How Generative Adversarial Networks and Their Variants Work: An Overview of GAN

Yongjun Hong, Uiwon Hwang, Jaeyoon Yoo and Sungroh Yoon
Department of Electrical & Computer Engineering
Seoul National University, Seoul, Korea
{yjhong, uiwon.hwang, yjy765, sryoon}@snu.ac.kr

Abstract

Generative Adversarial Networks (GAN) have received wide attention in the machine learning field for their potential to learn high-dimensional, complex real data distribution. Specifically, they do not rely on any assumptions about the distribution and can generate real-like samples from latent space in a simple manner. This powerful property leads GAN to be applied to various applications such as image synthesis, image attribute editing, image translation, domain adaptation and other academic fields. In this paper, we aim to discuss the details of GAN for those readers who are familiar with, but do not comprehend GAN deeply or who wish to view GAN from various perspectives. In addition, we explain how GAN operates and the fundamental meaning of various objective functions that have been suggested recently. We then focus on how the GAN can be combined with an autoencoder framework. Finally, we enumerate the GAN variants that are applied to various tasks and other fields for those who are interested in exploiting GAN for their research.

1 Introduction

Recently, in the machine learning field, generative models have become more important and popular because of their applicability in various fields. Their capability to represent complex and high-dimensional data can be utilized in treating images [2, 13, 60, 68, 132, 138, 150], videos [127, 130, 131], music generation [43, 69, 146], natural languages [51, 76] and other academic domains such as medical images [17, 80, 141] and security [114, 129]. Specifically, generative models are highly useful for image to image translation (See Figure 1) [10, 60, 142, 150] which transfers images to another specific domain, image super-resolution [68], changing some features of an object in an image [3, 39, 78, 98, 149] and predicting the next frames of a video [127, 130, 131]. In addition, generative models can be the solution for various problems in the machine learning field such as semisupervised learning [22, 70, 109, 120], which tries to address the lack of labeled data, and domain adaptation [2, 13, 50, 113, 116, 145], which leverages known knowledge for some tasks in other domains where only little information is given.

Formally, a generative model learns to model a real data probability distribution $p_{\text{data}}(x)$ where the data x exists in the d-dimensional real space R^d, and most generative models, including autoregressive models [96, 110], are based on the maximum likelihood principle with a model parametrized by parameters θ. With independent and identically distributed (i.i.d.) training samples x^i where $i \in \{1, 2, \cdots n\}$, the likelihood is defined as the product of probabilities that the model gives to each training data: $\prod_{i=1}^{n} p_\theta(x^i)$ where $p_\theta(x)$ is the probability that the model assigns to x. The maximum likelihood principle trains the model to maximize the likelihood that the model follows the real data distribution.

From this point of view, we need to assume a certain form of $p_\theta(x)$ explicitly to estimate the likelihood of the given data and retrieve the samples from the learned model after the training. In this way, some approaches [95, 96, 110] successfully learned the generative model in various fields including speech synthesis. However, while the explicitly defined probability density function brings about computational tractability, it may fail to represent the complexity of real data distribution and learn the high-dimensional data distributions [90].

Generative Adversarial Networks (GANs) [38] were proposed to solve the disadvantages of other generative models. Instead of maximizing the likelihood, GAN introduces the concept of adversarial learning between the generator and the discriminator. The generator and the discriminator act as adversaries with respect to each other to produce real-like samples. The generator is a continuous, differentiable transformation function mapping a prior distribution p_z from the latent space $\mathcal{Z}$ into the data space $\mathcal{X}$ that tries to fool the discriminator. The discriminator distinguishes its input whether it comes from the real data distribution or the generator. The basic intuition behind the adversarial learning is that as the generator tries to deceive the discriminator which also evolves against the generator, the generator improves. This adversarial process gives GAN notable advantages over the other generative models.

GAN avoids defining $p_\theta(x)$ explicitly, and instead trains the generator using a binary classification of the discriminator. Thus, the generator does not need to follow a certain form of $p_\theta(x)$. In addition, since the generator is a simple, usually deterministic feed-forward network from $\mathcal{Z}$ to $\mathcal{X}$, GAN can sample the generated data in a simple manner unlike other models using the Markov chain [118] in which the sampling is computationally slow and not accurate. Furthermore, GAN can parallelize the generation, which is not possible for other models such as PixelCNN [110], PixelRNN [96], and WaveNet [95] due to their autoregressive nature.

For these advantages, GAN has been gaining considerable attention, and the desire to use GAN in many fields is growing. In this study, we explain GAN [37, 38] in detail which generates sharper and better real-like samples than the other generative models by adopting two components, the generator and the discriminator. We look into how GAN works theoretically and how GAN has been applied to various applications.

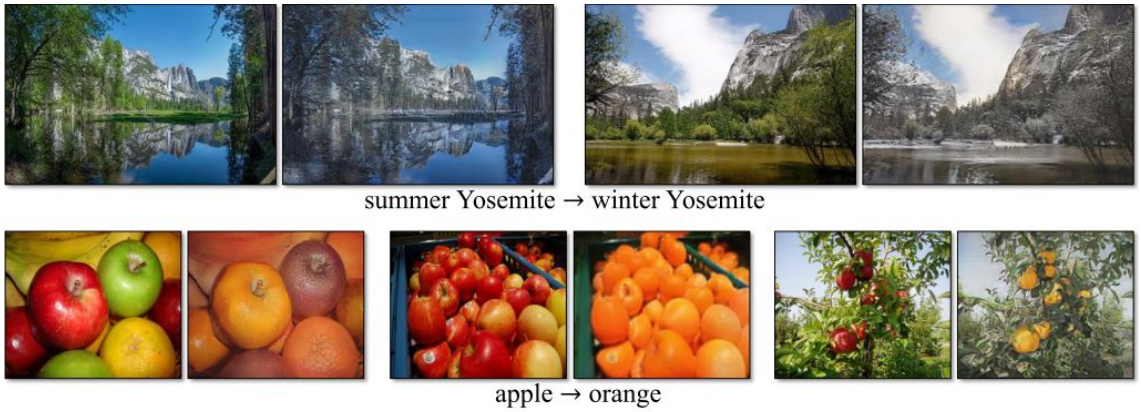


Figure 1: Examples of unpaired image to image translation from CycleGAN [150]. CycleGAN use a GAN concept, converting contents of the input image to the desired output image. There are many creative applications using a GAN, including image to image translation, and these will be introduced in Section 4. Images from CycleGAN [150].

1.1 Paper Organization

Table 1 shows GAN and GAN variants which will be discussed in Section 2 and 3. In Section 2, we first present a standard objective function of a GAN and describe how its components work. After that, we present various objective functions proposed recently, focusing on their similarities in terms of the feature matching problem. We then explain the architecture of GAN extending the discussion to dominant obstacles caused by optimizing a minimax problem, especially a mode collapse, and how to address those issues.

In Section 3, we discuss how GAN can be exploited to learn the latent space where a compressed and low dimensional representation of data lies. In particular, we emphasize how the GAN extracts the latent space from the data space with autoencoder frameworks. Section 4 provides several extensions of the GAN applied to other domains and various topics as shown in Table 2. In Section 5, we observe a macroscopic view of GAN, especially why GAN is advantageous over other generative models. Finally, Section 6 concludes the paper.

Table 1: An overview of GANs discussed in Section 2 and 3.

Subject	Topic	Reference
Object functions	f-divergence	GAN [38], f-GAN [92], LSGAN [79]
	IPM	WGAN [5], WGAN-GP [44], FISHER GAN [87], McGAN [88], MMDGAN [71]
Architecture	DCGAN	DCGAN [104]
	Hierarchy	StackedGAN [52], GoGAN [57], Progressive GAN [59]
	Auto encoder	BEGAN [11], EBGAN [148], MAGAN [133]
Issues	Theoretical analysis	Towards principled methods for training GANs [4]
		Generalization and equilibrium in GAN [6]
	Mode collapse	MRGAN [14], DRAGAN [64], MAD-GAN [35], Unrolled GAN [82]
Latent space	Decomposition	CGAN [83], ACGAN [93], InfoGAN [16], ss-InfoGAN [121]
		cGANs with projection discriminator [84]
	Encoder	ALI [27], BiGAN [25], Adversarial Generator-Encoder Networks [128]
	VAE	VAEGAN [67], α-GAN [107]

Table 2: Categorization of GANs applied for various topics.

Domain	Topic	Reference
Image	Image translation	Pix2pix [55], PAN [132], CycleGAN [150], DiscoGAN [60]
		DualGAN [142], One-sided unsupervised domain mapping [10]
	Super resolution	SRGAN [68]
	Object detection	SeGAN [29], Perceptual GAN for small object detection [72]
	Object transfiguration	GeneGAN [149], GP-GAN [137]
	3D image generation	3DGAN [138]
		3D shape induction from 2d views of multiple objects [33]
	Joint image generation	Coupled GAN [77]
	Video generation	VGAN [130], Pose-GAN [131], MoCoGAN [127]
	Text to image	Stack GAN [52], TAC-GAN [19]
	Change facial attributes	SD-GAN [24], SL-GAN [143], DR-GAN [126], AGEGAN [3]
Sequential data	Music generation	C-RNN-GAN [86], SeqGAN [146], ORGAN [43]
	Text generation	RankGAN [76]
	Speech conversion	VAW-GAN [51]
Others	Semi-supervised learning	Categorical GAN [120], SSL-GAN [22], Triple-GAN [70]
	Domain adaptation	DANN [2], CyCADA [50], DIRT-T [116], WDGRL [113]
		Unsupervised pixel-level domain adaptation [13]
	Continual learning	Deep generative replay [115]
	Medical image segmentation	DI2IN [141], SCAN [17], SegAN [139]
	Steganography	Steganography GAN [129], Secure steganography GAN [114]

2 Generative Adversarial Networks

As its name implies, GAN is a generative model that learns to make real-like data adversarially [38] containing two components, the generator G and the discriminator D. G takes the role of producing real-like fake samples from the latent variable z, whereas D determines whether its input comes from G or real data space. D outputs a high value as it determines that its input is more likely to be real. G and D compete with each other to achieve their individual goals, thus generating the term adversarial. This adversarial learning situation can be formulated as Equation 1 with parametrized networks G and D. $p_{\text{data}}(x)$ and $p_z(z)$ in Equation 1 denote the real data probability distribution defined in the data space $\mathcal{X}$ and the probability distribution of z defined on the latent space $\mathcal{Z}$, respectively.

$$\min_G \max_D V(G,D) = \min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}} \left[\log D(x)\right] + \mathbb{E}_{z\sim p_z} \left[\log(1 - D(G(z))\right]. \tag{1}$$

$V(G,D)$ is a binary cross entropy function that is commonly used in binary classification problems [79]. Note that G maps z from $\mathcal{Z}$ into the element of $\mathcal{X}$, whereas D takes an input x and distinguishes whether x is a real sample or a fake sample generated by G.

As D wants to classify real or fake samples, $V(G,D)$ is a natural choice for an objective function in aspect of the classification problem. From D's perspective, if a sample comes from real data, D will maximize its output, while if a sample comes from G, D will minimize its output; thus, the $\log(1 - D(G(z)))$ term appears in Equation 1. Simultaneously, G wants to deceive D, so it tries to maximize D's output when a fake sample is presented to D. Consequently, D tries to maximize $V(G,D)$ while G tries to minimize $V(G,D)$, thus forming the minimax relationship in Equation 1. Figure 2 shows an illustration of the GAN.

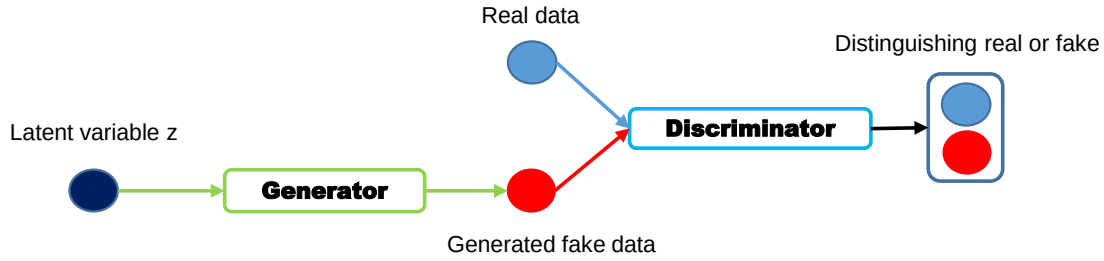


Figure 2: Generative Adversarial Network

Theoretically, assuming that the two models G and D both have sufficient capacity, the equilibrium between G and D occurs when $p_{\text{data}}(x) = p_g(x)$ and D always produces $\frac{1}{2}$ where $p_g(x)$ means a probability distribution of the data provided by the generator [38]. Formally, for fixed G the optimal discriminator $D^\star$ is $D^\star(x) = \frac{p_{\text{g}}(x)}{p_{\text{g}}(x)+p_{\text{data}}(x)}$ which can be shown by differentiating Equation 1. If we plug in the optimal $D^\star$ into Equation 1, the equation becomes the Jensen Shannon Divergence (JSD) between $p_{\text{data}}(x)$ and $p_{\text{g}}(x)$. Thus, the optimal generator minimizing JSD$(p_{\text{data}}||p_{\text{g}})$ is the data distribution $p_{\text{data}}(x)$ and D becomes $\frac{1}{2}$ by substituting the optimal generator into the optimal $D^\star$ term.

Beyond the theoretical support of GAN, the above paragraph leads us to infer two points. First, from the optimal discriminator in the above, GAN can be connected into the density ratio trick [107]. That is, the density ratio between the data distribution and the generated data distribution as follows:

$$Dr(x) = \frac{p_{\text{data}}(x)}{p_{\text{g}}(x)} = \frac{p(x|y=1)}{p(x|y=0)} = \frac{p(y=1|x)}{p(y=0|x)} = \frac{D^\star(x)}{1-D^\star(x)} \tag{2}$$

where $y = 0$ and $y = 1$ indicate the generated data and the real data, respectively and $p(y = 1) = p(y = 0)$ is assumed. This means that GAN addresses the intractability of the likelihood by just using the relative behavior of the two distributions [107], and transferring this information to the generator to produce real-like samples. Second, GAN can be interpreted to measure the discrepancy between the generated data distribution and the real data distribution and then learn to reduce it. The discriminator is used to implicitly measure the discrepancy.

Despite the advantage and theoretical support of GAN, many shortcomings has been found due to the practical issues and inability to implement the assumption in theory including the infinite capacity of the discriminator. There have been many attempts to solve these issues by changing the objective function, the architecture, etc. Holding the fundamental framework of GAN, we assess variants of the object function and the architectures proposed for the development of GAN. We then focus on the crucial failures of GAN and how to address those issues.

2.1 Object Functions

The goal of generative models is to match the real data distribution $p_{\text{data}}(x)$ from $p_{\text{g}}(x)$. Thus, minimizing differences between two distributions is a crucial point for training generative models. As mentioned above, standard GAN [38] minimizes JSD$(p_{\text{data}}||p_{\text{g}})$ estimated by using the discriminator. Recently, researchers have found that various distances or divergence measures can be adopted instead of JSD and can improve the performance of the GAN. In this section, we discuss how to measure the discrepancy between $p_{\text{data}}(x)$ and $p_{\text{g}}(x)$ using various distances and object functions derived from these distances.

2.1.1 f-divergence

The f-divergence $D_f(p_{\text{data}}||p_{\text{g}})$ is one of the means to measure differences between two distributions with a specific convex function f. Using the ratio of the two distributions, the f-divergence for p_{data} and p_{g} with a function f is defined as follows:

$$D_f(p_{\text{data}}||p_{\text{g}}) = \int_{\mathcal{X}} p_{\text{g}}(x) f\left(\frac{p_{\text{data}}(x)}{p_{\text{g}}(x)}\right) dx \tag{3}$$

It should be noted that $D_f(p_{\text{data}}||p_{\text{g}})$ can act as a divergence between two distributions under the conditions that f is a convex function and $f(1) = 0$ is satisfied. Because of the condition $f(1) = 0$, if two distributions are equivalent, their ratio becomes 1 and their divergence goes to 0. Though f is termed a generator function [92] in general, we call f an f-divergence function to avoid confusion with the generator G.

f-GAN [92] generalizes the GAN objective function in terms of f-divergence under an arbitrary convex function f. As we do not know the distributions exactly, Equation 3 should be estimated through a tractable form such as an expectation form. By using the convex conjugate $f(u) = \sup_{t \in dom f^\star}(tu - f^\star(t))$, Equation 3 can be reformulated as follows:

$$D_f(p_{\text{data}}||p_{\text{g}}) = \int_{\mathcal{X}} p_{\text{g}}(x) \sup_{t \in dom f^\star} \left(t\frac{p_{\text{data}}(x)}{p_{\text{g}}(x)} - f^\star(t) \right) dx \tag{4}$$

$$\geq \sup_{T \in \mathcal{T}} \left(\int_{\mathcal{X}} T(x) p_{\text{data}}(x) - f^\star\left(T(x)\right) p_{\text{g}}(x) \right) dx \tag{5}$$

$$= \sup_{T \in \mathcal{T}} \left(\mathbb{E}_{x \sim p_{\text{data}}}[T(x)] - \mathbb{E}_{x \sim p_{\text{g}}}[f^\star(T(x))] \right) \tag{6}$$

where $f^\star$ is a Fenchel conjugate [30] of a convex function f and $dom f^\star$ indicates a domain of $f^\star$.

Equation 5 follows from the fact that the summation of the maximum is larger than the maximum of the summation and $\mathcal{T}$ is an arbitrary function class that satisfies $\mathcal{X} \to \mathbb{R}$. Note that we replace t in Equation 4 by $T(x) : \mathcal{X} \to dom f^\star$ in Equation 5 to make t involved in $\int_{\mathcal{X}}$. If we express $T(x)$ in the form of $T(x) = a(D_\omega(x))$ with $a(\cdot) : \mathbb{R} \to dom f^\star$ and $D_\omega(x) : \mathcal{X} \to \mathbb{R}$, we can interpret $T(x)$ as the parameterized discriminator with a specific activation function $a(\cdot)$.

We can then create various GAN frameworks with the specified generator function f and activation function a using the parameterized generator G_θ and discriminator $\mathcal{T}_\omega$. Similar to the standard GAN [38], f-GAN first maximizes the lower bound in Equation 6 with respect to T_ω to make the lower bound tight to $D_f(p_{\text{data}}||p_{\text{g}})$, and then minimizes the approximated divergence with respect to G_θ to make $p_{\text{g}}(x)$ similar to $p_{data}(x)$. In this manner, f-GAN tries to generalize various GAN objectives by estimating some types of divergences given f as shown in Table 3

Table 3: GANs using f-divergence. Table reproduced from [92].

GAN	Divergence	Generator $f(t)$
	KLD	$t \log t$
GAN [38]	JSD - $2 \log 2$	$t \log t - (t+1)\log(t+1)$
LSGAN [79]	Pearson $\mathcal{X}^2$	$(t-1)^2$
EBGAN [148]	Total Variance	$\lvert t-1 \rvert$

Kullback-Leibler Divergence (KLD), reverse KLD, JSD and other divergences can be derived using the f-GAN framework with the specific generator function f, though they are not all represented in Table 3. Among f-GAN based GANs, LSGAN is one of the most widely used GANs due to its simplicity and high performance; we briefly explain LSGAN in the next paragraph. To summarize, f-divergence $D_f(p_{\text{data}}||p_{\text{g}})$ in Equation 3 can be indirectly estimated by calculating expectations of its lower bound to deal with the intractable form with the unknown probability distributions. f-GAN generalizes various divergences under an f-divergence framework and thus it can derive the corresponding GAN objective with respect to a specific divergence.

2.1.1.1 Least square GAN

The standard GAN uses a sigmoid cross entropy loss for the discriminator to classify whether its input is real or fake. However, if a generated sample is well classified as real by the discriminator, there would be no reason for the generator to be updated even though the generated sample is located far from the real data distribution. A sigmoid cross entropy loss can barely push such generated samples towards real data distribution since its classification role has been achieved.

Motivated by this phenomenon, least-square GAN (LSGAN) replaces a sigmoid cross entropy loss with a least square loss, which directly penalizes fake samples by moving them close to the real data distribution. Compared to Equation 1, LSGAN solves the following problems:

$$\max_D V_{LSGAN}(D) = \max_D \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}}[(D(x)-b)^2] + \mathbb{E}_{z\sim p_z}[(D(G(z))-a)^2] \tag{7}$$

$$\min_G V_{LSGAN}(G) = \min_G \frac{1}{2}\mathbb{E}_{z\sim p_z}[(D(G(z))-c)^2] \tag{8}$$

where a, b and c refer to the baseline values for the discriminator.

Equations 7 and 8 use a least square loss, under which the discriminator is forced to have designated values (a, b and c) for the real samples and the generated samples, respectively, rather than a probability for the real or fake samples. Thus, in contrary to a sigmoid cross entropy loss, a least square loss not only classifies the real samples and the generated samples but also pushes generated samples closer to the real data distribution. In addition, LSGAN can be connected to an f-divergence framework as shown in Table 3.

2.1.2 Integral probability metric

Integral probability metric (IPM) defines a critic function f, which belongs to a specific function class $\mathcal{F}$, and IPM is defined as a maximal measure between two arbitrary distributions under the frame of f. In a compact space $\mathcal{X} \subset R^d$, let $\mathcal{P}(\mathcal{X})$ denote the probability measures defined on $\mathcal{X}$. IPM metrics between two distributions p_{data}, $p_{\text{g}} \in \mathcal{P}(\mathcal{X})$ is defined as follows:

$$d_{\mathcal{F}}(p_{\text{data}}, p_{\text{g}}) = sup_{f\in\mathcal{F}}\ \mathbb{E}_{x\sim p_{\text{data}}}[f(x)] - \mathbb{E}_{x\sim p_{\text{g}}}[f(x)] \tag{9}$$

As shown in Equation 9, IPM metric $d_{\mathcal{F}}(p_{\text{data}}, p_{\text{g}})$ defined on $\mathcal{X}$ determines a maximal distance between $p_{\text{data}}(x)$ and $p_{\text{g}}(x)$ with functions belonging to $\mathcal{F}$ which is a set of measurable, bounded, real-valued functions. It should be noted that $\mathcal{F}$ determines various distances and their properties. Here, we consider the function class $\mathcal{F}_{v,w}$ whose elements are the critic f, which can be represented as a standard inner product of parameterized neural networks Φ_w and a linear output activation function v in real space as described in Equation 10. w belongs to parameter space Ω that forces the function space to be bounded. Under the function class in Equation 10, we can reformulate Equation 9 as the following equations:

$$\mathcal{F}_{v,w} = \{f(x) =< v, \Phi_w(x) > | v \in R^m, \Phi_w(x) : \mathcal{X} \to R^m\} \tag{10}$$

$$d_{\mathcal{F}_{v,w}}(p_{\text{data}}, p_{\text{g}}) = sup_{f\in\mathcal{F}_{v,w}}\ \mathbb{E}_{x\sim p_{\text{data}}} f(x) - \mathbb{E}_{x\sim p_{\text{g}}} f(x) \tag{11}$$

$$= \max_{w\in\Omega, v} < v, \mathbb{E}_{x\sim p_{\text{data}}} \Phi_w(x) - \mathbb{E}_{x\sim p_{\text{g}}} \Phi_w(x) > \tag{12}$$

$$= \max_{w\in\Omega} \max_{v} < v, \mathbb{E}_{x\sim p_{\text{data}}} \Phi_w(x) - \mathbb{E}_{x\sim p_{\text{g}}} \Phi_w(x) > \tag{13}$$

In Equation 13, the range of v determines the semantic meanings of the corresponding IPM metrics. From now on, we discuss IPM metric variants such as the Wasserstein metric, maximum mean discrepancy (MMD), and the Fisher metric based on Equation 13.

2.1.2.1 Wasserstein GAN

Wasserstein GAN (WGAN) [5] presents significant studies regarding the distance between $p_{\text{data}}(x)$ and $p_{\text{g}}(x)$. GAN learns the generator function g_θ that transforms a latent variable z into $p_{\text{g}}(x)$ rather than directly learning the probability distribution $p_{\text{data}}(x)$ itself. A measure between $p_{\text{g}}(x)$ and $p_{\text{data}}(x)$ thus is required to train g_θ. WGAN suggests the Earth-mover (EM) distance which is also called the Wasserstein distance, as a measure of the discrepancy between the two distributions. The Wasserstein distance is defined as follows:

$$W(p_{\text{data}}, p_{\text{g}}) = inf_{\gamma\in\sqcap(p_{\text{data}}, p_{\text{g}})}\ \mathbb{E}_{(x,y)\sim\gamma}\left[\|x-y\|\right] \tag{14}$$

where $\sqcap(p_{\text{data}}, p_{\text{g}})$ denotes the set of all joint distributions where the marginals of $\gamma(x, y)$ are $p_{\text{data}}(x)$ and $p_{\text{g}}(x)$ respectively.

Probability distributions can be interpreted as the amount of mass they place at each point, and EM distance is the minimum total amount of work required to transform $p_{\text{data}}(x)$ into $p_{\text{g}}(x)$.

From this view, calculating the EM distance is equivalent to finding a transport plan $\gamma(x, y)$, which defines how we distribute the amount of mass from $p_{\text{data}}(x)$ over $p_{\text{g}}(y)$. Therefore, a marginality condition can be interpreted that $p_{\text{data}}(x) = \int_y \gamma(x, y)dy$ is the amount of mass to move from point x and $p_{\text{g}}(y) = \int_x \gamma(x, y)dx$ is the amount of mass to be stacked at the point y. Because work is defined as the amount of mass times the distance it moves, we have to multiply the Euclidean distance $\|x - y\|$ by $\gamma(x, y)$ at each point x, y and the minimum amount of work is derived as Equation 14.

The benefit of the EM distance over other metrics is that it is a more sensible objective function when learning distributions with the support of a low-dimensional manifold. The article on WGAN shows that EM distance is the weakest convergent metric in that the convergent sequence under the EM distance does not converge under other metrics and it is continuous and differentiable almost everywhere under the Lipschitz condition, which standard feed-forward neural networks satisfy. Thus, EM distance results in a more tolerable measure than do other distances such as KLD and total variance distance regarding convergence of the distance.

As the inf term in Equation 14 is intractable, it is converted into a tractable equation via Kantorovich-Rubinstein duality with the Lipschitz function class [103], [46]; *i.e.*, $f : X \to R$, satisfying $d_R(f(x_1), f(x_2)) \leq 1 \times d_X(x_1, x_2), \forall x_1, x_2 \in X$ where d_X denotes the distance metric in the domain X. A duality of Equation 14 is as follows:

$$W(p_{\text{data}}, p_{\text{g}}) = sup_{|f|_L \leq 1} \mathbb{E}_{x \sim p_{\text{data}}}[f(x)] - \mathbb{E}_{x \sim p_{\text{g}}}[f(x)] \tag{15}$$

Consequently, if we parametrize the critic f with w to be a 1-Lipschitz function, the formulation becomes to a minimax problem in that we train f_w first to approximate $W(p_{\text{data}}, p_{\text{g}})$ by searching for the maximum as in Equation 15, and minimize such approximated distance by optimizing the generator g_θ. To guarantee that f_w is a Lipschitz function, weight clipping is conducted for every update of w to ensure that the parameter space of w lies in a compact space. It should be noted that $f(x)$ is called the critic because it does not explicitly classify its inputs as the discriminator, but rather scores its input.

2.1.2.2 Variants of WGAN

WGAN with gradient penalty (WGAN-GP) [44] points out that the weight clipping for the critic while training WGAN incurs a pathological behavior of the discriminator and suggests adding a penalizing term of the gradient's norm instead of the weight clipping. It shows that guaranteeing the Lipschitz condition for the critic via weight clipping constraints the critic in a very limited subset of all Lipschitz functions; this biases the critic toward a simple function. The weight clipping also creates a gradient problem as it pushes weights to the extremes of the clipping range. Instead of the weight clipping, adding a gradient penalty term to Equation 15 for the purpose of implementing the Lipschitz condition by directly constraining the gradient of the critic has been suggested [44].

In addition, loss sensitive GAN (LS-GAN) [102] also uses a Lipschitz constraint like WGAN but with a different method. It learns loss function L_θ instead of the critic such that the loss of a real sample should be smaller than a generated sample by a data-dependent margin, leading to more focus on fake samples whose margin is high. Moreover, LS-GAN assumes that the density of real samples $p_{\text{data}}(x)$ is Lipschitz continuous so that nearby data do not abruptly change. Therefore, the reason for adopting the Lipschitz condition is independent of WGAN's Lipschitz condition. The article on LS-GAN discusses that the nonparametric assumption that the model should have infinite capacity proposed by Goodfellow et al. [38] is too harsh a condition to satisfy even for deep neural networks and causes various problems in training; hence, it constrains a model to lie in Lipschitz continuous function space while WGAN's Lipschitz condition comes from the Kantorovich-Rubinstein duality and only the critic is constrained. In addition, LS-GAN uses a weight-decay regularization technique to impose the weights of a model to lie in a bounded area to ensure the Lipschitz function condition.

Another GAN related to the Wasserstein distance is relaxed Wasserstein GAN (RWGAN) [45]. It proposes a relaxed Wasserstein distance which is a combination of Bregman divergence [7] and the Wasserstein distance. It uses a *relaxed* term as it aims to generalize the Wasserstein-L^2 distance where the distance term $\|x - y\|$ in Equation 14 is replaced with the Bregman divergence with a convex function.

2.1.2.3 Mean feature matching

From Equation 10, we can generalize several IPM metrics under the measure of the inner product. If we constrain v with p norm where p is a nonnegative integer, we can derive Equation 18 as a feature matching problem as follows by adding the $\|v\|_p \leq 1$ condition where $\|v\|_p = \{\Sigma_{i=1}^{m} v_i^p\}^{1/p}$. It should be noted that, with conjugate exponent q of p such that $\frac{1}{p} + \frac{1}{q} = 1$, the dual norm of norm p satisfies $\|x\|_q = sup\{< v, x >: \|v\|_p \leq 1\}$ by Holder's inequality [125]. Motivated by this dual norm property [88], we can derive a l_q mean matching problem as follows:

$$d_{\mathcal{F}}(p_{\text{data}}, p_{\text{g}}) = \max_{w \in \Omega} \max_{\|v\|_p \leq 1} < v, \mathbb{E}_{x \sim p_{\text{data}}} \Phi_w(x) - \mathbb{E}_{x \sim p_{\text{g}}} \Phi_w(x) > \tag{16}$$

$$= \max_{w \in \Omega} \| \mathbb{E}_{x \sim p_{\text{data}}} \Phi_w(x) - \mathbb{E}_{x \sim p_{\text{g}}} \Phi_w(x) \|_q \tag{17}$$

$$= \max_{w \in \Omega} \|\mu_w(p_{\text{data}}) - \mu_w(p_{\text{g}})\|_q \tag{18}$$

where $\mathbb{E}_{x \sim \mathcal{P}} \Phi_w(x) = \mu_w(\mathcal{P})$ denotes an embedding mean from distribution $\mathcal{P}$ represented by a neural network Φ_w.

In terms of WGAN, WGAN uses the 1-Lipschitz function class condition by the weight clipping. The weight clipping indicates that the infinite norm $\|v\|_\infty = max_i |v_i|$ is constrained. Thus, WGAN can be interpreted as an l_1 mean feature matching problem. Mean and covariance feature matching GAN (McGAN) [88] extended this concept to match not only the l_q mean feature but also the second order moment feature by using the singular value decomposition concept; it aims to also maximize an embedding covariance discrepancy between $p_{\text{data}}(x)$ and $p_{\text{g}}(x)$. Geometric GAN [75] shows that the McGAN framework is equivalent to a support vector machine (SVM) [111], which separates the two distributions with a hyperplane that maximizes the margin. It encourages the discriminator to move away from the separating hyperplane and the generator to move toward the separating hyperplane. However, such high-order moment matching requires complex matrix computations. MMD addresses this problem with a kernel technique which induces an approximation of high-order moments and can be analyzed in a feature matching framework.

2.1.2.4 Maximum mean discrepancy (MMD)

Before describing MMD methodology, we must first list some mathematical facts. A Hilbert space $\mathcal{H}$ is a complete vector space with the metric endowed by the inner product in the space. Kernel k is defined as $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $k(y, x) = k(x, y)$. Then, for any given positive definite kernel $k(\cdot, \cdot)$, there exists a unique space of functions $f : \mathcal{X} \to \mathbb{R}$ called the reproducing kernel Hilbert space (RKHS), $\mathcal{H}_k$ which is a Hilbert space and satisfies $< f, k(\cdot, x) >_{\mathcal{H}_\mathcal{K}} = f(x)$, $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}_k$ (so called reproducing property).

MMD methodology can be seen as feature matching under the RKHS space for some given kernel with an additional function class restriction such that $\mathcal{F} = \{f | \|f\|_{\mathcal{H}_k} \leq 1\}$. It can be related to Equation 10 in that the RKHS space is a completion of $\{f | f(x) = \Sigma_{i=1}^{n} a_i \Phi_{x_i}(x), a_i \in \mathbb{R}, x_i \in \mathcal{X}\}$ where $\Phi_{x_i}(x) = k(x, x_i)$. Therefore, we can formulate MMD methodology as another mean feature matching from Equation 11 as follows:

$$d_{\mathcal{F}}(p_{\text{data}}, p_{\text{g}}) = \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{x \sim p_{\text{g}}} f(x) \tag{19}$$

$$= \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \{< f, \mathbb{E}_{x \sim p_{\text{data}}} \Phi_x(\cdot) - \mathbb{E}_{x \sim p_{\text{g}}} \Phi_x(\cdot) >_{\mathcal{H}_k}\} \tag{20}$$

$$= \|\mu(p_{\text{data}}) - \mu(p_{\text{g}})\|_{\mathcal{H}_k} \tag{21}$$

where $\mu(\mathcal{P}) = \mathbb{E}_{x \sim \mathcal{P}} \Phi_x(\cdot)$ denotes a kernel embedding mean and Equation 20 comes from the reproducing property.

From Equation 21, $d_{\mathcal{F}}$ is defined as the maximum kernel mean discrepancy in RKHS. It is widely used in statistical tasks, such as a two sample test to measure a dissimilarity. Given $p_{\text{data}}(x)$ with mean $\mu_{p_{\text{data}}}$, $p_{\text{g}}(x)$ with mean $\mu_{p_{\text{g}}}$ and kernel k, the square of the MMD distance $M_k(p_{\text{data}}, p_{\text{g}})$ can be reformulated as follows:

$$M_k(p_{\text{data}}, p_{\text{g}}) = \|\mu_{p_{\text{data}}} - \mu_{p_{\text{g}}}\|^2_{\mathcal{H}_k} = \mathbb{E}_{x, x' \sim p_{\text{data}}}[k(x, x')] - 2\,\mathbb{E}_{x \sim p_{\text{data}}, y \sim p_{\text{g}}}[k(x, y)] + \mathbb{E}_{y, y' \sim p_{\text{g}}}[k(y, y')] \tag{22}$$

Generative moment matching networks (GMMN) [73] suggest directly minimizing MMD distance with a fixed Gaussian kernel $k(x, x') = exp(-\|x - x'\|^2)$ by optimizing $\min_\theta M_k(p_{\text{data}}, p_{\text{g}})$. It appears quite dissimilar to the standard GAN [38] because there is no discriminator that estimates the discrepancy between two distributions. Unlike GMMN, MMDGAN [71] suggests adversarial kernel learning by not fixing the kernel but learning it itself. They replace the fixed Gaussian kernel with a composition of a Gaussian kernel and an injective function f_ϕ as follows: $\tilde{k}(x, x') = exp(-\|f_\phi(x) - f_\phi(x')\|^2)$ and learn a kernel to maximize the mean discrepancy. An objective function with optimizing kernel k then becomes $\min_\theta \max_\phi M_{\tilde{k}}(p_{\text{data}}, p_{\text{g}})$ and now is similar to the standard GAN objective as in Equation 1. To enforce f_ϕ modeled with the neural network to be injective, an autoencoder satisfying $f_{decoder} \approx f^{-1}$ is adopted for f_ϕ.

Similar to other IPM metrics, MMD distance is continuous and differentiable almost everywhere in θ. It can also be understood under the IPM framework with function class $\mathcal{F} = \mathcal{H}_K$ as discussed above. By introducing an RKHS with kernel k, MMD distance has an advantage over other feature matching metrics in that kernel k can represent various feature space by mapping input data x into other feature space. In particular, MMDGAN can also be connected with WGAN when f_ϕ is composited to a linear kernel with an output dimension of 1 instead of a Gaussian kernel. The moment matching technique using the Gaussian kernel also has an advantage over WGAN in that it can match even an infinite order of moments since exponential form can be represented as an infinite order via Taylor expansion while WGAN can be treated as a first-order moment matching problem as discussed above. However, a great disadvantage of measuring MMD distance is that computational cost grows quadratically as the number of samples grows [5].

Meanwhile, CramerGAN [9] argues that the Wasserstein distance incurs biased gradients, suggesting the energy distance between two distributions. In fact, it measures energy distance indirectly in the data manifold but with transformation function h. However, CramerGAN can be thought of as the distance in the kernel embedded space of MMDGAN, which forces h to be injective by the additional autoencoder reconstruction loss as discussed above.

2.1.2.5 Fisher GAN

In addition to standard IPM in Equation 9, Fisher GAN [87] incorporates a data-dependent constraint by the following equations:

$$d_{\mathcal{F}}(p_{\text{data}}, p_{\text{g}}) = \sup_{f \in \mathcal{F}} \frac{\mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{x \sim p_{\text{g}}} f(x)}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_{\text{data}}} f^2(x) + \mathbb{E}_{x \sim p_{\text{g}}} f^2(x))}} \tag{23}$$

$$= \sup_{f \in \mathcal{F}, \frac{1}{2}[\mathbb{E}_{x \sim p_{\text{data}}} f^2(x) + \mathbb{E}_{x \sim p_{\text{g}}} f^2(x)] = 1} \mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{x \sim p_{\text{g}}} f(x) \tag{24}$$

Equation 23 is motivated by fisher linear discriminant analysis (FLDA) [135] which not only maximizes the mean difference but also reduces the total with-in class variance of two distributions. Equation 24 follows from the constraining numerator of Equation 23 to be 1. It is also, as are other IPM metrics, interpreted as a mean feature matching problem under the somewhat different constraints. Under the definition of Equation 10, Fisher GAN can be converted into another mean feature matching problem with second order moment constraint. A mean feature matching problem derived from the FLDA concept is as follows:

$$d_{\mathcal{F}}(p_{\text{data}}, p_{\text{g}}) = \max_{w \in \Omega} \max_{v} \frac{< v, \mathbb{E}_{x \sim p_{\text{data}}} \Phi(x) - \mathbb{E}_{x \sim p_{\text{g}}} \Phi(x) >}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_{\text{data}}} f^2(x) + \mathbb{E}_{x \sim p_{\text{g}}} f^2(x))}} \tag{25}$$

$$= \max_{w \in \Omega} \max_{v} \frac{< v, \mathbb{E}_{x \sim p_{\text{data}}} \Phi(x) - \mathbb{E}_{x \sim p_{\text{g}}} \Phi(x) >}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_{\text{data}}} v^T \Phi(x) \Phi(x)^T v + \mathbb{E}_{x \sim p_{\text{g}}} v^T \Phi(x) \Phi(x)^T v)}} \tag{26}$$

$$= \max_{w \in \Omega} \max_{v} \frac{< v, \mu_w(p_{\text{data}}) - \mu_w(p_{\text{g}}) >}{\sqrt{v^T (\frac{1}{2} \sum_w (p_{\text{data}}) + \frac{1}{2} \sum_w (p_{\text{g}}) + \gamma I_m) v}} \tag{27}$$

$$= \max_{w \in \Omega} \max_{v, v^T (\frac{1}{2} \sum_w (p_{\text{data}}) + \frac{1}{2} \sum_w (p_{\text{g}}) + \gamma I_m) v = 1} < v, \mu_w(p_{\text{data}}) - \mu_w(p_{\text{g}}) > \tag{28}$$

where $\mu_w(\mathcal{P}) = \mathbb{E}_{x \sim \mathcal{P}} \Phi_w(x)$ denotes an embedding mean and $\sum_w(\mathcal{P}) = \mathbb{E}_{x \sim \mathcal{P}} \Phi_w(x) \Phi_w^T(x)$ denotes an embedding covariance for the probability $\mathcal{P}$.

Equation 26 can be induced by using the inner product of f defined as in Equation 10. γI_m of Equation 27 is an m by m identity matrix that guarantees a numerator of the above equations not to be zero. In Equation 28, Fisher GAN aims to find the embedding direction v which maximizes the mean discrepancy while constraining it to lie in a hyperellipsoid as $v^T(\frac{1}{2}\sum_w(p_{\text{data}}) + \frac{1}{2}\sum_w(p_{\text{g}}) + \gamma I_m)v = 1$ represents. It naturally derives the Mahalanobis distance [21] which is defined as a distance between two distributions given a positive definite matrix such as a covariance matrix of each class. More importantly, Fisher GAN has advantages over WGAN. It does not impose a data independent constraint such as weight clipping which makes training too sensitive on the clipping value, and it has computational benefit over the gradient penalty method in WGAN-GP [44] as the latter must compute gradients of the critic while Fisher GAN computes covariances.

2.1.2.6 Comparison to f-divergence

The f-divergence family, which can be defined as in Equation 3 with a convex function f, has restrictions in that as the dimension d of the data space $x \in \mathcal{X} = R^d$ increases, the f-divergence is highly difficult to estimate, and the supports of two distributions tends to be unaligned, which leads a divergence value to infinity [122]. Even though Equation 6 derives a variational lower bound of Equation 3 which looks very similar to Equation 9, the tightness of the lower bound to the true divergence is not guaranteed in practice and can incur an incorrect, biased estimation.

Sriperumbudur et al. [122] showed that the only non-trivial intersection between the f-divergence family and the IPM family is total variation distance; therefore, the IPM family does not inherit the disadvantages of f-divergence. They also proved that IPM estimators using finite i.i.d. samples are more consistent in convergence whereas the convergence of f-divergence is highly dependent on data distributions.

Consequently, employing an IPM family to measure distance between two distributions is advantageous over using an f-divergence family because IPM families are not affected by data dimension and consistently converge to the true distance between two distributions. Moreover, they do not diverge even though the supports of two distributions are disjointed. In addition, Fisher GAN [87] is also equivalent to the Chi-squared distance [136], which can be covered by an f-divergence framework. However, with a data dependent constraint, Chi-squared distance can use IPM family characteristics, so it is more robust to unstable training of the f-divergence estimation.

2.1.3 Auxiliary object functions

In Section 2.1.1 and Section 2.1.2, we demonstrated various objective functions for adversarial learning. Concretely, through a minimax iterative algorithm, the discriminator estimates a specific kind of distance between $p_{\text{g}}(x)$ and $p_{\text{data}}(x)$. The generator reduces the estimated distance to make two distributions closer. Based on this adversarial objective function, a GAN can incorporate with other types of objective functions to help the generator and the discriminator stabilize during training or to perform some kinds of tasks such as classification. In this section, we introduce auxiliary object functions attached to the adversarial object function, mainly a reconstruction objective function and a classification objective function.

2.1.3.1 Reconstruction object function

Reconstruction is to make an output image of a neural network to be the same as an original input image of a neural network. The purpose of the reconstruction is to encourage the generator to preserve the contents of the original input image [14, 60, 107, 128, 150] or to adopt auto-encoder architecture for the discriminator [11, 133, 148]. For a reconstruction objective function, mostly the L1 norm of the difference of the original input image and the output image is used.

When the reconstruction objective term is used for the generator, the generator is trained to maintain the contents of the original input image. In particular, this operation is crucial for tasks where semantic and several modes of the image should be maintained, such as image translation [60, 150] (detailed in Section 4.1.1.2) and auto-encoder reconstruction [14, 107, 128] (detailed in Section 3.2). The intuition of using reconstruction loss for the generator is that it guides the generator to restore the original input in a supervised learning manner. Without a reconstruction loss, the generator is to simply fool the discriminator, so there is no reason for the generator to maintain crucial parts of the input. As a supervised learning approach, the generator treats the

original input image as label information, so we can reduce the space of possible mappings of the generator in a way we desire.

There are some GAN variants using a reconstruction objective function for the discriminator, which is naturally derived from auto-encoder architecture for the discriminator [11, 133, 148]. These GANs are based on the aspect that views the discriminator as an energy function, and will be detailed in Section 2.2.3.

2.1.3.2 Classification object functions

A cross entropy loss for a classification is widely added for many GAN applications where labeled data exists, especially semi-supervised learning and domain adaptation. Cross entropy loss can be directly applied to the discriminator, which gives the discriminator an additional role of classification [93, 120]. Other approaches [2, 13, 70, 145] adopt classifier explicitly, training the classifier jointly with the generator and the discriminator through a cross entropy loss (detailed in Section 4.3 and 4.4).

2.2 Architecture

An architecture of the generator and the discriminator is important as it highly influences the training stability and performance of GAN. Various papers adopt several techniques such as batch normalization, stacked architecture, and multiple generators and discriminators to promote adversarial learning. We start with deep convolutional GAN (DCGAN) [104], which provides a remarkable benchmark architecture for other GAN variants.

2.2.1 DCGAN

DCGAN provides significant contributions to GAN in that its suggested convolution neural network (CNN) [65] architecture greatly stabilizes GAN training. DCGAN suggests an architecture guideline in which the generator is modeled with a transposed CNN [26] as shown in Figure 3; the discriminator is modeled with a CNN with an output dimension 1. It also proposes other techniques such as batch normalization and types of activation functions for the generator and the discriminator to help stabilize the GAN training. As it solves the instability of training GAN only through architecture, it becomes a baseline for modeling various GANs proposed later. For example, Im et al. [54] uses a recurrent neural network (RNN) [40] to generate images motivated by DCGAN. By accumulating images of each time step output of DCGAN and combining several time step images, it produces higher visual quality images.

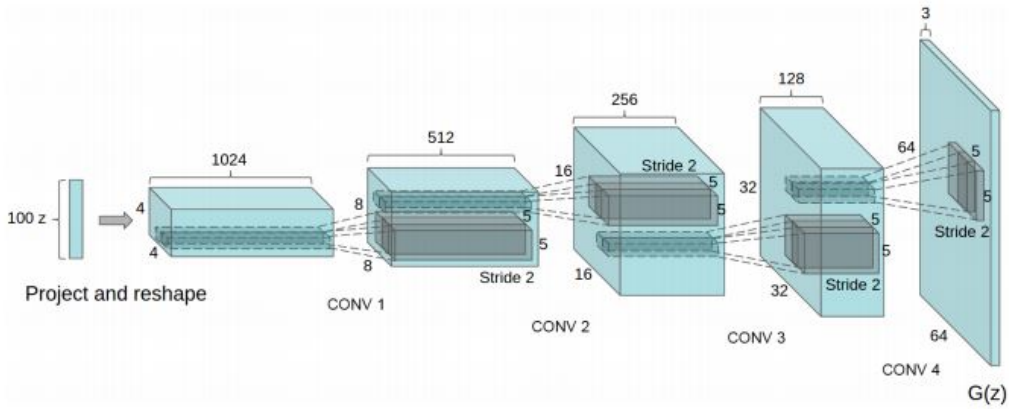


Figure 3: Basic generator architecture proposed by DCGAN [104]. The latent variables are projected to spatial convolutional representations, and conducted fractional-strided convolutions to generate a 64 × 64 image. Images from DCGAN [104]

2.2.2 Hierarchical architecture

In this section, we describe GAN variants that stack multiple generator-discriminator pairs. Commonly, these GANs generate samples in multiple stages to generate large-scale and high-

quality samples. The generator of each stage is utilized or conditioned to help the next stage generator to better produce samples as shown in Figure 4.

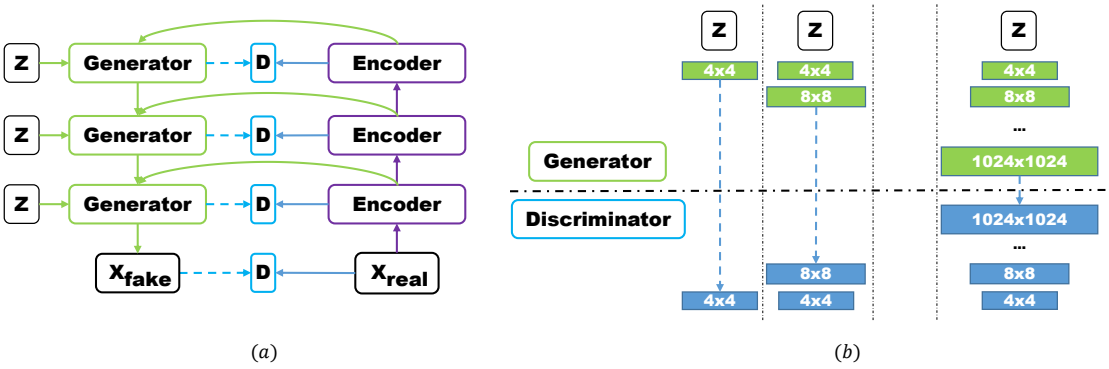


Figure 4: Illustrations of (a) StackedGAN [52] and (b) Progressive GAN [59].

2.2.2.1 Hierarchy using multiple GAN pairs

StackedGAN [52] attempts to learn a hierarchical representation by stacking several generator-discriminator pairs. For each layer of a generator stack, there exists the generator which produces level-specific representation, the corresponding discriminator training the generator adversarially at each level and an encoder which generates the semantic features of real samples. Figure 4a shows a flowchart of StackedGAN. Each generator tries to produce a plausible feature representation that can deceive the corresponding discriminator, given previously generated features and the corresponding hierarchically encoded features.

In addition, Gang of GAN (GoGAN) [57] proposes to improve WGAN [5] by adopting multiple WGAN pairs. For each stage, it changes the Wasserstein distance to a margin-based Wasserstein disitance as $[D(G(z)) - D(x) + m]^+$ so the discriminator focuses on generated samples whose gap $D(x) - D(G(z))$ is less than m. In addition, GoGAN adopts ranking loss for adjacent stages which induces the generator in later stages to produce better results than the former generator by using a smaller margin at the next stage of the generation process. By progressively moving stages, GoGAN aims to gradually reduce the gap between $p_{\text{data}}(x)$ and $p_{\text{g}}(x)$.

2.2.2.2 Hierarchy using a single GAN

Generating high resolution images is highly challenging since a large scale generated image is easily distinguished by the discriminator, so the generator often fails to be trained. Moreover, there is a memory issue in that we are forced to set a low mini-batch size due to the large size of neural networks. Therefore, some studies adopt hierarchical stacks of multiple generators and discriminators [28, 35, 53]. This strategy divides a large complex generator's mapping space step by step for each GAN pair, making it easier to learn to generate high resolution images. However, Progressive GAN [59] succeeds in to generating high resolution images in a single GAN, making training faster and more stable.

Progressive GAN generates high resolution images by stacking each layer of the generator and the discriminator incrementally as shown in Figure 4b. It starts training to generate a very low spatial resolution (*e.g.* 4×4), and progressively doubles the resolution of generated images by adding layers to the generator and the discriminator incrementally. With this stacking procedure, Progressive GAN notably generates large scale images sharply as shown in Figure 5. In addition, it proposes various training techniques such as pixel normalization, equalized learning rate and mini-batch standard deviation, all of which help GAN training to become more stable.

2.2.3 Auto encoder architecture

An auto encoder is a neural network for unsupervised learning, It assigns its input as a target value, so it is trained in a self-supervised manner. The reason for self-reconstruction is to encode a compressed representation or features of the input, which is widely utilized with a decoder. Its usage will be detailed in Section 3.2.

Figure 5: CelebA-HQ results of Progressive GAN [59]. Leftmost images are size of 1024 ×1024. As mentioned above, Progressive GAN amazingly produces large scale images very realistic, and this results was possible due to a hierarchical stacking. Images from Progressive GAN [59].

In this section, we describe GAN variants which adopt an auto encoder as the discriminator. These GANs view the discriminator as an energy function, not a probabilistic model that distinguishes its input as real or fake. An energy model assigns a low energy for a sample lying near the data manifold (a high data density region), while assigning a high energy for a contrastive sample lying far away from the data manifold (a low data density region). These variants are mainly Energy Based GAN (EBGAN), Boundary Equilibrium GAN (BEGAN) and Margin Adaptation GAN (MAGAN), all of which frame GAN as an energy model.

Since an auto encoder is utilized for the discriminator, a pixelwise reconstruction loss between an input and an output of the discriminator is naturally adopted for the discriminator's energy function and is defined as follows:

$$D(v) = \|v - AE(v)\| \tag{29}$$

where $AE : R^{N_x} \Rightarrow R^{N_x}$ denotes an auto encoder and R^{N_x} represents the dimension of an input and an output of an auto encoder. It is noted that $D(v)$ in Equation 29 is the pixelwise L1 loss for an autoencoder which maps an input $v \in R^{N_x}$ into a positive real number R^+.

A discriminator with an energy $D(v)$ is trained to give a low energy for a real v and a high energy for a generated v. From this point of view, the generator produces a contrastive sample for the discriminator, so that the discriminator is forced to be regularized near the data manifold. Simultaneously, the generator is trained to generate samples near the data manifold since the discriminator is encouraged to reconstruct only real samples. Table 4 presents the summarized details of BEGAN, EBGAN and MAGAN. L_G and L_D indicate the generator loss and the discriminator loss, respectively, and $[t]^+ = \max(0, t)$ represents a maximum value between t and 0, which act as a hinge.

Table 4: An autoencoder based GAN variants (BEGAN, EBGAN and MAGAN).

	Objective function	Details
BEGAN [11]	$L_D = D(x) - k_t D(G(z))$ $L_G = D(G(z))$ $k_{t+1} = k_t + \alpha(\gamma D(x) - D(G(z)))$	Wasserstein distance between loss distributions
EBGAN [148]	$L_D = D(x) + [m - D(G(z))]^+$ $L_G = D(G(z))$	Total Variance$(p_{\text{data}}, p_\theta)$
MAGAN [133]	$L_D = D(x) + [m - D(G(z))]^+$ $L_G = D(G(z))$	Margin m is adjusted in EBGAN's training

Boundary equilibrium GAN (BEGAN) [11] uses the fact that pixelwise loss distribution follows a normal distribution by CLT. It focuses on matching loss distributions through Wasserstein distance and not on directly matching data distributions. In BEGAN, the discriminator has two roles: one is to reconstruct real samples sufficiently and the other is to balance the generator and the discriminator via an equilibrium hyperparameter $\gamma = \mathbb{E}[L(G(z))]/\mathbb{E}[L(x)]$. γ is fed into an objective function to prevent the discriminator from easily winning over the generator; therefore, this balances the power of the two components.

Energy-based GAN (EBGAN) [148] interprets the discriminator as an energy agent, which assigns low energy to real samples and high energy to generated samples. Through the $[m - L(G(z))]^+$ term in an objective function, the discriminator ignores generated samples with higher energy than m so the generator attempts to synthesize samples that have lower energy than m to fool the discriminator, which allows that mechanism to stabilize training. Margin adaptation GAN (MAGAN) [133] takes a similar approach to EBGAN, where the only difference is that MAGAN does not fix the margin m. MAGAN shows empirically that the energy of the generated sample fluctuates near the margin m and that phenomena with a fixed margin make it difficult to adapt to the changing dynamics of the discriminator and generator. MAGAN suggests that margin m should be adapted to the expected energy of real data, thus, m is monotonically reduced, so the discriminator reconstructs real samples more efficiently.

In addition, because the total variance distance belongs to an IPM family with the function class $\mathcal{F} = \{f : \|f\|_\infty = \sup_x |f(x)| \leq 1\}$, it can be shown that EBGAN is equivalent to optimizing the total variance distance by using the fact that the discriminator's output for generated samples is only available for $0 \leq D \leq m$ [5]. Because the total variance is the only intersection between IPM and f-divergence [122], it inherits some disadvantages for estimating f-divergence as discussed by Arjovsky et al. [5] and Sriperumbudur et al. [122].

Figure 6: Random images sampled from the generator at varying $\gamma \in \{0.3, 0.5, 0.7\}$ of BEGAN [11]. Samples at lower γ shows similar images. At high γ values, image diversity seems to increase, but have some artifacts. Images from BEGAN [11].

2.3 Obstacles in Training GAN

In this section, we discuss theoretical and practical issues related to the training dynamics of GAN. We first evaluate a theoretical problem of the standard GAN, which is incurred from the fact that the discriminator of GAN aims to approximate the JSD [38] between $p_{\text{data}}(x)$ and $p_\theta(x)$ and the generator of GAN tries to minimize the approximated JSD, as discussed in Section 2.3.1. We then discuss practical issues, especially a mode collapse problem where the generator fails to capture the diversity of real samples, and generates only specific types of real samples, as discussed in Section 2.3.2.

2.3.1 Theoretical issues

As mentioned in Section 1, the traditional generative model is to maximize a likelihood of $p_{\text{g}}(x)$. It can be shown that maximizing the log likelihood is equivalent to minimizing the Kullback-Leibler Divergence (KLD) between $p_{\text{data}}(x)$ and $p_\theta(x)$ as the number of samples m increases:

$$\theta^\star = \underset{\theta}{\text{argmax}} \lim_{m\to\infty} \frac{1}{m} \sum_{i=1}^{m} \log p_{\text{g}}(x^i) \tag{30}$$

$$= \underset{\theta}{\text{argmax}} \int_x p_{\text{data}}(x) \log p_{\text{g}}(x) dx \tag{31}$$

$$= \underset{\theta}{\text{argmin}} \int_x -p_{\text{data}}(x) \log p_{\text{g}}(x) dx + p_{\text{data}}(x) \log p_{\text{data}}(x) dx \tag{32}$$

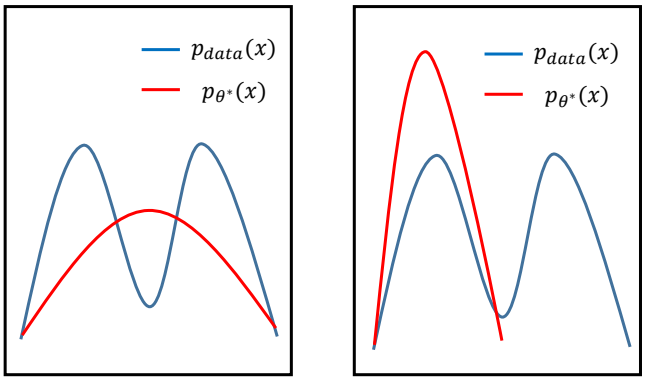
$$= \underset{\theta}{\text{argmin}} \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{g}}(x)} dx \tag{33}$$

$$= \underset{\theta}{\text{argmin}}\, KLD(p_{\text{data}}||p_{\text{g}}) \tag{34}$$

We note that we need to find an optimal parameter $\theta^\star$ for maximizing likelihood; therefore, argmax is used instead of max. In addition, we replace our model's probability distribution $p_\theta(x)$ with $p_{\text{g}}(x)$ for consistency of notation.

Equation 31 is established by the central limit theorem (CLT) [108] in that as m increases, the variance of the expectation of the distribution decreases. Equation 32 can be induced because $p_{\text{data}}(x)$ is not dependent on θ, and Equation 34 follows from the definition of KLD. Intuitively, minimizing KLD between these two distributions can be interpreted as approximating $p_{\text{data}}(x)$ with a large number of real training data because the minimum KLD is achieved when $p_{\text{data}}(x) = p_{\text{g}}(x)$.

Thus, the result of maximizing likelihood is equivalent to minimizing $\text{KLD}(p_{\text{data}}||p_{\text{g}})$ given infinite training samples. Because KLD is not symmetrical, minimizing $\text{KLD}(p_{\text{g}}||p_{\text{data}})$ gives a different result. Figure 7 from Goodfellow [37] shows the details of different behaviors of asymmetric KLD where Figure 7a shows minimizing $\text{KLD}(p_{\text{data}}||p_{\text{g}})$ and Figure 7b shows minimizing $\text{KLD}(p_{\text{g}}||p_{\text{data}})$ given a mixture of two Gaussian distributions $p_{\text{data}}(x)$ and the single Gaussian distribution $p_{\text{g}}(x)$. $\theta^\star$ in each figure denotes the argument minimum of each asymmetric KLD. For Figure 7a, the points where $p_{data} \neq 0$ contribute to the value of KLD and the other points at which p_{data} is small rarely affect the KLD. Thus, p_{g} becomes nonzero on the points where p_{data} is nonzero. Therefore, $p_{\theta^\star}(x)$ in Figure 7a is averaged for all modes of $p_{\text{data}}(x)$ as $KLD(p_{\text{data}}||p_{\text{g}})$ is more focused on covering all parts of $p_{\text{data}}(x)$. In contrast, for $\text{KLD}(p_{\text{g}}||p_{\text{data}})$, the points of which $p_{data} = 0$ but $p_{\text{g}} \neq 0$ contribute to a high cost. This is why $p_{\theta^\star}(x)$ in Figure 7b seeks to find an x which is highly likely from $p_{\text{data}}(x)$.



(a) $KLD(p_{\text{data}}||p_{\text{g}})$ (b) $KLD(p_{\text{g}}||p_{\text{data}})$

Figure 7: Different behavior of asymmetric KLD. Images reproduced from [37]

JSD has an advantage over the two asymmetric KLDs in that it accounts for both mode dropping and sharpness. It never explodes to infinity unlike KLD even though there exists a point x that lies outside of $p_{\text{g}}(x)$'s support which makes $p_{\text{g}}(x)$ equal 0. Goodfellow et al. [38] showed that the discriminator D aims to approximate $V(G, D^\star) = 2JSD(p_{\text{data}}||p_{\text{g}}) - 2\log 2$ for the fixed generator G between $p_{\text{g}}(x)$ and $p_{\text{data}}(x)$, where $D^\star$ is an optimal discriminator and $V(G, D)$ is defined in Equation 1. Concretely, if D is trained well so that it approximates $JSD(p_{\text{data}}||p_{\text{g}}) -$

$2\log 2$ sufficiently, training G minimizes the approximated distance. However, Arjovsky and Bottou [4] reveal mathematically why approximating $V(G, D^\star)$ does not work well in practice.

Arjovsky and Bottou [4] proved why training GAN is fundamentally unstable. When the supports of two distributions are disjointed or lie in low-dimensional manifolds, there exists the perfect discriminator which classifies real or fake samples perfectly and thus, the gradient of the discriminator is 0 at the supports of the two distributions. It has been proven empirically and mathematically that $p_{\text{data}}(x)$ and $p_{\text{g}}(x)$ derived from z have a low-dimensional manifold in practice [89], and this fact allows D's gradient transferred to G to vanish as D becomes to perfectly classify real and fake samples. Because, in practice, G is optimized with a gradient based optimization method, D's vanishing (or exploding) gradient hinders G from learning enough through D's gradient feedback. Moreover, even with the alternate $-\log D(G(z))$ objective proposed in [38], minimizing an objective function is equivalent to simultaneously trying to minimize $\text{KLD}(p_{\text{g}}||p_{\text{data}})$ and maximize $\text{JSD}(p_{\text{g}}||p_{\text{data}})$. As these two objectives are opposites, this leads the magnitude and variance of D's gradients to increase as training progresses, causing unstable training and making it difficult to converge to equilibrium. To summarize, training the GAN is theoretically guaranteed to converge if we use an optimal discriminator $D^\star$ which approximates JSD, but this theoretical result is not implemented in practice when using gradient based optimization. In addition to the D's improper gradient problem discussed in this paragraph, there are two practical issues as to why GAN training suffers from nonconvergence.

2.3.2 Practical issues

First, we represent G and D as deep neural networks to learn parameters rather than directly learning $p_{\text{g}}(x)$ itself. Modeling with deep neural networks such as the multilayer perceptron (MLP) or CNN is advantageous in that the parameters of distributions can be easily learned through gradient descent using back-propagation. This does not require further distribution assumptions to produce an inference; rather, it can generate samples following $p_{\text{g}}(\text{x})$ through simple feed-forward. However, this practical implementation causes a gap with theory. Goodfellow et al. [38] provide theoretical convergence proof based on the convexity of probability density function in $V(G, D)$. However, as we model G and D with deep neural networks, the convexity does not hold because we now optimize in the parameter space rather than in the function space (where assumed theoretical analysis lies). Therefore, theoretical guarantees do not hold anymore in practice. For a further issue related to parameterized neural network space, Arora et al. [6] discussed the existence of the equilibrium of GAN, and showed that a large capacity of D does not guarantee G to generate all real samples perfectly, meaning that an equilibrium may not exist under a certain finite capacity of D.

A second problem is related to an iterative update algorithm suggested in Goodfellow et al. [38]. We wish to train D until optimal for fixed G, but optimizing D in such a manner is computationally expensive. Naturally, we must train D in certain k steps and that scheme causes confusion as to whether it is solving a minimax problem or a maximin problem, because D and G are updated alternatively by gradient descent in the iterative procedure. Unfortunately, solutions of the minimax and maximin problem are not generally equal as follows:

$$\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D) \tag{35}$$

With a maximin problem, minimizing G lies in the inner loop in the right side of Equation 35. G is now forced to place its probability mass on the most likely point where the fixed nonoptimal D believes it likely to be real rather than fake. After D is updated to reject the generated fake one, G attempts to move the probability mass to the other most likely point for fixed D. In practice, real data distribution is normally a multi modal distribution but in such a maximin training procedure, G does not cover all modes of the real data distribution because G considers that picking only one mode is enough to fool D. Empirically, G tends to cover only a single mode or a few modes of real data distribution. This undesirable nonconvergent situation is called a mode collapse. A mode collapse occurs when many modes in the real data distribution are not represented in the generated samples, resulting in a lack of diversity in the generated samples. It can be simply considered as G being trained to be a non one-to-one function which produces a single output value for several input values.

Furthermore, the problem of the existence of the perfect discriminator we discussed in the above paragraph can be connected to a mode collapse. First, assume D comes to output almost 1 for all

real samples and 0 for all fake samples. Then, because D produces values near 1 for all possible modes, there is no need for G to represent all modes of real data probability. The theoretical and practical issues discussed in this section can be summarized as follows.

- Because the supports of distributions lie on low dimensional manifolds, there exists the perfect discriminator whose gradients vanish on every data point. Optimizing the generator may be difficult because it is not provided with any information from the discriminator.
- GAN training optimizes the discriminator for the fixed generator and the generator for fixed discriminator simultaneously in one loop, but it sometimes behaves as if solving a maximin problem, not a minimax problem. It critically causes a mode collapse. In addition, the generator and the discriminator optimize the same objective function $V(G, D)$ in opposite directions which is not usual in classical machine learning, and often suffers from oscillations causing excessive training time.
- The theoretical convergence proof does not apply in practice because the generator and the discriminator are modeled with deep neural networks, so optimization has to occur in the parameter space rather than in learning the probability density function itself.

2.3.3 Training techniques to improve GAN training

As demonstrated in Section 2.3.1 and 2.3.2, GAN training is highly unstable and difficult because GAN is required to find a Nash equilibrium of a non-convex minimax game with high dimensional parameters but GAN is typically trained with gradient descent [109]. In this section, we introduce some techniques to improve training of GAN, to make training more stable and produce better results.

- Feature matching [109]:
  This technique substitutes the discriminator's output in the objective function (Equation 1) with an activation function's output of an intermediate layer of the discriminator to prevent overfitting from the current discriminator. The generator then tries to capture discriminative features of real and fake samples. Feature matching does not aim on the discriminator's output, rather it guides the generator to see the statistics or features of real training data, in an effort to stabilize training.
- Label smoothing [109]:
  As mentioned previously, $V(G, D)$ is a binary cross entropy loss whose real data label is 1 and its generated data label is 0. However, since a deep neural network classifier tends to output a class probability with extremely high confidence [37], label smoothing encourages a deep neural network classifier to produce a more soft estimation by assigning label values lower than 1. Importantly, for GAN, label smoothing has to be made for labels of real data, not for labels of fake data, since, if not, the discriminator can act incorrectly [37].
- Spectral normalization [85]:
  As we see in Section 2.1.2.1 and 2.1.2.2, WGAN and Improved WGAN impose the discriminator to have Lipschitz continuity which constrain the magnitude of function differentiation. Spectral normalization aims to impose a Lipschitz condition for the discriminator in a different manner. Instead of adding a regularizing term or weight clipping, spectral normalization constrains the spectral norm of each layer of the discriminator where the spectral norm is the largest singular value of a given matrix. Since a neural network is a composition of multi layers, spectral normalization normalizes the weight matrices of each layer to make the whole network Lipschitz continuous. In addition, compared to the gradient penalty method proposed in Improved WGAN, spectral normalization is computationally beneficial since gradient penalty regularization directly controls the gradient of the discriminator.
- PatchGAN [55]:
  Thoroughly, PatchGAN is not a technique for stabilizing training of GAN. However, PatchGAN greatly helps to generate sharper results in various applications such as image translation [55, 150]. PatchGAN facilitates GAN's ability to capture high frequency parts of the image. Rather than producing a single output from the discriminator, which is a probability for its input's authenticity, PatchGAN [55] makes the discriminator produce a grid output,

not a single scalar as a conventional approach. For one element of the discriminator's output, its receptive field in the input image should be one small local patch in the input image, so the discriminator aims to distinguish each patch in the input image. To achieve this, one can remove the fully connected layer in the last part of the discriminator in the standard GAN, which outputs a single scalar. As the discriminator becomes fully convolution networks, it can be applied to the arbitrary size of the input image. As a matter of fact, PatchGAN is equivalent to adopting multiple discriminators for every patch of the image, making the discriminator help the generator to represent more sharp images locally.

2.4 Methods to Address Mode Collapse in GAN

Mode collapse which indicates the failure of GAN to represent various types of real samples is the main catastrophic problem of a GAN. From a perspective of the generative model, mode collapse is a critical obstacle for a GAN to be utilized in many applications, since the diversity of generated data needs to be guaranteed to represent the data manifold concretely. Unless multi modes of real data distribution are not represented by the generative model, such a model would be meaningless to use.

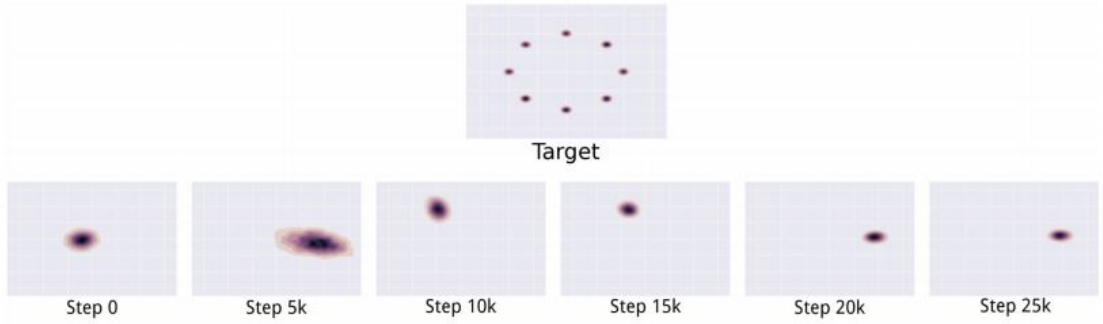


Figure 8: An illustration of the mode collapse problem. Images from Unrolled GAN [82].

Figure 8 shows a mode collapse problem for a toy example. A target distribution p_{data} has multi modes, which is a Gaussian mixture in two-dimensional space [82]. Figures in the lower row represent learned distribution p_{g} as the training progresses. As we see in Figure 8, the generator does not cover all possible modes of the target distribution. Rather, the generator covers only a single mode, switching between different modes as the training goes on. The generator learns to produce a single mode, believing that it can fool the discriminator. The discriminator counter acts the generator by rejecting the chosen mode. Then, the generator switches to another mode which is believed to be real. This training behavior keeps proceeding, and thus, the convergence to a distribution covering all the modes is highly difficult.

In this section, we present several studies that suggest methods to overcome the mode collapse problem. In Section 2.4.1, we demonstrate studies that exploit new objective functions to tackle a mode collapse, and in Section 2.4.2, we introduce studies which propose architecture modifications. Lastly, in Section 2.4.3, we describe mini-batch discrimination which is a notably and practically effective technique for the mode collapse problem.

2.4.1 Object function methods

Unrolled GAN [82] manages mode collapse with a surrogate objective function for the generator, which helps the generator predict the discriminator's response by unrolling the discriminator update k steps for the current generator update. As we see in the standard GAN [38], it updates the discriminator first for the fixed generator and then updates the generator for the updated discriminator. Unrolled GAN differs from standard GAN in that it updates the generator based on a k steps updated discriminator given the current generator update, which aims to capture how the discriminator responds to the current generator update. We see that when the generator is updated, it unrolls the discriminator's update step to consider the discriminator's k steps future response with respect to the generator's current update while updating the discriminator in the

same manner as the standard GAN. Since the generator is given more information about the discriminator's response, the generator spreads its probability mass to make it more difficult for the discriminator to react to the generator's behavior. It can be seen as empowering the generator because only the generator's update is unrolled, but it seems to be fair in that the discriminator can not be trained to be optimal in practice due to an infeasible computational cost while the generator is theoretically assumed to obtain enough information from the optimal discriminator.

Deep regret analytic GAN (DRAGAN) [64] suggests that a mode collapse occurs due to the existence of a spurious local Nash Equilibrium in the nonconvex problem. DRAGAN addresses this issue by proposing constraining gradients of the discriminator around the real data manifold. It adds a gradient penalizing term which biases the discriminator to have a gradient norm of 1 around the real data manifold. This method attempts to create linear functions by making gradients have a norm of 1. Linear functions near the real data manifolds form a convex function space, which imposes a global unique optimum. Note that this gradient penalty method is also applied to WGAN-GP [44]. They differ in that DRAGAN imposes gradient penalty constraints only to local regions around the real data manifold while Improved WGAN imposes gradient penalty constraints to almost everywhere around the generated data manifold and real data manifold, which leads to higher constraints than those of DRAGAN.

In addition, EBGAN proposes a repelling regularizer loss term to the generator, which encourages feature vectors in a mini-batch to be orthogonalized. This term is utilized with cosine similarities at a representation level of an encoder and forces the generator not to produce samples fallen in a few modes.

2.4.2 Architecture methods

Multi agent diverse GAN (MAD-GAN) [35] adopts multiple generators for one discriminator to capture the diversity of generated samples as shown in Figure 9a. To induce each generator to move toward different modes, it adopts a cosine similarity value as an additional objective term to make each generator produce dissimilar samples. This technique is inspired from the fact that as images from two different generators become similar, a higher similarity value is produced, thus, by optimizing this objective term, it may make each generator move toward different modes respectively. In addition, because each generator produces different fake samples, the discriminator's objective function adopts a soft-max cross entropy loss term to distinguish real samples from fake samples generated by multiple generators.

Mode regularized GAN (MRGAN) [14] assumes that mode collapse occurs because the generator is not penalized for missing modes. To address mode collapse, MRGAN adds an encoder which maps the data space $\mathcal{X}$ into the latent space $\mathcal{Z}$. Motivated from the manifold disjoint mentioned in Section 2.3.1, MRGAN first tries to match the generated manifold and real data manifold using an encoder. For manifold matching, the discriminator D_M distinguishes real samples x and its reconstruction $G \circ E(x)$, and the generator is trained with $D_M(G \circ E(x))$ with a geometric regularizer $d(x, G \circ E(x))$ where d can be any metric in the data space. A geometric regularizer is used to reduce the geometric distance in the data space, to help the generated manifold move to the real data manifold, and allow the generator and an encoder to learn how to reconstruct real samples. For penalizing missing modes, MRGAN adopts another discriminator D_D which distinguishes $G(z)$ as fake and $G \circ E(x)$ as real. Since MRGAN matches manifolds in advance with a geometric regularizer, this modes diffusion step can distribute a probability mass even to minor modes of the data space with the help of $G \circ E(x)$. An outline of MRGAN is illustrated in Figure 9b, where R denotes a geometric regularizing term (reconstruction).

2.4.3 Mini-batch Discrimination

Mini-batch discrimination [109] allows the discriminator to look at multiple examples in a mini-batch to avoid a mode collapse of the generator. The basic idea is that it encourages the discriminator to allow diversity directly in a mini-batch, not considering independent samples in isolation. To make the discriminator deal with not only each example but the correlation between other examples in a mini-batch simultaneously, it models a mini-batch layer in an intermediate layer of the discriminator, which calculates L1-distance based statistics of samples in a mini-batch. By adding such statistics to the discriminator, each example in a mini-batch can be estimated by how far or close to other examples in a mini-batch they are, and this information can be internally utilized by the discriminator, which helps the discriminator reflect samples' diversity to the output.

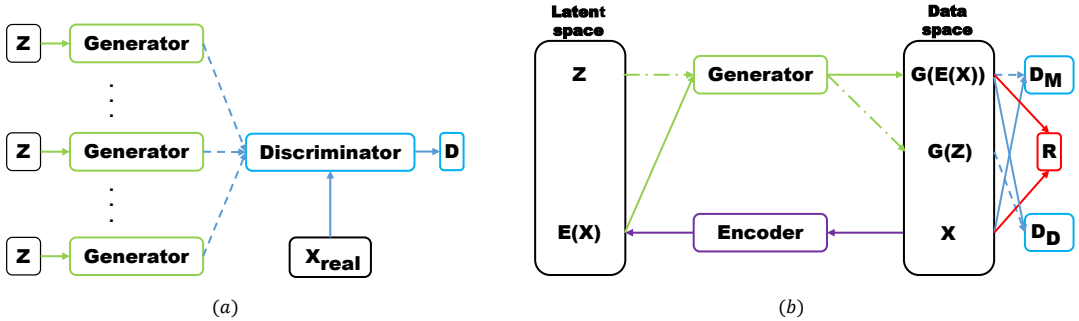


Figure 9: Illustrations of (a) MAD-GAN [35] and (b) MRGAN [14].

For the aspect of the generator, it tries to create statistics similar to those of real samples in the discriminator by adversarial learning procedure.

In addition, Progressive GAN [59] proposed a simplified version of mini-batch discrimination, which uses the mean of the standard deviation for each feature (channel) in each spatial location over the mini-batch. This way do not add trainable parameters which projects statistics of the mini-batch while maintaining its effectiveness for a mode collapse. To summarize, mini-batch discrimination reflects samples' diversity to the discriminator, helping the discriminator determine whether its input batch is real or fake.

3 Treating the Latent Space

Latent space, also called an embedding space, is the space in which a compressed representation of data lies. If we wish to change or reflect some attributes of an image (for example, a pose, an age, an expression or even an object of an image), modifying images directly in the image space would be highly difficult because the manifolds where the image distributions lie are high-dimensional and complex. Rather, manipulating in the latent space is more tractable because the latent representation expresses specific features of the input image in a compressed manner. In this section, we investigate how GAN handles latent space to represent target attributes and how a variational approach can be combined with the GAN framework.

3.1 Latent Space Decomposition

The input latent vector z of the generator is so highly entangled and unstructured that we do not know which vector point contains the specific representations we want. From this point of view, several papers suggest decomposing the input latent space to an input vector c, which contains the meaningful information and standard input latent vector z, which can be categorized into a supervised method and an unsupervised method.

3.1.1 Supervised Methods

Supervised methods require a pair of data and corresponding attributes such as the data's class label. The attributes are generally used as an additional input vector as explained below.

Conditional GAN (CGAN) [83] imposes a condition of additional information such as a class label to control the data generation process in a supervised manner by adding an information vector c to the generator and discriminator. The generator takes not only a latent vector z but also an additional information vector c and the discriminator takes samples and the information vector c so that it distinguishes fake samples given c. By doing so, CGAN can control the number of digits to be generated, which is impossible for standard GAN.

Miyato and Koyama [84] modified the method for using the information vector c in the discriminator. CGAN and the other CGAN-based methods simply concatenate c with the input. Unlike those, Miyato and Koyama [84] proposed a *projection* discriminator in which the information vector is multiplied by the vector output function of the input in the inner product form. This

architecture of the discriminator is supported by the assumption of simple model distributions but it may not fit well in the other rigorous distributions such as multimodal distributions.

Auxiliary classifier GAN (AC-GAN) [93] takes a somewhat different approach than CGAN. It is trained by minimizing the log-likelihood of class labels with the adversarial loss. The discriminator produces not only the probability that the input samples are from the real dataset but also the probability over the class labels. Figure 10 outlines CGAN and CGAN with a projection discriminator and ACGAN where CE denotes the cross entropy loss for the classification.

In addition, plug and play generative networks (PPGN) [91] are another type of generative model that produce data under a given condition. Unlike the other methods described above, PPGN does not use the labeled attributes while training the generator. Instead, PPGN learns the auxiliary classifier and the generator producing real-like data via adversarial learning independently. Then, PPGN produces the data for the given condition using the classifier and the generator through an MCMC-based sampler. An important characteristic of PPGN is that they can work as plug and play. When a classifier pretrained with the same data but different labels is given to the generator, the generator can synthesize samples under the condition without further training.

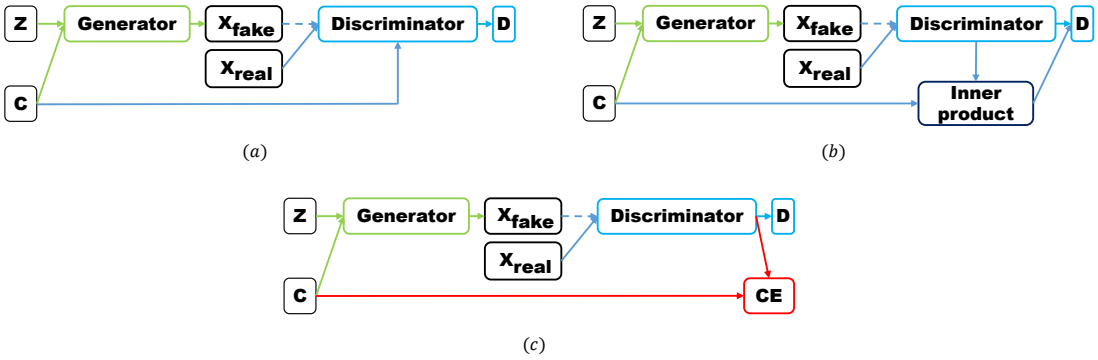


Figure 10: Illustrations of (a) CGAN [83], (b) CGAN with a projection discriminator [84] and (c) AC-GAN [93].

3.1.2 Unsupervised Methods

Different from the supervised methods discussed above, unsupervised methods do not exploit any labeled information. Thus, they require an additional algorithm to disentangle the meaningful features from the latent space.

InfoGAN [16] decomposes an input noise vector into a standard incompressible latent vector z and another latent variable c to capture salient semantic features of real samples. Then, InfoGAN maximizes the amount of mutual information between c and a generated sample $G(z, c)$ to allow c to capture some noticeable features of real data. In other words, the generator takes the concatenated input (z, c) and maximizes the mutual information, $I(c; G(z, c))$ between a given latent code c and the generated samples $G(z, c)$ to learn meaningful feature representations. However, evaluating mutual information $I(c; G(z, c))$ needs to directly estimate the posterior probability $p(c|x)$, which is intractable. InfoGAN, thus, takes a variational approach which replaces a target value $I(c; G(z, c))$ by maximizing a lower bound.

Both CGAN and InfoGAN learn conditional probability $p(x|c)$ given a certain condition vector c; however, they are dissimilar regarding how they handle condition vector c. In CGAN, additional information c is assumed to be semantically known (such as class labels), so we have to provide c to the generator and the discriminator during the training phase. On the other hand, c is assumed to be unknown in InfoGAN, so we take c by sampling from prior distribution $p(c)$ and control the generating process based on $I(c; G(z, c))$. As a result, the automatically inferred c in InfoGAN has much more freedom to capture certain features of real data than c in CGAN, which is restricted to known information.

Semi-supervised InfoGAN (ss-InfoGAN) [121] takes advantage of both supervised and unsupervised methods. It introduces some label information in a semi-supervised manner by decomposing latent code c in to two parts, $c = c_{ss} \dot{\bigcup} c_{us}$. Similar to InfoGAN, ss-InfoGAN attempts to learn the

semantic representations from the unlabeled data by maximizing the mutual information between the generated data and the unsupervised latent code c_{us}. In addition, the semi-supervised latent code c_{ss} is trained to contain features that we want by using labeled data. For InfoGAN, we cannot predict which feature will be learned from the training, while ss-InfoGAN uses labeled data to control the learned feature by maximizing two mutual informations; one between c_{ss} and the labeled real data to guide c_{ss} to encode label information y, and the other between the generated data and c_{ss}. By combining the supervised and unsupervised methods, ss-InfoGAN learns the latent code representation more easily with a small subset of labeled data than the fully unsupervised methods of InfoGAN. Figure 11 shows the flow of InfoGAN and ss-InfoGAN where Q in the blue rectangle represents an auxiliary parameterized network which approximates posterior probability $p(c|x)$, and I in the red rectangle represents the mutual information. It should be noted that the two mutual information terms for c_{ss} are combined into I_{ss} in Figure 11b. The reason why Q is rooted from the discriminator is that the discriminator network is commonly shared with auxiliary network Q in practical implementation.

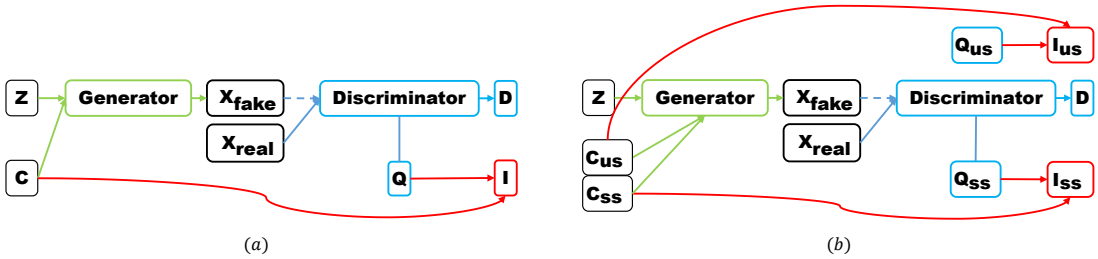


Figure 11: Illustrations of (a) InfoGAN [16] and (b) ss-InfoGAN [121].

3.1.3 Examples

Decomposing the latent space into meaningful attributes within the GAN framework has been exploited for various tasks. In this section, we introduce some noticeable GAN examples for image generation under the given condition.

StackGAN [147] was proposed for text to image generation that synthesizes corresponding images given text descriptions as shown in Figure 12. StackGAN synthesizes images conditioned on text descriptions in a two-stage process: a low-level feature generation (stage 1), and painting details from a given generated image at stage 1 (stage 2). The generators of each stage are trained adversarially given text embedding information φ_t from the text t. Notably, rather than directly concatenating φ_t to z as CGAN does, StackGAN proposes a conditional augmentation technique which samples conditioned text latent vectors c from the Gaussian distribution $\mathcal{N}(\mu(\varphi_t), \sum(\varphi_t))$. By sampling the text embedding vector c from $\mathcal{N}(\mu(\varphi_t), \sum(\varphi_t))$, this technique attempts to augment more training pairs given limited amount of image-text paired data.

Another algorithm for text to image generation, Text Auxiliary Classifier GAN (TAC-GAN) [19] uses fixed text embedding for conditioning the text description. TAC-GAN extends AC-GAN to a text-to-image generation. The main difference is that TAC-GAN concatenates the latent variable z with the text embedding, not a class label as AC-GAN does. Similar to AC-GAN, a one-hot encoded class label is still utilized for the classification task of the discriminator to encourage the generated images to reflect the text description correctly.

Semantically decomposing GAN (SD-GAN) [24] tries to generate a face having different poses by directly decomposing z into identity and pose parts of a face image and then sampling each latent variable separately from the independent latent distributions. Meanwhile, we do not need to restrict the attributes of an image to facial characteristics. Attributes can be not only characteristics of a face but also scenery features such as the weather. Karacan et al. [58] synthesized outdoor images having specific scenery attributes using the CGAN framework, and they also concatenated an attribute latent vector to z for the generator.

Figure 12: A text to image synthesis of StackGAN [147]. StackGAN shows higher output diversity than other text to image models. Images from [37].

3.2 With an Autoencoder

In this section, we explore efforts combining an autoencoder structure into the GAN framework. An autoencoder structure consists of two parts: an encoder which compresses data x into latent variable z: and a decoder, which reconstructs encoded data into the original data x. This structure is suitable for stabilizing GAN because it learns the posterior distribution $p(z|x)$ to reconstruct data x, which reduces mode collapse caused by the lack of GAN's inference ability to map data x to z. An autoencoder can also help manipulations at the abstract level become possible by learning a latent representation of a complex, high-dimensional data space with an encoder $\mathcal{X} \to \mathcal{Z}$ where $\mathcal{X}$ and $\mathcal{Z}$ denote the data space and the latent space. Learning a latent representation may make it easier to perform complex modifications in the data space through interpolation or conditional concatenation in the latent space. We demonstrate how GAN variants learn in the latent space in Section 3.2.1 and extend our discussion to proposed ideas which combine a VAE framework, another generative model with an autoencoder, with GAN in Section 3.2.2.

3.2.1 Learning the Latent Space

Adversarially learned inference (ALI) [27] and bidirectional GAN (BiGAN) [25] learn latent representations within the GAN framework combined with an encoder. As seen in Figure 13a, they learn the joint probability distribution of data x and latent z while GAN learns only the data distribution directly. The discriminator receives samples from the joint space of the data x and the latent variable z and discriminates joint pairs $(G(z), z)$ and $(x, E(x))$ where G and E represent a decoder and an encoder, respectively. By training an encoder and a decoder together, they can learn an inference $\mathcal{X} \to \mathcal{Z}$ while still being able to generate sharp, high-quality samples.

Ulyanov et al. [128] proposed a slightly peculiar method in which adversarial learning is run between the generator G and the encoder E instead of the discriminator. They showed that adversarial learning in the latent space using the generator and the encoder theoretically results in the perfect generator. Upon their theorems, the generator minimizes the divergence between $E(G(z))$ and a prior z in the latent space while the encoder maximizes the divergence. By doing so, they implemented adversarial learning together with tractable inference and disentangled latent space without additional computational cost.

In addition, they added reconstruction loss terms in each space to guarantee each component to be reciprocal. These loss terms can be interpreted as imposing each function to be a one to one mapping function so as not to fall into mode collapse as similar to in MRGAN. Recall that a geometric regularizing term of MRGAN [14] also aims to incorporate a supervised training signal which guides the reconstruction process to the correct location. Figure 13b shows an outline of Ulyanov et al. [128] where R in the red rectangle denotes the reconstruction loss term between the original and reconstructed samples.

3.2.2 Variational Autoencoder

Variational Autoencoder (VAE) [23] is a popular generative model using an autoencoder framework. Assuming some unobserved latent variable z affects a real sample x in an unknown manner,

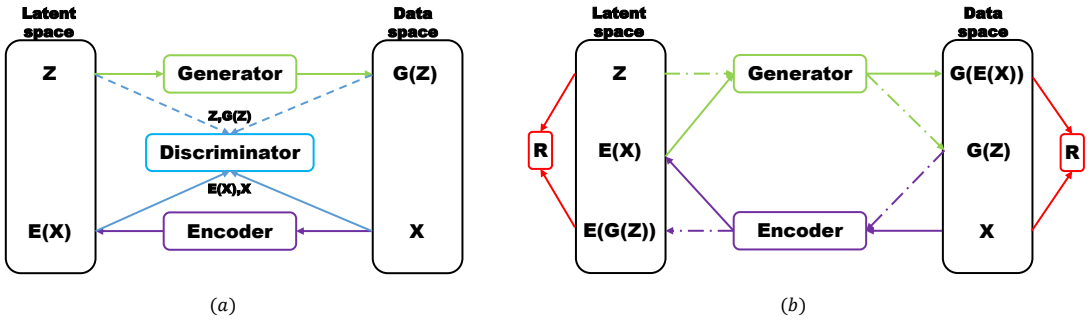


Figure 13: Illustrations of (a) ALI [27], BiGAN [25], and (b) AGE [128].

VAE essentially finds the maximum of the marginal likelihood $p_\theta(x)$ for the model parameter θ. VAE addresses the intractability of $p_\theta(x)$ by introducing a variational lower bound, learning the mapping of $\mathcal{X} \to \mathcal{Z}$ with an encoder and $\mathcal{Z} \to \mathcal{X}$ with a decoder. Specifically, VAE assumes a prior knowledge $p(z)$ and approximated posterior probability modeled by $Q_\phi(z|x)$ to be a standard normal distribution and a normal distribution with diagonal covariance, respectively for the tractability. More explicitly, VAE learns to maximize $p_\theta(x)$ where a variational lower bound of the marginal log-likelihood $\log p_\theta(x)$ can be derived as follows:

$$\log p_\theta(x) = \int_z Q_\phi(z|x) \log p_\theta(x) dz = \int_z Q_\phi(z|x) \log \left(\frac{p_\theta(x,z)}{p_\theta(z|x)} \frac{Q_\phi(z|x)}{Q_\phi(z|x)} \right) dz \tag{36}$$

$$= \int_z Q_\phi(z|x) \left(\log \left(\frac{Q_\phi(z|x)}{p_\theta(z|x)} \right) + \log \left(\frac{p_\theta(x,z)}{Q_\phi(z|x)} \right) \right) dz \tag{37}$$

$$= KL(Q_\phi(z|x)||p_\theta(z|x)) + \mathbb{E}_{Q_\phi(z|x)} \left[\log \frac{p_\theta(x,z)}{Q_\phi(z|x)} \right] \tag{38}$$

As $KL(Q_\phi(z|x)||p_\theta(z|X))$ is always nonnegative, a variational lower bound $L(\theta, \phi; x)$ of Equation 42 can be derived as follows:

$$\log p_\theta(x) \geq \mathbb{E}_{Q_\phi(z|x)}[\log p_\theta(x,z) - \log Q_\phi(z|x)] \tag{39}$$

$$= \mathbb{E}_{Q_\phi(z|x)}[\log p_\theta(z) - \log p_\theta(x|z) - \log Q_\phi(z|x)] \tag{40}$$

$$= -KL(Q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{Q_\phi(z|x)}[\log p_\theta(x|z)] \tag{41}$$

$$= L(\theta, \phi; x) \tag{42}$$

where $p_\theta(x|z)$ is a decoder that generates sample x given the latent z and $Q_\phi(z|x)$ is an encoder that generates the latent code z given sample x.

Maximizing $L(\theta, \phi; x)$ increases the marginal likelihood $p_\theta(x)$. The first term can be interpreted as leading an encoder $Q_\phi(z|x)$ to be close to a prior probability $p_\theta(z)$. It can be calculated analytically because $Q_\phi(z|x)$ and the prior probability are assumed to follow a Gaussian distribution. The second term can be estimated from the sample using a reparameterization method. To sum up, VAE learns by tuning the parameter of the encoder and the decoder to maximize the lower bound $L(\theta, \phi; x)$.

3.2.2.1 Hybrid with GAN

Recently, several approaches to incorporate each advantage of VAE and GAN have been proposed. Although VAE generates blurry images, VAE suffers less from the mode collapse problem because an autoencoder encourages all real samples to be reconstructed. However, GAN generates sharper images than VAE and does not need further constraints on the model while GAN suffers from mode collapse as mentioned in Section 2.4. In this section, we address two studies which attempt to combine VAE and GAN into one framework.

VAEGAN [67] combined VAE with GAN by assigning GAN's generator to a decoder. Its objective function combined VAE's objective function with an adversarial loss term to produce

sharp images while maintaining encoding ability for the latent space. Notably, it replaced the reconstruction of x in Equation 42 with the intermediate features of the discriminator to capture more perceptual similarity of real samples. Figure 14a shows an outline of VAEGAN where the discriminator D takes one real sample and two fake samples where one is sampled from an encoded latent space (z_{VAE}) and the other from a prior distribution (z).

Variational approaches for autoencoding GAN (α-GAN) [107] proposes adopting discriminators for the variational inference and transforms Equation 41 into a more GAN-like formulation. The most negative aspect of VAE is that we have to constrain a distribution form of $Q_\phi(z|x)$ to analytically calculate $L(\theta, \phi; x)$. α-GAN treats the variational posteriori distribution implicitly by using the density ratio technique which can be derived as follows:

$$KL(Q_\phi(z|x)||p_\theta(z)) = \mathbb{E}_{Q_\phi(z|x)}\left[\frac{Q_\phi(z|x)}{p_\theta(z)}\right] \approx \mathbb{E}_{Q_\phi(z|x)}\left[\frac{C_\phi(z)}{1 - C_\phi(z)}\right] \tag{43}$$

where $C_\phi(z) = \frac{Q_\phi(z|x)}{Q_\phi(z|x)+p_\theta(z)}$ which is an optimal solution of the discriminator for a fixed generator in standard GAN [38]. α-GAN estimates the KLD term using a learned discriminator from the encoder $Q_\phi(z|x)$ and the prior distribution $p_\theta(z)$ in the latent space. A KLD regularization term of a variational distribution, thus, no longer needs to be calculated analytically.

α-GAN also modifies a reconstruction term in Equation 42 by adopting two techniques. One is using another discriminator which distinguishes real and synthetic samples, and the other adds a normal l_1 pixelwise reconstruction loss term in the data space. Specifically, α-GAN changes a variational lower bound $L(\theta, \phi; x)$ into a more GAN-like formulation by introducing two discriminators using a density ratio estimation and reconstruction loss to prevent a mode collapse problem. Figure 14b shows an outline of α-GAN where D_L is the discriminator of the latent space, and D_D represents the discriminator which acts on the data space and R is the reconstruction loss term.

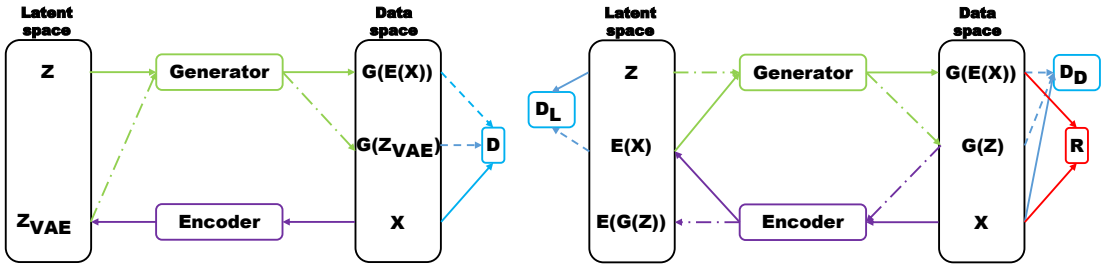


Figure 14: Illustrations of (a) VAEGAN [67] and (b) α-GAN [107].

3.2.3 Examples

Antipov et al. [3] proposed an encoder-decoder combined method for the face aging of a person. It produces facial aging of the target image under the given age vector y while maintaining the identity of the target image. The encoder takes a role to output a latent vector z which represents a personal identity to be preserved. However, practically, this CGAN-based approach has a problem in that the generator tends to ignore the latent variable z, concerning only conditional information y.

There are some approaches to training z in an unsupervised manner with an auto encoder. Mainly, semi latent GAN (SL-GAN) [143] proposed a method for changing a facial image from high-level semantic facial attributes (*i.e.*, male/female, skin/hair color), by decomposing the latent space from an encoder into annotated attributes (ground-truth attribute of an image) and data-driven attributes (as c of InfoGAN). Similarly to InfoGAN, SL-GAN tries to maximize the mutual information between the data-driven attribute and the generated image while using unsupervised training for the data-driven attributes.

In addition to that, disentangled representation GAN (DR-GAN) [126] addresses pose-invariant face recognition, which is a difficult problem because of the drastic changes of an image for each different pose, adopting an encoder-decoder structure for the generator. As the purpose of DR-GAN is to generate a face of the same identity given a target pose as seen in Figure 15, it has

to learn the identity feature to be invariant regardless of a facial pose. To achieve this, DR-GAN designs an encoder of the generator to represent an identity feature, while a decoder of the generator produces an image under the pose representing vectors and the encoded identity.

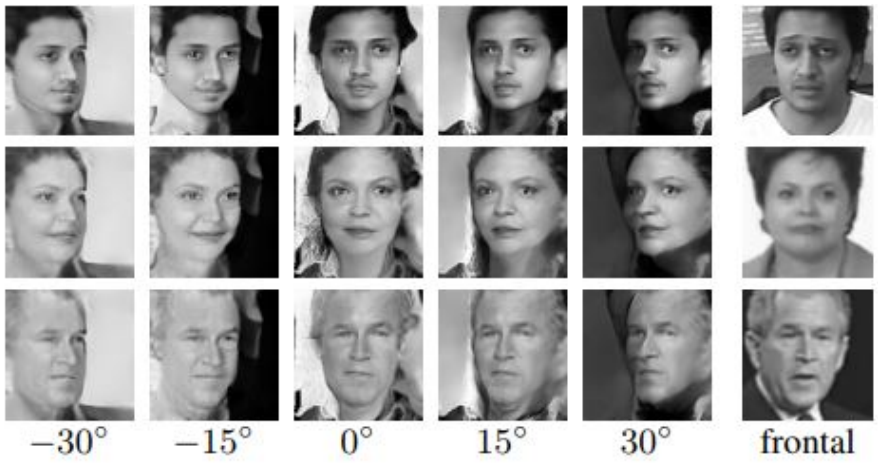


Figure 15: Multi-image face rotation on IJB-A dataset of DR-GAN [126]. DR-GAN shows facial rotation of an image maintaining persons' identities. Images from DR-GAN [126].

4 Applications Using GANs

As discussed in earlier sections, GAN is a very powerful generative model in that it can generate real-like samples with an arbitrary latent vector z. We do not need to know an explicit real data distribution nor assume further mathematical conditions. These advantages lead GAN to be applied in various academic and engineering fields. In this section, we discuss applications of GANs in several domains.

4.1 Image

4.1.1 Image translation

Image translation involves translating images in one domain X to images in another domain Y. Mainly, translated images have the dominant characteristic of domain Y maintaining their attributes in the original images. Image translation can be categorized into supervised and unsupervised techniques such as the classical machine learning.

4.1.1.1 Paired two domain data

Image translation with paired images can be regarded as supervised image translation in that an input image $x \in X$ to be translated always has the target image $y \in Y$ where X and Y are two distinctive domains. Pix2pix [55] suggests an image translation method with paired images using the U-NET architecture [106] which is widely used for biomedical image segmentation. It adopts a CGAN framework in which a generator produces a corresponding target image conditioned on an input image as seen in Figure 16. In contrast, Perceptual Adversarial Networks (PAN) [132] add the perceptual loss between a paired data (x, y) to the generative adversarial loss to transform input image x into ground-truth image y. Instead of using the pixelwise loss to push the generated image toward the target image, it uses hidden layer discrepancies of the discriminator between an input image x and ground truth image y. It tries to transform x to y to be perceptually similar by minimizing perceptual information discrepancies from the discriminator.

4.1.1.2 Unpaired two domain data

Image translation in an unsupervised manner learns a mapping between two domains given unpaired data from two domains. CycleGAN [150] and discover cross-domain relations with GAN

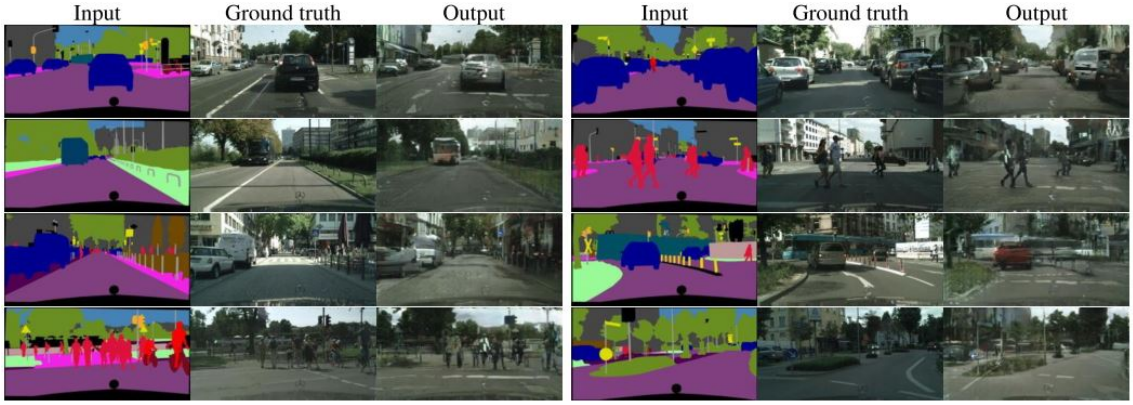


Figure 16: Paired image translation results proposed by pix2pix [55]: converting Cityscapes labels into real photo compared to ground truth. Images from pix2pix [55].

(DiscoGAN) [60] aim to conduct unpaired image-to-image translations using a cyclic consistent loss term in addition to an adversarial loss term. With a sole translator G: $X \rightarrow Y$, GAN may learn mappings from $X \rightarrow Y$ with meaningless translation or mode collapse (*i.e.*, several images in X are mapped to one image in Y.), resulting in an undesired translation. To reduce the space of mapping of the generator, they adopt another inverse translator T: $Y \rightarrow X$ and introduce the cyclic consistency loss which encourages $T(G(x)) \approx x$ and $G(T(y)) \approx y$. The cyclic loss term with an L1 norm leads to reconstruction from $T(G(x))$ to an original x and from $G(T(y))$ to an original y so that each translation finds a plausible mapping between the two domains as mentioned in Section 2.1.3.1. It can be thought of as an autoencoder scheme with a reconstruction loss term to impose meaningful translation to the desired output. In addition, their methods can also be interpreted in a similar manner described in Section 3.2.1 in that they add a supervised signal for reconstruction. In addition, DualGAN [142] also adopts cyclic reconstruction loss but uses the Wasserstein distance instead of a sigmoid cross entropy generative adversarial loss. Figure 17 shows the overflow of the unpaired image translation of CycleGAN [150] and DiscoGAN [60] where D denotes the discriminator in each domain and R is the reconstruction loss for the cyclic consistency.

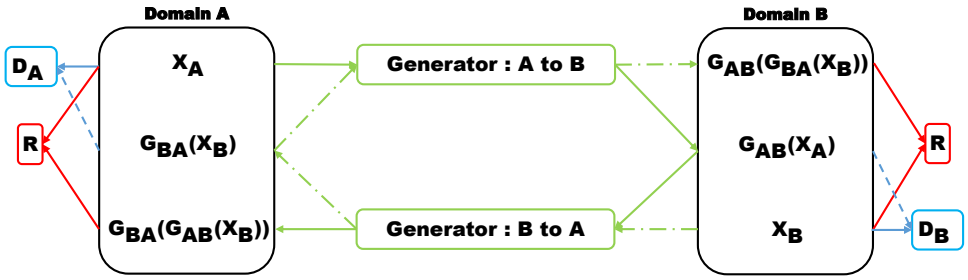


Figure 17: An illustration of unpaired image translation of CycleGAN [150] and DiscoGAN [60].

In addition to the cyclic consistency constraint, Benaim and Wolf [10] conducted unsupervised domain mapping by adding distance preserving loss to reduce further the possible mapping space of the generator. Their main idea was that the distance between two samples in X should be maintained in Y. Not only the distance preserving for each domain, but they also proposed self-distance loss which maintains the pairwise distance between two left and right halves of one sample image to perform one-sided domain mapping.

Attribute guided image translation was also considered to transfer the visual characteristic of an image. Conditional CycleGAN [78] embeds the target attribute vector z and concatenates it with an image to be translated, utilizing CGAN with a cyclic consistency framework. Kim et al. [61] attempted to transfer visual attributes. In addition to the cyclic consistency of an image, they also added an attribute consistency loss which forces the transferred image to have a target attribute of the reference image.

4.1.2 Super resolution

Acquiring super resolution images from low resolution images has the fundamental problem that the recovered high-resolution image misses high-level texture details during the upscaling of the image. Ledig et al. [68] adopted a perceptual similarity loss in addition to an adversarial loss, instead of pixelwise mean-squared error loss. It focuses on feature differences from the intermediate layer of the discriminator, not pixelwise because optimizing pixel-wise mean squared error induces the pixelwise average of a plausible solution, leading to perceptually poor smoothed details and it is not robust to drastic pixel value changes.

4.1.3 Object detection

Detecting small objects in an image typically suffers from low-resolution of an object, and thus, it is necessary to train models with images of various scales similar to You Look Only Once (YOLO) [105] and Single Shot Detection (SSD) [100] methods. Notably, Li et al. [72] tries to transform a small object with low resolution into a super resolved large object to make the object more discriminative. They utilized a GAN framework except decomposed the discriminator into two branches, namely, an adversarial branch and a perceptual branch. The generator produces a real-like large-scale object by the typical adversarial branch while the perceptual branch guarantees that the generated large-scale object is useful for the detection.

Ehsani et al. [29] proposed another framework to detect objects occluded by other objects in an image. It uses a segmentor, a generator, and a discriminator to extract the entire occluded-object mask and to paint it as a real-like image. The segmentor takes an image and a visible region mask of an occluded object and produces a mask of the entire occluded object. The generator and the discriminator are trained adversarially to produce an object image in which the invisible regions of the object are reconstructed.

4.1.4 Object transfiguration

Object transfiguration is a conditional image generation that replaces an object in an image with a particular condition while the background does not change. Zhou et al. [149] adopted an encoder-decoder structure to transplant an object, where the encoder decomposes an image into the background feature and the object feature, and the decoder reconstructs the image from the background feature and the object feature we want to transfigure. Importantly, to disentangle the encoded feature space, two separated training sets are required where one is the set of images having the object and the other is the set of images not having the object.

In addition, the GAN can be applied to an image blending task which implants an object into another image's background and makes the composited copy-paste images look more realistic. Gaussian-Poisson GAN (GP-GAN) [137] suggests a high-resolution image blending framework using GAN and a classic image blending gradient-based approach [32]. It decomposes images into low-resolution but well-blended images using a GAN and detailed textures and edges using a gradient constraint. Then, GP-GAN attempts to combine the information by optimizing a Gaussian-Poisson equation [99] to generate high-resolution well-blended images while maintaining captured high-resolution details.

4.1.5 3D image generation

Advances in 3-dimensional (3D) volumetric convolution networks have led to the application of GAN to generate 3D objects. Wu et al. [138] proposed a 3D GAN framework inspired by DCGAN [104] that expands the system by one more dimension to address 3-dimensional data. Gadelha et al. [33] inferred 3D objects from multiple 2D views of an object. It uses a 3D volumetric convolution network to generate 3D shapes in the generator, but its discriminator takes a 2D projected image as an input so that the projection of a synthesized 3D object matches the input 2D view.

4.1.6 Joint image generation

The GAN can be utilized to generate multiple domain images at once. Coupled GAN [77] suggests a method of generating multidomain images jointly by weight sharing techniques among GAN pairs. It first adopts GAN pairs to match the number of domains we want to produce. Then, it shares the weights of some layers of each GAN pair that represents high-level semantics.

Therefore, it attempts to learn joint distributions of a multidomain from samples drawn from a marginal domain distribution. It should be noted that because it aims to generate multidomain images which share high-level abstract representations, images from each domain have to be very similar in a broad view.

4.1.7 Video generation

In this paragraph, we discuss GANs generating video. Generally, the video is composed of relatively stationary background scenery and dynamic object motions. Video GAN (VGAN) [130] considers a two-stream generator. A moving foreground generator using 3D CNN predicts plausible future frames while a static background generator using 2D CNN makes the background stationary. Pose-GAN [131] takes a VAE and GAN combining approach. It uses a VAE approach to estimate future object movements conditioned on a current object pose and hidden representations of past poses. With a rendered future pose video and clip image, it uses a GAN framework to generate future frames using a 3D CNN. Recently, motion and content GAN (MoCoGAN) [127] proposed to decompose the content part and motion part of the latent space, especially modeling the motion part with RNN to capture the time dependency. Table 5 summarizes each video generating GAN.

Table 5: Comparison of video generators. Table reproduced from MocoGAN [127].

	VGAN	Pose-GAN	MoCoGAN
Video generation	Directly generating 3D volume	Frames with rendered future pose video	Frames are generated sequentially
Generator	3D CNN/2D CNN	3D CNN	2D CNN
Discriminator	3D CNN	3D CNN	3D CNN/2D CNN
Variable length	No	No	Yes
Details	2 stream generator	VAE+GAN	Decompose motion and content

4.2 Sequential Data Generation

GAN variants that generate discrete values mostly borrow a policy gradient algorithm of RL, to circumvent direct back-propagation of discrete values. To output discrete values, the generator, as a function, needs to map the latent variable into the domain where elements are not continuous. However, if we do the back-propagation as another continuous value generating process, the generator is steadily guided to generate real-like data by the discriminator, rather than suddenly jumping to the target discrete values. Thus, such a slight change of the generator cannot easily look for a limited real discrete data domain [146].

In addition, when generating a sequence such as music or language, we need to evaluate a partially generated sequence step-by-step, measuring the performance of the generator. However, the conventional GAN framework can only evaluate whole generated sequences unless there is a discriminator for each time-step. This, too, can be solved by the policy gradient algorithm, in that RL naturally addresses the sequential decision process of the agent.

4.2.1 Music

When we want to generate music, we need to generate the note and a tone of the music step-by-step, and these elements are not continuous values. A simple and direct approach is of continuous RNN-GAN (C-RNN-GAN) [86], where it models both the generator and discriminator as an RNN with long-short term memory (LSTM) [49] which is widely used for processing sequential data, directly extracting whole sequences of music. However, as mentioned above, we can only evaluate whole sequences, and not a partially generated sequence. Furthermore, its results are not highly satisfactory since it does not consider the discrete property of the music elements.

In contrast, sequence GAN (SeqGAN) [146], object reinforced GAN (ORGAN) [43], and Lee et al. [69] employed a policy gradient algorithm, and not generating whole sequences at once. The result of SeqGAN is shown in Figure 18. The reason why they adopted a policy gradient is that a standard GAN approach has poor ability to generate discrete output because the discriminator's

gradient flow to the generator incurs slight changes of the generator's output, rather than output proper to the discretized data space. Moreover, partially generated sequences cannot be evaluated, unlike an entire generated sequence in C-RNN-GAN. They treat a generator's output as a policy of an agent and take the discriminator's output as a reward. Selecting a reward with the discriminator is a natural choice as the generator acts to obtain a large output (reward) from the discriminator, similar to the agent learning to acquire a large reward in reinforcement learning. In addition, ORGAN is slightly different from SeqGAN, adding a hard-coded objective to the reward function to achieve the specified goal.

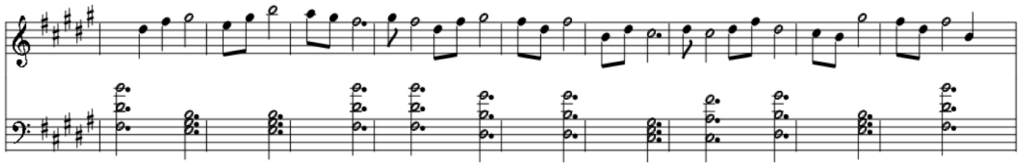


Figure 18: Polyphonic music sequences generated from SeqGAN [69].

4.2.2 Language and speech

RankGAN [76] suggests language (sentence) generation methods and a ranker instead of a conventional discriminator. The reason for using a ranker concept is that a conventional discriminator only does binary classification, whether its input is real or fake. In natural language processing, the expression power of natural language needs to be considered in addition to its authenticity. Thus, RankGAN adopts a relative ranking concept between generated sentences and reference sentences which are human-written. The generator tries its generated language sample to be ranked high, while the ranker evaluates the rank score of the human-written sentences higher than the machine-written sentences. As the generator outputs discrete symbols, it similarly adopts a policy gradient algorithm similar to SeqGAN and ORGAN. In RankGAN, the generator can be interpreted as a policy predicting next step symbol and the rank score can be thought of as a value function given a past generated sequence.

Variational autoencoding Wasserstein GAN (VAW-GAN) [51] is a voice conversion system combining GAN and VAE frameworks. The encoder infers a phonetic content z of the source voice, and the decoder synthesizes the converted target voice given a target speaker's information y, similar to conditional VAE [140]. As discussed in Section 3.2.2, VAE suffers from generating sharp results due to the oversimplified assumption of the Gaussian distribution. To address this issue, VAW-GAN incorporates WGAN [5] similarly to VAEGAN [67]. By assigning the decoder to the generator, it aims to reconstruct the target voice given the speaker representation.

4.3 Semi-Supervised Learning

Semi-supervised learning is a learning method that improves the classification performance by using unlabeled data in a situation where there are both labeled and unlabeled data. In the big data era, there exists a common situation where the size of the data is too large to label all the data, or the cost of labeling is expensive. Thus, it is often necessary to train the model with a dataset in which only a small portion of the total data has labels.

Variations of the discriminator allow the GAN framework to be used for classification. In addition, semi-supervised learning can be enabled using unlabeled data for classifier training. Several papers [22, 120] show that adversarial training regularizes learning of the classifier, thereby creating a more robust classifier. The generator can also help in semi-supervised classification by learning the conditional data distribution and acting as an over-sampler for missing labels.

4.3.1 Semi-supervised learning with a discriminator

The GAN-based semi-supervised learning method [109] demonstrates how unlabeled and generated data are available on the GAN framework. The generated data is allocated to a $K+1$ class beyond $1 \dots K$ class for the labeled data. The model learns to maximize the log probability of the output unit for the labeled real data corresponding to its correct label and to minimize the probability of the unlabeled data being allocated to the $K+1$ class which is given for the generated

data. For the generated data, the model learns to maximize the log-likelihood of the classifying of $y = K + 1$. The unlabeled data and the generated data serve to inform the model of the space where the real data resides. In other words, the unsupervised cost serves to guide the location of the optimum of the supervised cost of the real labeled data.

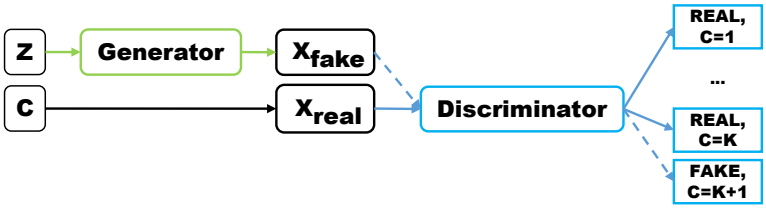


Figure 19: An illustration of GAN-based semi-supervised learning with a discriminator.

Figure 19 shows a flowchart of GAN-based semi-supervised learning with the discriminator. For labeled real data, the discriminator classifies their correct label (1 to K). For unlabeled real data and generated data, they are trained with a GAN minimax game. Their training objective can be expressed as follows:

$$L = L_s + L_{us} \tag{44}$$

$$L_s = -\mathbb{E}_{x,y\sim p_{\text{data}}(x,y)}[\log p_\theta(y|x, y < K + 1)] \tag{45}$$

$$L_{us} = -\mathbb{E}_{x\sim p_{\text{data}}(x)}[1 - \log p_\theta(y = K + 1|x)] + \mathbb{E}_{x\sim G}[\log p_\theta(y = K + 1|x)] \tag{46}$$

where L_s and L_{us} stand for the loss functions of the labeled data and the unlabeled data, respectively. It is noted that because only generated data is classified as the $K + 1$ class, we could think of L_{us} as a GAN standard minimax game.

Categorical GAN (CatGAN) [120] proposes an algorithm for the robust classification for which the generator regularizes the classifier. The discriminator has no classification head for distinguishing real and fake and is trained with three requirements: a small conditional entropy of $H(y|x)$ to make the correct class assignment for the real data, a large conditional entropy of $H(y|G(z))$ to make the class assignment for the generated data diverse and a large entropy of $H(y)$ to make a uniform marginal distribution with an assumption of a uniform prior $p(y)$ over classes where x, y and $G(z)$ are real data, labels, and generated data respectively. The generator, meanwhile, is trained with two requirements: a small conditional entropy of $H(y|G(z))$ to make the class assignment for the generated data certain and a large entropy of $H(y)$ to generate equally distributed samples over classes. The unlabeled data and the generated data help the classification by balancing classes through the adversarial act of the generator, so it helps semi-supervised learning.

Semi-supervised learning with context-conditional GAN (CC-GAN) [22] suggests an algorithm for both image in-painting and semi-supervised learning. CC-GAN's generator fills a missing image patch conditioned on the surrounding pixels where the discriminator distinguishes the full images as real data, and images with missing patches and filled images as fake data. The learned generator can be used for missing value imputation. Similar to the method used in CatGAN, semi-supervised learning may be enabled by adding a cross entropy term for the labeled data to the objective function. The discriminator has an output layer with an output neuron that distinguishes real and fake samples and another output neuron for classification. Hidden layers of the discriminator are shared between two heads and learn a hierarchical feature representation by discriminating real and fake samples, which helps semi-supervised classification.

4.3.2 Semi-supervised learning with an auxiliary classifier

The above GAN variants in semi-supervised learning have two problems: the first one is that the discriminator has two incompatible convergence points, one for discriminating real and fake data and the other for predicting the class label; and the second problem is that the generator cannot generate data in a specific class. Triple-GAN [70] addresses the two problems by a three-player formulation. Triple GAN consists of three agents: a generator G, a discriminator D and a classifier C. G samples the generated data X_g conditioned on the class label Y_g. C computes the

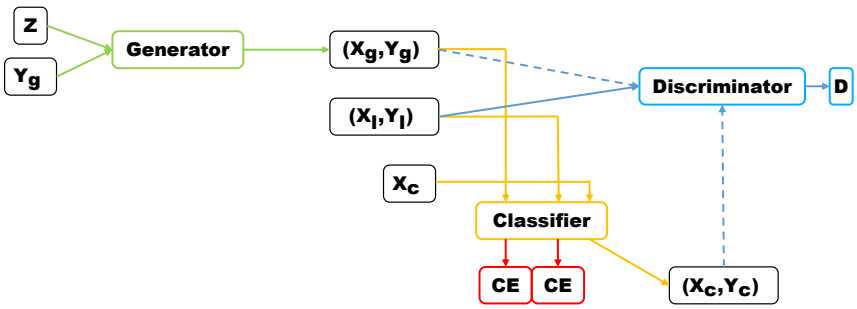


Figure 20: An illustration of Triple-GAN [70].

cross entropy of the generated data $(X_g, Y_g) \sim p_g(X, Y)$ and the labeled data $(X_l, Y_l) \sim p(X, Y)$ and predicts class Y_c of the unlabeled data X_c. D is simply trained to distinguish the labeled data (X_l, Y_l) as real, and the generated data under the class condition (X_g, Y_g) and the unlabeled data with a predicted label $(X_c, Y_c) \sim p_c(X, Y)$ as fake. In addition, X_g, Y_g and X_l, Y_l are trained with the cross entropy loss for the classifier. The model is illustrated in Figure 20 where CE refers to the cross entropy loss.

To summarize, Triple-GAN adopts an auxiliary classifier which classifies real labeled data and label-conditioned generated data, relieving the discriminator of classifying the labeled data. In addition, Triple-GAN generates data conditioned on Y_g, which means that it can generate label-specific data.

4.4 Domain Adaptation

Domain adaptation is a type of transfer learning which tries to adapt data from one domain (*i.e.*, the source domain) into another domain (*i.e.*, the target domain), while the classification task performance is preserved in the target domain [97]. The reason why domain adaptation has gained attention is that following situations frequently occur in the machine learning field: there is a source domain with labeled data and a target domain with unlabeled data where we want to solve some machine learning task. These situations widely occur in various fields because labeled data is difficult to obtain in the real world. Domain adaptation can be considered as transferring prior knowledge of the source domain to the target domain for performing some task (*e.g.* , classification) in the target domain. Formally, the unsupervised domain adaptation solves the following problem: with input data x and its label y, let a source domain distribution $\mathbb{D}_S(x, y)$ and a target domain distribution $\mathbb{D}_T(x, y)$ be defined over $X \times Y$ where X and Y are sets of a data space and a label space respectively. Given labeled source domain data $(x_s, y_s) \in \mathbb{D}_S(x, y)$ and unlabeled target domain data $x_t \in \mathbb{D}_T(x)$ (the marginal distribution of $\mathbb{D}_T(x, y)$), the unsupervised domain adaptation aims to learn a function $h : X \to Y$ which classifies well in the target domain without the label information of target domain data x_t.

4.4.1 Domain adaptation by feature space alignment via GAN

The main difficulty in domain adaptation is the difference between the source distribution and the target distribution, called domain shift. This domain shift allows the classifier trained with only the source data to fail in the target domain. One of the methods to address the domain shift is to project each domain data into the common feature space where the distributions of the projected data are similar. There have been a few studies to achieve the common feature space via GAN for the domain adaptation task.

Domain adversarial neural network (DANN) [2] first used GAN to obtain domain invariant features by making them indistinguishable as to whether it comes from the source domain or the target domain while still discriminative for the classifying task. There are two components sharing a feature extractor. One is a classifier which classifies the labels of the data. The other is a domain discriminator which discern where the data comes from. DANN trains a label classifier to classify the labels of the source data via a cross entropy loss and trains a domain discriminator adversarially, which leads the feature generator to extract domain invariant features. The feature

generator, thus, acts like the generator in GAN producing source-like features from the target domain, and the domain discriminator acts like the discriminator in that it discriminates the source domain features as real and the target domain features as fake with a binary cross entropy loss. To sum up, DANN takes CNN classification networks in addition to the GAN framework to classify data from the target domain without label information by learning the domain invariant features. Figure 21 shows the overall outline of DANN, ARDA and unsupervised pixel-level domain adaptation where I_S, I_T, and I_f stands for the source domain image, target domain image, and fake image, respectively. F_S and F_T means the extracted source features, and the target features, respectively. It should be noted that Y_s, which is the label information of I_s, is fed into the classifier, training the classifier with the cross entropy loss.

Most of the other methods for domain adaptation using GAN are based on DANN. Wasserstein distance guided representation learning (WDGRL) [113] also solves the domain adaptation problem with GAN, incorporating WGAN [5]. WDGRL has a very similar architecture with DANN consisting of three parts: the feature generator generating domain invariant features, the critic estimating Wasserstein distance between the generated features from the source and the target domains, and the classifier trained with the source labeled data. In contrast to DANN, it solves the minimax problem with the Wasserstein critic to train the domain discriminator.

Although DANN may achieve domain invariant marginal feature distributions, if the label of the source feature and that of the target feature do not contain a mismatch, the learned classifier should not work well in the target domain. Cycle-consistent adversarial domain adaptation (CyCADA) [50], thus, adds a cycle consistency [60, 150] to preserve the content of the data which is the crucial characteristic in determining the label while inheriting the most of the architecture of DANN. However, Shu et al. [116] used the cluster assumption that decision boundaries should not cross high-density regions [116] to improve the classification performance in the target domain. They implemented the cluster assumption by minimizing conditional entropy.

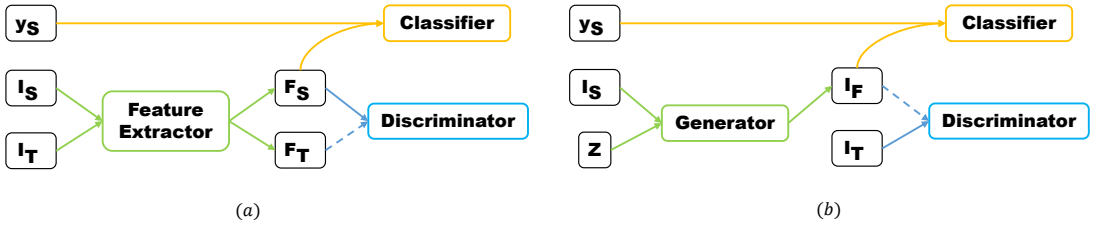


Figure 21: Illustrations of (a) DANN [2] and (b) unsupervised pixel-level domain adaptation [13].

4.4.2 Examples

Bousmalis et al. [13] used the domain adaptation via GAN for the grasping task. They used simulation data as the source domain data and ran the learned model in the real environment. Their method is slightly different than DANN in that it only adapts source images to be seen as if they were drawn from the target domain while DANN tries to obtain features similarly from both domains. It can be understood that the feature space of [13] is the target domain space and the feature generator in the target domain is the identity function. In addition, they added a content similarity loss defined as the pixelwise mean-squared error between the source image and the adapted image to preserve the content of the source image. By doing so, they achieved better performance than the supervised method in the grasping task. It should also be noted that the method of [13] can check whether the domain adaptation process is working well during the training phase because the transformed data is visible, while [2, 113] cannot visually check the domain adaptation process because the common representation space cannot be easily visualized.

Yoo et al. [145] achieved autonomous navigation without any real labeled data using domain adaptation. As in [13], they also exploited the simulation data as the source data and showed autonomous navigation in a real outdoor environment. They used cycle consistency to preserve the content and style loss term motivated by the style transfer task [34] to reduce the domain shift dramatically for the outdoor environment as seen in Figure 22. By doing so, they showed the applicability of the simulation via domain adaptation into the autonomous navigation tasks where collecting labels for various environments is difficult and expensive.

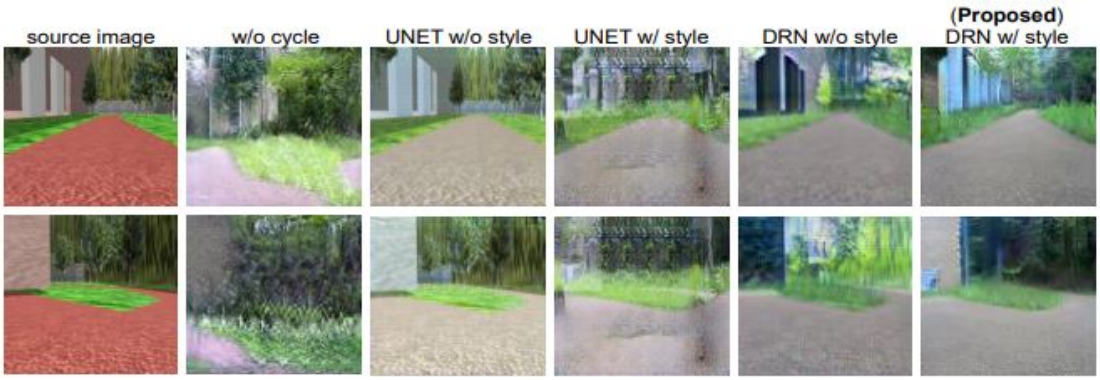


Figure 22: The source images and transferred images by five models proposed by Yoo et al. [145]. The leftmost images are source images made by a simulator, and figures of 2-6 columns are transferred images by five different models. They tried to transfer the source images to be like images of real environment for successful autonomous navigation. Images from Yoo et al. [145].

4.5 Other Tasks

Several variants of GAN have also been developed in other academic or practical fields other than the machine learning fields.

4.5.1 Medical Image Segmentation

Xue et al. [139] proposed a segmentor-critic structure to segment a medical image. A segmentor generates a predicted segmented image, and a critic maximizes the hierarchical feature differences between the ground-truth and the generated segmentation. This structure leads a segmentor to learn the features of the ground-truth segmentation adversarially similar to the GAN approach. There are also other medical image segmentation algorithms such as the deep image-to-image network (DI2IN) [141] and structure correcting adversarial network (SCAN) [17]. DI2IN conducts liver segmentation of 3D CT images through adversarial learning. SCAN tries to segment the lung and the heart from chest X-ray images through an adversarial approach with a ground-truth segmentation mask.

4.5.2 Steganography

It is also feasible to use GAN for steganography. Steganography is a technique that conceals secret messages in non-secret containers such as an image. A steganalyzer determines if a container contains a secret message or not. Some studies such as Volkhonskiy et al. [129], and Shi et al. [114] propose a steganography model with three components: a generator producing real-looking images that are used as containers and two discriminators, one of which classifies whether an image is real or fake, and the other determines whether an image contains a secret message.

4.5.3 Continual Learning

Deep generative replay [115] extends a GAN framework to continual learning. Continual learning solves multiple tasks and accumulates new knowledge continually. Continual learning in deep neural networks suffers from catastrophic forgetting which refers to forgetting a previously learned task while learning a new task. On the other hand, the mammalian brain has a mechanism called hippocampal replay that overcomes catastrophic forgetting by transferring information from short-term memory to long-term memory [94]. Inspired by the brain mechanism, catastrophic forgetting is addressed with a GAN framework called deep generative replay. Deep generative replay trains a scholar model in each task where a scholar model is composed of a generator. The generator produces samples of an old task, and the solver gives a target answer to an old task's sample. By sequentially training scholar models with old task values generated by old scholars, it attempts to overcome catastrophic forgetting while learning new tasks.

5 Discussion

We have discussed how GAN and its variants work and how they are applied to various applications. As we have viewed GAN in a microscopic perspective until now, we are going to discuss macroscopic view of GAN in this section.

5.1 Evaluation

Measuring the performance of GAN is related to capture the diversity and quality of generated data. As explained in the Section 1, the generative models mostly models the likelihood and learns by maximizing it. Thus, it is natural to evaluate the generator using log likelihood. However, GAN generates real-like data without estimating likelihood directly so evaluation with the likelihood is not quite proper. The most widely accepted evaluation metric of GAN is an inception score. The inception score was proposed by Salimans et al. [109], and the inception score is defined as follows:

$$\exp(\mathbb{E}_{x\sim p_g(x)}[KL(p(y|x)||p(y))]) \tag{47}$$

As seen in Equation 47, it computes the average KLD between conditional label distribution $p(y|x)$ and the marginal distribution of the generated data's label $p(y)$ with a pre-trained classifier such as VGG [117] and Imagenet data[66]. Since the KLD term between $p(y|x)$ and $p(y)$ in the exponent function is equivalent to their mutual information $I(y;x) = H(y) - H(y|x)$ [8], the high entropy of $p(y|x)$ and low entropy of $p(y)$ leads to a high inception score. The entropy of $p(y|x)$ measures how the generated data are sharp and clear to be well-classified. The other term $H(y)$ represents whether the generated data are diverse with respect to the generated class. In this way, the inception score is believed to measure the diversity and visual quality of generated data.

However, the inception score has a limitation in that it is unable to resist against the mode collapse problem. Even though the trained generator produces only one plausible data for each class, $p(y|x)$ can have low entropy leading the high inception score. To address this issue, AC-GAN [93] adopts Multi-Scale Structural Similarity (MS-SSIM) [134], which evaluates perceptual image similarity, and thus find out a mode collapse more reliably. In addition, MRGAN [14] proposes a MODE score based on the inception score and Danihelka et al. [18] suggests an independent Wasserstein critic which is trained independently for the validation dataset to measure over-fitting and mode collapse. Moreover, various object functions such as MMD, Total Variance and Wasserstein distance can be used as the approximated distance between $p_g(x)$ and $p_{data}(x)$. Theoretically, all metrics should produce the same result under the assumption of full capacity of the model and an infinite number of training samples. However, Theis et al. [124] show that minimizing MMD, JSD or KLD results in different optimal points even for the mixture of Gaussian distributions as a real data distribution and the convergence in one type of distance does not guarantee the convergence for another type of distance due to the fact that their object functions do not share a unique global optima.

The way of measuring the performance of GAN is still a disputable subject. Since GAN is naturally unsupervised learning, we can not measure accuracy or error rate as like in the supervised learning approaches. Evaluation metrics or different distances discussed above paragraph still do not measures the performance of GAN exactly, and there are lots of cases that images not looking natural have high score [8]. Thus, there is a room to improve the evaluation for GAN to be better.

5.2 Discrete Structured Data

Unlike other generative models like VAE, GAN has an issue to deal with the discrete data such as text sequences or discretized images. Since discrete data is non-differentiable, gradient descent update via back-propagation cannot directly be applied for a discrete output. The content of this section may overlap with Section 4.2 but we will shortly cover this issue in the discussion because it is one of the troublesome issue in GAN.

To address this issue, some methods adopt a policy gradient algorithm [123] in reinforcement learning (RL) of which objective is to maximize total rewards. By rolling out whole sequences, this method circumvents direct back-propagation for a discrete output. Intuitively, the generator which generates a fake data maximizing the discriminator's output can be thought of as a policy agent in RL which is a probability distribution to take an action maximizing a reward in a given state.

Maximum-Likelihood Augmented Discrete GAN (MLADGAN) [15] and Boundary-Seeking GAN (BSGAN) [47] both take a policy gradient algorithm to generate discrete structured data. They both treat the generator as a policy which outputs a probability for a discrete output. Importantly, they estimate the true distribution as $\tilde{p}_{data}(x) = \frac{1}{Z} p_{\mathrm{g}}(x) \frac{D(x)}{1-D(x)}$ where Z is a normalization factor. $\tilde{p}_{data}(x)$ is motivated from $p_{\mathrm{data}}(x) = \frac{D^*(x)}{1-D^*(x)} p_{\mathrm{g}}(x)$ where D^* is the optimal discriminator, and MLADGAN and BSGAN interpret $\tilde{p}_{data}(x)$ as a reward for a discrete output x. Because larger $D(x)$ means greater $\tilde{p}_{data}(x)$, taking $\tilde{p}_{data}(x)$ value as a reward brings about the same consequence as like when $D(x)$ is taken as a reward. They showed some successful performance but they are known that learning process is highly instable due to MCMC sampling in the training phase.

Adversarially Regularized Autoencoders (ARAE) [62] addresses this issue by combining a discrete auto-encoder which encodes a discrete input into the continuous code c. ARAE also adopts WGAN [5] acting on the latent space where encoded code lies. To avoid direct access to the discrete structure, the generator produces fake code $\tilde{c}$ from the sampled latent z and the critic evaluates such fake code $\tilde{c}$ and real code c. By jointly training an auto-encoder with WGAN on the code space, it avoids back-propagation on discrete space while learning latent representations of discrete structured data.

5.3 Relationship to Reinforcement Learning

Reinforcement Learning (RL) is a type of learning theory which focuses on teaching an agent to choose the best action given a current state. A policy $\pi(a|s)$ which is a probability for choosing an action a at state s, is learned via an on-policy or off-policy algorithms. RL has a very similar concept to GAN in the aspect of a policy gradient where it is very important to estimate the value function correctly given state s and action a.

Policy gradient algorithm [123] presented unbiased estimation of a policy gradient under the correct action-state value $Q(s, a)$. Because $Q(s, a)$ is the discounted total reward from state s and action a, $Q(s, a)$ must be estimated via an estimator called a critic. A wide variety of policy gradients such as deep deterministic policy gradient (DDPG) [74], trust region policy optimization (TRPO) [112], and Q-prop [42] can be thought of as estimating $Q(s, a)$ which has low bias and low variance, to correctly estimate the policy gradient. In that regard, they are similar to a GAN framework in that GAN's discriminator estimates the distance between two distributions and the approximated distance needs to be highly unbiased so as to make $p_{\mathrm{g}}(x)$ closely approximate $p_{data}(x)$. Pfau and Vinyals [101] detail the connection between GAN and actor-critic methods.

Inverse reinforcement learning (IRL) [1] is similar reinforcement learning in that its objective is to find the optimal policy. However, in the IRL framework, the experts' demonstrations are provided instead of a reward. It finds the appropriate reward function that makes the given demonstration as optimal as possible and then produces the optimal policy for the found reward function. There are many variants in IRL. Maximal entropy IRL [151] is one which finds the policy distribution that satisfies the constraints so that feature expectations of the policy distribution and the given demonstration are the same. To solve such an ill-posed problem, maximal entropy IRL finds the policy distribution with the largest entropy according to the maximal entropy principle [56]. Intuitively, the maximal entropy IRL finds the policy distribution which maximizes the likelihood of a demonstration and its entropy. Its constraint and convexity induce the dual minimax problem. The dual variable can be seen as a reward. The minimax formulation and the fact that it finds the policy which has the largest likelihood of demonstrations gives it a deep connection with the GAN framework. The primal variable is a policy distribution in IRL whereas it can be considered as a data distribution from the generator in GAN. The dual variable is a reward/cost in IRL while it can be seen as the discriminator in GAN.

Finn et al. [31], Yoo et al. [144], and Ho and Ermon [48] showed a mathematical connection between IRL and GAN. Ho and Ermon [48] converted IRL to the original GAN by constraining the space of dual variables and Yoo et al. [144] showed the relationship between EBGAN [148] and IRL using approximate inference.

5.4 Pros and Cons of GAN

5.4.1 Pros

As briefly mentioned in Section 1, the major advantage of GAN is that GAN does not need to define the shape of the probability distribution of the generator model. Thus, GAN naturally avoids concerning tractable density forms which need to represent complex and high dimensional distribution. Compared to other models using explicitly defined probability density [23, 95, 96, 110], GAN has following advantages:

- GAN can parallelize sampling of generated data. In case of Pixelcnn [110], Pixelrnn [96] and Wavenet [95], their speed of generation is highly slow due to their auto-regressive fashion, wherein $p_{\mathrm{g}}(x)$ is decomposed into a product of conditional distributions given previously generated values:

$$p_{\mathrm{g}}(x) = \prod_{i=1}^{d} p_{\mathrm{g}}(x_i | x_1, \dots x_{i-1}) \tag{48}$$

  where x is d-dimensional vector. For example, in image generation, auto-regressive models generate an image pixel by pixel where the probability distribution of future pixel can not be inherently computed until the value of previous pixel is computed. Thus, the generation process is naturally slow, which becomes more severe for high dimensional data generation such as speech synthesis [95].

  On the other hand, the generator of GAN is a simple feed-forward network mapping from $\mathcal{Z}$ to $\mathcal{X}$. The generator produces data at once, not pixel by pixel as auto-regressive models. Therefore, GAN can generate samples in parallel, which gives great speed up for sampling, and this property gives more opportunity for GAN to be used in various real applications.

- GAN does not need to approximate a likelihood by introducing a lower bound, like in VAE. As we mentioned in Section 3.2.2, VAE tries to maximize a likelihood by introducing a variational lower bound. The strategy of VAE is to maximize a tractable variational lower bound, leading a likelihood to be guaranteed at least as high as the lower bound, even when the likelihood is intractable. However, VAE still needs assumptions on a prior and posterior distributions, which do not guarantee the tight bound of Equation 42. This strong assumptions on distributions makes the approximation to the maximum likelihood biased.

  In contrast, GAN does not approximate the likelihood, and does not need any probability distribution assumptions. Instead, GAN is designed to solve an adversarial game between the generator and the discriminator, and a Nash equilibrium of GAN game corresponds to finding the real data distribution [37].

- GAN is empirically known to produce better and sharper result than other generative models, especially VAE. In VAE, a distribution of pixel values in the reconstructed image is modeled as a conditional Gaussian distribution. This causes the optimization of $\log p_{\mathrm{g}}(x|z)$ to be equivalent to minimizing the Euclidean term of $-\|x - Decoder(z)\|^2$, which can be interpreted as a regression problem fitting the mean.

  GAN is highly capable of capturing high frequency parts of an image. Since the generator tries to fool the discriminator to recover real data distribution, the generator evolves to lead even the high frequency parts to deceive the discriminator. In addition, some techniques such as PatchGAN in Section 2.3.3 helps GAN produce and capture sharper result more effectively.

5.4.2 Cons

GAN is developed to solve the minimax game between the generator and the discriminator. Though several studies discuss the convergence and the existence of the Nash equilibrium of GAN game, training of GAN is highly unstable and difficult to converge as mentioned in Section 2.3.1 and 2.3.2. GAN solves the minimax game through the gradient descent method iteratively for the generator and the discriminator. In perspective of cost function: $V(G, D)$ in Equation 1, a solution for GAN game is the Nash equilibrium which is a point of parameters where the discriminator's

cost and the generator's cost is minimum with respect to their parameters. However, the decrease of the discriminator's cost function can cause the increase of the generator's cost function and vice versa. Thus a convergence of GAN game may often fail and is prone to be unstable.

Another important issue for GAN is a mode collapse problem. This problem is very detrimental for GAN to be applied in real applications since a mode collapse restricts GAN's ability of diversity. In Equation 1, the generator is only forced to deceive the discriminator, not led for representing multi-modality of a real data distribution. A mode collapse thus can happen even in a simple experiment [14], and this highly discourages applying GAN due to the low diversity. As mentioned in Section 2.4, various studies tried to address the mode collapse by using a new object function [14, 64], or adding new components [14, 35]. However, for a highly complex and multi-modal real data distribution, the mode collapse still remains as a problem GAN has to solve.

5.4.3 Future research areas

As GAN has been popular throughout deep learning, the limitations of GAN mentioned above have recently been improved. For example, to stabilize the GAN training and increase generation performance, a new weight normalization technique [85] and a GAN using another optimization technique [20] was introduced. In order to utilize GAN for more realistic applications, a GAN for partial and noisy observations was proposed and showed improved performance [12].

New methods for using GAN have also been steadily developed so far. For instance, CausalGAN [63] combines a causal implicit generative model with the conditional GAN to replace conditioning by intervention, which enables the model to generate data with desired characteristic combinations that do not exist in the dataset. In addition, a GAN was proposed for new applications such as cipher cracking [36] and object tracking [119]. In the future, we anticipate that the limitations of existing GANs may be solved in novel ways, and GAN will be regarded as the important generative model by conquering areas that existing deep learning based models can not solve well.

6 Conclusion

We discussed how various object functions and architectures affect the behavior of GAN and the applications of GAN such as image translation, image attribute editing, domain adaptation, and other fields. The GAN originated from the theoretical minimax game perspective. In addition to the standard GAN [38], practical trials as well as mathematical approaches have been adopted, resulting in many variants of GAN. Furthermore, the relationship between GAN and other concepts such as imitation learning, and other generative models has been discussed and combined in various studies, resulting in rich theory and numerous application techniques. GAN has the potential to be applied in many application domains including those we have discussed. Despite GAN's significant success, there remain unsolved problems in the theoretical aspects as to whether GAN actually converges and whether it can perfectly overcome mode collapse, as Arora et al. [6], Grnarova et al. [41], and Mescheder et al. [81] discussed. However, with the power of deep neural networks and with the utility of learning a highly non-linear mapping from latent space into data space, there remain enormous opportunities to develop GAN further and to apply GAN to various applications and fields.

References

[1] Pieter Abbeel and Andrew Y Ng. Inverse reinforcement learning. In Encyclopedia of machine learning, pages 554–558. Springer, 2011.

[2] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. arXiv preprint arXiv:1412.4446, 2014.

[3] Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. Face aging with conditional generative adversarial networks. arXiv preprint arXiv:1702.01983, 2017.

[4] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862, 2017.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017.

[6] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). arXiv preprint arXiv:1703.00573, 2017.

[7] Arindam Banerjee, Xin Guo, and Hui Wang. On the optimality of conditional expectation as a bregman predictor. IEEE Transactions on Information Theory, 51(7):2664–2669, 2005.

[8] Shane Barratt and Rishi Sharma. A note on the inception score. arXiv preprint arXiv:1801.01973, 2018.

[9] Marc G Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients. arXiv preprint arXiv:1705.10743, 2017.

[10] Sagie Benaim and Lior Wolf. One-sided unsupervised domain mapping. arXiv preprint arXiv:1706.00826, 2017.

[11] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. arXiv preprint arXiv:1703.10717, 2017.

[12] Ashish Bora, Eric Price, and Alexandros G. Dimakis. AmbientGAN: Generative models from lossy measurements. In International Conference on Learning Representations, 2018. URL https://openreview.net/forum?id=Hy7fDog0b.

[13] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. arXiv preprint arXiv:1612.05424, 2016.

[14] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. arXiv preprint arXiv:1612.02136, 2016.

[15] Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. arXiv preprint arXiv:1702.07983, 2017.

[16] 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, pages 2172–2180, 2016.

[17] Wei Dai, Joseph Doyle, Xiaodan Liang, Hao Zhang, Nanqing Dong, Yuan Li, and Eric P Xing. Scan: Structure correcting adversarial network for chest x-rays organ segmentation. arXiv preprint arXiv:1703.08770, 2017.

[18] Ivo Danihelka, Balaji Lakshminarayanan, Benigno Uria, Daan Wierstra, and Peter Dayan. Comparison of maximum likelihood and gan-based training of real nvps. arXiv preprint arXiv:1705.05263, 2017.

[19] Ayushman Dash, John Cristian Borges Gamboa, Sheraz Ahmed, Muhammad Zeshan Afzal, and Marcus Liwicki. Tac-gan-text conditioned auxiliary classifier generative adversarial network. arXiv preprint arXiv:1703.06412, 2017.

[20] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training GANs with optimism. In International Conference on Learning Representations, 2018. URL https://openreview.net/forum?id=SJJySbbAZ.

[21] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. Chemometrics and intelligent laboratory systems, 50(1):1–18, 2000.

[22] Emily Denton, Sam Gross, and Rob Fergus. Semi-supervised learning with context-conditional generative adversarial networks. arXiv preprint arXiv:1611.06430, 2016.

[23] Carl Doersch. Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908, 2016.

[24] Chris Donahue, Akshay Balsubramani, Julian McAuley, and Zachary C Lipton. Semantically decomposing the latent spaces of generative adversarial networks. arXiv preprint arXiv:1705.07904, 2017.

[25] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. arXiv preprint arXiv:1605.09782, 2016.

[26] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285, 2016.

[27] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. arXiv preprint arXiv:1606.00704, 2016.

[28] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. arXiv preprint arXiv:1611.01673, 2016.

[29] Kiana Ehsani, Roozbeh Mottaghi, and Ali Farhadi. Segan: Segmenting and generating the invisible. arXiv preprint arXiv:1703.10239, 2017.

[30] Werner Fenchel. On conjugate convex functions. Canad. J. Math, 1(73-77), 1949.

[31] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. arXiv preprint arXiv:1611.03852, 2016.

[32] Robert T. Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. IEEE Transactions on pattern analysis and machine intelligence, 10(4): 439–451, 1988.

[33] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. arXiv preprint arXiv:1612.05872, 2016.

[34] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, pages 2414–2423. IEEE, 2016.

[35] Arnab Ghosh, Viveka Kulharia, Vinay Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. arXiv preprint arXiv:1704.02906, 2017.

[36] Aidan N. Gomez, Sicong Huang, Ivan Zhang, Bryan M. Li, Muhammad Osama, and Lukasz Kaiser. Unsupervised cipher cracking using discrete GANs. In International Conference on Learning Representations, 2018. URL https://openreview.net/forum?id=BkeqO7x0-.

[37] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160, 2016.

[38] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[39] Mahesh Gorijala and Ambedkar Dukkipati. Image generation and editing with variational info generative adversarialnetworks. arXiv preprint arXiv:1701.04568, 2017.

[40] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In Acoustics, speech and signal processing (icassp), 2013 ieee international conference on, pages 6645–6649. IEEE, 2013.

[41] Paulina Grnarova, Kfir Y Levy, Aurelien Lucchi, Thomas Hofmann, and Andreas Krause. An online learning approach to generative adversarial networks. arXiv preprint arXiv:1706.03269, 2017.

[42] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. arXiv preprint arXiv:1611.02247, 2016.

[43] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. arXiv preprint arXiv:1705.10843, 2017.

[44] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028, 2017.

[45] Xin Guo, Johnny Hong, Tianyi Lin, and Nan Yang. Relaxed wasserstein with applications to gans. arXiv preprint arXiv:1705.07164, 2017.

[46] Leonid G Hanin. Kantorovich-rubinstein norm and its application in the theory of lipschitz spaces. Proceedings of the American Mathematical Society, 115(2):345–352, 1992.

[47] R Devon Hjelm, Athul Paul Jacob, Tong Che, Kyunghyun Cho, and Yoshua Bengio. Boundary-seeking generative adversarial networks. arXiv preprint arXiv:1702.08431, 2017.

[48] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In Advances in Neural Information Processing Systems, pages 4565–4573, 2016.

[49] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[50] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. arXiv preprint arXiv:1711.03213, 2017.

[51] Chin-Cheng Hsu, Hsin-Te Hwang, Yi-Chiao Wu, Yu Tsao, and Hsin-Min Wang. Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. arXiv preprint arXiv:1704.00849, 2017.

[52] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. arXiv:1612.04357, 2016. URL https://arxiv.org/abs/1612.04357.

[53] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, page 4, 2017.

[54] Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. Generating images with recurrent adversarial networks. arXiv preprint arXiv:1602.05110, 2016.

[55] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004, 2016.

[56] Edwin T Jaynes. Information theory and statistical mechanics. Physical review, 106(4):620, 1957.

[57] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Gang of gans: Generative adversarial networks with maximum margin ranking. arXiv preprint arXiv:1704.04865, 2017.

[58] Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. arXiv preprint arXiv:1612.00215, 2016.

[59] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196, 2017.

[60] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. arXiv preprint arXiv:1703.05192, 2017.

[61] Taeksoo Kim, Byoungjip Kim, Moonsu Cha, and Jiwon Kim. Unsupervised visual attribute transfer with reconfigurable generative adversarial networks. arXiv preprint arXiv:1707.09798, 2017.

[62] Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. Adversarially regularized autoencoders for generating discrete structures. arXiv preprint arXiv:1706.04223, 2017.

[63] Murat Kocaoglu, Christopher Snyder, Alexandros G Dimakis, and Sriram Vishwanath. Causalgan: Learning causal implicit generative models with adversarial training. arXiv preprint arXiv:1709.02023, 2017.

[64] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. How to train your dragan. arXiv preprint arXiv:1705.07215, 2017.

[65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[67] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. arXiv preprint arXiv:1512.09300, 2015.

[68] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. arXiv preprint arXiv:1609.04802, 2016.

[69] Sang-gil Lee, Uiwon Hwang, Seonwoo Min, and Sungroh Yoon. A seqgan for polyphonic music generation. arXiv preprint arXiv:1710.11418, 2017.

[70] Chongxuan Li, Kun Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. arXiv preprint arXiv:1703.02291, 2017.

[71] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. arXiv preprint arXiv:1705.08584, 2017.

[72] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In IEEE CVPR, 2017.

[73] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pages 1718–1727, 2015.

[74] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

[75] Jae Hyun Lim and Jong Chul Ye. Geometric gan. arXiv preprint arXiv:1705.02894, 2017.

[76] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. arXiv preprint arXiv:1705.11001, 2017.

[77] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In Advances in neural information processing systems, pages 469–477, 2016.

[78] Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. Conditional cyclegan for attribute guided face image generation. arXiv preprint arXiv:1705.09966, 2017.

[79] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. arXiv preprint ArXiv:1611.04076, 2016.

[80] Morteza Mardani, Enhao Gong, Joseph Y Cheng, Shreyas Vasanawala, Greg Zaharchuk, Marcus Alley, Neil Thakur, Song Han, William Dally, John M Pauly, et al. Deep generative adversarial networks for compressed sensing automates mri. arXiv preprint arXiv:1706.00051, 2017.

[81] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. arXiv preprint arXiv:1705.10461, 2017.

[82] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. arXiv preprint arXiv:1611.02163, 2016.

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

[84] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. arXiv preprint arXiv:1802.05637, 2018.

[85] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957, 2018.

[86] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. arXiv preprint arXiv:1611.09904, 2016.

[87] Youssef Mroueh and Tom Sercu. Fisher gan. arXiv preprint arXiv:1705.09675, 2017.

[88] Youssef Mroueh, Tom Sercu, and Vaibhava Goel. Mcgan: Mean and covariance feature matching gan. arXiv preprint arXiv:1702.08398, 2017.

[89] Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. In Advances in Neural Information Processing Systems, pages 1786–1794, 2010.

[90] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In Advances in Neural Information Processing Systems, pages 3387–3395, 2016.

[91] Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. arXiv preprint arXiv:1612.00005, 2016.

[92] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In Advances in Neural Information Processing Systems, pages 271–279, 2016.

[93] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. arXiv preprint arXiv:1610.09585, 2016.

[94] H Freyja Ólafsdóttir, Daniel Bush, and Caswell Barry. The role of hippocampal replay in memory and planning. Current Biology, 28(1):R37–R50, 2018.

[95] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499, 2016.

[96] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759, 2016.

[97] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345–1359, 2010.

[98] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional gans for image editing. arXiv preprint arXiv:1611.06355, 2016.

[99] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In ACM Transactions on graphics (TOG), volume 22, pages 313–318. ACM, 2003.

[100] Dmitry Pestov, Xi Wang, Gombojav O Ariunbold, Robert K Murawski, Vladimir A Sautenkov, Arthur Dogariu, Alexei V Sokolov, and Marlan O Scully. Single-shot detection of bacterial endospores via coherent raman spectroscopy. Proceedings of the National Academy of Sciences, 105(2):422–427, 2008.

[101] David Pfau and Oriol Vinyals. Connecting generative adversarial networks and actor-critic methods. arXiv preprint arXiv:1610.01945, 2016.

[102] Guo-Jun Qi. Loss-sensitive generative adversarial networks on lipschitz densities. arXiv preprint arXiv:1701.06264, 2017.

[103] Svetlozar Todorov Rachev et al. Duality theorems for kantorovich-rubinstein and wasserstein functionals. 1990.

[104] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

[105] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 779–788, 2016.

[106] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 234–241. Springer, 2015.

[107] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. arXiv preprint arXiv:1706.04987, 2017.

[108] Murray Rosenblatt. A central limit theorem and a strong mixing condition. Proceedings of the National Academy of Sciences, 42(1):43–47, 1956.

[109] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In Advances in Neural Information Processing Systems, pages 2234–2242, 2016.

[110] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. arXiv preprint arXiv:1701.05517, 2017.

[111] Bernhard Schölkopf and Alexander J Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2002.

[112] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pages 1889–1897, 2015.

[113] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Adversarial representation learning for domain adaptation. arXiv preprint arXiv:1707.01217, 2017.

[114] Haichao Shi, Jing Dong, Wei Wang, Yinlong Qian, and Xiaoyu Zhang. Ssgan: Secure steganography based on generative adversarial networks. arXiv preprint arXiv:1707.01613, 2017.

[115] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. arXiv preprint arXiv:1705.08690, 2017.

[116] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. arXiv preprint arXiv:1802.08735, 2018.

[117] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[118] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE, 1986.

[119] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson Lau, and Ming-Hsuan Yang. Vital: Visual tracking via adversarial learning. arXiv preprint arXiv:1804.04273, 2018.

[120] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. arXiv preprint arXiv:1511.06390, 2015.

[121] Adrian Spurr, Emre Aksan, and Otmar Hilliges. Guiding infogan with semi-supervision. arXiv preprint arXiv:1707.04487, 2017.

[122] Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet. On integral probability metrics,\phi-divergences and binary classification. arXiv preprint arXiv:0901.2698, 2009.

[123] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems, pages 1057–1063, 2000.

[124] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844, 2015.

[125] Alberto Torchinsky and Shilin Wang. A note on the marcinkiewicz integral. In Colloquium Mathematicae, volume 1, pages 235–243, 1990.

[126] Luan Tran, Xi Yin, and Xiaoming Liu. Representation learning by rotating your faces. arXiv preprint arXiv:1705.11136, 2017.

[127] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. arXiv preprint arXiv:1707.04993, 2017.

[128] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Adversarial generator-encoder networks. arXiv preprint arXiv:1704.02304, 2017.

[129] Denis Volkhonskiy, Ivan Nazarov, Boris Borisenko, and Evgeny Burnaev. Steganographic generative adversarial networks. arXiv preprint arXiv:1703.05502, 2017.

[130] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In Advances In Neural Information Processing Systems, pages 613–621, 2016.

[131] Jacob Walker, Kenneth Marino, Abhinav Gupta, and Martial Hebert. The pose knows: Video forecasting by generating pose futures. arXiv preprint arXiv:1705.00053, 2017.

[132] Chaoyue Wang, Chang Xu, Chaohui Wang, and Dacheng Tao. Perceptual adversarial networks for image-to-image transformation. IEEE Transactions on Image Processing, 27(8): 4066–4079, 2018.

[133] Ruohan Wang, Antoine Cully, Hyung Jin Chang, and Yiannis Demiris. Magan: Margin adaptation for generative adversarial networks. arXiv preprint arXiv:1704.03817, 2017.

[134] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4):600–612, 2004.

[135] Max Welling. Fisher linear discriminant analysis. Department of Computer Science, University of Toronto, 3(1), 2005.

[136] Edwin B Wilson and Margaret M Hilferty. The distribution of chi-square. Proceedings of the National Academy of Sciences, 17(12):684–688, 1931.

[137] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Gp-gan: Towards realistic high-resolution image blending. arXiv preprint arXiv:1703.07195, 2017.

[138] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In Advances in Neural Information Processing Systems, pages 82–90, 2016.

[139] Yuan Xue, Tao Xu, Han Zhang, Rodney Long, and Xiaolei Huang. Segan: Adversarial network with multi-scale l_1 loss for medical image segmentation. arXiv preprint arXiv:1706.01805, 2017.

[140] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In European Conference on Computer Vision, pages 776–791. Springer, 2016.

[141] Dong Yang, Tao Xiong, Daguang Xu, Qiangui Huang, David Liu, S Kevin Zhou, Zhoubing Xu, JinHyeong Park, Mingqing Chen, Trac D Tran, et al. Automatic vertebra labeling in large-scale 3d ct using deep image-to-image network with message passing and sparsity regularization. In International Conference on Information Processing in Medical Imaging, pages 633–644. Springer, 2017.

[142] Zili Yi, Hao Zhang, Ping Tan Gong, et al. Dualgan: Unsupervised dual learning for image-to-image translation. arXiv preprint arXiv:1704.02510, 2017.

[143] Weidong Yin, Yanwei Fu, Leonid Sigal, and Xiangyang Xue. Semi-latent gan: Learning to generate and modify facial images from attributes. arXiv preprint arXiv:1704.02166, 2017.

[144] Jaeyoon Yoo, Heonseok Ha, Jihun Yi, Jongha Ryu, Chanju Kim, Jung-Woo Ha, Young-Han Kim, and Sungroh Yoon. Energy-based sequence gans for recommendation and their connection to imitation learning. arXiv preprint arXiv:1706.09200, 2017.

[145] Jaeyoon Yoo, Yongjun Hong, and Sungroh Yoon. Autonomous uav navigation with domain adaptation. arXiv preprint arXiv:1712.03742, 2017.

[146] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In AAAI, pages 2852–2858, 2017.

[147] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. arXiv preprint arXiv:1612.03242, 2016.

[148] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126, 2016.

[149] Shuchang Zhou, Taihong Xiao, Yi Yang, Dieqiao Feng, Qinyao He, and Weiran He. Genegan: Learning object transfiguration and attribute subspace from unpaired data. arXiv preprint arXiv:1705.04932, 2017.

[150] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. arXiv preprint arXiv:1703.10593, 2017.

[151] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In AAAI, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.