Estimate the Implicit Likelihoods of GANs with Application to Anomaly Detection

Shaogang Ren, Dingcheng Li, Zhixin Zhou, Ping Li
Cognitive Computing Lab
Baidu Research
10900 NE 8th St. Bellevue, WA 98004, USA
{shaogangren, lidingcheng, zhixinzhou, liping11}@baidu.com

ABSTRACT

The thriving of deep models and generative models provides approaches to model high dimensional distributions. Generative adversarial networks (GANs) can approximate data distributions and generate data samples from the learned data manifolds as well. In this paper, we propose an approach to estimate the implicit likelihoods of GAN models. A stable inverse function of the generator can be learned with the help of a variance network of the generator. The local variance of the sample distribution can be approximated by the normalized distance in the latent space. Simulation studies and likelihood testing on real-world data sets validate the proposed algorithm, which outperforms several baseline methods in these tasks. The proposed method has been further applied to anomaly detection. Experiments show that the method can achieve state-of-the-art anomaly detection performance on real-world data sets.

CCS CONCEPTS

• **Theory of computation → Unsupervised learning and clustering**.

KEYWORDS

anomaly detection, generative adversarial networks, unsupervised learning, density estimation

ACM Reference Format:
Shaogang Ren, Dingcheng Li, Zhixin Zhou, Ping Li. 2020. Estimate the Implicit Likelihoods of GANs with Application to Anomaly Detection. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380293

1 INTRODUCTION

Many real-world high dimensional data sets concentrate around low dimensional unknown manifolds. Deep models provide new approaches to estimate the density of extreme high dimensional data. Generative models, e.g., generative adversarial networks (GANs) [13], can learn the distributions of high dimensional data sets and generate samples as well. GANs use the adversarial loss as their training objective, which penalizes dissimilarity between the distribution of the generated samples and the real samples. Given infinite approximation power, the original GAN objectives aim to minimize the Jensen-Shannon divergence between the real data distribution and generated samples.

The generator in a GAN can be seen as a nonlinear parametric mapping $g : \mathcal{Z} \to \mathcal{X}$ from the low dimensional latent space to the data manifold. With the realistic pictures generated with GANs [6, 17], one can claim that a well regularized GAN models can approximate the true data manifolds well. With good data manifold approximations, people try to leverage the GANs to other tasks beyond image generation, e.g., anomaly detection [1, 30], photo style transformation [38], and text generation [34].

Due to the broad applications of GANs, it is important to measure the distribution likelihoods of any given samples. However, GANs are implicit models, and it means the sample likelihoods cannot be computed directly. The discriminator used in the GAN is designed to classify the samples from true data distribution and the samples from the generator. Thus the discriminator cannot estimate the likelihoods of the samples that do not belong to either distributions. It is proved by [5] that there exists an inverse function of the generator that can project samples in the sample space into the latent space. [29] provide an approach to estimate the spectral implicit density of generative models. However, this method does not provide an qualitative way to estimate the likelihood of any given samples. In this paper, we bring up a method that can learn an inverse function of the generator of GAN models that preserve the measurement consistence, and then apply the inverse function to data sample likelihood estimation. In the following subsections, we first review some manifold concepts for deep generative models, and then we briefly introduce neural network models for density estimation. In the last subsection, we present a short survey on the application of generative models to anomaly detection.

1.1 Deep Generative Models as Manifolds

Recently, people have tried to apply manifold analysis to generative models. For example, [22] improves GAN based semi-supervised learning by adding manifold invariant into the classifier. The authors of [33] try to perform geodesic clustering with deep generative models. In [4], the authors add stochastic variables to the VAE generator. With the variance network, the metric estimation on the data manifold can be improved. Mathematically, a deterministic generative model $x = g(z)$ can be seen as a surface model if the generator g is sufficiently smooth.

Here, we briefly review the basic concepts on surfaces. A deep generative model represents an embedding function, $g : \mathcal{Z} \to \mathcal{X}$, from a low-dimensional latent space $\mathcal{Z} \subseteq R^d$ to a submanifold $\mathcal{M} \subseteq R^D$. Usually we have $D \gg d$. We assume g is a smooth injective mapping, so that $\mathcal{M}$ is an embedded manifold. The Jacobian of g at

$z \in \mathcal{Z}, \mathbf{J}_g(z)$, provides a mapping from the tangent space at $z \in \mathcal{Z}$ into the tangent space at $x = g(z) \in \mathcal{X}$, i.e., $\mathbf{J}_g : T_z\mathcal{Z} \to T_x\mathcal{X}$. This mapping is not surjective and the range of $\mathbf{J}_z g$ is restricted to the tangent space of the manifold $\mathcal{M}$ at $x = g(z)$, denoted as $T_x\mathcal{M}$. Since GAN can generate realistic images, $\mathcal{M}$ is close to the true data manifold, namely $\mathcal{M}_{data}$.

We represent the Riemannian metric as a symmetric, positive definite matrix field, $\mathbf{M}(z)$, defined at each point z on the latent coordinate space, $\mathcal{Z}$. $\mathbf{M}(z)$ is given by the following formula:

$$\mathbf{M}(z) = \mathbf{J}_g(z)^\top \mathbf{J}_g(z).$$

Given two tangent vectors $u, v \in T_z\mathcal{Z}$ in coordinates, their inner product is defined as $\langle u, v\rangle = u^\top \mathbf{M}(z)v$. Let us consider a smooth curve $\gamma : [a, b] \to \mathcal{Z}$. This corresponds to a curve on the manifold, $g \circ \gamma(t) \in \mathcal{M}$. The arc length of γ is given by the following formula:

$$L(\gamma) = \int_a^b \sqrt{\dot{\gamma}(t)^\top \mathbf{M}_{\gamma(t)}\dot{\gamma}(t)}\mathrm{d}t,$$

where $\dot{\gamma}(t)$ is the first order derivative of γ at time t. For example, with the explicit formula of geodesic distance, we can then apply geodesic clustering on the data manifold [12, 33].

1.2 Change of Variable

When $\dim(\mathcal{Z}) < \dim(\mathcal{X})$, the change of variable theorem is known in the context of geometric measure theory as the smooth coarea formula [16, 20]. With the assumption that g is a topological embedding from $\mathcal{Z}$ to the submanifold $\mathcal{M}$ of $\mathcal{X}$. It says that

$$p_{\mathcal{X}}(x) = p_{\mathcal{Z}}(g^{-1}(x)) \det \left(\mathbf{J}_g(g^{-1}(x))^\top \mathbf{J}_g(g^{-1}(x))\right)^{-\frac{1}{2}},$$

where we assume the prior distribution on the latent space $p_{\mathcal{Z}}$ is $N(0, \mathbf{I}_d)$ throughout this paper. For $x \in \mathcal{M}$, we have $z = g^{-1}(x)$, and we obtain

$$\log\left(p_{\mathcal{X}}(g(z))\right) = \log\left(p_{\mathcal{Z}}(z)\right) - \frac{1}{2}\log\left(\det\left(\mathbf{J}_g(z)^\top \mathbf{J}_g(z)\right)\right). \quad (1)$$

For a real-world data set, typically we would not know the true tangent space dimension number of $\mathcal{M}_{data}$ at a given data sample x. People usually set $d = \dim\mathcal{Z}$ larger than the $T_x\mathcal{M}_{data}$ for all x. This leads to the problem that $\mathbf{J}_g$ does not have a full rank, and so does $\mathbf{M}(z)$. It means that it is impossible to directly apply Eq. (1) with logarithm of zero determinant to estimate the sample density.

1.3 Density Estimation with Neural Networks

Along with the thriving of deep neural networks and GANs, a variety of density estimation methods have been developed. The MADE model [11] is proposed to estimate the likelihood values with masked neural networks. RealNVP [8] and Glow [18] are generative models that can estimate samples' likelihood values. Both RealNVP and Glow employ reversible neural networks with special Jacobian metric whose determinate can be easily calculated. Similar to RealNVP and Glow, FFJord [14] is a generative model which can yield sample likelihood values with free-form Jacobian reversible neural network. FlowGAN [15] estimates data distribution by combining maximum likelihood estimation (MLE) and adversarial loss. The combination of MLE and adversarial loss may reduce the quality of generated samples. Most of these models use flow models to avoid the singularity issues coming with GANs. However, using the same dimension number for both the latent space and the data space space may violate the fact that most real-world data sets are following distributions on low dimensional manifolds. As discussed in previous subsections, GANs can approximate the real data distribution in a more reasonable approach. In this paper, we try to develop a method that can estimate the quantitative likelihood or density values of data samples by leveraging the manifold approximation power of GANs. Note that, even without likelihood estimation, people already apply GANs to anomaly detection. We will also apply our GAN likelihood estimation method to anomaly detection and demonstrate its effectiveness by comparing with existing anomaly detection methods. A review on anomaly detection with generative models is given in the next subsection.

1.4 Anomaly Detection with Generative Models

Generative models have been taken as effective ways to learn data representations and have been applied to anomaly detection. The auto-encoder based approaches [2, 37] first train a model that can reconstruct normal samples and then use the reconstruction errors to identify abnormal samples. The authors of [39] assume the latent spaces of data sets follow Gaussian mixture distributions. Instead of utilizing auto-encoder based methods that derive statistical anomaly criterion in light of data distributions or energies, they employ GAN-based framework to detect abnormal medical images in [30]. In their work, the latent variables of the samples are inferred with stochastic gradient descent. The anomaly score is estimated with the reconstruction error from the inferred latent variables in addition to the discrimination value from the discriminator of GAN model. Different from these existing methods, we try to develop a new anomaly detection method base on the proposed GAN likelihood estimation. In the sense of generativeness, our approach shares something in common with [30].

In summary, we aim to propose a framework that can estimate the log-likelihoods of samples by leveraging the GAN models. The proposed method can approximate the local variance of any given data points with the help of two learned networks, the variance for the generator and the inference network for the latent representation. With the variance network, the singularity of the generator's Jacobian matrix can be avoided. Thus, the likelihoods of samples can be calculated with the Riemannian metric. Our experiments on several real-world data sets reveal that, in the tasks of anomaly detection and likelihood testing, our approach considerably outperforms other baseline methods.

2 METHODOLOGY

GANs attempt to estimate the empirical distributions of high dimensional data sets. Given the learned generator g, and a latent variable z, we can estimate the generated sample $g(z)$'s log-likelihood with Eq. (1). The computation of Eq. (1) implies the full rank of g's Jacobian matrix regarding z. We can use a low dimensional latent space, i.e., a small d, to obtain a full rank Jacobian matrix for almost any $z \in \mathcal{Z}$. However, experimentally a small d may hurt the performance of GANs. In this paper, we try to resolve these issues with an extension of the classical GAN model. Our primary goal is to estimate the likelihoods of any given data samples. We need to learn an inference network to directly map a given data point x in the input space to a latent variable z that can be mapped back to a point close to x on the manifold $\mathcal{M}$. Then the sample x's log-likelihood value can be computed thanks to the generator and the proposed variance network.

2.1 The Variance Network of the Generator

We add a variance network σ to the generator g, and it extends the generator to a stochastic one,

$$f(z) = g(z) + \sigma(z) \odot \epsilon, \quad \text{and} \tag{2}$$
$$g : \mathcal{Z} \to \mathcal{X}, \sigma : \mathcal{Z} \to \mathbb{R}_+^D, \epsilon \sim N(0, \mathbf{I}_D).$$

Here g is the mean function, σ is the variance function, and $\odot$ stands for element-wise multiplication. For a given ϵ, f is an approximation of g, and $\mathbb{E}_{\epsilon\sim N(0,\mathbf{I}_D)} f(z) = g(z)$. The variance network σ extends the generator g to the whole input space $\mathbb{R}^D$. We will show that the singularity issue of the g's Jacobian matrix can be overcame with its variance network σ. To ensure that the variances are large in the regions without or less data samples, we formulate σ with a *radial basis function (RBF) neural network* [28]. The RBF network is a trainable kernel learning function.

The setup and training of σ is similar to [28]. First of all, a large number of data points are sampled from the latent space $\mathcal{Z}$, and then these data points are divided into K clusters with K-means. c_k is the center of cluster k and C_k is the number of data points in cluster k. For any data sample x, to compute the corresponding variance values, we first use the inference network h introduced in next subsection to estimate its latent variable, i.e., $z = h(x)$. The RBF network returns the x's variance values by aggregating the distance from z to all the cluster centers with learned kernel weights. This step can be done in the prepossessing stage and it can avoid computation overhead to the main algorithm.

The RBF function σ function is given by,

$$\sigma(z) = (\mathbf{W}^2 \mathbf{v}(z))^{-\frac{1}{2}},$$
$$v_k(z) = \exp\big(-\lambda_k ||z - c_k||_2^2\big), \; k = 1, .., K$$
$$\lambda_k = \frac{1}{2}\Bigg(\frac{a}{|C_k|}\sum_{z_j \in C_k} ||z_j - c_k||_2\Bigg)^{-2},$$

where a is a hyper-parameter for the kernel, and $\mathbf{W}$ is the network parameter need be learned from the training data samples. The samples with larger distance from the cluster centers will have larger variances. Given a generator network g, we can learn the variance network weights $\mathbf{W}$ by minimize the distance between $f(z)$ and x. With the stochastic generator f and the variance function, the Riemannian metric [4] can be written as

$$\begin{aligned}\bar{\mathbf{M}}_f^z &= \mathbb{E}_{\epsilon\sim N(0,\mathbf{I}_D)} \mathbf{J}_f^\top(z)) \mathbf{J}_f(z) \\ &= \mathbf{J}_g^\top(z)\mathbf{J}_g(z) + \mathbf{J}_\sigma^\top(z))\mathbf{J}_\sigma(z).\end{aligned} \tag{3}$$

We have the following lemma for the estimation of likelihoods.

Lemma 1. *With $K \geq \dim(\mathcal{Z}) + 1$ and a full rank $\mathbf{W}^2$, $\bar{\mathbf{M}}_f^z$ is a full rank matrix. The log-likelihood of the a generated sample is given by*

$$\mathbb{E}_{\epsilon\sim N(0,\mathbf{I}_D)} \log(p_\mathcal{X}(f(z))) = \log\big(p_\mathcal{Z}(z)\big) - \frac{1}{2}\log(\det(\bar{\mathbf{M}}_f^z)). \tag{4}$$

Proof. We have $\frac{\partial}{\partial z}\sigma(z) = B\mathbf{W}^2 VE$, where

$$B = \begin{bmatrix} -\frac{1}{2}\beta_1^{-\frac{3}{2}}(z) & & \\ & \ddots & \\ & & -\frac{1}{2}\beta_D^{-\frac{3}{2}}(z)\end{bmatrix}, \quad E = \begin{bmatrix} z - c_1 \\ \cdots \\ z - c_K\end{bmatrix}$$

$$V = \begin{bmatrix} -2\lambda_1 e^{-\lambda_1 ||z-c_1||_2^2)} & & \\ & \ddots & \\ & & -2\lambda_K e^{-\lambda_K ||z-c_K||_2^2)}\end{bmatrix}.$$

Here $\beta_i(z) = \mathbf{W}_{i.}^2 \mathbf{v}(z)$. As rank$(B) = D$, rank$(V) = K$, rank$(E) \geq \dim(\mathcal{Z})$, as B and V are diagonal and full rank, if $\mathbf{W}^2$ is full rank, than rank$(B\mathbf{W}^2 V) = $ rank$(W) = K$. As the K centers are different from each other, rank$(E) = \dim(\mathcal{Z})$, then rank$(\mathbf{J}_\sigma) = \dim(\mathcal{Z})$, thus $\bar{\mathbf{M}}_z^{(f)}$ is positive-definite. We have

$$\log\big(p_\mathcal{X}(f(z))\big) = \log\big(p_\mathcal{Z}(z)\big) - \frac{1}{2}\log\big(\det\big(\mathbf{J}_f^\top(z)\mathbf{J}_f(z)\big)\big),$$

$$\begin{aligned}&\mathbb{E}_{\epsilon\sim N(0,\mathbf{I}_D)} \log\big(p_\mathcal{X}(f(z))\big) \\ &= \log\big(p_\mathcal{Z}(z)\big) - \mathbb{E}_{\epsilon\sim N(0,\mathbf{I}_D)} \frac{1}{2}\log\big(\det\big(\mathbf{J}_f^\top(z)\mathbf{J}_f(z)\big)\big) \\ &= \log\big(p_\mathcal{Z}(z)\big) - \frac{1}{2}\log\big(\det(\bar{\mathbf{M}}_f^z)\big).\end{aligned}$$

The determinate of a $d \times d$ matrix, $\bar{\mathbf{M}}_f^z$ will be either a too small or a too large value that is out of the precision of the computer system. We can avoid this issue by using the eigenvalues of $\bar{\mathbf{M}}_f^z$ to estimate a sample's likelihood. □

Remark: Note that

$$\begin{aligned}&\mathbb{E}_{\epsilon\sim N(0,\mathbf{I}_D)} \log(p_\mathcal{X}(f(z))) \\ &= \log\big(p_\mathcal{Z}(z)\big) - \frac{1}{2}\log\big(\det(\bar{\mathbf{M}}_f^z)\big) = \log\big(p_\mathcal{Z}(z)\big) - \frac{1}{2}\sum_{i=1}^d \log(\lambda_i).\end{aligned}$$

Here $\lambda_i, 0 \leq i \leq d$ are the eigenvalues of $\bar{\mathbf{M}}_f^z$.

2.2 The Inference Network Learning

For a given data sample x, we need to find the corresponding latent space representation z before we can use Eq. (3) and Eq. (4) to compute the Riemannian metric and thus the likelihood value. As shown in Figure 1, we try to learn an approximate inverse of the generator network g, so that for $z \in \mathcal{Z}$, we have $h : \mathbb{E}_{\epsilon\sim N(0,\mathbf{I}_D)} h(f(z)) = \mathbb{E}_{\epsilon\sim N(0,\mathbf{I}_D)} h(g(z) + \sigma(z) \odot \epsilon) \approx z$. The latent variable of a point x in the input space can be approximated by the inference network (encoder) h. Given a ϵ and a full rank $\mathbf{J}_f(z)$ at z and ϵ, f is locally invertible in the open neighborhood $f(S)$, S being an open neighborhood of z. We try to lean a $h : f(S) \to S$ such that $\mathbb{E}_{\epsilon\sim N(0,\mathbf{I}_D)} h(f(z)) \approx z, \forall z \in S$.

With the variance network σ introduced in the previous subsection, we can avoid the singularity of Riemannian metric of $\mathcal{M}(z)$ as long as $K \geq \dim(\mathcal{Z}) + 1$. We maximize the following empirical log-likelihoods of data samples from the data distribution p_{data} to

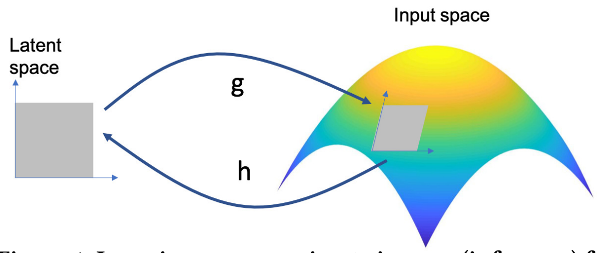

Figure 1: Learning an approximate inverse (inference) function of the generator.

learn the parameters of σ and h,

$$\max_{h,\sigma} \mathbb{E}_{x\sim p_{data}(x)}\left[\log(p(x|z))\big|_{z=h(x)}\right]. \quad (5)$$

Here $(p(x|z))$ is parametric with the extended generator f. According to Eq. (2), given a z, $p(x|z)$ is a multivariate Gaussian distribution with mean $g(z)$, and co-variance matrix Λ_σ, and $\Lambda_\sigma(z) = \text{diag}(\sigma(z))$. We add a constraint to Eq. (5) to include the prior distribution of z into the objective, and it forces the posterior $q(z|x)$ ($z = h(x)$, $x \sim p_{data}(x)$) to be close to the prior of z, $p(z)$ ($p(z)$ is the $p_{\mathcal{Z}}(z)$ in section 1.2),

$$\max_{h,\sigma} \mathbb{E}_{x\sim p_{data}(x), z=h(x)}\left[\log(p(x|z)) - \text{KL}(q(z|x)\|p(z))\right].$$

Each element of the empirical distribution $q(z|x)$ is a Gaussian distribution with the sample average of $h(x)$ as the mean, and the standard deviation of $h(x)$, $x \sim p_{data}(x)$ as the variance. The objective formulae is different from the VAE [19] model. In our model we do not model the variance of distribution $q(z|x)$ in order to reduce model complexity. Moreover, given the generator g, our objective is to learn a variance network σ for the generator g rather than the decoder in VAEs. It is easy to prove that, with $z \in \mathcal{Z}$, and $\epsilon \sim N(0, \mathbf{I}_D)$, f follows a Gaussian Process with g as the mean function, and Λ_σ as the covariance function, and Λ_σ is a diagonal matrix with σ along the diagonal.

Lemma 2. *Assuming g is a well learned generator function with GAN model, with $z \in \mathcal{Z}$ and $\epsilon \sim N(0, \mathbf{I}_D)$, the stochastic generator $f(z) = g(z) + \sigma(z) \odot \epsilon$ can be taken as*

$$f(z) \sim GP(g(z), \Lambda_\sigma).$$

Proof. Any collection of the entries of f follows a Gaussian distribution. According to the definition of Gaussian process, $f(z)$ follows Gaussian Process. □

By stacking f and h together, it is a regression model to reconstruct a data sample x ($x \sim p_{data}$), i.e., $\hat{x} = g(h(x)) + \sigma(h(x)) \odot \epsilon$. With the inferred latent variables and a collection of reconstructed data samples $\hat{x} \sim f(h(x))$, the whole framework is similar to the Gaussian Process latent variable model (GP-LVM, [23]). The objective for h and σ becomes

$$L_{h,\sigma} = \mathbb{E}_{x\sim p_{data}}\Bigg[\sum_{i=1}^{D}\Big[-\frac{1}{2}\big(g_i(h(x)) - x_i\big)^2/\sigma_i^2(h(x)) - \log\sigma_i(h(x))\Big] - \text{KL}(q(z|x)\|p(z))\Bigg].$$

2.3 Stabilize Training

Experimentally, it is rather difficult to learn a pair of perfect h and σ for a given g. We integrate the learning of both networks with the training of GANs. We have the following objective to adjust the generator g and h,

$$\min_{g,h} \mathbb{E}_{z\sim p(z)}\left[\|z - h(g(z))\|^2\right].$$

It is similar to the regularization term in InfoGAN [7] that enforces mutual information between the generated samples and partial latent variables. VEEGAN [32] also included this term to their regularization to reduce model collapses. Experiments show that with this term the model can converge fast and the training process is more stable. Figure 1 shows the scheme of the g and h.

2.4 Algorithm

The algorithm to learn h and σ is combined with the learning of the generator g and the discriminator d in GAN. The complete procedure is given in Algorithm 1.

Algorithm 1 The proposed algorithm to learn GAN with inference net h and variance net σ

```
1:  Input: Data sample set
2:  Output: Networks $g$, $d$, $h$, and $\sigma$
3:  Sample a fixed number of hidden variables from $P_{\mathcal{Z}}$ for K-means clustering; Compute $\lambda_k, c_k, k = 1, ..., K$;
4:  while Training not finished do
5:      Sample minibatch $m$ samples $\{x^{(1)}, ..., x^{(m)}\}$ from $P_{data}$;
6:      Sample minibatch $m$ samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $P_{\mathcal{Z}}$;
7:      Updating the discriminator $d$ with ascending the gradient
            $\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log d(x^{(i)})] + \log(1 - d(g(z^{(i)})))\right].$
8:      Sample minibatch $m$ samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $P_{\mathcal{Z}}$;
9:      Updating the generator $g$ by descending the gradient
            $\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\left[\log(1 - d(g(z^{(i)})))\right].$
10:     Sample minibatch $m$ samples $\{x^{(1)}, ..., x^{(m)}\}$ from $P_{data}$;
11:     Updating $h$ and $\sigma$ by ascending the gradient $\nabla_{\theta_{h,\sigma}} L_{h,\sigma}$;
12:     Sample minibatch $m$ samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $P_{\mathcal{Z}}$;
13:     Updating $h$ and $g$ by descending the gradient
            $\nabla_{\theta_{h,g}}\frac{1}{m}\sum_{i=1}^{m}\|z^{(i)} - h(g(z^{(i)}))\|^2.$
14: end while
```

Let $\mathbf{W}^t$ denote $\mathbf{W}$ at iteration t in Algorithm 1. We have the following Lemma 3 regarding the updating steps for $\mathbf{W}$. Lemma 3 shows that with a randomly initialized $\mathbf{W}$, we almost can ensure each $\mathbf{W}^t, t = 1, ..., T$ is with full rank, and thus a full rank Jacobian matrix for the stochastic generator f. This means we can safely compute the likelihood values for almost any testing samples. Algorithm 1 also avoids the step to directly fit the parameters of generator g with real data samples, and thus can preserve the capability of GAN to generate sharp and realistic images or data samples.

Lemma 3. *Let $\mathbf{W}^t$ be the matrix $\mathbf{W}$ at the updating step t of algorithm 1, then $\mathbf{W}^{t+1}$ will be the element-wise multiplication of $\mathbf{W}^t$ with another matrix.*

Proof. Given a sample x, the latent variable is $z = h(x)$. We have the loss regarding x,

$$L^x_{h,\sigma} = \sum_{i=1}^{D} \left[-\frac{1}{2}\big(g_i(h(x)) - x_i\big)^2/\sigma_i^2(h(x)) - \log \sigma_i(h(x))\right] - \mathrm{KL}(q(z|x)\|p(z)).$$

For the ith entry of x, the gradient regarding $\mathbf{W}$ is

$$\frac{\partial L^{x_i}_{h,\sigma}}{\partial \mathbf{W}} = -\left(\frac{\big(g_i(h(x)) - x_i\big)^2}{\sigma_i^3(h(x))} - \frac{1}{\sigma_i(h(x))}\right)\beta_i^{-\frac{3}{2}}\mathbf{W}_{i\cdot} \odot \mathbf{v}^T(z).$$

Here $\beta_i(z) = \mathbf{W}^2_{i\cdot}\mathbf{v}(z)$. Let

$$\mathbf{u}_i(x) = \frac{\beta_i^{-\frac{3}{2}}}{\sigma_i(h(x))}\left(1 - \frac{\big(g_i(h(x)) - x_i\big)^2}{\sigma_i^3(h(x))}\right),$$

we will get

$$\frac{\partial L_{h,\sigma}}{\partial \mathbf{W}} = \mathbf{W} \odot \big(\mathbf{u}(x)\mathbf{v}^T(z)\big).$$

Here $\mathbf{u}(x) = [\mathbf{u}_1(x), ..., \mathbf{u}_D(x)]^T$.

Without losing the generalization, we ignore the learning rate. For t, we have

$$\begin{aligned}\mathbf{W}^t &= \mathbf{W}^{t-1} + \mathbf{W}^{t-1} \odot \big(\mathbf{u}^{t-1}(x)(\mathbf{v}^{t-1}(z))^T\big)\\ &= \mathbf{W}^{t-1} \odot \left(\mathbf{1} + \mathbf{u}^{t-1}(x)(\mathbf{v}^{t-1}(z))^T\right).\end{aligned}$$

With a batch of training samples $\mathcal{B}$, $\mathbf{W}^t$ will be the element-wise product of $\mathbf{W}^{t-1}$ with an aggregated updating matrix, which is

$$\mathbf{W}^t = \mathbf{W}^{t-1} \odot \left(\mathbf{1} + \frac{1}{|\mathcal{B}|}\sum_{x\in\mathcal{B}, z=h(x)} \mathbf{u}^{t-1}(x)(\mathbf{v}^{t-1}(z))^T\right).$$

This concludes the lemma. □

Let $\mathbf{W}^0$ be the initial value of $\mathbf{W}$, the final step $\mathbf{W}^T$ will be the element-wise multiplication of $\mathbf{W}^0$ with the aggregated updating matrix regarding all steps. We are almost sure about the full rank of $\mathbf{W}^T$ as $\mathbf{W}^0$ is randomly initialized and the updating matrix is almost with full rank due to the randomness of training samples. The time consuming part is the computation of the likelihood in the testing stage. It is due to the calculation of $\mathbf{J}_g$, and g is a mapping from $\dim(\mathcal{Z})$ to $\dim(\mathcal{X})$.

3 THEORETICAL ANALYSIS

Algorithm 1 basically follows the same procedure to learn the generator g and the discriminator d in GANs. The only interaction between g and h is to enhance the mutual information between latent variables and the generated samples. The loss functions for d and g can be adapted to any other forms of divergence to measure distribution distances, such as Wasserstein distance. Without loss generality, we study the theoretical upper bound of distribution estimation with the proposed approach. We can follow existing approaches to analysis the estimation upper bound. We adopt the definition of *Earth Moving Distance*, or equivalently *Wasserstein-1* distance from [3].

Definition 1. *For distributions $\mathbb{P}$ and $\mathbb{Q}$ on $\mathbb{R}^d$, Wasserstein-1 distance between them*

$$W(\mathbb{P}, \mathbb{Q}) = \inf_{\gamma\in\prod(\mathbb{P},\mathbb{Q})} \mathbb{E}_{(X,Y)\sim\gamma}[\|X - Y\|].$$

In the following theorems, $\mathbb{P}^{(n)}$ denotes the empirical distributions of $\mathbb{P}$ with n samples. In other words, suppose x_i, $i = 1, \ldots n$ are i.i.d. samples of $\mathbb{P}$, then $\mathbb{P}^{(n)} = \frac{1}{n}\sum_{i=1}^n 1\{x = x_i\}$ is a empirical distribution of $\mathbb{P}^{(n)}$. For distributions $\mathbb{P}$ and $\mathbb{Q}$, $\mathbb{P} + \mathbb{Q}$ denotes the distribution of $X + Y$ for $X \sim \mathbb{P}$ and $Y \sim \mathbb{Q}$ independently. Suppose $\mathbb{Q}^{(n)} = \frac{1}{n}\sum_{i=1}^n 1\{y = y_i\}$ is a empirical distribution of $\mathbb{Q}^{(n)}$, then $\mathbb{P}^{(n)} + \mathbb{Q}^{(n)} = \frac{1}{n}\sum_{i=1}^n 1\{z = x_i + y_i\}$ is an empirical distribution of $\mathbb{P} + \mathbb{Q}$. It is worth noting that $\mathbb{P}^{(n)} + \mathbb{Q}^{(n)} \neq \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n 1\{z = x_i + y_j\}$.

Theorem 1. *Let $\mathcal{F}$ be certain function class and $g : \mathbb{R}^d \to \mathbb{R}^D$ be an L-Lipschitz function for $D > d > 2$. Let Z be random variable satisfying $\mathbb{E}[\|Z\|^2] \le c$ and $\mathbb{P}_1$ be the distribution of $g(Z)$. Let $\mathbb{P}^{(n)}$ be an n-sample empirical distribution of $g(Z) + \epsilon$, where Z follows distribution such that $\mathbb{E}[|Z|^2] \le \alpha$ and $\epsilon \sim N(0, diag(\sigma_1, \ldots, \sigma_D)/\sqrt{D})$, and*

$$\hat{g} = \arg\min_{\bar{g}\in\mathcal{F}, \bar{g}(Z)\sim\hat{\mathbb{P}}} W(\hat{\mathbb{P}}, \mathbb{P}^{(n)}).$$

Let $\hat{\mathbb{P}}$ be the distribution of $\hat{g}(Z)$, then

$$\mathbb{E}[W(\hat{\mathbb{P}}, \mathbb{P}_1)] \le 2C_d\sqrt{c}Ln^{-d} + 5\sqrt{\frac{\sum_{i=1}^D \sigma_i^2}{D}}$$

for the constant C_d only depending on d from Theorem 2.

Proof. Let $\mathbb{P}, \mathbb{P}_1$ and $\hat{\mathbb{P}}$ be the distribution of $g(Z) + \epsilon$, $g(Z)$ and $\hat{g}(Z)$ respectively. By triangle inequality,

$$W(\hat{\mathbb{P}}, \mathbb{P}_1) \le W(\hat{\mathbb{P}}, \mathbb{P}^{(n)}) + W(\mathbb{P}^{(n)}, \mathbb{P}) + W(\mathbb{P}, \mathbb{P}_1).$$

The first term on the RHS can be bounded by

$$W(\hat{\mathbb{P}}, \mathbb{P}^{(n)}) = \min_{\bar{g}\in\mathcal{F}, \bar{g}(Z)\sim\hat{\mathbb{P}}} W(\hat{\mathbb{P}}, \mathbb{P}^{(n)}) \le W(\mathbb{P}, \mathbb{P}^{(n)}),$$

where the inequality is due to the fact that $g \in \mathcal{F}$ and $g(Z) \sim \mathbb{P}$. Hence, we have

$$W(\hat{\mathbb{P}}, \mathbb{P}_1) \le 2W(\mathbb{P}^{(n)}, \mathbb{P}) + W(\mathbb{P}, \mathbb{P}_1).$$

It is sufficient to bound these two terms to obtain the desired result. We decompose $\mathbb{P}^{(n)} = \mathbb{P}_1^{(n)} + \mathbb{P}_2^{(n)}$ where $\mathbb{P}_1^{(n)}$ and $\mathbb{P}_2^{(n)}$ are n-sample empirical distribution of $g(Z)$ and ϵ respectively. We recall our assumption that $\epsilon \sim N(0, \text{diag}(\sigma_1, \dots, \sigma_D)/\sqrt{D}) = \mathbb{P}_2$.

$$W(\mathbb{P}^{(n)}, \mathbb{P}) \leq W(\mathbb{P}, \mathbb{P}_1) + W(\mathbb{P}_1, \mathbb{P}_1^{(n)}) + W(\mathbb{P}_1^{(n)}, \mathbb{P}^{(n)}).$$

By Lemma 4 and Jensen's inequality, we have

$$\begin{aligned} W(\mathbb{P}, \mathbb{P}_1) =& W(\mathbb{P}_1 + \mathbb{P}_2, \mathbb{P}_1) \\ \leq& \mathbb{E}_{\epsilon \sim \mathbb{P}_2}[\|\epsilon\|] \leq (E_{\epsilon \sim \mathbb{P}_2}[\|\epsilon\|^2])^{1/2} \leq \sqrt{\frac{\sum_{i=1}^D \sigma_i^2}{D}}. \end{aligned}$$

Since g is an L-Lipschitz function, Let $Z \in \mathbb{Q}$ and $\mathbb{Q}^{(n)}$ be n-sample empirical distribution of Z. Suppose $\mathbb{Q}^{(n)} = \frac{1}{n}\sum_{i=1}^n 1\{z = z_i\}$ and $\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^n 1\{x = g(z_i)\}$. Since g is an L-Lipschitz function, we have

$$\begin{aligned} W(\mathbb{P}_1, \mathbb{P}_1^{(n)}) =& \inf_{\gamma \in \prod(\mathbb{P}_1, \mathbb{P}_1^{(n)})} \mathbb{E}_{(X,Y)\sim\gamma}[\|X - Y\|] \\ \leq& \inf_{\gamma \in \prod(\mathbb{Q}, \mathbb{Q}^{(n)})} \mathbb{E}_{(X,Y)\sim\gamma}[\|g(X) - g(Y)\|] \\ \leq& \inf_{\gamma \in \prod(\mathbb{Q}, \mathbb{Q}^{(n)})} \mathbb{E}_{(X,Y)\sim\gamma}[L\|X - Y\|] \\ \leq& LW(\mathbb{Q}, \mathbb{Q}^{(n)}). \end{aligned}$$

By Theorem 2, we have

$$W(\mathbb{P}_1, \mathbb{P}_1^{(n)}) \leq LW(\mathbb{Q}, \mathbb{Q}^{(n)}) \leq C_d L\sqrt{c}n^{-d}.$$

By 5,

$$\begin{aligned} \mathbb{E}[W(\mathbb{P}_1^{(n)}, \mathbb{P}^{(n)})] =& \mathbb{E}[W(\mathbb{P}_1^{(n)}, \mathbb{P}_1^{(n)} + \mathbb{P}_2^{(n)})] \\ \leq& \mathbb{E}_{X \sim \mathbb{P}_2}[\|X\|] \leq \sqrt{\frac{\sum_{i=1}^D \sigma_i^2}{D}}. \end{aligned}$$

We combine these bounds and obtain

$$\begin{aligned} &\mathbb{E}[W(\mathbb{P}^{(n)}, \mathbb{P}_1)] \\ \leq& 3W(\mathbb{P}, \mathbb{P}_1) + 2\mathbb{E}[W(\mathbb{P}_1, \mathbb{P}_1^{(n)})] + 2\mathbb{E}[W(\mathbb{P}_1^{(n)}, \mathbb{P}^{(n)})] \\ \leq& 3\sqrt{\frac{\sum_{i=1}^D \sigma_i^2}{D}} + 2C_d L\sqrt{c}n^{-d} + 2\sqrt{\frac{\sum_{i=1}^D \sigma_i^2}{D}} \\ =& 2C_d L\sqrt{c}n^{-d} + 5\sqrt{\frac{\sum_{i=1}^D \sigma_i^2}{D}}, \end{aligned}$$

as desired. □

Lemma 4. *Let $\mathbb{P}_1, \mathbb{P}_2$ and $\mathbb{Q}$ be distributions on $\mathbb{R}^d$, then*

$$W(\mathbb{P}_1 + \mathbb{P}_2, \mathbb{Q}) \leq W(\mathbb{P}_1, \mathbb{Q}) + \mathbb{E}_{X \sim \mathbb{P}_2}[\|X\|]$$

Proof. Let γ is the optimal joint distribution for the infimum of $W(\mathbb{P}_1, \mathbb{Q})$. We note that γ does not necessarily exists in general. However, for every $\delta > 0$, there exists $\gamma_\delta \in \prod(\mathbb{P}_1, \mathbb{Q})$ such that

$$W(\mathbb{P}_1, \mathbb{Q}) + \delta \geq \mathbb{E}_{(X,Y)\sim\gamma_\delta}[\|X - Y\|].$$

Hence the argument can always be reduced to the case the optimal gamma exists. Hence without loss of generality, we may assume γ exists. Let γ' be the joint distribution of (X, Y) such that $Y \sim \mathbb{Q}$ and $X|Y \sim \gamma(\,\cdot\,, Y) + \mathbb{P}_2$. Then $\gamma' \in \prod(\mathbb{P}_1 + \mathbb{P}_2, \mathbb{Q})$.

$$\begin{aligned} W(\mathbb{P}_1 + \mathbb{P}_2, \mathbb{Q}) \leq& \mathbb{E}_{(X,Y)\sim\gamma'}[\|X - Y\|] \\ =& \mathbb{E}_{(X,Y)\sim\gamma, Z\sim\mathbb{P}_2}[\|X + Z - Y\|] \\ \leq& \mathbb{E}_{(X,Y)\sim\gamma, Z\sim\mathbb{P}_2}[\|X - Y\| + \|Z\|] \\ =& \mathbb{E}_{(X,Y)\sim\gamma}[\|X - Y\|] + \mathbb{E}_{Z\sim\mathbb{P}_2}[\|Z\|] \\ =& W(\mathbb{P}_1, \mathbb{Q}) + \mathbb{E}_{Z\sim\mathbb{P}_2}[\|Z\|], \end{aligned}$$

as desired. □

Lemma 5. *Let $\mathbb{P}$ and $\mathbb{Q}$ be distributions on $\mathbb{R}^d$. Let $\mathbb{P}^{(n)}$ and $\mathbb{Q}^{(n)}$ be the empirical distribution of $\mathbb{P}$ and $\mathbb{Q}$ respectively. Then*

$$\mathbb{E}[W(\mathbb{P}^{(n)}, \mathbb{P}^{(n)} + \mathbb{Q}^{(n)})] \leq \mathbb{E}_{X\sim\mathbb{Q}}[\|X\|].$$

Proof. We will use the definition $\mathbb{P}^{(n)} = \frac{1}{n}\sum_{i=1}^n 1\{x = x_i\}$, $\mathbb{Q}^{(n)} = \frac{1}{n}\sum_{i=1}^n 1\{y = y_i\}$ and $\mathbb{P}^{(n)} + \mathbb{Q}^{(n)} = \frac{1}{n}\sum_{i=1}^n 1\{z = x_i + y_i\}$. Let $\gamma \in \prod(\mathbb{P}^{(n)}, \mathbb{P}^{(n)} + \mathbb{Q}^{(n)})$ such that if $(X, Y) \sim \gamma$, then $\mathbb{P}(Y = x_i + y_i | X = x_i) = 1$. Then

$$\begin{aligned} &W(\mathbb{P}^{(n)}, \mathbb{P}^{(n)} + \mathbb{Q}^{(n)}) \\ \leq& \mathbb{E}_{(X,Y)\sim\gamma}[\|X - Y\|] \\ \leq& \frac{1}{n}\sum_{i=1}^n [\|x_i - (x_i + y_i)\|] = \frac{1}{n}\sum_{i=1}^n [\|y_i\|] \end{aligned}$$

We have $\frac{1}{n}\sum_{i=1}^n [\|y_i\|] = \mathbb{E}_{X\sim\mathbb{Q}}[\|X\|]$ as desired. □

We will introduce a useful theorem from [10], which provides the bound between empirical distribution and original distribution in Wasserstein distance. We refer readers to find the proof in the original paper.

Theorem 2 ([10], Theorem 1). *Let $\mathbb{P}$ be distribution on $\mathbb{R}^d$ and $\mathbb{P}_n$ be its n-sample empirical distribution. Suppose $\mathbb{E}_{X\sim\mathbb{P}}[\|X\|^2] \leq c$, then*

$$\mathbb{E}[W(\mathbb{P}, \mathbb{P}_n)] \leq C_d\sqrt{c}n^{-d},$$

for some constant C_d only depending on d.

4 EXPERIMENTS

In this section, we first evaluate the proposed likelihood estimation method with simulated and real-world data sets. Then we try to apply the proposed method to anomaly detection tasks. In the following experiments, we use 'InvGAN' to represent the proposed model. The implementation of the proposed InvGAN is based on the PaddlePaddle[1] platform.

4.1 Likelihood Estimation on Synthetic Data

In this subsection, we investigate the proposed approach with simulated data. We have two iid latent variables z_1, z_2 following $N(0, 1)$ distribution. The samples are simulated with $X = [w_1 \sin z_1, w_2 \cos z_2, w_3\, z_1^2, w_4 z_2, w_5\, z_1^3, w_6(z_1 + z_2^2)] + \epsilon$, and each entry of ϵ follow $N(0, 0.01)$. We can easily compute the Jacobian regarding X and z to get the ground truth likelihood value for each simulated data sample with Eq. (4). The simulated data set contents 50,000 samples for training and 50,000 samples for testing. In the proposed

[1]https://www.paddlepaddle.org.cn/

model, the generator and the discriminator both have two fully connected layers with 30 hidden nodes, and the invert function h has the same structure as the discriminator except the output dimension.

We compare the proposed model with two density estimation models FlowGAN, and FlowVAE. FlowGAN [15] is a hybrid model with maximum likelihood estimation (MLE) and adversarial loss. They employ the coupling layers proposed in [8] for the generator. With the coupling layers, the generator has the same number of dimensions for both latent space and input space. With the coupling layers [8], the determinate of the generator can be easily computed, and thus the likelihood of the samples. FlowGAN [15] train the generator based on both MLE and the adversarial the loss functions. We build a FlowVAE model with the decoder that has the same structure as the generator in the FlowGAN model. InvGAN and FlowGAN have the sample structure for the discriminator, which includes Four linear layers and three Relu layers. We use the same batch size and epoch number to train three models.

As state previously, the data set is simulated with 2 latent variables, and the dimension of the input (sample) space is 6. Since FlowGAN and FlowVAE use the same dimension number for both input space and the latent space, this may result in a constant offset between the model estimations and the ground truth likelihood. To fairly compare different models in figures, we add a constant value to the likelihood values of both FlowGAN and FlowVAE. Figure 2 gives the comparison between the ground truth and the estimated likelihood values for the testing set with three methods. We can see that compared with FlowGAN and FlowVAE, the proposed model can give more smooth estimations for the likelihood values, and improved fitting with the ground truth likelihood curve.

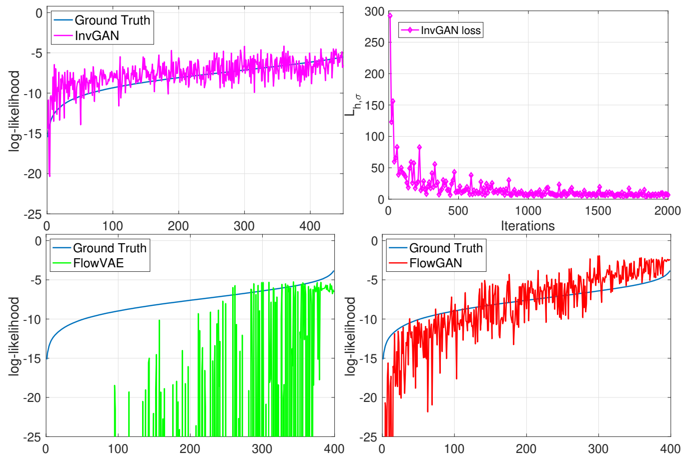

Figure 2: The upper left plot shows the InvGAN log-likelihoods of the simulated testing set; Upper right presents the objective loss values over iterations; The lower two plots give the log-likelihood values of the simulated data set using FlowVAE and FlowGAN. For plots with log-likelihood values, Y axe is the log-likelihood value, and X axe is sample index based on the increasing order of the ground truth log-likelihood value.

4.2 Likelihood Estimation on Real Data Sets

Table 1: Bits/dim on testing data sets of MNIST, CIFAR10, and ImageNet32 for different models; lower is better

Model	MNIST	CIFAR10	ImageNet32
RealNVP	1.06	3.49	4.28
Glow	1.05	3.35	4.09
FFJORD	0.99	3.40	-
MADE	2.04	5.67	-
MAF	1.89	4.31	-
FlowGAN	3.26	4.21	-
InvGAN	1.29	1.23	1.02

We compare our model with the other likelihood estimation methods using three real-world data sets, MNIST, CIFAR10, and ImageNet32. The methods listed in the Table 1 include RealNVP [8], Glow [18], FFJord [14], FlowGAN [15], MADE [11], and MAF [26]. Most of these methods are based on maximum likelihood estimation (MLE). FlowGAN is a hybrid model combining MLE and adversarial loss. RealNVP, Glow, and FlowGAN rely on revertible coupling layers to preserve the density mass between the input space to latent space. Different from invertible neural network models, MADE [11] and MAF [26] are mask based neural density estimation methods. For this set of experiments we use three convolution layers and one linear layer for the generator, the discriminator, and the inference network. More details about the network structures are given in Appendix A.

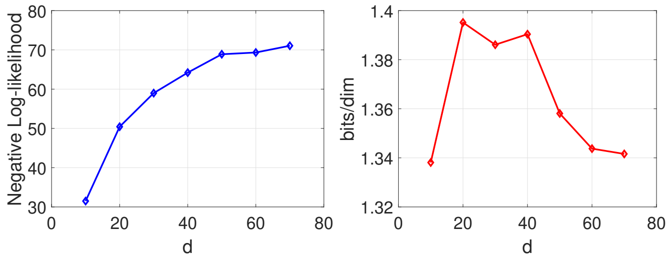

Figure 3: The left plot shows the average negative log-likelihood values of a MNIST testing image set using InvGAN at different latent space dimension (d). The right plot presents the corresponding bits/dim for the testing set under the revised metric at different d values.

For "InvGAN", different number of dim($\mathcal{Z}$) may lead to different values of log-likelihood values even for the same data sample (the left plot in Figure 3). The bits per dimension metric [14, 15], i.e., $-\log p_{\mathcal{X}}(x)/D, x \in R^D$, may not apply to our model, as the proposed model always yields smallest bits/dim value due the small latent space dimension number used in our model. To fairly compare with the other models, we propose to use the a revised version of bits per dimension metric in the comparison, which is $-\log p_{\mathcal{Z}}(z)/d + \frac{1}{2}\log\left(\det(\mathrm{J}_g(z)^\top \mathrm{J}_g(z))\right)/D$. Here d is the latent space dimension size. With the new metric, the bits/dim of InvGAN can be bounded in a reasonable range(as shown the right plot of Figure 3). The new metric can compensate the difference resulted from different latent space dimension numbers. For flow or reversible neural network based density estimation methods, the value of bits per dim does change as they take $D = d$. We follow the experimental setup in FFJord [14] and FlowGAN [15], and Table 1 compares different methods under the new metric on testing data sets. We use the likelihood values presented in FFJord [14] and FlowGAN [15]. We can see that even with the revised metric, our model can consistently perform better compared with other models.

4.3 Anomaly Detection

Table 2: Statistics of the data sets for anomaly detection

Data Set	Feature	Instance
MNIST	1,024	60,000
CIFAR10	3,072	60,000
Arrhythmia	274	452

In this subsection, we apply the proposed likelihood estimation to three anomaly detection tasks. The data sets we use for anomaly detection are MNIST [24], CIFAR10 [21], and Arrhythmia [9]. Table 2 gives more details on the three data sets. For all of the results, our anomaly score is defined similar to [1]. With the log-likelihood value set, $S = \{s_i : \mathrm{LLK}(x_i), x_i \in \mathcal{T}\}$, the anomaly score is

$$s_i' = \frac{s_i - \min S}{\max(S) - \min S}.$$

Here $\mathcal{T}$ is the testing set. The anomaly methods we compared our method with are given in next subsection.

4.3.1 Baseline Methods.

Deep Structured Energy Based Models (DSEBM) [36] are energy-based models for anomaly detection. Similarly to denoising autoencoder, the main idea is to accumulate the energy across layers of neural networks. Two types of anomaly scoring criteria were proposed: energy based and reconstruction error based. In our experimental results, DSEBM-r represents results with reconstruction error and DSEBM-e are results with energy based approach.

Deep Autoencoding Gaussian Mixture Model (DAGMM) [39] is an autoencoder-based method for anomaly detection, and it can achieve state-of-the-art results. In DAGMM, an auto-encoder as well as an estimator network are trained jointly. The auto-encoder is to generate latent space representations, and the estimator is to output parameters of a GMM modeling the lower-dimensional latent space. The likelihood value of a sample's latent representation computed with the learned GMM is taken as the anomaly detection metric.

AnoGAN [30] is a GAN-based method for anomaly detection. The method trains a GAN model to recover a latent representation for each test data sample. The anomaly score is calculated with a combination of the reconstruction error and the discrimination score from the discriminator network. The reconstruction error measures how well the GAN can reconstruct the data via the inferred latent variable and the generator. The authors of [30] compare the two components for the anomaly score and we picked the variant which performed best in the paper.

Efficient GAN-Based Anomaly Detection (EGBAD) [35] is another anomaly detection method based on GAN. Different from AnoGAN [30], their model learns an inverse function of the generator to speed up the inference of the latent variable. Similar to AnoGAN, The anomaly score in this approach includes two parts: the fitting error from reconstruction and the discrimination score.

GANomaly [1] uses a conditional generative adversarial network that jointly learns the generation of high-dimensional image space and the inference of latent space. Employing encoder-decoder-encoder sub-networks in the generator enables the model to map the input image to a lower dimension vector, which is then used to reconstruct the generated output image. The use of the additional encoder network maps this generated image to its latent representation. Minimizing the distance between these images and the latent vectors during training aids in learning the effective representations for the normal samples.

Adversarially Learned Anomaly Detection (ALAD) [35] is a recently proposed GAN based anomaly detection method. Similar to EGBAD [36], ALAD learns an encoding function to infer the latent variable for testing samples. Their model is enhanced with three discriminators. With a cycle consistence between sample space and latent space, the training of their model can be stabilized in learning representations for anomaly detection.

One Class Support Vector Machines (OC-SVM) [31] are a classic kernel method for anomaly detection and density estimation that learns a decision boundary around normal examples. The radial basis function (RBF) kernel is employed in the experiments. The v parameter is set to the expected anomaly proportion in the data set, which is assumed to be known, whereas the γ parameter is set to be inversely proportional to the number of input features.

Isolation Forests (IF) [25] are a classic machine learning technique that isolates abnormal samples rather than learning the distribution of normal data. The method constructs trees across randomly chosen features according to randomly split values . The anomaly score is defined as the average path length from a testing sample to the root. The experiments results for this model are obtained with the implementation in the scikit-learn [27] package.

4.3.2 MNIST. We take one class of the ten numbers in the MNIST [24] data set as the abnormal class, and the rest classes as the normal class. In the training phase, we use only the images from nine normal classes to train GAN and our model. In the testing state, we use the images from all ten classes in the testing set. The experiment set up is following GANomaly [1]. The results for GANomaly [1] is obtained by running the online code with the optimal parameters given in the script. The results for the other methods are based on the result section in [1]. The upper plot in Figure 4 shows the results of all of the methods on ten classes. The proposed method outperforms the others methods in all the ten tasks.

4.3.3 CIFAR10. Similar to the MNIST data set, we take one class as the abnormal class, and the rest classes as normal. We follow the experimental set up in [1]. The testing involves samples from both the normal classes and the abnormal class in the testing data

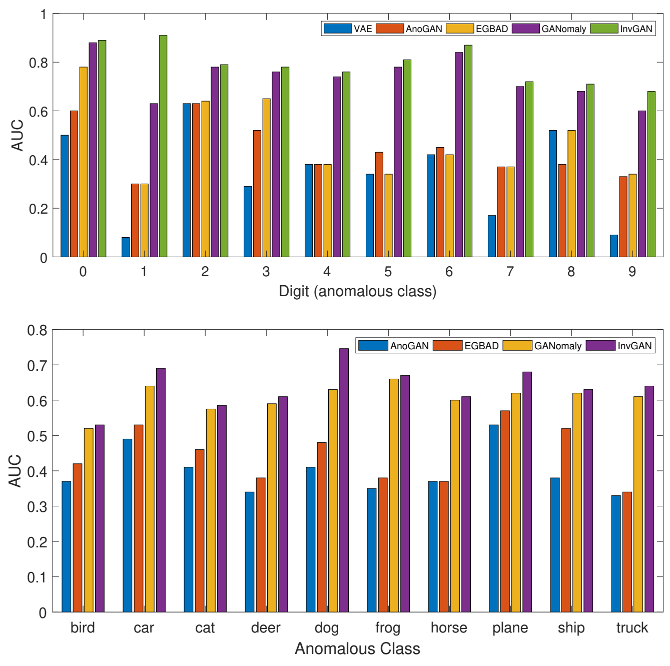

Figure 4: Anomaly detection with different models on the MNIST and CIFAR10 data sets.

set. As shown in the lower plot in Figure 4, our method leads the best results in all ten anomaly detection tasks, especially for the class dog. The anomaly detection results on two real data sets show that our model can perform well on the variance estimation for the distribution samples and outliers.

Table 3: Anomaly detection on MNIST and CIFAR10

Data Set	Model	AUROC
MNIST	VAE	0.3420 ±0.1861
	AnoGAN	0.4390 ±0.1116
	GANomaly	0.7390 ±0.0882
	InvGAN	**0.7920 ±0.0786**
CIFAR10	OC-SVM*	0.5843 ±0.0956
	IF*	0.6025 ±0.1040
	DSEBM-r*	0.6071 ±0.1007
	DSEBM-e*	0.5956 ±0.1151
	ALAD*	0.6072 ±0.1201
	AnoGAN	0.5949 ±0.1076
	EGBAD	0.4450 ±0.0786
	GANomaly	0.6065 ±0.0390
	InvGAN	**0.6391 ±0.0608**

Table 3 presents comparison among different anomaly detection methods for both image data sets. The experimental setups for AnoGAN, EGBAD, GANomaly, and the proposed InvGAN are the same and follow [1]. In order to obtain a comprehensive comparison with existing methods, we also include some anomaly detection results for CIFAR-10 from [35] (models marked with $*$ in Table 3).

Table 4: Anomaly detection on Arrhythmia data set

Data Set	Model	Precision	Recall	F1 Score
Arrhythmia	OC-SVM	**0.5397**	0.4082	0.4581
	IF	0.5147	0.5469	**0.5303**
	DSEBM-r	0.1515	0.1513	0.1510
	DSEBM-e	0.4667	0.4565	0.4601
	DAGMM	0.4909	0.5078	0.4983
	AnoGAN	0.4118	0.4375	0.4242
	ALAD	0.5000	0.5313	0.5152
	InvGAN	0.4390	**0.6000**	0.5070

We notice that there are some small experimental differences between [35] and what we used for CIFAR-10. In [35], they take one class any the normal one, and the rest classes as the abnormal ones. Anomaly detection is essentially a binary classification problem. The numerical results in Table 3 can validate the significance of the proposed model. In summary, the proposed method outperforms the other auto-encoder and GAN based approaches for both image data sets. More details about the implementation of the networks are given in Appendix A.

4.3.4 Arrhythmia. We further investigate the method on a tabular data set. The Arrhythmia [9] data set is obtained from the ODDS repository. The smallest classes, including 3, 4, 5, 7, 8, 9, 14, and 15, are combined to form the anomaly class, and the rest of the classes are combined to form the normal class. Table 4 shows the anomaly detection results with different methods. Due to the small training set, classic methods outperform deep generative models on precision and F1 score. However, our method achieves the highest recall values and a high F1 score as well. With the help of the variance network, our method is relatively robust compared with other deep neural network methods. More details about the implementation of the InvGAN can be found in Appendix B. Figure 5 gives the convergence of the loss $L_{h,\sigma}$ for different data sets.

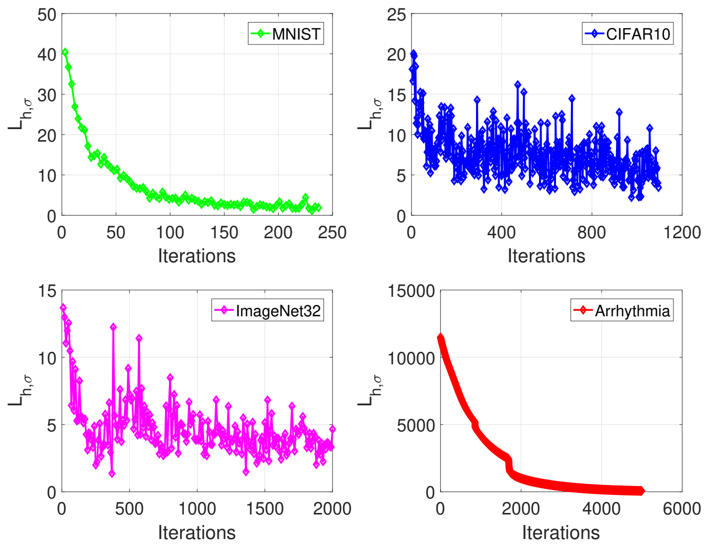

Figure 5: $L_{h,\sigma}$ at different iterations for different data sets.

5 CONCLUSIONS

In this paper, we propose an approach to estimate the implicit likelihoods of GANs. By leveraging an inference function and a variance network of the generator, the likelihoods of testing samples can be estimated. Simulation study and likelihood testing on real-world data sets valid the advantages of the proposed method. We further apply the method to three anomaly detection tasks. Experimental results show that the proposed approach can outperform classic and other deep neural network based anomaly detection methods. Our future work includes further theoretical study of the model and broadening the applications.

Appendix

A NETWORK STRUCTURES FOR IMAGE DATA SETS

The generator, discriminator, and the inference network for MNIST, CIFAR10 are given in Tables 5, 6, and 7. In the experiments, ImageNet32 use the same network structure as CIFAR10, and it is not stated in the tables.

Table 5: Generator of InvGAN in likelihood and anomaly detection experiments for MNIST and CIFAR10

Generator				
Layer	Number of Output	Kernel	Stride	Activation function
Input z~ $N(0,1)^{50}$	50			
Fully-Connected	128*4*4			ReLU
Transposed convolution	128*8*8	4*4	2	ReLU
Transposed convolution	64*16*16	4*4	2	ReLU
Transposed convolution	1*32*32 (MNIST) 3*32*32 (CIFAR10)	4*4 4*4	2 2	Tanh Tanh

Table 6: Discriminator of InvGAN in likelihood and anomaly detection experiments for MNIST and CIFAR10

Discriminator				
Layer	Number of Output	Kernel	Stride	Activation function
Input x	1*32*32 (MNIST) 3*32*32 (CIFAR10)			
Convolution	64*16*16	4*4	2	ReLU
Convolution	128*8*8	4*4	2	ReLU
Convolution	128*4*4	4*4	2	ReLU
Fully-Connected	1	1		Sigmoid

Table 7: Inference network of InvGAN in likelihood and anomaly detection experiments for MNIST and CIFAR10

Inference Network				
Layer	Number of Output	Kernel	Stride	Activation function
Input x	1*32*32 (MNIST) 3*32*32 (CIFAR10)			
Convolution	64*16*16	4*4	2	ReLU
Convolution	128*8*8	4*4	2	ReLU
Convolution	128*4*4	4*4	2	ReLU
Fully-Connected	50	50		

B NETWORK STRUCTURES FOR ARRHYTHMIA DATA SET

The network structures for the generator, the discriminator, and the inference network in the Arrhythmia experiments are given in Tables 8, 9, and 10, respectively.

Table 8: Generator of InvGAN in Arrhythmia anomaly detection experiments

Generator			
Layer	Number of Output	Batch Normalization	Activation function
Input z~ $N(0,1)^{z_{dim}}$	50		
Fully-Connected	128	Y	ReLU
Fully-Connected	256	Y	ReLU
Fully-Connected	274		

Table 9: Discriminator of InvGAN in Arrhythmia anomaly detection experiments

Discriminator			
Layer	Number of Output	Batch Normalization	Activation function
Input x	274		
Fully-Connected	256	Y	ReLU
Fully-Connected	128	Y	ReLU
Fully-Connected	1		Sigmoid

Table 10: Inference network of InvGAN in Arrhythmia anomaly detection experiments

Inference Network			
Layer	Number of Output	Batch Normalization	Activation function
Input x	274		
Fully-Connected	256	Y	ReLU
Fully-Connected	128	Y	ReLU
Fully-Connected	50		

REFERENCES

[1] Samet Akcay, Amir Atapour Abarghouei, and Toby P. Breckon. 2018. GANomaly: Semi-supervised Anomaly Detection via Adversarial Training. In *14th Asian Conference on Computer Vision (ACCV)*. Perth, Australia, 622–637.

[2] Jinwon An and Sungzoon Cho. 2015. *Variational Autoencoder based Anomaly Detection using Reconstruction Probability.* Technical Report.

[3] Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. *Wasserstein GAN.* Technical Report. arXiv:1701.07875

[4] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. 2018. Latent Space Oddity: on the Curvature of Deep Generative Models. In *6th International Conference on Learning Representations (ICLR)*. Vancouver, BC, Canada.

[5] Yu Bai, Tengyu Ma, and Andrej Risteski. 2019. Approximability of Discriminators Implies Diversity in GANs. In *7th International Conference on Learning Representations (ICLR)*. New Orleans, LA.

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2019. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *7th International Conference on Learning Representations (ICLR)*. Orleans, LA.

[7] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)*. Barcelona, Spain, 2172–2180.

[8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2017. Density estimation using Real NVP. In *5th International Conference on Learning Representations (ICLR)*. Toulon, France.

[9] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[10] Nicolas Fournier and Arnaud Guillin. 2015. On the rate of convergence in Wasserstein distance of the empirical measure. *Probability Theory and Related Fields* 162, 3-4 (2015), 707–738.

[11] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. 2015. MADE: Masked Autoencoder for Distribution Estimation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Lille, France, 881–889.

[12] Alvina Goh and René Vidal. 2008. Clustering and dimensionality reduction on Riemannian manifolds. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Anchorage, Alaska.

[13] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. 2016. *Deep Learning.* MIT Press.

[14] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. 2019. FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In *7th International Conference on Learning Representations (ICLR)*. New Orleans, LA.

[15] Aditya Grover, Manik Dhar, and Stefano Ermon. 2018. Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*. New Orleans, LA, 3069–3076.

[16] Andrew J. Hanson. 1994. *Graphics gems iv. chapter Geometry for N-dimensional Graphics.* Academic Press Professional, Inc.

[17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *6th International Conference on Learning Representations (ICLR)*. Vancouver, BC, Canada.

[18] Diederik P. Kingma and Prafulla Dhariwal. 2018. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*. Montréal, Québec, Canada, 10236–10245.

[19] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations (ICLR)*. Banff, AB, Canada.

[20] Steven Krantz and Harold Parks. 2008. *Analytical Tools: The Area Formula, the Coarea Formula, and Poincaré Inequalities.* Birkhäuser Boston, Boston, 1–33.

[21] Alex Krizhevsky. 2009. *Learning Multiple Layers of Features from Tiny Images.* Technical Report. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[22] Abhishek Kumar, Prasanna Sattigeri, and Tom Fletcher. 2017. Semi-supervised Learning with GANs: Manifold Invariance with Improved Inference. In *Advances in Neural Information Processing Systems*. Long Beach, CA, 5534–5544.

[23] Neil D. Lawrence. 2003. Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. In *Advances in Neural Information Processing Systems (NIPS)*. Vancouver, BC, Canada, 329–336.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86 (1998), 2278–2324.

[25] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*. Pisa, Italy, 413–422.

[26] George Papamakarios, Iain Murray, and Theo Pavlakou. 2017. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems (NIPS)*. Long Beach, CA, 2338–2347.

[27] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, and David Cournapeau. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 12825–2830.

[28] Qichao Que and Mikhail Belkin. 2016. Back to the Future: Radial Basis Function Networks Revisited. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cadiz, Spain, 1375–1383.

[29] Aditya Ramesh and Yann LeCun. 2018. *Backpropagation for Implicit Spectral Densities.* Technical Report. arXiv:1806.00499

[30] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. 2017. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In *Proceedings of 25th International Conference on Information Processing in Medical Imaging (IPMI)*. Boone, NC, 146–157.

[31] Bernhard Schölkopf, Robert C. Williamson, Alexander J. Smola, John Shawe-Taylor, and John C. Platt. 1999. Support Vector Method for Novelty Detection. In *Advances in Neural Information Processing Systems (NIPS)*. Denver, CO, 582–588.

[32] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles A. Sutton. 2017. VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *Advances in Neural Information Processing Systems (NIPS)*. Long Beach, CA, 3308–3318.

[33] Tao Yang, Georgios Arvanitidis, Dongmei Fu, Xiaogang Li, and Søren Hauberg. 2018. *Geodesic Clustering in Deep Generative Models.* Technical Report. arXiv:1809.04747

[34] Haiyan Ying, Dingcheng Li, Xu Li, and Ping Li. 2020. Meta-CoTGAN: A Meta Cooperative Training Paradigm for Improving Adversarial Text Generation. In *The Thirty-Forth AAAI Conference on Artificial Intelligence (AAAI)*. New York, NY.

[35] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. 2018. Adversarially Learned Anomaly Detection. In *IEEE International Conference on Data Mining (ICDM)*. Singapore, 727–736.

[36] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. 2016. Deep Structured Energy Based Models for Anomaly Detection. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*. New York, NY, 1100–1109.

[37] Chong Zhou and Randy C. Paffenroth. 2017. Anomaly Detection with Robust Deep Autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. Halifax, NS, Canada, 665–674.

[38] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *IEEE International Conference on Computer Vision (ICCV)*. Venice, Italy, 2242–2251.

[39] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Dae-ki Cho, and Haifeng Chen. 2018. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In *6th International Conference on Learning Representations (ICLR)*. Vancouver, BC, Canada.