How does Lipschitz Regularization Influence GAN Training?

Anonymous authors
Paper under double-blind review

Abstract

Despite the recent success of Lipschitz regularization in stabilizing GAN training, the exact reason of its effectiveness remains poorly understood. It is commonly believed that the main function of K-Lipschitz regularization is to restrict the $L2$-norm of the neural network gradient to be smaller than a threshold K (e.g. $K = 1$) such that $\|\nabla f\| \leq K$. While in this work, we uncover a counter-intuitive fact that under typical GAN setups, the choice of K does not matter. This finding suggests that instead of keeping the neural network gradients small, an even more important function of Lipschitz regularization is its restriction on the domain and interval of attainable gradient values of the loss function. This avoids the bias of the loss function over input samples. Empirically, we verify our proposition on the MNIST, CIFAR10 and CelebA datasets.

1 Introduction

Generative Adversarial Networks (GANs) are a class of generative models successfully applied to various applications, e.g., pose-guided image generation (Ma et al., 2017), image-to-image translation (Zhu et al., 2017), super-resolution (Ledig et al., 2017), high resolution image synthesis (Wang et al., 2018), urban modeling (Kelly et al., 2018), etc. Theoretically, Goodfellow et al. (2014) proved the convergence of GAN training by assuming that the generator is always updated according to the temporarily optimal discriminator at each training step. In practice, this assumption is too difficult to satisfy and GANs remain notoriously difficult to train. To stabilize the training of the GANs, various techniques have been proposed regarding the choices of architectures (Radford et al., 2015; He et al., 2016), loss functions (Arjovsky et al., 2017; Mao et al., 2017a), regularization and normalization (Arjovsky et al., 2017; Gulrajani et al., 2017; Mescheder et al., 2018; Miyato et al., 2018). We refer interested audiences to (Lucic et al., 2017; Kurach et al., 2018) for empirical studies.

Among them, the Lipschitz regularization (Gulrajani et al., 2017; Miyato et al., 2018) has shown great success in stabilizing the training of various GANs recently. For example, Mao et al. (2017b) and Fedus et al. (2018) observed that the gradient penalty Lipschitz regularizer helps to improve the training of the LS-GAN (Mao et al., 2017a) and the NS-GAN (Goodfellow et al., 2014) respectively; Miyato et al. (2018) observed that the NS-GAN with their spectral normalization Lipschitz regularizer works better than WGAN-GP (Gulrajani et al., 2017).

In this paper, we provide an analysis to better understand the impact of Lipschitz regularization and the loss function. This analysis is summarized below.

i) How does the Lipschitz constant K influence GAN training?

- For the neural network, under typical GAN setups, the choice of K does not matter. The only exception is that a very small K leads to numerical robustness issues and slow training.
- For the loss function, K restricts its domain and thereby the range of values of the attainable gradients. This means that, as $K \to 0$, all loss functions degenerate to linear functions with constant gradients that are similar to the Wasserstein loss.

In other words, K does not matter for the network itself, but it matters for the loss function.

ii) How does the loss function influence GAN training? There are actually two conflicting goals:

- The loss function should be strictly convex for better convergence than linear ones.
- The loss function should be linear to avoid mode collapse.

So our main insight is that the rule of thumb of using small Lipschitz constants is mainly to degenerate loss functions to almost linear ones. These degenerate losses improve GAN training by achieving a balance between the two conflicting goals mentioned above. Because of this, the exact shapes of the loss functions before degeneration do not seem to matter that much. We demonstrate this by two experiments. First, we show that when K is sufficiently small, even GANs trained with ridiculous loss functions (e.g. cosine) give comparable results to all other loss functions. Second, we can directly degenerate loss functions by introducing loss function scaling. This enables successful GAN training for a wide range of K for all loss functions, which only worked for the Wasserstein loss before. Our analysis is made more formal and expanded on in the following sections.

2 Lipschitz Regularization and GAN Loss Functions

Traditionally, Lipschitz regularization and GAN loss functions have been studied separately.

GAN Loss Functions. Under the assumption of optimal discriminators, the GAN loss function is usually interpreted as a measure of some sort of statistical divergence between the model distribution and the data distribution. Based on this understanding, a variety of GAN loss functions have been proposed. Goodfellow et al. (2014) first proposed to use the sigmoid cross-entropy loss function that minimizes the Jensen-Shannon (JS) divergence. However, because of the saturation at both ends of the sigmoid function, such a loss function can lead to vanishing gradients and thus fail to update the generator. To compensate for it, Goodfellow et al. (2014) proposed a variant named the *non-saturating* loss (NS-GAN), which heuristically amplifies the gradients when updating the generator. Observing that the JS divergence is a special case of the f-divergence, Nowozin et al. (2016) extended the idea of Goodfellow et al. (2014) and showed that any f-divergence can be used to train GANs. Their work suggested that the performance of GANs can be improved by employing "better" divergence measures. Following this direction, Arjovsky & Bottou (2017) first pointed out the flaws of the JS divergence used in GANs and then proposed to use the Wasserstein distance (WGAN) instead (Arjovsky et al., 2017). In addition, they employed the Kantorovich-Rubinstein dual of the Wasserstein distance to improve the computational efficiency, which simplifies the WGAN loss function to a simple identity function but at the cost of requiring the discriminator neural network to be 1-Lipschitz continuous. Meanwhile, Mao et al. (2017a) proposed to use the quadratic loss function that minimizes the Pearson χ^2 divergence, which is called the Least-Square GAN (LS-GAN).

Lipschitz Regularization. Instead of stabilizing GAN training, Lipschitz regularization is first designed to enforce the 1-Lipschitz continuity of the discriminator neural network, which is required by the Kantorovich-Rubinstein duality applied in WGAN (Arjovsky et al., 2017). Since then, several algorithms are proposed to implement Lipschitz regularization. Arjovsky et al. (2017) first proposed the *weight clipping*, whose main idea is to clamp the weights of each neural network layer to a small fixed range $[-c, c]$, where c is a small positive constant. Although weight clipping guarantees the Lipschitz continuity of the discriminator, the choice of parameter c is difficult and prone to invalid gradients. To this end, Gulrajani et al. (2017) proposed the *gradient penalty* (GP) which incorporates an extra regularization term of discriminator's gradient magnitude into the loss function. In their original setting, GP is one-centered to enforce 1-Lipschitz continuity. While Mescheder et al. (2018) showed that the zero-centered one is more reasonable for the convergence of GAN training. However, one major problem of GP is that it is computed with finite samples, which makes it intractable to be applied to the entire output space. To sidestep this problem, Gulrajani et al. (2017) proposed to heuristically sample from the straight lines connecting model distribution and data distribution. However, this makes their approach heavily dependent on the support of the model distribution (Miyato et al., 2018). Addressing this issue, Miyato et al. (2018) proposed the *spectral normalization* (SN) that enforces the Lipschitz continuity of a neural network in the operator space. Observing that the Lipschitz constant of the entire neural network is bounded by the product of those of its layers, they break down the problem to applying the Lipschitz regularization onto each neural network layer. These simplified sub-problems can then be solved by normalizing the weight matrix of each layer according to its largest singular value.

While recently, a tendency of applying 1-Lipschitz regularization to GANs without Lipschitz constraint requirements appears and links the two fields together. For example, Fedus et al. (2018) and Mao et al. (2017b) observed that applying the gradient penalty Lipschitz regularizer designed for the WGAN also helps to improve the training of the NS-GAN and the LS-GAN respectively. Similarly,

Table 1: The GAN loss functions used in our experiments. $f(\cdot)$ is the output of the discriminator neural network; $g(\cdot)$ is the output of the generator neural network; x is the sample from the training dataset; z is the sample from the noise distribution. For the NS-GAN, $f^*(\cdot) = \mathrm{sigmoid}[f(\cdot)]$.

GAN types	Discriminator Loss	Generator Loss
NS-GAN	$-\mathbb{E}[\log(f^*(x))] - \mathbb{E}[\log(1 - f^*(g(z)))]$	$-\mathbb{E}[\log(f^*(g(z)))]$
LS-GAN	$\mathbb{E}[(f(x)-1)^2] + \mathbb{E}[f(g(z))^2]$	$\mathbb{E}[(f(g(z))-1)^2]$
WGAN	$\mathbb{E}[f(x)] - \mathbb{E}[f(g(z))]$	$\mathbb{E}[f(g(z))]$
COS-GAN	$-\mathbb{E}[\cos(f(x)-1)] - \mathbb{E}[\cos(f(g(z))+1)]$	$-\mathbb{E}[\cos(f(g(z))-1)]$
EXP-GAN	$\mathbb{E}[\exp(f(x))] + \mathbb{E}[\exp(-f(g(z)))]$	$\mathbb{E}[\exp(f(g(z)))]$

Miyato et al. (2018) observed that the NS-GAN with their spectral normalization regularizer works better than WGAN-GP (Gulrajani et al., 2017). All these observations imply that:

Hypothesis 1. *Keeping neural network gradients small by enforcing 1-Lipschitz regularization is the key to successful GAN training while the choice of loss functions contributes little.*

However, due to its complex nature, deep learning is usually criticized as "alchemy" (Hutson, 2018) and prone to attribution error, i.e. the credit of success is often incorrectly attributed. To this end, we would like to question the above hypothesis: is it true?

2.1 Empirical Evaluations on Lipschitz Regularized GANs

To answer the above question, we empirically evaluate how the choice of Lipschitz constants influence the performance of various Lipschitz regularized GANs. First, we regularize the discriminator neural network with the state-of-the-art spectral normalization and enforce K-Lipschitz regularization by tuning the hyper-parameter k_{SN} (Miyato et al., 2018),

$$\bar{W}_{SN}(W, k_{SN}) := k_{SN} \cdot W/\sigma(W), \tag{1}$$

where W is the weight matrix of a neural network layer, $\sigma(W)$ is its largest singular value. The relationship between k_{SN} and K can be quantitatively approximated as $K \approx k_{SN}^n$ (Miyato et al., 2018), where n is the number of neural network layers in the discriminator. Then, we train these Lipschitz regularized GANs with a variety of GAN loss functions (Table 1) consisting of three most widely-used ones: NS-GAN, LS-GAN, WGAN, and two arbitrary ones: $\cos(\cdot)$ and $\exp(\cdot)$. Following the best practice, we employ the Fréchet Inception Distance (FID) metric (Heusel et al., 2017) to quantitatively measure the performance of these GANs. The smaller the FID score, the better the performance of the GAN. Please see Appendix B for the detailed neural network architectures and other experimental setups.

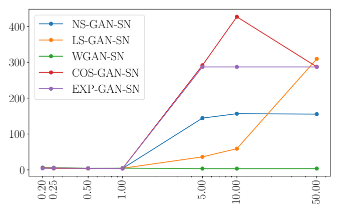
Table 2 shows the FID scores of different GAN loss functions with different k_{SN}. Qualitative results are shown in Appendix E. At the first glance, the results seem to be a solid support of Hypothesis 1. For example, when $k_{SN} \leq 1.0$, all the loss functions (including the arbitrary ones) can be used to train GANs stably; the FID scores of all loss functions (except the WGAN) are similar; the FID scores of all loss functions slightly worsen as k_{SN} further decreases. When $k_{SN} \geq 5.0$, the performance of all the GANs (except the WGAN) worsen and even break (very high FID scores, e.g. ≥ 100). However, the history tells us that significant scientific progresses are usually inspired by outliers instead of the observations as expected. In our case, the WGAN is such an outlier as it works normally when $k_{SN} \geq 5.0$. This suggests that the performance of WGAN is insensitive to the choice of Lipschitz constants, which contradicts Hypothesis 1. So does the choice of Lipschitz constant really matter?

3 Do Lipschitz Constants Matter?

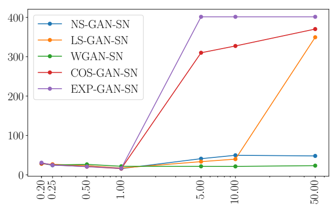
In a nutshell, the choice of Lipschitz constants does not matter when i) scale insensitive activation functions (e.g. ReLU) and ii) optimizers using normalized gradients (e.g. RMSProp) are used. Note that both techniques are dominating ones in practical applications due to their effectiveness in stabilizing neural network training (Glorot et al., 2011; Tieleman & Hinton, 2012).

Table 2: FID scores *vs.* k_{SN} on different datasets. k_{SN} is typically fixed as 1 to enforce 1-Lipschitz regularization (Miyato et al., 2018). The WGAN is an outlier that works normally even when $k_{SN} \geq 5.0$. For the line plots, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores. From left to right, the seven points on each line have $k_{SN} = 0.2, 0.25, 0.5, 1.0, 5.0, 10.0, 50.0$ respectively. Lower FID scores are better. SN: spectral normalization (Miyato et al., 2018).

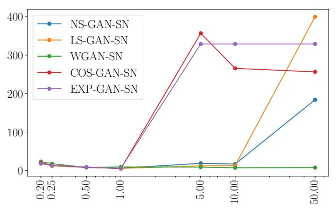
Dataset	GANs	FID Scores						
		k_{SN}=0.2	0.25	0.5	1.0	5.0	10.0	50.0
MNIST	NS-GAN-SN	5.41	3.99	4.20	3.90	144.28	156.60	155.41
	LS-GAN-SN	5.14	**3.96**	**3.90**	4.42	36.26	59.04	309.35
	WGAN-SN	6.35	6.06	4.44	4.70	**3.58**	**3.50**	**3.71**
	COS-GAN-SN	5.41	4.83	4.05	3.86	291.44	426.62	287.23
	EXP-GAN-SN	**4.38**	4.93	4.25	**3.69**	286.96	286.96	286.96
CIFAR10	NS-GAN-SN	29.23	**24.37**	23.29	**15.81**	41.04	49.67	48.03
	LS-GAN-SN	**28.04**	26.85	23.14	17.30	33.53	39.90	349.35
	WGAN-SN	29.20	25.07	26.61	21.75	**21.63**	**21.45**	**23.36**
	COS-GAN-SN	29.45	25.31	**20.73**	15.88	309.96	327.20	370.13
	EXP-GAN-SN	30.89	24.74	20.90	16.66	401.24	401.24	401.24
CelebA	NS-GAN-SN	18.59	12.71	**8.04**	6.11	18.95	17.04	184.06
	LS-GAN-SN	20.34	**12.14**	8.85	5.69	12.40	13.14	399.39
	WGAN-SN	23.26	17.93	8.48	9.41	**9.03**	**7.37**	**7.82**
	COS-GAN-SN	20.59	13.93	8.88	**5.20**	356.70	265.53	256.44
	EXP-GAN-SN	**18.23**	13.65	9.18	5.88	328.94	328.94	328.94



(a) MNIST



(b) CIFAR10



(c) CelebA

Let us consider a simple discriminator $D(x) = L(f(x))$, where x is the input, f is the discriminator neural network with scalar output, L is the loss function. Following Miyato et al. (2018), we omit the bias term for simplicity and describe a spectral normalized $f(x)$ as:

$$f(x) = \bar{W}_{SN}^{N+1}(a_N(\bar{W}_{SN}^{N}(a_{N-1}(\bar{W}_{SN}^{N-1}(...a_1(\bar{W}_{SN}^{1}x)...))))), \tag{2}$$

where, a_n is the activation function, $\bar{W}_{SN}^{n}$ is the normalized weight matrix of the n-th layer (Eq.1).

Forward Pass. As Glorot et al. (2011) and Hoffer et al. (2018) suggested, scale-invariant activation functions allow the scaling parameter like k_{SN} to move freely within the neural network without influencing its function. For example, let $W_{\text{1-}Lip} = W/\sigma(W)$, Eq.2 can be rewritten as

$$f(x) = K\left[W_{\text{1-}Lip}^{N+1}(a_N(W_{\text{1-}Lip}^{N}(a_{N-1}(W_{\text{1-}Lip}^{N-1}(...a_1(W_{\text{1-}Lip}^{1}x)...)))))\right], \tag{3}$$

where $K = (k_{SN})^{N+1}$ is the Lipschitz constant. Thus, it can be concluded that K only works as a scaling hyper-parameter and is independent of the overall network function.

Backpropagation. Similar to the case of forward pass, scale-insensitive activation functions allow k_{SN} to move freely during backpropagation.

$$\nabla a(k_{SN} \cdot W_{\text{1-}Lip} h_{in})) = k_{SN} \nabla a(W_{\text{1-}Lip} h_{in}) \tag{4}$$

where h_{in} is the input of a neural network layer. Eq.4 implies that the scaling parameter k_{SN} will be accumulated when the gradients are backpropagated through neural network layers, which may produce exploding or vanishing gradients that can heavily influence the training. However, for optimizers with normalized gradients (e.g. RMSProp), each gradient is divided by an exponentially decaying average of the past gradients, which largely eliminates the influence of the gradient scale. For example in RMSProp (Tieleman & Hinton, 2012),

$$w_{t+1} = w_t - \eta \frac{k_{SN}^m \cdot g_t}{\sqrt{E[(k_{SN}^m \cdot g)^2]_t + \epsilon}} = w_t - \eta \frac{g_t}{\sqrt{E[g^2]_t + \epsilon / k_{SN}^{2m}}} \tag{5}$$

where k_{SN}^m is a scaling parameter, w is the neural network weight, η is the learning rate, g is the gradient, $E[g^2]$ is the exponentially decaying average that $E[g^2]_t = \gamma E[g^2]_{t1} + (1-\gamma)g_t^2$, $\epsilon > 0$ is a small constant for numerical stability that can usually be ignored when $\epsilon \ll E[(k_{SN}^m \cdot g)^2]_t$.

Summarizing the above analysis, the major influence of k_{SN} is only a simple *scaling effect* applied to the neural network output during the forward pass, which has nothing to do with the overall neural network function. Thus, the choice of Lipschitz constant does not matter.

The above conclusion clarifies the "strange" robustness of WGAN against large k_{SN} ($k_{SN} \geq 5$) in Table 2: the WGAN loss functions are linear and have constant gradients which are independent of the scale of neural network outputs that is controlled by k_{SN}. Thus, the choice of k_{SN} does not influence the performance of WGAN. Similarly, it can be easily inferred that the big impact of Lipschitz constants on the performance of other GANs (Table 2) originates from its interaction with the loss functions.

Remark. The above analysis also clarifies the observation that when $k_{SN} \leq 1$, the FID scores of all loss functions slightly worsen as k_{SN} decreases, which is actually a numerical problem. Eq.5 shows that the effect of ϵ is heavily dependent on the scale of gradients. When k_{SN} is small (e.g. $k_{SN} = 0.2$), the gradients are dramatically scaled down by a factor of k_{SN}^m. As a result, ϵ cannot be ignored and reduces the magnitude of the gradients, which accounts for the degenerated performance.

4 Restrictions of GAN Loss Functions

In this section, we will show that: *compared to the choice of Lipschitz constants, the dominating factor of Lipschitz regularization is its side-effect of restricting the domain and interval of attainable gradient values of the loss functions*. To support our claim, we first theoretically analyze why a K-Lipschitz regularized discriminator *restricts* the domain and interval of attainable gradient values of the loss function to intervals bounded by K. Then, an empirical study is performed accordingly to verify our theoretical analysis. Finally, we show that such restrictions prevent the backpropagation of vanishing or exploding gradients as well as avoid the bias of loss functions over input samples, which stabilizes GAN training.

4.1 Theoretical Derivation

Let us consider a simple discriminator $D(x) = L(f(x))$, where x is the input, f is a neural network with scalar output, L is the loss function. During training, the loss function L works by backpropagating the gradient $\nabla L = \partial L(f(x))/\partial f(x)$ to update the neural network weights:

$$\frac{\partial D(x)}{\partial W^n} = \boldsymbol{\frac{\partial L(f(x))}{\partial f(x)}} \frac{\partial f(x)}{\partial W^n} \tag{6}$$

where W^n is the weight matrix of the n-th layer. Let X and Ω be the domain and the range of f respectively (i.e. $f : X \rightarrow \Omega$), it can be easily derived that the attainable values of ∇L is determined by Ω (i.e. $\nabla L : \Omega \rightarrow \Psi$). Without loss of generality, we assume that $x \in X = [-1, 1]^{m \times n \times 3}$ are normalized images and derive the bound of the size of Ω as follows:

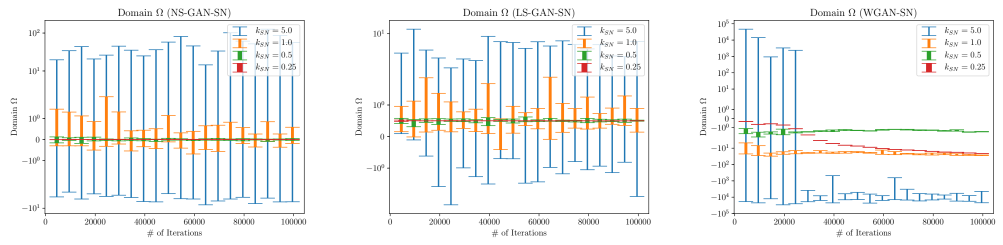
Theorem 1. *If the discriminator neural network f satisfies the k-Lipschitz continuity condition, we have $f : X \rightarrow \Omega \subset \mathbb{R}$ satisfying $|\min(\Omega) - \max(\Omega)| \leq k\sqrt{12mn}$.*

Please see Appendix A for the proof of Theorem 1. Theorem 1 shows that the size of Ω is bounded by k. However, k can be unbounded when Lipschitz regularization is not enforced during training, which results in an unbounded Ω and a large interval of attainable gradient values. On the contrary, when K-Lipschitz regularization is applied (i.e. $k \leq K$), the loss function L is restricted as follows:

Corollary 1.1 (**Restriction of Loss Function**). *Assume that f is a Lipschitz regularized neural network whose Lipschitz constant $k \leq K$, the loss function L is C^2-continuous with M as the maximum absolute value of its second derivatives in its domain. Let Ψ be the interval of attainable gradient values that $\nabla L : \Omega \rightarrow \Psi$, we have*

$$\big|\min(\Omega) - \max(\Omega)\big| \leq K\sqrt{12mn} \tag{7}$$

$$\big|\min(\Psi) - \max(\Psi)\big| \leq M \cdot K\sqrt{12mn} \tag{8}$$



(a) The domain Ω shrinks with decreasing k_{SN} for different loss functions. The x-axis shows the number of training iterations while the y-axis shows the domain Ω. Note that the y-axis is in log scale.

k_{SN}	NS-GAN-SN, $L(\cdot) = -\log(\text{sigmoid}(\cdot))$ Ω	Ψ	LS-GAN-SN, $L(\cdot) = (\cdot)^2$ Ω	Ψ
5.0	$[-8.130,\ 126.501]$	$[-1.000,\ -0.000]$	$[-2.460,\ 12.020]$	$[-4.921,\ 24.041]$
1.0	$[-0.683,\ 2.091]$	$[-0.664,\ -0.110]$	$[-0.414,\ 1.881]$	$[-0.828,\ 3.762]$
0.5	$[-0.178,\ 0.128]$	$[-0.545,\ -0.468]$	$[0.312,\ 0.621]$	$[0.623,\ 1.242]$
0.25	$[-0.006,\ 0.006]$	$[-0.502,\ -0.498]$	$[0.478,\ 0.522]$	$[0.956,\ 1.045]$

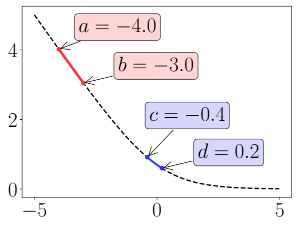
(b) The domain Ω and the interval of attained gradient values Ψ against k_{SN} ($\nabla L : \Omega \to \Psi$).

Figure 1: Empirical results regarding the restrictions of different GAN loss functions (Table 1) on the CelebA dataset. NS-GAN: Non-Saturating GAN; LS-GAN: Least-Square GAN; WGAN: Wasserstein GAN. k_{SN} is a parameter of spectral normalization (Eq.1).

Corollary 1.1 shows that under a mild condition (C^2-continuous), applying K-Lipschitz regularization restricts the domain Ω and thereby the interval of attainable gradient values Ψ of the loss function L to intervals bounded by K. When K is small, e.g., $K = 1$ (Gulrajani et al., 2017; Fedus et al., 2018; Mao et al., 2017b; Miyato et al., 2018), the interval of attainable gradient values of the loss function is considerably reduced, which prevents the backpropagation of vanishing or exploding gradients and thereby stabilizes the training.

Remark on the Changes of Ω_i During Training. So far we analyzed the restriction of the loss function by a static discriminator. However, the discriminator neural network f is dynamically updated during training and thus its range $\Omega^{\cup} = \cup_i \Omega_i$, where Ω_i is the discriminator range at each training step i. Therefore, we need to analyze two questions:

1. How does the size of Ω_i change during training?
2. Does Ω_i shift during training (figure on the right)?



$[a, b]$ shifts to $[c, d]$.

For question 1, the size of Ω_i is always bounded by the Lipschitz constant K throughout the training. While for question 2, the answer depends on the discriminator loss function. In a nutshell, common loss functions (e.g. NS-GAN, LS-GAN) are strictly convex and have a unique minimum respectively, which forces Ω_i to be positioned around it and prevents Ω_i from shifting. Thus, $\Omega^{\cup}$ is still roughly bounded by K. However, Ω_i may be allowed to shift when the discriminator loss functions is not strictly convex (e.g. WGAN). While for WGAN, its attainable gradient value Ψ is a constant that is independent of Ω_i. Thus, regarding to Ψ, we can view it as a degenerate loss function that still fits in our discussion.

4.2 Empirical Verification

Figure 1 shows the corresponding empirical results on the restriction of different loss functions. Note that the K-Lipschitz regularization is enforce by applying the spectral normalization (Miyato et al., 2018) that $K \approx k_{SN}^n$, where n is the number of neural network layers (Eq.1).

Restriction on Ω. In Figure 1 (a), we plot the domain Ω as intervals for different iterations under different k_{SN}. It can be observed that the interval Ω shrinks rapidly as k_{SN} decreases, which verifies that the domain Ω is restricted by the Lipschitz regularization (Eq.7).

Changes of Ω_i During Training Figure 1 (a) also shows that the domains Ω_i of the NS-GAN and the LS-GAN are rather fixed during training while those of the WGAN shift, which verifies our remark on the changes of Ω_i during training. More specifically, it can be observed that Ω_i of NS-GAN (Table 1) are relatively fixed around zero, which is the unique minimum of its loss function. A similar phenomenon can be observed for LS-GAN (Table 1) around 0.5. Interestingly, we empirically observed that the Ω_i of WGAN also get relatively fixed at late stages of the training.

Restriction on Ψ. In Figure 1 (b), we show the corresponding intervals of attainable gradient values Ψ when the domain Ω ($\Omega = \cup_i \Omega_i$) is restricted as in Figure 1. In particular, we compute the bounds of Ψ as follows,

$$\min(\Psi) = \nabla L\big(\min(\Omega)\big), \max(\Psi) = \nabla L\big(\max(\Omega)\big). \tag{9}$$

because ∇L is monotonically increasing in Ω for common GAN loss functions that are convex (e.g., NS-GAN, LS-GAN, WGAN). Note that the Ψ of WGAN is not included as it is a constant whose size is always zero. Similar to the domain Ω, Ψ also shrinks with the increasing strength of Lipschitz regularization, which avoids the saturating and exploding parts of the loss function. For example when $k_{SN} = 5.0$, the gradient of the NS-GAN loss function saturates to a value around 0 while that of the LS-GAN loss function explodes to 24.041. However, such problems do not happen when $k_{SN} \leq 1.0$. Note that we only compute Ψ on one of the two symmetric loss terms used in the discriminator loss function (Table 1). The interval of attained gradient values of the other loss term follows similar patterns.

4.3 How does the Restriction of Loss Functions Help Training?

At this point, we know that instead of influencing the overall neural network function, the choice of Lipschitz constant K only works by restricting the domain and thereby the interval of attainable gradient values of the loss function, which is implemented by scaling the neural network outputs by K (Eq.3). Thus, for a K-Lipschitz regularized neural network f, we have

$$f : X \to \Omega \iff f : X \to K \cdot \Omega_{1\text{-}Lip} \tag{10}$$

where $\Omega_{1\text{-}Lip}$ is the range of a corresponding 1-Lipschitz continuous neural network. This suggests that we can restrict the domain Ω of loss function L by a positive scaling hyper-parameter α directly,

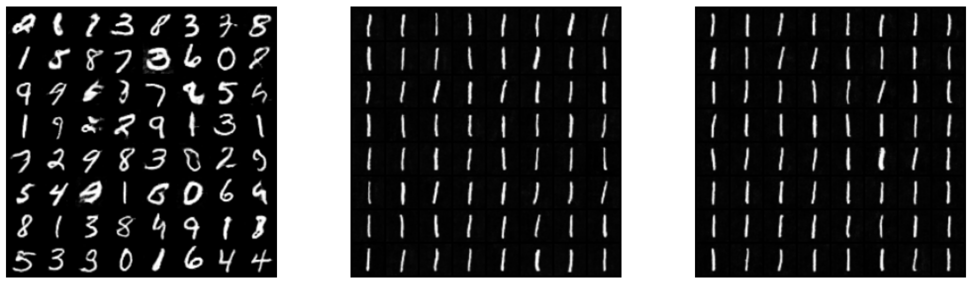
$$L_\alpha(\Omega) = L(\alpha \cdot \Omega)/\alpha. \tag{11}$$

Compared to tuning k_{SN}, the proposed *domain scaling* is more numerically stable as it does not influence the scale of the backpropagated gradients and thus does not suffer from the numerical problem caused by ϵ (discussed in section 3, Remark). This also motivates us to include α in the denominator of Eq.11, which helps to preserve the gradient scale of the loss function. Please see Appendix D for the extensive experiments on domain scaling.

The equivalency between tuning Lipschitz constants and domain scaling (Eq.11) implies that the performance improvements of various Lipschitz regularized GANs (Table 2) originates from the restriction of the loss function. But what kinds of restrictions are beneficial to the GAN training? Qualitatively, we hypothesize that:

1. The interval of attainable gradients values should be small. This prevents the backpropagation of exploding or vanishing gradients and guarantees that there is no significant bias over input samples from the same category (i.e. real or synthesized).
2. For better convergence, the restricted loss function should be strictly convex and has a unique minimum *locally* in its restricted domain.

For the first point, we omit the discussion of the well-studied exploding or vanishing gradients problem and focus on the bias of loss functions that may cause mode collapse (Figure 2). As we know, neural networks are naturally biased over inputs: they may "prefer" some input samples over the others by thinking they are "more real" and outputting higher/lower values, even though all the inputs are sample from unbiased real data. Biased (scale-sensitive) loss functions like NS-GAN backpropagate different gradients according to the biased network outputs, which further enhances the bias



(a) $\alpha = 1e^{-9}$, FID $= 3.61$ (b) $\alpha = 1e^{-3}$, FID $= 154.14$ (c) $\alpha = 1e^{-1}$, FID $= 155.54$

Figure 2: Samples of randomly generated images with NS-GAN against α ($k_{SN} = 50.0$, MNIST).

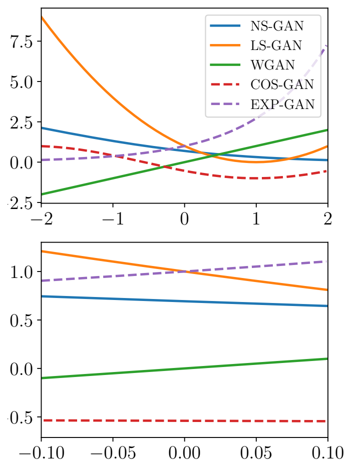
and may cause mode collapse (Figure 2). On the contrary, using unbiased (e.g. WGAN) or weakly biased loss functions (e.g. restricted loss functions) can sidestep the problem by backpropagating same or similar gradients for input samples within the same category (i.e. real or synthesized).

The second point makes sense because strictly convex loss functions with unique minima usually have better convergence. We observe that any loss function degenerates as its domain Ω shrinks to a single value. According to Taylor's expansion, let $\omega, \omega + \Delta\omega \in \Omega$, we have:

$$L(\omega + \Delta\omega) = L(\omega) + \frac{L'(\omega)}{1!}\Delta\omega + \frac{L''(\omega)}{2!}(\Delta\omega)^2 + \cdots . \quad (12)$$

As $|\max(\Omega) - \min(\Omega)|$ shrinks to zero, we have $L(\omega + \Delta\omega) \approx L(\omega) + L'(\omega)\Delta\omega$ showing that we can approximate any loss function by a linear function with constant gradient as its domain Ω shrinks to a single value (figure on the right). Let $\omega \in \Omega$, we implement the degeneration of a loss function by scaling its domain Ω with an extremely small constant α:

$$\lim_{\alpha \to 0} \frac{\partial L_\alpha(\omega)}{\partial \omega} = \frac{1}{\alpha} \cdot \frac{\partial L(\alpha \cdot \omega)}{\partial \omega} = \frac{\partial L(\alpha \cdot \omega)}{\partial(\alpha \cdot \omega)} = \nabla L(0). \quad (13)$$



We use $\alpha = 1e^{-25}$ in our experiments. Smaller values are not used due to numerical errors (NaN). More specifically, we scale the domain by $\alpha = 1e^{-25}$ and show the FID scores of WGAN and those of different loss functions in the table on the right. To verify that α is an alternative to k_{SN} and can compensate its effect, we use a large $k_{SN} = 50$. The LS-GAN$^{\#}$ (Mao et al., 2017b) is the zero-centered version of LS-GAN, whose rationale is discussed in Appendix D. It can be observed that all loss functions have similar performance as WGAN. However, due to the loss of strict convexity, their performance is slightly worsen than the best ones in Table 2 and in Appendix D.

GANs	FID Scores		
	MNIST	CIFAR10	CELEBA
WGAN	3.71	23.36	7.82
NS-GAN	3.74	21.92	8.10
LS-GAN$^{\#}$	3.81	21.47	8.51
COS-GAN	3.96	23.65	8.30
EXP-GAN	3.86	21.91	8.22

5 Conclusion

In this paper, we studied the influence of Lipschitz regularization on the GAN training. First, we point out a counter-intuitive fact that under typical GAN setups, the choice of Lipschitz constants is independent of the overall neural network function but just works as a scaling hyper-parameter of the neural network output. Second, we demonstrate that instead of keeping the neural network gradients small, the dominating factor of Lipschitz regularization is its restriction on the domain and interval of attainable gradient values of the loss function. Such restrictions stabilize GAN training by preventing the backpropagation of vanishing or exploding gradients and avoiding the bias of the loss function over input samples, which is a new step in understanding the exact reason for Lipschitz regularization's effectiveness.

REFERENCES

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 214–223, 2017.

William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M. Dai, Shakir Mohamed, and Ian Goodfellow. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. In *International Conference on Learning Representations*, 2018.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudk (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 315–323, 2011.

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. 5767–5777, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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 *Advances in Neural Information Processing Systems 30*, pp. 6626–6637. Curran Associates, Inc., 2017.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: Efficient and accurate normalization schemes in deep networks. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, 2018.

Matthew Hutson. Ai researchers allege that machine learning is alchemy. *Science*, 360(6388):861, 2018.

Tom Kelly, Paul Guerrero, Anthony Steed, Peter Wonka, and Niloy J. Mitra. Frankengan: Guided detail synthesis for building mass models using style-synchonized gans. *ACM Trans. Graph.*, 37 (6):1:1–1:14, 2018. doi: 10.1145/3272127.3275065.

Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The gan landscape: Losses, architectures, regularization, and normalization. *arXiv preprint arXiv:1807.04720*, 2018.

C. Ledig, L. Theis, F. Huszr, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 105–114, July 2017. doi: 10.1109/CVPR.2017.19.

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.

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

Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *Advances in Neural Information Processing Systems 30*, pp. 406–416. Curran Associates, Inc., 2017.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017a.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. On the effectiveness of least squares generative adversarial networks. *arXiv preprint arXiv:1712.06391*, 2017b.

Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 3478–3487, 2018.

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

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems 29*, pp. 271–279. Curran Associates, Inc., 2016.

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

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

A Proof of Theorem 1

Theorem 1. *If the discriminator neural network f satisfies the k-Lipschitz continuity condition, we have $f : X \to \Omega \subset \mathbb{R}$ satisfying $|\min(\Omega) - \max(\Omega)| \leq k\sqrt{12mn}$.*

Proof. Given a k-Lipschitz continuous neural newtork f, for all $x_1, x_2 \in X$, we have:

$$|f(x_1) - f(x_2)| \leq k\|x_1 - x_2\|. \tag{14}$$

Let $x_b, x_w \in X$ be the pure black and pure white images that maximize the Euclidean distance:

$$\|x_b - x_w\| = \sqrt{(-1-1)^2 \cdot m \cdot n \cdot 3} = \sqrt{12mn}. \tag{15}$$

Thus, we have:

$$\begin{aligned} |f(x_1) - f(x_2)| &\leq k\|x_1 - x_2\| \\ &\leq k\|x_b - x_w\| = k\sqrt{12mn}. \end{aligned} \tag{16}$$

Thus, the range of f is restricted to Ω, which satisfies:

$$|\min(\Omega) - \max(\Omega)| \leq k\sqrt{12mn} \tag{17}$$

□

B Detailed Experimental Setups

In our experiments, we use two variants of the standard CNN architecture (Radford et al., 2015; Arjovsky et al., 2017; Miyato et al., 2018) for the GANs to learn the distributions of the MNIST, CIFAR10 datasets at 32×32 resolution and the CelebA dataset (Liu et al., 2015) at 64×64 resolution. We use a batch size of 64 to train the GANs. We use an RMSProp optimizer with learning rate 0.00005. To make a fair comparison, we fix the number of discriminator updates in each iteration $n_{dis} = 1$ for all the GANs tested, i.e. we do not use multiple discriminator updates like (Arjovsky & Bottou, 2017; Arjovsky et al., 2017). Unless specified, we stop the training after 10^5 iterations. Details of the architectures are shown as follows. BN batch normalization; SN: spectral normalization; $c = 3$ for CIFAR10 dataset and $c = 1$ for MNIST dataset in input $x \in [-1, 1]^{64\times 64\times c}$.

$z \in \mathbb{R}^{100} \sim \mathcal{N}(0, 1)$
Reshape $\to 1 \times 1 \times 100$
4×4, stride=1, deconv. BN 512 ReLU
4×4, stride=2, deconv. BN 256 ReLU
4×4, stride=2, deconv. BN 128 ReLU
4×4, stride=2, deconv. BN 64 ReLU
4×4, stride=2, deconv. 3 Tanh

Generator

RGB image $x \in [-1, 1]^{32\times 32\times 3}$
4×4, stride=2, conv 64 SN lReLU(0.2)
4×4, stride=2, conv 128 SN lReLU(0.2)
4×4, stride=2, conv 256 SN lReLU(0.2)
4×4, stride=2, conv 512 SN lReLU(0.2)
4×4, stride=1, conv 1

Discriminator

(a) Network architectures (64×64 resolution).

$z \in \mathbb{R}^{100} \sim \mathcal{N}(0, 1)$
Reshape $\to 1 \times 1 \times 100$
4×4, stride=1, deconv. BN 256 ReLU
4×4, stride=2, deconv. BN 128 ReLU
4×4, stride=2, deconv. BN 64 ReLU
4×4, stride=2, deconv. 3 Tanh

Generator

Image $x \in [-1, 1]^{64\times 64\times c}$
4×4, stride=2, conv 64 SN lReLU(0.2)
4×4, stride=2, conv 128 SN lReLU(0.2)
4×4, stride=2, conv 256 SN lReLU(0.2)
4×4, stride=1, conv 1

Discriminator

(b) Network architectures (32×32 resolution).

C Restrictions of GAN loss Functions (Gradient Penalty)

Before spectral normalization, the best-performing Lipschitz regularizer is the gradient penalty (Gulrajani et al., 2017):

$$L = L_{GAN} + \lambda \mathop{\mathbb{E}}_{\hat{x} \in P_{\hat{x}}} [(\| \bigtriangledown_{\hat{x}} D(\hat{x})\| - k_{GP})^2], \tag{18}$$

where L_{GAN} is the GAN loss function without gradient penalty, λ is the weight of the gradient penalty term, $P_{\hat{x}}$ is the distribution of linearly interpolated samples between the target distribution and the model distribution, k_{GP} is the target gradient norm that controls the Lipschitz constant of the discriminator K. In general, the smaller k_{GP}, the smaller K. However, it is challenging to find a quantitative approximation between k_{GP} and K because the gradient penalty term $\lambda \mathbb{E}_{\hat{x} \in P_{\hat{x}}} [(\| \bigtriangledown_{\hat{x}} D(\hat{x})\| - k_{GP})^2]$ has no upper bound during training. Such a challenge makes gradient penalty less impactful in restricting the loss function.

As Figure 3 shows, we plot the domain Ω as intervals for different iterations under different k_{GP} and k_{SN} for the gradient penalty and the spectral normalization regularizers respectively. Similar to Gulrajani et al. (2017) and Fedus et al. (2018), we use $\lambda = 10$ for the gradient penalty. It can be observed that for both regularizers, the interval Ω shrinks as k_{GP} and k_{SN} decreases, which shows that the domain of the loss function is also restricted when gradient penalty is applied. However, k_{SN} is much more impactful than k_{GP} in restricting Ω, which is another reason for our choice of using spectral normalization to alter the strength of the Lipschitz regularization in this paper. The details of the architectures are shown below.

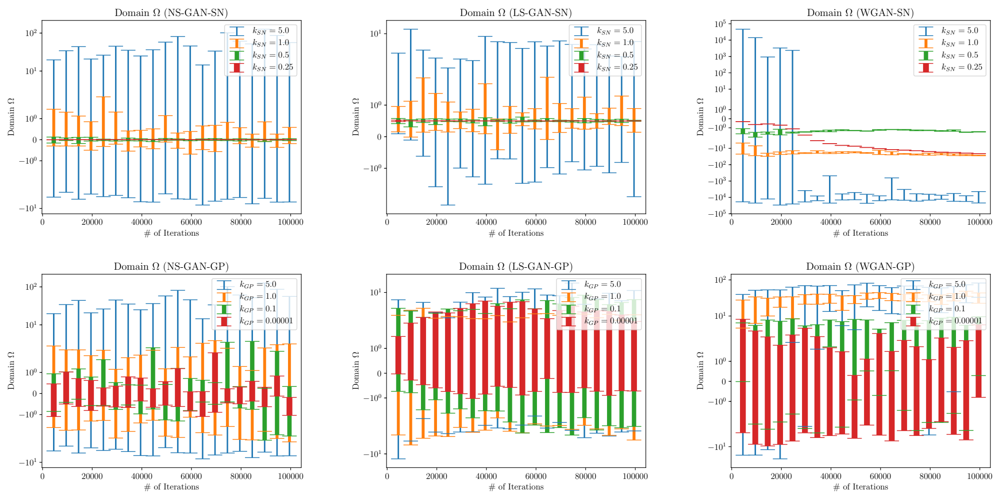


Figure 3: Relationship between domain Ω and k_{GP}, k_{SN} for different loss functions on CelebA dataset, where k_{GP}, k_{SN} are the parameters controlling the strength of the Lipschitz regularizers. The domain Ω shrinks with decreasing k_{GP} or k_{SN}. Each column shares the same loss function while each row shares the same Lipschitz regularizer. NS-GAN: Non-Saturating GAN (Goodfellow et al., 2014); LS-GAN: Least-Square GAN (Mao et al., 2017a); WGAN: Wasserstein GAN (Arjovsky et al., 2017); GP: gradient penalty (Gulrajani et al., 2017); SN: spectral normalization (Miyato et al., 2018). Note that the y-axis is in log scale.

$z \in \mathbb{R}^{100} \sim \mathcal{N}(0,1)$
Reshape $\rightarrow 1 \times 1 \times 100$
4×4, stride=1, deconv. BN 512 ReLU
4×4, stride=2, deconv. BN 256 ReLU
4×4, stride=2, deconv. BN 128 ReLU
4×4, stride=2, deconv. BN 64 ReLU
4×4, stride=2, deconv. 3 Tanh

Generator

RGB image $x \in [-1,1]^{64\times64\times3}$
4×4, stride=2, conv 64 BN lReLU(0.2)
4×4, stride=2, conv 128 BN lReLU(0.2)
4×4, stride=2, conv 256 BN lReLU(0.2)
4×4, stride=2, conv 512 BN lReLU(0.2)
4×4, stride=1, conv 1

Discriminator

(a) Network architectures with gradient penalty regularizer (64×64 resolution).

D Empirical Results on Domain Scaling

Table 5 shows the FID scores against α on the MNIST, CIFAR10 and CelebA datasets, where α is the hyper-parameter for domain scaling,

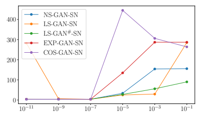
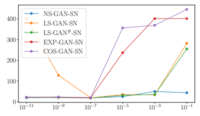
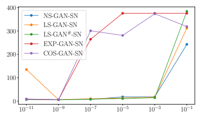
$$L_\alpha(\Omega) = L(\alpha \cdot \Omega)/\alpha. \tag{19}$$

Table 5 (a) shows that the FID scores of different loss functions generally improve with decreasing α, which is similar to those of k_{SN}. To further illustrate the similarity between α and k_{SN}, we show that a small α (e.g. $\alpha \leq 10^{-9}$) can compensate for the influence of an extremely large Lipschtiz constant ($K \approx k_{SN}^n = 50^4 = 6.25 \times 10^6$) and thereby stabilizes GAN training. Also, the FID scores when $\alpha \leq 10^{-11}$ are slightly worse than those when $\alpha \leq 10^{-9}$. As discussed in the main paper, the reason for this phenomenon is that restricting the domain of the loss function converges towards the performance of WGAN whose loss function does not have a unique minimum.

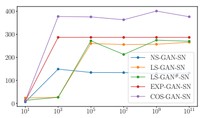
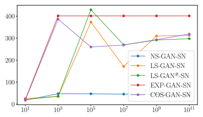
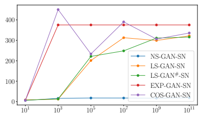
Table 5 (b) shows that the FID scores of different loss functions generally worsen with less restricted domains. Note that when $\alpha \geq 10^5$, the common practice of 1-Lipschitz regularization fails to stabilize the GAN training, which also implies that Lipschitz regularization only works as a scaling hyper-parameter of the neural network output.

Remark on LS-GAN and LS-GAN$^{\#}$. Note that the LS-GAN-SN has some abnormal behavior (e.g. $\alpha = 1e^{-11}$ in Table 5 (a) and $\alpha = 1e^{1}$ in Table 5 (b)) due to the conflict between its 0.5-centered domain and our zero-centered domain scaling method (Eq.11). This can be easily fixed by using the zero-centered LS-GAN$^{\#}$-SN (Mao et al., 2017b).

Table 5: FID scores *vs.* α. For the line plots, the x-axis shows α (in log scale) and the y-axis shows the FID scores. Lower FID scores are better.

Dataset	GANs	FID Scores						Line Plot
		$\alpha = 1e^{-11}$	$1e^{-9}$	$1e^{-7}$	$1e^{-5}$	$1e^{-3}$	$1e^{-1}$	
MNIST	NS-GAN-SN	3.67	3.61	3.81	33.48	154.14	155.54	
	LS-GAN-SN	281.82	6.48	3.74	24.93	29.31	285.46	
	LS-GAN$^{\#}$-SN	4.25	4.15	3.99	27.62	55.64	90.00	
	COS-GAN-SN	3.74	3.93	3.66	445.15	306.09	263.85	
	EXP-GAN-SN	4.14	4.01	3.54	134.76	286.96	286.96	
CIFAR10	NS-GAN-SN	19.58	22.46	18.73	24.57	49.56	43.42	
	LS-GAN-SN	400.91	127.38	18.68	34.78	33.17	282.11	
	LS-GAN$^{\#}$-SN	20.16	19.96	18.13	31.03	35.06	254.88	
	COS-GAN-SN	22.16	22.69	20.19	356.10	369.11	445.37	
	EXP-GAN-SN	21.93	21.70	18.50	236.77	401.24	401.24	
CelebA	NS-GAN-SN	9.08	7.05	7.84	18.51	18.41	242.64	
	LS-GAN-SN	135.17	6.57	10.67	13.39	17.42	311.93	
	LS-GAN$^{\#}$-SN	6.66	5.68	8.72	11.13	14.90	383.61	
	COS-GAN-SN	8.00	6.31	300.55	280.84	373.31	318.53	
	EXP-GAN-SN	8.85	6.09	264.49	375.32	375.32	375.32	

(a) $k_{SN} = 50.0$

Dataset	GANs	FID Scores						Line Plot
		$\alpha = 1e^{1}$	$1e^{3}$	$1e^{5}$	$1e^{7}$	$1e^{9}$	$1e^{11}$	
MNIST	NS-GAN-SN	6.55	148.97	134.44	133.82	130.21	131.87	
	LS-GAN-SN	23.37	26.96	260.05	255.73	256.96	265.76	
	LS-GAN$^{\#}$-SN	13.43	26.51	271.85	212.74	274.63	269.96	
	COS-GAN-SN	11.79	377.62	375.72	363.45	401.12	376.39	
	EXP-GAN-SN	11.02	286.96	286.96	286.96	286.96	286.96	
CIFAR10	NS-GAN-SN	17.63	47.31	46.85	45.44	45.67	39.90	
	LS-GAN-SN	25.55	34.44	373.07	171.18	309.55	312.96	
	LS-GAN$^{\#}$-SN	20.45	36.18	429.21	269.63	291.55	297.71	
	COS-GAN-SN	18.59	386.24	259.83	268.89	293.29	318.65	
	EXP-GAN-SN	21.56	401.24	401.24	401.24	401.24	401.24	
CelebA	NS-GAN-SN	5.88	16.14	17.75	17.67	16.87	18.81	
	LS-GAN-SN	8.41	12.09	201.22	312.83	299.30	321.84	
	LS-GAN$^{\#}$-SN	7.21	13.13	221.41	248.48	311.21	315.94	
	COS-GAN-SN	6.62	450.57	233.42	390.40	306.17	335.87	
	EXP-GAN-SN	6.91	375.32	375.32	375.32	375.32	375.32	

(b) $k_{SN} = 1.0$

E Additional Qualitative Results

In this section, we show the qualitative results (sample images) corresponding to the quantitative experiments in the main paper. The FID scores and line plots are shown together with the samples.

E.1 Samples of FID scores *vs.* k_{SN} Experiment

This subsection corresponds to the FID scores *vs.* k_{SN} experiment (Table 2 in the main paper).

With varying k_{SN} on the MNIST dataset,

- Figure 4 shows sample images of the **NS-GAN-SN**;
- Figure 5 shows sample images of the **LS-GAN-SN**;
- Figure 6 shows sample images of the **WGAN-SN**;
- Figure 7 shows sample images of the **COS-GAN-SN**;

- Figure 8 shows sample images of the **EXP-GAN-SN**.

With varying k_{SN} on the CIFAR10 dataset,

- Figure 9 shows sample images of the **NS-GAN-SN**;
- Figure 10 shows sample images of the **LS-GAN-SN**;
- Figure 11 shows sample images of the **WGAN-SN**;
- Figure 12 shows sample images of the **COS-GAN-SN**;
- Figure 13 shows sample images of the **EXP-GAN-SN**.

With varying k_{SN} on the CelebA dataset,

- Figure 14 shows sample images of the **NS-GAN-SN**;
- Figure 15 shows sample images of the **LS-GAN-SN**;
- Figure 16 shows sample images of the **WGAN-SN**;
- Figure 17 shows sample images of the **COS-GAN-SN**;
- Figure 18 shows sample images of the **EXP-GAN-SN**.

E.2 Samples of the FID scores *vs.* α Experiment

This subsection corresponds to the FID scores *vs.* α experiment (Table 5). The results show that instead of the restriction of the neural network gradients, the restriction of the loss function is the dominating factor of Lipschitz regularization.

Results for Table 5 (a) With varying α, $k_{SN} = 50.0$ and on MNIST dataset,

- Figure 19 shows sample images of the **NS-GAN-SN**;
- Figure 20 shows sample images of the **LS-GAN-SN**;
- Figure 21 shows sample images of the **LS-GAN$^{\#}$-SN**;
- Figure 22 shows sample images of the **EXP-GAN-SN**;
- Figure 23 shows sample images of the **COS-GAN-SN**.

With varying α, $k_{SN} = 50.0$ and on CIFAR10 dataset,

- Figure 24 shows sample images of the **NS-GAN-SN**;
- Figure 25 shows sample images of the **LS-GAN-SN**;
- Figure 26 shows sample images of the **LS-GAN$^{\#}$-SN**;
- Figure 27 shows sample images of the **EXP-GAN-SN**;
- Figure 28 shows sample images of the **COS-GAN-SN**.

With varying α, $k_{SN} = 50.0$ and on CelebA dataset,

- Figure 29 shows sample images of the **NS-GAN-SN**;
- Figure 30 shows sample images of the **LS-GAN-SN**;
- Figure 31 shows sample images of the **LS-GAN$^{\#}$-SN**;
- Figure 32 shows sample images of the **EXP-GAN-SN**;
- Figure 33 shows sample images of the **COS-GAN-SN**.

Results for Table 5 (b) With varying α, $k_{SN} = 1.0$ and on MNIST dataset,

- Figure 34 shows sample images of the **NS-GAN-SN**;
- Figure 35 shows sample images of the **LS-GAN-SN**;
- Figure 36 shows sample images of the **LS-GAN$^{\#}$-SN**;
- Figure 37 shows sample images of the **EXP-GAN-SN**;
- Figure 38 shows sample images of the **COS-GAN-SN**.

With varying α, $k_{SN} = 1.0$ and on CIFAR10 dataset,

- Figure 39 shows sample images of the **NS-GAN-SN**;
- Figure 40 shows sample images of the **LS-GAN-SN**;
- Figure 41 shows sample images of the **LS-GAN$^{\#}$-SN**;

- Figure 42 shows sample images of the **EXP-GAN-SN**;
- Figure 43 shows sample images of the **COS-GAN-SN**.

With varying α, $k_{SN} = 1.0$ and on CelebA dataset,

- Figure 44 shows sample images of the **NS-GAN-SN**;
- Figure 45 shows sample images of the **LS-GAN-SN**;
- Figure 46 shows sample images of the **LS-GAN$^{\#}$-SN**;
- Figure 47 shows sample images of the **EXP-GAN-SN**;
- Figure 48 shows sample images of the **COS-GAN-SN**.

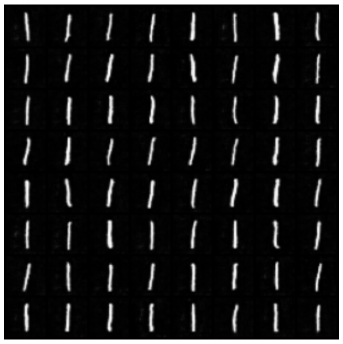
E.3 Samples of the Degenerate Loss Function

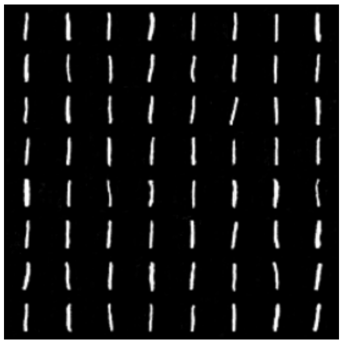
Figure 49 shows the sample images of WGAN-SN and some extremely degenerate loss functions, which corresponds to the following table in the main paper. It can be observed that all loss functions have similar performance.

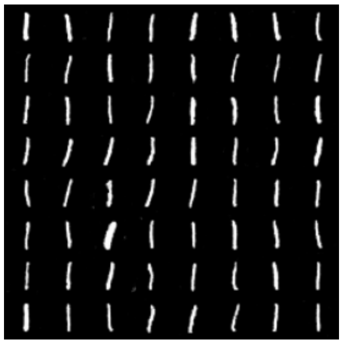
GANs	FID Scores		
	MNIST	CIFAR10	CELEBA
WGAN-SN	3.71	23.36	7.82
NS-GAN-SN	3.74	21.92	8.10
LS-GAN$^{\#}$-SN	3.81	21.47	8.51
COS-GAN-SN	3.96	23.65	8.30
EXP-GAN-SN	3.86	21.91	8.22

FID scores *vs.* k_{SN} of Different Loss Functions

- MNIST -


(a) $k_{SN} = 50.0$, FID $= 155.41$

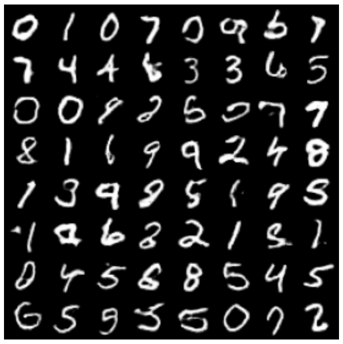

(b) $k_{SN} = 10.0$, FID $= 156.60$


(c) $k_{SN} = 5.0$, FID $= 144.28$

(d) $k_{SN} = 1.0$, FID $= 3.90$

(e) $k_{SN} = 0.5$, FID $= 4.20$

(f) $k_{SN} = 0.25$, FID $= 3.99$


(g) $k_{SN} = 0.2$, FID $= 5.41$

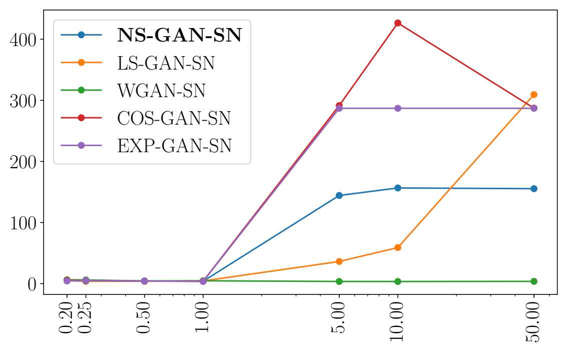
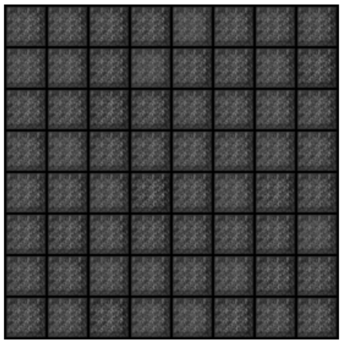


Figure 4: Samples of randomly generated images with NS-GAN-SN of varying k_{SN} (MNIST). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.


(a) $k_{SN} = 50.0$, FID $= 309.35$

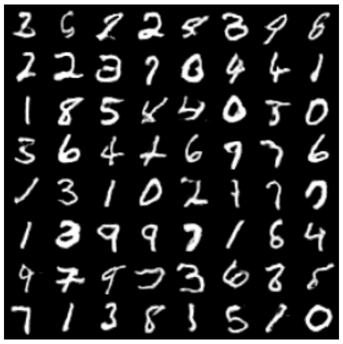
(b) $k_{SN} = 10.0$, FID $= 59.04$

(c) $k_{SN} = 5.0$, FID $= 36.26$

(d) $k_{SN} = 1.0$, FID $= 4.42$

(e) $k_{SN} = 0.5$, FID $= 3.90$

(f) $k_{SN} = 0.25$, FID $= 3.96$


(g) $k_{SN} = 0.2$, FID $= 5.14$

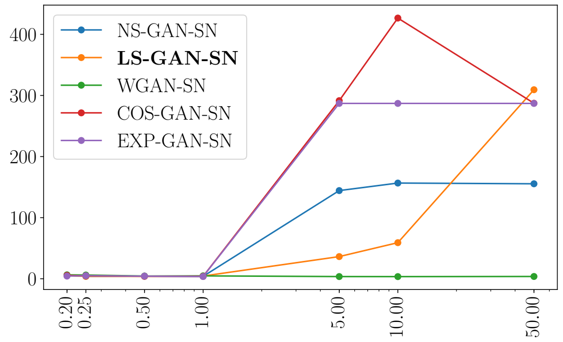


Figure 5: Samples of randomly generated images with LS-GAN-SN of varying k_{SN} (MNIST). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.

(a) k_{SN} = 50.0, FID = 3.71

(b) k_{SN} = 10.0, FID = 3.50

(c) k_{SN} = 5.0, FID = 3.58

(d) k_{SN} = 1.0, FID = 4.70

(e) k_{SN} = 0.5, FID = 4.44

(f) k_{SN} = 0.25, FID = 6.06

(g) k_{SN} = 0.2, FID = 6.35

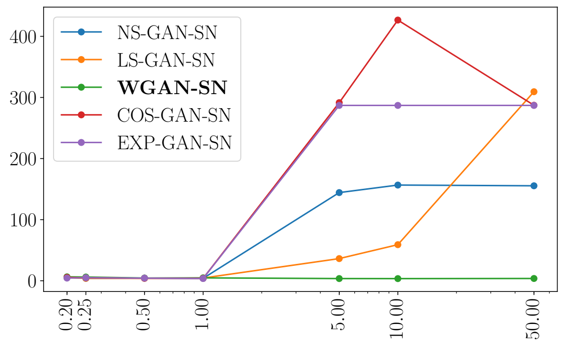
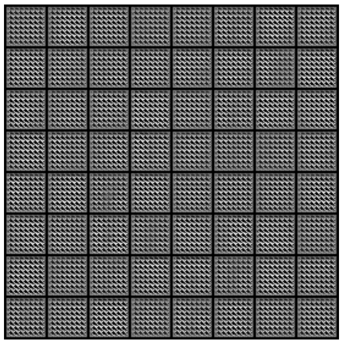
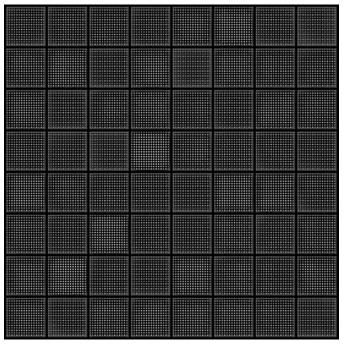
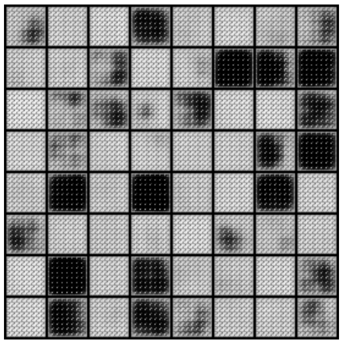


Figure 6: Samples of randomly generated images with WGAN-SN of varying k_{SN} (MNIST). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.


(a) $k_{SN} = 50.0$, FID $= 287.23$

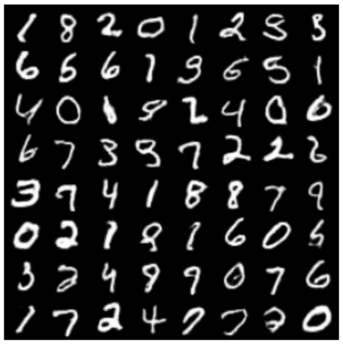

(b) $k_{SN} = 10.0$, FID $= 426.62$


(c) $k_{SN} = 5.0$, FID $= 291.44$

(d) $k_{SN} = 1.0$, FID $= 3.86$

(e) $k_{SN} = 0.5$, FID $= 4.05$

(f) $k_{SN} = 0.25$, FID $= 4.83$


(g) $k_{SN} = 0.2$, FID $= 5.41$

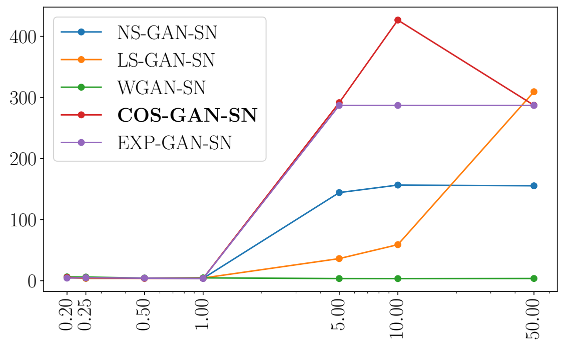
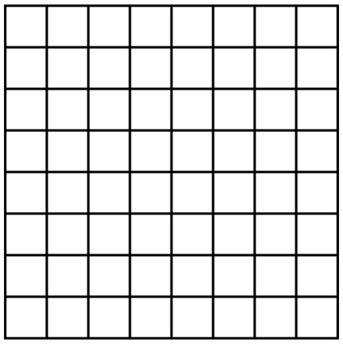
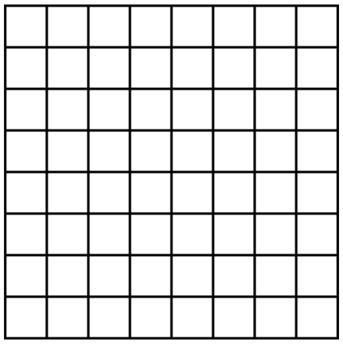
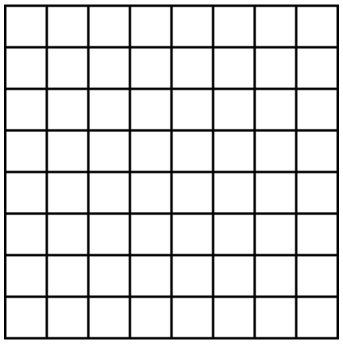


Figure 7: Samples of randomly generated images with COS-GAN-SN of varying k_{SN} (MNIST). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.


(a) $k_{SN} = 50.0$, FID $= 286.96$

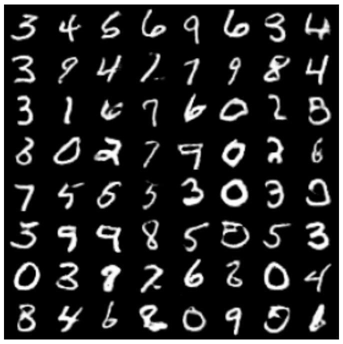

(b) $k_{SN} = 10.0$, FID $= 286.96$


(c) $k_{SN} = 5.0$, FID $= 286.96$

(d) $k_{SN} = 1.0$, FID $= 3.69$

(e) $k_{SN} = 0.5$, FID $= 4.25$

(f) $k_{SN} = 0.25$, FID $= 4.93$


(g) $k_{SN} = 0.2$, FID $= 4.38$

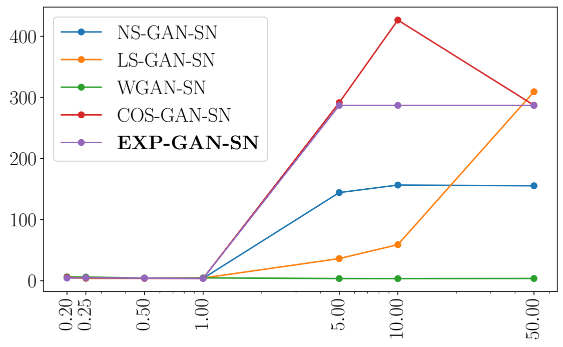
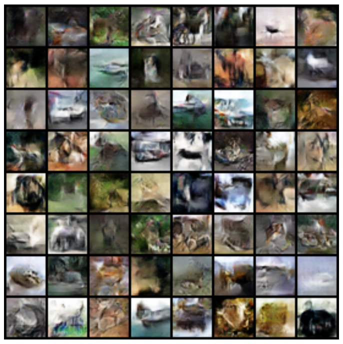


Figure 8: Samples of randomly generated images with EXP-GAN-SN of varying k_{SN} (MNIST). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.

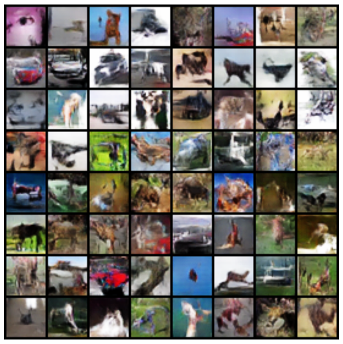
FID scores *vs.* k_{SN} of Different Loss Functions

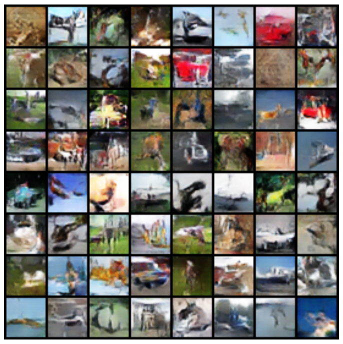
- CIFAR10 -

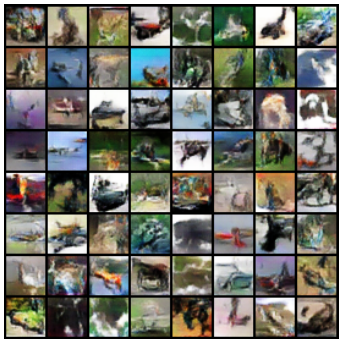

(a) $k_{SN} = 50.0$, FID $= 48.03$

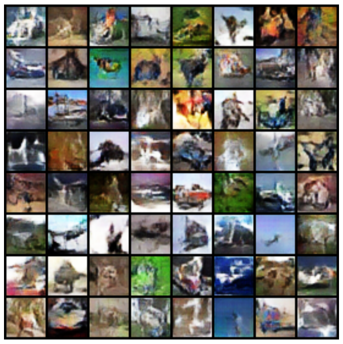
(b) $k_{SN} = 10.0$, FID $= 49.67$

(c) $k_{SN} = 5.0$, FID $= 41.04$


(d) $k_{SN} = 1.0$, FID $= 15.81$


(e) $k_{SN} = 0.5$, FID $= 23.29$


(f) $k_{SN} = 0.25$, FID $= 24.37$


(g) $k_{SN} = 0.2$, FID $= 29.23$

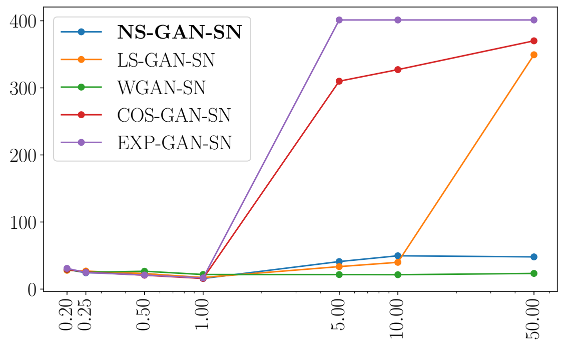
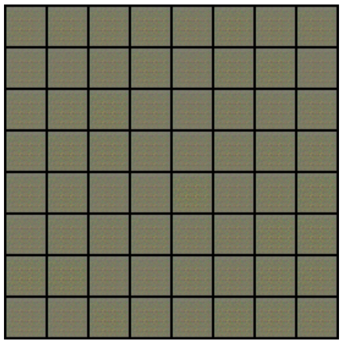
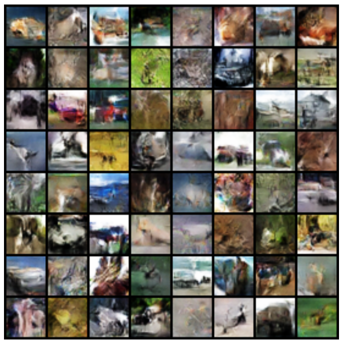
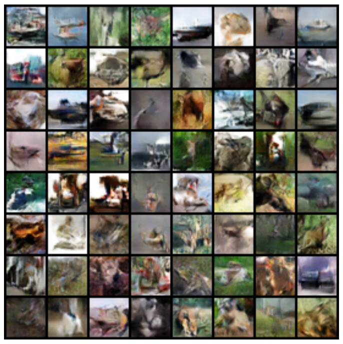
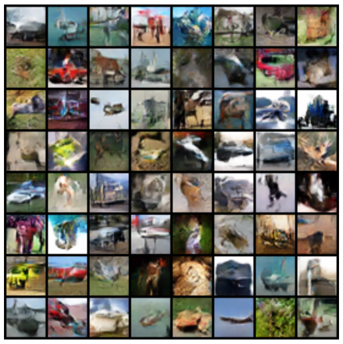


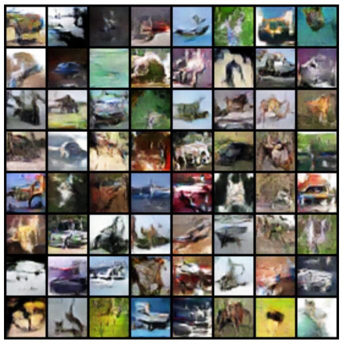
Figure 9: Samples of randomly generated images with NS-GAN-SN of varying k_{SN} (CIFAR10). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.

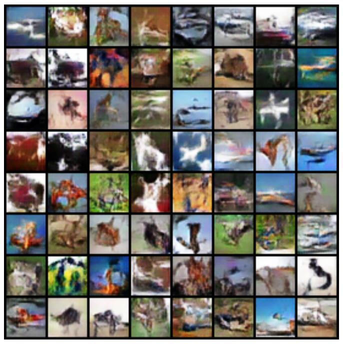

(a) $k_{SN} = 50.0$, FID $= 349.35$

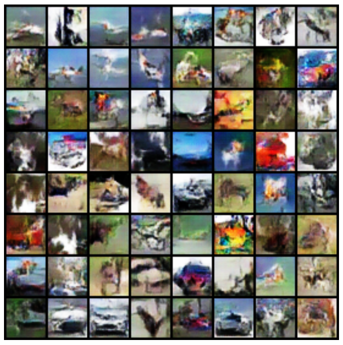

(b) $k_{SN} = 10.0$, FID $= 39.90$


(c) $k_{SN} = 5.0$, FID $= 33.53$


(d) $k_{SN} = 1.0$, FID $= 17.30$


(e) $k_{SN} = 0.5$, FID $= 23.14$


(f) $k_{SN} = 0.25$, FID $= 26.85$


(g) $k_{SN} = 0.2$, FID $= 28.04$

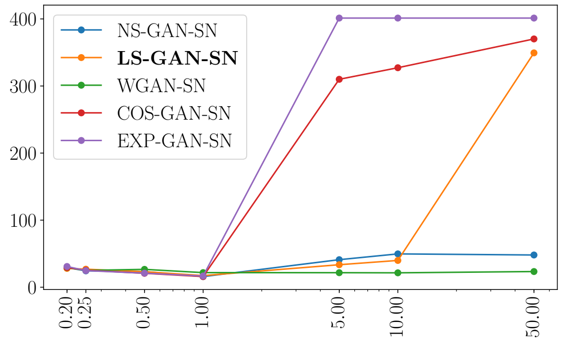
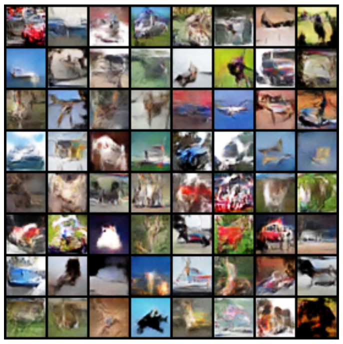
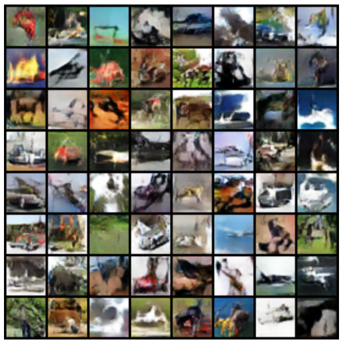
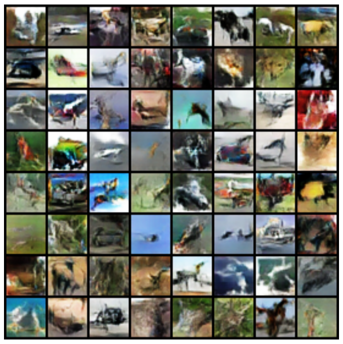
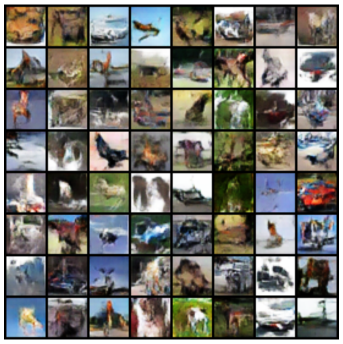


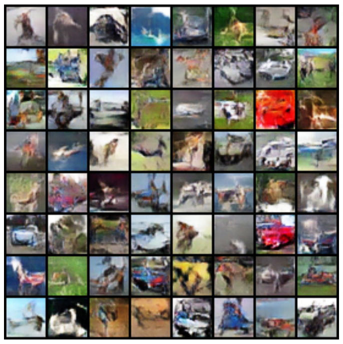
Figure 10: Samples of randomly generated images with LS-GAN-SN of varying k_{SN} (CIFAR10). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.

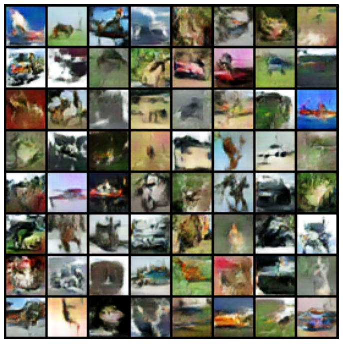

(a) $k_{SN} = 50.0$, FID $= 23.36$

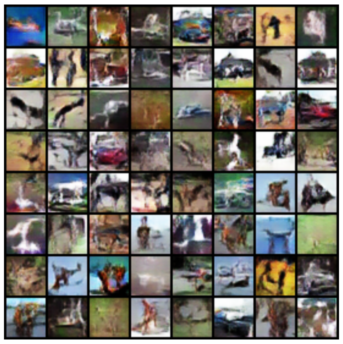

(b) $k_{SN} = 10.0$, FID $= 21.45$


(c) $k_{SN} = 5.0$, FID $= 21.63$


(d) $k_{SN} = 1.0$, FID $= 21.75$


(e) $k_{SN} = 0.5$, FID $= 26.61$


(f) $k_{SN} = 0.25$, FID $= 25.07$


(g) $k_{SN} = 0.2$, FID $= 29.20$

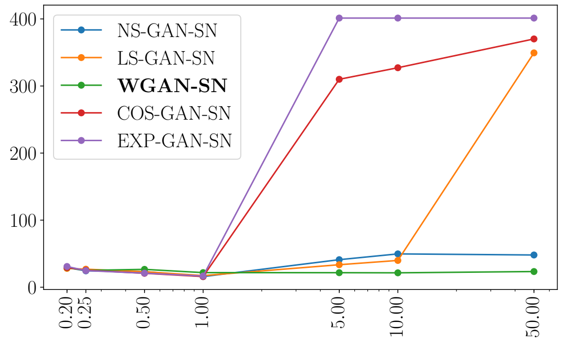
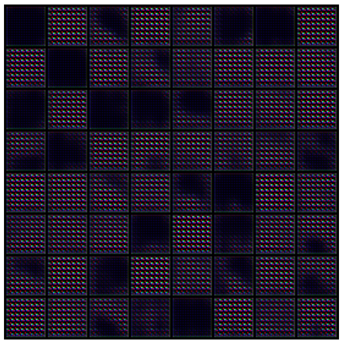
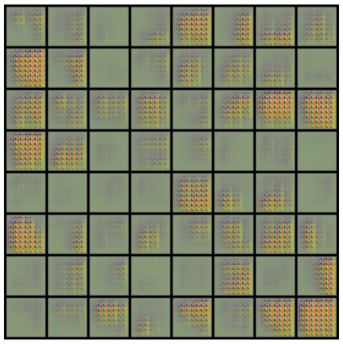
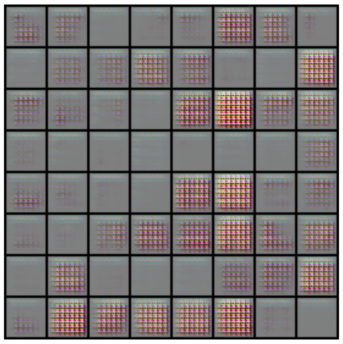


Figure 11: Samples of randomly generated images with WGAN-SN of varying k_{SN} (CIFAR10). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.


(a) $k_{SN} = 50.0$, FID $= 370.13$

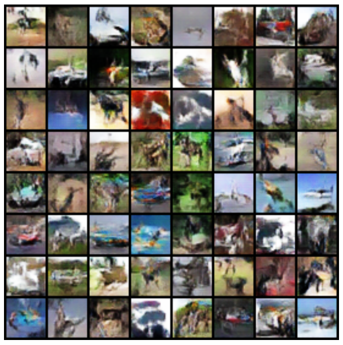

(b) $k_{SN} = 10.0$, FID $= 327.20$


(c) $k_{SN} = 5.0$, FID $= 309.96$

(d) $k_{SN} = 1.0$, FID $= 15.88$

(e) $k_{SN} = 0.5$, FID $= 20.73$

(f) $k_{SN} = 0.25$, FID $= 25.31$


(g) $k_{SN} = 0.2$, FID $= 29.45$

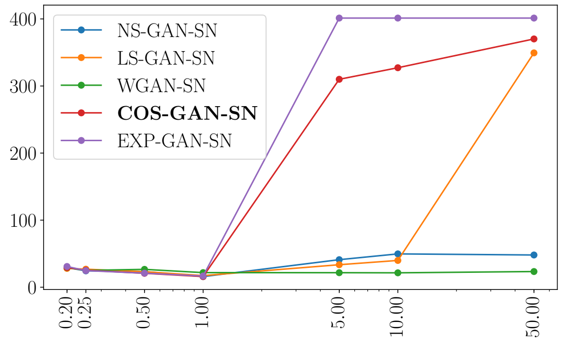
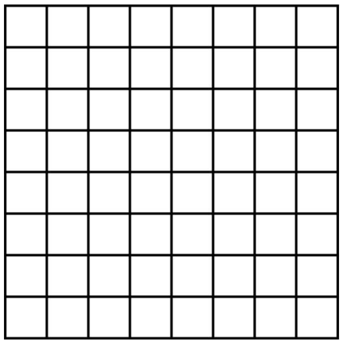
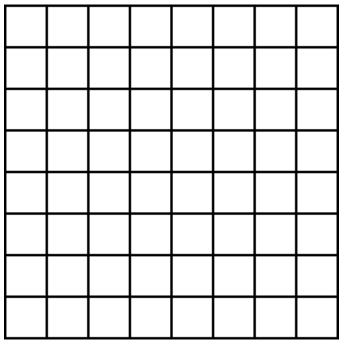
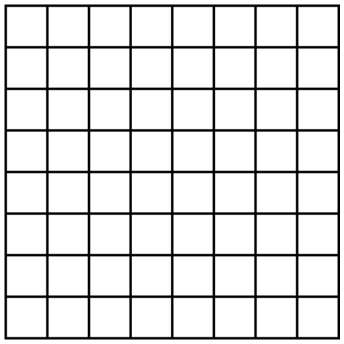
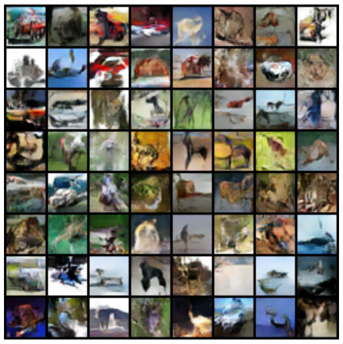


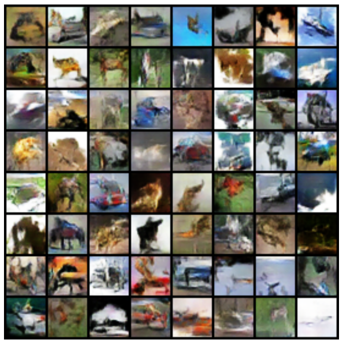
Figure 12: Samples of randomly generated images with COS-GAN-SN of varying k_{SN} (CIFAR10). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.

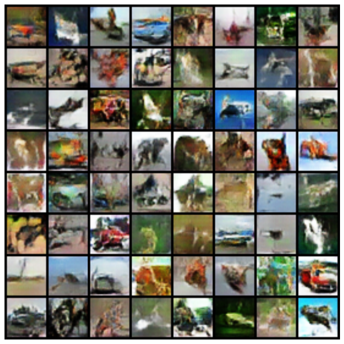

(a) $k_{SN} = 50.0$, FID $= 401.24$

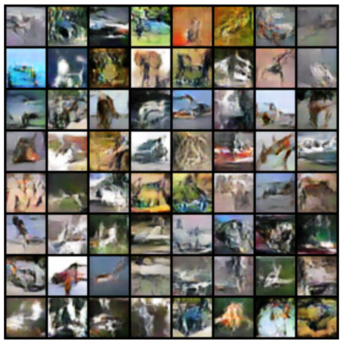

(b) $k_{SN} = 10.0$, FID $= 401.24$


(c) $k_{SN} = 5.0$, FID $= 401.24$


(d) $k_{SN} = 1.0$, FID $= 16.66$


(e) $k_{SN} = 0.5$, FID $= 20.90$


(f) $k_{SN} = 0.25$, FID $= 24.74$


(g) $k_{SN} = 0.2$, FID $= 30.89$

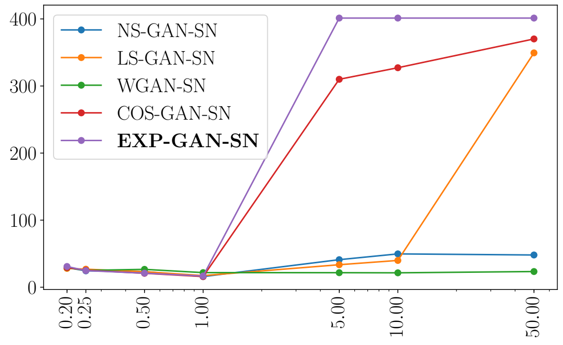
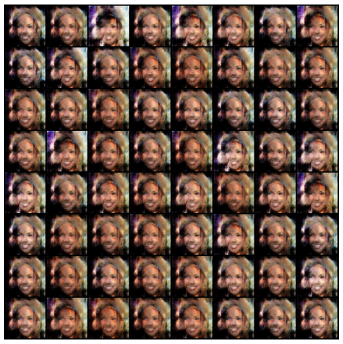


Figure 13: Samples of randomly generated images with EXP-GAN-SN of varying k_{SN} (CIFAR10). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.

FID scores *vs.* k_{SN} of Different Loss Functions

- CelebA -


(a) $k_{SN} = 50.0$, FID $= 184.06$

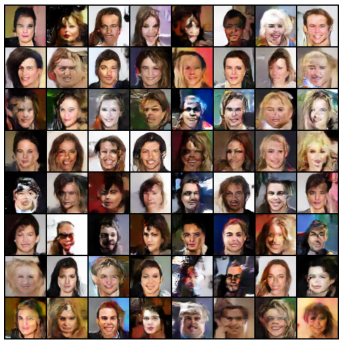
(b) $k_{SN} = 10.0$, FID $= 17.04$

(c) $k_{SN} = 5.0$, FID $= 18.95$

(d) $k_{SN} = 1.0$, FID $= 6.11$

(e) $k_{SN} = 0.5$, FID $= 8.04$

(f) $k_{SN} = 0.25$, FID $= 12.71$


(g) $k_{SN} = 0.2$, FID $= 18.59$

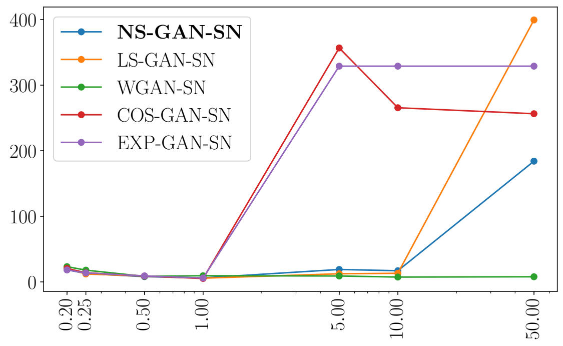
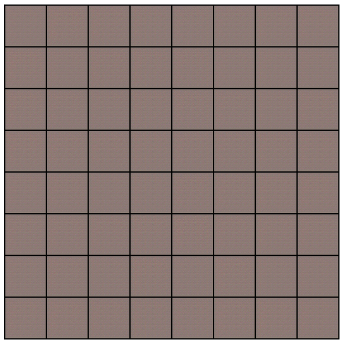
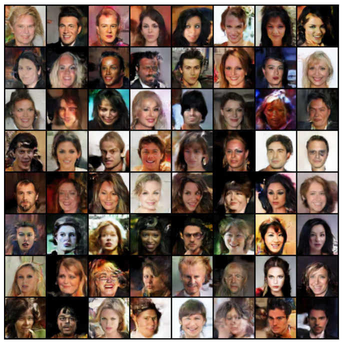


Figure 14: Samples of randomly generated images with NS-GAN-SN of varying k_{SN} (CelebA). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.

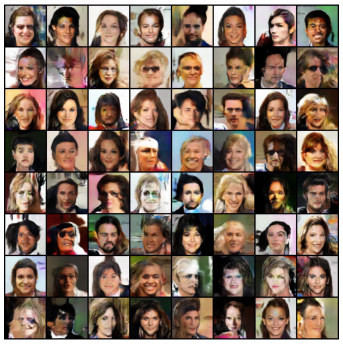

(a) $k_{SN} = 50.0$, FID $= 399.39$

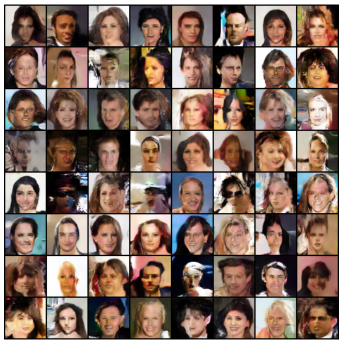
(b) $k_{SN} = 10.0$, FID $= 13.14$


(c) $k_{SN} = 5.0$, FID $= 12.40$

(d) $k_{SN} = 1.0$, FID $= 5.69$

(e) $k_{SN} = 0.5$, FID $= 8.85$


(f) $k_{SN} = 0.25$, FID $= 12.14$


(g) $k_{SN} = 0.2$, FID $= 20.34$

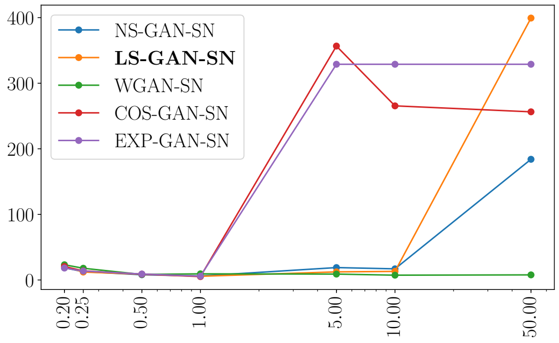


Figure 15: Samples of randomly generated images with LS-GAN-SN of varying k_{SN} (CelebA). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.

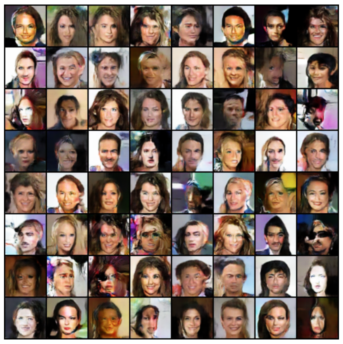
(a) $k_{SN} = 50.0$, FID $= 7.82$

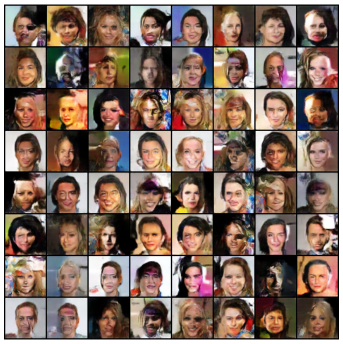
(b) $k_{SN} = 10.0$, FID $= 7.37$

(c) $k_{SN} = 5.0$, FID $= 9.03$

(d) $k_{SN} = 1.0$, FID $= 9.41$

(e) $k_{SN} = 0.5$, FID $= 8.48$


(f) $k_{SN} = 0.25$, FID $= 17.93$


(g) $k_{SN} = 0.2$, FID $= 23.26$

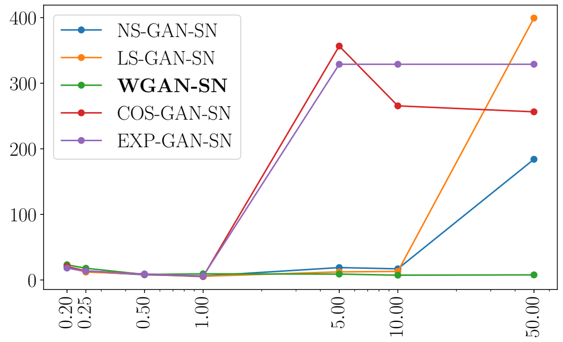
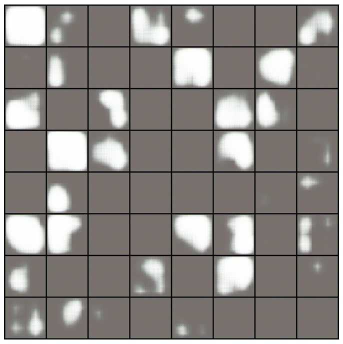
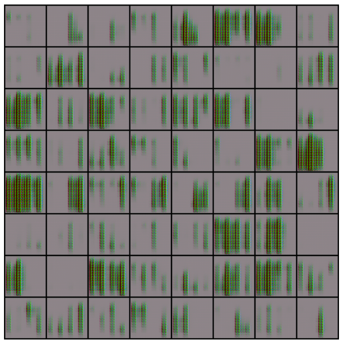
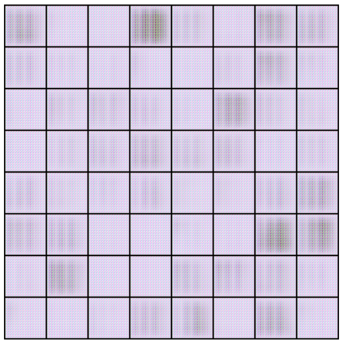
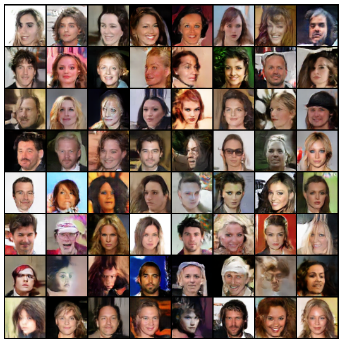


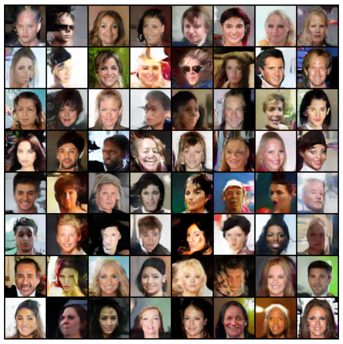
Figure 16: Samples of randomly generated images with WGAN-SN of varying k_{SN} (CelebA). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.


(a) $k_{SN} = 50.0$, FID $= 256.44$

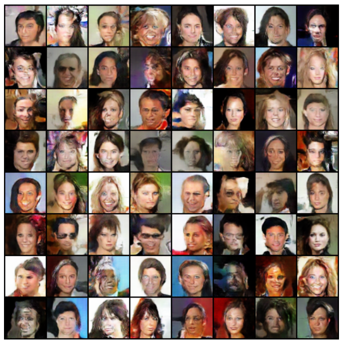

(b) $k_{SN} = 10.0$, FID $= 265.53$


(c) $k_{SN} = 5.0$, FID $= 356.70$


(d) $k_{SN} = 1.0$, FID $= 5.20$


(e) $k_{SN} = 0.5$, FID $= 8.88$

(f) $k_{SN} = 0.25$, FID $= 13.93$


(g) $k_{SN} = 0.2$, FID $= 20.59$

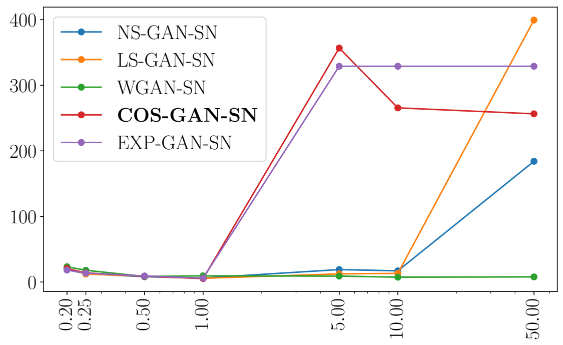
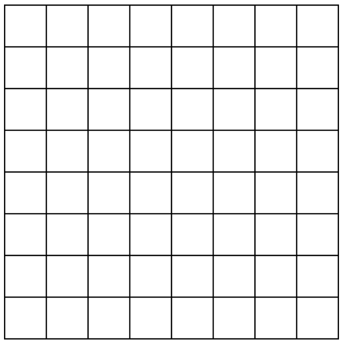
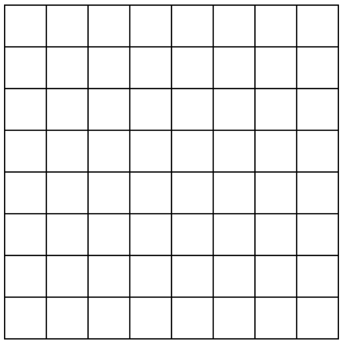
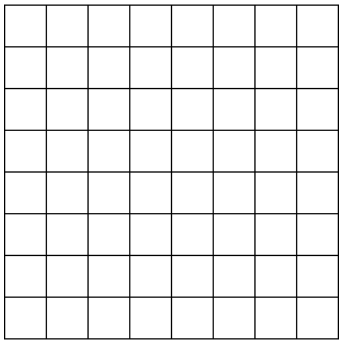


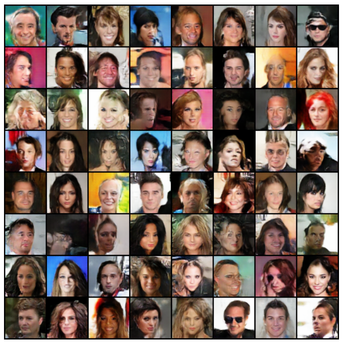
Figure 17: Samples of randomly generated images with COS-GAN-SN of varying k_{SN} (CelebA). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.

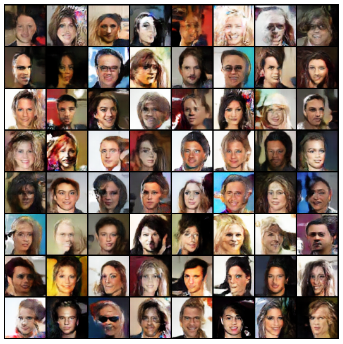

(a) $k_{SN} = 50.0$, FID $= 328.94$

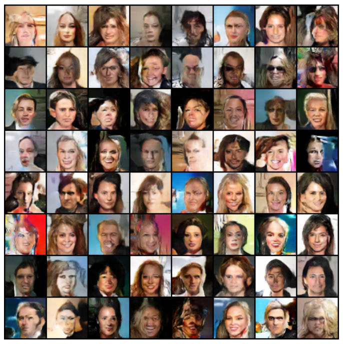

(b) $k_{SN} = 10.0$, FID $= 328.94$


(c) $k_{SN} = 5.0$, FID $= 328.94$

(d) $k_{SN} = 1.0$, FID $= 5.88$


(e) $k_{SN} = 0.5$, FID $= 9.18$


(f) $k_{SN} = 0.25$, FID $= 13.65$


(g) $k_{SN} = 0.2$, FID $= 18.23$

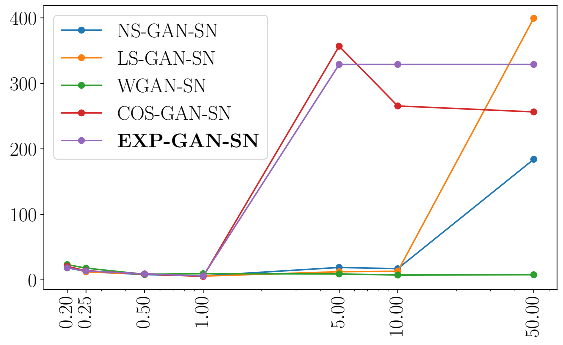


Figure 18: Samples of randomly generated images with EXP-GAN-SN of varying k_{SN} (CelebA). For the line plot, x-axis shows k_{SN} (in log scale) and y-axis shows the FID scores.

FID scores *vs.* α ($k_{SN} = 50.0$) of Different Loss Functions

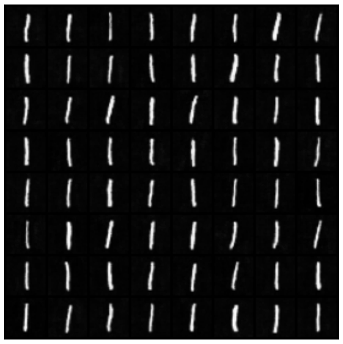
- MNIST -

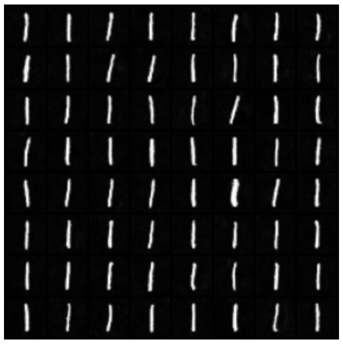
(a) $\alpha = 1e^{-11}$, FID $= 3.67$

(b) $\alpha = 1e^{-9}$, FID $= 3.61$

(c) $\alpha = 1e^{-7}$, FID $= 3.81$

(d) $\alpha = 1e^{-5}$, FID $= 33.48$


(e) $\alpha = 1e^{-3}$, FID $= 154.14$


(f) $\alpha = 1e^{-1}$, FID $= 155.54$

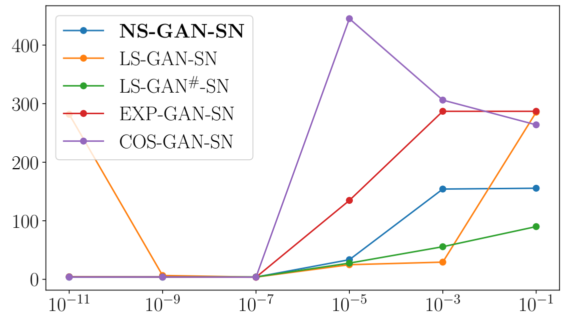
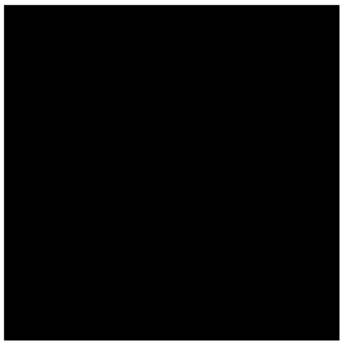


Figure 19: Samples of randomly generated images with NS-GAN-SN of varying α ($k_{SN} = 50.0$, MNIST). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

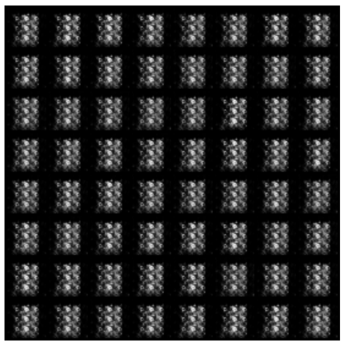

(a) $\alpha = 1e^{-11}$, FID = 281.82

(b) $\alpha = 1e^{-9}$, FID = 6.48

(c) $\alpha = 1e^{-7}$, FID = 3.74

(d) $\alpha = 1e^{-5}$, FID = 24.93

(e) $\alpha = 1e^{-3}$, FID = 29.31


(f) $\alpha = 1e^{-1}$, FID = 285.46

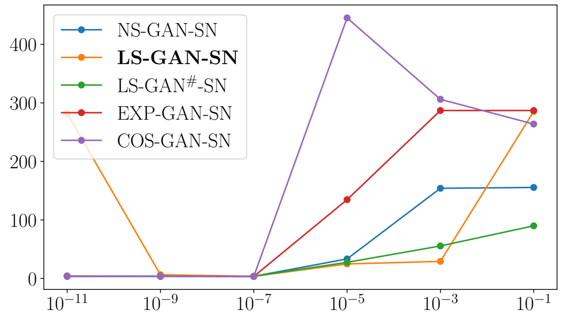


Figure 20: Samples of randomly generated images with LS-GAN-SN of varying α ($k_{SN} = 50.0$, MNIST). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

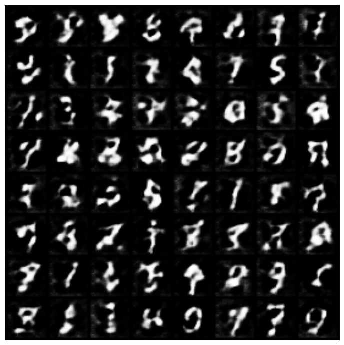
(a) $\alpha = 1e^{-11}$, FID $= 4.25$

(b) $\alpha = 1e^{-9}$, FID $= 4.15$

(c) $\alpha = 1e^{-7}$, FID $= 3.99$

(d) $\alpha = 1e^{-5}$, FID $= 27.62$

(e) $\alpha = 1e^{-3}$, FID $= 55.64$


(f) $\alpha = 1e^{-1}$, FID $= 90.00$

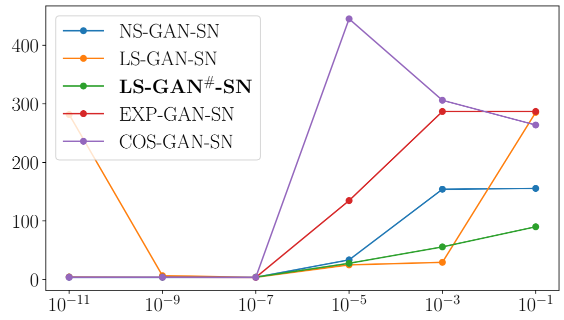
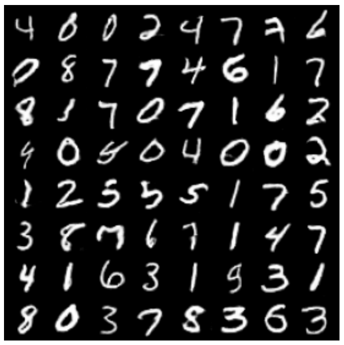
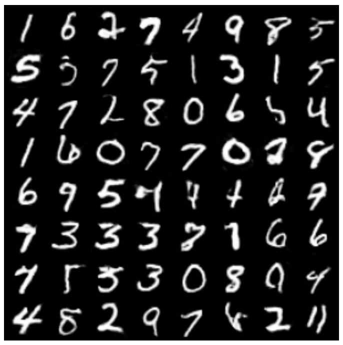
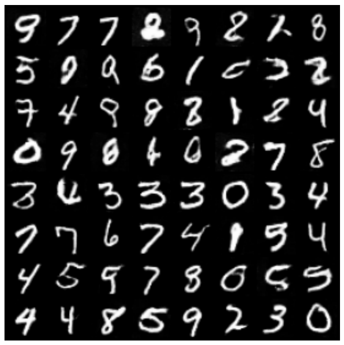
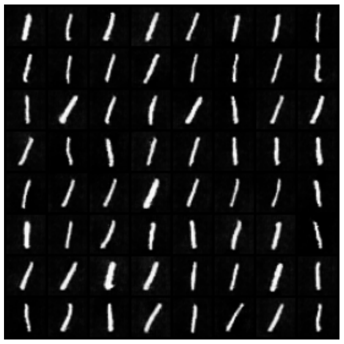


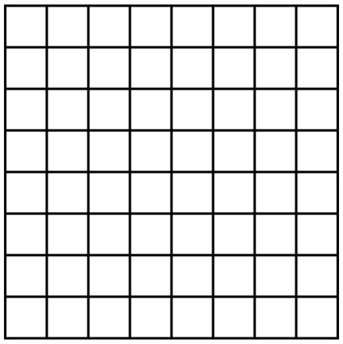
Figure 21: Samples of randomly generated images with LS-GAN$^{\#}$-SN of varying α ($k_{SN} = 50.0$, MNIST). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

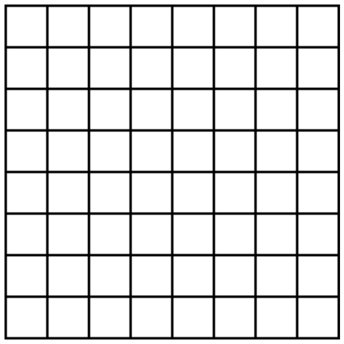

(a) $\alpha = 1e^{-11}$, FID $= 4.14$


(b) $\alpha = 1e^{-9}$, FID $= 4.01$


(c) $\alpha = 1e^{-7}$, FID $= 3.54$


(d) $\alpha = 1e^{-5}$, FID $= 134.76$


(e) $\alpha = 1e^{-3}$, FID $= 286.96$


(f) $\alpha = 1e^{-1}$, FID $= 286.96$

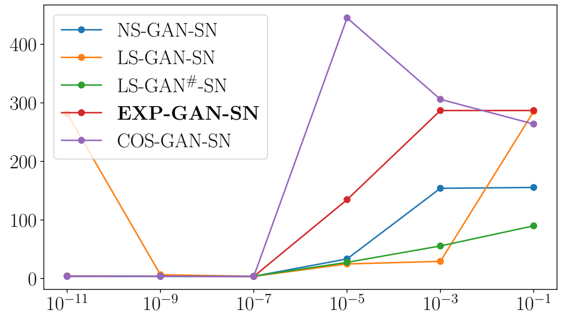
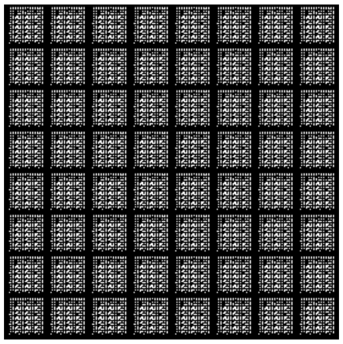


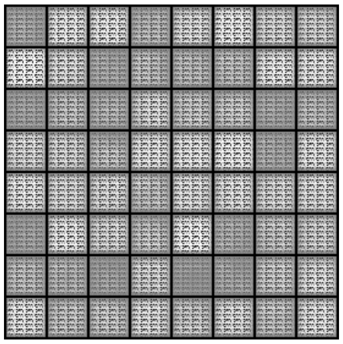
Figure 22: Samples of randomly generated images with EXP-GAN-SN of varying α ($k_{SN} = 50.0$, MNIST). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

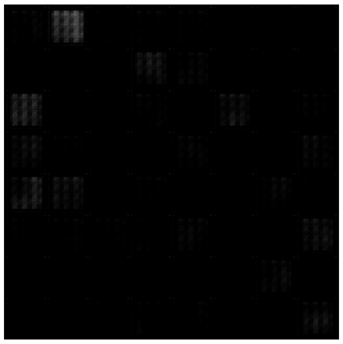
(a) $\alpha = 1e^{-11}$, FID $= 3.74$

(b) $\alpha = 1e^{-9}$, FID $= 3.93$

(c) $\alpha = 1e^{-7}$, FID $= 3.66$


(d) $\alpha = 1e^{-5}$, FID $= 445.15$


(e) $\alpha = 1e^{-3}$, FID $= 306.09$


(f) $\alpha = 1e^{-1}$, FID $= 263.85$

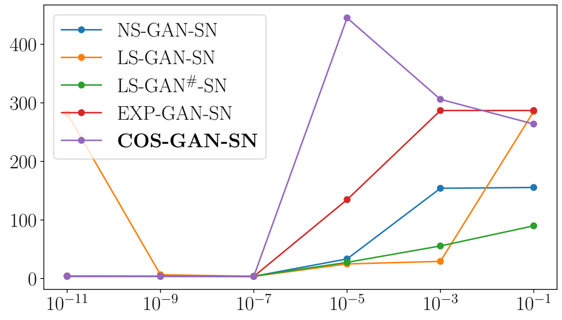
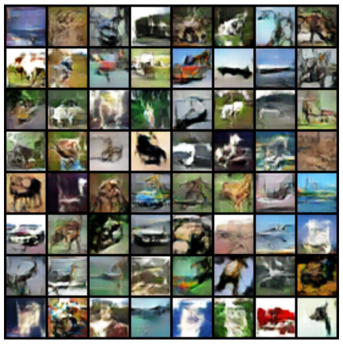
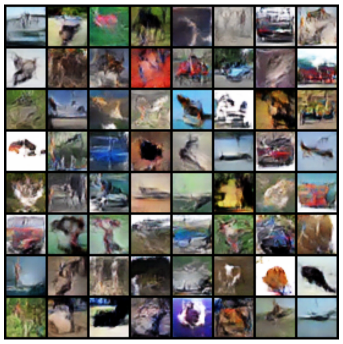


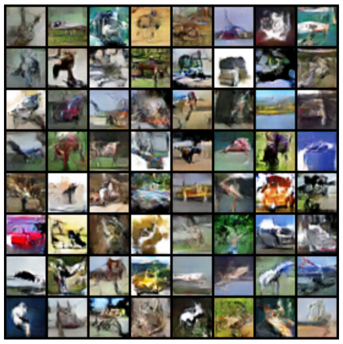
Figure 23: Samples of randomly generated images with COS-GAN-SN of varying α ($k_{SN} = 50.0$, MNIST). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

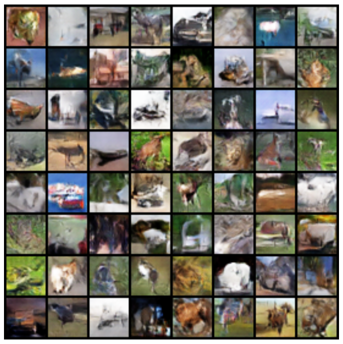
FID scores *vs.* α ($k_{SN} = 50.0$) of Different Loss Functions

- CIFAR10 -

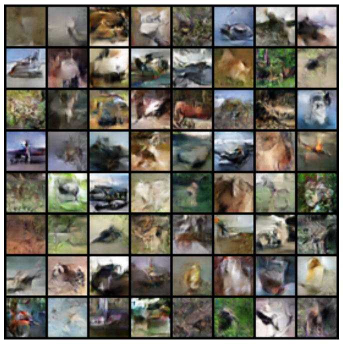

(a) $\alpha = 1e^{-11}$, FID $= 19.58$


(b) $\alpha = 1e^{-9}$, FID $= 22.46$


(c) $\alpha = 1e^{-7}$, FID $= 18.73$


(d) $\alpha = 1e^{-5}$, FID $= 24.57$

(e) $\alpha = 1e^{-3}$, FID $= 49.56$


(f) $\alpha = 1e^{-1}$, FID $= 43.42$

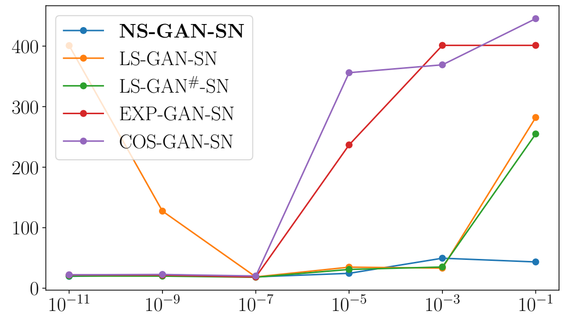
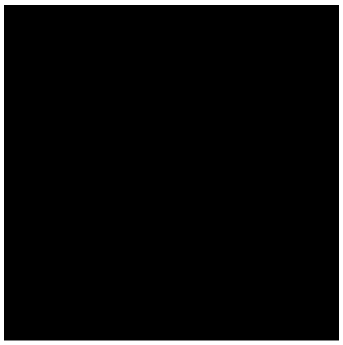
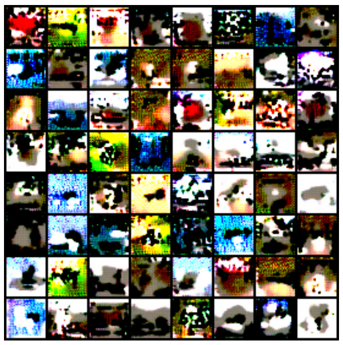
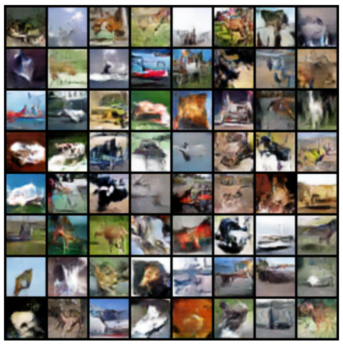


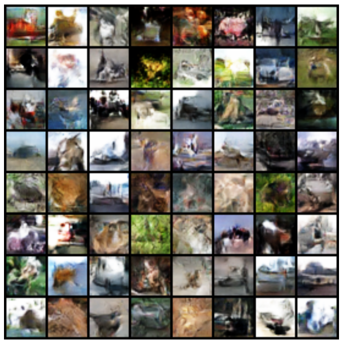
Figure 24: Samples of randomly generated images with NS-GAN-SN of varying α ($k_{SN} = 50.0$, CIFAR10). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

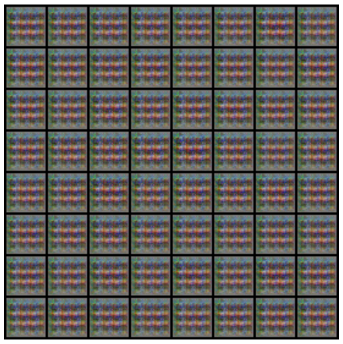

(a) $\alpha = 1e^{-11}$, FID $= 400.91$


(b) $\alpha = 1e^{-9}$, FID $= 127.38$


(c) $\alpha = 1e^{-7}$, FID $= 18.68$

(d) $\alpha = 1e^{-5}$, FID $= 34.78$


(e) $\alpha = 1e^{-3}$, FID $= 33.17$


(f) $\alpha = 1e^{-1}$, FID $= 282.11$

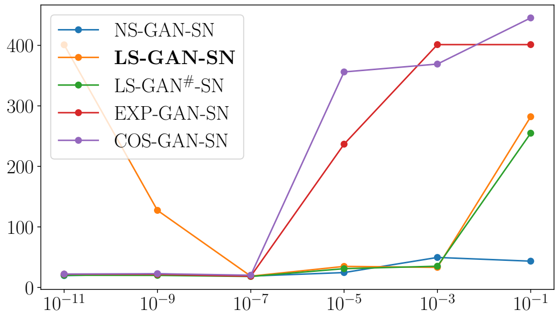
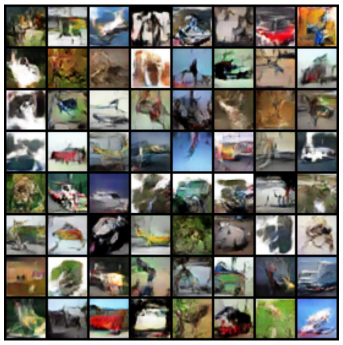
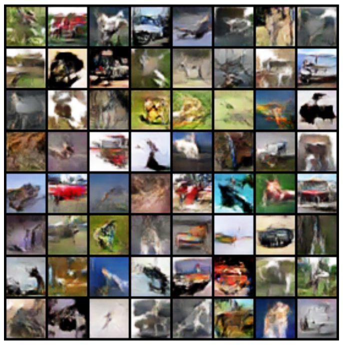


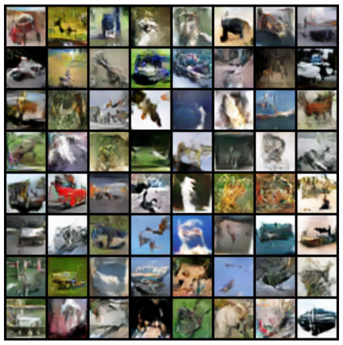
Figure 25: Samples of randomly generated images with LS-GAN-SN of varying α ($k_{SN} = 50.0$, CIFAR10). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.



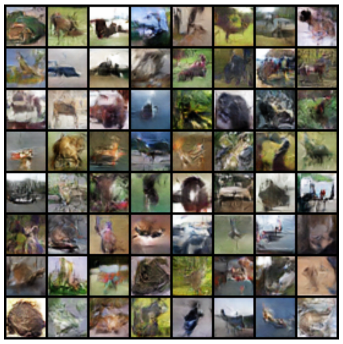
(a) $\alpha = 1e^{-11}$, FID = 20.16



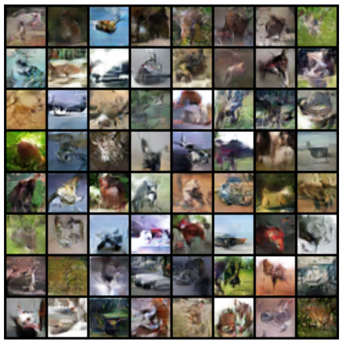
(b) $\alpha = 1e^{-9}$, FID = 19.96



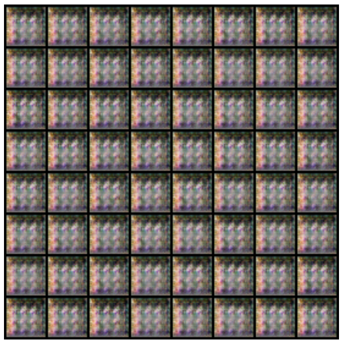
(c) $\alpha = 1e^{-7}$, FID = 18.13



(d) $\alpha = 1e^{-5}$, FID = 31.03



(e) $\alpha = 1e^{-3}$, FID = 35.06



(f) $\alpha = 1e^{-1}$, FID = 254.88

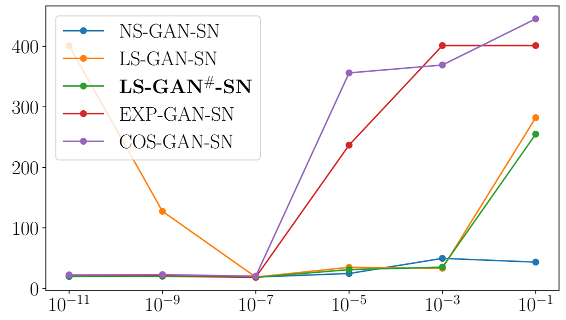
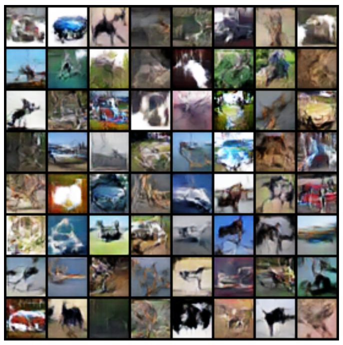
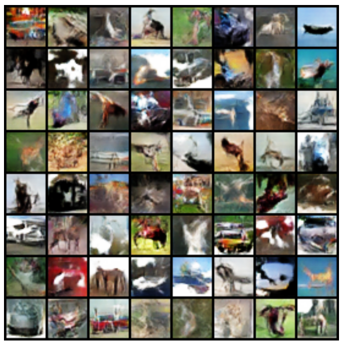
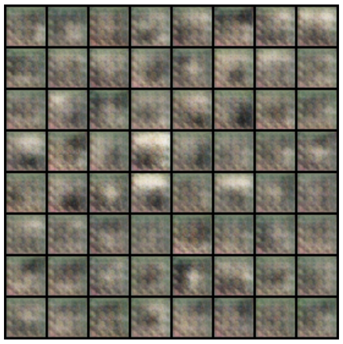


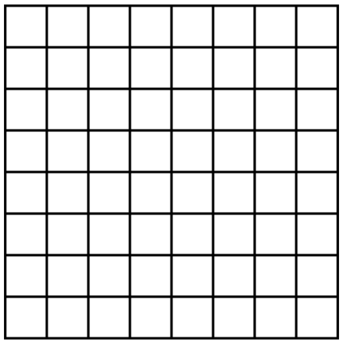
Figure 26: Samples of randomly generated images with LS-GAN$^{\#}$-SN of varying α ($k_{SN} = 50.0$, CIFAR10). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

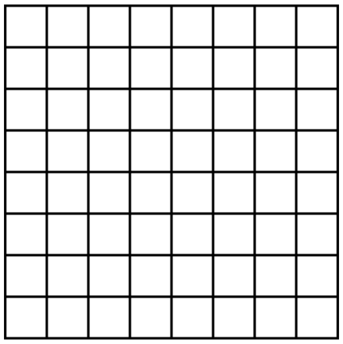
(a) $\alpha = 1e^{-11}$, FID $= 21.93$


(b) $\alpha = 1e^{-9}$, FID $= 21.70$


(c) $\alpha = 1e^{-7}$, FID $= 18.50$


(d) $\alpha = 1e^{-5}$, FID $= 236.77$


(e) $\alpha = 1e^{-3}$, FID $= 401.24$


(f) $\alpha = 1e^{-1}$, FID $= 401.24$

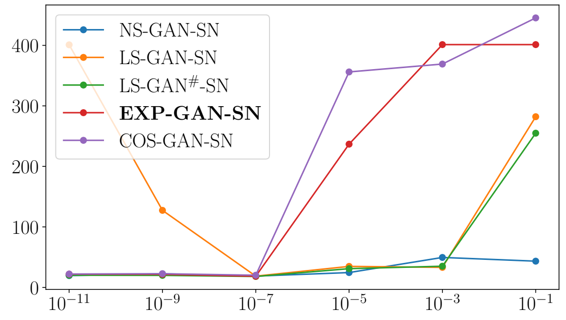
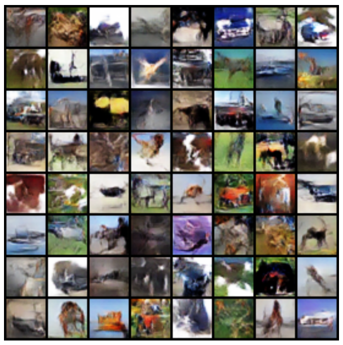
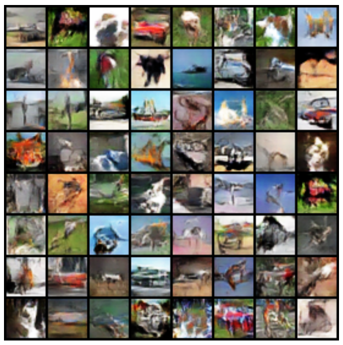
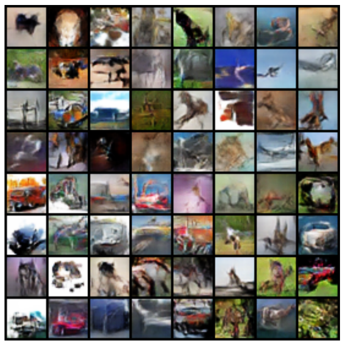
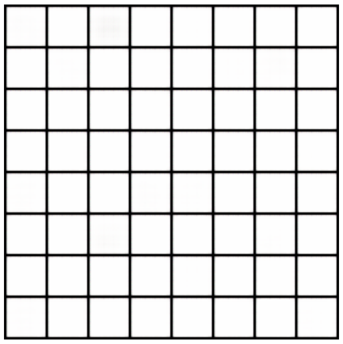


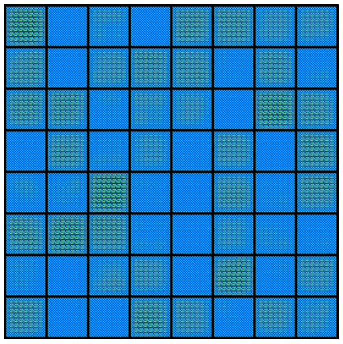
Figure 27: Samples of randomly generated images with EXP-GAN-SN of varying α ($k_{SN} = 50.0$, CIFAR10). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

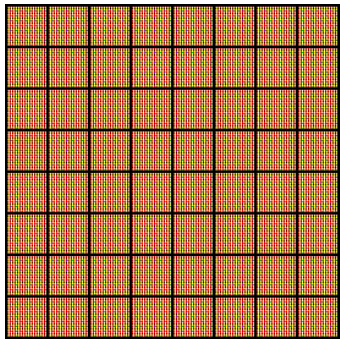

(a) $\alpha = 1e^{-11}$, FID $= 22.16$


(b) $\alpha = 1e^{-9}$, FID $= 22.69$


(c) $\alpha = 1e^{-7}$, FID $= 20.19$


(d) $\alpha = 1e^{-5}$, FID $= 356.10$


(e) $\alpha = 1e^{-3}$, FID $= 369.11$


(f) $\alpha = 1e^{-1}$, FID $= 445.37$

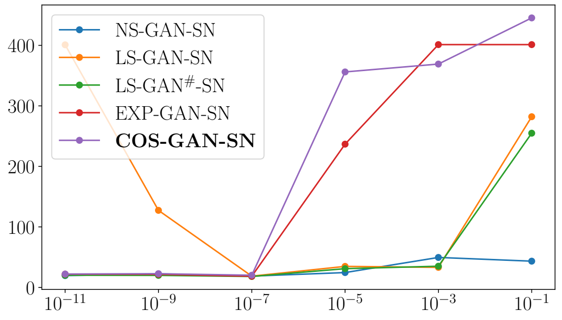


Figure 28: Samples of randomly generated images with COS-GAN-SN of varying α ($k_{SN} = 50.0$, CIFAR10). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

FID scores *vs.* α ($k_{SN} = 50.0$) of Different Loss Functions

- CelebA -

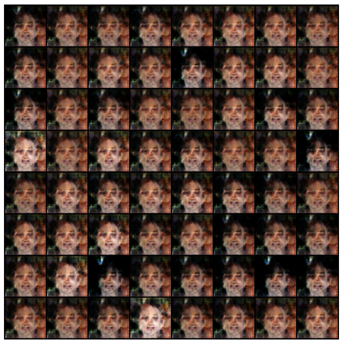
(a) $\alpha = 1e^{-11}$, FID $= 9.08$

(b) $\alpha = 1e^{-9}$, FID $= 7.05$

(c) $\alpha = 1e^{-7}$, FID $= 7.84$

(d) $\alpha = 1e^{-5}$, FID $= 18.51$

(e) $\alpha = 1e^{-3}$, FID $= 18.41$


(f) $\alpha = 1e^{-1}$, FID $= 242.64$

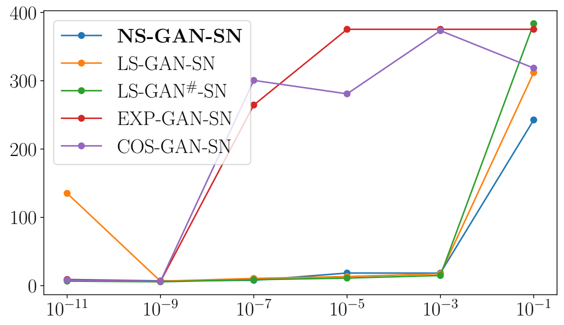
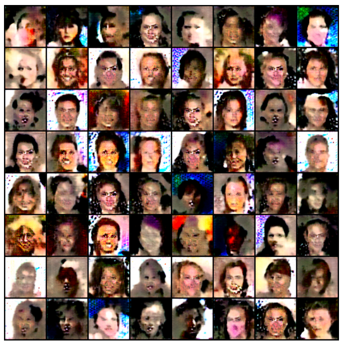
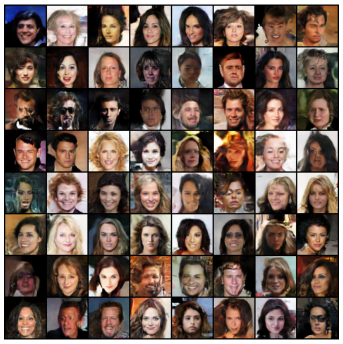


Figure 29: Samples of randomly generated images with NS-GAN-SN of varying α ($k_{SN} = 50.0$, CelebA). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

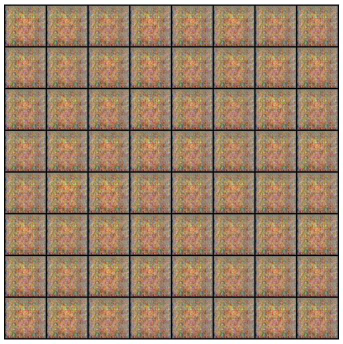

(a) $\alpha = 1e^{-11}$, FID = 135.17

(b) $\alpha = 1e^{-9}$, FID = 6.57

(c) $\alpha = 1e^{-7}$, FID = 10.67


(d) $\alpha = 1e^{-5}$, FID = 13.39

(e) $\alpha = 1e^{-3}$, FID = 17.42


(f) $\alpha = 1e^{-1}$, FID = 311.93

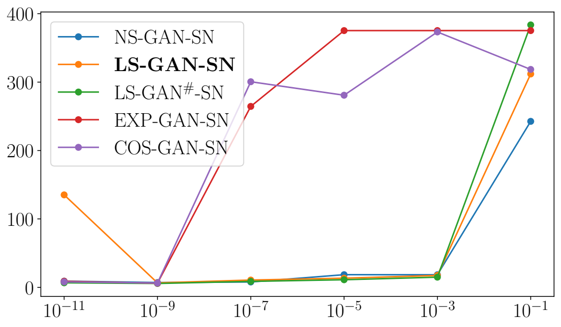
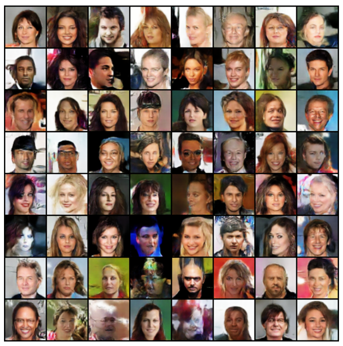
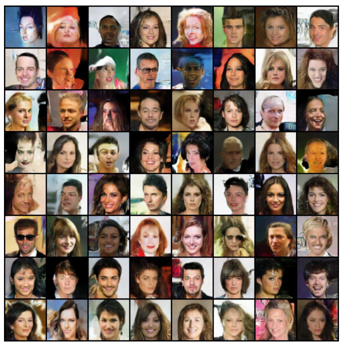


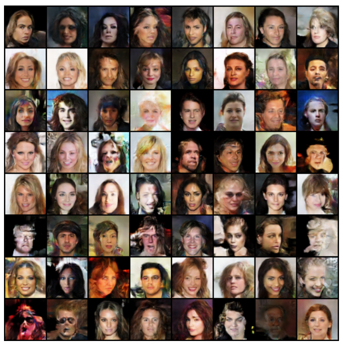
Figure 30: Samples of randomly generated images with LS-GAN-SN of varying α ($k_{SN} = 50.0$, CelebA). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

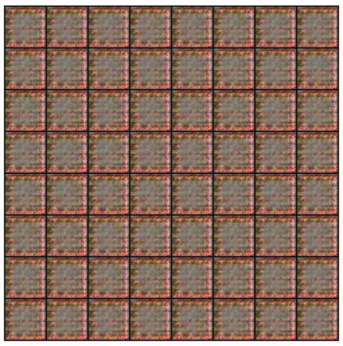

(a) $\alpha = 1e^{-11}$, FID $= 6.66$


(b) $\alpha = 1e^{-9}$, FID $= 5.68$

(c) $\alpha = 1e^{-7}$, FID $= 8.72$

(d) $\alpha = 1e^{-5}$, FID $= 11.13$


(e) $\alpha = 1e^{-3}$, FID $= 14.90$


(f) $\alpha = 1e^{-1}$, FID $= 383.61$

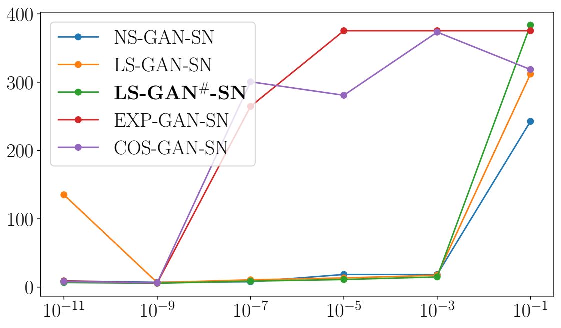
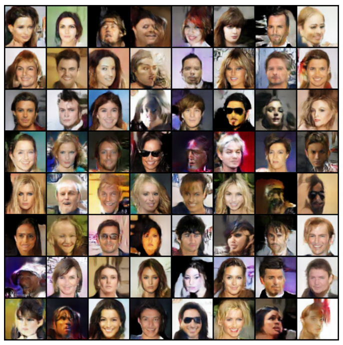
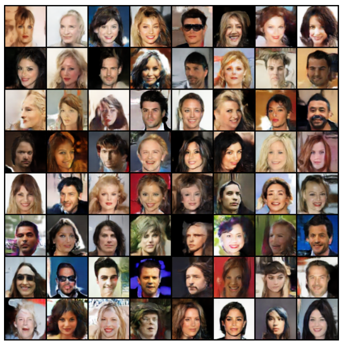
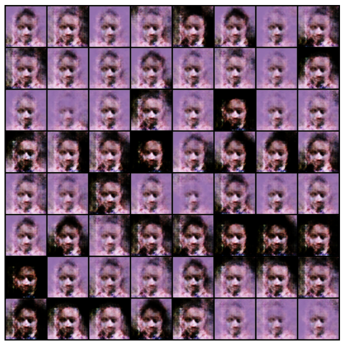
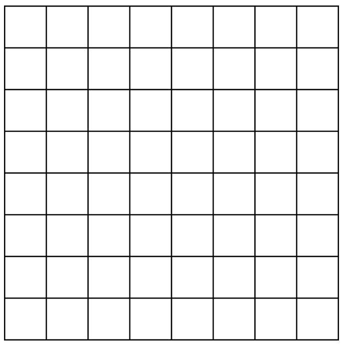


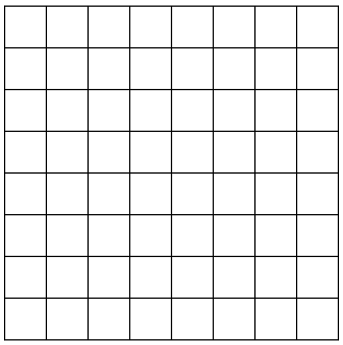
Figure 31: Samples of randomly generated images with LS-GAN$^{\#}$-SN of varying α ($k_{SN} = 50.0$, CelebA). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

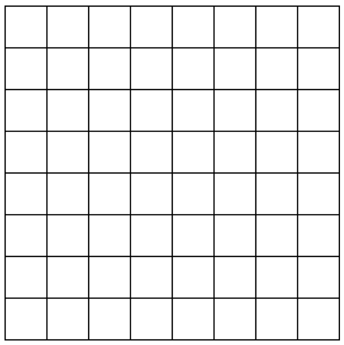

(a) $\alpha = 1e^{-11}$, FID = 8.85


(b) $\alpha = 1e^{-9}$, FID = 6.09


(c) $\alpha = 1e^{-7}$, FID = 264.49


(d) $\alpha = 1e^{-5}$, FID = 375.32


(e) $\alpha = 1e^{-3}$, FID = 375.32


(f) $\alpha = 1e^{-1}$, FID = 375.32

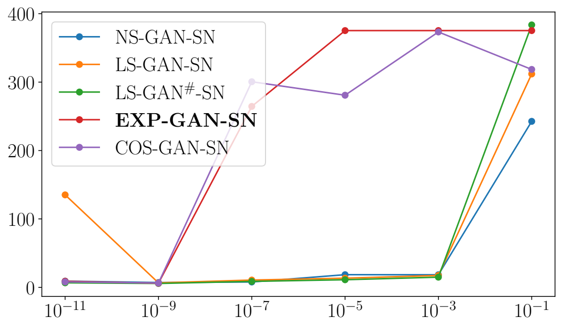
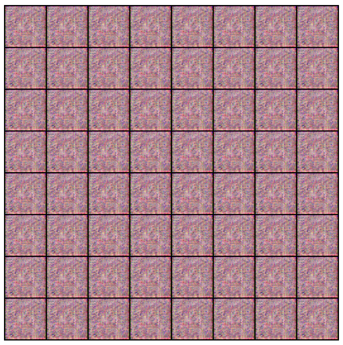
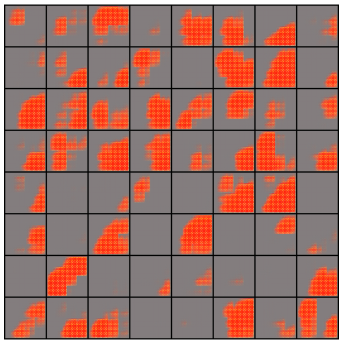


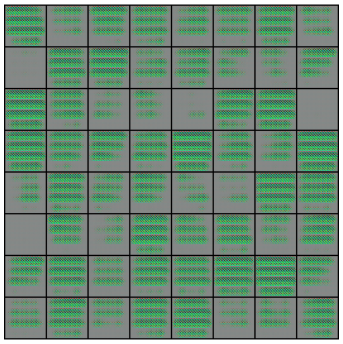
Figure 32: Samples of randomly generated images with EXP-GAN-SN of varying α ($k_{SN} = 50.0$, CelebA). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

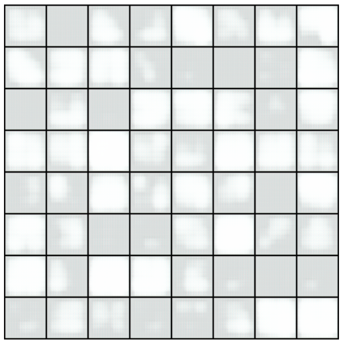
(a) $\alpha = 1e^{-11}$, FID = 8.00

(b) $\alpha = 1e^{-9}$, FID = 6.31


(c) $\alpha = 1e^{-7}$, FID = 300.55


(d) $\alpha = 1e^{-5}$, FID = 280.84


(e) $\alpha = 1e^{-3}$, FID = 373.31


(f) $\alpha = 1e^{-1}$, FID = 318.53

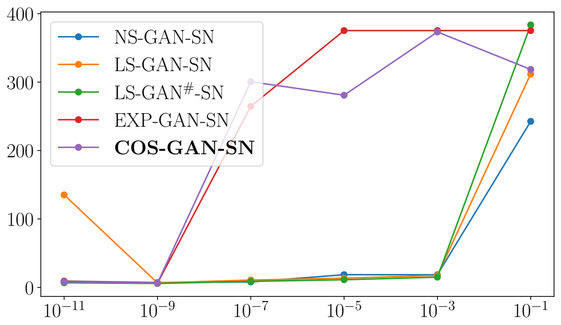
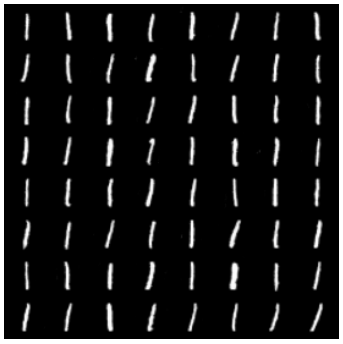


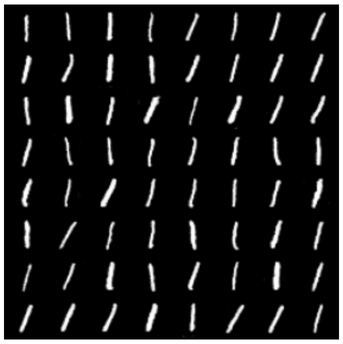
Figure 33: Samples of randomly generated images with COS-GAN-SN of varying α ($k_{SN} = 50.0$, CelebA). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

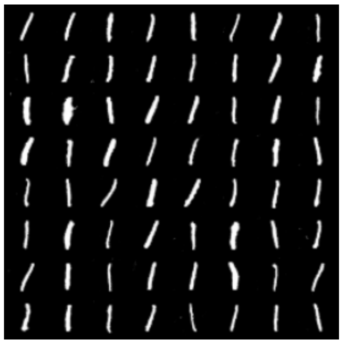
FID scores *vs.* α ($k_{SN} = 1.0$) of Different Loss Functions

- MNIST -

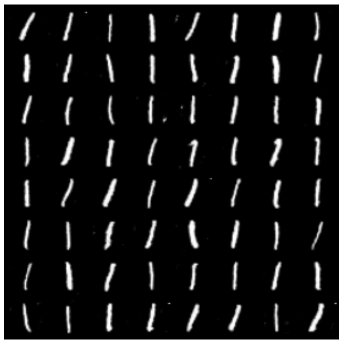
(a) $\alpha = 1e^1$, FID $= 6.55$


(b) $\alpha = 1e^3$, FID $= 148.97$


(c) $\alpha = 1e^5$, FID $= 134.44$


(d) $\alpha = 1e^7$, FID $= 133.82$

(e) $\alpha = 1e^9$, FID $= 130.21$


(f) $\alpha = 1e^{11}$, FID $= 131.87$

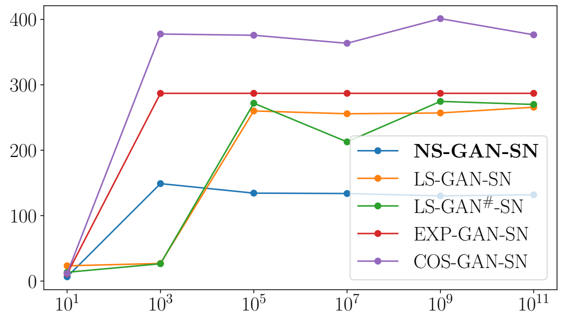
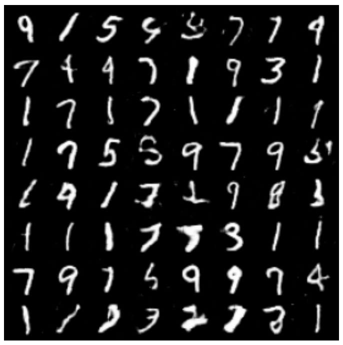
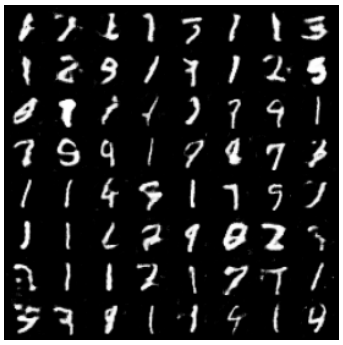
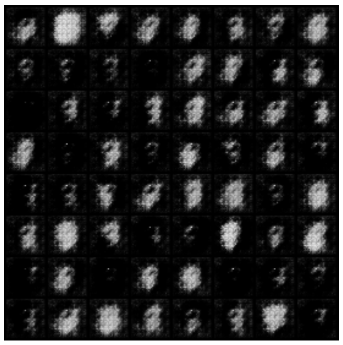
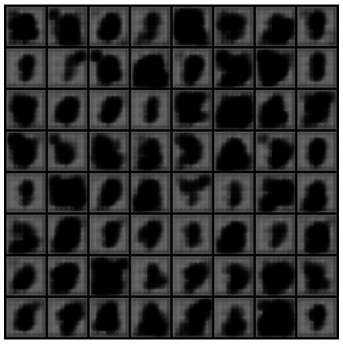


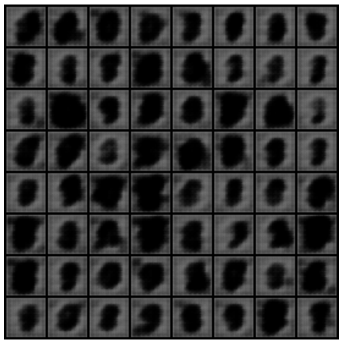
Figure 34: Samples of randomly generated images with NS-GAN-SN of varying α ($k_{SN} = 1.0$, MNIST). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

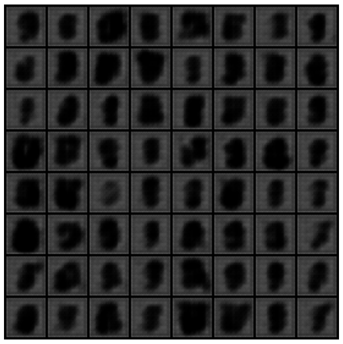

(a) $\alpha = 1e^1$, FID = 23.37


(b) $\alpha = 1e^3$, FID = 26.96


(c) $\alpha = 1e^5$, FID = 260.05


(d) $\alpha = 1e^7$, FID = 255.73


(e) $\alpha = 1e^9$, FID = 256.96


(f) $\alpha = 1e^{11}$, FID = 265.76

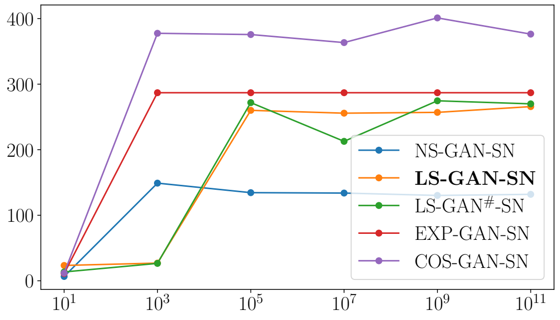
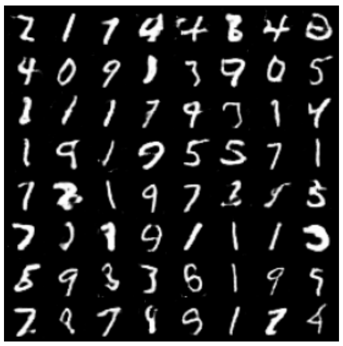
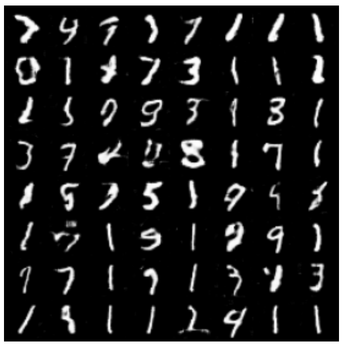
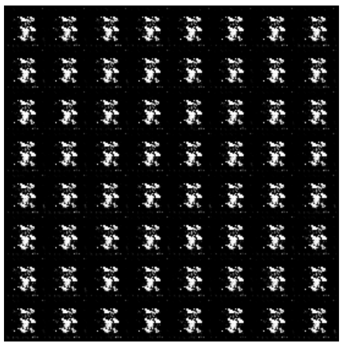
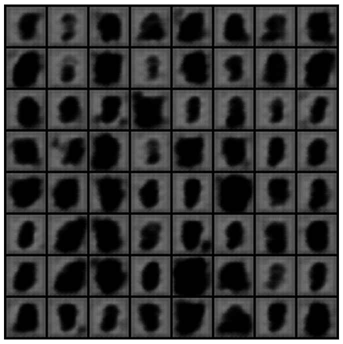


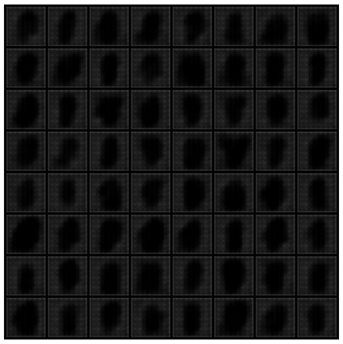
Figure 35: Samples of randomly generated images with LS-GAN-SN of varying α ($k_{SN} = 1.0$, MNIST). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

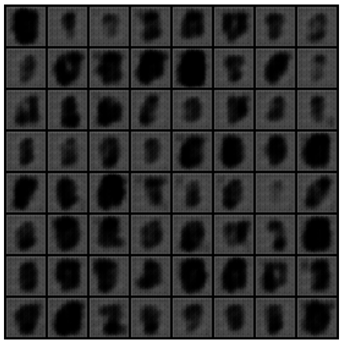

(a) $\alpha = 1e^1$, FID = 13.43


(b) $\alpha = 1e^3$, FID = 26.51


(c) $\alpha = 1e^5$, FID = 271.85


(d) $\alpha = 1e^7$, FID = 212.74


(e) $\alpha = 1e^9$, FID = 274.63


(f) $\alpha = 1e^{11}$, FID = 269.96

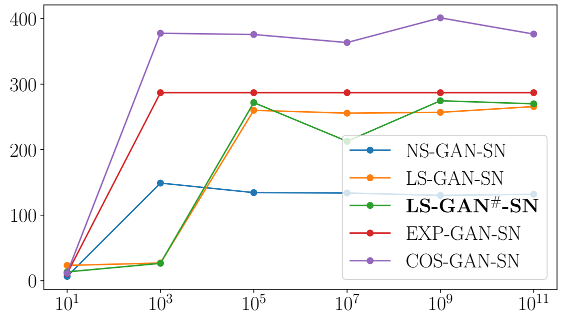
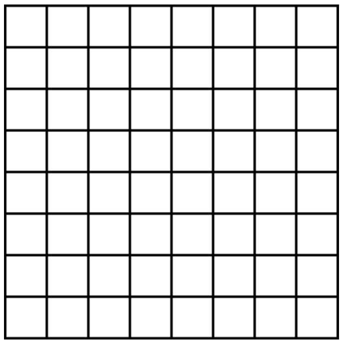
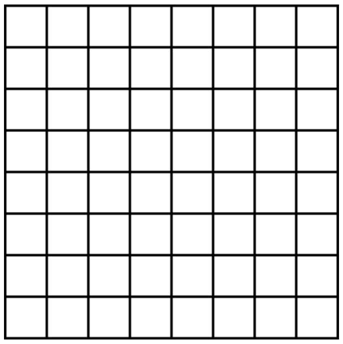
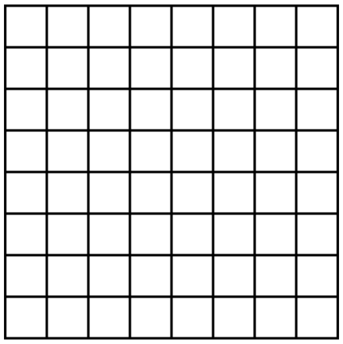


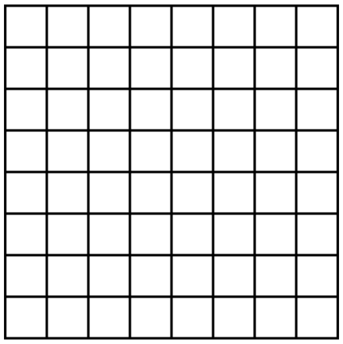
Figure 36: Samples of randomly generated images with LS-GAN$^{\#}$-SN of varying α ($k_{SN} = 1.0$, MNIST). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

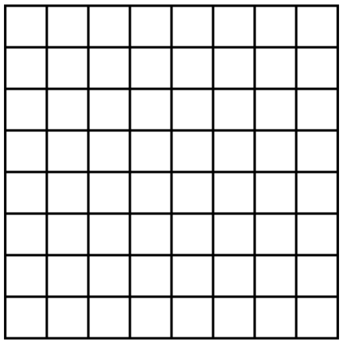
(a) $\alpha = 1e^1$, FID = 11.02


(b) $\alpha = 1e^3$, FID = 286.96


(c) $\alpha = 1e^5$, FID = 286.96


(d) $\alpha = 1e^7$, FID = 286.96


(e) $\alpha = 1e^9$, FID = 286.96


(f) $\alpha = 1e^{11}$, FID = 286.96

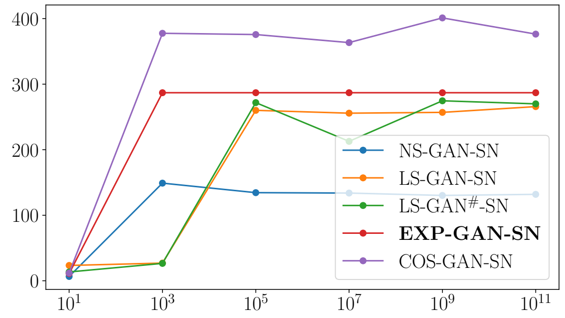
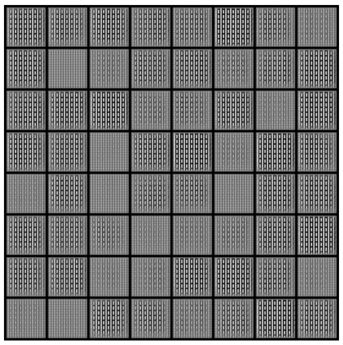
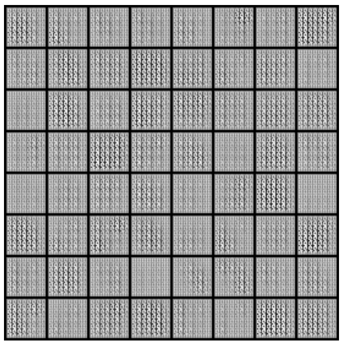
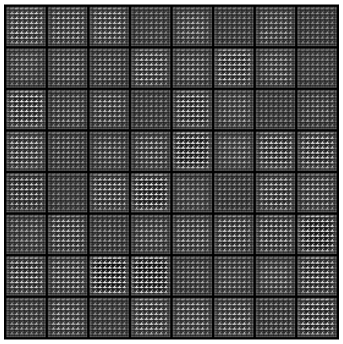


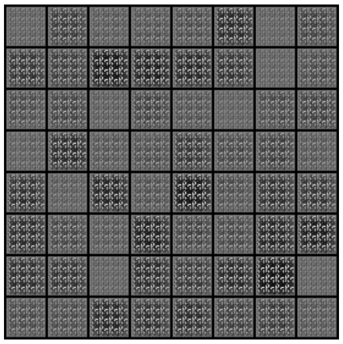
Figure 37: Samples of randomly generated images with EXP-GAN-SN of varying α ($k_{SN} = 1.0$, MNIST). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

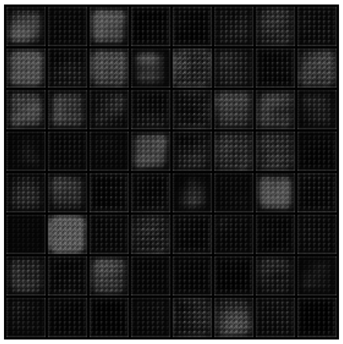
(a) $\alpha = 1e^1$, FID = 11.79


(b) $\alpha = 1e^3$, FID = 377.62


(c) $\alpha = 1e^5$, FID = 375.72


(d) $\alpha = 1e^7$, FID = 363.45


(e) $\alpha = 1e^9$, FID = 401.12


(f) $\alpha = 1e^{11}$, FID = 376.39

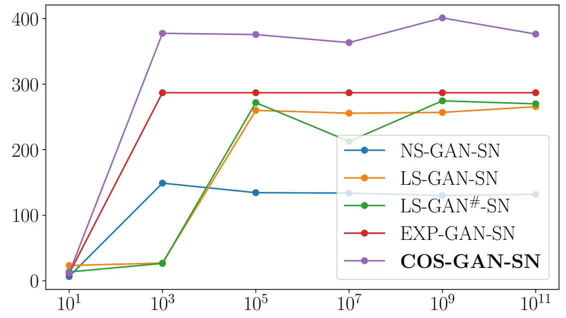
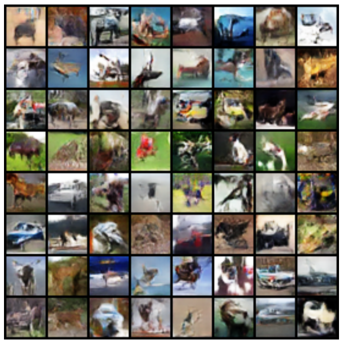


Figure 38: Samples of randomly generated images with COS-GAN-SN of varying α ($k_{SN} = 1.0$, MNIST). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

FID scores *vs.* α ($k_{SN} = 1.0$) of Different Loss Functions

- CIFAR10 -

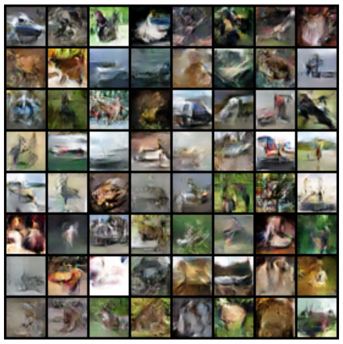

(a) $\alpha = 1e^1$, FID $= 17.63$

(b) $\alpha = 1e^3$, FID $= 47.31$

(c) $\alpha = 1e^5$, FID $= 46.85$

(d) $\alpha = 1e^7$, FID $= 45.44$

(e) $\alpha = 1e^9$, FID $= 45.67$


(f) $\alpha = 1e^{11}$, FID $= 39.90$

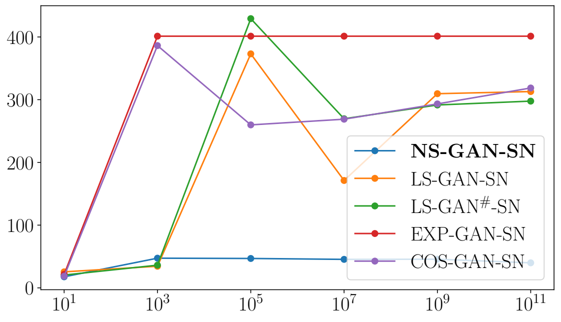
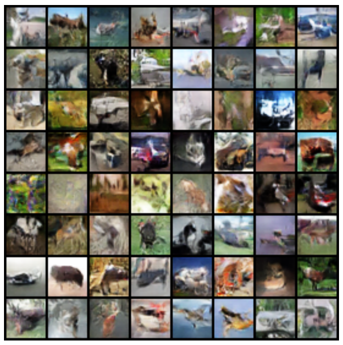
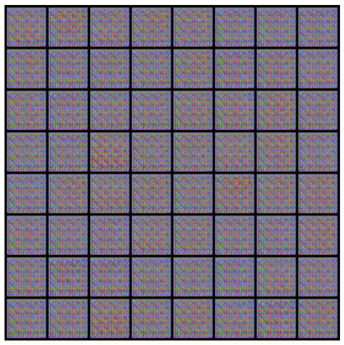
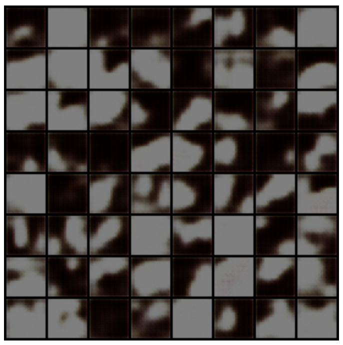


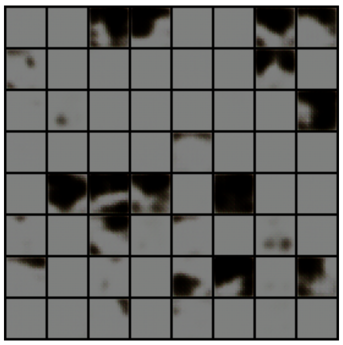
Figure 39: Samples of randomly generated images with NS-GAN-SN of varying α ($k_{SN} = 1.0$, CIFAR10). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

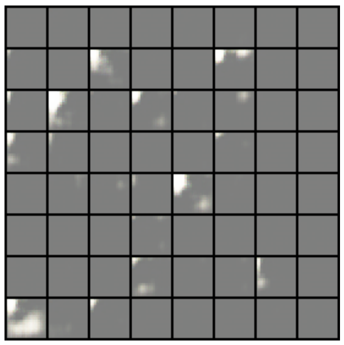

(a) $\alpha = 1e^1$, FID $= 25.55$

(b) $\alpha = 1e^3$, FID $= 34.44$


(c) $\alpha = 1e^5$, FID $= 373.07$


(d) $\alpha = 1e^7$, FID $= 171.18$


(e) $\alpha = 1e^9$, FID $= 309.55$


(f) $\alpha = 1e^{11}$, FID $= 312.96$

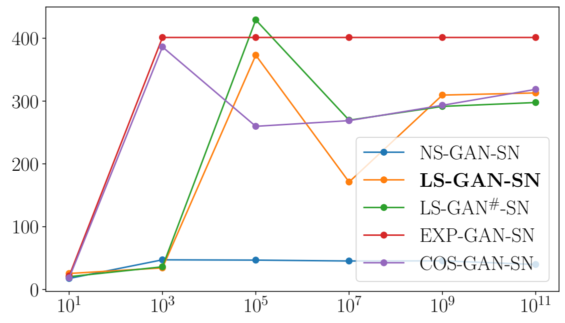
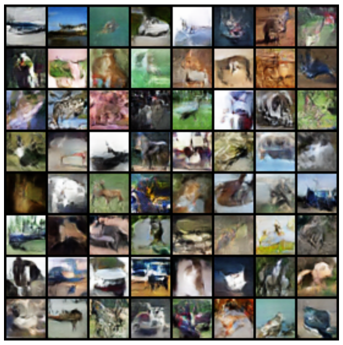
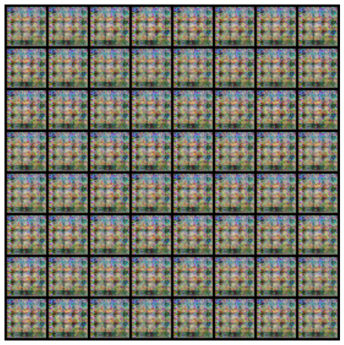
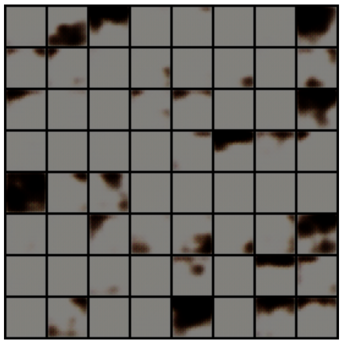


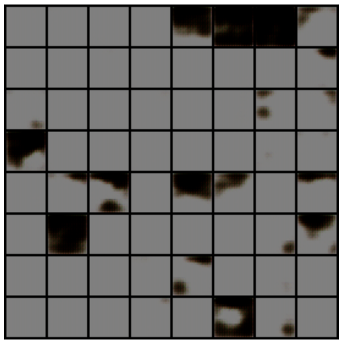
Figure 40: Samples of randomly generated images with LS-GAN-SN of varying α ($k_{SN} = 1.0$, CIFAR10). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

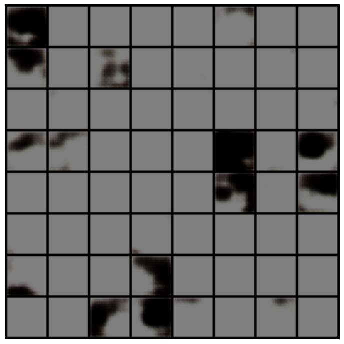

(a) $\alpha = 1e^{1}$, FID $= 20.45$

(b) $\alpha = 1e^{3}$, FID $= 36.18$


(c) $\alpha = 1e^{5}$, FID $= 429.21$


(d) $\alpha = 1e^{7}$, FID $= 269.63$


(e) $\alpha = 1e^{9}$, FID $= 291.55$


(f) $\alpha = 1e^{11}$, FID $= 297.71$

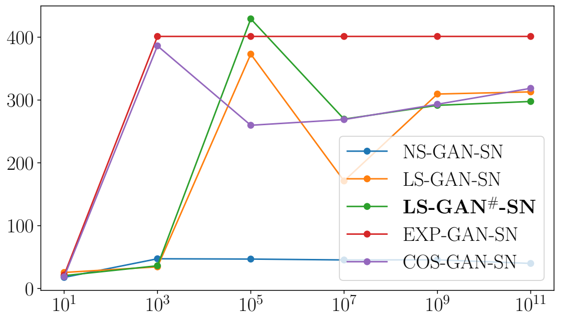
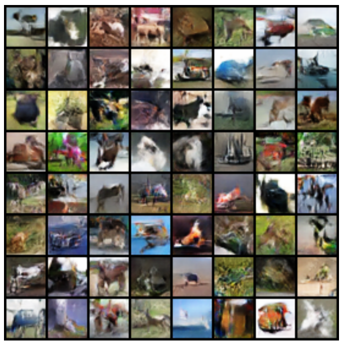
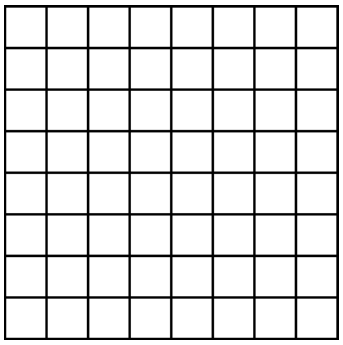
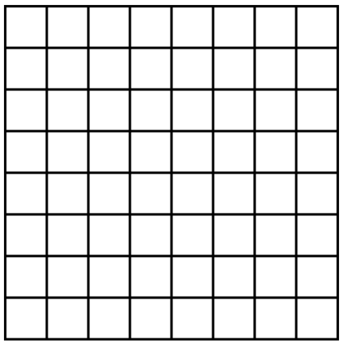
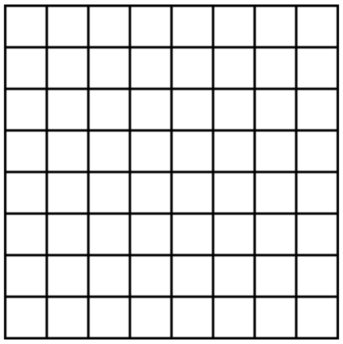


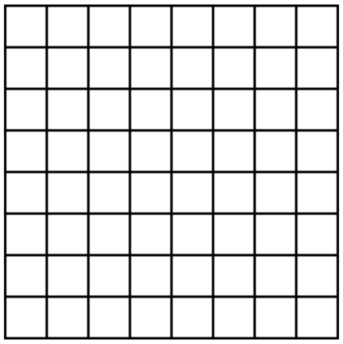
Figure 41: Samples of randomly generated images with LS-GAN$^{\#}$-SN of varying α ($k_{SN} = 1.0$, CIFAR10). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

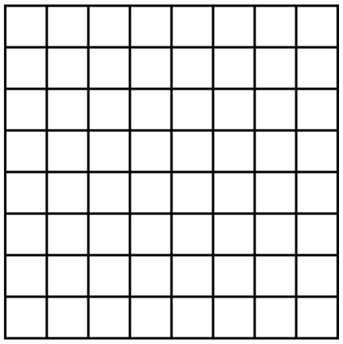

(a) $\alpha = 1e^1$, FID = 21.56


(b) $\alpha = 1e^3$, FID = 401.24


(c) $\alpha = 1e^5$, FID = 401.24


(d) $\alpha = 1e^7$, FID = 401.24


(e) $\alpha = 1e^9$, FID = 401.24


(f) $\alpha = 1e^{11}$, FID = 401.24

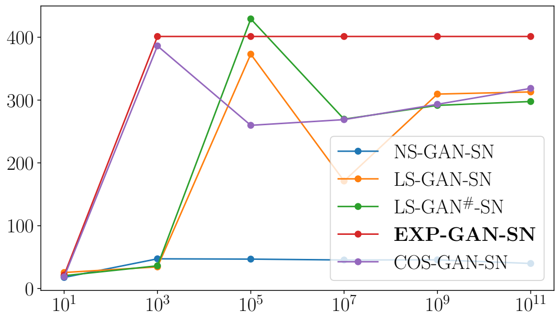
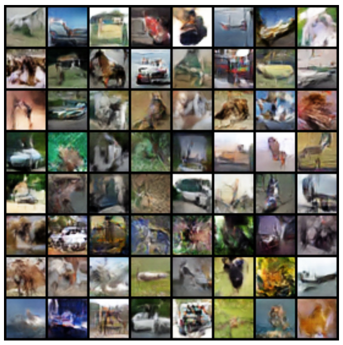
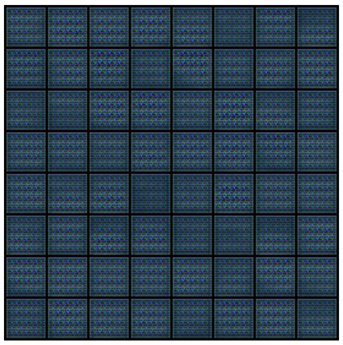
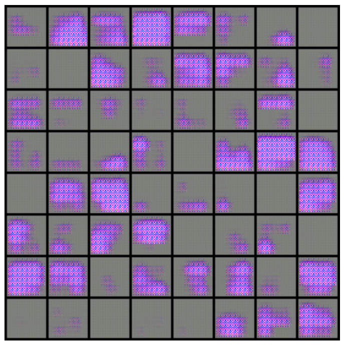
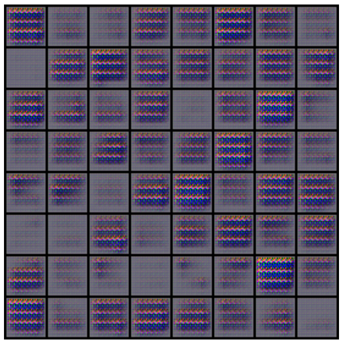


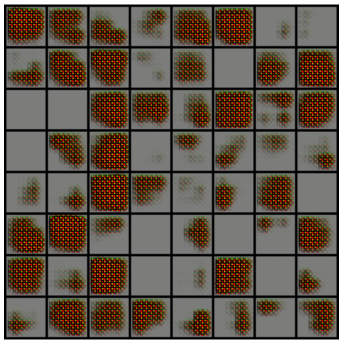
Figure 42: Samples of randomly generated images with EXP-GAN-SN of varying α ($k_{SN} = 1.0$, CIFAR10). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

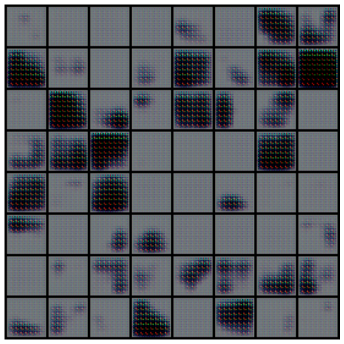

(a) $\alpha = 1e^1$, FID $= 18.59$


(b) $\alpha = 1e^3$, FID $= 386.24$


(c) $\alpha = 1e^5$, FID $= 259.83$


(d) $\alpha = 1e^7$, FID $= 268.89$


(e) $\alpha = 1e^9$, FID $= 293.29$


(f) $\alpha = 1e^{11}$, FID $= 318.65$

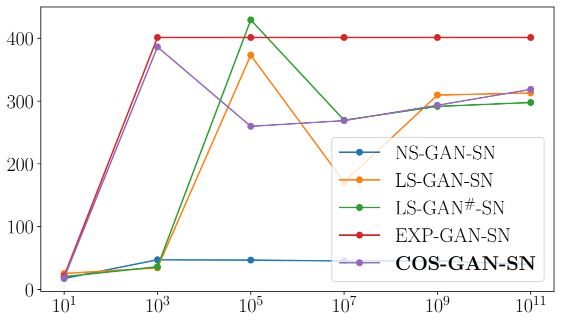


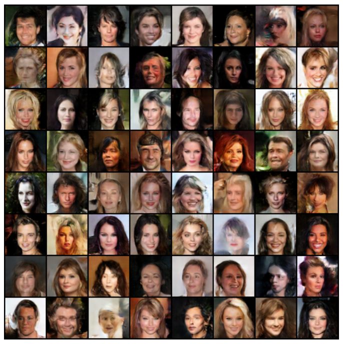
Figure 43: Samples of randomly generated images with COS-GAN-SN of varying α ($k_{SN} = 1.0$, CIFAR10). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

FID scores *vs.* α ($k_{SN} = 1.0$) of Different Loss Functions

- CelebA -

(a) $\alpha = 1e^1$, FID $= 5.88$

(b) $\alpha = 1e^3$, FID $= 16.14$


(c) $\alpha = 1e^5$, FID $= 17.75$

(d) $\alpha = 1e^7$, FID $= 17.67$

(e) $\alpha = 1e^9$, FID $= 16.87$

(f) $\alpha = 1e^{11}$, FID $= 18.81$

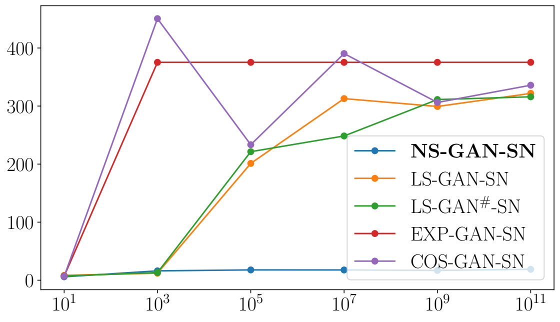
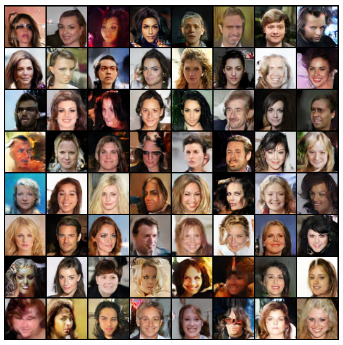
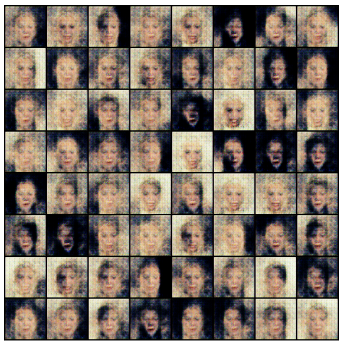
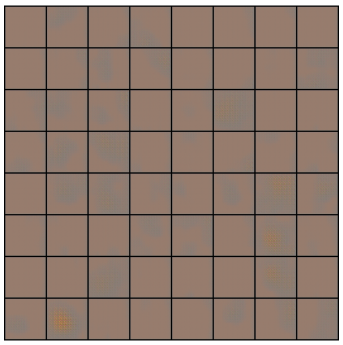


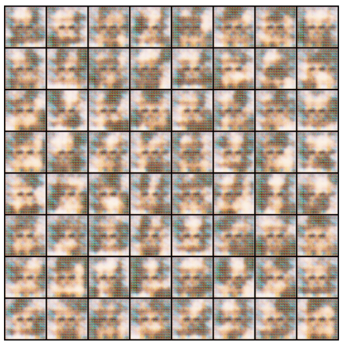
Figure 44: Samples of randomly generated images with NS-GAN-SN of varying α ($k_{SN} = 1.0$, CelebA). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

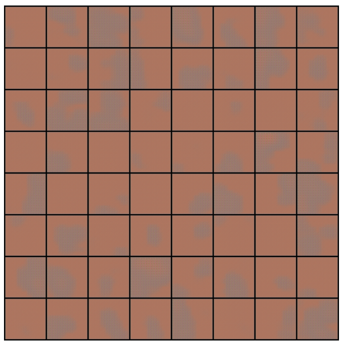

(a) $\alpha = 1e^1$, FID = 8.41

(b) $\alpha = 1e^3$, FID = 12.09


(c) $\alpha = 1e^5$, FID = 201.22


(d) $\alpha = 1e^7$, FID = 312.83


(e) $\alpha = 1e^9$, FID = 299.30


(f) $\alpha = 1e^{11}$, FID = 321.84

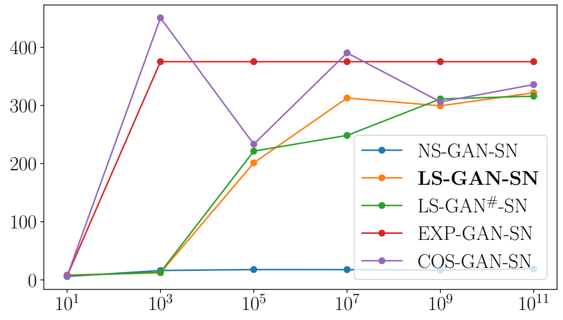
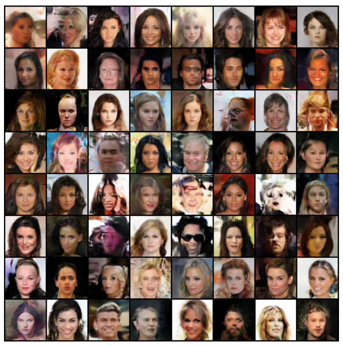
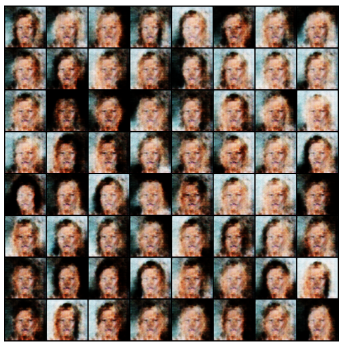


Figure 45: Samples of randomly generated images with LS-GAN-SN of varying α ($k_{SN} = 1.0$, CelebA). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

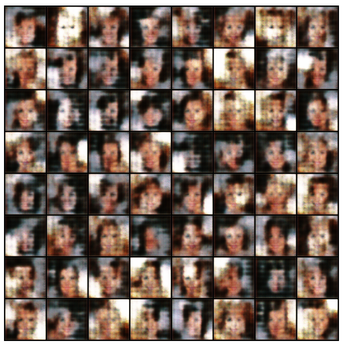
(a) $\alpha = 1e^{1}$, FID $= 7.21$



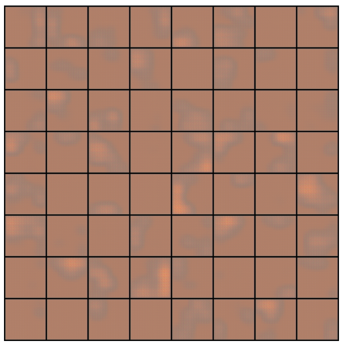
(b) $\alpha = 1e^{3}$, FID $= 13.13$



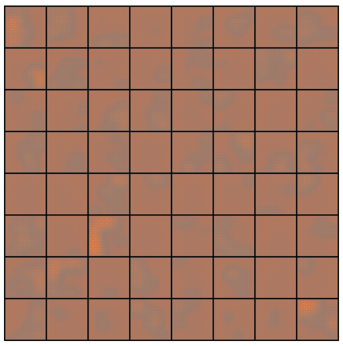
(c) $\alpha = 1e^{5}$, FID $= 221.41$



(d) $\alpha = 1e^{7}$, FID $= 248.48$



(e) $\alpha = 1e^{9}$, FID $= 311.21$



(f) $\alpha = 1e^{11}$, FID $= 315.94$

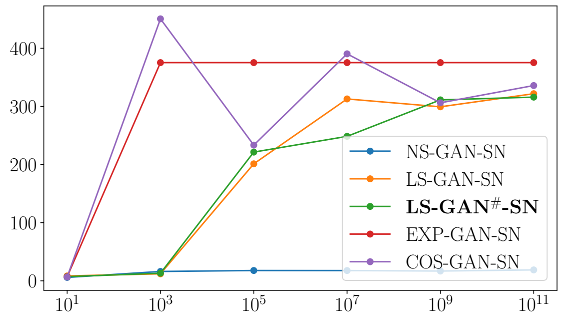
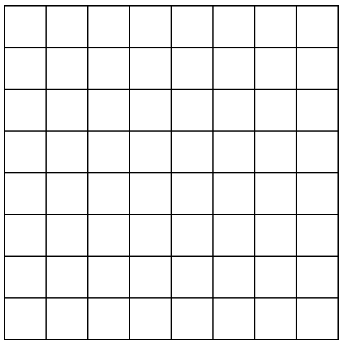
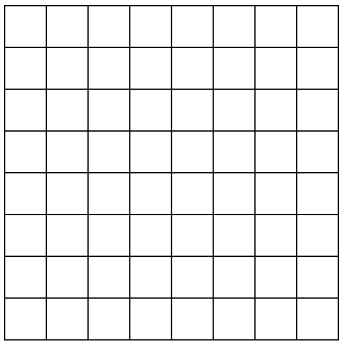
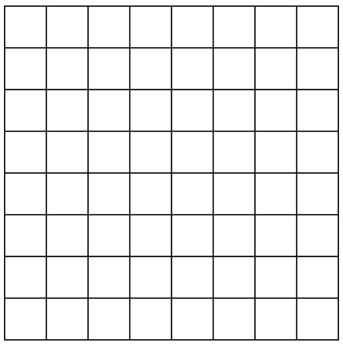


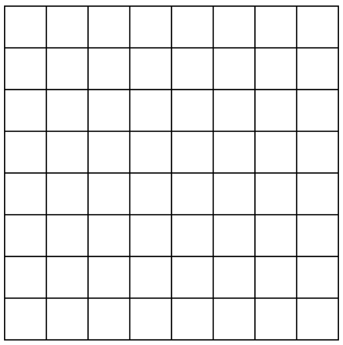
Figure 46: Samples of randomly generated images with LS-GAN$^{\#}$-SN of varying α ($k_{SN} = 1.0$, CelebA). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

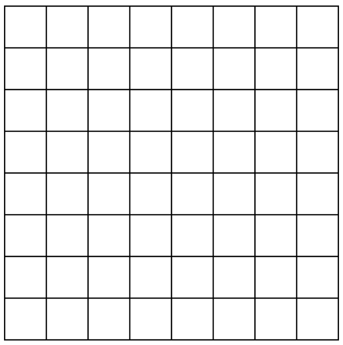
(a) $\alpha = 1e^1$, FID $= 6.91$


(b) $\alpha = 1e^3$, FID $= 375.32$


(c) $\alpha = 1e^5$, FID $= 375.32$


(d) $\alpha = 1e^7$, FID $= 375.32$


(e) $\alpha = 1e^9$, FID $= 375.32$


(f) $\alpha = 1e^{11}$, FID $= 375.32$

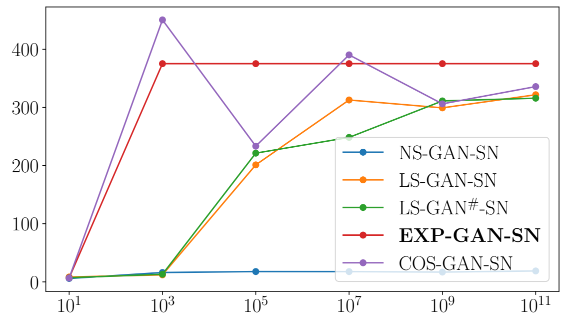
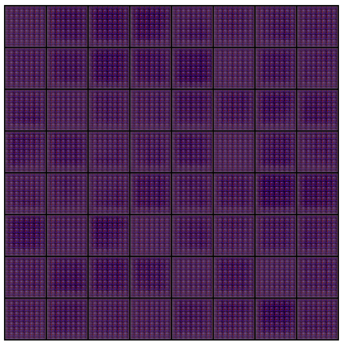
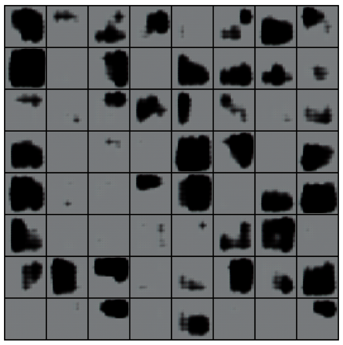
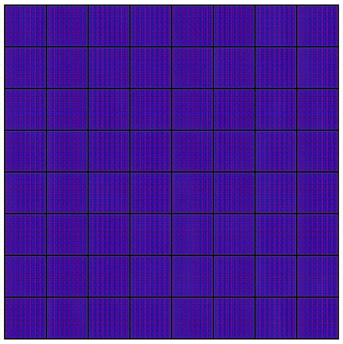


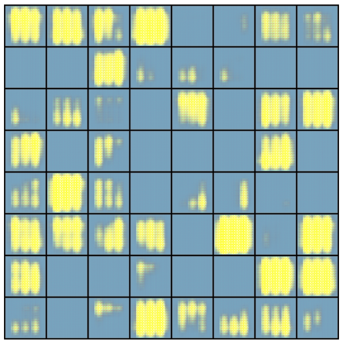
Figure 47: Samples of randomly generated images with EXP-GAN-SN of varying α ($k_{SN} = 1.0$, CelebA). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

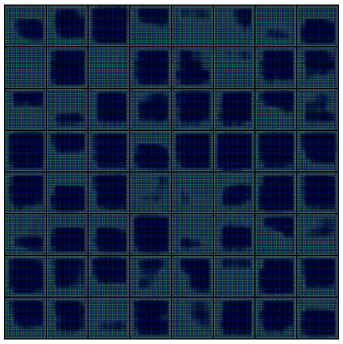
(a) $\alpha = 1e^1$, FID $= 6.62$


(b) $\alpha = 1e^3$, FID $= 450.57$


(c) $\alpha = 1e^5$, FID $= 233.42$


(d) $\alpha = 1e^7$, FID $= 390.40$


(e) $\alpha = 1e^9$, FID $= 306.17$


(f) $\alpha = 1e^{11}$, FID $= 335.87$

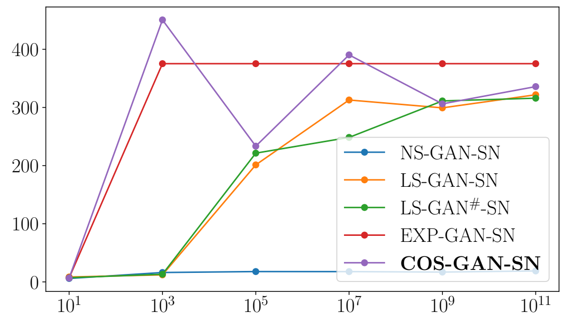
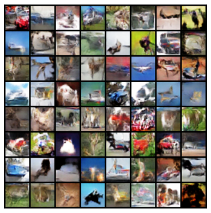
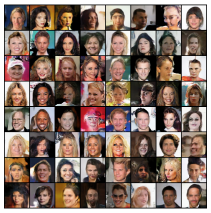


Figure 48: Samples of randomly generated images with COS-GAN-SN of varying α ($k_{SN} = 1.0$, CelebA). For the line plot, x-axis shows α (in log scale) and y-axis shows the FID scores.

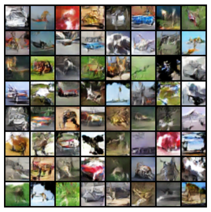
FID scores of WGAN-SN and some extremely degenerate loss functions ($\alpha = 1e^{-25}$) on different datasets

(a) WGAN, FID = 3.71


(b) WGAN, FID = 23.36

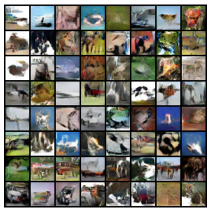

(c) WGAN, FID = 7.82

(d) NSGAN, FID = 3.74


(e) NSGAN, FID = 21.92

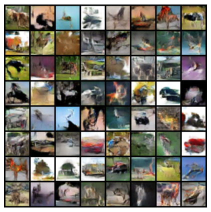
(f) NSGAN, FID = 8.10

(g) LSGAN$^{\#}$, FID = 3.81


(h) LSGAN$^{\#}$, FID = 21.47

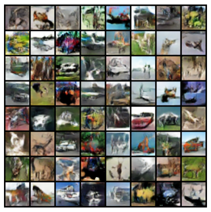
(i) LSGAN$^{\#}$, FID = 8.51

(j) COSGAN, FID = 3.96


(k) COSGAN, FID = 23.65

(l) COSGAN, FID = 8.30

(m) EXPGAN, FID = 3.86


(n) EXPGAN, FID = 21.91

(o) EXPGAN, FID = 8.22

Figure 49: Samples of randomly generated images with WGAN-SN and some extremely degenerate loss functions ($\alpha = 1e^{-25}$) on different datasets. We use $k_{SN} = 50$ for all our experiments.