Exploring Generative Adversarial Networks: Improving training techniques

Lekha Iyengar
li471@nyu.edu

Mudit Pandey
mp5099@nyu.edu

Riken Mehta
rpm390@nyu.edu

Abstract

The most problematic areas in training Generative Adversarial Networks[6] are mode collapse, low fidelity, failure to converge and non-scalability. We apply a few of the innovations and techniques presented in Large Scale GAN Training for High Fidelity Natural Image Synthesis[3] for BigGan[3] architecture for improving stability, scalability and robustness of Deep Convolutional Generative Adversarial Networks (DCGAN) [15]. Extensive experiments on ImageNet [5], CIFAR [9], CelebA[11] and CelebHQ[8] dataset demonstrate the effectiveness of these techniques and show promising results

1. Introduction

The High Fidelity Natural Image Synthesis [3] paper showed that Generative Adversarial Networks perform better as the model is scaled up and the number of parameters is increased. However, this scale up comes at huge computational cost, the 158 million parameter model was trained on 128 core Google TPU 3 pods. We implement the BigGan architecture and conduct some of the experiments listed in High Fidelity Natural Image Synthesis [3]. We also enhanced the DCGAN architecture by incorporating these changes and were able to see striking improvements over vanilla DCGAN implementation.

1.1. Related Works

1.1.1 Deep Convolutional Generative Adversarial Networks(DCGAN)

Deep Convolutional Generative Adversarial Networks(DCGAN)[15] is a simple Generative Adversarial Network architecture which uses convolutional stride and transposed convolution for downsampling and upsampling. It uses batch normalization, ReLU activation in generator and Leaky ReLU in discriminator.

1.1.2 Spectral Normalization

Spectral normalization[12] is a weight normalization technique used to stabilize the discriminator. It constrains the Lipschitz constant of the weights

$$W_{SN} = W/\sigma(W)$$

where σ(W) is largest singular value of W.
Normalizing the weights leads to the problem of slow training of discriminator. There are two major approaches to solve this problem, the first is to train discriminator few times before updating the generator and the second approach proposed in GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium [7] is to use different learning rates for the generator and discriminator. Stable training with 1:1 balanced updates is desirable for improving the convergence speed of the model. The Self-Attention Generative Adversarial Networks (SAGAN)[22] uses learning rate 0.0004 for the discriminator and the learning rate of 0.0001 for the generator. High Fidelity Natural Image Synthesis [3] uses both these methods to fix the slow learning of discriminator. Learning rate of 2.10^{-4} is used for discriminator and 5.10^{-5} for the generator. The discriminator is also trained for 2 steps for every generator step.

1.1.3 Hinge loss

Hinge loss[10][19] for adversarial networks is used as it has shown good results in the past for GAN training. The loss function is applied to both generator and discriminator as follows:

$$\begin{aligned} L_D = & -E_{(x,y)\sim p_{data}}[min(0, -1 + D(x,y))] \\ & - E_{z\ p_z, y\sim p_{data}}[min(0, -1 - D(G(z), y))] \end{aligned} \quad (1)$$

$$L_G = -E_{z\sim p_z, y\sim p_{data}}[D(G(z), y)] \quad (2)$$

1.1.4 Class conditional batch norm

Class-conditional batch norm [4] is used to provide class information to generator and to improve training. It does this by normalizing activations for each minibatch. Let $\gamma_{i,c}$ and $\beta_{i,c}$ be per channel trainable scalars which are a function of input x_i then

$$\gamma_{i,c} = f_c(x_i)$$

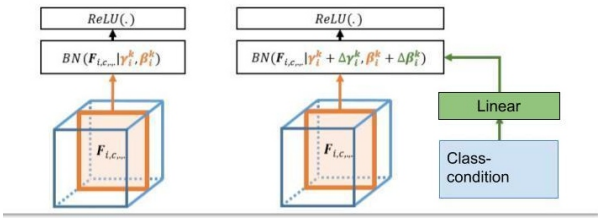


Figure 1: Computational Graph for batch norm(left) and class-conditional batch norm(right)

$$\beta_{i,c} = h_c(x_i)$$

here f_c and h_c are functions which can learn to control distribution of activations based on x_i

1.1.5 Self-attention

Self-attention[20][22] is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. A self-attention module computes the response at a position in a sequence by attending to all positions and taking their weighted average in an embedding space.The network exhibits a better balance between ability to model long-range dependencies and computational and statistical efficiency. The module is complementary to convolutions and helps with modeling long range, multi-level dependencies across image regions. Self-attention blocks achieve better results compared to residual blocks with the same number of parameters.

1.1.6 Orthogonal Initialization

Exact solutions to the nonlinear dynamics of learning in deep linear neural networks [17] shows that initializing the weights to a random orthogonal square matrix W such that $W^T W = I$ provide depth independent learning time and lead to faithful propagation of gradients in deep neural nets.

1.1.7 Hierarchical Latent Space

Latent space is the space where the extracted features of an image are projected. Images which are similar should lie close to each other in the latent space. Hierarchical latent spaces are used in High Fidelity Natural Image Synthesis [3] in which the noise vector z is split and fed to multiple layers (ResBlocks) of the generator rather than just the first layer (Figure 2). The latent vector is split into equal chunks along the channel dimension which are concatenated to the class embedding passed to a block. This allows the latent space to influence features and different resolution and levels of hierarchy. They also improve memory and compute

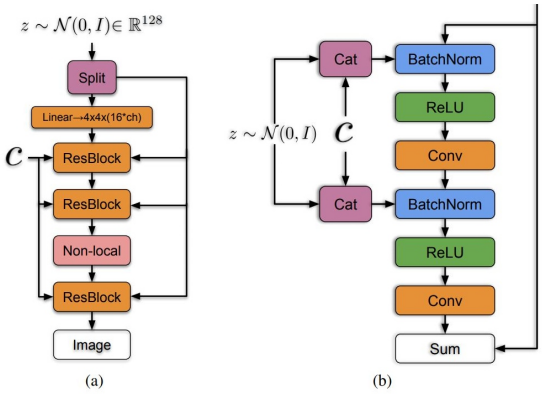


Figure 2: a) Architectural Layout for Generator in BigGan b) Residual Block in Generator

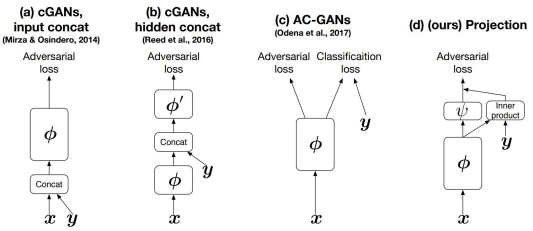


Figure 3: Discriminator model for Conditional GANS

cost by reducing the number of parameter in first linear layer.

1.1.8 Projection

Class information is provided to the discriminator using projections [13]. We take an inner product between the embedded condition vector y and the feature vector to improve the quality of class conditional image generation. The projection based discriminator is also better at reproducing diversity.

1.1.9 Linearly projected share embeddings

The embeddings C are shared across layers instead of having a separate layer for each embedding [14][22]. These embeddings are linearly projected to gains and bias of each Resnet Block in BigGan architecture proposed in High Fidelity Natural Image Synthesis [3]. This sharing reduces computation and memory costs and speeds up the training.

1.1.10 Truncation trick and Orthogonal Regularization

The truncation trick is one of the innovations used to boost Inception score [16][1] and Frchet Inception

$z \in \mathbb{R}^{120} \sim \mathcal{N}(0, I)$
dense, $4 \times 4 \times 16 \cdot ch$
ResBlock up $16 \cdot ch$
ResBlock up $8 \cdot ch$
ResBlock up $4 \cdot ch$
ResBlock up $2 \cdot ch$
Non-Local Block (64×64)
ResBlock up $1 \cdot ch$
BN, ReLU, 3×3 conv 3
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{128 \times 128 \times 3}$
ResBlock down $1 \cdot ch$
Non-Local Block (64×64)
ResBlock down $2 \cdot ch$
ResBlock down $4 \cdot ch$
ResBlock down $8 \cdot ch$
ResBlock down $16 \cdot ch$
ResBlock $16 \cdot ch$
ReLU
Global sum pooling
Embed(y)·$\boldsymbol{h}$ + (dense $\rightarrow$ 1)

(b) Discriminator

Figure 4: Architectures for BigGan for 128x128 resolution. ch represents the channel width multi- plier in each network.

Distance(FID)[7]. The noise vector z is truncated by resampling values with values above a chosen threshold. Reducing this threshold leads to an increase in Inception score and decrease in variety and consequently FID. Adding truncation to noise may produce saturation artifacts. To counteract the side effect of truncation, orthogonal regularization is used.

$$R_\beta(W) = \beta||W^T W \bigodot (1 - I)||_F^2$$

where 1 denotes a matrix with all elements set to 1. β value of 10^{-4} is used for BigGan architecture.

1.1.11 Mutual Information for Neural Estimation

Mode collapse in GANS can be palliated by regularizing generator loss with negative entropy of the sample. Mutual Information for Neural Estimation(MINE)[2] is an alternative approach to improve mode coverage of Generative Adversarial Networks which uses mutual information as an estimate of the negative entropy of generator loss. In addition to a generator and discriminator, an additional deep neural statistics network is used whose parameters are optimized for convergence.

2. Approach

2.1. Re-implementation of BigGan architecture

We implemented the BigGan architecture[3] and some of the incremental changes and innovations proposed in Large Scale GAN Training for High Fidelity Natural Image Synthesis[3]. We successfully implemented spectral normalization, hinge loss GAN training objective and imbalanced learning rate for generator and discriminator updates from the baseline Self-Attention GAN (SAGAN) paper which is the basis for High Fidelity Natural Image Synthesis. All convolution and linear layers in the discriminator and generator (except the non-local blocks) are followed by a spectral normalization layer. In addition we replicated key concepts like class conditional batch norm, hierarchical latent spaces for generators, projection for discriminators and linearly projected shared embeddings. For performance enhancement, we incorporated the truncation trick proposed in the paper.

While we could incorporate most of the proposed enhancements in our implementation, we could not implement orthogonal initialization and orthogonal regularization due to time constraints. The architecture proved to be quite expensive in terms of resource requirements, hence we could not experiment with larger batch size and channels. We used simple Batch Normalization instead of Cross Replica Batch Normalization in the generator. The reason for using cross replica batch normalization is to maintain batch normalization statistics across all devices. This seems to be crucial when training on large batch sizes as described in the paper[3]. All weights in both the discriminator and generator have default initialization (kaiming_uniform in PyTorch for both convolutional and linear layers) instead of orthogonal initialization. The training of the generative adversarial network also differs slightly. We train the discriminator once for every generator iteration as opposed to training the discriminator twice.

2.2. DCGAN

We re-implemented the DCGAN architecture[15]. Our implementation of DCGAN is generalized for images of power 2. We use this to generate 64 x 64 as well as 128 x 128 images. We also add spectral normalization layers after every convolution layers. When training on 128 x 128 images, we additionally add a dropout[18] layer with drop out rate of 0.5 after every convolution layers except the first and last layers of the discriminator. We also incorporated the truncation trick and observed improved sample quality.

3. Experiments

Figure 5: Generated images: BigGAN trained on lsun data

3.1. BigGAN

- Datasets - We test our implementation of BigGAN on 'church outdoors' and 'bridges' classes of the LSUN

dataset[1][21] and a 100 class subset of ImageNet[2][5].

- Pre-processing - Images were not pre-processed in any way except for normalization and resizing (if required)
- TTUR - We train our BigGAN implementation with the learning rate of the discriminator is 0.0004 and the generator is 0.0001 as used by SAGAN[22].
- Batch size - We use a batch size of 64 as opposed to large batch sizes like 512, 1024 and 2048 in the original paper[3]. The choice was simply because of computational constraints. As a result, we were not able to attain the benchmarks set by the original implementation since the original paper observed a 42% improvement in image quality (quantified by the inception score[1]) by simply increasing the batch size by a factor of 8.
- Channel size - Increasing the channel width by 50% also led to a 21% increase in performance. As increasing the channels led to more than 100% increase in parameters(81.5 million to 173.5 million), we couldn't experiment with channel width. We used a channel width of 64 for our implementation.
- Training - We trained the network for 200,000 iterations on 4 NVIDIA Tesla K80 GPus. The training time for 200,000 iterations was approximately 10 days. Our model hasn't converged at the time of publishing this report and we have reported results for 200,000 iterations. Our model learns slowly as compared to the original one as the batch size is smaller and it has 'seen' lesser data compared to a 2048 batch model.
- Truncation trick- The samples were generated using a truncated z vector. We observed that the samples generated in such a way are generally of better quality when compared to samples generated without truncation (even without orthogonal regularization) as claimed by the original paper.

3.2. DCGAN

- Datasets - We test our implementation of DCGAN on the CIFAR10[9], CelebA[11], Celeb-HQ[8] and a 100 class subset of ImageNet[4].
- Pre-processing - Images were not pre-processed in any way except for normalization and resizing (if required)

[1]Church outdoor: 126,227 images , Brides: 818,687 images

[2]ImageNet top-level classes: Dogs, Cats, Fish, Butterfly, Birds. A total of 259,048 images

[3]Inception score does not make sense for datasets largely different from ImageNet

[4]ImageNet top-level classes: Dogs, Cats, Fish, Butterfly, Birds

Figure 6: Generated images by SN-DCGAN trained on Celeb-HQ

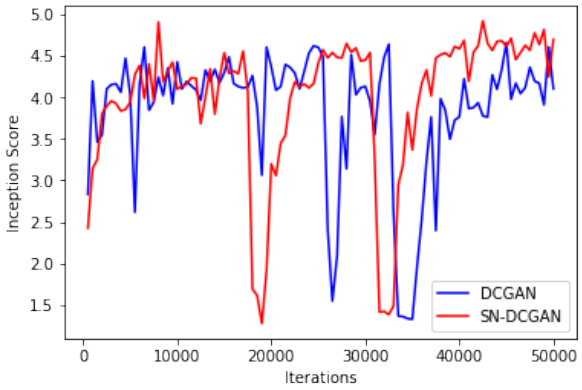
- Learning rate - The learning rate used for generator and discriminator was 0.0002. We retained the learning rate proposed in the original DCGAN implementation.
- Spectral normalization - We experiment with spectral normalization. The results can be found in table 1 and 2. It seems that spectral normalization does not help much in case of small networks. Although we find some gains in terms of inception scores in some cases, the results don't seem conclusive for small network.
- Batch size - We experiment with batch size of 32 and 64.The batch size however shows more promising results. An increase in batch size by a mere factor of 2 resulted in slight gains in both the inception score and frechet inception distance (FID). These gains would probably be more pronounced for a larger batch size increase factor. Note, however that increasing batch size tends to make training unstable. Thus, a better, stabler architecture or more robust regularization would be suitable for such experiments.
- Image resolution - Increasing training image quality in terms of resolution greatly boosts generated image quality as can be seen from the results. Celeb-HQ which is higher quality dataset of Celeb-A gives significantly better results in terms of image quality than its counterpart.
- Truncation - In all experiments, the samples generated by truncated z vector gives better quality results when compared to samples generated without truncation.
- Dropout[18] - We also tried generating 128 x 128 images for the Celeb-HQ and 100-class ImageNet subset. The training collapses immediately even with spectral normalization. We then added dropout layers to the discriminator and found that regularizing the discriminator stabilizes training. We were also able to observe good quality 128 x 128 results.
- Evaluation metrics - The inception scores were calculated using 50,000 generated samples using batch size = 64 and split = 10 for each model.

Architecture	Batch Size	CIFAR-10	ImageNet	CelebA[3]	Celeb-HQ
DCGAN	32	4.66 $\pm$0.053	6.45 $\pm$0.067	3.7 $\pm$0.076	3.56 $\pm$0.077
SN-DCGAN	32	4.92 $\pm$0.049	6.19 $\pm$0.095	3.42 $\pm$0.056	3.51 $\pm$0.039
DCGAN	64	4.65 $\pm$0.051	6.56 $\pm$0.063	3.8 $\pm$0.085	3.47 $\pm$0.055
SN-DCGAN	64	4.41 $\pm$0.057	6.50 $\pm$0.091	3.67 $\pm$0.096	3.22 $\pm$0.059

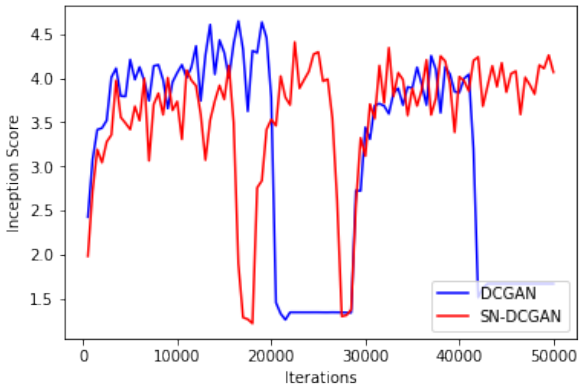
Table 1: Evaluation Metrics (Inception Score)

Architecture	Batch Size	CIFAR-10	ImageNet	CelebA	Celeb-HQ
DCGAN	32	81.8	62.49	53.64	56.14
SN-DCGAN	32	95.46	95.07	62.49	56.7
DCGAN	64	79.24	61.36	54.24	58.77
SN-DCGAN	64	95.07	63.36	56.85	59.1

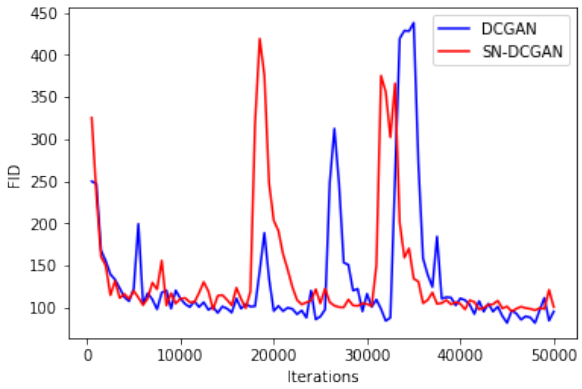
Table 2: Evaluation metrics (FID)



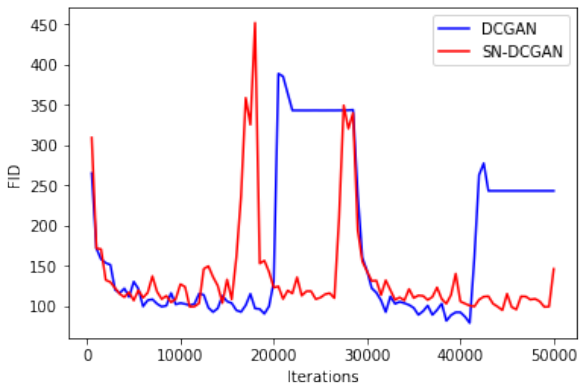
(a) Batch size 32



(b) Batch size 64



(c) Batch size 32

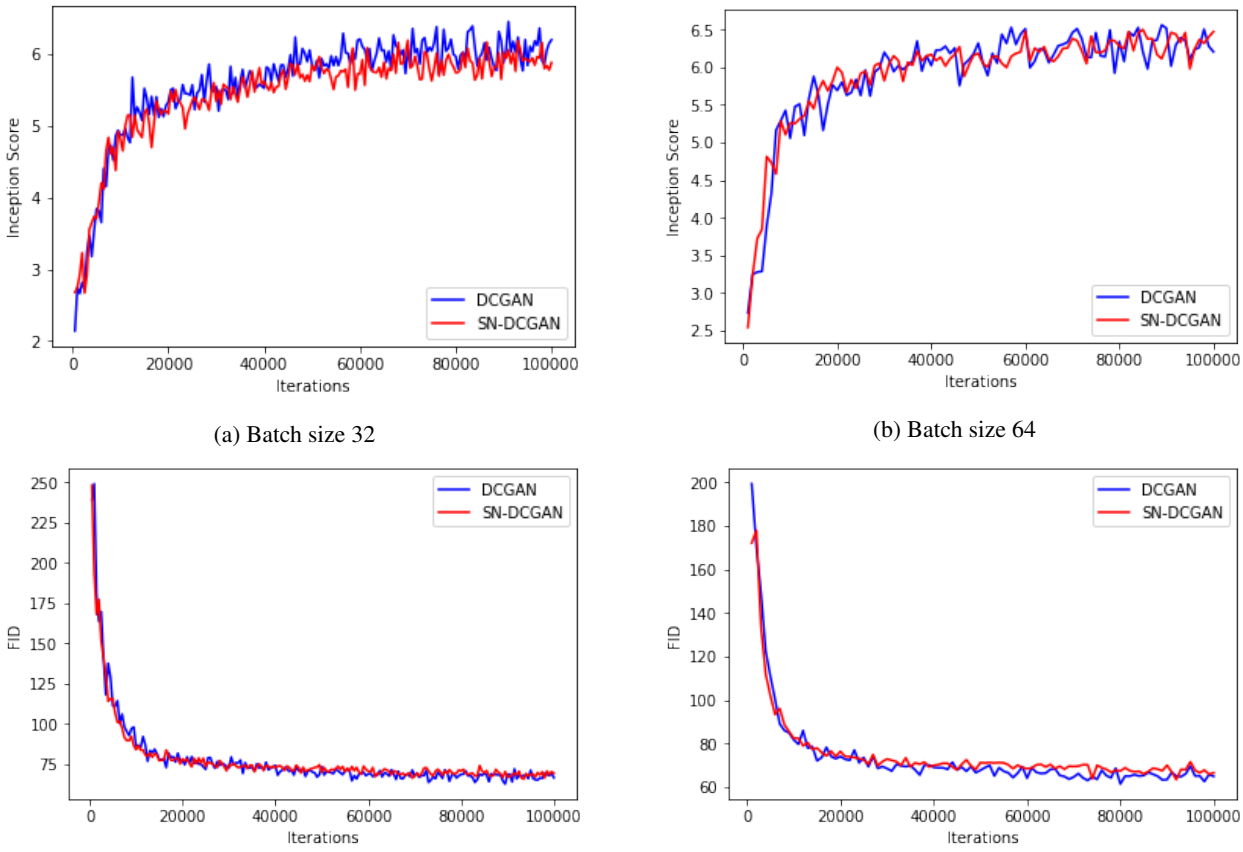


(d) Batch size 64

Figure 7: Training statistics (CIFAR10)

4. Discussion

The encouraging result from various experiments based on Large scale GAN training for High Fidelity Natural Im-



(a) Batch size 32

(b) Batch size 64

(c) Batch size 32

(d) Batch size 64

Figure 8: Training statistics (ImageNet)

age Synthesis conducted over 4 datasets have led us to the conclusion implementing orthogonal initialization, increasing batch size, channels and orthogonal regularization would improve the performance of our model. We plan to incorporate these changes and retrain our BigGan model. In addition to the innovations mentioned in the paper, we observed a strong correlation between image resolution and the fidelity of generated images. We also discovered that scaling DCGAN to generate 128 x 128 images requires heavy regularization of the discriminator network. In addition to this architecture, we have also implemented Mutual Information for Neural Estimation[2] for handling with mode collapse. We are currently in the process of evaluating this algorithm. We are also exploring the effect of attention on image fidelity based on the paper SAGAN[22] for DCGAN. Our future plan for improving training of DCGANs is to add Self attention, TTUR, latent spaces and regularization techniques in this network. [5]

[5]Our code is available here `https://github.com/rikenmehta03/gan_mine`

References

[1] S. Barratt and R. Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[2] I. Belghazi, S. Rajeswar, A. Baratin, R. D. Hjelm, and A. C. Courville. MINE: mutual information neural estimation. *CoRR*, abs/1801.04062, 2018.

[3] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018.

[4] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. *CoRR*, abs/1707.00683, 2017.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[7] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017.

[8] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

[9] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research).

[10] J. H. Lim and J. C. Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017.

[11] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[12] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018.

[13] T. Miyato and M. Koyama. cgans with projection discriminator. *CoRR*, abs/1802.05637, 2018.

[14] T. Miyato and M. Koyama. cgans with projection discriminator. *CoRR*, abs/1802.05637, 2018.

[15] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[16] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[17] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *CoRR*, abs/1312.6120, 2013.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014.

[19] D. Tran, R. Ranganath, and D. Blei. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, pages 5523–5533, 2017.

[20] X. Wang, R. B. Girshick, A. Gupta, and K. He. Non-local neural networks. *CoRR*, abs/1711.07971, 2017.

[21] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015.

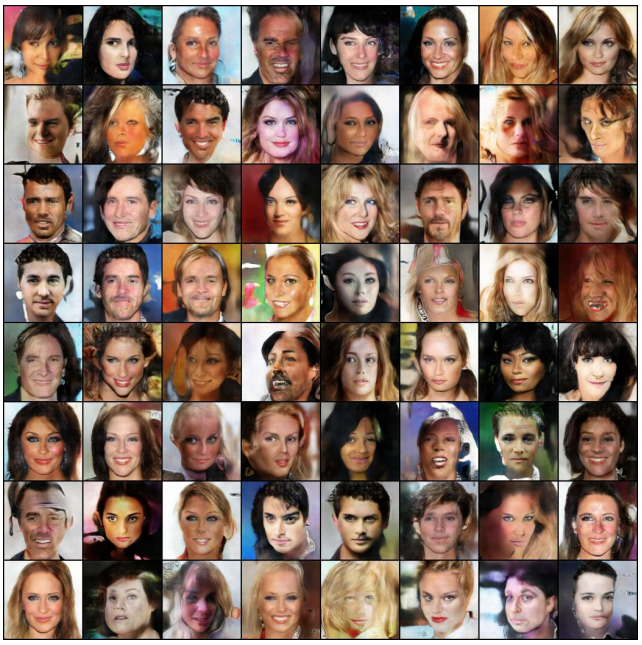
[22] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

5. APPENDIX

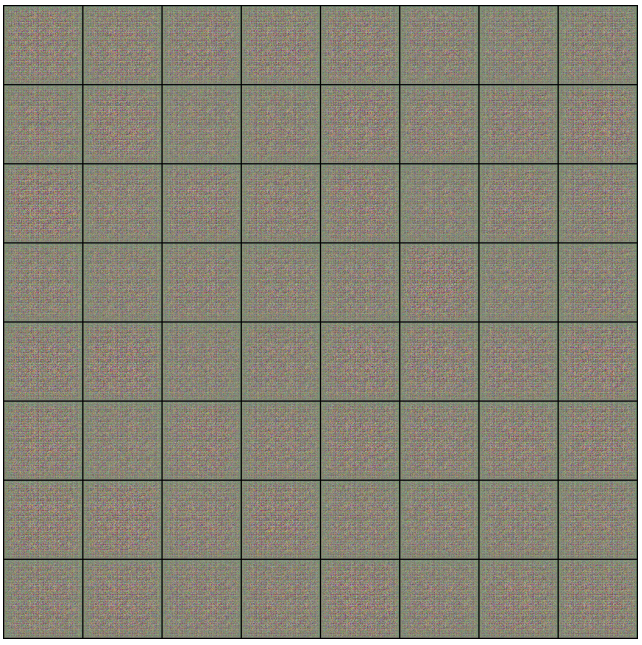
More generated samples from both DCGAN and BigGAN on total 5 different datasets including ImageNet, CIFAR10, CelebA, Celeb-HQ and LSUN.

Figure 11: Generated images by BigGAN trained on LSUN (128 x 128)

Figure 12: Generated images by SN-DCGAN trained on Celeb-HQ (64 x 64)



(a) With dropout



(b) Without dropout (14500 Iteration)

Figure 13: Generated images by SN-DCGAN on Celeb-HQ (128 x 128)

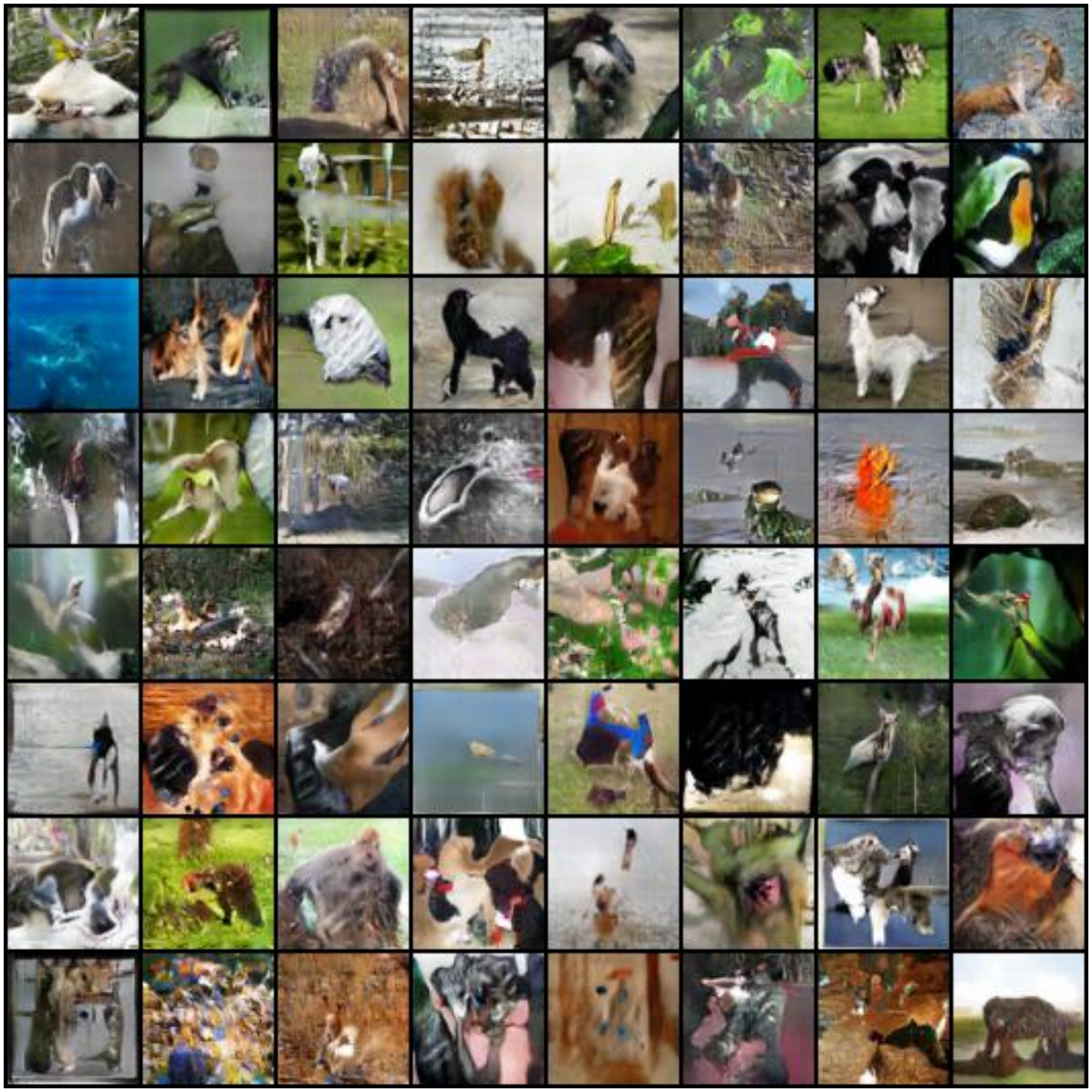


Figure 14: Generated images by SN-DCGAN trained on ImageNet (64 x 64)

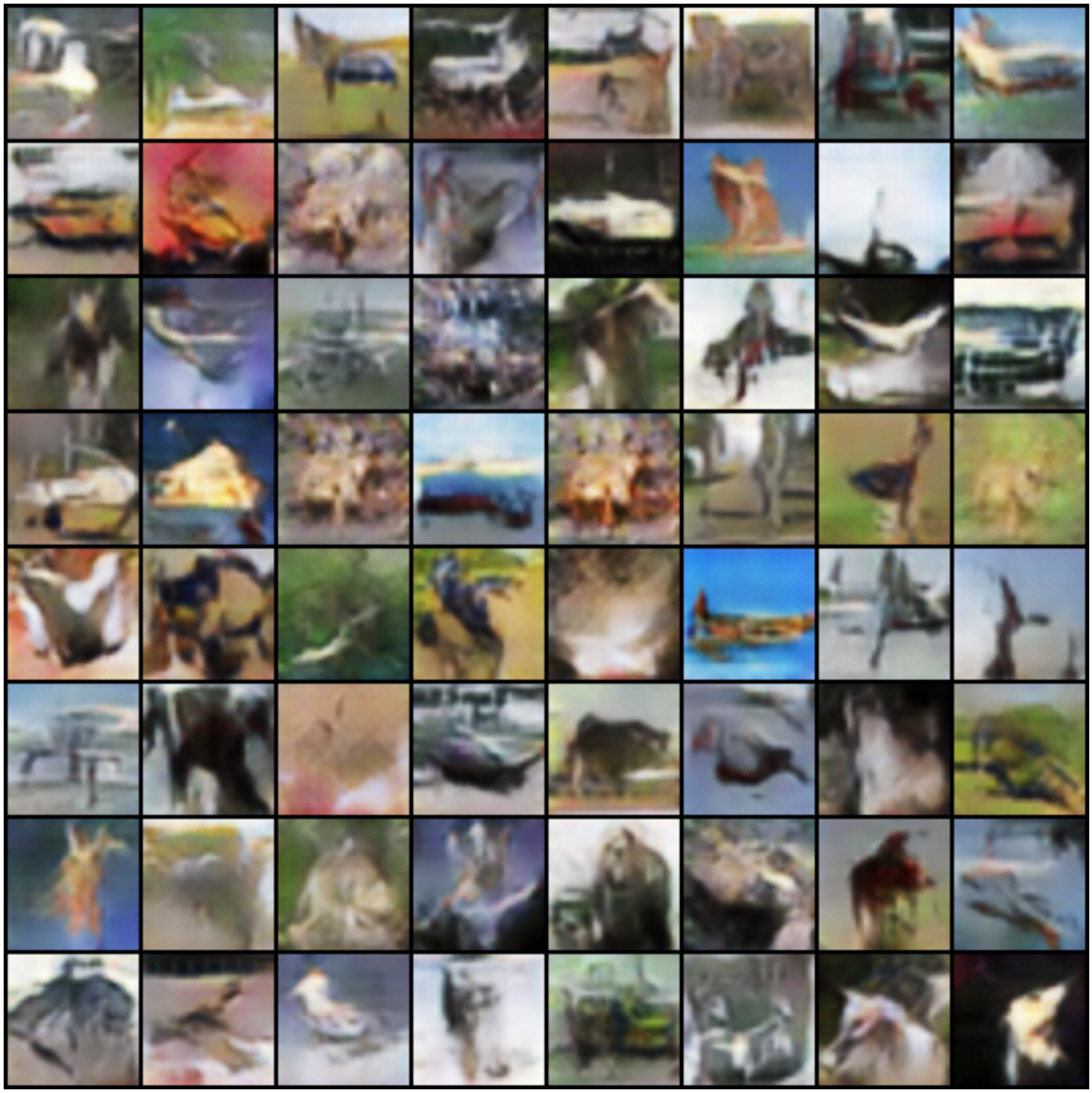


Figure 15: Generated images by SN-DCGAN trained on CIFAR10 (64 x 64)

Figure 16: Generated images by SN-DCGAN trained on CelebA (64 x 64)