* in our CorGAN. In the empirical experiments, we show that the model has more stable performance if we attach more importance to the *property 3*. The variance value is approximated by the Discrimination output values of a batch of generated samples.

In the proposed CorGAN, the D still provides information about divergence distance between $q_\theta(x)$ and $p(x)$. Since the training process does not converge, the Discriminator is always able to distinguish real samples of the positive class and generated samples of the negative class relatively well. We did not directly take the Discriminator as an outlier detector because it is still confused by the real and generated samples to some degree. Hence, we build another classifier that is optimized directly on the binary classification task (see Step 2 in Fig. 1). The discriminator can be taken as an initialized point for this classifier in practice.

IV. Experiments and Analysis

In the experiments, we first compare our model with other related models on an artificial abnormal image detection task.

We analyze the sensitivity of the hyperparameters and the efficiency of the detection process in our model. Additionally, we show the performance of our model on a real-world dataset, called KDD99 dataset. We compare our model with the following similar GAN-based anomaly detection models:

AnoGAN: AnoGAN [14] is first trained until convergence. The Generator thereof models the normal data distribution $p(x)$. In the anomaly detection process, given a test sample $\boldsymbol{x}$, the anomaly score depends on the reconstruction error, which is comprised of a Residual Loss and a Discrimination Loss. The Residual Loss $L_R(x) = \sum |\boldsymbol{x} - G(\theta, \boldsymbol{z})|$ describes the difference between the test sample and a generated sample from a variable $\boldsymbol{z}$ in latent space. The Discrimination Loss is the Feature Dismatching in Discriminator $L_{D(x)_{FM}} = \sum |f(x) - f(G(\theta, \boldsymbol{z}))|$ or the Cross-entropy Loss $L_{D(x)_\sigma}$. The final anomaly score is: $A(x) = \min_{\boldsymbol{z}}(1 - \lambda)L_{R(x)} + \lambda L_{D(x)}$.

Efficient BiGAN: [16] improves the AnoGAN by jointly training an encoder that reconstructs the corresponding latent code from the generated image $\boldsymbol{z} = E(G(\theta, \boldsymbol{z}))$. In the detection process, while AnoGAN searches for a latent code that minimizes the reconstruction error, the efficient BiGAN compute the reconstruction error directly using the latent code $\boldsymbol{z}$ from the Encoder.

A. Abnormal Image Detection on MNIST

Following the experimental setup in [16], [24], we construct the same datasets as in Section III-A using THE MNIST dataset. 10 different datasets are constructed by successively making each digit class an anomaly and treating the remaining 9 digits as positive examples. Only positive samples were used for training; the test samples contain both positive examples and negative ones.

We evaluate our models with the area under the precision-recall curve (AUPRC score). The score takes all possible thresholds into consideration, providing an overall performance of the evaluated models. The performance of our model and the compared models is shown in Fig. 3.

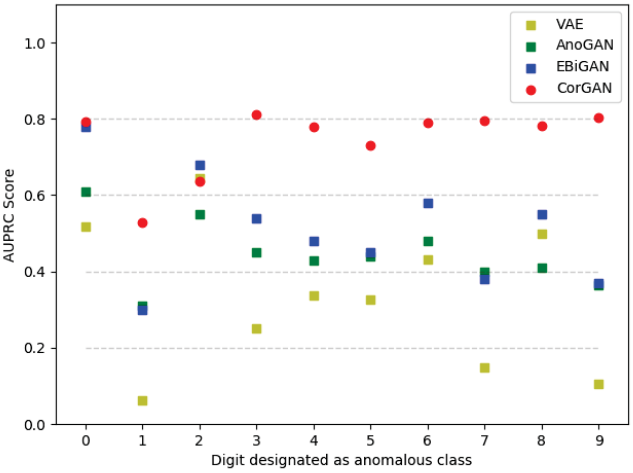


Fig. 3 The performance of CorGAN model other models. The scores of AnoGAN and EBigan are obtained from [16], the score of VAE-based method is obtained from [24]

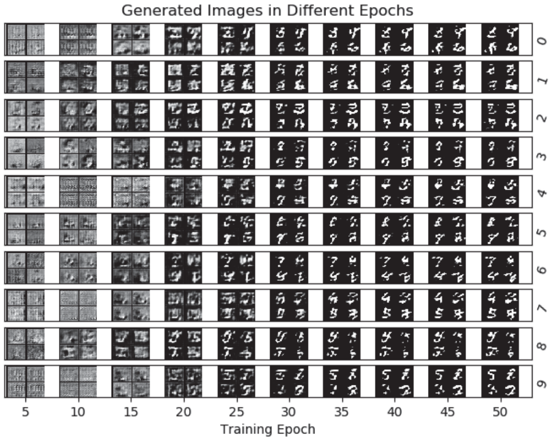


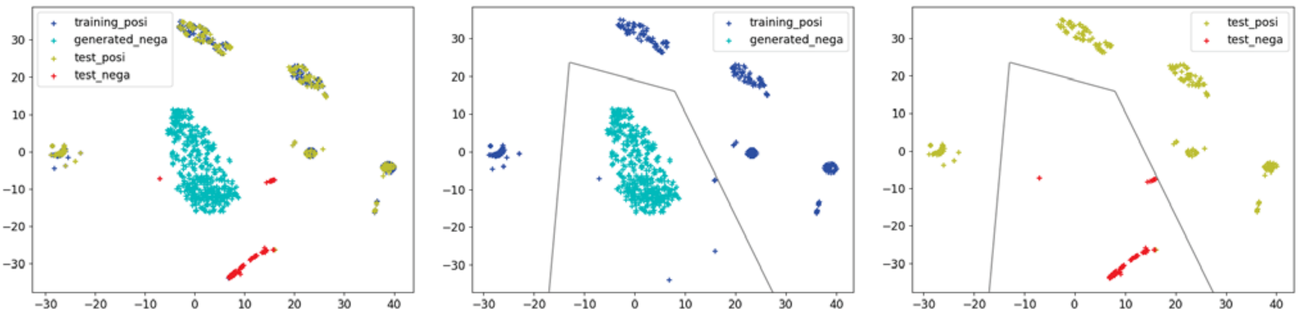
Fig. 4 The generated images of CorGAN in different epochs. They belong to the negative class, lie near the positive class and differ from each other.

Our CorGAN model significantly outperforms AnoGAN (Feature Matching), EBiGN(Feature Matching), Variational Autoencoder-based model. Reference [24] shows that the Variational Autoencoder-based method outperforms the traditional reconstruction-based methods such as PCA, kernel PCA, and Autoencoder Reconstruction Error-based method. Hence, we only list the performance of VAE-based methods in the figure.

Efficiency: [15] proposes generating both positive samples and negative sample for open-category classification using GANs. The generation and categorization process is based on a derivative-free optimization, which is inefficient in high-dimensional data space. AnoGAN [14] requires a SGD-based optimization to compute the anomaly score in the detection process. Reference [16] avoids the optimization process by training an extra Encoder. It reconstructs the desired latent code instead of searching via the optimization, especially, $D(G(E(\boldsymbol{x})))$, which runs approximately 800x faster than the AnoGAN. Our framework contains a classifier C. The anomaly score of a test sample $\boldsymbol{x}$ can be directly computed via a single forward inference of the Classifier $C(\boldsymbol{x})$. Theoretically, the detection process of our framework runs 3x faster than the improved AnoGAN (approximately 2400x faster than the original AnoGAN).

Sensitivity Analysis: The performance of the CorGAN is more robust and not as sensitive as that of our early-stopping solution. The reason is that all the generated images of CorGAN belong to the negative class (see Fig. 4). The proposed CorGAN model only slightly outperforms the best performance in the early-stopping solution. To show a clear comparison with other models, we only show the performance of the CorGAN model in Fig. 3.

The hyper-parameter λ in (3) specifies the tradeoff between *property 2* and *property 3* of the generated samples. We empirically test the $\lambda \in \{5, 2, 1, 0.5, 0.2\}$. The results show that the λ with different values have almost no impact on the final performance, and the large λ leads to more robust performance. The more penalty on the variance term ensures that the generated samples are not too close to the positive class. In all the experiments, we report the results with $\lambda = 1$.



(a) Visualizing the training samples, the generated samples and the test samples using t-SNE

(b) Building a Classifier on the training data and the generated data

(c) Detecting outliers using the classification boundary of the built Classifier

Fig. 5 Visualization of outlier detection process in CorGAN framework on the real-world KDDCUP99 dataset

B. Network Intrusion Detection on KDDCUP99

In this experiment, we show that our CorGAN can also perform well on high-dimensional non-image data. We evaluated our method on a real-world network intrusion dataset, called KDDCUP99 10 percent dataset [41]. The experimental setup followed the work [42], [10]. More concretely, 50% of the whole dataset is randomly sampled for testing. For the remaining 50% of the dataset, all anomalous samples were removed. Only data samples from the normal class were used for training models.

The evaluation measure used in this experiment are *precision*, *recall*, and *F1-score* (anomaly as the positive class in confusion matrix). The proportion of outliers in the original dataset is 20%. Hence, the threshold is chosen so that the 20% of samples with the highest anomaly scores are classified as anomalies.

The CorGAN used multi-layer perceptrons (MLPs) in its different function modules. The used Classifier is a Support Vector Machine with RBF kernel. For a fair comparison, the architectures and parameters of our CorGAN model, AnoGAN [14] and efficient BiGAN [16] are set the same. The more details can be found in Appendix V-B. The performance of our model and other models is shown in the table I. Our CorGAN model achieves state-of-the-art performance. The performance of CorGAN is slightly better than the Early-Stopping solution.

TABLE I
PERFORMANCE ON THE KDDCUP99 DATASET. VALUES FOR OC-SVM, DSEBM, DAGMM VALUES WERE OBTAINED FROM [42], [10]. VALUES FOR ANOGAN AND EFFICIENT BIGAN WERE OBTAINED FROM [16]. VALUES FOR GAN$_{early-stopping}$ AND OUR CORGAN MODEL ARE DERIVED FROM 10 RUNS

Model	Precision	Recall	F1-Score
OC-SVM	0.7457	0.8523	0.7954
DSEBM-r	0.8521	0.6472	0.7328
DSEBM-e	0.8619	0.6446	0.7399
DAGMM-NVI	0.9290	0.9447	0.9368
DAGMM	0.9297	0.9442	0.9369
AnoGAN$_{FM}$	0.8786 ± 0.0340	0.8297 ± 0.0345	0.8865 ± 0.0343
AnoGAN$_{\sigma}$	0.7790 ± 0.1247	0.7914 ± 0.1194	0.7852 ± 0.1181
EBiGAN$_{FM}$	0.8698 ± 0.1133	0.9523 ± 0.0224	0.9058 ± 0.0688
EBiGAN$_{\sigma}$	0.9200 ± 0.0740	0.9582 ± 0.0104	0.9372 ± 0.0440
GAN$_{early-stopping}$	0.9637± 0.0149	0.9791± 0.0152	0.9713± 0.0152
Our CorGAN	**0.9648**± **0.0135**	**0.9802**± **0.0136**	**0.9724**± **0.0136**

To gain more insight into the proposed framework, we visualized the outlier detection process in Fig. 5. We randomly chose 500 samples from the training dataset, the generated dataset, and the test dataset respectively. Using the t-SNE technique [43], Fig. 5a visualizes the training samples (blue points), the generated negative samples (cyan points), the positive samples (yellow points) and the negative ones (red points) in the test dataset. Since the positive samples in training data and the positive samples in test data are from the same distribution, the clusters of blue points overlap the ones of green points. Each cluster means one type of IP connections. The number of training samples and generated samples are the same; the generated negative sample cover more space although it only forms one cluster.

Fig. 5b shows the classification boundary of the classifier built on the positive training samples and the generated negative samples. The built classifier is used to distinguish the normal samples and the outliers in the test dataset, which is shown in Fig. 5c. As shown in the figures, the distribution of generated samples is different from that of the outliers in the test dataset. The generated samples are auxiliary to build a tight boundary of the normal distribution from the training data since they lie near the positive class in data space.

V. Conclusion

This work investigated the Generative Adversary Framework to detect outliers by generating outlier samples. An intuitive solution is first proposed and analyzed. Due to the limitation of the high sensitivity of the first solution, we propose, as a second solution, the CorGAN model. With a new objective function, the Generator of CorGAN is able to generate samples with the desired properties. The model is empirically evaluated on an image dataset and a high-dimensional non-image dataset. The results show that our model shows a competitive performance. The outlier detection tasks in the computer vision community often involve real-world high-resolution images. The outliers generated by our model only cover a small space of the outlier distribution. We leave the further exploration in future work.

Appendix

A. Annotations in This Paper

The anotations used in this paper are listed in the following table.

TABLE II
The Annotations Used in This Paper

$p, p(x)$	the real data distribution
$\boldsymbol{x}$	a variable subjective to $p(x)$
$\boldsymbol{x'}$	another variable subjective to $p(x)$
$p_z(\boldsymbol{z})$	the standard normal distribution
$\boldsymbol{z}$	a variable subjective to $p_z(\boldsymbol{z})$
$\boldsymbol{z'}$	another variable subjective to $p_z(\boldsymbol{z})$
$q_\theta, q_\theta(x)$	the generative model distribution
$E(x)$	the Encoder reconstructing latent code
$C(x)$	the Classifier in CorGAN framework

TABLE III
Architecture and Hyperparameters in CorGAN

Operation	Kernel	Strides	FMs/Units	BN?	NonLinear
$G(\theta, \boldsymbol{z})$					
Dense			1024	√	ReLU
Dense			$7 * 7 * 128$	√	ReLU
Trans Conv	4×4	2×2	64	√	ReLU
Trans Conv	4×4	2×2	1	×	Tanh
$D(\boldsymbol{x})$					
Convolution	4×4	2×2	64	×	Leaky ReLU
Convolution	4×4	2×2	64	√	Leaky ReLU
Dense			1024	√	Leaky ReLU
Dense			1	×	Sigmoid
$C(\boldsymbol{x}) = D(\boldsymbol{x})$					
Optimizer		Adam($\alpha = 10^{-5}, \beta 1 = 0.5$)			
Batch size		100			
Latent dimension		200			
GAN Epochs		10			
Classifier Epochs		10			
Leaky ReLU slope		0.1			
Weight		Isotropic gaussian ($\mu = 0, \sigma = 0.02$)			
Bias initialization		Constant(0)			

TABLE IV
Architecture and Hyperparameters in CorGAN

Operation	Units	NonLinear	Dropout
$G(\theta, \boldsymbol{z})$			
Dense	64	ReLU	0.0
Dense	128	ReLU	0.0
Dense	121	ReLU	0.0
$D(\boldsymbol{x})$			
Dense	256	Leaky ReLU	0.2
Dense	128	Leaky ReLU	0.2
Dense	128	Leaky ReLU	0.2
Dense	1	Sigmoid	0.0
$C(\boldsymbol{x})$ = SVM with RBF kernel			
Optimizer		Adam($\alpha = 10^{-5}, \beta 1 = 0.5$)	
Batch size		50	
Latent dimension		32	
GAN Epochs		1	
Leaky ReLU slope		0.1	
Weight, Bias initialization		Xavier Initializer, Constant(0)	

B. Architecture of GAN Model and the Classifier

The architectures and hyperparameters of AnoGAN and Efficient BiGAN can be found in [16]. All the GAN-based models in this paper used almost the same architectures and hyperparameters. In the abnormal image detection task on the MNIST dataset, the pixels of images were scaled to be in the range [-1,1] for the training and the test. The Table III lists the architectures and hyperparameters of our CorGAN model. Table IV lists the architectures and hyperparameters of our CorGAN model on the KDD99 dataset.

References

[1] V. Hodge and J. Austin, “A survey of outlier detection methodologies,” *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[2] M. M. Moya, M. W. Koch, and L. D. Hostetler, “One-class classifier networks for target recognition applications,” Sandia National Labs., Albuquerque, NM (United States), Tech. Rep., 1993.

[3] G. Ritter and M. T. Gallegos, “Outliers in statistical pattern recognition and an application to automatic chromosome classification,” *Pattern Recognition Letters*, vol. 18, no. 6, pp. 525–539, 1997.

[4] C. M. Bishop, “Novelty detection and neural network validation,” *IEE Proceedings-Vision, Image and Signal processing*, vol. 141, no. 4, pp. 217–222, 1994.

[5] N. Japkowicz, “Concept-learning in the absence of counter-examples: an autoassociation-based approach to classification,” Ph.D. dissertation, Rutgers, The State University of New Jersey, 1999.

[6] S. Basu, M. Bilenko, and R. J. Mooney, “A probabilistic framework for semi-supervised clustering,” in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 59–68.

[7] B. Agarwal *et al.*, “One-class support vector machine for sentiment analysis of movie review documents,” *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 9, no. 12, pp. 2458–2461, 2015.

[8] A. Lakhina, M. Crovella, and C. Diot, “Mining anomalies using traffic feature distributions,” in *ACM SIGCOMM Computer Communication Review*, vol. 35. ACM, 2005, pp. 217–228.

[9] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[10] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, “Deep structured energy based models for anomaly detection,” *arXiv preprint arXiv:1605.07717*, 2016.

[11] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *International conference on machine learning*, 2016, pp. 478–487.

[12] X. Yang, K. Huang, J. Y. Goulermas, and R. Zhang, “Joint learning of unsupervised dimensionality reduction and gaussian mixture model,” *Neural Processing Letters*, vol. 45, no. 3, pp. 791–806, 2017.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[14] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, “Unsupervised anomaly detection with generative adversarial networks to guide marker discovery,” in *International Conference on Information Processing in Medical Imaging*. Springer, 2017, pp. 146–157.

[15] Y. Yu, W.-Y. Qu, N. Li, and Z. Guo, “Open-category classification by adversarial sample generation,” *arXiv preprint arXiv:1705.08722*, 2017.

[16] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, “Efficient gan-based anomaly detection,” *arXiv preprint arXiv:1802.06222*, 2018.

[17] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, “A review of novelty detection,” *Signal Processing*, vol. 99, pp. 215–249, 2014.

[18] B. Lindsay, G. L. Mclachlan, K. E. Basford, and M. Dekker, “Mixture models: Inference and applications to clustering,” *Journal of the American Statistical Association*, vol. 84, no. 405, p. 337, 1989.

[19] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[20] E. Parzen, “On estimation of a probability density function and mode,” *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[21] P. Vincent and Y. Bengio, “Manifold parzen windows,” in *Advances in neural information processing systems*, 2003, pp. 849–856.

[22] Y. Bengio, H. Larochelle, and P. Vincent, “Non-local manifold parzen windows,” in *Advances in neural information processing systems*, 2006, pp. 115–122.

[23] M. Markou and S. Singh, “Novelty detection: a review—part 2:: neural network based approaches,” *Signal processing*, vol. 83, no. 12, pp. 2499–2521, 2003.

[24] J. An and S. Cho, “Variational autoencoder based anomaly detection using reconstruction probability,” *Special Lecture on IE*, vol. 2, pp. 1–18, 2015.

[25] D. M. Tax and K.-R. Müller, "Feature extraction for one-class classification," *Lecture notes in computer science*, pp. 342–349, 2003.

[26] S. D. Bay and M. Schwabacher, "Mining distance-based outliers in near linear time with randomization and a simple pruning rule," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 29–38.

[27] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29. ACM, 2000, pp. 93–104.

[28] D. Barbará, Y. Li, and J. Couto, "Coolcat: an entropy-based algorithm for categorical clustering," in *Proceedings of the eleventh international conference on Information and knowledge management*. ACM, 2002, pp. 582–589.

[29] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognition Letters*, vol. 24, no. 9, pp. 1641–1650, 2003.

[30] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.

[31] D. M. Tax and R. P. Duin, "Support vector domain description," *Pattern recognition letters*, vol. 20, no. 11, pp. 1191–1199, 1999.

[32] K. Hempstalk, E. Frank, and I. H. Witten, "One-class classification by combining density and class probability estimation," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 505–519.

[33] W. Fan, M. Miller, S. Stolfo, W. Lee, and P. Chan, "Using artificial anomalies to detect unknown and known network intrusions," *Knowledge and Information Systems*, vol. 6, no. 5, pp. 507–527, 2004.

[34] N. Abe, B. Zadrozny, and J. Langford, "Outlier detection by active learning," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 504–509.

[35] D. M. Tax and R. P. Duin, "Uniform object generation for optimizing one-class classifiers," *Journal of machine learning research*, vol. 2, no. Dec, pp. 155–173, 2001.

[36] A. Bánhalmi, A. Kocsor, and R. Busa-Fekete, "Counter-example generation-based one-class classification," in *ECML*. Springer, 2007, pp. 543–550.

[37] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[38] S. Mohamed and B. Lakshminarayanan, "Learning in implicit generative models," *arXiv preprint arXiv:1610.03483*, 2016.

[39] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Information Processing Systems*, 2016, pp. 271–279.

[40] M. Uehara, I. Sato, M. Suzuki, K. Nakayama, and Y. Matsuo, "Generative adversarial nets from a density ratio estimation perspective," *arXiv preprint arXiv:1610.02920*, 2016.

[41] M. Lichman *et al.*, "Uci machine learning repository," 2013.

[42] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," 2018.

[43] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.