REPRODUCIBILITY REPORT FOR RELATIVISTIC DISCRIMINATOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Adversarial Network (GAN) is one of the most widely used generative models. Despite its impressive performance in generating human pleasant images, training GAN is still a challenge problem. Relativistic discriminator (RGAN) is proposed to improve training stability and visual quality of wide variety of GANs. In this paper, we conduct a series of experiments, aiming to reproduce the results of the RGAN. We compare frechet inception distance (FID) and inception score (IS) among 10 different loss functions via 2 GAN architectures and 3 different normalization methods. Our tasks include generating 32x32 natural images and 64x64 human faces. The experimental results show that RGAN is able to stabilize training process and improve generated image quality in 15 out of 20 cases, and fail to generate reasonable images (unconverged) in 2 cases. While the relativistic average GAN (RaGAN) can improve the quality in 9 out of 20 cases and fail to generate reasonable images in 4 cases. Our code is available at `https://github.com/LoSealL/VideoSuperResolution`.

1 INTRODUCTION

Generative adversarial network (GAN) has been widely studied and significantly improved in the past few years. Karras et al. (2018) proposed a progressive growing GAN that can generate 1024x1024 photo-realistic human faces, and Brock et al. (2019) successfully generated 512x512 realistic natural images. Though GAN performs impressively in image generation, the training of GAN is still a challenging task (Fedus et al., 2018; Mescheder et al., 2018). GAN was originally proposed by Goodfellow et al. (2014) – as they claimed in Fedus et al. (2018) – one is minimax-GAN (MGAN) which optimizes the minimax loss and the other one is non-saturate GAN (NSGAN) which optimizes non-saturating loss. We follow their terms so we use **NSGAN** to act as the "original GAN".

Recently, many techniques have been proposed to address the training issue. Mao et al. (2017) proposed least-square loss which optimizes the Pearson χ^2 divergence. Arjovsky & Bottou (2017) proposed wasserstein loss which optimizes wasserstein distance and constrains the Lipschitz constant of the discriminator by weight clipping. Gulrajani et al. (2017) replaced weight clipping by gradient penalty to improve the quality of generated images. Miyato et al. (2018) proposed spectral normalization (SN) to constraint the Lipschitz constant. Gradient penalty (GP) can also apply to non-IPM based GAN (such as NSGAN) (Kurach et al., 2018; Mescheder et al., 2018). In this paper we distinguish different models via their loss functions, which are referred as **NSGAN**, **LSGAN**, **WGAN**, **WGAN-GP**, **NSGAN-GP** respectively.

Jolicoeur-Martineau (2019) proposed a novel loss function to stabilize GAN training and improve image quality, which is referred as relativistic GAN (RGAN) and relativistic average GAN (RaGAN). Jolicoeur-Martineau (2019) believes that the key missing property of GAN is that the probability of real data being real should decrease as the probability of fake data being real increase. The loss function of RGAN is then defined as follows:

$$L_D = \mathbb{E}_{x_r \sim \mathbb{P}, x_f \sim \mathbb{Q}}[f_1(C(x_r) - C(x_f))] + \mathbb{E}_{x_r \sim \mathbb{P}, x_f \sim \mathbb{Q}}[f_2(C(x_f) - C(x_r))] \tag{1}$$

$$L_G = \mathbb{E}_{x_r \sim \mathbb{P}, x_f \sim \mathbb{Q}}[g_1(C(x_r) - C(x_f))] + \mathbb{E}_{x_r \sim \mathbb{P}, x_f \sim \mathbb{Q}}[g_2(C(x_f) - C(x_r))] \tag{2}$$

Where $\mathbb{P}, \mathbb{Q}$ is the distribution of real and fake data respectively, $C(x)$ is the critic values of x, and f, g are scalar-to-scalar functions. And the RaGAN's loss function is defined as:

$$L_D = \mathbb{E}_{x_r \sim \mathbb{P}}[f_1(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}}[C(x_f)])] + \mathbb{E}_{x_f \sim \mathbb{Q}}[f_2(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}}[C(x_r)])] \quad (3)$$

$$L_G = \mathbb{E}_{x_r \sim \mathbb{P}}[g_1(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}}[C(x_f)])] + \mathbb{E}_{x_f \sim \mathbb{Q}}[g_2(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}}[C(x_r)])] \quad (4)$$

In this paper, our goal is to reproduce the experiment of RGAN to verify and compare its performance with other techniques. This is part of the *ICLR 2019 Reproducibility Challenge*.

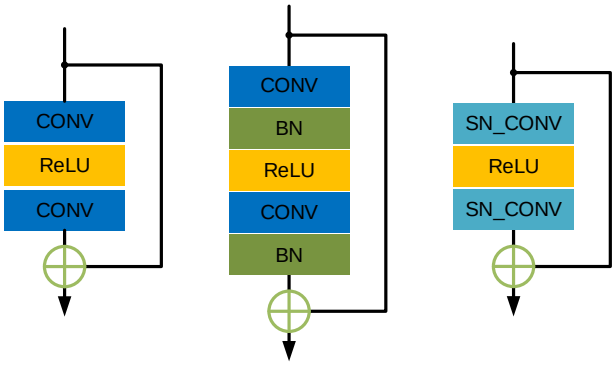
2 EXPERIMENTS

Jolicoeur-Martineau (2019) provides the pytorch implementation of RGAN on Github[1]. We re-implement the RGAN and reference works by tensorflow and provide on Github[2].

We enumerate 3 forms of relativistic GAN and 3 forms of relativistic average GAN, specifically, we use term **RGAN** to represent *relativistic NSGAN*, **RaGAN** to represent *relativistic average NSGAN*; **R(a)LSGAN** to represent *relativistic (average) LSGAN* and **R(a)GAN-GP** to represent *relativistic (average) NSGAN-GP*.

2.1 GAN ARCHITECTURE

In the original experiments, Jolicoeur-Martineau (2019) used a 4-layer (Radford et al., 2016) to generate 32x32 images, and a 5-layer DCGAN without bias after convolution to generate 64x64 images. In our reproduced experiments, we use the same DCGAN architecture to generate 32x32 images, and we also use a Resnet architecture to generate 32x32 and 64x64 images. The implementation of the residual block is shown in figure 1, and the architecture details is listed in table 1.



(a) Resblock (b) Resblock with BN (c) Resblock with SN

Figure 1: Resblock architecture with (a) no normalization; (b) batch normalization; (c) spectral normalization.

2.2 DATA SETS

It's very important to know how data is processed and feed into networks, because different processing may lead to very different results. We use CIFAR10 (Krizhevsky & Hinton, 2009) and CelebA (Liu et al., 2015) dataset for image generation. CIFAR10 dataset contains 50,000 training data, 10,000 testing data with 10 classes. CelebA dataset contains 202,599 human faces with various age, race and gender. For CelebA dataset, we first center crop all images into 64x64 patches and randomly select 10,000 images for testing and use the rest for training. As for CIFAR10 dataset, we directly load the training data and testing data via Keras API, class information is not used. In each iteration, a batch of images is randomly selected from training data, and the training data is randomly shuffled after a fixed number of steps (we shuffle every 200 steps in all the experiments).

[1]Available at `https://github.com/AlexiaJM/RelativisticGAN`
[2]Available at `https://github.com/LoSealL/VideoSuperResolution`

Table 1: *FC 4096* is short for full connected layer with 4096 units, *Conv f32k3s1* is short for 2D convolution layer with 32 filters, kernel size 3x3 and strides 1x1. *Deconv* is the transposed convolution. *BN* is short for batch normalization, *ReLU* is short for rectifier linear unit and *LReLU* is leaky ReLU. *Norm* can be either batch normalization (BN), spectral normalization (SN) or no normalization (None). *Up* is the nearest upsampling before Resblock. *AvgPooling* is the average-pooling to downsample features. The colored emphasized layers in ResNet are inserted for generating 64x64 images and are disabled for 32x32 images.

DCGAN 32x32		**ResNet**	
Generator	Discriminator	Generator	Discriminator
$z \in \mathbb{R}^{128} \sim N(0, I)$	$x \in \mathbb{R}^{32x32x3}$	$z \in \mathbb{R}^{128} \sim N(0, I)$	$x \in \mathbb{R}^{HxWx3}$
FC 8192	Conv f64k3s1	FC 4096	Resblock f64k3
BN, ReLU	Norm, LReLU	Up, Resblock f256k3	AvgPooling s2
Deconv f256k4s2	Conv f128k4s2	Up, Resblock f256k3	Resblock f128k3
BN, ReLU	Norm, LReLU	Up, Resblock f256k3	AvgPooling s2
Deconv f128k4s2	Conv f128k3s1	*Up, Resblock f256k3*	Resblock f256k3
BN, ReLU	Norm, LReLU	BN, ReLU	AvgPooling s2
Deconv f64k4s2	Conv f256k4s2	Conv f3k3s1	*Resblock f256k3*
BN, ReLU	Norm, LReLU	tanh	*AvgPooling s2*
Conv f3k3s1	Conv f256k3s1		ReLU
tanh	Norm, LReLU		FC 1
	Conv f512k4s2		
	Norm, LReLU		
	Conv f512k3s1		
	Norm, LReLU		
	FC 1		

2.3 Evaluation Metrics

We evaluate benchmarks by *Frechet Inception Distance* (FID, Heusel et al. (2017)) and *Inception Score* (IS, Salimans et al. (2016)). To perform a fairly comparison, we follow suggestions by Fedus et al. (2018). In detail, the inception score is obtained from 10,000 generated images, and FID is calculated between 10,000 test images and 10,000 generated images. The test images are directly provided by CIFAR10 dataset, while are randomly selected from CelebA as we described in section 2.2. The inception model we use is *Inception_v1* and the batch size is 50 when executing inception model.

2.4 Training Details

We perform 3 different normalization with 10 candidate loss functions on DCGAN and ResNet architecture. The batch size for training is fixed to 64, and the learning rate is fixed to 2.0×10^{-4} during training. We apply the Adam optimizer (Kingma & Ba, 2015) for all the experiments and the momentum of Adam optimizer is $(0.5, 0.999)$. For all experiments except those on WGAN and WGAN-GP, we update discriminator 1 step for every generator update, while for WGAN and WGAN-GP we update discriminator 5 steps for every generator update. We first train DCGAN and ResNet with CIFAR10 dataset, and end up at 100k iterations. Then we train ResNet with CelebA human faces (center cropped to 64x64) and end up at 40k iterations. The iteration is counted on the **discriminator** updates. Table 2 illustrates inception score and frechet inception distance for all the training results from CIFAR10, and table 3 shows the results from CelebA.

We also show FID and inception score of 32x32 generated images over different iterations in the appendix A. More samples are plotted in appendix B.

2.5 Analysis

Our results indicate that RGAN performs very well compared to its counterpart, the only negative result is RGAN-GP in generating 64x64 faces, which is not converged. RaGAN has an equal

Table 2: Benchmarks on CIFAR10. IS (1st value) and FID (2nd value) are reported. The best result of each column is bolded and the second best result is underlined. Short dash "-" means we don't train on that configuration because of limited computational resources.

	DCGAN 32x32			**Resnet 32x32**		
batch=64, lr=2e-4, Adam=(0.5, 0.999), 100k iter						
IS/FID	**None**	**BN**	**SN**	**None**	**BN**	**SN**
NSGAN	2.64/205.3	6.33/54.42	6.48/40.31	3.24/161.3	7.57/29.80	6.38/37.08
RGAN	6.99/36.91	6.01/58.52	7.11/32.55	6.72/**31.77**	7.05/40.05	6.92/33.37
RaGAN	7.47/31.46	6.89/60.74	6.70/52.16	6.42/41.77	7.53/31.96	6.21/47.34
LSGAN	2.47/316.2	1.49/272.6	6.06/52.05	2.60/224.3	7.50/27.52	6.53/39.22
RLSGAN	3.24/156.6	6.60/**44.22**	4.75/67.92	5.49/51.51	7.63/**26.57**	6.96/33.14
RaLSGAN	2.65/182.0	**7.34**/51.61	4.71/91.97	5.82/57.73	**7.87**/31.65	6.83/35.03
WGAN-GP	6.96/34.59	-	6.74/36.34	4.39/72.37	-	1.30/330.4
NSGAN-GP	7.07/31.51	-	7.40/28.47	6.64/35.14	-	6.81/33.51
RGAN-GP	**7.50/25.63**	-	**7.62/23.97**	**6.89**/33.92	-	**7.18/29.85**
RaGAN-GP	7.44/25.95	-	7.47/24.69	2.38/214.8	-	7.04/34.83

Table 3: Benchmarks on CelebA 64x64. We only report FID metric for CelebA dataset because inception score is not suitable for human faces.

ResNet 64x64: batch=64, lr=2e-4, Adam=(0.5, 0.999), 40k iter

FID	**NSGAN**	**RGAN**	**RaGAN**	**NSGAN-GP**	**RGAN-GP**	**RaGAN-GP**
None	30.46	22.46	17.17	**13.00**	246.8	13.63
SN	**22.73**	23.01	252.4	290.1	32.08	312.4

performance compared to RGAN in positive results, but RaGAN has more negative results. Using relativistic loss function does help to stabilize training and greatly improve the generated image quality. When apply Lipschitz constraint to discriminator (spectral normalization or gradient penalty), the training is more stable, and relativistic loss functions can still help to improve the quality. If we change to a deeper architecture (ResNet), the conclusion still holds.

It's recommended to combine RGAN together with spectral normalization and the gradient penalty. However, it's not very clear whether averaged relativistic GAN is really help to its counterpart and the meaning of averaging critic values in equation 3 and 4 is not well explained. It can be seen that RaGAN+BN/SN improves little, and RaGAN can't even converge in 4 cases (red highlighted in table 2 and 3).

3 Conclusion

In this work we compare relativistic GAN and relativistic average GAN to their counter-parts in totally 20 cases. We found in most cases, relativistic GAN is an effective method to stabilize training process and can also improve image quality, while relativistic average GAN is not very effective and sometimes corrupt training in our experiments. Apply relativistic GAN with spectral normalization on discriminator is strongly recommended. Apply gradient penalty to discriminator loss function can further improves the image quality at the cost of higher computation. We hope this work can help when training GANs.

References

Martin Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. In *ICLR*, 2017.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *ICLR*, 2019.

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 *ICLR*, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *NIPS*, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved Training of Wasserstein GANs. In *NIPS*, 2017.

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 *NIPS*, 2017.

Alexia Jolicoeur-Martineau. The relativistic discriminator : a key element missing from standard GAN. In *ICLR*, 2019.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *ICLR*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The GAN Landscape: Losses, Architectures, Regularization, and Normalization, *arXiv:1807.04720v1*. 2018.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale CelebFaces Attributes (CelebA) Dataset. In *ICCV*, 2015.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least Squares Generative Adversarial Networks. In *ICCV*, 2017.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which Training Methods for GANs do actually Converge? In *ICML*, 2018.

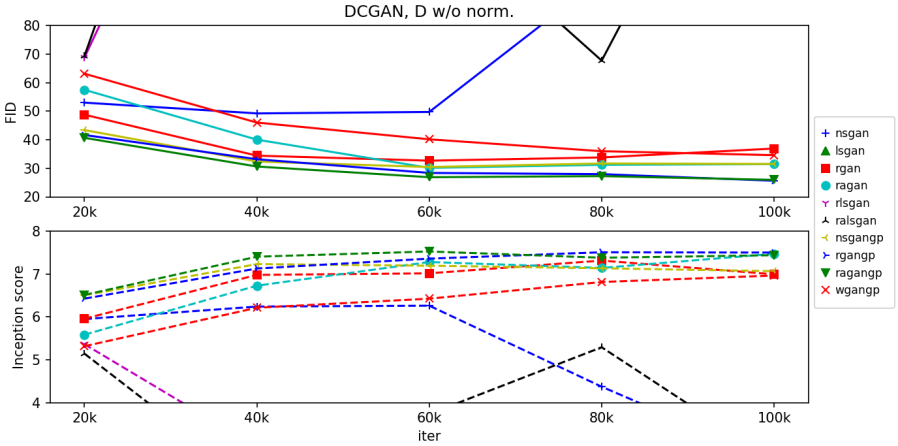
Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. In *ICLR*, 2018.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *ICLR*, 2016.

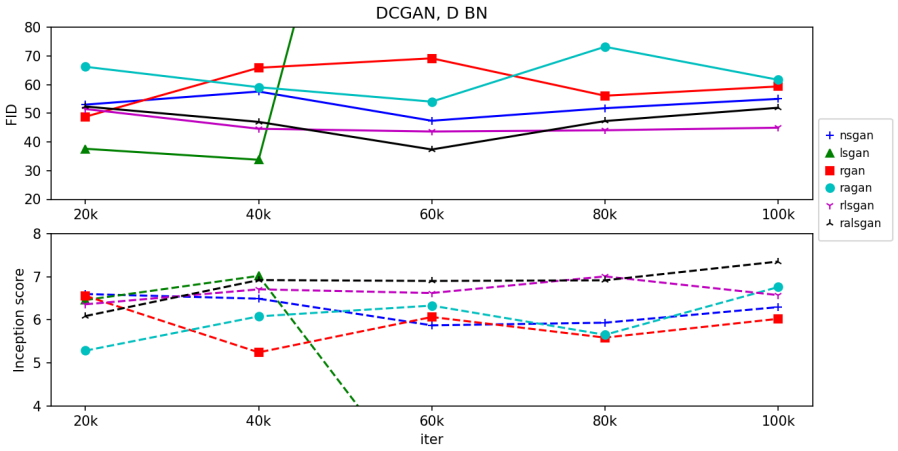
Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In *NIPS*, 2016.

APPENDICES

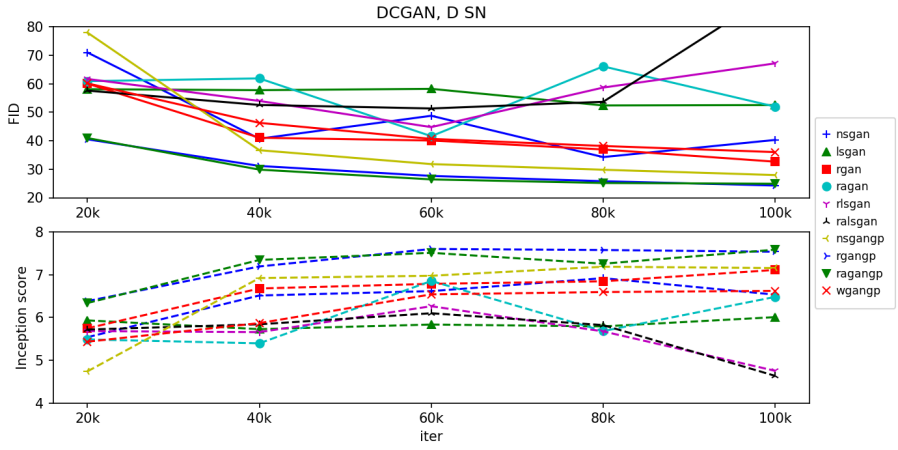
A FID AND INCEPTION SCORE



(a) Discriminator without normalization

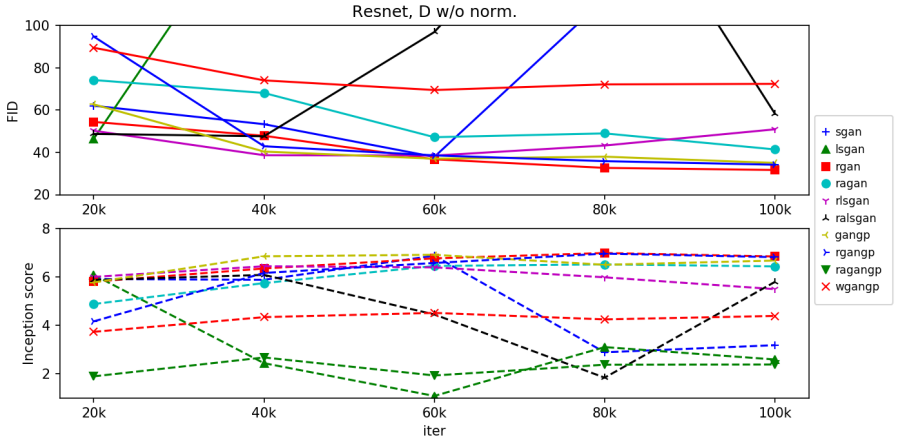


(b) Discriminator with batch normalization

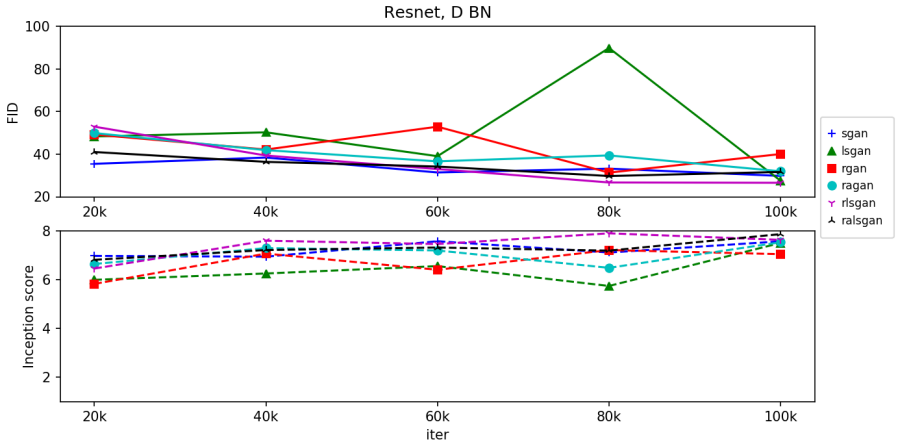


(c) Discriminator with spectral normalization

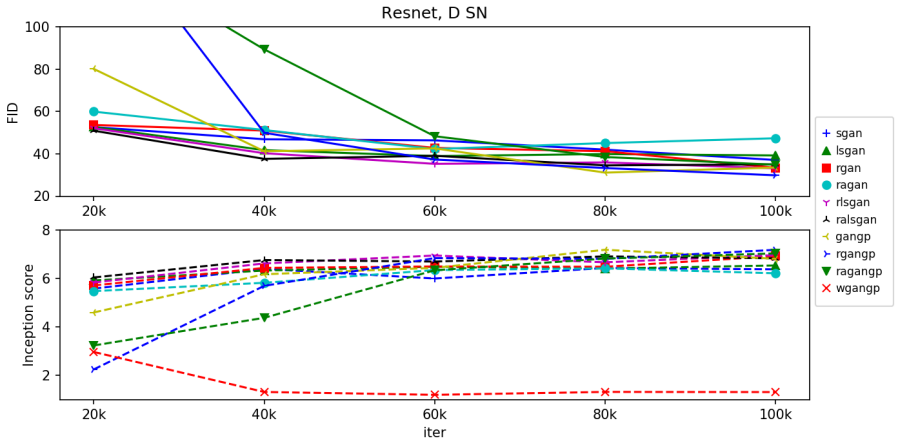
Figure 2: FID and inception score of DCGAN 32x32 over iterations.



(a) Discriminator without normalization



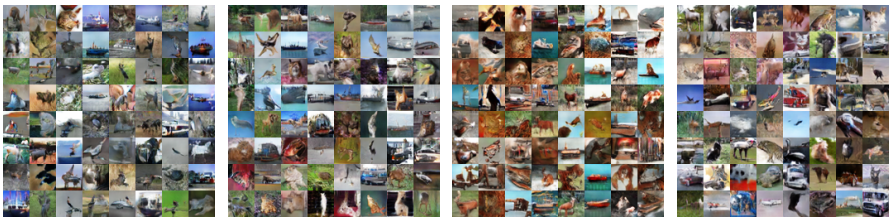
(b) Discriminator with batch normalization



(c) Discriminator with spectral normalization

Figure 3: FID and inception score of Resnet 32x32 over iterations.

B GENERATED IMAGE SAMPLES



(a) NSGAN (b) RGAN (c) RaGAN (d) NSGAN-GP

Figure 4: Generated 32x32 images by DCGAN. D with spectral normalization.

(a) NSGAN (b) RGAN (c) RaGAN (d) NSGAN-GP

Figure 5: Generated 32x32 images by Resnet. D with spectral normalization.

(a) NSGAN (b) RGAN

(c) RaGAN (d) NSGAN-GP

Figure 6: Generated 64x64 images by Resnet. D with spectral normalization.