GraN-GAN: Piecewise Gradient Normalization for Generative Adversarial Networks

Supplementary Material

Vineeth S. Bhaskara*[1] Tristan Aumentado-Armstrong*[1,2,3] Allan Jepson[1] Alex Levinshtein[1]
[1]Samsung AI Centre Toronto [2]University of Toronto [3]Vector Institute for AI
{s.bhaskara,allan.jepson,alex.lev}@samsung.com, tristan.a@partner.samsung.com

Appendix A: Training Details

We use Mimicry [12] with PyTorch [14] on a single NVIDIA V100 GPU for training our models. The generator G and discriminator (or critic) D architectures are identical across methods for a given dataset except for models with spectral normalization that replace convolutional and linear layers with their normalized variants. The number of learnable parameters are identical across methods for a fixed dataset size. Number of parameters for (G, D) are $\approx (4.3\text{M}, 1\text{M})$ for 32^2, $(4.9\text{M}, 10\text{M})$ for 48^2, and $(32\text{M}, 29\text{M})$ for 128^2 image sizes, respectively. G and D are both residual networks with ReLU activation functions, and G employs batch normalization [6] while D does not. We train our models on a single NVIDIA V100 GPU with the Adam [11] optimizer at a learning rate (LR) of 2×10^{-4}, $\beta_1 = 0.0$, $\beta_2 = 0.9$ and a batch size of 64 for 100K iterations. The number of discriminator updates per generator update n_{dis} is set to 5 for CIFAR-10/CIFAR-100/STL-10 and 2 for LSUN bedrooms/CelebA. All models (GraN or baseline) use a linear LR decay policy except models on CelebA that use the same learning rate throughout, following Mimicry [12]. However, GraNC-GAN on CelebA required a slight alteration: setting LRs for G and D to be 5×10^{-5} and 1×10^{-4}, respectively, and using linear LR decay.

Empirically we find it necessary to have a smaller piecewise Lipschitz constant $\mathcal{K}$ when training GANs on larger image resolutions with gradient normalization. We suspect that a smooth discriminator or critic with smaller gradient norms is essential for stable GAN training on larger image resolutions. We choose $\mathcal{K} = 1/\tau = 0.0909$ for our models on LSUN bedrooms/CelebA (except for $\mathcal{K} = 1/\tau = 0.2$ with GraNC-GAN on CelebA) and $\mathcal{K} = 1/\tau = 0.83$ for our models on CIFAR-10/CIFAR-100/STL-10.

WGAN-GP uses the Wasserstein distance based loss objectives for D in Eq. (7) and G in Eq. (8). SNGAN uses hinge loss for D in Eq. (10) and Eq. (8) for G.

For NSGAN-GP†, we adjust the gradient penalty loss to constrain the Lipschitz constant to $\mathcal{K}$ (instead of 1). For NSGAN-SN†, we scale the output of the network before the sigmoid by $\mathcal{K}$ to obtain an effective $\mathcal{K}$-Lipschitz constraint using SN. We also retrain the baselines WGAN-GP and SNGAN with similar modifications so that the Lipschitz constraint is identical to the piecewise Lipschitz constraint for our methods and call them WGAN-GP† and SNGAN†, respectively.

It is also worth highlighting that our method backpropagates through the GraN normalization term as well and does not simply treat it as a constant.

Evaluation We quantitatively evaluate the methods by Inception Score (IS) [15], FID [5], and KID [1] with 50K synthetic images randomly sampled from G and 50K real images from the dataset. We report the mean scores computed across 3 randomly sampled sets of 50K images for a given G. We note that across all methods and datasets, the standard deviations across 3 evaluation samplings for IS, FID, and KID are less than 0.05, 0.085, and 0.0004, respectively, and we therefore do not include them in our tables. IS is not used for LSUN and CelebA, as these comprise a single class, for which IS performs poorly [12].

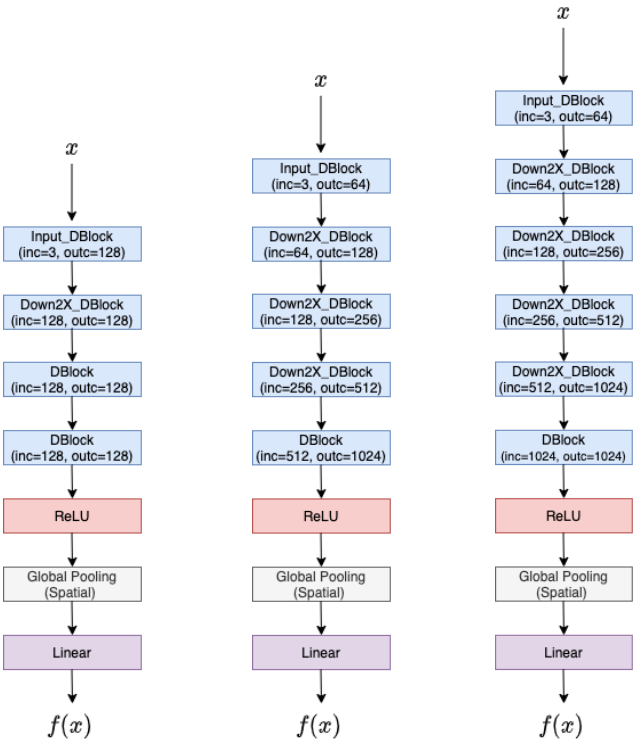
Appendix B: Model Architectures

Figure 1 presents the discriminator model architectures for inputs of dimensions 32^2, 48^2 and 128^2, respectively. Figure 2 presents the generator model architectures for outputs of dimensions 32^2, 48^2 and 128^2, respectively.

Note that for GraN-models, the output of the networks $f(x)$ is normalized to $g(x)$ as described in Equation (19) of the main paper and does not contain any additional learnable parameters.

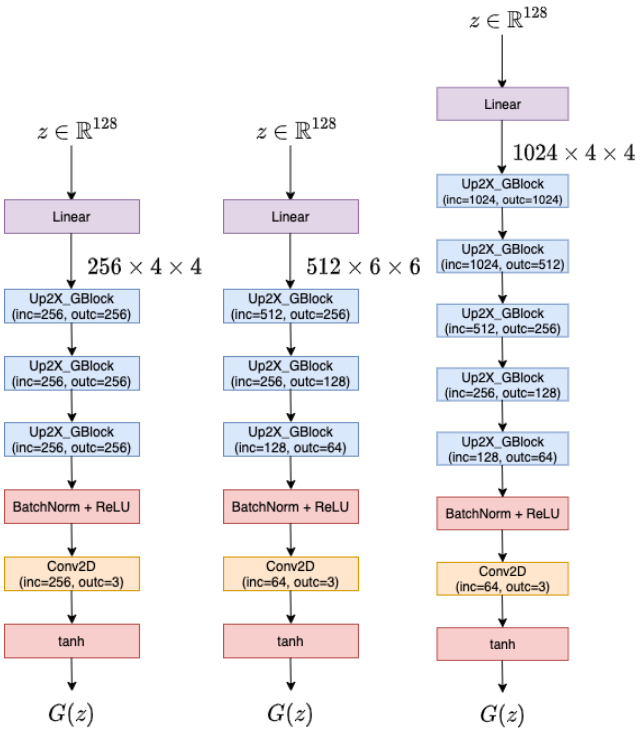
To modify the Lipschitz constant (LC) of baselines in-

*Equal contribution.



(a) $x \in \mathbb{R}^{32^2}$ (b) $x \in \mathbb{R}^{48^2}$ (c) $x \in \mathbb{R}^{128^2}$

Figure 1: Discriminator architectures for a) 32×32, b) 48×48 and c) 128×128 image sizes, respectively. The architectures for Input_DBlock, DBlock and Down2X_DBlock are described in Figure 3. All models use Global Spatial Average Pooling except SNGAN that uses Global Spatial Sum Pooling before the last Linear layer. For SNGAN only, the `Linear` and convolution `Conv2D` layers are the spectral normalized versions with 1 power iteration.



(a) $G(z) \in \mathbb{R}^{32^2}$ (b) $G(z) \in \mathbb{R}^{48^2}$ (c) $G(z) \in \mathbb{R}^{128^2}$

Figure 2: Generator architectures for a) 32×32, b) 48×48 and c) 128×128 image sizes, respectively. The architecture for Up2X_GBlock is described in Figure 3. Generator architectures are identical across all models for a given dataset resolution.

volving spectral normalized linear and convolutional layers (NSGAN-SN† and SNGAN†), we scale the output $f(x)$ by $\mathcal{K}$, i.e., $f(x) \rightarrow \mathcal{K}f(x)$, where $\mathcal{K}$ scales LC relative to the LC of the baseline model since $|f|_{\text{Lip}} \leq 1$, implies, $|\mathcal{K}f|_{\text{Lip}} \leq \mathcal{K}$.

For models WGAN-GP† and NSGAN-GP†, we instead only change the gradient penalty loss term in the objective for D to

$$\mathcal{L}_{\text{GP}} = \lambda \left(\|\nabla_x f(x)\| - \mathcal{K} \right)^2,$$

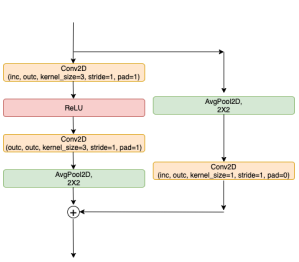
where $\lambda = 10$ (following defaults recommended in [4]) and $\mathcal{K} = 1$ corresponds to the default WGAN-GP model. As in the main paper, we denote † to represent models with adjusted LC relative to the original baselines.

See also §D and Fig. 4 for discussion and empirical results concerning the observed LC when using an SNGAN discriminator with a resnet-based convolutional architecture.

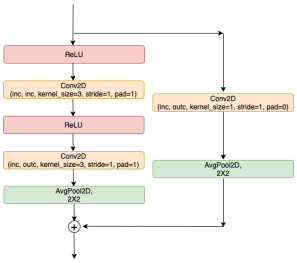
Appendix C: Wall-clock timings for a single training update

We summarize wall-clock times for a single training update that consists of one generator update and n_{dis} number of discriminator updates (including time for loading a batch of 64 images from the dataset). As can be noticed from Table 1, our method is roughly similar in wall-clock timings compared to WGAN-GP on smaller models (32^2 and 48^2) but slower than NSGAN or SNGAN. On 128^2 images GraN is 40% slower than WGAN-GP.

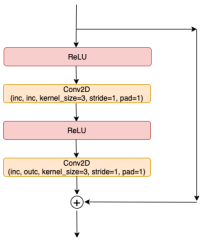
This is because gradient normalized discriminator (or critic) D requires computing the gradient norms on both the real and the fake samples when updating the parameters of D. In contrast, WGAN-GP only computes gradient norms on half the total number of real+fake samples which are random interpolates between the reals and fakes. Moreover, when updating G, computing generator loss $\mathcal{L}_G$ requires computing the gradient norm of D for GraN models, unlike WGAN-GP where gradient penalty affects only the param-



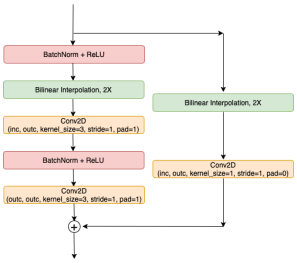
(a) Input_DBlock



(b) Down2X_DBlock



(c) DBlock



(d) Up2X_GBlock

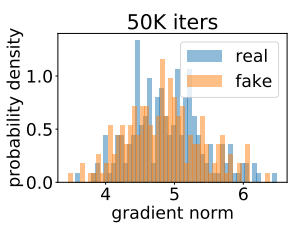
Figure 3: Residual block architectures for a) Input_DBlock, b) Down2X_DBlock, c) DBlock and d) Up2X_GBlock in Figures 1 and 2. `inc` and `outc` denote the input and output number of channels, respectively. Note that when `inc` $\neq$ `outc`, the skip connection in DBlock includes a 1×1 `Conv2D` appropriately. For SNGAN, the linear and convolution layers in Input_DBlock, DBlock and Down2X_DBlock are the spectral normalized versions.

Table 1: Wall-clock timings (in seconds $\times 10$) for a single training update across different dataset of different resolutions. Note that $n_{dis} = 5$ for CIFAR-10/100 and STL-10 while $n_{dis} = 2$ for LSUN/CelebA, following Mimicry [12].

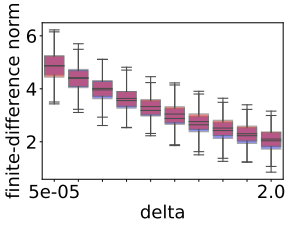
Method	**CIFAR-10** sec ($\times 10$)	**CIFAR-100** sec ($\times 10$)	**STL-10** sec ($\times 10$)	**LSUN** sec ($\times 10$)	**CelebA** sec ($\times 10$)
NSGAN	3.80±0.04	3.72±0.03	4.89±0.06	10.93±0.10	10.98±0.10
WGAN-GP	5.86±0.46	6.12±0.13	8.19±0.18	18.78±0.10	18.61±0.10
SNGAN	4.17±0.05	4.17±0.04	5.57±0.11	11.62±0.11	11.56±0.09
GraND-GAN (Ours)	5.66±0.04	5.69±0.04	8.83±0.07	26.34±0.08	26.13±0.10
GraNC-GAN (Ours)	5.69±0.04	5.65±0.04	8.83±0.08	26.11±0.10	26.18±0.18

eter updates for D at a given training iteration. GraN and WGAN-GP are both slower relative to NSGAN or SNGAN because they involve computing the gradient norm and back-propagating through it. We remark that further advances in the efficiency of back-propagation through network gradients could ameliorate this issue (e.g., AutoInt [13]).

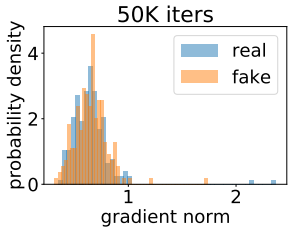
However, we note that, since the generator G architecture is identical across methods for a given dataset, at inference all methods fare equally in wall-clock timings for image generation.



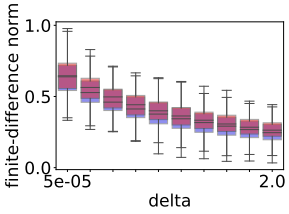
(a) Gradient norm (SNGAN with sum pooling)



(b) FD gradient norm (SNGAN with sum pooling)



(c) Gradient norm (SNGAN with average pooling)



(d) FD gradient norm (SNGAN with average pooling)

Figure 4: Gradient norms and finite-difference (FD) approximation to the gradient norms at increasing perturbation length `delta` along the gradient for SNGAN with global sum pooling ((a) and (b)) and global average pooling ((c) and (d)) at 50K iterations of training on CIFAR-10.

Appendix D: The Looseness of Layerwise Constraints

We consider the simple case of the composition of two linear layers, $z = f(g(x)) = B(Ax + a) + b$, where both f and g, have a sharp Lipschitz constant (LC) of one. The question is under what conditions does $f \circ g$ also have a sharp LC of one?

We first introduce some notation. Let M be an $m \times n$ matrix. We denote the singular value decomposition of M as $M = U_M \Sigma_M V_M^T$, where we take U_M and V_M to be square matrices (of sizes $m \times m$ and $n \times n$, respectively). Here Σ_M is a diagonal $m \times n$ matrix, with the non-negative singular values sorted in decreasing order down the diagonal [3]. Define $\sigma_1(M)$ to be the maximal SV of the matrix M. Moreover, define $\Gamma_\sigma(M)$ to be the projection from $\mathbb{R}^n$ to the subspace spanned by the right singular vectors of M for SV's equal to σ. That is,

$$\Gamma_\sigma(M) = V_M D_\sigma(M) V_M^T, \tag{1}$$

where $D_\sigma(M)$ is defined to be a diagonal $n \times n$ matrix where the i^{th} diagonal element is 1 when the corresponding element of Σ_M equals σ, and zero otherwise. It then follows that $\Gamma_\sigma(M^T)$ is the projection of $\mathbb{R}^m$ to the subspace spanned by the *left* singular vectors of M for the SV

σ. Finally, from the form of $\Gamma_\sigma(M)$ in (1) we can conclude

$$\sigma_1(\Gamma_\sigma(M)) = 1, \tag{2}$$
$$\Gamma_\sigma(M)\Gamma_\sigma(M) = \Gamma_\sigma(M), \tag{3}$$

so long as σ is an SV for M.

We can express the conditions that f, g, and $f \circ g$ all have sharp LC of one in terms of this notation. Specifically, the tight Lipschitz bounds for f, g and $f \circ g$ are $\sigma_1(B)$, $\sigma_1(A)$, and $\sigma_1(BA)$, respectively. Assuming SN has rescaled A and B appropriately, then the LC of g and f are both one and we have $\sigma_1(B) = \sigma_1(A) = 1$. Moreover, we see $f \circ g$ will have a sharp LC of one iff $\sigma_1(BA) = 1$. We examine this latter condition.

Theorem 1. *For A and B as above (with dimensions such that their product BA can be formed), with maximal SV's equal to one, the maximal SV of BA satisfies $\sigma_1(BA) \leq 1$. Further, equality of this bound holds if and only if*

$$\sigma_1(\Gamma_1(B)\Gamma_1(A^T)) = 1. \tag{4}$$

Proof of Theorem 1. Let $m_1 = \dim(\Gamma_1(A))$ and $n_1 = \dim(\Gamma_1(B))$ be the number of singular values equal to one in A and B, respectively. The assumption that $\sigma_1(A) = \sigma_1(B) = 1$ implies $m_1, n_1 > 0$.

Recall that the spectral norm of a matrix M, which is induced by the L_2 vector norm, can be defined via the largest singular value: $||M||_2 := \sigma_1(M)$, equivalently computed as

$$||M||_2 = \sup_{||x||=||y||=1} |y^T M x| = \sup_{x \neq 0} \frac{||Mx||}{||x||}. \tag{5}$$

It follows from (5) that $||BA||_2 \leq ||B||_2||A||_2$ (the submultiplicativity property), and hence

$$\sigma_1(BA) \leq \sigma_1(B)\sigma_1(A) = 1. \tag{6}$$

First we prove $\sigma_1(BA) = 1$ implies (4). Assume $\sigma_1(BA) = 1$. Then it follows from (5) that there exists a vector x such that $||x|| = 1$ and $z := BAx$ satisfies $||z|| = 1$. Let $y = Ax$. There are two cases to consider, either $||y|| < 1$, or $||y|| = 1$. However, since $z = By$ and $||B||_2 = \sigma_1(B) = 1$ we have $||z|| <= ||B||_2||y|| = ||y||$. Therefore the assumption $||y|| < 1$ leads to the contradiction $||z|| < 1$, and instead we must have

$$||y|| = ||Ax|| = ||x|| = 1, \tag{7}$$
$$||z|| = ||By|| = ||y|| = 1. \tag{8}$$

Equation (7) ensures $x \in \texttt{range}(\Gamma_1(A))$ and therefore $y = Ax \in \texttt{range}(\Gamma_1(A^T))$. Also, equation (8) ensures $y \in \texttt{range}(\Gamma_1(B))$. Therefore it follows that y is a unit vector such that $\Gamma_1(B)y = y$, and $\Gamma_1(A^T)y = y$. And thus, $\Gamma_1(B)\Gamma_1(A^T)y = y$. By (5) we then have $\sigma_1(\Gamma_1(B)\Gamma_1(A^T)) \geq 1$. But, from (2), it follows that $\sigma_1(\Gamma_1(B)) = \sigma_1(\Gamma_1(A^T)) = 1$ and therefore $\sigma_1(\Gamma_1(B)\Gamma_1(A^T)) \leq 1$. As a result we have shown (4), as desired.

For the reverse direction, assume $\sigma_1(\Gamma_1(B)\Gamma_1(A^T)) = 1$. Then (5) implies there exists a y such that $||y|| = 1$ and $z = \Gamma_1(B)\Gamma_1(A^T)y$ with $||z|| = 1$. But, since $\Gamma_1(B)$ and $\Gamma_1(A^T)$ are projection matrices (see (3)), it can be shown that we must have $\Gamma_1(A^T)y = y$, $\Gamma_1(B)y = y$, and $z = y$.[1] Moreover, since y is a right singular vector of A^T for singular value one, it follows that $x := A^T y = V_A \Sigma_A U_A^T y$ is a left singular vector of A^T for the SV at one and $||x|| = 1$. Therefore $x = \Gamma_1(A)x = V_A D_1(A) V_A^T x$. That is, x is in the right singular space of A for the SV at one and it follows that $y = Ax$. Taken together, we have $BAx = By$ and $y = \Gamma_1(B)y$, so $||By|| = 1$. Hence $||BAx|| = ||By|| = 1 = ||x||$. That is, from (5), it follows that $\sigma_1(BA) \geq 1$. Finally, from (6), we conclude $\sigma_1(BA) = 1$, as desired. □

Relation to Layerwise Spectral Normalization As described in the text, (Eq. 4) is a subspace alignment condition where $\sigma_1(\Gamma_1(B)\Gamma_1(A^T))$ equals the cosine of the first principal angle between the two subspaces $\Gamma_1(B)$ and $\Gamma_1(A^T)$ [3, 18]. It is therefore unlikely to be satisfied by chance, although during training the model may reduce this angle and approach $\sigma_1(\Gamma_1(B)\Gamma_1(A^T)) = 1$. Thus, with training, we might expect the norm of the gradients of $f(x)$ to increase towards an upper bound.

However, we note additional features of $f(x)$ for the architectures described in Figures 1 and 3. Specifically, the global sum pooling in Fig. 1 and the skip connections in Fig. 3 are both capable of amplifying the gradient norms through these stages by a factor greater than one. Thus, while the subspace alignment conditions can be expected to shrink the gradient magnitudes, these specific components can expand them. The consequence of these two opposing effects is not clear a priori.

The empirical results shown in Fig. 4a indicate that, for the cases tested, the net effect is for $f(x)$ to have a gradient norm larger than one. Moreover, when average pooling is used in place of the sum pooling, the norm of the gradient is predominantly less than one (see Fig. 4c). Similar properties are seen for the magnitudes of finite differences (FD) of $f(x)$ over steps of length δ (see Fig. 4b, 4d), as described in the paper.

Indeed, we can compute the LC for the 32×32 resnet-based convolutional discriminator (used on, e.g., CIFAR-10), shown in §B and Fig. 1, as follows. First, note that the four `DBlocks` have a skip connection, meaning the LC increases two-fold across each block, resulting in an LC of $2^4 = 16$ before pooling (assuming the SN keeps the convolutional

[1] The basic idea here is that if $z = Py$ for a projection P and $||z|| = ||y||$ then $||y||^2 = ||Py||^2 + ||(I-P)y||^2$ can be used to show $(I-P)y = 0$. Moreover, from (1), it then follows that $Py = y$.

Table 2: Mean $\pm$ standard deviation of IS, FID and KID across 3 training runs with random restarts on CIFAR-10. † indicates modified baselines with the Lipschitz constant $\mathcal{K} = 0.83$ that our methods use.

Model	**IS** $\uparrow$	**FID** $\downarrow$	**KID**($\times 1000$) $\downarrow$
NSGAN	7.35±0.25	26.85±5.16	17.81±3.79
WGAN-GP	7.42±0.02	22.44±0.35	20.67±0.31
SNGAN	8.06±0.04	17.22±0.16	12.44±0.25
NSGAN-GP†	8.01±0.04	15.69±0.15	12.95±0.21
NSGAN-SN†	7.72±0.06	21.12±0.59	15.79±0.42
WGAN-GP†	7.37±0.02	22.75±0.05	21.12±0.36
SNGAN†	7.98±0.01	16.86±0.40	12.16±0.38
GraND-GAN	8.00±0.01	15.60±0.47	12.80±0.42
GraNC-GAN	7.96±0.02	16.15±0.21	13.30±0.32

Table 3: Ablation of our method (GraND-GAN) on CIFAR-10 image generation under different values of the Lipschitz constant $\mathcal{K} = 1/\tau$ with $\epsilon = 0.1$.

$1/\tau$	**IS** $\uparrow$	**FID** $\downarrow$	**KID**($\times 1000$) $\downarrow$
0.1	7.709	18.303	15.4
0.5	7.919	15.689	12.8
0.83	8.031	14.965	12.3
1.0	8.011	15.469	12.2
1.33	8.111	14.561	10.9

Table 4: Ablation of our method (GraND-GAN) on CIFAR-10 image generation under different values of ϵ with $\mathcal{K} = 1/\tau = 0.83$.

ϵ	**IS** $\uparrow$	**FID** $\downarrow$	**KID**($\times 1000$) $\downarrow$
1e-08	8.065	15.076	11.9
0.0001	7.924	16.695	13.7
0.001	8.035	16.322	13.5
0.01	7.900	15.726	13.0
0.1	8.031	14.965	12.3
1.0	7.981	15.194	12.2

layer LCs at one). The first two blocks also have spatial downsampling, resulting in an 8×8 feature map that is sum-pooled. This pooling, along with the preceding skip connections, increases the final LC to $8 \times 8 \times 16 = 1024$, as mentioned in the main paper.

Appendix E: Variance of IS, FID and KID metrics across random training restarts for CIFAR-10

We report the mean and the standard deviations of the metrics reported (IS, FID, KID) across 3 different training runs with random restarts for CIFAR-10 in Table 2.

Table 5: Frequency of runs diverging (i.e., FID $\geq$ 40) on CIFAR-10 on three random restarts for GraND-GAN, GraNC-GAN, SNGAN, WGAN-GP, and NSGAN-GP† on CIFAR-10 for settings B, C and D.

Model	Setting	α	β_1	β_2	n_{dis}	#(FID $\geq$ 40)
WGAN-GP	B	0.0002	0.5	0.999	1	3/3
	C	0.001	0.5	0.999	5	3/3
	D	0.001	0.9	0.999	5	3/3
SNGAN	B	0.0002	0.5	0.999	1	3/3
	C	0.001	0.5	0.999	5	0/3
	D	0.001	0.9	0.999	5	0/3
NSGAN-GP†	B	0.0002	0.5	0.999	1	3/3
	C	0.001	0.5	0.999	5	3/3
	D	0.001	0.9	0.999	5	0/3
GraND-GAN	B	0.0002	0.5	0.999	1	2/3
	C	0.001	0.5	0.999	5	0/3
	D	0.001	0.9	0.999	5	0/3
GraNC-GAN	B	0.0002	0.5	0.999	1	2/3
	C	0.001	0.5	0.999	5	0/3
	D	0.001	0.9	0.999	5	0/3

Appendix F: Ablations on ϵ and τ

We also run ablations on our methods by varying the piecewise Lipschitz constant $\mathcal{K} = 1/\tau$ and ϵ for GraND-GAN on CIFAR-10 image generation. Tables 3 and 4 show that our method is fairly robust to a range of $\mathcal{K}$ and ϵ, respectively, on CIFAR-10. The role of hyperparameter ϵ is mainly numerical stability when the gradient norm becomes vanishingly small. Irrespective of ϵ used, we empirically find that the weights of the network scale up sufficiently large such that the input gradient norm of the GraNed output $g(x)$ is close to the upper bound $\mathcal{K}$, i.e., $\|\nabla_x g(x)\| \approx \mathcal{K}$. This is evident in Figure 3 of the main paper where the gradient norms for our methods have a very narrow distribution around $\mathcal{K}$ despite using $\epsilon = 0.1$.

Appendix G: Frequency of runs diverging on CIFAR-10 on three random restarts

We repeat the experiment in Figure 2 of the main paper for settings B ($\alpha = 0.0002,\ \beta_1 = 0.5,\ \beta_2 = 0.999,\ n_{\text{dis}} = 1$), C ($\alpha = 0.001,\ \beta_1 = 0.5,\ \beta_2 = 0.999,\ n_{\text{dis}} = 5$) and D ($\alpha = 0.001,\ \beta_1 = 0.9,\ \beta_2 = 0.999,\ n_{\text{dis}} = 5$) that have aggressive training settings over 3 random restarts. We call a run "diverging" when the FID ≥ 40 on CIFAR-10. Table 5 summarizes the number of diverging runs out of 3 random restarts.

Evidently, GraND-GAN and GraNC-GAN have the least number of runs that diverged in 3 random restarts across settings B, C and D. SNGAN comes close but diverges $3/3$ times for setting B when $n_{\text{dis}} = 1$. WGAN-GP performs

poorly across random restarts for B, C and D. NSGAN-GP† diverges $3/3$ times for settings B and C.

Appendix H: Qualitative Results

Figures 7, 8, 9, 5 and 6 present a sample of images generated by different methods for CIFAR-10, CIFAR-100, STL-10, LSUN-Bedrooms and CelebA, respectively. We compare the results of our methods qualitatively with the baselines (NSGAN, WGAN-GP, SNGAN) and the best model of the modified baselines (NSGAN-GP† where † represents an adjusted Lipschitz constraint to match the piecewise Lipschitz constant of our methods).

Appendix I: Soft versus Hard Hinge Performance

We tested GraNC-GAN on both soft and hard hinge losses (recalling that the soft hinge loss is obtained by replacing the ReLU non-linearity in the standard hard hinge loss with the softplus activation). On LSUN, GraNC-GAN struggles to converge with hard hinge, while it outperforms SNGAN with soft hinge loss. Moreover, if one lowers the LRs on LSUN (to be those used by GraNC-GAN on CelebA; see §A), the soft hinge version performs better by $\sim$6 FID (specifically, 20.2 vs. 26.3). On CelebA, using hard hinge resulted in an FID of 14, two points higher than that obtained via soft hinge (12), as displayed in the main paper. Altogether, these suggest the soft hinge loss is generally more performant and stable than the standard hard hinge function, at least for GraN. Previous works, such as SNGAN, also note such instabilities across different loss functions, and, therefore, switch from the Wasserstein loss to the (hard) hinge loss in their work. In our case, soft hinge loss was found to work the best.

Appendix J: Effect of ϵ_{Adam} in the Adam update on GAN training

To illustrate a qualitative effect of tuning ϵ_{Adam} in the Adam update on training GANs, we train GraNC-GAN on CIFAR-10 with Hinge loss for 1000 iterations, fixing the Lipschitz constant $\mathcal{K} = 1$. We train two models, one with $\epsilon_{\text{Adam}} = 1 \times 10^{-8}$ (default value) and another model with $\epsilon_{\text{Adam}} = 1 \times 10^{-7}$ (i.e., $10\times$ larger than the default). Figure 10 show the qualitative results of a few examples sampled from the generators.

As noted in the main paper, tuning the Lipschitz constant $\mathcal{K}$ has an effect that is equivalent to changing ϵ_{Adam}. Figure 10 qualitatively demonstrates that tuning ϵ_{Adam} (or $\mathcal{K}$, in effect) affects GAN training considerably.

Appendix K: Stability of Modern GANs

Recent families of GANs, including those based on BigGAN [2] and StyleGAN [9], have achieved unprecedented synthesis results; yet, they are not immune from instability issues. BigGAN devotes a significant portion of their paper to understanding stability (see, e.g., Sections 4.1 and 4.2 on "characterizing instability"). Furthermore, they note that "it is possible to enforce stability by strongly constraining D, but doing so incurs a dramatic cost in performance." Instability persists even within more recent methods that are based on BigGAN, such as U-net GAN [16], which experiences $\sim 40\%$ of its runs failing. While StyleGAN does not present a stability analysis, their network relies heavily on progressive growing [8] for stability, which induces artifacts (and additional training complexity) addressed in follow-up work (StyleGANv2 [10]). Similarly, MSG-GAN [7] demonstrates improved stability of its technique over progressive growing. In other words, despite steady improvements, GAN stability remains a significant challenge, even for modern architectures. See also [17] for a recent survey of stabilization techniques.

References

[1] Mikołaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[3] Gene H Golub and Charles F Van Loan. *Matrix computations (fourth edition)*. JHU press, 2013.

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5769–5779, 2017.

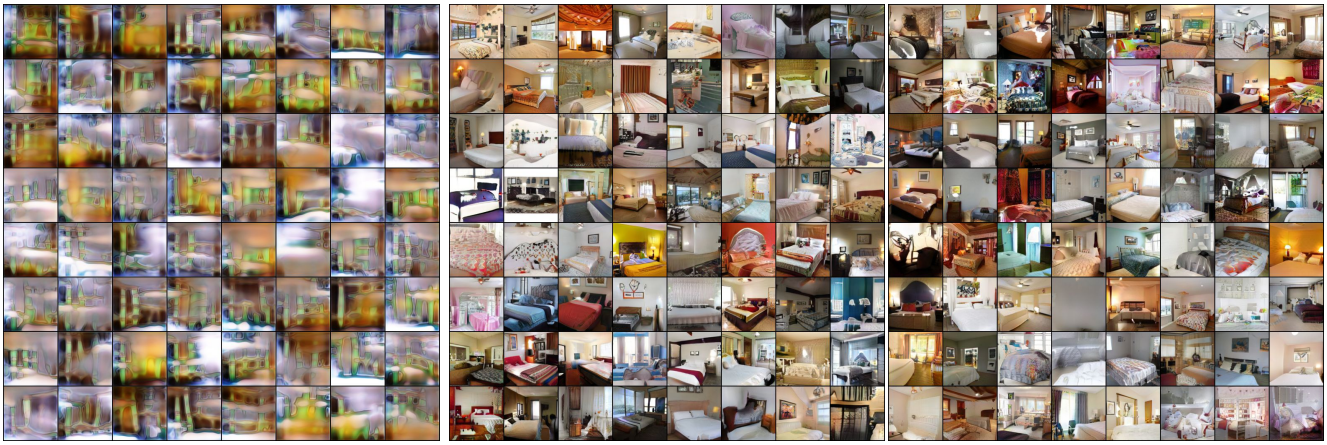
[5] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

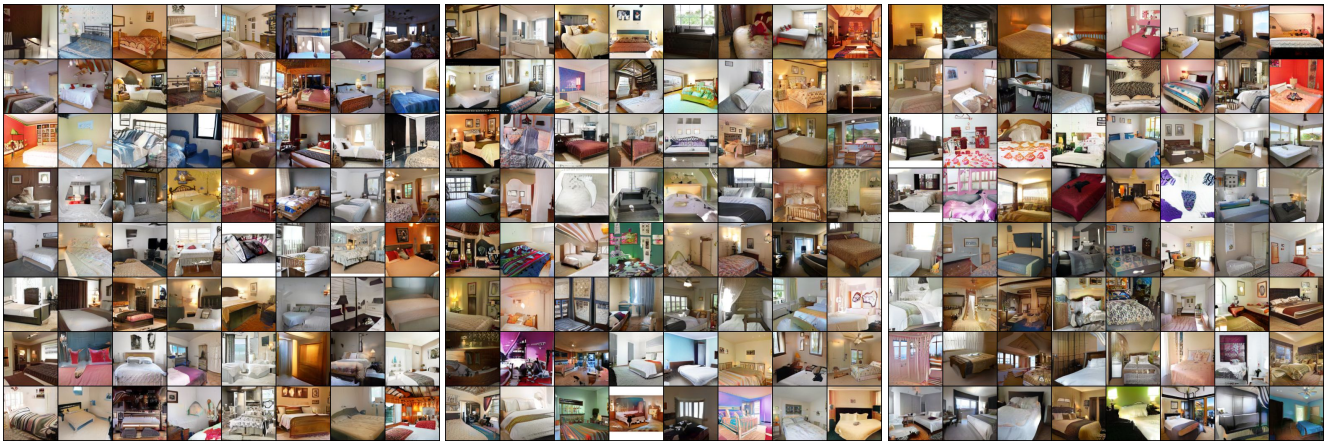
[7] Animesh Karnewar and Oliver Wang. Msg-gan: Multi-scale gradients for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7799–7808, 2020.

[8] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[9] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks.



(a) NSGAN (b) WGAN-GP (c) SNGAN



(d) NSGAN-GP† (e) GraND-GAN (f) GraNC-GAN

Figure 5: Qualitative results on LSUN-Bedrooms across different models, including baselines (NSGAN, WGAN-GP, SNGAN), the best performing modified baseline (NSGAN-GP†) and our methods (GraND-GAN and GraNC-GAN). Zoom in for better viewing.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

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

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

[12] Kwot Sin Lee and Christopher Town. Mimicry: Towards the reproducibility of gan research. *CVPR Workshop on AI for Content Creation*, 2020.

[13] David B. Lindell, Julien N. P. Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14556–14565, June 2021.

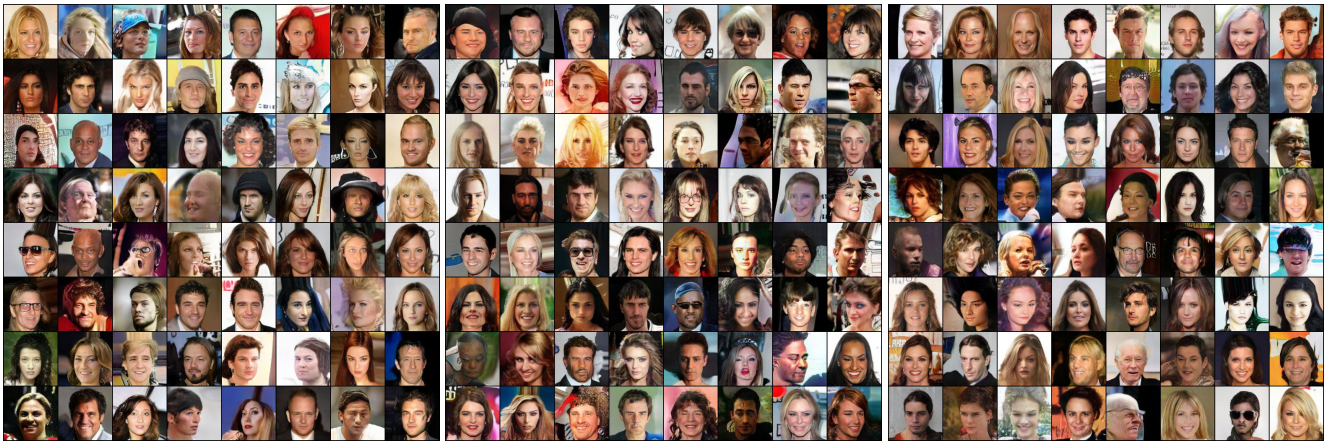
[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[15] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[16] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. A u-net based discriminator for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8207–8216, 2020.

[17] Maciej Wiatrak, Stefano V Albrecht, and Andrew Nystrom.

(a) NSGAN (b) WGAN-GP (c) SNGAN

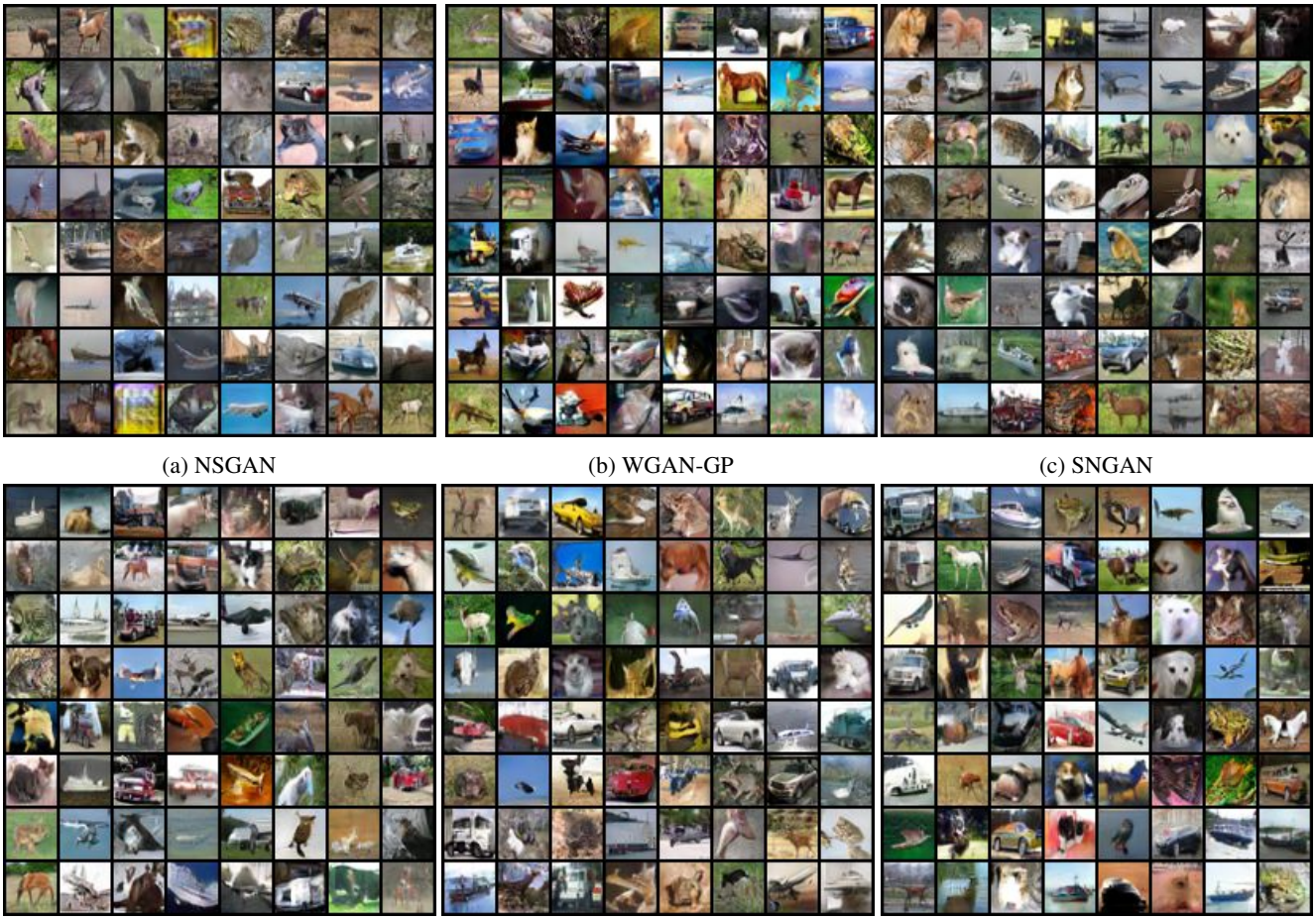


(d) NSGAN-GP† (e) GraND-GAN (f) GraNC-GAN

Figure 6: Qualitative results on CelebA across different models, including baselines (NSGAN, WGAN-GP, SNGAN), the best performing modified baseline (NSGAN-GP†) and our methods (GraND-GAN and GraNC-GAN). Zoom in for better viewing.

Stabilizing generative adversarial networks: A survey. *arXiv preprint arXiv:1910.00927*, 2019.

[18] Peizhen Zhu and Andrew V Knyazev. Angles between subspaces and their tangents. *arXiv preprint arXiv:1209.0523*, 2012.



(a) NSGAN (b) WGAN-GP (c) SNGAN

(d) NSGAN-GP† (e) GraND-GAN (f) GraNC-GAN

Figure 7: Qualitative results on CIFAR-10 across different models, including baselines (NSGAN, WGAN-GP, SNGAN), the best performing modified baseline (NSGAN-GP†) and our methods (GraND-GAN and GraNC-GAN). Zoom in for better viewing.

(a) NSGAN (b) WGAN-GP (c) SNGAN

(d) NSGAN-GP† (e) GraND-GAN (f) GraNC-GAN

Figure 8: Qualitative results on CIFAR-100 across different models, including baselines (NSGAN, WGAN-GP, SNGAN), the best performing modified baseline (NSGAN-GP†) and our methods (GraND-GAN and GraNC-GAN). Zoom in for better viewing.

(a) NSGAN (b) WGAN-GP (c) SNGAN

(d) NSGAN-GP† (e) GraND-GAN (f) GraNC-GAN

Figure 9: Qualitative results on STL-10 across different models, including baselines (NSGAN, WGAN-GP, SNGAN), the best performing modified baseline (NSGAN-GP†) and our methods (GraND-GAN and GraNC-GAN). Zoom in for better viewing.

(a) $\epsilon_{\text{Adam}} = 1 \times 10^{-8}$ (b) $\epsilon_{\text{Adam}} = 1 \times 10^{-7}$

Figure 10: Qualitative comparison of generated CIFAR-10 samples under two different ϵ_{Adam} hyperparameter settings. Tuning ϵ_{Adam} affects GAN training. Zoom in for better viewing.