NOISE-TEMPERED GENERATIVE ADVERSARIAL NETWORKS

Anonymous authors
Paper under double-blind review

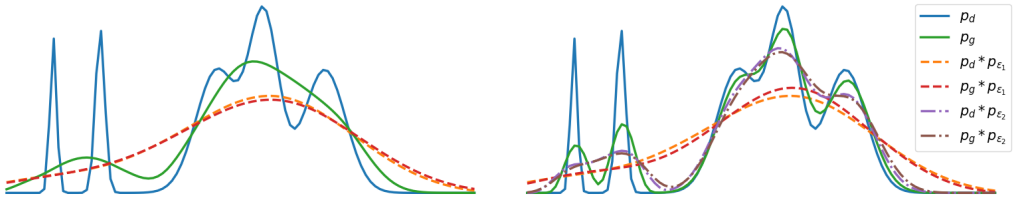
ABSTRACT

We present a novel method to stabilize the training of generative adversarial networks. The training stability is often undermined by the limited and low-dimensional support of the probability density function of the data samples. To address this problem we propose to simultaneously train the generative adversarial networks against different additive noise models, including the noise-free case. The benefits of this approach are that: 1) The case with noise added to both real and generated samples extends the support of the probability density function of the data, while not compromising the exact matching of the original data distribution, and 2) The noise-free case allows the exact matching of the original data distribution. We demonstrate our approach with both fixed additive noise and with learned noise models. 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 Goodfellow et al. (2014), 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, Sønderby et al. (2017); Roth et al. (2017); Arjovsky & Bottou (2017)). In particular, Arjovsky & Bottou (2017) point 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 or lies on a low-dimensional manifold, 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. To address this limitation, some research focused on novel formulations, as, for example, in Gulrajani et al. (2017); Mao et al. (2017) and on techniques to improve the training, such as, for example Miyato et al. (2018); Arjovsky et al. (2017). However, a recent large-scale evaluation by Lucic et al. (2017) shows that obtaining improvements over the seminal work of Goodfellow et al. (2014) is extremely challenging.

The recent work by Arjovsky & Bottou (2017) 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 and dimensionality. In fact, 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



(a) (b)

Figure 1: Illustration of uses of additive noise as a regularizer for generative models. Adding noise to samples from a probability density p_d results in a smoothing of p_d. (a) Shows that matching smooth distributions results in a low-quality probability density estimate p_g. (b) Shows that matching smooth probability densities for two noise instances results already in a much higher quality estimate p_g.

finding the model distribution p_g that best solves $p_{d,\epsilon} = p_{g,\epsilon}$. As shown in Figure 1 (a), this results in a low quality estimate p_g because $p_d * p_\epsilon$ has lost the high frequency content of p_d. Therefore, Arjovsky and Bottou propose 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 build on the findings of Arjovsky et al. (2017) to design a novel noise-based procedure that yields stable and accurate training. Our method is based on the following two observations: 1) If we add noise to the real and fake samples in a GAN training, we achieve $p_{d,\epsilon} = p_{g,\epsilon}$, i.e., $p_\epsilon * (p_d - p_g) = 0$, which is a necessary (but not sufficient) condition for $p_d = p_g$; 2) $p_{d,\epsilon} = p_{g,\epsilon}$ is a sufficient condition for $p_d = p_g$, if it holds for any p_ϵ. Based on the first observation, we propose not to anneal noise, but rather sustain it, as it does not prevent the desired solution. Based on the second observation, we propose to vary the noise probability density. In fact, as shown in Figure 1 (b) if we train a GAN on data with additive noise from two Gaussian noise instances $p_{\epsilon_1} = \mathcal{N}(0, \sigma_1 I_d)$ and $p_{\epsilon_2} = \mathcal{N}(0, \sigma_2 I_d)$, the matching of both $p_{d,\epsilon_1} = p_{g,\epsilon_1}$ and $p_{d,\epsilon_2} = p_{g,\epsilon_2}$ yields a much more accurate approximate probability density p_g. Both noise instances help the estimation of p_g: the low-noise instance preserves high-frequency content of p_d, while the high-noise instance extends the support of p_d, which then helps the training of the generator. Therefore, we suggest to obtain a generative model G by solving a multi-objective optimization

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

where we introduced a space $\mathcal{S}$ of probability density functions, and D denotes the discriminator. In fact, if we solve the innermost optimization problem, then we obtain the optimal discriminator $D(x) = {}^{p_{d,\epsilon}(x)}/_{(p_{d,\epsilon}(x) + p_{g,\epsilon}(x))}$, where we have defined $p_g(x) \triangleq \int \delta(x - G(z)) p(z) dz$, with δ the Dirac delta and $p(z) = \mathcal{N}(z; 0, I_d)$. If we substitute this in the problem above we have

$$\min_G 2\,\mathrm{JSD}(p_{d,\epsilon}, p_{g,\epsilon}) - 2\log 2, \qquad \forall p_\epsilon \in \mathcal{S} \quad (2)$$

where JSD is the Jensen-Shannon divergence. Under suitable assumptions on the space $\mathcal{S}$ and the dimensionality of z, the optimal solution of equation 2 is unique and $p_g = p_d$ regardless of the support of p_d. Thus, our formulation enjoys the following properties

- It defines a fitting of probability densities that is not affected by their support
- It promotes, rather than prevents, the exact matching of the data probability density function
- It can be easily applied to other GAN formulations.

In the next sections we introduce our analysis more in detail and then devise a computationally feasible approximation of the problem formulation equation 1. Our method is evaluated quantitatively on CIFAR-10 (Krizhevsky (2009)), STL-10 (Coates et al. (2011)), and CelebA (Liu et al. (2015)), and qualitatively on ImageNet (Russakovsky et al. (2015)) and LSUN bedrooms (Yu et al. (2015)).

2 NOISE-TEMPERED ADVERSARIAL TRAINING

2.1 MATCHING PROBABILITY DENSITY FUNCTIONS THROUGH NOISE

We are interested in finding a formulation that yields as optimal generator G a sampler of the data probability density function (pdf) p_d. The main difficulty in dealing with p_d is that its support often does not extend to the whole space where the data lives (see, for instance, Arjovsky et al. (2017)). This allows adversarial training to converge to degenerate solutions, where the model pdf p_g only partially overlaps with p_d (a scenario called *mode collapse*). It has been noticed that introducing additive noise on both real and fake data samples during training helps reduce this issue. In fact, additive noise $\epsilon \sim p_\epsilon$ corresponds to blurring the original pdfs to $p_d * p_\epsilon$ and $p_g * p_\epsilon$, an operation that is known to increase the support of both original pdfs and thus their likelihood to overlap.

However, even if the optimization of equation 2 yielded a generator G such that $p_g * p_\epsilon = p_d * p_\epsilon$, there would be no guarantee that $p_g = p_d$. As a consequence, prior work such as Arjovsky et al. (2017) starts the training of G with a large additive noise and then decreases it with iteration time. Unfortunately, as the noise variance is reduced, the model converges to the original formulation and may exhibit an unstable behavior. In this work, we notice that $p_g * p_\epsilon = p_d * p_\epsilon$ includes $p_g = p_d$ as a solution for any additive noise $\epsilon \sim p_\epsilon$. This means that the additive noise does not compromise our ability to still match p_g to the original p_d. Thus, rather than reducing the noise variance, we use the principle that matching convolutions of probability density distributions **for all possible additive noise distributions** is equivalent to matching directly the probability density distributions without additive noise. If we consider the range of image intensities Ω to be finite, then the support of the pdf p_d is bounded and contained in Ω. Then, we can prove the following:[1]

Lemma 1 *If for any $\omega \in \Omega$ and any bounded and continuous function p_ϵ*

$$(p_d * p_\epsilon)(\omega) = \int_\Omega p_d(x) p_\epsilon(\omega - x) dx = \int_\Omega p_g(x) p_\epsilon(\omega - x) dx = (p_g * p_\epsilon)(\omega), \tag{3}$$

then we have $p_d(\omega) = p_g(\omega)$ for all $\omega \in \Omega$.

Now recall the problem formulation in equation 2, which looks for the pdf p_g that minimizes the expectation of JSD$(p_{d,\epsilon}, p_{g,\epsilon})$ over $p_\epsilon \in \mathcal{S}$. The minimum value of the Jensen-Shannon divergence is 0 and it is achieved when $p_{d,\epsilon}(\omega) = p_{g,\epsilon}(\omega)$ for all $\omega \in \Omega$. The global minimum is then achieved when JSD$(p_{d,\epsilon}, p_{g,\epsilon}) = 0$ for all p_ϵ. Thus, by exploiting the above lemma, the optimal solution of equation 2 is still $p_d(\omega) = p_g(\omega)$ for all $\omega \in \Omega$, as in the original GAN formulation.

2.2 FORMULATION

There are two practical issues with the proposed formulation equation 1. One is that the multi-objective optimization requires, in principle, testing infinite pdfs. Another is that the use of constrained models (such as neural networks) for both the discriminator and the generator may prevent the optimal solution from reaching the global minimum. If we scalarize the multi-objective with some fixed weights then the optimization will only find a tradeoff across all noise distributions p_ϵ. This means that the case with no or little noise may have a small relevance compared to all other cases and this could lower the accuracy of the estimated p_g. Moreover, the case with no noise should be given the highest priority as it is the only one that ensures the exact matching between the data and model distributions. Our proposed solution is to define $\mathcal{S}$ with only two terms, the Dirac delta distribution (no noise) and a pdf with a large variance, and to scalarize the multi-objective with a random variable $\mu \sim \mathcal{U}(0, 1)$, where $\mathcal{U}(0, 1)$ is the uniform distribution between 0 and 1, which we sample at every iteration during training. This is then equivalent to

$$\min_G \max_D \mathbb{E}_{\epsilon \sim \mu\delta + (1-\mu)p_\epsilon} \Big[\mathbb{E}_{x \sim p_d}[\log D(x + \epsilon)] + \mathbb{E}_{z \sim \mathcal{N}(0, I_d)}[\log(1 - D(G(z) + \epsilon))] \Big]. \tag{4}$$

We then consider two configurations:

[1] We rely on the same principle as in Calculus of Variations (Gelfand & Fomin (1964)), whereby the necessary condition for an extremum initially written in its weak form, as the integral of the functional derivative against an arbitrary function, is then transformed into its strong form by the Fundamental Lemma of the Calculus of Variations.

Algorithm 1: Noise-Tempered GAN (NTGAN)

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 mini-batches $\{x_1, \dots, x_m\} \sim p_d$ and fake examples $\{\tilde{x}_1, \dots, \tilde{x}_m\} \sim p_g$;
    Sample noise $\{\epsilon_1, \dots, \epsilon_m\} \sim p_\epsilon$ and $\mu \sim \mathcal{U}(0, 1)$;
    $L_D^r(\phi, \omega) = \sum_{i=1}^m \mu \ln(D(x_i)) + (1 - \mu) \ln(D(x_i + \epsilon_i))$;
    $L_D^f(\phi, \omega) = \sum_{i=1}^m \mu \ln(1 - D(\tilde{x}_i)) + (1 - \mu) \ln(1 - D(\tilde{x}_i + \epsilon_i))$;
    $L_\epsilon(\omega) = \sum_{i=1}^m |\epsilon_i|^2$;
    $\phi \leftarrow \phi + \alpha \nabla_\phi (L_D^r(\phi, \omega) + L_D^f(\phi, \omega))$;
    $\omega \leftarrow \omega - \alpha \nabla_\omega \big(L_D^r(\phi, \omega) + L_D^f(\phi, \omega) + \lambda L_\epsilon(\omega)\big)$;
  end
  Sample fake examples $\{\tilde{x}_1, \dots, \tilde{x}_m\} \sim p_g$;
  Sample noise $\{\epsilon_1, \dots, \epsilon_m\} \sim p_\epsilon$ and $\mu \sim \mathcal{U}(0, 1)$;
  $L_G^f(\theta) = \sum_{i=1}^m \mu \ln(D(\tilde{x}_i)) + (1 - \mu) \ln(D(\tilde{x}_i + \epsilon_i))$;
  $\theta \leftarrow \theta + \alpha \nabla_\theta L_G^f(\theta)$;
end

1. **Gaussian noise** with a fixed/learned standard deviation σ: $\mathcal{S} = \{\delta(\epsilon), \mathcal{N}(\epsilon; 0, \sigma I_d)\}$;
2. A learned **noise generator**: $\mathcal{S} = \big\{\delta(\epsilon), \int \delta(\epsilon - N(w))p(w)dw\big\}$, with N a noise generator network (to be jointly trained) and $p(w) = \mathcal{N}(w; 0, I_d)$.

The proposed approximations result in the following problem formulations
Gaussian noise

$$\min_G \min_\sigma \max_D \mathbb{E}_{\epsilon \sim \mu\delta + (1-\mu)\mathcal{N}(0,\sigma)} \Big[\mathbb{E}_x \log D(x + \epsilon) + \mathbb{E}_z \log(1 - D(G(z) + \epsilon))\Big] + \lambda\sigma^2 \qquad (5)$$

Noise generator

$$\min_G \min_N \max_D \mathbb{E}_{\epsilon \sim \mu\delta + (1-\mu)N} \Big[\mathbb{E}_x \log D(x + \epsilon) + \mathbb{E}_z \log(1 - D(G(z) + \epsilon)) + \lambda|\epsilon|^2\Big]. \qquad (6)$$

In both formulations we may learn some optimal parameters of the noise distribution. We do so by minimizing the cost function after the maximization with respect to the discriminator. The minimization would encourage an infinite noise since this would make $p_{d,\epsilon}(\omega)$ more similar to $p_{g,\epsilon}(\omega)$ regardless of p_d and p_g. Such term however would not be very useful to the training. Therefore, to limit the noise magnitude we introduce as a regularization term the noise variance σ^2 or the Euclidean norm of the noise N output image, and multiply it by a positive scalar λ, which we tune.

2.3 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 NTGAN 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 as in the standard framework, but it is much more stable and yields an accurate generator.

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.

Generator CIFAR-10/(STL-10)
$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
fully-connected BN ReLU $4 \times 4 \times M$
(deconv 4×4 stride=2 BN ReLU 512)
deconv 4×4 stride=2 BN ReLU 256
deconv 4×4 stride=2 BN ReLU 128
deconv 4×4 stride=2 BN ReLU 64
deconv 3×3 stride=1 `tanh` 3

Discriminator CIFAR-10/(STL-10)
conv 3×3 stride=1 lReLU 64
conv 4×4 stride=2 BN lReLU 64
conv 4×4 stride=2 BN lReLU 128
conv 4×4 stride=2 BN lReLU 256
conv 4×4 stride=2 BN lReLU 512
(conv 4×4 stride=2 BN lReLU 1024)
fully-connected `sigmoid` 1

3 EXPERIMENTS

We compare and evaluate our model using two common GAN metrics: the Inception score `IS` (Salimans et al. (2016)) and the Fréchet Inception distance `FID` (Heusel et al. (2017)). 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 appendix.

Ablations. To verify our model we perform ablation experiments on two common image datasets: CIFAR-10 (Krizhevsky (2009)) and STL-10 (Coates et al. (2011)). 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 Radford et al. (2015) 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 (Kingma & Ba (2014)) with a learning rate of 10^{-4} and $\beta_1 = 0.5$. The results of 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 Arjovsky & Bottou (2017) and Sønderby et al. (2017) 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 2.3;

Application to Different GAN Models. We investigate the possibility of applying our proposed algorithm 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). Note that for the GAN with minimax loss (MMGAN) 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 (Krizhevsky (2009)) and CelebA (Liu et al. (2015)). 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 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

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 noisy 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 noisy 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) NTGAN ($\lambda = 0.1$)	27.7 ± 0.8	$7.31 \pm .06$	63.9 ± 1.7	$8.81 \pm .07$
(g) NTGAN ($\lambda = 1$)	26.5 ± 0.6	$7.49 \pm .04$	64.0 ± 1.4	$8.52 \pm .16$
(h) NTGAN ($\lambda = 10$)	29.8 ± 0.4	$6.55 \pm .08$	66.9 ± 3.2	$8.38 \pm .20$
(i) NTGAN alt. mini-batch ($\lambda = 1$)	28.7 ± 0.6	$7.3 \pm .05$	67.8 ± 3.2	$8.30 \pm .11$

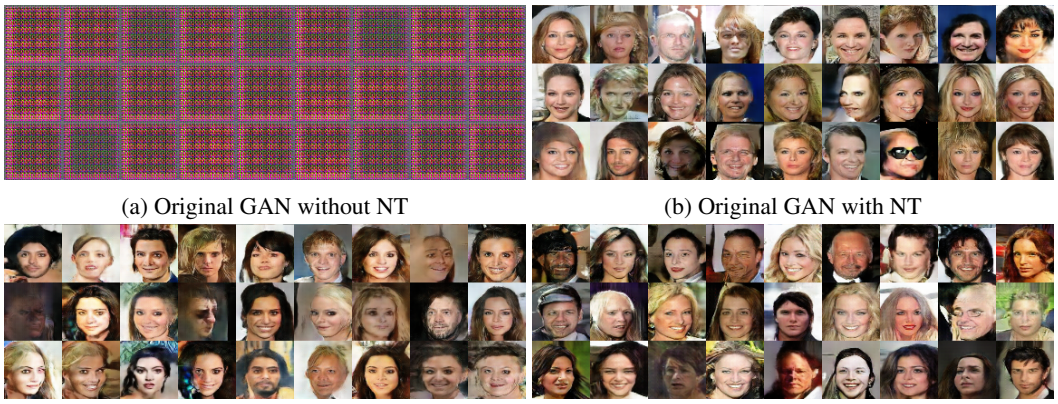
Table 3: We apply our proposed method to various previous GAN models trained on CIFAR-10 and CelebA. The same network architectures and hyperparameters as in the original works are used. We can observe that using our approach increases performance in most cases even with the suggested hyperparameter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

Model	**CIFAR-10**		**CelebA**	
	FID	IS	FID	IS
MMGAN (Goodfellow et al. (2014))	> 450	~ 1	> 350	~ 1
DCGAN (Radford et al. (2015))	33.4 ± 0.5	$6.73 \pm .07$	25.4 ± 2.6	$2.42 \pm .06$
WGAN-GP (Gulrajani et al. (2017))	37.7 ± 0.4	$6.55 \pm .08$	15.5 ± 0.2	$2.57 \pm .02$
LSGAN (Mao et al. (2017))	38.7 ± 1.8	$6.73 \pm .12$	21.4 ± 1.1	$2.55 \pm .06$
SNGAN (Miyato et al. (2018))	29.1 ± 0.4	$7.26 \pm .06$	13.2 ± 0.3	$2.31 \pm .02$
MMGAN +NT ($\lambda = 0.1$)	33.1 ± 0.7	$6.91 \pm .05$	16.6 ± 1.9	$2.44 \pm .02$
DCGAN + NT ($\lambda = 10$)	31.2 ± 0.3	$6.95 \pm .11$	14.7 ± 1.0	$2.57 \pm .05$
LSGAN + NT ($\lambda = 10$)	36.7 ± 1.2	$6.63 \pm .17$	19.9 ± 0.4	$2.60 \pm .06$
SNGAN + NT ($\lambda = 1$)	28.0 ± 1.6	$7.39 \pm .06$	11.0 ± 0.4	$2.39 \pm .03$
NTGAN (same generator as SNGAN)	26.5 ± 0.6	$7.49 \pm .04$	11.5 ± 0.9	$2.54 \pm .04$

that applying our method improves performance in most cases and even enables the training with the original saturation-prone minimax GAN objective, which is very unstable otherwise. We show random CelebA reconstructions from models trained with and without NT in Figure 2.

Robustness to Hyperparameters. We test the robustness of NTGANs 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 stabilization (Standard), a model with the gradient penalty regularization proposed by Roth et al. (2017) (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. Figure 3 shows that our NTGAN is stable and accurate across all settings.

Qualitative Results. We trained NTGANs on 128×128 images from two large-scale datasets: ImageNet (Russakovsky et al. (2015)) and LSUN bedrooms (Yu et al. (2015)). 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 iterations on ImageNet. Random samples of the models are shown in Figure 4.



(a) Original GAN without NT

(b) Original GAN with NT

(c) DCGAN without NT

(d) DCGAN with NT

Figure 2: Left column: Random reconstructions from models trained on CelebA without noise-tempering (NT). Right column: Random reconstructions with our proposed method.

Table 4: Hyperparameter settings used to evaluate the robustness of our proposed GAN stabilizer.

Exp.	**Learning Rate**	**BatchNorm in G**	**Optimizer**	**Activation**	n_{disc}	**#Filters G**	**#Samples**
a)	$2 \cdot 10^{-4}$	FALSE	ADAM	ReLU / lReLU	1	$[512, 256, 128, 64]$	50K
b)	$2 \cdot 10^{-4}$	TRUE	ADAM	tanh	1	$[512, 256, 128, 64]$	50K
c)	$1 \cdot 10^{-3}$	TRUE	ADAM	ReLU / lReLU	1	$[512, 256, 128, 64]$	50K
d)	$1 \cdot 10^{-2}$	TRUE	SGD	ReLU / lReLU	1	$[512, 256, 128, 64]$	50K
e)	$2 \cdot 10^{-4}$	TRUE	ADAM	ReLU / lReLU	5	$[512, 256, 128, 64]$	50K
f)	$2 \cdot 10^{-4}$	TRUE	ADAM	ReLU / lReLU	1	$[64, 64, 64, 64]$	50K
g)	$2 \cdot 10^{-4}$	FALSE	ADAM	ReLU / lReLU	1	$[512, 256, 128, 64]$	5K

4 Related Work

The inherent instability of GAN training was first addressed through a set of techniques and heuristics (Salimans et al. (2016)) and careful architectural design choices and hyper-parameter tuning (Radford et al. (2015)). Salimans et al. (2016) for example propose 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 & Bottou (2017). 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. Sønderby et al. (2017) made similar observations and also proposed the use of "instance noise" as a way to overcome these issues. Arjovsky et al. (2017) build on the work of Arjovsky & Bottou (2017) and introduce 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. Gulrajani et al. (2017) present a better way to enforce the Lipschitz constraint via a gradient penalty over interpolations between real and generated data (WGAN-GP). Roth et al. (2017) introduced a stabilizing regularizer based on a gradient norm penalty similar to that by Gulrajani et al. (2017). Their 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 Miyato et al. (2018). This method demonstrated 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., Mao et al. (2017), Zhao et al. (2016), Berthelot et al. (2017), Arjovsky et al. (2017), Zhao et al. (2016), Kodali et al. (2017)).

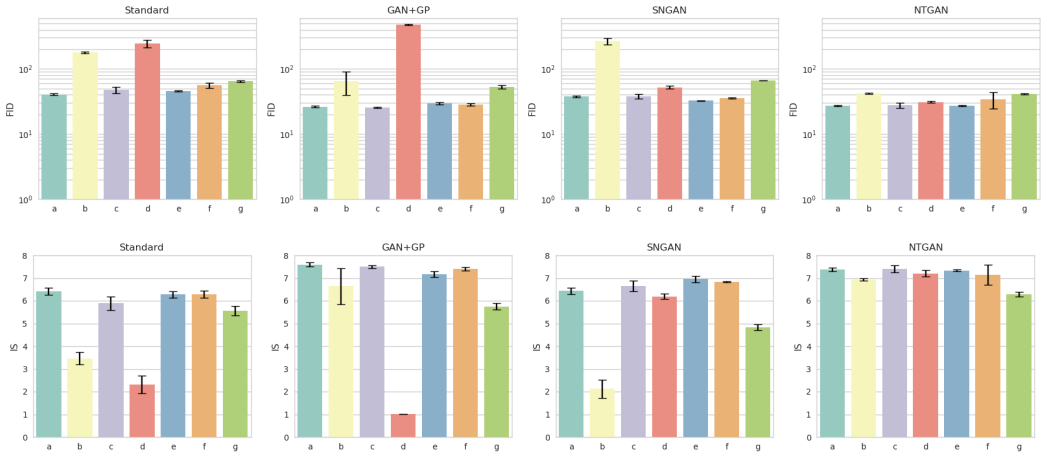


Figure 3: Results of the robustness experiments as specified in Table 4 and performed on CIFAR-10. We compare the standard GAN (*1st column*), a GAN with gradient penalty (*2nd column*), a GAN with spectral normalization (*3rd column*) and a GAN with our proposed method (*4th column*). Results are reported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

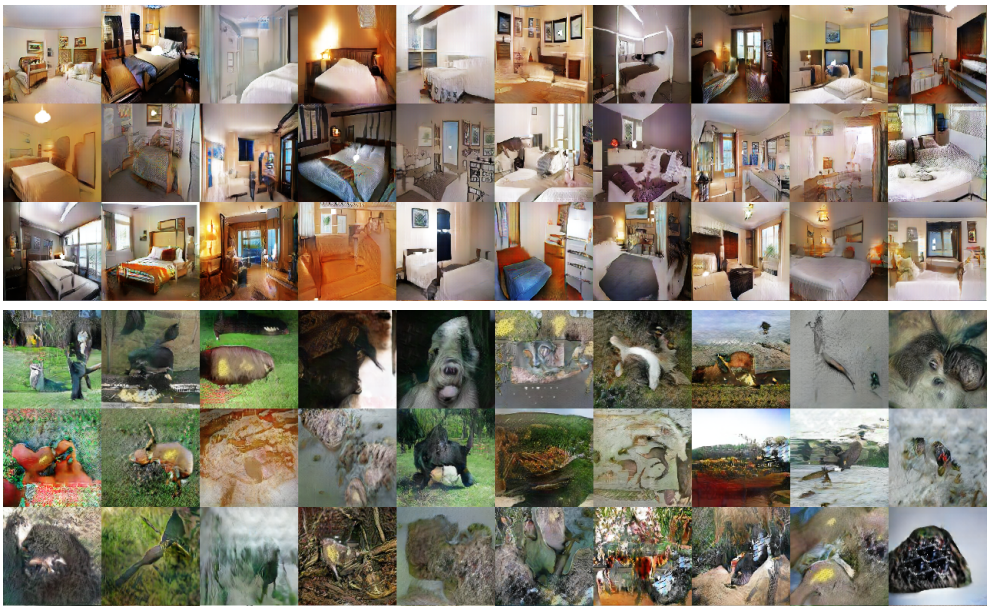


Figure 4: Reconstructions from NTGANs trained on 128×128 images from the LSUN bedrooms dataset (*top*) and ImageNet (*bottom*).

5 Conclusion

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 exploit additive noise to extend the support of the densities and thus facilitate the matching through gradient descent. More importantly, we show that using multiple loss terms with different additive noise (including the no-noise case) is necessary to achieve a highly accurate match of the original data distribution. We demonstrate the proposed training method on several common datasets of real images.

REFERENCES

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

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

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

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*, pp. 215–223, 2011.

I.M. Gelfand and S.V. Fomin. *Calculus of Variations*. Selected Russian publications in the mathematical sciences. Prentice-Hall, 1964. URL `https://books.google.ch/books?id=zylANwAACAAJ`.

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*, pp. 2672–2680, 2014.

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

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.

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*, pp. 448–456. JMLR. org, 2015.

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

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

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)*, 2015.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. *CoRR*, abs/1711.10337, 2017.

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)*, pp. 2813–2821. IEEE, 2017.

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

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

Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems*, pp. 2015–2025, 2017.

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.

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*, pp. 2234–2242, 2016.

Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. In *International Conference on Learning Representations*, 2017.

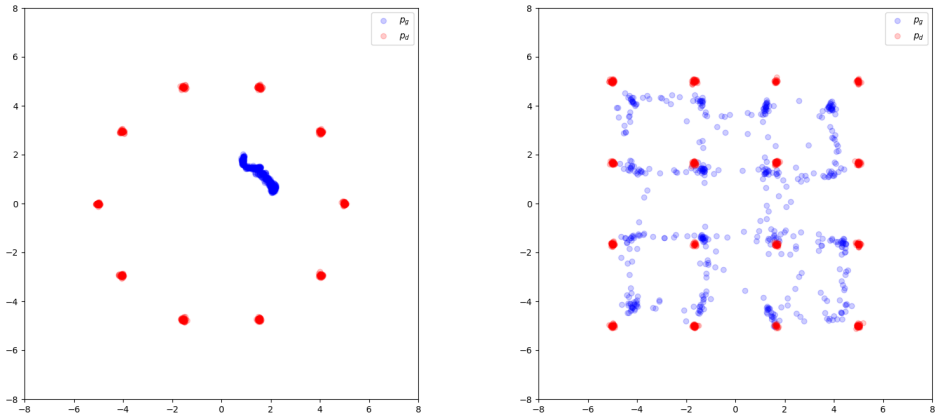
Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015.

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

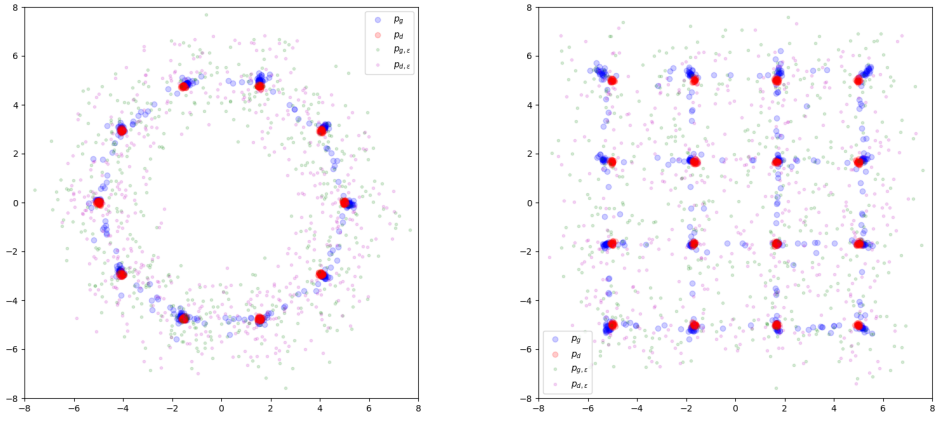
6 APPENDIX

6.1 EXPERIMENTS ON SYNTHETIC DATA

We performed experiments with a standard GAN and a NTGAN using Gaussian noise on synthetic 2-D data. The generator and discriminator architectures are both MLPs consisting of three fully-connected layers with a hidden-layer size of 512. We use ReLU activations and batch-normalization (Ioffe & Szegedy (2015)) in all but the first discriminator layer and the output layers. The Adam optimzer (Kingma & Ba (2014)) was used with a learning rate of 10^{-4} and we trained for 20K iterations. The results are shown in Figure 5. We can observe how the matching of both clean and blurred distribution leads to a better fit in the case of NTGAN.



(a) Standard GAN



(b) NTGAN with Gaussian noise

Figure 5: We performed experiments on synthetic 2D data with a standard GAN (*top*) and a NTGAN (*bottom*). The ground truth data is shown in *red* and the model generated data is shown in *blue*. For NTGAN we also show samples from the blurred data distribution $p_{d,\epsilon}$ in *green* and the blurred model distribution $p_{g,\epsilon}$ in *purple*.

Figure 6: We show examples of the generated noise (top row) and corresponding noisy training examples (rows 2 to 4). The different columns correspond to different iterations.

6.2 Examples of the Noise Generator

Examples of the noise generated by the noise generator of a NTGAN trained on CIFAR-10 are illustrated in Figure 6. We can observe noise patterns that resemble some of the commonly observed artifacts produced by a degenerate GAN generator. The noise patterns vary over time and therefore encourage the fitting of the distributions under different blurs.

6.3 Implementation Details

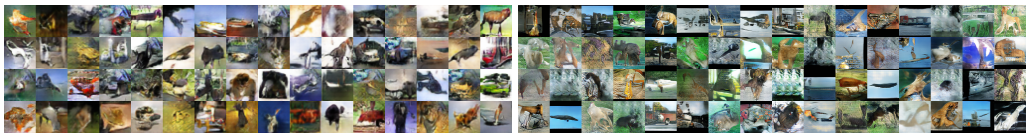
Noise Generator. The noise-generator architecture in all our experiments is equivalent to the generator architecture with the number of filters reduced by a factor of eight. The output of the noise-generator has a `tanh` activation scaled by a factor of two to allow more noise if necessary. We also experimented with a linear activation but didn't find a significant difference in performance.

GAN+GP. For the comparisons to the GAN regularizer proposed by Roth et al. (2017) we used the same settings as used in their work in experiments with DCGAN.

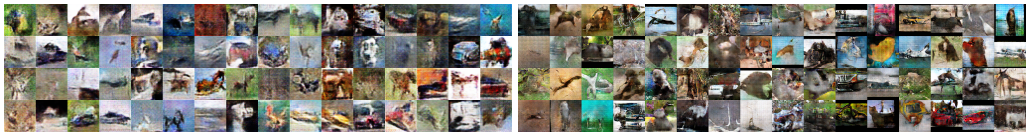
SNGAN+NT. We used the standard GAN loss (same as DCGAN) in all our experiments with models using spectral normalization. When combining SNGAN with NT we achieved the best results when batch-normalizing the inputs to the discriminator. For models with batch-normalization we found no benefit by adding this input normalization.

6.4 Qualitative Results for Experiments

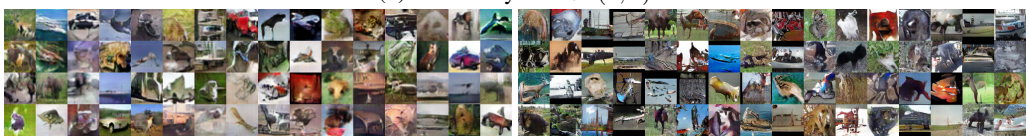
We provide qualitative results for some of the ablation experiments in Figure 7 and for the robustness experiments in Figure 8. As we can see in Figure 8, none of the tested settings led to degenerate solutions in the case of NTGAN while the other methods would show failure cases in some settings.



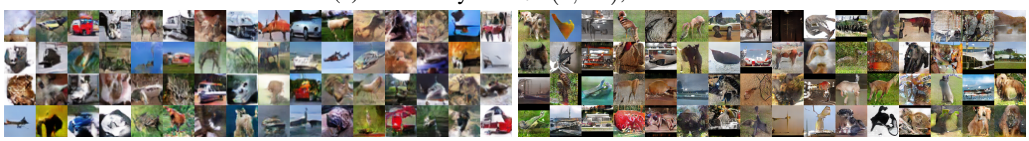
(a) Standard GAN



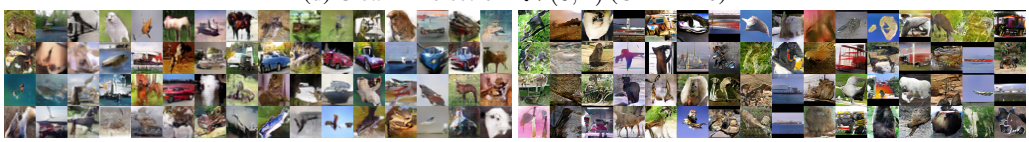
(b) Noise only: $\epsilon \sim \mathcal{N}(0, I)$



(c) Noise only: $\epsilon \sim \mathcal{N}(0, \sigma I)$, $\sigma \to 0$



(d) Clean + noise: $\epsilon \sim \mathcal{N}(0, I)$ (CIFAR-10)



(e) NTGAN ($\lambda = 1$)

Figure 7: We show random reconstructions for some of the ablation experiments listed in Table 2. The left column shows results on CIFAR-10 and the right column shows results on STL-10.

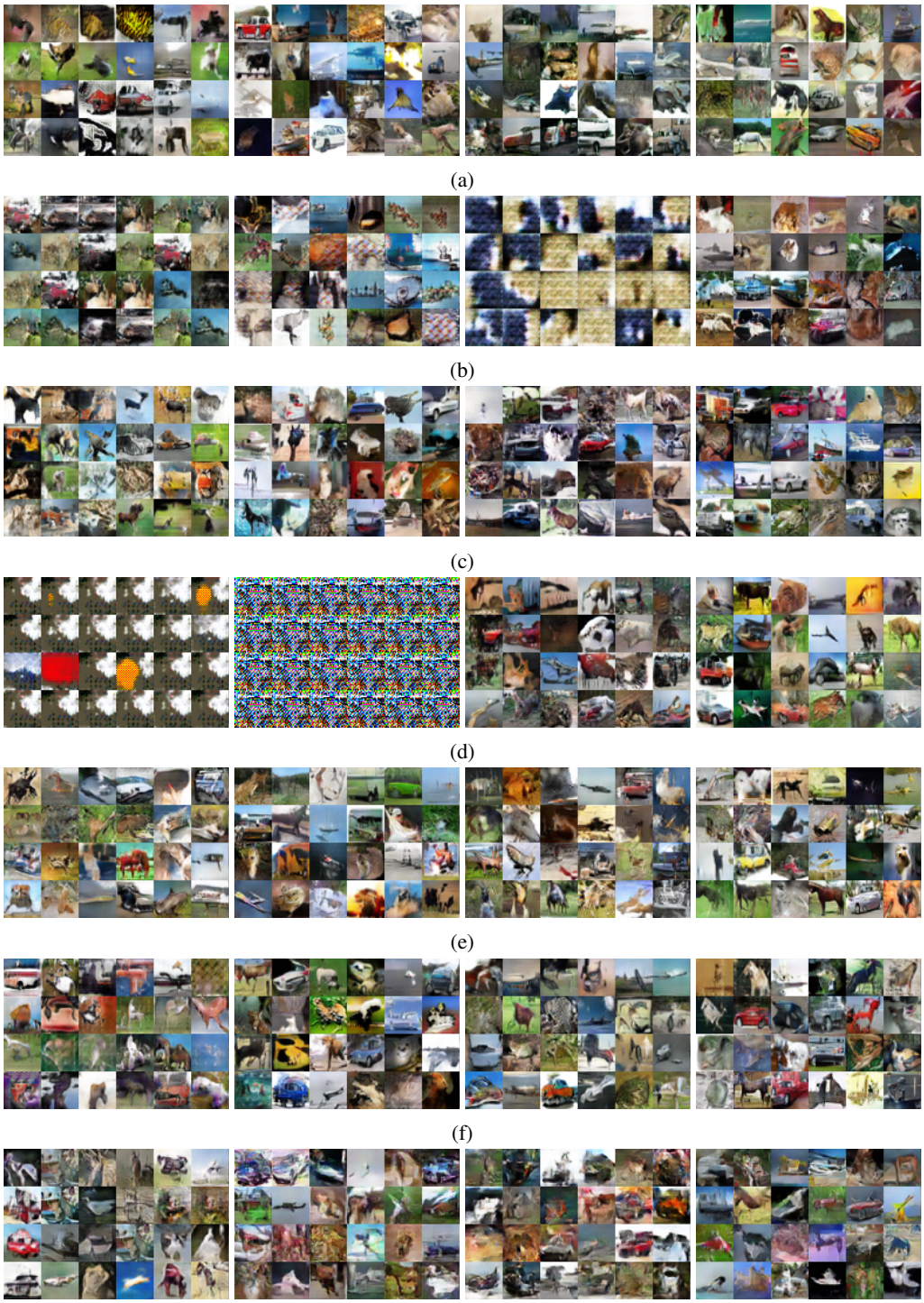


Figure 8: We show random reconstructions for the robustness experiments (see Table 4). We compare a standard GAN (*1st column*), a GAN with gradient penalty by Roth et al. (2017) (*2nd column*), a GAN with spectral normalization by Miyato et al. (2018) (*3rd column*) and a GAN with our proposed method (*4th column*).