Quickly Transforming Discriminator in Pre-Trained GAN to Encoder

Cheng Yu, Wenming Wang, *Member, IEEE*

***Abstract*—Fine-designed deep Generative Adversarial Networks (GANs) can generate high-quality (HQ) images. However, the discriminator in GAN only plays a role to distinguish candidates produced by the generator from the true data distribution, and numerous generated samples are still not clear and true. From pre-trained GAN, we offer a self-supervised method to quickly transform the discriminator into an encoder and fine-tune the pre-trained GAN to an auto-encoder. The parameters of the pre-trained discriminator are reused and converted into an encoder for outputting reformed latent space. The transformation changes the previous GAN to a symmetrical architecture and the generator can reconstruct the HQ image by reforming latent space. By fixing the generator, the reformed latent space can perform better representation than the pre-trained GAN, and the performance of the pre-trained GAN can be improved by the transformed encoder.**

***Index Terms*—Generative Adversarial Net (GAN), Auto-Encoder, Latent Space Representation, Image Reconstruction.**

I. INTRODUCTION

SINCE DCGAN [1] was proposed, using a convolutional neural network to implement the generative model becomes popular. However, with the increase of resolution, we need tremendous 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]), achieve stable training and effective results. These GANs take advantage of several techniques such as pixel normalization and equalized learning rate to control the upgrading rate for training-related parameters. Compared with DCGAN that needs doubling parameters to the next layer, the improvements above make better GAN that can generate higher resolution images in fewer parameters. However, those GANs have no ability to reconstruct images via latent variables. Actually, the methods above inevitably bring out many defective samples, with local details, blurred and perturbed generations.

GAN always utilizes low-dimensional latent space to generate high-dimensional images. In inverting process, embedding images to latent space is similar to the mapping from high dimensional data to low dimensional manifold. There are two types of methods to embed images to latent space. Previous work [7] tries to embed images to latent space and searches the corresponding latent space via perceptual loss [8], but the method does not use auto-encoder so that it needs to optimize every target. Besides, other works [9], [10], [11], [12] use auto-encoders to embed images, even though the reformed latent space produced by the encoder can not be reconstructed well, especially for HQ images, and the method always brings the problem of space entanglement.

So far, the auto-encoder can not do the embedding task well. There's still plenty of room for improvement. We propose a method to embed and reconstruct HQ images. Using pre-trained GAN, 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 improve the generative model's ability to reconstruct images and represent latent space. Meanwhile, the discriminator in the pre-trained GAN is slightly modified. We reuse the discriminative parameters to inherit the features of the pre-trained GAN and explore the model performance if we increase the symmetries of the model architecture and parameters. In conclusion, we highlight our contributions in the following three parts:

1. In pre-trained GANs, such as DCGAN and PGGAN, we propose a novel approach to quickly transform the discriminator into an encoder. Using a self-supervised manner with few training epochs (limited to 10 epochs), the transformed encoder can embed images into a new latent space. And using the new latent space, the performance of the transformed model is better than the baseline of pre-trained GAN.

2. We change the parameters located in the discriminator output layer, and let the output size of the discriminator is equal to the input size of the generator. On this basis, we train the modified discriminator to obtain a transformed encoder. By reusing the parameters of the pre-trained discriminator, the transformed encoder can inherit the features from pre-trained GAN and training quickly.

3. We analyze the relationship between the model performance and the model architecture, and we notice that when we increase the symmetries of the model which consists of transformed encoder and generator. The modified model can improve the quality of the generated images. Finally, the above improvements reduce the chaotic area in the generated image and improve the generative performance measured by PSNR [13], SSIM [14], FID [15] and LPIPS [16].

II. PROPOSED APPROACH

From the perspective of model training strategy, GAN is achieved by asynchronously training two networks: generator

---

This work was supported by the Science and Technology Development Fund (FDCT) of Macau (0016/2019/A1).

Cheng Yu is with the Faculty of Information Technology, Macau University of Science and Technology, Macau 999078, China, and also with Chongqing College of Electronic Engineering, Chongqing 401331, China. (E-mail: disanda@outlook.com)

Wenming Wang is the corresponding author, with the International Institute of Next Generation Internet, and also with the State Key Laboratory of Lunar and Planetary Sciences, Macau University of Science and Technology, Macau 999078, China. (E-mail: wmwang@must.edu.mo.)

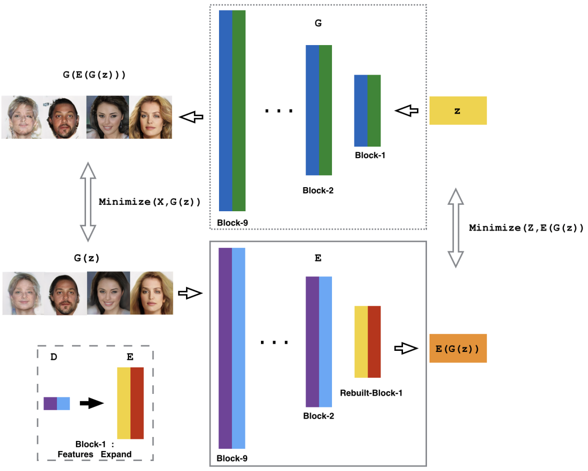

Fig. 1. An illustration for our proposed method: after increasing parameters in the discriminator last layer, we transform the discriminator into an encoder, and then we train the encoder corresponding to the fixed generator.

G and discriminator D. In each step, we upgrade the parameters of D according to the loss gap between the G outputs and ground-truth (GT) samples x. Then we upgrade the parameters of G according to the descent direction which comes from D outputs.

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. 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 target images x which sample from ground truth (GT), and the output results are generated images $G(z)$. Similar to VAE, we denote z as multi-dimensional random variables. Here, we choose $z \sim \mathcal{N}(0,1)$.

We briefly denote the loss function of vanilla VAE as Eq.1. The first term is the distribution gap between GT samples and generations that needs to be measured by the KL divergence. The second term is the reconstruction loss $\mathcal{L}_R$, which maximizes the expected log-likelihood for image generation.

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

Generalized latent spaces are located inside of the overall generative model and generated by the corresponding layers. There are two commonly used latent spaces. One comes from the encoder's output layer, $E(\cdot)$, and another one comes from the decoder's input layer, z. Motivated by [10], we deduce the gap between the z and $E(\cdot)$ when they have the same dimensional size. The first term of Eq.1 can measure the gap between the two latent spaces as follows:

$$D_{KL} \simeq \sum_{i=1}(1+\log(\sigma_i^2)-(\mu_i^2)-(\sigma_i^2)). \quad (2)$$

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 space gap in a self-supervised manner.

A. *Self-Supervised Transforming*

The loss function of original GAN is not adaptive for training the transformed auto-encoder. We modify the model architecture which similar to VAE. However, KL divergence is based on the Bayesian probabilistic model, and can not guide the details of the latent space representation. Meanwhile, E cannot effectively reconstruct the wild image when the target is far from the training dataset. Inspired by [7], [18], we propose a self-supervised way to transform E, replacing the log-likelihood loss with mean square error (MSE) loss and perceptual loss [8] as follows:

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

Eq.3 is the standard loss function that measures $G(z)$ and GT samples x. We translate x as a self-supervised output, $G(z)$, so that the gap between generations and its reconstructions can be reformulated as $G(z)$ and $G(E(G(z)))$. Finally, the optimized target is replaced, and there is no need to input x. The replacement as follows:

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

In addition, if we do not control the latent space values for E output, the two latent spaces (z and $E(\cdot)$) will be too far apart to embed images. Therefore, we assume a fixed mean $\mu = 0$ and variance $\sigma = 1$ to optimize $E(\cdot)$. That is the key to improve encoding efficiency. E should find a more suitable space for image embedding. The images will be embedded in an overlapping space. This change also improves the reconstruction performance, alleviating the coupling of latent variables. This regularization trick can reformulate Eq.2 as follows:

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

For pre-trained GANs, we usually do not know the specific value of μ and σ. So we use MSE to optimize the potential space which comes from E output. Eq.5 can be replaced by the following loss function:

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

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

$$\mathcal{L}_E = \mathcal{L}_{mse} + \mathcal{L}_{pcp} + \mathcal{L}_{reg}. \quad (7)$$

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

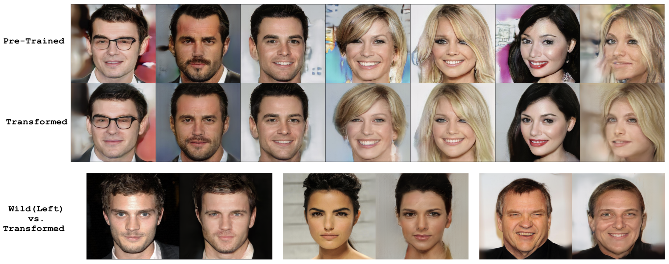

Fig. 2. The 1st row shows the face images that are generated by pre-trained PGGAN. The 2nd row shows corresponding reconstructions via our method. Most reconstructions are better than their previous images in local details. The 3rd row shows 3 pairs of wild images (left) with their reconstructions (right).

B. Fine-tune Networks to Symmetrical Architecture

In common GANs, the vocation of D is only to classify x and $G(z)$, and its output size usually is one dimension, which is smaller than G input size. 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 transforming 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 transforming by replacing the output layer of D (D_{output}). In each layer, we focus on the data input dimensions and out dimensions (input, output), and we only address the D output layer with its output dimensions. In the full connected layer (FC), we increase output nodes. In the convolutional layer (CONV), we increase its output channels. The modified D_{output} for different GANs are listed in Tab. I. We use 3-layer fully connected architecture (3-FC) and 5-layer convolutional architecture (5-FC) to process 28×28 images. Besides, DCGAN deals with 256× 256 images and PGGAN deals with 1024×1024 images.

TABLE I
DETAILS OF TRANSFORMING D TO E, BY CHANGING OUTPUT LAYER.

Layer Position:	G_{input}	D_{output}	E_{output}
3-FC	$(784, 2048)_{FC}$	$(2048, 1)_{FC}$	$(2048, \mathbf{784})_{FC}$
5-CONV	(32, 512)	(64, 1)	(64, **32**)
DCGAN [1]	(128, 2048)	(32, 1)	(32, **128**)
PGGAN [2]	(512, 512)	$(32768, 1)^{*}_{FC}$	$(32768, \mathbf{512})_{FC}$

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

C. Sharing Discriminator Parameters to Encoders

A recent work [19] reuses the pre-trained parameters in the discriminator to handle the task of image-to-image style translation. Similar with Cycle-GAN [20], it needs another network to form a double model to deal with the image-to-image translation task. The difference with this work is that we only reuse the parameters to boost the speed of training convergence. We do not use more pairs of networks. Algorithm 1 shows the pseudo-code of the whole proposed method. In Fig.2 and Fig. 5, we display pre-trained PGGAN samples and corresponding reconstructions.

```
Algorithm 1: Transforming discriminator to encoder
Input: Noise: $z \in \mathbb{R}^m$. GT: $x \in \mathbb{R}^n$, $m \ll n$.
Pre-trained GAN: $G$, $D$. where
  $D(x) \simeq 1, D(G(z)) \simeq 0$.
Learning Rate: $\alpha = 0.0015$
Output: Transformed $E$ satisfied with:
        $E(G(z)) \simeq z, G(E(x)) \simeq x$.
Initialize:
1. $W_{E_{output}} = W_{G_{input}}$ (Replace E's output layer),
  $E(\cdot) = z', z' \in \mathbb{R}^1 \to z' \in \mathbb{R}^m$.
2. $W_E = W_D$ (Re-using parameters).
while (no more 10 epochs & not converged) do
    $\nabla\mathcal{L} \leftarrow \mathcal{L}_{mse,pcp}(G(z), G(E(G(z))))$,
    $\nabla\mathcal{L} \leftarrow \mathcal{L}_{reg}(E(G(z)), z)$, $w_E \leftarrow w_E + \alpha\nabla\mathcal{L}$.
end
```

III. EXPERIMENT

We use three datasets for experiments. The first dataset is Fashion-MNIST. we use 60.000 training samples with image size 28×28, and we set batch size 128. The second one is CelebA [21]. There are 202,599 face image and we choose 30,000 samples to training, we resize image to 256×256 with batch size 30. The last one is HQ dataset CelebA-HQ [22]. There are 30,000 face images with image size 1024×1024. Here, we only use 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 choose Adam optimizer with a learning rate of 0.0015, β_1=0.5, β_2=0.99 and ϵ = 1e-8 to implement all above experiments.

A. Comparison between Different Architectures

In Fashion-MNIST, we use 5 convolutional layers (5-CONV) to build G and D, and use different numbers of fully connected layers and convolutional layers to construct E. These architectures include one-layer fully connected layer (1-FC), 3 convolutional layers (3-CONV), 3 fully connected layers (3-FC), and 5 convolutional layers (5-CONV). When we choose different architectures to build E for auto-encoding with G, the reconstructed images by reformed latent space are not much different. We report the evaluation metrics in Tab. II. The reconstructed 5,000 images are evaluated by PSNR and SSIM after training one epoch. We notice that the performance has improved when we increase the model symmetries, even though the improvement is not obvious in human perception.

TABLE II
COMPARISON OF DIFFERENT STRUCTURES (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**

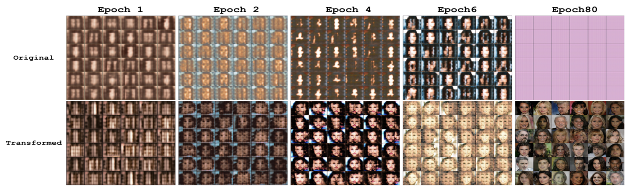

Fig. 3. Training comparison in DCGAN (original D and transformed D).

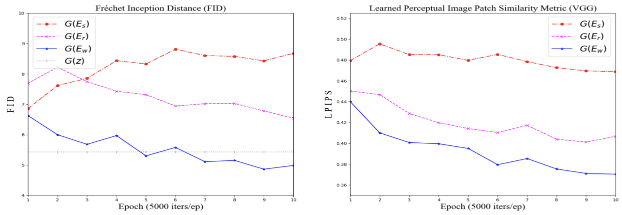

Fig. 4. Ablation study of training E_s, E_n and E_p with progressive epochs.

We choose DCGAN to evaluate the symmetrical architecture on CelebA. Different from vanilla DCGAN, we replace 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 test two architectures on 256×256 images. As shown in Tab. I, original D output channel is one dimension for classifying the sample's fake and truth. In modified D (we call it E), we increased the last layer parameters by changing its layer channels, and then E output size equal to the input size of G (see Tab. I). We train the two architectures separately without the pre-training process. The result has shown that the vanilla output layer training fails when the training epoch increases, but E can make the training more effective. We report the training process in Fig. 3.

B. Self-Supervised Training for Encoder Transform

On CelebA-HQ, we choose 10,000 samples for evaluation. To train E, we use 20,000 generated images from pre-trained PGGAN (with batch size 4). As shown in Fig.2, the pre-trained samples still have many distortions and blur blobs in local details. The first row shows the generated images by PGGAN. Our transformed method has shown in the second row. By encoding the reformed latent space, our method samples ($G(E(G(z)))$) are better than the pre-trained model samples ($G(z)$). On LSUN dataset (car, tower and horse) [24], we report more cases on Fig. 5 with pre-trained PGGAN and our reconstructions.

C. Reusing Parameters with Symmetric Architecture

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 Tab. III. In FID, we compare three encoders with GT (10,000 samples), and compare with $G(z)$ in LPIPS (2,000 samples). All results are obtained in the 10th epoch.

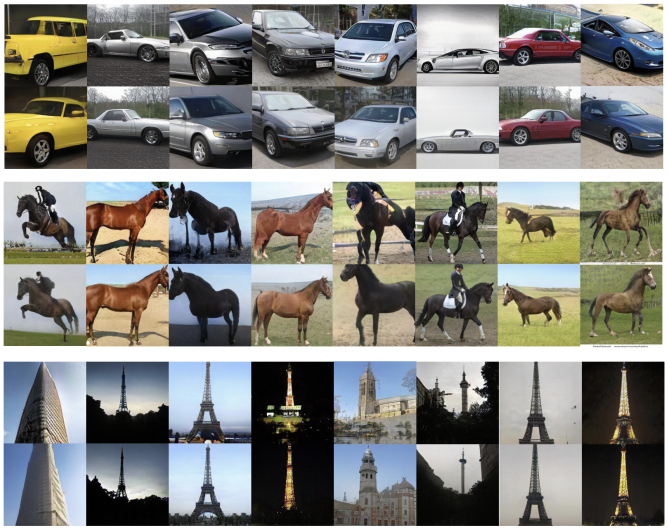

Fig. 5. Pre-trained PGGAN samples (ups) and corresponding reconstructions (downs). Dataset chooses from LSUN (car, horse and tower) and all image sizes are 256×256.

TABLE III
PERFORMANCE OF PRE-TRAINED MODEL AND TRANSFORMED MODELS

	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**

We also compare three encoders during the training process. As shown in Fig 4, 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 verifies our intuitive view that a transformed E will be better when we reuse D parameters and increase the model symmetries.

IV. CONCLUSION

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

REFERENCES

[1] A. Radford, L. Metz, and 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, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” in *Int. Conf. Learn. Represent. (ICLR)*, 2018.

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

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

[5] A. Brock, J. Donahue, and 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, and T. Aila, “Analyzing and improving the image quality of stylegan,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 8107–8116.

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

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

[9] J. Donahue, P. Krähenbühl, and T. Darrell, “Adversarial feature learning,” in *Int. Conf. Learn. Represent. (ICLR)*, 2017.

[10] A. Creswell and A. A. Bharath, “Inverting the generator of a generative adversarial network,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 7, pp. 1967–1974, 2019.

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

[12] N. Yang, M. Zhou, B. Xia, X. Guo, and L. Qi, “Inversion based on a detached dual-channel domain method for stylegan2 embedding,” *IEEE Signal Processing Letters*, vol. 28, pp. 553–557, 2021.

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

[14] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[15] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and 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, and 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 and M. Welling, “Auto-encoding variational bayes,” in *Int. Conf. Learn. Represent. (ICLR)*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6114

[18] J. Zhu, P. Kr, E. Shechtman, and 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, and 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, and 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, and 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, and 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, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” in *Int. Conf. Learn. Represent. (ICLR)*, 2018.

[24] 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,” *arXiv preprint*, vol. abs/1506.03365, 2015.