Semi-Supervised Outlier Detection Using Generative and Adversary Framework

Anonymous authors
Paper under double-blind review

Abstract

In a conventional binary/multi-class classification task, the decision boundary is supported by data from two or more classes. However, in one-class classification task, only data from one class are available. To build an robust outlier detector using only data from a positive class, we propose a corrupted GAN (CorGAN), a deep convolutional Generative Adversary Network requiring no convergence during training. In the adversarial process of training CorGAN, the Generator is supposed to generate outlier samples for negative class, and the Discriminator as an one-class classifier is trained to distinguish data from training datasets (i.e. positive class) and generated data from the Generator (i.e. negative class). To improve the performance of the Discriminator (one-class classifier), we also propose a lot of techniques to improve the performance of the model. The proposed model outperforms the traditional method PCA + PSVM [Schölkopf et al. (2000)] and the solution based on Autoencoder [Thompson et al. (2002)].

1 Introduction

[Hodge & Austin (2004)] addresses three fundamental approaches detecting outliers. The first approach is unsupervised clustering that determines outliers without prior knowledge of the data; In the second approach, supervised classification requires labelled data from both positive class and negative class; The third addressed approach detect outliers using data only from a positive class via semi-supervised learning. Semi-supervised learning has gained increasing attention in recent years. One-class classification (OCC), as a typical semi-supervised learning technique, is applied to detect outliers using only positive examples from one class. The semi-supervised learning in this paper focuses on OCC technique.

To motivate the importance of OCC, we first introduce you a classic application scenario. In industry, machine monitoring system is used everywhere to detect machine faults. A classifier should be constructed to detect when the machine behaves abnormally. Obviously, the training data for the positive class is easy to obtain by measuring the normal operations of the machine. However, the training data with negative labels are in short, even totally unavailable. In this case, a classifier should be built only on positive training data. This kind of task is known as OCC task. The name “one-class classification” originates from the paper [Moya et al. (1993)]. Other researchers also present similar tasks with other terms such as Outlier Detection [Ritter & Gallegos (1997)], Novelty Detection [Bishop (1994)] or Concept Learning [Japkowicz (1999)]. They are used interchangeably in this paper, even though they have specific meanings in other works. One-class classification can be used not only in machine monitoring task but also in many other domains, e.g. Text mining [Basu et al. (2004)], Sentiment Analysis [Agarwal et al. (2015)] and IT security [Lakhina et al. (2005)].

A lot of solutions have been proposed to solve the one-class classification problem. However, almost none of them shows acceptable performance in high dimension. Neural Network with deep architecture is well known for the ability to manipulate high-dimensional data. It achieves state-of-art result in lots of AI-related task [LeCun et al. (2015)]. This paper applies a neural network with deep architecture in outlier detection task. Generative adversary framework (GAN) composed of a Generator $G$, which can be used to generate outliers and a Discriminator $D$, which can be trained as a binary classifier. The framework is a potential solution to detect outliers though generating counter examples. Usually, the Nash equilibrium of the training process of GANs cannot be guaranteed in practice. Fortunately, our proposed model requires no convergence of the training process, since the
$G$ is used to generate only outliers instead of high-quality images like ones from the training dataset. The proposed deep architecture solution is implemented, analysed and compared to other methods.

The first section introduces the one-class classification problem and a potential solution with deep-architecture neural network. The second section presents the work related to one-class classification problems (i.e. semi-supervised outlier detection). Then, the two basic steps of our solution for one-class classification problem are described in the third section, namely, the training step to optimise model and the detecting step to do inference. Next, the fourth section proposes a technique to break Nash equilibrium so that the $G$ of GAN can keep generating artificial outliers. In addition, this section also proposes several other improved techniques. The fifth section shows experiments, analyses results and compares the performance with that of other methods. Finally, the last section concludes our work and describes future work remain to be further researched.

2 RELATED WORK

Five approaches to solve OCC problem are summarized in [Pimentel et al., (2014)]. Probabilistic approach estimates the generative probability density function (pdf) of the data from positive class. The boundaries of normality in 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. Thereinto, Gaussian Mixture Models (GMMs) [Lindsay et al., (1989); Bishop (2006) and Kernel Density Estimators [Parzen (1962); Vincent & Bengio (2003); Bengio et al. (2006)] have proven to be popular. This approach requires complete density estimation in the feature space. If the data in feature space are high dimensional, huge amounts of data are required to fit the model because of the curse of dimensionality. Only when the data from the target class are large enough, this kind of method can perform well. Another well known approach, Reconstruction-based approach, first train a model minimising 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. [Markou & Singh (2003)] reviews lots of the neural network-based methods. Additionally, PCA can also detect outliers by comparing the example before and after transformation. The reconstruction error approach abandons some information with low variance during reconstruction. However, the abandoned low-variance information has proven to be most informative [Tax & Müllner (2003)].

Additionally, Distance-based approach, e.g. Nearest neighbour-based methods [Bay & Schwabacher (2003); Breuning et al. (2000)] and Clustering-based methods [Barbara et al. (2002); He et al. (2003)], avoids estimating pdf explicitly, but it requires a well-defined distance/similarity measure, which is especially difficult 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 [Schölkopf et al. (2000)] and Support vector data description (SVDD) [Tax & Duin (1999)] are two basic ones. But the choice of an appropriate kernel function is not easy, which determines the computational cost. Moreover, the hyperparameters that control the tightness of the boundary are also difficult to select. Lastly, Information-theoretic approach tries to distinguish normal data and outliers by computing information content of dataset using 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 researched. [Hempstalk et al. (2008); Fan et al. (2004)] generate outlier with a predefined distribution. The strong assumptions about the outlier data distribution in these approaches may be violated in real datasets [Abe et al. (2006)]. [Tax & Duin (2001)] proposes a method for generating artificial outliers, uniformly distributed in a hypersphere. However, in high-dimensional data space, their proposed technique is not feasible any more because it is very hard to get a confident estimate of the target volume due to the large difference in volume of the target and outlier class. [Bánkalmi et al. (2007)] extends dataset by generating outlier examples distributed around positive class. The approach first finds boundary points explicitly using SVM, which is computationally expensive. Then it generates negative examples only around positive class using a distance measure, which causes infeasibility in high-dimensional space. Our proposed CorGan generates negative examples including both ones around the positive class and ones far from the positive class. Moreover, the model requires no explicit distance measure and does not need to find boundary points explicitly.
Neural networks with deep architecture have already been used in OCC task, but mostly in Reconstruction error approaches [Markou & Singh, 2003]. To our knowledge, our proposed CorGAN is the first work to generate outliers for OCC via deep architecture (i.e. Generative Adversary Network). Generative Adversary Nets (GAN) describes a framework, in which a generative model is trained via an adversarial process. A variant of the GAN framework (CatGAN) is applied to solve multi-class classification task in unsupervised or semi-supervised fashion [Springenberg, 2015]. Odena [2016] make a further research about semi-supervised learning using GANs. Schlegl et al. [2017] proposes AnoGAN to apply GAN in Anomaly Detection, which requires the Nash-equilibrium at the end of the training process. Nevertheless, all variants of GAN and it self are known for its unstable training process. Fortunately, the CorGAN proposed in this paper requires no convergence of the training process.

3 OUTLIER DETECTION USING CORGAN

The proposed model and improved techniques can be generalised to various kinds of data. In order to show the performance in high-dimensional space, we illustrate our model on image data. The proposed parametric method consists of two steps:

1. Training Step: Training the CorGAN with improved techniques;
2. Inference Step: Detecting outliers using the resultant \( D \) of the trained CorGAN.

3.1 GENERATIVE ADVERSARY NETWORK

Generative Adversary Network(GAN) is a framework for training generative models via an adversarial process [Goodfellow et al., 2014]. The framework consists of two components, a generative model (Generator \( G \)) and a discriminative model (Discriminator \( D \)). The \( G \) aims to capture the data distribution. The \( D \) estimates the probability that a sample came from the training data rather than Generator. This framework corresponds to a minimax two-player game. In the training procedure, the \( D \) is trained to distinguish samples in training datasets 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. After the Nash-equilibrium of the training process, the output probability of the \( D \) is always 0.5. In case of the convergence, the \( G \) is capable of generating realistic images that have same/similar distribution as in training dataset, and the \( D \) cannot make right discrimination any more. The biggest advantage of this framework is that no Markov chains or unrolled approximate inference networks are required in training and sampling process.

3.2 STEP1: TRAINING THE CORGAN

Architectures of Generator and Discriminator are generally neural network, such as, Multilayer Perceptron, Deep Convolutional Network [LeCun et al., 1989], Convolutional Neural Network Cascade [Springenberg, 2015] and Recurrent Neural Network [Rumelhart et al., 1988]. The Back-Propagation algorithm can be used to train both the generative model and the discriminative model. The architecture applied in proposed CorGAN is shown in Figure [1].

The \( G \) generally starts from prior distribution \( p_z \) (input noise variable \( z \)). In the case of convergent GANs, the \( G \) maps the prior distribution to training data distribution \( p_d \). However, the \( G \) of CorGAN is used to generate outlier examples. Hence, it is supposed to map the prior distribution to outlier data distribution \( G(z; \theta_g) \) instead of training data distribution by avoiding convergence. As usual, the \( D \) maps the input (i.e. training data or generated sample) to a single scalar, which represents the probability that the input came from training datasets instead of \( G \). The target value is \( a_t = 1 \) for input data from training dataset and \( a_o = 0 \) for input data generated by the \( G \). The \( D \), as a binary classifier, is trained to minimize the cost \( V(D) \):

\[
\min_D V(D) = \log(D(G(z)) - a_o) + \log(a_t - D(p_d)) \quad (1)
\]

The objective of the \( G \) of CorGAN tries to fool the \( D \), but not maximise the probability \( D \) making a mistake. Supposing that the new target value is \( a_{new} \in (0, 1) \) instead of 1 in convergent GANs, the new objective of \( G \) of CorGAN is to minimise the cost \( U(G) \):

\[
\min_G U(G) = \log(|a_{new} - D(G(z))|) \quad (2)
\]
The CorGAN model is updated via back-propagation algorithm. Generally, there is only finite datasets available. If the $D$ is overly optimised without updating the $G$, it would result in overfitting problem. To avoid the problem, the $D$ and the $G$ can be updated simultaneously or alternately, e.g., $k$ steps of optimizing $D$ and one step of optimizing $G$. The traditional GAN reaches Nash equilibrium after several training epochs. The new objective of the $G$ of CorGAN breaks Nash equilibrium of the training process, which causes that the $G$ can keep generating outlier examples.

The inlier data are taken as training data in CorGAN. In the adversarial process of training CorGAN, the $G$ is supposed to generate outlier samples for negative class, and the $D$ is trained to assign a high probability value to data from training datasets (i.e. positive class) and a small probability value to generated data from the $G$ (i.e. negative class). The generated outliers not only distribute around the positive class, but also cover feature space far away from positive class. In order that $G$ can map a prior distribution to a huge data space except from the positive class, we proposed a lot of improved techniques (section 4). After the training process, the resultant $D$ will be used to detect outliers by comparing its output probability and user specified threshold.

3.3 **STEP2: DETECTING OUTLIERS USING THE DISCRIMINATOR**

In detection process, the resultant Discriminator outputs a relatively high probability for data that have the same distribution as in training dataset and a relatively low probability for data with different distribution from in training data. That is to say that if the output is low probability in outlier-detecting process, the input is predicted as an outlier. But what is a low probability? So, we need a probability threshold to decide whether an output probability is high or low. Since the output after the sigmoid activation function of last layer is in interval $(0, 1)$, we can intuitively set 0.5 as the threshold. In that case, the input is outlier, if the output from $D$ is small than 0.5, otherwise inlier. Depending on the concrete trained $D$, the confidence threshold with a value of 0.5 not necessarily leads to an optimal performance. Sometimes, the deviation of 0.5 shows a better evaluation score.

The best value of the confidence threshold mainly depends on how the model is trained. If the model is trained by specifying a new objective for the $G$ (like in CorGAN), the $D$ model learns distribution from training datasets for a long time. How, the $D$ is trained with data from much larger outlier distribution using the same time. The resulting $D$ will present a relatively higher probability for data that follow the same distribution as the training data (i.e. for inliers). So, the confidence threshold
Outlier Detection, also called one-class classification, can be evaluated with F1-score, which is harmonic mean of precision and recall. The accepted fraction of the positive class $f_{T+}$ and the rejected fraction of the negative class $f_{O-}$ are both together also as a popular measure for OCC. However, the score of those measures strongly depends on the specified threshold. The measures show its highest score with different threshold in different applications and models. In order that the results can be comparable to the state of art reached by other various kinds of methods, the performance of the $D$ in this paper will be evaluated with Receiver operating characteristic curve (ROC) and Area under the ROC curve (AUC). The AUC score of the $D$ will be calculated by testing various test datasets.

4 IMPROVED TECHNIQUES FOR GAN IN OCC

If the training process reaches Nash equilibrium, the $G$ is able to generate examples following the same distribution as training data (see figure 2), and the output probability of the $D$ is always 0.5 for inliers and an unexpected value for outliers. It is difficult to distinguish them with a threshold. Our proposed corrupted generative adversary network (CorGAN) is a GAN without convergence. In order to avoid the Nash equilibrium that the training process could reach, this section proposes a lot of techniques to break the convergence and build a robust outlier identifier. Specifying a new objective for the $G$ is a basic one to keep it generating outlier examples, and other optional techniques improve the performance of the model.

![Figure 2: Comparison between the generated data and the training data: The images of handwritten digit nine are training data. After several training epochs, the generated images and the training data are visualised in the figure.](image)

4.1 EARLY STOPPING

In early training epochs (i.e. before convergence), the $G$ has no ability to generate data that follows the same distribution as the training datasets. Meanwhile, the $D$ is trained with the training data with positive labels and the generated data with negative labels. Distributions from $G$ are different from the distribution of training datasets before convergence. The $D$ recognizes the distribution of training datasets by presenting a high probability. Early Stopping before convergence can obtain a well-behaved Discriminator.

In term of implementation of this method, the performance of the $D$ is tested regularly during the training process. It is not necessary to decide whether the current performance is the best one that the discriminative model could be. We just save the best model we get so far. At end of the training process, the saved best model is the model that shows the best evaluation score. In other words, we do not explicitly stop training, but always save the best model. During the training process, the $D$ is evaluated regularly with Area Under the Curve of inlier accepted fraction, called positively biased AUC (see figure 3). The $D$ saved with best biased AUC score shows not optimal but near optimal performance on test dataset.

4.2 SPECIFY A NEW OBJECTIVE FOR THE GENERATOR

Even though Early Stopping avoids the problem the convergence causes, GAN can only be trained with a limited number of epochs. Hence, Early Stopping can only guarantee a high inlier accepted
Figure 3: Area Under the Curve of inlier accepted fraction: The figure describes the relationship between the inlier accepted fraction and the specified threshold. Given the specified threshold 0.7, the point $P$ in the curve corresponds to the accepted fraction of inliers 0.68. Since no outlier is available, the area under this curve (positively biased AUC) is a good measure to select the near optimal model.

Table 1: The behave of the $G$ and the performance of the $D$ in case of various new target values.

<table>
<thead>
<tr>
<th>new target value $a_{new}$</th>
<th>$a_{new} = 1$: The objective of $G$ is the exact same as the convergent GAN.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{new} \in (\sim 0.9, 1)$: The $G$ is still able to generate realistic images. But the speed of convergence is slowed down. In this case, there exists more training epochs before Nash equilibrium is reached. Combined with Early Stopping method, this method can train $D$ with more generated samples. Since the model is saved before convergence, $f_{T+}$ is still not affected by the new objective. Fortunately, $f_{O-}$ is improved because of more generated data with negative labels. Even though the convergence process is slowed down, it becomes more stable, which can be used to improve the training process [Salimans et al. (2016)].</td>
<td></td>
</tr>
<tr>
<td>$a_{new} \in (\sim 0.5, \sim 0.9)$: $f_{T+}$ is still closed to 1. The $D$ can be trained with much more generated samples that distribute both around and far from the positive class. Plenty of feature space can be explored by the $D$. $f_{O-}$ is improved further.</td>
<td></td>
</tr>
<tr>
<td>$a_{new} \in [0, \sim 0.5)$: The $G$ can only generate data that distribute far from the positive class. In case of $a_{new} = 0$, the new target minimises the probability $D$ making a mistake, i.e. the $G$ will generate data as far as possible.</td>
<td></td>
</tr>
</tbody>
</table>

fraction $f_{T+}$, but not necessary high outlier rejected fraction $f_{O-}$ because the $D$ is only trained with a certain number of generated samples (i.e. outliers). To build a robust outlier identifier against as many kinds of outlier distributions as possible, we should train the $D$ with as many generated samples as possible, which have different distribution from the distribution of the training dataset (i.e. inliers).

We can break Nash equilibrium by specifying a new objective for $G$. Without modification, the objective of $G$ is to maximise the probability of the $D$ making a mistake. We propose a new objective for $G$. Instead of maximising the probability that $D$ makes a mistake, the new objective is that the $D$ makes a mistake with a certain probability. The new target value used to calculate the cost for updating the $G$ is $a_{new} \in (0, 1)$ instead of 1 of convergent GANs. Depending on the new target value, the behavior of the $G$ and the performance of the $D$ is shown in the table.

4.3 ATTACHING MORE IMPORTANCE TO GENERATED DATA

Cost of the $D$ consists of two parts. These two parts are caused respectively by the training data and the generated data. Generally, the two parts are simply added together as the total cost for updating the parameters of the $D$. That is to say that the training data and the generated data are treated with the same importance. They can be treated differently by assigning a weight to one of them to broaden the search space of parameters:

$$d_{\text{cost}} = \text{weight} \times d_{\text{cost\_real}} + d_{\text{cost\_gen}}$$
Table 2: Training -, validation - and test datasets of experiments setting.

<table>
<thead>
<tr>
<th>Datasets:</th>
<th>Source of Images:</th>
<th>Number:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>digit of 9 in MNIST</td>
<td>4967</td>
</tr>
<tr>
<td>Validation</td>
<td>digit of 9 in MNIST</td>
<td>900</td>
</tr>
<tr>
<td>Test dataset</td>
<td>Inliers: digit of 9 in MNIST</td>
<td>900</td>
</tr>
<tr>
<td></td>
<td>1.Outliers: digits of 0-8 in MNIST</td>
<td>900</td>
</tr>
<tr>
<td></td>
<td>2.Outliers: CIFAR10</td>
<td>900</td>
</tr>
<tr>
<td></td>
<td>3.Outliers: Images composed of noise</td>
<td>900</td>
</tr>
</tbody>
</table>

While the outlier distributions are various and difficult to recover all of them, the inlier distribution is rather simplex and easy to learn. During the training process, the cost that generated data caused should be reduced as far as possible by updating parameters of the \( D \). In other words, the generated data should be attached more importance by specifying the value of weight. Compared to the general case that the two parts of cost are not treated differently, this method generally shows a better performance on the test datasets whose distributions are far from the training dataset.

4.4 COMBINING PREVIOUSLY GENERATED DATA

Compared to the method of Early Stopping, the method of specifying a new objective for \( G \) presents a better performance because the latter trains \( D \) with arbitrarily more generated data that have different distribution from of training data. With the new specified objective, the training procedure does not converge, and the \( G \) is able to keep generating data that have distributions different from in training data. The \( D \) can be trained with arbitrarily many generated distributions. However, the space of distribution learned by \( D \) is limited to a great extent. On one hand, the generated distribution always stays near the positive class after several training epoch. On the other hand, the \( D \) can forget the previously learned distributions because of the limited capacity.

In this subsection, we proposed a technique to broaden the learned distributions. The performance of the \( D \) can be improved by being regularly trained with previously generated data. We can train CorGAN with mini batches that combine the data generated recently and previously. The combined training data can avoid that the \( D \) forgets the learned distribution to some degree.

There exist a lot generated data in the training procedure. Which ones should be chosen to train \( D \) and prevent it forgetting the previously generated distributions? Because the generated data can be arbitrarily many, it is unadvisable and impossible to save all of them. In this case, the generated data can be treated as stream data. We apply a Reservoir Sampling Algorithm \cite{vitter1985random} to sample previously generated images. This algorithm samples examples from the stream data with the same probability and specifies a reservoir to save the sampled examples. Together with new generated examples, the sampled examples saved in a reservoir can be used to train \( D \). The resultant \( D \) can identify not only recently generated outliers but also previous ones.

5 EXPERIMENTS AND ANALYSIS

Datasets and Evaluation: To objectively evaluate the performance of outlier detection, lots of datasets are used in the experiments, namely, MNIST \cite{lecun1998mnist}, CIFAR10 \cite{krizhevsky2009learning}. The image size in MNIST is (28, 28). The size in CIFAR10 images are cropped into (28, 28) by removing pixels along the sides. Especially, we specify a dataset composed of three group noisy images with same size (28, 28), the value of their whose pixels are respectively subject to uniform distribution, Gaussian distribution and random values. The table 2 lists training dataset, validation dataset and test datasets. The performance of various approaches will be evaluated and compared with the Area Under the Receiver Operating Characteristic (ROC) Curve (AUC).

Experiments Setting: PCA+PSVM: PCA is used to reduce the dimensionality of the high-dimensional data (i.e. images). The number of components \( K \) is set such that 95% of the variance is retained (\( K=111 \)). One-class SVM proposed in \cite{scholkopf2000estimating} is plane-based, called PSVM. To identify outliers in the feature space, PSVM tries to find a hyperplane that best sepa-
rates the data from the origin. RBF kernel is used in this experiment. Other settings are defaults in sklearn.svm.OneClassSVM Pedregosa et al. (2011).

**Autoencoder:** Autoencoder detects outliers by computing reconstruction error and compares it with a specified threshold. The threshold is based on the difference between the inputs and outputs for the training data. If the reconstruction error for a test sample is larger than the threshold, then the sample is identified as outlier, otherwise as inlier. In order to justify our proposal, we compare to convolutional autoencoder. The encoder has the same architecture as the Discriminator in CorGAN except from output layer 4a. The decoder also has similar architecture as the Generator in CorGAN 4b. The model is regularised with weight decay $\lambda=0.01$. The parameters are updated with SGD optimisation algorithm, $\text{minibatch}=128$ and learning rate $lr=0.1$. The cost function is cross-entropy function. The model is trained for 30 epochs without pretraining.

![Figure 4: The architecture of the convolutional Autoencoder.](image)

**CorGAN:** The basic architecture of CorGAN as well as the number of its layers and units are shown in figure 1. We propose a lot of improved techniques. Since its combinations are numerous, we justify only three main models. The first model is basic one, CorGAN = GAN with early stopping technique and a new objective for the $G$ (see section 4.2). The new target value $a_{\text{new}}$ is set manually to 0.9 for the $G$. The $G$ is regularised with weight decay $\lambda=0.1$. The optimisation algorithm is Adam, $\text{minibatch}=128$ and learning rate $lr=0.0002$. No pretraining is performed. The second model to be justify is based on the first one, CorGAN2 = CorGAN + Attaching more importance to generated images (see section 4.3). The weight is set to 0.5 manually. The third illustrated model is also based on the first one, CorGAN3 = CorGAN + Combining previously generated images (see section 4.4). The minibatch size is composed of 64 images sampled from previous training epoch and 64 newly generated images.

**Results and Analysis:** The results of the experiments are shown in table 3. The outlier distribution of the handwritten digits images of the numbers (0-8) is relatively close to the inlier distribution of the number 9. Hence, all the approaches show the worse AUC scores on the first test dataset. The PCA+PSVM approach shows better score on the second test dataset than on the noise test dataset. The traditional approach is not robust enough for noise outliers. The solutions based on neural network often show a better performance against noise data because of the random initialisation of its parameters, especially, our proposed solution based on GAN framework, in which Generator generates lots of noise examples. The convolutional autoencoder can reconstruct natural images well by detects edges, corners and objects. Therefore, the convolutional autoencoder shows unsatisfactory score on natural images. Our proposed solution classifies test examples without reconstruction process, which shows robust performance against outlier natural images as well as noise images.

Compared to CorGAN, CorGAN2 attaches more importance to generated images, which makes classifier more robust again the outliers whose distribution is far from inlier distribution. In consequence, CorGAN2 shows low score on the first dataset, in which the distributions of inliers and outliers are relatively close. In the model CorGAN3, the outlier examples generated previously are combined with newly generated examples to train the Discriminator. In this way, the Discriminator learned a large space of outlier distribution. CorGAN3 shows the best scores on various test datasets.
Table 3: The AUC scores of various models are shown in the table. All the models are tested in three datasets: MNIST(9) + MNIST(0-8), MNIST(9) + CIFAR10, MNIST(9) + Noise. Within MNIST(9) images are inliers, and other images are outliers. CorGAN, CorGAN2 and CorGAN3 are described in section 5.

<table>
<thead>
<tr>
<th>Model</th>
<th>MNIST(9)</th>
<th>MNIST(0-8)</th>
<th>CIFAR10</th>
<th>Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA + PSVM</td>
<td>0.8623</td>
<td>0.9720</td>
<td>0.9302</td>
<td></td>
</tr>
<tr>
<td>Autoencoder</td>
<td>0.8974</td>
<td>0.9739</td>
<td>0.9995</td>
<td></td>
</tr>
<tr>
<td>CorGAN</td>
<td>0.8974</td>
<td>0.9739</td>
<td>0.9995</td>
<td></td>
</tr>
<tr>
<td>CorGAN2</td>
<td>0.8343</td>
<td>0.9937</td>
<td>0.9999</td>
<td></td>
</tr>
<tr>
<td>CorGAN3</td>
<td>0.9253</td>
<td>0.9943</td>
<td>0.9999</td>
<td></td>
</tr>
</tbody>
</table>

6 Conclusion and Future Work

In this paper, we present a solution to solve one-class classification problem based on GAN framework and successfully apply the Discriminator of the framework to detect outliers. We illuminate a lot techniques to improve the performance and verify the proposed techniques with experiments. First, we choose the near optimal model to detect outliers by saving a better model during training procedure. Then we specify a new objective for the $G$ so that it can keep generating outliers. The more importance are attached to generated images to further improve the performance of the $D$. In order to prevent the $D$ forgetting the previously generated outliers, we combine previously generated outliers from the Generator to train the outlier identifier. These techniques show comparable AUC scores.

In future work, We can further vary the generated distributions to train $D$. We can specify multiple Generators in the generative adversary framework. The mini batch can combine the data generated by different Generators, which have different objective, e.g. the different probabilities of $D$ making a mistake. To further explore more generated distribution used to train the $D$, we can even combine CorGAN with other generative models. Similarly, we must also change the objective of them to fit our goal, since the other generative models are also used to generate outliers.

All the proposals in this thesis do not leverage distance information (KLD) between distributions within a batch both in the training process and detecting process. Another topic worth studying is the clustering-based method to detect outlier using $D$ of GAN. One potential method of leveraging distance information is to model the closeness between examples in a minibatch. The modelling process is described in Minibatch Discrimination [Salimans et al. (2016)], an improved technique for training GANs.

In terms of the task of detecting of outlier images, we will try to identify more fine attribute of images. For instance, the built outlier identifier should be able to distinguish images taken under different illumination as well as different viewpoints, which describe the same object. Furthermore, we can take images of a group of objects as inliers. We will build a one-class classifier to make a decision about whether the object described by the given image comes from the group.

References


