# Efficient Integration of Perceptual VAE Into Dynamic Latent Scale GAN

Jeongik Cho

jeongik.jo.01@gmail.com

**Abstract**

Dynamic latent scale GAN is a learning-based GAN inversion method. In this paper, we propose a method to improve the performance of dynamic latent scale GAN by integrating perceptual VAE loss into dynamic latent scale GAN efficiently. When training dynamic latent scale with normal i.i.d. latent random variable, and latent encoder is integrated into discriminator, a sum of predicted latent random variable of real data and scaled normal random variable follows normal i.i.d. random variable. We can consider this random variable as VAE latent random variable and use it for VAE training since there are real data corresponding to latent codes. Considering the intermediate layer output of the discriminator as a feature encoder, we can train the generator with VAE latent random variable to minimize the perceptual distance between generated data and corresponding real data. Furthermore, we can use VAE latent random variable for adversarial training since it has the same distribution as GAN latent random variable. Both generated data and corresponding real data are used during adversarial training with VAE latent random variable, inference & backpropagation for VAE training can be integrated into those of adversarial training. Therefore, training the generator to minimize the perceptual VAE loss does not require additional computation. Perceptual VAE loss is only added to the generator because the encoder is naturally trained with encoder loss of dynamic latent scale GAN.

## 1. Introduction

Training encoder that inverts the generator is called GAN inversion [1]. Dynamic latent scale GAN [2] (DLSGAN) proposed a method of training an encoder that inverts the generator of GAN [3] through maximum likelihood estimation. When the entropy of the latent random variable is too high, it is difficult for the encoder to recover latent code from generated data point because the generator maps different latent codes to the same or similar generated data point. DLSGAN makes it easy for the encoder to invert the generator by appropriately adjusting the entropy of the latent random variable.

There were several works to improve the performance of GAN by utilizing GAN inversion and data reconstruction loss [5, 6, 7].

## 2. Perceptual VAE DLSGAN

The following equations show the loss function for training DLSGAN's encoder.

$$s = \frac{\sqrt{d_z} v_f^{\circ 1/2}}{\left\| v_f^{\circ 1/2} \right\|_2} \tag{1}$$

$$L_{enc} = \mathbb{E}_{z \sim Z} \left\| \left( z - E_l\big(G(z \circ s)\big) \right) \circ s \right\|_2^2 \tag{2}$$

In Eq. 1 and 2, $d_z$ represents a dimension of latent random variable $Z$. $E_l$ and $G$ represent the latent encoder and generator, respectively. $v_f$ and $s$ represent fake latent variance vector and latent scale vector, respectively. DLSGAN uses the moving average of the predicted fake latent vector $E_l\big(G(z \circ s)\big)$ to approximate the fake latent variance vector $v_f$. Operation "$\circ$" is the element-wise multiplication. $vec^{\circ 1/2}$ represents the element-wise square root of vector $vec$. Latent encoder $E_l$ and generator $G$ are trained to minimize encoder loss $L_{enc}$ in DLSGAN.

In this paper, we propose Perceptual VAE DLS GAN (PVDGAN), a method to efficiently integrate perceptual VAE [4] loss into dynamic latent scale GAN to improve the performance of dynamic latent scale GAN.

When training DLSGAN, latent encoder $E_l$ of DLSGAN is trained to predict latent random variable $Z$ from generated data $G(Z \circ s)$. It is clear that $E_l(X) = Z$ if generator $G$ perfectly generates real data random variable $X$, and the latent encoder $E_l$ perfectly inverts generator $G$.

During DLSGAN training (i.e., models are not perfect), if latent encoder $E_l$ and discriminator $D$ is integrated, it will be difficult to distinguish between real data random variable $X$ and generated data random variable $G(Z \circ s)$ for latent encoder $E_l$, since generator $G$ is trained to deceive discriminator $D$ that shares the hidden layers with latent encoder $E_l$.

Based on this intuition, when latent encoder $E_l$ and discriminator $D$ is integrated, we assumed that latent encoder $E_l$ tries to map real data random variable $X$ to latent random variable $Z$ during DLSGAN training, even without explicit loss. Under this assumption, we can generate VAE latent random variable $Z_X$ that follows GAN latent random variable $Z$ by adding noise to predicted real latent code $E_l(X)$.

When latent random variable $Z \sim N(0, I_{d_z})$, each element of predicted real data latent random variable $E_l(X)$ will follow $N(0, \sigma^2)$, where $0 \leq \sigma \leq 1$. Given real latent variance vector $v_r$, which is the element-wise variance of $E_l(X)$, one can simply add scaled normal distribution to $E_l(X)$ to get the same distribution as GAN latent random variable $Z$.

$$Z_X = E_l(X) + N(0, I_{d_z}) \circ (1 - v_r)^{\circ 1/2} \tag{3}$$

Eq. 3 shows VAE latent random variable $Z_X$. One can easily see that $Z_X \sim Z \sim N(0, I_{d_z})$. Unlike VAE, our method does not require variance output for the encoder since there is real latent variance vector $v_r$. Real latent variance vector $v_r$ is approximated through the element-wise variance of predicted real latent codes $E_l(x)$ from previous training steps like DLSGAN.

We can use VAE latent random variable $Z_X$ for VAE training since there is a corresponding real data random variable $X$. The following equations show the loss for VAE training.

$$z_x = E_l(x) + N(0, I_{d_z}) \circ (1 - v_r)^{\circ 1/2} \tag{4}$$

$$L_{rec} = \mathbb{E}_{x \sim X}\left[Dist\left(x, G(z_x \circ s)\right)\right] \tag{5}$$

Eq. 4 is a sample version of Eq. 3. $x$ and $z_x$ represent the real data point and VAE data point of $x$, respectively.

Eq. 5 shows reconstruction loss to train VAE. In Eq. 5, $L_{rec}$ represents reconstruction loss. $Dist$ represents a function that measures the distance between two inputs. $G(z_x \circ s)$ represents reconstructed data of real data point $x$. We can train generator $G$ to minimize reconstruction loss $L_{rec}$ since there are reconstructed data $G(z_x \circ s)$ and corresponding real data $x$. Our method assumes that latent encoder $E_l$ is trained with only encoder loss $L_{enc}$, so latent encoder $E_l$ is not trained with reconstruction loss $L_{rec}$.

$$Dist(a, b) = \frac{1}{d_f}\left\|E_f(a) - E_f(b)\right\|_2^2 \tag{6}$$

Eq. 6 shows the $Dist$ function for reconstruction loss $L_{rec}$. In Eq. 6, $d_f$ and $E_f$ represents feature vector dimension and feature encoder, respectively. One can see that function $Dist$ measures the perceptual distance between two data points with feature encoder $E_f$.

Finding a good $Dist$ function is not an easy problem. For example, if we simply use the mean squared error of pixel values for image VAE (i.e., $E_f(x) = x$), the generated images will be very blurry. Pixel-level mean squared error is a good choice if we want the minimize pixel-level distance between input data and reconstructed data, but in most cases, we want to minimize perceptual distance. One can simply think of using a pre-trained model as feature encoder $E_f$. However, if we use a pre-trained model, we need additional computations for inference & backpropagation of the pre-trained model to minimize $L_{rec}$. Also, there might be no good pre-trained models for some data domains. Furthermore, it is hard to customize a pre-trained model (e.g., input resolution is fixed).

Instead of using a pre-trained model, we proposed to use discriminator intermediate layer output as feature encoder $L_f$. We know that VAE latent random variable $Z_X$ is the same distribution as GAN latent random variable $Z$. Therefore, we can use VAE latent random variable $Z_X$ for adversarial training. During adversarial training with VAE latent code $z_x$, there are inference & backpropagation on generator $G$ and discriminator $D$ with real data $x$ and reconstructed data $G(z_x \circ s)$. Therefore, since inference & backpropagation for minimizing reconstruction loss $L_{rec}$ can be integrated into the inference & backpropagation of the adversarial training step, additional computation for minimizing reconstruction loss $L_{rec}$ is not required.

If VAE latent random variable $Z_X$ is different from GAN latent random variable $Z$, adversarial training with VAE latent random variable $Z_X$ is not only meaningless but rather makes GAN training more difficult because generator $G$ and discriminator $D$ should generate and discriminate with not only for latent distribution $Z$ but also for unknown distribution $Z_X$.

In short, when training DLSGAN with GAN latent random variable $Z \sim N(0, I_{d_z})$, and latent encoder $E_l$ is integrated into discriminator $D$, since VAE latent random variable $Z_X = E_l(X) + N(0, I_{d_z}) \circ (1 - v_r)^{\circ 1/2}$ follows GAN latent random variable $Z$, VAE latent random variable $Z_X$ can be used for adversarial training. During the adversarial training with VAE latent random variable $Z_X$, since there are inference & backpropagation with real data $x$ and reconstructed data $G(z_x)$, no additional computation is required for the generator $G$ to minimize reconstruction loss $L_{rec}$.

The following algorithm shows the algorithm to obtain loss for PVDGAN.

---

$function\ get\_loss(D^*, G, Z, X, b, v_r, v_f)$:

1: $x \leftarrow sample(X, b)$

2: $s \leftarrow \frac{\sqrt{d_z} v_f^{\circ 1/2}}{\left\| v_f^{\circ 1/2} \right\|_2}$

3: $a_r,\ z_r,\ y_r \leftarrow D^*(x)$

4: $z \leftarrow concat\left( \begin{matrix} sample(Z, b/2), \\ nograd(z_r[b/2:]) + sample(Z, b/2) \circ (1 - v_r)^{\circ 1/2} \end{matrix} \right)$

5: $a_f,\ z',\ y_r' \leftarrow D^*(G(z \circ s))$

6: $L_{enc} \leftarrow \frac{1}{b \times d_z} \|(z - z') \circ s\|_2^2$

7: $L_{rec} \leftarrow \frac{1}{b/2 \times d_y} \|y_r[b/2:] - y_r'[b/2:]\|_2^2$

8: $L_d \leftarrow adv(a_r, a_f) + \lambda_{enc} L_{enc}$

9: $L_f \leftarrow adv(a_f) + \lambda_{enc} L_{enc} + \lambda_{rec} L_{rec}$

10: $v_r \leftarrow update(v_r, z_r^{\circ 2})$

11: $v_f \leftarrow update(v_f, z'^{\circ 2})$

12: $return\ L_d, L_f, v_r, v_f$

---

Algorithm 1. Algorithm to obtain loss for PVDGAN

In Algo. 1, $D^*$, $G$, $Z$, and $X$ represent the integrated discriminator, generator, latent random variable, and data random variable, respectively. $Z$ follows $d_z$-dimensional i.i.d. normal distribution. In Algo. 1, it was assumed that the latent random variable $Z$ follows $N(0, I_{d_z})$ for convenience. $D^*$ is the integrated discriminator in which discriminator $D$, latent encoder $E_l$, and feature encoder $E_f$ are integrated. Therefore, integrated discriminator $D^*$ has 3 outputs. $b$ represents batch size. $v_r$ and $v_f$ represent traced real latent variance vector and traced fake latent variance vector, respectively. $v_f$ corresponds to the traced latent variance vector of DLSGAN.

In line 1, $sample(A, n)$ is a function that returns $n$ samples from random variable $A$. $x$ represents sampled real data points.

In line 2, $s$ is the $d_z$-dimensional latent scale vector of DLSGAN.

In line 3, one can see that integrated discriminator $D^*$ outputs 3 values. First, $a_r$ is $b \times 1$ shape real data adversarial value. Second, $z_r$ is $b \times d_z$ shape real latent code. Third, $y_r$ is $b \times d_f$ shape real feature vector. Unlike the other two outputs, the feature vector $y_r$ is the intermediate layer output of the integrated discriminator $D^*$.

In line 4, $nograd(k)$ is a function that prevents the gradient flow to input $k$. The output of $nograd$ is the same as the input. $z_r[b/2:]$ represents last $b/2$ samples of $z_r$. Therefore, $z_r[b/2:]$ is $\frac{b}{2} \times d_z$ matrix. One can see that $nograd(z_r[b/2:]) + sample(Z, b/2) \circ (1 - v_r)^{\circ 1/2}$ follows latent random variable $Z$. In an ideal case, all elements of $v_r$ are less than or equal to 1, but for stability, we recommend to use $\max(1 - v_r, 0)^{\circ 1/2}$ instead of $(1 - v_r)^{\circ 1/2}$ for stability. $concat$ represents concatenate function. Therefore, $z$ is $b \times d_z$ shape matrix, the first $b/2$ elements of which are latent codes sampled from latent random variable $Z$, and the last $b/2$ elements of which are generated from $z_r[b/2:]$.

In line 5, $a_f$ is $b \times 1$ shape fake data adversarial value. $z'$ and $y_r'$ represent predicted latent codes and predicted feature vectors.

In lines 6 and 7, $L_{enc}$ and $L_{rec}$ represent encoder loss of DLSGAN and perceptual reconstruction loss, respectively. One can see that $y_r[b/2:]$ and $y_r'[b/2:]$, which were generated from $x[b/2:]$ were used for reconstruction loss $L_{rec}$.

In lines 8 and 9, $L_d$ and $L_f$ represent discriminator loss and generator loss, respectively. $\lambda_{enc}$ and $\lambda_{rec}$ represent encoder loss weight and perceptual reconstruction loss weight, respectively. $adv$ represents adversarial loss function [8, 10]. One can see that there is no reconstruction loss $L_{rec}$ for integrated discriminator $D^*$.

In lines 10 and 11, traced real latent variance $v_r$ and traced fake latent variance $v_f$ are updated as DLSGAN, respectively.

One can see that the above algorithm requires $b$ generator inference & backpropagation and $2b$ discriminator inference & backpropagation. It is the same as the training step of a general GAN. Therefore, PVDGAN does not require additional computation compared to basic GAN or DLSGAN.

## 3. Experiments

We compared the performance of PVDGAN and DLSGAN.

We used the FFHQ dataset [9] resized to $256 \times 256$ resolution. Among 70k images, the first 60k images were used as a training set, and the left 10k images were used as test images. Pixel values were normalized from -1 to 1, and a 50% random left-right flip was used for data augmentation.

NSGAN with R1 regularization [10] was used as an adversarial loss. We used a simple model architecture consisting of only convolution layers and skip connections. We used upsample/downsample of SWAGAN [11] with equalized learning rate [12]. We did not use a direct skip connection to the input in the discriminator. It may make GAN inversion hard but increases generative performance. Both methods used the same architecture. The second last convolutional block output of the discriminator was used as the feature encoder.

We used FID [13], Precision & Recall [14] metrics with a pre-trained inception model for generative performance evaluation. 10k test images and 10k generated images were used for generative performance evaluation. Pre-trained inception model and size of the neighborhood $k = 3$ were used for Precision & Recall evaluation. Average PSNR and average SSIM were used for inversion and comprehensive performance evaluation as DLSGAN.

The following figures show the performance of DLSGAN and PVDGAN for each epoch.

$$\lambda_{r1} = 5.0$$

$$\lambda_{enc} = 1.0$$

$$d_z = 1024$$

$$Z \sim N(0, I_{d_z})$$

$$optimizer = Adam\begin{pmatrix} learning\ rate = 0.003 \\ \beta_1 = 0.0 \\ \beta_2 = 0.99 \end{pmatrix}$$

$$trainable\ weights\ ema\ decay\ rate = 0.999$$

$$latent\ variance\ vector\ ema\ decay\ rate = 0.999$$

$$batch\ size = 16$$

$$epochs = 50$$

PVDGAN used reconstruction loss weight $\lambda_{rec} = 1.0$.

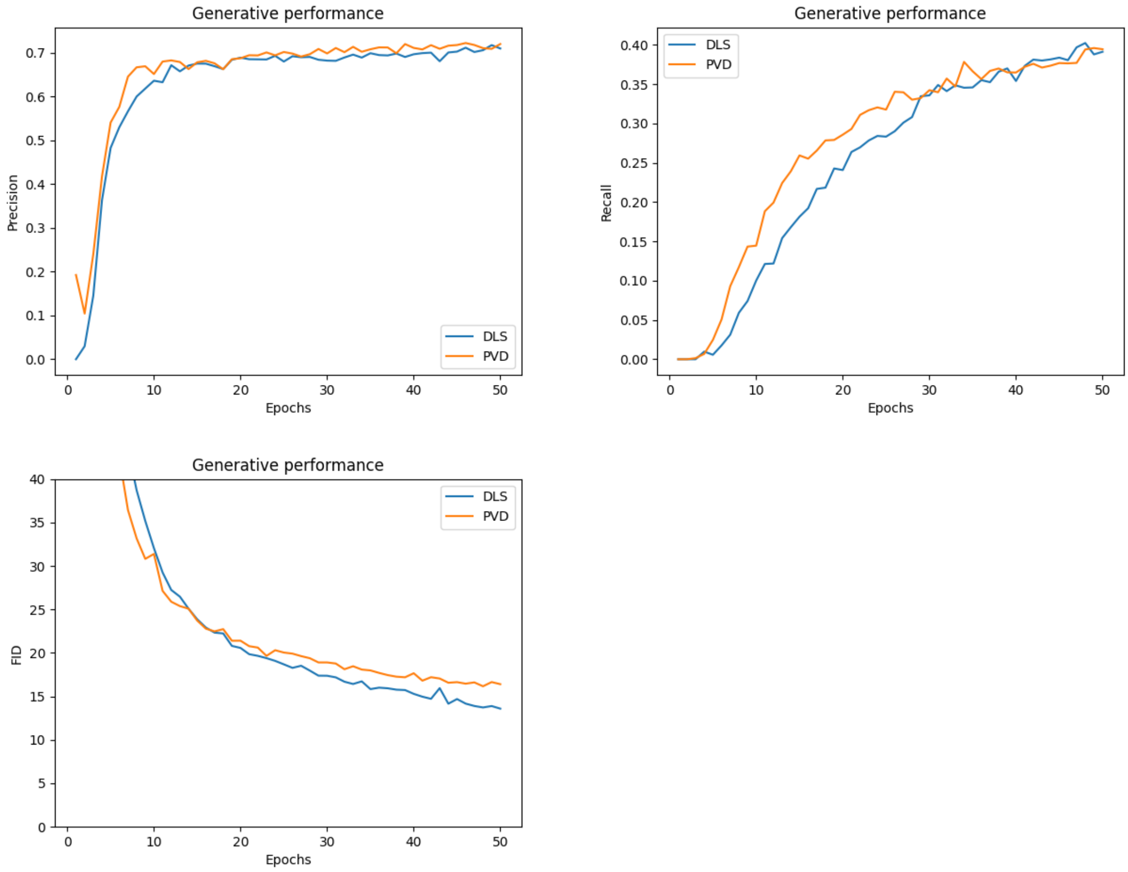


Figure 1. Generative performance for each epoch.

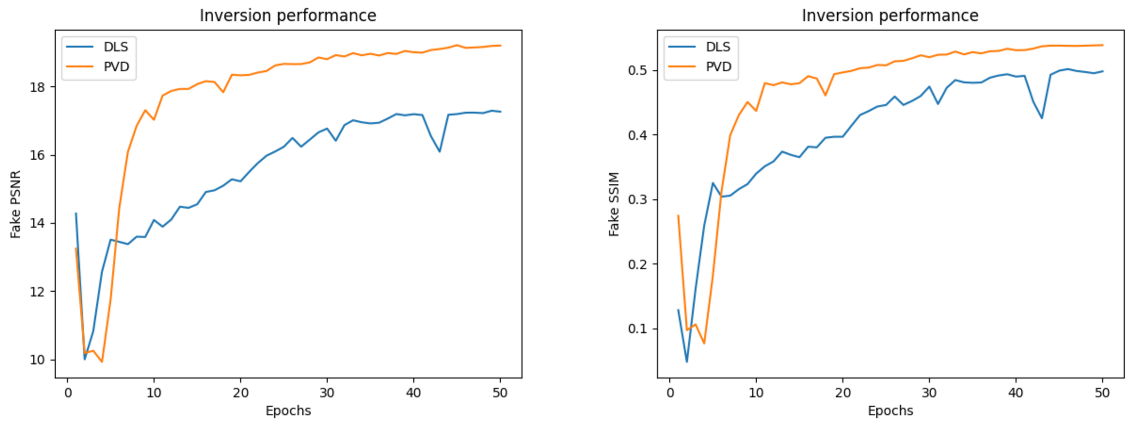


Figure 2. Inversion performance for each epoch.

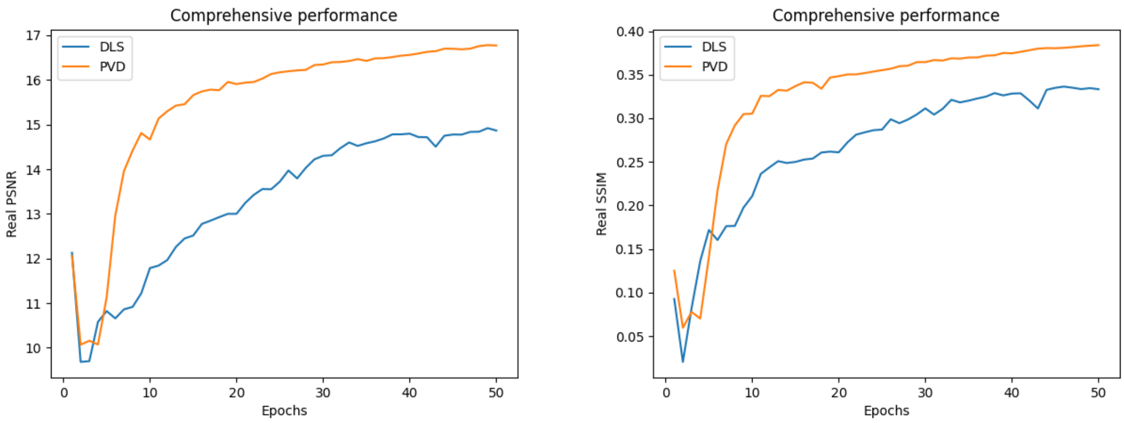


Figure 3. Comprehensive performance for each epoch.

Figs. 1-3 show the generative, inversion, and comprehensive performance of models, respectively.

In Fig. 1, DLSGAN shows better generative performance with FID evaluation. However, one can see that there is no significant difference between DLSGAN and PVDGAN with precision & recall evaluation.

In Figs. 2 and 3, PVDGAN clearly shows better inversion and comprehensive performance than DLSGAN. The following figure shows unseen test image reconstruction examples.


Figure 4. Test image reconstruction examples.

In Fig. 4, one can see that PVDGAN shows better perceptual real image reconstruction (e.g., rows 3, 4, 7 in left the part of Fig. 4)

## 4. Conclusion

In this paper, we propose a method to integrate the perceptual VAE loss into the DLSGAN generator very efficiently to improve the performance of DLSGAN. When the discriminator and latent encoder are integrated, and GAN latent random variable is normal i.i.d. random variable, a sum of the predicted real latent random variable and scaled normal random variable also follows GAN latent random variable. Therefore, we can use it for both adversarial training and VAE training. Considering discriminator intermediate layer output as a feature encoder, perceptual VAE training of the generator does not require additional computation. The proposed method improved the performance of DLSGAN.

## 5. References


[1] W. Xia, Y. Zhang, Y. Yang, J. -H. Xue, B. Zhou and M. -H. Yang, "GAN Inversion: A Survey," in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022, doi: 10.1109/TPAMI.2022.3181070. https://ieeexplore.ieee.org/abstract/document/9792208

[2] J. Cho, A. Krzyzak, “Dynamic Latent Scale for GAN Inversion,” in Proceedings of 11th ICPRAM, pp. 221-228, 2022. https://www.scitepress.org/Link.aspx?doi=10.5220/0010816800003122

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, and D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, “Generative adversarial networks,” in Communications of the ACM, Volume 63, Issue 11, November 2020, pp. 139–144. https://dl.acm.org/doi/abs/10.1145/3422622

[4] D. P Kingma, M. Welling, “Auto-Encoding Variational Bayes,” in arXiv preprint, Dec 2013. https://arxiv.org/abs/1312.6114v11

[5] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, O. Winther, “Autoencoding beyond pixels using a learned similarity metric” in arXiv preprint, Dec 2015. https://arxiv.org/abs/1512.09300

[6] D. Ulyanov, A. Vedaldi, V. Lempitsky, “It takes (only) two: adversarial generator-encoder networks,” in AAAI, 2018. https://dl.acm.org/doi/10.5555/3504035.3504188

[7] J. Zhu, Y. Shen, D. Zhao, B. Zhou, “In-Domain GAN Inversion for Real Image Editing,” in ECCV 2020, pp. 592-608. https://link.springer.com/chapter/10.1007/978-3-030-58520-4_35

[8] M. Lucic, K. Kurach, M. Michalski, S. Gelly, O. Bousquet, “Are GANs Created Equal? A Large-Scale Study,” in NIPS 2018. https://papers.nips.cc/paper/2018/hash/e46de7e1bcaaced9a54f1e9d0d2f800d-Abstract.html

[9] T. Karras, S. Laine and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4396-4405, doi: 10.1109/CVPR.2019.00453. https://ieeexplore.ieee.org/document/8953766

[10] L. Mescheder, A. Geiger, S. Nowozin, “Which Training Methods for GANs do actually Converge?,” in PMLR 2018. https://proceedings.mlr.press/v80/mescheder18a.html

[11] R. Gal, D. C. Hochberg, A. Bermano, D. Cohen-Or, “SWAGAN: a style-based wavelet-driven generative model,” in ACM Transactions on Graphics, Vol. 40, pp. 1-11, August 2021.

https://dl.acm.org/doi/10.1145/3450626.3459836

[12] T. Karras, T. Aila, S. Laine, J. Lehtinen, “Progressive Growing of GANs for Improved Quality, Stability, and Variation,” in ICLR 2018. https://openreview.net/forum?id=Hk99zCeAb

[13] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," in NIPS 2017.
https://papers.nips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html

[14] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, T. Aila, "Improved Precision and Recall Metric for Assessing Generative Models," in NIPS 2019.
https://proceedings.neurips.cc/paper/2019/hash/0234c510bc6d908b28c70ff313743079-Abstract.html