Fast Transformation of Discriminators into Encoders using Pre-Trained GANs

Cheng Yu[a], Wenmin Wang[b,c,*]

[a] *Faculty of Information Technology, Macau University of Science and Technology, Macau 999078, China*
[b] *International Institute of Next Generation Internet, Macau University of Science and Technology, Macau 999078, China*
[c] *State Key Laboratory of Lunar and Planetary Sciences, Macau University of Science and Technology, Macau 999078, China*

Abstract

Finely tuned deep generative adversarial networks (GANs) can generate high-quality (HQ) images. However, the discriminator in GAN is only able to distinguish true or fake images. Moreover, numerous synthesized images from GANs are imperfect, and we can not reconstruct those images via GANs. In this paper, we revisit pre-trained GANs and offer a self-supervised method to quickly transform GAN's discriminators into encoders. We reuse parameters of the GAN's discriminator and replace its output layer, so that it can be transformed into an encoder and output reformed latent vectors. The transformation makes the pre-trained GAN a symmetrical architecture and allows for better performance. Based on the method, GANs can be made to reconstruct synthesized images via encoders. Compared to synthesized images, these reconstructions can maintain or even attain higher quality.

Keywords: Keywords:
Generative Adversarial Net (GAN)
Auto-Encoder
Image Synthesis
Image Reconstruction

1. Introduction

Since DCGAN [1] was proposed, using a convolutional neural network to implement generative adversarial networks (GANs) has become popular. However, with the increase in resolution, we need large learning parameters and large-scale HQ datasets as training samples. Recent novel GANs, such as PGGAN [2], StyleGAN [3] and so forth ([4, 5, 6]), can achieve stable training and produce effective results. These GANs take advantage of several techniques such as pixel normalization and equalized learning rates to control the upgrading rate for training-related parameters. Compared to DCGAN that requires doubling parameters for the next layer, the improvements mentioned above produce better GANs that can generate higher resolution images with fewer parameters. However, those GANs have no ability to reconstruct synthesized images. Furthermore, the above methods inevitably create many defective samples, with local details, blurred, and perturbed synthesized images.

A GAN consists of a generator and a discriminator. The generator utilizes low-dimensional latent vectors to generate high-dimensional images. The latent vectors are always sampled from a normal distribution. The discriminator has no use other than to assist in training the generator. In GAN inversion [7], embedding images in latent vectors is the process of mapping data from high dimensions to low dimensions. There are two types of methods to embed images in latent vectors. Previous work [8] tried to embed an image into its latent vector via perceptual loss [9], but the method does not use GAN encoders so getting the latent vector from each image entails a lot of training costs. Besides, other works [10, 11, 12] use GAN encoders to embed images. However, training GAN encoders also incurs numerous costs, especially in using deep GANs. We transformed GAN's discriminators to an encoder so GANs can quickly get their encoders.

So far, GAN encoders cannot do the embedding task well if there are not enough training sources, such as large-scale images and training devices. We propose an efficient method that can embed images in latent vectors and reconstruct images. Using pre-trained GANs, we quickly transform its discriminator into an encoder, which uses the self-generated images from the generator to replace ground-truth samples, so that we can allow GANs to reconstruct synthesized images via the transformed encoder. The encoder is slightly different from the discriminator (the last layer), and we reuse the discriminator's parameters so that the encoder can inherit the discriminator's features. In addition, although previous works (such as BigGAN [5], StyleGANv2 [6]) found that increasing model parameter size could make better performance, the importance of the model symmetry is omitted. The symmetry represent parameter size differ-

*Corresponding author.
Email address: wmwang@must.edu.mo (Wenmin Wang)

ence between generator and discriminator. In this regard, we explore the model performance when we increase the symmetries by adjusting the discriminator's last layer. Meanwhile discoving the transformation of discriminator to encoder. In conclusion, we describe our three contributions in the following way:

- We propose a novel approach to quickly transform the discriminator into an encoder. Using a self-supervised method with few training epochs (limited to 10 epochs), the transformed encoder can embed images into reform latent vectors. Using the reformed latent vectors, the performance of reconstructions is better than the synthesized images if GAN performance is limited.
- We change the parameters of the GAN's discriminator and allow its output size to equal the input size of the generator. Based on the change, we train the modified discriminator to get the GAN encoder. By reusing the parameters of the discriminator, the transformed encoder can inherit the features from pre-trained GAN and can be trained quickly.
- We analyze the relationship between the model performance and the model architecture, and we notice that when we increase the symmetries of the generative model, which consists of transformed encoder and generator, the modified model can improve the quality of the synthesized images, such as reducing the unclear areas in the synthesized image and improve the performance when measured by PSNR [13], SSIM [14], FID [15] and LPIPS [16].

2. Proposed Approach

From the perspective of training strategy, GAN is achieved by asynchronously training two networks: generator G and discriminator D. In the training process, we upgrade the parameters of D according to the true or fake labels. Fake labels represent G outputs (x') and True labels represent ground-truth (GT) images (y). Then we upgrade the parameters of G so that G can produce better fake images. Here, we denote the loss function of GAN as $\mathcal{L}_{(D,G)}$.

Different from GAN, training an auto-encoder model is a synchronous process that is composed of an encoder (E) and a decoder. Here, we regard the decoder as a generator (G) and try to transform D to E. Then we train E to match pre-trained GAN. In Bayesian probability model, such as variational auto-encoder (VAE) [17], E and G can also be denoted as $q(z|x)$ and $p(x|z)$. Both E and G need to be upgraded in the same once back-propagation. The input of the auto-encoder is y which samples from the ground truth (GT), and the output image is the corresponding reconstruction $G(z)$. Similar to VAE, we denote z as latent vectors. Here, we choose $z \sim \mathcal{N}(0, 1)$.

We briefly denote the loss function of vanilla VAE as $\mathcal{L}_{(E,G)}$ (see Eq.1). The first term is the distribution gap between y and $G(z)$, and that gap needs to be measured by the KL divergence. The second term is the reconstruction loss $\mathcal{L}_R$, which maximizes the expected log-likelihood of the image reconstruction.

$$\mathcal{L}_{(E,G)} = -D_{KL}(q(z|x)\|p(z)) + \underbrace{\mathbb{E}_q[\log p(x|z)]}_{Reconstruction-Loss}. \quad (1)$$

There are two latent vectors that we should compare for their similarity. One is the encoder's output ($E(\cdot)$), and the other one is the decoder's input (z). Motivated by [11], we deduce the gap between the z and $E(\cdot)$ when they have the same dimensions. The first term of Eq.1 can measure the gap between the above two latent vectors as follows:

$$\mathcal{L}_{\mathcal{R}1} \simeq \sum_{i=1}(1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2). \quad (2)$$

Here, we denote one dimension of z as z_i. On the dimension, we denote its mean as μ_i and its standard deviation as σ_i. By only using z, we aim to minimize the latent vector gap using a self-supervised method.

2.1. Self-Supervised Transformation

The loss function of the ordinary GAN is not adaptive for training the transformed auto-encoder, because it only judges true or fake labels. We modify the GAN-based architecture so that it is similar to VAE. However, if we use the VAE loss function, the KL divergence is based on the Bayesian probabilistic model, and cannot guide the details of the latent vector representation. At the same time, E cannot effectively reconstruct the wild image when the target is far from the training dataset. Inspired by [8, 18], we propose a self-supervised method to train E, replacing the log-likelihood (the 2nd term of Eq. 1) loss with mean square error (MSE) loss and perceptual loss [9] as follows:

$$\mathcal{L}_{\mathcal{R}2} = \mathcal{L}_{mse}(G(z), y) + \mathcal{L}_{pcp}(G(z), y). \quad (3)$$

Eq.3 is the standard loss function that measures $G(z)$ and GT samples y. We translate y as a self-supervised output, $G(E(G(z)))$. In this way, the optimized target (y) is replaced, and there is no need for any dataset training. The replacement is as follows:

$$\begin{aligned}\mathcal{L}_{\mathcal{R}2} = \mathcal{L}_{mse}(G(z), G(E(G(z))))\\ +\mathcal{L}_{pcp}(G(z), G(E(G(z))))\end{aligned}. \quad (4)$$

In addition, if we do not consider the specific value of $E(\cdot)$, it will be too far apart from z. Therefore, we assume a fixed mean $\mu = 0$ and variance $\sigma = 1$ for $E(\cdot)$, that is the key to improving efficiency. E should find a more suitable latent vector for image reconstruction, and images will be embedded in an overlapping space between $E(\cdot)$ and z. This regularization trick can reformulate Eq.2 as follows:

$$\mathcal{L}_{\mathcal{R}1} \simeq \mathcal{L}_{reg} = \frac{1}{n}\sum_{i=1}^{n}(E(G(z_i)) + E(G(z_i))^2) - 1. \quad (5)$$

For pre-trained GANs, we do not know the specific value of μ and σ, so we use MSE to optimize the latent vector which

Figure 1 An illustration for our proposed method (Case of PGGAN 1024×1024). By adjusting the output channels in the last layer of the discriminator, we transform the discriminator into an encoder, and then we train the encoder corresponding to the fixed generator. The green-dashed arrows indicate the process of GAN, and the solid black and green arrows indicate our process.

comes from E output. Eq.5 can be replaced by the following loss function:

$$\mathcal{L}_{\mathcal{R}1} = \frac{1}{n}\sum_{i=1}^{n}(E(G(z_i)) - z_i)^2. \tag{6}$$

Finally, we briefly summarize the final loss function in Eq. 7, which replaces the original auto-encoder loss function as follows:

$$\mathcal{L}_E = \mathcal{L}_{\mathcal{R}1} + \mathcal{L}_{\mathcal{R}2}. \tag{7}$$

Fig.1 demonstrates the principles of transforming a pre-trained discriminator into an encoder.

2.2. *Fine-tuning Networks to Symmetrical Architecture*

In normal GANs, the usage of D is only to classify y and $G(z)$, and its output size usually is a one-dimensional value (the true or fake label), which is smaller than G input size z. For example, in PGGAN, the size of z is 512 which is needed to represent an HQ image with size (1024,1024). Transforming D to E is our goal, so we fix pre-trained G and only train the transformed E. In order to make G and E form a symmetrical encoding-decoding architecture, E output size (E_{output}) needs to equal G input size (G_{input}). The transformation means that we add more parameters to the output layer of D. Following the increased parameters, we increase the symmetries of the transformed model, which is composed of G and E. We achieve the transformation by replacing the output layer of D (D_{output}). In each layer, we focus on the data input size and output size (input, output), and we only address the D output layer with its output size. In the full connected layer (FC), we increase the output nodes. In the convolutional layer (CONV), we increase its output channels. The modified D_{output} for different GANs are listed in Table 1. We use a 3-layer fully connected architecture (3-FC) and a 5-layer convolutional architecture (5-CONV) to process 28×28 images. As it happens, DCGAN deals with 256× 256 images, PGGAN deals with 1024×1024 images, and PGGAN-FC deals with 256$times$256 images.

Table 1
Details of transforming D to E by changing output layers (default as CONV).

Layer Position:	G_{input}	D_{output}	E_{output}
3-FC	$(784, 2048)_{FC}$	$(2048, 1)_{FC}$	$(2048, \mathbf{784})_{FC}$
5-CONV	(32, 512)	(512, 1)	(512, **32**)
DCGAN	(128, 2048)	(2048, 1)	(2048, **128**)
PGGAN	(512, 512)	(512, 1)	(512, **512**)
PGGAN-FC	(512, 512)	$(32768, 1)^*_{FC}$	$(32768, \mathbf{512})_{FC}$

* In PGGAN-FC, D_{output} is the last block which has two fully connected layers. The number of their nodes $(input, output)_{FC}$ are $(32768, 4)_{FC}$ and $(4, 1)_{FC}$.

Figure 2 The 1st row shows the face images that are generated by pre-trained PGGAN (1024×1024). The 2nd row shows corresponding reconstructions via our method. Most reconstructions are better than the previous images in local details. The 3rd row shows 3 pairs of wild images (left) with their reconstructions (right).

2.3. Sharing Discriminator Parameters with Encoders

A recent work [19] reuses the pre-trained parameters in the discriminator to handle the task of image-to-image style translation. Similar to Cycle-GAN [20], it needs another network to form a double model to deal with the image-to-image translation task. The difference from this work is that we only reuse the parameters to boost the speed of training convergence. We do not use the double model which contains a pair of networks. Algorithm 1 shows the pseudo-code of the whole proposed method. In Fig.2 and Fig. 6, we display pre-trained PGGAN samples and corresponding reconstructions.

Algorithm 1:
Transforming discriminator into encoder

Input:
Pre-trained GAN: G and D.
Latent Vector: $z \sim \mathcal{N}(0, 1)$.
Learning Rate: $\alpha = 0.0015$.
Initialize:
$E = D$, $E_{output} = G_{input}$, $w_E = w_D$.
while *(no more 10 epochs & not converged)* **do**
  $\nabla\mathcal{L}_{R_1} \leftarrow \mathcal{L}_{R_1}(G(z), G(E(G(z))))$,
  $\nabla\mathcal{L}_{R_2} \leftarrow \mathcal{L}_{R_2}(E(G(z)), z)$,
  $w_E \leftarrow w_E + \alpha(\nabla\mathcal{L}_{R_1} + \nabla\mathcal{L}_{R_2})$.
end
Output: E.

3. Experiments

We used three datasets for doing experiments. The first dataset is Fashion-MNIST. We used 60.000 training samples with image size 28×28, and we set the batch size at 128. The second dataset is CelebA [21]. There are 202,599 face images and we choose 30,000 samples for training, we resized images to 256×256 with a batch size of 30. The last dataset is an HQ dataset (CelebA-HQ [22]). There are 30,000 face images with image size 1024×1024. Here, we only used CelebA-HQ for evaluation. The framework is PyTorch (version 1.5.1, CUDA 10.2) on a GPU Card (Nvidia Tesla V100-SXM3 32GB). We chose the Adam optimizer with a learning rate of 0.0015, β_1 = 0.5 and β_2 = 0.99.

Table 2
Comparison of different architectures (E) via Fashion-MNIST

Model Layers (E):	1-FC	3-CONV	3-FC	5-CONV
PSNR(±0.5)	15.18	16.95	15.36	**17.98**
SSIM(±0.1)	0.52	0.66	0.64	**0.71**

3.1. Comparison between Different Architectures

In Fashion-MNIST, we used 5 convolutional layers (5-CONV) to build G and D, and used different numbers of fully connected layers and convolutional layers to construct E. These architectures included one fully connected layer (1-FC), 3 convolutional layers (3-CONV), 3 fully connected layers (3-FC), and 5 convolutional layers (5-CONV). When we chose different architectures to build E for auto-encoding with G, the reconstructed images by reformed latent space were not much different. We report the evaluation metrics in Table 2. The reconstructed 5,000 images were evaluated by PSNR and SSIM after training for one epoch. We noticed that the performance has improved when we increased the model symmetries, even

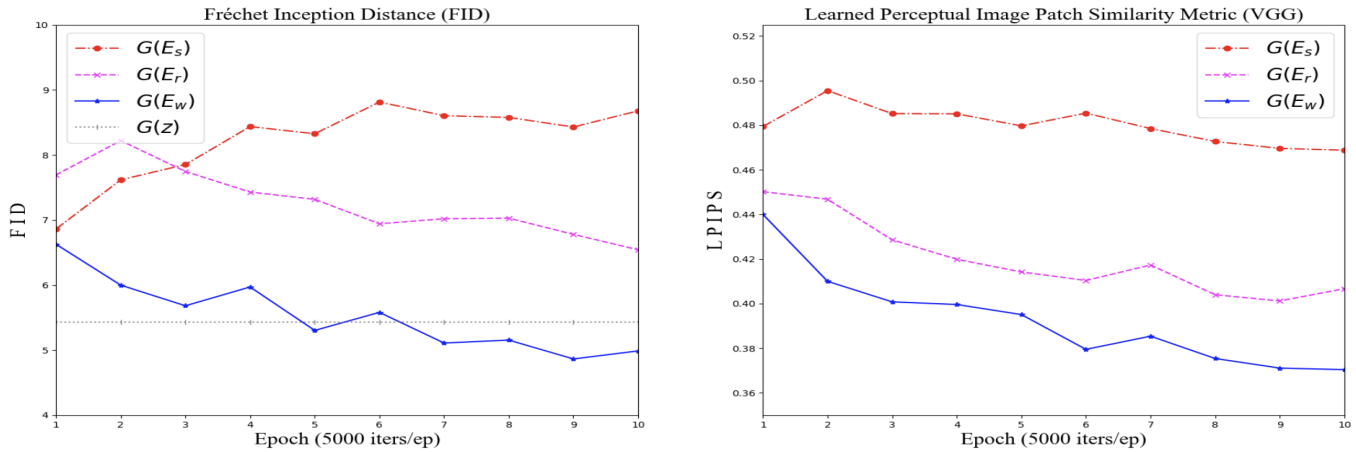

Figure 3 Ablation study of training E_s, E_n and E_p with progressive epochs. E_s is a small architecture which has half layer parameters of D. E_n is the same parameter size as D but without reusing parameters of D. E_p is same parameter size with E_n and reuse parameters of D.

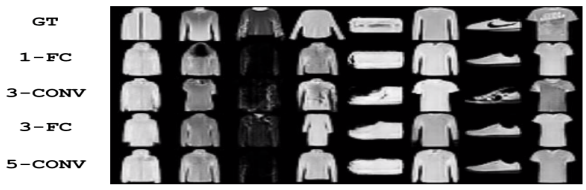

Figure 4 Comparison of Image reconstructions during different E (Fashion-MNIST).

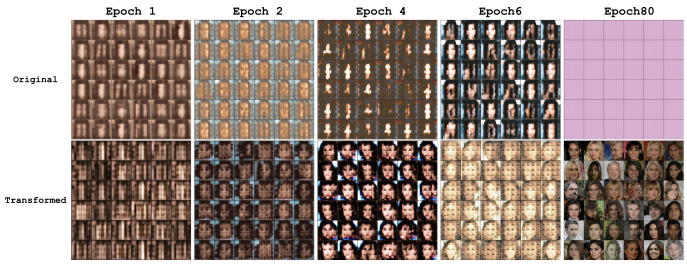

Figure 5 Training comparison in DCGANs (D with transformed D).

though the improvement is not obvious to human perception (see Fig. 4).

We chose DCGAN to evaluate the symmetrical architecture on CelebA. Different from vanilla DCGAN, we replaced batch normalization with spectral normalization [23] on D. Based on the replacement, D satisfies Lipschitz continuity and makes training more stable. Compared with the previous size 64×64, the modification can make DCGAN handle 256×256 images, but it is still difficult to train. So, we tested two architectures on 256×256 images. As shown in Table 1, D output channel is a one-dimensional label for classifying the sample's fake and truth labels. In transformed D (we also call it E), we increased the last layer parameters by changing its output channels, and then E output size equal to the input size of G (see Table 1). We trained the two architectures via the GAN loss function ($\mathcal{L}_{(D,G)}$) without auto-encoding training. The result has shown that D training fails when the training epoch increases, but E can make the training more effective. We report the training process in Fig. 5.

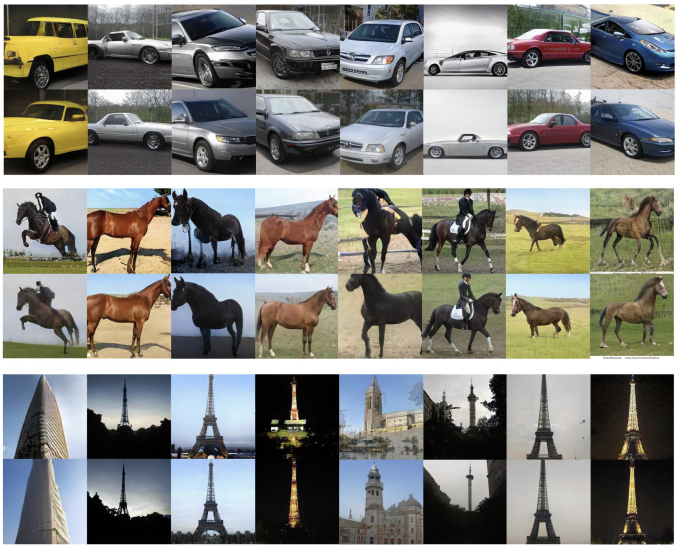

Figure 6 Pre-trained PGGAN-FC samples (ups) and corresponding reconstructions (downs). Results come from LSUN (car, horse and tower), with image size 256×256.

3.2. *Self-Supervised Training for the Encoder*

Using CelebA-HQ, we chose 10,000 real samples for evaluation. To train E, we synthesized 20,000 generated images from pre-trained PGGAN which is limited performance. We trained E with batch size 4. As shown in Fig.2, the pre-trained samples still had many distortions, blurs, and blobs in local details. The first row shows the generated images by PGGAN. Our transformed method is shown in the second row. By encoding the

reformed latent space, our method samples ($G(E(G(z))))$ were better than the pre-trained model samples ($G(z)$). On the LSUN dataset (car, tower, and horse) [24], we report more cases in Fig. 6 with pre-trained PGGAN-FC and our reconstructions.

3.3. Reusing Parameters with Symmetric Architecture

Table 3
Performance of pre-trained PGGAN and transformed architectures

	PSNR↑	Metrics		
		SSIM↑	LPIPS^{VGG} ↓ (±0.01)	FID↓ (±0.1)
$G(z)$	–	–	–	5.43
$G(E_s)$	59.89 ± 0.52	0.9915	0.4640	6.82
$G(E_n)$	61.38 ± 0.58	0.9942	0.3498	6.77
$G(E_p)$	61.73 ± 0.82	**0.9947**	**0.2997**	**4.86**

To evaluate reusing parameters, we designed three different encoders based on PGGAN's E (E_s, E_n and E_p). PGGAN's network needs 9 blocks from 4×4 to 1024×1024. Here, the output block of the three encoders is the same. As for the other blocks, E_s has half layer parameters of D (two-layer block to one-layer block), and E_n is the same parameter size as D with no reused D's parameters. E_p has the same parameter size as E_n but reuses D's parameters. We report the experimental results in Table 3. In FID, we compared three encoders with GT (10,000 samples) and with $G(z)$ in LPIPS (2,000 samples). All results were obtained in the 10th epoch.

We also compared three encoders during the training process. As shown in Fig 3, in the early epochs, FID of E_s and E_n are slightly higher than the baseline $G(z)$. With the epoch increase, E_p is better than E_s and E_n, and converges faster. LPIPS of E_p is also better than others. This verified our intuitive view that a transformed E would be better when we reused D parameters and increased the model symmetries.

4. Conclusion

We have offered a novel approach for quickly transforming a discriminator into an encoder via a pre-trained GAN, in which we adjusted the parameters of the discriminator output layer to the same size as the generator input layer. We used a self-supervised method to train the reformed encoder. By reusing the parameters and increasing the networks' symmetries, our proposed scheme quickly yielded an efficient encoder that enhances the performance of latent space representation and image reconstruction.

Declaration of Competing Interest

We confirm that the current version of the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed.

We further confirm that the order of authors listed in the manuscript has been approved by all of us. We also wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgments

This work was supported by the Science and Technology Development Fund (FDCT) of Macau (0016/2019/A1). Our code and pre-trained models are available upon request.

References

[1] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, in: *Int. Conf. Learn. Represent. (ICLR)*, 2016.

[2] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of gans for improved quality, stability, and variation, in: *Int. Conf. Learn. Represent. (ICLR)*, 2018.

[3] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019.

[4] A. Karnewar, O. Wang, Msg-gan: Multi-scale gradients for generative adversarial networks, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.

[5] A. Brock, J. Donahue, K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, in: *Int. Conf. Learn. Represent. (ICLR)*, 2019.

[6] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, T. Aila, Analyzing and improving the image quality of stylegan, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 8107–8116.

[7] W. Xia, Y. Zhang, Y. Yang, J. Xue, B. Zhou, M. Yang, GAN inversion: A survey, arXiv preprint abs/2101.05278.

[8] R. Abdal, Y. Qin, P. Wonka, Image2stylegan: How to embed images into the stylegan latent space?, in: *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 4431–4440.

[9] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: *Europ. Conf. Comput. Vis. (ECCV)*, 2016, pp. 694–711.

[10] J. Donahue, P. Krähenbühl, T. Darrell, Adversarial feature learning, in: *Int. Conf. Learn. Represent. (ICLR)*, 2017.

[11] A. Creswell, A. A. Bharath, Inverting the generator of a generative adversarial network, IEEE Trans. Neural Networks Learn. Syst. 30 (7) (2019) 1967–1974.

[12] S. Pidhorskyi, D. A. Adjeroh, G. Doretto, Adversarial latent autoencoders, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.

[13] A. Horé, D. Ziou, Image quality metrics: Psnr vs. ssim, in: *Proc. Int. Conf. Pattern Recognit. (ICPR)*, Istanbul, 2010, pp. 2366–2369.

[14] Zhou Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, *IEEE Trans. Image Process.* 13 (4) (2004) 600–612.

[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: *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6626–6637.

[16] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018.

[17] D. P. Kingma, M. Welling, Auto-encoding variational bayes, in: *Int. Conf. Learn. Represent. (ICLR)*, 2014.
URL `http://arxiv.org/abs/1312.6114`

[18] J. Zhu, P. Kr, E. Shechtman, A. A. Efros, Generative visual manipulation on the natural image manifold, in: *Europ. Conf. Comput. Vis. (ECCV)*, 2016, pp. 597–613.

[19] R. Chen, W. Huang, B. Huang, F. Sun, B. Fang, Reusing discriminators for encoding: Towards unsupervised image-to-image translation, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 8165–8174.

[20] J.-Y. Zhu, T. Park, P. Isola, A. A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2223–2232.

[21] Z. Liu, P. Luo, X. Wang, X. Tang, Deep learning face attributes in the wild, in: *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 3730–3738.

[22] C.-H. Lee, Z. Liu, L. Wu, P. Luo, Maskgan: Towards diverse and interactive facial image manipulation, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 5549–5558.

[23] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, in: *Int. Conf. Learn. Represent. (ICLR)*, 2018.

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