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.

International Scholarly and Scientific Research & Innovation 12(10) 2018 891 ISNI:0000000091950263
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 solving 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 an 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 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 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 $q_{θ}(x)$ 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 $q_{θ}(x) = p(x)$.

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 I 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. 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 \( q_0(x) \neq p(x) \), 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 top 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 \( V(D) \):

\[
\max_D V(D) = E_{x \sim p(x)}[-\log(D(x))] + E_{z \sim p_z(z)}[-\log(1 - D(G(z, \theta)))]
\]

(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

\[
D_G^* = \frac{p(x)}{p(x) + q_0(x)}
\]

(2)

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

\[
\min_G \mathbb{E}_{z \sim p(z)}[\log(D(z, \theta))]
+ \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z, \theta)))] - \lambda \log(\mathbb{E}_{z \sim p_z(z)}[D(G(z, \theta)) - \mathbb{E}_{z' \sim p_z(z)}[D(G(z', \theta))]])
\]

(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
The generated images in different training epochs of GAN

**Algorithm 1: CorGAN**

1. Initialize $p(x)$ and $q_0(x)$
2. For each epoch $t$
   a. Generate $\hat{x}$ using $p(x)$
   b. Update $q_0(x)$ with $\hat{x}$
   c. Update $p(x)$ with $\hat{x}$
3. Output $q_0(x)$ as the final model

The generated images in different training epochs of GAN

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 \( x \), the anomaly score depends on the reconstruction error, which is comprised of a Residual Loss and a Discrimination Loss. The Residual Loss \( L_R(x) = \sum |x - G(\theta, z)| \) describes the difference between the test sample and a generated sample from a variable \( z \) in latent space. The Discrimination Loss is the Feature Dismatching in Discriminator \( L_D(\theta, \lambda) = \sum |f(x) - f(G(\theta, z))| \) or the Cross-entropy Loss \( L_D(z) \). The final anomaly score is:

\[
A(x) = \min_{z}(1 - \lambda)L_R(x) + \lambda L_D(z).
\]

**Efficient BiGAN:** [16] improves the AnoGAN by jointly training an encoder that reconstructs the corresponding latent code from the generated image \( z = E(G(\theta, z)) \). 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 \( z \) 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 model is obtained from [24]](image)

Our CorGAN model significantly outperforms AnoGAN (Feature Matching), EBiGAN (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, \( D(G(E(x))) \), which runs approximately 800x faster than the AnoGAN. Our framework contains a classifier \( C \). The anomaly score of a test sample \( x \) can be directly computed via a single forward inference of the Classifier \( C(x) \). 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 \( \lambda \) 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 \( \lambda \) with different values have almost no impact on the final performance, and the large \( \lambda \) 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 \( \lambda = 1 \).
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 architectures and parameters of our CorGAN model, AnoGAN [14] and efficient BiGAN [16] are set the same. The more 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.
The architectures and hyperparameters of our CorGAN model. In the abnormal image detection task on the MNIST dataset, the pixels of images were scaled to be in the range [0, 1]. 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 [0, 1]. In the abnormal image detection task on the MNIST dataset, the pixels of images were scaled to be in the range [0, 1].

### TABLE II

<table>
<thead>
<tr>
<th>Operation</th>
<th>Kernel Size</th>
<th>FMs/Units</th>
<th>BN</th>
<th>Non-linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dθ(z)</td>
<td>4 × 4</td>
<td>64</td>
<td>ReLU</td>
<td>Leaky ReLU</td>
</tr>
<tr>
<td>Dθ(z)</td>
<td>4 × 2</td>
<td>64</td>
<td>ReLU</td>
<td>Leaky ReLU</td>
</tr>
<tr>
<td>Classifier</td>
<td>256</td>
<td></td>
<td>ReLU</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>1024</td>
<td></td>
<td>ReLU</td>
<td></td>
</tr>
</tbody>
</table>

### Table IV

<table>
<thead>
<tr>
<th>Table IV: ARCHITECTURE AND HYPERPARAMETERS IN COR-GAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Dθ(z)</td>
</tr>
<tr>
<td>Dθ(z)</td>
</tr>
<tr>
<td>Dθ(z)</td>
</tr>
<tr>
<td>Dθ(z)</td>
</tr>
<tr>
<td>Dθ(z)</td>
</tr>
<tr>
<td>Dθ(z)</td>
</tr>
<tr>
<td>Dθ(z)</td>
</tr>
<tr>
<td>Cθ(z) = Dθ(z)</td>
</tr>
<tr>
<td>Batch size</td>
</tr>
<tr>
<td>Latent dimension</td>
</tr>
<tr>
<td>GAN Epochs</td>
</tr>
<tr>
<td>Classifier Epochs</td>
</tr>
<tr>
<td>Leaky ReLU slope</td>
</tr>
<tr>
<td>Weight, Bias initialization</td>
</tr>
</tbody>
</table>

### 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 [0, 1].

### REFERENCES