Generalization and Equilibrium in Generative Adversarial Nets (GANs)

Sanjeev Arora* Rong Ge † Yingyu Liang‡ Tengyu Ma§ Yi Zhang¶

Abstract

This paper makes progress on several open theoretical issues related to Generative Adversarial Networks. A definition is provided for what it means for the training to *generalize*, and it is shown that generalization is not guaranteed for the popular distances between distributions such as Jensen-Shannon or Wasserstein distance. We introduce a new metric called *neural net distance* for which generalization does occur. We also show that an approximate pure equilibrium in the 2-player game exists for a natural training objective (Wasserstein). Showing such a result has been an open problem (for any training objective).

Finally, the above theoretical ideas lead us to propose a new training protocol, MIX+GAN, which can be combined with any existing method. We present experiments showing that it stabilizes and improves some existing methods.

1 Introduction

Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] have become the dominant method for fitting generative models to complicated real-life data, and even found unusual uses such as designing good cryptographic primitives [Abadi and 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. But many basic issues remain unresolved, as we now discuss.

Let's recall that the basic scenario is that we wish to train a *generator* deep net whose input is a standard Gaussian, and whose output is a sample from some distribution $\mathcal{D}$ on $\Re^d$. We wish the samples from $\mathcal{D}$ to closely resemble those drawn from some real-life distribution $\mathcal{D}_{real}$ (which could be, say, real-life images represented using raw pixels). Towards this end, a *discriminator* deep net is trained alongside the generator net, and it is trained 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 $\mathcal{D}$. This basic iterative framework has been tried with many training objectives; see Section 2. Recently Arjovsky et al. [2017] proposed another variant called *Wasserstein GAN* that appears to lead to more stable training. The unresolved issues stem

*Princeton University, Computer Science Department, email: arora@cs.princeton.edu
†Duke University, Computer Science Department, email: rongge@cs.duke.edu
‡Princeton University, Computer Science Department, email: yingyul@cs.princeton.edu
§Princeton University, Computer Science Department, email: tengyu@cs.princeton.edu
¶Princeton University, Computer Science Department, email: yz7@cs.princeton.edu

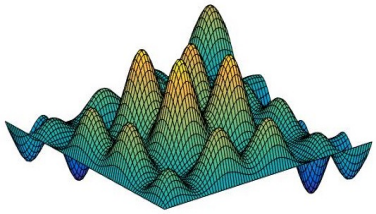

Figure 1: Probability density $\mathcal{D}_{real}$ with many peaks and valleys

from the fact that distribution $\mathcal{D}_{real}$ is complicated. The number of peaks and valleys in it may be large and possibly even 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.) In practice, the generator often "wins" the game by the end, meaning that the discriminator can do no better than random guessing when deciding whether or not a particular sample came from $\mathcal{D}$ or $\mathcal{D}_{real}$.

But it is unclear what to conclude from the generator's win: is $\mathcal{D}$ close to $\mathcal{D}_{real}$ in some metric? The fear of course is *overfitting*, intuitively described as the generator having simply memorized the presented samples (or parts theoreof). A glance at Figure 1 reinforces such fears. The number of samples from $\mathcal{D}_{real}$ (and from $\mathcal{D}$ for that matter) used in the entire training is moderately large, but still a lot fewer than the potential number of peaks and valleys in $\mathcal{D}_{real}$. Thus the samples from $\mathcal{D}_{real}$ may be a poor substitute for the full distribution, and there is no a priori reason that using them to train a new distribution will lead to a result $\mathcal{D}$ that meaningfully approximates $\mathcal{D}_{real}$ in unsampled peaks and valleys. Understanding *generalization* is open.

Finally, we don't know if an equilibrium exists. 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 absence of an equilibrium could cause some of the instability observed in training. Arjovsky et al. [2017] suggest that empirically, using their Wasserstein objective reduces instability, but we still lack proof of existence of an equilibrium. Game theory doesn't help 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 This paper

We formally define generalization in Section 3 and show that for previously studied notions of distance between distributions, generalization is not guaranteed (Theorem 3.2). The reason is that these formalisms involve an ideal discriminator, which can have exponentially high VC dimension (or any other suitable measure of complexity). Therefore we introduce a new metric on distributions, the *neural net distance*, for which we show that generalization does happen.

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

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

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 the objective changes only a little whether player 1 goes first, or player 2 does.) This is the first such proof we know of.

The 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 introduce 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 (reported in Section 6) show that for several previous techniques, MIX+GAN stabilizes the training, and in some cases improves the performance.

2 Preliminaries

Notations: Throughout the paper we use d for the dimension of samples. Both generators and discriminators will be neural networks with n parameters. In Section 3 we use m for number of samples. When we say with high probability we mean the probability is larger than $1 - d^{-\omega(1)}$.
Generators and Discriminators: For GANs, the generator and discriminators are multi-layer neural networks. We abstract the set of possible networks as a class of functions: $\{F_u, u \in \mathcal{U}\}$, where $u \in \mathcal{U}$ is a high-dimensional vector that specifies the trainable parameters: weights, biases etc. (This assumes that hyperparameters —number of layers, number of nodes, convolution structure etc.— have been pre-fixed, as is normally the case.) Without loss of generality assume $\mathcal{U}$ is a subset of the d dimensional unit ball[2].

Throughout the paper, $\{G_u, u \in \mathcal{U}\}$ denotes the class of generators and $\{D_v, v \in \mathcal{V}\}$ denotes the class of discriminators. Thus G_u is a function mapping a simple random input $h \sim \mathcal{D}_h$ (e.g., a Gaussian vector) to a sample x. Each D_v is a function that maps a sample x to a value in $[0, 1]$ – usually interpreted as the probability that the sample comes from the real distribution $\mathcal{D}_{real}$. In most neural network architectures, if the parameters (weights, biases) change by a small amount, this makes only a small difference in the output. We capture this phenomena by assuming G_u and D_v are L-Lipschitz with respect to their parameters. Thus for all $u, u' \in \mathcal{U}$ and any input h, we have $\|G_u(h) - G_{u'}(h)\| \leq L\|u - u'\|$. Similarly for all $v, v' \in \mathcal{V}$ and any sample x, we have $|D_v(x) - D_{v'}(x)| \leq L\|v - v'\|$. 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.
GAN objective. The standard GAN training [Goodfellow et al., 2014] consists of training u, v so as to optimize an objective such as:

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

Intuitively, this says that the discriminator D_v should give high values $D_v(x)$ to the real samples and low values $D_v(G(h))$ for the generator's outputs. The log function was suggested because of a nice information-theoretic interpretation described below, but in practice it can cause problems

[2]If the parameters can have norm at most r we can always consider $F_{r \cdot u}$ and then u can have norm at most 1.

since $\log x \to -\infty$ as $x \to 0$. But the objective still makes intuitive sense if we replace log by any monotone function $f : [0, 1] \to \mathbb{R}$, which yields the objective:

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

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

Standard Interpretation via Distance Metric. From the complicated min-max objective it can be difficult to tell in what sense the final distribution $G_u(\mathcal{D}_h)$ is close to $\mathcal{D}_{real}$. Thus researchers have relied upon a heuristic argument that assumes that the discriminator is chosen optimally within some class, which allows interpreting the max as defining a distance metric between $\mathcal{D}_{real}$ and $G_u(h)$. Different measuring functions lead to some popular distance metrics this way.

For the original objective function, if the optimal discriminator is allowed to be any function all (i.e., not just one computable by a finite nueral net) it can be checked that the optimal choice is $D(x) = \frac{P_{real}(x)}{P_{real}(x)+P_G(x)}$. Here $P_{real}(x)$ is the probability (or density) of x in the real distribution, and $P_G(x)$ is the probability (or density) of x in the distribution generated by generator G. Using this discriminator, up to linear transformation the maximum value achieved by discriminator isequivalent to the Jensen-Shannon (JS) divergence between $\mathcal{D}_{real}$ and $G_u(h)$. 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 f leads to different distance metrics. Notably, when $f(x) = x$, and the discriminator is chosen among all 1-Lipschitz functions (which is more realistic than all functions) the max part of the objective is equivalent to Wasserstein distance (up to additive constant)

$$d_W(\mu, \nu) = \sup_{f:1 \text{ Lipschitz}} \mathop{\mathbb{E}}_{x\sim\mu} [f(x)] - \mathop{\mathbb{E}}_{x\sim\nu} [f(x)]. \tag{3}$$

3 Distance Metric and Generalization

This interpretation of the GAN objective in terms of minimizing metrics such as JS divergence or Wasserstein distance is standard but it relies on two crucial assumptions: (i) The network has enough capacity to approximate the ideal discriminator and (ii) We have enough samples to estimate the expectations in Objectives (1) or (2). Neither assumption is satisfied in practice, and we now show this affects generalization ability.

The training actually uses only finite set of samples from $\mathcal{D}_{real}$ and $G_u(\mathcal{D}_h)$. Let $h_1, h_2, ..., h_m \sim \mathcal{D}_h$ be the inputs of the generator, and let $x_1, x_2, ..., x_m \sim \mathcal{D}_{real}$ be samples from the real distribution[3]. We use $\hat{\mathcal{D}}_{real}$ and $G_u(\hat{\mathcal{D}}_h)$ to denote the empirical distribution that assigns probability $1/m$ to each sample.

[3]Of course, the training allows using different number of samples from the two distributions, but for notational ease use m for both.

Thus the empirical objective function actually being optimized is

$$\min_{u\in\mathcal{U}}\max_{v\in\mathcal{V}}\frac{1}{m}\sum_{i=1}^{m}[f(D_v(x_i))]+\frac{1}{m}\sum_{i=1}^{m}[f(1-D_v(G_u(h_i)))]. \tag{4}$$

Thus even for the perfect discriminator, the training can only measure its effectiveness on distinguishing between these two empirical distributions. When $f(x)=\log x$ we can get $d_{JS}(\hat{\mathcal{D}}_{real},G_u(\hat{\mathcal{D}}_h))$; when $f(x)=x$ and discriminator is 1-Lipschitz, we get $d_W(\hat{\mathcal{D}}_{real},G_u(\hat{\mathcal{D}}_h))$.

One of the messages of our analysis will be that disciminators with bounded capacity perform very differently than the ideal discriminators analysed in Section 2. In fact, we will see (Corollary 3.2) that a discriminator of capacity n cannot distinguish too well between $\mathcal{D}_{real}$ and the empirical distribution $\hat{\mathcal{D}}_{real}$ when the number of samples m exceeds $(n\log n)/\epsilon^2$. Furthermore, this continues to hold no matter now many samples it draws from $\mathcal{D}_{real}$, so this is not a question of "insufficient training samples,"or the usual worry of "overfitting."One way to view this result is that a bounded capacity discriminator is unable to force the generator to produce a distribution with very high diversity. We note that similar results have been shown before in study of pseudorandomness [Trevisan et al., 2009] and model criticism [Gretton et al., 2012].

3.1 Warm up

Intuitively, overfitting for GANs means that the generator will memorize the training examples instead of generating the real distribution. We first show that if the GAN is really minimizing the JS divergence or Wasserstein distance, then unless the number of samples is exponential in the number of dimensions memorizing the examples is always better than generating the real distribution.

Theorem 3.1. *Let μ be uniform Gaussian distributions $\mathcal{N}(0,\frac{1}{d}I)$. Suppose $\hat{\mu}$ is empirical versions of μ with m samples. If $\log m\ll d$, then with high probability*

$$d_{JS}(\mu,\hat{\mu})=\log 2.$$
$$d_W(\mu,\hat{\mu})\geq 1.1.$$

Let $\mu=\mathcal{D}_{real}$, and think of $\hat{\mu}$ as the uniform distribution of the training examples. This Theorem shows that the real distribution $\mathcal{D}_{real}$ is actually very far from the uniform distribution of the training samples. On the other hand, if the generator actually memorized the training examples, of course we have $d_{JS}(\hat{\mu},\hat{\mu})=d_W(\hat{\mu},\hat{\mu})=0$. Therefore the generator is likely to overfit.

3.2 Generalization

In the warm up example we only considered the samples from real distribution. In reality even for the distribution generated by the generator we cannot use infinite number of samples.

We say that the training *generalizes* if the empirical distance (in whatever metric) well-approximates the distance (in the same metric) between the full distributions.

Similar to the intuitions in Theorem 3.1, we show that this may not be true even for very simple distributions unless if the sample has exponential size. The next theorem shows that the distance between the empirical distributions can be close to the maximum possible (namely, $\log 2$ for d_{JS} and $\sqrt{2}$ for Wasserstein) even if the samples are drawn from the *same* distribution. For JS

divergence, we also show even if we add noise to both empirical distributions, the distance can still be very large (even samples are noised as part of training.)

Theorem 3.2. *Let μ, ν be uniform Gaussian distributions $\mathcal{N}(0, \frac{1}{d}I)$. Suppose $\hat{\mu}, \hat{\nu}$ are empirical versions of μ, ν with m samples. If $\log m \ll d$, then with high probability*

$$d_{JS}(\mu, \nu) = 0, d_{JS}(\hat{\mu}, \hat{\nu}) = \log 2.$$
$$d_W(\mu, \nu) = 0, d_W(\hat{\mu}, \hat{\nu}) \geq 1.1.$$

Further, let $\tilde{\mu}, \tilde{\nu}$ be the convolution of $\hat{\mu}, \hat{\nu}$ with a Gaussian distribution $N(0, \frac{\sigma^2}{d}I)$, as long as $\sigma < \frac{c}{\sqrt{\log m}}$ for small enough constant c, we have with high probability

$$d_{JS}(\tilde{\mu}, \tilde{\nu}) > \log 2 - 1/m.$$

The proof idea is surprisingly simple – with m samples, with very high probability even the closest pair between a sample from $\hat{\mu}$ and a sample $\hat{\nu}$ will be at least 1.1. The JS-divergence is large because the supports are disjoint, the Wasserstein distance is large because no matter how we "couple" samples from $\hat{\mu}$ and $\hat{\nu}$ they need to be transported for a distance at least 1.1.

Theorem 3.2 shows that even if the generator happens to find the real distribution, if the discriminator is powerful enough to compute either the JS divergence or Wasserstein distance, then distance between the empirical distributions can still be large and the generator has no idea that it has succeeded (and may move away).

Therefore, the GAN objective should not be interpreted as minimizing these two distances between the distributions, and it is non-rigorous to use that intuition to reason about the design of the procedure.

3.3 New distance metric with generalization bounds

Can we still think of GANs as minimizing some distance metric between the real distribution $\mathcal{D}_{real}$ and $G_u(\mathcal{D}_h)$? We now define a new distance metric that captures the actual optimization being performed, and which does lead to the generalization desired.

Definition 1. For two distributions μ and ν the *neural network divergence with respect to a class of discriminators $\{D_v : v \in \mathcal{V}\}$ and measuring function f* is defined as:

$$d_{NN}(\mu \| \nu) = \max_{v \in \mathcal{V}} \mathop{\mathbb{E}}_{x \sim \mu}[f(D_v(x))] + \mathop{\mathbb{E}}_{x \sim \nu}[f(D_v(1-x))].$$

In particular when $f(x) = x$ we call this the *neural network distance*[4] :

$$d_{NN}(\mu, \nu) = \max_{v \in \mathcal{V}} \left| \mathop{\mathbb{E}}_{x \sim \mu}[D_v(x)] - \mathop{\mathbb{E}}_{x \sim \nu}[D_v(x)] \right|.$$

[4]Technically this is a pseudometric, because $d_{NN}(\mu, \nu) = 0$ does not imply $\mu = \nu$. On the other hand, this metric is symmetric and satisfies triangle inequality. Symmetry is easy to see, triangle inequality follows because for three distributions μ_1, μ_2, μ_3, if v is the optimal discriminator for μ_1, μ_3 then $d_{NN}(\mu_1, \mu_2) \geq |\mathbb{E}_{x \sim \mu_1}[D_v(x)] - \mathbb{E}_{x \sim \mu_2}[D_v(x)]|$ and similarly $d_{NN}(\mu_2, \mu_3) \geq |\mathbb{E}_{x \sim \mu_2}[D_v(x)] - \mathbb{E}_{x \sim \mu_3}[D_v(x)]|$. Hence $d_{NN}(\mu_1, \mu_3) \leq d_{NN}(\mu_1, \mu_2) + d_{NN}(\mu_2, \mu_3)$.

Note that if the class of discriminators D_v is the set of all 1-Lipschitz functions, then the definition is exactly the same as Wasserstein distance (3). (But not all 1-Lipschitz functions may be computable by small neural nets.)

Intuitively, the definition and the next theorem addresses the fear that the generator has overfitted to the idiosyncracies of the specific empirical sample of $\mathcal{D}_{real}$ through interacting with the adversarial discriminator that it trained with. The theorem says that if the generator achieves a certain payoff against the any discriminator on the training samples, then it can achieve a similar payoff against all the same discriminator even if the discriminator now have access to infinitely many samples from $\mathcal{D}_{real}$.

In the theorem, $\{D_v : v \in \mathcal{V}\}$ is any class of discriminators that is L-Lipschitz with respect to v, measuring function f takes value in $[-\Delta, \Delta]$ and is L_f-Lipschitz. The theorem shows that the distance between the empirical distributions closely tracks the distance between the true distributions, given (relatively) modest number of samples. As usual, n is the number of parameters of the neural net.

Theorem 3.3. *In the setting of previous paragraph, let μ, ν be two distributions and $\hat{\mu}, \hat{\nu}$ be empirical versions with m samples each. There is a some fixed constant C such that when $m \geq \frac{Cn\Delta^2 \log(LL_f n/\epsilon)}{\epsilon^2}$, then we have with high probability:*

$$|d_{NN}(\hat{\mu}\|\hat{\nu}) - d_{NN}(\mu\|\nu)| \leq \epsilon.$$

The proof (appearing in the supplementary) uses standard concentration inequalities and a standard ϵ-net argument: 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 3.3 shows that the neural network divergence (and neural network distance) has a much better generalization bound than Jensen-Shannon divergence or Wasserstein distance. If the GAN successfully minimized the neural network divergence over empirical samples, then we know the neural network divergence between the distributions $\mathcal{D}_{real}$ and $G_u(\mathcal{D}_h)$ is also small.

To be more rigorous one would need to argue that this generalization continues to hold at every iteration of the training. While we can resample from $\mathcal{D}_h$ for every generator G_u, it is infeasible to get more samples from $\mathcal{D}_{real}$ in every iteration. The following corollary shows the generalization bound holds for every iteration as long as the number of iterations is not too large.

Corollary 3.1. *Let $u_1, u_2, ..., u_t$ ($\log t \ll d$) be t sets of parameters for the generator. For each u_i, let $\hat{\mathcal{D}}_h[t]$ be a fresh set of m samples from the distribution $G_{u_i}(\mathcal{D}_h)$. Let $\hat{\mathcal{D}}_{real}$ be m samples from the real distribution. With high probability when $m \geq \frac{n\Delta^2 \log(LL_f n/\epsilon)}{\epsilon^2}$ we have for all t*

$$|d_{NN}(D_{u_i}(\hat{\mathcal{D}}_h[t])\|\hat{\mathcal{D}}_{real}) - d_{NN}(D_{u_i}(\mathcal{D}_h)\|\mathcal{D}_{real})| \leq \epsilon.$$

The key observation here is that the objective is separated into two parts and the generator is not directly related to $\mathcal{D}_{real}$. So even though we cannot resample real samples, the generalization bound still holds. Detailed proof appear in supplementary material.

Limitations of neural net distance Note that the ability to generalize also comes with a (necessary) cost. 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 Theorem 3.3, we obtain:

Corollary 3.2 (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 fixed constant C such that when $m \geq \frac{Cn\Delta^2 \log(LL_f d/\epsilon)}{\epsilon^2}$, then we have with high probability*

$$d_{NN}(\mu, \hat{\mu}) \leq \epsilon.$$

That is, the neural network distance cannot distinguish between the real distribution μ and an empirical distribution with roughly n/ϵ^2 samples. Thus the discriminator is incapable of detecting when the synthetic distribution has low diversity.

4 Expressive power and existence of 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 n. 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 n parameters, then a mixture of $\tilde{O}(n \log(n/\epsilon)/\epsilon^2)$ 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 (formalized below) 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 f, and works for $f(x) = x$ (i.e., Wasserstein GAN). For other f we only show existence of mixed equilibria with small mixtures.

4.1 General f: Mixed Equilibrium

For general measure function f 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 $\{G_v, v \in \mathcal{V}\}$, we can define the payoff $F(u, v)$ of the game between generator and discriminator

$$F(u,v) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{real}}[f(D_v(x))] + \mathop{\mathbb{E}}_{h\sim\mathcal{D}_h}[f(1 - D_v(G_u(h)))]. \tag{5}$$

Of course as we discussed in 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 D_v supported on $\mathcal{V}$.

Theorem 4.1 (vonNeumann)**.** *There exists a value V, and a pair of distributions ($\mathcal{D}_u$,$\mathcal{D}_v$) such that*

$$\forall v, \quad \mathop{\mathbb{E}}_{u\sim\mathcal{D}_u}[F(u,v)] \le V \quad \textit{and} \quad \forall u, \quad \mathop{\mathbb{E}}_{v\sim\mathcal{D}_v}[F(u,v)] \ge V.$$

Note that this equilibrium involves both parties announcing their strategies $\mathcal{D}_u, \mathcal{D}_v$ 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 $u \sim \mathcal{D}_u, h \sim \mathcal{D}_h$, 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 $v \sim \mathcal{D}_v$, and then output $D_v(x)$. Note that in general this is very different from a discriminator D that outputs $\mathbb{E}_{v\sim\mathcal{D}_v}[D_v(x)]$, because the measuring function f is in general nonlinear. In particular, the correct payoff function for a mixture of discriminator is:

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

Of course, this equilibrium involving an infinite mixture makes little sense in practice. We show that (as is folklore in game theory[Lipton et al., 2003]) that we can *approximate* this min-max solution with mixture of finitely many generators and discriminators. More precisely we define ϵ-approximate equilibrium

Definition 2. A pair of mixed strategy $(\mathcal{D}_u, \mathcal{D}_v)$ is an ϵ-approximate equilibrium, if for some value V

$$\forall v \in \mathcal{V}, \quad \mathop{\mathbb{E}}_{u\in\mathcal{U}}[F(u,v)] \le V + \epsilon;$$
$$\forall u \in \mathcal{U}, \quad \mathop{\mathbb{E}}_{v\in\mathcal{V}}[F(u,v)] \ge V - \epsilon.$$

If the strategies $\mathcal{D}_u, \mathcal{D}_v$ are pure strategies, then this pair is called an ϵ-approximate pure equilibrium.

Suppose f is L_f-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 large enough constant $C > 0$ such that for any ϵ, there exists $T = \frac{C\Delta^2 n \log(LL'L_f \cdot n/\epsilon)}{\epsilon^2}$ generators $G_{u_1}, \ldots G_{u_T}$ and T discriminators $D_{v_1}, \ldots D_{v_T}$, let $\mathcal{D}_u$ be a uniform distribution on u_i and $\mathcal{D}_v$ be a uniform distribution on v_i, then (D_u, D_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 $\tilde{O}(n)$ generators can achieve an ϵ-approximate equilibrium.

4.1.1 $f(x) = x$: Pure Equilibrium

Now we consider the special case of Wasserstein-GAN, which is equivalent to setting $f(x) = x$ in Equation (2). In this case, it is possible to augment the network structure, and achieve an approximate pure equilibrium for the GAN game for networks of size $\tilde{O}(n^2)$. This should be interpreted as: if deep nets of size n are capable of generating a Gaussian, then the W-GAN game for neural networks of size $\tilde{O}(n^2)$ has an approximate equilibrium in which the generator wins. (The theorem is stated for RELU gates but also holds for standard activations such as sigmoid.)

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

To prove this theorem, we consider the mixture of generators and discriminators as in Theorem 4.2, and show how to fold the mixture into a larger $p+1$-layer neural network. We sketch the idea; details are in the supplementary.

Mixture of Discriminators Given a mixture of T discriminators $D_{v_1}, ..., D_{v_T}$, we can just define $D(x) = \frac{1}{T}\sum_{i=1}^{T} D_{v_i}(x)$. Because $f(x) = x$, we know for any generator G_u

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}_{real}}[D(x)] = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{real}, i\in[T]}[D_{v_i}(x)]$$
$$\mathop{\mathbb{E}}_{h\sim\mathcal{D}_h}[(1 - D(G_u(h))] = \mathop{\mathbb{E}}_{h\sim\mathcal{D}_h, i\in[T]}[(1 - D_{v_i}(G_u(h))].$$

Therefore, the payoff for D is exactly the same as the payoff for the mixture of discriminators. Also, the discriminator D is easily implemented as a network T times as large as the original network by adding a top layer that averages the output of the T generators $D_{v_i}(x)$.

Mixture of Generators For mixture of generators, we construct a single neural network that approximately generates the mixture distribution using the gaussian input it has. To do that, we can pass the input h through all the generators $G_{u_1}, G_{u_2}, ..., G_{u_T}$. We then show how to implement a "multi-way selector" that will select a uniformly random output from $G_{u_i}(h)$ ($i \in [T]$). The selector involves a simple 2-layer network that selects a number i from 1 to T with the appropriate probability and "disables"all the neural nets except the ith one by forwarding an appropriate large negative input.

Theorem 4.3 suggests that the objective of Wasserstein GAN may have approximate pure equilibrium for certain architectures of neural networks, which lends credence that Wasserstein GAN may be more robust.

Remark: In practice, GANs use highly structured deep nets, such as convolutional nets. Our current proof of existence of pure equilibrium requires introducing less structured elements in the net, namely, the multiway selectors that implement the mixture within a single net. It is left for future work whether pure equilibria exist for the original structured architectures. In the meantime, in practice we recommend using, even for W-GAN, a mixture of structured nets for GAN training, and it seems to help in our experiments reported below.

5 MIX+GANs

Theorem 4.2 and Theorem 4.3 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, W-GAN etc., giving us new training methods MIX+DCGAN, MIX+W-GAN etc.

We use exponentiated gradient[Kivinen and Warmuth, 1997]: store the log-probabilities $\{\alpha_{u_i}, i \in [T]\}$, and then obtain the weights by applying soft-max function on them:

$$w_{u_i} = \frac{e^{\alpha_{u_i}}}{\sum_{k=1}^{T} e^{\alpha_{u_k}}}, \quad i \in [T]$$

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 (5). We use both measuring functions $f(x) = \log x$ (for original GAN) and $f(x) = x$ (for WassersteinGAN). In our experiments we alternatively update generators' and discriminators' parameters as well as their corresponding log-probabilities using ADAM [Kingma and 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 term that discourages the weights being too 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 MIX+GAN on image generation tasks: MNIST dataset [LeCun et al., 1998] of hand-written digits and the CeleA [Liu et al., 2015] dataset of human faces. Then for more quantitative evaluation we use the CIFAR-10 dataset [Krizhevsky and 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, CeleA 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 was done (and is left for further work). All related code is to be released. Please refer to our code for hyper-parameters and experimental setup.

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 CeleA using the authors' code as a black box, and compared visual qualities of generated images vs 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 MIX + DCGAN produces somewhat cleaner digits than DCGAN (note the fuzziness in the latter). Interestingly, each component of our mixture learns a slightly different style of strokes.

Figure 3 demonstrates results on CeleA dataset, using the same architecture as for MNIST, except the models are not conditioned on image labels anymore.

The MIX+DCGAN model generates more faithful and more diverse samples than the baseline DCGAN does on both datasets, even though one may need to zoom in to fully perceive the difference since the two datasets are rather easy for a powerful training like DCGAN.

6.2 Quantitative Results

Now we turn to quantitative measurement using Inception Score [Salimans et al., 2016]. Throughout, we use mixtures of 5 generators and discriminators. At first sight the comparison DCGAN vs MIX + DCGAN seems unfair because the latter uses as much capacity as 5 DCGAN's, with corresponding penalty in running time per epoch. On the other hand, in deep net training increased capacity isn't always a good idea, since training may use it badly (eg, to overfit).

To construct MIX+DCGAN, we build on top of the DCGAN trained with losses proposed by Huang et al. [2016b], namely adversarial loss, entropy loss and conditional loss, which is the best

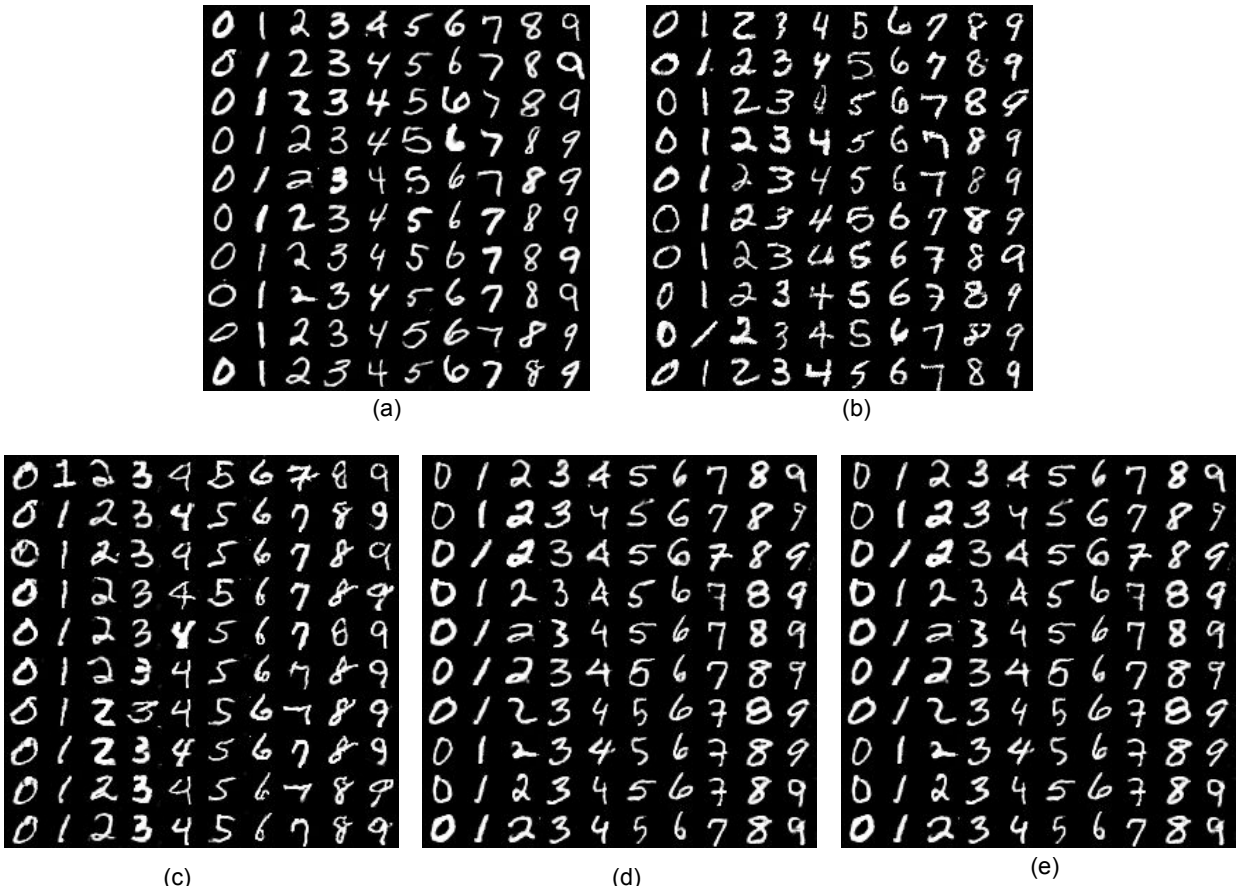

Figure 2: **MNIST Samples**. Digits generated from (a) MIX+DCGAN. (b) DCGAN. (c)-(d)-(e) From each of 3 components of MIX+DCGAN. (View on-screen with zooming in.)

DCGAN so far without improved training techniques. The same hyper-parameters are used for fair comparison. See Huang et al. [2016b] 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.

Table 1 is a quantitative comparison between our mixture models and other state-of-the-art GAN models on the CIFAR-10 dataset, with inception score calculated using 50,000 freshly generated samples from each model 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. We find surprisingly that, simply by applying MIX+ to the baseline models, our MIX+ models achieve 7.65 v.s. 7.16 gain in the score on DCGAN, and 4.04 v.s. 3.82 gain on WassersteinGAN. We also try to compare to a DCGAN with 5 times as many parameters. Without careful hyper-parameter tuning, the performance is only 7.22. We are running experiments to tune the hyper-parameters more carefully and applying MIX+ models to more GANs, due to the limit in GPU resources more data will be available in the next version.

Figure 4 shows how Inception Scores of MIX+DCGAN v.s. DCGAN evolve during training.

Figure 3: **CeleA Samples**. Faces generated from (a) MIX+DCGAN. (b) DCGAN.(c)-(d)-(e) Each of 3 components of MIX+DCGAN. (View on-screen with zooming in.)

MIX+DCGAN outperforms DCGAN throughout the entire training process, showing that it makes effective use of the additional 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 5 shows the training dynamics of neural network divergence of MIX+WassersteinGAN v.s. WassersteinGAN, which strongly indicates that MIX+WassersteinGAN is capable of achieving a much lower divergence as well as of improving the visual quality of generated samples.

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.

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 and 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., 2016b])	8.59± 0.12
DCGAN (as reported in Wang and Liu [2016])	6.58
DCGAN (best variant in Huang et al. [2016b])	7.16±0.10
DCGAN (5x size, no tuning)	7.22±0.07
MIX+DCGAN (Ours, with 5 components)	7.72±0.09
Wasserstein GAN	3.82±0.06
MIX+WassersteinGAN (Ours, with 5 components)	4.04±0.07
Real data	11.24±0.12

One issue raised by our analysis is that the current GANs objectives cannot enforce that the synthetic distribution has high diversity. This cannot be fixed by simply providing the discriminator with more training examples. Possibly some other change to the GANs setup can fix this.

The paper also made progress on other unexplained issues 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 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, note that existence of an equilibrium does not imply that a simple algorithm (in this case, backpropagation) would find it easily. That still defies explanation.

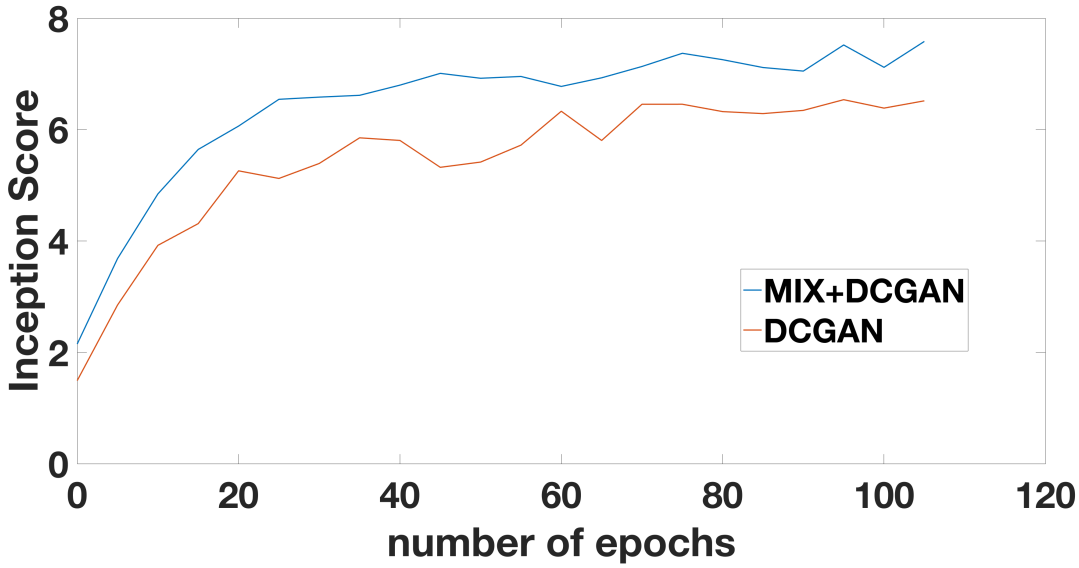

Figure 4: **MIX+DCGAN v.s. DCGAN Training Curve (Inception Score)**. MIX+DCGAN is consistently higher than DCGAN.

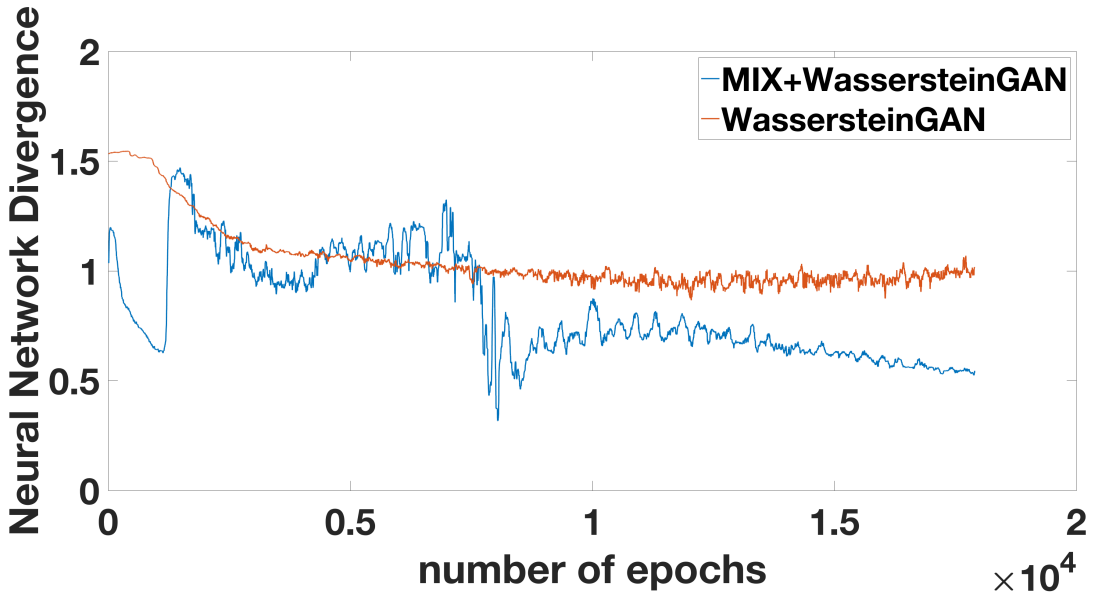

Figure 5: **MIX+WassersteinGAN v.s. WassersteinGAN Training Curve (Wasserstein Objective)**. MIX+WassersteinGAN is better towards the end but loss drops less smoothly, which needs further investigation.

References

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

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

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

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

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

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. *arXiv preprint arXiv:1612.04357*, 2016a.

xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. Technical report, 2016b.

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

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

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

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

Richard J Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 36–41. ACM, 2003.

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

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

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

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

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

Luca Trevisan, Madhur Tulsiani, and Salil Vadhan. Regularity, boosting, and efficiently simulating every high-entropy distribution. In *Computational Complexity, 2009. CCC'09. 24th Annual IEEE Conference on*, pages 126–136. IEEE, 2009.

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

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

A Omitted Proofs

In this section we give detailed proofs for the theorems in the main document.

A.1 Omitted Proofs for Section 3: Distance Metric and Generalization

We first show that traditional distance metrics can lead to overfitting.

Theorem A.1 (Theorem 3.1 restated)**.** *Let μ be uniform Gaussian distributions $\mathcal{N}(0, \frac{1}{d}I)$. Suppose $\hat{\mu}$ is empirical versions of μ with m samples. If $\log m \ll d$, then with high probability*

$$d_{JS}(\mu, \hat{\mu}) = \log 2.$$
$$d_W(\mu, \hat{\mu}) \geq 1.1.$$

Proof. The proof is very simple. For Jensen-Shannon divergence, observe that μ is a continuous distribution and $\hat{\mu}$ is discrete, therefore $d_{JS}(\mu, \hat{\mu}) = \log 2$.

For Wasserstein distance, let $x_1, x_2, ..., x_m$ be the empirical samples. For $y \sim \mathcal{N}(0, \frac{1}{d}I)$, by standard concentration and union bounds, we have

$$\Pr[\forall i \in [m] \|y - x_i\| \geq 1.2] \geq 1 - m\exp(-\Omega(d)) \geq 1 - o(1).$$

Therefore, using the earth-mover interpretation of Wasserstein distance, we know $d_W(\mu, \hat{\mu}) \geq 1.1$. □

Next we consider sampling for both the generated distribution and the real distribution, and show that the traditional distance metrics do not generalize.

Theorem A.2 (Theorem 3.2 restated)**.** *Let μ, ν be uniform Gaussian distributions $\mathcal{N}(0, \frac{1}{d}I)$. Suppose $\hat{\mu}, \hat{\nu}$ are empirical versions of μ, ν with m samples. Then when $\log m \ll d$, we have with high probability*

$$d_{JS}(\mu, \nu) = 0, d_{JS}(\hat{\mu}, \hat{\nu}) = \log 2.$$
$$d_W(\mu, \nu) = 0, d_W(\hat{\mu}, \hat{\nu}) \geq 1.1.$$

Further, let $\tilde{\mu}, \tilde{\nu}$ be the convolution of $\hat{\mu}, \hat{\nu}$ with a Gaussian distribution $N(0, \frac{\sigma^2}{d}I)$, as long as $\sigma < \frac{c}{\sqrt{\log m}}$ for small enough constant c, we have with high probability

$$d_{JS}(\tilde{\mu}, \tilde{\nu}) > \log 2 - 1/m.$$

Proof. For the Jensen-Shannon divergence, we know with probability 1 the supports of $\hat{\mu}, \hat{\nu}$ are disjoint, therefore $d_{JS}(\hat{\mu}, \hat{\nu}) = 1$.

For Wasserstein distance, note that for two random Gaussian vectors $x, y \sim N(0, \frac{1}{d}I)$, their difference is also a Gaussian with expected square norm 2. Therefore we have

$$\Pr[\|x - y\|^2 \leq 2 - \epsilon] \leq \exp(-\Omega(\epsilon^2 d)).$$

As a result, when $\log m \ll d$, we can union bound over all the m^2 pairwise distances for points in support of $\hat{\mu}$ and support of $\hat{\nu}$. With high probability, the closest pair between $\hat{\mu}$ and $\hat{\nu}$ has distance at least 1, therefore the Wasserstein distance $d_W(\hat{\mu}, \hat{\nu}) \geq 1.1$.

Finally we prove that even if we add noise to the two distributions, the JS divergence is still large. For distributions $\tilde{\mu}, \tilde{\nu}$, let ρ_1, ρ_2 be their density functions. Let $g(x) = \rho_1(x) \log \frac{2\rho_1(x)}{\rho_1(x)+\rho_2(x)} + \rho_2(x) \log \frac{2\rho_2(x)}{\rho_1(x)+\rho_2(x)}$, we can rewrite the JS divergence as

$$d_{JS}(\tilde{\mu}, \tilde{\nu}) = \int \frac{1}{2} g(x) dx.$$

Let z_x be a Bernoulli variable with probability $\rho_1(x)/(\rho_1(x) + \rho_2(x))$ of being 1. Note that $g(x) = (\rho_1(x) + \rho_2(x))(\log 2 - H(z_x))$ where $H(z_x)$ is the entropy of z_x. Therefore $0 \le g(x) \le (\rho_1(x) + \rho_2(x)) \log 2$. Let $\mathcal{X}$ be the union of radius-0.2 balls near the $2m$ samples in $\hat{\mu}$ and $\hat{\nu}$. Since with high probability, all these samples have pairwise distance at least 1, by Gaussian density function we know (a) the balls do not intersect; (b) within each ball $\frac{\max\{d_1(x), d_2(x)\}}{\min\{d_1(x), d_2(x)\}} \ge m^2$; (c) the union of these balls take at least $1 - 1/2m$ fraction of the density in $(\hat{\mu} + \hat{\nu})/2$.

Therefore for every $x \in \mathcal{X}$, we know $H(z_x) \le o(1/m)$, therefore

$$\begin{aligned}
d_{JS}(\tilde{\mu}, \tilde{\nu}) &= \int \frac{1}{2} g(x) dx \\
&\ge \int_{x \in \mathcal{X}} \frac{1}{2} g(x) dx \\
&\ge \int_{x \in \mathcal{X}} (\rho_1(x) + \rho_2(x))(\log 2 - o(1/m)) dx \\
&\ge \log 2 - 1/2m - o(1/m) \ge \log 2 - 1/m.
\end{aligned}$$

□

Next we prove the neural network distance does generalize, given enough samples.

Theorem A.3 (Theorem 3.3 restated)**.** *Let μ, ν be two distributions and $\hat{\mu}, \hat{\nu}$ be empirical versions with m samples each. There is a some fixed constant C such that when $m \ge \frac{Cn\Delta^2 \log(LL_f n/\epsilon)}{\epsilon^2}$, then we have with probability close to 1:*

$$|d_{NN}(\hat{\mu} \| \hat{\nu}) - d_{NN}(\mu \| \nu)| \le \epsilon.$$

Proof. The proof uses concentration bounds. We show that with high probability, for every discriminator D_v,

$$|\mathop{\mathbb{E}}_{x \sim \mu}[f(D_v(x))] - \mathop{\mathbb{E}}_{x \sim \hat{\mu}}[f(D_v(x))]| \le \epsilon/2, \tag{6}$$

$$|\mathop{\mathbb{E}}_{x \sim \nu}[f(1 - D_v(x))] - \mathop{\mathbb{E}}_{x \sim \hat{\nu}}[f(1 - D_v(x))]| \le \epsilon/2. \tag{7}$$

If $d_{NN}(\mu \| \nu) = t$, let D_v be the optimal discriminator, we then have

$$\begin{aligned}
d_{NN}(\mu \| \nu) &\ge \mathop{\mathbb{E}}_{x \sim \hat{\mu}}[f(D_v(x))] + \mathop{\mathbb{E}}_{x \sim \hat{1-\nu}}[f(D_v(x))]. \\
&\ge \mathop{\mathbb{E}}_{x \sim \mu}[f(D_v(x))] + \mathop{\mathbb{E}}_{x \sim \nu}[f(D_v(x))] \\
&\quad - |\mathop{\mathbb{E}}_{x \sim \mu}[f(D_v(x))] - \mathop{\mathbb{E}}_{x \sim \hat{\mu}}[f(D_v(x))]| \\
&\quad - |\mathop{\mathbb{E}}_{x \sim \nu}[f(1 - D_v(x))] - \mathop{\mathbb{E}}_{x \sim \hat{\nu}}[f(1 - D_v(x))]| \\
&\ge t - \epsilon.
\end{aligned}$$

The other direction is similar.

Now we prove the claimed bounds (6) (proof of (7) is identical). Let $\mathcal{X}$ be a finite set such that every point in $\mathcal{V}$ is within distance $\epsilon/8LL_f$ of a point in X (a so-called $\epsilon/8LL_f$-net). Standard constructions give an X satisfying $\log|\mathcal{X}| \leq O(n\log(LL_f n/\epsilon))$. For every $v \in \mathcal{X}$, by Chernoff bound we know

$$\Pr[|\mathop{\mathbb{E}}_{x\sim\mu}[f(D_v(x))] - \mathop{\mathbb{E}}_{x\sim\hat{\mu}}[f(D_v(x))]| \geq \frac{\epsilon}{4}] \leq \exp(-\Omega(\frac{\epsilon^2 m}{\Delta^2})).$$

Therefore, when $m \geq \frac{Cn\Delta^2\log(LL_f n/\epsilon)}{\epsilon^2}$ for large enough constant C, we can union bound over all $v \in \mathcal{X}$. With high probability, for all $v \in \mathcal{X}$ we have $|\mathbb{E}_{x\sim\mu}[f(D_v(x))] - \mathbb{E}_{x\sim\hat{\mu}}[f(D_v(x))]| \geq \frac{\epsilon}{4}$.

Now, for every $v \in \mathcal{V}$, we can find a $v' \in \mathcal{X}$ such that $\|v - v'\| \leq \epsilon/8LL_f$. Therefore

$$\begin{aligned}
&|\mathop{\mathbb{E}}_{x\sim\mu}[f(D_v(x))] - \mathop{\mathbb{E}}_{x\sim\hat{\mu}}[f(D_v(x))]| \\
\leq&|\mathop{\mathbb{E}}_{x\sim\mu}[f(D_{v'}(x))] - \mathop{\mathbb{E}}_{x\sim\hat{\mu}}[f(D_{v'}(x))]| \\
&+|\mathop{\mathbb{E}}_{x\sim\mu}[f(D_{v'}(x))] - \mathop{\mathbb{E}}_{x\sim\mu}[f(D_v(x))]| \\
&+|\mathop{\mathbb{E}}_{x\sim\hat{\mu}}[f(D_{v'}(x))] - \mathop{\mathbb{E}}_{x\sim\hat{\mu}}[f(D_v(x))]| \\
\leq&\epsilon/4 + \epsilon/8 + \epsilon/8 \\
\leq&\epsilon/2.
\end{aligned}$$

This finishes the proof of (6). □

Finally, we generalize the above Theorem to hold for *all* generators in a family.

Corollary A.1 (Corollary 3.1 restated)**.** *Let $u_1, u_2, ..., u_t$ ($\log t \ll d$) be t sets of parameters for the generator. For each u_i, let $\hat{\mathcal{D}}_h[t]$ be a fresh set of m samples from the distribution $G_{u_i}(\mathcal{D}_h)$. Let $\hat{\mathcal{D}}_{real}$ be a set of m samples from the real distribution. With high probability when $m \geq \frac{n\Delta^2\log(LL_f n/\epsilon)}{\epsilon^2}$ we have for all t*

$$|d_{NN}(D_{u_i}(\hat{\mathcal{D}}_h[t])\|\hat{\mathcal{D}}_{real}) - d_{NN}(D_{u_i}(\mathcal{D}_h)\|\mathcal{D}_{real})| \leq \epsilon.$$

Proof. This follows from the proof of Theorem 3.3. Note that we have fresh samples for every generator distribution, so Equation (7) is true with high probability by union bound. For the real distribution, notice that Equation (6) does not depend on the generator, so it is also true with high probability. □

A.2 Omitted Proof for Section 4: Expressive power and existence of equilibrium

Mixed Equilibrium We first show there is a finite mixture of generators and discriminators that approximates the equilibrium of infinite mixtures.

Theorem A.4 (Theorem 4.2 restated)**.** *In the settings of Theorem 4.2, there is a large enough constant $C > 0$ such that for any ϵ, there exists $T = \frac{C\Delta^2 n\log(LL'L_f\cdot n/\epsilon)}{\epsilon^2}$ generators $G_{u_1}, \ldots G_{u_T}$ and T discriminators $D_{v_1}, \ldots D_{v_T}$, let $\mathcal{D}_u$ be a uniform distribution on u_i and $\mathcal{D}_v$ be a uniform distribution on v_i, then (D_u, D_v) is an ϵ-approximate equilibrium.*

Further, if the class of generator can generate a Gaussian, and the class of discriminator includes constant functions, then the value of the game $V = 2f(1/2)$ (where f is the connection function in (2)).

Proof. Let ($\mathcal{D}_u$,$\mathcal{D}_v$) be the pair of optimal solutions as in Theorem 4.1 and V be the optimal value. We will show that randomly sampling T generators and discriminators from these two distributions gives the desired mixture with high probability.

Construct $\epsilon/4LL'L_f$-nets U, V for $\mathcal{U}$ and $\mathcal{V}$. By standard construction, the sizes of these ϵ-nets satisfy $\log(|U|+|V|) \leq C'n\log(LL'L_f \cdot n/\epsilon)$ for some constant C'. Let $u_1, u_2, ..., u_T$ be independent samples from $\mathcal{D}_u$, and $v_1, v_2, ..., v_T$ be independent samples from $\mathcal{D}_v$. By Chernoff bound, for any $u \in U$, we know

$$\Pr[\mathop{\mathbb{E}}_{i\in[T]}[F(u,v_i)] \leq \mathop{\mathbb{E}}_{v\in\mathcal{V}}[F(u,v)] - \epsilon/2] \leq \exp(-\frac{\epsilon^2 T}{2\Delta^2}).$$

When $T = \frac{C\Delta^2 n\log(L\cdot L_f\cdot n/\epsilon)}{\epsilon^2}$ and the constant C is large enough ($C \gg 2C'$), with high probability this inequality is true for all $u \in U$. Now, for any $u \in \mathcal{U}$, let u' be the closest point in the ϵ-net. By the construction of the net, $\|u - u'\| \leq \epsilon/4LL'L_f$. It is easy to check that $F(u,v)$ is $2LL'L_f$-Lipschitz in both u and v, therefore

$$\mathop{\mathbb{E}}_{i\in[T]}[F(u',v_i)] \geq \mathop{\mathbb{E}}_{i\in[T]}[F(u,v_i)] - \epsilon/2.$$

Combining the two inequalities we know for any $u' \in \mathcal{U}$,

$$\mathop{\mathbb{E}}_{i\in[T]}[F(u',v_i)] \geq V - \epsilon.$$

This finishes the proof for the second inequality. The proof for the first inequality is identical. By probabilistic argument we know there must exist such generators and discriminators.

Finally, we prove the fact that the value V must be equal to $2f(1/2)$. For the discriminator, one strategy is to just output $1/2$. This strategy has payoff $2f(1/2)$ no matter what the generator does, so $V \geq 2f(1/2)$. For the generator, consider the distribution $\mathcal{D}_\zeta = \mathcal{D}_{real} + \zeta N(0, I)$, which is the convolution of $\mathcal{D}_{real}$ and a Gaussian of variance ζI. For any ζ, $\mathcal{D}_\zeta$ can be expressed as a infinite mixture of Gaussians and is therefore a mixed strategy of the generator. The Wasserstein distance between $\mathcal{D}_\zeta$ and $\mathcal{D}_{real}$ is $O(\zeta)$. Since the discriminator is L'-Lipschitz, it cannot distinguish between $\mathcal{D}_\zeta$ and $\mathcal{D}_{real}$. In particular we know for any discriminator D_v

$$|\mathop{\mathbb{E}}_{x\sim\mathcal{D}_\zeta}[f(1-D_v(x))] - \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{real}}[f(1-D_v(x))]| \leq O(L_f L'\zeta).$$

Therefore,

$$\begin{aligned}
&\max_{v\in\mathcal{V}} \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{real}}[f(D_v(x))] + \mathop{\mathbb{E}}_{x\sim\mathcal{D}_\zeta}[f(1-D_v(x))] \\
\leq& O(L_f L'\zeta) + \max_{v\in\mathcal{V}} \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{real}}[f(D_v(x)) + f(1-D_v(x))] \\
\leq& 2f(1/2) + O(L_f L'\zeta).
\end{aligned}$$

Here the last step uses the assumption that f is concave. Therefore the value is upperbounded by $V \leq 2f(1/2) + O(L_f L'\zeta)$ for any ζ. Taking limit of ζ to 0, we have $V = 2f(1/2)$. □

Pure equilibrium Now we show for Wasserstein objective, there exists an approximate pure equilibrium

Theorem A.5 (Theorem 4.3 restated)**.** *Suppose the generator and discriminator are both p-layer neural networks ($p \geq 2$) with n parameters, and the last layer uses ReLU activation function. In the setting of Theorem 4.2 there exists $p+1$-layer neural networks of generators G and discriminator D with $O\left(\frac{\Delta^2 n^2 \log(LL'L_f \cdot n/\epsilon)}{\epsilon^2}\right)$ parameters, such that there exists an ϵ-approximate pure equilibrium. Furthermore, if the generator is capable of generating a Gaussian then the value $V = 1$.*

In order to prove this theorem, the major step is to construct a generator that works as a mixture of generators.

Mixture of Generators For mixture of generators, we need to construct a single neural network that approximately generates the mixture distribution using the gaussian input it has. To do that, we can pass the input h through all the generators $G_{u_1}, G_{u_2}, ..., G_{u_T}$. We then show how to implement a "multi-way selector" that will select a uniformly random output from $G_{u_i}(h)$ ($i \in [T]$).

In order to do that, we first observe that it is possible to compute a step function using a two layer neural network. This is fairly standard for many activation functions.

Lemma 1. *Fix an arbitrary $k \in \mathcal{N}$ and $z_1 < z_2 < \cdots < z_k$. For any $0 < \delta < \min\{z_{i+1} - z_i\}$, there is a two-layer neural network with a single input $h \in \Re$ that outputs $k+1$ numbers $x_1, x_2, ..., x_{k+1}$ such that (i) $\sum_{i=1}^{k+1} x_i = 1$ for all h; (ii) when $h \in [z_{i-1} + \delta/2, z_i - \delta/2]$, $x_i = 1$ and all other x_j's are 0*[5].

Proof. Using a two layer neural network, we can compute the function $f_i(h) = \max\{\frac{h - z_i - \delta/2}{\delta}, 0\} - \max\{\frac{h - z_i + \delta/2}{\delta}, 0\}$. This function is 0 for all $h < z_i - \delta/2$, 1 for all $h \geq z_i + \delta/2$ and change linearly in between. Now we can write $x_1 = 1 - f_1(h)$, $x_{k+1} = f_k(h)$, and for all $i = 2, 3, ..., k$, $x_k = f_i(h) - f_{i-1}(h)$. It is not hard to see that these functions satisfy our requirements. □

Using these step functions, we can essentially select one output from the T generators.

Lemma 2. *In the setting of Theorem 4.3, for any $\delta > 0$, there is a $p+1$-layer neural network with $O\left(\frac{\Delta^2 n^2 \log(LL'L_f \cdot n/\epsilon)}{\epsilon^2}\right)$ parameters that can generate a distribution that is within δ total variational difference with the mixture of $G_{u_1}, G_{u_2}, ..., G_{u_T}$.*

The idea is simple: since we have implemented step functions from Lemma 1, we can just pass through the input through all the generators $G_{u_1}, ..., G_{u_T}$. For the last layer of G_{u_i}, we add a large multiple of $-(1 - x_i)$ where x_i is the i-th output from the network in Lemma 1. Clearly, if $x_i = 0$ this is going to effectively disable the neural network; if $x_i = 1$ this will have no effect. By properties of x_i's we know most of the time only one $x_i = 1$, hence only one generator is selected.

Proof. Suppose the input for the generator is $(h_0, h) \sim N(0,1) \times \mathcal{D}_h$ (i.e. h_0 is sampled from a Gaussian, h is sampled according to $\mathcal{D}_h$ independently). We pass the input h through the generators and gets outputs $G_{u_i}(h)$, then we use h_0 to select one as the true output.

Let $z_1, z_2, ..., z_{T-1}$ be real numbers that divides the probability density of a Gaussian into T equal parts. Pick $\delta' = \delta/100T$ in Lemma 1, we know there is a 2-layer neural network that computes step functions $x_1, ..., x_T$. Moreover, the probability that $(x_1, ..., x_T)$ has more than 1 nonzero entry is smaller than δ. Now, for the output of $G_{u_i}(h)$, in each output ReLU gate, we add a very large multiple of $-(1 - x_i)$ (larger than the maximum possible output). This essentially "disables" the

[5] When $h \leq z_1 - \delta/2$ only x_1 is 1 and when $h \geq z_k + \delta/2$ only $x_{k+1} = 1$

output when $x_i = 0$ because before the result before ReLU is always negative. On the other hand, when $x_i = 1$ this preserves the output. Call the modified network $\hat{G}_{u_i}$, we know $\hat{G}_{u_i} = G_{u_i}$ when $x_i = 1$ and $\hat{G}_{u_i} = 0$ when $x_i = 0$. Finally we add a layer that outputs the sum of $\hat{G}_{u_i}$. By construction we know when $(x_1, ..., x_T)$ has only one nonzero entry, the network correctly outputs the corresponding $G_{u_i}(x_i)$. The probability that this happens is at least $1-\delta$ so the total variational distance with the mixture is bounded by δ. □

Using the generator and discriminators we constructed, it is not hard to prove Theorem 4.3. The only thing to notice here is that when the generator is within δ total variational distance to the true mixture, the payoff $F(u, v)$ can change by at most $2\Delta\delta$.

Proof of Theorem 4.3. Let T be large enough so that there exists an $\epsilon/2$-approximate mixed equilibrium. Let the new set of discriminators be the convex combination of T discriminators $\{D_v, v \in \mathcal{V}\}$. Let the new set of generators be constructed as in Lemma 2 with $\delta \leq \epsilon/4\Delta$ and $G_{u_1}, ..., G_{u_T}$ from the original set of generators. Let D be the discriminator which is the average of the T discriminators from the approximate mixed equilibrium, and G be the generator constructed by the T generators from the approximate mixed equilibrium. Define $F^\star(G, D)$ be the payoff of the new two-player game. Now, for any discriminator D', think of it as a distribution of G_v, we know

$$\begin{aligned} F^\star(G, D') &\geq \mathop{\mathbb{E}}_{i\in[T], v\in D'} F(u_i, v) \\ &\quad - |F^\star(G, D') - \mathop{\mathbb{E}}_{i\in[T], v\in D'} F(u_i, v)| \\ &\geq V - \epsilon/2 - 2\Delta\frac{\epsilon}{4\Delta} \\ &\geq V - \epsilon. \end{aligned}$$

The bound from the first term comes from Theorem 4.2, and the fact that the expectation is smaller than the max. The bound for the second term comes from the fact that changing a δ fraction of probability mass can change the payoff F by at most $2\Delta\delta$.

Similarly, for any generator G', we know it is close to a mixture of generators G_u, therefore

$$\begin{aligned} F^\star(G', D) &\leq \mathop{\mathbb{E}}_{i\in[T], u\in G'} F(u, v_i) \\ &\quad + |F^\star(G', D) - \mathop{\mathbb{E}}_{i\in[T], u\in G'} F(u, v_i)| \\ &\leq V + \epsilon/2 + 2\Delta\frac{\epsilon}{4\Delta} \\ &\leq V + \epsilon. \end{aligned}$$

This finishes the proof. □

B Examples when best response fail

In this section we construct simple generators and discriminators and show that if both generators and discriminators are trained to optimal with respect to Equation (2), the solution will cycle and cannot converge. For simplicity, we will show this when the connection function is f, but it is possible to show similar result even when f is the traditional log function.

We consider a simple example where we try to generate points on a circle. The true distribution $\mathcal{D}_{true}$ has $1/3$ probability to generate a point at angle $0, 2\pi/3, 4\pi/3$. Let us first consider a case when the generator does not have enough capacity to generate the true distribution.

Definition 3 (Example 1)**.** The generator G has one parameter $\theta \in [0, 2\pi)$, and always generates a point at angle θ. The discriminator D has a parameter $\phi \in [0, 2\pi)$, and $D_\phi(\tau) = \exp(-10d(\tau, \phi)^2)$. Here $d(\tau, \phi)$ is the angle between τ and ϕ and is always between $[0, \pi]$.

We will analyze the "best response" procedure (as in Algorithm 1). We say the procedure converges if $\lim_{i\to\infty} \phi^i$ and $\lim_{i\to\infty} \theta^i$ exist.

Algorithm 1 Best Response

Initialize θ^0.
for $i = 1$ to T **do**
  Let $\phi^i = \arg\max_\phi \mathbb{E}_{\tau\sim\mathcal{D}_{true}}[D_\phi(\tau)] - \mathbb{E}_{\tau\sim G(\theta)}[D_\phi(\tau)]$.
  Let $\theta^i = \arg\min_\theta - \mathbb{E}_{\tau\sim G(\theta)}[D_\phi(\tau)]$.
end for

Theorem B.1. *For generator G and discriminator D in Example 1, for every choice of parameter θ, there is a choice of ϕ such that $D_\phi(\theta) \le 0.001$ and $\mathbb{E}_{\tau\sim\mathcal{D}_{\sqcup\nabla\sqcap}}[D_\phi(\tau)] \ge 1/3$. On the other hand, for every choice of ϕ, there is always a θ such that $D_\phi(\theta) = 1$. As a result, the sequence of best response cannot converge.*

Proof. For any θ, we can just choose ϕ to be the farthest point in $\{0, 2\pi/3, 4\pi/3\}$. Clearly the distance is at least $2\pi/3$ and therefore $D_\phi(\theta) \le \exp(-10) \le 0.001$. On the other hand, for the true distribution, it has $1/3$ probability of generating the point ϕ, therefore $\mathbb{E}_{\tau\sim\mathcal{D}_{\sqcup\nabla\sqcap}}[D_\phi(\tau)] \ge 1/3$. For every ϕ, we can always choose $\theta = \phi$, and we have $D_\phi(\theta) = 1$.

By the construction of ϕ^i and θ^i in Algorithm 1, we know for any i $D_{\phi^i}(\theta^i) = 1$, but $\mathbb{E}_{\tau\sim\mathcal{D}_{true}}[D_\phi(\tau)] - D_{\phi^{i-1}}(\theta^i) \ge 1/4$. Therefore $|D_{\phi^i}(\theta^i) - D_{\phi^{i-1}}(\theta^i)| \ge 1/4$ for all i and the sequences cannot converge. □

This may not be very surprising as the generator does not even have the capacity to generate the true distribution. One might hope that once we use a large enough neural network, the generator will be able to generate the true distribution. However, our next example shows even in that case the best response algorithm may not converge.

Definition 4 (Example 2)**.** Let $\theta, \phi \in [0, 2\pi)^3$ will be 3 -dimensional vectors. The generator $G(\theta)$ generates the uniform distribution over points $\theta_1, \theta_2, \theta_3$. The discriminator function is chosen to be $D_\phi(\tau) = \frac{1}{3}\sum_{i=1}^3 \exp(-10d(\tau, \phi_i)^2)$.

Clearly in this example, the true distribution can be generated by the generator (just by choosing $\theta = (0, 2\pi/3, 4\pi/3)$). However we will show that the best response algorithm still cannot always converge.

Theorem B.2. *Suppose the generator and discriminator are described as in Example 2, and $\theta^0 = (0, 0, 0)$, then we have: (1) In every iteration the three points for generator $\theta^i_{1,2,3}$ are equal to each other. (2) In every iteration θ^i_1 is 0.1-close to one of the true points $\{0, 2\pi/3, 4\pi/3\}$, and its closest point is different from the previous iteration.*

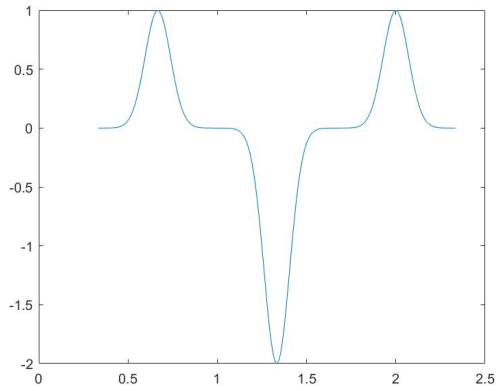

Figure 6: Optimization problem for ϕ

Before giving detailed calculations, we first give an intuitive argument. In this example, we will use induction to prove that at every iteration t, two properties are preserved: 1. The three points of the generator $(\theta_1^t, \theta_2^t, \theta_3^t)$ are close to the same real example ($0, 2\pi/3$ or $4\pi/3$); 2. The three points of the discriminator $(\phi_1^{t+1}, \phi_2^{t+1}, \phi_3^{t+1})$ will be close to the other two real examples. To go from 1 to 2, notice that in this case the three ϕ values can be optimized independently (and the final objective is the sum of the three), so it suffices to argue for one of them. For one ϕ, by our construction the objective function is really close to the sum of two Gaussians at the other two real examples, minus twice of a Gaussian at the real example that θ_i^t's are close to (see Figure 6). From the Figure it is clear that the maximum of this function is close to one of the real examples that is different from θ_i^t.Now all the three ϕ_i^{t+1}'s will be close to one of the two real examples, so one of them is going to get at least two ϕ_i^{t+1}'s. In the next iteration, as the generator is trying to maximize the output of discriminator, all three θ_i^{t+1}'s will be close to the real example with at least two ϕ_i^{t+1}'s.

Now we make this intuition formal through calculations.

Proof. The optimization problems here are fairly simple and we can argue about the solutions directly. Throughout the proof we will use the fact that $\exp(-10*(2\pi/3)^2) < 1e-4$ and $\exp(1+\epsilon) \approx 1+\epsilon$. The following claim describes the properties we need:

Claim 1. *If $\theta_1 = \theta_2 = \theta_3$ and θ_1 is 0.1-close to one of $\{0, 2\pi/3, 4\pi/3\}$, then the optimal ϕ must satisfy ϕ_1, ϕ_2, ϕ_3 be 0.05-close to the other two points in $\{0, 2\pi/3, 4\pi/3\}$.*

We first prove the Theorem with the claim. We will do this by induction. The induction hypothesis is that for every $j \le t$, we have $\theta_{1,2,3}^j$ are equal to each other, and θ_1^j is 0.1-close to one of the true points $\{0, 2\pi/3, 4\pi/3\}$. This is clearly true for $t = 0$. Now let us assume this is true for t and consider iteration $t+1$.

Without loss of generality we assume θ_1^t is close to 0. Now by Claim we know $\phi_1^t, \phi_2^t, \phi_3^t$ are 0.05-close to either $2\pi/3$ or $4\pi/3$. Without loss of generality assume there are more ϕ_i^t's close to $2\pi/3$ (the number of ϕ_i^t's close to the two point has to be different because there are 3 ϕ_i^t's). Now, by property of Gaussians, we know $D_{\phi^t}(\tau)$ has a unique maximum that is within 0.1 of $2\pi/3$ (the influence from the other point is much smaller than 0.05). Since the generator is trying to

maximize $\mathbb{E}_{\tau\sim G(\theta)}[D_\phi(\tau)]$, all three $\theta_i^t (i = 1, 2, 3)$ should be at this maximizer. This finishes the induction. □

Now we prove the claim.

Proof. The objective function is

$$\max_\phi \frac{1}{3}\left(\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}_{true}}[\sum_{i=1}^{3}\exp(-10d(\tau,\phi_i)^2)] - \mathop{\mathbb{E}}_{\tau\sim G(\theta)}[\sum_{i=1}^{3}\exp(-10d(\tau,\phi_i)^2)]\right).$$

In this objective function ϕ_i's do not interact with each other, so we can break the objective function into three (one for each ϕ_i). Without loss of generality, assume $\theta_{1,2,3}$ are 0.1-close to 0, we are trying to maximize

$$\max_\phi \left(\frac{1}{3}\sum_{i=1}^{3}\exp(-10d(\phi, i\cdot 2\pi/3)^2)] - \exp(-10d(\theta,\phi)^2)\right).$$

Clearly, if ϕ is not 0.05 close to either $2\pi/3$ or $4\pi/3$, we have $D \le 1/3 - 10\cdot 0.05^2 + \exp(-10) \le 1/3 - 0.02$. On the other hand, when $\phi = 2\pi/3$ or $4\pi/3$, we have $D \ge 1/3 - \exp(-10)$. Therefore the maximum must be 0.05-close to one of the two points. □

Note that this example is very similar to the *mode collapse* problem mentioned in NIPS 2016 GAN tutorial[Goodfellow, 2016].