IMPROVING IMPLICIT GENERATIVE MODELS VIA ONLINE KERNEL METHODS

A Dissertation
Presented to
The Academic Faculty

By

Yeojoon Youn

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

Aug 2023

IMPROVING IMPLICIT GENERATIVE MODELS VIA ONLINE KERNEL METHODS

Thesis committee:

Dr. Jacob Abernethy
School of Computer Science
Georgia Institute of Technology

Dr. Vidya Muthukumar
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Justin Romberg
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mark Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date approved: July 28, 2023

ACKNOWLEDGMENTS

I am profoundly indebted to my advisor, Dr. Jacob Abernethy, for his invaluable mentorship and unwavering support throughout my research. Engaging in discussions with Jake has been both enlightening and enjoyable. I wish to extend my gratitude to Dr. Vidya Muthukumar, my co-advisor, and to Dr. Justin Romberg and Dr. Mark Davenport, who graciously accepted the task of reviewing my MS thesis. My heartfelt thanks go to Neil Thistlethwaite and my longtime friend, Sang Keun Choe, for their significant contributions to this work. Lastly, but most importantly, I express my deepest appreciation to my parents and two sisters.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

GANs Generative Adversarial Networks

GMMN Generative Moment Matching Network

IKL Implicit Kernel Learning

MMD Maximum Mean Discrepancy

OKGANs Online Kernel-based Generative Adversarial Networks

PCA Principal Component Analysis

RKHS Reproducing Kernel Hilbert Space

SUMMARY

The objective of the proposed research is to solve unusual training behaviors of Generative Adversarial Networks (GANs), including mode collapse and cycling, by using online kernel-based non-parametric classifiers. One of the major breakthroughs in machine learning over the past several years has been GANs, a neural network-based implicit generative model which aims to mimic some underlying distribution given a dataset of samples. In contrast to many supervised problems, where one tries to minimize a simple objective function of the parameters, GAN training is formulated as a min-max problem over a pair of network parameters. GANs have shown impressive empirical success in some domains in the following sense: samples produced by the trained model do appear (subjectively) to fall within the support of the target distribution. But in prior work, researchers have been puzzled by aforementioned unusual training behavior, including cycling and so-called *mode collapse*, and almost all prior work has avoided careful analysis of whether the estimated distribution does actually match the target in a quantitatively measurable sense. Thus, in this work, we propose (a) new metrics to evaluate the performance of implicit generative models and (b) a new framework for training such a model using kernel-based non-parametric classifiers instead of the typical neural network discriminators. We call this the Online Kernel-based Generative Adversarial Networks (OKGANs), and we emphasize that it can be trained efficiently. We show that using classical results from online learning with kernels, OKGANs are guaranteed to match the target distribution under certain oracle guarantees on the generator family. Empirically, our models do a much better job of minimizing a reverse-KL-divergence between the trained model and the target distribution, and as expected from the theory, OKGAN does not exhibit mode collapse or cycling and performs dramatically better than many of the classical GAN formulations on low-dimensional synthetic data. We also compare performance on classical vision datasets such as MNIST, SVHN, CelebA, and CIFAR-10, and show comparable performance.

CHAPTER 1

INTRODUCTION

Generative Adversarial Networks (GANs) [1] frame the task of estimating a generative model as solving a particular two-player zero-sum game. One player, the generator G, seeks to produce samples which are indistinguishable from those generated from some true distribution p_{real}, and the other player, the discriminator D, aims to actually distinguish between such samples. In the classical setting each of these players is given by neural networks parameterized by θ_g and θ_d, respectively. For a random input seed vector z sampled from a base distribution p_z, the generator outputs a synthetic example $x = G_{\theta_g}(z)$, and the discriminator returns a probability (or score) according to whether the sample is genuine or not. In its original formulation, a GAN is trained by solving a min-max problem, $\min_{\theta_g} \max_{\theta_d} V(\theta_d, \theta_g)$:=

$$\mathbb{E}_{x\sim p_{\text{real}}}[\log D_{\theta_d}(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D_{\theta_d}(G_{\theta_g}(z)))] \tag{1.1}$$

From a theoretical perspective, this framework for learning generative models has two very appealing qualities. First, in a setting where V is convex in θ_g and concave in θ_d, a true equilibrium point of this game could be readily obtained using various descent-style methods or regret-minimization [2]. Second, it was observed in [1] that, given an infinite amount of training data, and optimizing over the space of all possible discriminators and all possible generative models, the equilibrium solution of (1.1) would indeed return a generative model that captures the true distribution p_{real}. [3, 4] also claim that their GANs can learn the true distribution based on this strong assumption.

The challenge, in practice, is that none of these assumptions hold, at least for the way that the most popular GANs are implemented. The standard protocol for GAN training is to find an equilibrium of (1.1) by alternately updating θ_d and θ_g via stochastic gradient descent/ascent using samples drawn from both the true distribution (dataset) and generated samples. It has been observed that simultaneous descent/ascent procedures can fail to find equilibria even in convex settings [5, 6, 7], as the equilibria are "unstable" when traversal through the parameter landscape is viewed as a dynamical system; one might even expect to see cycling around the min-max point without progress [8].

[9] raise another issue involving the capacity of the discriminator network. In the thought experiment from [1] we summarized above, the discriminator must have "enough capacity" to produce a suitably complex function. In practice, however, GANs are trained with discriminators from a parametric family of neural networks with a fixed number of parameters k. [9] observe that if the generator is allowed to produce distributions from mixtures of $O\left(\frac{k}{\epsilon^2}\right)$ base measures then the generator can fool the discriminator (up to ϵ) with the appropriate choice of mixture that differs substantially from p_{real}. Authors of this work have suggested that this issue of finite capacity explains GAN *mode collapse*, a phenomenon observed by practitioners whereby a GAN generator, when trained to estimate multi-modal distributions, ends up dropping several modes.

We have found a useful way to visualize the dance performed between D and G. While it is hard to observe cycling behavior in the high-dimensional parameter space of θ_d, we can take a natural projection of the discriminator as follows. Let $\theta_d^{(1)}, \theta_d^{(1)}, \ldots, \theta_d^{(T)}$ be the sequence of parameters produced in training the GAN via the standard sequential update steps on the objective (1.1), and let $x_1, \ldots, x_m$ be a random sample of examples from the training set. Consider the matrix $M = [D(x_j; \theta_d^{(i)})]_{i \in [T], j \in [m]}$, where each row represents the function value of the discriminator on a set of key points at a fixed training time i. We can perform a Principal Component Analysis (PCA) of this data, with two components, and obtain a projected version of the data $\tilde{M} \in \mathbb{R}^{T \times 2}$, so each row is a 2-dimensional projection of the discriminator at a given time. We display this data in Figure 1.1 (Left graph), where light pink nodes are earlier training time points and dark red are later rounds of training. The

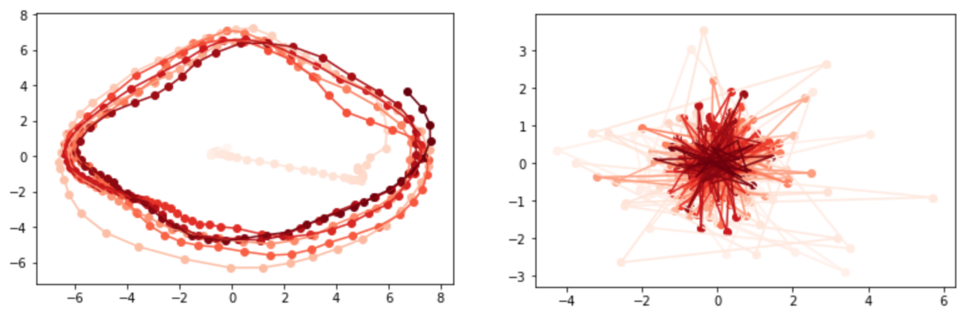


Figure 1.1: Qualitative Comparison of Vanilla GAN (Left) and OKGAN (Right, proposed) on Cycling Behavior on 2D-Grid Synthetic Dataset.

observation here is quite stark: the discriminator does indeed cycle in a relatively consistent way, and suggests that training may not always be highly productive.

In this research, we propose a GAN formulation that relies on a *non-parametric* family of discriminators, kernel-based classifiers, which can be efficiently trained online; we call this OKGAN (the Online Kernel-based Generative Adversarial Network). OKGANs exhibit several benefits over previous GAN formulations, both theoretical and empirical. On the theoretical side, the non-parametric nature of the family of functions helps avoid the two negative results we highlighted above: the limited-capacity challenge raised by [9] and the numerical training issues described by [5]. Given that the kernel classifiers can grow in complexity with additional data, the discriminator is able to adapt to the increasing complexity of the generator and the underlying distribution to be learned[1]. Furthermore, the discriminator formulation is now a convex problem, and we have a wealth of results on both the computational as well as statistical properties of kernel-based predictors. Kernel classifiers also allow additional flexibility on data representation through the choice of kernel, or combination of kernels. Moreover, Implicit Kernel Learning (IKL) [10] allows us to get proper kernels based on each dataset via implicit generative models parameterized by neural networks.

What is quite clear, empirically, is that OKGANs *do not suffer from mode collapse*,

[1]Our kernel classifiers do require a limited budget size for the number of examples to store, but this is mostly for computational reasons, and the budget size can be scaled up as needed

at least not on any of the low-dimensional synthetic examples that we have tested. On simple problems where the target exhibits a density function, we show OKGAN not only achieves high sample quality but also outperforms prior methods when it comes to capturing the target distribution via reverse KL-divergence. Additionally, when we include the use of an "encoder," we qualitatively demonstrate OKGANs work well on classical image datasets, including MNIST, SVHN, CelebA, and CIFAR-10. We observe the discriminator of OKGANs adapts in a more aggressive fashion, and the discriminator does not appear to exhibit cycling behavior, a common phenomenon for other GANs using neural net based discriminators.

1.1 Related Works

Many theoretical works have aimed at understanding GANs. [9, 11] study the generalization properties of GANs under neural distance, [12] study the convergence properties of GANs via "adversarial" divergence. [13] assert that the diversity in GANs can be improved by letting the discriminator class get strong distinguishing power against a certain generator class. [14, 5, 6, 15, 16, 17] consider a range of questions around GAN dynamics. Additionally, there are interesting works on analyzing the GAN dynamics such as [14, 5, 6, 15, 16, 17].

Mode collapse and cycling behavior are two of the main issues raised about GAN training. [18] observes the cycling behavior of the generator's output in an experiment with a 2D synthetic dataset when mode collapse occurs. [19] propose a new visualization technique called path-angle to study the game vector field of GANs and shows cycling behavior empirically with this technique. [7] try to improve GAN training via so-called "optimistic" mirror descent. [20] propose a mathematical definition of mode collapse and gives an information-theoretic analysis. Bourgain Theorem mentioned in [21] use metric embeddings to construct a latent Gaussian mixture, a direct approach to solve mode collapse. Moreover, Prescribed GANs [22] mitigate mode collapse by adding an entropy-regularized term. Additionally, there are several other interesting works about solving mode collapse [23, 24, 25, 26, 27].

Our work is not the first to implement kernel learning ideas into GANs. A statistical hypothesis testing framework called Maximum Mean Discrepancy (MMD) introduced by [28], which aims to compare two different distributions, provides researchers an intuition to apply the kernel method to GANs. [29, 30] propose Generative Moment Matching Network (GMMN) with a fixed Gaussian kernel for the MMD statistical testing. [31] introduce MMD GANs which improve GMMN by adding an injective function to the kernel and making the kernel trainable. [32] demonstrate the superiority of MMD GANs in terms of gradient bias, and [33] improve MMD GANs with a repulsive loss function and a bounded Gaussian kernel. Additionally, instead of using MMD, [34] integrate noise to smooth the Jensen-Shannon Divergence and use kernel density estimates for the true and the generator densities. However, in this thesis, we apply online kernel learning to the discriminator to get an approximately optimal discriminator in every discriminator update, which finally enables the generator to match the true distribution.

CHAPTER 2

ONLINE KERNEL GANS

2.1 Online Kernel Classifier

In the classical formulation of a GAN, the discriminator can generally be regarded as a classifier that aims to distinguish between data in the training set, sampled from p_{real}, and so-called "fake" data produced by the generator. The original GAN formulation of [1], and nearly all other generative models [35, 36, 37, 38, 39, 40, 41] inspired by this work, the discriminator is a finitely-parameterized neural network with parameters $\theta_d \in \mathbb{R}^k$. What we argue in this thesis is that one can instead use a non-parametric family of discriminator functions, a Reproducing Kernel Hilbert Space (RKHS) [42], and obtain both an efficient algorithm as well as a provable guarantee that the learned distribution will converge to the target under reasonably-mild assumptions.

Let us begin by recalling the formulation of the *Wasserstein GAN* [43]. Given two distributions p, q an some input metric space $\mathcal{X}$, we can define any number of distance measures between p and q, but let us focus on one formulation of the Wasserstein distance[1]

$$\Delta(p,q) := \sup_{\text{1-lip } f} \mathbb{E}_{(Y,X)\sim p\vee q}[Y f(X)],$$

where the supremum is taken over all 1-lipschitz functions f, and where the distribution $p \vee q$ on X, Y is taken as follows: Y is sampled uniformly at random from $\{-1, 1\}$, and if $Y = 1$ then $X \sim p$ and otherwise $X \sim q$.

Let us consider modifying this notion of distance in two ways: first, we can incorporate a hinge loss $\ell(z) := \max(0, 1 - z)$, and second, we can take the supremum over a restricted

[1]This isn't precisely the definition of Wasserstein distance, but is equivalent up to a factor of 2 via Kantorovich-Rubinstein duality [44]

class of functions $\mathcal{H}$. We may now define the distance

$$\Delta_{\mathcal{H},\ell}(p,q) := \sup_{f\in\mathcal{H}} \mathbb{E}_{(Y,X)\sim p\vee q}[1-\ell(Yf(X))]. \tag{2.1}$$

Claim 2.1.1. Let $q_1, q_2, \ldots$ be a sequence of distributions on $\mathcal{X}$, and let $\mathcal{H}$ be a family of functions that are dense in the space of all 1-lipschitz functions on $\mathcal{X}$. If $\Delta_{\mathcal{H},\ell}(p, q_t) \to 0$ as $t \to \infty$ then $q_t \to p$.

We omit the proof of this claim as it is straightforward. Restricting to a dense subset of functions does not affect the supremum, and since $1 - \ell(z) = z$ for all $z \in [-1, 1]$ function doesn't have a significant effect since it coincides with the identity function in the interval $[-1, 1]$.

But this observation highlights one of the key points of this thesis. Whereas [43] argued in favor of the Wasserstein GAN by suggesting that it is effectively minimizing $\Delta_{\mathcal{H}}(p, \cdot)$, where $\mathcal{H}$ is a parameterized family of neural networks. But this choice of $\mathcal{H}$ is *not* dense in the space of lipschitz functions. Given an appropriate choice of kernel, such as the Gaussian/RBF, the associated RKHS *is* dense in this space [45]. Moreover, we have efficient algorithms to estimate functions in an RKHS as the problem can be posed in a convex fashion. In this work, we utilize online training methods for kernel classifiers, which ensures that they exhibit vanishing regret, a fact that ensures our convergence result under an assumption on the generator class.

2.1.1 An Overview of Kernel Learning Methods

We now review the basics of kernel-based learning algorithms, and online learning with kernels; see [28, 46, 47, 48, 49] for further exposition. Let $\mathcal{X}$ be some abstract space of data, which typically, although not necessarily, is some finite-dimensional real vector space. A *kernel* $k : \mathcal{X} \times \mathcal{X}$ is called *positive semi-definite* if it is a symmetric function on pairs of examples from $\mathcal{X}$, and for every positive integer d and every set of examples $x_1, \ldots, x_d \in \mathcal{X}$

the matrix $[k(x_i, x_j)]_{i\in[d],j\in[d]}$ is positive semi-definite. Typically we view a PSD kernel as a dot product in some high-dimensional space, and indeed a classic result is that for any PSD k there is an associated feature map Φ which maps points in $\mathcal{X}$ to some (possibly infinite dimensional) Hilbert space for which $k(x, x') = \langle\Phi(x), \Phi(x')\rangle$ [49]. Given a kernel k, we can consider functions of the form $f(x) := \sum_{i=1}^{m} \alpha_i k(x_i, x)$, where the α_i's are arbitrary real coefficients and the x_i's are arbitrary points in $\mathcal{X}$. The set of functions of this form can be viewed as a *pre Hilbert space*, using the norm $\|f\|_{\mathcal{H}} := \sqrt{\sum_i \sum_j \alpha_i\alpha_j k(x_i, x_j)}$, and when we complete this set of functions we obtain the Reproducing Kernel Hilbert Space $\mathcal{H}$. Again, this is a very brief survey, but more can be found in the excellent book of [49].

Let us give an overview of learning in an RKHS associated to some kernel k. First, imagine we have a sequence of m examples $S_m := \{(x_1, y_1), \cdots, (x_m, y_m)\}$ sampled from some distribution p on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y} = \{-1, 1\}$. Our goal is to estimate a classifier $f : \mathcal{X} \to \mathbb{R}$ in $\mathcal{H}$. Assume we the convex loss $\ell : \mathbb{R} \to \mathbb{R}$, where $\ell(yf(x))$ is the cost of predicting $f(x)$ when the true label is y; typically we will use the *hinge loss* or the *logistic loss*. In a batch setting, we may estimate f given S_m by minimizing the regularized risk functional $R_{\text{reg},\lambda}$ defined as follows:

$$R_{\text{reg},\lambda}[f, S_m] := \tfrac{1}{m} \textstyle\sum_{i=1}^{m} \ell(y_i f(x_i)) + \tfrac{\lambda}{2}\|f\|_{\mathcal{H}}^2. \tag{2.2}$$

Assuming that the loss function satisfies a simple monotonicity property, as a result of the celebrated *representer theorem* [49] we may conclude that a solution to the above problem always exists in the linear span of the set $\{k(x_i, \cdot) : i = 1, \ldots, m\}$. In other words, estimating a function in an infinite dimensional space reduces to find m coefficients $\alpha_1, \ldots, \alpha_m$ which parameterize the resulting solution $\hat{f}_m := \sum_{i=1}^{m} \alpha_i k(x_i, \cdot)$.

2.1.2 Online Training

Researchers have known for some time that training kernel-based learning algorithms can be prohibitively expensive when the dataset size m is large; the problem is worse when the dataset is growing in size. Solving (2.2) naively can lead to computational cost that is at least cubic in m. A more scalable training procedure involves online updates to a current function estimate. A more thorough description of online kernel learning can be found in [46] and [48], but we give a rough outline here. Let f_t be the function reached at round t of an iterative process and ρ_t be the offset of f_t at round t.

$$f_t(x) = \rho_t + \textstyle\sum_{i=1}^{t-1} \alpha_i k(x_i, x). \tag{2.3}$$

A simple gradient update with step size η_t, using the instantaneous regularized risk $R_{\text{reg},\lambda}$ with respect to a single example (x_t, y_t), leads to the following iterative procedure:

$$f_{t+1} = f_t - \eta_t \partial_f R_{\text{reg},\lambda}[f, (x_t, y_t)]|_{f=f_t} \tag{2.4}$$

$$= (1 - \eta_t \lambda) f_t - \eta_t y_t \ell'(y_t f_t(x_t)) k(x_t, \cdot) \tag{2.5}$$

Here, $\ell'(yz) := \partial_z \ell(yz)$. The algorithm at time t maintains a set of points $x_1, \ldots, x_{t-1}$ and corresponding corresponding coefficients $\alpha_1, \ldots, \alpha_{t-1}$ and offset ρ_t, and when a new example x_t arrives, due to (2.5), the coefficient α_t is created, and the other α_is are scaled down:

$$\alpha_t := -\eta \ell'(y_t f_t(x_t)), \text{ for } i = t \tag{2.6}$$

$$\alpha_i \leftarrow (1 - \eta\lambda)\alpha_i, \text{ for } i \leq t \tag{2.7}$$

In our implementation, we add n multiple examples at once as a minibatch at every round. For example, at round t, the input is not a single example (x_t, y_t) but n examples $(x_{n(t-1)+1}, y_{n(t-1)+1}), \cdots, (x_{nt}, y_{nt})$. Thus, we change (2.6) and (2.7) as below. Also, ρ_t is updated as an average of n new coefficients.

$$\alpha_i := -\eta y_t \ell'(y_t f_t(x_i)), \text{ for } n(t-1) < i \leq nt$$

$$\alpha_i \leftarrow (1-\eta\lambda)\alpha_i, \text{ for } i \leq n(t-1) \qquad \rho_t := \tfrac{1}{n}\textstyle\sum_{j=n(t-1)+1}^{nt} \alpha_j$$

One may note that the above algorithm scales quadratically with t, since after a given number of t rounds must compute $k(x_i, x_t)$ for all $i < t$. But this can be alleviated with a careful budgeting policy, where only a limited cache of x_i's and α_i are stored. This is natural for a number of reasons, but especially given that α's decay exponentially and thus each α will fall below ϵ after only $\log(1/\epsilon)/(\eta\lambda)$ updates. The issue of budgeting and its relation to performance and computational issues was thoroughly explored by [48], and we refer the reader to their excellent work. In our experiments we relied on the "Remove-Oldest" method akin to first-in-first-out (FIFO) caching. Then, at round t, let's say a fixed budget size of our online kernel classifier is B, and $w_{nt-B+1}, w_{nt-B+2}, \cdots, w_{nt}$ are key examples saved in the budget. As the result, after we finish training on the minibatch at round t, we will get a classifier function f as following.

$$f_t(x) = \rho_t + \textstyle\sum_{i=nt-B+1}^{nt} \alpha_i k(w_i, x) \tag{2.8}$$

2.2 Objective Function and Training

In OKGAN, the discriminator in the original GAN formulation (1.1) is regarded as the online kernel classifier, and it is obtained not from a parametric family of neural networks but from RKHS $\mathcal{H}$. The goal of the online kernel classifier is to separate the real data and the fake data. When we obtain the classifier function after each batch, its value of real data and fake data is respectively positive and negative. That's why we use a hinge loss while formulating a min-max objective of OKGAN. If the generator G is parameterized by θ_g and makes the fake distribution p_G, the objective of OKGAN is

$$\max_{\theta_g} \min_{f \in \mathcal{H}} \mathop{\mathbb{E}}_{(Y,X) \sim p_{\text{real}} \vee p_G} [\ell(Y f(X))] \tag{2.9}$$

We recall from the previous section that $p_{\text{real}} \vee p_G$ is the distribution which first samples a uniformly random Y from $\{-1, 1\}$ and then samples $X \sim p_{\text{real}}$ if $Y = 1$ or $X \sim p_G$ if $Y = -1$. Notice that this objective is exactly $1 - \Delta_{\mathcal{H},\ell}(p_{\text{real}}, p_G)$ as defined in (2.1).

We obtain the online kernel classifier f through the process in 2.1 after training one batch of the dataset. Then, we use the objective function of the generator G as:

$$\min_{\theta_g} \left\{ V(\theta_g) = \mathbb{E}_{z \sim p_z}[\max(0, 1 - f(G_{\theta_g}(z)))] \right\} \tag{2.10}$$

We use (2.10) as the loss function for the generator rather than $-\mathbb{E}_{z \sim p_z}[\max(0, 1 + f(G_{\theta_g}(z)))]$, which is the second term in (2.9) with opposite sign. It is the same reason that non-saturating loss is preferred than minimax loss [50].

OKGAN has superior performance on low-dimensional data such as 2d synthetic datasets (See Table 4.1). But without additional representation power, it struggles to generate high-quality images that have been the hallmark of other GAN architectures. Thus, we add an encoder layer and let the online kernel classifier trained in a reduced feature space similar to a method in [31]. The objective function of OKGAN with encoder and the specific algorithm is shown in the following subsection.

2.2.1 Objective Function of OKGAN with Encoder

The encoder E enables us to calculate the kernel with high dimensional data such as complicated image datasets. E is also a neural network and trained in a way that separates real data(x_{real}) and fake data(x_{fake}) because the online kernel classifier should recognize $E(x_{\text{real}})$ as real and $E(x_{\text{fake}})$ as fake. From the perspective of OKGAN with the encoder, a combination of the encoder and the online kernel classifier is considered as the discriminator.

Thus, when G is parameterized by θ_g and E is parameterized by θ_e, we acquire an minmax objective of OKGAN as:

$$\max_{\theta_g} \min_{f \in \mathcal{H}, \theta_e} \mathbb{E}_{x \sim p_{\text{real}}}[\max(0, 1 - f(E_{\theta_e}(x)))] + \mathbb{E}_{z \sim p_z}[\max(0, 1 + f(E_{\theta_e}(G_{\theta_g}(z))))] \tag{2.11}$$

There are 3 steps to train OKGAN with the encoder. First, $2N$ samples($x_1, \cdots, x_{2N}$), where N samples are real and others are fake, after passing the encoder become $E(x_1), \cdots, E(x_{2N})$, and we get an online kernel classifier f from these. Second, the generator G is updated based on a generator objective function with the updated f and the existing E. Finally, the encoder E is updated based on a encoder objective function with the updated f and G. The objective function of G and E is shown as below.

$$\min_{\theta_g} \left\{ V(\theta_g) = \mathbb{E}_{z \sim p_z}[\max(0, 1 - f(E_{\theta_e}(G_{\theta_g}(z))))] \right\} \tag{2.12}$$

$$\min_{\theta_e} \left\{ V(\theta_e) = \mathbb{E}_{x \sim p_{\text{real}}}[\max(0, 1 - f(E_{\theta_e}(x)))] + \mathbb{E}_{z \sim p_z}[\max(0, 1 + f(E_{\theta_e}(G_{\theta_g}(z))))] \right\} \tag{2.13}$$

The entire process of training OKGAN with encoder is summarized in Algorithm 1.

2.3 Flexibility on Data Representation through Kernels

OKGAN successfully generates classical image datasets (see Chapter 4) by achieving flexibility on data representation through the choice of kernels. We implement commonly used kernels and mixtures of them in the online kernel classifier, which are Gaussian kernel(k_γ^{rbf}), linear kernel(k^{linear}), polynomial kernel(k^{poly}), rational quadratic kernel(k_α^{rq}), mixed Gaussian kernel(k^{rbf}), and mixed RQ-linear kernel($k^{\text{rq*}}$) [32].

Algorithm 1 OKGAN with encoder

Input: α learning rate, γ learning rate decay, N_b batch size, n_b batch size of mini-batch, B budget size, n_g number of iterations of generator per discriminator update.

while not converged **do**

  Sample $\{x_i\}_{i=1}^{N_b} \sim p_{\text{real}}$ & $\{z_i\}_{i=1}^{N_b} \sim p_z$

  if not first iteration **then**

    $g_{\theta_e} \leftarrow \nabla_{\theta_e} \frac{1}{N_b} \sum_{i=1}^{N_b} \max(0, 1 - f(E_{\theta_e}(x_i))) + \frac{1}{N_b} \sum_{i=1}^{N_b} \max(0, 1 + f(E_{\theta_e}(G_{\theta_g}(z_i))))$

    $\theta_e \leftarrow \theta_e - \alpha \cdot \text{Adam}(\theta_e, g_{\theta_e})$

  end if

  $2N_b$ inputs for the online kernel classifier are $E(x_1), \cdots, E(x_{N_b})$ and $E(G(z_1)), \cdots, E(G(z_{N_b}))$

  For the classifier, consider $2N_b$ as a total size of data and n_b as a batch size.

  $f \leftarrow \rho + \sum_{i=1}^{B} \alpha_i k(w_i, x)$ {refer to the process in Section 2.1}

  α_is and w_is are respectively coefficients and key examples saved in the current budget

  for $t = 1, \cdots, n_g$ **do**

    Sample $\{z_j\}_{j=1}^{N_b} \sim p_z$

    $g_{\theta_g} \leftarrow \nabla_{\theta_g} \frac{1}{N_b} \sum_{j=1}^{N_b} \max(0, 1 - f(E_{\theta_e}(G_{\theta_g}(z_j))))$

    $\theta_g \leftarrow \theta_g - \alpha \cdot \text{Adam}(\theta_g, g_{\theta_g})$

  end for

  $\alpha \leftarrow \alpha\gamma$

end while

$$k^{\text{rbf}}_{\gamma}(x, x') = \exp\left(-\gamma\|x - x'\|^2\right) \qquad k^{\text{rq}}_{\alpha}(x, x') = \left(1 + \frac{\|x - x'\|^2}{2\alpha}\right)^{-\alpha}$$

$$k^{\text{linear}} = \langle x, x' \rangle \qquad k^{\text{rq}*}(x, x') = k^{\text{linear}} + \sum_{\alpha \in \mathcal{A}} k^{\text{rq}}_{\alpha}(x, x')$$

$$k^{\text{poly}}(x, x') = (\gamma\langle x, x' \rangle + r)^d \qquad k^{\text{rbf}}(x, x') = \sum_{\gamma \in \Gamma} \exp\left(-\gamma\|x - x'\|^2\right)$$

Furthermore, we use an IKL method [10] that enables neural networks to learn the base kernel. First, by the Bochner's theorem [51], when a kernel function k is continuous, real valued, symmetric, and shift-invariant on $\mathbb{R}^d$, the fact that there exists a $\mathbb{P}_k(w)$ s.t.

$$k(x - x') = \int_{\mathbb{R}^d} e^{i\omega^\top (x - x')} d\mathbb{P}_k(\omega) = \mathbb{E}_{\omega \sim \mathbb{P}_k}\left[e^{i\omega^\top (x - x')}\right]$$

is equivalent to the positive definiteness of k. Let's say $\mathbb{P}(\nu)$ is a base distribution, and we sample ν from this distribution. Then, we can model $\mathbb{P}_k(\omega)$ through an implicit generative model $h_\psi(\nu) : \mathbb{R}^{d_b} \to \mathbb{R}^d$. This indicates that we indirectly sample ω from the base distribution $\mathbb{P}(\nu)$ by using the relationship $\omega = h_\psi(\nu)$ rather than directly sampling from $\mathbb{P}_k$. Thus, the above equation is changed into the following equation.

$$k_\psi(x, x') = \mathbb{E}_\nu\left[e^{ih_\psi(\nu)^\top (x - x')}\right]$$

We approximately compute the above kernel through random Fourier features [52]. Let's draw D iid samples $\nu_1, \cdots, \nu_D \in \mathbb{R}^{d_b}$ from $\mathbb{P}(\nu)$. Then, we define $z(x) : \mathbb{R}^{d_b} \to \mathbb{R}^{2D}$ and compute the kernel with this.

$$z(x) \equiv \sqrt{\frac{1}{D}}[\cos(h_\psi(\nu_1)^\top x) \ \cdots \ \cos(h_\psi(\nu_D)^\top x) \ \sin(h_\psi(\nu_1)^\top x) \ \cdots \ \sin(h_\psi(\nu_D)^\top x)]^\top$$
$$\Rightarrow k(x, x') = z(x)^\top z(x')$$

CHAPTER 3

THEORETICAL ANALYSIS

We will now turn our attention to showing that the procedure described in the previous chapter allows us to provably estimate the target distribution p_{real}. Our results rely on the use of regret minimizing algorithms over the RKHS, and are related to existing results that use online learning to guarantee convergence [53, 2], yet our results hold in a broader set of circumstances under a "generator oracle" assumption.

Let us recall the problem we aim to solve. Assume we have some family of distributions $\mathcal{D} \subset \Delta_{\mathcal{X}}$, which represent the "generator space". We are also given a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ and associated RKHS $\mathcal{H}$, our goal is to solve the following objective

$$\sup_{q \in \mathcal{D}} \inf_{f \in \mathcal{H}} \left\{ g(q, f) := \mathbb{E}_{(Y,X) \sim p_{\text{real}} \vee q}[\ell(Y f(X))] \right\}$$

We recall that for any distribution $q \in \mathcal{D}$ we have that $\inf_{f \in \mathcal{H}} g(q, f) = 1 - \Delta_{\mathcal{H},\ell}(p_{\text{real}}, q)$, and it is easy to see that $\inf_{f \in \mathcal{H}} g(q, f) \leq 1$ since we may always choose $f \equiv 0$. Now, if we can find a sequence of distributions $q_1, q_2, \ldots \in \mathcal{D}$ such that $\inf_{f \in \mathcal{H}} g(q_t, f) \to 1$ then $\Delta_{\mathcal{H},\ell}(p_{\text{real}}, q_t) \to 0$ and by Claim 2.1.1 we have that $q_t \to p_{\text{real}}$ as desired.

To obtain a provable guarantee, we need one algorithmic assumption on how efficiently our generator training is, and another on the size of the largest $\|f_t\|_{\mathcal{H}}$ we will be observed in training.

Assumption 3.0.1.

1. We have an oracle which, given any $f \in \mathcal{H}$ as input, can return a $q \in \mathcal{D}$ such that $g(q, f) \geq 1$
2. Let $f_t^* := \arg\min_{f \in \mathcal{H}} \sum_{s=1}^{t} g(q_s, f)$. Then there exists $U > 0$ s.t. $\|f_t\|_{\mathcal{H}}, \|f_t^*\|_{\mathcal{H}} \leq U$ for all t.

The first of these requirements states that, given a discriminator f, we may always train our generator g in order to "fool" this given f. This has proven to be true in practice, and simply relies on our ability to construct a generator (or space of distributions) which are suitably complex. The second assumption is somewhat stronger, as it states that the functions returned by our discriminator training algorithm must not grow too large with respect to the RKHS norm. This has proven to be true in practice, but it does depend on the complexity of the distribution p_{real}. We note that this assumption can be weakened to allow that $U = o(\sqrt{t})$ without hurting our asymptotic convergence.

Algorithm 2 Sequential Training Protocol

for $t = 1, \dots, T$ **do**
  Update: $f_t \leftarrow f_{t-1} - \eta_{t-1} \nabla_f g(q_{t-1}, \cdot)|_{f=f_{t-1}}$
  Update generator: select q_t such that $g(q_t, f_t) \geq 1$
end for
Return: $\bar{q}_T := \frac{1}{T} \sum_{t=1}^{T} q_t$

What follows is our main result. All proofs of following theorems and lemmas are elaborated in the following sections.

Theorem 3.0.2. *Let $\bar{q}_T$ be the output of Algorithm 2, and that Assumption 3.0.1 holds. Then we have that* $\lim_{T\to\infty} \inf_{f\in\mathcal{H}} g(\bar{q}_T, f) \geq 1$. *It follows that*

$$\bar{q}_T \to p_{real} \text{ as } T \to \infty.$$

This theorem relies on two key lemmas. First we show that the "quality" of $\bar{q}_T$ can be controlled in terms of the average regret of the online learning algorithm. Then we show that the average regret vanishes as $T \to \infty$.

Lemma 3.0.3. *Let's define l_t as $l_t = g(q_t, \cdot)$. Then,* $\inf_{f\in\mathcal{H}} g(\bar{q}_T, f) \geq 1 - \frac{Regret_T(l_1,\dots,l_T;\mathcal{H})}{T}$

Lemma 3.0.4. *Assume that k is a kernel with bounded norm $\|k(x,\cdot)\| \leq X$. Since ℓ is hinge loss, there exists c which satisfies $|\ell'(yf(x))| \leq c$ for all $x \in \mathcal{X}, y \in \mathcal{Y}$. The learning rate is $\eta_t = \eta t^{-\frac{1}{2}}$. Then, we have $Regret_T(l_1, \dots, l_T; \mathcal{H}) \leq \frac{1}{2}(2\eta c^2 X^2 + \frac{(U+\|f_T^*\|_{\mathcal{H}})^2}{\eta})T^{\frac{1}{2}} + \frac{\|f_T^*\|_{\mathcal{H}}^2}{\eta}$.*

3.1 Proof of Lemma 3.0.3

$$\text{Regret}_T(l_1,\ldots,l_T;\mathcal{H}) := \sum_{t=1}^{T} l_t(f_t) - \inf_{f\in\mathcal{H}}\sum_{t=1}^{T} l_t(f) = \sum_{t=1}^{T} g(q_t,f_t) - \inf_{f\in\mathcal{H}}\sum_{t=1}^{T} g(q_t,f)$$
$$\geq T - \inf_{f\in\mathcal{H}}\sum_{t=1}^{T} g(q_t,f) \ \ (\because\ g(q_t,f_t)\geq 1)$$
$$= T(1-\inf_{f\in\mathcal{H}}\frac{1}{T}\sum_{t=1}^{T} g(q_t,f)) \geq T(1-\inf_{f\in\mathcal{H}} g(\bar{q}_T,f))$$

The last inequality comes from concavity of $g(q,\cdot)$ on q. Therefore, we conclude that

$$\inf_{f\in\mathcal{H}} g(\bar{q}_T,f) \geq 1 - \frac{\text{Regret}_T(l_1,\ldots,l_T;\mathcal{H})}{T}$$

3.2 Proof of Lemma 3.0.4

The proof is closely followed by [46].

$$\|f_t - f_T^*\|_{\mathcal{H}}^2 - \|f_{t+1} - f_T^*\|_{\mathcal{H}}^2$$
$$= -\|f_{t+1}-f_t\|_{\mathcal{H}}^2 - 2 < f_{t+1}-f_t, f_t - f_T^* >_{\mathcal{H}}$$
$$= -\eta_t^2\|\nabla_f g(q_t,\cdot)|_{f=f_t}\|_{\mathcal{H}}^2 + 2\eta_t < \nabla_f g(q_t,\cdot)|_{f=f_t}, f_t - f_T^* >_{\mathcal{H}}$$
$$\geq -\eta_t^2\|\nabla_f g(q_t,\cdot)|_{f=f_t}\|_{\mathcal{H}}^2 + 2\eta_t g(q_t, f_t - f_T^*) \ \ (\because \text{ convexity of } g(\cdot,f) \text{ on } f)$$
$$= -\eta_t^2\|\mathop{\mathbb{E}}_{(y,x)\sim p_{\text{real}}\vee q_t}[\nabla_f \ell(yf(x))]|_{f=f_t}\|_{\mathcal{H}}^2 + 2\eta_t g(q_t, f_t - f_T^*)$$
$$\geq -\eta_t^2\mathop{\mathbb{E}}_{(y,x)\sim p_{\text{real}}\vee q_t}\|\nabla_f \ell(yf(x))|_{f=f_t}\|_{\mathcal{H}}^2 + 2\eta_t g(q_t, f_t - f_T^*)$$
$$= -\eta_t^2\mathop{\mathbb{E}}_{(y,x)\sim p_{\text{real}}\vee q_t}\|\ell'(yf_t(x))k(x,\cdot)\|\|_{\mathcal{H}}^2 + 2\eta_t g(q_t, f_t - f_T^*)$$
$$\geq -\eta_t^2 c^2 X^2 + 2\eta_t g(q_t, f_t - f_T^*) \ \ (\because\ |\ell'(yf(x))| \leq c \ \&\ \|k(x,\cdot)\|_{\mathcal{H}} \leq X)$$

Let's change the form of the above inequality. Since $\forall t, \|f_t\|_{\mathcal{H}} \leq U$.

$$
\begin{aligned}
&\frac{1}{\eta_t}\|f_t - f_T^*\|_{\mathcal{H}}^2 - \frac{1}{\eta_{t+1}}\|f_{t+1} - f_T^*\|_{\mathcal{H}}^2 \\
&= \frac{1}{\eta_t}(\|f_t - f_T^*\|_{\mathcal{H}}^2 - \|f_{t+1} - f_T^*\|_{\mathcal{H}}^2) + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t+1}}\right)\|f_{t+1} - f_T^*\|_{\mathcal{H}}^2 \\
&\geq -\eta_t c^2 X^2 + 2g(q_t, f_t) - 2g(q_t, f_T^*) + (U + \|f_T^*\|_{\mathcal{H}})^2 \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t+1}}\right)
\end{aligned}
$$

By telescoping,

$$
\begin{aligned}
\frac{\|f_1 - f_T^*\|_{\mathcal{H}}^2}{\eta} &\geq \frac{\|f_1 - f_T^*\|_{\mathcal{H}}^2}{\eta} - \frac{\|f_{T+1} - f_T^*\|_{\mathcal{H}}^2}{\eta_{T+1}} \\
&\geq \sum_{t=1}^{T}(-\eta_t c^2 X^2) + 2\left(\sum_{t=1}^{T} g(q_t, f_t) - \sum_{t=1}^{T} g(q_t, f_T^*)\right) \\
&\quad + (U + \|f_T^*\|_{\mathcal{H}})^2 \left(\frac{1}{\eta} - \frac{(T+1)^{\frac{1}{2}}}{\eta}\right)
\end{aligned}
$$

Also from $\eta_t = \eta t^{-\frac{1}{2}}$, we get

$$
\begin{aligned}
\sum_{t=1}^{T} \eta_t &= \eta(1 + \sum_{t=2}^{T} t^{-\frac{1}{2}}) \\
&\leq \eta(1 + \sum_{t=2}^{T} \frac{2}{\sqrt{t} + \sqrt{t-1}}) \\
&= \eta(1 + \sum_{t=2}^{T} 2(\sqrt{t} - \sqrt{t-1})) = \eta(2\sqrt{T} - 1) \leq 2\eta T^{\frac{1}{2}}
\end{aligned}
$$

Finally, since $f_1 = 0$, $(T+1)^{\frac{1}{2}} - 1 \leq T^{\frac{1}{2}}$, and $\sum_{t=1}^{T} \eta_t \leq 2\eta T^{\frac{1}{2}}$

$$
\begin{aligned}
\text{Regret}_T(l_1, \ldots, l_T; \mathcal{H}) &= \sum_{t=1}^{T} g(q_t, f_t) - \sum_{t=1}^{T} g(q_t, f_T^*) \\
&\leq \frac{1}{2}\left(2\eta c^2 X^2 + \frac{(U + \|f_T^*\|_{\mathcal{H}})^2}{\eta}\right) T^{\frac{1}{2}} + \frac{\|f_T^*\|_{\mathcal{H}}^2}{\eta}
\end{aligned}
$$

3.3 Proof of Theorem 3.0.2

By the results of Lemma 3.0.3 and Lemma 3.0.4,

$$\begin{aligned}
\lim_{T\to\infty} \inf_{f\in\mathcal{H}} g(\bar{q}_T, f) &\geq 1 - \lim_{T\to\infty} \frac{\text{Regret}_T(l_1,\dots,l_T;\mathcal{H})}{T} \\
&\geq 1 - \lim_{T\to\infty} \left(\frac{1}{2}\Big(2\eta c^2 X^2 + \frac{(U + \|f_T^*\|_{\mathcal{H}})^2}{\eta}\Big) T^{-\frac{1}{2}} + \frac{\|f_T^*\|_{\mathcal{H}}^2}{\eta T} \right) \\
&\geq 1 - \lim_{T\to\infty} \left(\frac{1}{2}\Big(2\eta c^2 X^2 + \frac{4U^2}{\eta}\Big) T^{-\frac{1}{2}} + \frac{U^2}{\eta T} \right) \\
&= 1
\end{aligned}$$

By Assumption 3.0.1 ($\inf_{f\in\mathcal{H}} g(q,f) \leq 1$) and Theorem 3.0.2,

$$\lim_{T\to\infty} \inf_{f\in\mathcal{H}} g(\bar{q}_T, f) = 1$$

We recall that for any distribution $q \in \mathcal{D}$ we have that $\inf_{f\in\mathcal{H}} g(q,f) = 1 - \Delta_{\mathcal{H},\ell}(p_{\text{real}}, q)$. Let's consider a sequence of distributions $\bar{q}_1, \bar{q}_2, \ldots \in \mathcal{D}$ such that $\inf_{f\in\mathcal{H}} g(\bar{q}_T, f) \to 1$. Then, we obtain

$$\lim_{T\to\infty} \Delta_{\mathcal{H},\ell}(p_{\text{real}}, \bar{q}_T) = 0$$

By Claim 2.1.1, we finally conclude that

$$\lim_{T\to\infty} \bar{q}_T = p_{\text{real}}$$

CHAPTER 4
EXPERIMENTS

In this chapter, we provide experimental results of OKGANs in both quantitative and qualitative ways. We quantitatively compare OKGANs with other GANs on 2D synthetic datasets and show how well OKGANs solve the mode collapse problem, using quantitative metrics proposed earlier in [24, 20, 21]. Also, we compute $D_{\text{KL}}(p_{\text{fake}}||p_{\text{real}})$ [54] of OKGANs and other GANs on MNIST and our new dataset called "Poisson Digits", where p_{real} is a real distribution and p_{fake} is a fake distribution. These quantitative analyses validate our claim that OKGANs are good at matching the target distribution. For the overall evaluation of OKGANs, we use standard metrics such as Inception score [35] and FID [55] on CIFAR-10. Moreover, we analyze OKGANs qualitatively on 2D synthetic datasets and classical image datasets and observe that OKGANs do not suffer from cycling behavior on 2D synthetic datasets, shown through our novel visualization technique.

4.1 Experimental Setup

4.1.1 Datasets

We use 2D synthetic datasets for the quantitative analysis on mode collapse, specifically, we use 2D-grid, 2D-ring, and 2D-gmm. The 2D-grid and 2D-ring datasets are Gaussian mixtures with 25 and 8 modes, organized in a grid shape and a ring shape respectively. The specific setup of these two datasets is the same as [20]. The standard deviation of each Gaussian in 2D-grid is 0.05, and a grid value of four edges is (-4, -4), (-4, 4), (4, -4), (4, 4). In the case of 2D ring, the standard deviation of each Gaussian is 0.01, and the radius of a ring shape is 1. Since we regard 2D-grid and 2D-ring as discrete distributions in our analysis, we further experiment on one of the continuous distributions, 2D-gmm, a mixture of 5

Gaussian distributions with different means and covariances. Each mean of the Gaussian mixture is (-3, 3), (0, 0), (2, 3), (5, 1), (7, 3), and we choose each Gaussian with a different ratio [1, 2, 2.5, 1, 1] to get samples. The corresponding covariance for each mean is

$$\begin{bmatrix} 0.83 & 0.5 \\ 0.5 & 0.83 \end{bmatrix} \quad \begin{bmatrix} 0.83 & -0.5 \\ -0.5 & 0.83 \end{bmatrix} \quad \begin{bmatrix} 0.77 & 0.57 \\ 0.57 & 0.77 \end{bmatrix} \quad \begin{bmatrix} 1.17 & 0.83 \\ 0.83 & 1.17 \end{bmatrix} \quad \begin{bmatrix} 0.70 & 0.002 \\ 0.002 & 0.70 \end{bmatrix}$$

We use other 2D synthetic datasets such as 2D-circle, 2D-swissroll, and 2D-2spirals for the qualitative analysis. For 2D-circle, we follow the experiment setup in [21]. 100 Gaussian distributions are on a circle with a radius 2, and three identical Gaussians are located at the center of the circle. The standard deviation of each Gaussian is 0.05. The geometric structures of the 2D-swissroll and the 2D-2spirals datasets can be found in Figure 4.1. Also, we do further quantitative analysis on MNIST and Poisson Digits. Each image of Poisson Digits dataset is 28×84, which consists of three horizontally connected MNIST images. The number in this dataset follows the Poisson distribution whose mean is 100. Additionally, SVHN (Street View House Numbers), CelebA, and CIFAR-10 are all used for the qualitative analysis.

4.1.2 Neural Network Architectures

For the quantitative analysis and cycling behavior on 2D synthetic datasets with OKGAN, we need a neural network architecture only for the generator, since the discriminator is formed by the online-kernel classifier. The generator architecture of OKGAN is the same as one of PacGAN [20], and we use the online kernel classifier instead of the discriminator of PacGAN. In terms of the quantitative analysis on MNIST and Poisson Digits, we use a pre-trained autoencoder and train the fully connected generator in the code space with a reduced dimension [29]. By means of the pre-trained autoencoder, we do not need an additional encoder to train our GAN model, which is necessary for more complicated datasets. In experiments with classical image datasets for the qualitative analysis, we use DCGAN [56] architecture to the OKGAN. The generator of OKGAN is the same as that of DCGAN, and the encoder of OKGAN is a reverse architecture of the generator. The output dimension of the encoder is 100 for SVHN, CelebA, and CIFAR-10.

4.1.3 Kernel Choice

The appropriate choice of a kernel is an essential part of the online kernel classifier. Since all 2D synthetic datasets are a Gaussian mixture distribution, we choose Gaussian kernel for experiments on 2D synthetic datasets. When it comes to learning a real distribution of the 2D synthetic datasets, it is significant to set the value γ in the kernel. Empirically, γ should be larger when modes become closer. The γ value for 2D-grid, 2D-ring, and 2D-gmm is respectively 1, 16, and 10, which is constant during the training.

We use the IKL method [10] with one hidden layer neural network on MNIST and Poisson Digits dataset, which enables neural networks to learn the base kernel. For the qualitative analysis on classical image datasets, polynomial kernel ($\gamma = 0.01, r = 0, d = 3$) works well on SVHN dataset. For CelebA and CIFAR-10 dataset, we use the mixed gaussian kernel, where $\Gamma = \{\frac{1}{2\times 2^2}, \frac{1}{2\times 5^2}, \frac{1}{2\times 10^2}, \frac{1}{2\times 20^2}, \frac{1}{2\times 40^2}, \frac{1}{2\times 80^2}\}$ [32]. All coefficients of kernels in these experiments are constant during training.

4.1.4 Evaluation Metrics

The evaluation metrics for 2D synthetic datasets such as 2D-grid and 2D-ring are also previously proposed by [24, 20], which are *# of mode*, *percentage of high-quality samples(% of high quality)*, and *reverse Kullback-Leibler(KL) divergence*. Let's say a standard deviation of Gaussian is σ, and a set of generated samples is $\{x_1, \cdots, x_N\} = \mathcal{X}$. An entire set of modes is $C = \{c_1, \cdots, c_M\}$. Then, *# of mode*, *% of high-quality* is defined as:

$$\textit{\# of mode} = \sum_{i=1}^{M} \mathbb{1}[\min_{x \in \mathcal{X}} \|c_i - x\|] < 3\sigma]$$
$$\textit{\% of high-quality} = \frac{100}{N} \sum_{i=1}^{N} \mathbb{1}[\min_{c \in \mathcal{C}} \|x_i - c\|] < 3\sigma]$$

We calculate *reverse KL divergence* [54] by considering real and fake distributions as discrete distributions. GAN with high *# of mode*, high *% of high-quality*, and low *reverse KL divergence* is regarded as good at solving mode collapse and learning real distribution well. For 2D-gmm, by means of a method in [57], We estimate reverse KL divergence of two distributions from their samples via k-nearest-neighbor(k-NN) since we regard distributions as continuous densities. In this case, we do not consider *# of mode* and *% of high-quality*. Let $\{X_1, \cdots, X_n\}$ and $\{Y_1, \cdots, Y_m\}$ be d-dimensional samples respectively from distribution p and q. If the Euclidean distance between X_i and its k-NN in $\{X_j\}_{j \neq i}$ is $\rho_k(i)$ and the distance between X_i and its k-NN in $\{Y_j\}$ is $\nu_k(i)$, the estimator for KL divergence becomes

$$D_{\mathrm{KL}}(p||q) \approx \frac{d}{n} \sum_{i=1}^{n} \log \frac{\nu_k(i)}{\rho_k(i)} + \log \frac{m}{n-1}$$

When computing reverse KL divergence on MNIST and Poisson Digits, we use a pre-trained LeNet-5 MNIST classifier [58] and classify each generated number based on this classifier. For example, when we do experiments on Poisson Digits, we determine three digits respectively in one image of Poisson Digits through the LeNet-5 classifier and form a discrete distribution with the generated fake samples. Then, we compute reverse KL divergence of the real and fake discrete distribution on MNIST and Poisson Digits, where real MNIST and Poisson Digits respectively follow the uniform distribution and Poisson distribution. We consider MNIST a uniform distribution because a training set of MNIST contains the same amount of examples from 0 to 9. Finally, we use the Inception score and FID to measure the performance of OKGAN on CIFAR-10.

Table 4.1: Quantitative Results on 2D Synthetic Datasets. The max # of modes of 2D-grid and 2D-ring is respectively 25 and 8.

	2D-grid			2D-ring			2D-gmm
	#modes	high quality(%)	reverse KL	#modes	high quality(%)	reverse KL	reverse KL
PacGAN	23.8	91.3	0.13	7.9	95.6	0.07	-
BourGAN	24.8	**95.1**	0.036	7.9	**100.0**	0.019	0.066
GMMN	**25.0**	88.2	**0.005**	**8.0**	93.8	**0.001**	**0.007**
OKGAN	**25.0**	94.3	**0.005**	**8.0**	96.2	0.002	0.009

4.2 Quantitative Analysis

4.2.1 2D Synthetic Datasets

For 2D synthetic datasets, we compare OKGAN with the three powerful unconditional GANs on solving mode collapse, called PacGAN [20], BourGAN [21], and GMMN [29]. We apply neural network architectures of PacGAN [20] to OKGAN, BourGAN, and GMMN for the fair comparison. The quantitative performance of these four GANs on 2D-grid, 2D-ring, and 2D-gmm is summarized in Table 4.1. Our results are averaged over 10 trials.

As you can see from Table 4.1 and Figure 4.1, 4.2, OKGAN and GMMN show a way better performance than other GANs in terms of mitigating mode collapse. In contrast with other GANs, OKGAN and GMMN capture all modes for 2D-grid and 2D-ring. The remarkable point is that *reverse KL divergence* of OKGAN and GMMN are much lower than other GANs for all three datasets. This indicates that OKGAN and GMMN not only produce all modes but also match the target distribution. Since OKGAN and GMMN both take advantage of a kernel method, these two GANs increase the diversity of generated samples. However, as GMMN does not have any discriminator and overly focuses on reducing Maximum Mean Discrepancy (MMD) of the target and fake distribution, the *percentage of high-quality samples* of GMMN is lower than that of OKGAN. On the other hand, due to the high discriminative power of the online kernel classifier as the discriminator, OKGAN achieves both diversity and high sample quality.

Table 4.2: Quantitative Results on Poisson Digits Dataset

	reverse KL
VGAN	2.269
GMMN (rbf)	0.126
OKGAN (rbf)	**0.291**
OKGAN (IKL)	**0.054**

4.2.2 MNIST & Poisson Digits

We evaluate OKGAN and Vanilla GAN (VGAN) on MNIST how much generated distributions match the true uniform distribution. We train OKGANs by using IKL and a pre-trained autoencoder. However, since VGAN fails to generate MNIST images with the pre-trained autoencoder, we compare reverse KL-divergence of OKGAN to that of original VGAN without the pre-trained autoencoder. The reverse KL-divergence of OKGAN is 0.01, and this value is lower than 0.015, which is the reverse KL-divergence of VGAN. Moreover, we measure reverse KL-divergence of VGAN, GMMN, and OKGAN on Poisson Digits. The Gaussian kernel is used for GMMN, and Gaussian kernel and IKL are used for OKGAN. We train all GAN models with the same fully-connected generator architecture and a pre-trained autoencoder for Poisson Digits. From the quantitative results shown in Table 4.2, we can conclude that OKGAN with IKL is the best at matching Poisson distribution.

4.2.3 CIFAR-10

The Inception score (IS) and FID are the most widely used quantitative metrics for GANs. As we use DCGAN architecture for CIFAR-10 training, we compare OKGAN to other GANs whose models are based on DCGAN. When it comes to comparing OKGAN to other kernel-based GANs, OKGAN performs much better than GMMN and Kernel-GAN [34] but shows a slightly lower score than MMD GAN [31].

Table 4.3: Inception Scores(IS) and FID

	IS	FID
GMMN [29]	3.47	-
WGAN-GP [38]	5.99	-
DRAGAN^g [2]	6.11	-
MMD GAN [31]	6.17	-
Kernel GAN [34]	4.22	-
SN-GAN [41]	7.42	29.3
OKGAN(ours)	**5.98**	**41.2**

4.3 Qualitative Analysis

4.3.1 Generated Image Quality

Initially, we engage in a qualitative comparison of images generated by the OKGAN method against those produced by the BourGAN and GMMN techniques, using 2D synthetic datasets (See Figure 4.1, 4.2). The comparison also includes the original images, providing a comprehensive evaluation of the respective methods' performance and capabilities. As you can observe in Figure 4.1, BourGAN sometimes fails to capture all modes in 2D-grid and the center in 2D-circle. Overall, only OKGAN achieves high sample quality and diversity at the same time. Then, we check each image quality of VGAN and OKGAN for MNIST, as in Figure 4.3. We do further qualitative analysis on Poisson Digits. We compare VGAN, GMMN with the Gaussian kernel, and OKGAN with IKL. In Figure 4.4, we can easily observe that GMMN and OKGAN generate various numbers, while VGAN only generates a few numbers. To qualitatively measure how much the true Poisson distribution and each fake distribution is close, we represent distributions by using histograms, as in Figure 4.5. Finally, We qualitatively compare OKGAN with DCGAN for SVHN, CelebA, and CIFAR-10. From Figure 4.6, 4.7, and 4.8, we observe that OKGANs successfully generate high-quality samples for all image datasets.

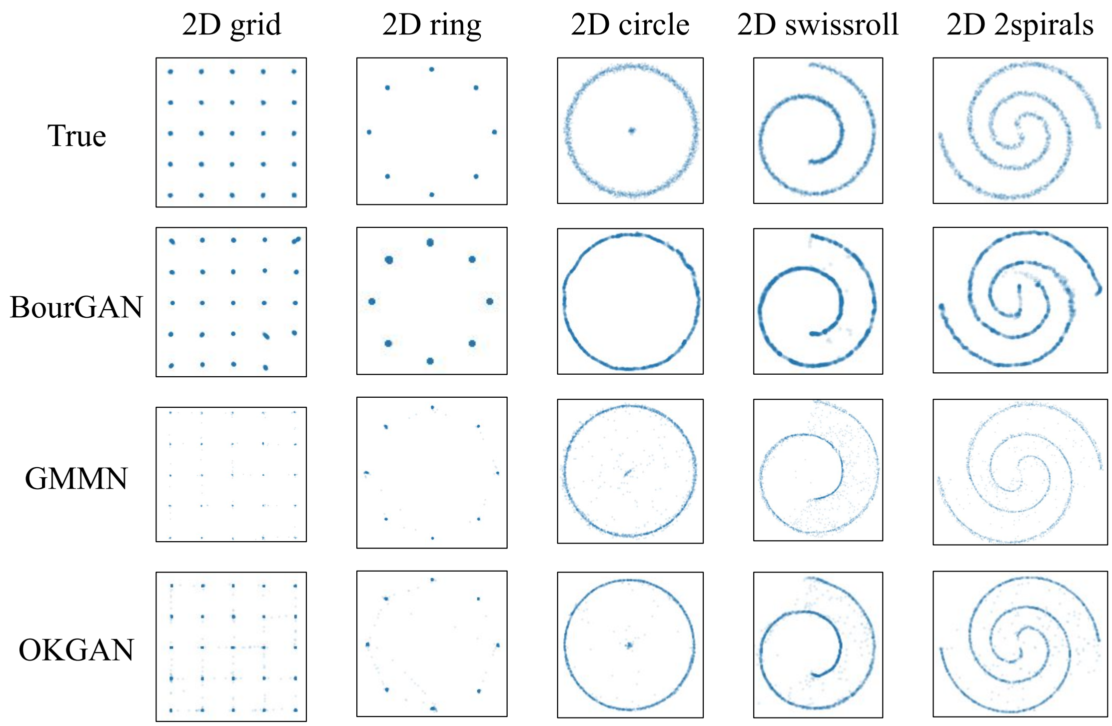


Figure 4.1: Qualitative comparison on 2D synthetic datasets.

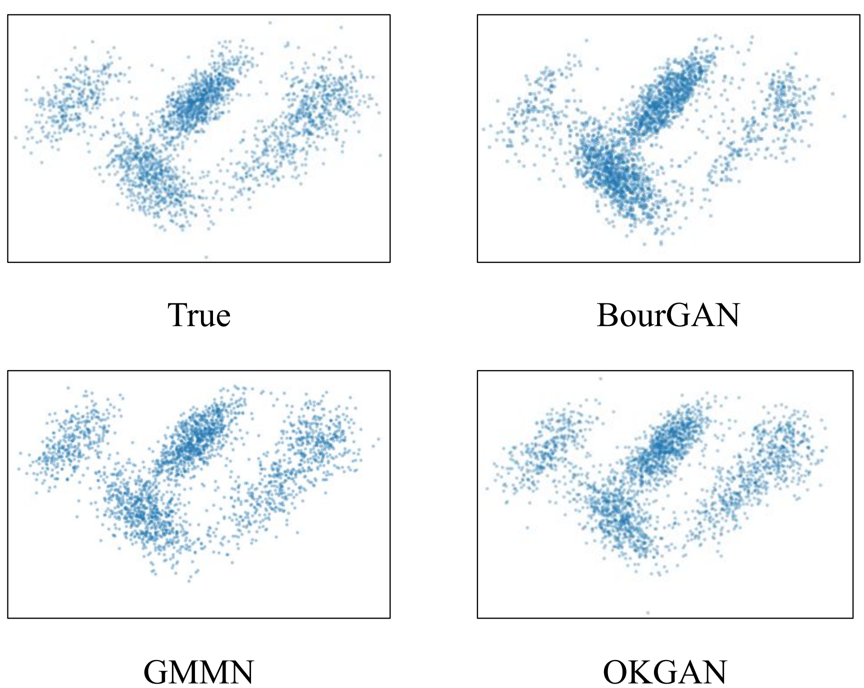
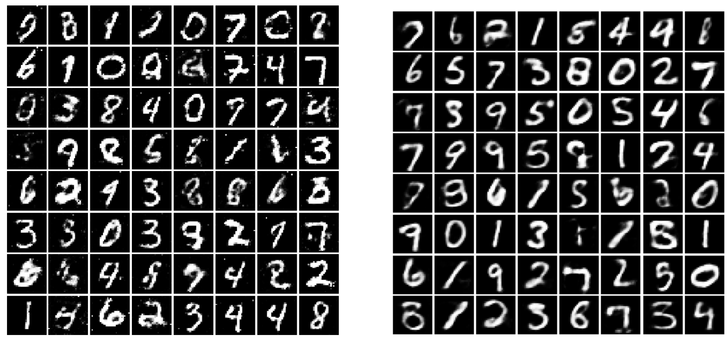
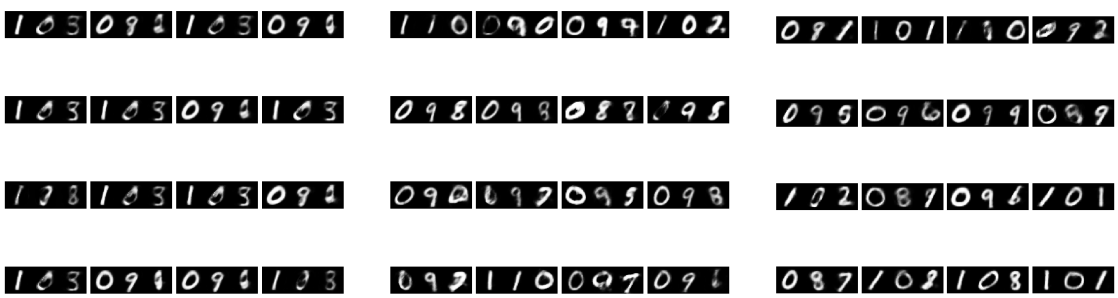


Figure 4.2: Qualitative comparison on the 2D-gmm dataset.



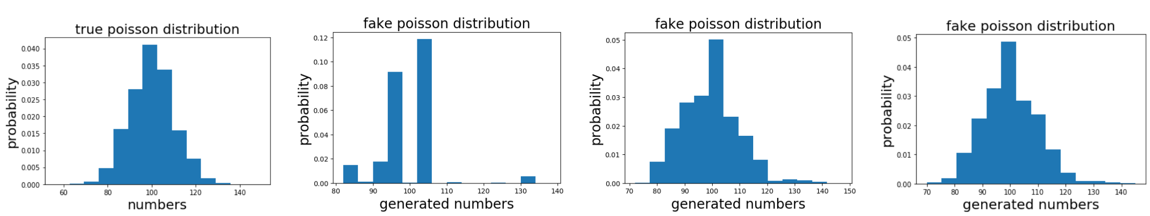
(a) VGAN (b) OKGAN

Figure 4.3: Qualitative comparison on MNIST dataset



(a) VGAN (b) GMMN (rbf) (c) OKGAN (IKL)

Figure 4.4: Qualitative comparison on Poisson Digits



(a) True Poisson (b) VGAN (c) GMMN (d) OKGAN

Figure 4.5: Histograms of the true Distribution and fake distributions generated by VGAN, GMMN, and OKGAN on Our New Dataset ”Poisson Digits”.

(a) DCGAN (b) OKGAN

Figure 4.6: Qualitative comparison on SVHN dataset

(a) DCGAN (b) OKGAN

Figure 4.7: Qualitative Comparison on CelebA Dataset

(a) DCGAN (b) OKGAN

Figure 4.8: Qualitative comparison on CIFAR-10 dataset

4.3.2 Cycling behavior

In Figure 1.1(a), we can clearly observe that Vanilla GAN (VGAN) shows the cycling behavior during training, which means the discriminator does cycle and fails to give meaningful information to the generator. In a parameter-based alternative update framework such as VGAN, it is challenging for the discriminator to chase the transitions of the generator with a slow pace of parameter updates. However, in the case of OKGAN, by obtaining a closed-form discriminator with a non-parametric kernel method, the discriminator is updated in a more aggressive fashion and separates real and fake data more effectively at every update. As you can see in Figure 1.1(b), the discriminator of OKGAN tends to find the optimal discriminator with no apparent cycling behavior, which also leads to solving the mode collapse problem in the end.

CHAPTER 5

DISCUSSION

In this work, we propose OKGAN, a new type of GAN whose discriminator contains the online kernel classifier. We provide a novel method for visualizing the cycling behavior of GANs and empirically show that OKGAN does not suffer from this issue. Moreover, with a kernel-based non-parametric discriminator, OKGAN successfully learns 2D synthetic data with no mode collapse and generates high quality samples in image datasets. In future, the deeper theoretical understanding on dynamics of GANs with the non-parametric discriminator can be discussed.

REFERENCES

[1] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[2] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of gans," *arXiv preprint arXiv:1705.07215*, 2017.

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

[4] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2794–2802.

[5] L. Mescheder, S. Nowozin, and A. Geiger, "The numerics of gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 1825–1835.

[6] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for gans do actually converge?" *arXiv preprint arXiv:1801.04406*, 2018.

[7] C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng, "Training gans with optimism," *arXiv preprint arXiv:1711.00141*, 2017.

[8] D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, and T. Graepel, "The mechanics of n-player differentiable games," *arXiv preprint arXiv:1802.05642*, 2018.

[9] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang, "Generalization and equilibrium in generative adversarial nets (gans)," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 224–232.

[10] C.-L. Li, W.-C. Chang, Y. Mroueh, Y. Yang, and B. Póczos, "Implicit kernel learning," *arXiv preprint arXiv:1902.10214*, 2019.

[11] P. Zhang, Q. Liu, D. Zhou, T. Xu, and X. He, "On the discrimination-generalization tradeoff in gans," *arXiv preprint arXiv:1711.02771*, 2017.

[12] S. Liu, O. Bousquet, and K. Chaudhuri, "Approximation and convergence properties of generative adversarial learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 5545–5553.

[13] Y. Bai, T. Ma, and A. Risteski, "Approximability of discriminators implies diversity in gans," *arXiv preprint arXiv:1806.10586*, 2018.

[14] V. Nagarajan and J. Z. Kolter, "Gradient descent gan optimization is locally stable," in *Advances in neural information processing systems*, 2017, pp. 5585–5595.

[15] J. Li, A. Madry, J. Peebles, and L. Schmidt, "On the limitations of first-order approximation in gan dynamics," *arXiv preprint arXiv:1706.09884*, 2017.

[16] T. Liang and J. Stokes, "Interaction matters: A note on non-asymptotic local convergence of generative adversarial networks," *arXiv preprint arXiv:1802.06132*, 2018.

[17] W. Nie and A. Patel, "Towards a better understanding and regularization of gan training dynamics," *arXiv preprint arxiv:1806.09235*, 2019.

[18] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[19] H. Berard, G. Gidel, A. Almahairi, P. Vincent, and S. Lacoste-Julien, "A closer look at the optimization landscapes of generative adversarial networks," *arXiv preprint arXiv:1906.04848*, 2019.

[20] Z. Lin, A. Khetan, G. Fanti, and S. Oh, "Pacgan: The power of two samples in generative adversarial networks," in *Advances in neural information processing systems*, 2018, pp. 1498–1507.

[21] C. Xiao, P. Zhong, and C. Zheng, "Bourgan: Generative networks with metric embeddings," in *Advances in Neural Information Processing Systems*, 2018, pp. 2269–2280.

[22] A. B. Dieng, F. J. Ruiz, D. M. Blei, and M. K. Titsias, "Prescribed generative adversarial networks," *arXiv preprint arXiv:1910.04302*, 2019.

[23] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.

[24] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, "Veegan: Reducing mode collapse in gans using implicit variational learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 3308–3318.

[25] S. Azadi, C. Olsson, T. Darrell, I. Goodfellow, and A. Odena, "Discriminator rejection sampling," *arXiv preprint arXiv:1810.06758*, 2018.

[26] C. Tao, L. Chen, R. Henao, J. Feng, and L. C. Duke, "Chi-square generative adversarial network," in *International conference on machine learning*, 2018, pp. 4887–4896.

[27] R. Turner, J. Hung, E. Frank, Y. Saatchi, and J. Yosinski, “Metropolis-hastings generative adversarial networks,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 6345–6353.

[28] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, “A kernel method for the two-sample-problem,” in *Advances in neural information processing systems*, 2007, pp. 513–520.

[29] Y. Li, K. Swersky, and R. Zemel, “Generative moment matching networks,” in *International Conference on Machine Learning*, 2015, pp. 1718–1727.

[30] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani, “Training generative neural networks via maximum mean discrepancy optimization,” *arXiv preprint arXiv:1505.03906*, 2015.

[31] 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. 2203–2213.

[32] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, “Demystifying mmd gans,” *arXiv preprint arXiv:1801.01401*, 2018.

[33] W. Wang, Y. Sun, and S. Halgamuge, “Improving mmd-gan training with repulsive loss function,” *arXiv preprint arXiv:1812.09916*, 2018.

[34] M. Sinn and A. Rawat, “Non-parametric estimation of jensen-shannon divergence in generative adversarial network training,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2018, pp. 642–651.

[35] 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.

[36] 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.

[37] 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.

[38] 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. 5767–5777.

[39] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[40] 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.

[41] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[42] L. Bottou and C.-J. Lin, "Support vector machine solvers," *Large scale kernel machines*, vol. 3, no. 1, pp. 301–320, 2007.

[43] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[44] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.

[45] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *Journal of machine learning research*, vol. 2, no. Nov, pp. 67–93, 2001.

[46] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE transactions on signal processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[47] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[48] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A Kernel-Based perceptron on a budget," *SIAM J. Comput.*, vol. 37, no. 5, pp. 1342–1372, Jan. 2008.

[49] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[50] W. Fedus, M. Rosca, B. Lakshminarayanan, A. M. Dai, S. Mohamed, and I. Goodfellow, "Many paths to equilibrium: Gans do not need to decrease a divergence at every step," *arXiv preprint arXiv:1710.08446*, 2017.

[51] W. Rudin, *Fourier analysis on groups*. Wiley Online Library, 1962, vol. 121967.

[52] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2008, pp. 1177–1184.

[53] P. Grnarova, K. Y. Levy, A. Lucchi, T. Hofmann, and A. Krause, "An online learning approach to generative adversarial networks," *arXiv preprint arXiv:1706.03269*, 2017.

[54] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[55] 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. 6626–6637.

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

[57] Q. Wang, S. R. Kulkarni, and S. Verdú, "Divergence estimation for multidimensional densities via k-nearest-neighbor distances," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2392–2405, 2009.

[58] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.