On Stabilizing Generative Adversarial Training with Noise

Simon Jenni Paolo Favaro
University of Bern
{simon.jenni,paolo.favaro}@inf.unibe.ch

Abstract

We present a novel method and analysis to train generative adversarial networks (GAN) in a stable manner. As shown in recent analysis, training is often undermined by the probability distribution of the data being zero on neighborhoods of the data space. We notice that the distributions of real and generated data should match even when they undergo the same filtering. Therefore, to address the limited support problem we propose to train GANs by using different filtered versions of the real and generated data distributions. In this way, filtering does not prevent the exact matching of the data distribution, while helping training by extending the support of both distributions. As filtering we consider adding samples from an arbitrary distribution to the data, which corresponds to a convolution of the data distribution with the arbitrary one. We also propose to learn the generation of these samples so as to challenge the discriminator in the adversarial training. We show that our approach results in a stable and well-behaved training of even the original minimax GAN formulation. Moreover, our technique can be incorporated in most modern GAN formulations and leads to a consistent improvement on several common datasets.

1. Introduction

Since the seminal work of [6], generative adversarial networks (GAN) have been widely used and analyzed due to the quality of the samples that they produce, in particular when applied to the space of natural images. Unfortunately, GANs still prove difficult to train. In fact, a vanilla implementation does not converge to a high-quality sample generator and heuristics used to improve the generator often exhibit an unstable behavior. This has led to a substantial work to better understand GANs (see, for instance, [23, 19, 1]). In particular, [1] points out how the unstable training of GANs is due to the (limited and low-dimensional) support of the data and model distributions. In the original GAN formulation, the generator is trained against a discriminator in a minimax optimization problem. The discriminator learns to distinguish real from fake samples, while the generator learns to generate fake samples that can fool the discriminator. When the support of the data and model distributions is disjoint, the generator stops improving as soon as the discriminator achieves perfect classification, because this prevents the propagation of useful information to the generator through gradient descent (see Fig. 1a).

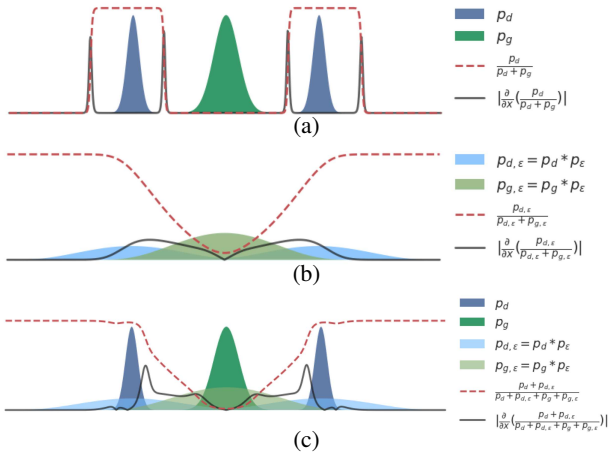


Figure 1: (a) When the probability density functions of the real p_d and generated data p_g do not overlap, then the discriminator can easily distinguish samples. The gradient of the discriminator with respect to its input is zero in these regions and this prevents any further improvement of the generator. (b) Adding samples from an arbitrary p_ϵ to those of the real and the generated data results in the filtered versions $p_d * p_\epsilon$ and $p_g * p_\epsilon$. Because the supports of the filtered distributions overlap, the gradient of the discriminator is not zero and the generator can improve. However, the high-frequency content of the original distributions is missing. (c) By varying p_ϵ, the generator can learn to match the data distribution accurately thanks to the extended supports.

The recent work by [1] proposes to extend the support of the distributions by adding noise to both generated and real images before they are fed as input to the discriminator. This procedure results in a smoothing of both data

and model probability distributions, which indeed increases their support extent (see Fig. 1b). For simplicity, let us assume that the probability density function of the data is well defined and let us denote it with p_d. Then, samples $\tilde{x} = x + \epsilon$, obtained by adding noise $\epsilon \sim p_\epsilon$ to the data samples $x \sim p_d$, are also instances of the probability density function $p_{d,\epsilon} = p_\epsilon * p_d$, where $*$ denotes the convolution operator. The support of $p_{d,\epsilon}$ is the Minkowski sum of the supports of p_ϵ and p_d and thus larger than the support of p_d. Similarly, adding noise to the samples from the generator probability density p_g leads to the smoothed probability density $p_{g,\epsilon} = p_\epsilon * p_g$. Adding noise is a quite well-known technique that has been used in maximum likelihood methods, but is considered undesirable as it yields approximate generative models that produce low-quality blurry samples. Indeed, most formulations with additive noise boil down to finding the model distribution p_g that best solves $p_{d,\epsilon} = p_{g,\epsilon}$. However, this usually results in a low quality estimate p_g because $p_d * p_\epsilon$ has lost the high frequency content of p_d. An immediate solution is to use a form of noise annealing, where the noise variance is initially high and is then reduced gradually during the iterations so that the original distributions, rather than the smooth ones, are eventually matched. This results in an improved training, but as the noise variance approaches zero, the optimization problem converges to the original formulation and the algorithm may be subject to the usual unstable behavior.

In this work, we design a novel adversarial training procedure that is stable and yields accurate results. We show that under some general assumptions it is possible to modify both the data and generated probability densities with additional noise without affecting the optimality conditions of the original noise-free formulation. As an alternative to the original formulation, with $z \sim \mathcal{N}(0, I_d)$ and $x \sim p_d$,

$$\min_G \max_D \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z)))], \quad (1)$$

where D denotes the discriminator, we propose to train a generative model G by solving instead the following optimization

$$\min_G \max_D \sum_{p_\epsilon \in \mathcal{S}} \mathbb{E}_{\epsilon \sim p_\epsilon} \left[\mathbb{E}_{x \sim p_d}[\log D(x + \epsilon)]\right] + \mathbb{E}_{\epsilon \sim p_\epsilon} \left[\mathbb{E}_{z \sim \mathcal{N}(0, I_d)}[\log(1 - D(G(z) + \epsilon))]\right], \quad (2)$$

where we introduced a set $\mathcal{S}$ of probability density functions. If we solve the innermost optimization problem in Problem (2), then we obtain the optimal discriminator

$$D(x) = \frac{\sum_{p_\epsilon \in \mathcal{S}} p_{d,\epsilon}(x)}{\sum_{p_\epsilon \in \mathcal{S}} p_{d,\epsilon}(x) + p_{g,\epsilon}(x)}, \quad (3)$$

where we have defined p_g as the probability density of $G(z)$, where $z \sim \mathcal{N}(0, I_d)$. If we substitute this in the problem above and simplify we have

$$\min_G \text{JSD}\left(\frac{1}{|\mathcal{S}|}\sum_{p_\epsilon \in \mathcal{S}} p_{d,\epsilon}, \frac{1}{|\mathcal{S}|}\sum_{p_\epsilon \in \mathcal{S}} p_{g,\epsilon}\right), \quad (4)$$

where JSD is the Jensen-Shannon divergence. We show that, under suitable assumptions, the optimal solution of Problem (4) is unique and $p_g = p_d$. Moreover, since $1/|\mathcal{S}| \sum_{p_\epsilon \in \mathcal{S}} p_{d,\epsilon}$ enjoys a larger support than p_d, the optimization via iterative methods based on gradient descent is more likely to achieve the global minimum, regardless of the support of p_d. Thus, our formulation enjoys the following properties: 1) It defines a fitting of probability densities that is not affected by their support; 2) It guarantees the exact matching of the data probability density function; 3) It can be easily applied to other GAN formulations. A simplified scheme of the proposed approach is shown in Fig. 2.

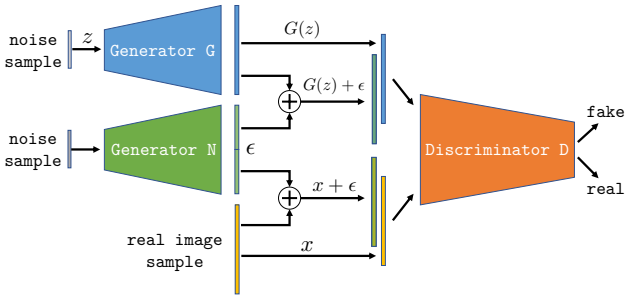


Figure 2: Simplified scheme of the proposed GAN training. We also show a noise generator N that is explained in detail in Section 3.1. The discriminator D needs to distinguish both noise-free and noisy real samples from fake ones.

In the next sections we introduce our analysis in detail and then devise a computationally feasible approximation of the problem formulation (2). Our method is evaluated quantitatively on CIFAR-10 [12], STL-10 [5], and CelebA [15], and qualitatively on ImageNet [20] and LSUN bedrooms [24].

2. Related Work

The inherent instability of GAN training was first addressed through a set of techniques and heuristics [22] and careful architectural design choices and hyper-parameter tuning [18]. [22] proposes the use of one-sided label smoothing and the injection of Gaussian noise into the layers of the discriminator. A theoretical analysis of the unstable training and the vanishing gradients phenomena was introduced by Arjovsky *et al.* [1]. They argue that the main source of instability stems from the fact that the real and the generated distributions have disjoint supports or lie on low-dimensional manifolds. In the case of an optimal discriminator this will result in zero gradients that then stop the

training of the generator. More importantly, they also provide a way to avoid such difficulties by introducing noise and considering "softer" metrics such as the Wasserstein distance. [23] makes similar observations and also proposed the use of "instance noise" which is gradually reduced during training as a way to overcome these issues. Another recent work stabilizes GAN training in a similar way by transforming examples before feeding them to the discriminator [21]. The amount of transformation is then gradually reduced during training. They only transform the real examples, in contrast to [23], [1] and our work. [2] builds on the work of [1] and introduces the Wasserstein GAN (WGAN). The WGAN optimizes an integral probability metric that is the dual to the Wasserstein distance. This formulation requires the discriminator to be Lipschitz-continuous, which is realized through weight-clipping. [7] presents a better way to enforce the Lipschitz constraint via a gradient penalty over interpolations between real and generated data (WGAN-GP). [19] introduces a stabilizing regularizer based on a gradient norm penalty similar to that by [7]. Its formulation however is in terms of f-divergences and is derived via an analytic approximation of adversarial training with additive Gaussian noise on the datapoints. Another recent GAN regularization technique that bounds the Lipschitz constant of the discriminator is the spectral normalization introduced by [17]. This method demonstrates state-of-the-art in terms of robustness in adversarial training. Several alternative loss functions and GAN models have been proposed over the years, claiming superior stability and sample quality over the original GAN (e.g., [16], [25], [3], [2], [25], [11]). Adversarial noise generation has previously been used in the context of classification to improve the robustness against adversarial perturbations [13].

3. Matching Filtered Distributions

We are interested in finding a formulation that yields as optimal generator G a sampler of the data probability density function (pdf) p_d, which we assume is well defined. The main difficulty in dealing with p_d is that it may be zero on some neighborhood in the data space. An iterative optimization of Problem (1) based on gradient descent may yield a degenerate solution, *i.e.*, such that the model pdf p_g only partially overlaps with p_d (a scenario called *mode collapse*). It has been noticed that adding samples of an arbitrary distribution to both real and fake data samples during training helps reduce this issue. In fact, adding samples $\epsilon \sim p_\epsilon$ corresponds to blurring the original pdfs p_d and p_g, an operation that is known to increase their support and thus their likelihood to overlap. This increased overlap means that iterative methods can exploit useful gradient directions at more locations and are then more likely to converge to the global solution. By building on this observation, we propose to solve instead Problem (2) and look for a way to increase the support of the data pdf p_d without losing the optimality conditions of the original formulation of Problem (1).

Our result below proves that this is the case for some choices of the additive noise. We consider images of $m \times n$ pixels and with values in a compact domain $\Omega \subset \mathbb{R}^{m \times n}$, since image intensities are bounded from above and below. Then, also the support of the pdf p_d is bounded and contained in Ω. This implies that p_d is also $L^2(\Omega)$.

Theorem 1. *Let us choose $\mathcal{S}$ such that Problem (4) can be written as*

$$\min_{p_g} JSD\left(\frac{1}{2}(p_d + p_d * p_\epsilon), \frac{1}{2}(p_g + p_g * p_\epsilon)\right), \quad (5)$$

where p_ϵ is a zero-mean Gaussian function with a positive definite covariance Σ. Let us also assume that the domain of p_g is restricted to Ω (and thus $p_g \in L^2(\Omega)$). Then, the global optimum of Problem (5) is $p_g(x) = p_d(x)$, $\forall x \in \Omega$.

Proof. The global minimum of the Jensens-Shannon divergence is achieved if and only if

$$p_d + p_d * p_\epsilon = p_g + p_g * p_\epsilon. \quad (6)$$

Let $p_g = p_d + \Delta$. Then, we have $\int |\Delta(x)|^2 dx < \infty$. By substituting p_g in eq. (6) we obtain $\Delta * p_\epsilon = -\Delta$. Since Δ and p_ϵ are in $L^2(\Omega)$, we can take the Fourier transform of both sides, compute their absolute value and obtain

$$\left|\hat{\Delta}(\omega)\right| |\hat{p}_\epsilon(\omega)| = \left|\hat{\Delta}(\omega)\right|, \quad \forall \omega \in \hat{\Omega}. \quad (7)$$

Because p_g and p_d integrate to 1, $\int \Delta(x)dx = 0$ and $\hat{\Delta}(0) = 0$. Suppose $\exists \omega \neq 0$ such that $\hat{\Delta}(\omega) \neq 0$. Since p_ϵ is Gaussian, $|\hat{p}_\epsilon(\omega)| = \left|e^{-\frac{1}{2}\omega^\top \Sigma^{-1} \omega}\right| < 1$, which contradicts the optimality condition (7). Thus, $\Delta(x) = 0, \forall x \in \Omega$ and we can conclude that $p_g(x) = p_d(x), \forall x \in \Omega$. □

3.1. Formulation

Based on the above theorem we consider two cases:

1. **Gaussian noise** with a fixed/learned standard deviation σ: $p_\epsilon(\epsilon) = \mathcal{N}(\epsilon; 0, \sigma I_d)$;
2. **Learned noise** from a noise generator network N with parameters σ: $p_\epsilon(\epsilon)$ such that $\epsilon = N(w, \sigma)$, with $w \sim \mathcal{N}(0, I_d)$.

In both configurations we can learn the parameter(s) σ. We do so by minimizing the cost function after the maximization with respect to the discriminator. The minimization encourages large noise since this would make $p_{d,\epsilon}(\omega)$ more similar to $p_{g,\epsilon}(\omega)$ regardless of p_d and p_g. This would not be very useful to gradient descent. Therefore, to limit the noise

Algorithm 1: Distribution Filtering GAN (DFGAN)

Input: Training set $\mathcal{D} \sim p_d$, number of discriminator updates n_{disc}, number of training iterations N, batch-size m, learning rate α, noise penalty λ

Output: Generator parameters θ

Initialize generator parameters θ, discriminator parameters ϕ and noise-generator parameters ω ;

for $1 \dots N$ **do**

  for $1 \dots n_{disc}$ **do**

    Sample $\{x_1, \dots, x_m\} \sim p_d$, $\{\tilde{x}_1, \dots, \tilde{x}_m\} \sim p_g$ and $\{\epsilon_1, \dots, \epsilon_m\} \sim p_\epsilon$;

    $L_D^r = \sum_{i=1}^m \ln(D(x_i)) + \ln(D(x_i + \epsilon_i))$;

    $L_D^f = \sum_{i=1}^m \ln(1 - D(\tilde{x}_i)) + \ln(1 - D(\tilde{x}_i + \epsilon_i))$;

    $L_\epsilon = \sum_{i=1}^m |\epsilon_i|^2$;

    $\phi \leftarrow \phi + \nabla_\phi L_D^r(\phi, \omega) + \nabla_\phi L_D^f(\phi, \omega)$;

    $\omega \leftarrow \omega - \nabla_\omega \big(L_D^r(\phi, \omega) + L_D^f(\phi, \omega) + \lambda L_\epsilon(\omega)\big)$;

  end

  Sample $\{\tilde{x}_1, \dots, \tilde{x}_m\} \sim p_g$ and $\{\epsilon_1, \dots, \epsilon_m\} \sim p_\epsilon$;

  $L_G^f = \sum_{i=1}^m \ln(D(\tilde{x}_i)) + \ln(D(\tilde{x}_i + \epsilon_i))$;

  $\theta \leftarrow \theta + \nabla_\theta L_G^f(\theta)$;

end

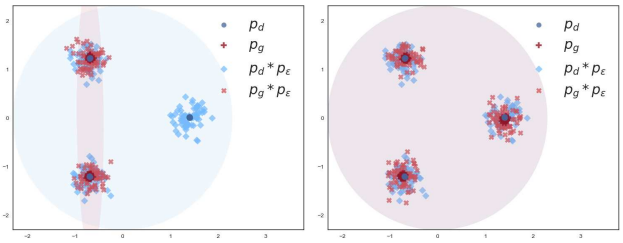
magnitude we introduce as a regularization term the noise variance $\Gamma(\sigma) = \sigma^2$ or the Euclidean norm of the noise output image $\Gamma(\sigma) = \mathbb{E}_{w \sim \mathcal{N}(0, I_d)} |N(w, \sigma)|^2$, and multiply it by a positive scalar λ, which we tune.

The proposed formulations can then be written in a unified way as:

$$\min_G \min_\sigma \max_D \lambda\Gamma + \mathbb{E}_x \Big[\log D(x) + \mathbb{E}_\epsilon \log D(x + \epsilon) \Big] + \mathbb{E}_z \Big[\log[1 - D(G(z))] + \mathbb{E}_\epsilon \log[1 - D(G(z) + \epsilon)] \Big]. \tag{8}$$

3.2. Implementation

Implementing our algorithm only requires a few minor modifications of the standard GAN framework. We perform the update for the noise-generator and the discriminator in the same iteration. Mini-batches for the discriminator are formed by collecting all the fake and real samples in two separate batches, *i.e.*, $\{x_1, \dots, x_m, x_1 + \epsilon_1, \dots, x_m + \epsilon_m\}$ is the batch with real examples and $\{\tilde{x}_1, \dots, \tilde{x}_m, \tilde{x}_1 + \epsilon_1, \dots, \tilde{x}_m + \epsilon_m\}$ the fake examples batch. The complete procedure is outlined in Algorithm 1. The noise-generator architecture is typically the same as the generator, but with a reduced number of convolutional filters. Since the inputs to the discriminator are doubled when compared to the standard GAN framework, the DFGAN framework can be 1.5 to 2 times slower. Similar and more severe performance drops are present in existing variants (*e.g.*, WGAN-GP). Note that by constructing the batches as $\{x_1, \dots, x_{m/2}, x_{m/2+1} + \epsilon_1, \dots, x_m + \epsilon_m\}$ the training time is instead comparable to the standard framework, but it is much more stable and yields an accurate generator. For a comparison of the runtimes, see Fig. 4.



(a) (b)

Figure 3: Illustration of how separate normalization of fake and real mini-batches discourages mode collapse. In (a) no normalization is applied and mode collapse is observed. Since the covered modes are indistinguishable, the generator receives no signal that encourages better mode coverage. In (b) separate normalization of the real and fake data is applied. The mismatch in the batch statistics (mean and standard deviation) can now be detected by the discriminator, forcing the generator to improve.

3.3. Batch-Normalization and Mode Collapse

The current best practice is to apply batch normalization to the discriminator separately on the real and fake mini-batches [4]. Indeed, this showed much better results when compared to feeding mini-batches with a 50/50 mix of real and fake examples in our experiments. The reason for this is that batch normalization implicitly takes into account the distribution of examples in each mini-batch. To see this, consider the example in Fig. 3. In the case of no separate normalization of fake and real batches we can observe mode-collapse. The modes covered by the generator are indistinguishable for the discriminator, which observes each example independently. There is no signal to the generator that leads to better mode coverage in this case. Since the first two moments of the fake and real batch distribution are clearly not matching, a separate normalization will help the discriminator distinguish between real and fake examples and therefore encourage better mode coverage by the generator.

Using batch normalization in this way turns out to be crucial for our method as well. Indeed, when no batch normalization is used in the discriminator, the generator will often tend to produce noisy examples. This is difficult to detect by the discriminator, since it judges each example independently. To mitigate this issue we apply separate normalization of the noisy real and fake examples before feeding them to the discriminator. We use this technique for models without batch normalization (*e.g.* SNGAN).

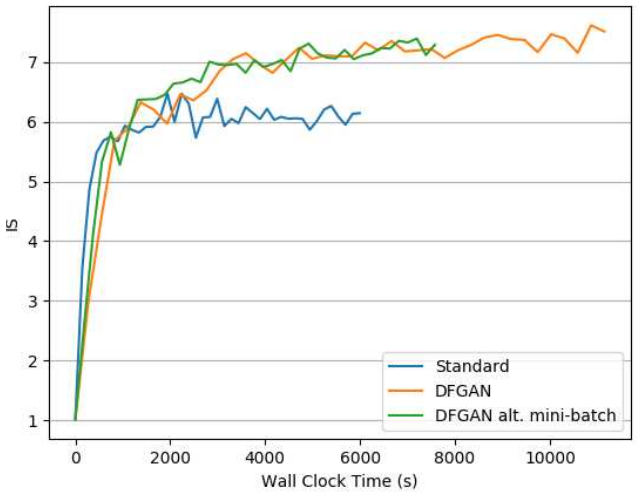


Figure 4: A comparison of wall clock time vs IS for GANs with and without distribution filtering. The models use the architecture specified in Table 1 and were trained on CIFAR-10. The computational overhead introduced by our method does not negatively affect the speed of convergence.

4. Experiments

We compare and evaluate our model using two common GAN metrics: the Inception score IS [22] and the Fréchet Inception distance FID [8]. Throughout this section we use 10K generated and real samples to compute IS and FID. In order to get a measure of the stability of the training we report the mean and standard deviation of the last five checkpoints for both metrics (obtained in the last 10% of training). More reconstructions, experiments and details are provided in the supplementary material.

4.1. Ablations

To verify our model we perform ablation experiments on two common image datasets: CIFAR-10 [12] and STL-10 [5]. For CIFAR-10 we train on the 50K 32×32 RGB training images and for STL-10 we resize the 100K 96×96 training images to 64×64. The network architectures resemble the DCGAN architectures of [18] and are detailed in Table 1. All the models are trained for 100K generator iterations using a mini-batch size of 64. We use the ADAM optimizer [10] with a learning rate of 10^{-4} and $\beta_1 = 0.5$. Results on the following ablations are reported in Table 2:

(a)-(c) Only noisy samples: In this set of experiments we only feed noisy examples to the discriminator. In experiment **(a)** we add Gaussian noise and in **(b)** we add learned noise. In both cases the noise level is not annealed. While this leads to stable training, the resulting samples are of poor quality which is reflected by high FID and low IS. The generator will tend to also produce noisy samples since there is no incentive to remove the noise. Annealing the added noise during training as proposed by [1] and [23] leads to an improvement over the standard GAN. This is demonstrated in experiment **(c)**. The added Gaussian noise is linearly annealed during the 100K iterations in this case;

(d)-(i) Both noisy and clean samples: The second set of experiments consists of variants of our proposed model. Experiments **(d)** and **(e)** use a simple Gaussian noise model; in **(e)** the standard deviation of the noise σ is learned. We observe a drastic improvement in the quality of the generated examples even with this simple modification. The other experiments show results of our full model with a separate noise-generator network. We vary the weight λ of the L^2 norm of the noise in experiments **(f)-(h)**. Ablation **(i)** uses the alternative mini-batch construction with faster runtime as described in Section 3.2;

Table 1: Network architectures used for experiments on CIFAR-10 and STL-10. Images are assumed to be of size 32×32 for CIFAR-10 and 64×64 for STL-10. We set $M = 512$ for CIFAR-10 and $M = 1024$ for STL-10. Layers in parentheses are only included for STL-10. The noise-generator network follows the generator architecture with the number of channels reduced by a factor of 8. BN indicates the use of batch-normalization [9].

Generator CIFAR-10/(STL-10)	**Discriminator CIFAR-10/(STL-10)**
$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$	conv 3×3 str.=1 lReLU 64
fully-conn. BN ReLU $4 \times 4 \times M$	conv 4×4 str.=2 BN lReLU 64
(deconv 4×4 str.=2 BN ReLU 512)	conv 4×4 str.=2 BN lReLU 128
deconv 4×4 str.=2 BN ReLU 256	conv 4×4 str.=2 BN lReLU 256
deconv 4×4 str.=2 BN ReLU 128	conv 4×4 str.=2 BN lReLU 512
deconv 4×4 str.=2 BN ReLU 64	(conv 4×4 str.=2 BN lReLU 1024)
deconv 3×3 str.=1 tanh 3	fully-connected sigmoid 1

Application to Different GAN Models. We investigate the possibility of applying our proposed training method to several standard GAN models. The network architectures are the same as proposed in the original works with only the necessary adjustments to the given image-resolutions of the datasets (*i.e.*, truncation of the network architectures). The only exception is SVM-GAN, where we use the architecture in Table 1. Note that for the GAN with minimax loss (MM-GAN) and WGAN-GP we use the architecture of DCGAN. Hyper-parameters are kept at their default values for each model. The models are evaluated on two common GAN benchmarks: CIFAR-10 [12] and CelebA [15]. The image resolution is 32×32 for CIFAR-10 and 64×64 for CelebA. All models are trained for 100K generator iterations. For the alternative objective function of LSGAN and

Table 2: We perform ablation experiments on CIFAR-10 and STL-10 to demonstrate the effectiveness of our proposed algorithm. Experiments (a)-(c) show results where only filtered examples are fed to the discriminator. Experiment (c) corresponds to previously proposed noise-annealing and results in an improvement over the standard GAN training. Our approach of feeding both filtered and clean samples to the discriminator shows a clear improvement over the baseline.

Experiment	CIFAR-10		STL-10	
	FID	IS	FID	IS
Standard GAN	46.1 ± 0.7	$6.12 \pm .09$	78.4 ± 6.7	$8.22 \pm .37$
(a) Noise only: $\epsilon \sim \mathcal{N}(0, I)$	94.9 ± 4.9	$4.68 \pm .12$	107.9 ± 2.3	$6.48 \pm .19$
(b) Noise only: ϵ learned	69.0 ± 3.4	$5.05 \pm .14$	107.2 ± 3.4	$6.39 \pm .22$
(c) Noise only: $\epsilon \sim \mathcal{N}(0, \sigma I)$, $\sigma \to 0$	44.5 ± 3.2	$6.85 \pm .20$	75.9 ± 1.9	$8.49 \pm .19$
(d) Clean + noise: $\epsilon \sim \mathcal{N}(0, I)$	29.7 ± 0.6	$7.16 \pm .05$	66.5 ± 2.3	$8.64 \pm .17$
(e) Clean + noise: $\epsilon \sim \mathcal{N}(0, \sigma I)$ with learnt σ	28.8 ± 0.7	$7.23 \pm .14$	71.3 ± 1.7	$8.30 \pm .12$
(f) DFGAN ($\lambda = 0.1$)	27.7 ± 0.8	$7.31 \pm .06$	$\mathbf{63.9} \pm 1.7$	$\mathbf{8.81} \pm .07$
(g) DFGAN ($\lambda = 1$)	$\mathbf{26.5} \pm 0.6$	$\mathbf{7.49} \pm .04$	64.0 ± 1.4	$8.52 \pm .16$
(h) DFGAN ($\lambda = 10$)	29.8 ± 0.4	$6.55 \pm .08$	66.9 ± 3.2	$8.38 \pm .20$
(i) DFGAN alt. mini-batch ($\lambda = 1$)	28.7 ± 0.6	$7.3 \pm .05$	67.8 ± 3.2	$8.30 \pm .11$

SVM-GAN we set the loss of the noise generator to be the negative of the discriminator loss, as is the case in our standard model. The results are shown in Table 3. We can observe that applying our training method improves performance in most cases and even enables the training with the original saturation-prone minimax GAN objective, which is very unstable otherwise. Note also that applying our method to SNGAN [17] (the current state-of-the-art) leads to an improvement on both datasets. We also evaluated SNGAN with and without our method on 64×64 images of STL-10 (same as in Table 2) where our method boosts the performance from an FID of 66.3 ± 1.1 to 58.3 ± 1.4. We show random CelebA reconstructions from models trained with and without our approach in Fig. 5.

Robustness to Hyperparameters. We test the robustness of DFGANs with respect to various hyperparamters by training on CIFAR-10 with the settings listed in Table 4. The network is the same as specified in Table 1. The noise penalty term is set to $\lambda = 0.1$. We compare to a model without our training method (Standard), a model with the gradient penalty regularization proposed by [19] (GAN+GP) and a model with spectral normalization (SNGAN). To the best of our knowledge, these methods are the current state-of-the-art in terms of GAN stabilization. Fig. 7 shows that our method is stable and accurate across all settings.

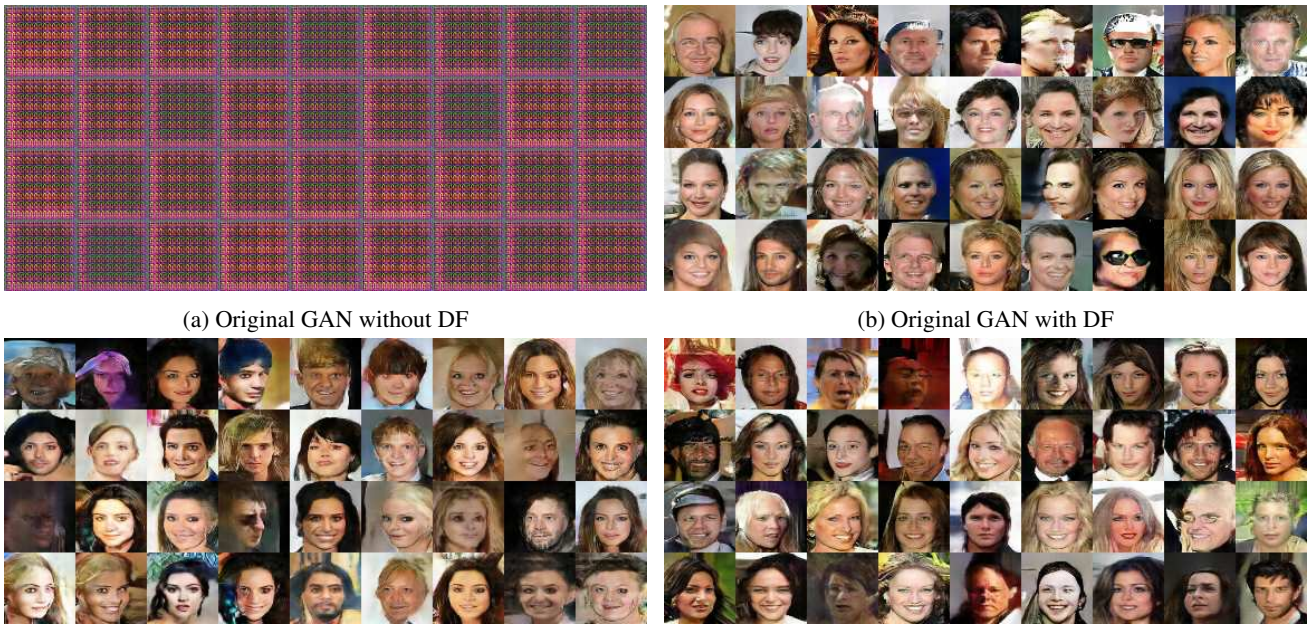
Robustness to Network Architectures. To test the robustness of DFGANs against non-optimal network architectures we modified the networks in Table 1 by doubling the number of layers in both generator and discriminator. This leads to significantly worse performance in terms of FID in all cases: 46 to 135 (Standard), 33 to 111 (SNGAN), 28 to 36 (GAN+GP), and 27 to 60 (DFGAN). However, SNGAN+DF leads to good results with a FID of 27.6.

Table 3: We apply our proposed GAN training to various previous GAN models trained on CIFAR-10 and CelebA. The same network architectures and hyperparameters as in the original works are used (for SVM-GAN we used the network in Table 1). We observe that our method increases performance in most cases even with the suggested hyperparameter settings. Note that our method also allows successful training with the original minimax MMGAN loss as opposed to the commonly used heuristic (*e.g.*, in DCGAN).

Model	CIFAR-10		CelebA
	FID	IS	FID
MMGAN [6]	> 450	~ 1	> 350
DCGAN [18]	33.4 ± 0.5	$6.73 \pm .07$	25.4 ± 2.6
WGAN-GP [7]	37.7 ± 0.4	$6.55 \pm .08$	15.5 ± 0.2
LSGAN [16]	38.7 ± 1.8	$6.73 \pm .12$	21.4 ± 1.1
SVM-GAN [14]	43.9 ± 1.0	$6.25 \pm .09$	26.5 ± 1.9
SNGAN ([17]	29.1 ± 0.4	$7.26 \pm .06$	13.2 ± 0.3
MMGAN +DF ($\lambda = 0.1$)	33.1 ± 0.7	$6.91 \pm .05$	16.6 ± 1.9
DCGAN + DF ($\lambda = 10$)	31.2 ± 0.3	$6.95 \pm .11$	14.7 ± 1.0
LSGAN + DF ($\lambda = 10$)	36.7 ± 1.2	$6.63 \pm .17$	19.9 ± 0.4
SVM-GAN + DF ($\lambda = 1$)	28.7 ± 1.1	$7.31 \pm .11$	12.7 ± 0.7
SNGAN + DF ($\lambda = 1$)	$\mathbf{25.9} \pm 0.3$	$\mathbf{7.47} \pm .08$	$\mathbf{10.5} \pm 0.4$

4.2. Qualitative Results

We trained DFGANs on 128×128 images from two large-scale datasets: ImageNet [20] and LSUN bedrooms [24]. The network architecture is similar to the one in Table 1 with one additional layer in both networks. We trained the models for 100K iterations on LSUN and 300K iter-



(a) Original GAN without DF

(b) Original GAN with DF

(c) DCGAN without DF

(d) DCGAN with DF

Figure 5: Left column: Random reconstructions from models trained on CelebA without distribution filtering (DF). Right column: Random reconstructions with our proposed method.

Figure 6: Reconstructions from DFGANs trained on 128×128 images from the LSUN bedrooms dataset (*top*) and ImageNet (*bottom*).

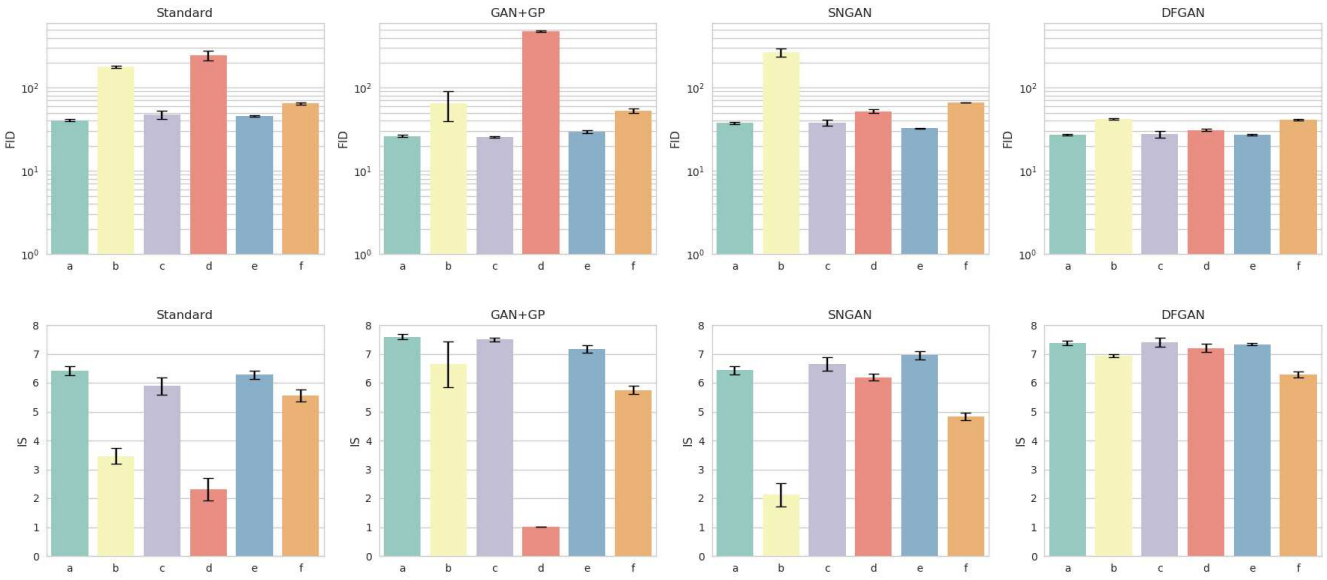


Figure 7: Results of the robustness experiments in Table 4 on CIFAR-10. We compare the standard GAN (*1st column*), a GAN with gradient penalty (*2nd column*), a GAN with spectral normalization (*3rd column*) and a GAN with our proposed method (*4th column*). Results are reported in Fréchet Inception Distance `FID` (*top*) and Inception Score `IS` (*bottom*).

Table 4: Hyperparameter settings used to evaluate the robustness of our proposed GAN training method. We vary the learning rate α, the normalization in G, the optimizer, the activation functions, the number of discriminator iterations n_{disc} and the number of training examples n_{train}.

Exp.	**LR** α	**BN in** G	**Opt.**	**ActFn**	n_{disc}	n_{train}
a)	$2 \cdot 10^{-4}$	FALSE	ADAM	(l)ReLU	1	50K
b)	$2 \cdot 10^{-4}$	TRUE	ADAM	tanh	1	50K
c)	$1 \cdot 10^{-3}$	TRUE	ADAM	(l)ReLU	1	50K
d)	$1 \cdot 10^{-2}$	TRUE	SGD	(l)ReLU	1	50K
e)	$2 \cdot 10^{-4}$	TRUE	ADAM	(l)ReLU	5	50K
f)	$2 \cdot 10^{-4}$	TRUE	ADAM	(l)ReLU	1	5K

ations on ImageNet. Random samples of the models are shown in Fig. 6. In Fig. 8 we show some examples of the noise that is produced by the noise generator at different stages during training. These examples resemble the image patterns that typically appear when the generator diverges.

5. Conclusions

We have introduced a novel method to stabilize generative adversarial training that results in accurate generative models. Our method is rather general and can be applied to other GAN formulations with an average improvement in generated sample quality and variety, and training stability. Since GAN training aims at matching probability density distributions, we add random samples to both generated and real data to extend the support of the densities and thus facilitate their matching through gradient descent. We demonstrate the proposed training method on several common datasets of real images.

Figure 8: Examples of the generated noise (top row) and corresponding noisy training examples (rows 2 to 4). The columns correspond to different iterations. The noise varies over time to continually challenge the discriminator.

Acknowledgements. This work was supported by the Swiss National Science Foundation (SNSF) grant number 200021_169622. We also wish to thank Abdelhak Lemkhenter for discussions and for help with the proof of Theorem 1.

References

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017. 1, 2, 3, 5

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017. 3

[3] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 3

[4] Soumith Chintala, Emily Denton, Martin Arjovsky, and Michael Mathieu. How to train a gan? tips and tricks to make gans work. https://github.com/soumith/ganhacks, 2016. 4

[5] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011. 2, 5

[6] 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. 1, 6

[7] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017. 3, 6

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017. 5

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 448–456. JMLR. org, 2015. 5

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[11] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. How to train your dragan. *arXiv preprint arXiv:1705.07215*, 2017. 3

[12] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 2, 5

[13] Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. Generative adversarial trainer: Defense to adversarial perturbations with gan. *arXiv preprint arXiv:1705.03387*, 2017. 3

[14] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 6

[15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. 2, 5

[16] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821. IEEE, 2017. 3, 6

[17] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 3, 6

[18] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2, 5, 6

[19] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems*, pages 2015–2025, 2017. 1, 3, 6

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 2, 6

[21] Mehdi SM Sajjadi and Bernhard Schölkopf. Tempered adversarial networks. *arXiv preprint arXiv:1802.04374*, 2018. 3

[22] 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*, pages 2234–2242, 2016. 2, 5

[23] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016. 1, 3, 5

[24] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015. 2, 6

[25] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016. 3