The GAN Landscape: Losses, Architectures, Regularization, and Normalization

Karol Kurach* **Mario Lucic*** **Xiaohua Zhai** **Marcin Michalski** **Sylvain Gelly**
Google Brain

Abstract

Generative Adversarial Networks (GANs) are a class of deep generative models which aim to learn a target distribution in an unsupervised fashion. While they were successfully applied to many problems, training a GAN is a notoriously challenging task and requires a significant amount of hyperparameter tuning, neural architecture engineering, and a non-trivial amount of "tricks". The success in many practical applications coupled with the lack of a measure to quantify the failure modes of GANs resulted in a plethora of proposed losses, regularization and normalization schemes, and neural architectures. In this work we take a sober view of the current state of GANs from a practical perspective. We reproduce the current state of the art and go beyond fairly exploring the GAN landscape. We discuss common pitfalls and reproducibility issues, open-source our code on Github, and provide pre-trained models on TensorFlow Hub.

1 Introduction

Deep generative models can be applied to the task of learning a target distribution. They were recently exploited in a variety of applications unleashing their full potential in the context of natural images [14, 23, 1, 28]. Generative adversarial networks (GANs) [9] are one of the main approaches to learning such models in a fully unsupervised fashion. The GAN framework can be viewed as a two-player game where the first player, the *generator*, is learning to transform some simple input distribution (usually a standard multivariate Normal or uniform) to a distribution on the space of images, such that the second player, the *discriminator*, cannot tell whether the samples belong to the true distribution or were synthesized [9]. Both players aim to minimize their own loss and the solution to the game is the Nash equilibrium where neither player can improve their loss unilaterally. The GAN framework can also be derived by minimizing a statistical divergence between the model distribution and the true distribution [9, 21, 19, 2].

Training GANs requires solving a minimax problem over the parameters of the generator and the discriminator. Since both generator and discriminator are usually parametrized as deep convolutional neural networks, this minimax problem is notoriously hard in practice [10, 2, 18]. As a result, a plethora of loss functions, regularization and normalization schemes, coupled with neural architecture choices, have been proposed [9, 25, 20, 10, 2, 19]. Some of these are derived based on theoretical insights, while others were inspired by practical considerations.

Our contributions. In this work we provide a thorough empirical analysis of these competing approaches, and help the researchers and practitioners navigate this space. We first define the GAN landscape – the set of loss functions, normalization and regularization schemes, and the most commonly used architectures. We explore this search space on several modern large-scale data sets

*Indicates equal authorship. Correspondence to Karol Kurach (kkurach@google.com) and Mario Lucic (lucic@google.com).

2nd Reproducibility in Machine Learning Workshop at ICML 2018, Stockholm, Sweden.

by means of hyperparameter optimization, considering both "good" sets of hyperparameters reported in the literature, as well as ones obtained by Gaussian Process regression. By analyzing the impact of the loss function, we conclude that the non-saturating loss [9] is sufficiently stable across data sets, architectures and hyperparameters. We then proceed to decompose the effect of various normalization and regularization schemes, as well as varying architectures. We show that both gradient penalty [10] as well as spectral normalization [20] are useful in the context of high-capacity architectures. We then show that one can further benefit from simultaneous regularization and normalization. Finally, we propose a discussion of common pitfalls, reproducibility issues, and practical considerations. We provide all the reference implementations, including training and evaluation code on Github[2], and provide pre-trained models on TensorFlow Hub[3].

2 The GAN Landscape

2.1 Loss Functions

Let P denote the target (true) distribution and Q the model distribution. The original GAN formulation [9] suggests two loss functions: the minimax GAN and the non-saturating (NS) GAN. In the former the discriminator minimizes the negative log-likelihood for the binary classification task (i.e. is the sample true or fake) and is equivalent to minimizing the Jensen-Shannon (JS) divergence between P and Q. In the latter the generator maximizes the probability of generated samples being real. In this work we consider the non-saturating loss as it is known to outperform the minimax variant [9]. The corresponding loss functions are defined as $\mathcal{L}_{\mathrm{D}} = -\mathbb{E}_{x\sim P}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim Q}[\log(1 - D(\hat{x}))]$ and $\mathcal{L}_{\mathrm{G}} = -\mathbb{E}_{\hat{x}\sim Q}[\log(D(\hat{x}))]$.

In Wasserstein GAN (WGAN) [2] it is proposed to minimize the Wasserstein distance between P and Q. Exploiting the connection to optimal transport they prove that under an optimal discriminator, minimizing the value function with respect to the generator minimizes the Wasserstein distance between P and Q. The drawback is that one has to ensure a 1-Lipschitz discriminator due to exploited Kantorovich-Rubenstein duality. To achieve this, the discriminator weights are clipped to a small absolute value. The corresponding loss functions are defined as $\mathcal{L}_{\mathrm{D}} = -\mathbb{E}_{x\sim P}[D(x)] + \mathbb{E}_{\hat{x}\sim Q}[D(\hat{x})]$ and $\mathcal{L}_{\mathrm{G}} = -\mathbb{E}_{\hat{x}\sim Q}[D(\hat{x})]$. As the final considered loss function, we consider the least-squares loss (LS) which corresponds to minimizing the Pearson χ^2 divergence between P and Q [19]. The intuition is that this loss function is smooth and saturates slower than the sigmoid cross-entropy loss of the JS formulation [9]. The corresponding loss functions are defined as $\mathcal{L}_{\mathrm{D}} = -\mathbb{E}_{x\sim P}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x}\sim Q}[D(\hat{x})^2]$ and $\mathcal{L}_{\mathrm{G}} = -\mathbb{E}_{\hat{x}\sim Q}[(D(\hat{x}) - 1)^2]$.

2.2 Regularization and Normalization of the Discriminator

Gradient norm penalty. In the context of Wasserstein GANs this penalty can be interpreted as a soft penalty for the violation of 1-Lipschitzness (WGAN GP) [10]. Hereby, the gradient is evaluated on a linear interpolation between training points and generated samples as a proxy to the optimal coupling [10]. The gradient penalty can also be evaluated around the data manifold which encourages the discriminator to be piece-wise linear in that region (DRAGAN) [17]. However, the gradient norm penalty can be considered purely as a regularizer for the discriminator and it was shown that it can improve the performance for other losses, not only the WGAN [8]. Furthermore, the penalty can be scaled by the "confidence" of the discriminator in the context of f-divergences [24]. A drawback of gradient penalty (GP) regularization scheme [10] is that it can depend on the model distribution Q which changes during training. The drawback of DRAGAN is that it is unclear how to exactly define the manifold. Finally, computing the gradient norms implies a non-trivial running time penalty – essentially doubling the running time. Finally, we also investigate the impact of a regularizer ubiquitous in supervised learning – the L_2 penalty on all the weights of the network.

Discriminator normalization. Normalizing the discriminator can be useful from both the optimization perspective (more efficient gradient flow, a more stable optimization), as well as from the representation perspective – the representation richness of the layers in a neural network depends on the spectral structure of the corresponding weight matrices.

[2] Available at `http://www.github.com/google/compare_gan`.

[3] Available at `http://www.tensorflow.org/hub`.

From the optimization point of view, several techniques have found their way into the GAN literature, namely Batch normalization [13] and Layer normalization (LN) [3]. Batch normalization (BN) in the context of GANs was suggested by [7] and further popularized by [23]. It normalizes the pre-activations of nodes in a layer to mean β and standard deviation γ, where both β and γ are parameters learned for each node in the layer. The normalization is done on the batch level and for each node separately. In contrast, with Layer normalization, all the hidden units in a layer share the same normalization terms β and γ, but different samples are normalized differently [3]. Layer normalization was first applied in the context of GANs in [10].

From the representation point of view, one has to consider the neural network as a composition of (possibly non-linear) mappings and analyze their spectral properties. In particular, for the discriminator to be a bounded linear operator it suffices to control the maximum singular value. This approach is followed in [20] where the authors suggest dividing each weight matrix, including the matrices representing convolutional kernels, by their spectral norm. We note that, while this approach guarantees 1-Lipschitzness for linear layers and ReLu activation units, bounding the spectral norm of the kernel of the convolutional map to 1 *does not* bound the spectral norm of the convolutional mapping to 1. In fact, depending on stride and padding used, the norm might be off by a factor proportional to the number of filters. We discuss the practical implications of this issue in Section 5. Furthermore, the authors argue that a key advantage of spectral normalization over competing approaches is that it results in discriminators of higher rank [20].

2.3 Generator and Discriminator Architecture

We explore two classes of architectures in this study: deep convolutional generative adversarial networks (DCGAN) [23] and residual networks (ResNet) [11]. These architectures are ubiquitous in GAN research [9, 2, 25, 10, 20]. **DCGAN** [23] extended the GAN idea to deep convolutional networks for image generation. Both the discriminator and generator networks contain 5 layers. Recently, [20] defined a variation of DCGAN with spectral normalization, so called *SNDCGAN*. Apart from minor updates (cf. Section 4) the main difference to DCGAN is the use of an eight-layer discriminator network. The detailes of both networks are summarized in Table 2. **ResNet19** is an architecture with five ResNet blocks in the generator and six ResNet blocks in the discriminator, that can operate on 128×128 images. We follow the ResNet setup from [20], with the small difference that we simplified the design of the discriminator. In particular, we downsample in every discriminator block and the first block does not contain any custom changes. Each ResNet block consists of three convolutional layers, which results in 19 layers in total for the discriminator. The detailed parameters of discriminator and generator are summarized in Table 3a and Table 3b. With this setup we were able to reproduce and improve on the current state of the art results.

2.4 Evaluation Metrics

An in-depth overview of available metrics is outside of the scope of this work and we refer the reader to [5]. We instead focus on several recently proposed metrics well suited to the image domain.

Inception Score (IS). Proposed by [25], IS offers a way to quantitatively evaluate the quality of generated samples. Intuitively, the conditional label distribution of samples containing meaningful objects should have low entropy, and the variability of the samples should be high. which can be expressed as $\texttt{IS} = \exp(\mathbb{E}_{x\sim Q}[d_{KL}(p(y \mid x), p(y))])$. The authors found that this score is well-correlated with scores from human annotators [25]. Drawbacks include insensitivity to the prior distribution over labels and not being a proper *distance*.

As an alternative [12] propose the **Frechet Inception Distance (FID)**. Samples from P and Q are first embedded into a feature space (a specific layer of InceptionNet). Then, assuming that the embedded data follows a multivariate Gaussian distribution, the mean and covariance are estimated. Finally, the Fréchet distance between these two Gaussians is computed, i.e.

$$\texttt{FID} = ||\mu_x - \mu_y||_2^2 + \mathrm{Tr}(\Sigma_x + \Sigma_y - 2(\Sigma_x\Sigma_y)^{\frac{1}{2}}),$$

where (μ_x, Σ_x), and (μ_y, Σ_y) are the mean and covariance of the embedded samples from P and Q, respectively. The authors argue that FID is consistent with human judgment and more robust to noise than IS. Furthermore, the score is sensitive to the visual quality of generated samples – introducing noise or artifacts in the generated samples will reduce the FID. In contrast to IS, FID can detect

intra-class mode dropping, i.e. a model that generates only one image per class can score a perfect IS, but will have a bad FID [18]. As an unbiased alternative, [4] introduced the **Kernel Inception distance (KID)**. A thorough empirical investigation is outside of the scope of this work due to a significantly higher computational cost.

Multi-scale Structural Similarity for Image Quality (MS-SSIM) and Diversity. A critical issue in GANs are *mode collapse* and *mode-dropping* – failing to capture a mode, or low-diversity of generated samples from a given mode. The MS-SSIM score [30] is used for measuring the similarity of two images where higher MS-SSIM score indicates more similar images. Several recent works suggest using the average pairwise MS-SSIM score within a given class as a proxy for the diversity of generated samples [22, 8]. The drawback of this approach is that we do not know the class corresponding to the generated sample, so it is usually applied on one-class data sets, such as CELEBA-HQ-128. In this work we use the same setup as in [8]. In particular, given a batch size b, we compute the average pairwise MS-SSIM score on 5 batches, of $5 \times b \times (b-1)/2$ image pairs in total. We stress that the diversity should only be taken into account *together with the FID and IS metrics*.

2.5 Data Sets

We consider three data sets, namely CIFAR10, CELEBA-HQ-128, and LSUN-BEDROOM. The LSUN-BEDROOM data set [31] contains slightly more than 3 million images and was already explored in several papers [23, 10][4]. We randomly partition the images into a train and test set whereby we use 30588 images as the test set. Secondly, we use the CELEBA-HQ data set of 30k images [15]. We use the $128 \times 128 \times 3$ version obtained by running the code provided by the authors.[5] We use 3000 examples as the test set and the remaining examples as the training set. Finally, in order to reproduce existing results, we also include the CIFAR10 data set which contains 70K images (32x32x3), partitioned into 60000 training instances and 10000 testing instances. The baseline FID scores are 12.6 for CELEBA-HQ-128, 3.8 for LSUN-BEDROOM, and 5.19 for CIFAR10. Details on FID computation can be found in Section 4.

2.6 Exploring the GAN Landscape

The search space for GANs is prohibitively expensive: exploring all combinations of all losses, normalization and regularization schemes, and architectures is outside of the practical realm. Instead, in this study we analyse several slices of this tensor for each data set. In particular, to ensure that we can reproduce existing results, we perform a study over the subset of this tensor on CIFAR10. We then proceed to analyze the performance of these models across CELEBA-HQ-128 and LSUN-BEDROOM. In Section 3.1 we fix everything but the loss. In Section 3.2 we fix everything but the regularization and normalization scheme. Finally, in Section 3.3 we fix everything but the architecture. This allows us to decouple some of these design choices and provide some insight on what matters most.

As noted in [18], one major issue preventing further progress is the hyperparameter tuning – currently, the community has converged to a small set of parameter values which work on some data sets, and may completely fail on others. In this study we combine the best hyperparameter settings found in the literature [20], and perform Gaussian Process regression in the bandit setting [27] to possibly uncover better hyperparameter settings. Then, we consider the union of these results and select the top performing models and discuss the impact of the computational budget [18].

We summarize the fixed hyperparameter settings in Table 1a which contains the „good" parameters reported in recent publications [8, 20, 10]. In particular, we consider the cross product of these parameters to obtain 24 hyperparameter settings to reduce the bias. Finally, to provide a fair comparison, we perform Gaussian Process optimization in the bandit setting [27] on the parameter ranges provided in Table 1b. We run 12 rounds (i.e. we communicate with the oracle 12 times) of the optimization, each with a batch of 10 hyperparameter sets selected based on the FID scores from the results of the previous iterations. As we explore the number of discriminator updates per generator update (1 or 5), this leads to an additional 240 hyperparameter settings which in some cases outperform the previously known hyperparameter settings. Batch size is set to 64 for all the experiments. We use a fixed the number of discriminator update steps of 100K for LSUN-BEDROOM data set and CELEBA-HQ-128 data set, and 200K for CIFAR10 data set. For stochastic optimization we apply the Adam optimizer [16].

[4] The images are preprocessed to $128 \times 128 \times 3$ using TensorFlow resize_image_with_crop_or_pad.

[5] Available online at https://github.com/tkarras/progressive_growing_of_gans.

Table 1: Hyperparameter ranges used in this study. The Cartesian product of the fixed values suffices to uncover the existing results. Gaussian Process optimization in the bandit setting [27] is used to select good hyperparameter settings from the specified ranges.

(a) Fixed values

PARAMETER	DISCRETE VALUE
Learning rate α	$\{0.0002, 0.0001, 0.001\}$
Reg. strength λ	$\{1, 10\}$
$(\beta_1, \beta_2, n_{dis})$	$\{(0.5, 0.900, 5), (0.5, 0.999, 1), (0.5, 0.999, 5), (0.9, 0.999, 5)\}$

(b) Gaussian Process regression ranges

PARAMETER	RANGE	LOG
Learning rate α	$[10^{-5}, 10^{-2}]$	Yes
λ for L_2	$[10^{-4}, 10^{1}]$	Yes
λ for non-L_2	$[10^{-1}, 10^{2}]$	Yes
$\beta_1 \times \beta_2$	$[0, 1] \times [0, 1]$	No

3 Results and Discussion

Given that there are 4 major components (loss, architecture, regularization, normalization) to analyze for each data set, it is infeasible to explore the whole landscape. Hence, we opt for a more pragmatic solution – we keep some dimensions fixed, and vary the others. For each experiment we highlight three aspects: (1) FID distribution of the top 5% of the trained models, (2) the corresponding sample diversity score, and (3) the tradeoff between the computational budget (i.e. number of models to train) and model quality in terms of FID. Each model from the fixed seed set was trained 5 times with a different random seed and we report the median score. The variance for the seeds obtained by Gaussian Process regression is handled implicitly so we train each model once.

3.1 Impact of the Loss Function

Here the loss is either the non-saturating loss (NS) [9], the least-squares loss (LS) [19], or the Wasserstein loss (WGAN) [2]. We use the ResNet19 with generator and discriminator architectures detailed in Table 3a. We consider the most prominent normalization and regularization approaches: gradient penalty [10], and spectral normalization [20]. Both studies were performed on CELEBA-HQ-128 and LSUN-BEDROOM with hyperparameter settings shown in Table 1a.

The results are presented in Figure 2. We observe that the non-saturating loss is stable over both data sets. Spectral normalization improves the quality of the model on both data sets. Similarly, the

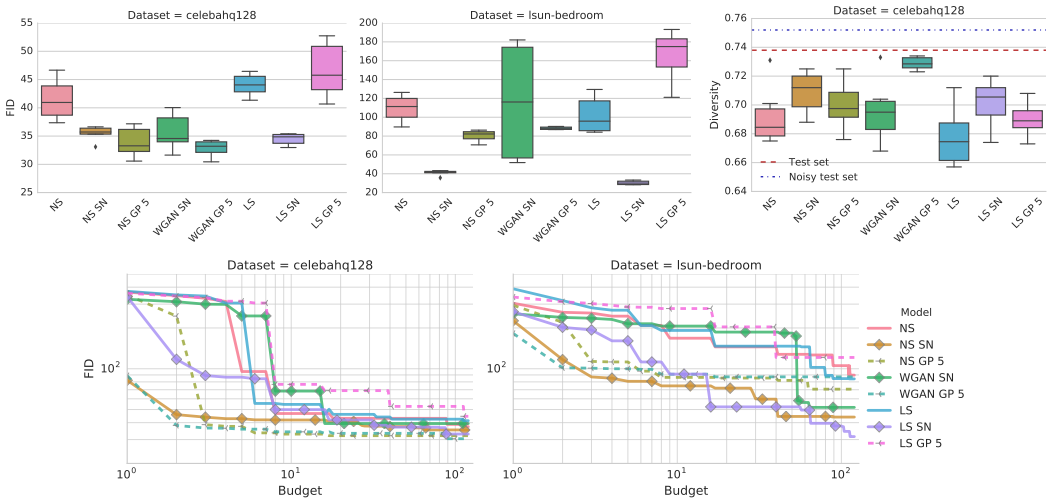


Figure 1: The non-saturating (NS) loss is stable over both data sets. Gradient penalty and spectral normalization improve the model quality. From the computational budget perspective (i.e. how many models one needs to train to reach a certain FID), both spectral normalization and gradient penalty perform better than the baseline, but the former is more efficient.

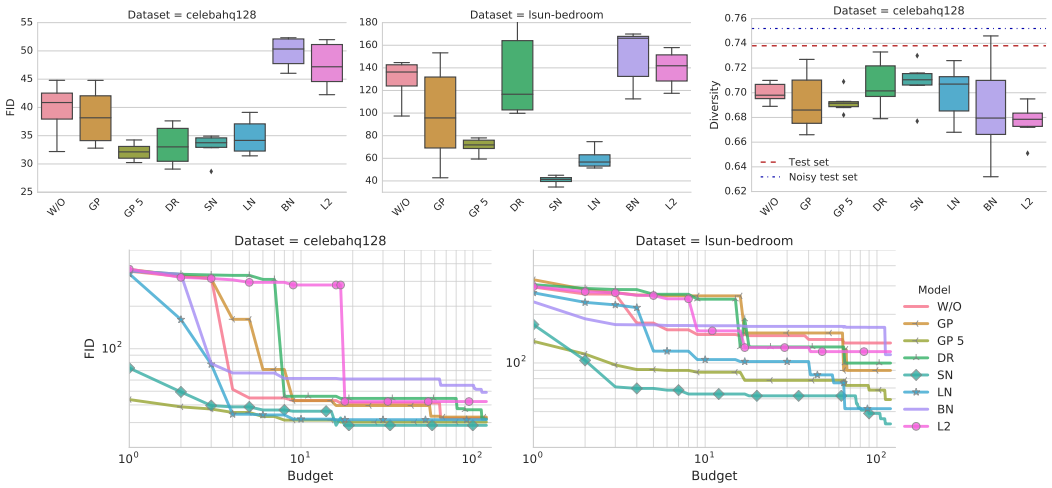


Figure 2: Both the gradient penalty (GP) and spectral normalization (SN) perform well and should be considered as viable approaches, and the latter is is computationally cheaper. Unfortunately none fully address the stability issues.

gradient penalty can help improve the quality of the model, but finding a good regularization tradeoff is non-trivial and requires a high computational budget. Models using the GP penalty benefit from 5:1 ratio of discriminator to generator updates as suggested by [10].

3.2 Impact of Regularization and Normalization

The goal of this study is to compare the relative performance of various regularization and normalization methods presented in the literature. To this end, and based on the loss study, we fix the loss to non-saturating loss [9]. We use the ResNet19 with generator and discriminator architectures described in Table 3a. Finally, we consider Batch normalization (BN) [13], Layer normalization (LN) [3], Spectral normalization (SN), Gradient penalty (GP) [10], Dragan penalty (DR) [17], or L_2 regularization. We consider both CELEBA-HQ-128 and LSUN-BEDROOM with the hyperparameter settings shown in Table 1a and Table 1b.

The results are presented in Figure 2. We observe that adding batch norm to the discriminator hurts the performance. Secondly, gradient penalty can help, but it doesn't stabilize the training. In fact, it is non-trivial to strike a balance of the loss and regularization strength. Spectral normalization helps improve the model quality and is more computationally efficient than gradient penalty. This is consistent with recent results in [32]. Similarly to the loss study, models using GP penalty benefit from 5:1 ratio of discriminator to generator updates. Furthermore, in a separate ablation study we observed that running the optimization procedure for an additional 100K steps is likely to increase the performance of the models with GP penalty, as suggested by [10].

Impact of Simultaneous Regularization and Normalization. Given the folklore that the Lipschitz constant of the discriminator is critical for the performance, one may expect simultaneous regularization and normalization could improve model quality. To quantify this effect, we fix the loss

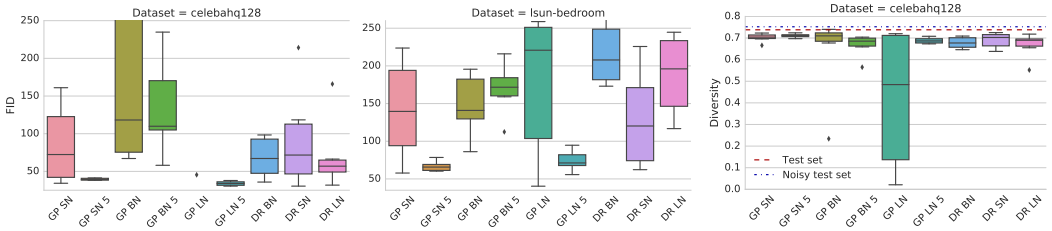


Figure 3: Gradient penalty coupled with spectral normalization (SN) or layer normalization (LN) strongly improves the performance over the baseline.

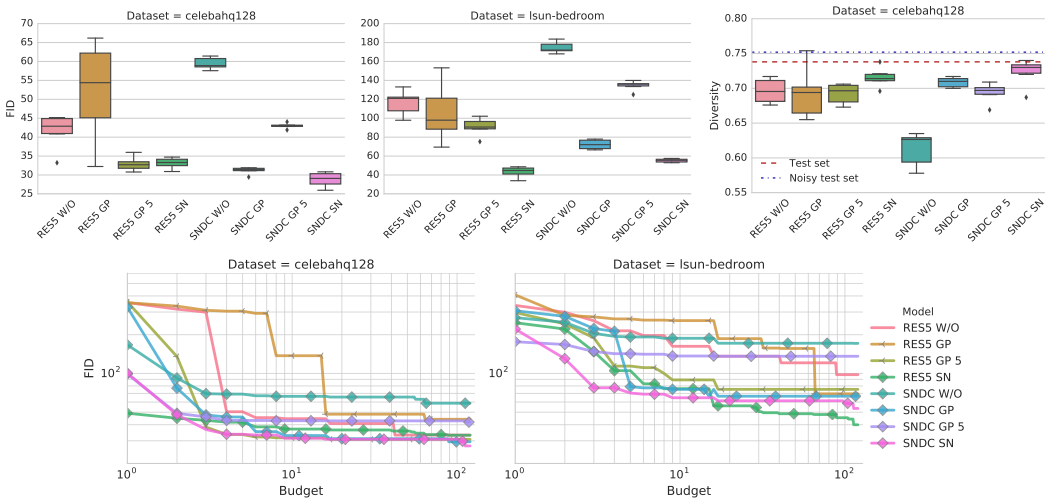


Figure 4: Impact of the discriminator and generator architecture for the non-saturating GAN loss. Both Spectral normalization and Gradient penalty can help improve upon the non-regularized baseline.

to non-saturating loss [9], use the Resnet19 architecture (as above), and combine several normalization and regularization schemes, with hyperparameter settings shown in Table 1a coupled with 24 randomly selected parameters. The results are presented in Figure 3. We observe that one may benefit from additional regularization and normalization. However, a lot of computational effort has to be invested for somewhat marginal gains in FID. Nevertheless, given enough computational budget we advocate simultaneous regularization and normalization – spectral normalization and layer normalization seem to perform well in practice.

3.3 Impact of Generator and Discriminator Architectures

An interesting practical question is whether our findings also hold for a different model capacity. To this end, we opt for the DCGAN [23] style architecture from [20]. We again perform the study with the non-saturating GAN loss, the Gradient penalty and Spectral normalization. We contrast the results obtained without regularization. For smaller architectures it was already noted in [18] that the gradient penalty is not essential. Here however, the regularization and normalization effects become more relevant, due to deeper architectures and optimization stability.

The results are presented in Figure 4. We observe that both architectures achieve comparable results and benefit from regularization and normalization. Spectral normalization strongly outperforms the baseline for both architectures.

4 Common Pitfalls

In this section we focus on several pitfalls we encountered while trying to reproduce existing results and provide a fairly and accurate comparison.

Metrics. There already seems to be a divergence in how the FID score is computed: (1) Some authors report the score on training data, yielding a FID between 50k training and 50k generated samples [29]. Some opt to report the FID based on 10k test samples and 5k generated samples [20]. Finally, [18] report the score with respect to the test data, in particular FID between 10k test samples, and 10k generated samples. The subtle differences will result in an mismatch between the reported FIDs, in some cases of more than 10%. We argue that FID should be computed with respect to the test data set [18] and use 10000 test samples and 10000 generated samples on CIFAR10 and LSUN-BEDROOM, and 3000 vs 3000 on CELEBA-HQ-128. Similarly, there are several ways to compute a diversity score using MS-SSIM and we follow the approach from [8]. We provide the implementation details in Section E of the Appendix.

Details of neural architectures. Even in popular architectures, like ResNet, there is still a number of design decision one needs to make, that are often omitted from the reported results. Those include the exact design of the ResNet cell (order of layers, when is ReLu applied, when to upsample and

downsample, how many filters to use). On top of these choices authors often include essentially arbitrary "customizations", which may result in wildly differing model capacity and hence potentially unfair comparison. Based on several ablation studies we can confidently say that these modifications often result in marginal improvements or deterioration, and hence serve no purpose other than introducing friction in the research process. As a result, we suggest to use the architectures presented within this work as a solid baseline.

Data sets. A common issue is related to data set processing – does LSUN-BEDROOM always correspond to the same data set? The authors usually don't bother to mention how precisely was the data set scaled down or up, which introduces inconsistencies between results on the "same" data set.

Implementation details and non-determinism. One major issue is the mismatch between the algorithm presented in a paper and the code provided online. We are aware that there is an embarrassingly large gap between a good implementation and a bad implementation of a given model. Hence, when no code is available, one is forced to guess which modifications were done. Another particularly tricky issue is removing randomness from the training process. After one fixes the data ordering and the initial weights, obtaining the same score by training the same model twice is non-trivial due to randomness present in certain GPU operations [6]. Disabling the optimizations causing the non-determinism often results in an order of magnitude running time penalty.

While each of these issues taken in isolation seems minor, they compound to produce a mist around existing results which introduces friction in practical applications and the research process [26].

5 Related Work

A recent large-scale study on GANs and VAEs was presented in [18]. The authors consider several loss functions and regularizers, and study the effect of the loss function on the FID score, with low-to-medium complexity data sets (MNIST, CIFAR10, CELEBA), and a single (InfoGAN style) architecture. In this limited setting, the authors found that there is no statistically significant difference between the recently introduced models, versus the non-saturating GAN originally proposed in [9]. A study of the effects of gradient-norm regularization in GANs was recently presented in [8]. The authors posit that the gradient penalty can also be applied to the non-saturating GAN, and that, to a limited extent, it reduces the sensitivity to hyperparameter selection. In a recent work which introduced Spectral normalization [20] the authors perform a small study of the competing regularization and normalization approaches. We are happy to report that we could reproduce all but one result. Namely, to get the FID score of 21 on CIFAR10 it was necessary to use *both spectral normalization and gradient penalty*. Plots with reproduced results can be found in the Appendix.

Inspired by these works and building on the available open-source code from [18], we take one additional step in all dimensions considered therein: more complex neural architectures, more complex data sets, and more involved regularization and normalization schemes.

6 Conclusion

In this work we study the GAN landscape: losses, regularization and normalization schemes, and neural architectures, and their impact on the on the quality of generated samples which we assess by recently introduced quantitative metrics. Our fair and thorough empirical evaluation suggests that one should consider non-saturating GAN loss and spectral normalization as default choices when applying GANs to a new data set. Given additional computational budget, we suggest adding the gradient penalty from [10] and train the model until convergence. Furthermore, additional marginal gains can be obtained by combining normalization and regularization empirically confirming the importance of the Lipschitz constant of the discriminator [2, 10, 20]. Furthermore, both types of architectures proposed up-to this point perform reasonably well. A separate ablation study uncovered that most of the tricks applied in the ResNet style architectures lead to marginal changes in the quality and should be avoided due to the high computational cost. As a result of this large-scale study we identify the commong pitfalls standing in the way of accurate and fair comparison and propose concrete actions to demistify the future results – issues with metrics, data set preprocessing, non-determinisim, and missing implementation details are particularly striking. We hope that this work, together with the open-sourced reference implementaions and trained models, will serve as a solid baseline for future GAN research.

References

[1] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Generative adversarial networks for extreme learned image compression. *CoRR*, abs/1804.02958, 2018.

[2] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2017.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations (ICLR)*, 2018.

[5] Ali Borji. Pros and cons of GAN evaluation measures. *arXiv preprint arXiv:1802.03446*, 2018.

[6] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[7] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[8] William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M Dai, Shakir Mohamed, and Ian Goodfellow. Many paths to equilibrium: Gans do not need to decrease adivergence at every step. In *International Conference on Learning Representations (ICLR)*, 2018.

[9] 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 (NIPS)*, 2014.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a Nash equilibrium. *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *International Conference on Learning Representations (ICLR)*, 2018.

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[17] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of GANs. *arXiv preprint arXiv:1705.07215*, 2017.

[18] 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.

[19] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016.

[20] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *International Conference on Learning Representations (ICLR)*, 2018.

[21] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[22] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *International Conference on Machine Learning (ICML)*, 2017.

[23] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *International Conference on Learning Representations (ICLR)*, 2016.

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

[25] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[26] D. Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. Winner's curse? On pace, progress, and empirical rigor, 2018.

[27] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning (ICML)*, 2010.

[28] Michael Tschannen, Eirikur Agustsson, and Mario Lucic. Deep generative models for distribution-preserving lossy compression. *arXiv preprint arXiv:1805.11057*, 2018.

[29] Thomas Unterthiner, Bernhard Nessler, Calvin Seward, Günter Klambauer, Martin Heusel, Hubert Ramsauer, and Sepp Hochreiter. Coulomb GANs: Provably optimal nash equilibria via potential fields. In *International Conference on Learning Representations (ICLR)*, 2018.

[30] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems and Computers*, 2003.

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

[32] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

A FID and Inception scores on CIFAR10

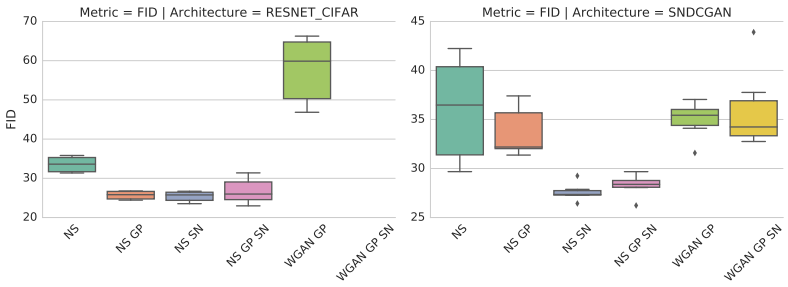


Figure 5: An empirical study with SNDCGAN and RESNET_CIFAR architectures on CIFAR10. We recover the state of the art results recently presented in [20]. We note that we could obtain FID of 22 only by combining the gradient penalty and spectral normalization.

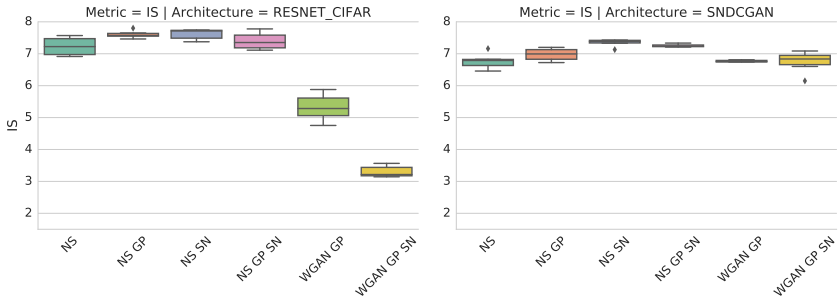


Figure 6: We show the Inception Score for each model within our study which corresponds to recently reported results [20].

B Architectures

B.1 SNDCGAN

We used the same architecture as [20], with the parameters copied from the GitHub page[6]. In Table 2a and Table 2b, we describe the operations in layer column with order. Kernel size is described in format $[filter\_h, filter\_w, stride]$, input shape is $h \times w$ and output shape is $h \times w \times channels$. The slopes of all lReLU functions are set to 0.1. The input shape $h \times w$ is 128×128 for CELEBA-HQ-128 and LSUN-BEDROOM, 32×32 for CIFAR10.

B.2 ResNet Architecture

We described ResNet19 in Table 3. RS column stands for the resample of the residual block, with downscale(D)/upscale(U)/none(-) setting. MP stands for mean pooling and BN for batch normalization. ResBlock is defined in Table 4. The addition layer merges two paths by adding them. The first path is a shortcut layer with exactly one convolution operation, while the second path consists of two convolution operations. The downscale layer and upscale layer are marked in Table 4. We used average pool with kernel $[2, 2, 2]$ for downscale, after the convolution operation. We used unpool from https://github.com/tensorflow/tensorflow/issues/2169 for upscale, before conv operation. h and w are the input shape to the ResNet block, output shape depends on the RS parameter. c_{in} and c_{out} are the input channels and output channels for a ResNet block. Table 5 described the ResNet cifar architecture we used in Figure 6 for reproducing the existing results. Note that RS is set to none for third ResBlock and fourth ResBlock in discriminator. In this case, we used the same ResNet block defined in Table 4 without resampling.

[6]https://github.com/pfnet-research/chainer-gan-lib

Table 2: SNDCGAN architecture.

(a) SNDCGAN discriminator

LAYER	KERNEL	OUTPUT SHAPE
Conv, lReLU	$[3, 3, 1]$	$h \times w \times 64$
Conv, lReLU	$[4, 4, 2]$	$h/2 \times w/2 \times 128$
Conv, lReLU	$[3, 3, 1]$	$h/2 \times w/2 \times 128$
Conv, lReLU	$[4, 4, 2]$	$h/4 \times w/4 \times 256$
Conv, lReLU	$[3, 3, 1]$	$h/4 \times w/4 \times 256$
Conv, lReLU	$[4, 4, 2]$	$h/8 \times w/8 \times 512$
Conv, lReLU	$[3, 3, 1]$	$h/8 \times w/8 \times 512$
Linear	-	1

(b) SNDCGAN generator

LAYER	KERNEL	OUTPUT SHAPE
z	-	128
Linear, BN, ReLU	-	$h/8 \times w/8 \times 512$
Deconv, BN, ReLU	$[4, 4, 2]$	$h/4 \times w/4 \times 256$
Deconv, BN, ReLU	$[4, 4, 2]$	$h/2 \times w/2 \times 128$
Deconv, BN, ReLU	$[4, 4, 2]$	$h \times w \times 64$
Deconv, Tanh	$[3, 3, 1]$	$h \times w \times 3$

Table 3: ResNet 19 architecture corresponding to "resnet_small" in https://github.com/pfnet-research/sngan_projection

(a) ResNet19 discriminator

LAYER	KERNEL	RS	OUTPUT SHAPE
ResBlock	$[3, 3, 1]$	D	$64 \times 64 \times 64$
ResBlock	$[3, 3, 1]$	D	$32 \times 32 \times 128$
ResBlock	$[3, 3, 1]$	D	$16 \times 16 \times 256$
ResBlock	$[3, 3, 1]$	D	$8 \times 8 \times 256$
ResBlock	$[3, 3, 1]$	D	$4 \times 4 \times 512$
ResBlock	$[3, 3, 1]$	D	$2 \times 2 \times 512$
ReLU, MP	-	-	512
Linear	-	-	1

(b) ResNet19 generator

LAYER	KERNEL	RS	OUTPUT SHAPE
z	-	-	128
Linear	-	-	$4 \times 4 \times 512$
ResBlock	$[3, 3, 1]$	U	$8 \times 8 \times 512$
ResBlock	$[3, 3, 1]$	U	$16 \times 16 \times 256$
ResBlock	$[3, 3, 1]$	U	$32 \times 32 \times 256$
ResBlock	$[3, 3, 1]$	U	$64 \times 64 \times 128$
ResBlock	$[3, 3, 1]$	U	$128 \times 128 \times 64$
BN, ReLU	-	-	$128 \times 128 \times 64$
Conv	$[3, 3, 1]$	-	$128 \times 128 \times 3$
Sigmoid	-	-	$128 \times 128 \times 3$

Table 4: ResNet block definition.

(a) ResBlock discriminator

LAYER	KERNEL	RS	OUTPUT SHAPE
Shortcut	$[3, 3, 1]$	D	$h/2 \times w/2 \times c_{out}$
BN, ReLU	-	-	$h \times w \times c_{in}$
Conv	$[3, 3, 1]$	-	$h \times w \times c_{out}$
BN, ReLU	-	-	$h \times w \times c_{out}$
Conv	$[3, 3, 1]$	D	$h/2 \times w/2 \times c_{out}$
Addition	-	-	$h/2 \times w/2 \times c_{out}$

(b) ResBlock generator

LAYER	KERNEL	RS	OUTPUT SHAPE
Shortcut	$[3, 3, 1]$	U	$2h \times 2w \times c_{out}$
BN, ReLU	-	-	$h \times w \times c_{in}$
Conv	$[3, 3, 1]$	U	$2h \times 2w \times c_{out}$
BN, ReLU	-	-	$2h \times 2w \times c_{out}$
Conv	$[3, 3, 1]$	-	$2h \times 2w \times c_{out}$
Addition	-	-	$2h \times 2w \times c_{out}$

C Recommended hyperparameter settings

To make the future GAN training simpler, we propose a set of best parameters for three setups: (1) Best parameters without any regularizer. (2) Best parameters with only one regularizer. (3) Best parameters with at most two regularizers. Table 6, Table 7 and Table 8 summarize the top 2 parameters for SNDCGAN architecture, ResNet19 architecture and ResNet cifar architecture, respectively. Models are ranked according to the median FID score of five different random seeds. Note that ranking models according to the best FID score of different

Table 5: ResNet cifar architecture.

(a) ResNet cifar discriminator

LAYER	KERNEL	RS	OUTPUT SHAPE
ResBlock	$[3, 3, 1]$	D	$16 \times 16 \times 128$
ResBlock	$[3, 3, 1]$	D	$8 \times 8 \times 128$
ResBlock	$[3, 3, 1]$	-	$8 \times 8 \times 128$
ResBlock	$[3, 3, 1]$	-	$8 \times 8 \times 128$
ReLU, MP	-	-	128
Linear	-	-	1

(b) ResNet cifar generator

LAYER	KERNEL	RS	OUTPUT SHAPE
z	-	-	128
Linear	-	-	$4 \times 4 \times 256$
ResBlock	$[3, 3, 1]$	U	$8 \times 8 \times 256$
ResBlock	$[3, 3, 1]$	U	$16 \times 16 \times 256$
ResBlock	$[3, 3, 1]$	U	$32 \times 32 \times 256$
BN, ReLU	-	-	$32 \times 32 \times 256$
Conv	$[3, 3, 1]$	-	$32 \times 32 \times 3$
Sigmoid	-	-	$32 \times 32 \times 3$

seeds will achieve better but unstable result. For ResNet19 architecture with at most two regularizers, we have run it only once due to computational overhead. To show the model stability, we listed the best FID score out of five seeds from the same parameters in column best. Spectral normalization is clearly outperforms the other normalizers on SNDCGAN and ResNet cifar architectures, while on ResNet19 both layer normalization and spectral normalization work well.

To visualize the FID score on each data set, Figure 7, Figure 8 and Figure 9 show the generated examples by GANs. We select the examples from the best FID run, and then increase the FID score for two more plots.

Table 6: SNDCGAN parameters

DATA SET	MEDIAN	BEST	LR($\times 10^{-3}$)	β_1	β_2	n_{disc}	λ	NORM
CIFAR10	29.75	28.66	0.100	0.500	0.999	1	-	-
CIFAR10	36.12	33.23	0.200	0.500	0.999	1	-	-
CELEBA-HQ-128	66.42	63.13	0.100	0.500	0.999	1	-	-
CELEBA-HQ-128	67.39	64.59	0.200	0.500	0.999	1	-	-
LSUN-BEDROOM	180.36	160.12	0.200	0.500	0.999	1	-	-
LSUN-BEDROOM	188.99	162.00	0.100	0.500	0.999	1	-	-
CIFAR10	26.66	25.27	0.200	0.500	0.999	1	-	SN
CIFAR10	27.32	26.97	0.100	0.500	0.999	1	-	SN
CELEBA-HQ-128	31.14	29.05	0.200	0.500	0.999	1	-	SN
CELEBA-HQ-128	33.52	31.92	0.100	0.500	0.999	1	-	SN
LSUN-BEDROOM	63.46	58.13	0.200	0.500	0.999	1	-	SN
LSUN-BEDROOM	74.66	59.94	1.000	0.500	0.999	1	-	SN
CIFAR10	26.23	26.01	0.200	0.500	0.999	1	1	SN+GP
CIFAR10	26.66	25.27	0.200	0.500	0.999	1	-	SN
CELEBA-HQ-128	31.13	30.80	0.100	0.500	0.999	1	10	GP
CELEBA-HQ-128	31.14	29.05	0.200	0.500	0.999	1	-	SN
LSUN-BEDROOM	63.46	58.13	0.200	0.500	0.999	1	-	SN
LSUN-BEDROOM	66.58	65.75	0.200	0.500	0.999	1	10	GP

D Which parameters really matter?

For each architecture and hyper-parameter we estimate its impact on the final FID. Figure 10 presents heatmaps for hyperparameters, namely the learning rate, β_1, β_2, n_{disc}, and λ for each combination of neural architecture and data set.

Table 7: ResNet19 parameters

DATA SET	MEDIAN	BEST	LR($\times 10^{-3}$)	β_1	β_2	n_{disc}	λ	NORM
CELEBA-HQ-128	43.73	39.10	0.100	0.500	0.999	5	-	-
CELEBA-HQ-128	43.77	39.60	0.100	0.500	0.999	1	-	-
LSUN-BEDROOM	160.97	119.58	0.100	0.500	0.900	5	-	-
LSUN-BEDROOM	161.70	125.55	0.100	0.500	0.900	5	-	-
CELEBA-HQ-128	32.46	28.52	0.100	0.500	0.999	1	-	LN
CELEBA-HQ-128	40.58	36.37	0.200	0.500	0.900	1	-	LN
LSUN-BEDROOM	70.30	48.88	1.000	0.500	0.999	1	-	SN
LSUN-BEDROOM	73.84	60.54	0.100	0.500	0.900	5	-	SN
CELEBA-HQ-128	29.13	-	0.100	0.500	0.900	5	1	LN+DR
CELEBA-HQ-128	29.65	-	0.200	0.500	0.900	5	1	GP
LSUN-BEDROOM	55.72	-	0.200	0.500	0.900	5	1	LN+GP
LSUN-BEDROOM	57.81	-	0.100	0.500	0.999	1	10	SN+GP

Table 8: ResNet cifar parameters

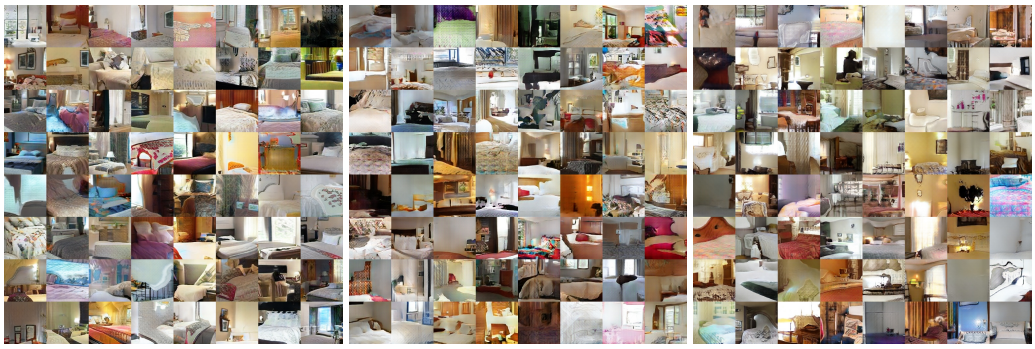
DATA SET	MEDIAN	BEST	LR($\times 10^{-3}$)	β_1	β_2	n_{disc}	λ	NORM
CIFAR10	31.40	28.12	0.200	0.500	0.999	5	-	-
CIFAR10	33.79	30.08	0.100	0.500	0.999	5	-	-
CIFAR10	23.57	22.91	0.200	0.500	0.999	5	-	SN
CIFAR10	25.50	24.21	0.100	0.500	0.999	5	-	SN
CIFAR10	22.98	22.73	0.200	0.500	0.999	1	1	SN+GP
CIFAR10	23.57	22.91	0.200	0.500	0.999	5	-	SN

E Variations of MS-SSIM

We used the MS-SSIM scorer from TensorFlow with default power_factors [30]. Note that the default filter size for each scale layer is 11, the minimum image edge is $11 \times 2^4 = 176$. To adapt it to CELEBA-HQ-128 data set with size 128×128, we used the minimum of filter size 11 and image size in last scale layer to allow the computation followed the previous work [8].

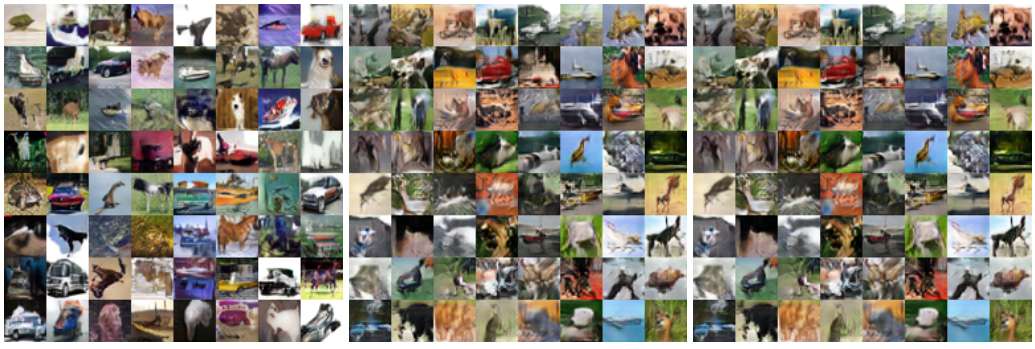
(a) FID = 24.7 (b) FID = 34.6 (c) FID = 45.2

Figure 7: Examples generated by GANs on CELEBA-HQ-128 data set.



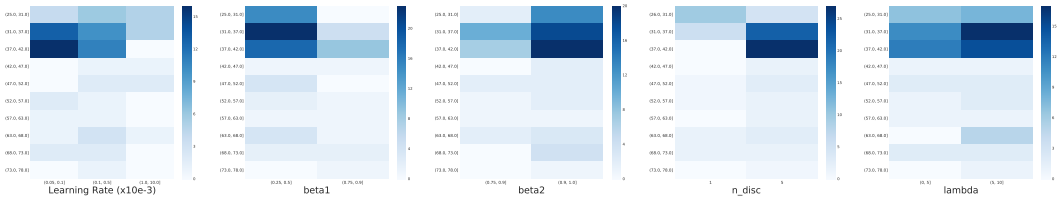
(a) FID = 40.4 (b) FID = 60.7 (c) FID = 80.2

Figure 8: Examples generated by GANs on LSUN-BEDROOM data set.

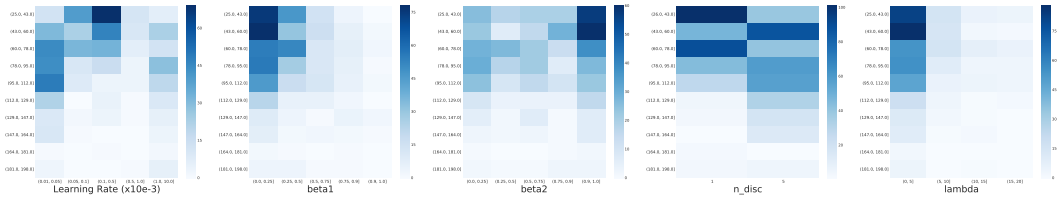


(a) FID = 22.7 (b) FID = 33.0 (c) FID = 42.6

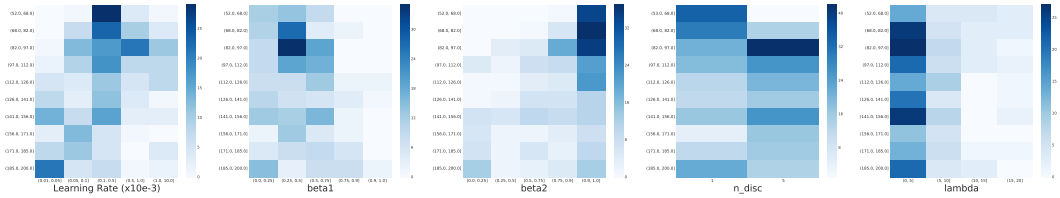
Figure 9: Examples generated by GANs on CIFAR10 data set.



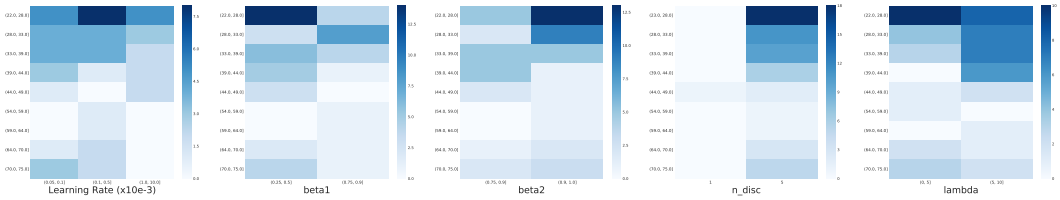
(a) SNDCGAN on CIFAR10



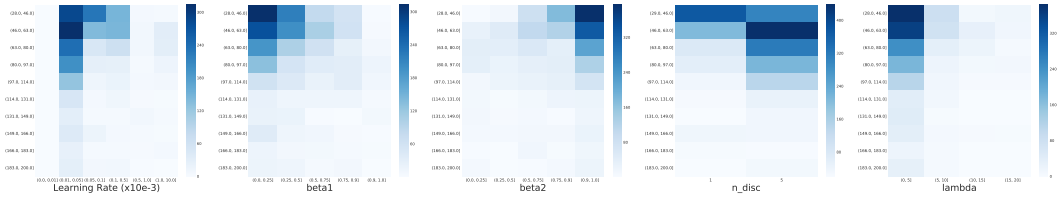
(b) SNDCGAN on CELEBA-HQ-128



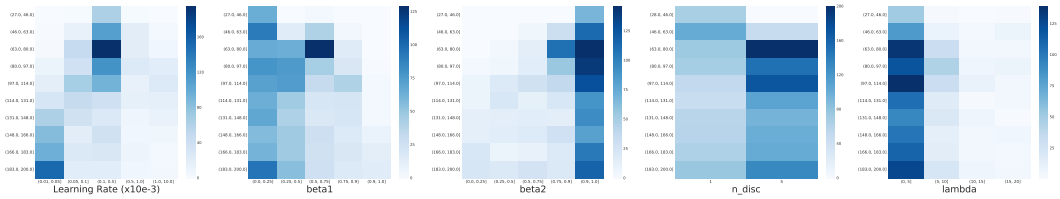
(c) SNDCGAN on LSUN-BEDROOM



(d) Resnet cifar on CIFAR10



(e) ResNet19 on CELEBA-HQ-128



(f) ResNet19 on LSUN-BEDROOM

Figure 10: Heat plots for hyper-parameters on each architecture and dataset combination.