Efficient Integration of Perceptual VAE into Dynamic Latent Scale GAN

Jeongik Cho
jeongik.jo.01@gmail.com

Abstract

Dynamic latent scale GAN is a method to train an encoder that inverts the generator of GAN with maximum likelihood estimation. 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 GAN with normal i.i.d. latent random variable, and latent encoder is integrated into discriminator, a sum of a predicted latent random variable of real data and a scaled normal noise follows normal i.i.d. random variable. This random variable can be used for both VAE and GAN training. Considering the intermediate layer output of the discriminator as a feature encoder output, the generator can be trained to minimize perceptual VAE loss. Also, inference & backpropagation for perceptual VAE loss can be integrated into those for GAN training. Therefore, perceptual VAE training does not require additional computation. Also, the proposed method does not require prior loss or variance estimation like VAE.

1 Introduction

Recently, generative adversarial network [1] (GAN) has shown impressive performance in generating high-quality data but is still suffering from low diversity of fake data. Variational autoencoder [2] (VAE) has shown better diversity of fake data, but the quality of the fake data is low [4]. There were several works to integrate VAE or autoencoder into GAN [6, 7, 8, 9] to have better generative performance.

Dynamic latent scale GAN [5] (DLSGAN) is a GAN inversion [3] method to train an encoder that inverts the generator of GAN with maximum likelihood estimation. In this paper, we propose Perceptual VAE DLSGAN (PVDGAN), a method to integrate perceptual VAE into DLSGAN efficiently. When training DLSGAN with normal i.i.d. latent random variable, and latent encoder is integrated into discriminator, we assumed that the latent encoder tries to map real data random variable to latent random variable without explicit loss. Therefore, a sum of a predicted real latent random variable and a scaled normal noise follows a normal i.i.d. random variable. This random variable is used for both VAE and GAN training in PVDGAN.

Furthermore, considering the intermediate layer output of the discriminator as a feature encoder output, the generator can be trained to minimize perceptual VAE loss. Also, inference & backpropagation for perceptual VAE loss can be integrated into those for GAN training. Therefore, perceptual VAE training does not require additional computation.

One can think our work is a variation of VAEGAN [6]. The first difference between VAEGAN and PVDGAN is that PVDGAN does not use VAE prior loss or variance estimation. Since DLSGAN is a maximum likelihood estimation method, encoder loss of DLSGAN can replace VAE prior loss. For the same reason, the encoder does not need to estimate VAE variance. PVDGAN replaces VAE variance estimation through the variance of the predicted latent random variable. The second difference is that VAEGAN uses 3 models (encoder, decoder, discriminator) for training, while PVDGAN uses only 2 models. Using only 2 models in PVDGAN is not only computationally efficient but also produces a better perceptual loss.

PVDGAN improved DLSGAN performance without additional computation.

2 Perceptual VAE DLSGAN

DLSGAN [5] is a GAN inversion method to train an encoder that inverts the generator of GAN with maximum likelihood estimation when the latent random variable is an i.i.d. random variable. If the entropy of the latent random variable is too high, it is difficult for the encoder to recover latent code from the fake data. This is because the generator maps different latent codes to the same or similar fake data point. DLSGAN dynamically adjusts the element-wise scale of the latent random variable so that the scaled latent random variable is appropriate to be transformed into real data random variable.

The following equations show the encoder loss of DLSGAN.

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

$$L_{enc} = \frac{1}{d_z} \|(Z - E_l(G(Z \circ s))) \circ s\|_2^2 \tag{2}$$

In Eqs. 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 and s represent the latent variance vector and latent scale vector, respectively. DLSGAN uses the moving average of the predicted fake latent code $E_l(G(z \circ s))$ to approximate the latent variance vector v. 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 PVDGAN, a method to efficiently integrate perceptual VAE [2] loss into DLSGAN to improve DLSGAN performance.

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

During DLSGAN training, if latent encoder E_l and discriminator D are integrated, it will be difficult to distinguish between real data random variable X and fake data random variable $G(Z \circ s)$ for latent encoder E_l. This is because latent encoder E_l shares hidden layers with discriminator D in adversarial training with generator G. Based on this intuition, we assumed that latent encoder E_l tries to map real data random variable X to the latent random variable Z during DLSGAN training, even without explicit loss.

Under this assumption, VAE latent random variable Z_x that follows GAN latent random variable Z can be generated by adding scaled normal noise to the predicted real latent random variable $E_l(X)$.

When latent random variable $Z \sim N(0, I_{d_z})$, we assumed that predicted real latent random variable $E_l(X)$ follows $N(0, I_{d_z}) \circ v^{\circ 1/2}$, where $0 \le v \le 1$. Therefore, adding scaled normal noise to the real latent random variable $E_l(X)$ can follow normal i.i.d. random variable.

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

Eq. 3 shows VAE latent random variable Z_x. One can easily find that $Z_x \sim Z \sim N(0, I_{d_z})$ because $E_l(X) \sim N(0, I_{d_z}) \circ v^{\circ 1/2}$.

VAE latent random variable Z_x can be used for VAE training since there is a corresponding real data random variable X. The following equations show the loss for VAE training.

$$L_{rec} = Dist(X, G(Z_x \circ s)) \tag{4}$$

In Eq. 4, L_{rec} represents reconstruction loss. $Dist$ represents a function that measures the distance between two random variables. $G(Z_x \circ s)$ represents the VAE reconstructed data random variable of real data random variable X. DLSGAN 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} in PVDGAN.

$$Dist(a, b) = \frac{1}{d_f} \|E_f(a) - E_f(b)\|_2^2 \tag{5}$$

Eq. 5 shows the $Dist$ function for reconstruction loss L_{rec}. In Eq. 5, d_f and E_f represent feature vector dimension and feature encoder, respectively. One can see that function $Dist$ measures the perceptual distance between two data with feature encoder E_f.

Finding a good $Dist$ function is not an easy problem. For example, if image VAE is trained with pixel-wise mean squared error (i.e., $E_f(x) = x$), the fake images will be very blurry. In most cases, we want to minimize perceptual distance. One can simply think of using a pre-trained model (e.g., pre-trained inception model). 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, the model is too large or small).

Instead of using a pre-trained model, PVDGAN uses discriminator intermediate layer output as feature encoder L_f output like VAEGAN [6]. Since VAE latent random variable Z_x follows GAN latent random variable Z, it can be used for GAN training as well. During GAN 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, inference & backpropagation for minimizing reconstruction loss L_{rec} can be integrated into the inference & backpropagation for the GAN training. It means that no additional inference & backpropagation for minimizing reconstruction loss L_{rec} is required.

If VAE latent random variable Z_x is different from GAN latent random variable Z, GAN training with VAE latent random variable Z_x is not only meaningless but rather makes GAN training more difficult. This is because generator G and discriminator D should generate and discriminate not only for the latent random variable Z but also for VAE latent random variable 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, VAE latent random variable $Z_x = E_l(X) + N(0, I_{d_z}) \circ (1-v)^{\circ 1/2}$ follows GAN latent random variable Z. Therefore, VAE latent random variable Z_x can be used for GAN training. Also, there are already inference & backpropagation with real data x and reconstructed data $G(z_x \circ s)$ in GAN training with VAE latent random variable Z_x. Therefore, VAE training (minimizing reconstruction loss L_{rec}) does not require additional inference & backpropagation.

Algorithm 1 Algorithm to obtain PVDGAN loss

```
Require: $D^*, G, Z, X, b, v$
 1: $x \Leftarrow sample(X, b)$
 2: $s \Leftarrow \frac{\sqrt{d_z} v^{\circ 1/2}}{\|v^{\circ 1/2}\|_2}$

 3: $a_r, z_r, y_r \Leftarrow D^*(x)$
 4: $z_x \Leftarrow nograd(z_r[b/2:]) + sample(Z, b/2) \circ (1-v)^{\circ 1/2}$
 5: $z \Leftarrow concat(sample(Z, b/2), z_x)$
 6: $a_f, z', y'_r \Leftarrow D^*(G(z \circ s))$

 7: $L_{enc} \Leftarrow \frac{1}{b \times d_z} \|(z - z') \circ s\|_2^2$
 8: $L_{rec} \Leftarrow \frac{1}{b/2 \times d_y} \|y_r[b/2:] - y'_r[b/2:]\|_2^2$

 9: $L_d \Leftarrow adv(a_r, a_f) + \lambda_{enc} L_{enc}$
10: $L_g \Leftarrow adv(a_f) + \lambda_{enc} L_{enc} + \lambda_{rec} L_{rec}$

11: $v \Leftarrow update(v, z'^{\circ 2})$

12: $return\ L_d, L_g, v$
```

Algo. 1 shows the algorithm to obtain loss for PVDGAN. In Algo. 1, D^*, G, Z, and X represent the integrated discriminator, generator, latent random variable, and real data random variable, respectively. In Algo. 1, it was assumed that the latent random variable Z follows $N(0, I_{d_z})$. D^* is the integrated discriminator in which discriminator D, latent encoder E_l, and feature encoder E_f are integrated. Therefore, the integrated discriminator D^* has 3 outputs. b and v represent the batch size and latent variance vector, respectively.

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. The first output a_r is $b \times 1$ shape real data adversarial values. The second output z_r is $b \times d_z$ shape predicted real latent codes. The third output y_r is $b \times d_f$ shape real feature vectors. Unlike the other two outputs, the feature vectors y_r are the intermediate layer outputs of the integrated discriminator D^*.

In line 4, z_x represents VAE latent codes. $nograd(k)$ represents 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_x is $\frac{b}{2} \times d_z$ matrix. In general cases, all elements of latent variance vector v are less than or equal to 1, but for stability, we recommend using $\max(1-v, 0)^{\circ 1/2}$ instead of $(1-v)^{\circ 1/2}$.

In line 5, $concat$ represents the concatenation function. Therefore, z is $b \times d_z$ shape matrix, the first $b/2$ elements are sampled from latent random variable Z, and the last $b/2$ elements are generated from VAE latent codes z_x.

In line 6, a_f is $b \times 1$ shape fake data adversarial value. z' and y'_r represent predicted latent codes and predicted feature vectors, respectively.

In lines 7 and 8, L_{enc} and L_{rec} represent encoder loss of DLSGAN and perceptual reconstruction loss, respectively.

In lines 9 and 10, L_d and L_g represent discriminator loss and generator loss, respectively. λ_{enc} and λ_{rec} represent encoder loss weight and perceptual reconstruction loss weight, respectively. adv represents adversarial loss function [10, 12]. One can see that there is no reconstruction loss L_{rec} for integrated discriminator D^*.

In line 11, the latent variance vector v is updated as DLSGAN (i.e., moving average).

One can see that no additional inference & backpropagation is required to minimize L_{rec} in Algo. 1. Therefore, PVDGAN does not require additional computation compared to basic GAN or DLSGAN (to be precise, there is an additional d_f-dimensional vector operation to calculate L_{rec}, but it is very small compared to inference & backpropagation in most deep learning models).

Unlike VAEGAN [6] or VAE [2], PVDGAN does not use VAE prior loss and encoder variance estimation. Since DLSGAN is a maximum likelihood estimation method, encoder loss L_{enc} can replace VAE prior loss. For the same reason, latent variance vector v can replace encoder variance estimation.

Also, VAEGAN uses 3 models (encoder, decoder, discriminator) for training, while PVDGAN uses only 2 models. Using only 2 models in PVDGAN is computationally efficient and expected to produce a better perceptual loss. This is because the discriminator does not need all information of input data to discriminate input data, so the discriminator of VAEGAN may lose the information of input data.

Simply, assume that the penultimate layer of the discriminator has only one unit (i.e., output dimension is 1), and using this penultimate layer output as feature encoder output in VAEGAN (i.e., $D(x) = E_f(x) \cdot w + b$, where w and b are both 1-dimensional trainable weight). One can find that perceptual loss with this feature encoder is almost meaningless. On the other hand, the dimension of the PVDGAN feature encoder output is at least $d_z + 1$. Also, the latent encoder E_l is trained not to lose the information of fake data. Therefore, one can expect PVDGAN generates a more useful perceptual loss.

3 Experiments

We compared the performance of PVDGAN and DLSGAN.

We used the FFHQ dataset [11] resized to 256×256 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 [1] with R1 regularization [12] 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 [13] with equalized learning rate [14]. 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 model architecture. The second last convolutional block output of the discriminator was used as the feature encoder output of PVDGAN.

We used FID [15], Precision & Recall [16] metrics with a pre-trained inception model for generative performance evaluation. 10k test real images and 10k fake 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 hyperparameters were used for experiments.

$$\lambda_{r1} = 3.0$$
$$\lambda_{enc} = 1.0$$
$$d_z = 1024$$
$$Z \sim N(0, I_{d_z})$$
$$optimizer = Adam \begin{pmatrix} learning\ rate = 0.003 \\ \beta_1 = 0.0 \\ \beta_2 = 0.99 \end{pmatrix}$$
$$trainable\ weights\ ema\ decay\ rate = 0.999$$
$$latent\ variance\ vector\ ema\ decay\ rate = 0.999$$
$$batch\ size = 16$$
$$epochs = 49$$

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

Figs. 1, 2, 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.

Fig. 4 shows unseen test image reconstruction examples. One can see that PVDGAN shows perceptually slightly better real image reconstruction. For example, PVDGAN showed better eye reconstruction in the first image of the left part and better hair reconstruction in the first, second, and fourth images of the right part.

4 Conclusion

In this paper, we proposed PVDGAN to integrate the perceptual VAE loss into the DLSGAN generator efficiently to improve the performance of DLSGAN. When training DLSGAN with normal i.i.d. latent random variable, and latent encoder is integrated into the discriminator, a sum of a predicted latent random variable of real data and a scaled normal noise follows normal i.i.d. random variable. PVDGAN uses this random

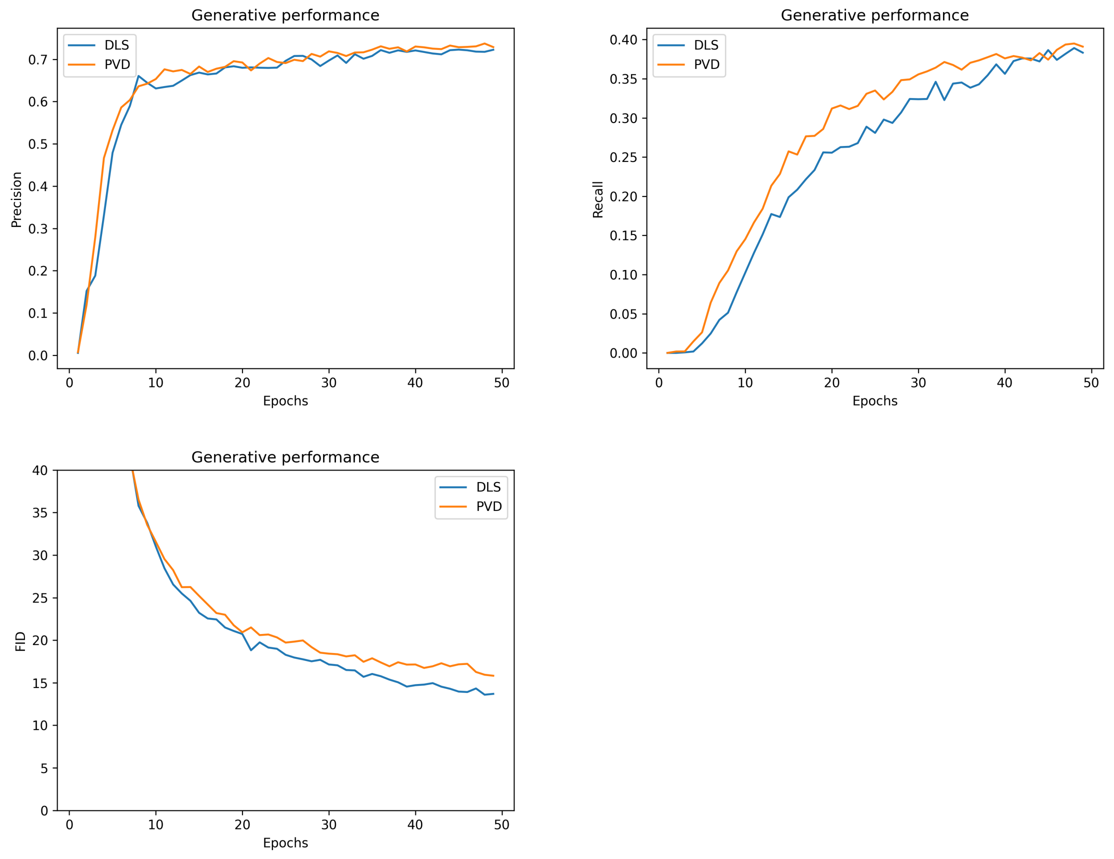


Figure 1: Generative performance for each epoch.

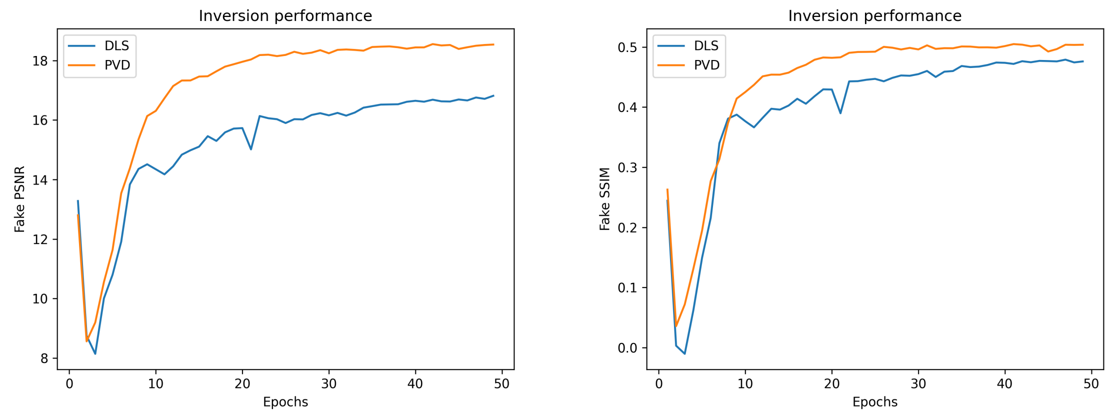
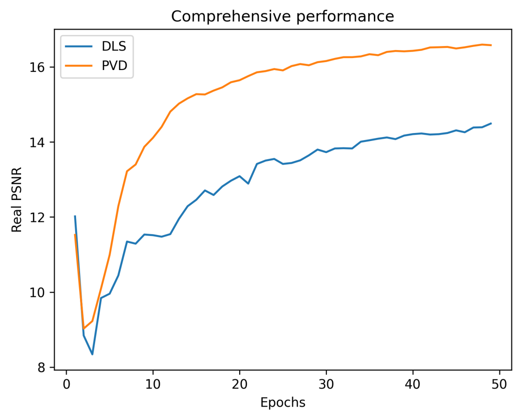


Figure 2: Inversion performance for each epoch.



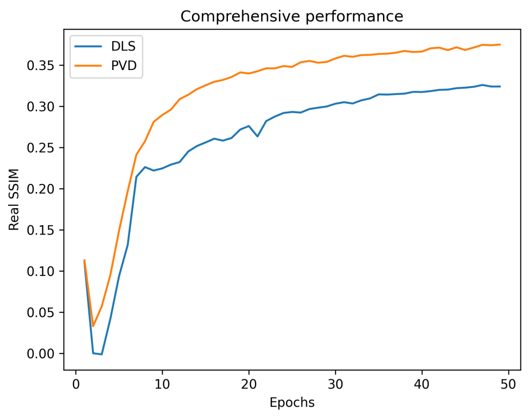


Figure 3: Comprehensive performance for each epoch.

variable for both VAE and GAN training. Considering the intermediate layer output of the discriminator as a feature encoder output, the generator of PVDGAN is trained to minimize perceptual VAE loss. Inference & backpropagation for perceptual VAE loss can be integrated into those for GAN training. PVDGAN does not require prior loss or variance estimation like VAE. The proposed method is kind of improved VAEGAN, and it improved the performance of DLSGAN.

References

[1] 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

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

[3] 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

[4] Z. Xiao, K. Kreis, A. Vahdat, "Tackling the Generative Learning Trilemma with Denoising Diffusion GANs," in International Conference on Learning Representations (ICLR) 2022. https://openreview.net/forum?id=JprM0p-q0Co

[5] 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

[6] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, O. Winther, "Autoencoding beyond pixels using a learned similarity metric" in Proceedings of The 33rd International

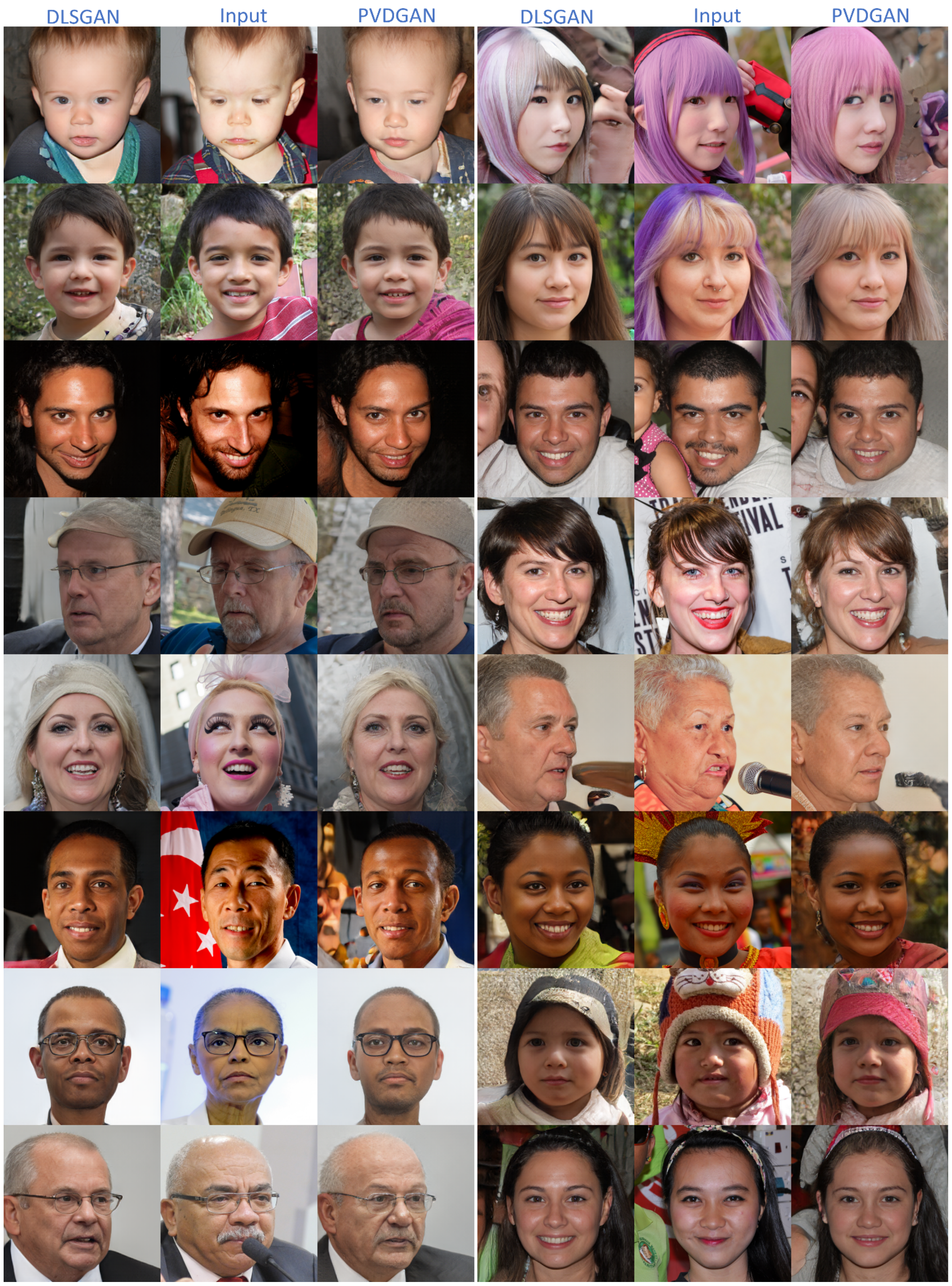


Figure 4: Test image reconstruction examples.

Conference on Machine Learning (PMLR), 2016. http://proceedings.mlr.press/v48/larsen16.html

[7] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, "Adversarial Autoencoders," in arXiv preprint, 2015. https://arxiv.org/abs/1511.05644

[8] 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

[9] J. Zhu, Y. Shen, D. Zhao, B. Zhou, "In-Domain GAN Inversion for Real Image Editing," in European Conference on Computer Vision (ECCV) 2020, pp. 592-608. https://link.springer.com/chapter/10.1007/978-3-030-58520-4_35

[10] M. Lucic, K. Kurach, M. Michalski, S. Gelly, O. Bousquet, "Are GANs Created Equal? A Large-Scale Study," in Advances in Neural Information Processing Systems 31 (NeurIPS 2018) 2018. https://papers.nips.cc/paper/2018/hash/e46de7e1bcaaced9a54f1e9d0d2f800d-Abstract.html

[11] 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

[12] L. Mescheder, A. Geiger, S. Nowozin, "Which Training Methods for GANs do actually Converge?," in Proceedings of Machine Learning Research (PMLR) 2018. https://proceedings.mlr.press/v80/mescheder18a.html

[13] 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

[14] T. Karras, T. Aila, S. Laine, J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," in International Conference on Learning Representations (ICLR) 2018. https://openreview.net/forum?id=Hk99zCeAb

[15] 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 Advances in Neural Information Processing Systems 30 (NIPS 2017). https://papers.nips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html

[16] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, T. Aila, "Improved Precision and Recall Metric for Assessing Generative Models," in Advances in Neural Information Processing Systems 32 (NeurIPS 2019). https://proceedings.neurips.cc/paper/2019/hash/0234c510bc6d908b28c70ff313743079-Abstract.html