# How Generative Adversarial Networks and Its 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 (GANs) have received wide attention in the machine learning field because of their potential to learn high-dimensional, complex real data. Specifically, they do not perform distribution assumptions and can simply infer real-like samples from latent space. This powerful property leads GANs to be applied to various applications such as image synthesis, image attribute editing, image translation, domain adaptation and other academic fields. In this review, we aim to discuss details of GANs for those readers who are familiar but do not comprehend GANs deeply, or who wish to evaluate GANs from various perspectives. We discuss how a GAN operates and the fundamental meaning of various objective functions suggested recently. We then focus on how the GAN can be combined with an auto-encoder framework which makes it possible handle the latent space. As an extension, we also discuss GAN variants that are applied to various tasks and other fields.

## 1 Introduction

Recently in the machine learning field, a generative model has become more important and popular because of its usage in various applications. By estimating real data probability distribution $p_{\text{data}}(x)$ where $x$ exists in the $d$-dimensional real space $R^d$, generative models can be applied to various tasks such as semi-supervised learning [97], [22],[64], image translation [52],[56], [11], domain adaptation [2],[13],[100] and other academic domains [74],[86],[118]. We can produce real-like data, which can be used for filling missing data in semi-supervised learning by training a generative model [105],[109]. In addition, we can transform an image to other specific domains [140],[122],[133] and perform various tasks such as an object transfiguration [139], super resolution [62], 3D image generation [128], [32], and control attributes of data [72],[4],[88],[36].

Formally, because generative models estimate real data probability distribution $p_{\text{data}}(x)$ with real samples, they can be thought as maximizing the likelihood of real samples with parameter $\theta$. With an assumption of independent and identically distributed (i.i.d.) $m$ training samples $x^i$ where $i \in \{1, 2, \cdots m\}$, generative models can be formulated as $\text{argmax}_\theta \prod_{i=1}^{m} p_\theta(x^i)$ where $p_\theta(x)$ stands for a probability distribution of generated samples with parameter $\theta$. We need to find an optimal parameter $\theta^\star$ for maximizing likelihood; therefore, argmax is used instead of max. Intuitively, such a generating process can be thought as minimizing a certain type of distance between $p_{\text{data}}(x)$ and $p_\theta(x)$. Importantly, 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. This relationship is derived as follows:

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

$$= \underset{\theta}{\text{argmax}} \int_x p_{\text{data}}(x) \log p_\theta(x) dx \tag{2}$$

$$= \underset{\theta}{\text{argmin}} \int_x -p_{\text{data}}(x) \log p_\theta(x) dx + p_{\text{data}}(x) \log p_{\text{data}}(x) dx \tag{3}$$

$$= \underset{\theta}{\text{argmin}} \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_\theta(x)} dx \tag{4}$$

$$= \underset{\theta}{\text{argmin}}\, KLD(p_{\text{data}}||p_\theta) \tag{5}$$

Equation 2 is established by the central limit theorem (CLT) [96] in that as $m$ increases, variance of the expectation distribution decreases. Equation 3 can be induced because $p_{\text{data}}(x)$ does not depend on $\theta$ and Equation 5 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 of KLD is achieved when $p_{\text{data}}(x) = p_\theta(x)$. However, we need the model distribution $p_\theta(x)$ to explicitly calculate KLD and more importantly, KLD is not defined when $x$ is outside the supports of $p_\theta(x)$ (approaching infinity).

From this point of view, maximizing the likelihood of given training data can be branched into mainly two types, where one is to explicitly define the model distribution $p_\theta(x)$ and the other is to avoid defining $p_\theta(x)$, and instead indirectly training the generative model via sampling data from $p_\theta(x)$. Figure 1 shows the two major generative models which this paper mostly discusses.

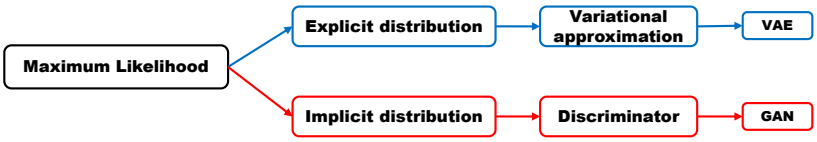



Figure 1: Taxonomy of GAN and VAE [34]

In this study, we focus on generative adversarial networks (GANs) [34] which generates real-like samples by adopting two components, 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 aims to produce real-like samples to fool the discriminator and the discriminator aims to distinguish samples into are real or fake ones. GANs generate real-like samples through indirect information of the discriminator based on an adversarial training framework, so it learns a function which maps the latent variable $z$ into data $x$ which follows $p_\theta(x)$ where $\theta$ is a parameter of the generator. Because GAN does not approximate $p_\theta(x)$ in a certain type, but instead estimates distribution indirectly through the adversarial interaction between the generator and the discriminator, GANs belong to the implicit distribution category of Figure 1. In addition, we look into another important generative model called variational auto-encoder (VAE) [23], which has to explicitly assume a prior distribution and likelihood distribution. Then, we discuss how they differ and various attempts to combine GAN and VAE frameworks to compensate for their individual shortcomings. Furthermore, we also extend our discussion to various applications using GANs such as image translation, semi-supervised learning and other academic fields.

## Paper Organization

Figure 2 shows a tree of the categorized GAN papers. We categorize GAN papers based on theory, treating latent space, application, and this scheme will be consistently maintained throughout this paper. Based on Figure 2, this paper starts by presenting a standard objective function of a GAN and how its components work. After that, we present various objective functions proposed recently, focusing on their similarities in terms of a feature matching problem and architectures adopting multiple components. We extend the discussion to dominant obstacles caused by optimizing a minimax problem, especially mode collapse, and how to address those issues. In Section 3,

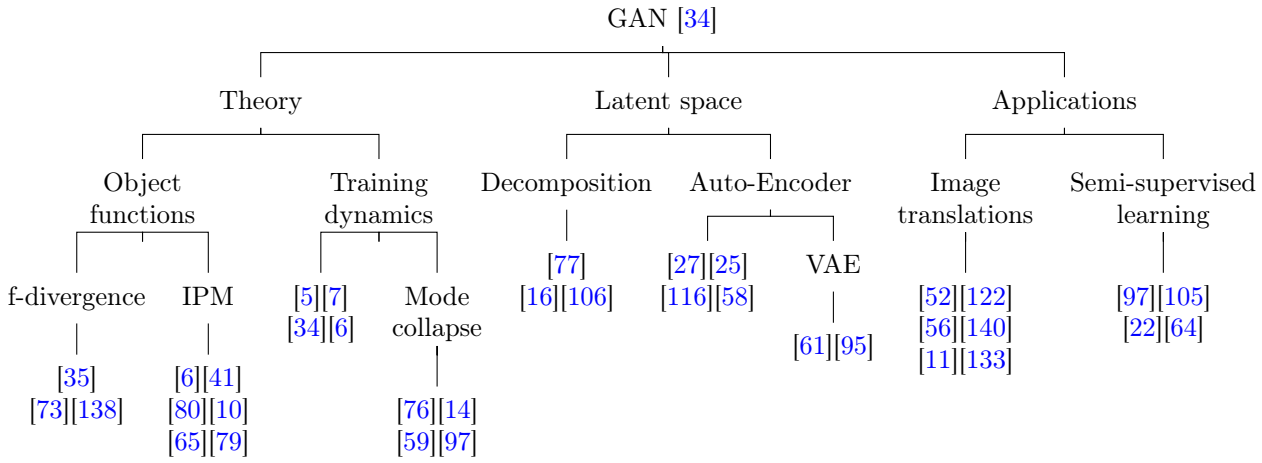



Figure 2: Brief tree of GAN papers. [35]: Standard GAN, [73]: LSGAN, [138]: EBGAN, [6]: WGAN, [97]: Improved WGAN, [80]: McGAN, [10]: CramerGAN, [65]: MMDGAN, [79]: Fisher-GAN, [76]: Unrolled GAN, [14]: MRGAN, [59]: DRAGAN, [77]: Conditional GAN, [16]: InfoGAN, [106]: ss-InfoGAN, [27]: ALI, [25]: BiGAN, [116]: AGE, [58]: ARAE, [61]: VAEGAN, [95]: $\alpha$-GAN, [52]: pix2pix, [122]: PAN, [56]: DiscoGAN, [140]: CycleGAN, [11]: DistanceGAN, [133]: DualGAN, [105]: CatGAN, [22]: SSL-GAN, [64]: Triple-GAN. Arjovsky and Bottou [5], Salimans et al. [97], Goodfellow [34] are theoretical or technical papers so those papers do not have specified name.

we discuss GAN variants representing hybrids with the auto-encoder framework. Particularly, we emphasize how the GAN learns latent space from data space with auto-encoder frameworks. Section 4 provides several extensions of the GAN applied to other domains, especially on image tasks. Section 5 addresses an evaluation problem of the GAN, the relationship with reinforcement learning, and Section 6 concludes the paper.

## 2 Generative Adversarial Networks

Goodfellow et al. [35] proposed a GAN, which is composed of two components, the generator $G$ and the discriminator $D$. $G$ produces fake samples from the latent variable $z$ whereas $D$ takes both fake samples and real samples and decides whether its input is real or fake. $D$ produces higher probability as it determines its input is more likely to be real. $G$ and $D$ oppose each other to achieve their individual goals, so the adversarial term is coined. When this adversarial situation is formulated as the objective function, GAN solves minimax Equation 6 with parametrized networks $G$ and $D$ [35]. $p_{\text{data}}(x)$ and $p_z(z)$ in Equation 6 denote a real data probability distribution defined in the data space $\mathcal{X}$ and a probability distribution of the latent variable $z$ defined on the latent space $\mathcal{Z}$. It should be noted that $G$ maps the latent variable $z$ from $\mathcal{Z}$ into the element of $\mathcal{X}$, whereas $D$ takes an input $x$ and distinguishes whether $x$ comes from real samples or from $G$. The objective function is thus represented as follows:

$$\min_G \max_D V(G,D) = \min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1-D(G(z))] \tag{6}$$

where $V(G,D)$ is a binary cross entropy function which is commonly used in binary classification problems [73].

As $D$ wants to classify real or fake samples, $V(G,D)$ is a natural choice for an objective function. From $D$'s perspective, if a sample comes from real data, $D$ will maximize its output and if a sample comes from $G$, $D$ will minimize its output; thus, the $\log(1-D(G(z)))$ term is naturally derived in Equation 6. Meanwhile, $G$ 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 $V(G,D)$, and which is where the minimax relationship in Equation 6 comes from. Figure 3 shows an outline of the GAN where $D$ produces higher probability (near 1) when it decides its input is more likely to come from real data.

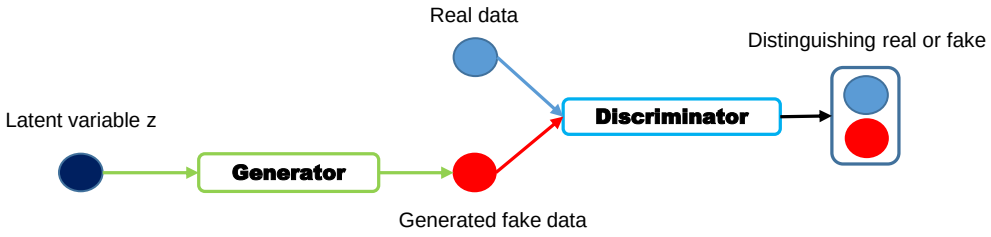



Figure 3: Generative Adversarial Network

Theoretically, assuming that models of $G$ and $D$ both have sufficient capacity, the Nash equilibrium for Equation 6 can be achieved through the following procedure. First, $D$ is trained to obtain an optimal discriminator for a fixed $G$, and then $G$ tries to fool $D$ to enable $D(G(z))$ produce a high probability. By iteratively optimizing such a minimax problem, $D$ cannot discriminate whether its input is real or fake anymore because $p_{\text{data}}(x) = p_\theta(x)$ has been achieved, and $D(x)$ produces a probability of $\frac{1}{2}$ for all real and fake samples. Particularly, Goodfellow et al. [35] shows that solving Equation 6 is equivalent to minimizing the Jensen Shannon Divergence (JSD) between $p_{\text{data}}(x)$ and $p_\theta(x)$.

With this fundamental framework of GAN, we look into variants of object function and architectures proposed for the development of GAN. We then focus on crucial failures of GAN and how to address those issues.

## 2.1 Object Functions

As briefly mentioned in Section 1, generative models aim to match real data distribution $p_{\text{data}}(x)$ from $p_\theta(x)$; thus, minimizing differences between two distributions is a crucial point for training generative models. In this section, we discuss how to measure differences between $p_{\text{data}}(x)$ and $p_\theta(x)$ in mathematical frameworks and various object functions derived from these.

### 2.1.1 f-divergence

f-divergence is a way of measuring differences between two distributions. f-GAN [83] generalizes the GAN objective function in terms of f-divergence under an arbitrary function $f$. The measure for two distributions in f-divergence is defined as follows:

$$D_f(p_{\text{data}}||p_\theta) = \int_{\mathcal{X}} p_\theta(x) f(\frac{p_{\text{data}}(x)}{p_\theta(x)})dx \tag{7}$$

where $D_f(p_{\text{data}}||p_\theta)$ can act as a divergence between two distributions under the conditions that the generator function $f$ is a convex function and $f(1) = 0$ is satisfied. $f(1) = 0$ can be intuitively understood in that if two distributions are equal, their divergence becomes 0.

As we do not know the distributions exactly, Equation 7 should be estimated as an expectation form. Equation 7 can be reformulated with a convex conjugate $f(u) = \sup_{t \in dom f^\star}(tu - f^\star(t))$ as in Equation 8 where $f^\star$ is a Fenchel conjugate [29] of a convex function $f$:

$$D_f(p_{\text{data}}||p_\theta) = \int_{\mathcal{X}} p_\theta(x) \sup_{t \in dom f^\star} (t\frac{p_{\text{data}}(x)}{p_\theta(x)} - f^\star(t))dx \tag{8}$$

$$\geq \sup_{T \in \mathcal{T}} (\int_{\mathcal{X}} (T(x)p_{\text{data}}(x) - f^\star(T(x))p_\theta(x))dx \tag{9}$$

$$= \sup_{T \in \mathcal{T}} (\mathbb{E}_{x \sim p_{\text{data}}}[T(x)] - \mathbb{E}_{x \sim p_\theta}[f^\star(T(x))]) \tag{10}$$

Equation 9 follows from the fact that summation of the maximum is larger than the maximum of the summation and $\mathcal{T}$ is an arbitrary function class which satisfies $\mathcal{X} \to \mathcal{R}$. We can parametrize the generator $G_\theta$ and the discriminator $\mathcal{T}_\omega$ with neural networks, making $T_\omega$ lie in a domain of $f^\star$ because it is derived from $t$ in a domain of Fenchel conjugate $f^\star$. From these parametrized networks, we can create various GAN objective functions with the specified generator function $f$ and the function class $\mathcal{T}$ by maximizing the lower bound in Equation 10 with respect to $T_\omega$ to

make the lower bound tight to $D_f(p_{\text{data}}||p_\theta)$. We then minimize the approximated divergence with respect to $G_\theta$ to make the two distributions similar. This f-GAN framework is quite similar with the standard GAN [35] in that the standard GAN also approximates $JSD(p_{\text{data}}||p_\theta)$ and minimizes the approximated JSD. In fact, the standard GAN is also a member of f-GAN as shown in Table 1.

| GAN | Divergence | Generator $f(t)$ |
|---|---|---|
| | KLD | $t \log t$ |
| GAN [35] | JSD - $2 \log 2$ | $t \log t - (t+1) \log(t+1)$ |
| LSGAN [73] | Pearson $\mathcal{X}^2$ | $(t-1)^2$ |
| EBGAN [138] | Total Variance | $\lvert t-1 \rvert$ |

Table 1: GANs using f-divergence [83]

KLD, reverse KLD, JSD and other divergences can be derived using the f-GAN framework with the specific generator function $f(t)$, even though they are not all represented in Table 1. It should be pointed out that f-divergence $D_f(p_{\text{data}}||p_\theta)$ can be estimated from calculating expectations by introducing a lower bound, not by directly managing unknown probability distributions as in Equation 7.

### 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 in the frame of $f$. In a compact space $\mathcal{X} \subset R^d$, let $\mathcal{P}(\mathcal{X})$ denote the probability measures defined on $\mathcal{X}$. We now define IPM metrics between two distributions $p_{\text{data}}$, $p_\theta \in \mathcal{P}(\mathcal{X})$ as follows:

$$d_{\mathcal{F}}(p_{\text{data}}, p_\theta) = sup_{f \in \mathcal{F}} \; \mathbb{E}_{x \sim p_{\text{data}}}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)] \tag{11}$$

As shown in Equation 11, IPM metric $d_{\mathcal{F}}(p_{\text{data}}, p_\theta)$ defined on $\mathcal{X}$ determines a maximal distance between $p_{\text{data}}(x)$ and $p_\theta(x)$ with functions belonging to $\mathcal{F}$. From Equation 11, we note that $\mathcal{F}$ is a set of measurable, bounded, real-valued functions in that if they are not, $d_{\mathcal{F}}(p_{\text{data}}, p_\theta)$ would not be properly defined. How to define $\mathcal{F}$ determines various distances and their properties. We consider the function class $\mathcal{F}_{v,w}$ whose elements are the critic $f$, which scores its input as a single value so that it can be defined as an inner product of parameterized neural networks $\Phi_w(x)$ and a linear output activation function $v$. It should be noted that $w$ belongs to parameter space $\Omega$ that forces the function space to be bounded. With the definition of the function class in Equation 12, we can reformulate Equation 11 as the following equations:

$$\mathcal{F}_{v,w} = \{f(x) =< v, \Phi_w(x) > | v \in R^m, \Phi_w(x) : \mathcal{X} \to R^m\} \tag{12}$$

$$\begin{aligned} d_{\mathcal{F}_{v,w}}(p_{\text{data}}, p_\theta) &= sup_{f \in \mathcal{F}_{v,w,p}} \mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{x \sim p_\theta} f(x) && (13) \\ &= \max_{w \in \Omega, v} < v, \mathbb{E}_{x \sim p_{\text{data}}} \Phi_w(x) - \mathbb{E}_{x \sim p_\theta} \Phi_w(x) > && (14) \\ &= \max_{w \in \Omega} \max_{v} < v, \mathbb{E}_{x \sim p_{\text{data}}} \Phi_w(x) - \mathbb{E}_{x \sim p_\theta} \Phi_w(x) > && (15) \end{aligned}$$

From Equation 15, how we restrict $v$ determines the semantic meanings of 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 15.

#### 2.1.2.1 Wasserstein GAN

Wasserstein GAN (WGAN) [6] proposes a significant result regarding the distance between $p_{\text{data}}(x)$ and $p_\theta(x)$. In a GAN, we are likely to learn the generator function $g_\theta$ that transforms an existing function $z$ into $p_\theta(x)$ rather than directly learning the probability distribution $p_{\text{data}}(x)$ itself. From this view, a measure between $p_\theta(x)$ and $p_{\text{data}}(x)$ is necessary to train $g_\theta$ and WGAN suggests the Earth-Mover (EM) distance which is also called as Wasserstein distance, as a measure of two distributions. The Wasserstein distance is defined as follows:

$$W(p_{\text{data}}, p_\theta) = inf_{\gamma \in \sqcap(p_{\text{data}}, p_\theta)} \mathbb{E}_{(x,y) \sim \gamma}[\|x - y\|] \tag{16}$$

where $\sqcap(p_{\text{data}}, p_\theta)$ denotes the set of all joint distributions and marginals of $\gamma(x, y)$, which are $p_{\text{data}}(x)$ and $p_\theta(x)$ 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 transforming $p_{\text{data}}(x)$ into $p_\theta(x)$. From this view, calculating the EM distance is equal to finding a transport plan $\gamma(x, y)$, which defines how we distribute the amount of mass from $p_{\text{data}}(x)$ over $p_\theta(y)$. Therefore, a marginality condition can be interpreted in that $p_{\text{data}}(x) = \int_y \gamma(x, y)dy$ is the amount of mass to move from point $x$ and $p_\theta(y) = \int_x \gamma(x, y)dx$ is the amount of mass to be transported to 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 (Equation 16).

The benefit of the EM distance over other metrics is that it is a more sensible objective function when learning distributions supported by low-dimensional manifolds. The article on WGAN shows that EM distance is the weakest convergent metric in that the converging sequence under the EM distance does not converge under other metrics and it is continuous and differentiable almost everywhere under the Lipschitz condition, which general feed-forward neural networks satisfy. This benefit can be intuitively thought of in that the EM distance is a more tolerable measure than other distances such as $KLD$ and total variance distance regarding convergence of the distance. Distances in strongest order among $KLD$, total variance distance, and Wasserstein distance [6] are as follows:

$$KL(p_{\text{data}}||p_\theta),\ KL(p_\theta||p_{\text{data}}) > JS(p_{\text{data}}||p_\theta) = TV(p_{\text{data}}||p_\theta) > W(p_{\text{data}}||p_\theta) \tag{17}$$

where $TV$ and $W$ stand for total variance and Wasserstein distance respectively

As the inf term in Equation 16 is highly intractable, it is converted into a more suitable equation via Kantorovich-Rubinstein duality with the Lipschitz function class [92], [43] for a set of functions such that for the critic $f : X \to R$, satisfies $d_R(f(x_1), f(x_2)) \leq 1 \times d_X(x_1, x_2)$, $\forall x_1, x_2 \in X$ where $d_X$ denotes the distance metric in a domain $X$. A duality of Equation 16 is as follows:

$$W(p_{\text{data}}, p_\theta) = sup_{|f|_L \leq 1}\, \mathbb{E}_{x \sim p_{\text{data}}}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)] \tag{18}$$

Consequently, if we parametrize the critic $f$ with $w$ to be a 1-Lipschitz function, it transforms to solving the minimax problem in that we train $f_w$ first to approximate $W(p_{\text{data}}, p_\theta)$ by searching for the maximum as in Equation 18, and minimize such approximated distance by optimizing the generator $g_\theta$. To guarantee $f_w$ to be a Lipschitz function, weight clipping is conducted for every update of $w$ to ensure the parameter space of $w$ lies in a compact space. It should be noted that $f(x)$ is called the critic because it is not explicitly classifying inputs as the discriminator, rather it scores its input.

#### 2.1.2.2 Variants of WGAN

Improved WGAN [41] points out that 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 weight clipping. It shows that guaranteeing the Lipschitz condition for the critic via weight clipping represents a very limited subset of all Lipschitz functions; this biases the critic toward a simpler function. Weight clipping also creates a gradient problem as it pushes weights to the extremes of the clipping range. Instead of weight clipping, it suggests adding a gradient penalty term to Equation 18 with the purpose of directly constraining the gradient of the critic, which the Lipschitz condition represents.

In addition, LS-GAN [91] also uses a Lipschitz constraint as in WGAN but with a different approach. It learns loss function $L_\theta$ instead of the critic such that the loss of a real sample should be smaller than a generated sample by a data-dependent margin, creating higher focus on fake samples whose margin is high. Moreover, LS-GAN assumes the density of real samples $p_{\text{data}}(x)$ is Lipschitz continuous, so that nearby data does 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 non-parametric assumption that the model has infinite capacity proposed by Goodfellow et al. [35] is too harsh a condition to be satisfied, even using 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 weights of a model to lie in bounded area to ensure the Lipschitz function condition.

Another GAN related to the Wasserstein distance is RWGAN [42]. It proposes a relaxed Wasserstein distance which is a combination of Bregman divergence [9] 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 16 is replaced with the Bregman divergence with a convex function.

#### 2.1.2.3 Feature matching

From Equation 12, we can generalize several IPM metrics under the measure of the inner product. If we constrain $v$ with $p$ norm where $p$ is an integer and $p \in [0, \infty)$, we can derive Equation 21 as a feature matching problem as follows by addition of the $\|v\|_p \leq 1$ condition. It should be noted that the dual norm of $p$ is $q$ such that $\frac{1}{p} + \frac{1}{q} = 1$ and the dual norm $q$ of norm $p$ satisfies $\|x\|_q = sup\{< v, x >: \|v\|_p \leq 1\}$ by Holder's inequality [113]. Motivated by this dual norm property [80], we can derive a $l_q$ mean matching problem as follows:

$$d_{\mathcal{F}}(p_{\text{data}}, p_\theta) = \max_{w \in \Omega} \max_{\|v\|_p \leq 1} < v, \mathbb{E}_{x \sim p_{\text{data}}} \Phi_w(x) - \mathbb{E}_{x \sim p_\theta} \Phi_w(x) > \tag{19}$$

$$= \max_{w \in \Omega} \| \mathbb{E}_{x \sim p_{\text{data}}} \Phi_w(x) - \mathbb{E}_{x \sim p_\theta} \Phi_w(x) \|_q \tag{20}$$

$$= \max_{w \in \Omega} \|\mu_w(p_{\text{data}}) - \mu_w(p_\theta)\|_q \tag{21}$$

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 $\Phi_w$.

In terms of WGAN, WGAN uses the 1-Lipschitz function class condition by weight clipping. Weight clipping means that even the maximum of $v$ is clamped so $p$ norm in Equation 21 can be interpreted as infinite norm $\|v\|_\infty = max_i |v_i|$. Because $p$ is infinite, dual norm $q$ is 1 from the $\frac{1}{p} + \frac{1}{q} = 1$ constraint and WGAN can be interpreted as an $l_1$ mean feature matching problem. McGAN [80] 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 $p_\theta(x)$. Geometric GAN [69] is inspired from McGAN in that the McGAN framework is equivalent to a support vector machine (SVM) [98], which separates a hyperplane that maximizes the margin between two distributions. It details adversarial training in that the discriminator is updated as moving away from the separating hyperplane and the generator is updated as moving 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. Before doing this, we first define some mathematical definitions of a Hilbert space and a kernel.

#### 2.1.2.4 Maximum mean discrepancy

A Hilbert space $\mathcal{H}$ is a vector space where the inner product is defined. Kernel $k$ is defined as $k : \mathcal{X} \times \mathcal{X} \rightarrow R$ such that $k(\cdot, x) \in H, \forall x \in \mathcal{X}$. Reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ is a Hilbert space $\mathcal{H}$ with kernel $k$, where $k$ satisfies a reproducing property, which is defined as $< f, k(\cdot, x) >_{\mathcal{H}} = f(x), \forall x \in \mathcal{X}$. By the Riesz representation theorem [3], we can induce RKHS with indefinite dimensional feature mapping $\Phi : \mathcal{X} \rightarrow \mathcal{H}_k$ such that $k(x, y) =< \Phi(x), \Phi(y) >_{\mathcal{H}_k}$ which satisfies the reproducing property by a positive semidefinite kernel $k$. To summarize, a function $f \in \mathcal{H}_k$ can be evaluated by its inner product with $k(\cdot, x) = \Phi(x)$ as $f(x) =< f, \Phi(x) >_{\mathcal{H}_k}$. Therefore we can redefine Equation 13 with a function class $\mathcal{F} = \{f | \|f\|_{\mathcal{H}_k} \leq 1\}$ and we can derive another mean feature matching relationship with the reproducing property as follows:

$$d_{\mathcal{F}}(p_{\text{data}}, p_\theta) = \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{x \sim p_\theta} f(x) \tag{22}$$

$$= \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \{< f, \mathbb{E}_{x \sim p_{\text{data}}} \Phi(x) - \mathbb{E}_{x \sim p_\theta} \Phi(x) >_{\mathcal{H}_k}\} \tag{23}$$

$$= \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \|\mu(p_{\text{data}}) - \mu(p_\theta)\|^2_{\mathcal{H}_k} \tag{24}$$

where $\mu(\mathcal{P}) = \mathbb{E}_{x\sim\mathcal{P}}\,\Phi(x)$ denotes a kernel embedding mean.

From Equation 24, $d_{\mathcal{F}}$ is defined as the maximum kernel mean discrepancy in RKHS. This method is widely accepted in statistics and is known as a two sample test. Given $p_{\text{data}}(x)$ with mean $\mu_{p_{\text{data}}}$, $p_\theta(x)$ with mean $\mu_{p_\theta}$ and kernel $k$, the square of MMD distance $M_k(p_{\text{data}}, p_\theta)$ is defined as follows:

$$M_k(p_{\text{data}}, p_\theta) = \|\mu_{p_{\text{data}}} - \mu_{p_\theta}\|^2_{\mathcal{H}_k} = \mathbb{E}_{p_{\text{data}}}[k(x,x')] - 2\,\mathbb{E}_{p_{\text{data}},p_\theta}[k(x,y)] + \mathbb{E}_{p_\theta}[k(y,y')] \tag{25}$$

GMMN [67] suggests directly minimizing MMD distance with fixed Gaussian kernel $k(x,x') = exp(-\|x-x'\|^2)$ by optimizing $\min_\theta M_k(p_{\text{data}}, p_\theta)$. It appears quite dissimilar to the standard GAN [35] because there is no discriminator which estimates the discrepancy between two distributions. MMDGAN [65] suggests adversarial kernel learning which replaces the fixed Gaussian kernel into a composition of a Gaussian kernel and a injective function $f_\phi$ as follows: $\tilde{k}(x,x') = exp(-\|f_\phi(x) - f_\phi(x')\|^2)$. Therefore, an objective function with optimizing kernel $k$ becomes to $\min_\theta \max_\phi M_{\tilde{k}}(p_{\text{data}}, p_\theta)$ and is similar to the standard GAN objective as in Equation 6. It should be noted that because $f_\phi$ is modeled with a neural network, an auto-encoder is adopted to enforce $f_\phi$ to be injective function satisfying $f_{decoder} \approx f^{-1}$.

Similar to other IPM metrics, MMD distance is continuous and differentiable almost everywhere in $\theta$. It can also be understood under the IPM framework with function class $\mathcal{F} = \mathcal{H}_K$ as in Equation 12 where its inner product is defined in an RKHS. By introducing an RKHS with kernel $k$, MMD distance has an advantage over other feature matching metrics in that kernel $k$ can map input data $x$ into even infinite dimensional features in linear form. For example, MMDGAN can also be connected with WGAN when $f_\phi$ is composited to a linear kernel with output dimension of 1 instead of Gaussian kernel. However, the moment matching technique using the Gaussian kernel takes 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 [6].

Meanwhile, CramerGAN [10] 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 imposes $h$ to be injective by the additional auto-encoder reconstruction loss as discussed above.

#### 2.1.2.5 Fisher GAN

Instead of using standard IPM in Equation 11, Fisher GAN [79] emphasizes a standardized mean discrepancy which naturally induces a data-dependent constraint by the following equations:

$$d_{\mathcal{F}}(p_{\text{data}}, p_\theta) = \sup_{f\in\mathcal{F}} \frac{\mathbb{E}_{x\sim p_{\text{data}}} f(x) - \mathbb{E}_{x\sim p_\theta} f(x)}{\sqrt{\frac{1}{2}(\mathbb{E}_{x\sim p_{\text{data}}} f^2(x) + \mathbb{E}_{x\sim p_\theta} f^2(x))}} \tag{26}$$

$$= \sup_{f\in\mathcal{F},\, \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}} f^2(x) + \mathbb{E}_{x\sim p_\theta} f^2(x) = 1} \mathbb{E}_{x\sim p_{\text{data}}} f(x) - \mathbb{E}_{x\sim p_\theta} f(x) \tag{27}$$

Equation 26 is motivated from Fisher linear discriminant analysis (FLDA) [125] in that it not only maximizes the mean difference but also reduces the total with-in class variance of two distributions. Equation 27 follows from the constraining numerator of Equation 26 to be 1. It is also, as other IPM metrics, interpreted as a mean feature matching problem but with different constraints. With the definition of Equation 12, Fisher GAN derives another mean feature matching problem with second order moment constraint. A mean feature matching problem derived from the FLDA

concept is as follows:

$$d_{\mathcal{F}}(p_{\text{data}}, p_\theta) = \max_{w \in \Omega} \max_{v} \frac{< v, \mathbb{E}_{x \sim p_{\text{data}}} \Phi(x) - \mathbb{E}_{x \sim p_\theta} \Phi(x) >}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_{\text{data}}} f^2(x) + \mathbb{E}_{x \sim p_\theta} f^2(x))}} \tag{28}$$

$$= \max_{w \in \Omega} \max_{v} \frac{< v, \mathbb{E}_{x \sim p_{\text{data}}} \Phi(x) - \mathbb{E}_{x \sim p_\theta} \Phi(x) >}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_{\text{data}}} v^T \Phi(x) \Phi(x)^T v + \mathbb{E}_{x \sim p_\theta} v^T \Phi(x) \Phi(x)^T v)}} \tag{29}$$

$$= \max_{w \in \Omega} \max_{v} \frac{< v, \mu_w(p_{\text{data}}) - \mu_w(p_\theta) >}{\sqrt{v^T (\frac{1}{2} \sum_w (p_{\text{data}}) + \frac{1}{2} \sum_w (p_\theta) + \gamma I_m) v}} \tag{30}$$

$$= \max_{w \in \Omega} \max_{v, v^T (\frac{1}{2} \sum_w (p_{\text{data}}) + \frac{1}{2} \sum_w (p_\theta) + \gamma I_m) v = 1} < v, \mu_w(p_{\text{data}}) - \mu_w(p_\theta) > \tag{31}$$

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.

Equation 29 can be induced using the inner product of $f$ defined as in Equation 12. $\gamma I_m$ of Equation 30 is an $m$ by $m$ identity matrix that guarantees a numerator of the above equations not to be zero. In Equation 31, Fisher GAN aims to find the embedding direction $v$ which maximizes the mean discrepancy while constraining it to lie in a hyper-ellipsoid as $\frac{1}{2} \sum_w (p_{\text{data}}) + \frac{1}{2} \sum_w (p_\theta) + \gamma I_m) v = 1$ represents. It naturally derives the Mahalanobis distance [20] which is defined as 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 in that it does not impose a data independent constraint such as weight clipping which makes training too sensitive on the clipping value and also has computational benefit over the gradient penalty method in Improved WGAN [97] 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 7 with a convex function $f$, has restrictions where as the dimension of the data space $x \in \mathcal{X} = R^d$ increases, estimation of f-divergence becomes less-tractable because of the integral term for each dimension and the supports of two distributions tends to be unaligned, which leads to infinity [107]. Even though Equation 10 derives a variational lower bound of Equation 7 which looks very similar to Equation 11, a tight lower bound to the true divergence is not guaranteed in practice and can incur an incorrect, biased estimation.

Sriperumbudur et al. [107] showed that only a non-trivial intersection between f-divergence family and IPM family represents total variation distance; therefore, the IPM family does not inherit the disadvantages of f-divergence. Estimating IPM is independent of the data dimension because it directly maximizes mean discrepancy of the critic. Sriperumbudur et al. [107] 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 advantageous over using f-divergence family in that 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 disjoint. In addition, Fisher GAN [79] also is equivalent to the Chi-squared distance [126], which can be covered by an f-divergence framework. However, with a data dependent constraint, Chi-squared distance can take IPM family characteristics, so it is more robust to unstable training of the f-divergence estimation.

### 2.1.3 Auto encoder reconstruction loss

An auto-encoder is widely adopted for GAN architecture to encourage the generator to represent all real samples more accurately using reconstruction loss. Especially, using an auto-encoder as the discriminator, a pixel-wise loss between an input and an output of an auto-encoder is naturally adopted for variants of GAN. A pixel-wise loss for the reconstruction of an auto-encoder 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 is defined as follows:

$$D(v) = \|v - AE(v)\| \tag{32}$$

It is noted that $D(v)$ in Equation 32 is the pixel-wise $l_1$ loss for an auto-encoder which maps an input $v \in R^{N_x}$ into a positive real number $R^+$.

| | Objective function | Details |
|---|---|---|
| BEGAN [12] | $L_D = D(x) - k_t D(G(z))$<br>$L_G = D(G(z))$<br>$k_{t+1} = k_t + \alpha(\gamma D(x) - D(G(z)))$ | Wasserstein distance between loss distributions |
| EBGAN [138] | $L_D = D(x) + [m - D(G(z))]^+$<br>$L_G(z) = D(G(z))$ | Total Variance$(p_{\text{data}}, p_\theta)$ |
| MAGAN [123] | $L_D = D(x) + [m - D(G(z))]^+$<br>$L_G(z) = D(G(z))$ | Margin $m$ is adjusted in EBGAN's training |

Table 2: An auto-encoder based GAN variants [12], [138], [123]

BEGAN [12] uses the fact that pixel-wise loss follows a normal distribution by CLT. It focuses on matching loss distributions through Wasserstein distance, not on matching data distributions. In BEGAN, the discriminator has two roles: one is to auto-encode real samples sufficiently and the other is to balance the generator and the discriminator via equilibrium hyper-parameter $\gamma = \mathbb{E}[L(G(z))]/\mathbb{E}[L(x)]$. $\gamma$ is fed into an objective function to prevent the discriminator from easily winning over the generator; therefore, balances the power of the two components.

EBGAN [138] 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 that the generator attempts to synthesize samples that have lower energy than $m$ to fool the discriminator and that mechanism stabilizes training. MAGAN [123] takes a similar approach to EBGAN, where the only difference is that MAGAN does not fix margin $m$. MAGAN shows empirically that energy of the generated sample fluctuates near margin $m$ and that phenomena with 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, $m$ monotonically reduces so the discriminator auto-encode real samples better.

Table 2 presents summarized details of BEGAN, EBGAN and MAGAN. It should be noted that a pixel-wise loss $L$ is denoted as $D$ to prevent confusion with the generator loss $L_G$ and the discriminator loss $L_D$ and $[t]^+ = \max(0, t)$ represents a maximum value between $t$ and 0.

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 total variance distance by using the fact that the discriminator's output for generated samples is only available for $0 \leq D \leq m$ [6]. Because total variance is the only intersection between IPM and f-divergence [107], it inherits some disadvantages for estimating f-divergence as discussed by Arjovsky et al. [6] and Sriperumbudur et al. [107].

## 2.2 Architecture

An architecture of the generator and the discriminator is important as it greatly influences to the training stability and performance of GAN. Various papers adopt several techniques such as batch normalization, stacked architecture, and multiple generators and discriminators. We start with DCGAN [93], which suggests a remarkable benchmark architecture for other GAN variants.

### 2.2.1 DCGAN

DCGAN provides significant contributions to GAN in that its suggested convolution neural networks (CNN) [60] architecture greatly stabilizes GAN training. DCGAN suggests an architecture guideline in which the generator is modeled with a transposed CNN [26] and the discriminator is modeled with a CNN with output dimension 1. It also details other techniques such as batch normalization and types of activation functions for the generator and the discriminator to help stabilize training of GAN. As it solves the instability of training GAN only through architecture,

it becomes a baseline for modeling various GANs proposed later. For example, GRAN [51] uses a recurrent neural network (RNN) [37] to generate images motivated by DCGAN. By accumulating images of each time step output of DCGAN, it aims to produce higher visual quality images by combining several time step images.

### 2.2.2 GANs using multiple components

There are various proposed architectures which adopt multiple generators and discriminators. For instance, MAD-GAN [33] addresses the mode collapse issue by having multiple generators to learn the modes of real data distribution. The mode collapse issue will be detailed in Section 2.3. Other papers propose using multiple component attempts to decompose the difficult problem of learning image distribution directly into a hierarchical problem in which former layer information helps to learn a later layer generation.

StackedGAN [50] aims to learn 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 semantic features of real samples. Its objective function has two more loss terms in addition to an adversarial objective function. The first is a feature similarity loss which enforces capture of higher level real semantic features from a next level encoder. The second is an entropy loss which encourages the generator to produce diverse representations conditioned on higher level real representations. By adopting a hierarchy in GAN, it aims to produce plausible feature representations given higher level representations.

GoGAN [54] proposes to improve WGAN [6] by adopting multiple WGAN pairs. For each stage, it changes the Wasserstein distance to a margin based one such 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 Wasserstein distance. By progressively moving stages, GoGAN aims to reduce the gap between $p_{\text{data}}(x)$ and $p_\theta(x)$ gradually.

In particular, LR-GAN [132] adopts one discriminator for two generators. LR-GAN generates an image in recursive fashion in that its background generator produces the background of an image and the foreground generator produces an object which is pasted to the generated background. It unfolds generation process over several time steps. The first step generates the background, and the second step generates the foreground which is composed of an object and its mask of it and composites it to the background by element-wise multiplication with its mask. Lastly, it maintains the appropriate transformed foreground given the composited image from last time step. LR-GAN naturally partitions background and foreground from an independent prior $z$ so that composition to a different background with a transformed object becomes more tractable.

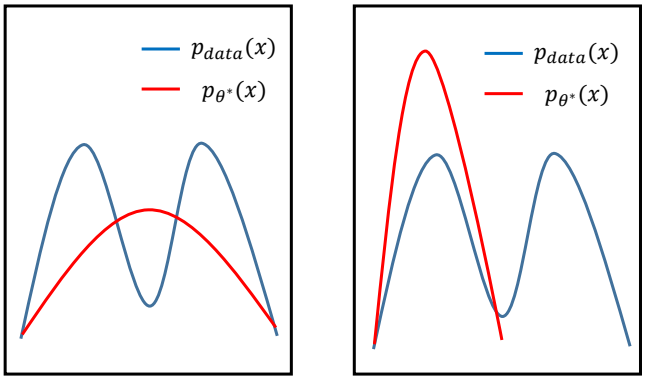
## 2.3 Obstacles in Training GAN

In this section, we discuss theoretical and practical issues related to the training dynamics of GAN. We first evaluate a theoretical problem that is incurred from the fact that the discriminator of GAN aims to approximate the JSD between $p_{\text{data}}(x)$ and $p_\theta(x)$ and the generator of GAN tries to minimize the approximated JSD [35] 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 discussed in Section 1, a traditional generative model approach is to minimize $KL(p_{\text{data}}||p_\theta)$. Because KLD is not symmetrical, minimizing $KL(p_\theta||p_{\text{data}})$ behaves differently. Figure 4 from Goodfellow [34] shows details of different behaviors of asymmetric KLD where Figure 4a shows minimizing $KL(p_{\text{data}}||p_\theta)$ and Figure 4b shows minimizing $KL(p_\theta||p_{\text{data}})$ given a mixture of two Gaussian distributions $p_{\text{data}}(x)$ and the single Gaussian distribution $p_\theta(x)$. $\theta^\star$ in each figure denotes the argument minimum of each asymmetric KLD. For Figure 4a, if $p_{\text{data}}(x) > p_\theta(x)$, mode dropping occurs because $p_\theta(x)$ does not cover some parts of the data distribution and high cost will be incurred. Conversely, if $p_{\text{data}}(x) < p_\theta(x)$, the generator produces samples that do not look sharp but are inexpensive to produce. Therefore, $p_{\theta^\star}(x)$ in Figure 4a is averaged for all modes of $p_{\text{data}}(x)$

as $KL(p_{\text{data}}||p_\theta)$ is more focused on covering all parts of $p_{\text{data}}(x)$. In contrast, $KL(p_\theta||p_{\text{data}})$ has opposite effects, which incur high cost owing for non-realistic generated samples in the case of $p_{\text{data}}(x) > p_\theta(x)$. This is why $p_{\theta^\star}(x)$ in Figure 4b seeks to find an $x$ which is highly likely from $p_{\text{data}}(x)$.




(a) $KL(p_{\text{data}}||p_\theta)$ (b) $KL(p_\theta||p_{\text{data}})$

Figure 4: Different behavior of asymmetric KLD [34]

JSD has an advantage over the two asymmetric KLDs in that it accounts for both mode dropping and sharpness. It never explodes to infinity as KLD even though there exists a point $x$ that lies outside of $p_\theta(x)$'s support which makes $p_\theta(x)$ be 0 as seen in Table 3. Goodfellow et al. [35] showed that the discriminator $D$ aims to approximate the distance between $p_\theta(x)$ and $p_{\text{data}}(x)$, and that approximated distance is equal to $V(G, D^\star) = 2JSD(p_{\text{data}}||p_\theta) - 2\log 2$ for the fixed generator $G$, where $D^\star$ is an optimal discriminator and $V(G, D)$ is defined in Equation 6. Concretely, we attempt to train $D$ well so that it approximates $JSD(p_{\text{data}}||p_\theta) - 2\log 2$ sufficiently and then train $G$ to minimize the approximated distance. However, Arjovsky and Bottou [5] reveal mathematically why approximating $V(G, D^\star)$ does not work well in practice.

| Divergence | Equation | Minimum | Maximum | Symmetry |
|---|---|---|---|---|
| $KLD(p_{\text{data}}\|\|p_\theta)$ | $\int_\chi p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_\theta(x)} dx$ | 0 | $\infty$ | No |
| $JSD(p_{\text{data}}\|\|p_\theta)$ | $\frac{1}{2}KL(p_{\text{data}}\|\|\frac{p_{\text{data}}+p_\theta}{2}) + \frac{1}{2}KL(p_\theta\|\|\frac{p_{\text{data}}+p_\theta}{2})$ | 0 | $\log 2$ | Yes |

Table 3: Comparison between KLD and JSD

Arjovsky and Bottou [5] proved why training GAN is unstable fundamentally. When optimizing $D$ with fixed $G$, $V(G, D)$ tends to approach 0, which does not approximate theoretical distance $V(G, D^\star)$. It means that $JSD(p_{\text{data}}||p_\theta)$ is maxed out to $\log 2$ so $V(G, D^\star)$ goes to 0. This occurs when the supports of two distributions are disjoint or lie in low-dimensional manifolds. For both cases, there exists the perfect discriminator which classifies real or fake samples perfectly so its gradient is 0 at the supports of the two distributions. It is proven empirically and mathematically that $p_{\text{data}}(x)$ and $p_\theta(x)$ derived from $z$ have a low-dimensional manifold in practice [81], and this fact allows $D$'s gradient transferred to $G$ to tend to vanish as $D$ approaches perfectly classifying real or fake samples. Moreover, even with the alternate $-\log D(G(z))$ objective proposed in [35], minimizing an objective function is equivalent to simultaneously trying to minimize $KL(p_\theta||p_{\text{data}})$ and maximizing $JSD(p_\theta||p_{\text{data}})$. As these two objectives are opposites, this leads the magnitude and variance of $D$'s gradients to increase as training progresses and this causes unstable training and making it difficult to converge to the equilibrium. $D$'s vanishing (or exploding) gradient hinders $G$ from learning enough from $D$'s gradient feedback because any update of $G$ is incurred through a composition of $D(G(z))$. 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 guaranteed in practice. 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 non-convergence.

### 2.3.2 Practical issues

First, we represent $G$ and $D$ as deep neural networks to learn parameters rather than directly learning $p_\theta(x)$ itself. Modeling with deep neural networks such as the multi layer perceptron (MLP) or CNN is advantageous in that parameters of distributions can be easily learned through gradient descent using back-propagation and it does not need further distribution assumptions to produce an inference; rather, it can generate samples following $p_\theta(\mathrm{x})$ through simple feed-forward. However, this practical implementation causes a gap with theory. Goodfellow et al. [35] 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. [7] discussed the existence of the equilibrium of GAN, and showed that a large capacity of $D$ does not the guarantee $G$ to generalize all real samples and this means that an equilibrium does not exist under a certain finite capacity of $D$ (capacity is the number of training variables).

A second problem is with an iterative update algorithm suggested in Goodfellow et al. [35]. We wish to train $D$ until optimal for fixed $G$, but optimizing $D$ in such 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 simultaneously by gradient descent in the iterative procedure. Unfortunately, solutions of the minimax and maximin problem are not equal as follows:

$$\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D) \tag{33}$$

With a maximin problem, minimizing $G$ lies in the inner loop in the right side of Equation 33. $G$ is now forced to place its probability mass on the most likely point where the fixed non-optimal $D$ 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 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 real data distribution because $G$ considers itself as fooling $D$ enough only by picking up one mode. Actually, $G$ covers only a single mode or a few modes of real data distribution. This undesirable non-convergent 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. Because the supports of two distributions are disjoint or have low-dimensional manifolds in practice, $D$ improves so that it tends to output close to 1 for real samples while it produces near 0 for fake samples. Because $D$ produces values near 1 for all possible modes, there is no reason 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.

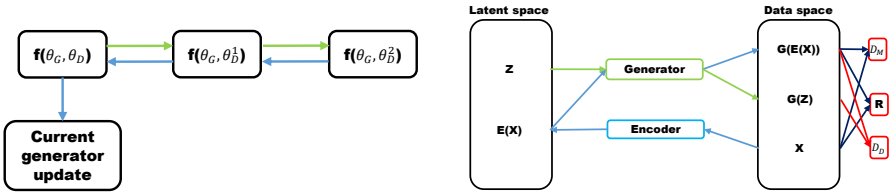
- The theoretical convergence proof does not apply in practice because the generator and the discriminator are modeled with deep neural networks, so optimization has to occur in the parameter space rather than in learning probability density function itself.

## 2.4 Methods to Address Issues in GAN

Mode collapse is the main catastrophic problem for a GAN in that the GAN fails to represent various types of real samples. In this section, we present several studies which suggest new techniques or architectures to address mode collapse phenomena. We also look into Salimans et al. [97], which suggests several methods to improve training GAN.

Unrolled GAN [76] manages mode collapse with a surrogate objective function for updating the generator, which helps the generator to predict the discriminator's response by unrolling the discriminator update $k$ steps for the current generator update. As we see in the standard GAN [35], it updates the discriminator first for the fixed generator and then updates the generator for the updated discriminator. Unrolled GAN differs from the 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 on current generator update. Figure 5a represents an outline of Unrolled GAN where the green arrow stands for the forward pass and the blue arrow represents the generator update flow. 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 fair in that the discriminator can not be trained to optimal in practice due to infeasible computational cost while the generator is assumed to obtain enough information from the optimal discriminator theoretically.

MRGAN [14] assumes that mode collapse occurs because the generator is not penalized for missing modes. To address mode collapse, it adds two regularizing terms for missing modes in its objective function. One is a geometric regularizing term to jointly train an encoder $E(x) : X \to Z$ with a term of $\mathbb{E}_{x \sim p_{\text{data}}}[d(x, G \circ E(x))]$, where $d$ is a $L^2$ metric in the data space. By minimizing $L^2$ distance in the data manifold, it attempts to match the data manifold and the generation manifold. The other term is a mode regularizing term which enforces $G \circ E(x)$ to visit every mode of training samples even to minor modes based on the assumption that $G \circ E(x)$ is a good auto-encoder. In the manifold matching step, the discriminator distinguishes real samples and its reconstruction so that the generator learns how to reconstruct and in distribution diffusion step, another discriminator classifies real samples and generated samples so that the generator learns to produce real-like samples. By matching manifolds of real samples and generated samples, MRGAN aims to make the discriminator pass smooth gradient information to the generator. In the diffusion step, the generator distributes probability mass across different modes fairly by diffusion step. Figure 5b shows an outline of MRGAN where $D_M$ denotes the discriminator which distinguishes real samples and reconstructed ones in the manifold matching step, $D_D$ denotes the other discriminator that distinguishes whether a sample is real or fake in the distribution diffusion step and $R$ denotes a geometric regularizing term.




(a) Unrolled GAN (b) Mode Regularized GAN

Figure 5: Unrolled GAN [76] and Mode Regularized GAN [14]

VEEGAN [108] takes a similar approach to MRGAN in that it also introduces an additional encoder to reconstruct, but differs from MRGAN in that it auto-encodes Gaussian noise vector $z$ rather than data $x$ itself. An advantage of noise auto-encoding is that choosing the $l_2$ reconstruction loss for latent space is more robust to blurry images than pixel-wise $l_2$ reconstruction in the data space and is a more natural choice for standard normal distribution $z$. This noise auto-encoding scheme will be detailed in Section 3.2.3

DRAGAN [59] points out that a mode collapse occurs owing to the existence of a spurious

local Nash Equilibrium in the non-convex 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 Improved WGAN [41]. 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 more higher constraints than DRAGAN.

MAD-GAN [33] adopts multiple generators for one discriminator to capture the diversity of generated samples. 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 results, so it aims to make each generator move toward different modes respectively by optimizing this objective term. 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.

Lastly, we present the methods of Salimans et al. [97], which discuss techniques for training GAN and a method for semi-supervised learning using GAN. Some important methods suggested are listed as follows.

- Feature matching:
  This technique does not focus on the discriminator's output; rather, it substitutes the discriminator's output with an activation function's output of an intermediate layer of the discriminator to prevent over-fitting from the current discriminator. The generator then tries to capture discriminative features of real and fake samples.

- Mini-batch discrimination:
  This approach allows the discriminator look at multiple examples in a mini-batch to avoid a mode collapse of the generator. To make the discriminator process each example with the correlation of other examples, it models a mini-batch layer in an intermediate layer of the discriminator. Therefore, the discriminator can use other examples of a mini-batch while maintaining its natural distinguishing role.

- Label smoothing:
  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. Label smoothing assigns values lower than 1 such as 0.9 to real data labels and values higher than 0 such as 0.1 to generated data labels. By doing this, we prevent the discriminator from passing large gradient signals to the generator.

It should be noted that other techniques such as virtual batch normalization and historical averaging are also proposed. More importantly, a proposed semi-supervised learning method will be presented in Section 4.3

# 3 Treating the Latent Space

Latent space, also called an embedding space when using an auto-encoder, is the space in which a compressed representation of data lies. In VAE, an encoder maps data point $x$ into latent space $z$ and a decoder reconstructs real data while GAN directly maps a Gaussian or uniform latent vector $z$ into the data space. In this section, we investigate how to handle latent space to represent target attributes and how a variational approach can be combined with the GAN framework.

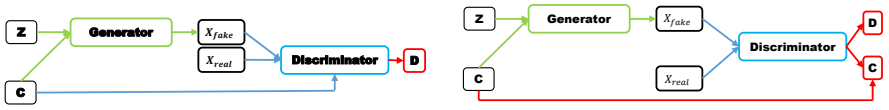
## 3.1 Latent Space Decompose

The input latent vector $z$ to the generator is a highly entangled and unstructured vector in that we do not know which specific vector point contains the representations we want. From this point of view, several papers suggest decomposing the input latent space into a standard input latent vector $z$ and another input vector $c$ we wish to handle.

### 3.1.1 Supervised

Conditional GAN (CGAN) [77] proposes a method of conditioning additional information such as a class label with a latent vector $z$ to control the data generation process in a supervised manner. It performs conditioning by feeding known information vector $c$ to the generator and the discriminator. The generator takes not only a latent vector $z$ but also an additional information vector $c$ and the discriminator also takes samples and a conditional information vector $c$ so that it distinguishes conditioned fake samples given in $c$.

AC-GAN [84] takes a somewhat different approach with CGAN in that it adds an auxiliary label classifying loss term to an adversarial loss term to maximize the log-likelihood of class label of generated samples. Therefore, the discriminator produces not only samples' distinguishing probability but also a probability over the set of class labels. Figure 6 outlines CGAN and ACGAN where $C$ in the red rectangle in Figure 6b denotes the class label probability. It should be noted that condition vector $c$ is fed into the label classifying loss function in ACGAN while $c$ is fed into the discriminator directly in CGAN.




(a) CGAN (b) ACGAN

Figure 6: CGAN [77] and AC-GAN [84]

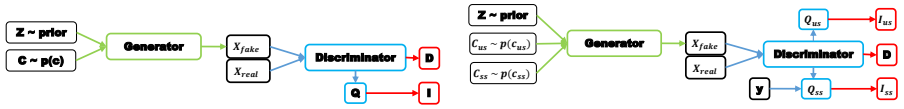
### 3.1.2 Unsupervised

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. Therefore, the generator takes concatenated input as $G(z, c)$ and maximizes mutual information $I(c; G(z, c))$ between a given latent code $c$ and generated samples $G(z, c)$ to learn meaningful feature representations. Because $c$ is to capture some noticeable features of real data, it is reasonable that InfoGAN's objective function aims to maximize the amount of information of $c$ when a generated sample $G(z, c)$ is observed. However, as mutual information $I(c; G(z, c))$ needs to directly access to posterior probability $p(c|x)$, so it takes a variational approach which approximates a target value $I(c; G(z, c))$ by maximizing a lower bound with auxiliary $Q$ networks estimating posterior probability $p(c|x)$.

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 by class labels, so we have to supply $c$ to the generator and the discriminator during the training phase. Meanwhile, $c$ in InfoGAN is assumed to be unknown, 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 in capturing certain features of real data than $c$ in CGAN, which is restricted to known information.

Semi-Supervised InfoGAN (ss-InfoGAN) [106] aims to take advantage of both supervised and unsupervised methods. It improves the unsupervised method of InfoGAN by introducing some label information as a semi-supervised manner by decomposing latent code $c$ in two parts. As in InfoGAN, it still attempts to learn semantic representations of unlabeled data by maximizing mutual information between synthetic data and the unsupervised latent code $c_{us}$. In addition, it adds semi-supervised latent code $c_{ss}$ which accomplishes $c = c_{ss} \cup c_{us}$ and maximizes two mutual information terms. By maximizing the mutual information between labeled real data and $c_{ss}$, it guides $c_{ss}$ to encode label information $y$ and maximizes another mutual information between synthetic data and $c_{ss}$ to pass encoded label information $y$ to the generator via $c_{ss}$. By combining supervised and unsupervised methods, ss-InfoGAN aims to learn the latent code representation more easily than fully unsupervised way of InfoGAN with a small subset of labeled data. Figure 7 shows the flow of InfoGAN and ss-InfoGAN where $Q$ in the blue rectangle represents an auxiliary parametrized network to approximate posterior probability $p(c|x)$ and $I$ in the red rectangle stands for mutual information to be maximized. It should be noted that the two mutual

information terms for $c_{ss}$ are combined into $I_{ss}$ in Figure 7b. The reason why $Q$ is derived from the discriminator is that the discriminator network is commonly shared with auxiliary network $Q$ in the implementation.




(a) InfoGAN (b) ss-InfoGAN

Figure 7: InfoGAN [16] and ss-InfoGAN [106]

## 3.2 With an Auto-Encoder

In this section, we learn efforts on how to combine an auto-encoder structure into the GAN framework. An encoder-decoder structure in which an encoder encodes data $x$ into latent variable $z$ and a decoder reconstructs encoded data into original data $x$ is suitable for stabilizing GAN because it learns the posterior distribution $p(z|x)$ to reconstruct data $x$, thus reducing mode collapse as GAN lacks the inference ability which maps data $x$ into $z$. By learning a latent representation of complex, high-dimensional data space with an encoder $\mathcal{X} \rightarrow \mathcal{Z}$ where $\mathcal{X}$ and $\mathcal{Z}$ denote the data space and the latent space, respectively, manipulations at the abstract level become possible and this facilitates complex modification in the data space to be more easily conducted in the latent space by latent interpolation or conditional concatenation on the latent variable. We discuss studies that adopt an encoder to learn the latent space in Section 3.2.1. We also present techniques for GAN to produce discrete output in Section 3.2.2 and PPGN [82] which represents a sampling technique in the latent space in Section 3.2.3.

We then extend our discussion to proposed ideas which relate a VAE framework into GAN in Section 3.2.4. These studies commonly focus on combining the advantages of VAE and GAN in that VAE suffers less from mode collapse and GAN produces sharper real-like samples than VAE.

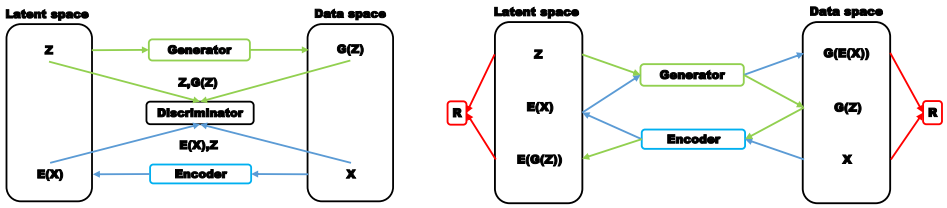
### 3.2.1 Learning the latent space using an encoder

ALI [27] and BiGAN [25] proposed learning latent representations within the GAN framework. As we see in Figure 8a, they aimed to learn the joint probability distribution of data $x$ and latent $z$ while GAN aims to learn directly only the data distribution. The discriminator acts on the joint space of the latent space and the data space and discriminates $(G(z), z)$ and $(x, E(x))$ joint pairs where $G$ and $E$ represent a decoder and an encoder respectively. By training an encoder and a decoder together, they aimed to learn an inference $\mathcal{X} \rightarrow \mathcal{Z}$ while generating sharp high-quality samples.

AGE [116] only focuses on the generator $G$ and an encoder $E$. The two parametrized networks are trained in the latent space and the data space by imposing reconstruction loss terms in each space while minimizing divergence in the latent space. The generator and an encoder are trained adversarially as that the generator aims to minimize divergence between $E(G(z))$ and a prior $z$ in the latent space while an encoder tries to match $E(x)$ and $z$ to align encoded data $E(x)$ with prior $z$ and not match $E(G(z))$ and $z$ by maximizing its divergence. It should be noted that an encoder and the generator act adversarially about the divergence between $E(G(z))$ and a prior $z$ so it skips adopting the discriminator. In addition, it adds reconstruction loss terms in each space to guarantee each component to be reciprocal and these terms can be interpreted as imposing each function to be an one to one mapping function so as not to fall into mode collapse. It is a similar motivation with the geometric regularizing term of MRGAN [14] in that it aims to incorporate a supervised training signal which guides the reconstruction process to the correct location. Figure 8b shows an outline of AGE where $R$ in the red rectangle denotes the reconstruction loss term between the original and reconstructed samples.

### 3.2.2 Discrete structured data

Applying GAN to generate discrete structured data such as text sequences or discretized images is another issue contrasting with generating continuous data, as gradient descent update via




(a) ALI, BiGAN (b) AGE

Figure 8: ALI [27], BiGAN [25], and AGE [116]

back-propagation can not directly be applied for a discrete output which is non-differentiable. Some methods adopt a policy gradient algorithm [110] in reinforcement learning (RL) whose objective is to maximize total rewards by rolling out whole sequences which circumvents direct back-propagation for a discrete output. MaliGAN [15] and BSGAN [44] take a similar approach in that both are inspired from $p_{\text{data}}(x) = \frac{D^*(x)}{1-D^*(x)} p_\theta(x)$ for the optimal discriminator $D^*$ and estimate the true distribution as $\tilde{p}_{data}(x) = \frac{1}{Z} p_\theta(x) \frac{D(x)}{1-D(x)}$ where $Z$ is a normalization factor. They both treat the generator as a policy which outputs a probability for a discrete output which is analogous to RL in that a policy in RL is a probability distribution for taking an action in a given state. They both take $\tilde{p}_{data}(x)$ as a reward for a discrete output $x$. Because higher $\tilde{p}_{data}(x)$ is derived from high $D(x)$ values, it follows that the $\tilde{p}_{data}(x)$ value is taken as a reward. However, this policy gradient approach has a limit in that it can not access latent representations as it does not learn a posterior inference from data $x$ to latent variable $z$.

ARAE [58] addresses this issue by combining a discrete auto-encoder which encodes a discrete input into the continuous code space and adopts WGAN [6] acting on the code space where a code stands for the latent representation of discrete data. To avoid direct access to the discrete structure, WGAN is adopted for the continuous code space where the generator produces fake code $\tilde{c}$ from 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 aims to avoid back-propagation while learning latent representations of discrete structured data.

### 3.2.3 Sampling from latent space

Plug and Play Generative Networks (PPGN) [82] is a type of a generative model derived from sampling techniques. It consists of a generator network $G$ and a condition network $C$, where $G$ produces data, and $C$ performs conditioning through an image classifying function. More specifically, the PPGN framework is derived from activation maximization. Using a chain rule, a joint probability can be split into a constraint that allows data to be in class $y$ and a constraint that allows data to be on the manifold of the image distribution to sample the data as follows:

$$p(x, y) = p(x)p(y|x) \tag{34}$$

In fact, it is common to generate data with a class given as $y = y_c$. This task can be interpreted as a problem to sample data from conditional distribution $p(x|y = y_c)$ and this can be represented by Bayes' rule as follows:

$$p(x|y = y_c) \propto p(x)p(y = y_c|x) \tag{35}$$

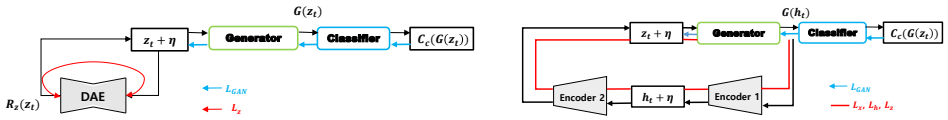
Maximizing $p(x|y = y_c)$ by the activation maximization method is equivalent to maximizing the likelihood that $x$ remains in the manifold of the image distribution and likelihood that the generated data is in the $y_c$ class. $G$ and $C$ perform two tasks respectively. To maximize these likelihoods, we need samples from the data distribution.

Data can be sampled directly from the sampler, but sampling in a high-dimensional space such as image data suffers from poor mixing [8]. Therefore, PPGN introduces latent variable $z_t$ in a low-dimensional latent space. It samples $z_t$ and $z_t$ is used as the input of the generator to extract data. The generator is trained by the feature matching loss proposed in Salimans et al. [97], which is the $l_2$ loss between the real data and the generated data in the feature space. The

next sample $z_{t+1}$ is selected which increases the class probability by the gradient back-propagated from the classifier output as follows:

$$z_{t+1} = z_t + \epsilon_1(R_z(z_t) - z_t) + \epsilon_2 \frac{\partial \log C_c(G(z_t))}{\partial G(z_t)} \frac{\partial G(z_t)}{\partial z_t}, \tag{36}$$

where $R_h(\cdot)$ is the reconstruction function of the denoising auto-encoder (DAE) [117] in the latent space. DAE is an unsupervised learning algorithm that extracts robust features by reconstructing original data from noisy data. $C_c(\cdot)$ denotes the output unit of classifier corresponding to class $y_c$. Sometimes, DAE does not model well owing to unconstrained density estimation. Joint PPGN learns that latent variables become generated data and then decodes such generated data back to latent variables. In this manner, Joint PPGN can sample a latent variable $h$ which is more constrained for $x$. It learns the DAEs for $h_t$, $z_t$ (with the noise vector $\eta$), and $x_t$ that share parameters as seen in Figure 9b. Actually, encoders are frozen and only the generator is learned. Its loss function has four loss terms, an adversarial loss term, and the $l_2$ reconstruction loss terms for $h_t$, $z_t$, and $x_t$.




(a) PPGN-z (b) Joint PPGN-z

Figure 9: PPGN [82]

### 3.2.4 Variational Auto-Encoder

VAE [23] is a popular generative model using an auto-encoder framework. As seen in Figure 1, VAE approximates $p_\theta(x)$ by introducing a variational lower bound, so that it aims to learn the mapping of $\mathcal{X} \rightarrow \mathcal{Z}$ with an encoder and $\mathcal{Z} \rightarrow \mathcal{X}$ with a decoder. With an assumption of some unobserved latent variable $z$ affecting real sample $X$ in an unknown manner, VAE essentially finds the maximum of the marginal likelihood $p_\theta(x)$ for model parameter $\theta$ given the latent variable $z$. Due to the latent variable $z$ affecting data $x$, maximum a posteriori (MAP) method with a prior knowledge of $z$ must be considered instead of maximum likelihood method. However, in general, posteriori probability $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$ is intractable because $p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz$ is not tractable. VAE addresses this issue with a variational inference [120] which induces a variational lower bound for $p_\theta(x)$. Specifically, VAE estimates posteriori probability with an assumption of a prior knowledge $p(z)$ being a normal Gaussian distribution and drives the approximating model $Q_\phi(z|x)$ to approximate real posteriori probability $p(z|x)$ by reducing $KL(Q_\phi(z|x)||p(z|x)) = \int_z Q_\phi(z|x) \log \frac{Q_\phi(z|x)}{p(z|x)} dz$ where $KL$ denotes KLD. More explicitly, we wish to maximize $p_\theta(x)$ where a variational lower bound of the marginal log-likelihood $\log p_\theta(x)$ can be derived as follows:

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

$$= \int_z Q_\phi(z|x)(\log(\frac{Q_\phi(z|x)}{p_\theta(z|x)}) + \log(\frac{p_\theta(x,z)}{Q_\phi(z|x)})) dz \tag{38}$$

$$= KL(Q_\phi(z|x)||p_\theta(z|x)) + \mathbb{E}_{Q_\phi(z|x)}[\frac{\log p_\theta(x,z)}{\log Q_\phi(z|x)}] \tag{39}$$

As $KL(Q_\phi(z|x)||p_\theta(z|X))$ is always a semi-positive number, a variational lower bound $L(\theta, \phi; x)$ of Equation 43 can be derived as follows:

$$\log p_\theta(x) \geq \mathbb{E}_{Q_\phi(z|x)}[\log p_\theta(x,z) - \log Q_\phi(z|x)] \tag{40}$$
$$= \mathbb{E}_{Q_\phi(z|x)}[\log p_\theta(z) - \log p_\theta(x|z) - \log Q_\phi(z|x)] \tag{41}$$
$$= -KL(Q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{Q_\phi(z|x)}[\log p_\theta(x|z)] \tag{42}$$
$$= L(\theta, \phi; x) \tag{43}$$

Therefore, we wish to maximize $L(\theta, \phi; x)$ to maximize marginal likelihood $p_\theta(x)$ and this method can be interpreted as maximizing likelihood $p_\theta(x|z)$ by reducing $KL(Q_\phi(z|x)||p_\theta(z))$ so that an encoder $Q_\phi(z|x)$ approximates a prior $p_\theta(z)$. $p_\theta(x|z)$ 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 $x$ assuming that $Q_\phi(z|x)$ follows conditional Gaussian distribution $\mathcal{N}(z|\mu(x), \sum(x))$ and imposing over $p_\theta(z) \sim \mathcal{N}(0, I)$. Therefore, this model is called VAE and the *variational* term can be considered as the latent $z$ not being fixed for the same input data $x$ since $z$ is sampled from a specific type of distribution.

#### 3.2.4.1 Comparison between VAE and GAN

To deal with the intractable marginal likelihood, VAE attempts to approximate $p_\theta(x)$ via introducing a variational lower bound with a further assumption over a latent variable distribution. VAE generates real-like images through an encoder $Q_\phi(z|x)$ and a decoder $p_\theta(x|z)$ which maximizes likelihood $\log p_\theta(x|z)$ with a prior regularizing term that minimizes $KL(Q_\phi(z|x)||p_\theta(z))$. A tight bound can be accomplished by jointly optimizing an encoder parameter $\phi$ and a decoder parameter $\theta$ via stochastic gradient descent. VAE can be thought of as transforming intractable marginal likelihood into an expectation over $Q_\phi(z|x)$ with a latent variable distributional assumption. Especially, because we assume $Q_\phi(z|x)$ to be a conditional Gaussian distribution given $x$, a distribution of pixel values in the reconstructed image are also modeled as a conditional Gaussian distribution and this causes $\log p_\theta(x|z)$ to be Euclidean term of $-\|x - Decoder(z)\|^2$. This can be interpreted as a regression problem fitting to the mean and causes VAE to generate blurry images.

Moreover, VAE cannot be trained via standard back-propagation. As the model has a stochasticity incurred from the fact that $z$ is sampled from $Q_\Phi(z|x)$ when calculating an expectation term of $\mathbb{E}_{Q_\Phi(z|x)}[\log p_\theta(x|z)]$ in Equation 42, $\max_{\theta,\phi} L(\theta, \phi; x)$ does not work directly and is not differentiable while GAN can be simply trained via standard back-propagation because the generator and the discriminator are assumed to be differentiable. VAE circumvents this problem via reparameterization technique [23]. Lastly, it is emphasized again that VAE tries to fit the real data distribution $p_{\text{data}}(x)$ by optimizing a variational lower bound. However, even if that lower bound is extremely optimized, it is not guaranteed that the tight bound of Equation 43 is achieved because of the simple assumptions of distributions.

In contrast, a GAN's way of solving the intractability of marginal likelihood is to introduce the discriminator which distinguishes its input as real or fake, not approximating nor computing marginal likelihood. GAN does not require an inference of the latent $z$ through an encoder, rather it updates the generator parameter $\theta$ via the discriminator's learning information with very few restrictions. If we parameterize the discriminator $D_\omega$ and denote $y = 1$ as a label of real samples, $y = 0$ as a label of generated samples, the discriminator acts in the role of emitting the probability that an input sample $x$ comes from real data, which can be represented as a conditional probability $p(y = 1|x)$. This binary classifying role can be converted to the density ratio $Dr(x)$ [34], [95] as follows:

$$Dr(x) = \frac{p_{\text{data}}(x)}{p_\theta(x)} = \frac{p(x|y=1)}{p(x|y=0)} = \frac{p(y=1|x)}{p(y=0|x)} = \frac{D_\omega(x)}{1 - D_\omega(x)} \tag{44}$$

where $\frac{p(y=1|x)}{p(y=0|x)}$ can be derived from Bayes rule with an assumption of $p(y = 1) = p(y = 0)$.

Intuitively, we can reformulate equation 44 as $D_\omega(x) = \frac{p_\theta(x)}{p_\theta(x)+p_{\text{data}}(x)}$ and we can notice that this is the optimal discriminator formulation as seen in Goodfellow et al. [35]. Therefore, learning $Dr(x)$ is directly connected to optimizing $V(G, D)$ in Equation 6 in that we train the discriminator only by drawing samples from two distributions, so that GAN focuses on the relative behavior of marginal likelihood $p_\theta(x)$ with respect to true data distribution $p_{\text{data}}(x)$ and the discriminator passes this information to the generator to induce the generator to make more real-like samples. Consequently, GAN does not require approximating or calculating distributions analytically and

therefore, it is branched as an implicit likelihood distribution with the discriminator as shown in Figure 1.

#### 3.2.4.2 Hybrid with GAN

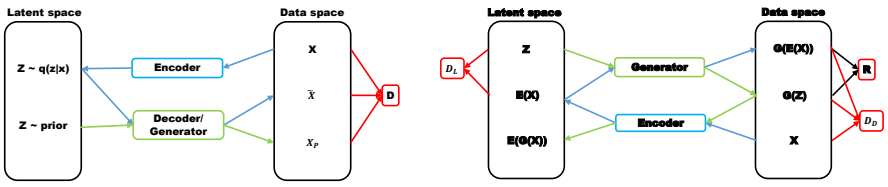
Recently, approaches to incorporate each advantage of VAE and GAN have been proposed. Although VAE generates blurry images, VAE suffers less from mode collapse problem because an auto-encoder encourages all real samples it has seen to be reconstructed. Meanwhile, GAN generates sharper images than VAE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

VAEGAN [61] combines VAE with GAN by assigning GAN's generator to a decoder part. Consequently, its objective function is comprised of VAE's objective function with an adversarial loss term to produce sharp images while maintaining encoding ability for the latent space. Notably, it replaces $x$ of a reconstruction term in Equation 43 with the intermediate features of the discriminator to capture more perceptual similarity of real samples by attempting reconstruction on the feature space. Figure 10a 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 and the other from a prior distribution.

$\alpha$-GAN [95] proposes adopting discriminators to the variational inference and transforms Equation 42 into a more GAN-like formulation. The most negative aspect of VAE is that we have to choose a distribution form of $Q_\phi(z|x)$ to analytically calculate $L(\theta, \phi; x)$. $\alpha$-GAN treats the variational posteriori distribution implicitly by using the density ratio trick technique can be derived as follows:

$$KL(Q_\phi(z|x)||p_\theta(z)) = \mathbb{E}_{Q_\phi(z|x)}[\frac{Q_\phi(z|x)}{p_\theta(z)}] \approx \mathbb{E}_{Q_\phi(z|x)}[\frac{C_\phi(z)}{1 - C_\phi(z)}] \tag{45}$$

Thus, a KLD regularization term of a variational distribution does not need to be calculated analytically anymore by adopting the latent discriminator $C_\phi(z)$ which distinguishes whether the latent variable $z$ comes from an encoder or a prior distribution $p_\theta(z) = \mathcal{N}(0, I)$. $\alpha$-GAN also manages a reconstruction term in Equation 43 by adopting two techniques. One is adopting another discriminator which distinguishes real or synthetic samples, the other is to add a normal $l_1$ pixel-wise reconstruction loss term from the data space. Specifically, $\alpha$-GAN changes a variational lower bound $L(\theta, \phi; x)$ into a more GAN-like formulation by introducing two discriminators using density ratio estimation and reconstruction loss to prevent a mode collapse problem. Figure 10b shows an outline of $\alpha$-GAN where $D_L$ is the latent discriminator, $D_D$ represents the discriminator which acts on the data space and $R$ is the reconstruction loss term.




(a) VAEGAN (b) $\alpha$-GAN

Figure 10: VAEGAN [61] and $\alpha$-GAN [95]

# 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 explicit real data distribution nor assume further mathematical conditions. These advantages lead GAN to be applied to various engineering fields such as images, videos, languages, and other academic 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 from domain X to images from another domain Y. Mainly, translated images have dominants characteristic of domain Y maintaining its attributes before being translated. As in classical machine learning, there are supervised and unsupervised techniques to conduct image translation.

#### 4.1.1.1 Paired two domain data

Image translation with paired images can be regarded as supervised image translation in that image $x \in domain\, X$ always has the target image $y \in domain\, Y$. Pix2pix [52] suggests an image translation method with paired images using the U-NET architecture [94] which is widely used for biomedical image segmentation. It adopts a CGAN framework in that its generator produces a corresponding target image conditioned on an input image. In contrast, PAN [122] adds the perceptual loss between a paired data $(x, y)$ to the generative adversarial loss to transform input image $x$ into ground-truth image $y$ without adopting the CGAN framework [77]. Instead of using the pixel-wise 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 manners is that given unpaired data from two domains, it learns a mapping between two domains without supervision. CycleGAN [140] and DiscoGAN [56] aim to conduct unpaired image-to-image translation using a cyclic consistent loss term. With a sole translator $G$: $X \rightarrow Y$, there can be many possible mappings from $X \rightarrow Y$ so that meaningless translation can occur or mode collapse occurs in which several images in $X$ are mapped to one image in $Y$. They adopt another inverse translator $T$: $Y \rightarrow X$ and introduce the cyclic consistency loss which encourages $T(G(x)) \approx x$ and $G(T(y)) \approx y$. It can be thought of as an auto-encoder scheme with reconstruction loss term to impose meaningful translation of the desired output. In addition, as we discussed in Section 3.2.1, it can be interpreted in similar manner in that it adds a supervised signal for reconstruction to in the correct location. Figure 11 shows the baseline of unpaired image translation where $D$ denotes the discriminator in each domain and $R$ is the for reconstruction loss. In addition, DualGAN [133] also adopts cyclic reconstruction loss but uses the Wasserstein distance instead of a sigmoid cross entropy generative adversarial loss.

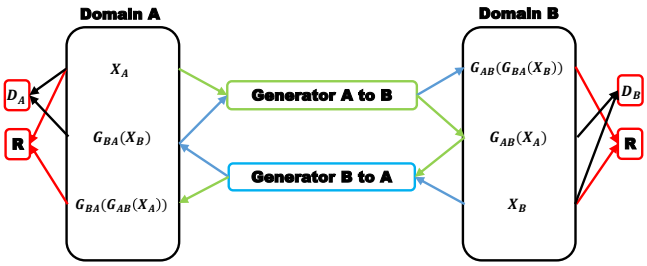



Figure 11: Unpaired image translation

DistanceGAN [11] conducts unsupervised domain mapping without the cyclic concept. It adds distance preserving loss to adversarial loss instead of cyclic consistency loss. Its main idea is that distance between two samples in $X$ should be maintained in $Y$. As it does not use cyclic consistency loss, one-sided domain mapping can be possible without an inverse translator. It also proposes self-distance loss for one mini-batch sample instead of two-sample distance. These concepts for unsupervised image translation can be merged into one equation as follows:

$$\begin{aligned} Loss = {} & \alpha_{1x} L_{GAN}(G_{XY}, D_Y, X, Y) + \alpha_{1y} L_{GAN}(G_{YX}.D_X, Y, X) + \alpha_{2x} L_{cycle}(G_{XY}, G_{YX}, X) \\ & + \alpha_{2y} L_{cycle}(G_{YX}, G_{XY}, Y) + \alpha_{3x} L_{distance}(G_{XY}, X) + \alpha_{3y} L_{distance}(G_{YX}, Y) \end{aligned} \tag{46}$$

Equation 46, which is reproduced from DistanceGAN [11], represents loss terms managing unpaired image translation. The loss function is composed of generative adversarial loss, cyclic consistency loss, and distance preserving loss. In addition, SimGAN [103] tries to refine a simulated synthetic image while maintaining feature characteristics through a self-regularizing loss. It is trained with sets of unpaired sets of synthetic data and real data. Table 4 the studies discussed in terms of Equation 46.

| GAN | $\alpha_{1x}$ | $\alpha_{1y}$ | $\alpha_{2x}$ | $\alpha_{2y}$ | $\alpha_{3x}$ | $\alpha_{3y}$ |
|---|---|---|---|---|---|---|
| DiscoGAN | 1 | 1 | 1 | 1 | 0 | 0 |
| cycleGAN | 1 | 1 | 1 | 1 | 0 | 0 |
| DualGAN | 1 | 1 | 1 | 1 | 0 | 0 |
| DistanceGAN, $X \rightarrow Y$ | 1 | 0 | 0 | 0 | 1 | 0 |
| DistanceGAN, $Y \rightarrow X$ | 0 | 1 | 0 | 0 | 0 | 1 |

Table 4: Comparison of unsupervised image translation [11]

#### 4.1.2 Using latent representation

Modifying images directly in image space is highly difficult because distributions which lie in the image manifold are high-dimensional and complex. Rather than directly manipulating in data space, editing specific attributes through latent space is more tractable in that latent representation expresses specific features of the corresponding image itself and the latent manifold normally has lower dimension as we discussed in Section 3. If we wish to change some attributes, we can concatenate other latent information which contains target attribute, or decompose latent space into identity preserving parts and attribute modifying parts. In this section, we look into various GAN applications which aim to manage attributes of an image with an attribute latent vector.

##### 4.1.2.1 Object transfiguration

Object transfiguration is a type of conditional image generation that replaces an object in an image with a particular condition while the background is maintained unchanged. GeneGAN [139] adopts an encoder-decoder structure to transplant an object. An encoder first decomposes an image to the background feature and the object figure to transfigure, and the decoder reconstructs the image with the decomposed object to transplant the combination with the background feature of the target image.

##### 4.1.2.2 Object detection

Detecting small objects in an image typically suffers from low resolution of an object so that it requires learning at different scales to detect correctly. Perceptual GAN [66] tries to leverage a small object with low resolution into a super-resolved large object, which is similar to the real large scale object using a GAN framework. Perceptual GAN decomposes the discriminator into two branches. The generator tries to produce a real-like large scale object by typical adversarial branch while the perceptual branch guarantees that generated large scale object is reasonable for detection.

SeGAN [28] proposes another framework to detect occluded objects by other objects in an image. It uses a segmentor, generator, and discriminator to extract the entire occluded-object mask and to paint it as if it were looks real-like. 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 a segmented, full painted object with the invisible part conditioned on an intermediate object mask from the segmentor.

##### 4.1.2.3 Image blending

Image blending is a task which implants an object into another image's background and makes such composited copy-paste images look more realistic. GP-GAN [127] applies a high-resolution image blending framework using GAN and a classic image blending gradient based approach [31]. It decomposes images into low-resolution but well-blended images using a GAN and detailed textures

and edges using gradient constraint. Then, GP-GAN attempts to combine the information by optimizing a Gaussian-Poisson equation [89] which generates high-resolution well-blended images while maintaining captured high-resolution details.

#### 4.1.2.4 Text to image

StackGAN [137] synthesizes images conditioned on text description by decomposing the process into two parts: low-level feature generation (stage 1), and painting details from a given generated image at stage 1 (stage 2). Generators of each stage are trained adversarial given a text embedding information $\varphi_t$ for text $t$. It should be noted that conditional augmentation technique is used for generating the text conditioned image, which samples condition vectors from $\mathcal{N}(\mu(\varphi_t), \sum(\varphi_t))$ for text embedding rather than using fixed text embedding $\varphi_t$. It is concatenated with $z$ as in the CGAN and this technique alleviates discontinuity of the latent space arising from a limited amount of image-text data. In contrast, TAC-GAN [19] is based on AC-GAN by conditioning on the text description instead of the class label by simply concatenating fixed text embedding and noise vector $z$. However, with addition of a wrong image and label data, which does not correspond to the text description, it tries to assign the correct text description to the generated image, as does AC-GAN.

#### 4.1.2.5 Editing attributes

AGEGAN [4] trains an encoder to obtain latent vector $z$ which represents a personal identity to be preserved. Then given age vector $y$, it aims to produce facial aging of the target image while maintaining the identity of the original image. ICGAN [88] takes a somewhat different approach to AGEGAN. It adopts two encoders, each produces latent representation and conditional attribute representation, respectively. By conditioning on target-specific attribute information, it generates an image with target attribute conditionally as AGEGAN does.

ViGAN [36] proposes combination of VAE and InfoGAN in that VAE generates the latent representation and the InfoGAN framework produces an attribute-conditioned image with latent variable $c$. SL-GAN [134] takes a similar approach to ViGAN in that it also combines VAE and InfoGAN. However, SL-GAN decomposes the latent space from an encoder into user-defined and data-driven attributes. It tries to maximize mutual information between the data-driven attribute and the generated image given by the user-defined attribute while the InfoGAN part of ViGAN takes input with sampled $z$ from an encoder and known description of data $c$.

DR-GAN [114] and TP-GAN [48] aim to address challenging issues in face recognition. DR-GAN focuses on pose-invariant face recognition, which is a difficult problem because of drastic changes for different poses. It also adopts an encoder-decoder scheme. An encoder tries to represent identity while a decoder produces an image with target pose and identity-given encoded features. these features are concatenated with a pose code $c$ and a code $z$ which represents face rotation. DR-GAN attempts to have the discriminator learn pose classification using a pose code $c$ in decoding so that the discriminator is able to detect pose-variation given the same identity. Meanwhile TP-GAN involves a frontal view synthesis method from other angled profile images. It adopts a two-pathway generator in which a local pathway captures detailed texture and a global pathway infers the global structure of a frontal view.

Attribute guided image translation is considered in Lu et al. [72] and Kim et al. [57]. Conditional CycleGAN [72] embeds target attribute vector $z$ into the CGAN framework with a cyclic consistency framework. Meanwhile, Kim et al. [57] decompose an attribute vector and aim to only transfer target attributes while non-target attributes are kept the same. This technique is imposed by a back transfer loss inspired from the cyclic consistency loss, which is mentioned in Section 4.1.1.2

Synthesizing outdoor images with specific scenery attributes is similar to editing face attributes in that it also adopts the CGAN framework with the attribute latent code $c$. AL-CGAN [55] also takes this conditional approach that takes side information of the layout and content of the scene.

### 4.1.3 Other tasks on images

#### 4.1.3.1 Super resolution

Estimating super resolution images from relatively low resolution images has a fundamental problem in that the recovered high-resolution image misses high-level texture details when upscaling

the image. SRGAN [62] adopts perceptual similarity loss with adversarial loss instead of optimizing pixel-wise mean squared error. It focuses on feature differences from the intermediate layer of the discriminator, not pixel-wise because optimizing pixel-wise mean squared error tries to find the pixel-wise average of a plausible solution, so this leads to perceptually poor smoothed details and is not robust to drastic pixel value changes.

#### 4.1.3.2 3D image generation

Advances in 3D volumetric convolution networks have led to application of GAN to generate 3D objects. 3D-GAN [128] proposes a 3D GAN framework inspired from Radford et al. [93] in that they simply expand to one more dimension to deal with 3-dimensional data. They also extend the 3D framework into 3D-VAE-GAN which aims to map a 2D image into a corresponding 3D object. Meanwhile, PrGAN [32] aims to infer 3D object from multiple 2D views of an object. It also uses a 3D volumetric convolution network to generate 3D shapes in the generator. However, its discriminator takes input as a 2D projected image so that projection of a synthesized 3D object matches to the input 2D view.

#### 4.1.3.3 Joint image generation

Coupled GAN [71] suggests a method of generating multi-domain images especially by weight sharing techniques among GAN pairs. It first adopts GAN pairs to be as many as the number of domains we want to produce. Then, it shares weights of some layers of each GAN pair that represents high-level semantics. Therefore, it attempts to learn joint distributions of a multi-domain from samples drawn from a marginal domain distribution. It should be noted that because it aims to generate multi-domain images which share high-level abstract representations, each domain has to be very similar in a broad view.

SD-GAN [24] takes a similar approach to Coupled GAN. It also adopts multi-GAN pairs but does not utilize a weight sharing technique. It decomposes latent space as discussed in the above paragraph. By partitioning latent variable $z$ into identity and pose parts, it attempts to produce two images which have the same identity while attaining different poses from two generators.

## 4.2 Sequential Data Generation

Besides images, GAN is also applicable to generating sequential data such as music, language and even videos.

### 4.2.1 Music

C-RNN-GAN [78] models both generator and discriminator as an RNN with Long-short Term Memory (LSTM) [46] which is widely used in managing sequential data. It aims to directly extract whole sequences of music. In contrast, SeqGAN [136], ORGAN [40], and Lee et al. [63] employ policy gradient algorithm, not generating whole sequences at once. The reason why they adopt a policy gradient is that normal GAN has poor ability in generating discrete output because the discriminator's gradient hardly affects the generator's output. Moreover, partially generated sequence cannot be evaluated, unlike an entire generated sequence such as in the case of 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 would act to obtain a large output from the discriminator which supports the aim of reinforcement learning. In addition, ORGAN differs from SeqGAN in that it adds a hard-coded objective to reward function to achieve the specified goal.

### 4.2.2 Language and speech

SEGAN [87] attempts to incorporate speech using 1-dimensional dilated convolution inspired from the Wavenet architecture [85]. RankGAN [70] suggests language generation methods using text data generation and a ranker instead of a conventional discriminator. The generator tries to rank the synthetic high among hand-written real data and the ranker evaluates the rank score for each input. As the generator outputs discrete symbols, it also adopts a policy gradient algorithm as does SeqGAN [136]. The generator can be interpreted as a policy predicting next step symbol and the rank score can be interpreted as a value function given a past generated sequence respectively.

VAW-GAN [47] is a voice conversion system combining GAN and VAE frameworks. It adopts a conditional-VAE framework [130]. The encoder represents phonetic content $z$ of source voice and the decoder synthesizes the converted voice given a target speaker's information $y$. As discussed in Section 3.2.4, VAE suffers from generating sharp results due to the over-simplified assumption of Gaussian distribution. To deal with this issue, VAW-GAN incorporates WGAN [6] similarly to VAEGAN [61]. The reason why it takes a WGAN framework with VAEGAN is that VAW-GAN wishes to learn voice conversion more directly by feeding a reconstructed source and target voice directly to the Wasserstein loss function. By assigning the decoder to the generator, it aims to reconstruct the target voice given the speaker representation.

### 4.2.3 Video

In this paragraph, we discuss GAN variants for generating video. Generally, video is composed of relatively stationary background scenery and dynamic object motions. VGAN [119] considers these points and therefore suggests a two-stream generator. A moving foreground generator using 3D CNN predicts plausible futures of frames while a static background generator using 2D CNN makes the background stationary. Pose-GAN [121] takes a mixed VAE and GAN 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, MoCoGAN [115] proposes decomposing the content part and motion part of the latent space, especially modeling the motion part with RNN to capture time dependency. Table 5 summarizes each video generating GAN.

| | 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 |

Table 5: Comparison of video generator [115]

## 4.3 Semi-Supervised Learning

Semi-supervised learning is a learning method that improves classification performance by using unlabeled data in a situation where there are both labeled and unlabeled data. In these times of big data, there commonly exists a situation where the size of the data is too large, or the cost of labeling, such as medical data, 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.

Salimans et al. [97] enabled semi-supervised learning in GAN by constructing the discriminator wherein a unit corresponding to an additional generated class is added to the K output units of the standard classifier. For the labeled real data, the model learns to maximize the log probability of the output unit corresponding to its label, and for unlabeled real data, it maximizes $\log p_\theta(y \in 1, ..., K|x)$. For generated data, the model learns to maximize the log-likelihood of the additional unit corresponding to $y = K + 1$. The objective function for this semi-supervised learning can be expressed as follows:

$$L = L_s + L_{us} \tag{47}$$

$$L_s = -\mathbb{E}_{x,y\sim p_{data}(x,y)}[\log p_\theta(y|x, y < K + 1)] \tag{48}$$

$$L_{us} = -\mathbb{E}_{x\sim p_{data}(x)}[1 - \log p_\theta(y = K + 1|x)] - \mathbb{E}_{x\sim G}[\log p_\theta(y = K + 1|x)] \tag{49}$$

where $L_s$ and $L_{us}$ stand for loss functions of labeled data and unlabeled data respectively. It is also noted that $G$ is the generator of GAN and because generated data also has no label, $L_{us}$ can be expressed as a GAN standard minimax game.

CatGAN [105] proposes an algorithm for robust classification in the regularized information maximization framework. The model is trained on three requirements: To make the class assignment for real data have small conditional entropy $H[p(y|x, D)]$; To make the class assignment for generated data have large conditional entropy $H[p(y|G(z), D)]$; and to make a uniform marginal distribution, i.e. to maximize $H[p(y|D)]$. Combining these three requirements, the objective functions for the generator and the discriminator are composed as follows:

$$L_G = H_G\Big[p(y|D)\Big] - \mathbb{E}_{z\sim P(z)}\Big[H[p(y|G(z), D)]\Big] \tag{50}$$

$$L_D = H_{\mathcal{X}}\Big[p(y|D)\Big] - \mathbb{E}_{x\sim\mathcal{X}}\Big[H[p(y|x, D)]\Big] + \mathbb{E}_{z\sim P(z)}\Big[H[p(y|G(z), D)]\Big]. \tag{51}$$

The mutual information is defined as the difference between the entropy of the marginal class distribution and the entropy of the conditional class distribution, i.e. $I(X;Y) = H(X) - H(X|Y)$. Thus, with $L_G$, we train the generator to minimize the mutual information between the data distribution and the predicted class distribution for the generated data. However, with $L_D$, we train the discriminator to minimize the mutual information between the data distribution and the class distribution for the real data, and to maximize the generated data. For semi-supervised learning, a cross entropy term for labeled data, $\mathbb{E}_{(x,\mathbf{y})\sim\mathcal{X}^L}\Big[CE[\mathbf{y}, p(y|x, D)]\Big]$, is added to $L_D$ where $CE$ refers to cross entropy.

SSL-GAN [22] suggests an algorithm for semi-supervised learning with CC-GAN [22]. CC-GAN's generator fills a missing image patch conditioned on surrounding pixels where the discriminator distinguishes full images as real data, and images with missing patches and inputed images as fake data. The learned generator can be used for missing value computation. Similar to the method used in Springenberg [105], semi-supervised learning can be enabled by adding a cross entropy term $\mathbb{E}_{x,y\sim\mathcal{X}_L}[log(D_c(y|x)]$ for labeled data to the objective function.

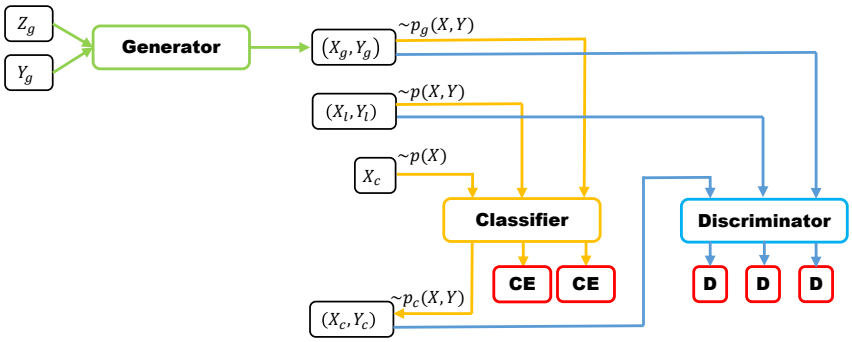



Figure 12: Triple-GAN [64]

However, GAN variants in semi-supervised learning have two problems: the first problem is that the discriminator has two incompatible convergence points, one is for discriminating real and fake data and the other is for predicting the class label, and the second problem is that the generator cannot generate data in a specific class. Triple-GAN [64] addresses both 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 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 labeled data $(X_l, Y_l)$ as real, and generated data with class condition $(X_g, Y_g)$, unlabeled data with predicted label $(X_c, Y_c) \sim p_c(X, Y)$ as fake. The model is illustrated in Figure 12. The three player game of Triple-GAN is formulated as follows:

$$\begin{aligned}\min_{C,G}\max_{D} L(C,G,D) &= \mathbb{E}_{(x,y)\sim p(x,y)}\Big[\log D(x,y)\Big] + CE \\ &+ (1-\alpha)\mathbb{E}_{y\sim p(y),z\sim p_z(z)}\Big[\log(1-D(G(y,z),y))\Big] + \alpha\mathbb{E}_{x\sim p(x),y\sim p_c(y|x)}\Big[\log(1-D(x,y))\Big].\end{aligned} \tag{52}$$

where $CE$ refers to the cross entropy loss. With the cross entropy loss of the classifier, Triple-GAN shows that the unique global optimal is achieved if and only if $p(x, y) = p_g(x, y) = p_c(x, y)$.

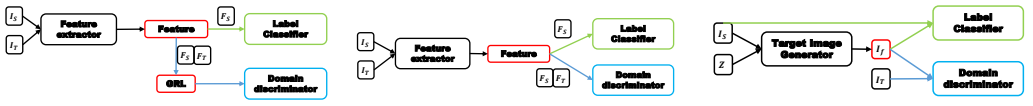
## 4.4 Domain Adaptation

Domain adaptation is a type of transfer learning which tries to match two domain distributions. It can be considered as transferring prior knowledge of a known source domain to a target domain because labeled training data is difficult to obtain in the real world. Explicitly, unsupervised domain adaptation solves the following problem: let a source domain distribution $\mathbb{D}_S$ and a target domain distribution $\mathbb{D}_T$ be defined over $X \times Y$ where $X$, $Y$ are sets of a data space and a label space respectively. Given labeled source domain data $(x_s, y) \in \mathbb{D}_S$ and unlabeled target domain data $x_t \in \mathbb{D}_T$, unsupervised domain adaptation aims to learn a function $h : X \to Y$ which classifies well in a target domain without label information of target domain data $x_t$.

DANN [2] approaches this problem by maintaining discriminative and attaining domain invariance concepts. Maintaining discriminative means maintaining a label prediction ability to classify well in the target domain. Domain invariance stands for making generated features trained for classifying source data, not distinguishing whether it comes from $\mathbb{D}_S$ or $\mathbb{D}_T$. As seen in Figure 13a, there are two components sharing a feature extractor. One is a label classifier which classifies labels of source data. The other is a domain discriminator which distinguishes where data comes from. We train a label classifier to classify labels of source data via a cross entropy loss. We also train a domain discriminator adversarially via gradient reversal layer (GRL), which induces the feature generator to extract domain invariant features. Therefore, the feature generator acts like the generator, which produces source-like features from the target domain, and the domain discriminator acts like the discriminator in that it discriminates source domain samples as real and target domain samples as fake with a binary cross entropy loss. To summarize, DANN takes CNN classification networks in addition with the GAN framework to achieve the goals to classify data from the target domain without label information.

Unsupervised pixel-level domain adaptation [13] takes a different approach to DANN. It is an attempt to adapt source images to be seen as if they were drawn from the target domain while DANN tries to make generated features from both domains similar. There are three components in the suggested model similar to DANN. The generator maps a source image $x_s$ and a noise vector $z$ to a target adapted image $x_f$. The discriminator distinguishes a real target image $x_t$ and a target adapted fake image $x_f$. The classifier assigns a class label for source image $x_s$. In addition, it adds a content similarity loss which reflects prior knowledge of adaptation. Content similarity loss is defined as the pixel-wise mean squared error between source image and adapted image only for the masked area. It informs the adapted image that the masked area is necessarily kept during adaptation.

ARDA [100] aims to solve the target domain unlabeled data classification problem by incorporating WGAN [6]. As other adversarial approaches in the above paragraphs, ARDA also consists of three parts: the feature generator generating domain invariant features, the critic estimating Wasserstein distance between generated features from source and target domains, and the classifier passing the supervised signal from source domain to target domain with cross entropy loss. It differs from DANN in that it accesses the minimax problem with the Wasserstein critic to train the domain discriminator. Figure 13 shows the overall outline of DANN, Unsupervised pixel-level domain adaptation, and ARDA where $I_S, I_T$, and $I_f$ stand for source domain image, target domain image, and fake image and $F_S, F_T$ are mean extracted source features, and target features, respectively.




(a) DANN (b) ARDA (c) Unsupervised pixel-level DA

Figure 13: DANN [2], ARDA [100] and Unsupervised pixel-level domain adaptation [13]

## 4.5 Other Fields

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

### 4.5.1 Medical images

GANCS [74] is used in MRI reconstruction which is a severely ill-posed problem with artifact aliasing. GANCS adopts LSGAN [73] to generate texture details and uses the $l_1$ pixel-wise loss to control high frequency noise. SegAN [129] aims to segment a medical image. It proposes a segmentor-critic structure. A segmentor generates a predicted segmentation image. A critic maximizes hierarchical feature differences between ground-truth and generated segmentation. This induces a segmentor to learn the features of ground-truth segmentation adversarially. In addition, there are other medical image segmentation algorithms such as DI2IN [131] and SCAN [17]. DI2IN takes 3D CT images for the task of liver segmentation through adversarial learning. SCAN tries to segment the lung field and the heart from chest X-Ray images also through an adversarial approach with a ground truth segmentation mask.

### 4.5.2 Steganography

Attempts to use GAN for steganography are also feasible. Steganography involves concealing secret messages in non-secret containers such as an image. Steganalyser determines if a container contains secret message. GAN variants which apply GAN to steganography such as Volkhonskiy et al. [118], and Shi et al. [101] propose a steganography model which consists of three components: a generator which produces real-looking images that will be used for containers and two discriminators where $D$ classifies whether an image is real or fake and $S$ is steganalyser which determines whether an image contains a secrete message.

### 4.5.3 Other applications

GAN variants such as CaloGAN [86] and LAGAN [21] are applied to physics, attempting to generate particle images to represent energy distributions.

Reinforcement Learning (RL) is a type of learning theory that 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 algorithm. Especially, when using off-policy algorithm, past trajectories are used to optimize $\pi(a|s)$. In early training stages, data batches in experience replay are highly biased because a policy $\pi(a|s)$ is far from optimum. EGAN [49] generates an $(s, a, r, s')$ sample from real data via maximizing mutual information between $(s, a)$ and $(r, s')$. By pre-training an agent using such generated data, it aims to train an agent more quickly. Meanwhile, 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 $a$. This similarity will be discussed in Section 5.2.

Shin et al. [102] extend GAN framework to continual learning. Continual learning involves solving multiple tasks without catastrophic forgetting which occurs when a learned previous task is forgotten when a model learns a new task. It addresses this issue with a GAN framework called deep generative replay. It 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 of an old task's sample. By sequentially training scholar models with old task values generated by an old scholar, it aims to overcome catastrophic forgetting even while learning new tasks.

# 5 Discussion

## 5.1 Evaluation

As we start out representing the generative model as maximizing likelihood, it is logical to evaluate the generator using log likelihood. However, as we discussed above, GANs aim to generate a real-like image implicitly without a likelihood assumption so that evaluation with likelihood is not quite proper. The most widely accepted evaluation metric is an inception score. Inception score was proposed by Salimans et al. [97] and was inspired from the Inception Network [111], and is a widely adopted evaluation metric in GAN. An inception score is defined as follows:

$$\exp(\mathbb{E}_x[KL(p(y|x)||p(y))]) \tag{53}$$

As seen in Equation 53, it computes the average KLD between conditional label distribution $p(y|x)$ and marginal label distribution $p(y)$ with a pre-trained classifier such as VGG [104] so

that it aims to capture a well-generated image having a conditional label distribution with low entropy which leads to a high inception score. However it lacks capturing mode collapse because $p(y|x)$ may have low entropy even when mode collapse occurs. To address this issue, Odena et al. [84] adopt MS-SSIM [124], which evaluates perceptual image similarity. In addition, various evaluation metrics are proposed in several papers. For example, MRGAN [14] proposes a MODE score based on inception score and Danihelka et al. [18] suggest 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, TV and Wasseerstein distance can be used as the approximated distance between $p_\theta(x)$ and $p_{data}(x)$.

However, how to evaluate the performance of GAN is difficult in that different objective functions and training procedures lead to different results. All methods discussed in this paper share the same goal which is to fit $p_\theta(x)$ to $p_{data}(x)$ by sampling real samples from $p_{data}(x)$ and training the generator adversarially from the discriminator. Particularly in Section 2.1, we discussed various objective functions where the discriminator approximates true divergence between two distributions and the generator minimizes it. Theoretically, all metrics should produce the same result in assumption of full capacity of a model and an infinite number of training samples. However, Theis et al. [112] show that minimizing MMD, JSD or KLD results in different optimal points even in a mixture of Gaussian distributions and also indicate that convergence in one type of distance does not guarantee convergence for another type of distance and object functions do not share unique global optima. In this regard, we need to be cautious when choosing a proper evaluation metric for a given target application.

### 5.2 Relationship to Reinforcement Learning

Policy gradient algorithm [110] 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) [68], trust region policy optimization (TRPO) [99], and Q-prop [39] 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 $p_\theta(x)$ closely approximate $p_{data}(x)$. Pfau and Vinyals [90] 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 [141] 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 [53]. 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. [30], Yoo et al. [135], and Ho and Ermon [45] showed a mathematical connection between IRL and GAN. Ho and Ermon [45] converted IRL to the original GAN by constraining the space of dual variables and Yoo et al. [135] showed the relationship between EBGAN [138] and IRL using approximate inference.

## 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 [35], 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 theoretical aspects as to in whether GANs actually converge and whether it can perfectly overcome mode collapse, as Arora et al. [7], Grnarova et al. [38], and Mescheder et al. [75] discussed. However, with the power of deep neural networks and with the utility of learning highly non-linear mapping from latent space into data space, there remain enormous opportunities to develop the GAN further and to apply GANs 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] Charalambos D Aliprantis and Kim C Border. Riesz representation theorems. In *Infinite Dimensional Analysis*, pages 455–472. Springer, 1999.

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

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

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

[7] 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.

[8] Yves F Atchade. An adaptive version for the metropolis adjusted langevin algorithm with a truncated drift. *Methodology and Computing in applied Probability*, 8(2):235–254, 2006.

[9] 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.

[10] 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.

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

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

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

[21] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. Learning particle physics by example: Location-aware generative adversarial networks for physics synthesis. *arXiv preprint arXiv:1701.05927*, 2017.

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

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

[30] 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.

[31] 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.

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

[33] 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.

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

[35] 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.

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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

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

[43] 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.

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

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

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

[47] 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.

[48] Rui Huang, Shu Zhang, Tianyu Li, and Ran He. Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis. *arXiv preprint arXiv:1704.04086*, 2017.

[49] Vincent Huang, Tobias Ley, Martha Vlachou-Konchylaki, and Wenfeng Hu. Enhanced experience replay generation for efficient reinforcement learning. *arXiv preprint arXiv:1705.08245*, 2017.

[50] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. *arXiv preprint arXiv:1612.04357*, 2016.

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

[52] 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.

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

[54] 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.

[55] 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.

[56] 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.

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

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

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

[60] 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.

[61] 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.

[62] 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.

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

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

[65] 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.

[66] 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.

[67] 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.

[68] 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.

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

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

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

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

[73] 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.

[74] 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.

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

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

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

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

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

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

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

[82] 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.

[83] 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.

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

[85] 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.

[86] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. Calogan: Simulating 3d high energy particle showers in multi-layer electromagnetic calorimeters with generative adversarial networks. *arXiv preprint arXiv:1705.02355*, 2017.

[87] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*, 2017.

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

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

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

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

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

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

[94] 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.

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

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

[97] 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.

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

[99] 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.

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

[101] 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.

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

[103] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828*, 2016.

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

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

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

[107] 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.

[108] Akash Srivastava, Lazar Valkov, Chris Russell, Michael Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. *arXiv preprint arXiv:1705.07761*, 2017.

[109] Ilya Sutskever, Rafal Jozefowicz, Karol Gregor, Danilo Rezende, Tim Lillicrap, and Oriol Vinyals. Towards principled unsupervised learning. *arXiv preprint arXiv:1511.06440*, 2015.

[110] 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.

[111] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

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

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

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

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

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

[117] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

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

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

[120] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

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

[122] Chaoyue Wang, Chang Xu, Chaohui Wang, and Dacheng Tao. Perceptual adversarial networks for image-to-image transformation. *arXiv preprint arXiv:1706.09138*, 2017.

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

[124] 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.

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

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

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

[128] 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.

[129] 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.

[130] 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.

[131] 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.

[132] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. *arXiv preprint arXiv:1703.01560*, 2017.

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

[134] 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.

[135] 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.

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

[137] 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.

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

[139] 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.

[140] 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.

[141] 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.