Posterior Promoted GAN with Distribution Discriminator for Unsupervised Image Synthesis

Xianchao Zhang, Ziyang Cheng, Xiaotong Zhang*, Han Liu
School of Software, Dalian University of Technology
Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province
xczhang@dlut.edu.cn, chengziyang2@mail.dlut.edu.cn
zxt.dut@hotmail.com, liu.han.dut@gmail.com

Abstract

Sufficient real information in generator is a critical point for the generation ability of GAN. However, GAN and its variants suffer from lack of this point, resulting in brittle training processes. In this paper, we propose a novel variant of GAN, Posterior Promoted GAN (P^2GAN), which promotes generator with the real information in the posterior distribution produced by discriminator. In our framework, different from other variants of GAN, the discriminator maps images to a multivariate Gaussian distribution and extracts real information. The generator employs the real information by AdaIN and a latent code regularizer. Besides, reparameterization trick and pretraining is applied to guarantee a stable training process in practice. The convergence of P^2GAN is theoretically proved. Experimental results on typical high-dimensional multi-modal datasets demonstrate that P^2GAN has achieved comparable results with the state-of-the-art variants of GAN on unsupervised image synthesis.

1. Introduction

Generative Adversarial Network (GAN) [13] has been widely developed since it was proposed, and its variants have made remarkable progresses in the field of image synthesis. Sufficient real information in generator is the key point for such models. The only real information contributing to generator in original GAN is the gradient transferred from discriminator, which is indirect and jejune for generator, leading to brittle training. Some supervised variants of GAN [32, 34, 8, 26, 12] use the attributes like labels in a direct and strong way with some constraints on the attributes, which supplies sufficient real information to generator. For unsupervised variants of GAN, there are two major ways to improve the real information in generator. One approach is to provide more refined gradient from discriminator by *introducing new loss functions* [15, 3, 31, 11, 46, 41]. But the information from gradient is still indirect and frangible, which cannot supply a continuous support for generator. The other approach is to enrich the real information by *adding priors into input noise* [4, 6, 10, 40, 47, 25, 44, 9, 28]. Nevertheless, the real information in prior is only added into input noise, which may be decayed by stacking convolutional layers gradually.

To tackle the above lack of real information issue, in this paper, we propose a novel framework of GAN called Posterior Promoted GAN (P^2GAN). Firstly, unlike traditional discriminator that cannot provide sufficient real information, P^2GAN learns abundant real information in the posterior distribution by discriminator, and applies the information to generator in a more straight and robust way. In specific, our discriminator maps images to a multivariate Gaussian distribution, and matches this distribution with two different given prior distributions for distinguishing. The discriminator structure is inspired by the encoder of Variational Autoencoder (VAE) [23]. Secondly, unlike previous works [4, 6, 10, 40, 47, 25, 44, 9, 28] which apply a prior distribution to the input noise, our generator is promoted by introducing the external real information into each convolutional layer via Adaptive Instance Normalization (AdaIN) [18] and a latent code regularizer. Directly affecting the features in convolutional layers by AdaIN is more powerful than only changing the prior of the input noise. Thirdly, we change the losses of Least Squares Generative Adversarial Network (LSGAN) [31] to a distribution version, and provide theoretical guarantees for its convergence. Besides, we employ reparameterization trick for reasonably calculating the losses at the implementation level and a pretraining step for initialization.

Our contributions are summarized as follows. First, we propose a novel GAN framework called P^2GAN. The discriminator outputs a posterior distribution for distinguish-

*Corresponding author.

ing real and fake, and extracts plentiful real information in the meantime. The generator utilizes the real information via AdaIN layers to promote its generation ability. Second, we modify the losses of LSGAN to a distribution version, and theoretically prove its convergence via the Pearson $\mathcal{X}$ divergence. Third, we propose a new latent code regularizer as a robust constraint of generator, which can prevent the abundant real information in the posterior distribution from fading away during training. Fourth, we introduce two methods to guarantee model stability in practice: the reparameterization trick training and the pretraining step. Our model produces a state-of-the-art performance on multiple image datasets in a fully unsupervised way.

2. Related Works

Generative Adversarial Network (GAN) uses the generator G and the discriminator D for adversarial learning. G inputs random noise $\boldsymbol{\epsilon}$ and produces generated images. D receives the real or the generated images, and tries to classify between them.

In order to improve GAN, some works introduce new loss functions for obtaining refined gradient from discriminator, e.g., WGAN [3] and WGAN-GP [15] use Wasserstein distance as adversarial losses to avoid gradient vanishing and brittle training. To precisely estimate the realness of an input sample, RealnessGAN [46] expands the scalar realness score of the original GAN into a distributional one. Some works increase real information by changing the prior of the input noise. InfoGAN [6] decomposes the input noise into an incompressible part and a structured semantic feature part artificially. [40] and [10] extend InfoGAN by proposing a reverse reconstructor network which maps the images to a random Gaussian latent code as input noise. [47] and [4] introduce a mixture of Gaussian distributions for sampling input noise. [25] proposes a family of distributions—Tensor Ring Induced Prior (TRIP)—as a prior for generative models. Some works enable precise control about the properties of the synthetic images via incorporating 3D prior knowledge into generator [39, 9]. Clustering on feature space to generate a self-conditioned label is another way to get prior [38, 30, 28]. [44] encodes a speech to get a speech latent embedding as an extra input noise, generating images from speech. Some image editing works use pre-trained GAN models as prior without retraining or modification [1, 14, 2].

Different from these variants, in this work, we modify the discriminator to map images into a multivariate Gaussian distribution, which can supply stronger real information via adversarial learning. Then the real information is applied in every convolutional layer of generator through AdaIN in a more robust and straightforward way, which can greatly promote the ability of generator.

3. Preliminary Works

3.1. LSGAN

Unlike the original GAN [13], LSGAN [31] adopts the least square as loss functions. It punishes those samples that are far from the decision boundary, which can perform a more stable learning process. Since P^{2}GAN uses the losses of LSGAN as basis, we introduce its objective functions first:

$$\begin{aligned}\min_D V_{LSGAN}(D) =& \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[(D(\mathbf{x})-b)^2]+ \\ & \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[(D(\mathbf{x})-a)^2] \\ \min_G V_{LSGAN}(G) =& \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[(D(\mathbf{x})-c)^2]\,, \end{aligned} \tag{1}$$

where $p_{data}(\mathbf{x})$ is the distribution of real images, and $p_g(\mathbf{x})$ is the distribution of generated images. a and b are the labels for fake and real data respectively while c denotes the value that G wants D to believe for fake data.

3.2. AdaIN

AdaIN [18] is proposed for style transfer, which can fuse external style image information into normalization. Given the i-th convolutional feature $\mathbf{m}_i$, AdaIN normalizes $\mathbf{m}_i$ by Instance Normalization (IN) [42], and then scales and biases it using the corresponding mean and variance supplied by IN on the style feature $\mathbf{s}_i$. The operation is defined as:

$$\mathrm{AdaIN}(\mathbf{m}_i, \mathbf{s}_i) = \sigma_{IN}(\mathbf{s}_i)\frac{\mathbf{m}_i - \mu_{IN}(\mathbf{m}_i)}{\sigma_{IN}(\mathbf{m}_i)} + \mu_{IN}(\mathbf{s}_i)\,, \tag{2}$$

In this work, we use the AdaIN layers to apply the real information to generator. Similar explorations are confirmed to be useful in style transfer [22] and image synthesis [20, 36, 7].

3.3. Reparameterization Trick

Reparameterization trick is used in VAE [23] for applying gradient descent in variational inference. Once we get the factors of a multivariate Gaussian distribution, the samples of the distribution can be generated by reparameterization trick. In detail, we first sample a random noise $\boldsymbol{\epsilon}$ from a standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, then the samples from the specific multivariate Gaussian distribution can be:

$$\mathbf{z} = \boldsymbol{\epsilon} \odot \boldsymbol{\sigma} + \boldsymbol{\mu}\,, \tag{3}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the Gaussian factors.

4. The Proposed Method

We first define a distribution discrepancy, then describe the discriminator and the generator. Finally we give the theoretical convergence analysis and show the training methods. The entire network architecture is shown in Figure 1.

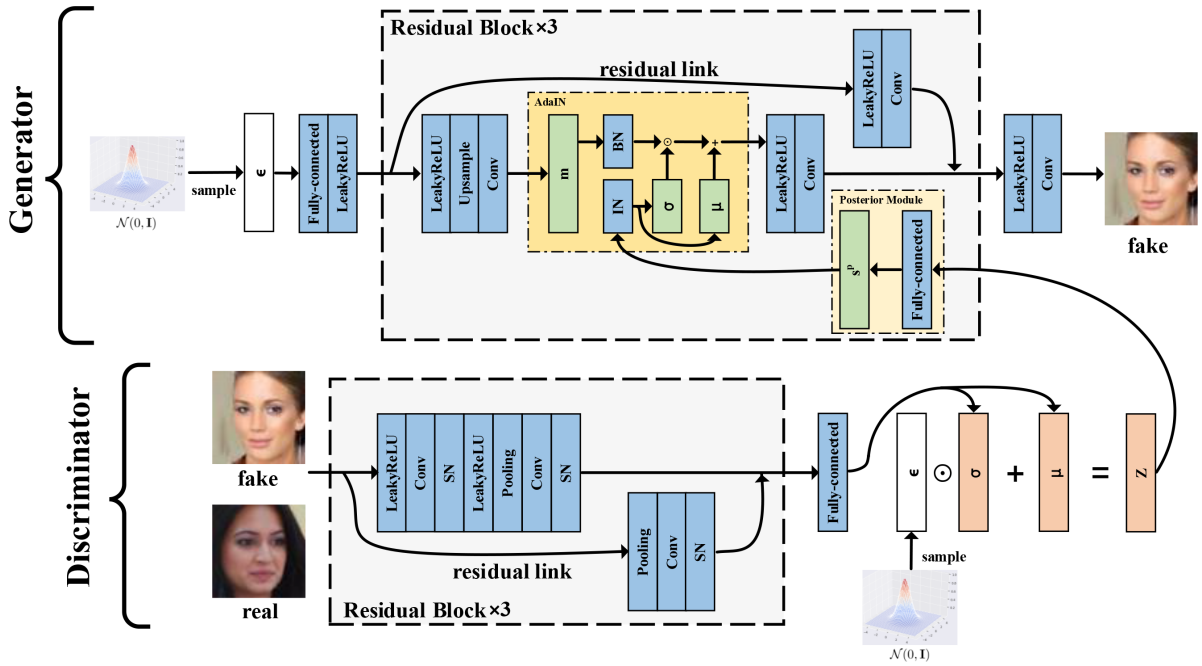


Figure 1: The network architecture. The discriminator takes images as input and outputs the factors $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ of a multivariate Gaussian distribution. The generator produces images from random noise $\boldsymbol{\epsilon}$ and posterior latent code $\mathbf{z}$ with the help of posterior modules and AdaIN in training process. "×3" means that the same structure repeats three times.

Notation	Explanation
$\mathbf{x}$	real or generated images
$\mathbf{z}$	posterior latent code
$\boldsymbol{\epsilon}$	random noise
P_i	the i-th posterior module
$p_{data}(\mathbf{x})$	distribution of real images
$p_g(\mathbf{x})$	distribution of generated images
$D(\mathbf{x})$	posterior distribution on $\mathbf{z}$
P_a, P_b, P_c	prior distribution on $\mathbf{z}$
μ_{IN}, σ_{IN}	mean and variance produced by IN
μ_{BN}, σ_{BN}	mean and variance produced by BN

Table 1: Notations

4.1. Distribution Discrepancy

Hypothesis: Suppose a_1 and a_2 are sampled from two different one-dimensional Gaussian distributions $P_1 = \mathcal{N}(\mu_1, \sigma_1^2)$ and $P_2 = \mathcal{N}(\mu_2, \sigma_2^2)$ by reparameterization trick: $a_1 = \epsilon\sigma_1 + \mu_1$, $a_2 = \epsilon\sigma_2 + \mu_2$. The least square distance between a_1 and a_2 is:

$$Dis = \frac{1}{2}\mathbb{E}_{a_1\sim P_1,\, a_2\sim P_2}[(a_1 - a_2)^2]\,, \tag{4}$$

Then for $\forall\epsilon \sim \mathcal{N}(0,1)$, if Dis is approximately 0, we can reach a conclusion that two Gaussian distributions P_1 and P_2 are basically equal. We define the discrepancy between two Gaussian distributions under the hypothesis as:

$$Dis = \frac{1}{2}(P_1 - P_2)^2\,, \tag{5}$$

Proof: Based on reparameterization trick, we can get:

$$a_\Delta = a_1 - a_2 = \epsilon(\sigma_1 - \sigma_2) + (\mu_1 - \mu_2) \tag{6}$$

For $\forall\epsilon \sim \mathcal{N}(0,1)$, we have $a_\Delta \sim \mathcal{N}(\mu_\Delta, \sigma_\Delta^2)$ and $\mu_\Delta = \mu_1 - \mu_2, \sigma_\Delta = \sigma_1 - \sigma_2$. Then the Dis can be rewritten as:

$$\begin{aligned} 2Dis &= \mathbb{E}[a_\Delta^2] \\ &= \mathbb{E}[(a_\Delta - \mu_\Delta + \mu_\Delta)^2] \\ &= \mathbb{E}[(a_\Delta - \mu_\Delta)^2] + 2\mathbb{E}[(a_\Delta - \mu_\Delta)\mu_\Delta] + \mathbb{E}[\mu_\Delta^2] \\ &= \sigma_\Delta^2 + 2\mu_\Delta(\mathbb{E}[a_\Delta] - \mu_\Delta) + \mu_\Delta^2 \\ &= \sigma_\Delta^2 + \mu_\Delta^2 \end{aligned} \tag{7}$$

If $2Dis = \sigma_\Delta^2 + \mu_\Delta^2 = (\sigma_1 - \sigma_2)^2 + (\mu_1 - \mu_2)^2 = 0$, there must be $\sigma_1 = \sigma_2, \mu_1 = \mu_2$, which means that two Gaussian distributions P_1 and P_2 are equal. The proof also holds for multivariate Gaussian distributions based on the assumption that each dimension is mutually independent.

4.2. Posterior Distribution Discriminator

The discriminator outputs a posterior distribution on latent code $\mathbf{z}$ instead of a probability scalar. Structurally, our

discriminator has two fully-connected layers without activation functions for outputing two factors of a multivariate Gaussian posterior, i.e., mean $\boldsymbol{\mu} \in \mathbb{R}^{b\times d}$ and standard deviation $\boldsymbol{\sigma} \in \mathbb{R}^{b\times d}$, where b is the batch size, and d is the number of Gaussian dimension. Note that each dimension is mutually independent by reparameterization trick [23]. Spectral Normalization (SN) [33] is applied in every layer in discriminator for stabilizing training. Given a batch of images $\mathbf{x} \in \mathbb{R}^{b\times h\times w\times c}$, the discriminator outputs $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ simultaneously, which represents the posterior distribution over latent code $\mathbf{z} \in \mathbb{R}^{b\times d}$. The posterior can be expressed as $D(\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \sigma^2\mathbf{I})$, where $\mathbf{x}$ can be sampled from real image distribution $p_{data}(\mathbf{x})$ or generated image distribution $p_g(\mathbf{x})$. In order to construct the adversarial losses, we introduce two priors $P_a = \mathcal{N}(\boldsymbol{\mu_a}, \sigma_a^2\mathbf{I})$ and $P_b = \mathcal{N}(\boldsymbol{\mu_b}, \sigma_b^2\mathbf{I})$ of the latent code $\mathbf{z}$, which are multivariate Gaussian distributions given by users. The similar approach can be found in VAE, where the prior is a standard Gaussian distribution. Some main notations used in this paper are listed in Table 1.

In this work, we generalize the losses of LSGAN to a distribution version by modifying the labels to the priors representing two virtual ground-truth distributions. Using the definition of distribution discrepancy in Section 4.1, the loss of our discriminator can be described as below:

$$\begin{aligned}\mathcal{L}_D =& \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[(D(\mathbf{x}) - P_b)^2] + \\ & \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[(D(\mathbf{x}) - P_a)^2]\,,\end{aligned} \tag{8}$$

where the first term shortens the distance between the posterior $D(\mathbf{x})$ given real images and the virtual ground-truth prior P_b of real images. The second term shortens the distance between the posterior given generated images and another prior P_a.

By minimizing $\mathcal{L}_D$, the posterior given generated or real images is drawn to match different priors, which can distinguish real and fake. Then the real information in discriminator $D(\mathbf{x})$, $\mathbf{x} \sim p_{data}(\mathbf{x})$ will be used to promote the generation process.

4.3. Posterior Promoted Generator

The generator inputs a random noise $\boldsymbol{\epsilon}$ sampled from a standard Gaussian along with a latent code $\mathbf{z}$ from the posterior given real images and outputs the generated images. We intend to reenforce the generator with AdaIN coordinating with some posterior modules, which allows the posterior $D(\mathbf{x})$, $\mathbf{x} \sim p_{data}(\mathbf{x})$ from discriminator to strengthen the generation procedure. Meanwhile a latent code regularizer will be introduced for preventing the real information from vanishing during training.

We present the adversarial loss $\mathcal{L}_{adv}$ for generator at first. It draws the posterior given generated images to prior $P_c = \frac{P_a+P_b}{2} = \mathcal{N}(\frac{\boldsymbol{\mu_a}+\boldsymbol{\mu_b}}{2}, (\frac{\sigma_a+\sigma_b}{2})^2\mathbf{I})$ for the purpose of cheating the discriminator and engendering a rival relationship between generator and discriminator:

$$\mathcal{L}_{adv} = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[(D(\mathbf{x}) - P_c)^2]. \tag{9}$$

Now we describe how to use the posterior given real images and introduce the latent code regularizer. Traditionally the generator receives a random noise $\boldsymbol{\epsilon} \in \mathbb{R}^{b\times n}$ sampled from a standard Gaussian for producing images, where n is the dimension of the random noise. We depart from this design by involving a posterior latent code branch $\mathbf{z} \sim D(\mathbf{x})$ given $\mathbf{x} \sim p_{data}(\mathbf{x})$, where the posterior is from discriminator. Then the posterior modules receive the latent code $\mathbf{z}$ as input, together with AdaIN to embed the real information contained in posterior in each layer of the generator. So we denote $p_g(\mathbf{x}) = G(\boldsymbol{\epsilon}, \mathbf{z})$.

The posterior modules are mutually independent fully-connected layers without any activation functions as shown in Figure 1. They are used for mapping $\mathbf{z}$ to the features in different dimensions corresponding to different convolutional channels. In addition to applying the real information in every layer, posterior modules also provide a chance to adjust the information according to the demand of different convolutional layers:

$$\mathbf{s}_i^p = \mathrm{P}_i(\mathbf{z})\,, \tag{10}$$

where $\mathbf{z}$ is sampled from $D(\mathbf{x})$ given $\mathbf{x} \sim p_{data}(\mathbf{x})$ and $\mathrm{P}_i(\mathbf{z})$ is the i-th posterior module.

Our approach uses the posterior features $\mathbf{s}_i^p$ as the external information in AdaIN and replaces IN with Batch Normalization (BN) [19] for normalizing $\mathbf{m}_i$. The AdaIN layers used in our generator is as follows:

$$\mathrm{AdaIN}(\mathbf{m}_i, \mathbf{s}_i^p) = \sigma_{IN}(\mathbf{s}_i^p)\frac{\mathbf{m}_i - \mu_{BN}(\mathbf{m}_i)}{\sigma_{BN}(\mathbf{m}_i)} + \mu_{IN}(\mathbf{s}_i^p). \tag{11}$$

We first normalize $\mathbf{m}_i$ by BN, making it have zero mean and unit variance, then bias and scale it with the mean and the variance of $\mathbf{s}_i^p$ which have the real information. The reason we change IN to BN for normalizing $\mathbf{m}_i$ is that BN shows a better performance in our experiments. And recent research shows combining BN and IN improves the performance in various scenarios [35]. By involving the posterior latent code $\mathbf{z}$ and AdaIN to generator, we make a "style transfer" on generated images with the real images as "style reference". As a result, AdaIN fuses the real information to each layer by changing the statistics of the features.

In order to preserve the real information during training, we propose a latent code regularizer $\mathcal{L}_z$, which minimizes the distance between the posterior given generated and real images:

$$\mathcal{L}_z = \mathbb{E}_{\mathbf{x}_g\sim p_g(\mathbf{x}), \mathbf{x}_r\sim p_{data}(\mathbf{x})}[|D(\mathbf{x}_g) - D(\mathbf{x}_r)|]\,, \tag{12}$$

where the subscripts g and r are used for distinguishing between images sampled from different distributions $p_g(\mathbf{x})$ and $p_{data}(\mathbf{x})$. $|*|$ means absolute value.

Intuitively, we use the latent code of posterior $D(\mathbf{x}_r)$ to generate $\mathbf{x}_g$, then the posterior $D(\mathbf{x}_g)$ produced by the discriminator should approach to the posterior $D(\mathbf{x}_r)$ which the latent code belongs to. In other words, we reconstruct the latent code of posterior $D(\mathbf{x}_r)$. What's more, $\mathcal{L}_z$ can provide the generator with more gradient information when the quality of the generated images is poor at the early stage of training, which avoids stopping updating generator because of a good discriminator.

The total loss of generator is composed of $\mathcal{L}_{adv}$ and $\mathcal{L}_z$:

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda \mathcal{L}_z \,, \tag{13}$$

where λ is the weight to balance the two given parts.

4.4. Theoretical Analysis

In this section, we start to discuss the relation between P^{2}GAN and f-divergence. Given the fixed G, we can derive the optimal discriminator based on LSGAN [31]:

$$D^*(\mathbf{x}) = \frac{P_b p_{data}(\mathbf{x}) + P_a p_g(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} \,, \tag{14}$$

Note that when D is optimal, the latent code regularizer $\mathcal{L}_z$ is zero. In order to analyze the optimal value for G, we rewrite the loss $\mathcal{L}_G$ as below:

$$\begin{aligned} C(G) =& \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[(D(\mathbf{x}) - P_c)^2] + \\ & \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[(D(\mathbf{x}) - P_c)^2] \,, \end{aligned} \tag{15}$$

where the first term does not have the parameters of G, so the optimal value keeps the same.

We remove the $\mathbf{x}$ in distributions for simplicity in the following equations. With the optimal D, the loss for G can be reformulated as:

$$\begin{aligned} 2C(G) =& \mathbb{E}_{\mathbf{x}\sim p_{data}}[(D^* - P_c)^2] + \mathbb{E}_{\mathbf{x}\sim p_g}[(D^* - P_c)^2] \\ =& \mathbb{E}_{\mathbf{x}\sim p_{data}}[(\frac{P_b p_{data} + P_a p_g}{p_{data} + p_g} - P_c)^2] + \\ & \mathbb{E}_{\mathbf{x}\sim p_g}[(\frac{P_b p_{data} + P_a p_g}{p_{data} + p_g} - P_c)^2] \\ =& \int \frac{[(P_b - P_c)p_{data} + (P_a - P_c)p_g]^2}{p_{data} + p_g} \, dx \\ =& \int \frac{[(P_b - P_a)p_g - (P_b - P_c)(p_{data} + p_g)]^2}{p_{data} + p_g} \, dx \,, \end{aligned} \tag{16}$$

where $P_c = \frac{P_a + P_b}{2} = \mathcal{N}(\frac{\boldsymbol{\mu_a} + \boldsymbol{\mu_b}}{2}, (\frac{\sigma_a + \sigma_b}{2})^2\mathbf{I})$, then we can get:

$$2C(G) = \int (\frac{P_b - P_a}{2})^2 \frac{[2p_g - (p_{data} + p_g)]^2}{p_{data} + p_g} \, dx \,, \tag{17}$$

where the term $\frac{P_b - P_a}{2}$ is with respect to the variable $\mathbf{z}$, so we can put it out of the integration of $\mathbf{x}$. The term can be regarded as a constant value. The equation can be:

$$\begin{aligned} 2C(G) =& \frac{(P_b - P_a)^2}{4} \int \frac{[2p_g - (p_{data} + p_g)]^2}{p_{data} + p_g} \, dx \\ =& \frac{(P_b - P_a)^2}{4} \mathcal{X}^2_{Pearson}((p_{data} + p_g) || 2p_g). \end{aligned} \tag{18}$$

If $P_a \neq P_b$, minimizing $2C(G)$ means minimizing the Pearson $\mathcal{X}$ divergence between $p_{data}(\mathbf{x}) + p_g(\mathbf{x})$ and $2p_g(\mathbf{x})$. The optimal $2C(G)$ is achieved if and only if $p_{data}(\mathbf{x}) + p_g(\mathbf{x}) = 2p_g(\mathbf{x})$, so $p_{data}(\mathbf{x}) = p_g(\mathbf{x})$ for any valid $\mathbf{x}$. And the optimal posterior from D is:

$$D^*(\mathbf{x}) = \frac{P_b p_{data}(\mathbf{x}) + P_a p_g(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} = \frac{P_b + P_a}{2}. \tag{19}$$

In other words, we get:

$$D^*(\mathbf{x}) = \mathcal{N}(\frac{\boldsymbol{\mu_a} + \boldsymbol{\mu_b}}{2}, (\frac{\sigma_a + \sigma_b}{2})^2\mathbf{I}). \tag{20}$$

4.5. Training

We sample $\mathbf{z}$ from each posterior and prior by reparameterization trick with the distribution factors, and employ $\mathbf{z}$ to loss functions for training. By applying sampling training in limited times, we can finally achieve the convergence stage, which can be confirmed in Section 5.5. Sampling can also add more randomicity and prevent overfitting [46]. Each time we select n_b samples, where n_b is the batch size.

Pretraining step is utilized to initialize the parameters of the model and help discriminator extract initial real information in posterior $D(\mathbf{x})$ given $\mathbf{x} \sim p_{data}(\mathbf{x})$. Specifically, we pretrain the whole model without the posterior modules in generator but reserve the loss $\mathcal{L}_z$. The reason is two-fold. First, we do not have mature real information to strengthen generator through the posterior modules in pretraining period. Second, $\mathcal{L}_z$ can still help provide a better gradient for generator without the posterior modules.

Factors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ used in generator are important for better performance in testing. In order to generate high-quality images, we provide two methods for choosing the posterior factors used in generator in testing period. One method is to use the Exponential Moving Average (EMA) with smooth factor 0.9 to generate images. The other method is to use the factors of the theoretical convergence distribution given by Equation 19. These two methods show similar performance in experiments, so we just present the results of using the theoretical convergence distribution.

5. Experiments

5.1. Datasets

CIFAR10 [24] consists of 60,000 images of $32 \times 32 \times 3$ in 10 classes, with 6,000 images per class.

Method	FID↓	
	CIFAR10	CelebA 64×64
BEGAN [5]	71.4	38.1
WGAN [3]	51.3±1.5	37.1±1.9
MD-GAN [11]	36.8	24.5
LSGAN [31]	27.6	51.7
SN-GAN [33]	21.7±0.2	21.7±1.5
WGAN-GP [15]	19.0±0.8	18.0±0.7
Liu *et al.* [28]	18.7±1.2	-
CTGAN [43]	17.6±0.7	15.8±0.6
SWGAN [45]	17.0±1.0	13.2±0.7
WGAN-GP-TRIP [25]	16.7	-
COCO-GAN [27]	-	4.0
P^{2}GAN	**15.5**	**3.9**

Table 2: FID for CIFAR10 and CelebA.

Method	FID↓
RealnessGAN (Ob1)	36.7
RealnessGAN (Ob2)	34.6
RealnessGAN (Ob3)	36.2
P^{2}GAN	**29.7**

Table 3: Comparison with RealnessGAN for CIFAR10.

Method	FID↓	
	CelebA 32×32	CelebA 64×64
StyleGAN	15.9	13.9
P^{2}GAN	**15.3**	**13.4**

Table 4: Comparison with StyleGAN for CelebA.

CelebA [29] consists of 202,599 celebrity images with large variations in 40 facial attributes. We crop the images and retain the main regions of faces in $64\times64\times3$ resolution.

We utilize all the images in each dataset for training, and the model is optimized in a totally unsupervised way.

5.2. Implementation Details

We train models using Adam with $\beta_1 = 0.0$, $\beta_2 = 0.999$, batch size $b = 32$, weight $\lambda = 10.0$ and channel size $c = 64$. The latent code size d is 5 and noise size n is 64 for CIFAR10. For CelebA, d is 20 and n is 128. The learning rate is 0.0002 with a delay rate 0.9 for every 50 epochs for CIFAR10 and CelebA. We set $P_a = \mathcal{N}(-\mathbf{4.0}, 1.0^2\mathbf{I})$ and $P_b = \mathcal{N}(\mathbf{4.0}, 3.0^2\mathbf{I})$ for CIFAR10, while $P_a = \mathcal{N}(-\mathbf{5.0}, 1.0^2\mathbf{I})$ and $P_b = \mathcal{N}(\mathbf{5.0}, 3.0^2\mathbf{I})$ for CelebA. The pretraining epochs for CIFAR10 and CelebA are 10 and 2 respectively. In our ablation studies, the number of training epochs is 20 as our goal is not to get the best results but to check the effectiveness of each method.

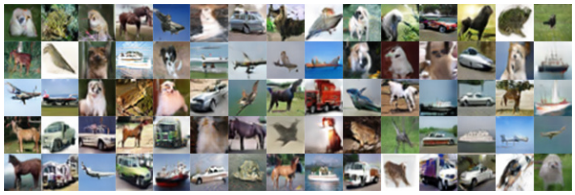


Figure 2: Generated images for CIFAR10 in an unsupervised way.

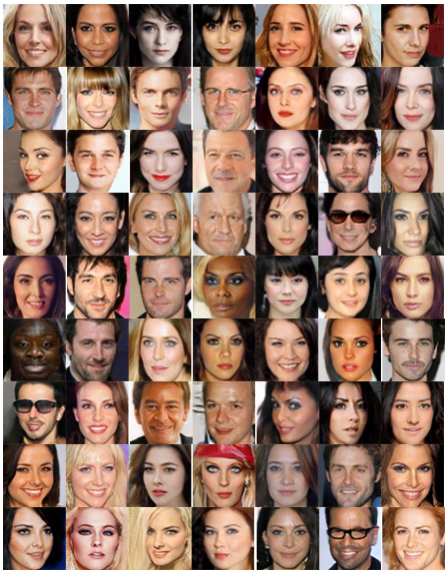


Figure 3: Generated images for CelebA in an unsupervised way.

5.3. Quality Analysis

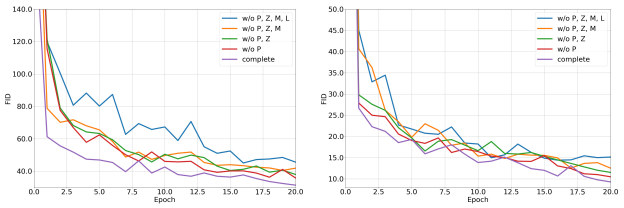
Frechet Inception Distance (FID) [17] is used to evaluate the sample quality for both datasets. FID computes the Wasserstein-2 distance between the generated images and the real images, which is a more principled and comprehensive metric. Lower FID indicates better image quality. In all experiments, 50,000 images are randomly sampled for computing FID. We report the best results for quality analysis. The implementation of FID in this work is based on the original code provided at https://github.com/bioinf-jku/TTUR.

A ResNet [16] structure is applied to get a better score for comparing with several state-of-the-art baselines. **Unless illustrating particularly, we use this ResNet structure to train our model.** Figure 1 shows the structure of this network. We present the FID for CIFAR10 and CelebA in Table 2, where P^{2}GAN is the best among all the baselines. The samples of generated images are provided in

Method	FID↓	
	CIFAR10	CelebA 64×64
P^{2}GAN (**w/o P, Z, M, L**)	45.6	15.2
P^{2}GAN (**w/o P, Z, M**)	41.9	12.5
P^{2}GAN (**w/o P, Z**)	38.0	11.7
P^{2}GAN (**w/o P**)	35.8	10.5
P^{2}GAN	**31.1**	**9.4**

Table 5: Ablation results on FID with different settings.

Figure 4: Generated images by different models.



(a) CIFAR10 (b) CelebA

Figure 5: FID on different ablation settings.

Figure 2 and 3. For qualitative comparison, we show the visual results in Figure 4.

A basic DCGAN [37] structure is utilized for showing the advantage of P^{2}GAN compared with the most recent baseline RealnessGAN. We apply the same DCGAN network settings to our model as RealnessGAN for fairness. The results are listed in Table 3.

A StyleGAN [21] structure is adjusted to fit the loss of P^{2}GAN and use "real information" to strengthen generator. Due to the limited computing resources, we only conduct experiments on CelebA under 32 and 64 resolutions. The results in Table 4 show that P^{2}GAN improves upon StyleGAN. Note that we shorten the training time and set "total_kimg=7k" in StyleGAN and P^{2}GAN, just for showing the effectiveness of P^{2}GAN, not for getting the best scores.

5.4. Ablation Analysis

Several ablation experiments about the pretraining step (**P**), the posterior latent code (**Z**), the posterior modules (**M**) and the latent code regularizer (**L**) are carried out. Table 5 shows the results of four ablation settings of our model. We first train the model without all the parts, which gives the worst performance (**w/o P, Z, M, L**). Then we add the latent code regularizer $\mathcal{L}_z$ for providing more gradient information (**w/o P, Z, M**). Next the posterior modules are brought into training, but we replace the posterior latent code $\mathbf{z}$ with a random noise from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ for reflecting the validity of real information better (**w/o P, Z**). Then we use $\mathbf{z}$ as input to the posterior modules (**w/o P**). Finally we add the pretraining step to train P^{2}GAN completely, which provides the best score. Figure 5 shows the FID of different ablation settings in different epochs.

P_a	P_b	FID↓	
		CIFAR10	CelebA 64× 64
$\mathcal{N}(\mathbf{0.1}, 0.5^2\mathbf{I})$	$\mathcal{N}(\mathbf{0.3}, 0.7^2\mathbf{I})$	35.0	11.2
$\mathcal{N}(-\mathbf{5.0}, 2.0^2\mathbf{I})$	$\mathcal{N}(\mathbf{1.0}, 1.0^2\mathbf{I})$	32.3	9.7
$\mathcal{N}(-\mathbf{5.0}, 1.0^2\mathbf{I})$	$\mathcal{N}(\mathbf{5.0}, 3.0^2\mathbf{I})$	**31.5**	**9.4**
$\mathcal{N}(-\mathbf{20.0}, 1.0^2\mathbf{I})$	$\mathcal{N}(\mathbf{20.0}, 3.0^2\mathbf{I})$	67.5	37.6

Table 6: Ablation results for different priors.

Method	FID↓	
	CIFAR10	CelebA 64×64
P^{2}GAN (**IN**)	34.8	12.2
P^{2}GAN (**EMA**)	31.4	9.8
P^{2}GAN	**31.1**	**9.4**

Table 7: Ablation results for IN and EMA.

Multiple settings for priors P_a and P_b are used to check the influence of the priors. We give the results of four groups of prior settings in Table 6, and the distribution discrepancy of the two priors increases in turn. It can be observed that the more dissimilar the priors, the better the results, which is similar to RealnessGAN [46]. But the results may be bad if the dissimilarity is beyond the network capacity.

Using EMA for applying factors in testing time shows a similar performance compared with the theoretical convergence distribution. **Using IN to normalize $\mathbf{m}_i$ in AdaIN** presents an unsatisfactory FID score compared with using BN. The results are shown in Table 7.

5.5. Convergence Analysis

The theoretical convergence analysis mentioned in Section 4.4 can be validated by checking a global measure of convergence:

$$\mathcal{M}_c = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[(D(\mathbf{x}) - D^*(\mathbf{x}))^2] + \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[(D(\mathbf{x}) - D^*(\mathbf{x}))^2], \quad (21)$$

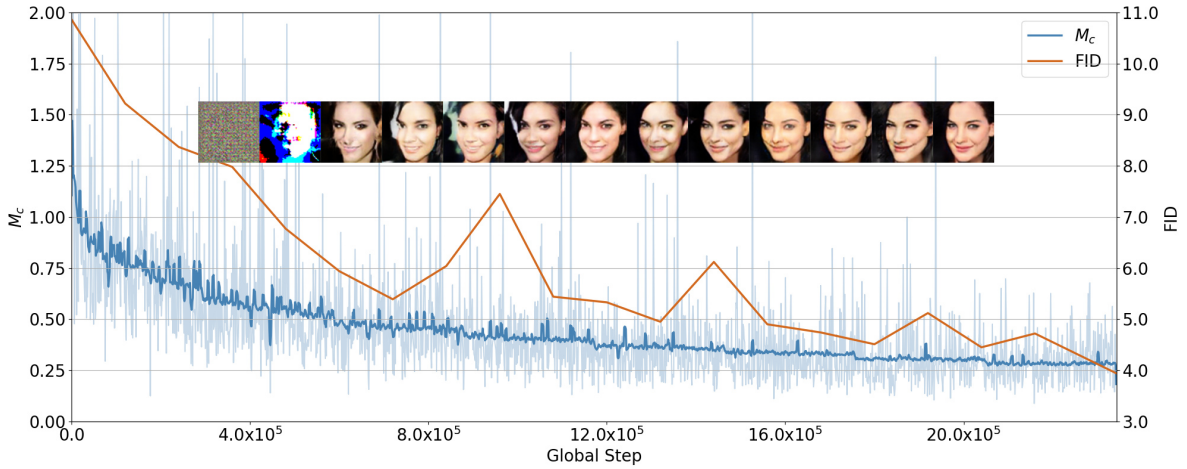
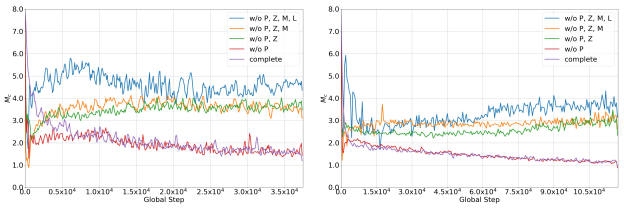


Figure 6: The global measure of convergence. We show the value of $\mathcal{M}_c$ and FID during the whole training for CelebA. The baby blue curve is the original $\mathcal{M}_c$, while the deep blue is the curve achieved by exponential moving average. The images are generated by the same input, indicating that the image quality of the samples improves stably along with the training process.



(a) CIFAR10 (b) CelebA

Figure 7: $\mathcal{M}_c$ on different ablation settings.

Figure 8: Interpolation of generated images.

where $D^*(\mathbf{x}) = \dfrac{P_b + P_a}{2}$ given by Equation 19. We show the values of $\mathcal{M}_c$ during training in Figure 6 and $\mathcal{M}_c$ with different ablation settings in Figure 7. The complete training method provides the most smooth curve and keeps decreasing, which validates the convergence analysis of P^{2}GAN.

5.6. Interpolation Analysis

Image interpolation is conducted to estimate the manifold continuity. We sample two random noises $\boldsymbol{\epsilon}_a$, $\boldsymbol{\epsilon}_b$ and two posterior latent codes $\mathbf{z}_a$, $\mathbf{z}_b$, then they are interpolated to get several noises $\{\boldsymbol{\epsilon}_i\}_{i=1}^{l}$ and codes $\{\mathbf{z}_i\}_{i=1}^{l}$ used for generating and the images are shown in Figure 8. The interpolation images are natural and high-caliber without otiose texture, which implies that our model has learned disentangling features. The experiments verify that P^{2}GAN generates images through the learned features instead of just memorizing the datasets.

6. Conclusion

We propose a novel GAN variant called P^{2}GAN, which modifies the discriminator to produce rich real information by mapping images to a multivariate Gaussian distribution, and enhances the generator through AdaIN and the real information. We modify the losses of LSGAN to a distribution version and prove the convergence theoretically. The convergence analysis in experiments indicates the validity of our training methods, while the interpolation results indicate that we have learned a robust and sufficient latent manifold structure. The proposed P^{2}GAN has achieved comparable results with the state-of-the-art models on high-dimensional multi-modal datasets.

Acknowledgments

This work was supported by National Science Foundation of China (No. 61632019, No. 61876028), the Fundamental Research Funds for the Central Universities (No. DUT20RC(3)066, No. DUT20RC(3)040).

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *IEEE International Conference on Computer Vision (ICCV)*, pages 4431–4440, 2019. 2

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8293–8302, 2020. 2

[3] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 214–223, 2017. 1, 2, 6

[4] Matan Ben-Yosef and Daphna Weinshall. Gaussian mixture generative adversarial networks for diverse datasets, and the unsupervised clustering of images. *arXiv preprint arXiv:1808.10356*, 2018. 1, 2

[5] David Berthelot, Tom Schumm, and Luke Metz. BEGAN: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 6

[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, pages 2172–2180, 2016. 1, 2

[7] Hyeong-Seok Choi, Changdae Park, and Kyogu Lee. From inference to generation: End-to-end fully self-supervised generation of human face from speech. In *8th International Conference on Learning Representations (ICLR)*, 2020. 2

[8] Jiankang Deng, Shiyang Cheng, Niannan Xue, Yuxiang Zhou, and Stefanos Zafeiriou. UV-GAN: adversarial facial UV map completion for pose-invariant face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7093–7102, 2018. 1

[9] Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3d imitative-contrastive learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5153–5162, 2020. 1, 2

[10] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. In *5th International Conference on Learning Representations (ICLR)*, 2017. 1, 2

[11] Hamid Eghbal-zadeh, Werner Zellinger, and Gerhard Widmer. Mixture density generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5820–5829, 2019. 1, 6

[12] Zhenglin Geng, Chen Cao, and Sergey Tulyakov. 3d guided fine-grained face manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9821–9830, 2019. 1

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27 (NeurIPS)*, pages 2672–2680, 2014. 1, 2

[14] Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code GAN prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3009–3018, 2020. 2

[15] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 5767–5777, 2017. 1, 2, 6

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 6

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 6626–6637, 2017. 6

[18] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1510–1519, 2017. 1, 2

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456, 2015. 4

[20] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. 2

[21] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. 7

[22] Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwanghee Lee. U-GAT-IT: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. In *8th International Conference on Learning Representations (ICLR)*, 2020. 2

[23] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR)*, 2014. 1, 2, 4

[24] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research), 2010. URL `http://www.cs.toronto.edu/~kriz/cifar.html`. 5

[25] Maksim Kuznetsov, Daniil Polykovskiy, Dmitry P. Vetrov, and Alexander Zhebrak. A prior of a googol gaussians: a tensor ring induced prior for generative models. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 4104–4114, 2019. 1, 2, 6

[26] Soochan Lee, Junsoo Ha, and Gunhee Kim. Harmonizing maximum likelihood with gans for multimodal conditional generation. In *7th International Conference on Learning Representations (ICLR)*, 2019. 1

[27] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. COCO-GAN: Generation by parts via conditional coordinating. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4511–4520, 2019. 6

[28] Steven Liu, Tongzhou Wang, David Bau, Jun-Yan Zhu, and Antonio Torralba. Diverse image generation via self-conditioned gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14274–14283, 2020. 1, 2, 6

[29] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738, 2015. 6

[30] Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generation with fewer labels. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 4183–4192, 2019. 2

[31] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821, 2017. 1, 2, 5, 6

[32] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1

[33] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations (ICLR)*, 2018. 4, 6

[34] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *6th International Conference on Learning Representations (ICLR)*, 2018. 1

[35] Hyeonseob Nam and Hyo-Eun Kim. Batch-instance normalization for adaptively style-invariant neural networks. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 2563–2572, 2018. 4

[36] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 7587–7596, 2019. 2

[37] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations (ICLR)*, 2016. 7

[38] Alexander Sage, Eirikur Agustsson, Radu Timofte, and Luc Van Gool. Logo synthesis and manipulation with clustered generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5879–5888, 2018. 2

[39] Yujun Shen, Bolei Zhou, Ping Luo, and Xiaoou Tang. Facefeat-gan: a two-stage approach for identity-preserving face synthesis. *arXiv preprint arXiv:1812.01288*, 2018. 2

[40] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. VEEGAN: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 3308–3318, 2017. 1, 2

[41] Akash Srivastava, Kai Xu, Michael U. Gutmann, and Charles Sutton. Generative ratio matching networks. In *8th International Conference on Learning Representations (ICLR)*, 2020. 1

[42] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 2

[43] Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. Improving the improved training of wasserstein gans: A consistency term and its dual effect. In *6th International Conference on Learning Representations (ICLR)*, 2018. 6

[44] Yandong Wen, Bhiksha Raj, and Rita Singh. Face reconstruction from voice using generative adversarial networks. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 5266–5275, 2019. 1, 2

[45] Jiqing Wu, Zhiwu Huang, Dinesh Acharya, Wen Li, Janine Thoma, Danda Pani Paudel, and Luc Van Gool. Sliced wasserstein generative models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3713–3722, 2019. 6

[46] Yuanbo Xiangli, Yubin Deng, Bo Dai, Chen Change Loy, and Dahua Lin. Real or not real, that is the question. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 2, 5, 7

[47] Chang Xiao, Peilin Zhong, and Changxi Zheng. BourGAN: Generative networks with metric embeddings. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 2275–2286, 2018. 1, 2