Generalization and Equilibrium in Generative Adversarial Nets

Sanjeev Arora
Rong Ge
Yingyu Liang
Tengyu Ma
Yi Zhang

Background: GAN

- GANs work by comparing two distributions where the generator net part of the GAN attempts to “fool” the discriminator function into thinking that the generated image is a real image.

- The GAN is trained through backpropagation of the result from the discriminator and so it can be thought of as a game between two players solving a min max scenario.

Simplified objective function (Wasserstein 2017):

$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \mathbf{E}_{x \sim \mathcal{D}_{real}}[D_v(x)] - \mathbf{E}_h[D_v(G_u(h))]$$

Background: GAN (2)

$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \mathbf{E}_{x \sim \mathcal{D}_{real}}[D_v(x)] - \mathbf{E}_h[D_v(G_u(h))]$$

The objective function and training is complete when the model converges. However, there are issues with convergence as the convergence points for the discriminator and generator can oscillate.

So the traditional training stopping point is when the gradient arrived at is 0.

The paper, instead, suggests that an equilibrium should be reached where the value of the objective function remains unchanged. No more improvements to either nets essentially.

Problem Statement:

- GANS do not have an equilibrium/finding one leads to extreme instability.

- GANs do not have good generalization
  - Ex: GAN is trained and the generator wins over the discriminator (it is stuck performing random guessing) but the generated distribution is not able to match the target distribution accurately and is severely worse off compared to the target dist.

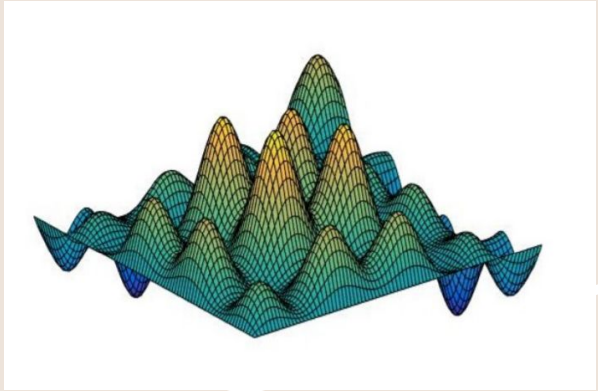
- How to get good a GAN that is able to match the target distribution?
  - Invent a new type of generator deep neural network that is able to read in a Gaussian Distribution and output samples from a distribution to be as close as possible to the target distribution. **Fundamentally this concept would work but in practice there are a lot of issues.**

- Can we achieve this all with a small number of parameters?

Problem Statement Continued:

- Taking the GAN model mentioned earlier: train a discriminator deep net by taking values from target distribution and the generator net to improve the discriminator. Stop when the discriminator is unable to differentiate the output from the target distribution and from the generated one.
    - This works for simple cases, but what about a complicated target distribution?
    - What metrics do you use to tell when they are similar?
    - Hard to tell when results when dealing with complicated target distributions.
- Complicated target distributions mean a lot of peaks and valleys.

Problem Statement Continued: Complicated Target Distributions



With the example target distribution above there is a large problem with dimensionality and so there could be an exponential number of various extremes. The biggest issue is when it comes to sampling you may not be able to sample properly to represent all the various peaks and valleys (ex: not enough samples for instance? How to tell if your samples have good coverage?). A sufficient sample size could also be exponential and thus makes it reasonable impossible to work with.

Other Problems

- WGANs and the usage of a zero gradient for halting were invented to improve the GAN model.

  However, there is another big issue.

- Equivalence is another factor that must exist similar to convergence. If the GAN model doesn't have a balance factor between the generator and the discriminator then there will be major instability.

Innovations from the Paper:

- Old measures of distance between distributions are not as useful, thus the paper invents a new type of distance.
    - Old distance formulas / standard metrics have situations where they selecta generator function, but shouldn't as there is a big gap between the two distributions.

- The neural net distance invented to better gauge the distance between the target distribution and the generated/trained distribution.
    - Some issues with this as the meric is more loose in its acceptance criteria. This leads to scenarios where metric can be near-zero even when the two distributions are far apart.

- How to measure and find equilibrium? On paper an infinite mix of generators is the best way to go about representing a probabilistic distribution. In reality, the paper attempts to do this with a reasonable size of generators mixed together.

- Use all the information above to create a new architecture for the generator network such that an approximate equilibrium exists that is pure. (Reduces number of generators needed for a probabilistic distribution from exponential to quadratic). Thus the MIX-GAN.

Defining Generalization:

Inspired by supervised learning where training is considered generalized if the training and test error are closely tied.

Let x1...xm be the training examples and let $\hat{D}_{real}$ be the uniform distribution for x1...xm. Also let $G_u(h1)...G_u(h_r)$ be a set of r examples from the generated distribution D_G. In training, we use: $\mathbb{E}_{x\sim\hat{\mathcal{D}}_{real}}[\phi(D_v(x))]$ to approximate the value of $\mathbb{E}_{x\sim\mathcal{D}_{real}}[\phi(D_v(x))]$. Attempt to minimize the distance (or divergence) between D_{real} and D_G.

Thus:

Given $\hat{D}_{real}$, an empirical version of the true distribution with m samples, a generated distribution D_G generalizes under the divergence or distance between distributions d(·, ·) with generalization error ε if the following holds with high probability:

$$\left|d(\mathcal{D}_{real}, \mathcal{D}_G) - d(\hat{\mathcal{D}}_{real}, \hat{\mathcal{D}}_G)\right| \leq \varepsilon$$

where and $\hat{D}_G$ is an empirical version of the generated distribution D_G with polynomial number of samples (drawn after D_G is fixed)

Generalization Explained:

Generalization for a GAN can be simplified down to the idea that the population distance (divergence) between the true and generated distributions is close to the empirical distance between the empirical distributions. The goal is to make the population distance as small as possible. The empirical distance is minimized through practice.

The goal of the team that worked on this paper was to improve the GAN performance by improving how generalization is done and reduce runtime of trying to implement a good generalization. Reducing the exponential time to a more reasonable polynomial time.

Current models just as the WGAN don’t generalize when given a polynomial number of examples because the population distance is not reflected by the empirical distance.

Model: Original GAN Function

Take the GAN training function (Goodfellow et al., 2014) as a base:

$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \mathop{\mathbb{E}}_{x \sim \mathcal{D}_{real}} [\log D_v(x)] + \mathop{\mathbb{E}}_{x \sim \mathcal{D}_{G_u}} [\log(1 - D_v(x))].$$

- u, v are parameters for optimization.
- D_v generally is the class of discriminator functions, but in this case it is a specific function.
  - Assign high values to $D_v(x)$ when dealing with a real sample and a small value when dealing with generated samples.
- The log function is there to an interpretation for the likelihood (recall log-likelihood). Problems exist with the log function (log x) as x approaches 0.

Model: MIX-GAN Training Func.

Take the GAN training function (Goodfellow et al., 2014) as a base and change the measuring function:

$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \mathop{\mathbb{E}}_{x \sim \mathcal{D}_{real}} [\phi(D_v(x))] + \mathop{\mathbb{E}}_{x \sim \mathcal{D}_{G_u}} [\phi(1 - D_v(x))].$$

- Mostly everything is the same except for the log function which was replaced with this new function called the “measuring function”.
  - The measuring function should be concave for handling when D_{real} and D_G are the same function (real being the target distribution and G being the generated).
  - Best case when the distributions are the same or are identical: output ½ since the training function is unable to tell the difference (ideal outcome).

Model: Getting the Optimal Discriminator

Researchers assume they have infinite samples to work with to find the best discriminator function from a class containing all neural nets. This has led to deciphering a formula to figure out the best D(x).

$$D(x) = \frac{P_{real}(x)}{P_{real}(x)+P_G(x)}$$

- This function was first derived in *Goodfellow et al. [2014].* Here $P_{real}(x)$ is the density of x in the real distribution and $P_G(x)$ is the density of x in the distribution generated by generator G.
  - Computationally this function is impossible to get.

Neural Net Distance (1)

- Distance metric designed to be a GAN objective that can be minimized and able to be analyzed for generalization performance.
- Derivation based off F-Distance, recall F-Divergence:

Definition 2 ($\mathcal{F}$-distance)**.** Let $\mathcal{F}$ be a class of functions from $\mathbb{R}^d$ to $[0,1]$ such that if $f \in \mathcal{F}, 1-f \in \mathcal{F}$. Let ϕ be a concave measuring function. Then the *$\mathcal{F}$-divergence with respect to ϕ between two distributions* μ and ν supported on $\mathbb{R}^d$ is defined as

$$d_{\mathcal{F},\phi}(\mu,\nu) = \sup_{D\in\mathcal{F}} \mathop{\mathbb{E}}_{x\sim\mu}[\phi(D(x))] + \mathop{\mathbb{E}}_{x\sim\nu}[\phi(1-D(x))] - 2\phi(1/2)$$

When $\phi(t) = t$, we have that $d_{\mathcal{F},\phi}$ is a distance function [6] , and with slightly abuse of notation we write it simply as $d_{\mathcal{F}}(\mu,\nu) = \sup_{D\in\mathcal{F}} |\mathbb{E}_{x\sim\mu}[D(x)] - \mathbb{E}_{x\sim\nu}[D(x)]|$.

Neural Net Distance (2)

- The formula:

$$d_{\mathcal{F},\phi}(\mu,\nu) = \sup_{D\in\mathcal{F}} \mathop{\mathbb{E}}_{x\sim\mu}[\phi(D(x))] + \mathop{\mathbb{E}}_{x\sim\nu}[\phi(1-D(x))] - 2\phi(1/2)$$

Is heavily based off the F-Divergence and so you can see that they use the same GAN objective function mentioned in *S. Nowozin, B. Cseke, R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. NeurIPS, 2016.*

- The property of φ(t) = t, then $d_F(P, P)$ remains true here.

Examples:

- If φ(t) = log(t) and F is the class of all functions then d_F will be the same as the Jensen-Shannon divergence.

- If φ(t) = t and F is the class of all 1-Lipschitz functions then d_F will be the same as the Wasserstein distance. (Lipschitz refers to the training parameters and indicates that changing a parameter by delta changes the output of the deep net by less than constant * delta).

Neural Net Distance (3)

Essentially, Neural net distance can be defined as D_F as mentioned in the previous two slides.

Taking the results of the longer formula based off F-Divergence, JS Divergence, and the WGAN model distance. We are able to create a simplified equation for Neural Net Distance with certain assumptions.

GAN training uses F to be a class of neural nets with a bound p on the number of parameters. An assumption can be made regarding the measuring function such that it takes in values between $[-\Delta, \Delta]$ and that it is L_ϕ-Lipschitz. Even more so, F is a class of discriminators that is L-Lipschitz to the target distribution where p denotes the number of parameters in the target distribution.

Theorem 3.1. *In the setting of previous paragraph, let μ, ν be two distributions and $\hat{\mu}, \hat{\nu}$ be empirical versions with at least m samples each. There is a universal constant c such that when $m \geq \frac{cp\Delta^2 \log(LL_\phi p/\epsilon)}{\epsilon^2}$, we have with probability at least $1 - \exp(-p)$ over the randomness of $\hat{\mu}$ and $\hat{\nu}$,*

$$|d_{\mathcal{F},\phi}(\hat{\mu}, \hat{\nu}) - d_{\mathcal{F},\phi}(\mu, \nu)| \leq \epsilon.$$

In a more general sense, you get the equation: $\left| d_{\mathcal{F},\phi}(\mathcal{D}_{real}, \mathcal{D}_{G^{(t)}}) - d_{\mathcal{F},\phi}(\hat{\mathcal{D}}_{real}, \hat{\mathcal{D}}_{G^{(t)}}) \right| \leq \varepsilon\,.$

The downside of the NND is that it will return a small value even if u and v are are far from each other. This is due to the capacity being limited by p.

Equilibrium using Mixture of Generators:

Equilibrium in our context is finding a point where the min max problem of the GAN is satisfied.

$$F(u,v) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{real}}[\phi(D_v(x))] + \mathop{\mathbb{E}}_{x\sim\mathcal{D}_G}[\phi(1-D_v(x))].$$

Taking a look at the function, we can isolate the generator and the discriminator to get:

Theorem 4.1 (von Neumann). *There exists a value V, and a pair of mixed strategies $(\mathcal{S}_u,\mathcal{S}_v)$ such that*

$$\forall v, \quad \mathop{\mathbb{E}}_{u\sim\mathcal{S}_u}[F(u,v)] \le V \quad \text{and} \quad \forall u, \quad \mathop{\mathbb{E}}_{v\sim\mathcal{S}_v}[F(u,v)] \ge V.$$

The thing to keep in mind here is that the discriminator is not as important as the generator since the mixture of generators is linear whereas the mixture of discriminator is not. Also the mixture of discriminators is also unable to effectively differentiate between the generated and real distribution. Therefore we after putting the focus on the mixture of generators and accounting for error we get.

Definition 3. A pair of mixed strategies $(\mathcal{S}_u,\mathcal{S}_v)$ is an ϵ-approximate equilibrium, if for some value V

$$\forall v \in \mathcal{V}, \quad \mathop{\mathbb{E}}_{u\sim\mathcal{S}_u}[F(u,v)] \le V+\epsilon;$$
$$\forall u \in \mathcal{U}, \quad \mathop{\mathbb{E}}_{v\sim\mathcal{S}_v}[F(u,v)] \ge V-\epsilon.$$

If the strategies $\mathcal{S}_u,\mathcal{S}_v$ are pure strategies, then this pair is called an ϵ-approximate pure equilibrium.

The paper insists that if the discriminator and generator are deep nets with p parameters then there exists an approximate equilibrium when the mixture size is large enough. How large is good enough?

Defining the Mixture of Generators:

The mixture of generators is defined by the following equation:

Theorem 4.2. *In the settings above, if the generator can approximate any point mass*[7]*, there is a universal constant* $C > 0$ *such that for any* ϵ*, there exists* $T = \frac{C\Delta^2 p \log(LL'L_\phi \cdot p/\epsilon)}{\epsilon^2}$ *generators* $G_{u_1}, \ldots G_{u_T}$*. Let* $\mathcal{S}_u$ *be a uniform distribution on* u_i*, and* D *is a discriminator that outputs only* $1/2$*, then* $(\mathcal{S}_u, D)$ *is an* ϵ*-approximate equilibrium.*

All the assumptions from when the neural net distance formula was defined are applied here.

The understanding of the equation above is:

- The proof above suggests that using a standard probabilistic argument and epsilon net argument to show that if T generators and discriminators are sampled from an infinite mixture, they form an approximate equilibrium with high probability.
- Generators are able to approximate any point mass, so an infinite mixture of generators can approximate the real distribution accurately. Therefore:
  a mixture of $O(p)$ generators can achieve an ϵ-approximate equilibrium. This works for many measuring functions (φ) as long as they are concave.

Pure Equilibrium:

Pure equilibrium follows the same process of defining the mixture generator.

Pure equilibrium is the most desired course of action simply because it means both sides of game are being played optimally to ensure the best payoff result.

The formula for the base is the same except that the pure equilibrium the architecture of the generator nets changes a bit. We move to nets with a size of p^2.

Theorem 4.3. *Suppose the generator and discriminator are both k-layer neural networks ($k \geq 2$) with p parameters, and the last layer uses ReLU activation function. In the setting of Theorem 4.2, there exists $k+1$-layer neural networks of generators G and discriminator D with $O\left(\frac{\Delta^2 p^2 \log(LL'L_\phi \cdot p/\epsilon)}{\epsilon^2}\right)$ parameters, such that there exists an ϵ-approximate pure equilibrium with value $2\phi(1/2)$.*

MIX-GANs

A mix GAN is a GAN that makes use of the idea of a mixture of generators within reason. So they make use of a mixture of T components, where T is constrained by GPU memory (< 5). Maintain T generators and T discriminators. All the respective weights for each generator is maintained and everything is trained through backwards propagation.

$$w_{u_i} = \frac{e^{\alpha_{u_i}}}{\sum_{k=1}^{T} e^{\alpha_{u_k}}}, \quad i \in [T]$$

It is necessary to define the weights so we can see the payoff function for the MIX+GAN. Which boils down to:

$$\min_{\{u_i\},\{\alpha_{u_i}\}} \max_{\{v_j\},\{\alpha_{v_j}\}} \mathop{\mathbb{E}}_{i,j\in[T]} F(u_i, v_j)$$
$$= \min_{\{u_i\},\{\alpha_{u_i}\}} \max_{\{v_j\},\{\alpha_{v_j}\}} \sum_{i,j\in[T]} w_{u_i} w_{v_j} F(u_i, v_j).$$

Some modifications can be made for the purpose of computing, for example empirically discouraging distant weights:

$$R_{ent}(\{w_{u_i}\}, \{w_{v_i}\}) = -\frac{1}{T} \sum_{i=1}^{T} (\log(w_{u_i}) + \log(w_{v_i}))$$

Experiments:

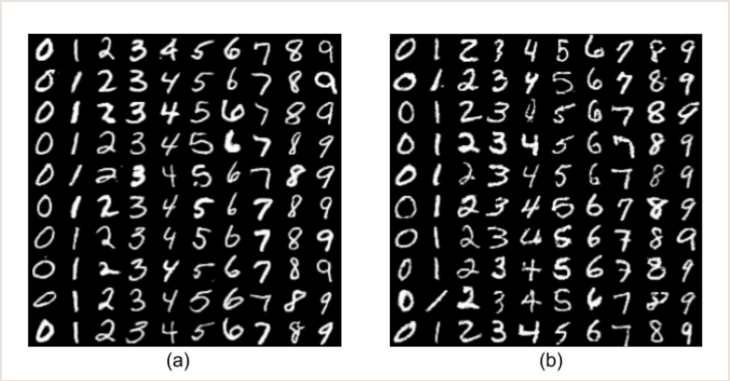
Table 1: **Inception Scores on CIFAR-10.** Mixture of DCGANs achieves higher score than any single-component DCGAN does. All models except for WassersteinGAN variants are trained with labels.

Method	Score
SteinGAN [Wang and Liu, 2016]	6.35
Improved GAN [Salimans et al., 2016]	8.09±0.07
AC-GAN [Odena et al., 2016]	8.25 ± 0.07
S-GAN (best variant in [Huang et al., 2017])	8.59± 0.12
DCGAN (as reported in Wang and Liu [2016])	6.58
DCGAN (best variant in Huang et al. [2017])	7.16±0.10
DCGAN (5x size)	7.34±0.07
MIX+DCGAN (Ours, with 5 components)	7.72±0.09
Wasserstein GAN	3.82±0.06
MIX+WassersteinGAN (Ours, with 5 components)	4.04±0.07
Real data	11.24±0.12

Table1 is run CIFAR10. The MIX variations perform better than all the base models.

Another key thing to note is that the MIX-GAN models are not better simply because they have more parameters. The DCGAN labeled 5x has 5 times as many parameters at the MIX + DCGAN but still performs worse.

Additional Experiments:



The a side is the MIX-DCGAN and the b side is the regular DCGAN. There are minor differences between the two but mainly the a side is more clear.

Summary:

The paper solved the issue regarding other GAN models failing to generalize by introducing a new GAN concept and a new distance metric.

Ideally for a GAN if a pure equilibrium exists that would be great for finding a good generalization point, however; it may not always exist and so the invention of the MIX-GAN.

MIX-GAN = small mixture of generators and discriminators and it is seen that it is able to improve the quality of certain existing GAN models.