Quantitative Evaluation of Generative Adversarial Networks and Improved Training Techniques

by

Yadong Li

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday June 1, 2018 at 15:45 PM.

Student number:	4608283	
Project duration:	September 1, 2017 – June 1, 2018	
Thesis committee:	Prof. M. J. T. Reinders,	TU Delft
	Dr. D. M. J. Tax,	TU Delft, supervisor
	Dr. ir J. N. Driessen,	TU Delft

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

TU Delft Delft University of Technology

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Objectives . . . 4
- 1.3 Outline . . . 5

2 Related Work **7**
- 2.1 Deep Learning . . . 7
- 2.2 Generative Models . . . 8
- 2.3 Generative Adversarial Networks . . . 9
- 2.4 Evaluating Generative Adversarial Networks . . . 12
- 2.5 Improving Techniques for GANs . . . 15
  - 2.5.1 Alternative Training Objectives . . . 15
  - 2.5.2 Regularization . . . 17

3 Quantitative Evaluation on Artificial Datasets **19**
- 3.1 Artificial Datasets . . . 19
- 3.2 Evaluating The Diversity . . . 20
- 3.3 Evaluating The Quality . . . 23

4 Technique for improving the performance of GANs **25**
- 4.1 Adding Regularization . . . 25
- 4.2 Smooth-to-sharp Training . . . 28

5 Experiments **33**
- 5.1 Exp 1: 1D Examples . . . 33
- 5.2 Exp 2: Evaluation of Adding Regularization . . . 35
- 5.3 Exp 3: Evaluation of GANs' Performance on Images with Different Smoothness . . . 39
- 5.4 Exp 4: Smooth-to-sharp Training Method . . . 43
- 5.5 Exp 5: Experiments on the MNIST Dataset . . . 46
- 5.6 Exp 6: Wasserstein Distance, Diversity and Quality . . . 52

6 Summary and Conclusions **57**
- 6.1 Summary . . . 57
- 6.2 Strengths and Limitations . . . 58
- 6.3 Future work . . . 59

References . . . 60

1 Introduction

1.1. Motivation

Generative models belong to a branch of unsupervised learning techniques in machine learning, for which the goal is to learn from training data in a certain domain (e.g., images, sentences, or sounds, etc.), and then to generate data similar to them. From a statistical point of view, generative models take a training set sampled from a data distribution, and learn to represent an estimate of the probability distributions of the given data.

Traditional generative models provide a parametric specification of a probability distribution function (*pdf*) and can then be trained by maximizing the log likelihood. However, such models have limited ability to generative high dimensional data, such as images. The limitation stems from the fact that high dimensionality generally makes the computation intractable and that high dimensional space is almost empty with meaningful samples lying in some specific locations, which makes the estimation of *pdf* very difficult.

Generative Adversarial Networks (GANs), first proposed by Ian Goodfellow in 2014[1], are a class of generative models that has achieved considerable success in modeling high dimensional data such as images. A simple GAN is composed of two parts: a generator and a discriminator. Both the generator and the discriminator are artificial neural networks with a large number of learnable parameters. The generator takes random noises as input and outputs samples that look like the true data. The discriminator is an adversarial classifier that is trained to discriminate between samples from the true data distribution and samples from the generator. The discriminator is introduced in GANs to provide signals to the generator, based on which we could optimize parameters of the generator to make its generated samples more confusing for the discriminator. Therefore, these two networks are in competition with each other in a way that the discriminator is trying to distinguish real samples from fake ones produced by the generator, while the generator is trying to create samples that make the discriminator think they are real. Thus, GANs do

1

not make estimate of a *pdf* directly but learn to generate samples according to the *pdf*, which makes modeling high dimensional data an easier task.

GANs have gained notable success in visual domains, such as realistic images generation [2–4], semantic meaningful representation learning [5], and image-to-image translation [6]. Figure 1.1 exemplifies the performance of GANs in generating handwritten digits using the MNIST dataset[7] as the training dataset. The figure on the left shows original handwritten digits from the true MNIST dataset, containing images of handwritten digits with different classes and with various thickness, orientation and shape. GANs take samples from the training dataset and learn to generate similar samples, shown in the figure on the right. The generated images contain all possible digits and look realistic, but not exactly the same as the digits in the training dataset.

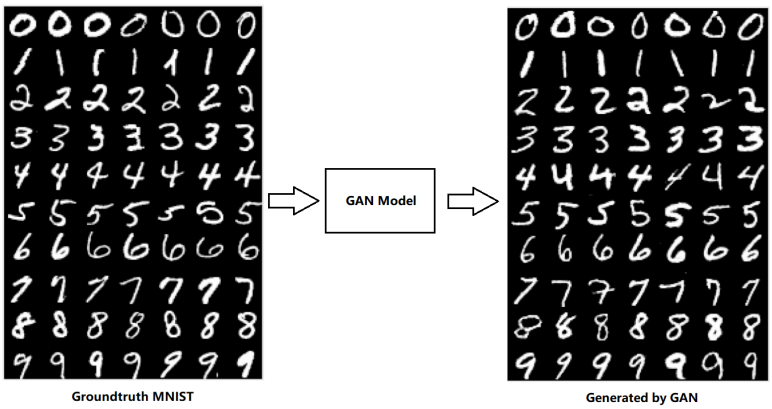


Figure 1.1: An illustration of what GANs would generate. A well-trained GAN would be able to train on examples (MNIST [7]) as shown on the left and then create similar examples from the same distribution as shown on the right. Figure reproduced from [8].

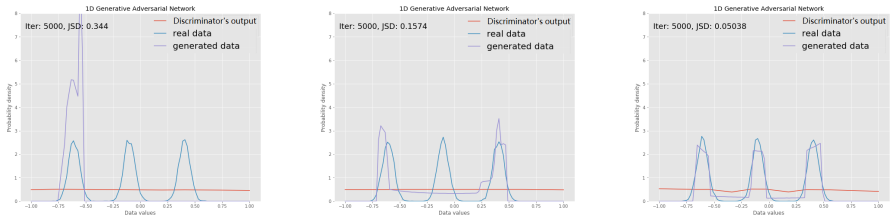
While GANs achieve compelling results in visual domain, the drawback is that it is hard to evaluate. The challenge mainly comes from the fact that GANs, unlike some generative models that define an explicit *pdf*, directly generate samples according to the input without defining a *pdf*. In low dimensional toy data, we may estimate the density function $p_{model}(x;\theta)$ by Parzen density estimation, but it is impractical for high dimensional data such as images. Thus, classic measures, such as computing log-likelihood on test data, is not applicable.

If we only have access to generated samples, what are the alternative ways to evaluate GAN's performance? One intuitive method is to visually inspect the quality of generated samples. Images that contain meaningful objects and that look realistic and detailed are considered as of high visual quality. For instance, the generated images in figure 1.1 on the right-hand side contain meaningful digits and look realistic with clear background, thus these images are of high visual quality.

Additionally, we are not only interested in generating a single high-quality image, but also in generating different samples that could cover all variations in the training

set, such as different objects with different shapes. These variations are also known as different modes in one dataset.

However, mode collapse is a well-recognized problem in GANs [5, 8, 9], where GANs only generate data with some modes in the training data. Figure 1.2 illustrate mode collapse in 1d space. In this 1d space, the real data distribution is a mixture of three Gaussian distributions that has three modes. Ideally, we would like GANs to generate all three modes, as shown in figure 1.2(c), however, In figure 1.2(a) and 1.2(b), GANs are only able to generate data from one or two modes and mode collapse occurs.



(a) Mode collapse example: generating only one mode

(b) Mode collapse example: generating only two modes

(c) No mode collapse: generating all three modes

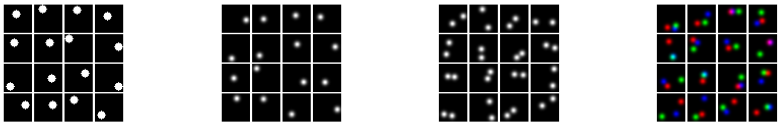
Figure 1.2: An example of mode collapse: the real data distribution has three modes(colored in blue). 1.2(a) shows the situation where GANs only generate data from one mode, and figure 1.2(b) shows the situation where GANs only generate data from two modes. 1.2(c) shows the ideal case where all three modes are generated.

To summarize, we would like to evaluate GANs from two aspects: the diversity and the quality. The diversity evaluates GANs' ability to generate diverse samples capturing different modes in real data and the quality evaluates GANs' ability to generate realistic samples. Take the groundtruth MNIST samples and generated samples in figure 1.1 as an example. For a well-trained GAN, we would like it to generate not only digits that look realistic, but also images that cover all digits from 0 to 9.

A common method adopted by researchers to evaluate GANs, as briefly discussed before, is to inspect the visual quality of generated samples. However, this manual work suffers from low scalability and high subjectivity. Meanwhile, unconscious bias from subjective evaluation is inevitable, such as when researchers report only their best results and lead to unfair comparisons for different GANs. Moreover, a human observer could probably overlook mode collapse and fail to evaluate diversity. Although for simple dataset like the MNIST, it is not difficult for a human observer to notice the missing digits by inspecting a sequence of samples, in more realistic settings, where a GAN is trained on data with tens of thousands of modes, a human observer would hardly detect the missing variation by manual inspection.

Since evaluating log likelihood is intractable for images and since manual inspection is not a reliable and scalable approach, we hereby propose a quantitative evaluation approach for GANs using an artificial image dataset. The artificial dataset, as shown in figure 3.1, contains images that have one or more artificial

spheres in different locations. By introducing this artificial dataset, we limit the evaluation of GANs to a specific dataset, and enables the quantitative evaluation from both the diversity aspect and the quality aspect. The variation of images, or different modes within this artificial dataset, is entirely represented by difference locations of spheres, thus we are able to assess GANs' ability to generate different samples, which is the diversity, by examining the locations of generated spheres. In the meantime, since we know the groundtruth shape of these spheres, we are able to assess GANs' ability to generate realistic samples, which is the quality, by comparing the generated samples and the ground-truth images in a pixel-by-pixel manner.



(a) Sharp sphere examples (b) Gaussian sphere examples (c) Two-sphere examples (d) Stack sphere examples

Figure 1.3: Illustration of our artificial image dataset: each sample image is located in 4×4 grids.

Objective evaluation of GANs' performance is a challenging but important task. It is essential for two reasons. First, as a variety of GANs are promoted for diverse training objectives and with modified training techniques, it helps researchers compare different GANs fairly and objectively. Second, to make faster progresses towards better algorithms and models, it is useful to find out which modifications have significant influence on GANs' performance.

In this work, we also discover two improved techniques for training GANs with the help of our quantitative evaluation scheme. First, adding proper regularization on networks improve and stabilize the performance. Second, we develop a Smooth-to-Sharp training framework to improve the performance, in which training starts with smooth images and progresses gradually to sharp ones.

1.2. Objectives

The aim of our work is to quantitatively evaluate the performance of GANs and to discover improved techniques for training GANs with the help of our developed evaluation scheme.

1. To develop quantitative evaluation scheme for GANs' performance. We would like to evaluate GANs' performance from two aspects: the diversity and the quality.

2. To propose training techniques to improve GANs' performance. We proposed two techniques, adding proper regularization and the "Smooth-to-Sharp" training framework.

Our work fills current research gaps from four aspects. First, we invent an artificial image dataset that is easy to create and available for quantitative evaluation. Second, we propose a quantitative approach to evaluate GANs based on both diversity and quality and use our Artificial image dataset to train different GANs and to make quantitative comparisons. Third, we empirically analyze the benefits of regularization on GANs and validate our idea by experimental results. Forth, we develop a ”smooth-to-sharp” training method, which boosts GANs’ performance for sharp images.

1.3. Outline

The outline of this thesis report is as follows. Chapter 2 gives background knowledge of GANs and summarizes related work of evaluating and improved training of GANs from literature. Chapter 3 introduces the proposed toy dataset and quantitative evaluation scheme. Chapter 4 discusses two ways to improve the training of GANs: adding regularization and the ”Smooth-to-Sharp” training. Chapter 5 presents key experiments and their results. Finally, chapter 6 summarizes this work, discusses the results and possible continuations of our research.

2

Related Work

2.1. Deep Learning

Deep Learning is a sub-field of machine learning, which uses multiple layers of artificial neurons to learn representations [10]. It has gained notable success in speech recognition, visual object recognition, and many other domains. The basic building blocks for most deep learning approaches are artificial neurons with trainable parameters, and these parameters are trained by the backpropagation procedure [11].

Neural networks The most commonly used architecture in Deep Learning is called deep neural networks. A neural network is composed of connected artificial neurons. Figure 2.1 illustrates the idea of such a neuron.

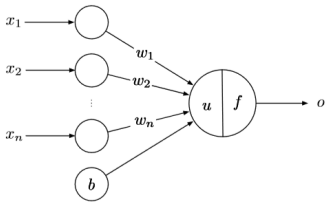


Figure 2.1: An illustration of a single neuron with n inputs, $n+1$ learnable weights and bias parameters. After the affine transformation, a function f, referred as activation functions, is applied.

Each neuron applies an affine transformation of the input $\mathbf{x} = [x_1, x_2, ..., x_n]^T$:

$$u = \sum_{i=1}^{n} w_i x_i + b, \tag{2.1}$$

where w_i is the weight corresponding to x_i and b is the bias term. A non-linear activation function f is applied after this affine transformation:

$$o = f(u). \tag{2.2}$$

Thus, we get the final output o after the affine transformation and the non-linear activation. By connecting multiple neurons in different layers, a neural network is formed. Figure 2.2 exemplifies a three-layer neural network.

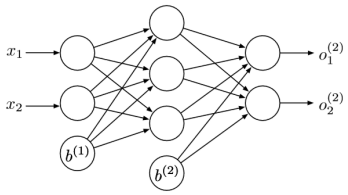


Figure 2.2: A three-layer neural network with two inputs, three hidden neurons and two outputs.

Optimization Neural networks are usually initialized with random weights and biases, and these parameters need to be updated such that a defined scalar cost J is minimized. This learning process is achieved by the gradient descent algorithm using the gradient computed by backpropagation [11]. A specialized set of gradient descent techniques has been developed to optimize neural networks, including Stochastic Gradient Descent (SGD) [12], RMSProp [13], Adam [14], and so forth.

Universal approximation theorem The universal approximation theorem states that a neuron network with at least one hidden layer can approximate any continuous function under mild assumptions, provided that the network is given enough hidden units[15]. However, we are not guaranteed that the optimization algorithm will be able to learn the arbitrary function, although the network is able to represent the function.

2.2. Generative Models

Generative models refer to any model that takes a training set sampled from a distribution P_{data}, and then learns to represent an estimate P_{model}. Typical generative models are density models that explicitly provide an estimate of the probability density function. A parametric Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ is an example of the simplest generative model, which gives an explicit probability density function as μ and σ are learned from training samples. More complicated density models include the Parzen density [16], Boltzmann machines [17] and so forth.

However, such density models have a limited ability to model and generate high dimensional data, such as images. First, most density models rely on a prior specified parametric family, which makes assumptions of the estimated probability distribution and limits the functional form of it. Furthermore, estimating the density

is often computationally intractable since high dimensional space is almost empty, with meaningful samples lying in some specific locations [18]. Lastly, if we are interested in generating new samples, the typical method is *inverse transform sampling*, which involves computing the cumulative distribution function of the estimated distribution and inverting the computed function. However, this method is computationally intractable for high dimensional data. Other approximation methods, such as the Markov Chain Monte Carlo (MCMC), are also inefficient in high-dimensional spaces[19].

These difficulties motivate the development of "generative machines" [1]: models that do not explicitly represent the density function yet are able to generate samples from the desired distribution. By avoiding defining the density explicitly, the generative machines often have fewer restrictions and are more flexible to the desired distribution, compared to the traditional density models. Moreover, the generative machines are capable of generating new samples directly, some of which are even capable of generating a batch of samples in parallel. One important representative of generative machines, with remarkable success achieved in recent years, is the Generative Adversarial Networks (GANs).

2.3. Generative Adversarial Networks

Framework GANs are a class of generative machines with the capability to generate samples directly and are proposed by Goodfellow et al. [1] in 2014 as a novel framework for training generative models. A simple GAN consists of two components: a generator *G* and a discriminator *D*. Goodfellow and his colleagues proposed an adversarial training process where these two components are simultaneously trained. *G* is trained to generate samples that look realistic, while *D* is trained to distinguish the samples produced by *G* from those belonging to the real dataset. The most commonly used architecture for both *G* and *D* are artificial neural networks. As stated by the universal approximation theorem, the advantage of using artificial neural networks is that these networks are capable of modelling very complex mappings, and that they can be optimized by backpropagation algorithm.

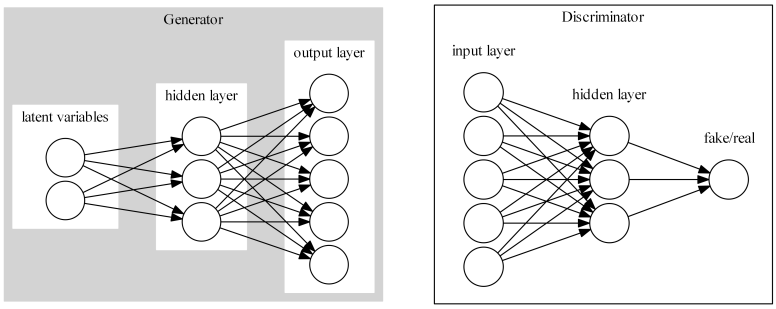
An example of a GAN is shown in figure 2.3, in which both *G* and *D* are neural networks with one hidden layer. G is a generative function that takes random inputs drawn from a prior distribution z (i.e., uniform or normal distribution), which are often referred as *latent variables*, and outputs generated samples $x \sim p_{model}$:

$$x = G(z),\ x \sim p_{model} \tag{2.3}$$

D is a discriminative function that maps samples x from the true distribution p_{data} or the generated distribution p_{model} to a scalar o to determine whether the input samples are fake or real:

$$o = D(x),\ x \sim p_{model}\ or\ x \sim p_{data} \tag{2.4}$$

Training GANs At first, GAN is initialized with random weights, so a random latent variable input to G would be output as a completely random image. Our



(a) Generator (b) Discriminator

Figure 2.3: An example of the structure of the Generator and the Discriminator. Generator has 2D latent variables and one hidden layer with three units. The discriminator takes real data or generated data as input, and outputs a scalar indicating the input being real or fake.

goal is to train these initial random weights in order to make the generated image samples look similar to the training data. In other words, we aim to match the generated image data distribution to the true data distribution. G gets trained by receiving training signals from D when we backpropagate through both D and G to learn to change the weights for G in order to make generated images more confusing for D. Therefore, these two networks are in competition with each other in a way that D is trying to distinguish real samples from fake ones produced by G, while G is trying to create samples that make D think they are real. In theory, G could eventually reproduce the true data distribution and D would be unable to find a difference. More formally, D and G are jointly trained with the objective function defined by the following minmax game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{data}}[\log D(x)] + \mathbb{E}_{z\sim P_z}[\log(1 - D(G(z)))], \tag{2.5}$$

Figure 2.4 gives a visualization of the training progress of GANs in 1D space. The real data distribution is a Gaussian distribution, and the GAN is trained to generate samples according to a similar distribution. At the beginning of the training process (2.4(a)), G generates samples that lie far away from the real samples. The real samples follow a normal distribution and the generated samples are randomly distributed. The learning signal provided by D guides G to generate samples that are gradually closer to the real samples, as shown in 2.4(b) 2.4(c). Finally, the generated samples lie very similar to real samples, and D cannot distinguish fake from real (2.4(d)).

Optimality The loss functions for respective models are shown as:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim P_{data}}[\log D(x)] - \mathbb{E}_{x\sim P_{model}}[\log(1 - D(x))] \tag{2.6}$$

$$\mathcal{L}_G = \mathbb{E}_{x\sim P_{model}}[\log(1 - D(x))] \tag{2.7}$$

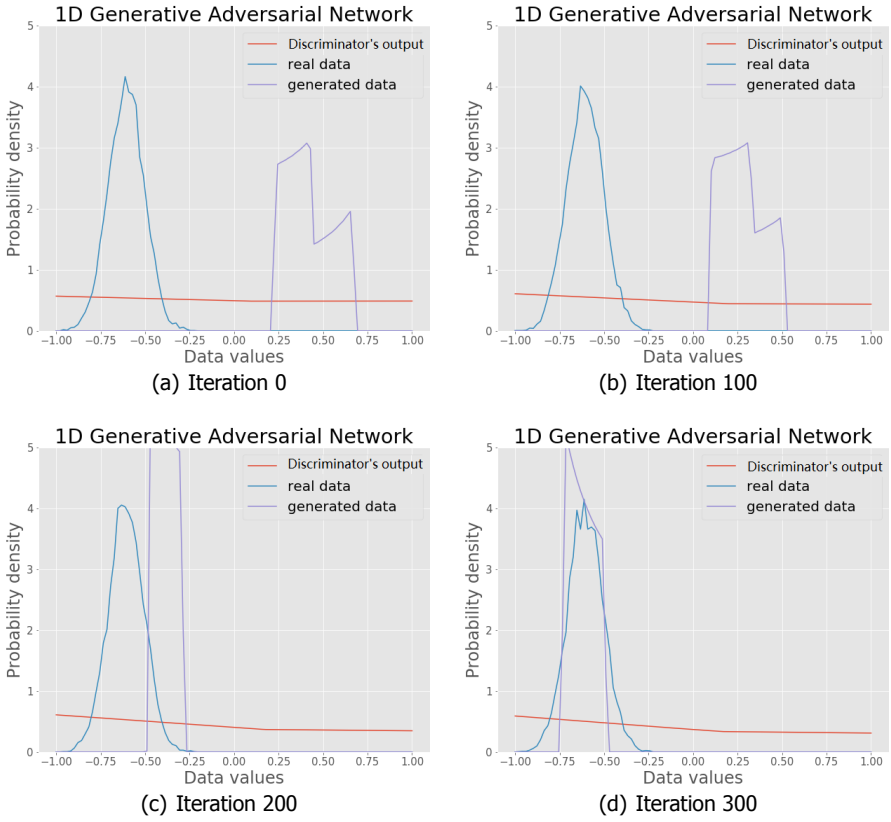


Figure 2.4: An example of the training progress in 1D space. The real data distribution is a Gaussian distribution, and GANs are trained to generate samples according to a similar distribution. The red line shows the output of the discriminator. It gives higher value to real samples and thus guides the generator to generate samples move towards real samples.

For a fixed generator G, the optimum discriminator D^* given a sample x is [1]:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{model}(x)} \tag{2.8}$$

The loss for a optimum discriminator would be: [20]:

$$\mathcal{L}_{D^*} = -2D_{JS}(P_{data} \parallel P_{model}) + 2\log 2, \tag{2.9}$$

where D_{JS} is the Jensen-Shannon divergence. The Jensen-Shannon divergence is a symmetrical, non-negative measure of the divergence between two probability distributions p and q. It is zero if and only if $p = q$. Thus, $\mathcal{L}_{D^*}$ has a global minimum of $-\log 4$ if and only if $p_{model} = p_{data}$. At this optimum, the optimal discriminator D^* cannot do better than randomly guessing and output 0.5 for any sample x (see Eq. 2.8).

2.4. Evaluating Generative Adversarial Networks

In this section, we summarize current evaluation methods that are popularly applied. We are interested in evaluating GANs from two aspects: the diversity and the quality. From the diversity aspect, we evaluate GANs' ability to generate diverse samples capturing different modes in the real data; and from the quality aspect, we evaluate GANs' ability to generate realistic samples.

Visual Inspection GANs are generative machines that are able to generate samples directly, thus practitioners often evaluate GANs by visually inspecting generated samples.

One straightforward way is to visualize a batch of samples generated by GANs, and to evaluate their qualities by researchers themselves. This method is deployed in many research works [1, 2, 8, 21, 22]. Figure 2.5 exemplifies this method. In this work, the authors visualized 24 samples from four different GAN models and compared their performance by visually inspecting the results. Although they claimed the top-left model be the best, it was very subjective. It is generally very hard to make a fair comparison based on visual inspection.

Another approach deployed by Salimans et al. [8] is to automate the human evaluation process using the Amazon Mechanical Turk, a crowd-sourcing platform on which researchers asked annotators to distinguish between generated images and real images.

Unfortunately, these manual works suffer from low scalability and high subjectivity. Unconscious bias from subjective evaluation is inevitable, such as when researchers report only their best results, which lead to unfair comparisons for different GANs. Moreover, a human observer could probably overlook mode collapse and fail to evaluate the diversity. Although for simple dataset like the MNIST, it is easy for a human observer to notice the missing digits by inspecting a sequence of samples, in more realistic settings, where a GAN is trained on data with tens of thousands of modes, a human observer would hardly detect the missing variation by manual inspection.

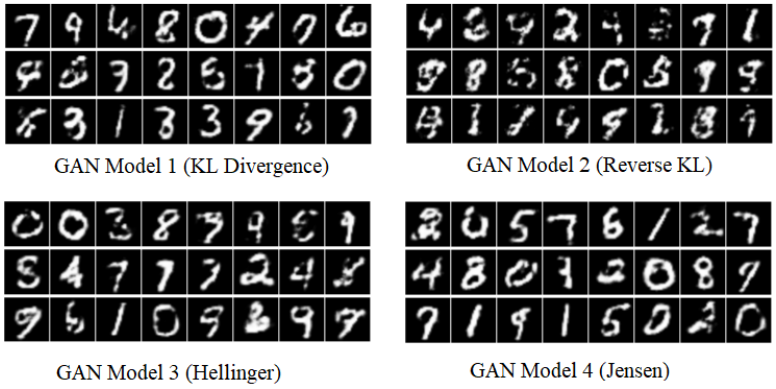


Figure 2.5: An example of manual inspection method used by researchers (Figure from [21]). They compared four different GANs by visualizing selected samples and by manually inspecting them.

Likelihood Method For generative models that define an explicit probability density function (*pdf*), the standard way to evaluate the performance is to compute the log-likelihood on reserved test data. We first reserve m samples as test data: $\{x_1, x_2, ..., x_m\}$, and then estimate the probability density $p_{model}(x)$:

$$LL = \sum_{i=1}^{m} \log(p_{model}(x_i)), \tag{2.10}$$

GANs do not define an explicit *pdf* but generate samples according to the *pdf*. When the *pdf* is not available, one common alternative is to use density estimators such as Parzen window method. Let x_i be a sample drawn from a distribution with an unknown density p. Parzen window method estimates the following function p:

$$\hat{p}_h(x) = \frac{1}{n} \sum_{i=1}^{n} \Phi_h(x - x_i), \tag{2.11}$$

where Φ is a window function parameterized by the bandwidth h.

Generated samples are used to fit such a density estimator. Then the log-likelihood of the test data is evaluated under this estimator and is used as a proxy for the true model's log-likelihood. This method, first introduced by Breuleux et al. [23], are widely used by researchers [1, 22, 24] to evaluate GANs. However, this method of estimating the likelihood has high variance and does not perform well in high dimensional spaces [1]. Theis et al. [18] also provided examples showing that in high dimensional spaces the Parzen window estimation of the likelihood is biased and is even unable to produce useful rankings when comparing different models. Thus, the authors suggested that this method should be avoided for evaluating generative models in high-dimensional spaces.

Evaluation by A Well-trained Classifier A more indirect evaluation method is to apply a well-trained classifier to the generated samples and calculate statistics of its output or at a particular hidden layer.

One widely used metric taking this approach is the Inception Score [8]. The inception Score (IS) uses a well-trained neural network and calculates a statistic of the network's outputs when applied to generated samples:

$$IS(G) = \exp(\mathbb{E}_{x \sim p_{model}}[D_{KL}(p(y|x) \parallel p(y)]), \tag{2.12}$$

where $x \sim p_{model}$ states that x is a sampled image from p_{model}, $p(y|x)$ is the conditional class distribution, $p(y) = \int p(y|x)p_{model}(x)dx$ is the marginal class distribution, and $D_{KL}(p \parallel q)$ is the *Kullback-Leibler (KL) divergence* between the distributions p and q.

The KL divergence measures how different two distributions are:

$$D_{KL}(p \parallel q) = \mathbb{E}_{x \sim p}[\log p(x) - \log q(x)]. \tag{2.13}$$

One notable property of the KL divergence is that it is non-negative. The KL divergence is 0 if and only if P and Q are the same. Since the KL divergence is non-negative and measures the difference between these distributions, it is often conceptualized as measuring a certain distance between these distributions.

The authors proposed the IS to combine two objectives into a metric:

1. The images generated should be contain meaningful objects, or the conditional class label distribution $p(y|x)$ should have low entropy.

2. The variability of samples should be high so the marginal $\int p(y|x = G(z))dz$ should have high entropy.

If both of these desirable qualities are satisfied by a GAN, then we expect a large KL-divergence between $p(y)$ and $p(y|x)$, resulting a large IS.

Another classification based method is the Frechet Inception distance (FID), proposed by Heusel [25]. Instead of using the class label distribution, the FID makes use of a certain layer of Inception Net to compare embeddings of the real and the generated data. Although the FID does not perform a classification task as the IS, it still needs a pre-trained classifier. Thus, we categorize the FID as a classification based approach. The FID views the embedding layer as a continuous multivariate Gaussian, and the mean and the covariance is estimated for both the generated data and the real data. Then, the Frechet distance between these two distributions is computed as:

$$FID(x, g) = ||\mu_x - \mu_g||_2^2 + Tr(\Sigma_x + \Sigma_g - 2(\Sigma_x\Sigma_g)^{\frac{1}{2}}), \tag{2.14}$$

where μ_x, μ_g are the means of embeddings from the true data distribution and the generated distribution, and Σ_x, Σ_g are the covariances.

These two classification based approaches have at least three drawbacks. Firstly, their evaluation largely relies on a well-trained classifier. GANs are unsupervised learning frameworks that do not require labels but class labels for training data

are required in order to train classifiers. Although we could evaluate GANs' performance of generating images using classifiers that are trained on a different image dataset, for instance, the ImageNet [26], the mismatch of datasets would probably introduce biases. Secondly, when evaluating the diversity of generated data, these approaches only capture the variation of different classes but fail to capture the variation of different shapes within one class. Thirdly, when evaluating the quality of generated data, these approaches assume that images classified with high confidence are of high quality, but this assumption is not always true.

Evaluating on An Artificial Image Dataset In this thesis, we create an artificial image dataset to evaluate GANs. Currently available datasets for evaluation are either too simple, for example, of a mixture of Gaussians in 1D or 2D space, or too complex, for example, the ImageNet [26]. By introducing an artificial dataset, we limit the evaluation of GANs to a specific dataset, and enable the quantitative evaluation from both the diversity aspect and the quality aspect. The artificial dataset we propose has two properties. First, the variation of images within the dataset is controllable and is easy to detect, and thus we are able to examine GANs' ability to generate diverse samples. Second, the content of images are relatively simple and are easy to compare in image spaces. Therefore, we can evaluate GANs' ability to generate high quality samples by comparing these samples with ground-truth images in a pixel-by-pixel manner.

To our best knowledge, only one published paper[27] discussed evaluating GANs quantitatively using an artificial image dataset. The authors created a simple image dataset: the manifold of convex polygons, as shown in Figure 2.6. They proposed two evaluation aspects, which are the precision and the recall. The precision aspect is similar to the quality in our evaluation metric, and the recall aspect is similar to the diversity in our evaluation metric. However, their method requires solving non-convex optimization problem and inverting the generator, and they have not open-sourced their setups and implementations yet.

2.5. Improving Techniques for GANs

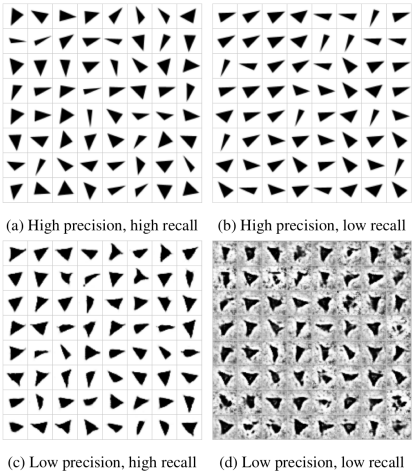
2.5.1. Alternative Training Objectives

$f-$GAN Nowozin et al. [21] showed that GANs can be viewed as a general divergence estimation principle. The authors extended the GAN framework to minimize any divergence measure belonging to a class of divergences called f-divergence.

$f-$divergence is a function measuring the difference between two probability distributions. Given two probability distribution P and Q, with continuous density functions p and q defined over a domain χ, $f-$divergence is defined as:

$$D_f(P||Q) = \int_{\chi} q(x) f\left(\frac{p(x)}{q(x)}\right) dx, \tag{2.15}$$

where f is a convex function. For instance, the Kullbeck-Leibler divergence belongs to this class of divergence. For a list of different $f-$divergences, see [21].



(a) High precision, high recall (b) High precision, low recall

(c) Low precision, high recall (d) Low precision, low recall

Figure 2.6: Figure taken form [27]. The authors created an artificial dataset: the manifold of convex polygons.

The authors also provided an alternative view of the training algorithm of GANs. In this view, the discriminator tries to estimate a lower bound of a chosen divergence measure, while the generator tries to minimize the estimated divergence.

The training objective of the original GAN [1] is to minimize the Jensen-Shannon divergence. However, there are many different choices of f-divergences possible. We can easily adapt the original GAN formulation to minimize other divergences.

Wasserstein GAN In [20], the authors analyzed why the traditional GAN is hard to train. The reason lies in the divergence, Jensen-Shannon divergence used in the original GAN [1] cannot accurately measure the distance between two distributions when the overlapped part is negligible. Thus, in [28] the authors proposed to use Wasserstein distance to measure the differences between the real distribution and the fake one. The Wasserstein distance, also known as the *Earth Mover* distance [29], of two distributions P and Q is defined as:

$$W(P,Q) = \inf_{\gamma \in \mathrm{T}} \mathbb{E}_{(x,y)\sim\gamma} \left[\| x - y \| \right], \tag{2.16}$$

where T denotes the set of all joint distributions with marginal distributions P and Q.

Solving Eq.(2.16) is computationally intractable. However, it can be re-formulated using the *Kantorovich-Rubinstein duality* [30] to

$$W(P,Q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim P} \left[f(x) \right] - \mathbb{E}_{x\sim Q} \left[f(x) \right]. \tag{2.17}$$

The supreme is over all 1-Lipschitz functions that satisfy: $\| f(x) - f(y) \| \leq \| x - y \|$.

If we choose the Wasserstein distance as the divergence measure, the original GAN formulation is rewritten as:

$$\min_G \max_D V(D,G)_{D\in 1-Lipschitz} = \mathbb{E}_{x\sim P_{data}}[D(x)] - \mathbb{E}_{z\sim P_z}[D(G(z))], \tag{2.18}$$

where the 1-Lipschitz constraint is enforced by weight-clipping [28] or by penalizing the gradient norm of the discriminator [31].

2.5.2. Regularization

GANs are generally considered to be hard to train. Meanwhile, mode collapse is a common failure pattern. These issues have been addressed in several recent papers [8, 9, 32] proposing training techniques to stabilize GAN's training and encourage its samples' diversity, many of which can be categorized as regularization strategies.

Regularization on Parameters Many regularization methods aim at limiting the capacity of models, by adding a parameter penalty $\Omega(\theta)$ to the objective function J. We denoted the regularized objective function by $\tilde{J}$:

$$\tilde{J}(\theta) = J(\theta) + \alpha\Omega(\theta), \tag{2.19}$$

where α is a non-negative hyperparameter that controls the relative weight of the penalty term, Ω.

One common choice of parameter penalty is the L2-Regularization, commonly known as *weight decay*. This strategy adds a regularization term

$$\Omega(\theta) = \frac{1}{2} \parallel \theta \parallel_2^2 \tag{2.20}$$

to the objective function.

The 1-Lipschitz constraint (2.18) in Wasserstein GAN can be viewed as a regularization on the discriminant function. In [28], this constraint is implemented via weight clipping. That is, simply clipping all weight values that are outside of some allowed range. Recently, [31] proposed a gradient penalty regularization method, which enforces the 1-Lipschitz constraint by penalizing the gradient norm of the discriminator.

Adding Noise and Data Augmentation Some work suggested to add continuous noise to the inputs of the discriminator[20, 33]. Adding noise can be seen as a process of constructing new inputs, and thus it is considered as a form of data augmentation. Data augmentation is an effective way to improve the robustness of a model and to make it generalize better. For GANs, adding noise to the inputs of the discriminator encourages a smooth and robust discriminator and helps ease the problem of instability.

3

Quantitative Evaluation on Artificial Datasets

3.1. Artificial Datasets

In this thesis, we create an artificial image dataset to evaluate GANs. Currently available datasets for evaluation are either too simple, for example, a mixture of Gaussians in 1D or 2D space, or too complex, for example, the ImageNet [26].

By introducing an artificial dataset, we limit the evaluation of GANs to a specific dataset, and enable the quantitative evaluation from both the diversity aspect and the quality aspect. The artificial dataset we propose has two properties. First, the variation of images within the dataset is controllable and is easy to detect, and thus we are able to examine GANs' ability to generate diverse samples. Second, the content of images are relatively simple and are easy to compare in image spaces. Therefore, we can evaluate GANs' ability to generate high quality samples by comparing these samples with ground-truth images in a pixel-by-pixel manner.

Our artificial image dataset consists of three datasets of images with one or two spheres in an arbitrary location.

The Sharp-sphere Dataset The simplest dataset is a series of single-channel binary images of one sphere, which only have two possible values 0 and 1 for each pixel, as presented in Figure 3.1(a). The radius of the sphere is fixed, but the center of the sphere is uniformly distributed over all possible locations.

If we let the center of the sphere be $c(c_1, c_2)$, and the pixel value be $I_{x,y}$, then $I_{x,y}$is determined by its distance to the center:

$$I_{x,y} = \begin{cases} 1 & (x-c_1)^2 + (y-c_2)^2 \leq r^2 \\ 0 & (x-c_1)^2 + (y-c_2)^2 > r^2 \end{cases}, \tag{3.1}$$

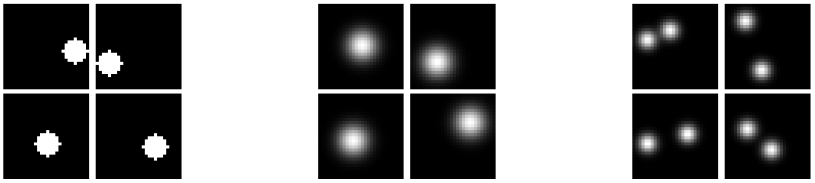
where r is the radius of the sphere.

The Smooth-sphere Dataset By applying the Gaussian blur, we create the smooth-sphere dataset, which is the gray-scale smooth version of images in the sharp-sphere dataset, as presented in figure 3.1(b). If we let the center of the sphere be $c(c_1, c_2)$, and the pixel value be $I_{x,y}$, then $I_{x,y}$ is determined by its distance to the center:

$$I_{x,y} = \exp\left(-\frac{((x-c_1)^2 + (y-c_2)^2)}{2\sigma^2}\right), \tag{3.2}$$

where σ controls the radius size.

The Two-sphere Dataset To increase the complexity of our artificial images, we further create the two-sphere dataset by putting two non-overlapping spheres in one image, as shown in figure 3.1(c).



(a) Four sample images from the sharp-sphere dataset (b) Four sample images from the smooth-sphere dataset (c) Four sample images from the two-sphere dataset

Figure 3.1: Samples from our artificial image dataset.

3.2. Evaluating The Diversity

The variation of images in our artificial dataset is solely represented by the different locations of the centers of spheres. Thus, we can evaluate GAN's ability to generate diverse samples by comparing the distributions of center locations of real samples and generated samples.

Detecting center locations One advantage of our artificial datasets is that we can detect the variations: the center locations of the generated images easily. The main technique used here is the convolution template matching [34].

One-sphere For one-sphere images, let $c = [c_1, c_2]$ be the center of a sphere then the location of the center c of real data follows a uniform distribution q in 2D space. Similarly, the location of the center $\hat{c}$ of generated samples follows an unknown distribution q in 2D space. The location $\hat{c}$ can be easily detected by convolving with a template filter which has the ground-truth shape, and the location with maximum convolution intensity would be the center. This operation is illustrated in figure 3.2.

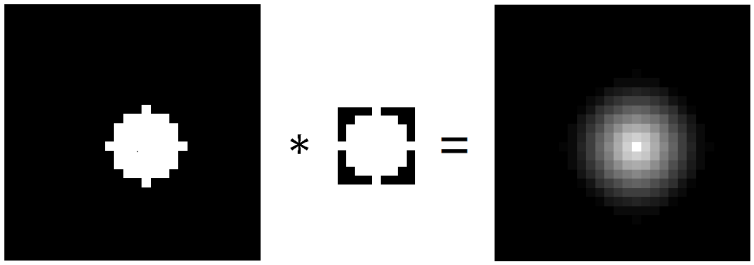


Figure 3.2: An illustration of using convolution operation to detect center location.

Two-sphere For two-sphere images, let $c^{(1)} = [c_1^{(1)}, c_2^{(1)}]$ be the center of one sphere, and $c^{(2)} = [c_1^{(2)}, c_2^{(2)}]$ be the center of the other sphere, then $[c^{(1)}, c^{(2)}]$ of real data follows a fixed distribution q in 4D space. Similarly, $[c^{\hat{(1)}}, c^{\hat{(1)}}]$ of generated samples follows an unknown distribution q in 4D space. To detect $c^{(1)}$ and $c^{(2)}$, we can also apply a convolution operation to the image and find out the two locations with maximum convolution intensity, as shown in figure 3.3. However, for generated images tat may not have perfect shapes, it is highly likely that our algorithm finds multiple detections of the same sphere. We use a technique called non-max suppression to make sure that our algorithm detects each sphere only once. Non-max suppression means that we are going to output a single local maximum value of intensity, but suppress the close-by ones that are non-maximal.

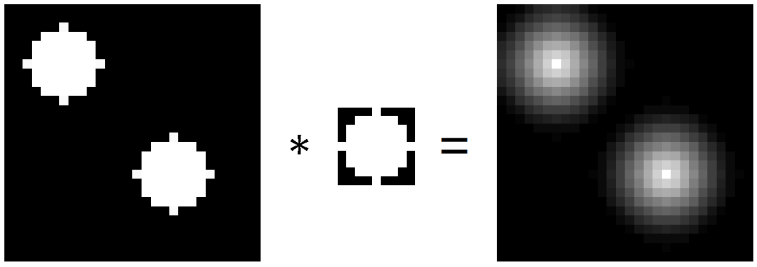


Figure 3.3: An illustration of using convolution operation to detect center locations for two-sphere dataset.

Estimating center location distribution After detecting the center locations, we can estimate the pdf of the distribution of these locations (q) using Parzen window estimates method. We need to decide the number of samples drawing to estimate the distribution.

Approximating the KL Divergence We compare the difference between p and q to measure the diversity of generated samples. One common approach to measure differences between p and q is the KL-divergence. In theory, we could choose

either $D_{KL}(p \parallel q)$ or $D_{KL}(q \parallel p)$. In order to avoid numerical instability caused by computing $\log 0$, we choose to compute $D_{KL}(q \parallel p)$. To compute the KL divergence $D_{KL}(q \parallel p)$, we need to know the pdf of p and of q explicitly. However, the pdf of distribution q, which is the distribution of the locations of the center for generated images, is not available directly. Thus, we use the Parzen window method to obtain an estimate.

In summary, we first draw a certain number of samples from q and estimate its pdf by the Parzen window estimates method, and then we approximate the KL divergence by discretization and summation over finite sets. We need to take care of two parameters here: the number of samples and the size of discretization intervals.

We choose these two parameters empirically. Without loss of generality, we use the smooth-sphere dataset to analyze these two parameters. We compute $D_{KL}(q \parallel p)$, where p and q are both ground-truth distribution of the location of the center, but the pdf of q is estimated by Parzen window estimates from samples and the pdf of p is given explicitly: $p(c) = \frac{1}{m^2}$, where m is the length of feasible area. Figure 3.4 shows the estimated KL divergence against the number of samples, for different discretization interval sizes: 1, 0.5, 0.1,0.05. Ideally, the KL divergence of all these curves should be 0, since p and q are the same distribution. We find that the larger the number of samples, the closer the KL divergence to 0, and that the smaller the discretization interval, the closer the curve to 0.

Although larger sample number and smaller interval size helps to make better estimation, they also increase computation cost. Hence, we take sample size as 10,000 and interval size as 0.1 in our analysis.

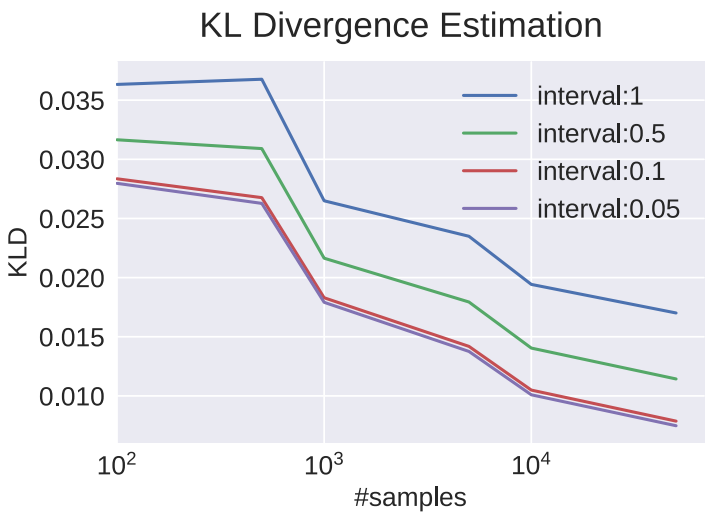


Figure 3.4: Empirical analysis of two parameters: the number of samples, and the size of interval. We find that the larger the number of samples, the closer the KL divergence to 0, and that the smaller the discretization interval, the closer the curve to 0.

3.3. Evaluating The Quality

Since we know the ground-truth shape of these spheres, we are able to assess GANs' ability to generate realistic samples, which is the quality aspect of our evaluation metrics, by comparing the generated samples and the ground-truth images in a pixel-by-pixel manner. Let the pixel coordinate be x, y, the pixel value of a ground-truth image be $I_{x,y}$, and the pixel value of a generated sample be $\hat{I}_{x,y}$, the Root Mean Square Error (RMSE) of these two images is computed as:

$$\text{RMSE} = \sqrt{\frac{\sum_{x=1}^{m}\sum_{y=1}^{n}(I(x,y) - \hat{I}(x,y))^2}{mn}}, \tag{3.3}$$

where m is the height of the image and n is the width of the image.

4

Technique for improving the performance of GANs

In this work, we also discover two training techniques that improve the performance of GANs with the help of our quantitative evaluation scheme. First, adding proper regularization on networks improves and stabilizes the performance. Second, we develop a Smooth-to-Sharp training framework to improve the performance, in which training starts with smooth images and progresses gradually to sharp ones.

4.1. Adding Regularization

In this section, we discuss the benefits of adding regularization terms to GANs. First, we take a close look at the optimal discriminator assumption, and discuss whether an optimal discriminator is needed. After that, we present the necessity for adding a proper regularization term and we exemplify the process by adding L2-regularization on both the discriminator and generator.

Optimal Discriminator Assumption In Chapter 2, we derived that in the original GAN [1], if the discriminator is optimized by training, updating the generator is equivalent to minimizing the following Jensen-Shannon divergence: the optimal discriminator loss $\mathcal{L}_{D^*}$ represents the Jensen-Shannon divergence of real data distribution p_{data} and p_{model}:

$$\mathcal{L}_{D^*} = -2D_{JS}(p_{data} \parallel p_{model}) + 2\log 2. \tag{4.1}$$

This result is based on the assumption that the discriminator D is constantly optimal. But how strong does this optimality assumption hold in practice? One intuitive method is to compare this optimal discriminator loss $\mathcal{L}_{D^*}$ with the real loss $\hat{L_D}$ observed in training.

We first examine the optimality assumption using a 1D toy example, where the real data distribution is a single Gaussian in 1D and GANs are trained to generate samples according to a similar distribution, as shown in Figure 4.1. The red line shows the output of the discriminator. Higher probability is given to real samples and thus the generator is guided by the discriminator to generate samples closer to real samples. Figure 4.2 illustrates the discriminator losses. The green curve is the actual discriminator loss, and the red curve shows the optimal discriminator loss L_{D^*} obtained from estimating JS Divergence. The zigzag pattern of the red curve after some iterations is caused by the fact that the generated samples are bouncing around the real samples. When comparing the optimal discriminator loss L_{D^*} (the red curve) with the actual discriminator loss $\hat{L_D}$ (the green curve), we spot a large gap between the two colored curves, suggesting that the discriminator is not optimal, which does not satisfy the optimal discriminator assumption.

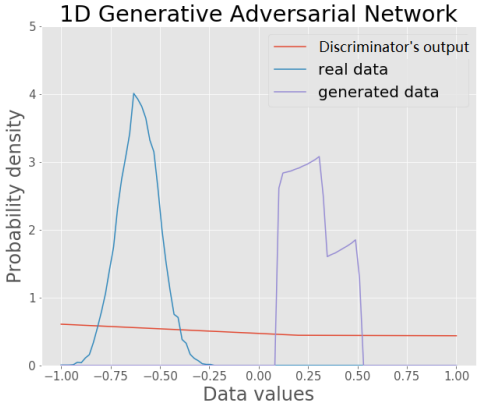


Figure 4.1: a 1D toy example: the real data distribution is a single Gaussian in 1D and GANs are trained to generate samples according to a similar distribution. The red line shows output of the discriminator, which gives higher value to real samples and thus guides the generator to generate samples closer to real samples.

Furthermore, we find another evidence suggesting the optimal assumption not hold. We examine the optimality assumption by training our discriminator using the same 1D toy example with the generator being frozen. Since the generated data and the real data can be completely separated, the optimal discriminator loss is 0. However, the actual discriminator loss curve, as presented in figure 4.3, shows that it takes more than 2000 iterations to decrease L_D to below 0.01. It again provides evidence that the optimal discriminator assumption is not always guaranteed in practice.

Reasons for the optimal discriminator assumption being violated in some circumstances may be the limited computational power of any neural network and the fact that D is not trained until convergence.

Do we need an optimal discriminator? But do we really need an optimal discriminator? An almost optimal discriminator is shown in figure 4.4. We obtained it

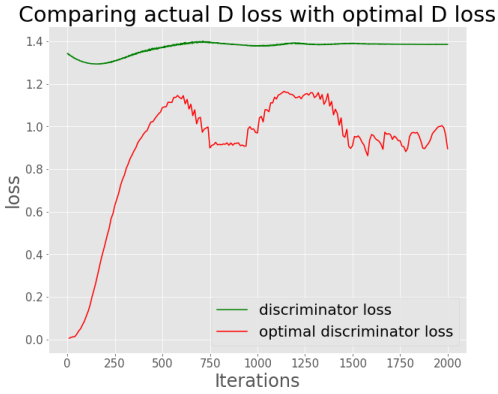


Figure 4.2: Comparison between the actual discriminator loss $\hat{L_D}$ (the green curve) and the optimal discriminator loss L_{D^*} (the red curve). There is a large gap between the two losses. The zigzag pattern of the red curve after some iterations is caused by the fact that the generated samples are bouncing around the real samples.

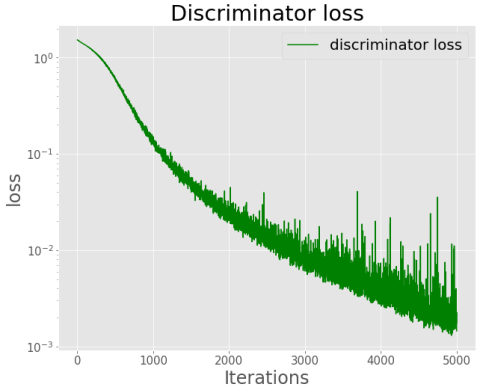


Figure 4.3: The actual discriminator loss curve when the generated data and the real data is separable. The generator is frozen and only the discriminator is updated.

by freezing the generator and updating the discriminator for some iterations. The red curve shows the output of the discriminator. The generator receives updating gradients from the discriminator. This optimal discriminator, however, cannot provide gradients to the discriminator for updating since the slope of the red curve is nearly all zero for generated data). This issue is also known as vanishing gradient that usually causes training failure [31].

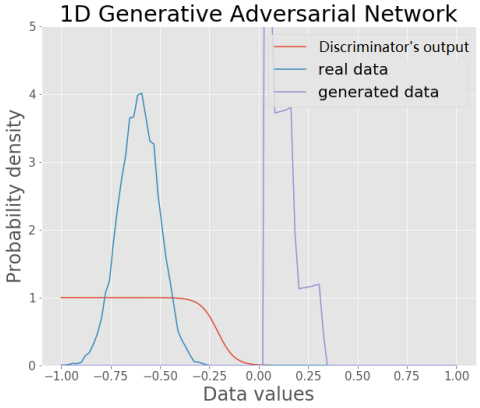


Figure 4.4: This is an almost optimal discriminator example for 1D case. The red curve shows the output of the discriminator.

To conclude, an optimal discriminator is not what we want here, although we do not want a bad discriminator neither. What we want is a discriminator with a smooth discriminant function that could provide proper gradient as learning signals to the generator. Hence, as many researchers claimed, when selecting a discriminator in real practice, we need to balance the discriminator to avoid it being too good, which leads to vanishing gradient problem, or too bad, which leads to failure in discriminating.

L2-Regularization One common method to impose a penalty on the complexity of a model is to add L2-regularization. This strategy adds a regularization term $\Omega(\theta) = \frac{1}{2} \parallel \theta \parallel_2^2$ to the original objective function (2.6)(2.7).

To learn a smooth discriminant function D, we can add a proper L2-regularization term to the objective function of D (2.6). Experimental results also show that both the quality and the diversity are improved if a proper L2-regularization term is added. Additionally, we find that by adding L2-regularization on the generator G, the performance of GAN is improved as well. These experiments are discussed in detail in the next chapter.

4.2. Smooth-to-sharp Training

Smooth or sharp? In our artificial dataset, we create two slightly different sets of images. One is of a sharp sphere, in which values of all pixels inside the sphere

are set to 1 and all other pixels are set to 0. The other is of a smooth sphere with smooth transitions in the boundary where pixels near the center of the sphere have higher values than pixels far from the center.

In our experiments, we find that a GAN obtains much better performance when trained with the smooth images, comparing to that with the sharp ones. Real world images are considered as sharp ones, as they have many sharp edges. Observing that GAN models learn better in the smooth setting, we train a GAN model with smooth images first and progress gradually to sharp ones to exploit if such training algorithm improves the performance of GANs.

Artificial datasets

Normalized Tunable Sigmoid function For our artificial datasets, we can control the smoothness of images by the Normalized Tunable Sigmoid Function:

$$f(x) = \frac{(2x-1) - k(2x-1)}{2(k - 2k|2x-1| + 1)} + 0.5, \tag{4.2}$$

where $k \in [-1, 0]$ is the smoothness parameter that controls the smoothness of an image. An illustration of this function is shown in Figure 4.5. When $k = 0$, it is just an identity function $f(x) = x$ that does not change the input. When k approaches -1, the output approaches 0 when the input is less than 0.5 and approaches 1 when the input is greater than 0.5.

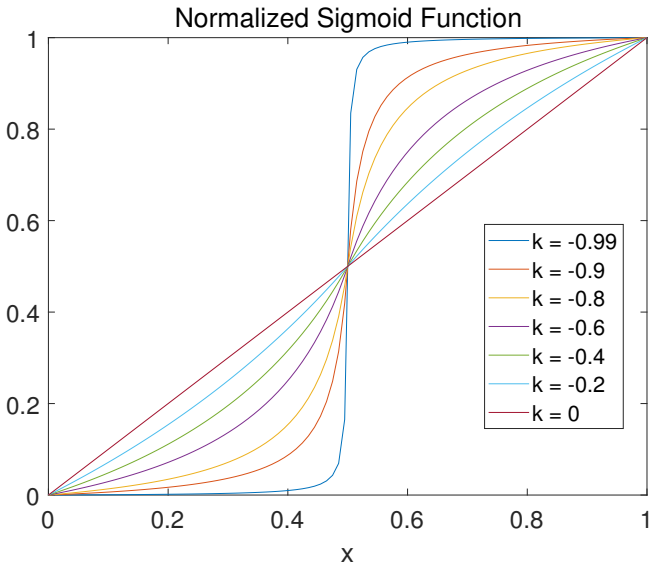


Figure 4.5: Normalized Tunable Sigmoid Function

Visualization We can apply the Normalized Tunable Sigmoid Function (4.2) to the smooth images, and we can obtain images with different smoothness strength. As shown in figure 4.6, the smaller the k, the sharper the image. When k approaches -1, we can obtain sharp images that are almost the same as the binary images in our Sharp-sphere dataset.

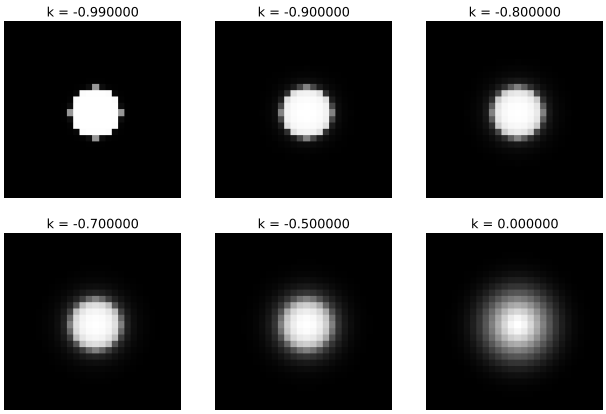


Figure 4.6: circle images with different smoothness strength

Real-world image dataset For real world image dataset, we could smooth the image by Gaussian smoothing technique. Mathematically, we could convolve the image with a Gaussian function in two-dimension. The equation of a Gaussian function $G(x, y)$ is:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{4.3}$$

where x is the distance in horizontal axis and y in vertical axis, and σ is the standard deviation of the Gaussian distribution which controls the smoothness power. Figure 4.7 shows an example of applying Gaussian smoothing to an image with different σ.

Algorithm The general idea of our Smooth-To-Sharp (STS) training algorithm is that we would like to start training with smooth images, and then with sharper ones, until the end we train GANs with our target image dataset: the sharp ones.

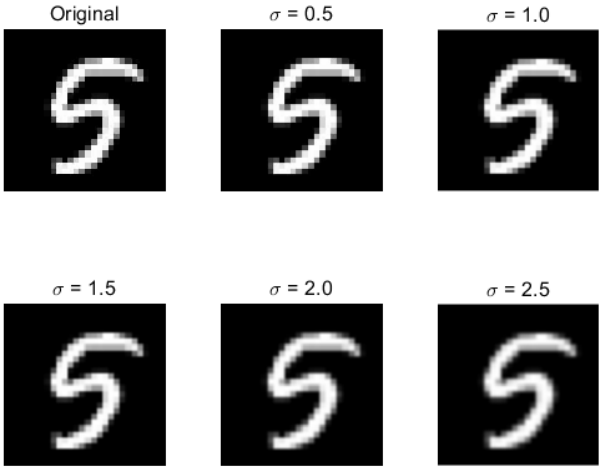


Figure 4.7: images (digit 5) with different smooth strength

5

Experiments

5.1. Exp 1: 1D Examples

We first present an example of a GAN trained on a 1D Gaussian distribution. We present and visualize the training outcomes in different training stages. We also present the quantitative evaluation of the performance of GANs when regularization terms are added during the trainings.

Setups The 1D training samples follow a Gaussian Distribution: $\mathcal{N}(-0.6, 1)$. Table 5.1 and 5.2 show the architecture of the GAN model used and the hyperparameters. A simple GAN model is used here: the discriminator and the generator each has one hidden layer with 16 units and the random inputs is a 1D uniform distribution.

Table 5.1: The structure of GAN used in this experiment. $FC.16 + relu$ = Fully connected layer with 16 units and Relu non-linearity.

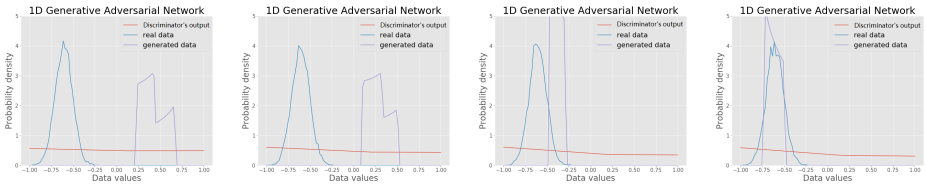
Model	Generator	Discriminator
1d-GAN	$FC.1 \rightarrow FC.16 + relu \rightarrow FC.1$	$FC.1 \rightarrow FC.16 + relu \rightarrow FC.1$

Table 5.2: List of hyperparameters and their values for training 1d-GAN.

Hyperparameter	Value	Description
Dimensions of latent space	1	Number of random inputs to the generator
Learning rate (D)	1×10^{-3}	The learning rate used for optimizing the discriminator.
Learning rate (G)	1×10^{-3}	The learning rate used for optimizing the generator.
L2-Regularization(D)	$0, 1 \times 10^{-5}, 1 \times 10^{-3}, 1 \times 10^{-2}$	The strength of L2 penalty on the discriminator.
L2-Regularization(G)	$0, 1 \times 10^{-5}, 1 \times 10^{-3}, 1 \times 10^{-2}$	The strength of L2 penalty on the generator.
Optimizer	RMSProp	The optimizer used for updating the generator and the discriminator.
Training steps	5,000	Total number of updates during training.
Batch size	64	Number of samples in each mini-batch fed to the network.
Number of repeats	20	Number of times of repeating an experiment.

Results We first visualize training results in different training steps. We use samples from the aforementioned 1D distribution as real data, on which we train GANs to generate data that follow a similar distribution. Figure 5.1 demonstrates the density estimates of the generated data (purple curve) and the real data (blue curve) and visualize the output of the discriminator (red line). The output represents the predicted probability that these samples come from the real data distribution. The output value is around 0.5 in figure 5.1, suggesting that the discriminator is not optimal, as we have discussed in Chapter 4. However, samples from the real distribution (the blue curve) have slightly higher output value of the discriminator than samples from generated distribution (the purple curve). As the generator is updated to generate samples with higher output value of the discriminator, the generated samples would be "guided" to move close to real samples.

Another observation of the result is that as the number of iterations grows from 0 to 1500, the estimated density of the generated data become closer and closer to that of the real data, which illustrates that the actually training progresses in a step-by-step manner, in other words, the performance of GANs improves with the number of iterations.



(a) Iteration 0 (b) Iteration 500 (c) Iteration 1000 (d) Iteration 1500

Figure 5.1: Visualization of training results in different training steps. The real data distribution is a single Gaussian in 1D space (blue curve) and GANs are trained to generate samples (purple curve) according to a similar distribution. The red line shows the output of the discriminator.

We further conduct an experiment to explore the relationship between sample qualities and the L2-Regularization strength λ. We analyze the relationship quantitatively by estimating the KL divergence between the generated data distribution and the real data distribution. The results are shown in figure 5.2. When we add a weak regularization ($\lambda = 1 \times 10^{-5}$) to the network, comparing to when no regularizaiton is added, the KL divergence decreases but its variance does not change obviously. When we slightly increase the strength of the added regularization ($\lambda = 1 \times 10^{-4}$), the value of KL divergence continue to decrease while the variance decreases remarkably. When we further increase the strength of the regularization (or $\lambda = 1 \times 10^{-3}$), although the value of KL divergence still decreases, the variance of it starts to increase. Eventually, when a very strong L2-Regularization ($\lambda = 1 \times 10^{-2}$) is added, both the value and the variance of KL divergence increase considerably, and become even larger than those of the network with no regularization.

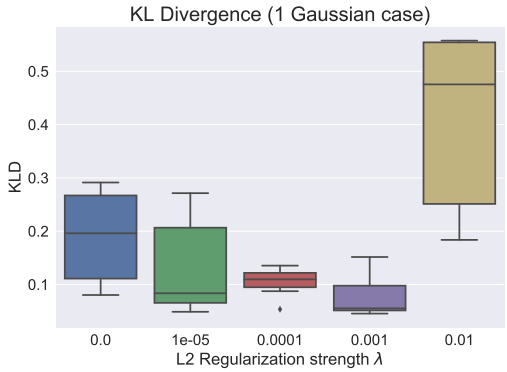


Figure 5.2: Boxplot of the relationship between the L2-Regularization strength and the KL divergence of the real data distribution and the generated distribution.

Discussion

1. Training results from this simple 1D example shows that a GAN is able to learn from training samples and to generate samples that follow the learned density function. The visualization of training results in different training steps in figure 5.1 illustrates that the training progresses in a step-by-step manner, in other words, the performance of GANs improve with the number of iterations.

2. Our experiment on the same 1D example further provides evidence supporting that adding a proper L2-Regularization to GAN models is helpful. As shown in figure 5.2, by adding a proper L2-Regularization (e.g., $\lambda = 1 \times 10^{-4}$), the performance of GAN can be largely boosted, as the value of KL divergence becomes much smaller comparing with the case when no regularization is added. It can also be inferred that the added regularization stabilizes the training, as the variance of the KL divergence drops.

5.2. Exp 2: Evaluation of Adding Regularization

In this section, we present the quantitative evaluation of the performance of GANs on our proposed artificial image datasets, and further present the evaluation of performance when L2-Regularization is added.

Setups The training datasets used in this experiment are the sharp-sphere dataset and the smooth-sphere dataset, as we described in chapter 3. Details of the datasets are described in table 5.3.

Table 5.4 and 5.5 show the architecture of the GAN models used in this experiment and hyperparameters of the models. A simple GAN model is used here: the discriminator and the generator each has one hidden layer with 64 units. The generator receives a uniform random inputs of two dimensions.

Table 5.3: The training datasets used in this experiment are listed below.

Training dataset	Description
Sharp-sphere Dataset	400 binary images samples, image size: $28 * 28$, radius = 2.
Smooth-sphere Dataset	400 smooth images samples, image size: $28 * 28$, radius = 2.

Table 5.4: The structure of GAN used in this experiment. $FC.16 + relu$ = Fully connected layer with 16 units and Relu non-linearity

Model	Generator	Discriminator
One-sphere GAN	$FC.2 \rightarrow FC.64 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.64 + relu \rightarrow FC.1$

Table 5.5: List of hyperparameters of GANs in this experiment and their values for training one-sphere samples.

Hyperparameter	Value	Description
Dimensions of latent space	2	Number of random inputs to the generator
Learning rate (D)	1×10^{-3}	The learning rate used for optimizing the discriminator.
Learning rate (G)	1×10^{-3}	The learning rate used for optimizing the generator.
L2-Regularization(D)	$0 \sim 1 \times 10^{-2}$	The strength of L2 penalty on the discriminator.
L2-Regularization(G)	$0 \sim 1 \times 10^{-2}$	The strength of L2 penalty on the generator.
Optimizer	RMSProp [13]	The optimizer used for updating the generator and the discriminator.
Training steps	10,000	Total number of updates during training.
Batch size	128	Number of samples in each mini-batch fed to the network.

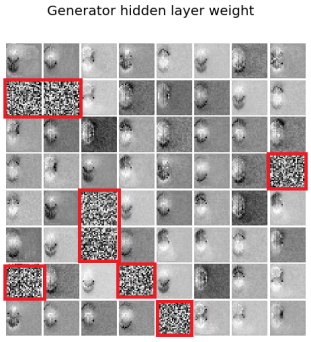
5

Results

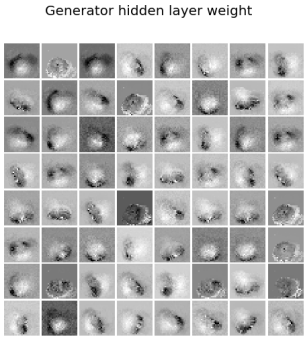
Visualization of learned weights In chapter 4, we discussed the benefits of adding a proper regularization term. For the discriminator, we would like to learn a smooth discriminant function, and thus L2-Regularization is added. The visualization of learned weights of the generator provide a support for adding regularization to the generator as well.

In figure 5.3, we visualize the learned weights of 64 hidden units in the generator under different regularization strength ($\lambda = 0$ or $\lambda = 1 \times 10^{-5}$). When no regularization is added to the generator ($\lambda = 0$), the weights of some units are merely random noise, as noted with red rectangle in figure 5.3(a), which suggests that some of the units are not utilized. When L2-Regularization is added, the weights of all units are not random, indicating that all units are utilized to learn certain patterns, as shown in figure 5.3(b).

Quantitative evaluation We also evaluate the benefits of adding L2-Regularization quantitatively from two aspects: the quality and the diversity. To evaluate the quality of sample generation, we compute the Root Mean Square Error (RMSE) with respect to the ground-truth image. To evaluate the diversity, we estimate the KL divergence between the distribution of locations of the center of the real samples and that of the generated samples. A lower KL divergence indicates higher similarity between two distributions of center locations, and thus better diversity of generate samples, while a higher KL divergence indicates lower similarity in center



(a) G: $\lambda = 0$



(b) G: $\lambda = 1 \times 10^{-5}$

Figure 5.3: Visualization of the learned weights of the generator under different L2-Regularization term. When $\lambda = 0$, the weights of some units are merely random noises. When $\lambda = 1 \times 10^{-5}$ the weights of all units are not random.

locations and poorer sample diversity.

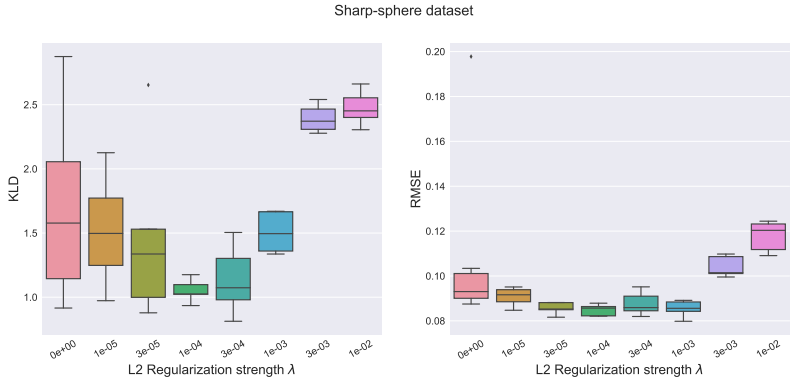


Figure 5.4: Evaluation on the diversity (the KL divergence) and the quality(RMSE) under different L2 Regularization strengths in the Sharp-sphere dataset. The boxplots visualize the results of 20 random repetitions.

Figure 5.4 demonstrates the change in the quality and the diversity of samples generated with different strengths of added regularization in the Sharp-sphere dataset. Comparing to cases with no regularization ($\lambda = 0$), if we add a moderate regularization ($\lambda = 1 \times 10^{-5}$), both the value and the variance of KLD and RMSE decrease, suggesting that the quality and the diversity of generated samples get improved and GANs become more stable. This is consistent with what we discussed in Chapter 4, that adding a proper regularization eases the vanishing gradient problem and thus improves the performance. However, if λ gets too

large (when $\lambda \geq 3 \times 10^{-3}$), the value and the variance of KLD and RMSE would increase back again, suggesting that quality and the diversity of generated samples get worse. This is reasonable since too strong regularization would result in limited modeling power of both the generator and the discriminator and affect the performance negatively.

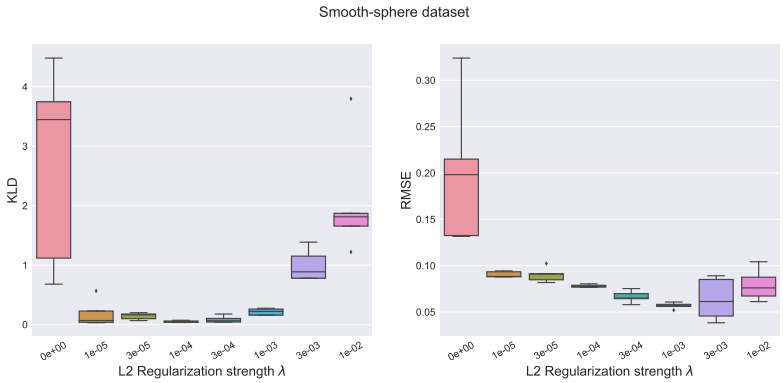


Figure 5.5: Evaluation on the diversity (the KL divergence) and the quality(RMSE) under different L2 Regularization strengths in the Smooth-sphere dataset. The boxplots visualize the results of 20 random repetitions.

For another dataset (Smooth-spheres), there is a similar trend of the change in the quality and the diversity of samples generated with different regularization strengths, as demonstrated in figure 5.5. When a moderate regularization ($\lambda = 1 \times 10^{-5}$) is added, both the quality and the diversity of generated samples get improved and models are more stable comparing to cases with no regularization ($\lambda = 0$). However, if λ gets too large (when $\lambda \geq 3 \times 10^{-3}$), both the quality and the diversity of generated samples get worse.

Table 5.6 presents the mean and standard deviation of the RMSE and the KL divergence of our GANs with different regularization strengths in the Sharp-sphere and the Smooth-sphere datasets, respectively. RMSE and KL divergence with the minimal value and variance are in bold, which represent the observed best performance of our GANs in the current experience. In the Sharp-sphere dataset, GANs achieve optimal performance with a regularization strengths of $\lambda = 1 \times 10^{-4}$ (RMSE: 0.0849 ± 0.0026, KLD: 1.0511 ± 0.0905). In the Smooth-sphere dataset, GANs achieve the optimal RMSE and KL divergence with regularization strengths of $\lambda = 1 \times 10^{-3}$ and $\lambda = 1 \times 10^{-4}$, respectively (RMSE: 0.0568 ± 0.0032, KLD: 0.0521 ± 0.0150). The quantitative comparison shows that by adding a moderate regularization, both the diversity and the quality are largely boosted for these two artificial datasets.

Discussions

1. Our results support the necessity of adding a proper L2-Regularization to improve of the performance of GANs. As shown in figure 5.3, adding regular-

Table 5.6: A summary of evaluation on the quality and the diversity using the Smooth-sphere dataset and the Sharp-sphere dataset, respectively. The mean and standard deviation of 20 repeats are displayed.

	Sharp-sphere Dataset		Smooth-sphere Dataset	
λ	KL Divergence	RMSE	KL Divergence	RMSE
0	1.6936 ± 0.7360	0.1107 ± 0.0430	2.6956 ± 1.6895	0.2002 ± 0.0788
1×10^{-5}	1.5232 ± 0.4890	0.0908 ± 0.0046	0.1865 ± 0.2274	0.0904 ± 0.0032
3×10^{-5}	1.4795 ± 0.7061	0.0856 ± 0.0027	0.1420 ± 0.0558	0.0902 ± 0.0079
1×10^{-4}	**1.0511 ±0.0905**	**0.0849 ±0.0026**	**0.0521 ±0.0150**	0.0780 ± 0.0016
3×10^{-4}	1.1345 ± 0.2722	0.0877 ± 0.0053	0.0857 ± 0.0570	0.0665 ± 0.0065
1×10^{-3}	1.5053 ± 0.1601	0.0854 ± 0.0037	0.2151 ± 0.0547	**0.0568 ±0.0032**
3×10^{-3}	2.3929 ± 0.1096	0.1041 ± 0.0047	0.9973 ± 0.2653	0.0639 ± 0.0228
1×10^{-2}	2.4748 ± 0.1378	0.1178 ± 0.0069	2.0718 ± 0.9975	0.0793 ± 0.0171

ization to the generator helps utilize more neuron units to learn meaningful representations. Moreover, as discussed in Chapter 4, an optimal discriminator cannot provide proper gradients to guide the generator when the real data distribution and the generated distribution have disjoint supports and thus is not what we want. Adding a proper L2-Regularization can ease this problem because the regularization term imposes a penalty on the complexity of a model and helps the discriminator to learn a smooth discriminant function.

2. Our experiments further provide quantitative evidence that adding L2-Regularization on the generator and the discriminator improves both the quality and the diversity of generated samples. As shown in figure 5.4, figure 5.5, and table 5.6, adding a proper L2-Regularization enables GANs to generate more diverse samples with higher quality, regardless of the smoothness of input images. Moreover, by adding a proper L2-Regularization, the variance of the diversity and the quality metrics get decreased dramatically, suggesting that the model gets more stable.

5.3. Exp 3: Evaluation of GANs' Performance on Images with Different Smoothness

Our artificial dataset contains two slightly different sets of images. The Sharp-sphere images have all pixel values inside the sphere set to 1 and all other pixels set to 0. The Smooth–sphere images have smooth transitions in the boundary.

We are interested in the performance of GANs in these two sets of images with different smoothness. Since real world images are considered as sharp images because of the large amount of sharp edges, if GANs learn better in the smooth setting, the algorithm of training a GAN model with smooth images as start and progressing to sharp images gradually will have the potential to improve the performance of the model.

In this experiment, we quantitatively evaluate the performance of original GAN [1] and WGAN[28, 31] in training images with different smoothness factor k values.

Setups

Dataset We evaluate GANs in training images with three different k values: 0, -0.2 and -0.99999. All experiments are repeated for five times. Other settings of the experiment are shown in table 5.7.

The training datasets used in this experiment are the Sharp-sphere dataset and the Smooth-sphere dataset. Details of the datasets are presented in table 5.7. In this experiment, the location of centers of training samples is limited to integer positions, and thus the maximum sample size is 400 for a 28*28 sized image and 1600 for a 48*48 sized image.

Table 5.7: The training datasets used in this experiment are listed below.

Training dataset	Description
Tunable Dataset-28*28	400 tunable images samples, image size: $28 * 28$, radius = 4.
Tunable Dataset-48*48	1600 tunable images samples, image size: $48 * 48$, radius = 4.
Tunable Dataset-two-sphere	67230 two-sphere samples, image size:$28 * 28$, radius = 4.

5

GAN architecture and hyperparameters Table 5.19 and 5.20 shows the architecture of the GAN model used and the hyperparameters. A simple GAN model is used here: both the discriminator and the generator have one hidden layer with 64 units. The generator receives random inputs of two dimensions.

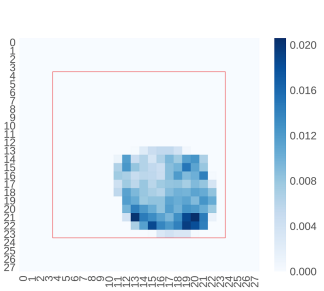
Table 5.8: The structure of GAN used in this experiment. $FC.64 + relu$ = Fully connected layer with 64 units and Relu non-linearity

Model	Generator	Discriminator
GAN (28*28)	$FC.2 \rightarrow FC.64 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.64 + relu \rightarrow FC.1$
WGAN (28*28)	$FC.2 \rightarrow FC.64 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.64 + relu \rightarrow FC.1$
WGAN (48*48)	$FC.2 \rightarrow FC.64 + relu \rightarrow FC.2304 + Sigmoid$	$FC.2304 \rightarrow FC.64 + relu \rightarrow FC.1$
WGAN (two-sphere)	$FC.4 \rightarrow FC.128 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.128 + relu \rightarrow FC.1$

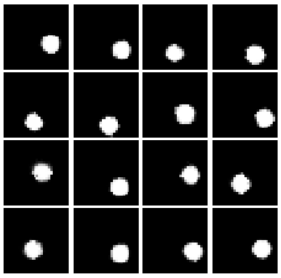
Table 5.9: List of hyperparameters and their values for training one-sphere GAN.

Hyperparameter	Value	Description
Dimensions of latent space (one-sphere)	2	Number of random inputs to the generator (for one-sphere dataset)
Dimensions of latent space (two-sphere)	4	Number of random inputs to the generator (for two-sphere dataset)
Learning rate (D)	1×10^{-3}	The learning rate used for optimizing the discriminator.
Learning rate (G)	1×10^{-3}	The learning rate used for optimizing the generator.
Generator update frequency (WGAN)	5	The frequency of which the generator is updated.
Gradient Penalty (WGAN)	10	
L2-Regularization(D, GAN)	1×10^{-4}	The strength of L2 penalty on the discriminator.
L2-Regularization(G, GAN)	1×10^{-4}	The strength of L2 penalty on the generator.
Optimizer	RMSProp	The optimizer used for updating the generator and the discriminator.
Training steps	20,000	Total number of updates during training.
Batch size	64	Number of samples in each mini-batch fed to the network.

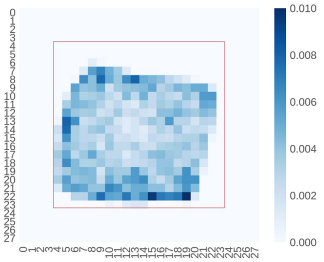
Quantitative evaluation We are interested in evaluating both diversity and quality. For evaluating diversity, KL divergence is used. For evaluating quality, root mean square error (RMSE) is used.



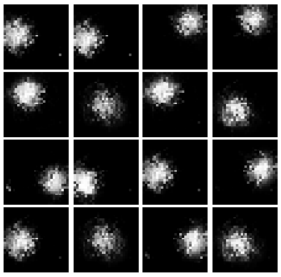
(a) center location of generated sharp spheres visualization



(b) 16 samples of generated sharp images



(c) center location of generated smooth spheres visualization



(d) 16 samples of generated smooth images

Figure 5.6: visualization of center location of generated samples and 16 samples. The red box indicates the feasible locations of centers.

Results

Qualitative evaluation We first compare the generated samples trained with smooth images and with sharp images qualitatively. Figure 5.6 shows the visualization of center location of generated samples and 16 samples. Ideally, the center location would be distributed uniformly over all feasible locations, which would cover all locations expect four borders that are as wide as the radius of spheres. Figure 5.6(a) visualize the center locations for sharp-sphere dataset, and the red box shows the feasible locations. We find that he center locations of generated images only cover the bottom-right part of the red box. However, for smooth-sphere dataset, the center locations of generated images cover a much larger area, as shown in figure 5.6(c), which indicates that GANs can generate more diverse images trained

with smooth images than with sharp ones.

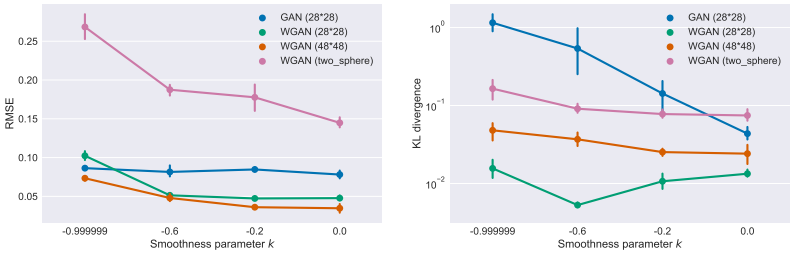


Figure 5.7: An overview of evaluating the quality (RMSE) and the diversity (KLD) for all models under different smoothness parameter k.

5

Quantitative evaluation We perform quantitative evaluation for four different models: the original GAN trained with $28*28$ one-sphere dataset (GAN (28*28)), the WGAN trained with $28*28$ one-sphere dataset (WGAN (28*28)), the WGAN trained with $48*48$ one-sphere dataset, and the WGAN trained with two-sphere dataset (WGAN (two-sphere)). We plot the diversity performance against different smoothness parameter k for these four models and the quality performance in figure 5.7.

1. GAN (28*28): The performance of this dataset is shown as the blue curve in figure 5.7. The KL divergence decreases from more than 1 to less than 0.1 when the smoothness parameter k increases, suggesting that the smoother the training images are, the more diverse the generated samples are. However, the RMSE does not change obviously with the increase of smoothness parameter k, suggesting that the quality of generate samples is less sensitive to the smoothness of the training images.

2. WGAN (28*28): The performance of this dataset is shown as the green curve in figure 5.7. The KL divergence decreases when the smoothness parameter k increases from -0.99999 to 0, though it does not decrease monotonously. One possible explanation is that the WGAN already has the ability to generate diverse images in the very sharp setting (k = -0.99999), so there is not much room for improvement with smoother settings. The differences in green curve may be caused by the four borders of feasible locations. A smooth sphere has larger radius as it has smooth transitions in boundaries, thus the spheres that are close to the four borders do not have complete shapes, which add some difficulties for GANs to learn. That may explain why the performance of diversity gets slightly worse as the images get smoother. The RMSE, however, decreases as k gets larger, suggesting that the smoother the training images are, the better the quality of generated samples is.

3. WGAN (48*48): The performance of this dataset is shown as the orange curve in figure 5.7. Both the KL divergence and RMSE decrease when the

smoothness parameter k increases, suggesting that the smoother the training images are, the more diverse and of the higher quality the training images are. Comparing to WGAN (28*28), the feasible center locations get larger, which leads to a more difficult learning task for GANs. As shown in figure 5.7 on the right, the KL divergence for WGAN (48*48) is indeed higher than the one for WGAN (28*28). Meanwhile, unlike WGAN (28*28), we observe that the KL divergence decreases monotonously as k increases.

4. WGAN (two-sphere): The performance of this dataset is shown as the pink curve in figure 5.7. Both the KL divergence and RMSE decrease when the smoothness parameter k increases, suggesting that the smoother the training images are, the more diverse and of the higher quality the training images are. Obviously, generating two-sphere images is a harder task, and we observe that the values of both RMSE and KL divergence are higher than WGAN (28*28) and WGAN (48*48).

Discussion

1. The experimental results provide a strong evidence suggesting that GANs generate more diverse and higher quality images in the smooth setting than in the sharp setting.

2. Comparing the normal GAN and the WGAN, WGAN has better performances both in diversity and quality, especially for diversity. This is consistent with our discussion in Chapter 2 and the claim in [28, 31] that optimizing the Wasserstein distance is a better objective for GANs than optimizing the original Jensen-Shannon divergence.

3. Comparing WGAN trained with 28*28 image size and WGAN trained with 48*48 image size, WGAN (48*48) has higher KL divergence value than WGAN (28*28), but similar RMSE results. This is reasonable since larger image size leads to larger feasible center locations, and generating diverse spheres with larger feasible center locations is generally a harder task for WGAN.

5.4. Exp 4: Smooth-to-sharp Training Method

Observing the fact that a GAN model performs better in smooth image settings, in this experiment, we are encouraged to explore whether the performance of GANs can be improved by training a GAN model with smooth images first and progress gradually to sharp ones.

Setups

Dataset The training datasets used in this experiment are one-sphere dataset and two-sphere dataset. The smoothness of an image in one dataset is tunable and is controlled by the smoothness factor k. Although the smoothness is controllable,

the training goal is to generate sharp images in the end. Details of the datasets are described in table 5.10.

Table 5.10: The training datasets used in this experiment are listed below.

Training dataset	Description
Tunable Dataset-28*28-half	200 tunable images samples, image size: $28 * 28$, radius = 4.
Tunable Dataset-28*28-full	400 tunable images samples, image size: $28 * 28$, radius = 4.
Tunable Dataset-48*48-half	800 tunable images samples, image size: $48 * 48$, radius = 4.
Tunable Dataset-48*48-full	1600 tunable images samples, image size: $48 * 48$, radius = 4.
Tunable Dataset-two-sphere	67230 two-sphere samples, image size:$28 * 28$, radius = 4.

GAN architecture and hyperparameters Table 5.19 and 5.20 shows the architecture of the GAN model used and the hyperparameters.

Table 5.11: The structure of GAN used in this experiment. $FC.64 + relu$ = Fully connected layer with 64 units and Relu non-linearity

Model	Generator	Discriminator
GAN (28*28)	$FC.2 \rightarrow FC.64 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.64 + relu \rightarrow FC.1$
WGAN (28*28)	$FC.2 \rightarrow FC.64 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.64 + relu \rightarrow FC.1$
WGAN (48*48)	$FC.2 \rightarrow FC.64 + relu \rightarrow FC.2304 + Sigmoid$	$FC.2304 \rightarrow FC.64 + relu \rightarrow FC.1$
WGAN (two-sphere)	$FC.4 \rightarrow FC.128 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.128 + relu \rightarrow FC.1$

Table 5.12: List of hyperparameters and their values for training one-sphere GAN.

Hyperparameter	Value	Description
Dimensions of latent space (one-sphere)	2	Number of random inputs to the generator (for one-sphere dataset)
Dimensions of latent space (two-sphere)	4	Number of random inputs to the generator (for two-sphere dataset)
Learning rate (D)	1×10^{-3}	The learning rate used for optimizing the discriminator.
Learning rate (G)	1×10^{-3}	The learning rate used for optimizing the generator.
Generator update frequency (WGAN)	5	The frequency of which the generator is updated.
Gradient Penalty (WGAN)	10	
L2-Regularization(D, GAN)	1×10^{-4}	The strength of L2 penalty on the discriminator.
L2-Regularization(G, GAN)	1×10^{-4}	The strength of L2 penalty on the generator.
Optimizer	RMSProp	The optimizer used for updating the generator and the discriminator.
Training steps	20,000	Total number of updates during training.
Batch size	64	Number of samples in each mini-batch fed to the network.

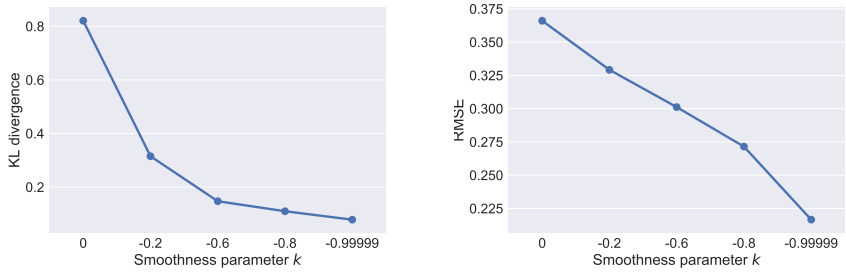
Smooth-to-sharp algorithm The key idea of our proposed smooth-to-sharp algorithm is to start training with smooth images, and progresses to sharp ones. In this experiment, we exemplify this algorithm by the settings shown in table 5.13. The total number of training steps is 20000, and we allocate these steps for different smoothness parameter k: 0, -0.2, -0.6, -0.8 and -0.99999.

Results We exemplify the progress of Smooth-to-Sharp training by the WGAN with Two-sphere dataset in figure 5.8. After the GAN model finishes training with

Table 5.13: Smooth-to-sharp training algorithm: the training starts with the very smooth images, then progresses to sharp ones. The total number of updates is 20000.

smoothness parameter k	0	-0.2	-0.6	-0.8	-0.99999
number of steps	2500	2500	2500	2500	10000

a certain smoothness images, we conduct the quantitative evaluation and record the KL divergence and the RMSE. In figure 5.8(a), we visualize the KL divergence against different smoothness factor. We find that the KL divergence goes down as the training samples progress from smooth to sharp. In figure 5.8(b), we also find that the RMSE goes down as the training samples progress from smooth to sharp.



(a) WGAN (two-sphere): KL divergence

(b) WGAN (two-sphere): RMSE

Figure 5.8: Plots showing the progress of Smooth-to-Sharp training of WGAN (two-sphere). We conduct the quantitative evaluation and record the KL divergence and the RMSE after the GAN model finishes training with a certain smoothness images.

Furthermore, we conduct a quantitative comparison of the original direct training method and our proposed Smooth-to-Sharp training method. We compare the diversity (KL divergence) and the quality (RMSE) of generated samples for four different GAN models. The result is shown in table 5.14. We find that for all listed GAN models, our Smooth-to-Sharp method obtains lower KL divergence and RMSE for all listed GAN models (shown with bold font in table 5.14), suggesting that under the Smooth-to-Sharp method, a GAN generates more diverse and higher quality images.

Discussion

1. The comparison results provide a strong evidence suggesting that the performance of GANs can be improved by training progressively: training a GAN model with smooth images first and progress gradually to sharp ones. The direct training method and our Smooth-to-Sharp method are compared fairly with all other settings the same. As shown in table 5.14, our Smooth-to-Sharp method outperform the original direct training method notably with respect to

Table 5.14: Experiment results for GAN models, either trained in the direct way or our proposed "Smooth-To-Sharp (STS)" method. Each experiment is repeated five times and we record the average KL divergence and RMSE.

Model	Method	KL Divergence	RMSE
GAN (28*28)	direct	1.1525	0.0860
	STS	**0.8648**	**0.0819**
WGAN (28*28)	direct	0.0425	0.1070
	STS	**0.0117**	**0.0845**
WGAN (48*48)	direct	0.0480	0.0733
	STS	**0.0260**	**0.0682**
WGAN (two-sphere)	direct	0.1644	0.2684
	STS	**0.0679**	**0.2062**

both the KL divergence and RMSE. Moreover, the progressive training method especially boosts the performance of diversity, as the improvement of KL divergence is more significant than RMSE.

2. Our proposed Smooth-to-Sharp method is an example of the progressively training, which is easy to implement, suitable for different GAN models and has little additional computational cost. It can be considered as a form of data augmentation as the GAN is trained with images with different smoothness.

5.5. Exp 5: Experiments on the MNIST Dataset

In this section, we will apply the two techniques to the MNIST dataset [35] that contains hand-written digits: adding L2-Regularization and the smooth-to-sharp method. These two techniques have been shown to improve GANs' performance on our proposed artificial dataset. We use the MNIST dataset to validate these two techniques on real-world datasets.

Setups

Dataset The training dataset used in this experiment is the MNIST dataset. The MNIST dataset contains 60,000 examples of hand-written digits from 0 to 9. The size of one image is 28 * 28.

The architecture and hyperparameters of GAN Table 5.15 and 5.16 shows the architecture of the GAN model used and its hyperparameters.

Evaluation of adding L2-Regularization Since the proposed quantitative evaluation of quality and diversity can be only applied to our artificial dataset of spheres, we evaluate GANs' performance on the MNIST dataset in a qualitative manner. The GAN is trained with different L2-Regularization strength λ: 0, 1×10^{-5},

Table 5.15: The structure of GAN used in this experiment. $FC.512 + relu$ = Fully connected layer with 512 units and Relu non-linearity

Model	Generator	Discriminator
GAN	$FC.48 \rightarrow FC.512 + relu \rightarrow FC.512 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.128 + relu \rightarrow FC.128 + relu \rightarrow FC.1$
WGAN	$FC.48 \rightarrow FC.512 + relu \rightarrow FC.512 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.128 + relu \rightarrow FC.128 + relu \rightarrow FC.1$

Table 5.16: List of hyperparameters and their values for training GANs in this experiment.

Hyperparameter	Value	Description
Dimensions of latent space	48	Number of random inputs to the generator (for one-sphere dataset)
Learning rate (D)	1×10^{-3}	The learning rate used for optimizing the discriminator.
Learning rate (G)	1×10^{-3}	The learning rate used for optimizing the generator.
Generator update frequency (WGAN)	5	The frequency of which the generator is updated.
Gradient Penalty (WGAN)	10	
L2-Regularization(D, GAN)	1×10^{-4}	The strength of L2 penalty on the discriminator.
L2-Regularization(G, GAN)	1×10^{-4}	The strength of L2 penalty on the generator.
Optimizer	RMSProp	The optimizer used for updating the generator and the discriminator.
Training steps	20,000	Total number of updates during training.
Batch size	64	Number of samples in each mini-batch fed to the network.

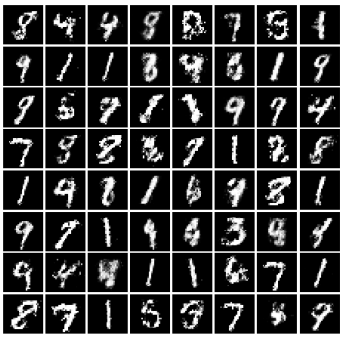
1×10^{-4} and 1×10^{-3}. For every λ, we repeat the training for 20 times and report the results.

Evaluation of Smooth-to-Sharp training Here we apply the Smooth-to-Sharp training method to the MNIST dataset. We smoothen the images by Gaussian smoothing technique. The smoothness parameter here is the standard deviation σ of the Gaussian distribution. In this experiment, we exemplify the Smooth-to-Sharp method using the settings shown in table 5.17. We train a GAN model with smoothened images ($\sigma = 2$) first, and then progress gradually to less smoothened images ($\sigma = 1.5, 1, 0.5$). Finally, we train the model with the original sharp images ($\sigma = 0$).

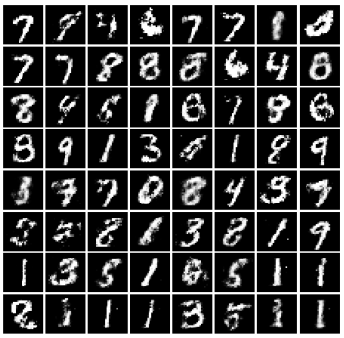
Table 5.17: Smooth-to-sharp training algorithm: the training starts with very smooth images, then progresses to sharp ones.

smoothness parameter σ	2	1.5	1	0.5	0
number of steps	2500	2500	2500	2500	10000

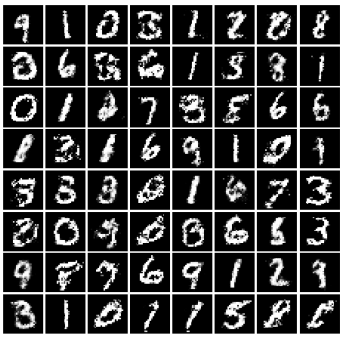
Results



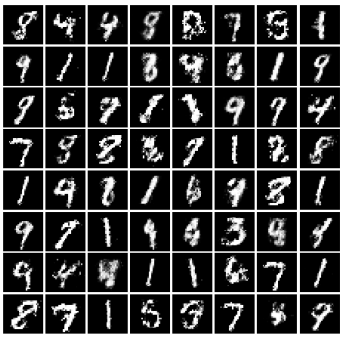
(a) $\lambda = 0$



(b) $\lambda = 1 \times 10^{-5}$



(c) $\lambda = 1 \times 10^{-4}$



(d) $\lambda = 1 \times 10^{-3}$

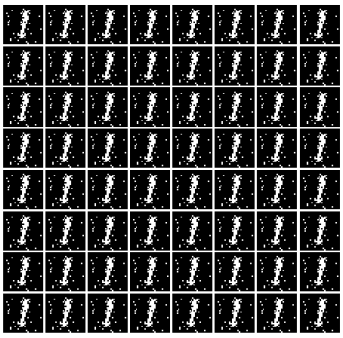
Figure 5.9: Visualization of samples from successfully trained cases.

Adding L2-Regularization To evaluate the benefit of adding L2-Regularization, we repeat the training of a GAN model for 20 times. We find that not all training results contain recognizable digits. We roughly categorize these GAN models into successfully trained and unsuccessfully trained. Some of the trained GAN models can generate relatively "good" images, and we categorize them as successful trained cases. In figure 5.9 we visualize 64 generated samples from successfully trained GANs with different L2-Regularizations. Some of the trained GAN models, however, cannot generate meaningful images,and we categorize them as unsuccessful trained cases. We also visualize generated samples from failed training cases with different L2-Regularizations in figure 5.10.

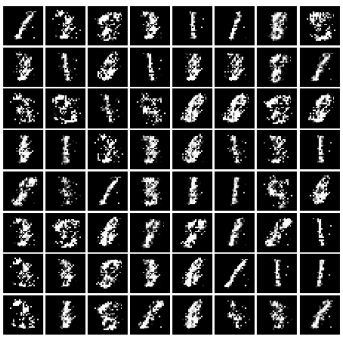
We compare the generated samples by visual inspection. In figure 5.9, we demonstrate collections of 64 samples from GANs with differen L2-Regularizations.

Though this comparison is subjective, we find that among these four different collections, images generated by GAN with $\lambda = 1 \times 10^{-4}$ (figure 5.9(c)) have the best image quality and diversity, as they contain different digits and most digits are clearer and sharper than other collections of images.

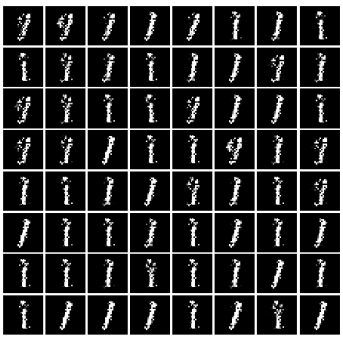
(a) $\lambda = 0$



(b) $\lambda = 1 \times 10^{-5}$



(c) $\lambda = 1 \times 10^{-4}$



(d) $\lambda = 1 \times 10^{-3}$

Figure 5.10: Visualization of samples from unsuccessfully trained cases. These images are of low visual quality.

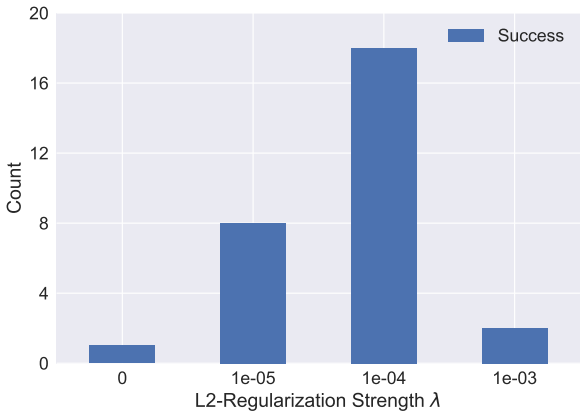


Figure 5.11: The relationship between λ and the number of successfully trained cases in 20 repeats. When no regularization is added ($\lambda = 0$), only 1 out of 20 trainings succeed. When a suitable regularization is added ($\lambda = 1 \times 10^{-4}$), 18 out of 20 training cases succeed. However, if the regularization strength is too strong ($\lambda = 1 \times 10^{-3}$), only 2 out of 20 cases succeed.

Furthermore, we analyze the relationship between λ and the number of successfully trained cases, as shown in figure 5.11. If we do not add a regularization term, only 1 out of 20 trained GANs can generate "good" images, categorized as successfully trained. If we add a suitable regularization term ($\lambda = 1 \times 10^{-4}$), the successful training rate is raised to 18 out of 20. However, if the regularization strength is too strong ($\lambda = 1 \times 10^{-3}$), only 2 out of 20 cases succeed.

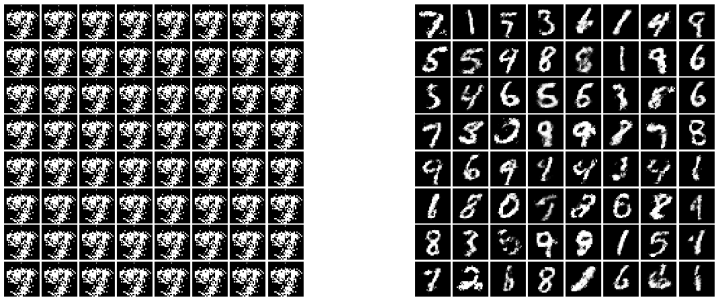
Smooth-to-sharp training Our Smooth-to-Sharp training method stabilizes the training of normal GANs, as all 20 repeats are successful, for all λs from 0 to 1×10^{-3}. We visualize and compare the generated samples from direct method and from our Smooth-to-Sharp method, as shown in figure 5.12, 5.13, and 5.14. Although a visual inspection is objective, we find that images generated by GANs with Smooth-to-Sharp training method are generally of high quality and high diversity.

Discussions

Adding L2-Regularization

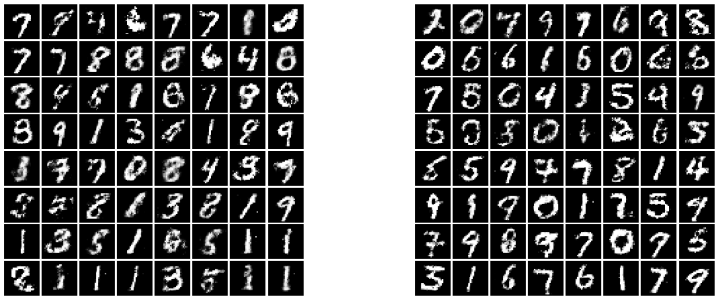
1. We find that adding L2-Regularizations with certain strength significantly increase the proportion of successful trainings, suggesting that adding a proper L2-Regularization stabilize the training process. If no regularization is added, the majority of trainings fail and generate meaningless samples.

2. It is difficult to compare the quality and the diversity of generated images objectively by simply inspecting the samples because they do not have notable differences.



(a) $\lambda = 0$, direct method (b) $\lambda = 0$, smooth-to-sharp method

Figure 5.12: Visualization of samples from direct training method and from the Smooth-to-Sharp method. $\lambda = 0$.



(a) $\lambda = 1 \times 10^{-5}$, direct method (b) $\lambda = 1 \times 10^{-5}$, smooth-to-sharp method

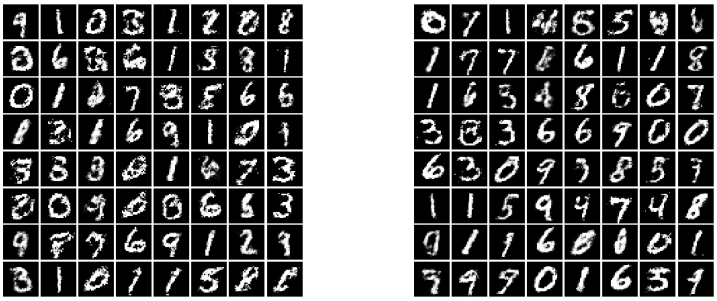
Figure 5.13: Visualization of samples from direct training method and and from the Smooth-to-Sharp method. $\lambda = 1 \times 10^{-5}$.

Smooth-to-Sharp training

1. We find that all trainings taking the Smooth-to-Sharp algorithm are successful, suggesting the capability of this algorithm to stabilize the training process. Compared with direct method, the number of successfully trained cases is raised from $1/20$ to $20/20$ with the Smooth-Sharp algorithm alone.

2. It is difficult to compare the quality and the diversity of generated images objectively by simply inspecting the samples because they do not have notable differences.

Previous experiments are all performed on artificial datasets. Our quantita-



(a) $\lambda = 1 \times 10^{-4}$, direct method (b) $\lambda = 1 \times 10^{-4}$, smooth-to-sharp method

Figure 5.14: Visualization of samples from direct training method and from the Smooth-to-Sharp method. $\lambda = 1 \times 10^{-4}$.

tive evaluation on artificial datasets demonstrates that both adding a proper L2-Regularization and using the Smooth-to-Sharp training could boost the performance of GANs. This experiment on the MNIST dataset validates the efficacy of these two methods on real-world dataset.

5.6. Exp 6: Wasserstein Distance, Diversity and Quality

To explore alternative evaluation metrics, we investigate the relationship between the estimated Wassterstein distance and GAN's performance. The estimated Wasserstein distance is obtained from the discriminator's output, and GAN's performance is represented by both the diversity metrics and the quality metrics proposed in Chapter 3.

Setups We analyze this relationship using our proposed artificial dataset because both the diversity and the quality metrics can only be evaluated quantitatively on our artificial datasets. Three different datasets are used in this experiment: the One-sphere dataset with 28*28 images, the One-sphere dataset with 48*48 images and the Two-sphere dataset with 28*28 images. The details of these datasets are shown in table 5.18.

Table 5.18: The training datasets used in this experiment are listed below.

Training dataset	Description
One-sphere Dataset-28*28	200 images samples, image size: $28 * 28$, radius = 4.
One-sphere Dataset-48*48	1600 images samples, image size: $48 * 48$, radius = 4.
Two-sphere Dataset	67230 two-sphere samples, image size:$28 * 28$, radius = 4.

Table 5.19 and 5.20 show the architecture of the GAN model used in this experiment and its hyperparameters.

Table 5.19: The structure of GAN used in this experiment. $FC.64 + relu$ = Fully connected layer with 64 units and Relu non-linearity

Model	Generator	Discriminator
WGAN (28*28)	$FC.2 \rightarrow FC.64 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.64 + relu \rightarrow FC.1$
WGAN (48*48)	$FC.2 \rightarrow FC.64 + relu \rightarrow FC.2304 + Sigmoid$	$FC.2304 \rightarrow FC.64 + relu \rightarrow FC.1$
WGAN (two-sphere)	$FC.4 \rightarrow FC.128 + relu \rightarrow FC.784 + Sigmoid$	$FC.784 \rightarrow FC.128 + relu \rightarrow FC.1$

Table 5.20: List of hyperparameters and their values for training one-sphere GAN.

Hyperparameter	Value	Description
Dimensions of latent space (one-sphere)	2	Number of random inputs to the generator (for one-sphere dataset)
Dimensions of latent space (two-sphere)	4	Number of random inputs to the generator (for two-sphere dataset)
Learning rate (D)	1×10^{-3}	The learning rate used for optimizing the discriminator.
Learning rate (G)	1×10^{-3}	The learning rate used for optimizing the generator.
Generator update frequency (WGAN)	5	The frequency of which the generator is updated.
Gradient Penalty (WGAN)	10	
Optimizer	RMSProp	The optimizer used for updating the generator and the discriminator.
Training steps	20,000	Total number of updates during training.
Batch size	64	Number of samples in each mini-batch fed to the network.

In this experiment, we record the estimated Wasserstein distance, the estimated KL-divergence, and the estimated RMSE. All training procedures are repeated five times and for each training procedure we record those three values every 1000 iterations (20000 iterations in total), which means that there are 100 records for every dataset.

Results

Wassterstein distance and KL divergence We apply logarithm transformation to the KL divergence in order to give a better linear relationship between the Wassterstein distance and the KL divergence. To reveal the relationship, we fit two lines per dataset for smooth images and sharp images, respectively. The scatter plots and fitted lines are shown in figure 5.15. For each fitted line, we report the r^2 and $p - value$ in table 5.21. The r^2 value indicates the percentage of Wasserstein distance variation that is explained by a fitted line, and a low $p - value$ suggests significant linear relationship between these two variables.

Wassterstein distance and RMSE To reveal the relationship between Wasserstein distance and the RMSE of generated samples compared to ground-truth images, we fit two lines per dataset for smooth images and sharp images, respectively. The scatter plots and fitted lines are shown in figure 5.16. For each fitted line, we report the r^2 and *p-value* in table 5.22.

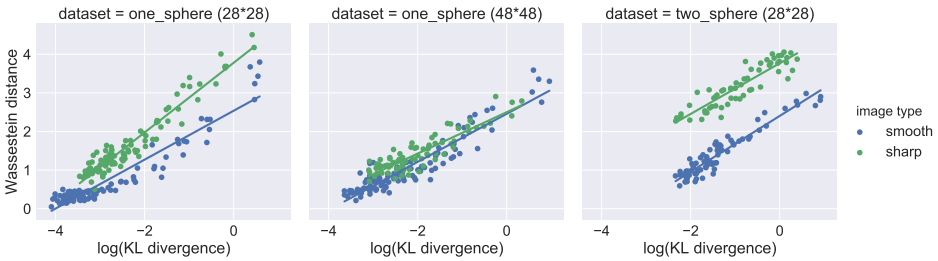


Figure 5.15: The scatter plots and the fitted lines for the Wasserstain distance and the logarithm of KL divergence. Green dots represent sharp images and blue dots represent smooth images in each dataset. The fitted lines indicate strong linear relationship between these two variables.

Table 5.21: The r^2 and $p - value$ of fitted linear models: $\log$(KL divergence) $\sim$ Wasserstein distance.

dataset	image type	r^2	p-value
One-sphere (28*28)	sharp	0.9216	4.0×10^{-58}
	smooth	0.9025	5.4×10^{-51}
One-sphere (48*48)	sharp	0.8104	3.6×10^{-29}
	smooth	0.8649	2.0×10^{-45}
Two-sphere (28*28)	sharp	0.8281	1.2×10^{-23}
	smooth	0.9216	4.3×10^{-43}

Discussion

1. We find that the estimated Wasserstein distance (obtained from the discriminator's loss) is strongly correlated with GANs' performance, with respect to the quality and the diversity evaluation metrics on our artificial dataset, as shown in figure 5.16 and 5.15. Our results provide a quantitative support for using the Wasserstein distance as a metric to evaluate GANs' performance. When first promoted [28], the Wasserstein distance has been suggested to be potentially correlated with the visual quality of generated images. To our knowledge, our experiment is the first to demonstrate this correlation quantitatively. Moreover, we further find that the Wasserstein distance is highly correlated not only with the quality but also the diversity of generated sam-

Table 5.22: The r^2 and *p-value* of fitted linear models: RMSE $\sim$ Wasserstein distance.

dataset	image type	r^2	p-value
One-sphere (28*28)	sharp	0.9216	6.7×10^{-54}
	smooth	0.9409	2.8×10^{-61}
One-sphere (48*48)	sharp	0.8104	3.5×10^{-29}
	smooth	0.9216	2.1×10^{-54}
Two-sphere (28*28)	sharp	0.8104	1.4×10^{-24}
	smooth	0.8836	1.3×10^{-38}

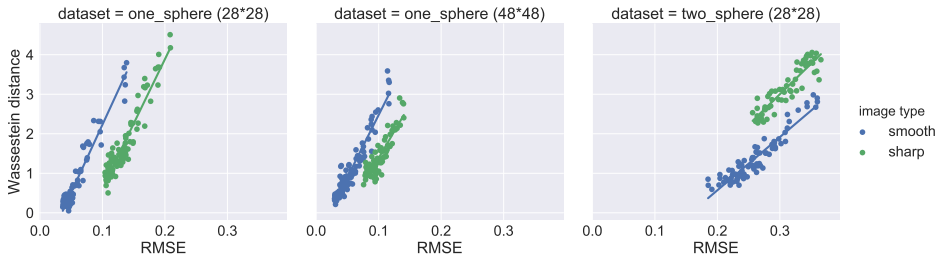


Figure 5.16: The scatter plots and the fitted lines for Wasserstain distance and the RMSE. Green dots represent sharp images and blue dots represent smooth images. The fitted lines indicate strong linear relationship between these two variables.

ples.

2. The linear relationship between the Wasserstein distance and the performance of GANs is only consistent in the same dataset and under the same GAN's architecture. Our results exhibit that the linear relationships are different for different datasets and for images with different smoothness. Hence, a solid comparison of GANs' performance can only be conducted using Wasserstein distance when the dataset and the architecture of GANs are identical.

6

Summary and Conclusions

6.1. Summary

This thesis work developed a systematic evaluation method to quantitatively assess the performance of GANs on a novel artificial dataset and proposed two techniques to improve the training of GANs with the help of our evaluation scheme.

We proposed to evaluate GANs from two aspects: the quality, which evaluates the visual quality of generated images, and the diversity, which evaluates GANs' ability to generate samples with different modes in the real data distribution.

We invented a series of artificial datasets (figure 3.1), consisting of images with one or two spheres in arbitrary locations, to evaluate the performance of GANs. Two important and straightforward properties about the images in our artificial datasets enable quantitative evaluations from both the quality aspect and the diversity aspect, which are the known variations (i.e., all possible locations of the centers of the spheres are predefined) and the regular shapes (i.e., all images are regularly spherical in shape).

We further explored the validity of an alternative evaluation metric, the Wasserstein distance, as an indicator of the quality and the diversity for Wasserstein GANs. The Wasserstein distance was estimated by the discriminator and could be obtained directly from the discriminator's output, making it a cost-effective indicator when considering application in real-world datasets. We compared, using our artificial datasets, the relationship between the Wasserstein distance and our proposed quantitative evaluation metrics, and found that the Wasserstein distance was highly correlated with both the quality and the diversity of generated samples (figure 5.15,5.16). To our knowledge, our study was the first to demonstrate this correlation in a quantitative manner.

Furthermore, We explored two improving techniques that boosted the performance of GANs, namely the addition of regularization terms and the "Smooth-to-Shape" training algorithm. We provided quantitative support that adding a proper L2-Regularization on both the generator and the discriminator stabilized the training

of GANs and improved the quality and the diversity of generated samples (figure 5.4, 5.5). We developed a progressive training method named as the "Smooth-to-Sharp" training algorithm, which trains a GAN model with images of smooth spheres first and progresses gradually to those of sharp spheres. We demonstrated that the "Smooth-to-Sharp" algorithm boosted GANs' capabilities to generate samples with high quality and high diversity, and that the improvement was more significant in the diversity aspect (table 5.14).

We further validated the efficacy of these two improving techniques on the MNIST dataset. We presented that adding L2-Regularizations with certain strength significantly increased the proportion of successful trainings, supporting that adding a proper L2-Regularization stabilized the training process in a real-world dataset. We also observed that all trainings taking the Smooth-to-Sharp algorithm were successful, validating the capability of the Smooth-to-Shape algorithm to improve GANs' performance in a real-world setting.

6.2. Strengths and Limitations

Our work has several strengths.

1. We developed a systematic evaluation method to quantitatively assess the performance of GANs, which was independent of a pre-trained classifier and was achieved by evaluating on a novel artificial dataset.
2. We were the first to quantitatively demonstrate the strong correlation between the Wasserstein distance and both the quality and the diversity of generated samples, through our proposed evaluation method and on our artificial datasets, providing evidence to support the use of the Wasserstein distance to indicate the quality and the diversity of Wasserstein GANs.
3. We proposed two techniques to improve the training of GANs and validated them both on our artificial datasets and on the MNIST dataset. The two techniques proposed were both simple to implement and required little additional computation power.

Our work also has limitations.

1. Our quantitative evaluation method can be only applied using the proposed artificial datasets, because ground truth about the variation and the quality of samples were required by the evaluation metrics. We would make our artificial datasets open-source to enable the application of our evaluation methods by other researchers.
2. Our artificial datasets only contained images of one or two spheres, which were relatively simple comparing with real-world datasets. We planned to increase the complexity of our datasets in future work.
3. We only validated the two improving techniques we proposed on the artificial datasets and the MNIST dataset using one GAN model due to limited time. Validation of these techniques on different datasets and using various GANs were planned as future work to increase reliability.

6.3. Future work

Firstly, we plan to increase the complexity of our artificial datasets. The currently proposed datasets are of one sphere or two non-overlapping and otherwise independent spheres. Approaches to improve the complexity of our datasets include increasing the type of sample shapes, such as introducing into the datasets images of other regular shapes (e.g., ovals and rectangles), creating dependency between the two spheres on each image, such as letting the distance between the two sphere centers follow a Normal distribution, and introducing more complex background (e.g., background with different colors and with random noises).

Secondly, we plan to further validate the two improving techniques proposed in this work on more complicated real-world datasets, such as the CIFAR-10 dataset [36], the ImageNet dataset [26] and so on.

Thirdly, systematical review, assessment, and comparison of the performance of all typical GAN models using our quantitative evaluation method could help researchers select better models for different research purposes and promote the improvement of GANs' performances. With increased number of GANs being promoted, such as LS-GAN [37], MMD-GAN [38], Fisher-GAN [39] that were published during the composition of this report, such quantitative evaluation is highly warranted.

Lastly, we plan to further investigate the validity of using the estimated Wasserstein distance as an alternative evaluation metric. We demonstrated that the estimated Wasserstein distance was strongly correlated with both the quality and the diversity of generated samples using our artificial dataset. The estimated Wasserstein distance could be obtained directly from the discriminator's output without additional cost, making it a promising indicator when considering application in real-world dataset. We can conduct experiments on real-world datasets to explore the relationship between Wasserstein distance and the quality and the diversity of generated samples.

References

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

[2] A. Radford, L. Metz, and S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks,* arXiv preprint arXiv:1511.06434 (2015).

[3] E. L. Denton, S. Chintala, R. Fergus, *et al.*, *Deep generative image models using a laplacian pyramid of adversarial networks,* in *Advances in neural information processing systems* (2015) pp. 1486–1494.

[4] T. Karras, T. Aila, S. Laine, and J. Lehtinen, *Progressive growing of gans for improved quality, stability, and variation,* arXiv preprint arXiv:1710.10196 (2017).

[5] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, *Infogan: Interpretable representation learning by information maximizing generative adversarial nets,* in *Advances in Neural Information Processing Systems* (2016) pp. 2172–2180.

[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, *Unpaired image-to-image translation using cycle-consistent adversarial networks,* arXiv preprint arXiv:1703.10593 (2017).

[7] Y. LeCun, C. Cortes, and C. Burges, *Mnist handwritten digit database,* AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist **2** (2010).

[8] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, *Improved techniques for training gans,* in *Advances in Neural Information Processing Systems* (2016) pp. 2234–2242.

[9] I. Goodfellow, *Nips 2016 tutorial: Generative adversarial networks,* arXiv preprint arXiv:1701.00160 (2016).

[10] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning,* nature **521**, 436 (2015).

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, *Backpropagation applied to handwritten zip code recognition,* Neural computation **1**, 541 (1989).

[12] L. Bottou, *Large-scale machine learning with stochastic gradient descent,* in *Proceedings of COMPSTAT'2010* (Springer, 2010) pp. 177–186.

[13] T. Tieleman and G. Hinton, *Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,* COURSERA: Neural networks for machine learning **4**, 26 (2012).

[14] D. Kingma and J. Ba, *Adam: A method for stochastic optimization,* arXiv preprint arXiv:1412.6980 (2014).

[15] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators,* Neural networks **2**, 359 (1989).

[16] E. Parzen, *On estimation of a probability density function and mode,* The annals of mathematical statistics **33**, 1065 (1962).

[17] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, *A learning algorithm for boltzmann machines,* Cognitive science **9**, 147 (1985).

[18] L. Theis, A. v. d. Oord, and M. Bethge, *A note on the evaluation of generative models,* arXiv preprint arXiv:1511.01844 (2015).

[19] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, Vol. 1 (MIT press Cambridge, 2016).

[20] M. Arjovsky and L. Bottou, *Towards principled methods for training generative adversarial networks,* arXiv preprint arXiv:1701.04862 (2017).

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

[22] M. Mirza and S. Osindero, *Conditional generative adversarial nets,* arXiv preprint arXiv:1411.1784 (2014).

[23] O. Breuleux, Y. Bengio, and P. Vincent, *Quickly generating representative samples from an rbm-derived process,* Neural computation **23**, 2058 (2011).

[24] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, *Least squares generative adversarial networks,* in *2017 IEEE International Conference on Computer Vision (ICCV)* (IEEE, 2017) pp. 2813–2821.

[25] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, *Gans trained by a two time-scale update rule converge to a local nash equilibrium,* in *Advances in Neural Information Processing Systems* (2017) pp. 6629–6640.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database,* in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (IEEE, 2009) pp. 248–255.

[27] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, *Are gans created equal? a large-scale study,* arXiv preprint arXiv:1711.10337 (2017).

[28] M. Arjovsky, S. Chintala, and L. Bottou, *Wasserstein gan,* arXiv preprint arXiv:1701.07875 (2017).

[29] Y. Rubner, C. Tomasi, and L. J. Guibas, *A metric for distributions with applications to image databases,* in *Computer Vision, 1998. Sixth International Conference on* (IEEE, 1998) pp. 59–66.

[30] L. V. Kantorovich and G. S. Rubinstein, *On a space of completely additive functions,* Vestnik Leningrad. Univ **13**, 52 (1958).

[31] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, *Improved training of wasserstein gans,* in *Advances in Neural Information Processing Systems* (2017) pp. 5769–5779.

[32] A. Srivastava, L. Valkov, C. Russell, M. Gutmann, and C. Sutton, *Veegan: Reducing mode collapse in gans using implicit variational learning,* arXiv preprint arXiv:1705.07761 (2017).

[33] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann, *Stabilizing training of generative adversarial networks through regularization,* in *Advances in Neural Information Processing Systems* (2017) pp. 2015–2025.

[34] R. Brunelli, *Template matching techniques in computer vision: theory and practice* (John Wiley & Sons, 2009).

[35] Y. LeCun, *The mnist database of handwritten digits,* http://yann. lecun. com/exdb/mnist/ (1998).

[36] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images,* (2009).

[37] G.-J. Qi, *Loss-sensitive generative adversarial networks on lipschitz densities,* arXiv preprint arXiv:1701.06264 (2017).

[38] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, *Mmd gan: Towards deeper understanding of moment matching network,* in *Advances in Neural Information Processing Systems* (2017) pp. 2200–2210.

[39] Y. Mroueh and T. Sercu, *Fisher gan,* in *Advances in Neural Information Processing Systems* (2017) pp. 2510–2520.