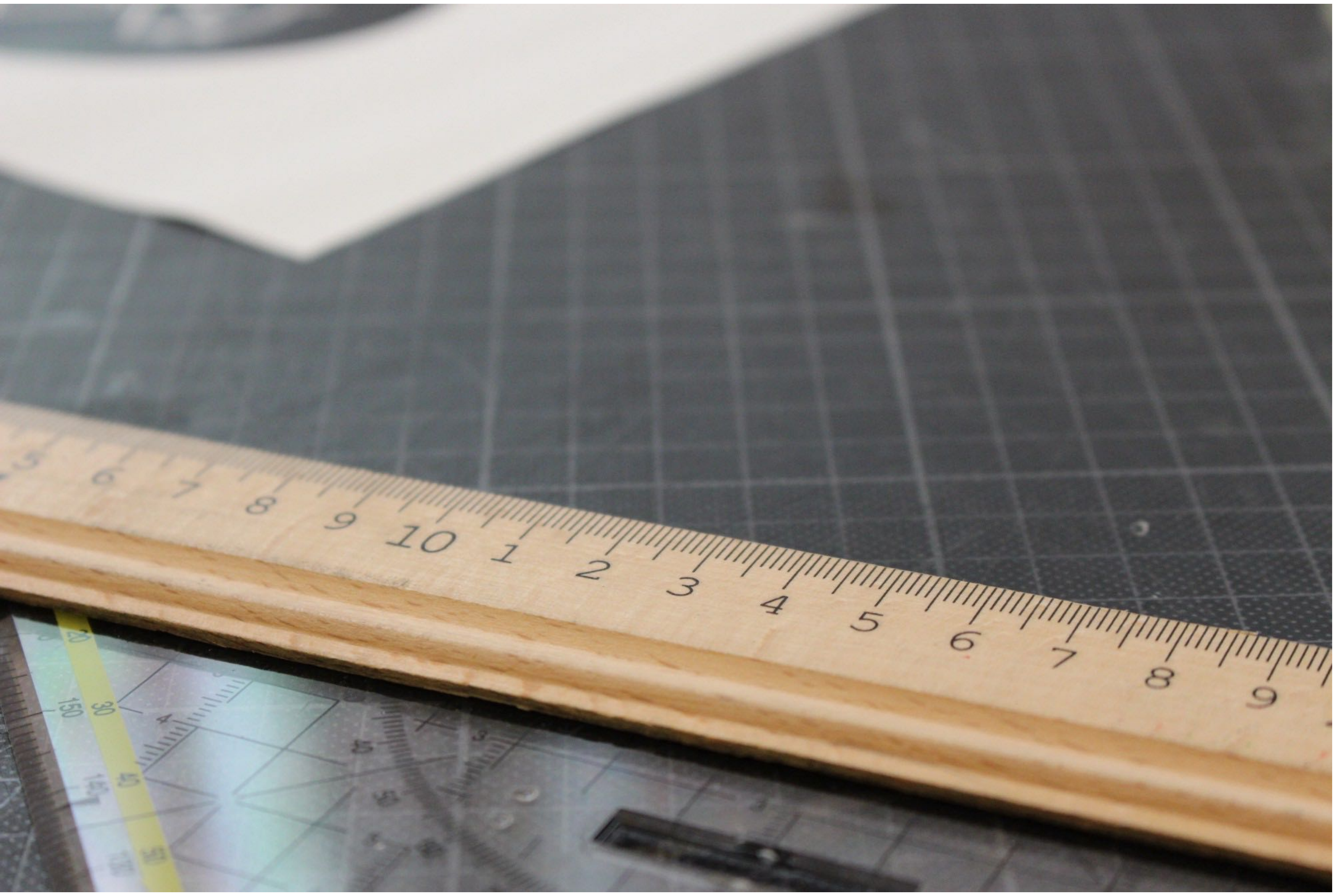


Photo by Christian Kaindl

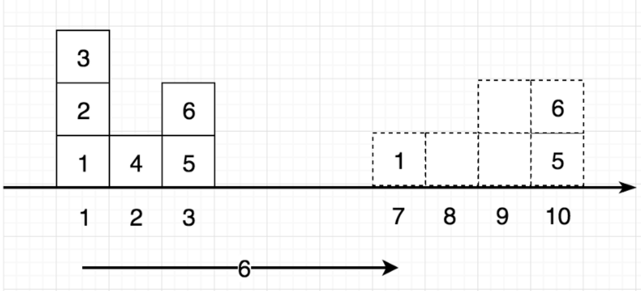
GAN — Wasserstein GAN & WGAN-GP

Jonathan Hui Follow
Jun 14, 2018 · 11 min read

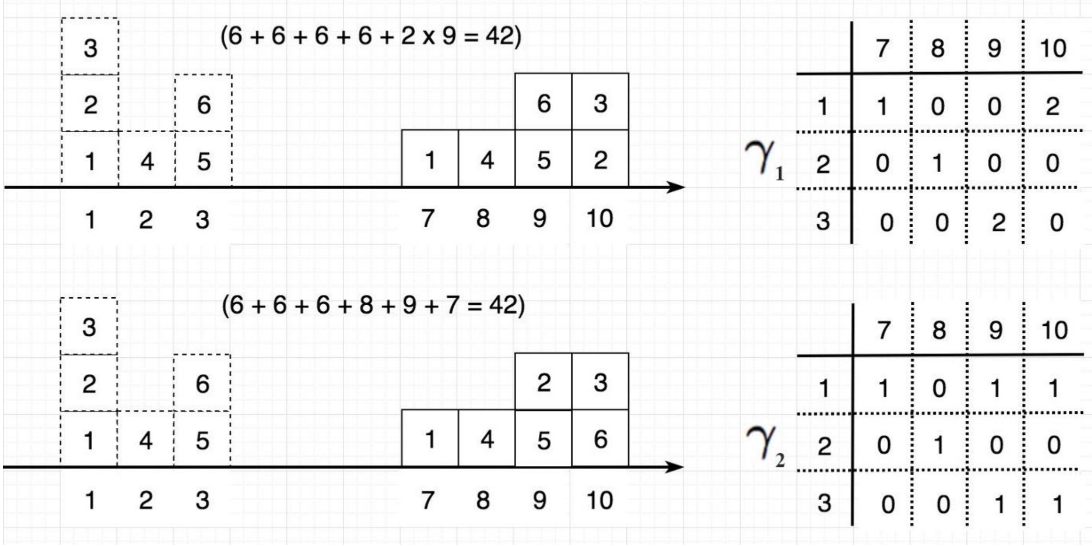
Training GAN is hard. Models may never converge and mode collapses are common. To move forward, we can make incremental improvements or embrace a new path for a new cost function. **Do cost functions matter in the GAN training?** This article is part of the GAN series which looks into the Wasserstein GAN (WGAN) and the WGAN-Gradient penalty in details. The equations for Wasserstein GAN look very unapproachable. In reality, they are pretty simple and we will explain them with examples.

Earth-Mover (EM) distance/ Wasserstein Metric

Let's complete a simple exercise on moving boxes. We get 6 boxes and we want to move them from the left to the locations marked by the dotted square on the right. For box #1, we move it from location 1 to location 7. The moving cost equals to its weight times the distance. For simplicity, we will set the weight to be 1. Therefore the cost to move box #1 equals to 6 (7–1).



The figure below presents two different moving plans γ. The tables in the right illustrates how boxes are moved. For example, in the first plan, we move 2 boxes from location 1 to location 10 and the entry γ(1, 10) is therefore set to two. The total transport cost of either plan below is 42.



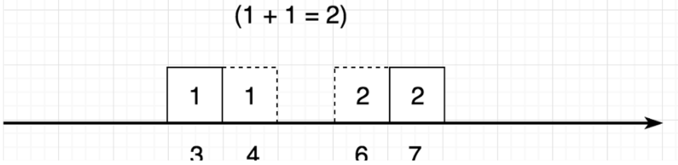
γ_1

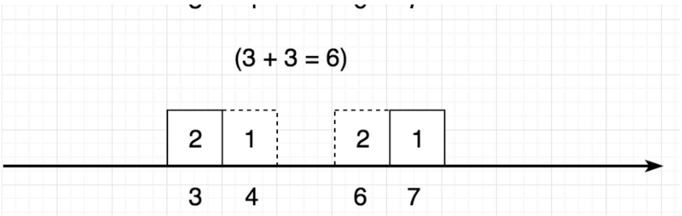
	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0

γ_2

	7	8	9	10
1	1	0	1	1
2	0	1	0	0
3	0	0	1	1

However, not all transport plans bear the same cost. The **Wasserstein distance** (or the **EM distance**) is the cost of the cheapest transport plan. In the example below, both plans have different cost and the Wasserstein distance (minimum cost) is two.





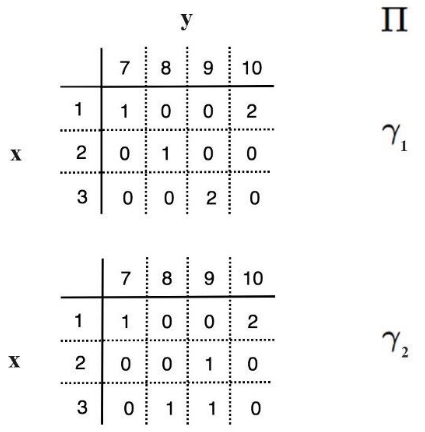
Let’s throw in some complicated terms before explaining it. The Wasserstein distance is the minimum cost of transporting mass in converting the data distribution ***q*** to the data distribution ***p***. The Wasserstein distance for the real data distribution ***Pr*** and the generated data distribution ***Pg*** is mathematically defined as the greatest lower bound (infimum) for any transport plan (i.e. the cost for the cheapest plan):

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

From the WGAN paper:

> *Π(Pr, Pg) denotes the set of all joint distributions γ(x, y) whose marginals are respectively Pr and Pg.*

Don’t get scared by the mathematical formula. The equation above is the equivalent of our example in the continuous space. ***Π*** contains all the possible transport plan ***γ***.



y

Π

x

	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0

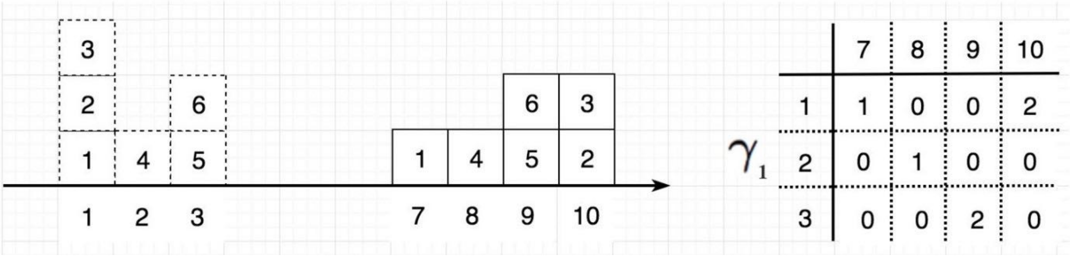
γ_1

x

	7	8	9	10
1	1	0	0	2
2	0	0	1	0
3	0	1	1	0

γ_2

We combine variable *x* and *y* to form a joint distribution *γ(x, y)* and *γ(1, 10)* is simply how many boxes at location 10 is from location 1. The number of boxes in location 10 must originally come from any position, i.e. ∑ *γ(*, 10) = 2*. That is the same as saying *γ(x, y)* must have marginals ***Pr*** and ***Pg*** respectively.



	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0

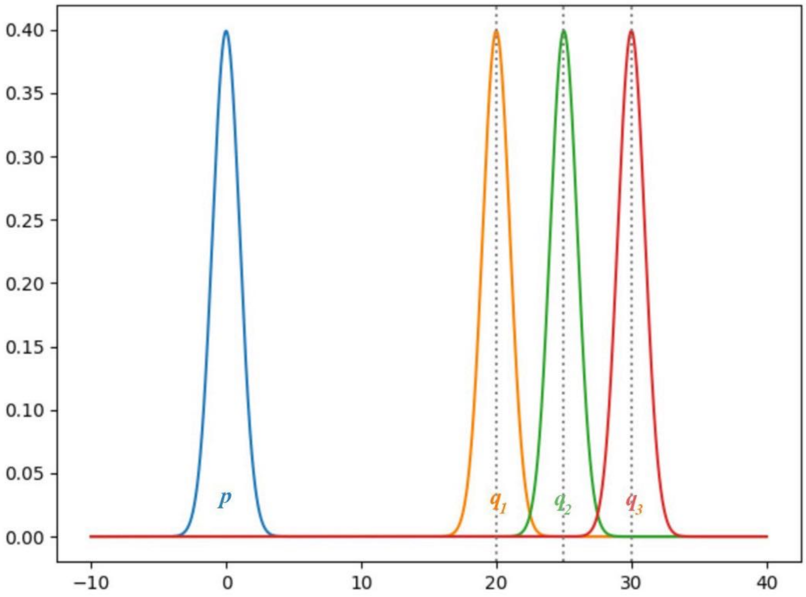
KL-Divergence and JS-Divergence

Before advocating any new cost functions, let's look at the two common divergences used in generative models first, namely the KL-Divergence and the JS-Divergence.

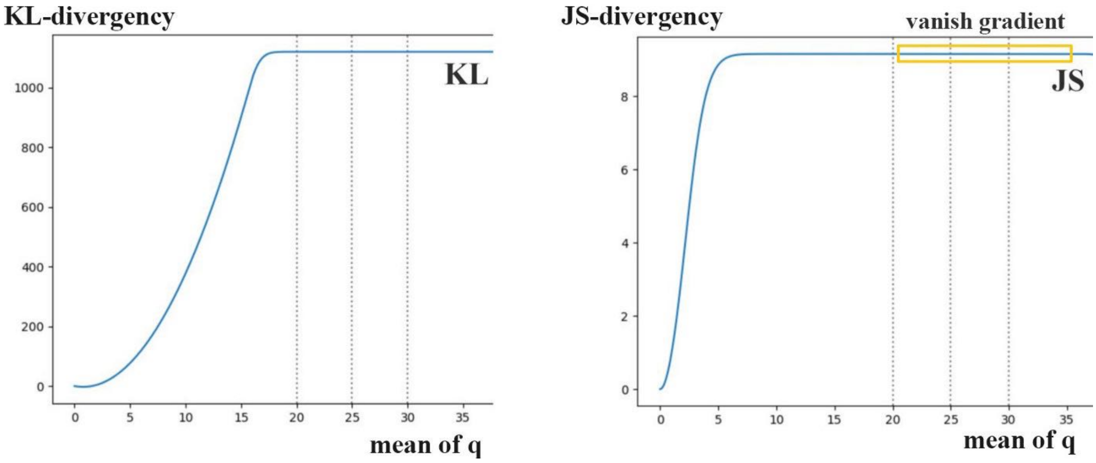
$$D_{KL}(P||Q) = \sum_{x=1}^{N} P(x) \log \frac{P(x)}{Q(x)}$$

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

where ***p*** is the real data distribution and ***q*** is the one estimated from the model. Let's assume they are Gaussian distributed. In the diagram below, we plot ***p*** and a few ***q*** having different means.



Below, we plot the corresponding KL-divergence and JS-divergence between ***p*** and ***q*** with means ranging from 0 to 35. As anticipated, when both ***p*** and ***q*** are the same, the divergence is 0. As the mean of ***q*** increases, the divergence increases. The gradient of the divergency will eventually diminish. We have close to a zero gradient, i.e. the generator learns nothing from the gradient descent.



Criticizing is easy. In practice, GAN can optimize the discriminator easier than the generator. Minimizing the GAN objective function with an optimal discriminator is equivalent to minimizing the JS-divergence (proof). As illustrated above, if the generated image has distribution *q* far away from the ground truth *p*, the generator barely learns anything.

Arjovsky et al 2017 wrote a paper to illustrate the GAN problem mathematically with the following conclusions:

- An optimal discriminator produces good information for the generator to improve. But if the generator is not doing a good job yet, the gradient for the generator diminishes and the generator learns nothing (the same conclusion we just explain).

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

orignal GAN generator's gradient

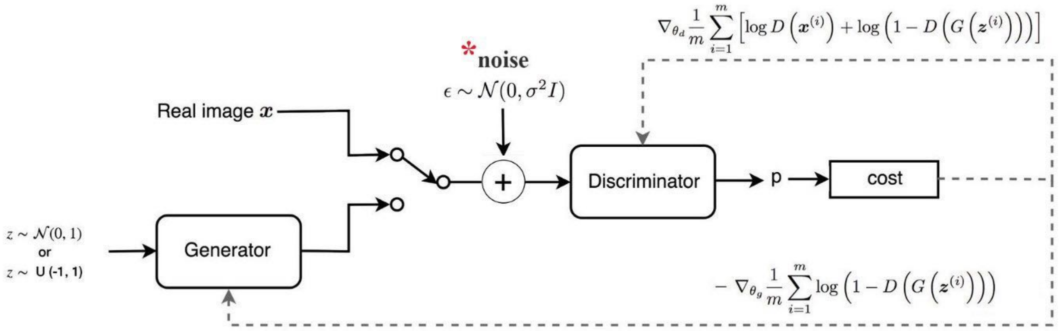
- The original GAN paper proposes an alternative cost function to address this gradient vanishing problem. However, Arjovsky illustrates the new functions have large variance of gradients that make the model unstable.

$$\nabla_{\theta_g}\log D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)$$

unstable with large variance of gradients

Alternative proposal

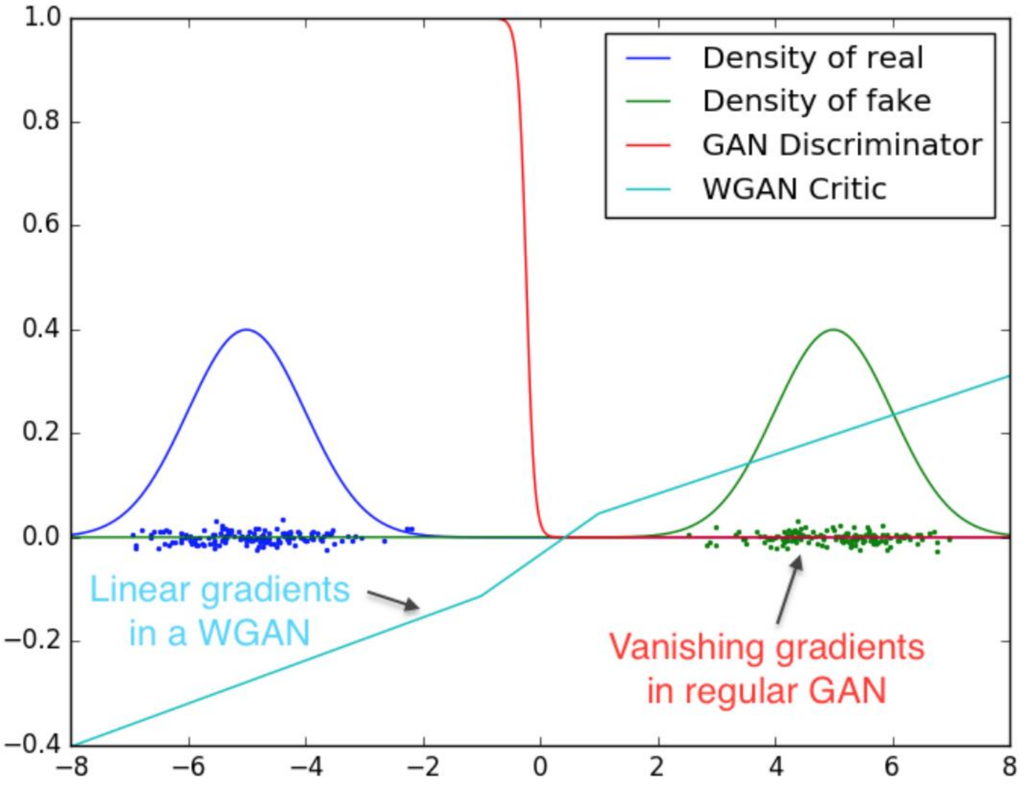
- Arjovsky proposes adding noise to generated images to stabilize the model.


$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right]$
*noise
$\epsilon \sim \mathcal{N}(0, \sigma^2 I)$
Real image x
+
Discriminator
p
cost
$z \sim \mathcal{N}(0,1)$
or
$z \sim$ U (-1, 1)
Generator
$-\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$

For more details, here is another article in our series that summarizes some of the important mathematical claims.

Wasserstein Distance

Instead of adding noise, Wasserstein GAN (WGAN) proposes a new cost function using Wasserstein distance that has a smoother gradient everywhere. WGAN learns no matter the generator is performing or not. The diagram below repeats a similar plot on the value of ***D(X)*** for both GAN and WGAN. For GAN (the red line), it fills with areas with diminishing or exploding gradients. For WGAN (the blue line), the gradient is smoother everywhere and learns better even the generator is not producing good images.



Source

Wasserstein GAN

However, the equation for the Wasserstein distance is highly intractable. Using the Kantorovich-Rubinstein duality, we can simplify the calculation to

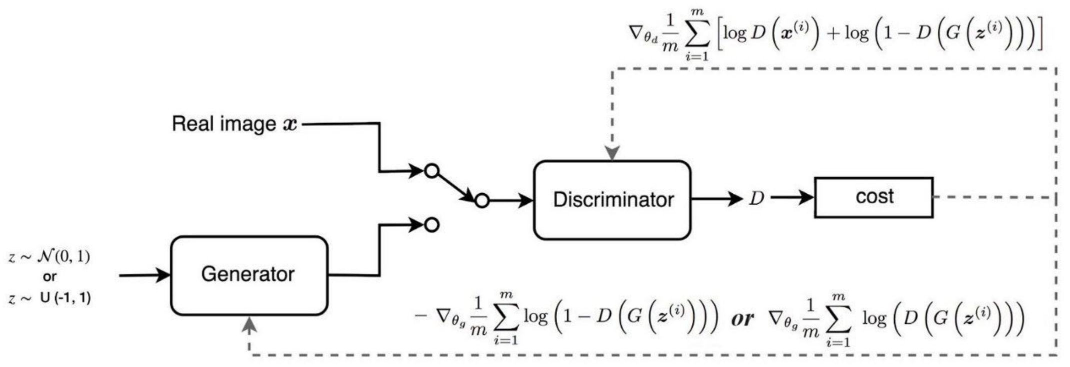
$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

where *sup* is the least upper bound and ***f*** is a 1-Lipschitz function following this constraint (see here for more info on the Lipschitz constraint):

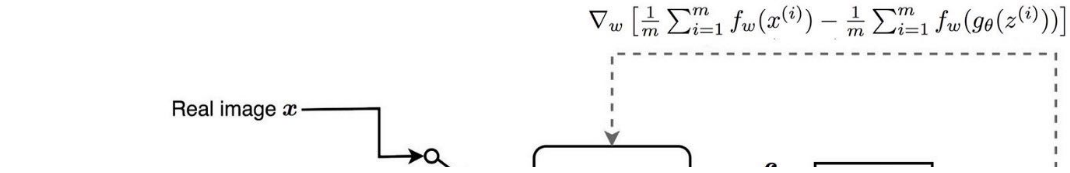
$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|.$$

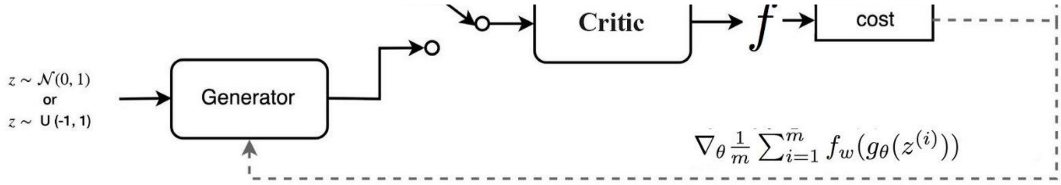
So to calculate the Wasserstein distance, we just need to find a 1-Lipschitz function. Like other deep learning problem, we can build a deep network to learn it. Indeed, this network is very similar to the discriminator ***D,*** just without the sigmoid function and outputs a scalar score rather than a probability. This score can be interpreted as how real the input images are. In reinforcement learning, we call it the **value function** which measures how good a state (the input) is. We rename the discriminator to **critic** to reflect its new role. Let's show GAN and WGAN side-by-side.

GAN:


$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(x^{(i)}\right)+\log\left(1-D\left(G\left(z^{(i)}\right)\right)\right)\right]$
Real image x
Discriminator
D
cost
$z\sim\mathcal{N}(0,1)$
or
$z\sim$ U (-1, 1)
Generator
$-\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(z^{(i)}\right)\right)\right)$ or $\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\left(D\left(G\left(z^{(i)}\right)\right)\right)$

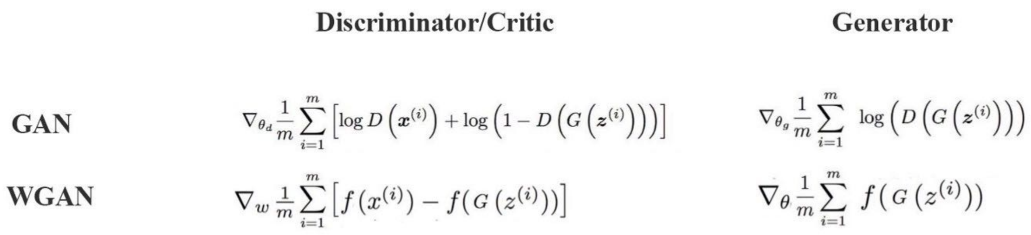
WGAN


$\nabla_w\left[\frac{1}{m}\sum_{i=1}^{m}f_w(x^{(i)})-\frac{1}{m}\sum_{i=1}^{m}f_w(g_\theta(z^{(i)}))\right]$
Real image x



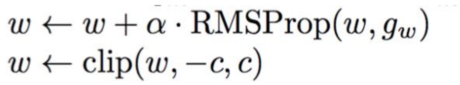
$z \sim \mathcal{N}(0,1)$
or
$z \sim U(-1,1)$
Generator
Critic
f
cost
$\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$

The network design is almost the same except the critic does not have an output sigmoid function. The major difference is only on the cost function:



	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$	$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
WGAN	$\nabla_w \frac{1}{m}\sum_{i=1}^{m}\left[f\left(x^{(i)}\right) - f\left(G\left(z^{(i)}\right)\right)\right]$	$\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f\left(G\left(z^{(i)}\right)\right)$

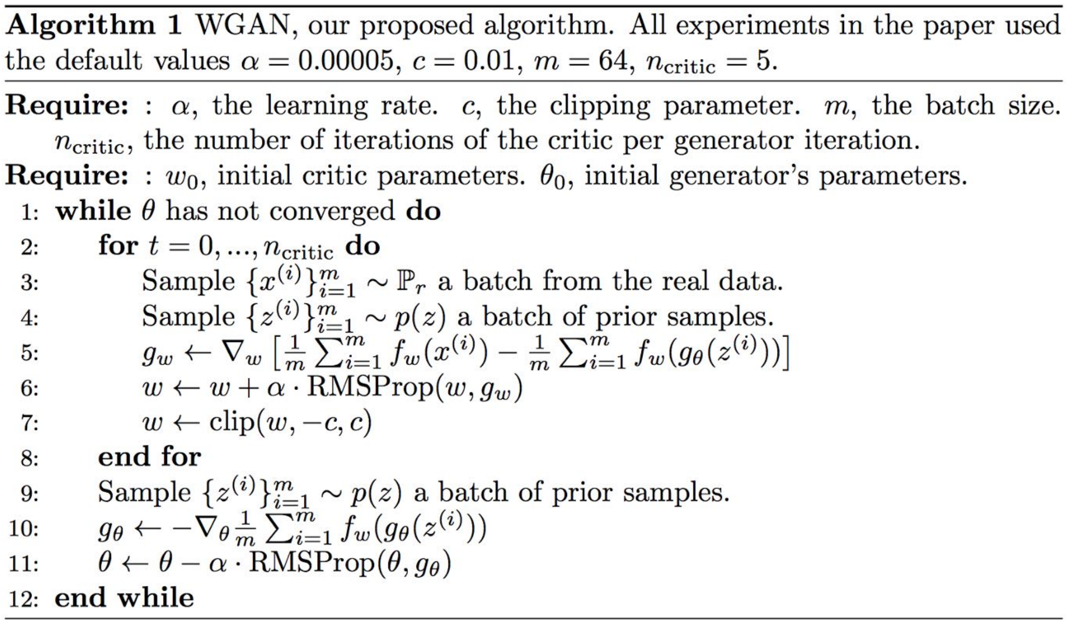
However, there is one major thing missing. ***f*** has to be a 1-Lipschitz function. To enforce the constraint, WGAN applies a very simple clipping to restrict the maximum weight value in ***f***, i.e. the weights of the discriminator must be within a certain range controlled by the hyperparameters ***c***.



$$w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$$
$$w \leftarrow \text{clip}(w, -c, c)$$

Algorithm

Now we can put everything together in the pseudo-code below.



Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

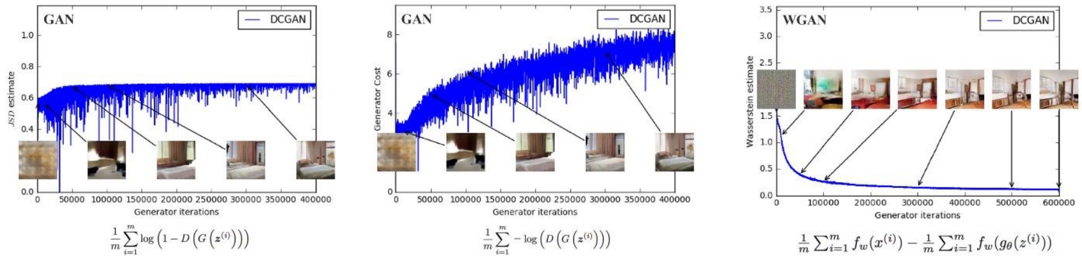
```
while $\theta$ has not converged do
    for $t = 0, ..., n_{\text{critic}}$ do
        Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
        Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
        $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
        $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
        $w \leftarrow \text{clip}(w, -c, c)$
    end for
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while
```

Source

Experiment

Correlation between loss metric and image quality

In GAN, the loss measures how well it fools the discriminator rather than a measure of the image quality. As shown below, the generator loss in GAN does not drop even the image quality improves. Hence, we cannot tell the progress from its value. We need to save the testing images and evaluate it visually. On the contrary, WGAN loss function reflects the image quality which is more desirable.



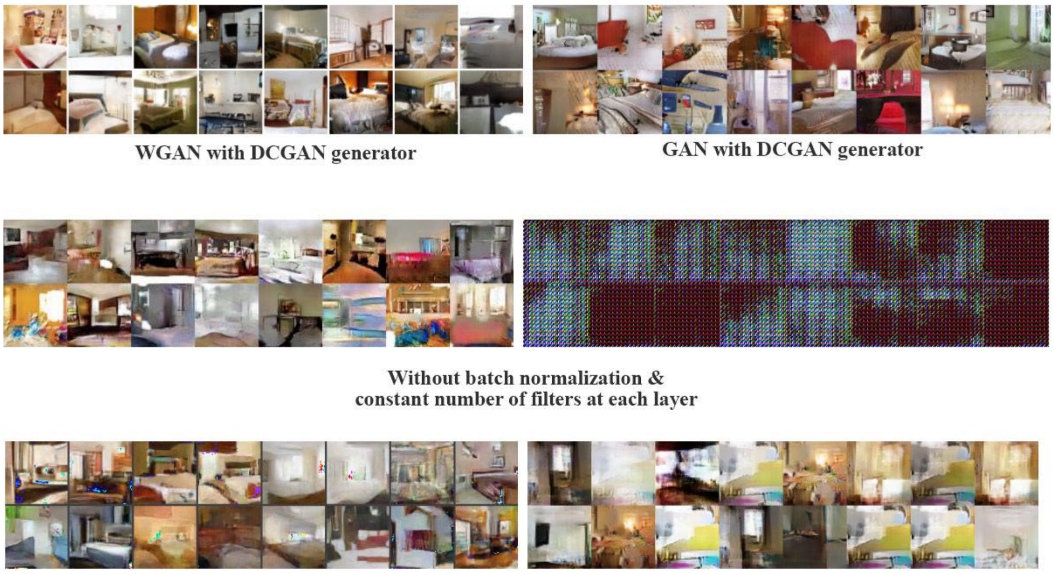
Source

Improve training stability

Two significant contributions for WGAN are

- it has no sign of mode collapse in experiments, and
- the generator can still learn when the critic perform well.

As shown below, even though we remove the batch normalization in DCGAN, WGAN can still perform.



WGAN with DCGAN generator

GAN with DCGAN generator

Without batch normalization &
constant number of filters at each layer

Using a MLP as the generator

All critics and discriminators follow the same discriminator design in DCGAN

Source

WGAN — Issues

Lipschitz constraint

Clipping allows us to enforce the Lipschitz constraint on the critic's model to calculate the Wasserstein distance.

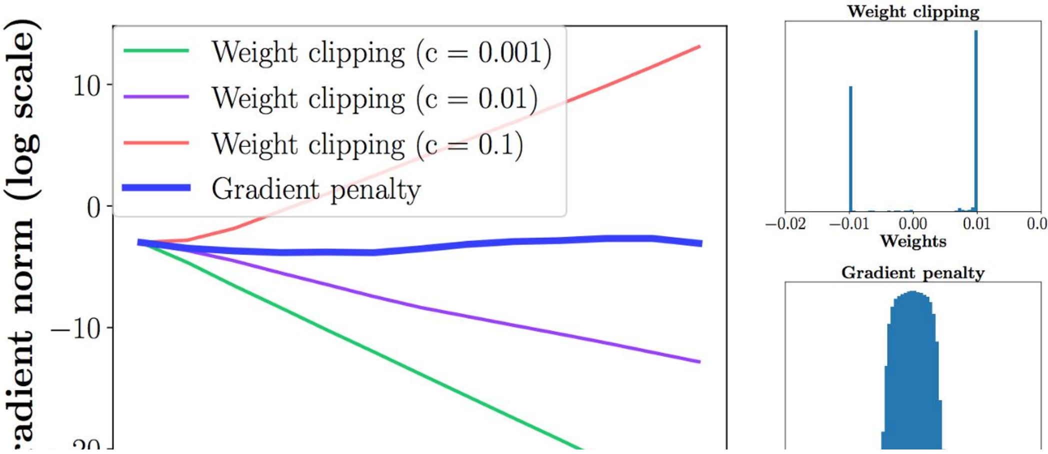
$$|f(x_1) - f(x_2)| \le K|x_1 - x_2|.$$

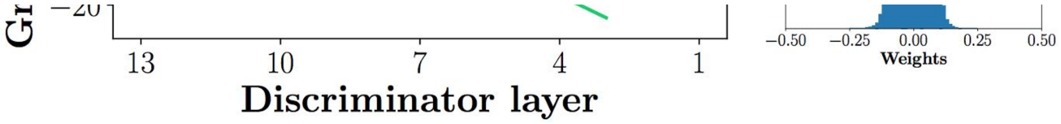
> *Quote from the research paper: Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. If the clipping parameter is large, then it can take a long time for any weights to reach their limit, thereby making it harder to train the critic till optimality. If the clipping is small, this can easily lead to vanishing gradients when the number of layers is big, or batch normalization is not used (such as in RNNs) … and we stuck with weight clipping due to its simplicity and already good performance.*

The difficulty in WGAN is to enforce the Lipschitz constraint. Clipping is simple but it introduces some problems. The model may still produce poor quality images and does not converge, in particular when the hyperparameter **c** is not tuned correctly.

$$w \leftarrow \text{clip}(w, -c, c)$$

The model performance is very sensitive to this hyperparameter. In the diagram below, when batch normalization is off, the discriminator moves from diminishing gradients to exploding gradients when **c** increases from 0.01 to 0.1.

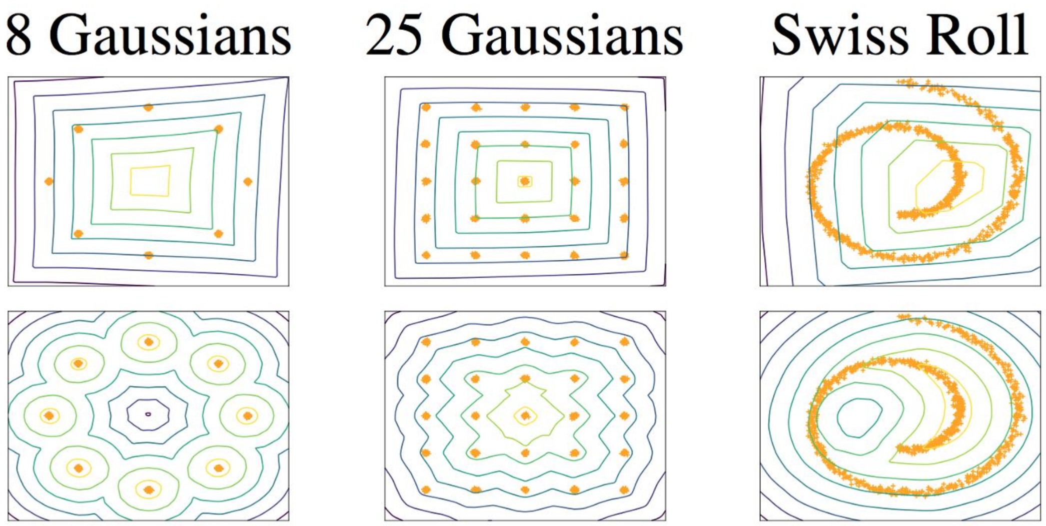




Source

Model capacity

The weight clipping behaves as a weight regulation. It reduces the capacity of the model *f* and limits the capability to model complex functions. In the experiment below, the first row is the contour plot of the value function estimated by WGAN. The second row is estimated by a variant of WGAN called WGAN-GP. The reduced capacity of WGAN fails to create a complex boundary to surround the modes (orange dots) of the model while the improved WGAN-GP can.



Source

Wasserstein GAN with gradient penalty (WGAN-GP)

WGAN-GP uses gradient penalty instead of the weight clipping to enforce the Lipschitz constraint.

$$|f(x_1) - f(x_2)| \le K|x_1 - x_2|.$$

Gradient penalty

A differentiable function *f* is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere.

f^* *has gradient norm 1 almost everywhere under* $\mathbb{P}_r$ *and* $\mathbb{P}_g$.

In specific, Appendix A in WGAN-GP paper proves that

$$\textit{if } f^* \textit{ is differentiable }, \pi(x = y) = 0 \textit{ , and } x_t = (1-t)y \textit{ with } 0 \leq t \leq 1 \textit{ ,}$$

$$\textit{it holds that } \mathbb{P}_{(x,y)\sim\pi}\left[\nabla f^*(x_t) = \frac{y-x_t}{\|y-x_t\|}\right] = 1.$$

Source

Points interpolated between the real and generated data should have a gradient norm of 1 for *f*.

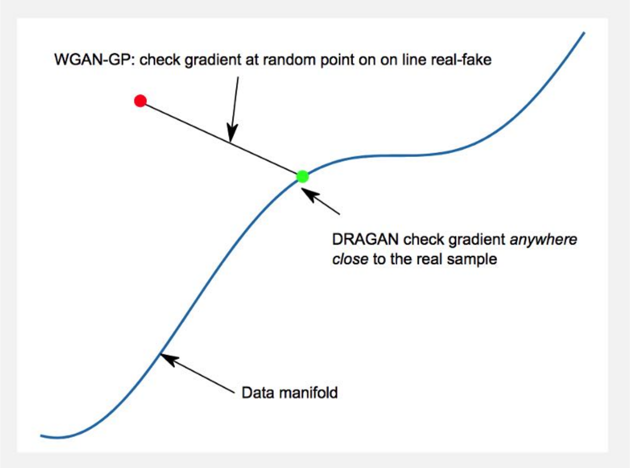
So instead of applying clipping, WGAN-GP penalizes the model if the gradient norm moves away from its target norm value 1.

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

$$\textit{where } \hat{\boldsymbol{x}} \textit{ sampled from } \tilde{\boldsymbol{x}} \textit{ and } \boldsymbol{x} \textit{ with t uniformly sampled between 0 and 1}$$

$$\hat{\boldsymbol{x}} = t\tilde{\boldsymbol{x}} + (1-t)\boldsymbol{x} \textit{ with } 0 \leq t \leq 1$$

λ is set to 10. The point *x* used to calculate the gradient norm is any points sampled between the *Pg* and *Pr*. (It will be easier to understand this with the pseudo-code later.)


WGAN-GP: check gradient at random point on on line real-fake
DRAGAN check gradient *anywhere close* to the real sample
Data manifold

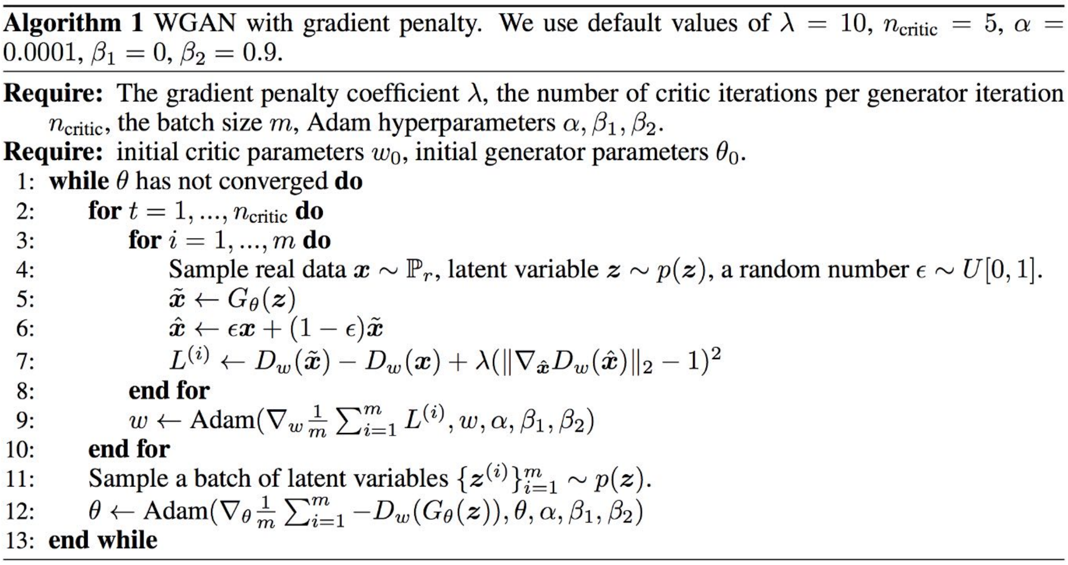
Source

Batch normalization is **avoided** for the critic (discriminator). Batch normalization creates correlations between samples in the same batch. It impacts the effectiveness of the gradient penalty which is confirmed by experiments.

By design or not, some new cost functions add gradient penalty to the cost function. Some is purely based on empirical observation that models misbehave when the gradient increases. However, gradient penalty adds computational complexity that may not be desirable but it does produce some higher-quality images.

Algorithm

Let’s look into the pseudo code in detailing how the sample point is created and how the gradient penalty is computed.



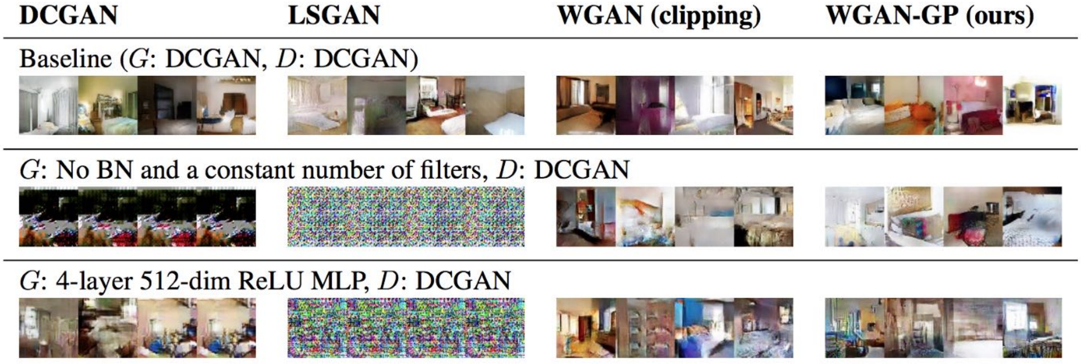
Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

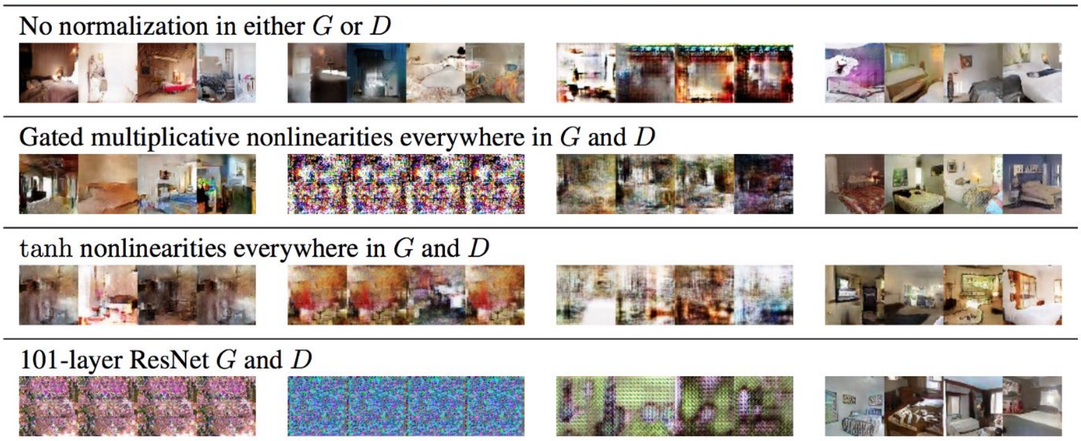
```
Require: The gradient penalty coefficient $\lambda$, the number of critic iterations per generator iteration
    $n_{\text{critic}}$, the batch size $m$, Adam hyperparameters $\alpha, \beta_1, \beta_2$.
Require: initial critic parameters $w_0$, initial generator parameters $\theta_0$.
while $\theta$ has not converged do
    for $t = 1, ..., n_{\text{critic}}$ do
        for $i = 1, ..., m$ do
            Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
            $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
            $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
            $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
        end for
        $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
    end for
    Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while
```

Source

WGAN-GP Experiments

WGAN-GP enhances training stability. As shown below, when the model design is less optimal, WGAN-GP can still create good results while the original GAN cost function fails.


DCGAN
LSGAN
WGAN (clipping)
WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)
G: No BN and a constant number of filters, D: DCGAN
G: 4-layer 512-dim ReLU MLP, D: DCGAN



No normalization in either G or D

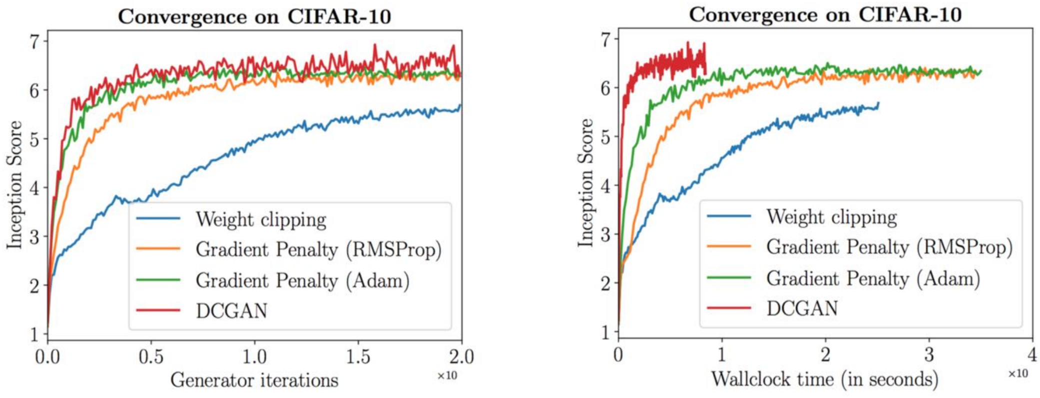
Gated multiplicative nonlinearities everywhere in G and D

tanh nonlinearities everywhere in G and D

101-layer ResNet G and D

Source

Below is the inception score using different methods. The experiment from the WGAN-GP paper demonstrates better image quality and convergence comparing with WGAN. However, DCGAN demonstrates slightly better image quality and it converges faster. But the inception score for WGAN-GP is more stable when it starts converging.

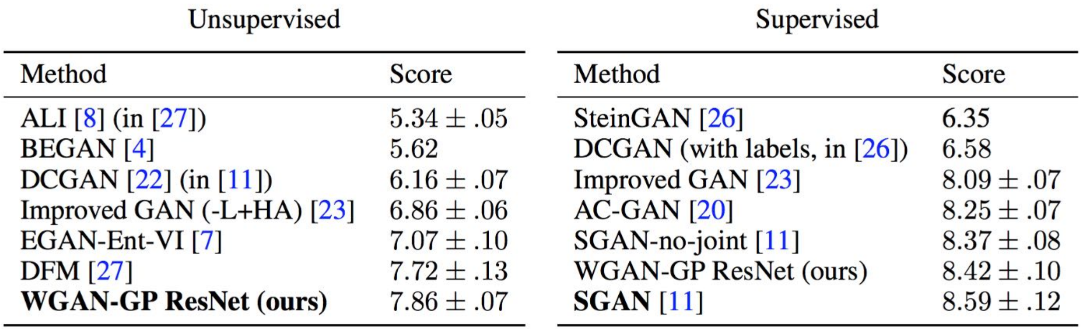


Convergence on CIFAR-10

Convergence on CIFAR-10

Source

So what is the benefit of WGAN-GP if it cannot beat DCGAN? The major advantage of WGAN-GP is its convergency. It makes training more stable and therefore easier to train. As WGAN-GP helps models to converge better, we can use a more complex model like a deep ResNet for the generator and the discriminator. The following are the inception score (the higher the better) using ResNet with WGAN-GP.



Unsupervised

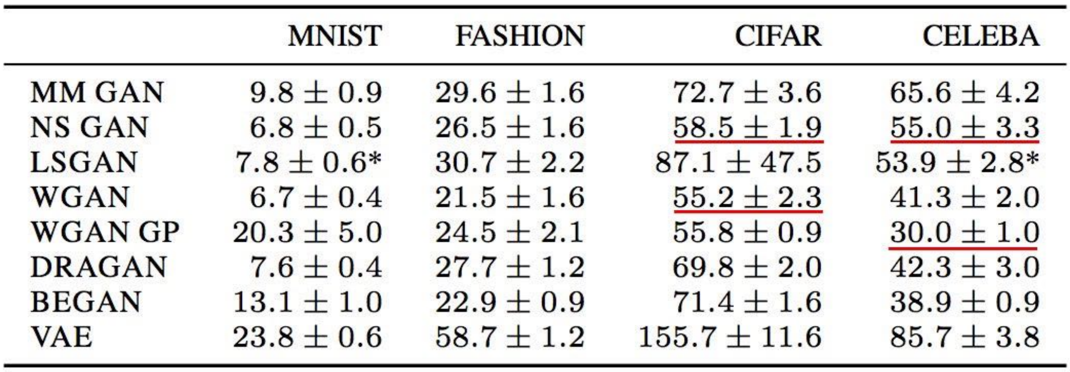
Method	Score
ALI [8] (in [27])	5.34 ± .05
BEGAN [4]	5.62
DCGAN [22] (in [11])	6.16 ± .07
Improved GAN (-L+HA) [23]	6.86 ± .06
EGAN-Ent-VI [7]	7.07 ± .10
DFM [27]	7.72 ± .13
WGAN-GP ResNet (ours)	7.86 ± .07

Supervised

Method	Score
SteinGAN [26]	6.35
DCGAN (with labels, in [26])	6.58
Improved GAN [23]	8.09 ± .07
AC-GAN [20]	8.25 ± .07
SGAN-no-joint [11]	8.37 ± .08
WGAN-GP ResNet (ours)	8.42 ± .10
SGAN [11]	8.59 ± .12

Source

In an independent study from the Google Brain, WGAN and WGAN-GP do achieve some of the best FID score (the lower the better).



	MNIST	FASHION	CIFAR	CELEBA
MM GAN	9.8 ± 0.9	29.6 ± 1.6	72.7 ± 3.6	65.6 ± 4.2
NS GAN	6.8 ± 0.5	26.5 ± 1.6	58.5 ± 1.9	55.0 ± 3.3
LSGAN	7.8 ± 0.6*	30.7 ± 2.2	87.1 ± 47.5	53.9 ± 2.8*
WGAN	6.7 ± 0.4	21.5 ± 1.6	55.2 ± 2.3	41.3 ± 2.0
WGAN GP	20.3 ± 5.0	24.5 ± 2.1	55.8 ± 0.9	30.0 ± 1.0
DRAGAN	7.6 ± 0.4	27.7 ± 1.2	69.8 ± 2.0	42.3 ± 3.0
BEGAN	13.1 ± 1.0	22.9 ± 0.9	71.4 ± 1.6	38.9 ± 0.9
VAE	23.8 ± 0.6	58.7 ± 1.2	155.7 ± 11.6	85.7 ± 3.8

Source

Further readings

For those want to understand GAN better:

GAN — A comprehensive review into the gangsters of GANs (Part 1)

Are we there yet? In this GAN series, we identify a general pattern on how GAN is applied to deep learning problems and…

medium.com

For all articles in the GAN series:

GAN — GAN Series (from the beginning to the end)

A full listing of our articles covers the applications of GAN, the issues, and the solutions.

medium.com

Reference

Wasserstein GAN

WGAN-GP: Improved Training of Wasserstein GANs

Towards principled methods for training Generative Adversarial Networks