Generalization and Equilibrium in Generative Adversarial Nets (GANs)

Sanjeev Arora [1] **Rong Ge** [2] **Yingyu Liang** [1] **Tengyu Ma** [1] **Yi Zhang** [1]

Abstract

It is shown that training of generative adversarial network (GAN) may not have good *generalization* properties; e.g., training may appear successful but the trained distribution may be far from target distribution in standard metrics. However, generalization does occur for a weaker metric called *neural net distance*. It is also shown that an approximate pure equilibrium exists in the discriminator/generator game for a natural training objective (Wasserstein) when generator capacity and training set sizes are moderate. This existence of equilibrium inspires MIX+GAN protocol, which can be combined with any existing GAN training, and empirically shown to improve some of them.

1. Introduction

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have become one of the dominant methods for fitting generative models to complicated real-life data, and even found unusual uses such as designing good cryptographic primitives (Abadi & Andersen, 2016). See a survey by (Goodfellow, 2016). Various novel architectures and training objectives were introduced to address perceived shortcomings of the original idea, leading to more stable training and more realistic generative models in practice (see (Odena et al., 2016; Huang et al., 2017; Radford et al., 2016; Tolstikhin et al., 2017; Salimans et al., 2016; Jiwoong Im et al., 2016; Durugkar et al., 2016) and the reference therein).

The goal is to train a *generator* deep net whose input is a standard Gaussian, and whose output is a sample from some distribution $\mathcal{D}$ on $\mathbb{R}^d$, which has to be close to some target distribution $\mathcal{D}_{real}$ (which could be, say, real-life images represented using raw pixels). The training uses samples from $\mathcal{D}_{real}$ and together with the generator net also trains a *discriminator* deep net trying to maximise its ability to distinguish between samples from $\mathcal{D}_{real}$ and $\mathcal{D}$. So long as the discriminator is successful at this task with nonzero probability, its success can be used to generate a feedback (using backpropagation) to the generator, thus improving its distribution $\mathcal{D}$. Training is continued until the generator *wins*, meaning that the discriminator can do no better than random guessing when deciding whether or not a particular sample came from $\mathcal{D}$ or $\mathcal{D}_{real}$. This basic iterative framework has been tried with many training objectives; see Section 2. But it has been unclear what to conclude when the generator wins this game: is $\mathcal{D}$ close to $\mathcal{D}_{real}$ in some metric? One seems to need some extension of *generalization theory* that would imply such a conclusion. The hurdle is that distribution $\mathcal{D}_{real}$ could be complicated and may have many peaks and valleys; see Figure 1. The number of peaks (modes) may even be exponential in d. (Recall the *curse of dimensionality*: in d dimensions there are $\exp(d)$ directions whose pairwise angle exceeds say $\pi/3$, and each could be the site of a peak.) Whereas the number of samples from $\mathcal{D}_{real}$ (and from $\mathcal{D}$ for that matter) used in the training is a lot fewer, and thus may not reflect most of the peaks and valleys of $\mathcal{D}_{real}$.

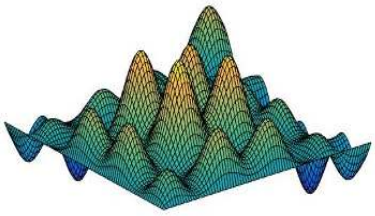

Figure 1. Probability density $\mathcal{D}_{real}$ with many peaks and valleys

A standard analysis due to (Goodfellow et al., 2014) shows that when the discriminator capacity (= number of parameters) and number of samples is "large enough", then a win by the generator implies that $\mathcal{D}$ is very close to $\mathcal{D}_{real}$ (see Section 2). But the discussion in the previous paragraph raises the possibility that "sufficiently large" in this analysis may need to be $\exp(d)$.

Another open theoretical issue is whether an equilibrium always exists in this game between generator and discrim-

Authors listed in alphabetical order. [1]Princeton University, Princeton NJ [2]Duke University, Durham NC. Correspondence to: Rong Ge <rongge@cs.duke.edu>, Yi Zhang <y.zhang@cs.princeton.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

inator. Just as a zero gradient is a necessary condition for standard optimization to halt, the corresponding necessary condition in a two-player game is an equilibrium. Conceivably some of the instability often observed while training GANs could just arise because of lack of equilibrium. (Recently Arjovsky et al. (2017) suggest that using their Wasserstein objective in practice reduces instability, but we still lack proof of existence of an equilibrium.) Standard game theory is of no help here because we need a so-called pure equilibrium, and simple counter-examples such as *rock/paper/scissors* show that it doesn't exist in general[1].

1.1. Our Contributions

We formally define generalization for GANs in Section 3 and show that for previously studied notions of distance between distributions, generalization is not guaranteed (Lemma 1). In fact we show that the generator can win even when $\mathcal{D}$ and $\mathcal{D}_{real}$ are arbitrarily far in any standard metric.

However, we can guarantee some weaker notion of generalization by introducing a new metric on distributions, the *neural net distance*. We show that generalization does happen with moderate number of training examples (i.e., when the generator wins, the two distributions must be close in neural net distance). However, this weaker metric comes at a cost: it can be near-zero even when the trained and target distributions are very far (Section 3.1)

To explore the existence of equilibria we turn in Section 4 to infinite mixtures of generator deep nets. These are clearly vastly more expressive than a single generator net: e.g., a standard result in bayesian nonparametrics says that every probability density is closely approximable by an infinite mixture of Gaussians (Ghosh et al., 2003). Thus unsurprisingly, an infinite mixture should win the game. We then prove rigorously that even a finite mixture of fairly reasonable size can closely approximate the performance of the infinite mixture (Theorem 4.2).

This insight also allows us to show for a natural GAN setting with Wasserstein objective there exists an *approximate* equilibrium that is pure. (Roughly speaking, an approximate equilibrium is one in which neither of the players can gain much by deviating from their strategies.)

This existence proof for an approximate equilibrium unfortunately involves a quadratic blowup in the "size" of the generator (which is still better than the naive exponential blowup one might expect). Improving this is left for future theoretical work. But we propose a heuristic approximation to the mixture idea to introduce a new framework for training that we call MIX+GAN. It can be added on top of any existing GAN training procedure, including those that use divergence objectives. Experiments in Section 6 show that for several previous techniques, MIX+GAN stabilizes the training, and in some cases improves the performance.

[1]Such counterexamples are easily turned into toy GAN scenarios with generator and discriminator having finite capacity, and the game lacks a pure equilibrium. See supplementary material.

2. Preliminaries

Notations. Throughout the paper we use d for the dimension of samples, and p for the number of parameters in the generator/discriminator. In Section 3 we use m for number of samples.

Generators and discriminators. Let $\{G_u, u \in \mathcal{U}\}$ ($\mathcal{U} \subset \mathbb{R}^p$) denote the class of generators, where G_u is a function — which is often a neural network in practice — from $\mathbb{R}^\ell \to \mathbb{R}^d$ indexed by u that denotes the parameters of the generators. Here $\mathcal{U}$ denotes the possible ranges of the parameters and without loss of generality we assume $\mathcal{U}$ is a subset of the unit ball[2]. The generator G_u defines a distribution $\mathcal{D}_{G_u}$ as follows: generate h from ℓ-dimensional spherical Gaussian distribution and then apply G_u on h and generate a sample $x = G_u(h)$ of the distribution $\mathcal{D}_{G_u}$. We drop the subscript u in $\mathcal{D}_{G_u}$ when it's clear from context.

Let $\{D_v, v \in \mathcal{V}\}$ denote the class of discriminators, where D_v is function from $\mathbb{R}^d$ to $[0, 1]$ and v is the parameters of D_v. The value $D_v(x)$ is usually interpreted as the probability that the sample x comes from the real distribution $\mathcal{D}_{real}$ (as opposed to the generated distribution $\mathcal{D}_G$).

We assume G_u and D_v are L-Lipschitz with respect to their parameters. That is, for all $u, u' \in \mathcal{U}$ and any input h, we have $\|G_u(h) - G_{u'}(h)\| \leq L\|u - u'\|$ (similar for D).

Notice, this is distinct from the assumption (which we will also sometimes make) that functions G_u, D_v are Lipschitz: that focuses on the change in function value when we change x, while keeping u, v fixed[3].

Objective functions. The standard GAN training (Goodfellow et al., 2014) consists of training parameters u, v so as to optimize an objective function:

$$\min_{u\in\mathcal{U}} \max_{v\in\mathcal{V}} \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{real}} [\log D_v(x)] + \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{G_u}} [\log(1 - D_v(x))]. \tag{1}$$

Intuitively, this says that the discriminator D_v should give high values $D_v(x)$ to the real samples and low values $D_v(x)$ to the generated examples. The log function was suggested because of its interpretation as the likelihood,

[2]Otherwise we can scale the parameter properly by changing the parameterization.

[3]Both Lipschitz parameters can be exponential in the number of layers in the neural net, however our Theorems only depend on the log of the Lipschitz parameters

and it also has a nice information-theoretic interpretation described below. However, in practice it can cause problems since $\log x \to -\infty$ as $x \to 0$. The objective still makes intuitive sense if we replace log by any monotone function $\phi : [0,1] \to \mathbb{R}$, which yields the objective:

$$\min_{u\in\mathcal{U}} \max_{v\in\mathcal{V}} \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{real}} [\phi(D_v(x))] + \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{G_u}} [\phi(1-D_v(x))]. \quad (2)$$

We call function ϕ the *measuring function*. It should be concave so that when $\mathcal{D}_{real}$ and $\mathcal{D}_G$ are the same distribution, the best strategy for the discriminator is just to output $1/2$ and the optimal value is $2\phi(1/2)$. In later proofs, we will require ϕ to be bounded and Lipschitz. Indeed, in practice training often uses $\phi(x) = \log(\delta + (1-\delta)x)$ (which takes values in $[\log\delta, 0]$ and is $1/\delta$-Lipschitz) and the recently proposed Wasserstein GAN (Arjovsky et al., 2017) objective uses $\phi(x) = x$.

Training with finite samples. The objective function (2) assumes we have infinite number of samples from $\mathcal{D}_{real}$ to estimate the value $\mathbb{E}_{x\sim\mathcal{D}_{real}}[\phi(D_v(x))]$. With finite training examples $x_1, \ldots, x_m \sim \mathcal{D}_{real}$, one uses $\frac{1}{m}\sum_{i=1}^{m}[\phi(D_v(x_i))]$ to estimate the quantity $\mathbb{E}_{x\sim\mathcal{D}_{real}}[\phi(D_v(x))]$. We call the distribution that gives $1/m$ probability to each of the x_i's the *empirical version* of the real distribution. Similarly, one can use a empirical version to estimate $\mathbb{E}_{x\sim\mathcal{D}_{G_u}}[\phi(1-D_v(x))]$.

Standard interpretation via distance between distributions. Towards analyzing GANs, researchers have assumed access to *infinite* number of examples and that the discriminator is chosen optimally within some *large* class of functions that contain all possible neural nets. This often allows computing analytically the optimal discriminator and therefore removing the maximum operation from the objective (2), which leads to some interpretation of how and in what sense the resulting distribution $\mathcal{D}_G$ is close to the true distribution $\mathcal{D}_{real}$.

Using the original objective function (1), then the optimal choice among all the possible functions from $\mathbb{R}^d \to (0,1)$ is $D(x) = \frac{P_{real}(x)}{P_{real}(x)+P_G(x)}$, as shown in (Goodfellow et al., 2014). Here $P_{real}(x)$ is the density of x in the real distribution, and $P_G(x)$ is the density of x in the distribution generated by generator G. Using this discriminator — though it's computationally infeasible to obtain it — one can show that the minimization problem over the generator correspond to minimizing the Jensen-Shannon (JS) divergence between the true distribution $\mathcal{D}_{real}$ and the generative distribution $\mathcal{D}_G$. Recall that for two distributions μ and ν, the JS divergence is defined by $d_{JS}(\mu,\nu) = \frac{1}{2}(KL(\mu\|\frac{\mu+\nu}{2}) + KL(\nu\|\frac{\mu+\nu}{2}))$.

Other measuring functions ϕ and choice of discriminator class leads to different distance function between distribution other than JS divergence. Notably, (Arjovsky et al., 2017) shows that when $\phi(t) = t$, and the discriminator is chosen among all 1-Lipschitz functions, maxing out the discriminator, the generator is attempting to minimize the Wasserstein distance between $\mathcal{D}_{real}$ and $D_u(h)$. Recall that Wasserstein distance between μ and ν is defined as $d_W(\mu,\nu) = \sup_D$ is 1-Lipschitz $|\mathbb{E}_{x\sim\mu}[D(x)] - \mathbb{E}_{x\sim\nu}[D(x)]|$.

3. Generalization Theory for GANs

The above interpretation of GANs in terms of minimizing distance (such as JS divergence and Wasserstein distance) between the real distribution and the generated distribution relies on two crucial assumptions: (i) very expressive class of discriminators such as the set of all bounded discriminator or the set of all 1-Lipschitz discriminators, and (ii) very large number of examples to compute/estimate the objective (1) or (2). Neither assumption holds in practice, and we will show next that this greatly affects the generalization ability, a notion we introduce in Section 3.1.

3.1. Definition of Generalization

Our definition is motivated from supervised classification, where training is said to generalize if the training and test error closely track each other. (Since the purpose of GANs training is to learn a distribution, one could also consider a stronger definition of successful training, as discussed in Section 3.4.)

Let $x_1, \ldots, x_m$ be the training examples, and let $\hat{\mathcal{D}}_{real}$ denote the uniform distribution over $x_1, \ldots, x_m$. Similarly, let $G_u(h_1), \ldots, G_u(h_r)$ be a set of r examples from the generated distribution $\mathcal{D}_G$. In the training of GANs, one implicitly uses $\mathbb{E}_{x\sim\hat{\mathcal{D}}_{real}}[\phi(D_v(x))]$ to approximate the quantity $\mathbb{E}_{x\sim\mathcal{D}_{real}}[\phi(D_v(x))]$. Inspired by the observation that the training objective of GANs and its variants is to minimize some distance (or divergence) $d(\cdot,\cdot)$ between $\mathcal{D}_{real}$ and $\mathcal{D}_G$ using finite samples, we define the generalization of GANs as follows:

Definition 1. A divergence or distance $d(\cdot,\cdot)$ between distributions *generalizes* with m training examples and error ε if for the learnt distribution $\mathcal{D}_G$, the following holds with high probability,

$$\left| d(\mathcal{D}_{real}, \mathcal{D}_G) - d(\hat{\mathcal{D}}_{real}, \hat{\mathcal{D}}_G) \right| \le \varepsilon \quad (3)$$

where $\hat{\mathcal{D}}_{real}$ is an empirical version of the true distribution (with m samples) and $\hat{\mathcal{D}}_G$ is an empirical version of the generated distribution with polynomial number of samples.

In words, generalization in GANs means that the population distance between the true and generated distribution is close to the empirical distance between the empirical distributions. Our target is to make the former distance small,

whereas the latter one is what we can access and minimize in practice. The definition allows only polynomial number of samples from the generated distribution because the training algorithm should run in polynomial time.

3.2. JS Divergence and Wasserstein don't Generalize

As a warm-up, we show that JS divergence and Wasserstein distance don't generalize with any polynomial number of examples because the population distance (divergence) is not reflected by the empirical distance.

Lemma 1. *Let μ be uniform Gaussian distributions $\mathcal{N}(0, \frac{1}{d}I)$ and $\hat{\mu}$ be an empirical versions of μ with m examples. Then we have $d_{JS}(\mu, \hat{\mu}) = \log 2$, $d_W(\mu, \hat{\mu}) \geq 1.1$.*

There are two consequences of Lemma 1. First, consider the situation where $\mathcal{D}_{real} = \mathcal{D}_G = \mu$. Then we have that $d_W(\mathcal{D}_{real}, \mathcal{D}_G) = 0$ but $d_W(\hat{\mathcal{D}}_{real}, \hat{\mathcal{D}}_G) > 1$ as long as we have polynomial number of examples. This violates the generalization definition equation (3).

Second, consider the case $\mathcal{D}_{real} = \mu$ and $\mathcal{D}_G = \hat{\mathcal{D}}_{real} = \hat{\mu}$, that is, $\mathcal{D}_G$ memorizes all of the training examples in $\hat{\mathcal{D}}_{real}$. In this case, since $\mathcal{D}_G$ is a discrete distribution with finite supports, with enough (polynomial) examples, in $\hat{\mathcal{D}}_G$, effectively we also have that $\hat{\mathcal{D}}_G \approx \mathcal{D}_G$. Therefore, we have that $d_W(\hat{\mathcal{D}}_{real}, \hat{\mathcal{D}}_G) \approx 0$ whereas $d_W(\mathcal{D}_{real}, \mathcal{D}_G) > 1$. In other words, with any polynomial number of examples, it's possible to overfit to the training examples using Wasserstein distance. The same argument also applies to JS divergence. See supplementary material for the formal proof.

Notice, this result does not contradict the experiments of (Arjovsky et al., 2017) since they actually use not Wasserstein distance but a surrogate distance that does generalize, as we show next.

3.3. Generalization Bounds for Neural Net Distance

Which distance measure between $\mathcal{D}_{real}$ and $\mathcal{D}_G$ is the GAN objective actually minimizing and can we analyze its generalization performance? Towards answering these questions in full generality (given multiple GANs objectives) we consider the following general distance measure that unifies JS divergence, Wasserstein distance, and the neural net distance that we define later in this section.

Definition 2 ($\mathcal{F}$-distance)**.** Let $\mathcal{F}$ be a class of functions from $\mathbb{R}^d$ to $[0, 1]$ and ϕ be a concave measuring function. Then the *$\mathcal{F}$-divergence with respect to ϕ between two distributions* μ and ν supported on $\mathbb{R}^d$ is defined as

$$d_{\mathcal{F},\phi}(\mu, \nu) = \sup_{D \in \mathcal{F}} \left| \mathop{\mathbb{E}}_{x \sim \mu}[\phi(D(x))] + \mathop{\mathbb{E}}_{x \sim \nu}[\phi(1 - D(x))] \right| - 2\phi(1/2)$$

When $\phi(t) = t$, we have that $d_{\mathcal{F},\phi}$ is a distance function [4] , and with slightly abuse of notation we write it simply as

$$d_{\mathcal{F}}(\mu, \nu) = \sup_{D \in \mathcal{F}} \left| \mathop{\mathbb{E}}_{x \sim \mu}[D(x)] - \mathop{\mathbb{E}}_{x \sim \nu}[D(x)] \right|.$$

Example 1. When $\phi(t) = \log(t)$ and $\mathcal{F} = \{\text{all functions from } \mathbb{R}^d \text{ to } [0, 1]\}$, we have that $d_{\mathcal{F},\phi}$ is the same as JS divergence. When $\phi(t) = t$ and $\mathcal{F} = \{\text{all 1-Lipschitz functions from } \mathbb{R}^d \text{ to } [0, 1]\}$, then $d_{\mathcal{F},\phi}$ is the Wasserstein distance.

Example 2. Suppose $\mathcal{F}$ is a set of neural networks and $\phi(t) = \log t$, then original GAN objective function is equivalent to $\min_G d_{\mathcal{F},\phi}(\hat{\mathcal{D}}_{real}, \hat{\mathcal{D}}_G)$.

Suppose $\mathcal{F}$ is the set of neural networks, and $\phi(t) = t$, then the objective function used empirically in (Arjovsky et al., 2017) is equivalent to $\min_G d_{\mathcal{F}}(\hat{\mathcal{D}}_{real}, \hat{\mathcal{D}}_G)$.

GANs training uses $\mathcal{F}$ to be a class of neural nets with a bound p on the number of parameters. We then informally refer to $d_{\mathcal{F}}$ as the neural net distance. The next theorem establishes generalization in the sense of equation (3) does hold for it (with a uniform convergence) . We assume that the measuring function takes values in $[-\Delta, \Delta]$ and that it is L_ϕ-Lipschitz. Further, $\mathcal{F} = \{D_v, v \in \mathcal{V}\}$ is the class of discriminators that is L-Lipschitz with respect to the parameters v. As usual, we use p to denote the number of parameters in v.

Theorem 3.1. *In the setting of previous paragraph, let μ, ν be two distributions and $\hat{\mu}, \hat{\nu}$ be empirical versions with at least m samples each. There is a universal constant c such that when $m \geq \frac{cp\Delta^2 \log(LL_\phi p/\epsilon)}{\epsilon^2}$, we have with probability at least $1 - \exp(-p)$ over the randomness of $\hat{\mu}$ and $\hat{\nu}$,*

$$|d_{\mathcal{F},\phi}(\hat{\mu}, \hat{\nu}) - d_{\mathcal{F},\phi}(\mu, \nu)| \leq \epsilon.$$

See supplementary material for the proof. The intuition is that there aren't too many distinct discriminators, and thus given enough samples the expectation over the empirical distribution converges to the expectation over the true distribution for *all* discriminators.

Theorem 3.1 shows that the neural network divergence (and neural network distance) has a much better generalization properties than Jensen-Shannon divergence or Wasserstein distance. If the GAN successfully minimized the neural network divergence between the empirical distributions, that is, $d(\hat{\mathcal{D}}_{real}, \hat{\mathcal{D}}_G)$, then we know the neural network divergence $d(\mathcal{D}_{real}, \mathcal{D}_G)$ between the distributions $\mathcal{D}_{real}$ and $\mathcal{D}_G$ is also small. It is possible to change the proof to also show that this generalization continues to hold at every iteration of the training. See supplementary material.

[4] Technically it is a pseudometric. This is also known as integral probability metrics(Müller, 1997).

3.4. Generalization vs Diversity

Since the final goal of GANs training is to learn a distribution, it is worth understanding that though weak generalization in the sense of Section 3.3 is guaranteed, it comes with a cost (albeit a necessary one). For JS divergence and Wasserstein distance, when the distance between two distributions μ, ν is small, it is safe to conclude that the distributions μ and ν are almost the same. However, the neural net distance $d_{NN}(\mu, \nu)$ can be small even if μ, ν are not very close. As a simple Corollary of Lemma 3.1, we obtain:

Corollary 3.1 (Low-capacity discriminators cannot detect lack of diversity)**.** *Let $\hat{\mu}$ be the empirical version of distribution μ with m samples. There is a some universal constant c such that when $m \leq \frac{cp\Delta^2 \log(LL_\phi p/\epsilon)}{\epsilon^2}$, we have that with probability at least $1 - \exp(-p)$, $d_{\mathcal{F},\phi}(\mu, \hat{\mu}) \leq \epsilon$.*

That is, the neural network distance for nets with p parameters cannot distinguish between a distribution μ and a distribution with support $\tilde{O}(p/\epsilon^2)$. In fact the proof still works if the disriminator is allowed to take many more samples from μ; the reason they don't help is that its capacity is limited to p.

4. Expressive Power and Equilibrium

Section 3 clarified the notion of generalization for GANs: namely, neural-net divergence between the generated distribution $\mathcal{D}$ and $\mathcal{D}_{real}$ on the empirical samples closely tracks the divergence on the full distribution (i.e., unseen samples). But this doesn't explain why in practice the generator usually "wins" so that the discriminator is unable to do much better than random guessing at the end. In other words, was it sheer luck that so many real-life distributions $\mathcal{D}_{real}$ turned out to be close in neural-net distance to a distribution produced by a fairly compact neural net? This section suggests no luck may be needed.

The explanation starts with a thought experiment. Imagine allowing a much more powerful generator, namely, an infinite mixture of deep nets, each of size p. So long as the deep net class is capable of generating simple gaussians, such mixtures are quite powerful, since a classical result says that an infinite mixtures of simple gaussians can closely approximate $\mathcal{D}_{real}$. Thus an infinite mixture of deep net generators will "win" the GAN game, not only against a discriminator that is a small deep net but also against more powerful discriminators (e.g., any Lipschitz function).

The next stage in the thought experiment is to imagine a much less powerful generator, which is a mix of only a few deep nets, not infinitely many. Simple counterexamples show that now the distribution $\mathcal{D}$ will not closely approximate arbitrary $\mathcal{D}_{real}$ with respect to natural metrics like ℓ_p. Nevertheless, could the generator still win the GAN game against a deep net of bounded capacity (i.e., the deep net is unable to distinguish $\mathcal{D}$ and $\mathcal{D}_{real}$)? We show it can.

INFORMAL THEOREM: *If the discriminator is a deep net with p parameters, then a mixture of $\tilde{O}(p \log(p/\epsilon)/\epsilon^2)$ generator nets can produce a distribution $\mathcal{D}$ that the discriminator will be unable to distinguish from $\mathcal{D}_{real}$ with probability more than ϵ. (Here $\tilde{O}(\cdot)$ notation hides some nuisance factors.)*

This informal theorem is also a component of our result below about the existence of an approximate pure equilibrium. With current technique this existence result seems sensitive to the measuring function ϕ, and works for $\phi(x) = x$ (i.e., Wasserstein GAN). For other ϕ we only show existence of mixed equilibria with small mixtures.

4.1. General ϕ: Mixed Equilibrium

For general measuring function ϕ we can only show the existence of a mixed equilibrium, where we allow the discriminator and generator to be finite mixtures of deep nets.

For a class of generators $\{G_u, u \in \mathcal{U}\}$ and a class of discriminators $\{D_v, v \in \mathcal{V}\}$, we can define the payoff $F(u, v)$ of the game between generator and discriminator

$$F(u, v) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}_{real}}[\phi(D_v(x))] + \mathop{\mathbb{E}}_{x \sim \mathcal{D}_G}[\phi(1 - D_v(x))]. \quad (4)$$

Of course as we discussed in the previous section, in practice these expectations should be with respect to the empirical distributions. Our discussions in this section does not depend on the distributions $\mathcal{D}_{real}$ and $\mathcal{D}_h$, so we define $F(u, v)$ this way for simplicity.

The well-known min-max theorem (v. Neumann, 1928) in game theory shows if both players are allowed to play *mixed strategies* then the game has a min-max solution. A mixed strategy for the generator is just a distribution $\mathcal{D}_u$ supported on $\mathcal{U}$, and one for discriminator is a distribution $\mathcal{D}_v$ supported on $\mathcal{V}$.

Theorem 4.1 (vonNeumann)**.** *There exists value V, and a pair of mixed strategies ($\mathcal{S}_u$,S_v) s.t. $\forall v, \quad \mathbb{E}_{u \sim \mathcal{S}_u}[F(u, v)] \leq V$, $\forall u, \quad \mathbb{E}_{v \sim \mathcal{S}_v}[F(u, v)] \geq V$.*

Note that this equilibrium involves both parties announcing their strategies $\mathcal{S}_u, \mathcal{S}_v$ at the start, such that neither will have any incentive to change their strategy after studying the opponent's strategy. The payoff is generated by the generator first sample $u \sim \mathcal{S}_u, h \sim \mathcal{D}_h$, and then generate an example $x = G_u(h)$. Therefore, the mixed generator is just a linear mixture of generators. The discriminator will first sample $v \sim \mathcal{S}_v$, and then output $D_v(x)$. Note that in general this is very different from a discriminator D that outputs $\mathbb{E}_{v \sim \mathcal{S}_v}[D_v(x)]$, because the measuring function ϕ is in

general nonlinear. In particular, the correct payoff function for a mixture of discriminator is:

$$\mathop{\mathbb{E}}_{v\sim\mathcal{S}_v}[F(u,v)] = \mathop{\mathbb{E}}_{\substack{x\sim\mathcal{D}_{real}\\ v\sim\mathcal{S}_v}}[\phi(D_v(x))] + \mathop{\mathbb{E}}_{\substack{h\sim\mathcal{D}_h\\ v\sim\mathcal{S}_v}}[\phi(1-D_v(G_u(h)))].$$

Of course, this equilibrium involving an infinite mixture makes little sense in practice. We show that (as is folklore in game theory (Lipton & Young, 1994)) that we can *approximate* this min-max solution with mixture of finitely many generators and discriminators. More precisely we define ϵ-approximate equilibrium:

Definition 3. A pair of mixed strategies $(\mathcal{S}_u, \mathcal{S}_v)$ is an ϵ-approximate equilibrium, if for some value V $\forall v \in \mathcal{V}$, $\mathbb{E}_{u\sim\mathcal{S}_u}[F(u,v)] \leq V+\epsilon$; $\forall u \in \mathcal{U}$, $\mathbb{E}_{v\sim\mathcal{S}_v}[F(u,v)] \geq V-\epsilon$. If the strategies $\mathcal{S}_u, \mathcal{S}_v$ are pure strategies, then this pair is called an ϵ-approximate pure equilibrium.

Suppose ϕ is L_ϕ-Lipschitz and bounded in $[-\Delta, \Delta]$, the generator and discriminators are L-Lipschitz with respect to the parameters and L'-Lipschitz with respect to inputs, in this setting we can formalize the above Informal Theorem as follows:

Theorem 4.2. *In the settings above, there is a universal constant $C > 0$ such that for any ϵ, there exists $T = \frac{C\Delta^2 p\log(LL'L_\phi\cdot p/\epsilon)}{\epsilon^2}$ generators $G_{u_1},\ldots,G_{u_T}$ and T discriminators $D_{v_1},\ldots,D_{v_T}$, let $\mathcal{S}_u$ be a uniform distribution on u_i and $\mathcal{S}_v$ be a uniform distribution on v_i, then $(\mathcal{S}_u, \mathcal{S}_v)$ is an ϵ-approximate equilibrium. Furthermore, in this equilibrium the generator "wins," meaning discriminators cannot do better than random guessing.*

The proof uses a standard probabilistic argument and epsilon net argument to show that if we sample T generators and discriminators from infinite mixture, they form an approximate equilibrium with high probability. For the second part, we use the fact that every distribution can be approximated by infinite mixture of Gaussians, so the generator must be able to approximate the real distribution $\mathcal{D}_{real}$ and win. Therefore indeed a mixture of $\tilde{O}(p)$ generators can achieve an ϵ-approximate equilibrium. See supplementary material for details.

In the special case of $\phi(x) = x$ (Wasserstein GAN), we show that a mixture of generator/discriminator is equivalent to a specially designed, larger generator/discriminator, therefore an approximate pure equilibrium exists. See supplementary material for more details.

Theorem 4.3. *Suppose the generator and discriminator are both k-layer neural networks ($k \geq 2$) with p parameters, and the last layer uses ReLU activation function. In the setting of Theorem 4.2, when $\phi(x) = x$ there exists $k+1$-layer neural networks of generators G and discriminator D with $O\left(\frac{\Delta^2 p^2\log(LL'L_\phi\cdot p/\epsilon)}{\epsilon^2}\right)$ parameters, such that there exists an ϵ-approximate pure equilibrium. Furthermore, if the generator is capable of generating a Gaussian then the value $V = 1$.*

5. MIX+GANs

Theorem 4.2 show that using a mixture of (not too many) generators and discriminators guarantees existence of approximate equilibrium. This suggests that using a mixture may lead to more stable training.

Of course, it is impractical to use very large mixtures, so we propose MIX+GAN: use a mixture of T components, where T is as large as allowed by size of GPU memory (usually $T \leq 5$). Namely, train a mixture of T generators $\{G_{u_i}, i\in[T]\}$ and T discriminators $\{D_{v_i}, i\in[T]\}$) which share the same network architecture but have their own trainable parameters. Maintaining a mixture means of course maintaining a weight w_{u_i} for the generator G_{u_i} which corresponds to the probability of selecting the output of G_{u_i}. These weights are also updated via backpropagation. This heuristic can be combined with existing methods like DCGAN (Radford et al., 2016), WASSERSTEIN-GAN (Arjovsky et al., 2017) etc., giving us new training methods MIX+DCGAN, MIX+WASSERSTEINGAN etc.

We use exponentiated gradient (Kivinen & Warmuth, 1997): store the log-probabilities $\{\alpha_{u_i}, i\in[T]\}$, and then obtain the weights by applying soft-max function on them: $w_{u_i} = \frac{e^{\alpha_{u_i}}}{\sum_{k=1}^T e^{\alpha_{u_k}}}$, $i\in[T]$.

Note that our algorithm is maintaining weights on different generators and discriminators. This is very different from the idea of *boosting* where weights are maintained on samples. ADAGAN (Tolstikhin et al., 2017) uses ideas similar to boosting and maintains weights on training examples.

Given payoff function F, training MIX+GAN boils down to optimizing:

$$\min_{\{u_i\},\{\alpha_{u_i}\}}\max_{\{v_j\},\{\alpha_{v_j}\}}\mathop{\mathbb{E}}_{i,j\in[T]}F(u_i,v_j) = \min_{\{u_i\},\{\alpha_{u_i}\}}\max_{\{v_j\},\{\alpha_{v_j}\}}\sum_{i,j\in[T]}w_{u_i}w_{v_j}F(u_i,v_j).$$

Here the payoff function is the same as Equation (4). We use both measuring functions $\phi(x) = \log x$ (for original GAN) and $\phi(x) = x$ (for WASSERSTEINGAN). In our experiments we alternatively update generators' and discriminators' parameters as well as their corresponding log-probabilities using ADAM (Kingma & Ba, 2015), with learning rate $lr = 0.0001$.

Empirically, it is observed that some components of the mixture tend to collapse and their weights diminish during

the training. To encourage full use of the mixture capacity, we add to the training objective an entropy regularizer that discourages the weights being far away from uniform: $R_{ent}(\{w_{u_i}\},\{w_{v_i}\}) = -\frac{1}{T}\sum_{i=1}^{T}(\log(w_{u_i}) + \log(w_{v_i}))$.

6. Experiments

In this section, we first explore the qualitative benefits of our method on image generation tasks: MNIST dataset (LeCun et al., 1998) of hand-written digits and the CelebA (Liu et al., 2015) dataset of human faces. Then for more quantitative evaluation we use the CIFAR-10 dataset (Krizhevsky & Hinton, 2009) and use the *Inception Score* introduced in (Salimans et al., 2016). MNIST contains 60,000 labeled 28×28-sized images of hand-written digits, CelebA contains over 200K 108×108-sized images of human faces (we crop the center 64×64 pixels for our experiments), and CIFAR-10 has 60,000 labeled 32×32-sized RGB natural images which fall into 10 categories.

To reinforce the point that this technique works out of the box, no extensive hyper-parameter search or tuning is necessary. Please refer to our code for experimental setup. [5]

6.1. Qualitative Results

The DCGAN architecture (Radford et al., 2016) uses deep convolutional nets as generators and discriminators. We trained MIX+DCGAN on MNIST and CelebA using the authors' code as a black box, and compared visual qualities of generated images to those by DCGAN.

Results on MNIST is shown in Figure 2. In this experiment, the baseline DCGAN consists of a pair of a generator and a discriminator, which are 5-layer deconvoluitonal neural networks, and are conditioned on image labels. Our MIX+DCGAN model consists of a mixture of such DCGANs so that it has 3 generators and 3 discriminators. We observe that our method produces somewhat cleaner digits than the baseline (note the fuzziness in the latter).

Results on CelebA dataset are also in Figure 2, using the same architecture as for MNIST, except the models are not conditioned on image labels anymore. Again, our method generates more faithful and more diverse samples than the baseline. Note that one may need to zoom in to fully perceive the difference, since both the two datasets are rather easy for DCGAN.

6.2. Quantitative Results

Now we turn to quantitative measurement using Inception Score. Our method is applied to DCGAN and WASSERSTEINGAN (Arjovsky et al., 2017), and throughout, mixtures of 5 generators and 5 discriminators are used. At first sight the comparison DCGAN v.s. MIX+DCGAN seems unfair because the latter uses 5 times the capacity of the former, with corresponding penalty in running time per epoch. To address this, we also compare our method with larger versions of DCGAN with roughly the same number of parameters, and we found the former is consistently better than the later, as detailed below.

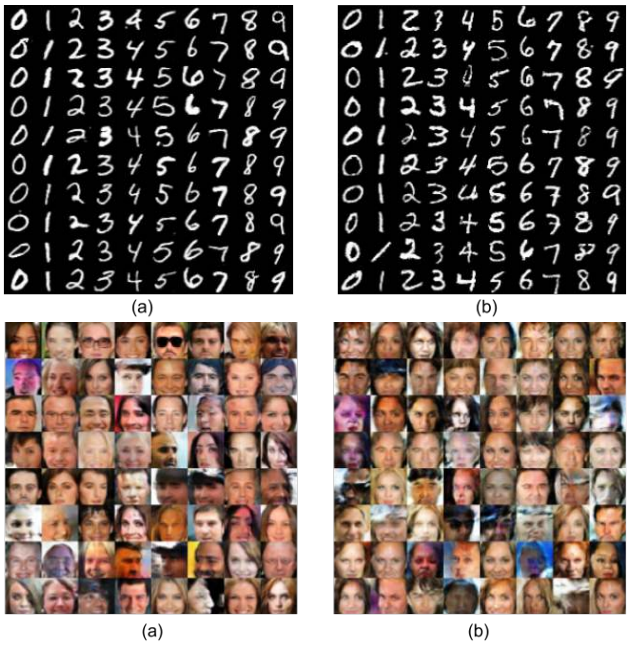

Figure 2. **MNIST and CelebA Samples**. Digits and Faces generated from (a) MIX+DCGAN. (b) DCGAN.

To construct MIX+DCGAN, we build on top of the DCGAN trained with losses proposed by Huang et al. (2017), which is the best variant so far without improved training techniques. The same hyper-parameters are used for fair comparison. See (Huang et al., 2017) for more details. Similarly, for the MIX+WASSERSTEINGAN, the base GAN is identical to that proposed by Arjovsky et al. (2017) using their hyper-parameter scheme.

For a quantitative comparison, inception score is calculated for each model, using 50,000 freshly generated samples that are not used in training. To sample a single image from our MIX+ models, we first select a generator from the mixture according to their assigned weights $\{w_{u_i}\}$, and then draw a sample from the selected generator.

Table 1 shows the results on the CIFAR-10 dataset. We find that, simply by applying our method to the baseline models, our MIX+ models achieve 7.72 v.s. 7.16 on DCGAN, and 4.04 v.s. 3.82 on WASSERSTEINGAN. To confirm that the superiority of MIX+ models is not solely due to more parameters, we also tested a DCGAN model with 5 times many parameters (roughly the same number of parameters

[5] Related code is public online at https://github.com/PrincetonML/MIX-plus-GANs.git

Table 1. **Inception Scores on CIFAR-10.** Mixture of DCGANs achieves higher score than any single-component DCGAN does. All models except for WASSERSTEINGAN variants are trained with labels.

Method	Score
SteinGAN (Wang & Liu, 2016)	6.35
Improved GAN (Salimans et al., 2016)	8.09±0.07
AC-GAN (Odena et al., 2016)	8.25 ± 0.07
S-GAN (best variant in (Huang et al., 2017))	8.59± 0.12
DCGAN (as reported in (Wang & Liu, 2016))	7.37
DCGAN (best variant in (Huang et al., 2017))	7.16±0.10
DCGAN (5x size)	7.34±0.07
MIX+DCGAN (with 5 components)	7.72±0.09
WASSERSTEINGAN	3.82±0.06
MIX+WASSERSTEINGAN (with 5 components)	4.04±0.07
Real data	11.24±0.12

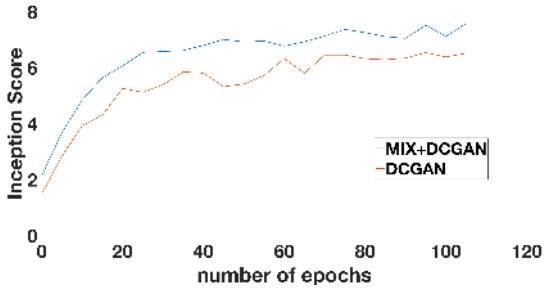

Figure 3. **Training Curve of MIX+DCGAN v.s. DCGAN (Inception Score)**. MIX+DCGAN is consistently higher than DCGAN.

as a 5-component MIX+DCGAN), which is tuned using a grid search over 27 sets of hyper-parameters (learning rates, dropout rates, and regularization weights). It gets only 7.34 (labeled as "5x size" in Table 1), which is lower than that of a MIX+DCGAN. It is unclear how to apply MIX+ to S-GANs. We tried mixtures of the upper and bottom generators separately, resulting in worse scores somehow. We leave that for future exploration.

Figure 3 shows how Inception Scores evolve during training. MIX+DCGAN outperforms DCGAN throughout the entire training process, showing that it makes effective use of the capacity.

Arjovsky et al. (2017) shows that (approximated) Wasserstein loss, which is the neural network divergence by our definition, is meaningful because it correlates well with visual quality of generated samples. Figure 4 shows the training dynamics of neural network divergence of MIX+WASSERSTEINGAN v.s. WASSERSTEINGAN, which clearly indicates our method is capable of achieving a much lower divergence as well as of improving the visual quality of generated samples.

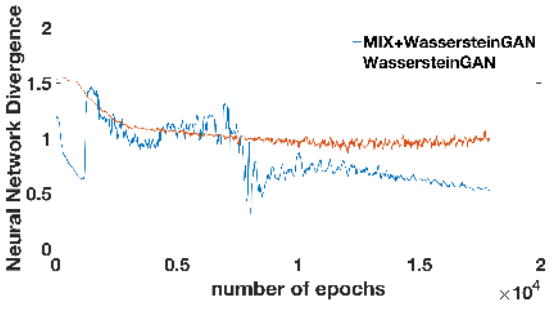

Figure 4. **Training Curve of MIX+WASSERSTEINGAN v.s. WASSERSTEINGAN (Wasserstein Objective)**. MIX+WASSERSTEINGAN is better towards the end but loss drops less smoothly, which needs further investigation.

7. Conclusions

The notion of generalization for GANs has been clarified by introducing a new notion of distance between distributions, the neural net distance. (Whereas popular distances such as Wasserstein and JS may not generalize.) Assuming the visual cortex also is a deep net (or some network of moderate capacity) generalization with respect to this metric is in principle sufficient to make the final samples look realistic to humans, even if the GAN doesn't actually learn the true distribution.

One issue raised by our analysis is that the current GANs objectives cannot even enforce that the synthetic distribution has high diversity (Section 3.4). Furthermore this cannot be fixed by simply providing the discriminator with more training examples. Possibly some other change to the GANs setup are needed.

The paper also made progress another unexplained issue about GANs, by showing that a pure approximate equilibrium exists for a certain natural training objective (Wasserstein) and in which the generator wins the game. No assumption about the target distribution $\mathcal{D}_{real}$ is needed.

Suspecting that a pure equilibrium may not exist for all objectives, we recommend in practice our MIX+GAN protocol using a small mixture of discriminators and generators. Our experiments show it improves the quality of several existing GAN training methods.

Finally, existence of an equilibrium does not imply that a simple algorithm (in this case, backpropagation) would find it easily. Understanding convergence remains wide open.

Acknowledgements

This paper was done in part while the authors were hosted by Simons Institute. We thank Moritz Hardt, Kunal Talwar, Luca Trevisan, and the referees for useful comments. This research was supported by NSF, Office of Naval Research, and the Simons Foundation.

References

Abadi, Martín and Andersen, David G. Learning to protect communications with adversarial neural cryptography. *arXiv preprint arXiv:1610.06918*, 2016.

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Durugkar, I., Gemp, I., and Mahadevan, S. Generative Multi-Adversarial Networks. *ArXiv e-prints*, November 2016.

Ghosh, Jayanta K, Ghosh, RVJK, and Ramamoorthi, RV. Bayesian nonparametrics. Technical report, 2003.

Goodfellow, Ian. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Huang, Xun, Li, Yixuan, Poursaeed, Omid, Hopcroft, John, and Belongie, Serge. Stacked generative adversarial networks. In *Computer Vision and Patter Recognition*, 2017.

Jiwoong Im, D., Ma, H., Dongjoo Kim, C., and Taylor, G. Generative Adversarial Parallelization. *ArXiv e-prints*, December 2016.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kivinen, Jyrki and Warmuth, Manfred K. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. Technical report, 2009.

LeCun, Yann, Cortes, Corinna, and Burges, Christopher JC. The mnist database of handwritten digits, 1998.

Lipton, Richard J and Young, Neal E. Simple strategies for large zero-sum games with applications to complexity theory. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pp. 734–740. ACM, 1994.

Liu, Ziwei, Luo, Ping, Wang, Xiaogang, and Tang, Xiaoou. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738, 2015.

Müller, Alfred. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(02):429–443, 1997.

Odena, Augustus, Olah, Christopher, and Shlens, Jonathon. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

Tolstikhin, Ilya, Gelly, Sylvain, Bousquet, Olivier, Simon-Gabriel, Carl-Johann, and Schölkopf, Bernhard. Adagan: Boosting generative models. *arXiv preprint arXiv:1701.02386*, 2017.

v. Neumann, J. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

Wang, Dilin and Liu, Qiang. Learning to draw samples: With application to amortized mle for generative adversarial learning. Technical report, 2016.