ON CONVERGENCE AND STABILITY OF GANS

Anonymous authors
Paper under double-blind review

ABSTRACT

We analyze convergence of GANs through the lens of online learning and game theory, to understand what makes it hard to achieve consistent stable training in practice. We identify that the underlying game here can be *ill-posed* and *poorly conditioned*, and propose a simple regularization scheme based on local perturbations of the input data to address these issues. Currently, the methods that improve stability either impose additional computational costs or require the usage of specific architectures/modeling objectives. Further, we show that WGAN-GP, which is the state-of-the-art stable training procedure, is similar to LS-GAN, does not follow from KR-duality and can be too restrictive in general. In contrast, our proposed algorithm is fast, simple to implement and achieves competitive performance in a stable fashion across a variety of architectures and objective functions with minimal hyperparameter tuning. We show significant improvements over WGAN-GP across these conditions.

1 INTRODUCTION

Generative modeling involves taking a set of samples drawn from an unknown data generating distribution P_{real} and finding an estimate P_{model} that closely resembles it. Generative adversarial networks (GANs) (Goodfellow et al. (2014)) are a class of powerful *implicit* generative models with wide-ranging applications. The basic setup consists of two networks, the generator and the discriminator, playing against each other in a repeated zero-sum game setting. The main goal here is to reach an equilibrium where the JS-divergence between P_{real} and P_{model} is minimized. However, there is little theoretical understanding regarding the convergence of GANs. It is not clear how many and what kind of equilibria exist in this game and how to reach our desired solution. Further, the usual training procedure which involves alternating gradient updates is highly unstable and this is widely attributed to the adversarial component of GANs (Goodfellow (2017), Salimans et al. (2016)). We show that this is not always the case.

In this paper, we address these questions by first analyzing the convex case of GANs. This setting has a 'unique solution' and we show guaranteed convergence when no-regret algorithms like online gradient descent (Zinkevich (2003)) are used. This establishes that the adversarial component is not wholly responsible for the training issues. However, this result does not hold when the players' strategy spaces are non-convex, which is usually the case when neural networks are used. This is due to the fact that, in non-convex settings, there can be local equilibria (Ratliff et al. (2013)) which can prevent training from converging to the global equilibrium. In this sense, the GAN game can be ill-posed in general, meaning that it has multiple solutions of varying desirability. Further, the game can be also poorly conditioned if players are allowed to use arbitrary functions. We hypothesize these to be the reasons for training instability and mode collapse. To mitigate these issues, we propose a novel regularization penalty on the discriminator which uses local perturbations of the input data.

Several recent works focus on stabilizing the training of GANs. While some solutions (Radford et al. (2015), Salimans et al. (2016)) require the usage of specific architectures/modeling objectives, some (Che et al. (2016), Zhao et al. (2016)) significantly deviate from the original GAN framework. Other promising works in this direction (Metz et al. (2016), Arjovsky et al. (2017), Qi (2017), Gulrajani et al. (2017)) impose a significant computational overhead. Thus, a fast and versatile method for consistent stable training of GANs is still missing in the literature. Our proposed algorithm is simple to implement, fast (using only alternating gradient updates), and achieves competitive performance in a stable fashion across different architectures (150 random setups), datasets (MNIST, CIFAR-10, CelebA) and objective functions (f-divergence measures) with minimal hyperparameter tuning. We

show improvements over WGAN-GP, which is the state-of-the-art stable training procedure, in both the modeling performance and training stability. Further, we show that its penalty is similar to LS-GAN's (Loss-Sensitive GAN) proposal, doesn't follow from KR-duality (Villani (2008)) and can be too restrictive in general.

To summarize, our contributions are as follows:

- We analyze the convex case of the GAN setup and show that convergence guarantees can be obtained when no-regret algorithms are used.
- We hypothesize that ill-posedness and poor conditioning are responsible for the training instability and mode collapse in GANs.
- We propose a simple, versatile regularization scheme to address these issues and provide experimental evidence to demonstrate the resulting improvements.
- We show that WGAN-GP's penalty does not follow from KR-duality, is very similar to LS-GAN and can be too restrictive in general. Our algorithm beats WGAN-GP, in both the modeling performance and training stability.

2 THEORY

In this section, we start with the formulation of the GAN game (section 2.1). Then, we consider its convex case and show proof of guaranteed convergence to the 'unique solution' using no-regret algorithms (section 2.2). However, these results do not apply to the non-convex case as there can be multiple local equilibria, in general. To mitigate this issue and improve the conditioning of the game, we propose a simple regularization scheme using local perturbations of the input data and provide geometric intuition about its effects (section 2.3). Finally, we show that WGAN-GP's gradient penalty is similar to LS-GAN's proposal, does not follow from KR duality as claimed in (Gulrajani et al. (2017)) and demonstrate how it can be too restrictive in general (section 2.4).

2.1 GAME FORMULATION

The GAN framework consists of two players - the *generator*, which produces synthetic data given some noise source and the *discriminator*, which is trained to distinguish generator's samples from the real data. The generator model G is parameterized by ϕ, takes a noise vector $\mathbf{z}$ as input, and produces a synthetic sample $G_\phi(\mathbf{z})$. The discriminator model D is parameterized by θ, takes a sample $\mathbf{x}$ as input and computes $D_\theta(\mathbf{x})$, which can be interpreted as the probability that $\mathbf{x}$ is real.

The models G, D can be arbitrary functions, although in practice, it is common to use deep networks. Their cost functions are defined as -

$$J^{(D)}(\phi, \theta) = -\mathbb{E}_{x \sim p_{real}} \log D_\theta(x) - \mathbb{E}_{\mathbf{z}} \log(1 - D_\theta(G_\phi(z)))$$

$$J^{(G)}(\phi, \theta) = \mathbb{E}_{\mathbf{z}} \log(1 - D_\theta(G_\phi(z)))$$

And the complete game can be specified as -

$$\min_\phi \max_\theta \Big\{ J(\phi, \theta) = \mathbb{E}_{x \sim p_{real}} \log D_\theta(x) + \mathbb{E}_{\mathbf{z}} \log(1 - D_\theta(G_\phi(z))) \Big\}$$

At equilibrium, the generator distribution P_{model} converges to the real distribution P_{real} if updates are made in the function space (Goodfellow et al. (2014)).

2.2 CONVEX CASE AND NO-REGRET ALGORITHMS

The minimax theorem for zero-sum games has been significantly generalized since its original formulation and the work of Sion (1958) provides a broad class of scenarios in which $\inf\sup\{\} = \sup\inf\{\}$. For instance, if we are given convex and compact subsets or $\Phi \subset \mathbb{R}^m$ and $\Theta \subset \mathbb{R}^n$ and a function $J : \Phi \times \Theta \to \mathbb{R}$ that is convex in its first argument and concave in its second, then Sion's theorem implies that

$$\min_{\phi \in \Phi} \max_{\theta \in \Theta} J(\phi, \theta) = \max_{\theta \in \Theta} \min_{\phi \in \Phi} J(\phi, \theta).$$

For such min/max optimization scenarios, we will use the term *Nash equilibrium pair* to refer to any ϕ^*, θ^* such that $\phi^* \in \arg\min_{\phi\in\Phi} \max_{\theta\in\Theta} J(\phi,\theta)$ and $\theta^* \in \arg\max_{\theta\in\Theta} \min_{\phi\in\Phi} J(\phi,\theta)$. Note that there can be multiple equilibria but all of them will have the same payoff (unique *value* of the game (Neumann (1928))).

Many problems, including the development of GANs, can be posed as a min/max formulation and hence a natural question is how we can find such equilibrium pairs. The most straightforward procedure by which players might search for an equilibrium is best-response dynamics (BRD). In each round, best-responding players play their optimal strategy given their opponent's current strategy. Despite its simplicity, BRD does not necessarily converge and can lead to oscillations or instability. At least in the setting where the payoff $J(\cdot,\cdot)$ is convex/concave, we have a technique that is both efficient and provably works: *no-regret learning algorithms* (Cesa-Bianchi & Lugosi (2006)). If both players treat the selection of their strategy parameters ϕ_t, θ_t as a sequential game ($t = 1, 2, \ldots$), and they update these parameters via no-regret dynamics, then it is easy to show that the time-average parameter vectors will converge to a Nash equilibrium pair (Nisan et al. (2007)). Let us first define no-regret algorithms.

Definition 2.1 (**No-regret algorithm**). Given a sequence of convex loss functions $L_1, L_2, \ldots : K \to \mathbb{R}$, an algorithm that selects a sequence of k_t's, each of which may only depend on previously observed $L_1, \ldots, L_{t-1}$, is said to have *no regret* if $\frac{R(T)}{T} = o(1)$, where we define

$$R(T) := \sum_{t=1}^{T} L_t(k_t) - \min_{k\in K} \sum_{t=1}^{T} L_t(k)$$

We can apply no-regret learning to our problem of equilibrium finding in a game $J(\cdot,\cdot)$ as follows. The ϕ player imagines the function $J(\cdot,\theta_t)$ as his loss function on round t, and similarly the θ player imagines $-J(\phi_t,\cdot)$ as her loss function at t. After T rounds of play, each player computes the average iterates $\bar{\phi}_T := \frac{1}{T}\sum_{t=1}^{T}\phi_t$ and $\bar{\theta}_T := \frac{1}{T}\sum_{t=1}^{T}\theta_t$. If V^* is the equilibrium value of the game, and the ϕ and θ players suffer regret $R_1(T)$ and $R_2(T)$ respectively, then one can show that (Freund & Schapire (1999)) -

$$V^* - \frac{R_2(T)}{T} \leq \max_{\theta\in\Theta} J(\bar{\phi}_T,\theta) - \frac{R_2(T)}{T} \leq \min_{\phi\in\Phi} J(\phi,\bar{\theta}_T) + \frac{R_1(T)}{T} \leq V^* + \frac{R_1(T)}{T}.$$

In other words, $\bar{\theta}_T$ and $\bar{\phi}_T$ are "almost optimal" solutions to the game, where the "almost" approximation factor is given by the average regret terms $\frac{R_1(T)+R_2(T)}{T}$. Under the no-regret condition, the former will vanish, and hence we can guarantee convergence in the limit. Next, we define a popular family of no-regret algorithms.

Definition 2.2 (**Follow The Regularized Leader**). FTRL (Hazan et al. (2016)) selects k_t on round t by solving for $\arg\min_{k\in K}\{\sum_{s=1}^{t-1} L_s(k) + \frac{1}{\eta}\Omega(k)\}$, where $\Omega(\cdot)$ is some convex regularization function and η is a learning rate.

***Remark*:** Roughly speaking, if you select the regularization as $\Omega(\cdot) = \frac{1}{2}\|\cdot\|^2$, then FTRL becomes the well-known Online Gradient Descent (OGD). Ignoring the case of constraint violations, OGD can be written in a simple iterative form: $k_t = k_{t-1} - \eta_{t-1}\nabla L_{t-1}(k_{t-1})$.

Now we show how OGD can be applied in the GAN game, assuming G and D are chosen such that $J(\phi,\theta)$ is convex/concave. Notice that the min/max objective function in GANs involves a stochastic component, with two randomized inputs given on each round, x and z which are sampled from the data distribution and a standard multivariate normal, respectively. Let us write $J_{x,z}(\phi,\theta) := \log D_\theta(x) + \log(1 - D_\theta(G_\phi(z)))$. Taking expectations with respect to $\mathbf{x}$ and $\mathbf{z}$, we define the full (non-stochastic) game as $J(\phi,\theta) = \mathbb{E}_{\mathbf{x},\mathbf{z}}[J_{x,z}(\phi,\theta)]$. But the above online training procedure is still valid with stochastic inputs. That is, the equilibrium computation would proceed similarly, where on each round we sample x_t and z_t, and follow the updates

$$\phi_{t+1} \leftarrow \phi_t - \eta_t \nabla_\phi J_{x_t,z_t}(\phi_t,\theta_t). \quad \text{and} \quad \theta_{t+1} \leftarrow \theta_t + \eta_t^{'} \nabla_\theta J_{x_t,z_t}(\phi_t,\theta_t)$$

A benefit of this stochastic perspective is that we immediately get a generalization bound on the mean parameters $\bar{\phi}_T$ after T rounds of optimization. The celebrated "online-to-batch conversion" Cesa-Bianchi et al. (2004), now a standard result in online learning theory, implies that $\mathbb{E}_{\mathbf{x},\mathbf{z}}[J_{x,z}(\bar{\phi}_T,\theta)]$,

for any θ, is no more than the optimal value $\mathbb{E}_{\mathbf{x},\mathbf{z}}[J_{x,z}(\phi^*, \theta)]$ plus an "estimation error" bounded by $\mathbb{E}\left[\frac{R_1(T)+R_2(T)}{T}\right]$, where the expectation is taken with respect to the sequence of samples observed along the way, and any randomness in the algorithm. A limitation of this result, however, is that it requires a fresh sample x_t to be used on every round.

To summarize, we are guaranteed to converge to the unique solution of the GAN game (using OGD) if $J(\phi, \theta)$ is convex/concave. Thus, despite the adversarial setup, alternating gradient updates will be stable in this case. In the next section, we will discuss what happens in non-convex settings.

2.3 Non-convex case and Local equilibria

In practice, we choose G, D to be deep neural networks and the function $J(\phi, \theta)$ need not be convex/concave anymore. The discussion in section 2.2, and the very general results one can prove using no-regret techniques, relies on the crucial assumption that the payoff function $J(\cdot, \cdot)$ is convex-concave. Indeed convexity is used in a number of places in the proofs: vanishing regret relies on a sequence of convex loss functions, and the online-to-batch conversion rests on Jensen's inequality to show $\frac{1}{T}\sum_{t=1}^{T} J(\phi_t, \theta) \geq J\left(\frac{1}{T}\sum_{t=1}^{T} \phi_t, \theta\right)$. Regret minimization isn't always tractable beyond convex settings and we resort to heuristics to improve convergence.

In non-convex settings, we are not guaranteed that a unique solution exists. There can be multiple *local equilibria*, which are essentially local minima of the game. We define this notion below.

Definition 2.3 (**Local Equilibrium**). A pair (ϕ^*, θ^*) is called an ϵ-local equilibrium if it holds that

$$\forall \phi^{'}, ||\phi^{'} - \phi^*|| \leq \epsilon : J(\phi^*, \theta^*) \leq J(\phi^{'}, \theta^*)$$

$$\forall \theta^{'}, ||\theta^{'} - \theta^*|| \leq \epsilon : J(\phi^*, \theta^*) \geq J(\phi^*, \theta^{'})$$

In a local equilibrium, both of the players do not have an incentive to switch to any other strategy within a small neighborhood of their current strategies. It should be clear that all Nash equilibria of the game are also local equilibria, but its not the other way around. This implies that, unlike the convex case where all equilibria have the same payoff (unique solution), it is now possible to have spurious local equilibria. Therefore, the GAN setup is ill-posed in practice (w.r.t gradient descent dynamics at least), since there can be many solutions of varying desirability. Next, we reason about the dynamics of alternating gradient updates (OGD) in this setting.

As we noted, in local equilibria, each player is at a locally optimal choice of strategies, given his opponent's choice. So, alternating gradient updates and other such training procedures can get stuck in them. Another possibility by which we can reach a steady state is when the iterates just oscillate. In both the cases, the average iterates $\bar{\theta}_T$ and $\bar{\phi}_T$ converge to some fixed points $\hat{\theta}$ and $\hat{\phi}$, albeit potentially non-optimal. However, in the worst case, it is also possible that the algorithm does not converge at all. These phenomena explain why the training process of GANs can be quite unstable, often resulting in oscillations and mode collapse.

Given our understanding about convergence in GANs and how this affects training stability, we propose a novel regularization scheme for the discriminator that mitigates these issues.

Proposed Regularization:

$$\lambda \cdot \mathbb{E}_{x \sim P_{real}, \delta \sim N_d(0, cI)} \left[\|\nabla_{\mathbf{x}} D_\theta(x + \delta)\| - 1 \right]^2$$

The complete training procedure is specified in Algorithm 1. Notice that we are biasing the discriminator function to have norm-(≈ 1) gradients in a region around the real-data manifold. We recall an important fact, that *any smooth function with norm-1 gradients on a compact set is necessarily a linear function*, which suggests that our procedure encourages "close to linearity" at least along the data manifold. We believe that the following conjecture accounts for the dramatic reduction in mode collapse events when DRAGAN is used for training (demonstrated in section 3): *the proposed regularization scheme restricts the set of possible discriminator functions so as to significantly reduce the number of non-optimal fixed points.*

Further, due to our regularization scheme, $D_\theta(x)$ cannot change too rapidly (or arbitrarily) in domain space and over successive iterations. This improves the conditioning of the game. The score assigned by the discriminator to any $\hat{x}$ is affected by multiple real samples nearby and these effects go across iterations, leading to gradual variation in the iterates, which are each individually smooth in the feature space (atleast close to data-manifold). We hypothesize that this helps in convergence and improves training stability. This effect can be observed in the swissroll experiment of section 2.3 (notice how level sets change). We leave it to future works to theoretically study this and explore if an appropriate penalty with similar effects can be developed for the generator.

Some minor details about our algorithm:

- Though we use the original GAN objective in Algorithm 1, our penalty improves stability in a variety of other settings. This is demonstrated in section 3.3.
- We use small pixel-level noise here but it is possible to find better ways of imposing this penalty. However, this task is beyond the scope of our paper.
- We found that DRAGAN requires minimal hyperparameter tuning (suggested setting: $\lambda = 10, c \sim 10$) and is robust to small changes.

Algorithm 1 DRAGAN (Deep Regret Analytic Generative Adversarial Networks)

Initial weights (ϕ_0, θ_0)
for number of training iterations **do**
  Get a mini-batch of examples $\{x^1, x^2, ..., x^m\}$ from training data.
  Get a mini-batch of samples $\{z^1, z^2, ..., z^m\}$ from $\mathcal{N}_k(0, 1)$.
  Generate $\{x^1 + \delta^1, x^2 + \delta^2, ..., x^m + \delta^m\}$ where each δ^i is a small pixel-wise noise vector.
  Update the discriminator by descending along:
    $$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \left[-\log D_\theta(x^i) - \log(1 - D_\theta(G_\phi(z^i))) + \lambda \cdot (||\nabla_{\mathbf{x}} D_\theta(x^i + \delta^i)||_2 - 1)^2 \right]$$
  Update the generator by descending along:
    $$\nabla_{\boldsymbol{\phi}} \frac{1}{m} \sum_{i=1}^{m} \log \left[1 - D_{\boldsymbol{\theta}}(G_{\boldsymbol{\phi}}(z^i)) \right]$$
end for

2.4 Regularized GANs

Several recent works have proposed regularized models of GANs to improve stability. Our proposed approach roughly falls under this class of models and WGAN-GP, LS-GAN are the closest related approaches. First, we show that these two approaches are very similar. Qi (2017) introduced LS-GAN with the idea of maintaining a margin between losses assigned to real and fake samples. Specifically, they enforce the following constraint on the discriminator for any real and fake sample pair -

$$D_\theta(x) - D_\theta(G_\phi(z)) \geq ||x, G_\phi(z)||$$

Further, they also impose Lipschitz constraint on D which ensures that for any $(x, x^{'})$ -

$$|D_\theta(x) - D_\theta(x^{'})| \leq ||x, x^{'}||$$

The two constraints together result in a situation where the following holds for any real and fake sample pair (roughly) -

$$D_\theta(x) - D_\theta(G_\phi(z)) \approx ||x, G_\phi(z)|| \tag{1}$$

The authors argue that the resulting function would have non-vanishing gradients almost everywhere between real and fake samples (section 6 of Qi (2017)).

Gulrajani et al. (2017) proposed an extension to address various shortcomings of the original WGAN and they impose the following condition on D -

$$||\nabla_x D_\theta(\hat{x})||_2 \approx 1 \tag{2}$$

where $\hat{x} = (\epsilon)x + (1-\epsilon)G_\phi(z)$ is some point on the line between a real and a fake sample, both chosen independently at random. This leads to D having norm-1 gradients almost everywhere between

real and fake samples. Notice that this behavior is very similar to that of LS-GAN's discriminator function. Thus, WGAN-GP is a slight variation of the original LS-GAN algorithm and we refer to these techniques as "coupled smoothing" methods.

Next, we show that WGAN-GP's penalty doesn't follow from KR-duality as claimed. We restate Lemma 1 from Gulrajani et al. (2017) here.

Lemma 2.1 (**Optimal critic and KR-duality**). Let $\mathbb{P}_r$ and $\mathbb{P}_g$ be two distributions in X, a compact metric space. Then, there is a 1-Lipschitz function f^* which is the optimal solution of

$$\max_{||f||_L \leq 1} \mathbb{E}_{y \sim \mathbb{P}_r}[f(y)] - \mathbb{E}_{x \sim \mathbb{P}_g}[f(x)]$$

Let π be the optimal coupling between $\mathbb{P}_r$ and $\mathbb{P}_g$, defined as the minimizer of:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\pi \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \pi}[||x - y||]$$

Where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ is the set of joint distributions $\pi(x, y)$ whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$, respectively. Then, if f^* is differentiable, $\pi(x = y) = 0$ and $x_t = (t)x + (1 - t)y$ with $0 \leq t \leq 1$,

$$\mathbb{P}_{(x,y) \sim \pi} \left[\nabla f^*(x_t) = \frac{y - x_t}{||y - x_t||} \right] = 1$$

This, in particular, implies the norms of the gradients are 1.

It should be clear from Lemma 2.1 that the optimal discriminator D^* will have norm-1 gradients (almost everywhere) only between those x and $G_\phi(z)$ pairs which are sampled from the optimal coupling or joint distribution π. Therefore, there is no basis for WGAN-GP's penalty (equation 2) where arbitrary pairs of real and fake samples are used. We show with a simple toy experiment below that indeed this penalty can be quite restrictive in practice.

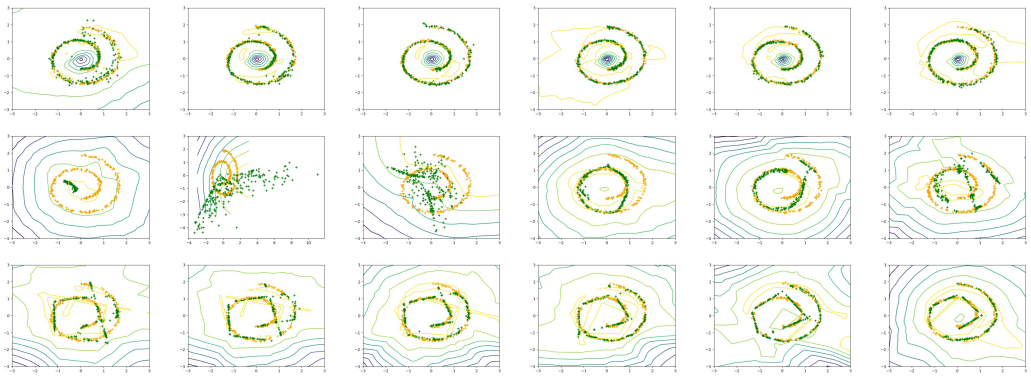


Figure 1: Swissroll experiment (different phases of training) - Vanilla GAN (top), WGAN-GP (middle), and DRAGAN (bottom). Real samples are marked orange and generated samples are green. Level sets of $D_\theta(x)$ are shown in the background where yellow is high and purple is low.

The most important distinction between coupled smoothing methods and ours is that we only impose gradient constraints in local regions around real samples. We refer to this technique as "local smoothing". Coupled smoothing involves imposing gradient constraints between real and generated samples. Note that the generated samples can be anywhere in the domain space and they change from one iteration to the next. In contrast, we consistently restrict $D_\theta(\cdot)$ only along the real-data manifold, which allows for broader flexibility in the vastly larger region outside of true density. These differences have implications for modeling performance, speed and stability of the algorithm (see section 3).

The resulting class of functions when coupled smoothing is used will be highly restricted compared to our method (which affects modeling performance). Our algorithm works with alternating gradient updates, while WGAN-GP needs multiple inner iterations to optimize D (since generated samples

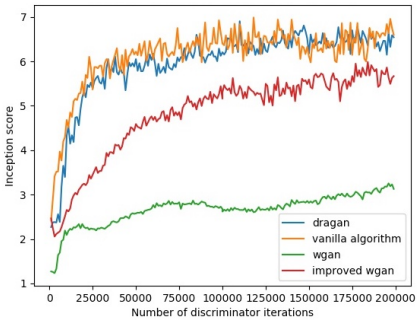
change every round). Further, D can change rapidly depending on the sequence of generated sample sets it receives and this affects stability/convergence. To conclude, appropriate regularization of the discriminator can improve training stability of GANs due to the reduction in number of spurious local equilibria and better conditioning of the underlying game. However, we should be careful about how we restrict the discriminator so that it doesn't negatively affect the modeling performance.

3 EXPERIMENTAL RESULTS

In section 3.1, we compare the modeling performance of our algorithm against vanilla GAN and WGANs in the standard DCGAN/CIFAR-10 setup. Section 3.2 demonstrates DRAGAN's superior stability properties across a variety of architectures. In section 3.3, we show that our method also works with other objective functions. Appendix contains samples for inspection, some of the plots and some additional results. Throughout, we use inception score (well-studied, reliable metric in the literature) and sample quality to measure the performance.

3.1 INCEPTION SCORES ON CIFAR-10 USING DCGAN ARCHITECTURE

DCGAN is a family of architectures designed to perform well with the vanilla training procedure. They are ubiquitous in the GAN literature owing to the instability of vanilla GAN in general settings. We use this architecture to model CIFAR-10 and compare against vanilla GAN, WGAN and WGAN-GP. As WGANs need 5 discriminator iterations for every generator iteration, comparing the modeling performance can be tricky. To address this, we report two scores for vanilla GAN and DRAGAN - one using the same number of generator iterations as WGANs and one using the same number of discriminator iterations. The results are shown in Figure 2 and samples are included in the appendix (Figure 5). Notice that DRAGAN beats WGANs in both the configurations, while vanilla GAN performs the best. A key point to note here is that our algorithm is fast compared to WGANs, so in practice, the performance will be closer to the DRAGAN^d case.



(a) Inception score plot

Algorithm	Score
WGAN	3.25
WGAN-GP	5.99
DRAGAN^g	6.11
DRAGAN^d	6.90
Vanilla GAN^g	6.3
Vanilla GAN^d	6.99

(b) Inception scores

Figure 2: Comparison of modeling performance on CIFAR10

3.2 BOGONET SCORE TO MEASURE STABILITY ACROSS ARCHITECTURES

One of the key advantages of our algorithm is that it improves stability in a wider range of architectures (other than DCGANs). Similar to Arjovsky et al. (2017); Gulrajani et al. (2017), we removed stabilizing components of DCGAN and demonstrated improved stability compared to the vanilla GAN (see section 5.2.4 in appendix). However, this is a small set of architectures and it is not clear if stability is improved in general.

To address this, we introduce a metric termed the *BogoNet* score to compare stability of different training procedures in the GAN setting. The basic idea is to choose random architectures for players G and D independently and evaluate the performance of different algorithms in the resulting games. A good algorithm should achieve stable performance without failing to learn or resulting in mode collapse despite the potentially imbalanced architectures. In our experiment, each player is assigned a network from a diverse pool of architectures belonging to three families (MLP, ResNet, DCGAN).

Table 1: Summary of inception score statistics across 100 architectures

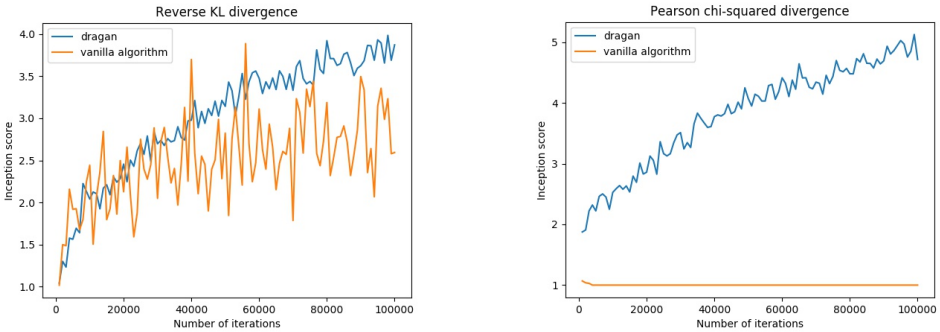
Algorithm	Final score		Area under curve		Qual. score
	Mean	Std	Mean	Std	Total
Vanilla GAN	2.91	1.44	277.72	126.09	92.5
DRAGAN	**3.70**	1.71	**312.15**	135.35	**157.5**
WGAN-GP	3.49	1.30	300.09	100.96	-

To demonstrate that our algorithm is more stable compared to vanilla GAN and WGAN-GP, we created 100 such instances of hard games. Each instance is trained using these algorithms on CIFAR-10 and we plot how inception score changes over time. For each algorithm, we calculated the average of final inception scores and area under the curve (AUC) over all 100 instances. The results are shown in Table 1. Notice that we beat the other algorithms in both metrics, which indicates some improvement in stability.

Further, we perform some qualitative analysis to verify that BogoNet score indeed captures the improvements in stability. We create another set of 50 hard architectures and compare DRAGAN against vanilla GAN. Each instance is worth 5 points and we split this bounty between the two algorithms depending on their performance. If both perform well or perform poorly, they get 2.5 points each. However, if one algorithm achieves stable performance compared to the other (in terms of failure to learn or mode collapses), we assign it higher portions of the bounty. Results were judged by two of the authors in a blind manner: The curves were shown side-by-side with the choice of algorithm for each side being randomized and unlabeled. The vanilla GAN received an average score of 92.5 while our algorithm achieved an average score of 157.5 and this correlates with BogoNet score from earlier. See section 5.3 in appendix for some additional details regarding this experiment.

3.3 Stability using different objective functions

Our algorithm can be used with a variety of objective functions and we demonstrate this using the following experiment. Nowozin et al. (2016) show that we can interpret GAN training as minimizing various f-divergences when an appropriate game objective function is used. We show experiments using the objective functions developed for Forward KL, Reverse KL, Pearson χ^2, Squared Hellinger, and Total Variation divergence minimization. We use a hard architecture from the previous subsection to demonstrate the improvements in stability. Our algorithm is stable in all cases except for Total Variation while the vanilla algorithm failed in all cases (see Figure 3 for two examples and Figure 10 in appendix for all five). Thus, practitioners can now choose their game objective from a larger set of functions and use DRAGAN (unlike WGANs which requires a specific objective function).



(a) Reverse KL

(b) Pearson χ^2

Figure 3: Inception score plots for two divergence measures, demonstrating superior stability for our algorithm.

4 CONCLUSIONS

In this paper, we analyze convergence of GANs through the lens of online learning and game theory to understand why consistent stable training is hard. The traditional way of interpreting GAN training as consistently minimizing some form of divergence can be misleading. We connect the idea of regret minimization to GANs and provide a novel way to reason about the dynamics in this game. In practice, the GAN game can be ill-posed and ill-conditioned, and we hypothesize this to be responsible for training instability and mode collapse. These issues are general in nature and a common theme in optimization literature. A simple regularization scheme is proposed that mitigates these problems and we provide empirical evidence for its strong, stable modeling performance in a variety of settings. We leave it to future works to analyze our ideas in more depth and use the resulting findings to develop better algorithms.

REFERENCES

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

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

Yoav Freund and Robert E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017. URL http://arxiv.org/abs/1701.00160.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *CoRR*, abs/1611.02163, 2016. URL http://arxiv.org/abs/1611.02163.

J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928. URL http://eudml.org/doc/159291.

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pp. 271–279, 2016.

Guo-Jun Qi. Loss-sensitive generative adversarial networks on lipschitz densities. *CoRR*, abs/1701.06264, 2017. URL http://arxiv.org/abs/1701.06264.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434 [cs]*, November 2015. URL http://arxiv.org/abs/1511.06434. arXiv: 1511.06434.

Lillian J Ratliff, Samuel A Burden, and S Shankar Sastry. Characterization and computation of local nash equilibria in continuous games. In *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*, pp. 917–924. IEEE, 2013.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. URL http://arxiv.org/abs/1606.03498.

Maurice Sion. On general minimax theorems. *Pacific J. Math*, 8(1):171–176, 1958.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 928–936, 2003.

5 APPENDIX

5.1 SAMPLES AND LATENT SPACE WALKS

In this section, we provide additional samples across algorithms and datasets. Further, Radford et al. Radford et al. (2015) suggest that walking on the manifold learned by the generator can expose signs of memorization. We use DCGAN to model MNIST and CelebA datasets using our algorithm and the results shown below demonstrate that our generator learns smooth transitions between different images.

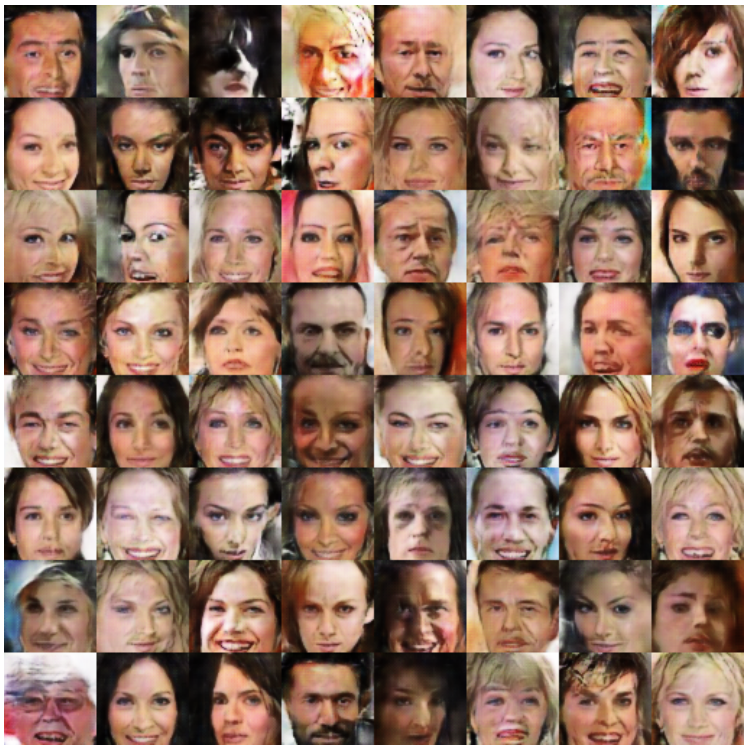
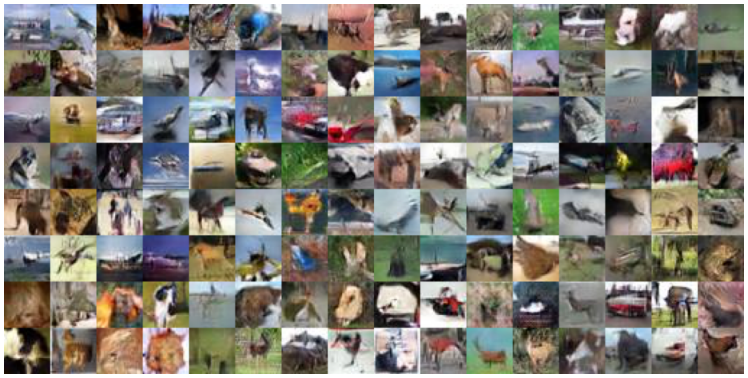


Figure 4: Samples for CelebA dataset using our algorithm with DCGAN architecture.

(a) Vanilla GAN

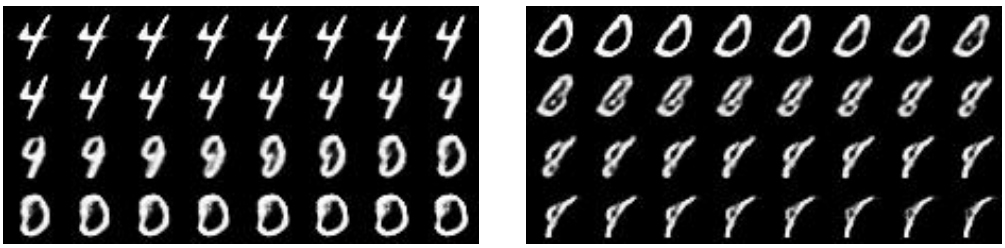


(b) Our Algorithm

(c) WGAN

(d) Improved WGAN

Figure 5: Samples for CIFAR-10 dataset using DCGAN architecture



(a) (b)

Figure 6: Latent space walk for MNIST using DCGAN architecture and our algorithm

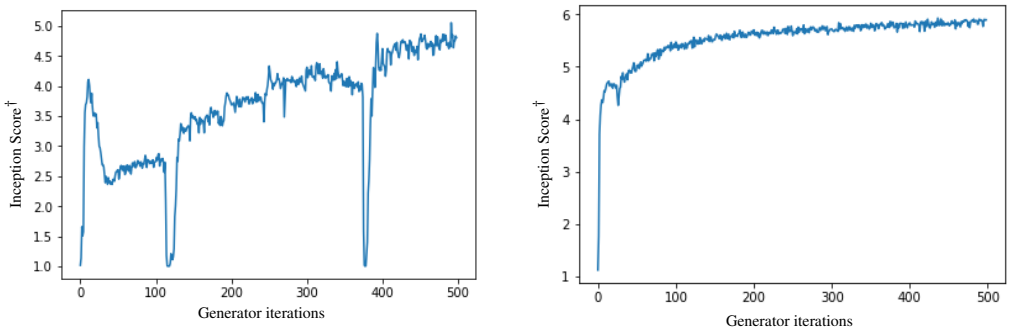
Figure 7: Latent space walk for CelebA dataset using DCGAN architecture and our algorithm

5.2 Additional Experiments

The inception score was introduced as a rough guide to evaluate generative models. To compute this score, we need a classifier model and the original paper suggests use of the inception model for this purpose. However, this model is too powerful for smaller datasets like MNIST and this makes it hard to analyze differences in the modeling capabilities of various algorithms. Therefore, we use a custom classifier built for the MNIST dataset to calculate inception scores in the next two subsections (we use a dagger to make this distinction).

5.2.1 ONE LAYER NETWORKS WITH MNIST DATASET

We design a simple experiment where G and D are both fully connected networks with just one hidden layer. Vanilla GAN performs poorly even in this simple setting and we observe severe mode collapse events. In contrast, our algorithm is stable throughout and obtains decent quality samples despite the constrained setup.



(a) Vanilla GAN

(b) Our algorithm

Figure 5: One layer networks with MNIST dataset - Inception score† plots

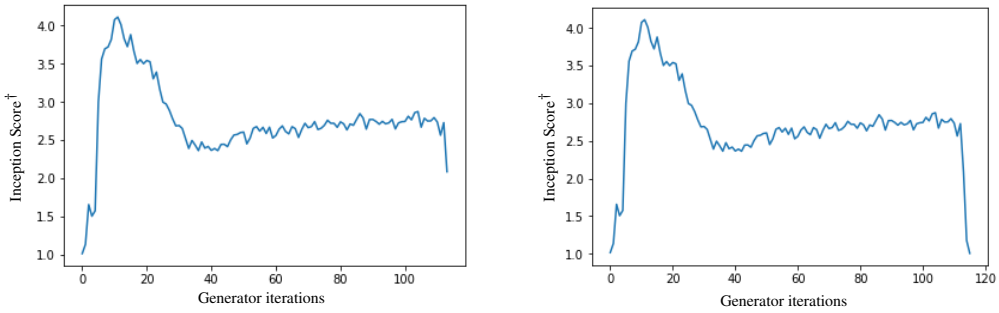
(a) Vanilla GAN

(b) Our algorithm

Figure 6: One layer networks with MNIST dataset - Samples

5.2.2 SMOOTHING PENALTY TRACKING (RECOVERING IS MISSING)

The stability in our algorithm comes from regularizing D with the penalty $(||\nabla_{\hat{\mathbf{x}}} D_{\boldsymbol{\theta}}(\hat{\mathbf{x}})||_2 - 1)^2$ as formulated in step 9 of Algorithm 1. We now take a closer look at mode collapse events that occur in the vanilla GAN and track the penalty in parallel (see Figure 7). Notice how the penalty increases as we enter mode collapse and drops as we recover. This indicates that the nature of the discriminator's gradients has a strong correlation with stability in GANs.



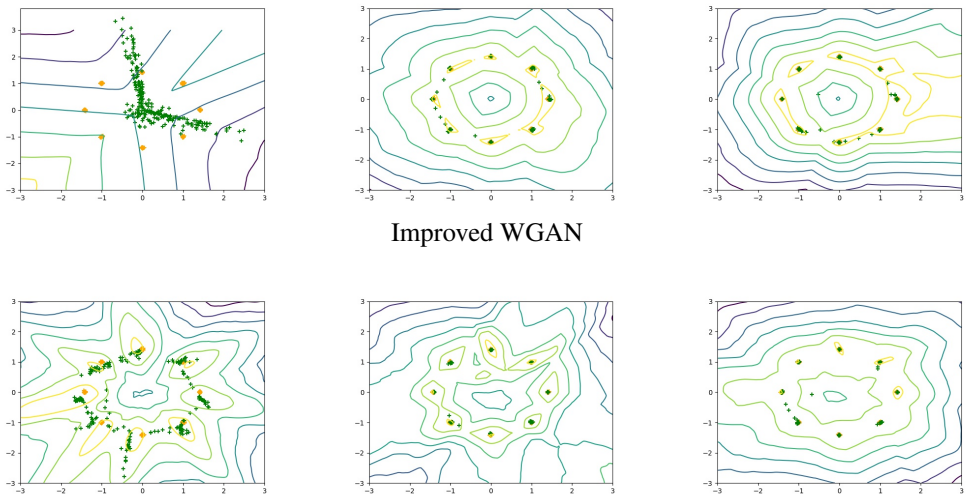
(i) Gradient penalty = 4.103e+03

(ii) Gradient penalty = 6.267e+03

Figure 7: Penalty values at two points in the training of a vanilla GAN.

5.2.3 8-GAUSSIANS EXPERIMENT

We track the performance of improved WGAN and our algorithm on the 8-Gaussians dataset over time. As seen in Figure 8, both of them approximately converge to the real density but notice that in case of improved WGAN, the discriminator function is more constrained throughout the training period. .



Improved WGAN

Our algorithm

Figure 8: Comparing the performance of different algorithms on 8-Gaussians dataset. Orange is real samples, green is generated samples and the contour plot represents $D_{\boldsymbol{\theta}}(x)$ function

5.2.4 ROBUSTNESS ACROSS DCGAN ARCHITECTURE VARIATIONS

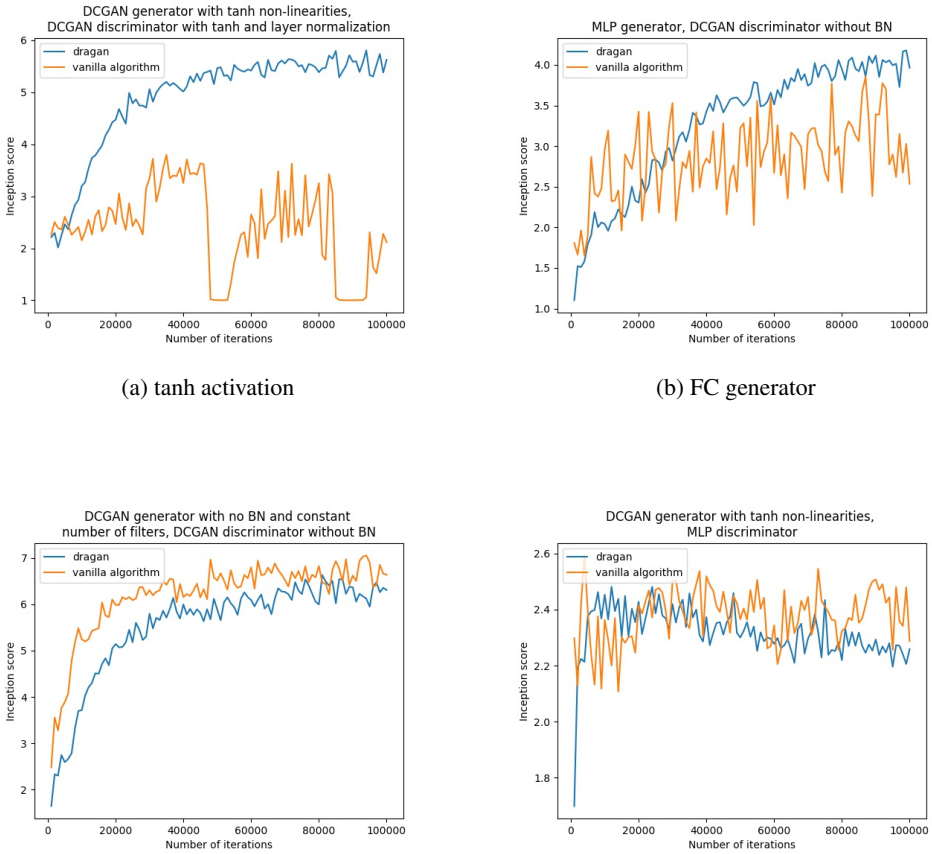
DCGANs have to be designed following specific guidelines to make them stable Radford et al. (2015). We restate the suggested rules here.

1. Use all convolutional networks which learn their own spatial downsampling (discriminator) or upsampling (generator)
2. Remove fully connected hidden layers for deeper architectures
3. Use batch normalization in both the generator and the discriminator
4. Use ReLU activation in the generator for all layers except the output layer, which uses $tanh$

5. Use LeakyReLU activation in the discriminator for all layers

We show that such constraints can be relaxed for our algorithm and hence, practitioners are free to choose from a more diverse set of architectures. Below, we present a series of experiments in which we remove different stabilizing components from DCGAN architecture and analyze the performance of our algorithm. In each case, our algorithm is stable while the vanilla procedure fails. A similar approach is used to establish the robustness of training procedures in Arjovsky et al. (2017); Gulrajani et al. (2017). Note that we add layer normalization to the discriminator (in our architecture experiments) in place of batch normalization. We chose the following four architectures which are difficult to train (in each case, we start with base DCGAN architecture and apply the changes) -

- No BN and a constant number of filters in the generator
- 4-layer 512-dim ReLU MLP generator
- $tanh$ nonlinearities everywhere
- $tanh$ nonlinearity in the generator and 4-layer 512-dim LeakyReLU MLP discriminator



(a) tanh activation

(b) FC generator

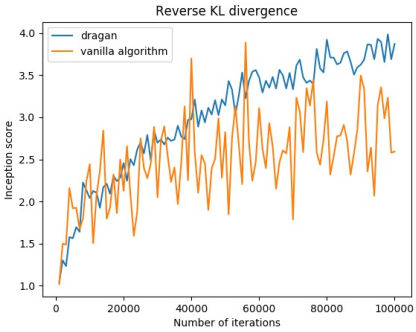
(c) Constant filter

(d) FC discriminator

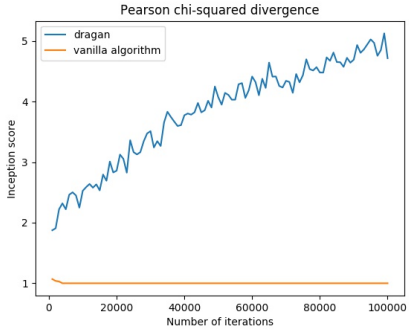
Figure 9: Inception performance on variations of DCGAN architectures.

5.2.5 Robustness across Divergence Measures (Complete)

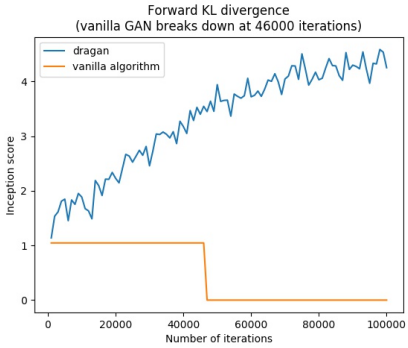
Due to space limitations, we only showed two examples for varying the divergence measures. Below we show results for all five measures.



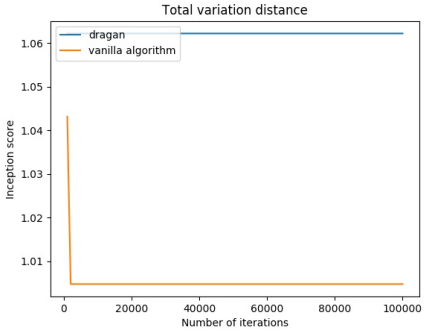
(a) Reverse KL



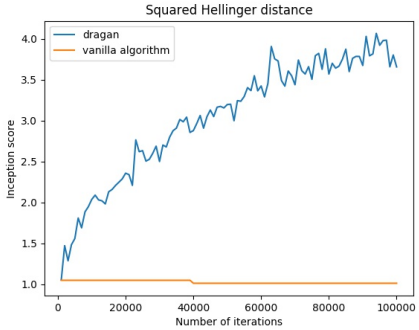
(b) Pearson χ^2



(c) Forward KL



(d) Total Variation



(e) Squared Hellinger

Figure 10: Inception score curves for five divergence measures, demonstrating superior stability for our algorithm in all but one case.

5.3 BogoNet Details

First, we choose from three families of architectures with probabilities - DCGAN (0.6), ResNet (0.2), MLP (0.2). Next, we further parameterize each family to create additional variation. For instance, the DCGAN family can result in networks with or without batch normalization, have LeakyReLU or Tanh non-linearities. The number and width of filters, latent space dimensionality are some other variations we chose from. Similarly, the number of layers and hidden units in each layer for MLP

are chosen randomly. For ResNets, we chose their depth randomly. This creates a set of hard games which test the stability of a given training algorithm.

Bounty splits examples -

DRAGAN-Vanilla order A - (5, 0), B - (3.5, 1.5), C – (2.25, 2.75), D – (2, 3)