ASSISTING THE ADVERSARY TO IMPROVE GAN TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a method for improved training of generative adversarial networks (GANs). Some of the most popular methods for improving the stability and performance of GANs involve constraining or regularizing the discriminator. Our method, on the other hand, involves regularizing the generator. It can be used alongside existing approaches to GAN training and is simple and straightforward to implement. Our method is motivated by a common mismatch between theoretical analysis and practice: analysis often assumes that the discriminator reaches its optimum on each iteration. In practice, this is essentially never true, often leading to poor gradient estimates for the generator. To address this, we introduce the *Adversary's Assistant* (AdvAs). It is a theoretically motivated penalty imposed on the generator based on the norm of the gradients used to train the discriminator. This encourages the generator to move towards points where the discriminator is optimal. We demonstrate the effect of applying AdvAs to several GAN objectives, datasets and network architectures. The results indicate a reduction in the mismatch between theory and practice and that AdvAs can lead to improvement of GAN training, as measured by FID scores.

1 INTRODUCTION

The generative adversarial network (GAN) framework (Goodfellow et al., 2014) trains a neural network known as a *generator* which maps from a random vector to an output such as an image. Key to training is another neural network, the *adversary* (sometimes called a discriminator or critic), which is trained to distinguish between "true" and generated data. This is done by *maximizing* one of the many different objectives proposed in the literature; see for instance Goodfellow et al. (2014); Arjovsky et al. (2017); Nowozin et al. (2016). The generator directly competes against the adversary: it is trained to *minimize* the same objective, which it does by making the generated data more similar to the true data. GANs are efficient to sample from, requiring a single pass through a deep network, and highly flexible, as they do not require an explicit likelihood. They are especially suited to producing photo-realistic images (Zhou et al., 2019) compared to competing methods like normalizing flows, which impose strict requirements on the neural network architecture (Kobyzev et al., 2020; Rezende & Mohamed, 2015) and VAEs (Kingma & Welling, 2014; Razavi et al., 2019; Vahdat & Kautz, 2020). Counterbalancing their appealing properties, GANs can have unstable training dynamics (Kurach et al., 2019; Goodfellow, 2017; Kodali et al., 2017; Mescheder et al., 2018).

Substantial research effort has been directed towards improving the training of GANs. These endeavors can generally be divided into two camps, albeit with significant overlap. The first develops better learning objectives for the generator/adversary to minimize/maximize. These are designed to have properties which improve training (Arjovsky et al., 2017; Li et al., 2017; Nowozin et al., 2016). The other camp develops techniques to regularize the adversary and improve its training dynamics (Kodali et al., 2017; Roth et al., 2017; Miyato et al., 2018). The adversary can then provide a better learning signal for the generator. Despite these contributions, stabilizing the training of GANs remains unsolved and continues to be an active research area.

We take an overlooked approach by training the generator in a way that accounts for the fact that the adversary is not trained to convergence. Specifically, we introduce a regularizer to the generator's loss that encourages it to move towards points where the adversary's current parameters are optimal.

Informally, the regularizer "assists" the adversary. For this reason we name our method the *Adversary's Assistant* (AdvAs). Importantly, AdvAs does not affect the solution to the minimax game, but only the training dynamics. That is, it does not change the fact that the distribution of generated data will converge to the distribution of true data, under standard assumptions.

AdvAs is hyperparameter-free and straightforward to implement. Furthermore, we release a library which makes it simple to integrate into existing code. We demonstrate its application to a standard architecture with the WGAN-GP objective (Arjovsky et al., 2017; Gulrajani et al., 2017); the state-of-the-art StyleGAN2 architecture and objective introduced by Karras et al. (2020); and the AutoGAN architecture and objective introduced by Gong et al. (2019). We test these on the MNIST (Lecun et al., 1998), CelebA (Liu et al., 2015), CIFAR10 (Krizhevsky et al., 2009) datasets respectively. We show that AdvAs improves training on all datasets, as measured by the Fréchet Inception Distance (FID) (Heusel et al., 2017), and the inception score (Salimans et al., 2016) where applicable.

2 BACKGROUND

A generator is a neural network $g : \mathcal{Z} \to \mathcal{X} \subseteq \mathbb{R}^{d_x}$ which maps from a random vector $\boldsymbol{z} \in \mathcal{Z}$ to an output $\boldsymbol{x} \in \mathcal{X}$ (e.g., an image). Due to the distribution over $\boldsymbol{z}$, the function g induces a distribution over its output $\boldsymbol{x} = g(\boldsymbol{z})$. If g is invertible and differentiable, the probability density function (PDF) over $\boldsymbol{x}$ from this "change of variables" could be computed. This is not necessary for training GANs, meaning that no such restrictions need to be placed on the neural network g. We denote this distribution $p_\theta(\boldsymbol{x})$ where $\theta \in \Theta \subseteq \mathbb{R}^{d_g}$ denotes the generator's parameters. The GAN is trained on a dataset $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, where each $\boldsymbol{x}_i$ is in $\mathcal{X}$. We assume that this is sampled i.i.d. from a data-generating distribution p_{true}. Then the aim of training is to learn θ so that p_θ is as close as possible to p_{true}. Section 2.1 will make precise what is meant by "close."

The adversary $a_\phi : \mathcal{X} \to \mathcal{A}$ has parameters $\phi \in \Phi \subseteq \mathbb{R}^{d_a}$ which are typically trained alternately with the generator. It receives as input either the data or the generator's outputs. The set that it maps to, $\mathcal{A}$, is dependent on the GAN type. For example, Goodfellow et al. (2014) define an adversary which maps from $x \in \mathcal{X}$ to the probability that x is a "real" data point from the dataset, as opposed to a "fake" from the generator. They therefore choose $\mathcal{A}$ to be $[0, 1]$ and train the adversary by maximizing the associated log-likelihood objective,

$$h(p_\theta, a_\phi) = \mathbb{E}_{x \sim p_{true}} \left[\log a_\phi(x)\right] - \mathbb{E}_{x \sim p_\theta} \left[\log(1 - a_\phi(x))\right]. \tag{1}$$

Using the intuition that the generator should generate samples that seem real and therefore "fool" the adversary, the generator is trained to minimize $h(p_\theta, a_\phi)$. Since we find θ to minimize this objective while fitting ϕ to maximize it, training a GAN is equivalent to solving the minimax game,

$$\min_\theta \max_\phi h(p_\theta, a_\phi). \tag{2}$$

Eq. (1) gives the original form for $h(p_\theta, a_\phi)$ used by Goodfellow et al. (2014) but this form varies between different GANs, as we will discuss in Section 2.1. The minimization and maximization in Eq. (2) are performed with gradient descent in practice. To be precise, we define $L_{\text{gen}}(\theta, \phi) = h(p_\theta, a_\phi)$ and $L_{\text{adv}}(\theta, \phi) = -h(p_\theta, a_\phi)$. These are treated as losses for the generator and adversary respectively, and both are minimized. In other words, we turn the maximization of $h(p_\theta, a_\phi)$ w.r.t. ϕ into a minimization of $L_{\text{adv}}(\theta, \phi)$. Then on each iteration, θ and ϕ are updated one after the other using gradient descent steps along their respective gradients:

$$\boldsymbol{\nabla}_\theta L_{\text{gen}}(\theta, \phi) = \boldsymbol{\nabla}_\theta h(p_\theta, a_\phi), \tag{3}$$

$$\boldsymbol{\nabla}_\phi L_{\text{adv}}(\theta, \phi) = -\boldsymbol{\nabla}_\phi h(p_\theta, a_\phi). \tag{4}$$

2.1 GANS MINIMIZE DIVERGENCES

A common theme in the GAN literature is analysis based on what we call the *optimal adversary assumption*. This is the assumption that, before each generator update, we have found the adversary a_ϕ which maximizes $h(p_\theta, a_\phi)$ given the current value of θ. To be precise, we define a class of permissible adversary functions $\mathcal{F}$. This is often simply the space of all functions mapping $\mathcal{X} \to \mathcal{A}$ (Goodfellow et al., 2014), but is in some GAN variants constrained by, e.g., a Lipschitz constant (Arjovsky et al., 2017). Then we call the adversary a_ϕ optimal for a particular value of θ if and only if $h(p_\theta, a_\phi) = \max_{a \in \mathcal{F}} h(p_\theta, a)$.

In practice, the neural network a_ϕ cannot represent every $a \in \mathcal{F}$ and so it may not be able to parameterize an optimal adversary for a given θ. As is common in the literature, we assume that the neural network is expressive enough that this is not an issue, i.e., we assume that for any θ, there exists at least one $\phi \in \Phi$ resulting in an optimal adversary. Then, noting that there may be multiple such $\phi \in \Phi$, we define $\Phi^*(\theta)$ to be the set of all optimal adversary parameters. That is, $\Phi^*(\theta) = \{\phi \in \Phi \mid h(p_\theta, a_\phi) = \max_{a \in \mathcal{F}} h(p_\theta, a)\}$ and the optimal adversary assumptions says that before each update of θ we have found $\phi \in \Phi^*(\theta)$. We emphasize that, in part due to the limited number of gradient updates performed on ϕ, this assumption is essentially never true in practice. This paper presents a method to improve the training of GANs by addressing this issue.

The optimal adversary assumption simplifies analysis of GAN training considerably. Instead of being a two-player game, it turns into a case of minimizing an objective with respect to θ alone. We denote this objective

$$\mathcal{M}(p_\theta) = \max_{a \in \mathcal{F}} h(p_\theta, a) = h(p_\theta, a_{\phi^*}) \quad \text{where} \quad \phi^* \in \Phi^*(\theta). \tag{5}$$

For example, Goodfellow et al. (2014) showed that using the objective presented in Eq. (1) results in $\mathcal{M}(p_\theta) = 2 \cdot \text{JSD}(p_{\text{true}} || p_\theta) - \log 4$, where JSD is the Jensen-Shannon divergence. By making the optimal adversary assumption, they could prove that their GAN training procedure would converge, and would minimize the Jensen-Shannon divergence between p_{true} and p_θ.

A spate of research following the introduction of the original GAN objective has similarly made use of the optimal adversary assumption to propose GANs which minimize different divergences. For example, Wasserstein GANs (WGANs) (Arjovsky et al., 2017) minimize a Wasserstein distance. MMD GANs (Li et al., 2017) minimize a distance known as the maximum mean discrepancy. Nowozin et al. (2016) introduce f-GANs which minimize f-divergences, a class including the Kullback-Leibler and Jensen-Shannon divergences. We emphasize that this is by no means an exhaustive list. Like these studies, this paper is motivated by the perspective that, under the optimal adversary assumption, GANs minimize a divergence. However, the GAN framework can also be viewed from a more game-theoretic perspective (Kodali et al., 2017; Grnarova et al., 2018).

3 Does an optimal adversary lead to optimal gradients?

As introduced above, the training of an adversary does not need to be considered in any analysis if it is simply assumed to always be optimal. From this perspective, the goal of training GANs can be seen as learning the generator to minimize $\mathcal{M}(p_\theta)$. This leads to the question: assuming that we have an optimal adversary, can we compute the gradient required for the generator update, $\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta)$? To clarify, assume that we have generator parameters θ', and have found $\phi^* \in \Phi^*(\theta')$ such that $h(p_{\theta'}, a_{\phi^*})$ and $\mathcal{M}(p_{\theta'})$ are equal in value. We then want to take a gradient step on θ' to minimize $\mathcal{M}(p_{\theta'})$. Virtually all GAN methods do this by assuming that $\mathcal{M}(p_{\theta'})$ and $h(p_{\theta'}, a_{\phi^*})$ have equal gradients with respect to θ at θ'. That is, it is assumed that $\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) \mid_{\theta=\theta'}$ is equal to the *partial derivative*[1] $D_1 h(p_\theta, a_{\phi^*}) \mid_{\theta=\theta'}$. It is not immediately obvious that this is true.

In the GAN literature this concern has largely been overlooked, with a few treatments for specific GAN types, see e.g. Arjovsky et al. (2017); Goodfellow et al. (2014). In particular, Arjovsky et al. (2017) invoke (but do not explicitly prove) an extension of Theorem 1 in Milgrom & Segal (2002) to prove that the Wasserstein GAN has optimal gradients if the adversary is optimal, i.e. $D_1 h(p_\theta, a_{\phi^*}) \mid_{\theta=\theta'} = \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) \mid_{\theta=\theta'}$. We note that this extension can, in fact, be used to prove that GANs in general have this property under fairly weak assumptions:

Theorem 1. *Let $\mathcal{M}(p_\theta) = h(p_\theta, a_{\phi^*})$ for any $\phi^* \in \Phi^*(\theta)$, as defined in Eq.* (5)*. Assuming that $\mathcal{M}(p_\theta)$ is differentiable w.r.t. θ and $h(p_\theta, a_\phi)$ is differentiable w.r.t. θ for all $\phi \in \Phi^*(\theta)$, then if $\phi^* \in \Phi^*(\theta)$ we have*

$$\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) = D_1 h(p_\theta, a_{\phi^*}). \tag{6}$$

See Appendix D.1 for a proof. We emphasize Theorem 1 applies only if the adversary is optimal. If this is not the case we cannot quantify, and so cannot directly minimize or account for, the discrepancy between $\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta)$ and $D_1 h(p_\theta, a_{\phi^*})$. Instead of attempting to do so, our method drives the parameters towards regions where $\phi^* \in \Phi^*(\theta)$ so that Theorem 1 can be invoked.

[1] We use $D_1 h(p_\theta, a_\phi)$ to denote the partial derivative of $h(p_\theta, a_\phi)$ with respect to θ with ϕ kept constant. Similarly, we will use $D_2 h(p_\theta, a_\phi)$ to denote the derivative of $h(p_\theta, a_\phi)$ with respect to ϕ, with θ held constant.

3.1 ADVERSARY CONSTRUCTORS

To see how we may impose the constraint that Eq. (6) is true, we consider a trivial relationship between any generator and the corresponding optimal adversary. If an optimal adversary exists for every $\theta \in \Theta$ then there exists some, possibly non-unique, function $f : \Theta \to \Phi$ that maps from any generator to a corresponding optimal adversary. That is, for all $\theta \in \Theta$, $f(\theta) = \phi^* \in \Phi^*(\theta)$ in which case $h(p_\theta, a_{f(\theta)}) = \max_{a \in \mathcal{F}} h(p_\theta, a)$. We refer to such a function as an *adversary constructor*. In an ideal scenario, we could compute the output of an adversary constructor, $f(\theta)$, for any θ. We could then invoke Theorem 1 and the generator could be updated with the gradient $\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) = D_1 h(p_\theta, a_{f(\theta)})$. In practice, computing $f(\theta)$ is infeasible and we can only approximate the optimal adversary parameters with gradient descent. There is therefore a mismatch between GAN theory, where Theorem 1 is often invoked, and practice, where the conditions to invoke it are essentially never satisfied. How then, can we address this problem? We look to the adversary constructors, which provide a condition that must be satisfied for the optimal adversary assumption to be true. Adversary constructors allow us to account for the influence of θ on ϕ by considering the total derivative $\boldsymbol{\nabla}_\theta h(p_\theta, a_{f(\theta)})$. We prove in Appendix D.2 that a comparison with the result of Theorem 1 leads to Corollary 1. In the next section, we motivate AdvAs as an attempt fulfill a condition suggested by this corollary.

Corollary 1. *Let $f : \Theta \to \Phi$ be a differentiable mapping such that for all $\theta \in \Theta$, $\mathcal{M}(p_\theta) = h(p_\theta, a_{f(\theta)})$. If the conditions in Theorem 1 are satisfied and the Jacobian matrix of f with respect to θ, $\boldsymbol{J}_\theta(f)$ exists for all $\theta \in \Theta$ then*

$$D_2 h(p_\theta, a_{f(\theta)})^{\mathrm{T}} \boldsymbol{J}_\theta(f) = 0. \tag{7}$$

4 ASSISTING THE ADVERSARY

Corollary 1 tells us that $D_2 h(p_\theta, a_\phi)^{\mathrm{T}} \boldsymbol{J}_\theta(f)$ will be zero whenever Theorem 1 can be invoked. This makes Eq. (7) a necessary, but not sufficient, condition for the invocation of Theorem 1. This suggests that the magnitude of $D_2 h(p_\theta, a_\phi)^{\mathrm{T}} \boldsymbol{J}_\theta(f)$ could be a measure of how "close" $D_1 h(p_\theta, a_{\phi^*})$ is to the desired gradient $\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta)$. However, the Jacobian $\boldsymbol{J}_\theta(f)$ is not tractable so $D_2 h(p_\theta, a_\phi)^{\mathrm{T}} \boldsymbol{J}_\theta(f)$ cannot be computed. The only term we can calculate in practice is $D_2 h(p_\theta, a_\phi)$, exactly the gradient used to train the adversary. If $D_2 h(p_\theta, a_\phi)$ is zero, then $D_2 h(p_\theta, a_\phi)^{\mathrm{T}} \boldsymbol{J}_\theta(f)$ is zero. The magnitude of $D_2 h(p_\theta, a_\phi)$ could therefore be an approximate measure of "closeness" instead of $D_2 h(p_\theta, a_\phi)^{\mathrm{T}} \boldsymbol{J}_\theta(f)$. This leads to our proposed augmented generator loss, which regularizes generator updates to reduce the magnitude of $D_2 h(p_\theta, a_\phi)$. It has a scalar hyperparameter $\lambda \geq 0$, but Section 4.3 provides a heuristic which can remove the need to set this hyperparameter.

$$L_{\mathrm{gen}}^{\mathrm{AdvAs}}(\theta, \phi) = L_{\mathrm{gen}}(\theta, \phi) + \lambda r(\theta, \phi), \tag{8}$$

with

$$r(\theta, \phi) = \left\| \boldsymbol{\nabla}_\phi L_{\mathrm{adv}}(\theta, \phi) \right\|_2^2, \tag{9}$$

recalling that $\boldsymbol{\nabla}_\phi L_{\mathrm{adv}}(\theta, \phi) = -D_2 h(p_\theta, a_\phi)$. We emphasize that while $r(\theta, \phi)$ appear similar to the gradient penalties found in previous work (Kodali et al., 2017; Roth et al., 2017; Gulrajani et al., 2017), $r(\theta, \phi)$ differs fundamentally in two ways: (1) it regularizes the generator and not the adversary and (2) the gradient is with respect to the adversary parameters as opposed to the adversary's input.

Figuratively, AdvAs changes the generator updates to move in a conservative direction that does not over-exploit the adversary's sub-optimality. Consider the generator and adversary as two players attempting to out-maneuver one another. From Eq. (2), we see that the generator should learn to counteract the best possible adversary, rather than the current adversary. If the current adversary is sub-optimal, allowing it to catch up would yield better updates to the generator. One way to achieve this is to update the generator in a way that helps make the current adversary optimal. This behavior is exactly what AdvAs encourages. In this sense, it assists the adversary, leading to it's name, the Adversary's Assistant. We emphasize that AdvAs is a small modification to existing GAN training algorithms, but for completeness we include pseudocode in Appendix A.

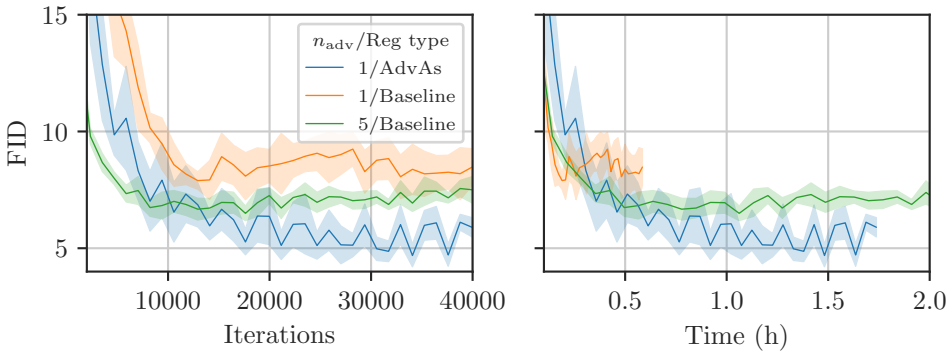

Figure 1: FID scores throughout training for the WGAN-GP objective on MNIST, estimated using 60 000 samples from the generator. We plot up to a maximum of 40 000 iterations. When plotting against time (right), this means some lines end before the two hours we show. The blue line shows the results with AdvAs, while the others are baselines with different values of n_{adv}.

4.1 AdvAs preserves convergence results

AdvAs has several desirable properties which support its use as a regularizer: (1) it does not interfere with the update on ϕ, and recall that perfectly optimizing ϕ leads to $h(p_\theta, a_\phi) = \mathcal{M}(p_\theta)$. (2) Under mild conditions, $\boldsymbol{\nabla}_\phi r(\theta; \phi) \mid_{\phi=\phi^*}$ is zero for an optimal ϕ^* and so $\boldsymbol{\nabla}_\theta L_{\text{gen}}^{\text{AdvAs}}(\theta, \phi^*) = \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta)$. These properties imply that, under the optimal adversary assumption, optimizing $L_{\text{gen}}^{\text{AdvAs}}$ is in fact equivalent to optimizing L_{gen}. See Appendix D.4 for a proof. Therefore any convergence analysis which relies on the optimal adversary assumption is equally applicable when AdvAs is included in the loss. Regarding the mild conditions in property (2), we require that ϕ^* be a stationary point of $h(p_\theta, a_\phi)$. This is true as long as h is differentiable w.r.t. ϕ at (θ, ϕ^*) and ϕ^* does not lie on a boundary of Φ. The optimal adversary parameters, ϕ^*, cannot lie on a boundary unless, for example, weight clipping is used as in Arjovsky et al. (2017). In such cases, we cannot speak to the efficacy of applying AdvAs.

We make the additional observation that for some GAN objectives, minimizing $r(\theta, \phi)$ alone (as opposed to L_{gen} or $L_{\text{gen}}^{\text{AdvAs}}$) may match p_θ and p_{true}. We show this in Appendix D.3 for the WGAN objective (Arjovsky et al., 2017). In particular, for all $\phi \in \Phi$, $r(\theta, \phi)$ is zero and at a global minimum whenever $p_\theta = p_{\text{true}}$. Experimental results in Appendix C.3 support this observation. However, the results appear worse than those obtained by optimizing either L_{gen} or $L_{\text{gen}}^{\text{AdvAs}}$.

4.2 Estimating the AdvAs loss

It is not always possible, and seldom computationally feasible, to compute the AdvAs regularization term $r(\theta, \phi)$ exactly. We instead use a stochastic estimate. This is computed by simply estimating the gradient $\boldsymbol{\nabla}_\phi L_{\text{adv}}(\theta, \phi)$ with a minibatch and then taking the squared L2-norm of this gradient estimate. That is, defining $\tilde{L}_{\text{adv}}(\theta, \phi)$ as an unbiased estimate of the adversary's loss, we estimate $r(\theta, \phi)$ with

$$\tilde{r}(\theta, \phi) = \left\| \boldsymbol{\nabla}_\phi \tilde{L}_{\text{adv}}(\theta, \phi) \right\|_2^2 . \tag{10}$$

Although the gradient estimate is unbiased, taking the norm results in a biased estimate of $r(\theta, \phi)$. However, comparisons with a more computationally expensive unbiased estimate[2] did not reveal a significant difference in performance.

4.3 Removing the hyperparameter λ

Eq. (8) introduces a hyperparameter, λ, which we would prefer not to perform a grid-search on. Setting λ to be too great can destabilize training. Conversely, setting it to be too small gives similar

[2]Computing an unbiased estimate involves estimating the variance of the gradient across the minibatch. This is not currently efficiently supported by, for example, PyTorch. See Appendix B for details.

results to not using AdvAs. We therefore introduce a heuristic which can be used to avoid setting the hyperparameter. Our experiments suggests that this is often a good choice, although manually tuning λ may yield greater gains. This heuristic involves considering the magnitudes of three gradients, and so we first define the notation,

$$\begin{aligned} \boldsymbol{g}_{\text{orig}}(\theta, \phi) &= \boldsymbol{\nabla}_\theta L_{\text{gen}}(\theta, \phi), \\ \boldsymbol{g}_{\text{AdvAs}}(\theta, \phi) &= \boldsymbol{\nabla}_\theta \tilde{r}(\theta, \phi), \\ \boldsymbol{g}_{\text{total}}(\theta, \phi, \lambda) &= \boldsymbol{\nabla}_\theta L_{\text{gen}}^{\text{AdvAs}}(\theta, \phi) \\ &= \boldsymbol{g}_{\text{orig}}(\theta, \phi) + \lambda \boldsymbol{g}_{\text{AdvAs}}(\theta, \phi). \end{aligned}$$

The heuristic can be interpreted as choosing λ at each iteration to prevent the total gradient, $\boldsymbol{g}_{\text{total}}(\theta, \phi, \lambda)$, from being dominated by the AdvAs term. Specifically, we ensure the magnitude of $\lambda \boldsymbol{g}_{\text{AdvAs}}(\theta, \phi)$ is less than or equal to the magnitude of $\boldsymbol{g}_{\text{orig}}$ by setting

$$\lambda = \min\left(1, \frac{\|\boldsymbol{g}_{\text{orig}}(\theta, \phi)\|_2}{\|\boldsymbol{g}_{\text{AdvAs}}(\theta, \phi)\|_2}\right) \tag{11}$$

at every iteration. We then perform gradient descent along $\boldsymbol{g}_{\text{total}}(\theta, \phi, \lambda)$. This technique ensures that λ is bounded above by 1.

Table 1: FID and IS scores on CIFAR10 using AutoGAN with and without AdvAs.

	IS $\pm\sigma$	FID $\pm\sigma$
AdvAs ($n_{\text{adv}} = 2$)	8.4 ± 0.1	14.5 ± 1.0
Baseline ($n_{\text{adv}} = 5$)	8.3 ± 0.1	15.0 ± 0.7

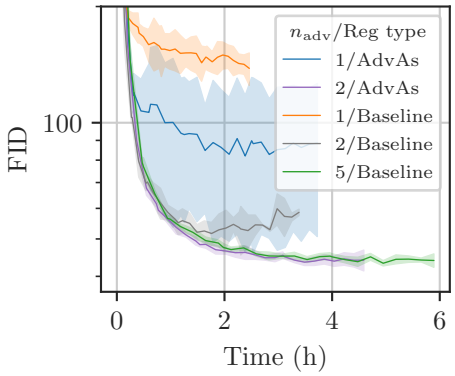

Figure 2: FID scores on CIFAR10 using AutoGAN from baselines and AdvAs plotted with a log y-axis against running time for different values of n_{adv}. We see that AdvAs with $n_{\text{adv}} = 2$ yields the lowest FID scores at every point during training.

5 Experiments

We demonstrate the effect of incorporating AdvAs into GAN training using several GAN architectures, objectives, and datasets. In each case, we compare GANs trained with AdvAs with baselines that do not use AdvAs but are otherwise identical. We first demonstrate the use of AdvAs in conjunction with the WGAN-GP objective (Gulrajani et al., 2017) to model MNIST (Lecun et al., 1998). In this experiment, we compare the performance gains achieved by AdvAs to a reasonable upper bound on the gains achievable with this type of regularization. We further support these findings with experiments on CIFAR10 (Krizhevsky et al., 2009) using AutoGAN (Gong et al., 2019), an architecture found through neural architecture search. We then demonstrate that AdvAs can improve training on larger images using StyleGAN2 (Karras et al., 2020) on CelebA (Liu et al., 2015). We quantify each network's progress throughout training using the FID score (Heusel et al., 2017). Since AdvAs increases the computation time per iteration, we plot training progress against time for each experiment. We also present inception scores (IS) (Salimans et al., 2016) where applicable. We estimate scores in each case with 5 random seeds and report the standard deviation (σ) as a measure of uncertainty.

AdvAs aims to improve performance by coming closer to having an optimal adversary. Another common way to achieve this is to use a larger number of adversary updates (n_{adv}) before each generator update. For each experiment, we show baselines with the value of n_{adv} suggested in the literature. Noting that the computational complexity is $\mathcal{O}(n_{\text{adv}})$ and so keeping n_{adv} low is desirable, we find that AdvAs can work well with lower values of n_{adv} than the baseline. For a fair comparison, we also report baselines trained with these values of n_{adv}.

For MNIST and CelebA, we avoid setting the hyperparameter λ by using the heuristic proposed in Section 4.3. We found for CIFAR10 that manually tuning λ gave better performance, and so set $\lambda = 0.01$. Additionally, on MNIST and CelebA, the methods we consider use regularization in the form of a gradient penalty (Gulrajani et al., 2017) for training the adversary. This is equivalent to including a regularization term $\gamma_{\text{adv}}(\phi)$ in the definition of L_{adv}. That is, $L_{\text{adv}}(\theta, \phi) = -\boldsymbol{\nabla}_\phi h(p_\theta, a_\phi) + \gamma_{\text{adv}}(\phi)$. Following Eq. (9) this regularization term is included in the AdvAs term $r(\theta, \phi)$. Another practical detail is that AutoGAN and StyleGAN2 are trained with a hinge loss (Lim & Ye, 2017).

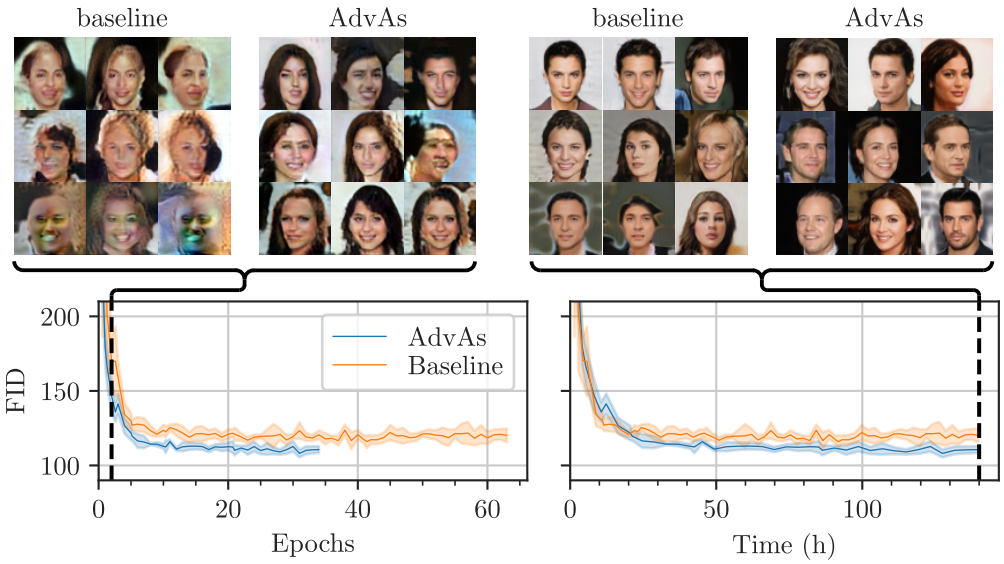

Figure 3: **Bottom:** FID scores throughout training estimated with 1000 samples, plotted against number of epochs (left) and training time (right). FID scores for AdvAs decrease more on each iteration at the start of training and converge to be 7.5% lower. **Top:** The left two columns show uncurated samples with and without AdvAs after 2 epochs. The rightmost two columns show uncurated samples from networks at the end of training. In each grid of images, each row is generated by a network with a different training seed and shows 3 images generated by passing a different random vector through this network. AdvAs leads to obvious qualitative improvement early in training.

That is, when computing the adversary's loss $L_{\text{adv}}(\theta, \phi)$, its output $a_\phi(\boldsymbol{x})$ is truncated to be below $+1$ for real images, or above -1 for generated images. This prevents it from receiving gradient feedback when its predictions are both accurate and confident. Intuitively, this stops its outputs becoming too large and damaging the generator's training. However, this truncation is not present when updating the generator. This means that the generator minimizes a different objective to the one maximized by the adversary, and so it is not exactly a minimax game. It is not clear that it is beneficial to calculate the AdvAs regularization term using this truncation. We found that better performance was obtained by computing $r(\theta, \phi)$ without truncation, and do this in the reported experiments.

5.1 WGAN-GP on MNIST

We use a simple neural architecture: the generator consists of a fully-connected layer followed by two transposed convolutions. The adversary has three convolutional layers. Both use instance normalization (Ulyanov et al., 2017) and ReLU non-linearities; see Appendix E for details. We compare using AdvAs with $n_{\text{adv}} = 1$ against the baseline for $n_{\text{adv}} \in \{1, 5\}$ where $n_{\text{adv}} = 5$ is suggested by Gulrajani et al. (2017). Fig. 1 shows the FID scores for each method throughout training. We see that using AdvAs with $n_{\text{adv}} = 1$ leads to better performance on convergence; even compared to the baseline with $n_{\text{adv}} = 5$, the best FID score reached is improved by 28%.

5.2 AutoGAN on CIFAR10

We next experiment on the generation of CIFAR10 (Krizhevsky et al., 2009) images. We use AutoGAN (Gong et al., 2019), which has a generator architecture optimized for CIFAR10 using neural architecture search. It is trained with a hinge loss, as described previously, an exponential moving average of generator weights, and typically uses $n_{\text{adv}} = 5$. Figure 2 shows FID scores throughout training for various values of n_{adv}, with and without AdvAs, each computed with 1000 samples. Table 1 shows FID scores at the end of training for the best performing value of n_{adv} for each method, estimated with 50 000 samples. For a fixed n_{adv} of either 1 or 2, using AdvAs improves the FID score. In fact, with $n_{\text{adv}} = 2$, the performance with AdvAs is indistinguishable from the baseline with the suggested setting of $n_{\text{adv}} = 5$. Unlike for MNIST, AdvAs does not outperform the baseline with high enough n_{adv}. We hypothesize that this is because, with an architecture highly optimized

for $n_{\mathrm{adv}} = 5$, the adversary is closer to being optimal when trained with $n_{\mathrm{adv}} = 5$. Assuming this is the case, we would not expect AdvAs to improve training compared to a baseline with sufficient n_{adv}. Still, our results show that applying AdvAs allows the same performance with a lower n_{adv}.

5.3 StyleGAN2 on CelebA

To demonstrate that AdvAs improves state-of-the-art GAN architectures and training procedures, we consider StyleGAN2 (Karras et al., 2020). We train this as proposed by Karras et al. (2020) with a WGAN-like objective with gradient penalty (Gulrajani et al., 2017), an exponential moving average of the generator weights, and various forms of regularization including path length, R1, and style-mixing regularization. More detail on these can be found in Karras et al. (2020), but we merely wish to emphasize that considerable effort has been put into tuning this training procedure. For this reason, we do not attempt to further tune n_{adv}, which is 1 by default. Any improvements from applying AdvAs indicate a beneficial effect not provided by other forms of regularization used.

Figure 3 compares the training of StyleGAN2 on CelebA at 64×64 resolution with and without the AdvAs regularizer. Using AdvAs has two main effects: (1) the generated images show bigger improvements per epoch at the start of training; and (2) the final FID score is improved by 7.5%. Even accounting for its greater time per iteration, the FID scores achieved by AdvAs overtake the baseline after one day of training. We verify that the baseline performance is similar to that reported by Zhou et al. (2019) with a similar architecture.

6 Related work

We proposed a method motivated by fulfilling the optimal adversary assumption. In this sense, it is similar to a large body of work aiming to improve and stabilize GAN training by better training the adversary. We differ fundamentally due to our focus on the training the generator rather than the adversary. This prior work generally affects the discriminator in one of two broad ways: weight constraints and gradient penalties Brock et al. (2019). Weight normalization involves directly manipulating the parameters of the adversary, such as through weight clipping (Arjovsky et al., 2017) or spectral normalization (Miyato et al., 2018). Gradient penalties (Kodali et al., 2017; Roth et al., 2017; Gulrajani et al., 2017) impose soft constraints on the gradients of the adversary's output with respect to its input. Various forms exist with different motivations; see Mescheder et al. (2018) for a summary and analysis. Our approach, AdvAs, may appear similar to a gradient penalty, as it operates on gradients of the adversary. However, the gradients are w.r.t. the adversary's parameters rather than its input. Furthermore, AdvAs is added to the generator's loss and not the adversary's.

Regularizing generator updates has recently received more attention in the literature (Chu et al., 2020; Gulrajani et al., 2017). In particular, Chu et al. (2020) show theoretically that the effectiveness of different forms of regularization for both the generator and adversary is linked to the smoothness of the objective function. They present a set of conditions on the generator and adversary that ensure a smooth objective function, which they argue will stabilize GAN training. Their results partly explain the success of different regularization approaches for the adversary. However, they leave the imposition of the required regularization on the generator to future work. Although our approach is not motivated by creating a smooth objective, it continues this line of work by regularizing the training of the generator.

7 Discussion and conclusions

We have introduced the Adversary's Assistant, a novel method for improved GAN training. AdvAs addresses the mismatch between theory, where the adversary is assumed to be trained to optimality, and practice, where this is never the case. We show improved training across three datasets, architectures, and GAN objectives, indicating that it successfully reduces this disparity. This can lead to substantial improvements in final performance. We note that, while applying AdvAs in preliminary experiments with BEGAN (Berthelot et al., 2017) and LSGAN (Mao et al., 2017), we did not observe either a significant positive effect, or a significant negative effect other than the increased time per iteration. Nevertheless, AdvAs is simple to apply and will, in many cases, improve both training speed and final performance.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, July 2017.

David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv:1703.10717 [cs, stat]*, May 2017.

Dimitri P Bertsekas. *Nonlinear Programming*, volume 48 of *Journal of the Operational Research Society*. Taylor & Francis, 1997.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv:1809.11096 [cs, stat]*, February 2019.

Casey Chu, Kentaro Minami, and Kenji Fukumizu. Smoothness and stability in gans. In *Eighth International Conference on Learning Representations*, April 2020.

Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3224–3234, 2019.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv:1701.00160 [cs]*, April 2017. Comment: v2-v4 are all typo fixes. No substantive changes relative to v1.

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.

Paulina Grnarova, Kfir Y. Levy, Aurelien Lucchi, Thomas Hofmann, and Andreas Krause. An online learning approach to generative adversarial networks. In *International Conference on Learning Representations*, February 2018.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5767–5777. Curran Associates, Inc., 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6626–6637. Curran Associates, Inc., 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. February 2015.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, February 2018.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119, 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114 [cs, stat]*, May 2014.

Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020. ISSN 1939-3539. doi: 10.1109/TPAMI.2020.2992934.

Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv:1705.07215 [cs]*, December 2017. Comment: Analysis of convergence and mode collapse by studying GAN training process as regret minimization. Some new results.

Alex Krizhevsky, Geoffrey Hinton, et al. *Learning Multiple Layers of Features from Tiny Images*. Citeseer, 2009.

Karol Kurach, Mario Lučić, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. A large-scale study on regularization and normalization in gans. In *International Conference on Machine Learning*, pp. 3581–3590. PMLR, May 2019.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi: 10.1109/5.726791.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabas Poczos. Mmd gan: Towards deeper understanding of moment matching network. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2203–2213. Curran Associates, Inc., 2017.

Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pp. 1718–1727, 2015.

Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv:1705.02894 [cond-mat, stat]*, May 2017.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 3730–3738, December 2015. doi: 10.1109/ICCV.2015.425.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2017.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning*, pp. 3481–3490. PMLR, July 2018.

Paul Milgrom and Ilya Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2): 583–601, 2002. ISSN 1468-0262. doi: 10.1111/1468-0262.00296.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, February 2018.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. F-gan: Training generative neural samplers using variational divergence minimization. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 271–279. Curran Associates, Inc., 2016.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 14866–14876. Curran Associates, Inc., 2019.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538. PMLR, June 2015.

Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2018–2028. Curran Associates, Inc., 2017.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 2234–2242. Curran Associates, Inc., 2016.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022 [cs]*, November 2017.

Thomas Unterthiner, Bernhard Nessler, Calvin Seward, Günter Klambauer, Martin Heusel, Hubert Ramsauer, and Sepp Hochreiter. Coulomb gans: Provably optimal nash equilibria via potential fields. *arXiv preprint arXiv:1708.08819*, 2017.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv:2007.03898 [cs, stat]*, July 2020. Comment: Some images are downsized to meet arXiv requirements. Check https://arash-vahdat.github.io/NVAE_arxiv.pdf for a high-resolution version (24 MB).

Sharon Zhou, Mitchell Gordon, Ranjay Krishna, Austin Narcomey, Li F Fei-Fei, and Michael Bernstein. Hype: A benchmark for human eye perceptual evaluation of generative models. In *Advances in Neural Information Processing Systems*, pp. 3449–3461, 2019.