Generative Adversarial Networks

Speaker: Mingde Wang

Tutor: Professor Qiao Yan


望月
中秋节快乐！
Happy mid-Autumn Festival

Question

How to use the online anime image to make the computer generate indistinguishable anime avatar?

http://qiita.com/mattya/items/e5bfe5e04b9d2f0bbd47

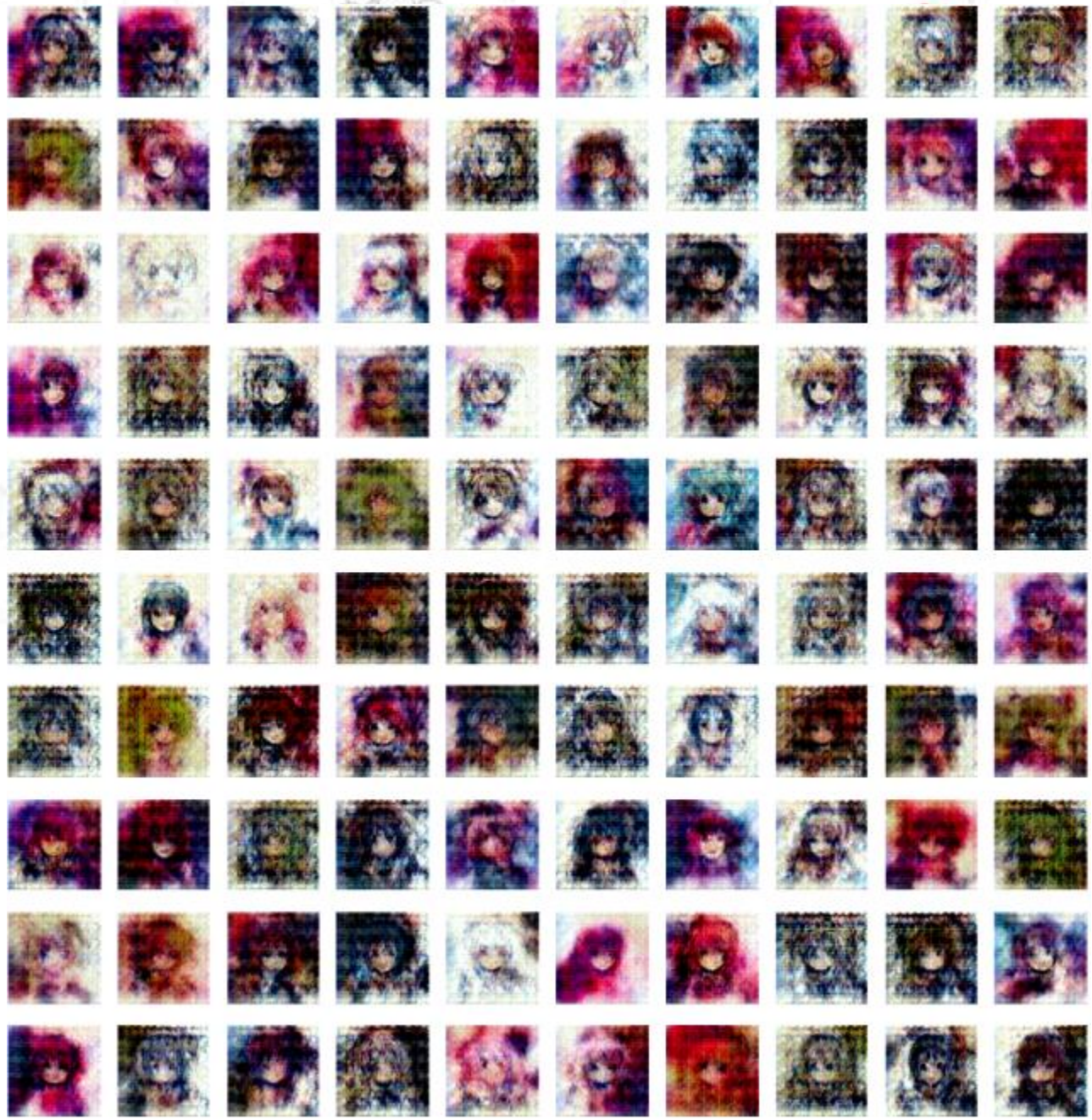
Solution

Naturally, we think of using machine learning to let the machine learn to draw.

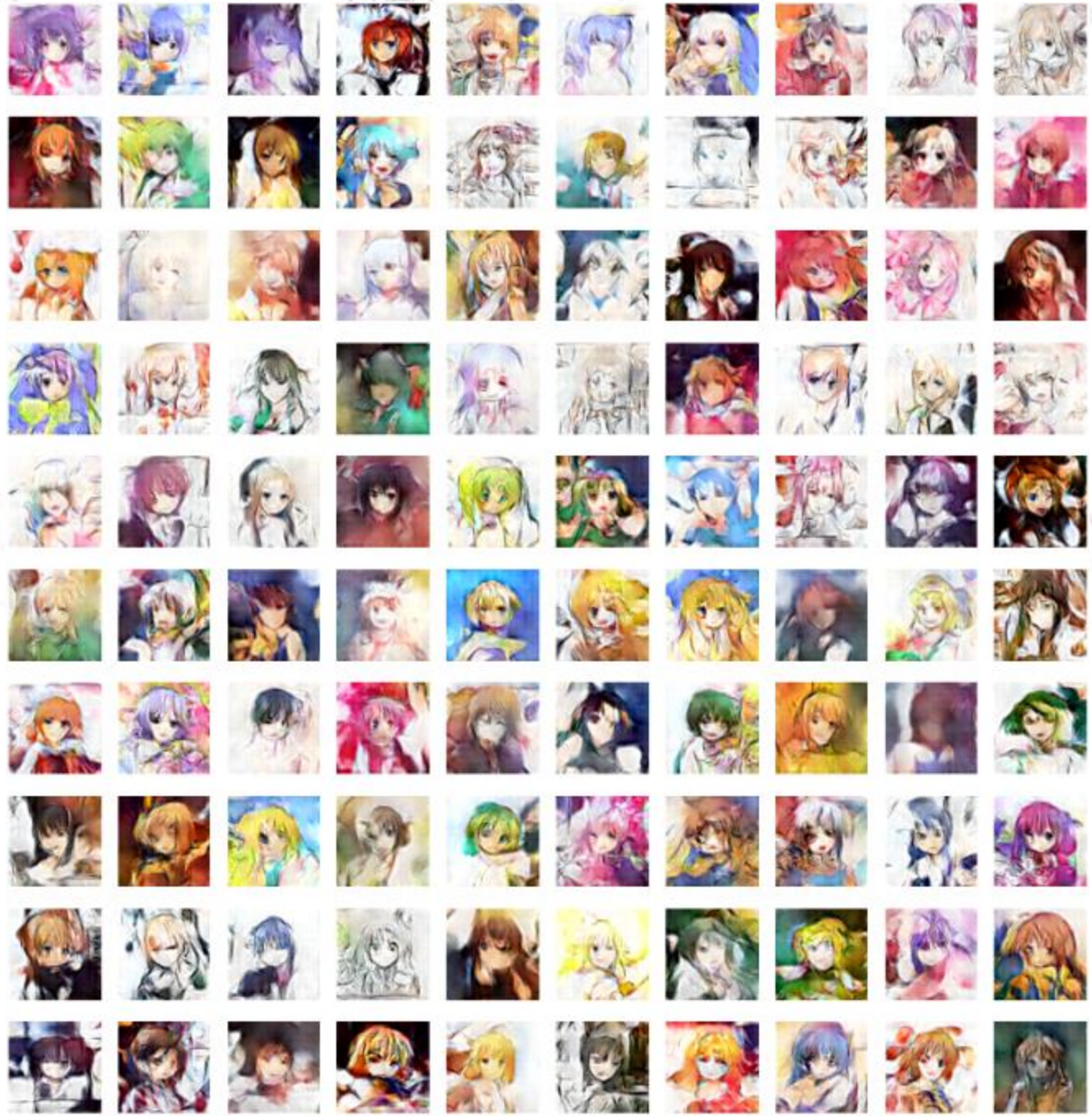
The general methods of machine learning can't achieve results, so some people have designed a new network architecture for this.

In 2014, Ian J. Goodfellow et al. proposed a new framework for estimating generative models via an adversarial process, which is **Generative Adversarial Networks（GAN）**.

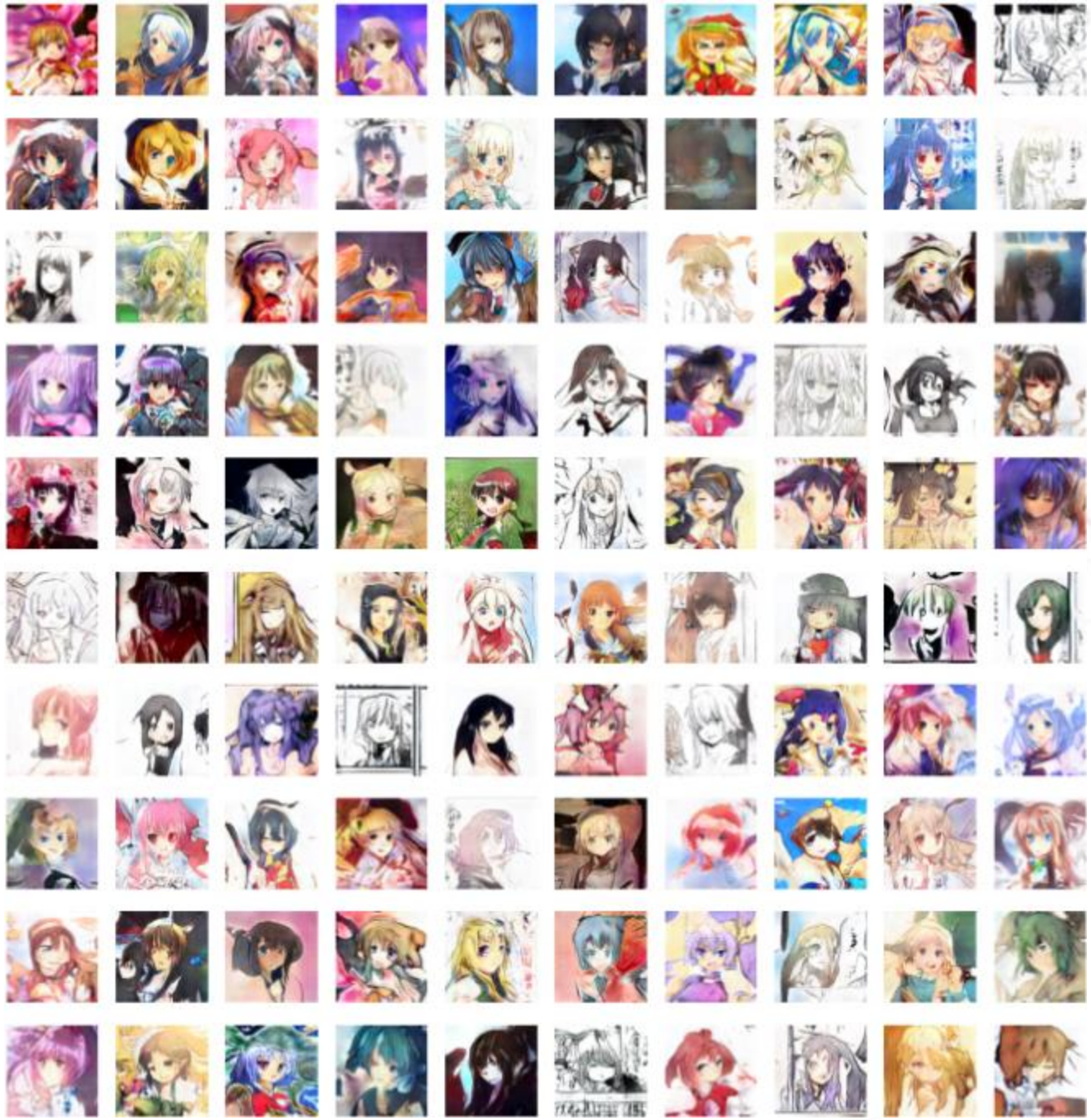
30 minute later



2 hours later



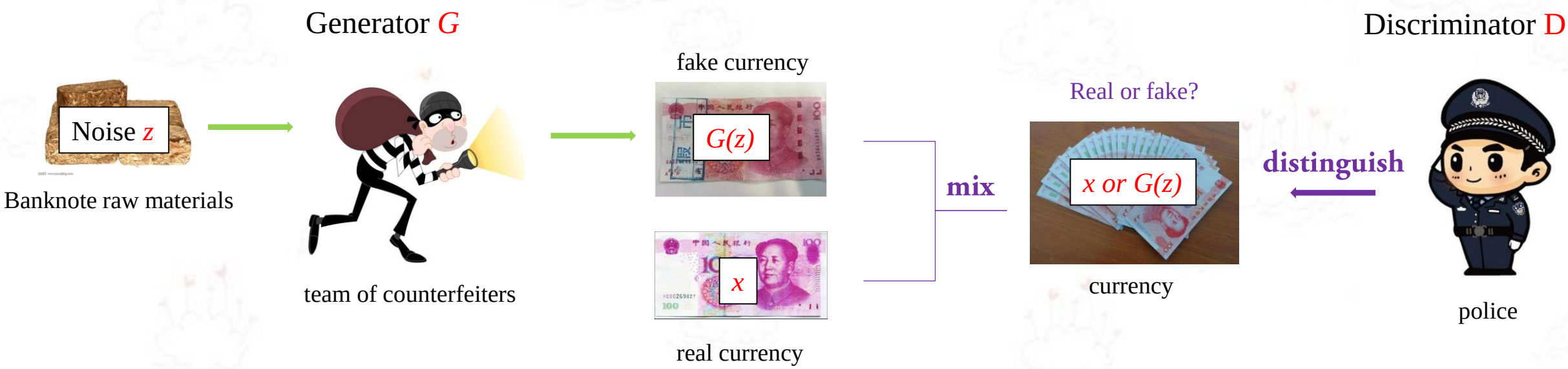
After one day



https://www.saluzi.com/t/google-ai/27020
http://mattya.github.io/chainer-DCGAN/

Symbolization

GAN



- D learns to determine whether a currency is from the fake currency or the real currency.
- G try to produce fake currency and use it without detection.
- Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.

In other words, D and G play the following two-player minimax game with value function *V (G,D)*:

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

P_g and P_{data} represent the distribution of the samples *G(z)* and real data *x*, respectively.

So this minimax game has a global optimum for $P_g = P_{data}$.

For *G* fixed, the optimal discriminator *D* is

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

$if\ P_{data}(x) = 0, P_g(x) \neq 0, D_G^*(x) = 0.$ fake

$if\ P_{data}(x) \neq 0, P_g(x) = 0, D_G^*(x) = 1.$ real

collapse mode

In 2016, Ian J. Goodfellow et al. find that one of the main failure modes for GAN is for the generator to collapse to a parameter setting where it always emits the same point. ----**"Improved Techniques for Training GANs"**

When collapse to a single mode is imminent, the gradient of the discriminator may point in similar directions for many similar points.

Because the discriminator processes each example independently, there is no coordination between its gradients, and thus no mechanism to tell the outputs of the generator to become more dissimilar to each other.

Instead, all outputs race toward a single point that the discriminator currently believes is highly realistic.

In 2016, Martin Arjovsky et al. pushed a set of formula theorems in "**Towards Principled Methods for Training Generative Adversarial Networks**" to theoretically analyze the problem of the original GAN.

Review

In the original GAN, the discriminator should minimize the following loss function, and divide the real sample into positive examples, and generate samples into negative examples.

$$-\mathbb{E}_{x\sim P_r}[\log D(x)] - \mathbb{E}_{x\sim P_g}[\log(1-D(x))] \tag{1}$$

For generator, the loss function is

$$\mathbb{E}_{x\sim P_g}[\log(1-D(x))] \tag{2}$$

or $$\mathbb{E}_{x\sim P_g}[-\log D(x)] \tag{3}$$

Problem 1: The training is unstable.

In one words: Why do updates get worse as the discriminator gets better?

First, we can get from Equation 1 what the optimal discriminator D should be when the generator was fixed. For a specific sample, it may come from the real distribution or from the generated distribution, and its contribution to the loss function of Equation 1 is

$$-P_r(x)\log D(x) - P_g(x)\log[1 - D(x)]$$

Let derivatives for *D(x)* be 0, we get

$$-\frac{P_r(x)}{D(x)} + \frac{P_g(x)}{1 - D(x)} = 0$$

Simplified, the optimal discriminator D is

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)} \tag{4}$$

When the discriminator is optimal, what does the generator's loss function become?

Add a term that does not depend on the generator to Equation 2, making it

$$\mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1 - D(x))]$$

Obtained by formula 4,

$$\mathbb{E}_{x\sim P_r} \log \frac{P_r(x)}{\frac{1}{2}[P_r(x) + P_g(x)]} + \mathbb{E}_{x\sim P_g} \log \frac{P_g(x)}{\frac{1}{2}[P_r(x) + P_g(x)]} - 2\log 2 \qquad (5)$$

In addition, we introduced Kullback-Leibler (KL) divergence and Jensen-Shannon (JS) divergence,

$$KL(P_1||P_2) = \mathbb{E}_{x\sim P_1} \log \frac{P_1}{P_2}$$

$$JS(P_1||P_2) = \frac{1}{2}KL(P_1||\frac{P_1 + P_2}{2}) + \frac{1}{2}KL(P_2||\frac{P_1 + P_2}{2})$$

Formula 5 be,

$$2JS(P_r||P_g) - 2\log 2$$

When the discriminator is optimal, we can equalize the loss function of the generator to minimize the JS divergence between Pr and Pg.

What is the JS divergence? There are only four cases for x,

$P_1(x) = 0$ 且 $P_2(x) = 0$ → JS=0

$P_1(x) \neq 0$ 且 $P_2(x) \neq 0$

$P_1(x) = 0$ 且 $P_2(x) \neq 0$ → JS=log2

$P_1(x) \neq 0$ 且 $P_2(x) = 0$ → JS=log2

The point x is located in the portion where P_r and P_g overlap.

In fact, The only way this can happen is if the distributions are not continuous, or they have disjoint supports. Because their supports lie on low dimensional manifolds.

Lemma 1. *Let $g : \mathcal{Z} \to \mathcal{X}$ be a function composed by affine transformations and pointwise nonlinearities, which can either be rectifiers, leaky rectifiers, or smooth strictly increasing functions (such as the sigmoid, tanh, softplus, etc). Then, $g(\mathcal{Z})$ is contained in a countable union of manifolds of dimension at most* $\dim \mathcal{Z}$*. Therefore, if the dimension of $\mathcal{Z}$ is less than the one of $\mathcal{X}$, $g(\mathcal{Z})$ will be a set of measure 0 in $\mathcal{X}$.*

When the support of P_r and P_g is a low-dimensional manifold in a high-dimensional space, the probability that the measure of the overlap part of P_r and P_g is 0 is 1.

In mathematics, a manifold is a topological space that locally resembles Euclidean space near each point. More precisely, each point of an n-dimensional manifold has a neighbourhood that is homeomorphic to the Euclidean space of dimension n. In this more precise terminology, a manifold is referred to as an n-manifold.

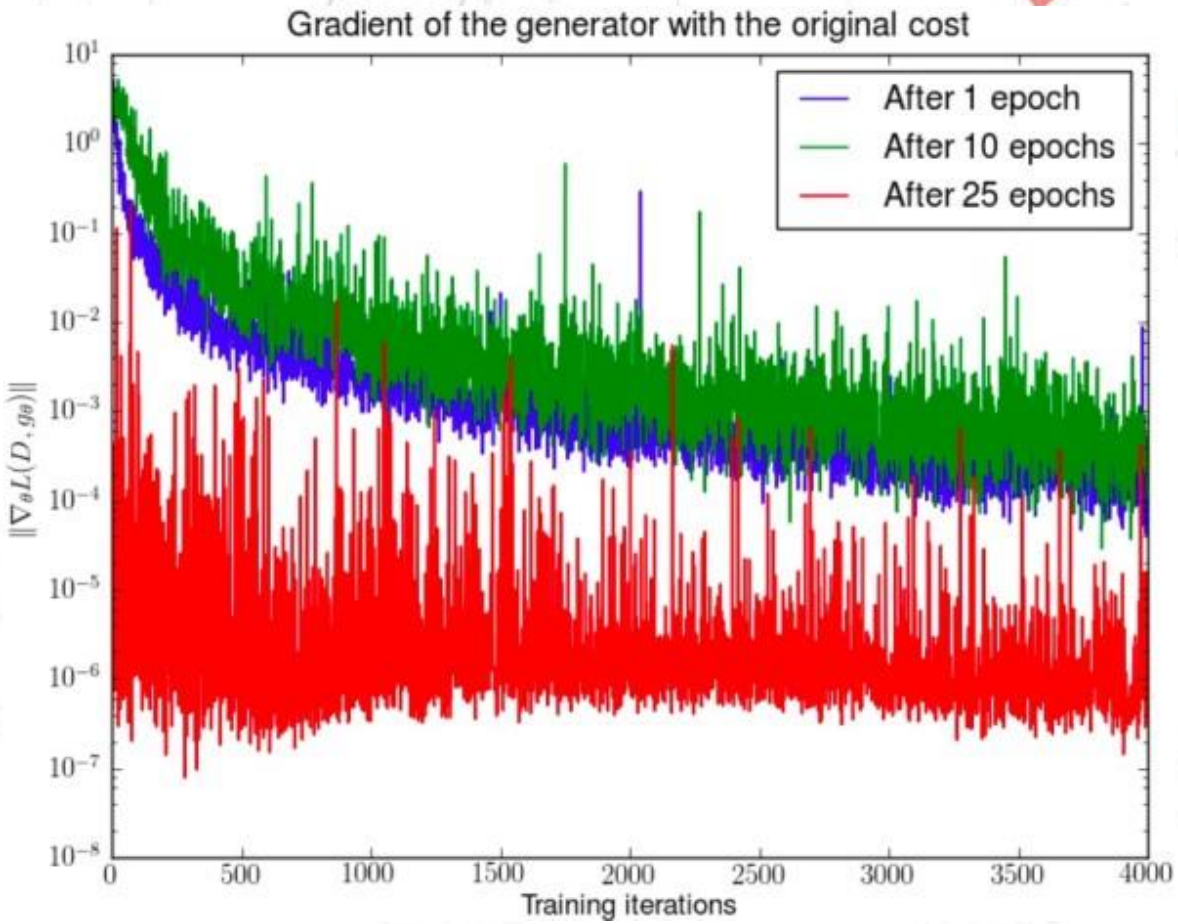
One-dimensional manifolds include lines and circles. Two-dimensional manifolds are also called surfaces. Examples include the plane, the sphere, and the torus, which can all be embedded (formed without self-intersections) in three dimensional real space.

In mathematical analysis, a measure on a set is a systematic way to assign a number to each suitable subset of that set, intuitively interpreted as its size. In this sense, a measure is a generalization of the concepts of length, area, and volume.

In the case of GANs, P_g is defined via sampling from a simple prior $z \sim p(z)$ (for example,100-dimensional), and then applying a function $g: Z \rightarrow \chi$, so the support of P_g has to be contained in *g(Z).* If the dimensionality of Z is less than the dimension of χ (for example, a picture:64×64,4096-dimensional), then it's imposible for P_g to be continuous. This is because in most cases *g(Z)* will be contained in a union of low dimensional manifolds, and therefore have measure 0 in χ.

Randomly select two curves in two-dimensional space, they are likely to have cross points, but its length is 0. The three-dimensional space is similar, take two surfaces randomly, they are more likely to exist cross lines, but the area is 0.

When the discriminator is optimal, we can equalize the loss function of the generator to minimize the JS divergence between P_r and P_g. However, their JS divergence is constant *log2*, which eventually causes the generator's gradient to be approximately *0* and the gradient to disappear.



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 original cost function. We see the gradient norms decay quickly, in the best case 5 orders of magnitude after 4000 discriminator iterations. Note the logarithmic scale.

Problem 2: collapse mode.

We have

$$\mathbb{E}_{x\sim P_r}[\log D^*(x)] + \mathbb{E}_{x\sim P_g}[\log(1 - D^*(x))] = 2JS(P_r||P_g) - 2\log 2$$

We transform the KL divergence into a form containing D:

$$\begin{aligned} KL(P_g||P_r) &= \mathbb{E}_{x\sim P_g}[\log \frac{P_g(x)}{P_r(x)}] \\ &= \mathbb{E}_{x\sim P_g}[\log \frac{P_g(x)/(P_r(x)+P_g(x))}{P_r(x)/(P_r(x)+P_g(x))}] \\ &= \mathbb{E}_{x\sim P_g}[\log \frac{1-D^*(x)}{D^*(x)}] \\ &= \mathbb{E}_{x\sim P_g}\log[1 - D^*(x)] - \mathbb{E}_{x\sim P_g}\log D^*(x) \end{aligned}$$

Therefore,

$$\begin{aligned} Loss_G = \boxed{\mathbb{E}_{x\sim P_g}[-\log D^*(x)]} &= KL(P_g||P_r) - \mathbb{E}_{x\sim P_g}\log[1 - D^*(x)] \\ &= KL(P_g||P_r) - 2JS(P_r||P_g) + 2\log 2 + \mathbb{E}_{x\sim P_r}[\log D^*(x)] \end{aligned}$$

- **Case 1**: If $P_r(x) > P_g(x)$, then *x* is a point with higher probability of coming from the data than being a generated sample. This is the core of the phenomenon commonly described as 'mode dropping': when there are large regions with high values of P_r, but small or zero values in P_g. It is important to note that when $P_r(x) > 0$ but $P_g(x) \to 0$, the integrand inside the KL grows quickly to infinity, meaning that this cost function assigns an extremely high cost to a generator's distribution not covering parts of the data.
- **Case 2**: If $P_r(x) < P_g(x)$, then x has low probability of being a data point, but high probability of being generated by our model. This is the case when we see our generator outputting an image that doesn't look real. In this case, when $P_r(x) \to 0$ and $P_g(x) > 0$, we see that the value inside the KL goes to *0*, meaning that this cost function will pay extremely low cost for generating fake looking samples.

Clearly, if we would minimize $KL(P_g|P_r)$ instead, the weighting of these errors would be reversed, meaning that this cost function would pay a high cost for generating not plausibly looking pictures.

Wasserstein distance

The Earth-Mover (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right] \tag{6}$$

where $\prod(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

Compared JS, the advantage of the EM distance.

The fact that the EM distance is continuous and differentiable a.e. means that we can (and should) train the critic till optimality. The argument is simple, the more we train the critic, the more reliable gradient of the Wasserstein we get, which is actually useful by the fact that Wasserstein is differentiable almost everywhere.

For the JS, as the discriminator gets better the gradients get more reliable but the true gradient is 0 since the JS is locally saturated and we get vanishing gradients.

Wasserstein GAN (WGAN)

In 2017, Martin Arjovsky et al. provided a comprehensive theoretical analysis of how the Earth Mover (EM) distance behaves in comparison to popular probability distances and divergences used in the context of learning distributions. And they also defined a form of GAN called Wasserstein-GAN that minimizes a reasonable and efficient approximation of the EM distance.

Author pointed to the fact that $W(\mathbb{P}_r, \mathbb{P}_g)$ might have nicer properties when optimized than $JS(\mathbb{P}_r, \mathbb{P}_g)$. However, the infimum in formula 6 is highly intractable. On the other hand, the Kantorovich-Rubinstein duality tells us that

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

where the supremum is over all the **1-Lipschitz** functions $f : \chi \to \mathbb{R}$.

Lipschitz continuity

In mathematical analysis, Lipschitz continuity is a strong form of uniform continuity for functions. Intuitively, a Lipschitz continuous function is limited in how fast it can change: there exists a real number such that, for every pair of points on the graph of this function, the absolute value of the slope of the line connecting them is not greater than this real number; the smallest such bound is called the Lipschitz constant of the function (or modulus of uniform continuity). For instance, every function that has bounded first derivatives is Lipschitz.

In particular, a real-valued function $f : \mathrm{R} \to \mathrm{R}$ is called Lipschitz continuous if there exists a positive real constant K such that, for all real x1 and x2,

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$$

Any such K is referred to as a Lipschitz constant for the function f. The smallest constant is sometimes called the (best) Lipschitz constant;

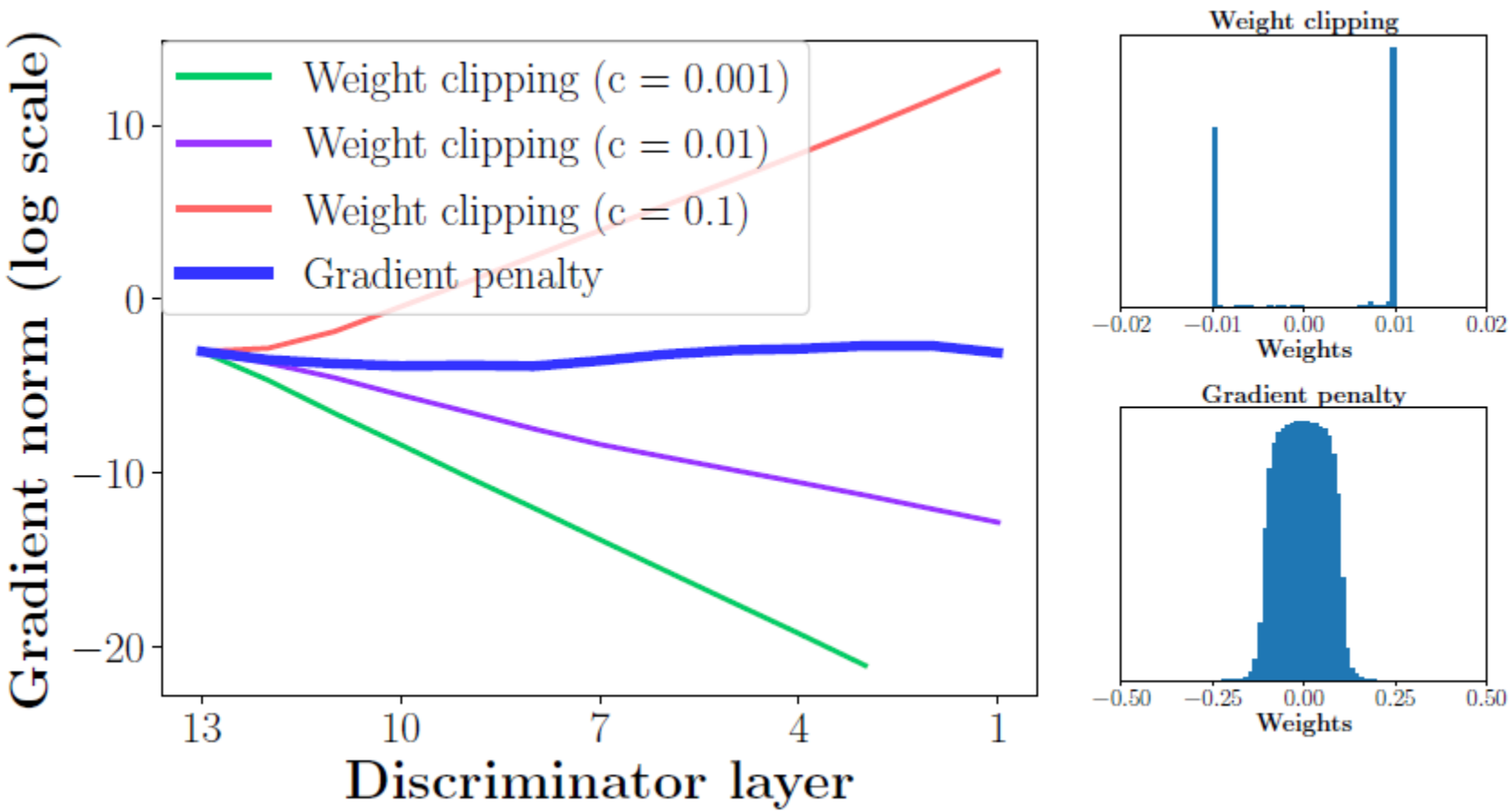
Compared the original GAN, WGAN has only changed four points:

- The last layer of the discriminator removes *sigmoid*.
- The loss of the generator and discriminator doesn't compute the *log*.
- Each updated the parameters of the discriminator, their values are truncated to no more than a fixed constant c. (called clipping — to satisfy the condition of **1-Lipschitz function.**)
- Do not use momentum-based optimization algorithms (including momentum and Adam), recommend RMSProp, SGD.

Improved Training of Wasserstein GANs

The WGAN makes progress toward stable training of GANs, but can still generate low-quality samples or fail to converge in some settings. Martin Arjovsky et al. finded that these problems are often due to the use of weight clipping in WGAN to enforce a Lipschitz constraint on the critic, which can lead to pathological behavior.

In 2017, Martin Arjovsky et al. proposed an alternative to clipping weights: penalize the norm of gradient of the critic with respect to its input (called **gradient penalty**) on **"Improved Techniques for Training GANs"**.



(left) Gradient norms of deep WGAN critics during training on toy datasets either explode or vanish when using weight clipping, but not when using a gradient penalty. (right)Weight clipping (top) pushes weights towards two values (the extremes of the clipping range), unlike gradient penalty (bottom).

gradient penalty

As previously mentioned, the **1-Lipschitz** limit requires that the discriminator's gradient not exceed *K*.

$$||\nabla_x D(x)||_p \leq K, \forall x \in \mathcal{X}$$

So why not just set an extra loss item to reflect this? For example:

$$ReLU[||\nabla_x D(x)||_p - K] \tag{7}$$

Or

$$[||\nabla_x D(x)||_p - K]^2 \tag{8}$$

In this paper, the author chose formula 8.

gradient penalty

Our new objective is

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\text{Our gradient penalty}}.$$

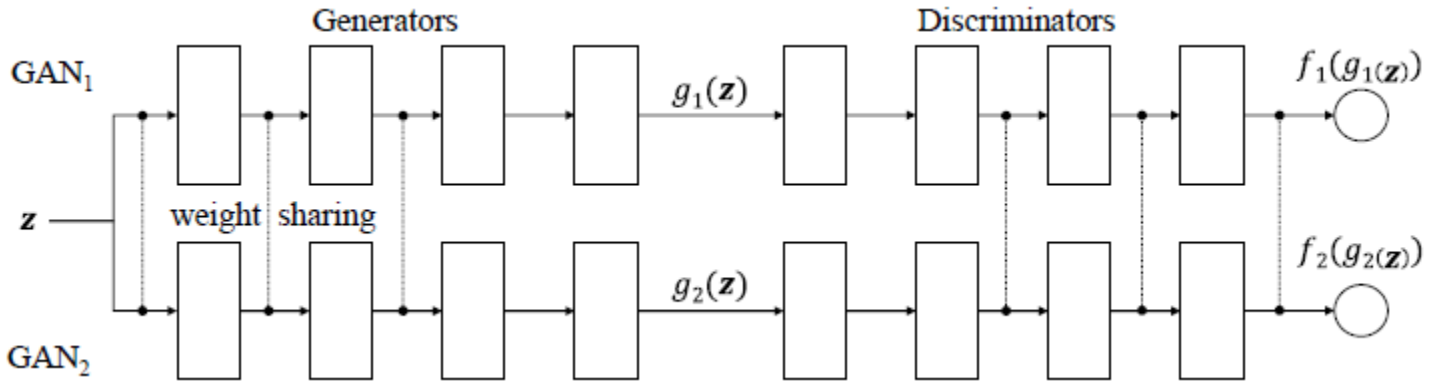
We implicitly define $\mathbb{P}_{\hat{x}}$ sampling uniformly along straight lines between pairs of points sampled from the data distribution $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_g$. This is motivated by the fact that the graph of the optimal critic consists of straight lines connecting points from $\mathbb{P}_r$ and $\mathbb{P}_g$.

$$x_r \sim P_r, x_g \sim P_g, \epsilon \sim Uniform[0, 1]$$

$$\hat{x} = \epsilon x_r (1 - \epsilon) x_g$$

In general, the penalty coefficient $\gamma = 10$.

Extension: Variants of GAN



Coupled Generative Adversarial Networks(CoGAN) consists of a pair of GANs: GAN1 and GAN2. Each has a generative model for synthesizing realistic images in one domain and a discriminative model for classifying whether an image is real or synthesized. We tie the weights of the first few layers (responsible for decoding high-level semantics) of the generative models, g1 and g2. We also tie the weights of the last few layers (responsible for encoding high-level semantics) of the discriminative models, f1 and f2. This weight-sharing constraint allows CoGAN to learn a joint distribution of images without correspondence supervision. A trained CoGAN can be used to synthesize pairs of corresponding images—pairs of images sharing the same high-level abstraction but having different low-level realizations.

Least Squares Generative Adversarial Networks(LSGAN)

In 2017, Xudong Mao et al. proposed the Least Squares Generative Adversarial Networks (LSGANs) which adopt the least squares loss function for the discriminator. We show that minimizing the objective function of LSGAN yields minimizing the Pearson X^2 divergence.

The objective function of LSGAN is

$$\min_D V_{LSGAN}(D) = \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}[(D(x)-1)^2] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[D(G(z)+1)^2]$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\left[D\left(G(z)\right)^2\right]$$

BEGAN: Boundary Equilibrium Generative Adversarial Networks

In paper "BEGAN: Boundary Equilibrium Generative Adversarial Networks", David Berthelot et al. proposed a new equilibrium enforcing method paired with a loss derived from the Wasserstein distance for training auto-encoder based Generative Adversarial Networks. This method balances the generator and discriminator during training.

The Wasserstein distance can be expressed as:

$$W_1(\mu_1, \mu_2) = \inf_{\gamma \in \Gamma(\mu_1, \mu_2)} \mathbb{E}_{(x_1, x_2) \sim \gamma}[|x_1 - x_2|]$$

Using Jensen's inequality, we can derive a lower bound to $W_1(\mu_1, \mu_2)$:

$$inf\,\mathbb{E}[|x_1 - x_2|] \geq inf\,|\mathbb{E}[x_1 - x_2]| = |m_1 - m_2|$$

It is important to note that we are aiming to optimize a lower bound of the Wasserstein distance between auto-encoder loss distributions, not between sample distributions.

BEGAN

We consider them to be at equilibrium when:

$$\mathbb{E}[\mathcal{L}(x)] = \mathbb{E}\big[\mathcal{L}\big(G(z)\big)\big]$$

We can relax the equilibrium with the introduction of a new hyper-parameter $\gamma \in [0,1]$ defined as

$$\gamma = \frac{\mathbb{E}\big[\mathcal{L}\big(G(z)\big)\big]}{\mathbb{E}[\mathcal{L}(x)]}$$

The BEGAN objective is:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t \cdot \mathcal{L}\big(G(z_D)\big) & for\ \theta_D \\ \mathcal{L}_G = \mathcal{L}\big(G(z_D)\big) & for\ \theta_G \\ k_{t+1} = k_t + \lambda_k \left(\gamma\mathcal{L}(x) - \mathcal{L}\big(G(z_D)\big)\right) & for\ each\ training\ step\ t \end{cases}$$

END

Thank you !