Jonathan Hui Follow

Jun 21, 2018 · 13 min read

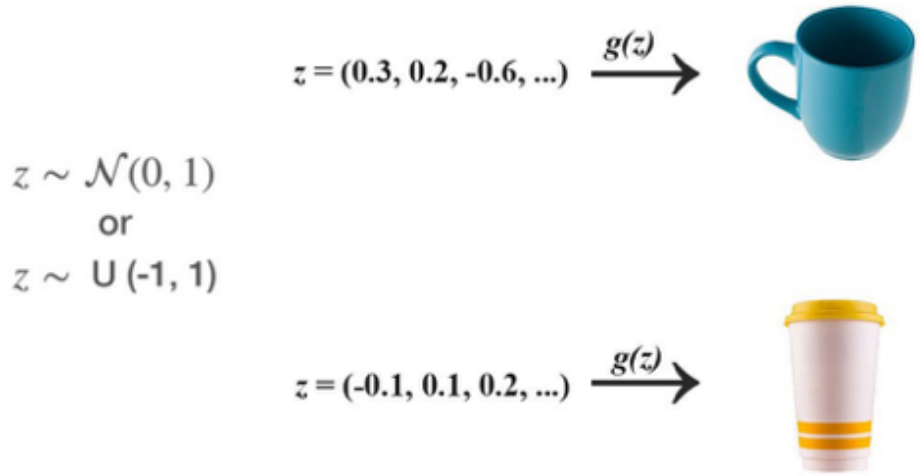
GAN — Why it is so hard to train Generative Adversarial Networks!

It is easier to recognize a Monet's painting than drawing one. Generative models (creating data) are considered much harder comparing with the discriminative models (processing data). Training GAN is also hard. This article is part of the GAN series and we will investigate why the training is so elusive. Through the study, we understand some fundamental problems that drive the directions of many researchers. We will look into some disagreements so we know where the research may head to. Before

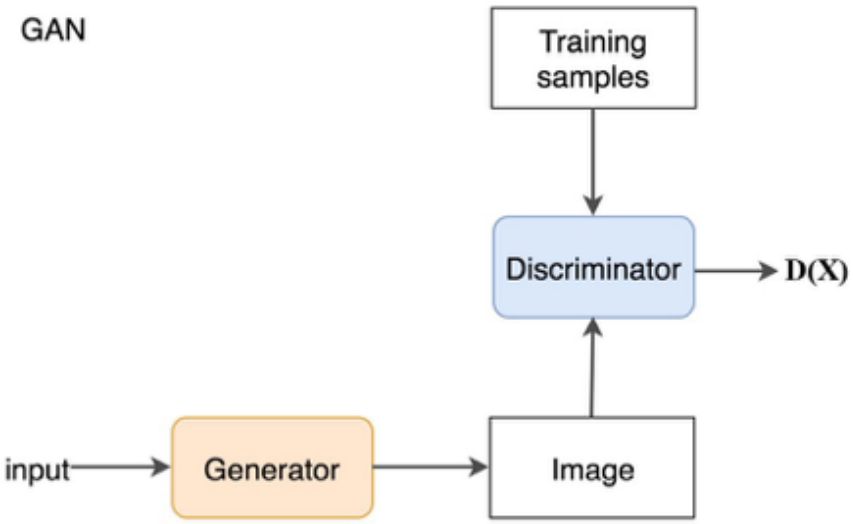
looking into the problems, let's have a quick recap on some of the GAN equations.

GAN

GAN samples noise ***z*** using normal or uniform distribution and utilizes a deep network generator ***G*** to create an image ***x (x=G(z))***.



In GAN, we add a discriminator to distinguish whether the discriminator input is real or generated. It outputs a value ***D(x)*** to estimate the chance that the input is real.

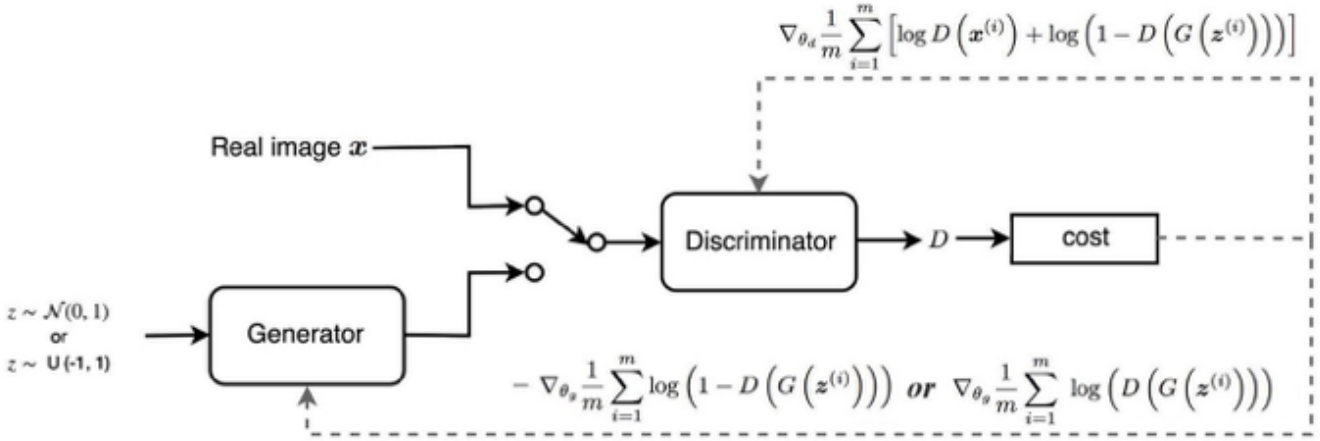


Objective function and gradients

GAN is defined as a minimax game with the following objective function.

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

The diagram below summarizes how we train the discriminator and the generator using the corresponding gradient.



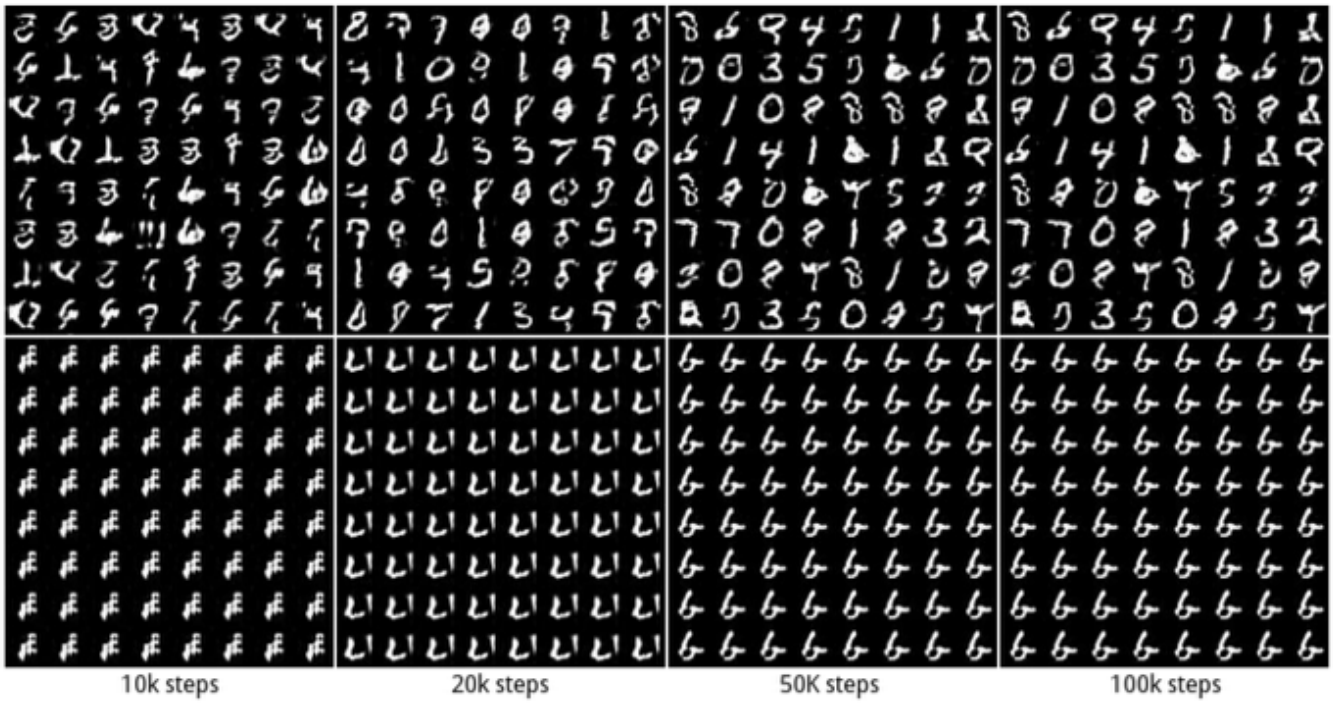
GAN Problems

Many GAN models suffer the following major problems:

- **Non-convergence**: the model parameters oscillate, destabilize and never converge,
- **Mode collapse**: the generator collapses which produces limited varieties of samples,
- **Diminished gradient**: the discriminator gets too successful that the generator gradient vanishes and learns nothing,
- Unbalance between the generator and discriminator causing overfitting, &
- Highly sensitive to the hyperparameter selections.

Mode

Real-life data distributions are multimodal. For example, in MNIST, there are 10 major **modes** from digit '0' to digit '9'. The samples below are generated by two different GANs. The top row produces all 10 modes while the second row creates a single mode only (the digit "6"). This problem is called **mode collapse** when only a few modes of data are generated.



Source

Nash equilibrium

GAN is based on the zero-sum non-cooperative game. In short, if one wins the other loses. A zero-sum game is also called minimax. Your opponent wants to maximize its actions and your actions are to minimize them. In game theory, the GAN model converges when the discriminator and the generator reach a Nash equilibrium. This is the optimal point for the minimax equation below.

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

Since both sides want to undermine the others, a Nash equilibrium happens when one player will not change its action regardless of what the opponent may do. Consider two player A and B which control the value of $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively. Player A wants to maximize the value $\boldsymbol{xy}$ while $\boldsymbol{B}$

wants to minimize it.

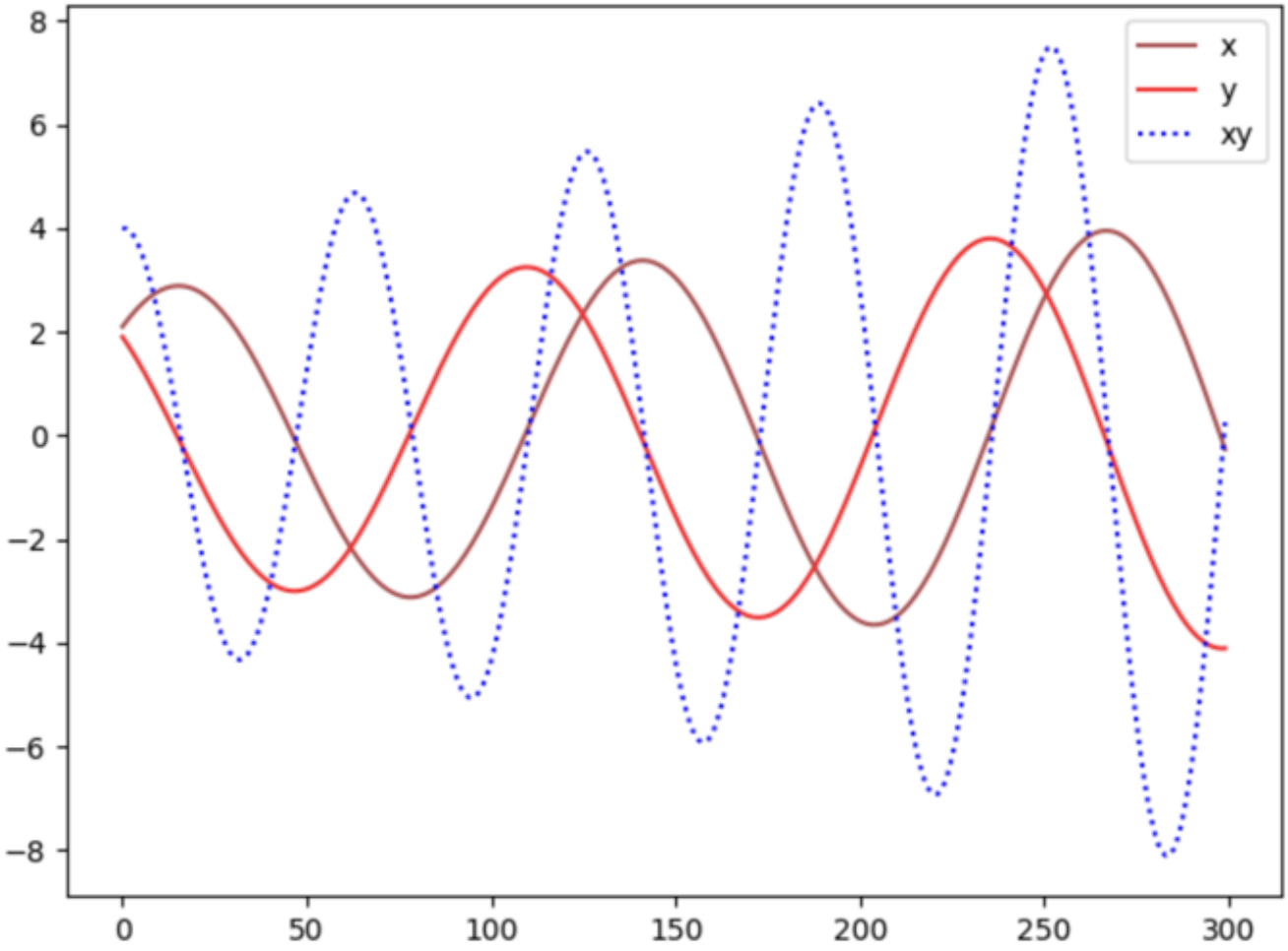
$$\min_{B} \max_{A} V(D, G) = xy$$

The Nash equilibrium is $x=y=0$. This is the only state where the action of your opponent does not matter. It is the only state that any opponents' actions will not change the game outcome.

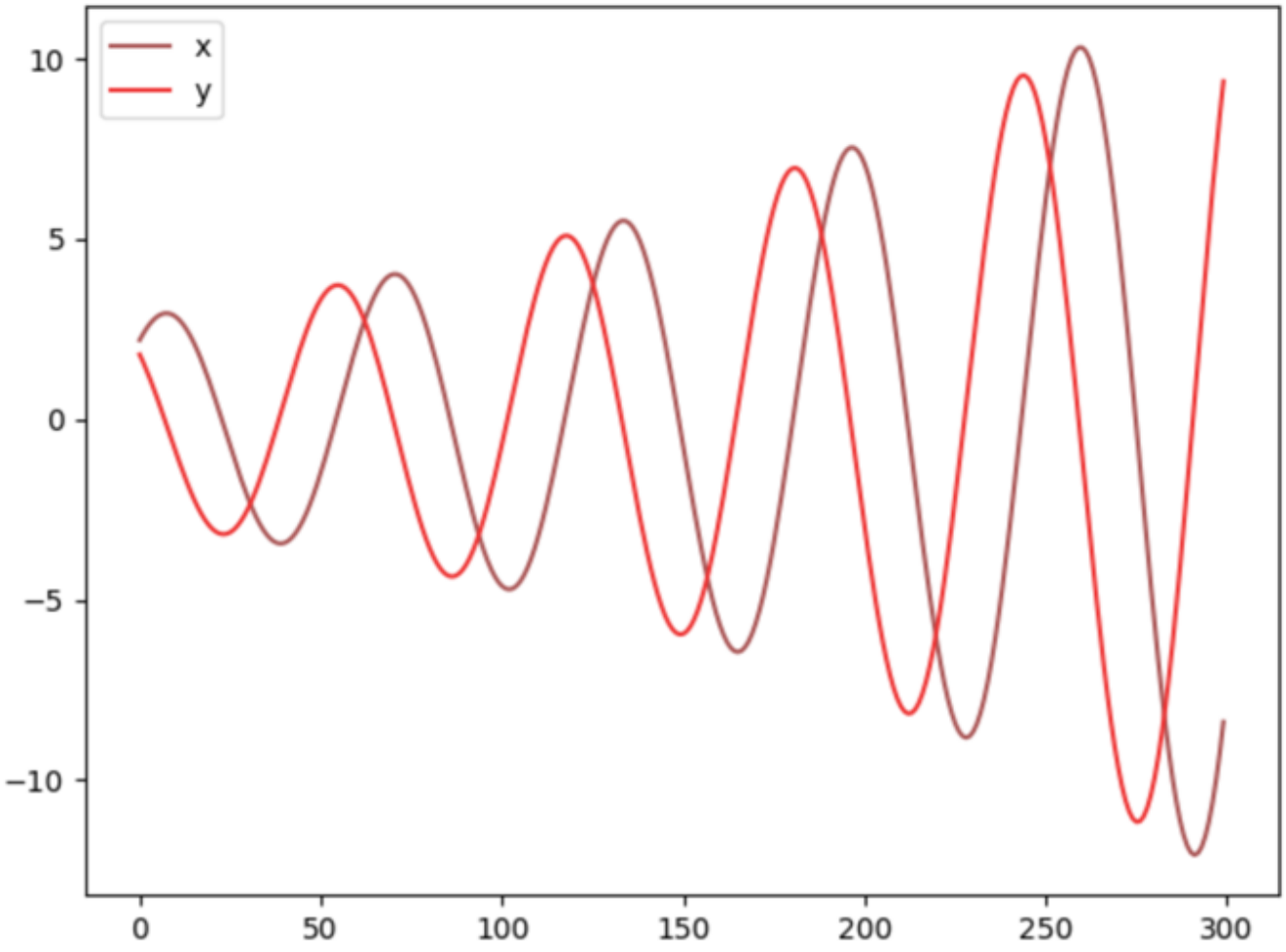
Let's see whether we can find the Nash equilibrium easily using the gradient descent. We update the parameter x and y based on the gradient of the value function V.

$$\Delta x = \alpha \frac{\partial(xy)}{\partial(x)}$$
$$\Delta y = - \alpha \frac{\partial(xy)}{\partial(y)}$$

where α is the learning rate. When we plot x, y, and xy against the training iterations, we realize our solution does not converge.



If we increase the learning rate or train the model longer, we can see the parameters *x, y* is unstable with big swings.



Our example is an excellent showcase that some cost functions will not converge with gradient descent, in particular for a non-convex game. We can also view this issue in an intuitive way: your opponent always countermeasures your actions which makes the models harder to converge.

> *Cost functions may not converge using gradient descent in a minimax game.*

Generative model with KL-Divergence

To understand the convergence issue in GAN, we will study the KL-

divergence and the JS-divergence first. Before GAN, many generative models create a model $\boldsymbol{\theta}$ that maximizes the Maximum Likelihood Estimation **MLE**. i.e. finding the best model parameters that fit the training data the most.

$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{N} p(x_i|\theta)$$

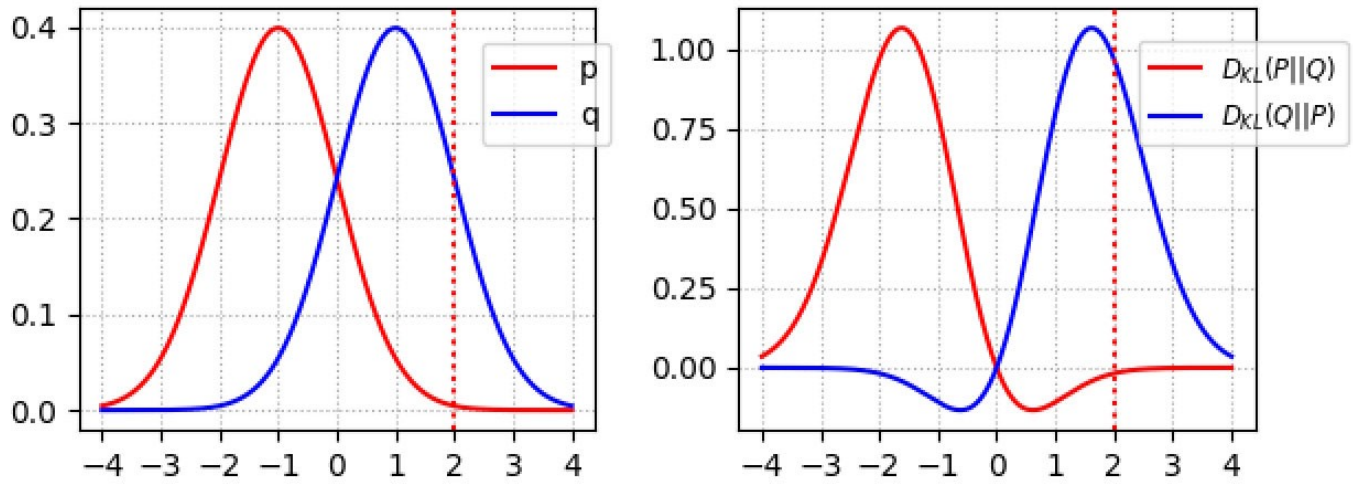
This is the same as minimizing the KL-divergence $\boldsymbol{KL(p,q)}$ (proof) which measures how the probability distribution $\boldsymbol{q}$ (estimated distribution) diverges from the expected probability distribution $\boldsymbol{p}$ (the real-life distribution).

$$D_{KL}(p||q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$

KL-divergence is not symmetrical.

$$D_{KL}(p||q) \neq D_{KL}(q||p)$$

$\boldsymbol{KL(x)}$ drops to 0 for area where $\boldsymbol{p(x) \rightarrow 0}$. For example, in the figure on the right below, the red curve corresponds to $D(p, q)$. It drops to zero when $\boldsymbol{x>2}$ where $\boldsymbol{p}$ approaches 0.



Note: KL(p, q) is the integral of the red curve in the right.

What is the implication? The KL-divergence ***DL(p, q)*** penalizes the generator if it misses some modes of images: the penalty is high where $p(x) > 0$ but $q(x) \rightarrow 0$. Nevertheless, it is acceptable that some images do not look real. The penalty is low when $p(x) \rightarrow 0$ but $q(x) > 0$. **(Poorer quality but more diverse samples)**

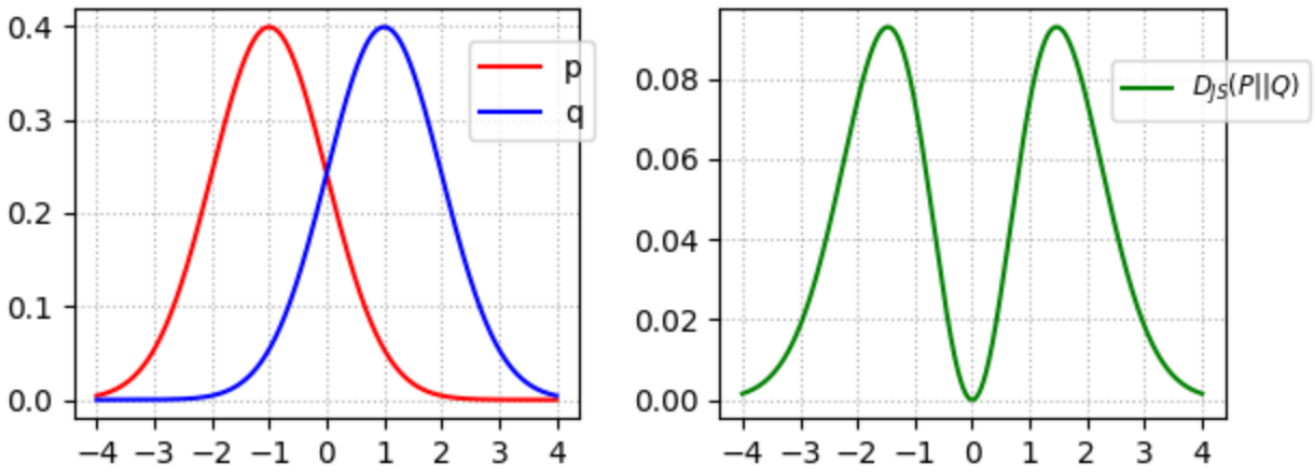
On the other hand, the reverse KL-divergence ***DL(q, p)*** penalizes the generator if the images do not look real: high penalty if $p(x) \rightarrow 0$ but $q(x) > 0$. But it explores less variety: low penalty if $q(x) \rightarrow 0$ but $p(x) > 0$. **(Better quality but less diverse samples)**

Some generative models (other than GANs) use MLE (a.k.a KL-divergence) to create models. It was originally believed that KL-divergence causes poorer quality of images (blurry images). But be warned that some empirical experiments may have disputed this claim.

JS-Divergence

JS-divergence is defined as:

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q||\frac{p+q}{2})$$



JS-divergence is symmetrical. Unlike KL-divergence, it will penalize poor images badly. (when $p(x) \rightarrow 0$ and $q(x) > 0$) In GAN, if the discriminator is optimal (performing well in distinguishing images), the generator's objective function becomes (proof):

$$\min_G V(D^*, G) = 2D_{JS}(p_r \| p_g) - 2\log 2$$

So optimizing the generator model is treated as optimizing the JS-divergence. In experiments, GAN produces nicer pictures comparing to other generative models that use KL-divergence. Follow the logic in the last section, early research speculates that optimizing JS-divergence, rather than KL-divergence, creates better but less diverse images. However, some researchers have since retracted those claims because

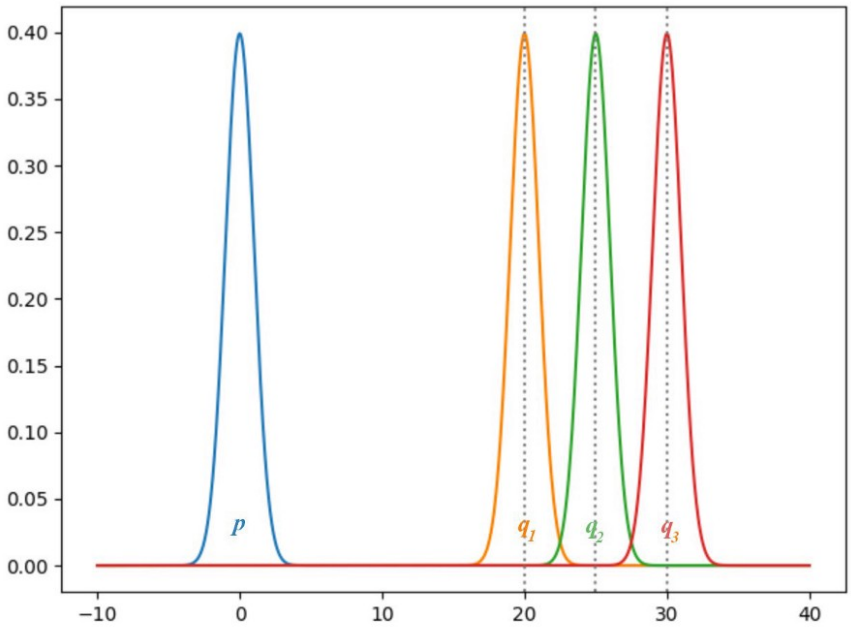
experiments with GAN using MLE produces similar image quality but still suffer image diversity problem. But significant efforts have already taken place in studying the weakness of JS-Divergence in GAN training. These works are significant regardless of the debates. Therefore, we will dig deeper into the issues of the JS-divergence next.

Vanishing gradients in JS-Divergence

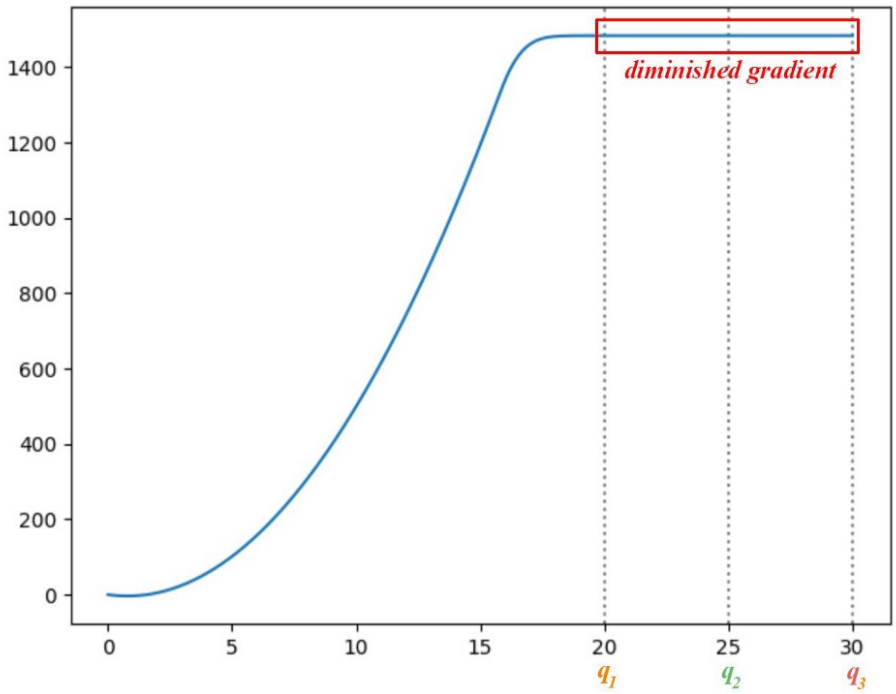
Recall that when the discriminator is optimal, the objective function for the generator is:

$$\min_G V(D^*, G) = 2D_{JS}(p_r \| p_g) - 2\log 2$$

What happens to the JS-divergence gradient when the data distribution $\boldsymbol{q}$ of the generator's images does not match with the ground truth $\boldsymbol{p}$ for the real images. Let's consider an example in which $\boldsymbol{p}$ and $\boldsymbol{q}$ are Gaussian distributed and the mean of $\boldsymbol{p}$ is zero. Let's consider $\boldsymbol{q}$ with different means to study the gradient of $JS(p, q)$.



Here, we plot the JS-divergence *JS(p, q)* between *p* and *q* with means of *q* ranging from 0 to 30. As shown below, the gradient for the JS-divergence vanishes from *q1* to *q3*. The GAN generator will learn extremely slow to nothing when the cost is saturated in those regions. In particular, in early training, *p* and *q* are very different and the generator learns very slow.



Unstable gradients

Because of the vanishing gradient, an **alternative cost function** is proposed by the original GAN paper to address the gradient vanishing problem.

$$\nabla_{\theta_g} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \rightarrow 0 \quad \textit{change to} \quad \nabla_{\theta_g} - \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

The corresponding gradient according to a different research paper from Arjovsky is:

$$\mathbb{E}_{z\sim p(z)}\left[-\nabla_\theta \log D^*(g_\theta(z))|_{\theta=\theta_0}\right] = \nabla_\theta\left[KL(\mathbb{P}_{g_\theta}\|\mathbb{P}_r) - 2JSD(\mathbb{P}_{g_\theta}\|\mathbb{P}_r)\right]|_{\theta=\theta_0}$$

It includes a **reverse** KL-divergence term which Arjovsky uses it to explain why GAN has higher quality but less diverse image comparing to generative models based on KL-divergence. But the same analysis claims that the gradients fluctuate and cause instability to the model. To illustrate the point, Arjovsky freezes the generator and trains the discriminator continuously. The gradient for the generator starts increasing with larger variants.

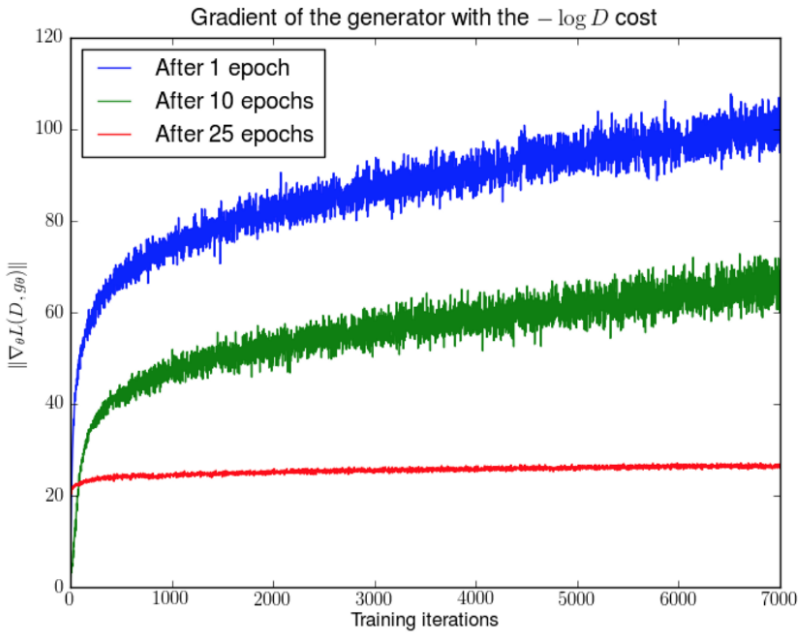


Figure 3: First, we trained a DCGAN for 1, 10 and 25 epochs. Then, with the generator fixed we train a discriminator from scratch and measure the gradients with the $-\log D$ cost function. We see the gradient norms grow quickly. Furthermore, the noise in the curves shows that the variance of the gradients is also increasing. All these gradients lead to updates that lower sample quality notoriously.

Source

The experiment above is not how we train GAN. However, mathematically, Arjovsky shows the first GAN generator's objective function has vanishing gradients and the alternative cost function has fluctuating gradients that cause instability to the models. Since the original GAN paper, there is a gold rush in finding new cost functions, like LSGAN, WGAN, WGAN-GP, BEGAN etc… Some methods are based on new mathematical models and others based on intuition back up by experiments. The objective is to find a cost function with smoother and non-vanishing gradients.

However, a 2017 Google Brain paper "Are GANs Created Equal?" claims that

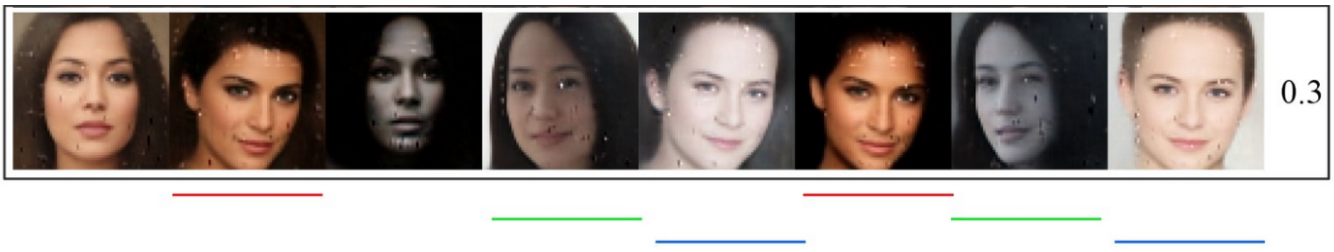
> *Finally, we did not find evidence that any of the tested algorithms*

> *consistently outperforms the original one.*

If any new proposed cost functions deliver a knockout punch in improving image quality, we will not have this debate. The doomsday picture on the original cost functions from the Arjovsky's mathematical model does not fully materialize either. But I will cautious readers of prematurely declare cost functions does not matter. My thoughts on the Google Brain paper can be find here. **What is my take?** Training GAN fails easily. Instead of trying out many cost functions at the beginning, debug your design and code first. Next try hard to tune the hyperparameter because GAN models are sensitive to them. Do it before trying cost functions randomly.

Why mode collapse in GAN?

Mode collapse is one of the hardest problems to solve in GAN. A complete collapse is not common but a partial collapse happens often. The images below with the same underlined color look similar and the mode starts collapsing.



Modified from source

Let's see how it may occur. The objective of the GAN generator is to create images that can fool the discriminator ***D*** the most.

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

But let's consider one extreme case where ***G*** is trained extensively without updates to ***D***. The generated images will converge to find the optimal image ***x**** that fool ***D*** the most, the most realistic image from the discriminator perspective. In this extreme, ***x**** will be independent of ***z***.

$$x^* = argmax_x D(x)$$

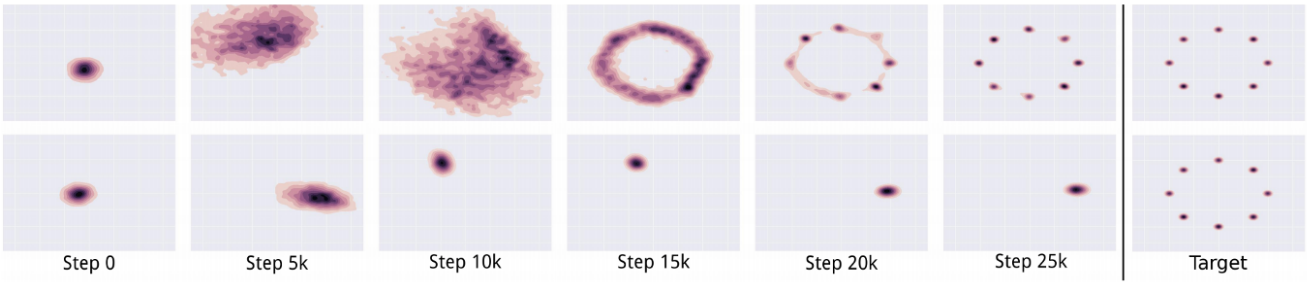
This is bad news. The mode collapses to a **single point**. The gradient associated with ***z*** approaches zero.

$$\frac{\partial J}{\partial z} \approx 0$$

When we restart the training in the discriminator, the most effective way to detect generated images is to detect this single mode. Since the generator desensitizes the impact of ***z*** already, the gradient from the discriminator will likely push the single point around for the next most vulnerable mode. This is not hard to find. The generator produces such an imbalance of modes in training that it deteriorates its capability to detect others. Now, both networks are overfitted to exploit short-term opponent weakness. This turns into a cat-and-mouse game and the model will not converge.

In the diagram below, the Unroll GAN manages to produce all 8 expected modes of data. The second row shows another GAN which the mode

collapses and rotates to another mode when the discriminator catches up.



Source

During training, the discriminator is constantly updated to detect adversaries. Therefore, the generator is less likely to be overfitted. In practice, our understanding of mode collapse is still limited. Our intuitive explanation above is likely oversimplified. Mitigation methods are developed and verified with empirical experiments. However, GAN training is still a heuristic process. Partial collapse is still common.

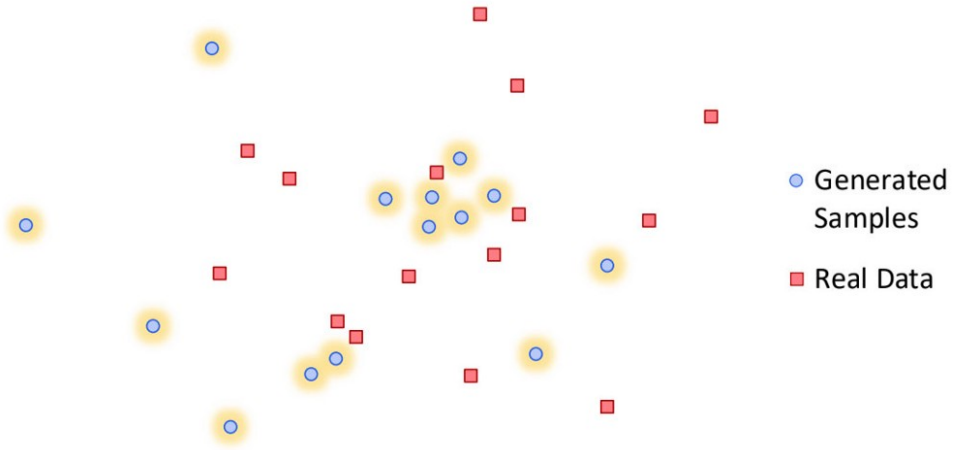
But mode collapse is not all bad news. In style transfer using GAN, we are happy to convert one image to just a good one, rather than finding all variants. Indeed, the specialization in the partial mode collapse sometimes creates higher quality images.

> *But mode collapse remains one of the most important issues to be solved for GAN.*

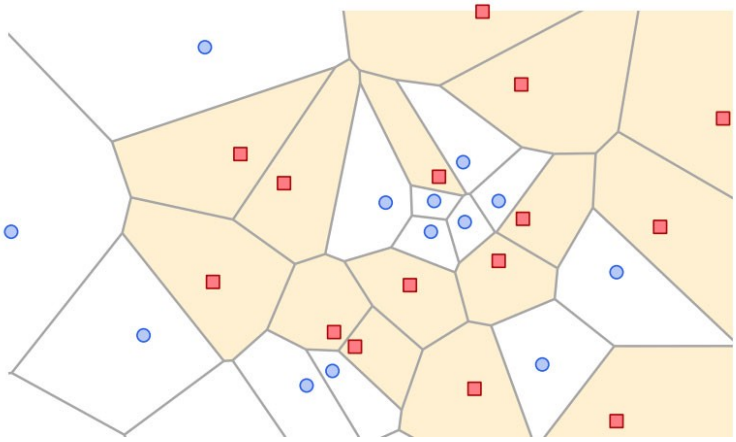
Implicit Maximum Likelihood Estimation (IMLE)

(Credit: Diagrams in this section are originated or modified from the IMLE presentation.)

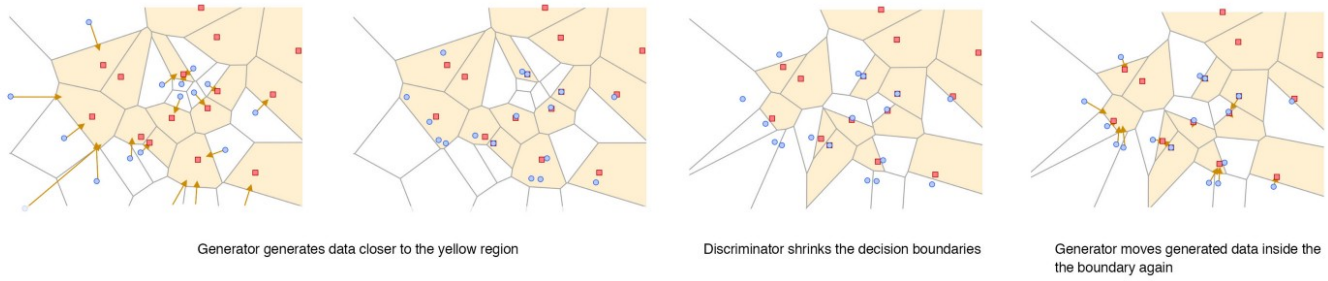
After this article is first written, a new research paper is published in explaining and solving the mode collapsing problem. Let's consider the red squares below are the real data and the blue as the generated samples.



The GAN discriminator creates regions in yellow to distinguish the real data away from the generated data.



In the training process, the generator will generate data to move generated samples towards the decision boundary while the discriminator will shrink the boundary further.

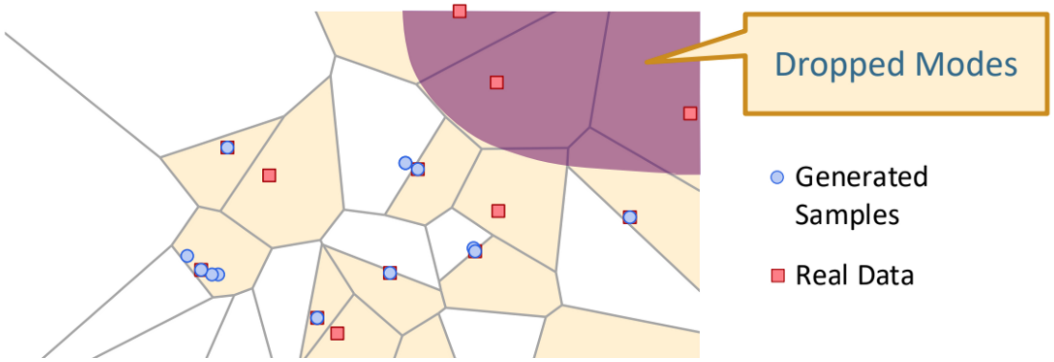


Generator generates data closer to the yellow region

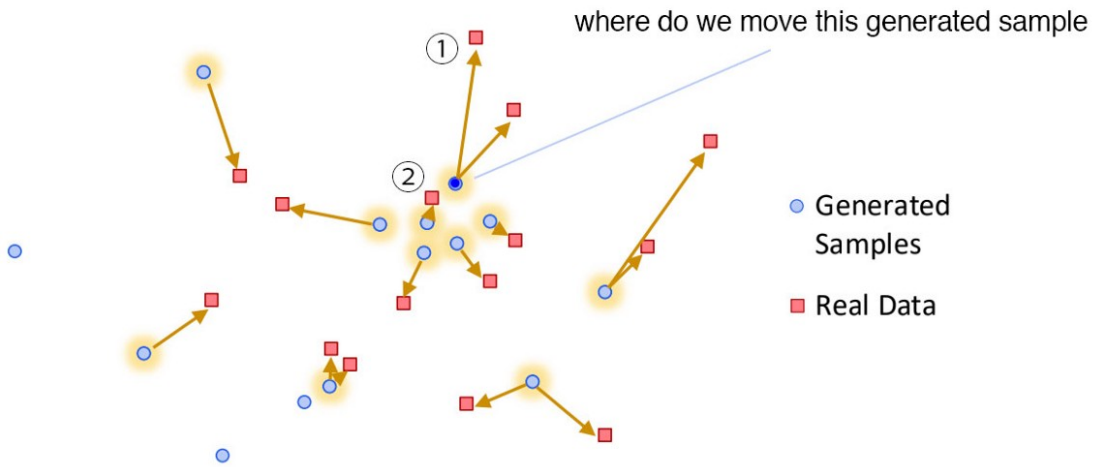
Discriminator shrinks the decision boundaries

Generator moves generated data inside the the boundary again

But the generator has no guarantee that it will generate samples that cover every mode in the process. As shown in the example, some modes can be left out in the process and cannot be recovered.



IMLE flips the mechanism the other way round. In GAN, we move generated samples toward the closest boundary. For example, we move the blue dot below towards ②. But in IMLE, for each real data, we ask what is the closet generated sample. So we train the model to move the blue dot to ① instead.



IMLE is not a GAN model. So we will not elaborate it here further.

Hyperparameters & training

No cost functions will work without good hyperparameters and tune them takes time and a lot of patience. New cost functions may introduce hyperparameter(s) that has sensitive performance.

> *Hyperparameter tuning needs patience. No cost functions will work without spending time on the hyperparameter tuning.*

Balance between the discriminator and generator

The non-convergence and mode collapse is often explained as an imbalance between the discriminator and the generator. The obvious solution is to balance their training to avoid overfitting. However, very few progress has made but not because of the lack of trying. Some researchers believe that this is not a feasible or desirable goal since a good discriminator gives good feedback. Some of the attention is therefore shifted for cost functions with non-vanishing gradients instead.

Cost v.s. image quality

In a discriminative model, the loss measures the accuracy of the prediction and we use it to monitor the progress of the training. However, the loss in GAN measures how well we are doing compared with our opponent. Often, the generator cost increases but the image quality is actually improving. We fall back to examine the generated images manually to verify the progress. This makes model comparison harder which leads to difficulties in picking the best model in a single run. It also complicates the tuning process.

Further reading

Now you hear the problems and you may want to hear the solutions. We provide two different articles. The first one provides key summaries on the solutions.

GAN — A comprehensive review into the gangsters of GANs (Part 2)

This article studies the motivation and the direction of the GAN research in improving GANs. By reviewing them in a...

medium.com

If you want to be much deeper, the second one will have more in-depth discussion:

GAN — Ways to improve GAN performance

GAN models can suffer badly in the following areas comparing to other deep networks.

medium.com

If you want to study the mathematical model further on the gradient and stability problem, the following article will elaborate on it. But be warned, the equations may look overwhelming. However, if you are not afraid of equations, it gives nice reasoning on some of their claims.

GAN — What is wrong with the GAN cost function?

We work hard to produce mathematical models for deep learning. But often, we are not successful and fall back to...

medium.com

Reference

Towards principled methods for training Generative Adversarial Networks

Improved techniques for training GANs

NIPS 2016 Tutorial: Generative Adversarial Networks

Generative Adversarial Networks

Are GANs created equal? A large-scale study

Implicit Maximum Likelihood Estimation