Generative Adversarial Networks - II

Viraj Shah
ECE, UIUC

Jiachen Tu
ECE, UIUC

Neeraj Wagh
BIOE, UIUC

March 5, 2022

1 Introduction

1.1 Generative Adversarial Networks

Generative Adversarial Network (GAN) is a novel machine learning tool that grants the capability to learn and sample from high-dimensional data distributions such as datasets of natural images [4]. A standard GAN is made of two components, both of them are typically represented by trainable deep neural networks, termed as generator (G) and discriminator (D). The goal of a discriminator (D) is to learn to distinguish between the images generated by the generator (fake) and the images coming from the training data (real). On the other hand, the generator (G) takes a low-dimensional latent variable z as its input, and aims to fool the discriminator by generating realistic images that mimic the samples from the real data. As depicted in Fig. 1, the generator (G) and discriminator (D) are trained together by making them compete against each other, *i.e.* in adversarial fashion. In other words, the generator trains under the supervision of a discriminator, while the discriminator itself is learning the underlying data distribution from real data. If trained sufficiently well, when equilibrium is reached, the generator is able to synthesize images that closely resemble the images of the real data [1, 6, 2, 3].

1.2 Objective Function for Minimax game in GANs

Formally put, the generator (G) takes a $k-$dimensional vector z as the input, where k is smaller than the ambient image dimensions. z is generally randomly sampled from a Gaussian or uniform random distribution. Generator (G) and discriminator (D) are parameterized by network parameters $\theta^{(G)}$ and $\theta^{(D)}$ respectively. The minimax objective of GAN training can be written as:

$$J\left(D,G\right)=\min_{G}\max_{D}\ \mathbb{E}_{x\sim p_{\text{data}}}\left[\log\left(D(x)\right)\right]+\mathbb{E}_{x\sim p_{z}}\left[\log\left(1-D(G(z))\right)\right], \quad (1)$$

where p_{data} depicts the distribution of training data, p_g depicts the distribution of synthetic samples generated by generator (G), and p_z depicts the distribution of latent variable z that is known apriori.

To understand how the objective in Eq. 1 helps us mimic the real data distribution, let us take a closer look at a role a discriminator (D) plays in the minimax game.

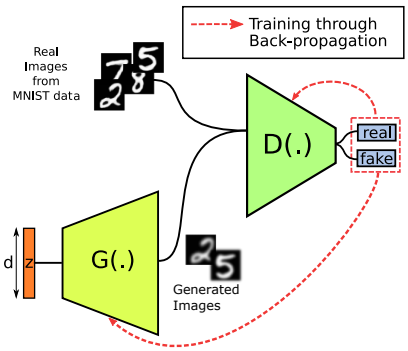

Figure 1: Illustration of Generative Adversarial Network model being trained on MNIST dataset

According to Eq. 1, we first aim to maximize the GAN objective with respect to discriminator parameters ($D^{(\theta)}$). Thus, let us evaluate the optimal discriminator D for a fixed generator G. Note that we do not put any limit on model capacity of G and D by choosing a non-parametric setting.

For a fixed generator (G), optimal discriminator (D^*_G) is given by:

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

Further,

$$\begin{aligned} J(D,G) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz, \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))\ dx. \end{aligned} \tag{2}$$

Note that the Eq. 2 is obtained by expanding the domain of the integration to contain generated data $\tilde{x} = G(z)$ as well. That way, the support of discriminator becomes $Supp(p_{\text{data}}) \cup Supp(p_g)$, and it is not needed to be defined outside its support.

To maximize the objective in Eq. 2, we can maximize the integrand over D at every point in its support, *i.e.*, optimize D in a function space so that for every value of x in the support, the value of $D(x)$ maximizes the integrand. Thus,

$$D^*(x) = \max_D\ p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x})) \tag{3}$$

Since $D(x)$ indicates probabilities of sample being real or fake, $D(x) \in [0, 1]$. Taking gradients of integrand and setting it zero to find optimal D:

$$\begin{aligned} &\frac{p_{\text{data}}(\boldsymbol{x})}{D(\boldsymbol{x})} - \frac{p_g(\boldsymbol{x})}{1 - D(\boldsymbol{x})} = 0, \\ \implies &p_{\text{data}}(\boldsymbol{x}) - p_{\text{data}}(\boldsymbol{x})D - p_g(\boldsymbol{x})D = 0, \\ \implies &D^*_G(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \end{aligned} \tag{4}$$

Plugging D_G^* from Eq. 4 into GAN objective, we get:

$$
\begin{aligned}
J(D_G^*, G) =& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \qquad (5)\\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))]\\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]\\
=& -\log(4) + KL\left(p_{\text{data}} \,\middle\|\, \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \,\middle\|\, \frac{p_{\text{data}} + p_g}{2}\right) \qquad (6)
\end{aligned}
$$

where KL is the Kullback–Leibler divergence. The later part of the expression is referred as Jensen-Shannon divergence between two distributions:

$$J(D_G^*, G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \,\|\, p_g\right) \qquad (7)$$

The above equation shows that if the discriminator is trained optimally, then at each generator update the minimization of the GAN objective turns out to be the minimization of the Jenson-Shannon divergence between p_{data} and p_g, reaching optimality when $p_{\text{data}} = p_g$.

2 Maximum Likelihood Objective for GAN

Maximum likelihood objective is widely used in many machine learning algorithms. As the name suggests, the idea is to maximize the likelihood of the real data given the observations. Note that maximizing the likelihood is equivalent to minimizing the KL divergence between the data distribution and model distribution - due to similar notion of minimizing the separation between two data distributions, it is rather intuitive to replace the Jenson-Shannon divergence objective with maximum likelihood objective in the case of GANs.

To obtrain MLE-based GAN objective, given an optimal discriminator, we equate the generator gradients with the gradients of KL divergence between real and generated distribution. In other words, the expected gradients of

$$J^{(G)} = \mathbb{E}_{x\sim p_g} f(x)$$

is equal to the expected gradient of $KL(p_{\text{d}ata}\|p_g)$. $f(\cdot)$ is a function we aim to determine.

Taking derivative of KL divergence with repsect to θ:

$$\frac{\partial}{\partial\theta} KL(p_{\text{d}ata}\|p_g) = -\mathbb{E}_{x\sim p_{\text{d}ata}} \frac{\partial}{\partial\theta} \log p_g(x). \qquad (8)$$

Taking derivative of generator objective $J^{(G)}$:

$$\frac{\partial}{\partial\theta} J^{(G)} = \frac{\partial}{\partial\theta} \mathbb{E}_{x\sim p_g} f(x) \log p_g(x).$$

Here, we use the $\log p_g(x)$ with an assumption that $p_g(x) \geq 0$. Further, we assume that Leibniz's rule can be applied here to interchange the order of integration and differentiation:

$$\frac{\partial}{\partial\theta}J^{(G)} = \mathbb{E}_{x\sim p_g} f(x)\frac{\partial}{\partial\theta}\log p_g(x).$$

Note that the expectation is computed over samples from p_g, while it is preferable that it is computed over samples from $p_{\mathrm{d}ata}$. To make such modification, we can employ *importance sampling trick* and can set $f(x) = \frac{p_{\mathrm{d}ata}(x)}{p_g(x)}$. In a way, this setup compensates for drawing samples from generator by reweighing the contribution to the gradient from each generator sample. Such ratio is already being calculated by the dscriminator, and with the use of some algebra, computationally stable version of $f(x)$ can be achieved.

3 Mode Collapse

Up to this point, GAN is formulated as a minimax game (zero-sum) problem between 2 neural networks: a discriminator and a generator. Most people would think that GAN is successful based on twp criteria: 1) generator reliably generates data that fools discriminator; 2) creates samples that are as diverse as distribution of real-world. Model collapse happens when the GAN fails to achieve 2) by achieving 1) through a concentrated distribution.

From the point of view from game theory, model collapse may arise because the maximum solution of GAN is different from the minimax solution. During the training process, the minimax and maximini approach might coincide, which is typical in a GAN game and here we will investigate how this phenomenon happens.

Suppose we formulate a GAN model G^* with value function $V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}\,(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$, then the minimax solution is represented by the equation 9:

$$G^* = \min_G \max_D (V(G,D)) \tag{9}$$

In this case, the model G^* will draw samples from the full distribution, which is the ideal goal for our training objective. On the other hand, during the implementation of training process, the order of the min and max optimizing process could be exchange to become a maximin solving process which is represented by the equation 10

$$G^* = \min_D \max_G (V(G,D)) \tag{10}$$

In the maximin process, the $\max_G$ is in the inner loop of optimization, and the generator is asked to map every latent z value to the single output variable x that the discriminator D believes is most likely to be real rather than false.

During simultaneous gradient descent, the optimization process does not clearly privilege minimax over maximin. The result of this behavior – getting maximin solution although hoping to use minimax – often leads to the mode collapse for the model G^*.

4 Methods to handle mode collapse

4.1 UnrolledGAN

With the intention of reducing the effect of mode collapse, unrolled GAN addresses the challenges of unstable optimization and mode collapse in GANs by unrolling optimization of the discriminator objective during training. By updating the generator's loss function to back propagate through k steps of gradient updates for discriminator, the generator would be able to see k steps into future to encourage more diverse samples. The intuition behind this is that one believes that the game strategy could be improved by looking ahead for the next few moves of your opponent and better prepare the next moves with more choices. Unrolled GAN lowers the chance that the generator is overfitted for a specific discriminator. This lessens mode collapse and improves stability.

Specifically, the GAN learning problem is formulated to find the optimal parameters θ_G^* for a generator function $G(z;\theta_G)$ in a minimax objective,

$$\theta_G^* = \underset{\theta_G}{\operatorname{argmin}}\, \theta_D f(\theta_G, \theta_D) \tag{11}$$

$$= \underset{\theta_G}{\operatorname{argmin}}\, f(\theta_G, \theta_D^*(\theta_G)) \tag{12}$$

$$\theta_D^*(\theta_G) = \underset{\theta_D}{\operatorname{argmax}}\, f(\theta_G, \theta_D), \tag{13}$$

where f is commonly chosen to be

$$f(\theta_G, \theta_D) = \mathbb{E}_{x\sim p_{data}}[\log(D(x;\theta_D))] + \mathbb{E}_{z\sim\mathcal{N}(0,I)}[\log(1 - D(G(z;\theta_G);\theta_D))]. \tag{14}$$

Explicitly solving for the optimal discriminator parameters $\theta_D^*(\theta_G)$ for every update step of the generator G is computationally infeasible for discriminators based on neural networks. Therefore this minimax optimization problem is typically solved by alternating gradient descent on θ_G and ascent on θ_D. The optimal solution $\theta^* = \{\theta_G^*, \theta_D^*\}$ is a fixed point of these iterative learning dynamics. Additionally, if $f(\theta_G, \theta_D)$ is convex in θ_G and concave in θ_D, then alternating gradient descent (ascent) trust region updates are guaranteed to converge to the fixed point, under certain additional weak assumptions. However in practice $f(\theta_G, \theta_D)$ is generally not convex in θ_G and concave in θ_D, and updates are not constrained in an appropriate way. As a result GAN training suffers from mode collapse, undamped oscillations, and other related problems.

A local optimum of the discriminator parameters θ_D^* can be expressed as the fixed point of an iterative optimization procedure,

$$\theta_D^0 = \theta_D \tag{15}$$

$$\theta_D^{k+1} = \theta_D^k + \eta^k \frac{\mathrm{d}f\left(\theta_G, \theta_D^k\right)}{\mathrm{d}\theta_D^k} \tag{16}$$

$$\theta_D^*(\theta_G) = \lim_{k\to\infty} \theta_D^k, \tag{17}$$

where η^k is the learning rate schedule.

By unrolling for K steps, a surrogate objective is created for the update of the generator,

$$f_K(\theta_G, \theta_D) = f\left(\theta_G, \theta_D^K(\theta_G, \theta_D)\right). \tag{18}$$

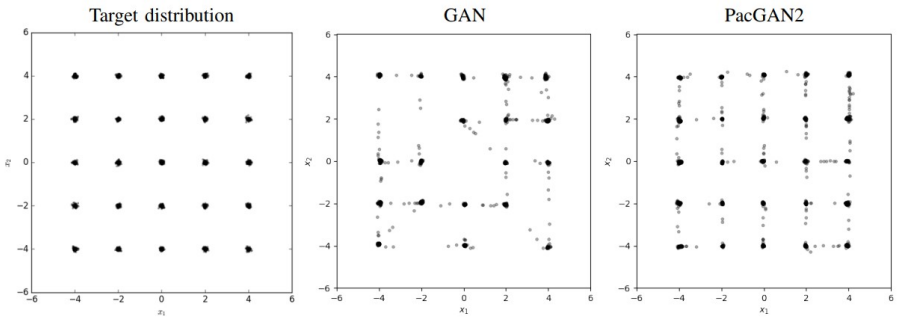

Figure 2: An illustration of the efficacy of packing. **Left:** 25-mode synthetic Gaussian data to be learned; **Middle:** Mode collapse in samples generated by regular GANs; **Right:** Samples generated with the idea of packing, which is able to equally samples from all the 25 modes. Source: `https://par.nsf.gov/servlets/purl/10168460`

When $K = 0$ this objective corresponds exactly to the standard GAN objective, while as $K \to \infty$ it corresponds to the true generator objective function $f\left(\theta_G, \theta_D^*\left(G\right)\right)$. By adjusting the number of unrolling steps K, we are thus able to interpolate between standard GAN training dynamics with their associated pathologies, and more costly gradient descent on the true generator loss.

The generator and discriminator parameter updates using this surrogate loss are

$$\theta_G \leftarrow \theta_G - \eta \frac{\mathrm{d} f_K\left(\theta_G, \theta_D\right)}{\mathrm{d}\theta_G} \tag{19}$$

$$\theta_D \leftarrow \theta_D + \eta \frac{\mathrm{d} f\left(\theta_G, \theta_D\right)}{\mathrm{d}\theta_D}. \tag{20}$$

4.2 Packing

Context: Many attempts to resolve the mode collapse issue have relied on modifications to model architectures, loss functions, and optimization algorithms. Despite achieving empirical success, these proposals lack a formal framework for explaining why they work and the extent of their sensitivity to hyperparameters. In PacGAN [5], the authors propose the idea of "packing", which examines GANs through the lens of classical statistical hypothesis testing. Intuitively, packing develops the idea of using multiple samples for the discriminator D to mitigate mode collapse since the lack of diversity is easier for D to detect when presented multiple samples rather than one. Their study shows that packing theoretically and empirically mitigates mode collapse without significant computational overhead. An empirical example of PacGAN is shown in Figure 2.

Key idea: Use an augmented discriminator $D(X_1, X_2, ..., X_m)$ that maps m "packed" or concatenated samples to a single (soft) label. These m samples are drawn independently from the same distribution - either real (jointly labelled Y=1) or generated (Y=0). The discriminator is then conceptualized to per-

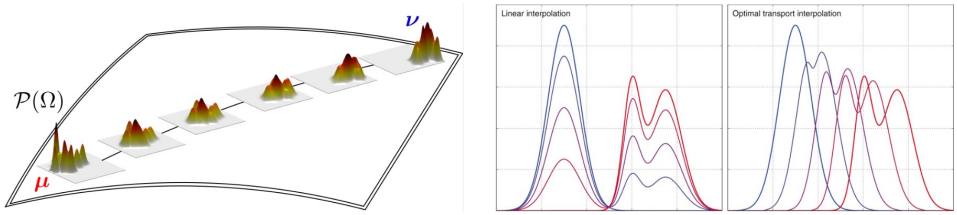

Figure 3: **Left:** The Wasserstein distance in some geometric space between two densities μ and ν is given by the length of the black line. This black line's trajectory represents the least cost with/shortest distance along which the mass of μ can be transported into ν; **Right:** An example of how optimal transport geometry is different from traditional linear geometry. Linear interpolation weights the two densities up or down but optimal transport interpolates with awareness of the underlying geometry. Source: A primer on optimal transport, NIPS 2017 Tutorial

form a binary hypothesis test on the product distributions (P^m, Q^m) of these m samples i.e., whether they were drawn from the true data distribution or the implicitly generated one. In other words, packed samples can be seen as a single sample drawn from the product distributions P^m (for real) or Q^m (for generated).

5 1-Wasserstein Distance

5.1 Definition

Let $x_i, ..., x_n \sim P$ and $y_i, ..., y_n \sim Q$ and let the corresponding density functions of the empirical distributions be p and q. We assume that $x_i, y_i \in \mathcal{R}^d$. The notion of 1-Wasserstein distance between the empirical distributions P and Q depicted in Figure 3 can be defined as:

$$W_1(P, Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

where $\Pi(P, Q)$ is the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively P and Q. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform distribution P into distribution Q. The Wasserstein distance $W_1(P, Q)$ then is the "cost" associated with the optimal/shortest transport plan.

5.2 Properties

The Wasserstein distance offers certain advantageous properties over other distance measures between probability distributions such as total variation, KL divergence, Hellinger, Euclidean, and JS divergence. Source: `https://www.stat.cmu.edu/~larry/=sml/Opt.pdf`:

1. P and Q can be either discrete or continuous.

2. Owing to its origin in optimal transport, it captures the geometry of the probability space. This can be seen from Figure 3.

3. Average of multiple distributions yields a similar distribution.

4. The interpolative optimal transport map associated with the distance measure gives us a deeper understanding of the difference between P and Q. This can also be seen from Figure 3.

5. The distance is continuous everywhere and as a loss function provides a computationally usable gradient over the parameter space.

5.3 Connection to GANs

A variant of GAN called Wasserstein GAN (WGAN) [1] minimizes an efficient approximation of the Wasserstein distance. Essentially, WGAN formulates the loss to more directly represent minimizing the difference between two probability distributions. As a result, winning the game correlates with minimizing the difference in these distributions rather than have the generator merely fool the discriminator. WGANs offer significant practical benefits to training: 1) there is no need to carefully balance generator and discriminator training, 2) the training is not highly sensitive to changes in model architecture, 3) it reduces mode collapse, and 4) the distance correlates well with sample quality and therefore makes debugging easier. However, the infimum in the definition above makes computation intractable, therefore an approximation or relaxation is necessary for practical purposes. This relaxation is admitted by the Kantorovich-Rubenstein duality, which will be introduced and proved in the next lecture.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *ArXiv:1701.07875*, 2017.

[2] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. *ArXiv:1703.10717*, 2017.

[3] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *ArXiv:1809.11096*, 2018.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. Adv. in Neural Processing Systems (NeurIPS)*, pages 2672–2680, 2014.

[5] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. *Advances in neural information processing systems*, 31, 2018.

[6] J.-Y. Zhu, T. Park, P. Isola, and A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. IEEE Conf. Comp. Vision and Pattern Recognition (CVPR)*, 2017.