# How Generative Adversarial Nets and its variants Work: An Overview of GAN

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

**Abstract**

Generative Adversarial Networks gets wide attention in machine learning field because of its massive potential to learn high dimensional, complex real data. Specifically, it does not need to do further distribution assumption and can simply infer real-like samples from latent space. This powerful property leads GAN to be applied various applications such as image synthesis, image attribute editing and semantically decomposing of image. In this review paper, we look into details of GAN that firstly show how it operates and fundamental meaning of objective functions and point to GAN variants applied to vast amount of tasks.

## 1 Introduction

Generative model is a model which estimates real data probability distribution. By estimating high-dimensional, multi modal distribution of real data, it can be applied to various domains and tasks. Since it estimates real data probability distribution with training real samples, it can be thought as maximizing likelihood of training data with parameter $\theta$. It can be formulated as follows with identical independent distributed $m$ training samples $x^i$.

$$\max_{\theta} \prod_{i=1}^{m} p_\theta(x^i)$$

It can be shown that maximizing log likelihood of training data is equivalent to minimizing Kullback-Leibler Divergence(KLD) between $p_{data}(x)$ and $p_\theta(x)$ as the number of samples $m \to \infty$ .

$$\begin{aligned}
\max_{\theta} \log \prod_{i=1}^{m} p_\theta(x^i) &\Longrightarrow \lim_{m\to\infty} \max_{\theta} \frac{1}{m} \sum_{i=1}^{m} \log p_\theta(x^i) \\
&\approx \max_{\theta} \int_x p_{data}(x) \log p_\theta(x) dx \\
&= \min_{\theta} \int_x -p_{data}(x) \log p_\theta(x) dx + p_{data}(x) \log p_{data}(x) dx \\
&= \min_{\theta} \int_x p_{data}(x) \log \frac{p_{data}(x)}{p_\theta(x)} dx \\
&= \min_{\theta} KL(p_{data} || p_\theta) \qquad (1)
\end{aligned}$$

Third line of equation (1) can be established because $p_{data}(x)$ does not depend on $\theta$ and last line is followed from the definition of KLD. Intuitively, minimizing KLD between these two distributions can be interpreted as approximating $p_{data}(x)$ with large number of training real data since the minimum of KL divergence is achieved when $p_{data}(x) = p_\theta(x)$. However, we need the model distribution $p_\theta(x)$ and more importantly, KLD is not defined when $x$ is outside of supports of $p_\theta(x)$, going to infinity.

Figure 1 stands for two major generative models derived from maximizing likelihood where it is inspired from [1]. In this paper, we focus on Generative Adversarial Networks(GAN) which

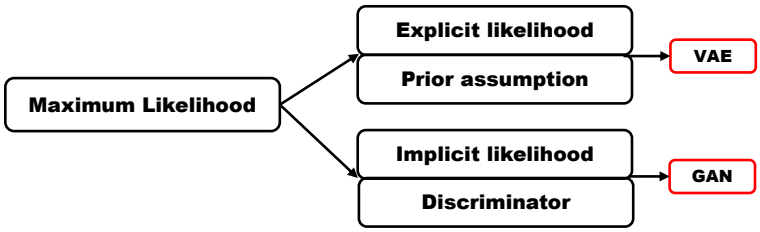



Figure 1: Taxonomy of GAN and VAE

generates data through indirect information of the discriminator so that learns a function which maps a known distribution $z$ into data $x$ which follows $p_\theta(x)$. Since GAN does not approximate $p_\theta(x)$ in certain type, rather manages distribution indirectly through interaction between the generator and the discriminator, GAN is placed lower part of figure 1. Also, we look into another generative model called Variational Auto-Encoder(VAE) which has to explicitly assume a prior distribution and discuss various attempts on how GAN adopts VAE framework to compensate its shortcomings.

### Paper organization

This paper starts by presenting a standard objective function of GAN and how its components work. After that, we will look into various objective functions proposed recently, focusing on their similarities in terms of features matching problem and also architectures adopting multiple components. We extend discussion to dominant obstacles caused by optimizing minimax problem especially mode collapse and how to address those issues. In section 3, we discuss GAN variants focusing on hybrid with auto-encoder framework. Especially, we focus on how GAN learns latent space from data space with auto-encoder framework. Section 4 provides several extensions of GAN applied to other domains, especially on image task. Section 5 comments evaluation problem of GAN, relationship with Reinforcement Learning and section 6 is a conclusion.

### Notations

This paper will consistently use some notations for simplicity. $x$ denotes elements in data space $\mathcal{X} = R^d$ where $d$ stands for the number of dimension. $p_{data}(x)$ denotes real data probability distribution and $p_\theta(x)$ represents generated data probability distribution which comes from the generator function $g_\theta(z) : \mathcal{Z} \to \mathcal{X}$.

## 2 Generative Adversarial Networks

Generative Adversarial Networks(GAN) is composed of two component, the generator $G$ and the discriminator $D$. [2] describes their role as counterfeiter and police officer. $G$ produces fake samples from latent variable $z$ while $D$ takes both fake samples and real samples and decides whether its input is real or fake. $D$ outputs higher probability when it determines a sample is more likely to be real. $G$ and $D$ fight each other to achieve each goals and that's what adversarial term stands for. When it is formulated as an objective function, it is solves following minimax problem.

$$\min_G \max_D V(G,D) = \min_G \max_D \mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z))] \tag{2}$$

$V(G,D)$ is a binary cross entropy commonly used in binary classification problem. Since $D$ wants to classify real or fake samples so $V(G,D)$ is a natural choice for an objective function. Let's look into why minimax comes from. In an angle of $D$, if a sample comes from real data, $D$ would maximize its output and if a sample comes from the generator, $D$ would minimize its output so $\log(1-D(G(z)))$ term is naturally derived in $\mathbb{E}_{z\sim p_z}$ in equation 2. [2] also suggests $\log(1-D(G(z)))$ term can be substituted with $-\log D(G(z))$ heuristically to prevent gradient saturation from the discriminator with guarantee of same result. Meanwhile, $G$ wants to deceive $D$ so tries to maximize

$D$'s output when fake sample gets into $D$. Consequently, $D$ tries to maximize $V(G, D)$ while $G$ tries to minimize $V(G, D)$. Figure 2 shows outline of GAN where $D$ produces higher probability near 1 when it decides input is more likely to come from real world.

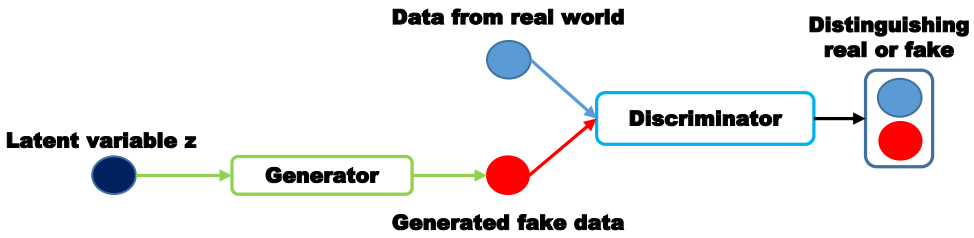



Figure 2: Generative Adversarial Network

Theoretically, assumed that model of $G$ and $D$ have both enough capacity, the Nash equilibrium for equation 2 can be achieved through following procedure. At first, train $D$ to get optimal discriminator for fixed generator. And then the generator tries to fool the discriminator so make $D(G(z))$ outputs high probability. By iteratively optimizing such minimax problem, the discriminator can not anymore discriminate whether coming sample is a real or fake since $D(x) = \frac{1}{2}$ for all real and fake samples $x$ because $p_{data}(x) = p_\theta(x)$ has achieved. Particularly, [2] shows that solving equation 2 is equivalent to minimizing JSD between $p_{data}(x)$ and $p_\theta(x)$.

With this fundamental framework of GAN, we look into variants of object function, architecture proposed for development of GAN. After that, we focus on crucial failures of GAN and how to address those issues.

## 2.1 Object functions

### 2.1.1 f-divergence

f-GAN [3] generalizes GAN objective from f-divergence under arbitrary function $f$ using the ratio of two distributions as follows.

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

$D_f(p_{data}||p_\theta)$ in equation 3 can act as a divergence between two distributions under the conditions that the generator function $f$ is convex and $f(1) = 0$ are satisfied. $f(1) = 0$ can be intuitively understood in that if two distributions are equal, divergence becomes 0. Since we do not know distribution exactly, equation 3 should be estimated as expectation form. Equation 3 can be reformulated with convex conjugate $f^\star(u) = \sup_{t \in dom f}(< t, u > -f(t))$ as equation 4.

$$\begin{aligned} D_f(p_{data}||p_\theta) &= \int_{\mathcal{X}} p_\theta(x) \sup_{t \in dom f^\star} (t\frac{p_{data}(x)}{p_\theta(x)} - f^\star(t)) dx \\ &\geq \sup_{T \in \mathcal{T}} (\int_{\mathcal{X}} (T(x) p_{data}(x) - f^\star(T(x)) p_\theta(x)) dx \\ &= \sup_{T \in \mathcal{T}} (\mathbb{E}_{x \sim p_{data}}[T(x)] - \mathbb{E}_{x \sim p_\theta}[f^\star(T(x))]) \end{aligned} \tag{4}$$

Inequality in equation 4 follows from Jensen's inequality and $\mathcal{T}$ is an arbitrary function class which satisfies $\mathcal{X} \to \mathcal{R}$. If we parametrize the generator $G_\theta$ and the discriminator $\mathcal{T}_\omega$ with neural networks especially making $T_\omega$ lie in domain $f^\star$ since it is derived from $t$ being in domain of convex conjugate $f^\star$, whenever we specify the generator function $f$ with convex conjugate $f^\star$ and the function class $\mathcal{T}$, we can make various GAN objective function for desired divergence by maximizing equation 4 with respect to $T_\omega$ to make lower bound tight to $D_f(p_{data}||p_\theta)$ and minimizing approximated divergence with respect to $G_\theta$ to make two distributions similar. It is pretty alike that standard GAN also approximates $JS(p_{data}||p_\theta)$ and minimize it and in fact, standard GAN is a member of f-GAN as shown in table 1.

| GAN | Divergence | Generator $f(t)$ |
|---|---|---|
| | KLD | $t \log t$ |
| GAN | JSD - $2 \log 2$ | $t \log t - (t+1)\log(t+1)$ |
| LSGAN [4] | Pearson $\mathcal{X}^2$ | $(t-1)^2$ |
| EBGAN [5] | Total Variance | $\|t-1\|$ |

Table 1: GANs using f-divergence

Such as KL divergence, reverse KL divergence and other divergences can be derived using f-GAN framework even though they are not all represented in table 1. It should be pointed out that by introducing a lower bound, f-divergence $D_f(p_{data}||p_\theta)$ can be estimated from calculating expectations, not by directly managing unknown probability distribution as in equation 3. Moreover, it generalizes various divergence under f-divergence framework so that draws corresponding GAN objective with respect to target divergence.

### 2.1.2 Integral Probability Metric

Integral Probability Metric(IPM) defines an a critic function $f$ which belongs to specific function class $\mathcal{F}$, that maximally discriminate between arbitrarily two distributions. Formally, in a compact space $\mathcal{X} \subset R^d$, let $\mathcal{P}(\mathcal{X})$ denote the probability measures defined on $\mathcal{X}$. We can now define IPM metrics between two distributions $p_{data}, p_\theta \in \mathcal{P}(\mathcal{X})$ as in equation 5 and we then solve equation 6 for the generator.

$$d_{\mathcal{F}}(p_{data}, p_\theta) = sup_{f \in \mathcal{F}} \ \mathbb{E}_{x \sim p_{data}}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)] \tag{5}$$

$$\min_\theta d_{\mathcal{F}}(p_{data}, p_\theta) \tag{6}$$

As shown in equation 5, IPM metric $d_{\mathcal{F}}(p_{data}, p_\theta)$ defined on $\mathcal{X}$ finds a maximal distance between $p_{data}(x)$ and $p_\theta(x)$ with functions belongs to $\mathcal{F}$. From equation 5, we can notice that $\mathcal{F}$ is a set of a measurable, bounded and real valued functions in that if they are not, $d_{\mathcal{F}}(p_{data}, p_\theta)$ would not be defined. How to define $\mathcal{F}$ determines various distances and its property. More formally, 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 can be defined as the inner product of parameterized neural networks $\Phi_w(x)$ and linear output activation function $v$ where $w$ without non-linear activation function. It should be noted that $w$ belongs to parameter space $\Omega$ which enforces function space to be bounded. With the definition of the function class as equation 7, we can reformulate equation 5 into equation 8.

$$\mathcal{F}_{v,w} = \{f(x) =< v, \Phi_w(x) > | v \in R^m, \Phi_w(x) : \mathcal{X} \to R^m\} \tag{7}$$

$$\begin{aligned} d_{\mathcal{F}_{v,w}}(p_{data}, p_\theta) &= sup_{f \in \mathcal{F}_{v,w,p}} \mathbb{E}_{x \sim p_{data}} f(x) - \mathbb{E}_{x \sim p_\theta} f(x) \\ &= \max_{w \in \Omega, v} < v, \mathbb{E}_{x \sim p_{data}} \Phi_w(x) - \mathbb{E}_{x \sim p_\theta} \Phi_w(x) > \\ &= \max_{w \in \Omega} \max_{v} < v, \mathbb{E}_{x \sim p_{data}} \Phi_w(x) - \mathbb{E}_{x \sim p_\theta} \Phi_w(x) > \end{aligned} \tag{8}$$

From equation 8, how we restrict $v$ determines semantic meanings of corresponding IPM metric. From now on, we will consider following IPM metric variants such as Wasserstein metric, Maximum Mean Discrepancy(MMD) and fisher metric. Later in this section, we will discuss IPM metrics under the aspect of equation 8.

#### 2.1.2.1 Wasserstein GAN

Wasserstein GAN [6] proposes a significant result about distance between $p_{data}$ and $p_\theta$. In GAN, we are likely to learn a function $g_\theta$ that transforms an existing function $z$ into $p_\theta$ rather directly learn the probability distribution itself. In that point of view, a measure of distance between $p_\theta$ and $p_{data}$ is necessary to train $g_\theta$ and Wasserstein GAN suggests Earth-Mover(EM) distance as a measure of distributions.

$$W(p_{data}, p_\theta) = inf_{\gamma \in \sqcap(p_{data}, p_\theta)} \mathbb{E}_{(x,y) \sim \gamma}[\|x - y\|] \tag{9}$$

$\sqcap(p_{data}, p_\theta)$ denotes the set of all joint distributions and $\gamma(x, y)$ whose marginals are respectively $p_{data}, p_\theta$. Probability distributions can be interpreted with how much mass they put on each point and EM distance is the minimum total amount of work transforming from $p_{data}$ into $p_\theta$. Since probability distributions can be interpreted with how much mass they put on each point finding EM distance is equal to find transport plan $\gamma(x, y)$ which defines how we distribute the amount of mass from $p_{data}(x)$ over $p_\theta(y)$. Therefore, marginality condition is necessary in that $p_{data}(x) = \int_y \gamma(x, y)dy$ 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 Euclidian distance $\|x - y\|$ and $\gamma(x, y)$ at each point $x$, $y$ and the minimum amount of work derives equation 9.

Then what benefits of EM distance over other metrics such as Total Variance(TV), KL divergence and JS divergence? That is EM distance is more sensible objective function than others when learning distributions supported by low dimensional manifolds. [6] shows that EM distance is the weakest convergent metrics in that converging sequence under EM distance does not converge under other metrics and it is continuous and differentiable almost everywhere under Lipschitz condition which feed-forward neural network satisfies. Metrics in strongest order as follows where $TV$ and $W$ stand for Total Variance and Wasserstein distance respectively.

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

Since inf term in equation 9 is highly intractable, it is converted to equation 10 via Kantorovich-Rubinstein Duality [7] with 1-Lipschitz function class which a set of functions such that for $f : X \to R$, 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 distance metric in domain $X$. Consequently, if we parametrize $f_w$ to be a 1-Lipschitz function, it becomes solving minimax problem in that train $f_w$ first to approximate $W(p_{data}, p_\theta)$ by looking for maximum in equation 10 and minimize Wasserstein distance by optimizing the generator $g_\theta$. To guarantee $f_w$ is Lipschitz function, weight clipping to $w$ is conducted for every update of $w$ to ensure parameter space of $w$ lie in compact space. It should be noted that $f(x)$ is called as critic because it is not explicitly classifying inputs real of fake as the discriminator rather it scores its input.

$$W(p_{data}, p_\theta) = sup_{|f|_L \leq 1} \mathbb{E}_{x \sim p_{data}}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)] \tag{10}$$

Improved WGAN [8] points out that weight clipping in critic in training of WGAN incurs a pathological behavior of discriminator and suggests adding penalize term of the gradient's norm instead of weight clipping. It shows that guaranteeing Lipschitz function via weight clipping represents very limited subset of all Lipschitz function so biases critic toward simpler function. Also weight clipping arises gradient problem as it pushes weight to extremes of the clipping range. Instead of weight clipping, it suggests adding gradient penalty term to (7) with purpose of directly constraining the gradient of critic as Lipschitz condition stands for. It is also noticeable that this method is very closed related to DRAGAN in that they both add gradient penalizing term. However they differ in that DRAGAN impose gradient penalty constraint only to local regions around real data manifold while improved WGAN imposes it to almost everywhere around generated data manifold and real data manifold which leads to more highly constraint than DRAGAN.

In addition, LS-GAN [9] also uses Lipschitz constraint as in WGAN but with different approach. It learns loss function $L_\theta$ instead of critic, which loss of real sample should be smaller than that of generated sample by a data-dependent margin so that it more focuses on samples whose margin is high. Moreover, LS-GAN assumes the density of real samples $p_{data}$ is Lipschitz continuous, which represents that nearby data does not abruptly change and the reason for that is quite independent with WGAN's function class Lipschitz condition. It focuses on non-parametric assumption that model has infinite capacity of [2] is too harsh condition to be satisfied even with deep neural networks and causes various problems in training, so it constraints a model to lie in Lipschitz continuous function space while WGAN's Lipschitz comes from Kantorovich-Rubinstein Duality and only the discriminator is constrained. Also, LS-GAN uses weight-decay regularization technique to impose weights to lie in bounded area to ensure Lipschitz function condition.

Lastly, RWGAN [10] proposes relaxed wasserstein distance which is a combination of Bregman divergence [11] and wasserstein distance. It uses a *relaxed* term since it aim to generalize wasserstein-$L^2$ disitance where distance in equation 9 is replaced with Bregman divergence with convex function.

#### 2.1.2.2 Feature Matching

In aspect of equation 7, we can generalize several IPM metrics under the measure of inner product. If we constraint $v$ with $p$ norm where $p$ is an integer and $p \in [0, \infty)$, we can derive equation 11 into feature matching problem as follows in addition of $\|v\|_p \leq 1$ condition. It should be noted that dual norm of $p$ is $q$ such that $\frac{1}{p} + \frac{1}{q} = 1$ and dual norm $q$ of norm $p$ satisfies $\|x\|_q = sup\{< v, x >: \|v\|_p \leq 1\}$ by Holder's inequality. Notice that $\mathbb{E}_{x \sim \mathcal{P}} \Phi_w(x) = \mu_w(\mathcal{P})$ means features embedding from distribution $\mathcal{P}$ and neural networks $\Phi_w$.

$$\begin{aligned} d_{\mathcal{F}}(p_{data}, p_\theta) &= \max_{w \in \Omega} \max_{v, \|v\|_p \leq 1} < v, \mathbb{E}_{x \sim p_{data}} \Phi_w(x) - \mathbb{E}_{x \sim p_\theta} \Phi_w(x) > \\ &= \max_{w \in \Omega} \| \mathbb{E}_{x \sim p_{data}} \Phi_w(x) - \mathbb{E}_{x \sim p_\theta} \Phi_w(x) \|_q \\ &= \max_{w \in \Omega} \|\mu_w(p_{data}) - \mu_w(p_\theta)\|_q \end{aligned} \tag{11}$$

In terms of WGAN, WGAN uses IPM metrics under 1-Lipschitz function class and it imposes this condition by weight clipping. It means that even maximum of $v$ is clamped so $p$ norm in equation 11 is infinite norm as $\|v\|_\infty = max_i |v_i|$. Since $p$ is infinite, dual norm $q$ is 1 by $\frac{1}{p} + \frac{1}{q} = 1$ constraint and WGAN can be interpreted as $l_1$ mean feature matching problem. McGAN [12] more extends this concept to match not only $l_q$ mean feature but also second order moment by using singular value decomposition concept which to also maximize the covariance embedding discrepancy between $p_{data}(x)$ and $p_\theta(x)$. In addition to that, Geometric GAN [13] is inspired from McGAN in that McGAN framework is equivalent to Support Vector Machine(SVM) [14] which separates hyperplane that maximize margin between two distributions. It details adversarial training in frame that the discriminator is updated away from separating hyperplane and the generator is updated toward separating hyperplane. However, such high-order momentum matching requires hard matrix computation. Maximum Mean Discrepancy(MMD) tackles those problem with kernel trick which induces an approximation of high order moments and it can also be analyzed in feature matching framework. Before investigation, we first define some mathematical definitions of Hilbert space and kernel.

#### 2.1.2.3 Maximum Mean Discrepancy

Hilbert space $\mathcal{H}$ is a vector space where inner product is defined and function space whose functions $f : \mathcal{X} \to R$ is also Hilbert space since it defines norm and inner product on its own. Kernel $k$ is defined as $k : \mathcal{X} \times \mathcal{X} \to R$ such that $k(\cdot, x) \in H, \forall x \in \mathcal{X}$. Lastly, Reproducing Kernel Hilbert Space(RKHS) $\mathcal{H}_k$ is a Hilbert space $\mathcal{H}$ with a kernel $k$ which a kernel $k$ satisfies reproducing property which is defined by $< f, k(\cdot, x) >_{\mathcal{H}} = f(x), \forall x \in \mathcal{X}$. By Riesz Representation Theorem [15], we can induce RKHS with indefinite dimensional feature mapping $\Phi : \mathcal{X} \to \mathcal{H}_k$ such that $k(x, y) =< \Phi(x), \Phi(y) >_{\mathcal{H}_k}$ which satisfies reproducing property from positive semidefinite kernel $k$. To summarize, a function $f \in \mathcal{H}_k$ can be evaluated by inner product with $k(\cdot, x) = \Phi(x)$. Therefore we can redefine equation 8 with a function class $\mathcal{F} = \{f | \|f\|_{\mathcal{H}_k} \leq 1\}$ and derive another mean feature matching with reproducing property $f(x) =< f, \Phi(x) >_{\mathcal{H}_k}$ as follows where $\mu(\mathcal{P}) = \mathbb{E}_{x \sim \mathcal{P}} \Phi(x)$ denotes kernel mean embedding.

$$\begin{aligned} d_{\mathcal{F}}(p_{data}, p_\theta) &= \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \mathbb{E}_{x \sim p_{data}} f(x) - \mathbb{E}_{x \sim p_\theta} f(x) \\ &= \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \{< f, \mathbb{E}_{x \sim p_{data}} \Phi(x) - \mathbb{E}_{x \sim p_\theta} \Phi(x) >_{\mathcal{H}_k}\} \\ &= \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \|\mu(p_{data}) - \mu(p_\theta)\|^2_{\mathcal{H}_k} \end{aligned} \tag{12}$$

From equation 12, $d_{\mathcal{F}}$ is defined as the maximum kernel mean discrepancy in RKHS. This method is widely accepted in statistics known as a two sample test. Given $p_{data}$ with mean $\mu_{p_{data}}$, $p_\theta$ with mean $\mu_{p_\theta}$ and a kernel $k$, the square of MMD distance $M_k(p_{data}, p_\theta)$ is defined as follows.

$$M_k(p_{data}, p_\theta) = \|\mu_{p_{data}} - \mu_{p_\theta}\|^2_{\mathcal{H}_k} = \mathbb{E}_{p_{data}}[k(x, x')] - 2\,\mathbb{E}_{p_{data}, p_\theta}[k(x, y)] + \mathbb{E}_{p_\theta}[k(y, y')] \tag{13}$$

GMMN [16] suggests minimize directly MMD distance with fixed Gaussian kernel $k(x, x') = exp(-\|x - x'\|^2)$ by optimizing $\min_\theta M_k(p_{data}, p_\theta)$. It looks quite dissimilar with classical GAN

because there is no discriminator which estimates the discrepancy between two distributions. MMDGAN [17] suggests adversarial kernel learning which replaces fixed Gaussian kernel to composition kernel of Gaussian kernel and injective functions $\tilde{k}(x, x') = exp(-\|f_\phi(x) - f_\phi(x')\|^2)$. So objective function with optimizing kernel $k$ is $\min_\theta \max_\phi M_{\tilde{k}}(p_{data}, p_\theta)$ that looks similar with original GAN objective as equation 2. It should be noticed that since $f_\phi$ is modeled with neural network, auto-encoder is adopted to enforce $f_\phi$ to be injective function as $f_{decoder} \approx f^{-1}$.

Like other IPM metrics, MMD distance is continuous and differentiable almost everywhere in $\theta$. It can also be understood under IPM framework with function class $\mathcal{F} = \mathcal{H}_K$ as equation 7 where its inner product is defined in RKHS. By introducing an RKHS with a kernel $k$, it has a gain over other feature matching metrics in that a 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 linear kernel with output dimension of 1 instead of Gaussian kernel. However, moment matching techniques using Gaussian kernel takes advantage over WGAN in that it can match even infinite order of moments since exponential form can be represented into infinite order via Taylor expansion while WGAN can be treated as first-order moment matching problem as discussed above. However, great disadvantage of measuring MMD distance is that computational cost grows quadratically as the number of samples grows [6].

Meanwhile, CramerGAN [18] criticizes that Wasserstein distance incurs biased gradient so suggests energy distance between two distributions. In fact, it measures energy distance not directly in the data manifold but with transformation function $h$. But it can be thought as the distance in kernel embedded space as MMDGAN impose transformation function to be injective by additional auto-encoder reconstruction loss.

#### 2.1.2.4 Fisher GAN

Instead of using standard IPM in equation 5, FisherGAN [19] focuses on standardized mean discrepancy which naturally induces a data dependent constraint as gradient penalty by following equation.

$$\begin{aligned} d_{\mathcal{F}}(p_{data}, p_\theta) &= \sup_{f \in \mathcal{F}} \frac{\mathbb{E}_{x \sim p_{data}} f(x) - \mathbb{E}_{x \sim p_\theta} f(x)}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_{data}} f^2(x) + \mathbb{E}_{x \sim p_\theta} f^2(x))}} \\ &= \sup_{f \in \mathcal{F}, \frac{1}{2}\mathbb{E}_{x \sim p_{data}} f^2(x) + \mathbb{E}_{x \sim p_\theta} f^2(x) = 1} \mathbb{E}_{x \sim p_{data}} f(x) - \mathbb{E}_{x \sim p_\theta} f(x) \end{aligned} \tag{14}$$

Equation 14 is motivated from Fisher Linear Discriminant Analysis(FLDA) in that it not only maximizes mean differences, but also reduces total with-in class variance from two distributions. Last line of equation 14 follows from constraining numerator of first line of equation 14 to 1. It is also, as other IPM metrics, concluded to mean feature matching problem as equation (9) but with little different constraint. With definition in equation 7, [19] derives another mean feature matching with second order moments constraint where $\mu_w(\mathcal{P}) = \mathbb{E}_{x \sim \mathcal{P}} \Phi_w(x)$ is a embedding mean and $\sum_w(\mathcal{P}) = \mathbb{E}_{x \sim \mathcal{P}} \Phi_w(x)\Phi_w^T(x)$ is a embedding covariance.

$$\begin{aligned} d_{\mathcal{F}}(p_{data}, p_\theta) &= \max_{w \in \Omega} \max_v \frac{< v, \mathbb{E}_{x \sim p_{data}} \Phi(x) - \mathbb{E}_{x \sim p_\theta} \Phi(x) >}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_{data}} f^2(x) + \mathbb{E}_{x \sim p_\theta} f^2(x))}} \\ &= \max_{w \in \Omega} \max_v \frac{< v, \mathbb{E}_{x \sim p_{data}} \Phi(x) - \mathbb{E}_{x \sim p_\theta} \Phi(x) >}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_{data}} v^T\Phi(x)\Phi(x)^T v + \mathbb{E}_{x \sim p_\theta} v^T\Phi(x)\Phi(x)^T v)}} \\ &= \max_{w \in \Omega} \max_v \frac{< v, \mu_w(p_{data}) - \mu_w(p_\theta) >}{\sqrt{v^T(\frac{1}{2}\sum_w(p_{data}) + \frac{1}{2}\sum_w(p_\theta) + \gamma I_m)v}} \\ &= \max_{w \in \Omega} \max_{v^T(\frac{1}{2}\sum_w(p_{data}) + \frac{1}{2}\sum_w(p_\theta) + \gamma I_m)v = 1} < v, \mu_w(p_{data}) - \mu_w(p_\theta) > \end{aligned} \tag{15}$$

Second line of equation 15 can be induced using inner product that $f$ is defined. $\gamma I_m$ in third line is $m$ by $m$ identity matrix which to guarantee numerator not to be zero. As we see the last line of equation 15, FisherGAN aims to find embedding direction $v$ which maximizes mean discrepancy while constraining it to lie in hyper-ellipsoid. It naturally derives Mahalanobis distance which is defined as distance between two distributions given positive definite matrix such as covariance

matrix of each class. More importantly, FisherGAN has advantages over WGAN in that it does not impose a data independent constraint such as weight clipping which makes training too sensitive on clipping value and also has computational benefit over gradient penalty method since it has to compute gradients of critic while Fisher method have only to compute covariance.

#### 2.1.2.5 Comparison to f-divergence

f-divergence family which can be defined from equation 3 with convex function $f$ has restrictions which as the dimension of the data $x \in \mathcal{X} = R^d$ increases, estimation of f-divergence becomes more not to be tractable because of the integral term for each dimension and the supports of two distributions tends not to be aligned which leads to infinity. Even though equation 4 derives variational lower bound of equation 4 which looks very similar to equation 5, tight bound to the true divergence is not guaranteed in practice so incurs incorrect, biased estimation.

Actually, [20] shows that only non-trivial intersection between f-divergence family and IPM family is total variation distance so therefore IPM family does not inherit disadvantages of f-divergence and totally different from it. Estimating IPM is independent from the data dimension since it directly maximizes mean discrepancy of the critic and [20] also proves that IPM estimators using finite i.i.d samples are consistent to the convergence while convergence of f-divergence is highly dependent on data distributions.

Consequently, employing IPM family to measure distance between two distributions takes advantage over using f-divergence family in that IPM family are not affected by data dimension and consistently converge to true distance. Moreover, they does not diverge even though the supports of two distribution are disjoint. In addition to that, [19] also expresses that FisherGAN is equivalent to Chi-squared distance which can be covered by f-divergence framework. But with data dependent constraint, Chi-squared distance can take IPM family characteristic so more robust to unstable training of f-divergence estimation.

### 2.1.3 Auto encoder reconstruction loss

Auto-encoder is widely adopted for GAN architecture to encourage generator to represent better all the real samples using reconstruction loss. Especially, using an auto-encoder as the discriminator, a following table 2 represents object functions of auto-encoder based GAN with pixel-wise loss $L : R^{N_x} \rightarrow R^+$ which is a natural choice for auto-encoder reconstruction. Pixel-wise loss $L$ is denoted as $D$ which is the discriminator as above, to prevent confusion with the generator loss $L_G$ and the discriminator loss $L_D$.

$$D(v) = |v - AE(v)|, \, where \, AE : R^{N_x} \Rightarrow R^{N_x}, \, v \in R^{N_x}, \, [t]^+ = \max(0, t) \tag{16}$$

| GAN | Objective function | Details |
|---|---|---|
| BEGAN | $L_D = D(x) - k_t D(G(z))$<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 | $L_D = D(x) + [m - D(G(z))]^+$<br>$L_G(z) = D(G(z))$ | Total Variance$(p_{data}, p_\theta)$ |
| MAGAN | $L_D = D(x) + [m - D(G(z))]^+$<br>$L_G(z) = D(G(z))$ | Margin $m$ is adjusted in EBGAN's training |

Table 2: Objective functions of auto-encoder based GAN

BEGAN [21] uses the fact pixel-wise loss in equation 16 follows normal distribution by Central Limit Theorem. It focuses on matching loss distributions through Wasserstein distance, not on matching data distributions. In BEGAN, discriminator has two role which one is to auto-encode well real samples and other is to balance generator and discriminator via equilibrium hyper-parameter $\gamma = \mathbb{E}[L(G(z))] / \mathbb{E}[L(x)]$. $\gamma$ is fed into objective function to prevent discriminator from easily winning generator so that balances the power of each component.

EBGAN [5] interprets the discriminator as an energy agent, which assigns low energy to real samples and assigns high energy to generated samples. Through $[m - L(G(z))]^+$ term in

objective function, the discriminator ignores generated samples with higher energy than $m$ so that the generator tries to synthesize samples which has lower energy than $m$ to fool discriminator and that mechanism much stabilizes training. MAGAN [22] takes same approach with EBGAN. Only difference is that it does not fix margin $m$. It shows empirically that energy of generated sample fluctuates near margin and that phenomena with fixed margin makes harsh to adapt to the changing dynamics of the discriminator and generator. MAGAN suggests that margin $m$ is adapted to expected energy of real data, so $m$ monotonically reduces as discriminator auto-encode real samples better.

Since total variance distance belongs to 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, it inherits some disadvantages of estimating f-divergence as [6], [20] represent.

## 2.2 Architectures

Architecture of generator and discriminator is important since it massively influences to stability of training GAN and the performance. Various papers adopt several techniques such as batch normalization, stacked architecture or multiple generator/discriminator models.

LAPGAN [23] generates image using Laplacian pyramid which lower level generated image conditioned on higher level of pyramid after up-sampled. Importantly, DCGAN [24] gives an enormous contribution in that its suggested Convolution Neural Networks(CNN) architecture greatly stabilize GAN training. Architecture of DCGAN is in figure 3. DCGAN [24] suggests architecture guideline such as batch normalization, kind of activation functions. Since it solves the instability of training GAN through only architecture, DCGAN becomes a baseline for modeling various GAN proposed later. For example, GRAN [25] uses Recurrent Neural Network(RNN) to generate images motivated by DCGAN. By accumulating images of each time step DCGAN's output, it aim to produce higher visual quality images by piling up several time step images.

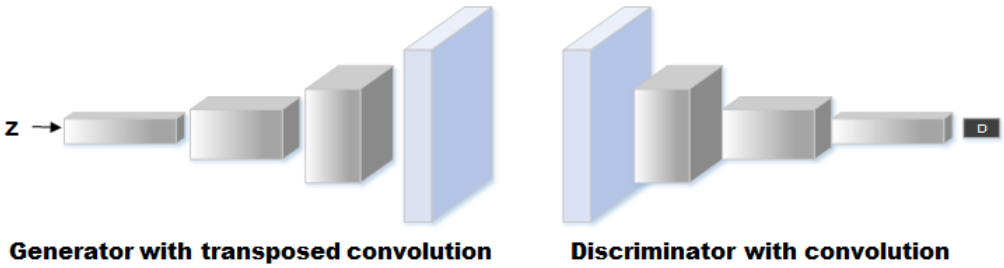



Figure 3: DCGAN architecture

### 2.2.1 GANs using multiple components

There are various proposed architectures which adopts multiple generators and discriminators. For instance, MAD-GAN tries to address mode collapse issue by having multiple generators to learn each different modes as mentioned above. Other papers proposing to use multiple components attempts to decompose hard problem of learning image distribution directly into hierarchical problem which former layer information helps to learn later layer generation.

Stacked GAN [26] aims to learn hierarchical representation by stacking several GAN. For each layer of generator stack, there are the generator which produces level-specific representation, the corresponding discriminator to train generator adversarial at each level and an encoder which generates semantic features of real samples. Its objective function has 2 more loss term in addition to an adversarial objective function. First one is a feature similarity loss which enforces to capture higher level real semantic features from next level encoder. The other one an entropy loss which encourages the generator to produce diverse representation conditioned on higher level real representation. By adopting hierarchy between GAN, it aims to produce plausible feature representations given higher level representation.

GoGAN [27] proposes to improve WGAN by adopting multiple WGAN pairs. For each stage, it changes Wasserstein loss to margin based one as EBGAN like $[D(G(z)) - D(x) + m]^+$ so that

makes the discriminator focus on generated samples whose gap $D(x) - D(G(z))$ is smaller than $m$. In addition to that, GoGAN adopts ranking loss for adjacent stages which induces the generator in later stage to produce better than former generator by using smaller margin at next stage Wasserstein loss. By progressively moving stages, GoGAN aims to reduce gap between $p_{data}$ and $p_\theta$ gradually.

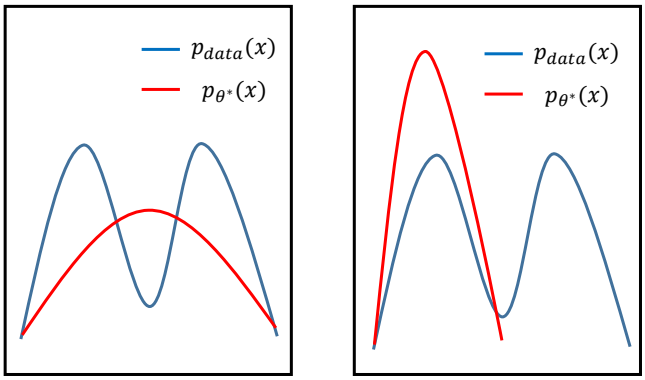
In particular, LR-GAN [28] adopts one discriminator for two generators. LR-GAN generates image in recursive fashion in that its background generator produces background of image and foreground generator output an object which is pasted to the generated background. It unfolds generation process into several time step. The first step generates the background, and the second step generates foreground which is composed of an object and a mask of it and composite it to the background by element-wise multiplication with mask. Lastly, it keeps to produce appropriate transformed foreground given composited image from last time step. It is noticeable that it naturally partitions background and foreground from independent prior $z$ so that composition to different background with transformed object become more tractable.

## 2.3 Obstacles for training GAN

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

Table 3: Comparison between KLD and JSD

As discussed in section 1, traditional generative model approach is minimizing $KL(p_{data}||p_\theta)$. Since KLD is not symmetry, minimizing $KL(p_\theta||p_{data})$ behaves different. Figure 4 from [1] shows details different behavior of asymmetric KLD where Figure 4a shows minimizing $KL(p_{data}||p_\theta)$ and figure 4b shows minimizing $KL(p_\theta||p_{data})$ given mixture of two Gaussian distributions $p_{data}(x)$ with single Gaussian distribution $p_\theta(x)$. $\theta^\star$ in each figure denotes argument minimum of each asymmetric KLD. For figure 4a, if $p_{data}(x) > p_\theta(x)$, mode dropping occurs since distribution of $G$ does not cover some parts of data distribution and high cost would be derived. Conversely, if $p_{data}(x) < p_\theta(x)$, $G$ generates samples that does not look sharp and outputs low cost. Therefore, $p_{\theta^*}(x)$ in figure 4a is averaged out for all modes of $p_{data}(x)$ since $KL(p_{data}||p_\theta)$ is more focusing on covering all parts of $p_{data}(x)$. In contrast to that, $KL(p_\theta||p_{data})$ has opposite effect which derives high cost to non realistic generated samples in case of $p_{data}(x) > p_\theta(x)$. That's why $p_{\theta^*}(x)$ in figure 4b seeks to find $x$ which is highly likely from $p_{data}(x)$.




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

Figure 4: Different behavior of asymmetric KLD

It looks like that Jenson-Shannon Divergence(JSD) has an advantage over those two asymmetric KLD in that it cares both mode dropping and sharpness. Also it never explodes to infinity like KLD even though there exists a point $x$ such that lies outside $p_\theta$'s support which makes $p_\theta(x)$

0 as seen in table 3. [2] shows that $G$ aims to minimize JSD between $p_{data}(x)$ and $p_\theta(x)$ and optimal discriminator $D^\star$ for the fixed generator, $V(G, D^\star)$ is equal to $2JSD(p_{data}||p_\theta) - 2\log 2$. Concretely, we hope that train D enough so that it approximates $JSD(p_{data}||p_\theta) - 2\log 2$ and then minimize G. However, [29] reveals mathematically why approximating $V(G, D^\star)$ does not work well in practice.

[29] empirically and theoretically proves that why training of GAN is unstable fundamentally. When optimizing $D$ with fixed G, $V(G, D)$ tends to go 0 not approximating theoretical $V(G, D^\star)$. It means that $JSD(p_{data}||p_\theta)$ is maxed out to $\log 2$ so $V(G, D^\star)$ goes to 0. It happens when the supports of the distributions are disjoint or lie on low dimensional manifolds. For both cases, there exists perfect discriminator which classifies perfectly so its gradient is 0 at supports of the distributions. It is proved empirically and mathematically that $p_{data}(x)$ and $p_\theta(x)$ came from $z$ have low dimensional manifold in practice [30] and this fact let the discriminator's gradient transferred to generator tends to vanish as discriminator becomes perfectly classifying real or fake data. Moreover, even with alternate $-\log D(G(z))$ objective, minimizing loss function is equivalent to simultaneously trying to minimize $KL(p_\theta||p_{data})$ and maximizing $JSD(p_\theta||p_{data})$. As those two objectives are counter-interactive, this leads the magnitude and variance of gradient to increase as training is in progress and this makes training unstable and hard to converge to equilibrium. Such discriminator's vanishing(or exploding) gradient hinders generator from learning enough from gradient feedback because update of $G$ is incurred through composition of $D(G(z))$. To summarize, training of GAN is theoretically guaranteed to converge if we get optimal discriminator $D^\star$ which approximates JS divergence but theoretical result is not guaranteed in practice. Besides discriminator's improper gradient problem discussed in above paragraph, there are 2 more practical issues why GAN training suffers from non-convergence.

First of all, 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 MLP 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 do an inference, rather it can generate samples following $p_\theta$(x) through a simple feed-forward. But this practical implementation derives a gap with theory. [2] provides 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 network, the convexity does not hold because we now optimize in parameter space rather than in function space where assumed theoretical analysis lies on. Therefore, theoretical guarantees do not hold anymore in practice. Further issue related to parameterized neural network space, [31] discusses the existence of equilibrium of GAN. It shows that large capacity of the discriminator does not the guarantee the generator to generalize all real data and this means that there does not exist equilibrium under a certain finite capacity of the discriminator(Capacity means the number of training variables). Second problem comes from iterative update algorithm suggested in [2]. Moreover we want to train discriminator until optimal for fixed generator but optimizing $D$ such that way is highly computational expensive. Naturally, we have to train $D$ for just certain $k$ steps and that scheme arises confusion that whether it is solving minimax problem or maxmin problem because $D$ and $G$ are updated simultaneously by gradient descent in iterative procedure. Unfortunately, solutions of minimax and maxmin problem are not equal.

$$\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D) \tag{17}$$

With maxmin problem, minimizing $G$ lies in inner loop in maxmin problem. G now is forced to place its probability mass on the most likely point where the fixed non-optimal D believes likely to be real rather than fake. After $D$ is updated to reject generated fake one, $G$ attempts to move probability mass to other most likely point for fixed $D$. In practice, real data distribution is a normally multi modal distribution but such maxmin training procedure consequences generator does not cover all modes of real data distribution since the generator considers itself that it fools the discriminator enough by only picking up one mode. Actually, the generator covers only single mode or few modes of real data distribution. This undesirable non-convergent situation is called mode collapse. Mode collapse is that many modes in real data distribution are not all represented in the generated samples so results in the lack of diversity of generated samples. It can be simply thought as generator is trained to be non one-to-one function which produces single output value for several input values.

In addition to that, the problem of the existence of perfect discriminator we discussed above paragraph can be connected to mode collapse. Since supports of two distributions are disjoint

or have low dimensional manifold in practice, the discriminator gets better so that it tends to output to be close to 1 for real samples while it produces near 0 for fake samples. Because the discriminator produces values near 1 for all possible modes, there is no reason for the generator to represent all modes in data manifold. Theoretical and practical issues discussed in this section can be summarized as follows.

- Since supports of distributions lie on low dimensional manifolds, there exists perfect discriminator whose gradients vanish on data point. Generator could have trouble being optimized since it is not provided any information from discriminator.
- GAN training optimizes generator for fixed discriminator and discriminator for fixed generator simultaneously in one loop, it sometimes behaves as solving maxmin problem not minimax problem. It critically causes mode collapse. Also generator and discriminator optimize same objective function $V(G, D)$ in opposite directions which not usual in classical machine learning, it suffers from oscillation which makes training take long time.
- Theoretical convergence proof do not apply in practice because generator and discriminator is modeled with deep neural networks, so optimization has to be made in parameter space rather than probability density function itself.

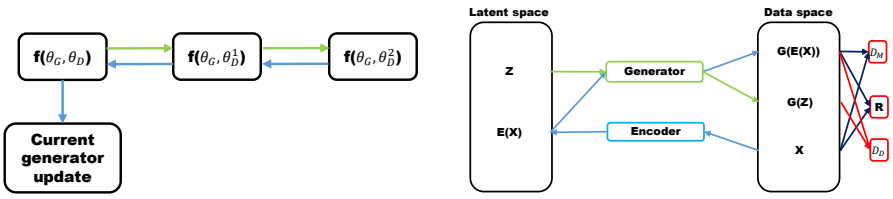
## 2.4 Methods to address mode collapse

Since mode collapse yields the limit of application of GAN so many researchers suggest new techniques and architecture to address this issue. [32] suggests several methods to improve training of GAN.

- Feature matching:
  This trick do not focus on discriminator‘s output, rather substitute it with an activation on an intermediate layer of the discriminator to prevent over-fitting from current discriminator. Generator is then trying to catch discriminative features of real data and fake data.
- Mini-batch discrimination:
  This approach is letting discriminator look at multiple examples in combination to avoid mode collapse of generator. To make discriminator process each example with correlation of the other examples, it models mini-batch layer in intermediate layer of discriminator. So discriminator can use the other examples in mini-batch while maintaining a natural role of it.
- Label smoothing:
  As mentioned above, $V(G, D)$ is a binary cross entropy loss which real data label is 1 and generated data label is 0. Label smoothing is to assign values lower than 1 such as 0.9 to real data label and values higher than 0 such as 0.1 to generated data label. By doing that, we can let discriminator prevent from passing large gradient signal to generator.

Unrolled GAN [33] manages mode collapse issue with surrogate objective function for update of the generator, which helps the generator to predict discriminator‘s response by unrolling discriminator update $k$ steps for current generator update. As we see in [2], GAN updates the discriminator first for fixed generator and updates the generator for updated and fixed discriminator. Unrolled GAN differs with [2] in that it updates generator based on $k$ steps updated discriminator given current generator update so that it hopes to capture how the discriminator responds on current generator update. Figure 5a represents simple outline of Unrolled GAN where green arrow stands for forward pass and blue arrow represents the generator update flow. We see that when the generator is updated, it unrolls the discriminator's update step to consider discriminator's $k$ steps future response with respect to generator's current update while updating the discriminator in the same way as before. Since the generator is given more information about discriminator's response, generator spreads out its probability mass to make discriminator more hard to react on the generator's behavior. It can be seen as empowering generator because the only generator's update is unrolled, but it seems quite fair in aspect that the discriminator can not be trained to optimal in practice due to infeasible computational cost while the generator is assumed to get information from the optimal discriminator theoretically.

MRGAN [34] focuses that mode collapse happens because the generator is not penalized for missing modes. To address mode collapse in this point of view, it adds 2 regularizing term for missing modes to objective function. One is a geometric regularizer to jointly train an encoder $E(x) : X \to Z$ with regularizing term $\mathbb{E}_{x \sim p_{data}}[d(x, G \circ E(x))]$, where $d$ is a $L^2$ metric in the data space. By minimizing $L^2$ distance in data manifold, it attempts to match the data manifold and generation manifold. The other one is a mode regularizer which enforces $G \circ E(x)$ to visit every modes of training samples even to minor modes in assumption of $G \circ E(x)$ is a good auto-encoder. In manifold matching step, the discriminator distinguishes real data and its reconstruction so that the generator learns how to reconstruct and in distribution diffusion step, another discriminator classifies real data and generated data so that the generator learn to produce real-like samples. By matching manifolds of real data and generated data so as to make discriminator pass smooth gradient information to generator and distributes probability mass across different modes fairly by diffusion step. Figure 5b shows outline of MRGAN where $D_M$ denotes the discriminator which distinguishes data and reconstructed one in manifold matching step, $D_D$ denotes the other discriminator that distinguishes whether an image is real or fake in distribution diffusion step and $R$ denotes a geometric regularizer.




(a) Unrolled GAN (b) Mode Regularized GAN

Figure 5: Unrolled GAN and Mode Regularized GAN

VEEGAN [35] takes similar approach 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. Advantage of noise auto-encoding is that choosing $l_2$ reconstruction loss for latent space is more robust to blurry image than pixel-wise $l_2$ reconstruction in data space and is more natural choice for standard normal distribution $z$. Such noise auto-encoding scheme will be detailed in section 3.2.1.2

DRAGAN [36] points out that mode collapse happens due to the existence of spurious local Nash Equilibrium in non-convex problem. DRAGAN address this issue by proposing constraining the gradients of the discriminator around real data manifold. It adds gradient penalizing term which biasing the discriminator to have gradient of norm 1 around the real data manifold. This method hopes functions in training procedure to have linearity which arises from making have norm-1 gradients. Linear functions near real data manifolds form convex function space, which imposes a global unique optimum. This gradient penalty method is also applied to WGAN from different approach but in same manner and it will be discussed later.

MAD-GAN [37] adopts multiple generator for one discriminator to capture diversity of generated samples. To induce each generator to move toward to each different mode, it adopts cosine similarity value as additional objective term so as to make each generator produce dissimilar samples from other ones. This technique is inspired from the fact that as images from two different generators become similar, higher similarity value comes out, so that it aims to optimize when such case happens. Also, since each generator produce each fake samples, the discriminator's objective adopts softmax cross entropy loss to distinguish real samples from fake samples generated by multiple generators.

# 3 Treating the latent space

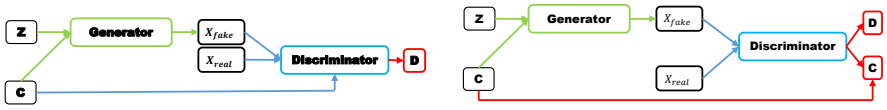
Latent space, also called as embedding space when using auto-encoder, is the space where a compressed representation of data lies in. In VAE, an encoder maps data point $x$ into latent space $z$ and a decoder reconstructs real data while GAN directly maps a Gaussian or uniform latent vector $z$ into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

## 3.1 Latent space decompose

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

### 3.1.1 Supervised

Conditional GAN(CGAN) [38] proposes a method of conditioning additional information such as class label with a noise vector $z$ to control 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 noise vector $z$ but also additional information vector $c$ and the discriminator also takes samples as well as condition information vector $c$ so that it distinguishes conditioned fake samples given $c$. AC-GAN [39] takes little different approach with CGAN in that it adds an auxiliary label classifying loss term to adversarial loss term so as to maximize the log-likelihood of class label of generated samples. Therefore, the discriminator produces not only samples' real or fake probability but also probability over the class label. Figure 6 shows outlines of CGAN and ACGAN where C in red rectangle in figure 6b denotes 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: Supervised latent decompose

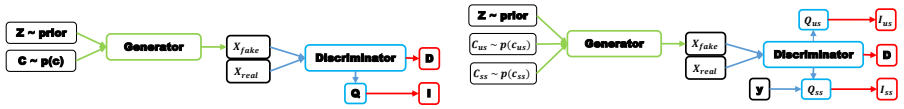
### 3.1.2 Unsupervised

InfoGAN [40] decompose the input noise vector into a standard incompressible noise 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 representation. Since the latent code $c$ is to capture some noticeable features of real data, it is quite convincing that its objective function aim to maximize the amount of information of the latent code $c$ when a generated sample $G(z, c)$ is observed. However, since mutual information $I(c; G(z, c))$ needs to directly access to posterior probability $p(c|x)$, so InfoGAN takes variational approach which approximates a target $I(c; G(z, c))$ by maximizing a lower bound with auxiliary Q networks estimating posterior $p(c|x)$.

Both CGAN and InfoGAN learn conditional probability $p(x|c)$ given a certain condition $c$, but they are dissimilar in regard how to handle condition $c$. In CGAN, additional information $c$ is assumed to semantically known such as class labels, so during training we have to supply it to the generator and the discriminator. Meanwhile, $c$ in InfoGAN is assumed to be unknown and we want to find posterior $p(c|x)$, so we put a prior for $c$ by sampling from $p(c)$ and infer it based on posterior probability $p(c|x)$. As a result, 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(ssInfoGAN) [41] takes both advantages of supervised and unsupervised methods. It improves unsupervised manner of InfoGAN by introducing few labels as semi-supervised manner by decomposing the latent code $c$ again in two parts. As InfoGAN, it still tries to learn semantic representation of unlabeled data by maximizing mutual information between synthetic data and the unsupervised latent code $c_{us}$. In addition to that, ss-InfoGAN adds additional semi-supervised latent code $c_{ss}$ which accomplishes $c = c_{ss} \cup c_{us}$ and maximizes two mutual information terms. By maximizing mutual information between labeled real data and

$c_{ss}$, guides $c_{ss}$ to encode the label information by interpreting label $y$ as $c_{ss}$ and maximizes another mutual information between synthetic data and $c_{ss}$ to pass encoded label information to the generator. By combining supervised and unsupervised way, ss-InfoGAN aim to the latent code representation more easily than fully unsupervised way like InfoGAN with a small subset of labeled data. Figure 7 represents flow of InfoGAN and ssInfoGAN where Q in blue rectangle represents auxiliary parametrized network to approximate posterior $p(c|x)$ and I in red rectangle stands for mutual information to be maximized. It should be noted that two mutual information term for $c_{ss}$ is 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 implementation.




(a) InfoGAN (b) ssInfoGAN

Figure 7: Unsupervised latent decompose

## 3.2 With an Auto-Encoder

### 3.2.1 Learn latent space using an encoder

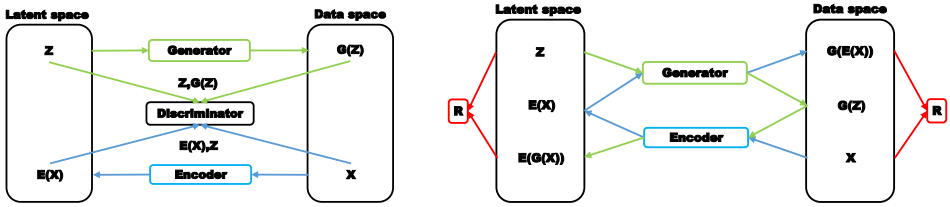
In this section, we look into efforts on how to combine auto-encoder structure into GAN framework. An encoder-decoder structure which an encoder encodes data $x$ into latent variable $z$ and a decoder reconstruct to data $x$ is suitable for stabilizing GAN training since it learns the posterior distribution $p(z|x)$ to reconstruct data $x$ so that alleviates mode collapse while GAN lacks of inference ability which maps from data $x$ into $z$. By learning latent representation of complex, high-dimensional data space with an encoder $X \rightarrow Z$, manipulations at the abstract level become possible and this encourages complex modification in data space to be more easily conducted at the latent space by doing such as latent interpolation or conditional concatenation on latent variable.

ALI [42], BiGAN [43] proposes learning latent representation within GAN framework. As we see in figure 8a, they aim to learn joint probability distribution of data $x$ and latent $z$ while GAN aims to learn directly only data distribution. The discriminator acts on joint space of latent space and data space and discriminates $(G(z), z)$ and $(x, E(x))$ joint pairs where $G$ and $E$ represents decoder and encoder respectively. By training encoder and decoder together, they hope to learn inference $X \rightarrow Z$ while still generating high-quality samples.

AGE [44] only focuses on the generator $G$ and the encoder $E$. Those two parametrized network are trained in the latent space and the data space via imposing reconstruction losses in each space with minimizing divergence in the latent space. The generator and the encoder are trained adversarial as that the generator aims to minimize divergence between $E(G(z))$ and prior $z$ in the latent space while the encoder tries to match $E(x)$ and $z$ to align encoded data with prior $Z$ and not to match $E(G(z))$ and $z$ by maximizing its divergence. Also it adds reconstruction errors in each space to guarantee each component to be reciprocal and this term can be interpreted as imposing each function to be one to one mapping function not to fall into mode collapse. In fact, it is a quite similar motivation with a geometric regularizer of MRGAN [34] in that it aim to incorporate a supervised training signal which guides the reconstruction process to the right place and this will be introduced again in section 4.1.1.2. Figure 8b shows outline of AGE where 'R' in red rectangle denotes reconstruction loss between original and reconstructed ones.

#### 3.2.1.1 Discrete structured data

Applying GAN to generate discrete structured data such as text sequence or discretized images is an another issue is contrast with continuous data since gradient descent update via back-propagation can not directly be applied for discrete output which is non-differentiable. Some methods adopt policy gradient [45] in Reinforcement Learning(RL) whose object is to maximize total reward for rolling out whole sequence so that circumvents direct back-propagation for discrete output. MaliGAN [46] and BSGAN [47] takes similar approach in that they are both inspired from $p_{data}(x) = \frac{D^*(x)}{1-D^*(x)} p_\theta(x)$ for optimal discriminator $D^*$ and estimate true distribution




(a) ALI, BiGAN (b) AGE

Figure 8

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 output a probability for discrete output which is analogous in that a policy in RL is a probability distribution for taking an action in given states. They both take $\tilde{p}_{data}(x)$ as a reward for discrete output $x$. Since higher $\tilde{p}_{data}(x)$ is derived from high $D(x)$ value, it seems to make sense that $\tilde{p}_{data}(x)$ value is taken as a reward. However, this policy gradient approach has a limit in that it can not access latent representation since it does not learn posterior inference from data $x$ to $z$.

ARAE [48] addresses this issue by combining discrete auto-encoder which encode discrete input into continuous code space and WGAN acting on code space where a code stands for latent representation of discrete data. To avoid directly access to discrete structure, WGAN is adopted for continuous code space where the generator produces fake code $\tilde{c}$ from sampled latent $z$ and the critic evaluates such fake code and real code $c$. By jointly training auto-encoder with WGAN on code space, it aims to dodge back-propagation while learning latent representations of discrete structured data.

#### 3.2.1.2 Sampling from latent space

Plug and Play Generative Networks(PPGN) [49] are generative models derived from sampling techniques. PPGN consists of a generator network $G$ and a condition network $C$, where $G$ functions produce data, and $C$ performs conditioning via the function of image classification. More specifically, the PPGN framework is derived from Activation Maximization. Using chain rule, 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.

$$p(x, y) = p(x)p(y|x) \tag{18}$$

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

$$p(x|y = y_c) \propto p(x)p(y = y_c|x) \tag{19}$$

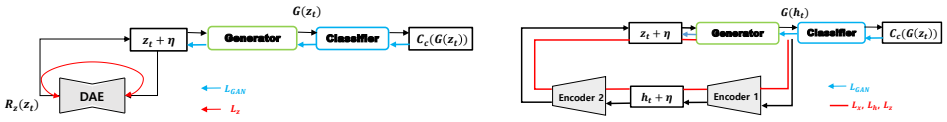
Maximizing $p(x|y = y_c)$ by the Activation Maximization method is equivalent to maximizing likelihood that $x$ stays in the manifold of the image distribution and likelihood that the generated data is in $y_c$ class. $G$ and $C$ perform two tasks respectively.

Data can be sampled directly from the sampler, but sampling in high dimensional space such as image data suffers from poor mixing. So [49] introduced latent variable $z_t$ in low dimensional latent space. PPGN samples $z_t$ and $z_t$ is used as the input of Generator to extract data. Generator is trained by feature matching loss proposed in [32], which is $l_2$ loss between the real data and the generated data in feature space. The next sample $z_{t+1}$ is selected which increases the class probability by the gradient back-propagated from the classifier output:

$$z_{t+1} = z_t + \epsilon_1(R_z(z_t) - z_t) + \epsilon_2 \frac{\partial \log C_c(G(z_t))}{\partial G(z_t)} \frac{\partial G(z_t)}{\partial z_t} \tag{20}$$

where $G$ is Generator, $R_h(\cdot)$ is the reconstruction function of the DAE in $z$ space and $C_c(\cdot)$ denotes the output unit of classifier corresponding to class $y_c$. Based on this, better performance can be obtained by adding additional constraints and noise.

Sometimes, DAE does not model well due to difficulty of unconstrained density estimation. Joint PPGN learns that latent variables become generated data and then decode them back into latent variables. In this way, Joint PPGN can sample 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, GAN loss and $L_2$ reconstruction loss for $h_t$, $z_t$, and $x_t$. It In this way, PPGN can generate high-resolution, high-quality images successfully.




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

Figure 9: PPGN

### 3.2.2 Variational Auto-Encoder

Variational Auto-Encoder(VAE) [50] is a popular generative model using auto-encoder. As seen in figure 1, VAE approximates $p_\theta(x)$ by introducing variational lower bound. With an assumption of some unobserved latent variable $z$ affecting real sample in unknown way, VAE is a kind of finding maximum 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) with a prior knowledge $z$ must be considered instead of Maximum Likelihood(ML) but in general, posteriori probability $p(z|x)$ is intractable because $p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz$ is not tractable. VAE address this issue with variational inference which induces a variational lower bound for $p_\theta(x)$. Concretely, VAE estimates posteriori probability with an assumption of a prior knowledge $p(z)$ being normal Gaussian distribution and drives approximating model $Q_\phi(z|x)$ close to real posteriori probability $p(z|x)$ by reducing $KL(Q_\phi(z|x)||p(z|x) = \int_z Q_\phi(z|x) \log \frac{Q_\phi(z|x)}{p(z|x)} dz$. More formally, we want to maximize marginal likelihood $p_\theta(x)$ for data $x$ and a variational lower bound can be derived as follows.

$$\begin{aligned}
\log p_\theta(x) &= \int_z Q_\phi(z|x) \log p_\theta(x) dz = \int_z Q_\phi(z|x) \log(\frac{p_\theta(x,z)}{p_\theta(z|x)} \frac{Q_\phi(z|x)}{Q_\phi(z|x)}) dz \\
&= \int_z Q_\phi(z|x)(\log(\frac{Q_\phi(z|x)}{p_\theta(z|x)}) + \log(\frac{p_\theta(x,z)}{Q_\phi(z|x)})) dz \\
&= KL(Q_\phi(z|x)||p_\theta(z|x)) + \mathbb{E}_{Q_\phi(z|x)}[\frac{\log p_\theta(x,z)}{\log Q_\phi(z|x)}]
\end{aligned} \tag{21}$$

Since $KL(Q_\phi(z|x)||p_\theta(z|X))$ is always semi-positive number, variational lower bound $L(\theta, \phi; x)$ can be formulated as follows.

$$\begin{aligned}
\log p_\theta(x) &\geq \mathbb{E}_{Q_\phi(z|x)}[\log p_\theta(x,z) - \log Q_\phi(z|x)] \\
&= \mathbb{E}_{Q_\phi(z|x)}[\log p_\theta(z) - \log p_\theta(x|z) - \log Q_\phi(z|x)] \\
&= -KL(Q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{Q_\phi(z|x)}[\log p_\theta(x|z)] \\
&= L(\theta, \phi; x)
\end{aligned} \tag{22}$$

Therefore, we want to maximize $L(\theta, \phi; x)$ to maximize marginal likelihood $p_\theta(x)$ and this method can be interpreted as maximizing likelihood $p_\theta(x|z)$ with reducing KL divergence so that $Q_\phi(z|x)$ approximates a prior $p_\theta(z)$. Also $p_\theta(x|z)$ 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 $x$ with assuming $Q_\phi(z|x)$ follows conditional Gaussian distribution $\mathcal{N}(z|\mu(x), \sum(x))$ and imposing over $p_\theta(z) \sim \mathcal{N}(0, I)$. That's why this model is named as Variational Auto-Encoder and variational term can be thought that the latent $z$ is not fixed for the same input data $x$ since $z$ is sampled from a specific kind of variational distribution but not a constant value.

#### 3.2.2.1 Comparison between VAE and GAN

To deal with intractable marginal likelihood, VAE tries to approximate it via introducing lower bound with a further assumption over latent variable distribution. VAE generates real-like images through a 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 divergence between $Q_\phi(z|x)$ and $p_\theta(z)$. A tight bound can be accomplished through jointly optimizing variational parameter $\phi$ and decoder parameter $\theta$ via stochastic gradient descent. VAE can be thought as transforming intractable marginal likelihood into expectation over a variational distribution $Q_\phi(z|x)$ through a encoder and a decoder with latent variable distributional assumption. Especially, since we assume $Q_\phi(z|x)$ to be conditional Gaussian given $x$, pixel values in the reconstructed image are modeled also as Gaussian and this leads maximizing likelihood to minimizing Euclidean loss as $\log p_\theta(x|z) = -\|x - Decoder(z)\|^2$. This can be interpreted as regression problem fitting to mean and this causes VAE to generate blurry images. Moreover, VAE can not be trained via standard back-propagation. Since the model has stochasticity incurred from that $z$ is sampled from $Q_\Phi(z|x)$ when calculating $\mathbb{E}_{Q_\Phi(z|x)}[\log p_\theta(x|z)]$ term of equation 22, $\max_{\theta,\phi} L(\theta, \phi; x)$ directly does not work so that this term is not differentiable while GAN can be trained via standard back-propagation since the generator and the discriminator are assumed to be differentiable. VAE circumvents this problem via reparameterization trick [50]. Lastly, it is emphasized again that VAE tries to fit $p_{data}(x)$ through variational lower bound. However, even if that lower bound is extremely optimized, it is not guaranteed that tight bound to the $p_{data}(x)$ is achieved because simple assumption of a prior and approximated posterior distribution.

In contrast, GAN's way of solving the intractability is to introduce the discriminator which distinguishes real or fake not to approximate nor compute marginal likelihood. GAN does not require an inference of the latent $z$ through encoder, rather update 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 real samples label, $y = 0$ as generated samples, the discriminator $D_\omega$ acts a role of emitting the probability that an input sample $x$ comes from real data which can be represented as conditional probability $p(y = 1|x)$. This binary classifying role can be turned up as the density ratio $Dr(x)$ as follows.

$$Dr(x) = \frac{p_{data}(x)}{p_\theta(x)} = \frac{p(x|y=1)}{p(x|y=0)} = \frac{p(y=1|x)}{p(y=0|x)} = \frac{D_\omega(x)}{1 - D_\omega(x)} \tag{23}$$

$\frac{p(y=1|x)}{p(y=0|x)}$ in equation 23 can be derived from Bayes rule with assumption of $p(y = 1) = p(y = 0)$. Intuitively, we can reformulate equation 23 as $D_\omega(x) = \frac{p_\theta(x)}{p_\theta(x)+p_{data}(x)}$ and we can notice this is the optimal discriminator formulation as seen in [2]. Therefore, learning $Dr(x)$ is directly connected to optimize $V(G, D)$ in equation 2 and training the discriminator only by drawing samples from two distributions, GAN focuses on the marginal likelihood $p_\theta(x)$ relative behavior with respect to true data distribution $p_{data}(x)$ and the discriminator pass this information to the generator so as to induce the generator to make real-like samples. Consequently, GAN does not need to approximate or calculate distributions analytically and therefore it is classified as implicit likelihood with the discriminator in figure 1.

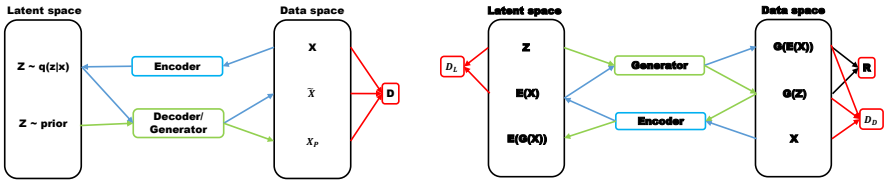
#### 3.2.2.2 Hybrid with GAN

Recently, approaches to incorporate the each advantages of VAE and GAN have been proposed. Even though VAE generates blurry images, it less suffers mode collapse problem since auto-encoder encourage all real samples it has seen. Meanwhile, GAN generates more sharp images than VAE and does not need further constraint about model, it suffers from mode collapse as mentioned in earlier section.

VAEGAN [51] combines VAE with GAN assigning GAN's generator to the decoder part. Consequently, its loss function is comprised of VAE loss function with adversarial loss term to produce sharp images while maintaining encoding ability to the latent space. Notably, it replaces $x$ of reconstruction term in equation 22 with the intermediate features of discriminator so as to capture more perceptually similarity of real samples by attempting reconstruction on feature-wise. Figure 10a shows outline of VAEGAN where the discriminator $D$ takes one real samples and two fake samples that one is sampled from encoded latent and the other is sampled from prior distribution.

$\alpha$-GAN [52] proposes adopting discriminators to the variational inference so that transforming equation 22 into more GAN-like formulation. The most unpleasant issue about VAE is we have to choose 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 like equation 24. By doing that, KL divergence 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 $p_\theta(z) = \mathcal{N}(0, I)$. $\alpha$-GAN also manages the reconstruction term in equation 22 by adopting 2 loss terms. One is adopting another discriminator which distinguishes real or synthetic samples, the other one is a normal $l_1$ pixel-wise reconstruction loss in data space. Concretely, $\alpha$-GAN change variational lower bound $L(\theta, \phi; x)$ into more GAN-like formulation by introducing two discriminators using density ratio estimation and reconstruction loss to prevent mode collapse. Figure 10b shows framework of $\alpha$-GAN where $D_L$ stands for the latent discriminator, $D_D$ represents the discriminator which acts on data space and $R$ stands for reconstruction loss.

$$KL(Q_\phi(z|x)||p_\theta(z)) = \mathbb{E}_{Q_\phi(z|x)}[\frac{Q_\phi(z|x)}{p_\theta(z)}] \approx \mathbb{E}_{Q_\phi(z|x)}[\frac{C_\phi(z)}{1 - C_\phi(z)}] \tag{24}$$




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

Figure 10: VAE hybrid with GAN

# 4 Applications using GAN

As discussed earlier sections, GAN is a really powerful generative model in that it can generate real-like samples with arbitrary noise vector $z$. We do not need to know explicit real data distribution nor assume further mathematical conditions. This advantages lead GAN to be applied to various engineering fields such as images, videos, languages and even other academic fields. In this section, we would like to deep dive into application of GAN in several domains.

## 4.1 Image

### 4.1.1 Image translation

Image translation is translating images from domain X to images from other domain Y. Mainly, translated images have dominants characteristic of domain Y maintaining attributes of before translated. As in classical machine learning, there are supervised and unsupervised ways to conduct image translation.

#### 4.1.1.1 Paired 2 domain data

Image translation with $x \in domain\, X$, $y \in domain\, Y$ paired images can be seen supervised image translating in that image in domain X has always target image in domain Y. pix2pix [53] suggests image translation method with paired images using U-NET architecture [54] which is widely used for biomedical image segmentation. It adopts conditional GAN framework in that its generator produces corresponding target image conditioned on an input image. In contrast to that, PAN [55] adds perceptual loss between a paired data $(x, y)$ to the generative adversarial loss to transform input image $x$ into ground-truth image $y$ not adopting CGAN framework. Instead of using pixel-wise loss to push generated image to 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 2 domain data

Image translation with unsupervised way is that just given unpaired 2 domain data, learns a mapping between 2 domain without supervision. CycleGAN [56], DiscoGAN [57] aim to conduct unpaired image-to-image translation using cyclic consistent loss term. With sole translator $G$: $X \rightarrow Y$, there can be lots of possible mapping from $X \rightarrow Y$ so that meaningless translation can happen or also it induces mode collapse which several images in $X$ are mapped to one image in $Y$. They adopts another inverse translator $T$: $Y \rightarrow X$ and introduces cyclic consistency loss which encourages $T(G(x)) \approx x$ and $G(T(y)) \approx y$. It can be thought as auto-encoder scheme with reconstruction loss term to impose meaningful translation to desired output. Also, as we discussed in section 3.2.1, it can be interpreted as similar manner in that it adds a supervised signal for reconstruction to be right place. Figure 11 shows baseline of unpaired image translation where $D$ denotes the discriminator in each domain and $R$ stands for reconstruction loss. In addition, DualGAN [58] also adopts cyclic reconstruction loss but use Wasserstein distance instead of sigmoid cross entropy generative adversarial loss.

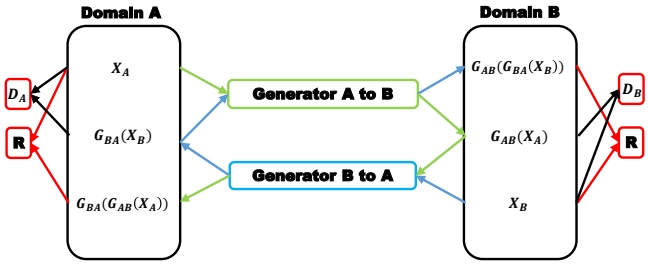



Figure 11: Unpaired image translation

DistanceGAN [59] conducts unsupervised domain mapping without circular 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 $Y$. Since it does not use cyclic consistency loss, one-sided domain mapping can be possible without inverse translator. It also suggests self-distance loss for one mini-batch sample instead of two samples distance. Lastly, SimGAN [60] tries to refine simulated synthetic image while maintaining feature characteristic through self-regularizing loss. It is trained with set of unpaired set of synthetic data and real data. Equation 25 which is reproduced from DistanceGAN [59], represents loss terms managing unpaired image translation, composed of generative adversarial loss, cyclic consistency loss, distance preserving loss. Table 4 summarizes discussed papers in terms of loss function.

$$\begin{aligned} Loss = {} & \alpha_{1x} L_{GAN}(G_{XY}, D_Y, X, Y) + \alpha_{1y} L_{GAN}(G_{YX}.D_X, Y, X) + \alpha_{2x} L_{cycle}(G_{XY}, G_{YX}, X) \\ & + \alpha_{2y} L_{cycle}(G_{YX}, G_{XY}, Y) + \alpha_{3x} L_{distance}(G_{XY}, X) + \alpha_{3y} L_{distance}(G_{YX}, Y) \end{aligned} \tag{25}$$

| 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

### 4.1.2 Using latent representation

Modifying images directly in image space is highly tough since distributions lie in image manifold are high-dimensional and complex. Rather directly manipulating in data space, editing specific attributes through latent space is more tractable in that latent representation expresses

specific features of corresponding image itself and latent manifold normally has lower dimension as we discussed in section 3. If we want to change some attributes, we can concatenate other latent information which contains target attribute or decompose latent space into identity preserving part and attribute modifying part. In this section, we look into various GAN application which aims to manage attributes of image with attribute latent vector.

#### 4.1.2.1 Object transfiguration

Object transfiguration is a kind of conditional image generation which replaces an object in an image with a particular condition while background is maintained unchanged. GeneGAN [61] adopts encoder-decoder structure to transplant 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 combined with the background feature of target image.

#### 4.1.2.2 Object detection

Detecting small object in an image typically suffers from low-resolution of an object so that it incurs to learn at different scales to detect correctly. Perceptual GAN [62] tries to leverage low-resolution of small object into super-resolved large object which to be similar with real large scale object using GAN framework. Perceptual GAN decomposes the discriminator into two branches. The generator tries to produce real-like large scale object by typical adversarial branch and the other is perceptual branch which guarantees that generated large scale object is reasonable to be used for detection.

SeGAN [63] proposes another framework to detect occluded object by other objects in an image. It suggests segmentor, generator and discriminator to extract the whole occluded object mask and to paint as if it looks real-like. The segmentor takes an image and a visible region mask of an occluded object and produces a mask of whole occluded object. The generator and the discriminator are trained adversarial to produce a segmented full painted object with invisible part conditioned on an intermediate object mask from the segmentor.

#### 4.1.2.3 Image blending

Image blending is the task which implants the object into other image's background and makes such composited copy-paste image look more realistic. GP-GAN [64] suggests high-resolution image blending framework using GAN and classical image blending gradient based approach. It decomposes images into low-resolution but well-blended images using blending GAN and detailed textures and edges using gradient constraint. Then, GP-GAN tries to combine those departed information by optimizing Gaussian-Poisson equation which generates high-resolution well-blended images while maintaining captured high-resolution details.

#### 4.1.2.4 Text to Image

StackGAN [65] synthesizes images conditioned on text description by decomposing stage 1 low-level feature generation part and stage 2 painting details part from given generated image at stage 1. Each stage generator 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 text conditioned image, which is sampling text embedding condition vector from $\mathcal{N}(\mu(\varphi_t), \sum(\varphi_t))$ rather than using fixed text embedding $\varphi_t$. It is concatenated with $z$ as conditional GAN and this technique alleviate discontinuity of the latent space arising from limited amount of image-text data. In contrast, TAC-GAN [66] is based on AC-GAN by conditioning on text description instead of class label by simply concatenating fixed text embedding and noise vector $z$. But with addition of wrong image and label data which does not correspond to text description, it tries to assign the correct text description to the generated image as AC-GAN.

#### 4.1.2.5 Editing attributes

AGEGAN [67] trains an encoder to get latent vector $z$ which represents personal identity to be preserved, and then given age vector $y$, it aims to produce face aging of target image while maintaining identity of original image. ICGAN [68] takes little different approach with AGEGAN in that it adopts two encoders, each produces latent representation and conditional attribute

representation respectively. By conditioning on target specific attribute information, it generates image with target attribute conditionally as AGEGAN does.

ViGAN [69] proposes combination of VAE and InfoGAN in that VAE generates latent representation and InfoGAN framework produces attribute conditioned image with latent variable $c$. SL-GAN [70] takes similar approach with ViGAN in that it also suggests to combine VAE and InfoGAN. But SL-GAN decompose latent space from an encoder into user-defined attribute and data-driven attribute and tries to maximize mutual information between data-driven attribute and generated image given user-defined attribute while InfoGAN part of ViGAN takes input with sampled $z$ from an encoder and known description of data $c$.

DR-GAN [71] and TP-GAN [72] aim to address challenging issues in face recognition topic. DR-GAN focuses on pose-invariant face recognition which is a tough problem because of drastic change on different poses. It also adopts encoder-decoder scheme that an encoder tries to represent identity while a decoder produces image with target pose and identity given encoded features concatenated with a pose code $c$ and $z$ which represents face rotation. DR-GAN hopes the discriminator to learn pose classification using a pose code $c$ in decoding so that the discriminator is able to detect pose-variation given same identity. Meanwhile TP-GAN proposes frontal view synthesis method from other angled profile image. It adopts two pathway generator which local pathway captures detailed texture and global pathway infers global structure of a frontal view.

Attribute guided image translation is also considered in [73], [74]. Conditional CycleGAN [73] embeds target attribute vector $z$ into CGAN framework with cyclic consistency framework. Meanwhile, [74] decomposes attribute vector and aim to only transfer target attribute while non-target attributes are kept same which is imposed by back transfer loss inspired from cyclic consistency loss in section 4.1.1.2

Synthesizing outdoor image with specific scenery attribute is quite similar with editing face attribute in that it also adopts conditional GAN framework with attribute latent code $c$. AL-CGAN [75] also takes this conditional approach that takes side information of layout and content of the scene.

### 4.1.3 Other tasks on image

#### 4.1.3.1 Super resolution

Estimating super resolution image from relative low resolution image has a fundamental problem that recovered high resolution image misses high-level texture details when upscaling image. SRGAN [76] adopts perceptual similarity loss with adversarial loss instead of optimizing pixel-wise Mean Square Error(MSE). It focuses on feature differences from intermediate layer of the discriminator, not on pixel-wise because optimizing pixel-wise MSE which tries to find the pixel-wise average of plausible solution so 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 network incur to apply GAN to generate 3D object. 3D-GAN [77] suggests 3D GAN framework inspired from [24] in that they just expand one more dimension to deal with 3 dimension. They also extend 3D framework into 3D-VAE-GAN which aims to map from 2D image into corresponding 3D object. Meanwhile, PrGAN [78] aim to infer 3D object from multiple 2D views of an object. It also uses 3D volumetric convolution network to generate 3D shape, but its discriminator takes input as 2D projected image so that projection of synthesized 3D object matches to input 2D view.

#### 4.1.3.3 Joint image generation

Coupled GAN [79] suggests method of generating multi-domain images especially by weight sharing techniques among GAN pairs. It firstly adopt GAN pair as many as the number of domain we want to produce, then share weights of some layers of each GAN pair that represents high-level semantics so that hopes to learn joint distribution of multi-domain from samples drawn from marginal domain distribution. It should be noted that since it aim to generate multi-domain images which share high-level abstract representation, each domain has to be quite similar in a broad view.

SD-GAN [80] takes similar approach with Coupled GAN in that it also adopts multi GAN pairs but does not utilizes weight sharing technique, rather its proposal is to decompose latent space as discussed above paragraph. By partitioning latent variable $z$ into identity part and pose part so that it tries to produce two images which has 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 [81] models both generator and discriminator as RNN with Long-short Term Memory(LSTM) which is widely used in sequential data managing and it aims to directly extract whole sequence of music. Whereas SeqGAN [82], ORGAN [83], [84] employ policy gradient algorithm not rather generating whole sequence at once. The reason why they adopt policy gradient is that normal GAN has poor ability in generating discrete output since the discriminator's gradient hardly affect to the generator's output. Moreover, it can not evaluate to partially generated sequence but only to whole generated sequence such as in case of C-RNN-GAN. They treat generator's output as a policy of an agent and take discriminator's output as a reward. Selecting reward with discriminator is a natural choice since generator would act to get large output of discriminator and it is corresponds to aim of reinforcement learning. In addition, ORGAN differs from SeqGAN in that it only adds hard-coded objective to reward function to achieve specified goal.

### 4.2.2 Language and Speech

SEGAN [85] attempts to incorporate speech using 1-d dilated convolution inspired from Wavenet architecture [86]. RankGAN [87] suggests language generation methods using text data generation and ranker instead of conventional discriminator. Generator tries to let the generated synthetic text be ranked high among hand-written real data and ranker evaluate rank score for each input. Since generator outputs discrete symbols, it also adopts policy gradient algorithm as [82] in that interpret generator as a policy predicting next step symbol and rank score as a value function given past generated sequence respectively.

VAW-GAN [88] suggests voice conversion system by combining GAN and VAE frameworks. It adopts conditional-VAE framework that the encoder represents phonetic content $z$ of source voice and the decoder synthesizes converted voice given target speaker information $y$. As discussed in section 3.2.2, VAE suffers from generating sharp result due to over-simplified assumption of Gaussian distribution. To deal with this issue, VAW-GAN incorporates WGAN similarly with VAEGAN [51] fashion. The reason why it takes WGAN framework with VAEGAN is that VAW-GAN wants to learn voice conversion more directly via giving reconstructed source and target voice directly to Wasserstein loss function. By assigning the decoder to the generator, it aims to reconstruct target voice given speaker representation.

### 4.2.3 Video

In this paragraph, we look into GAN variants about generating video. Generally video is composed of relatively stationary background scenery and dynamic object motions. VGAN [89] takes this points therefore suggests two stream generator. Moving foreground generator using 3D CNN predicts plausible futures of frames while static background generator using 2D CNN makes background stationary. Pose-GAN [90] takes mixture of VAE and GAN approach. It uses a VAE approach to estimate future object movements conditioned on current object pose and hidden representation of past poses. With a rendered future pose video and clip image, it uses a GAN framework to generate future frames using 3D convolution neural network. Recently, MoCoGAN [91] proposes decomposing content part and motion part of latent space of generative model, especially modeling 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

### 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 is both labeled and unlabeled data. In 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. At that time, it is often necessary to train the model with which only a small portion of the total data has labels.

[32] enables semi-supervised learning in GAN by constructing a discriminator wherein a unit corresponding to 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, to maximize $\log p_\theta(y \in 1, ..., K|x)$. For generated data, the model learn to maximize the log-likelihood of the additional unit corresponding to $y = K + 1$. The loss function for learning the model can be expressed as follows:

$$Loss = L_{supervised} + L_{unsupervised} \tag{26}$$

$$L_{supervised} = -\mathbb{E}_{x,y\sim p_{data}(x,y)}[\log p_\theta(y|x, y < K + 1)] \tag{27}$$

$$L_{unsupervised} = -\mathbb{E}_{x\sim p_{data}(x)}[1 - \log p_\theta(y = K + 1|x)] - \mathbb{E}_{x\sim G}[\log p_\theta(y = K + 1|x)], \tag{28}$$

where $G$ is the generator of GAN and since generated data also has no label, the unsupervised loss can be expressed as a GAN standard minimax game.

To surpass classification performance, unsupervised loss using unlabeled and generated data should help minimize supervised loss. The output distribution matching (ODM) cost introduced in [92] guarantees this to work well. The optimal solution for GAN is that the distribution of unlabeled real data and generated data are the same, which means the minimax game of GAN is the unsupervised ODM cost. Since the optimum of the supervised cost and the optimum of the unsupervised ODM cost are consistent, minimizing unsupervised loss improves classification performance.

CatGAN [93] is an algorithm for robust classification in the regularized information maximization (RIM) framework. The model is trained on three requirements: To make class assignment for real data have small conditional entropy $H[p(y|x, D)]$; To make class assignment for generated data have large conditional entropy $H[p(y|G(z), D)]$; and To make uniform marginal distribution, i.e. to maximize $H[p(y|D)]$. Combining these three requirements, the loss functions for the generator and discriminator is composed as follows:

$$L_G = H_G\Big[p(y|D)\Big] - \mathbb{E}_{z\sim P(z)}\Big[H[p(y|G(z), D)]\Big] \tag{29}$$

$$L_D = H_{\mathcal{X}}\Big[p(y|D)\Big] - \mathbb{E}_{x\sim\mathcal{X}}\Big[H[p(y|x, D)]\Big] + \mathbb{E}_{z\sim P(z)}\Big[H[p(y|G(z), D)]\Big]. \tag{30}$$

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, the loss $L_G$ trains the generator to minimize the mutual information between the data distribution and the predicted class distribution for the generated data. On the other hand, loss $L_D$ trains the discriminator to minimize the mutual information between the data distribution and the class distribution for the real data, and to maximize for the generator data. For semi-supervised learning, 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 [94] suggests an algorithm for semi-supervised learning with CC-GANs. CC-GAN is one of conditional GANs whose generator fills a missing image patch conditioned on surrounding pixels and the discriminator distinguishes full images as real data, and images with missing patch and imputed image as fake data. The learned generator can be used for missing value computation. Similar to the method used in [93], semi-supervised learning can be enabled by adding cross entropy term $\mathbb{E}_{x,y\sim\mathcal{X}_L}[log(D_c(y|x)]$ for labeled data to the loss function.

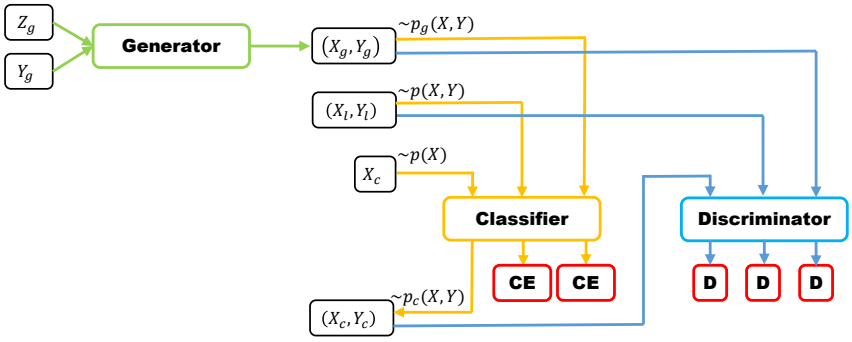



Figure 12: Triple-GAN

However, GANs 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 correct class; and the second problem is that the generator cannot generate data in a specific class. Triple GAN [95] addresses both problems in three-player formulation. Triple GAN consists of three agents: a generator $G$; a discriminator $D$; and a classifier $C$. The generator samples the generated data $X_g$ conditioned on the class label $Y_g$. The classifier 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 predict class $Y_c$ of the unlabeled data $X_c$. Discriminator is learned 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. And the three player game is defined as follows where $CE$ refers to cross entropy loss:

$$\min_{C,G}\max_{D} L(C,G,D) = \mathbb{E}_{(x,y)\sim p(x,y)}\Big[\log D(x,y)\Big] + CE \\ + (1-\alpha)\mathbb{E}_{y\sim p(y),z\sim p_z(z)}\Big[\log(1-D(G(y,z),y))\Big] + \alpha\mathbb{E}_{x\sim p(x),y\sim p_c(y|x)}\Big[\log(1-D(x,y))\Big]. \tag{31}$$

with the cross entropy loss of the classifier, [95] proves 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

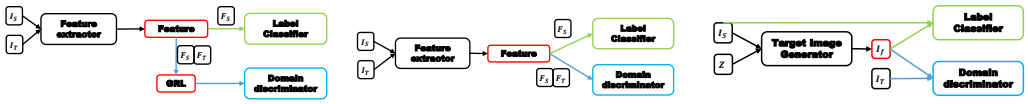
Domain adaptation is a way of transfer learning which tries to match 2 domain distribution. It can be thought as transferring prior knowledge of known source domain $\mathbb{D}_S$ to target domain $\mathbb{D}_T$ because labeled training data is difficult to obtain in real world. Especially, unsupervised domain adaptation solves following problem: given labeled source domain data $(x_s, y) \in \mathbb{D}_S \times Y$ and unlabeled target domain data $x_t \in \mathbb{D}_T$, learns a function $h : T \to Y$ also well classifying in target domain without label information of it.

DANN [96] approaches this problem with maintaining discriminative and attaining domain invariance concepts. Maintaining discriminative means maintaining label prediction ability to classify well also in target domain and domain invariance stands for making generated features trained for classifying source data not differentiating whether it comes from $\mathbb{D}_S$ or $\mathbb{D}_T$. As seen in 13a, there are 2 classifiers sharing feature extractor that a label classifier which classifies label of source data and domain discriminator which distinguishes where data comes from. We train a label classifier to classify label of source data via normal classification loss and train domain discriminator adversarial via Gradient Reversal Layer(GRL) which induces the feature generator

to extract domain invariant features. Therefore, it can be thought as in GAN framework that the feature generator acts like the generator, which produces source-like features from target domain and the domain discriminator acts like the discriminator in that it discriminates source(real) and target(fake) samples with binary cross entropy loss. To summarize, DANN takes CNN classification networks in addition with standard GAN framework to achieve the goals to classify target domain without label information.

Unsupervised pixel-level domain adaptation [97] takes different approach with DANN in that it is attempt to adapt source images to be seen as if they were drawn from target domain while DANN tries to make generated features from both domains similar. There are 3 components in suggested model as like DANN, that 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 $x_f$ and the classifier assigns class label for source image $x_s$. In addition to that, it also adds content similarity loss which reflects prior knowledge of adaptation. Content similarity loss is defined as Mean Square Error(MSE) between source image and adapted image only for masked area so as to inform to adapted image that masked area is necessarily kept during adaptation.

ARDA [98] jumps to solve target domain unlabeled data classification problem by incorporating Wasserstein GAN. As other adversarial approaches in above, ARDA also consists of three part: the feature generator generating domain invariant features, the critic estimating Wasserstein distance between generated features from source and target domains and the classifier passing supervised signal from source domain to target domain with cross entropy loss. It differs with DANN in that it accesses minimax problem with Wasserstein critic to train domain discriminator. Figure 13 shows overall outline of [96], [97] and [98] where $I_S, I_T, I_f$ stand for source domain image, target domain image, fake image and $F_S, F_T$ mean extracted source features, target features respectively.




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

Figure 13: Domain adaptation

## 4.5 Other fields

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

### 4.5.1 Medical images

GANCS [99] is used in MRI reconstruction which is a severely ill-posed problem with artifact aliasing. GANCS adopts LSGAN [4] to generate texture details and uses $l_1$ pixel-wise loss to control high frequency noise. SegAN [100] aims to segment medical image. It proposes a segmentor-critic structure which a segmentor generates predicted segmentation image and a critic maximizes hierarchical feature differences between ground-truth and generated segmentation so that induces a segmentor to learn the features of ground-truth segmentation adversarial. Also there are another medical image segmentations such as DI2IN [101] and SCAN [102]. DI2IN takes 3D CT images and do the task of liver segmentation through adversarial learning and SCAN tries to segment lung field and heart from Chest X-Ray images also through adversarial approach with ground truth segmentation mask.

### 4.5.2 Steganography

Attempts to use GAN to steganography is also feasible. Steganography is the way of concealing secret messages in non-secret container such as an image and steganalyser determines if a container contains secret message. GAN variants which apply GAN to steganography such as [103], [104], propose steganography model which consists of 3 components. First one is the generator which

produces real looking images that will be used for container. The others are 2 discriminators where $D$ classifies whether an image if real or fake and $S$ is steganalyser which determines whether an image contains secrete message.

#### 4.5.3 Others

GAN variants such as CaloGAN [105], LAGAN [106] are applied to physics, attempt to generate particle image to represent energy distribution.

Reinforcement Learning(RL) is a kinds of learning theory field which focuses on teaching an agent to choose the best action given current state. A policy $\pi(a|s)$ which is a probability of choosing an action $a$ at state $s$, is learned via on-policy or off-policy algorithm. Especially when using off-policy algorithm, past trajectories at experience replay are used to optimize $\pi(a|s)$. In early training stage, data batches in experience replay are quite high biased since a policy $\pi(a|s)$ is much far from optimum. EGAN [107] generates $(s, a, r, s')$ 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 fast. Meanwhile, RL has very similar concept with GAN in that in aspect of policy gradient, it is very important to estimate value function correctly given state $s$ and action $a$. This similarity will be discussed in section 5.2.

[108] extends GAN framework to continual learning. Continual learning is to solve multiple tasks without catastrophic forgetting which learned previous task is forgotten when a model learns a new task. [108] address this issue with GAN framework called deep generative replay. It trains a scholar model in each task where a scholar model is composed of the generator which produces samples of an old task and the solver which gives a target answer of an old task's sample. By sequentially training scholar models with old task's values generated by old scholar, it aim to overcome catastrophic forgetting even while learning new tasks.

# 5 Discussion

## 5.1 Evaluation

As we starts out representing the generative model as maximizing likelihood, it is quite sensible that evaluates the generator with log likelihood. However, as we discussed above, GAN aim to generate real-like image implicitly without likelihood assumption so that evaluation with likelihood is not quite proper. The most widely accepted evaluation metric is inception score. Inception score which is proposed by [32] and inspired from Inception Network [109] in image classification problem, is widely adopted evaluation metric in GAN. As seen in equation 32, It computes average KLD between conditional label distribution $p(y|x)$ and marginal label distribution $p(y)$ with pre-trained classifier such as VGG [110] so that it aim to capture that well generated image have conditional label distribution with low entropy so leads to high inception score. However it lacks of capturing mode collapse since $p(y|x)$ still can have low entropy even when mode collapse happens. To address this issue, [39] adopts MS-SSIM [111] which evaluates perceptual image similarity. Also, various evaluation metrics are proposed in several papers. For examples, MRGAN [34] proposes MODE score based on Inception score and [112] suggests independent Wasserstein critic which critic is trained independently for validation dataset so as 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 $p_{data}(x)$.

$$\exp(\mathbb{E}_x[KL(p(y|x)||p(y))]) \tag{32}$$

However, how to evaluate the performance of GAN is quite tricky in that each different objective functions and training procedure leads to different results. All methods discussed above share the same goal which fits $p_\theta(x)$ to $p_{data}(x)$ by sampling real samples from $p_{data}$ and training the generator adversarial from the discriminator. Particularly in section 2.1, we discussed various objective functions in that the discriminator approximates true divergence between two distributions and the generator minimizes it. Theoretical, all metrics should produce same result in assumption of full capacity of model and infinite number of training samples. However, [113] shows that minimizing MMD, JSD, KLD results in different optimal point even in mixture of Gaussian distribution and it also tells us that convergence in one kind of distance does not guarantee convergence in other

kind of distance and each object functions does not share unique global optima. In this point of view, we need to be care about choosing a proper evaluation metric for a given target application.

### 5.2 Relation to reinforcement learning

Policy gradient algorithm [45] states unbiased estimation of policy gradient under the correct action-state value $Q(s, a)$. Since $Q(s, a)$ is the discounted total reward from state $s$ and action $a$, $Q(s, a)$ must be estimated via estimator called as critic. Wide variety of policy gradient such as Deep Deterministic Policy Gradient(DDPG) [114], Trust Region Policy Optimization(TRPO) [115] and Q-prop [116] can all be though as estimating $Q(s, a)$ which has low bias and low variance to correctly estimate policy gradient. In that point of view, it is quite similar to GAN framework in that GAN's discriminator estimates distance between two distributions and the approximated distance needs to be very unbiased so as to make $p_\theta(x)$ get correctly close to $p_{data}(x)$. [117] details the connection between GAN and actor-critic methods.

Inverse Reinforcement Learning(IRL) [118] is similar reinforcement learning in that its objective is to find the optimal policy. But in IRL framework, the experts' demonstrations are given instead of reward. It finds the appropriate reward function that makes the given demonstration as optimal as possible and then outputs the optimal policy for the found reward function. There are many variants in IRL. Maximal entropy IRL [119] is the one of them and it finds the policy distribution which satisfies the constraints that feature expectations of the policy distribution and the given demonstration should be same. To solve such ill-posed problem, maixmal entropy IRL finds the policy distribution with the largest entropy according to maximal entropy principle [120]. Intuitively, it can be understand that maximal entropy IRL finds the policy distribution which maximizes the likelihood of demonstrations and its entropy. Its constraint and convexity induces 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 large likelihood of demonstrations which gives the deep connection with GAN framework. The primal variable is a policy distribution in IRL whereas it can be interpreted 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.

[121], [122], [123] have shown its mathematical connection between IRL and GAN. [123] convert IRL to the original GAN by constraining the space of dual variables and [122] has shown the relation between EBGAN and IRL using approximate inference.

## 6 Conclusion

So far, we have discussed how various object functions and architectures affect behavior of GAN and lots of GAN variants which are applied vast image related task such as image translation, image attribute edits, domain adaptation and even to other fields. There still remains open problems in theoretical aspects of GAN in whether GAN really converges and it can perfectly overcome mode collapse, as [31], [124], [125] discuss. However, with power of deep neural networks and with the utility of learning highly non-linear mapping from latent space into data space, there are still enormous opportunities to develop GAN itself and to apply GAN to various applications and fields.

## References


[1] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[3] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Information Processing Systems*, pp. 271–279, 2016.

[4] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," *arXiv preprint ArXiv:1611.04076*, 2016.

[5] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[6] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[7] S. T. Rachev *et al.*, "Duality theorems for kantorovich-rubinstein and wasserstein functionals," 1990.

[8] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[9] G.-J. Qi, "Loss-sensitive generative adversarial networks on lipschitz densities," *arXiv preprint arXiv:1701.06264*, 2017.

[10] X. Guo, J. Hong, T. Lin, and N. Yang, "Relaxed wasserstein with applications to gans," *arXiv preprint arXiv:1705.07164*, 2017.

[11] A. Banerjee, X. Guo, and H. Wang, "On the optimality of conditional expectation as a bregman predictor," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2664–2669, 2005.

[12] Y. Mroueh, T. Sercu, and V. Goel, "Mcgan: Mean and covariance feature matching gan," *arXiv preprint arXiv:1702.08398*, 2017.

[13] J. H. Lim and J. C. Ye, "Geometric gan," *arXiv preprint arXiv:1705.02894*, 2017.

[14] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[15] C. D. Aliprantis and K. C. Border, "Riesz representation theorems," in *Infinite Dimensional Analysis*, pp. 455–472, Springer, 1999.

[16] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1718–1727, 2015.

[17] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "Mmd gan: Towards deeper understanding of moment matching network," *arXiv preprint arXiv:1705.08584*, 2017.

[18] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos, "The cramer distance as a solution to biased wasserstein gradients," *arXiv preprint arXiv:1705.10743*, 2017.

[19] Y. Mroueh and T. Sercu, "Fisher gan," *arXiv preprint arXiv:1705.09675*, 2017.

[20] B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. Lanckriet, "On integral probability metrics,\phi-divergences and binary classification," *arXiv preprint arXiv:0901.2698*, 2009.

[21] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[22] R. Wang, A. Cully, H. J. Chang, and Y. Demiris, "Magan: Margin adaptation for generative adversarial networks," *arXiv preprint arXiv:1704.03817*, 2017.

[23] E. L. Denton, S. Chintala, R. Fergus, *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, pp. 1486–1494, 2015.

[24] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[25] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, "Generating images with recurrent adversarial networks," *arXiv preprint arXiv:1602.05110*, 2016.

[26] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, "Stacked generative adversarial networks," *arXiv preprint arXiv:1612.04357*, 2016.

[27] F. Juefei-Xu, V. N. Boddeti, and M. Savvides, "Gang of gans: Generative adversarial networks with maximum margin ranking," *arXiv preprint arXiv:1704.04865*, 2017.

[28] J. Yang, A. Kannan, D. Batra, and D. Parikh, "Lr-gan: Layered recursive generative adversarial networks for image generation," *arXiv preprint arXiv:1703.01560*, 2017.

[29] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[30] H. Narayanan and S. Mitter, "Sample complexity of testing the manifold hypothesis," in *Advances in Neural Information Processing Systems*, pp. 1786–1794, 2010.

[31] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang, "Generalization and equilibrium in generative adversarial nets (gans)," *arXiv preprint arXiv:1703.00573*, 2017.

[32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

[33] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.

[34] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," *arXiv preprint arXiv:1612.02136*, 2016.

[35] A. Srivastava, L. Valkov, C. Russell, M. Gutmann, and C. Sutton, "Veegan: Reducing mode collapse in gans using implicit variational learning," *arXiv preprint arXiv:1705.07761*, 2017.

[36] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "How to train your dragan," *arXiv preprint arXiv:1705.07215*, 2017.

[37] A. Ghosh, V. Kulharia, V. Namboodiri, P. H. Torr, and P. K. Dokania, "Multi-agent diverse generative adversarial networks," *arXiv preprint arXiv:1704.02906*, 2017.

[38] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[39] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," *arXiv preprint arXiv:1610.09585*, 2016.

[40] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2172–2180, 2016.

[41] A. Spurr, E. Aksan, and O. Hilliges, "Guiding infogan with semi-supervision," *arXiv preprint arXiv:1707.04487*, 2017.

[42] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville, "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016.

[43] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016.

[44] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Adversarial generator-encoder networks," *arXiv preprint arXiv:1704.02304*, 2017.

[45] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, pp. 1057–1063, 2000.

[46] T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio, "Maximum-likelihood augmented discrete generative adversarial networks," *arXiv preprint arXiv:1702.07983*, 2017.

[47] R. D. Hjelm, A. P. Jacob, T. Che, K. Cho, and Y. Bengio, "Boundary-seeking generative adversarial networks," *arXiv preprint arXiv:1702.08431*, 2017.

[48] Y. Kim, K. Zhang, A. M. Rush, Y. LeCun, *et al.*, "Adversarially regularized autoencoders for generating discrete structures," *arXiv preprint arXiv:1706.04223*, 2017.

[49] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune, "Plug & play generative networks: Conditional iterative generation of images in latent space," *arXiv preprint arXiv:1612.00005*, 2016.

[50] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[51] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[52] M. Rosca, B. Lakshminarayanan, D. Warde-Farley, and S. Mohamed, "Variational approaches for auto-encoding generative adversarial networks," *arXiv preprint arXiv:1706.04987*, 2017.

[53] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016.

[54] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, Springer, 2015.

[55] C. Wang, C. Xu, C. Wang, and D. Tao, "Perceptual adversarial networks for image-to-image transformation," *arXiv preprint arXiv:1706.09138*, 2017.

[56] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.

[57] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," *arXiv preprint arXiv:1703.05192*, 2017.

[58] Z. Yi, H. Zhang, P. T. Gong, *et al.*, "Dualgan: Unsupervised dual learning for image-to-image translation," *arXiv preprint arXiv:1704.02510*, 2017.

[59] S. Benaim and L. Wolf, "One-sided unsupervised domain mapping," *arXiv preprint arXiv:1706.00826*, 2017.

[60] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *arXiv preprint arXiv:1612.07828*, 2016.

[61] S. Zhou, T. Xiao, Y. Yang, D. Feng, Q. He, and W. He, "Genegan: Learning object transfiguration and attribute subspace from unpaired data," *arXiv preprint arXiv:1705.04932*, 2017.

[62] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan, "Perceptual generative adversarial networks for small object detection," in *IEEE CVPR*, 2017.

[63] K. Ehsani, R. Mottaghi, and A. Farhadi, "Segan: Segmenting and generating the invisible," *arXiv preprint arXiv:1703.10239*, 2017.

[64] H. Wu, S. Zheng, J. Zhang, and K. Huang, "Gp-gan: Towards realistic high-resolution image blending," *arXiv preprint arXiv:1703.07195*, 2017.

[65] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *arXiv preprint arXiv:1612.03242*, 2016.

[66] A. Dash, J. C. B. Gamboa, S. Ahmed, M. Z. Afzal, and M. Liwicki, "Tac-gan-text conditioned auxiliary classifier generative adversarial network," *arXiv preprint arXiv:1703.06412*, 2017.

[67] G. Antipov, M. Baccouche, and J.-L. Dugelay, "Face aging with conditional generative adversarial networks," *arXiv preprint arXiv:1702.01983*, 2017.

[68] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, "Invertible conditional gans for image editing," *arXiv preprint arXiv:1611.06355*, 2016.

[69] M. Gorijala and A. Dukkipati, "Image generation and editing with variational info generative adversarialnetworks," *arXiv preprint arXiv:1701.04568*, 2017.

[70] W. Yin, Y. Fu, L. Sigal, and X. Xue, "Semi-latent gan: Learning to generate and modify facial images from attributes," *arXiv preprint arXiv:1704.02166*, 2017.

[71] L. Tran, X. Yin, and X. Liu, "Representation learning by rotating your faces," *arXiv preprint arXiv:1705.11136*, 2017.

[72] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis," *arXiv preprint arXiv:1704.04086*, 2017.

[73] Y. Lu, Y.-W. Tai, and C.-K. Tang, "Conditional cyclegan for attribute guided face image generation," *arXiv preprint arXiv:1705.09966*, 2017.

[74] T. Kim, B. Kim, M. Cha, and J. Kim, "Unsupervised visual attribute transfer with reconfigurable generative adversarial networks," *arXiv preprint arXiv:1707.09798*, 2017.

[75] L. Karacan, Z. Akata, A. Erdem, and E. Erdem, "Learning to generate images of outdoor scenes from attributes and semantic layouts," *arXiv preprint arXiv:1612.00215*, 2016.

[76] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv:1609.04802*, 2016.

[77] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *Advances in Neural Information Processing Systems*, pp. 82–90, 2016.

[78] M. Gadelha, S. Maji, and R. Wang, "3d shape induction from 2d views of multiple objects," *arXiv preprint arXiv:1612.05872*, 2016.

[79] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in neural information processing systems*, pp. 469–477, 2016.

[80] C. Donahue, A. Balsubramani, J. McAuley, and Z. C. Lipton, "Semantically decomposing the latent spaces of generative adversarial networks," *arXiv preprint arXiv:1705.07904*, 2017.

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

[82] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient.," in *AAAI*, pp. 2852–2858, 2017.

[83] G. L. Guimaraes, B. Sanchez-Lengeling, P. L. C. Farias, and A. Aspuru-Guzik, "Objective-reinforced generative adversarial networks (organ) for sequence generation models," *arXiv preprint arXiv:1705.10843*, 2017.

[84] S.-g. Lee, U. Hwang, S. Min, and S. Yoon, "A seqgan for polyphonic music generation," *arXiv preprint arXiv:1710.11418*, 2017.

[85] S. Pascual, A. Bonafonte, and J. Serrà, "Segan: Speech enhancement generative adversarial network," *arXiv preprint arXiv:1703.09452*, 2017.

[86] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[87] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," *arXiv preprint arXiv:1705.11001*, 2017.

[88] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, "Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks," *arXiv preprint arXiv:1704.00849*, 2017.

[89] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances In Neural Information Processing Systems*, pp. 613–621, 2016.

[90] J. Walker, K. Marino, A. Gupta, and M. Hebert, "The pose knows: Video forecasting by generating pose futures," *arXiv preprint arXiv:1705.00053*, 2017.

[91] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "Mocogan: Decomposing motion and content for video generation," *arXiv preprint arXiv:1707.04993*, 2017.

[92] I. Sutskever, R. Jozefowicz, K. Gregor, D. Rezende, T. Lillicrap, and O. Vinyals, "Towards principled unsupervised learning," *arXiv preprint arXiv:1511.06440*, 2015.

[93] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," *arXiv preprint arXiv:1511.06390*, 2015.

[94] E. Denton, S. Gross, and R. Fergus, "Semi-supervised learning with context-conditional generative adversarial networks," *arXiv preprint arXiv:1611.06430*, 2016.

[95] C. Li, K. Xu, J. Zhu, and B. Zhang, "Triple generative adversarial nets," *arXiv preprint arXiv:1703.02291*, 2017.

[96] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, "Domain-adversarial neural networks," *arXiv preprint arXiv:1412.4446*, 2014.

[97] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," *arXiv preprint arXiv:1612.05424*, 2016.

[98] J. Shen, Y. Qu, W. Zhang, and Y. Yu, "Adversarial representation learning for domain adaptation," *arXiv preprint arXiv:1707.01217*, 2017.

[99] M. Mardani, E. Gong, J. Y. Cheng, S. Vasanawala, G. Zaharchuk, M. Alley, N. Thakur, S. Han, W. Dally, J. M. Pauly, *et al.*, "Deep generative adversarial networks for compressed sensing automates mri," *arXiv preprint arXiv:1706.00051*, 2017.

[100] Y. Xue, T. Xu, H. Zhang, R. Long, and X. Huang, "Segan: Adversarial network with multi-scale $l\_1$ loss for medical image segmentation," *arXiv preprint arXiv:1706.01805*, 2017.

[101] D. Yang, T. Xiong, D. Xu, Q. Huang, D. Liu, S. K. Zhou, Z. Xu, J. Park, M. Chen, T. 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*, pp. 633–644, Springer, 2017.

[102] W. Dai, J. Doyle, X. Liang, H. Zhang, N. Dong, Y. Li, and E. P. Xing, "Scan: Structure correcting adversarial network for chest x-rays organ segmentation," *arXiv preprint arXiv:1703.08770*, 2017.

[103] D. Volkhonskiy, I. Nazarov, B. Borisenko, and E. Burnaev, "Steganographic generative adversarial networks," *arXiv preprint arXiv:1703.05502*, 2017.

[104] H. Shi, J. Dong, W. Wang, Y. Qian, and X. Zhang, "Ssgan: Secure steganography based on generative adversarial networks," *arXiv preprint arXiv:1707.01613*, 2017.

[105] M. Paganini, L. de Oliveira, and B. Nachman, "Calogan: Simulating 3d high energy particle showers in multi-layer electromagnetic calorimeters with generative adversarial networks," *arXiv preprint arXiv:1705.02355*, 2017.

[106] L. de Oliveira, M. Paganini, and B. Nachman, "Learning particle physics by example: Location-aware generative adversarial networks for physics synthesis," *arXiv preprint arXiv:1701.05927*, 2017.

[107] V. Huang, T. Ley, M. Vlachou-Konchylaki, and W. Hu, "Enhanced experience replay generation for efficient reinforcement learning," *arXiv preprint arXiv:1705.08245*, 2017.

[108] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *arXiv preprint arXiv:1705.08690*, 2017.

[109] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[110] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[111] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[112] I. Danihelka, B. Lakshminarayanan, B. Uria, D. Wierstra, and P. Dayan, "Comparison of maximum likelihood and gan-based training of real nvps," *arXiv preprint arXiv:1705.05263*, 2017.

[113] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," *arXiv preprint arXiv:1511.01844*, 2015.

[114] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[115] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1889–1897, 2015.

[116] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine, "Q-prop: Sample-efficient policy gradient with an off-policy critic," *arXiv preprint arXiv:1611.02247*, 2016.

[117] D. Pfau and O. Vinyals, "Connecting generative adversarial networks and actor-critic methods," *arXiv preprint arXiv:1610.01945*, 2016.

[118] P. Abbeel and A. Y. Ng, "Inverse reinforcement learning," in *Encyclopedia of machine learning*, pp. 554–558, Springer, 2011.

[119] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *AAAI*, vol. 8, pp. 1433–1438, Chicago, IL, USA, 2008.

[120] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957.

[121] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," *arXiv preprint arXiv:1611.03852*, 2016.

[122] J. Yoo, H. Ha, J. Yi, J. Ryu, C. Kim, J.-W. Ha, Y.-H. Kim, and S. Yoon, "Energy-based sequence gans for recommendation and their connection to imitation learning," *arXiv preprint arXiv:1706.09200*, 2017.

[123] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

[124] P. Grnarova, K. Y. Levy, A. Lucchi, T. Hofmann, and A. Krause, "An online learning approach to generative adversarial networks," *arXiv preprint arXiv:1706.03269*, 2017.

[125] L. Mescheder, S. Nowozin, and A. Geiger, "The numerics of gans," *arXiv preprint arXiv:1705.10461*, 2017.