Wasserstein GANs and GAN Hacks

Pavlos Protopapas

Outline

- Wasserstein GAN
- Wasserstein-GP GAN
- GAN Hacks

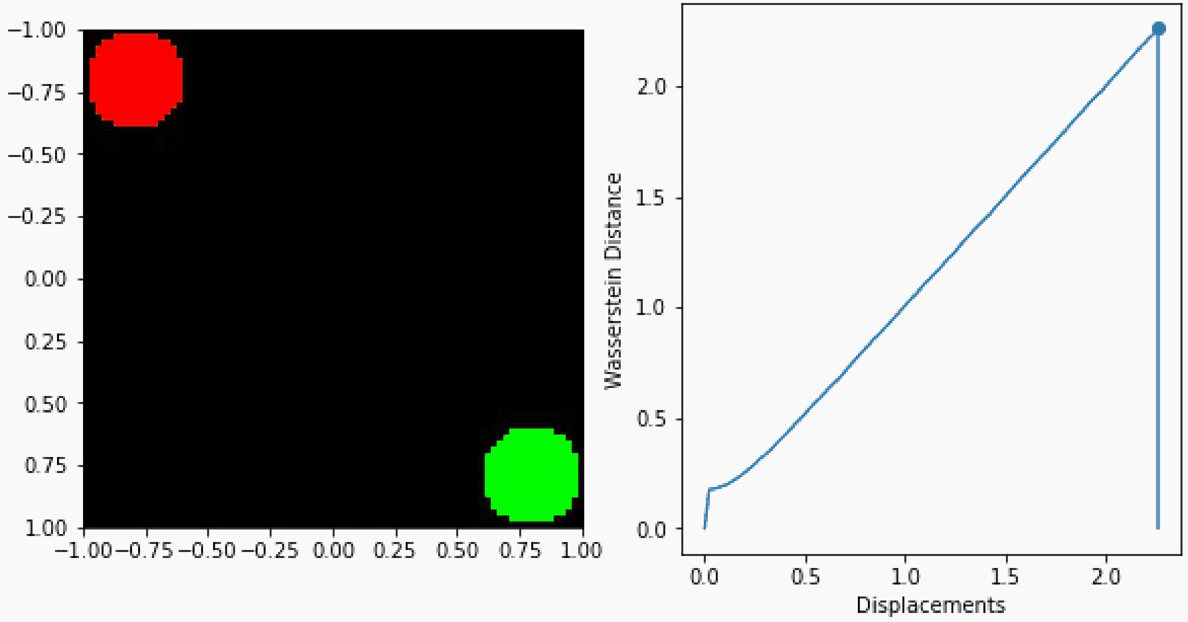
Outline

- **Wasserstein GAN**
- Wasserstein-GP GAN
- GAN Hacks

Different Distances

Distance is everything.

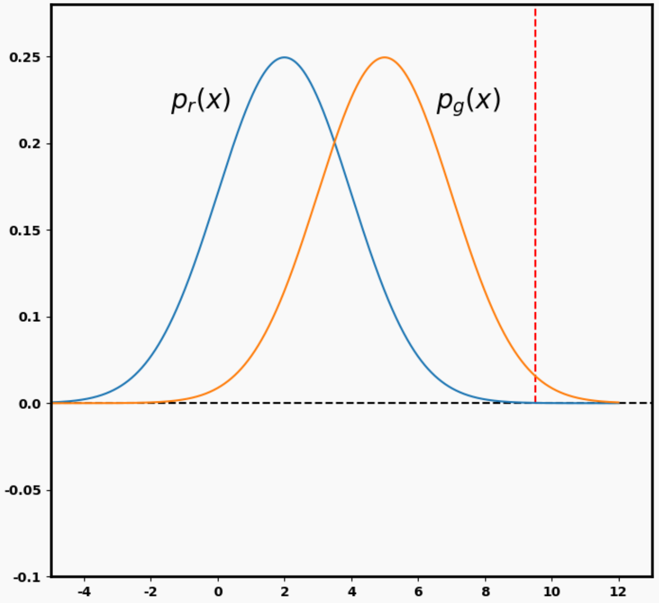
In general, generative models seek to minimize the distance between **real and learned distribution**.

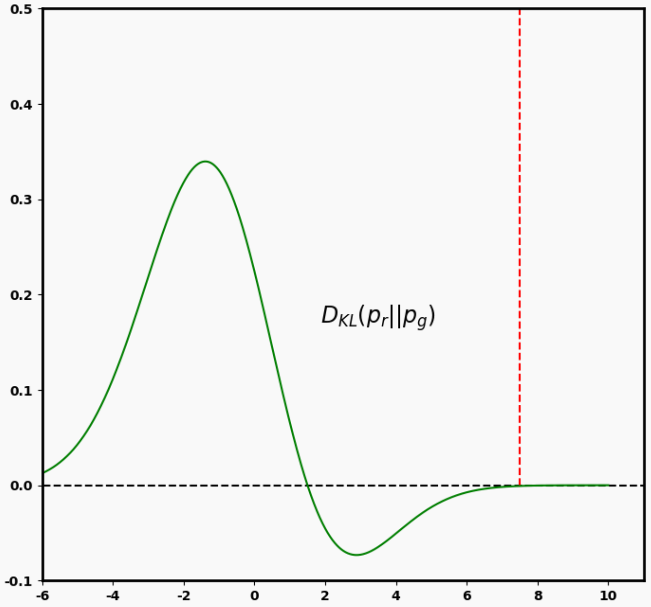


Different Distances: KL Divergence

1. Kullback-Leibler (KL) Divergence:

$$D_{KL}(p_r \parallel p_g) = \int p_r(x) \log\left(\frac{p_r(x)}{p_g(x)}\right) dx$$

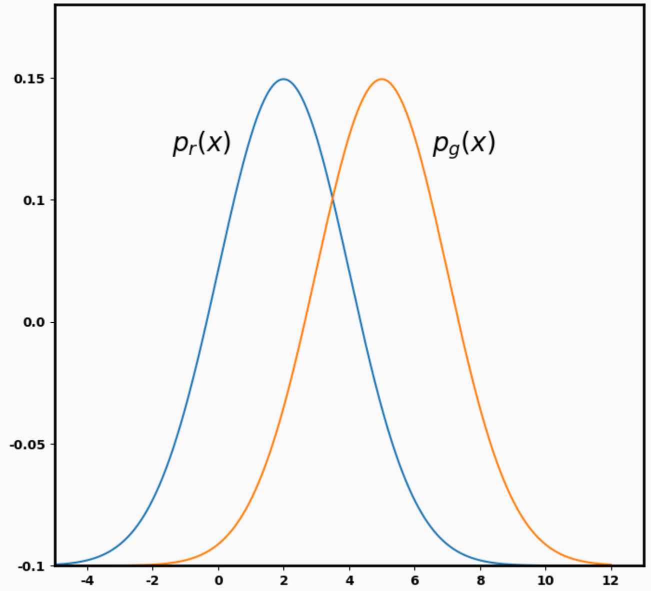


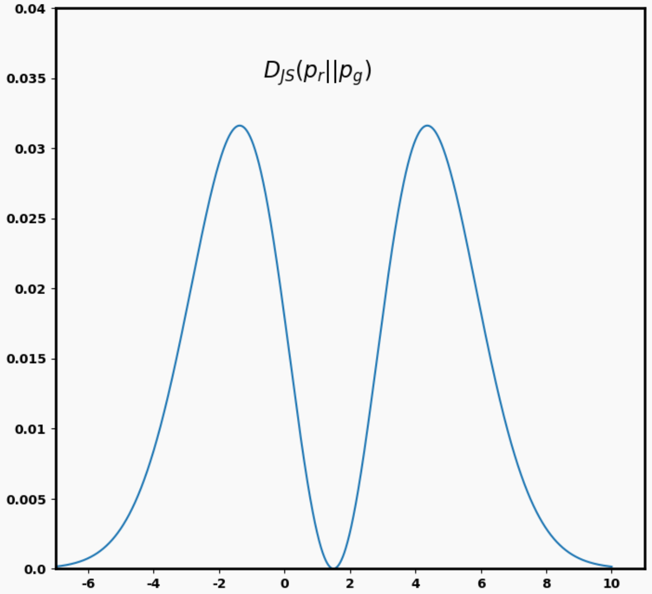


Different Distances: JS Divergence

2. Jensen Shannon (JS) Divergence:

$$D_{JS}(p_r \parallel p_g) = \frac{1}{2} D_{KL}\left(p_r \parallel \frac{p_r + p_g}{2}\right) + \frac{1}{2} D_{KL}\left(p_g \parallel \frac{p_r + p_g}{2}\right)$$





Different Distances: Wasserstein Distance

3. Earth movers/Wasserstein Distance:

Imagine we started with a distribution p_g and wanted to move mass around to change it into p_r. Moving mass m by distance $d = |\mathrm{x} - \mathrm{y}]$ would cost $m \cdot d$.
To execute this for all (x, y), move $\gamma(x, y)$ mass from x to y.

Then W-distance is:

$$W(p_r, p_g) = \inf_{\gamma \epsilon \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\| x - y \|\right]$$

What is inf?

What is γ?

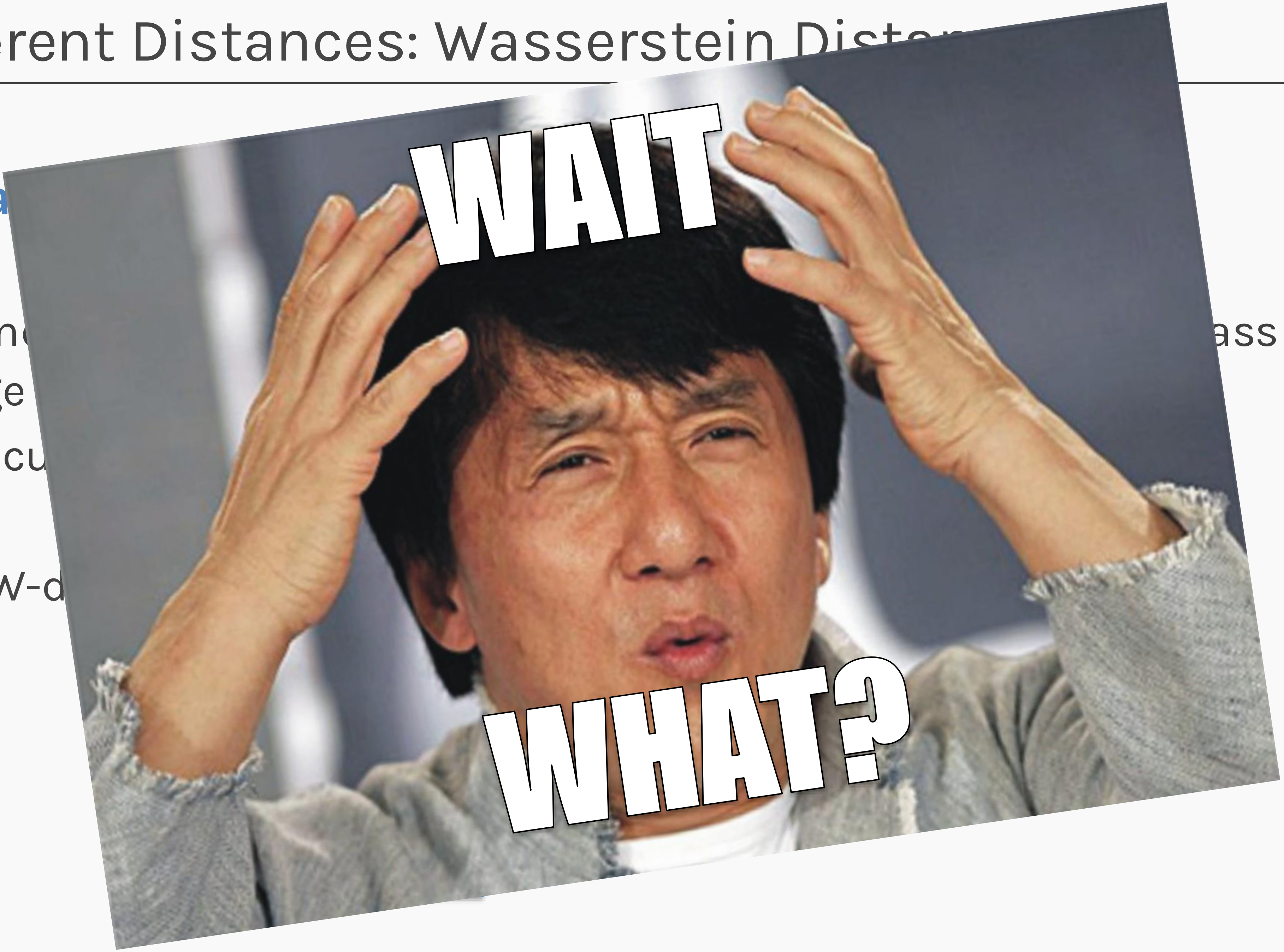
Different Distances: Wasserstein Distan

3. Ea

Imagin ass around to change

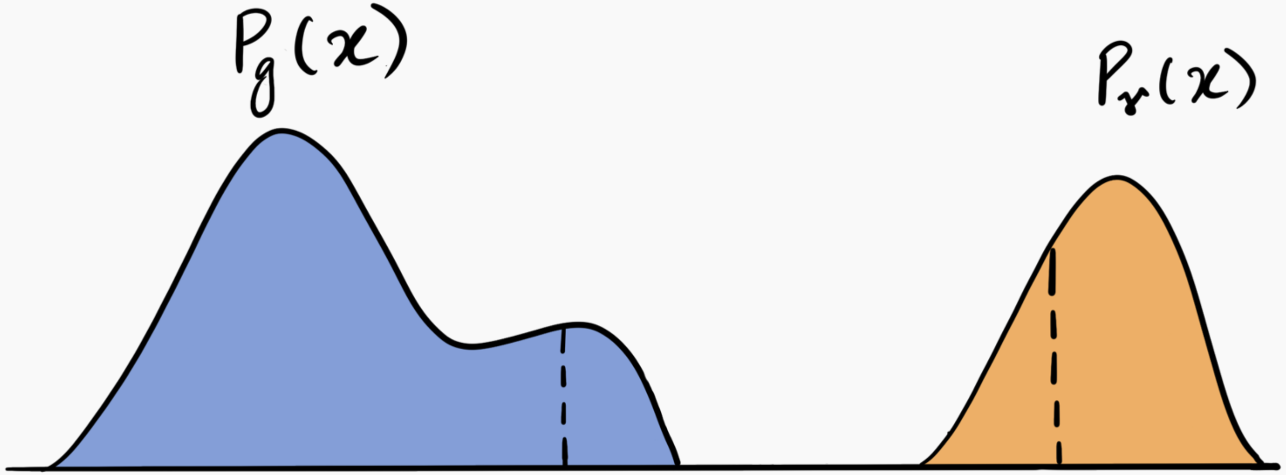
To execu

Then W-d



Different Distances: Wasserstein Distance

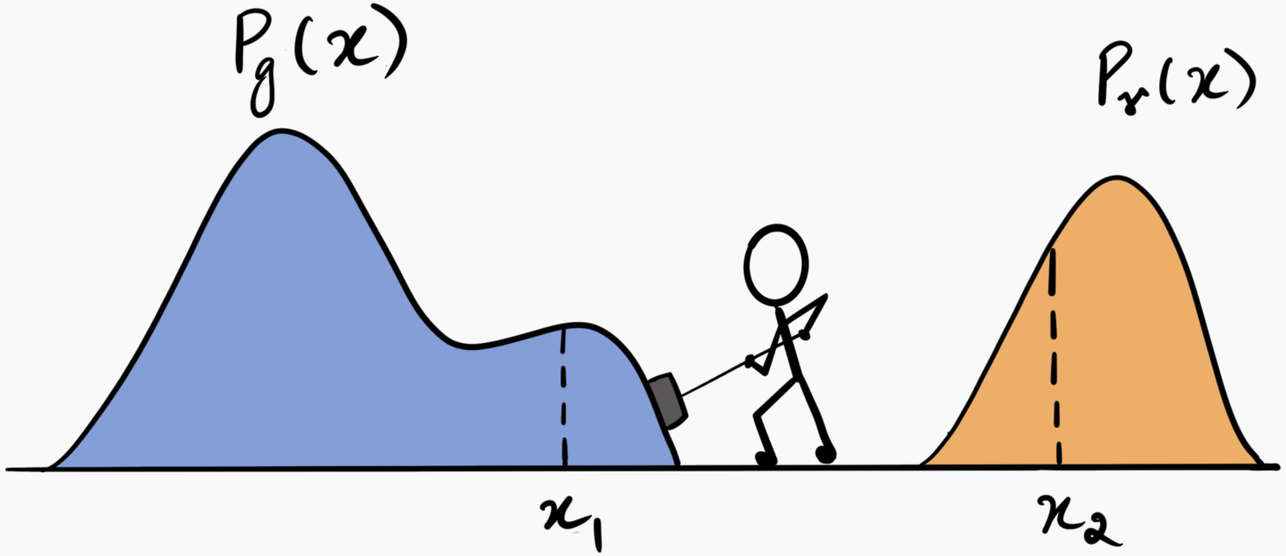
The real and generated distributions can be interpreted as two different ways of piling up a certain amount of dirt.



Different Distances: Wasserstein Distance

The real and generated distributions can be interpreted as two different ways of piling up a certain amount of dirt.

The Wasserstein distance is the minimum cost of turning one pile into the other.

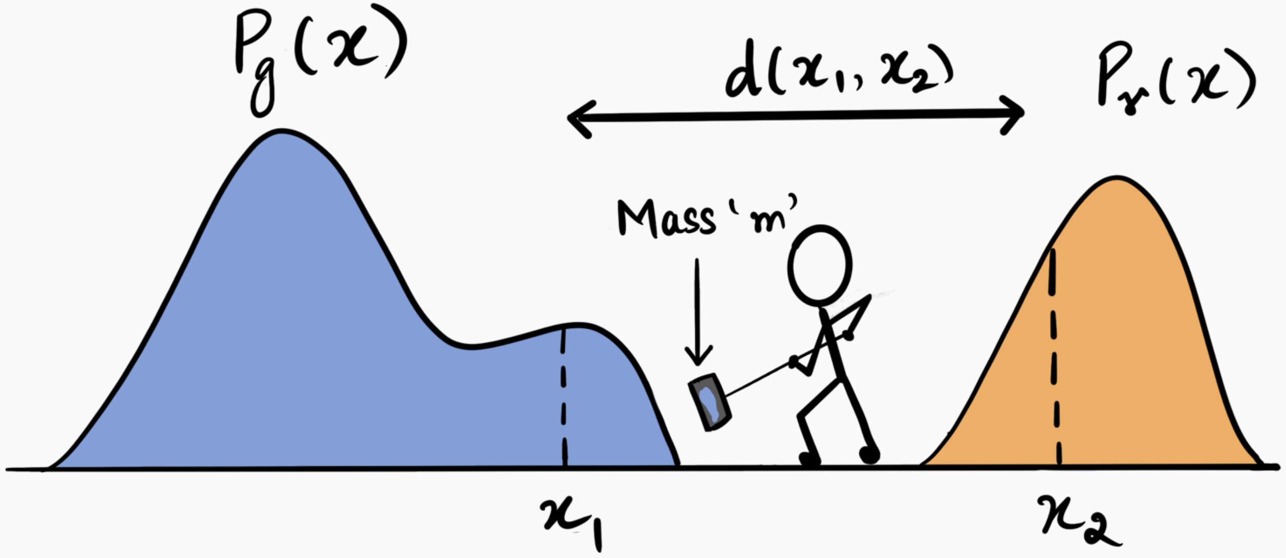


Different Distances: Wasserstein Distance

The real and generated distributions can be interpreted as two different ways of piling up a certain amount of dirt.

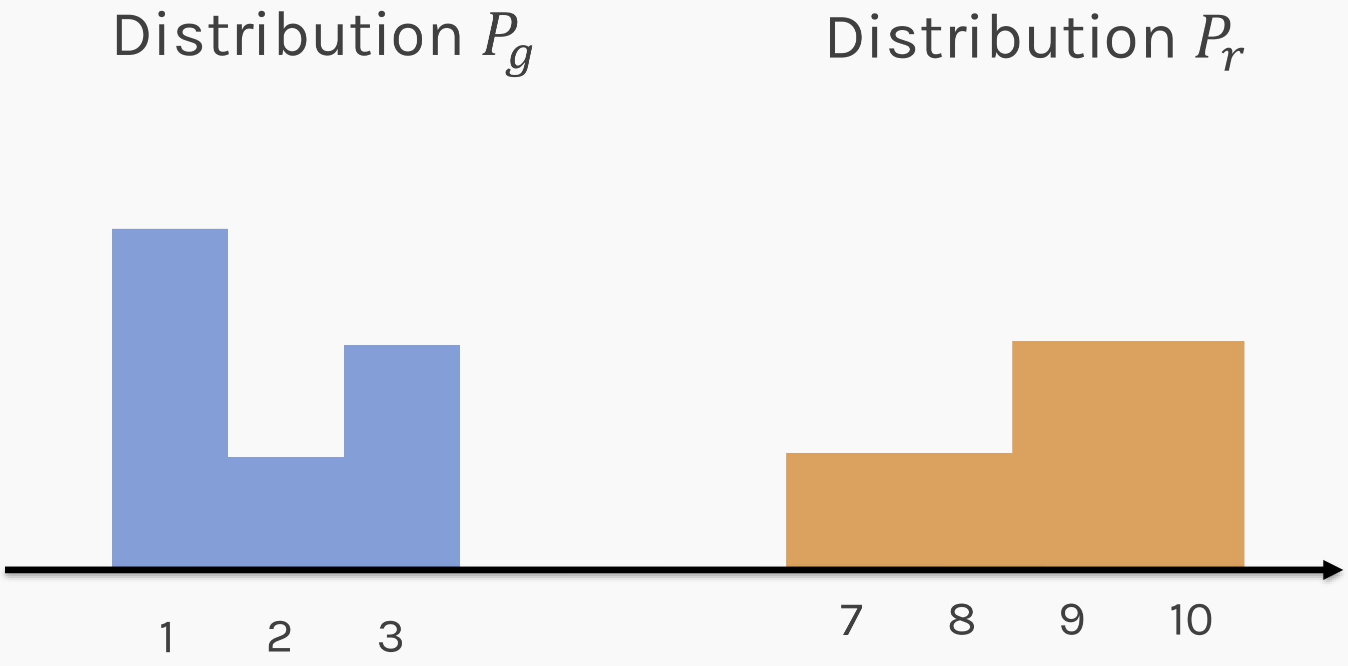
The Wasserstein distance is the minimum cost of turning one pile into the other.

The cost is assumed to be the **amount of dirt moved times the distance by which it is moved**.



Different Distances: Wasserstein Distance

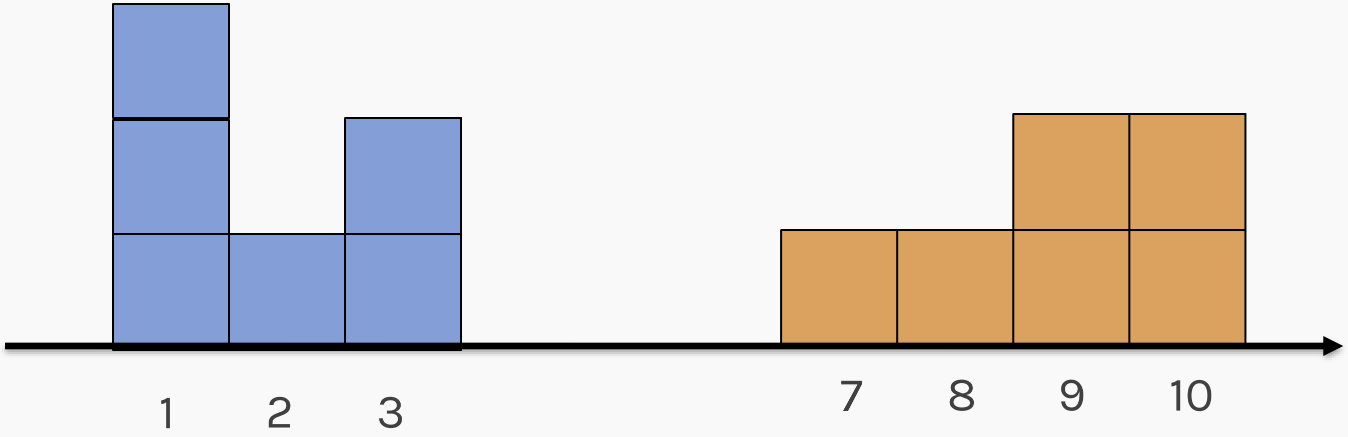
Starting with two distributions, P_g and P_r.



Different Distances: Wasserstein Distance

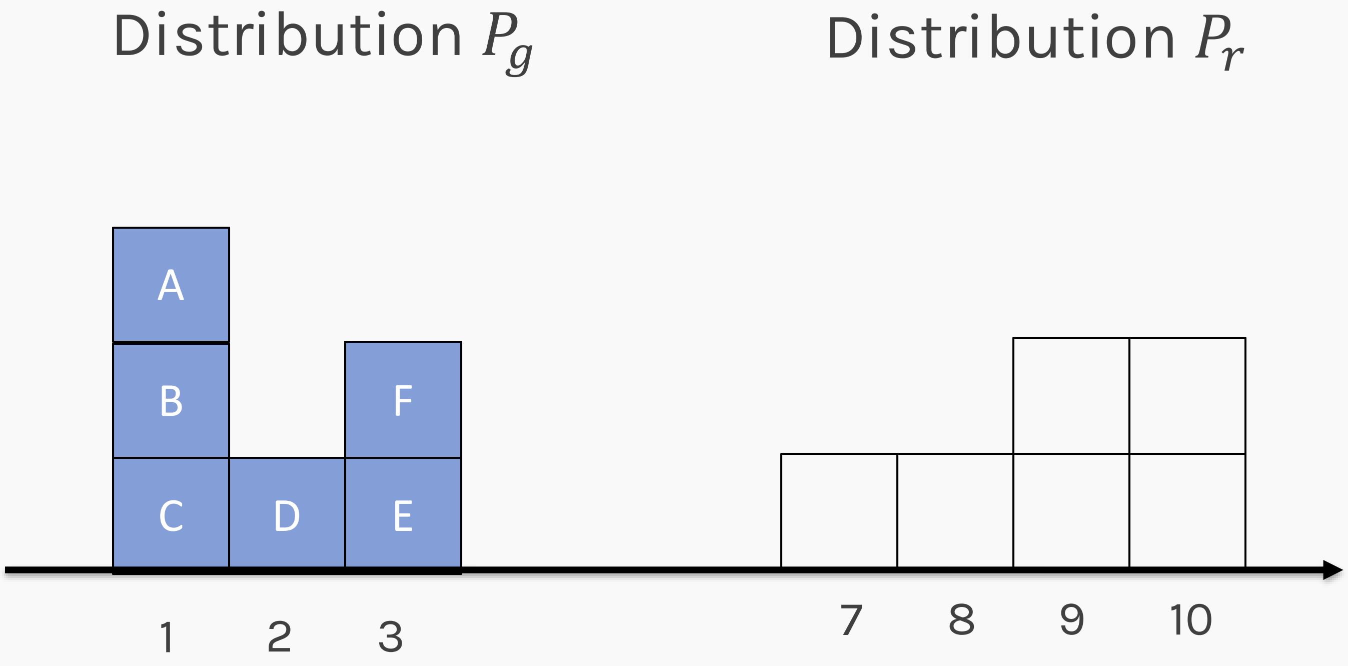
We think of the distribution as a histogram with boxes.

Distribution P_g Distribution P_r



Different Distances: Wasserstein Distance

The goal here is to move boxes from P_g in order to reproduce P_r.

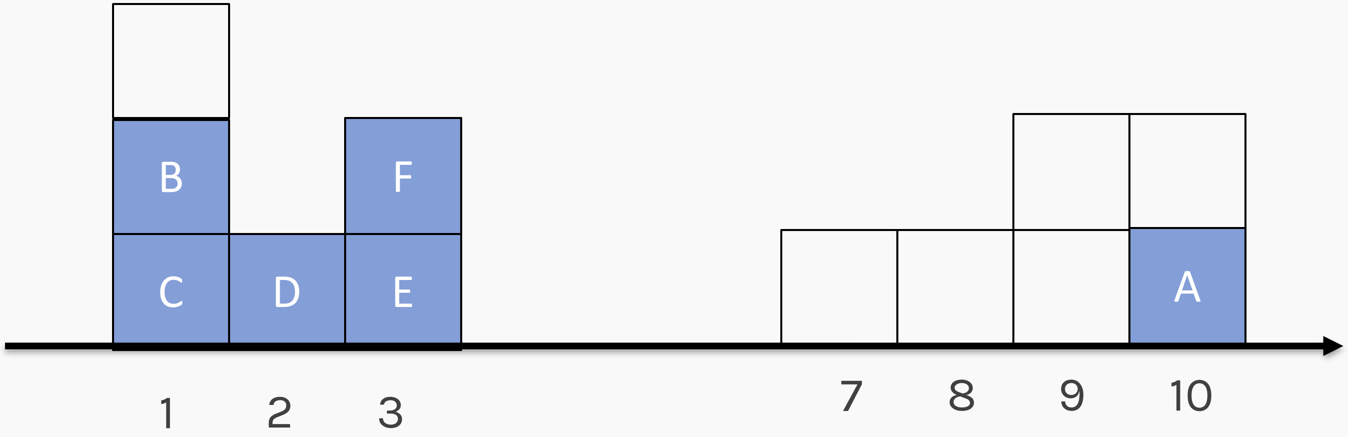


Different Distances: Wasserstein Distance

The goal here is to move boxes from P_g in order to reproduce P_r.

Distribution P_g

Distribution P_r



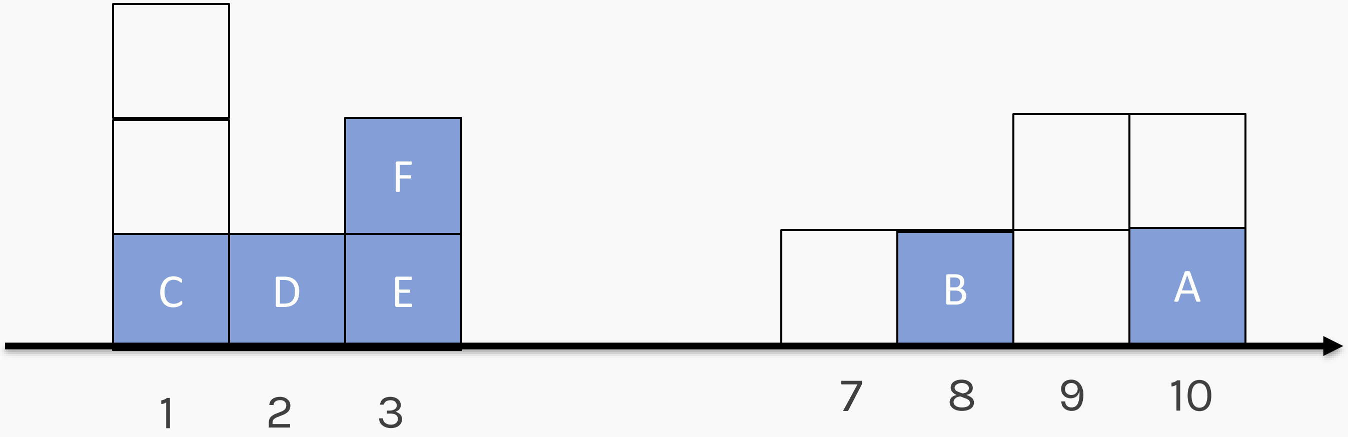
Cost: mass x distance = 1 x (10-1) = 9

Different Distances: Wasserstein Distance

The goal here is to move boxes from P_g in order to reproduce P_r.

Distribution P_g

Distribution P_r

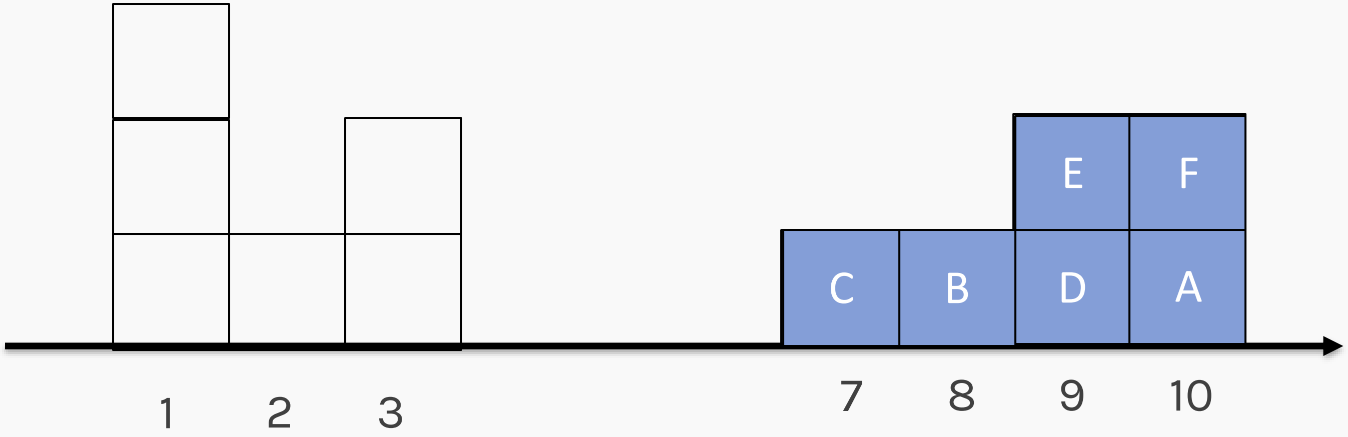


Cost: mass x distance = 9 + (1 x (8-1)) = 9 + 7

Different Distances: Wasserstein Distance

The goal here is to move boxes from P_g in order to reproduce P_r.

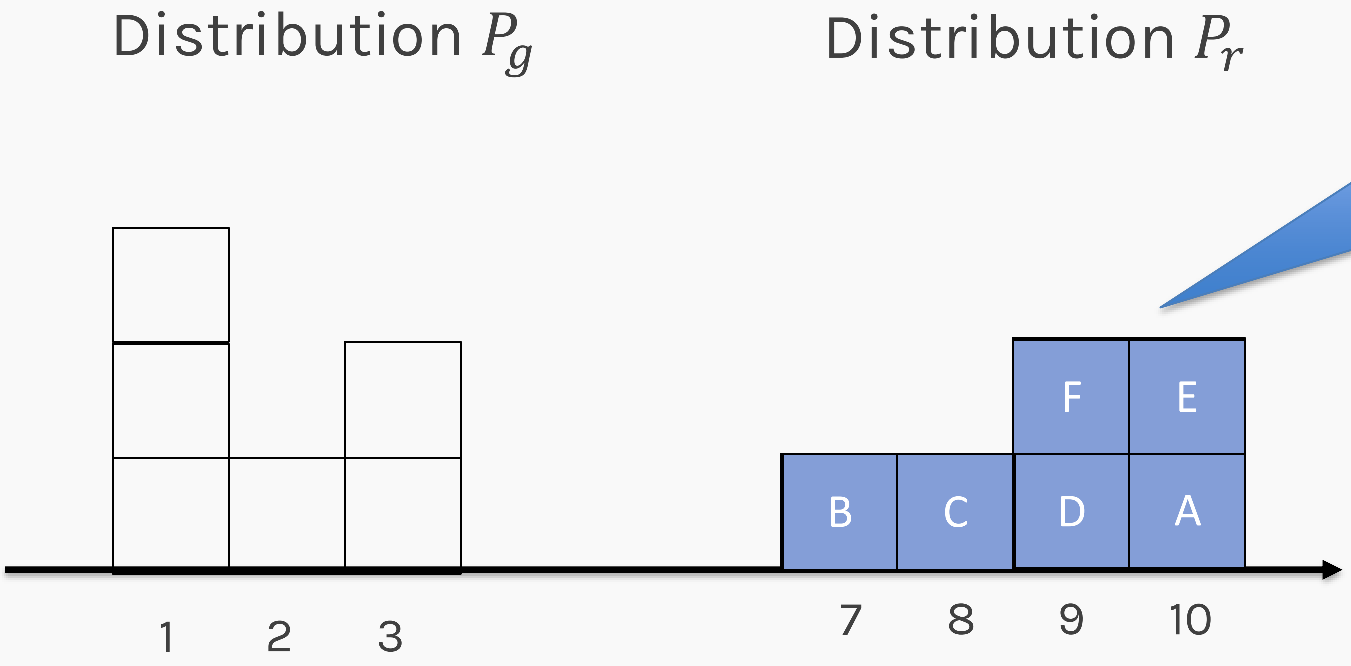
Distribution P_g Distribution P_r



Cost: mass x distance = 9 + 7 + 6 + 7 + 6 + 7= 42

Different Distances: Wasserstein Distance

The goal here is to move boxes from P_g in order to reproduce P_r.

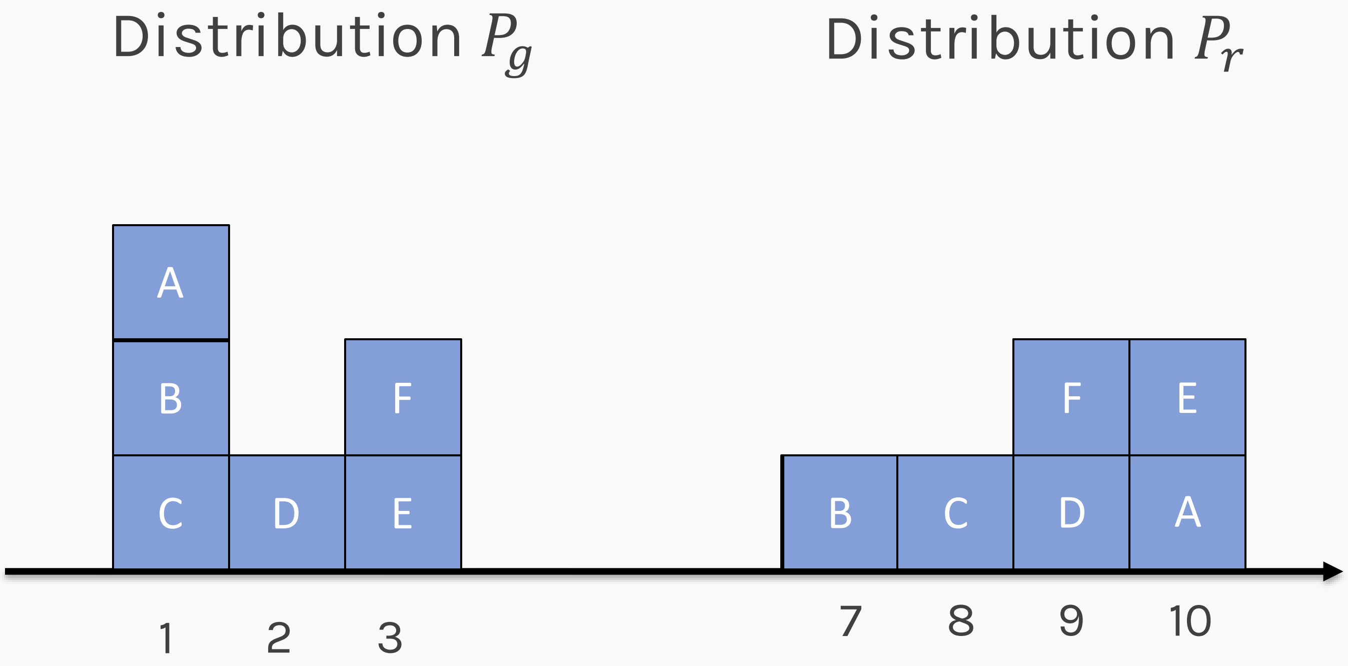


Even if the boxes moved around in a different way, the cost may remain the same.

Cost: mass x distance = 9 + 6 + 7 + 7 + 7 + 6= 42

Different Distances: Wasserstein Distance

What is the transport plan γ?



Distribution P_g

Distribution P_r

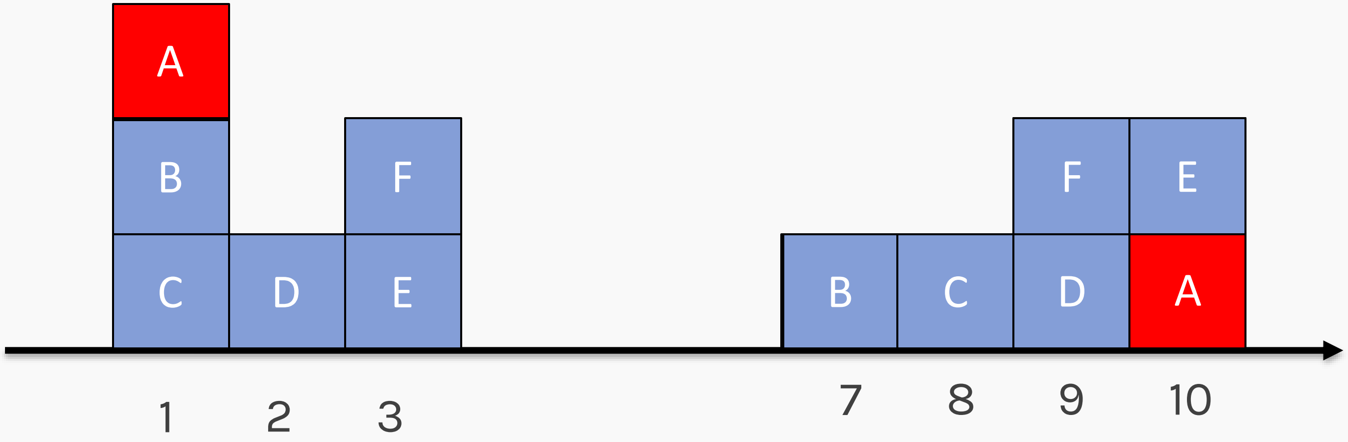
γ

	7	8	9	10
1				
2				
3				

Different Distances: Wasserstein Distance

What is the transport plan γ?

Distribution P_g Distribution P_r



γ

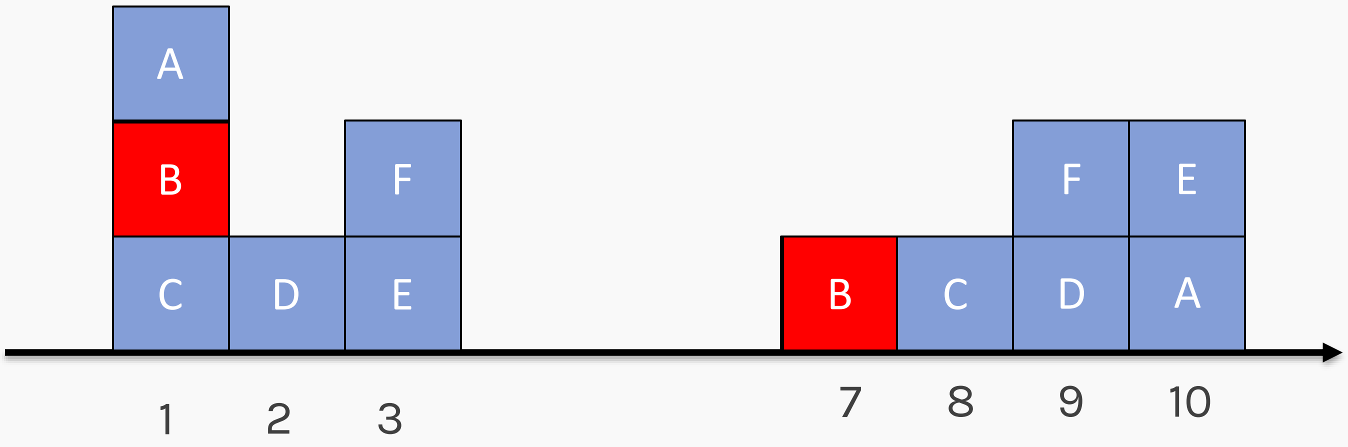
	7	8	9	10
1				1
2				
3				

Different Distances: Wasserstein Distance

What is the transport plan γ?

Distribution P_g

Distribution P_r

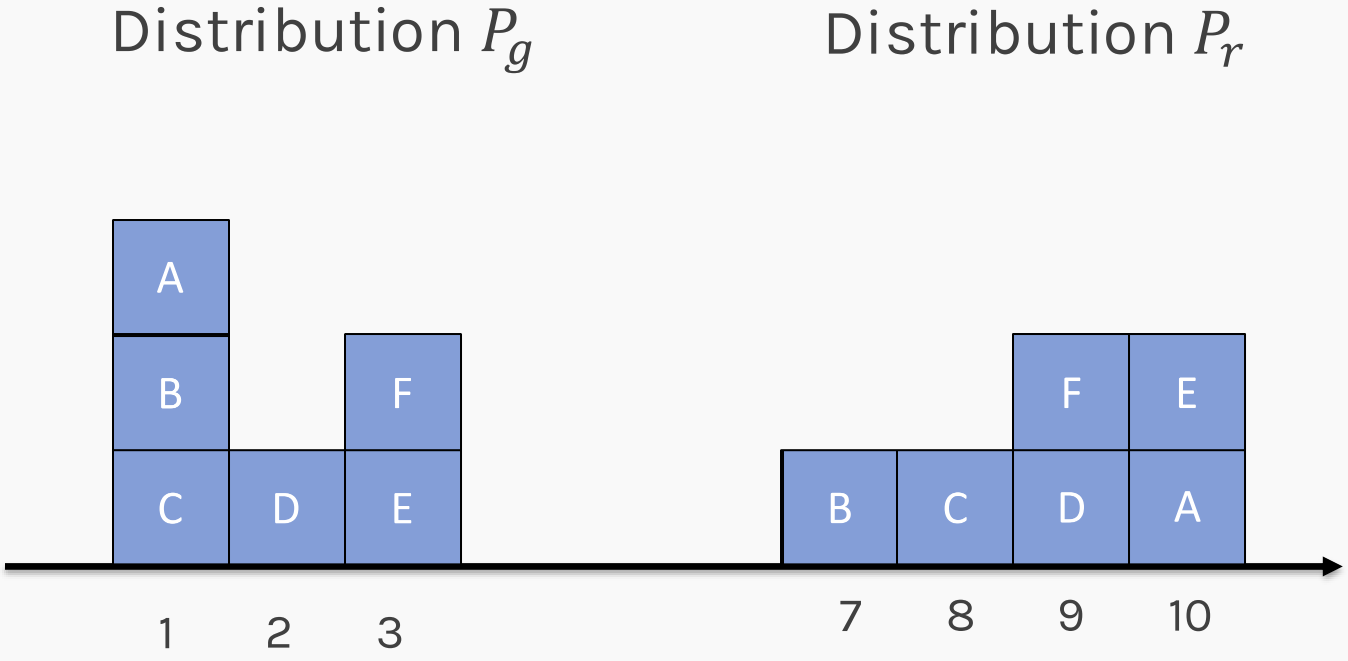


γ

	7	8	9	10
1	1			1
2				
3				

Different Distances: Wasserstein Distance

What is the transport plan γ?



γ

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

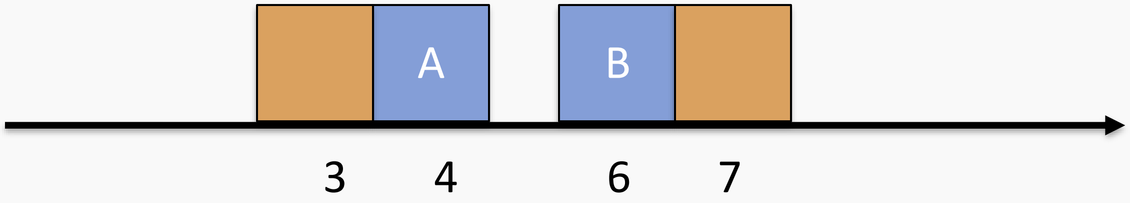
Different Distances: Wasserstein Distance

Not all transport plans bear the same cost.

The Wasserstein distance (or the EM distance) is the cost of the cheapest transport plan.

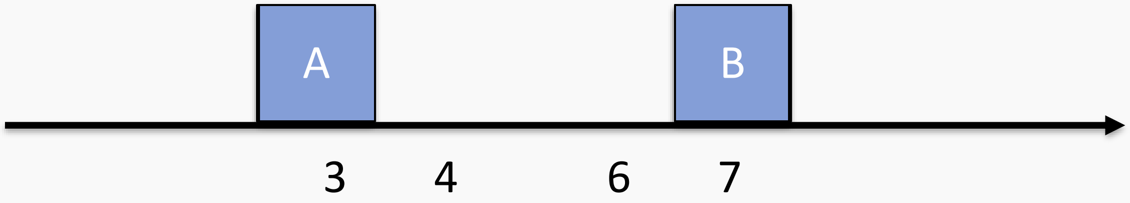
Different Distances: Wasserstein Distance

Plan A



Different Distances: Wasserstein Distance

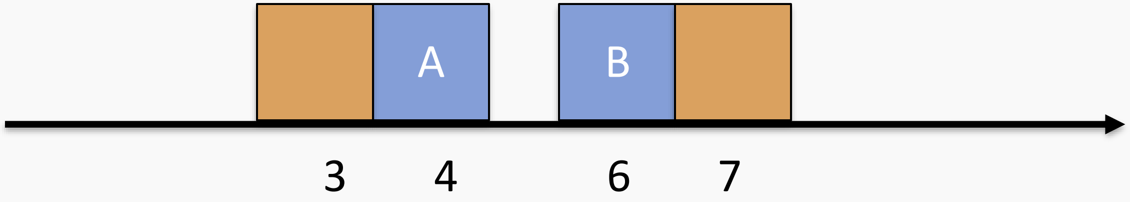
Plan A



Cost: mass x distance = 1 + 1 = 2

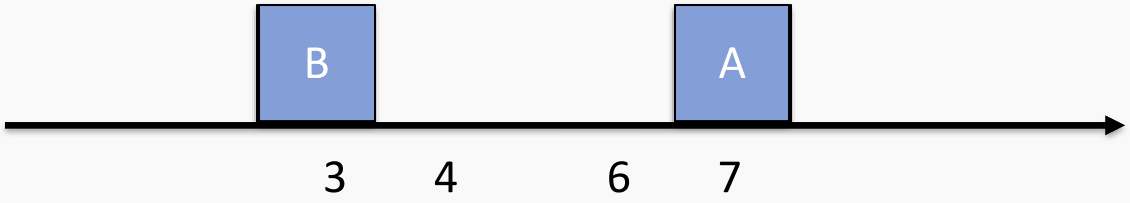
Different Distances: Wasserstein Distance

Plan B



Different Distances: Wasserstein Distance

Plan B

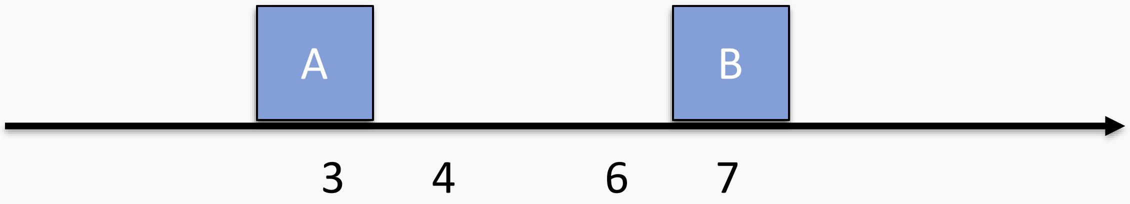


Cost: mass x distance = 3 + 3 = 6

Different Distances: Wasserstein Distance

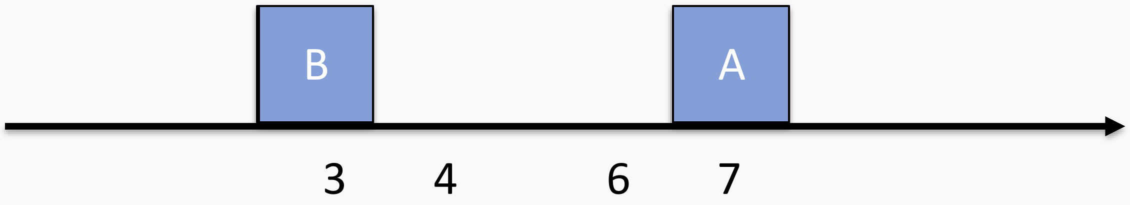
Therefore, between Plan A and Plan B, the Wasserstein distance will follow plan A as the transport plan is cheaper.

Plan A



Cost: mass x distance = 1 + 1 = 2

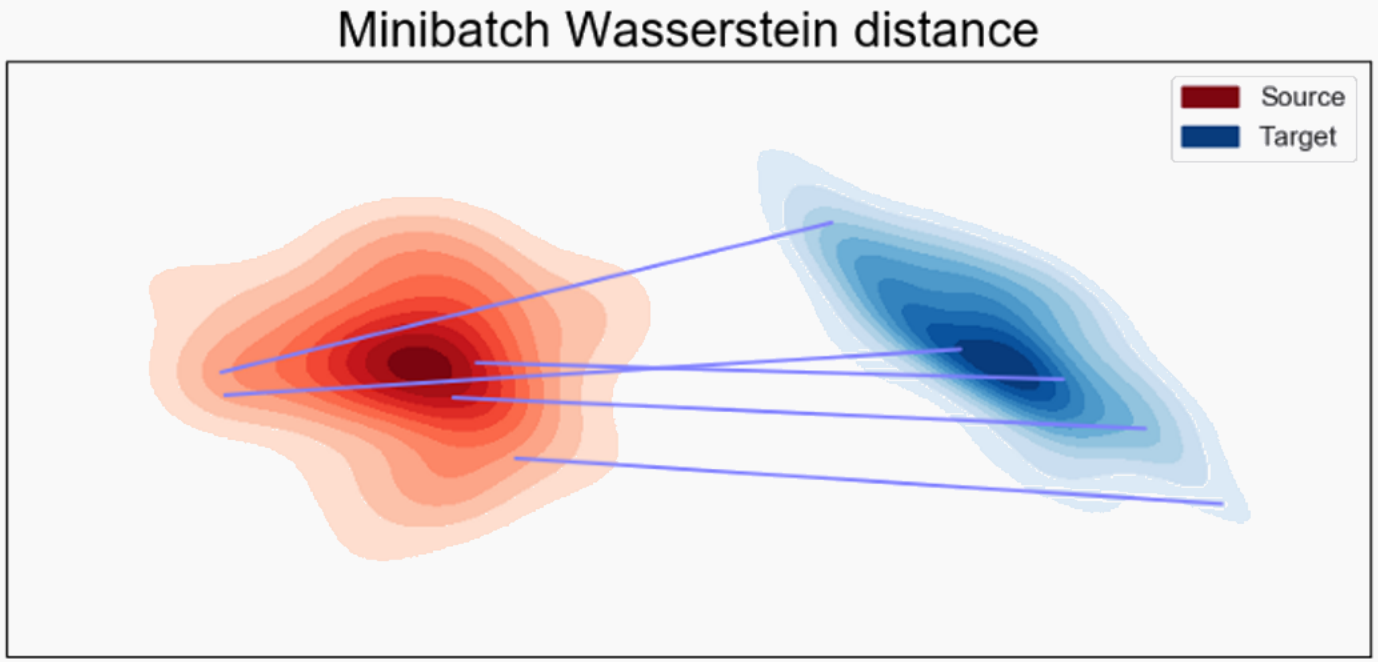
Plan B



Cost: mass x distance = 3 + 3 = 6

Wasserstein GAN

[Arjovsky et al. 2017] proposed a GAN framework based on the Wasserstein or Earth Movers distance.



[Source]

Wasserstein GAN

But why do we need a new distance? What was wrong with the original GAN?

GANs can optimize the discriminator easier than the generator.

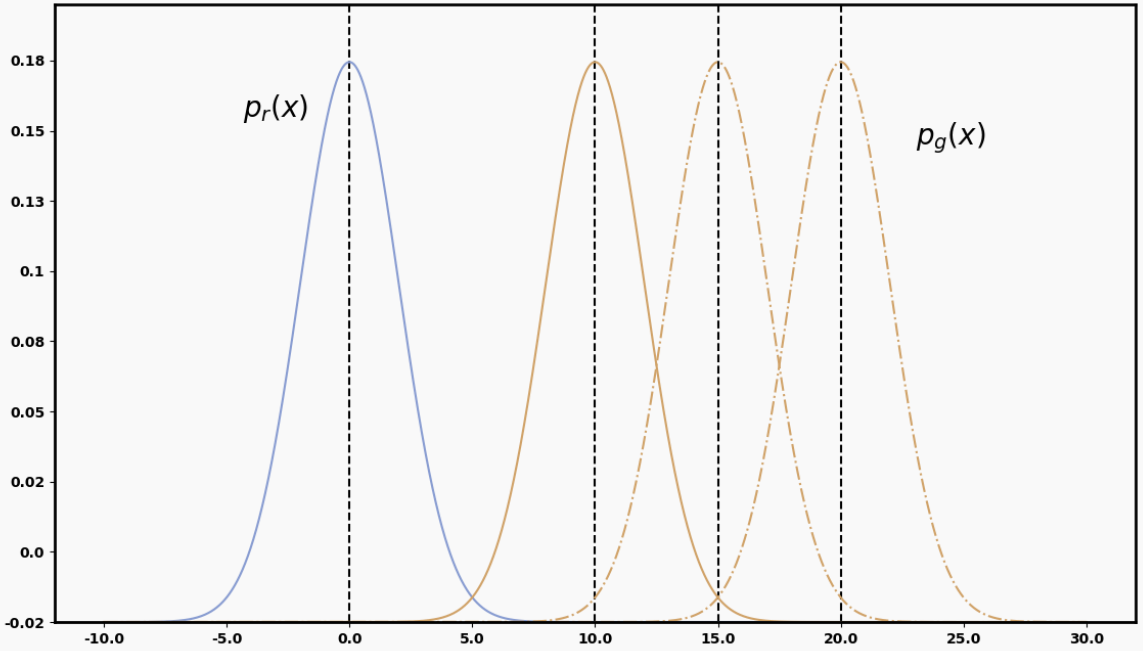
Minimizing the GAN objective function with an optimal discriminator is equivalent to minimizing the JS divergence. The loss function becomes:

$$\min_G V(D_{optimal}, G) = 2D_{JS}\left(p_r \parallel p_g\right) - 2\log(2)$$

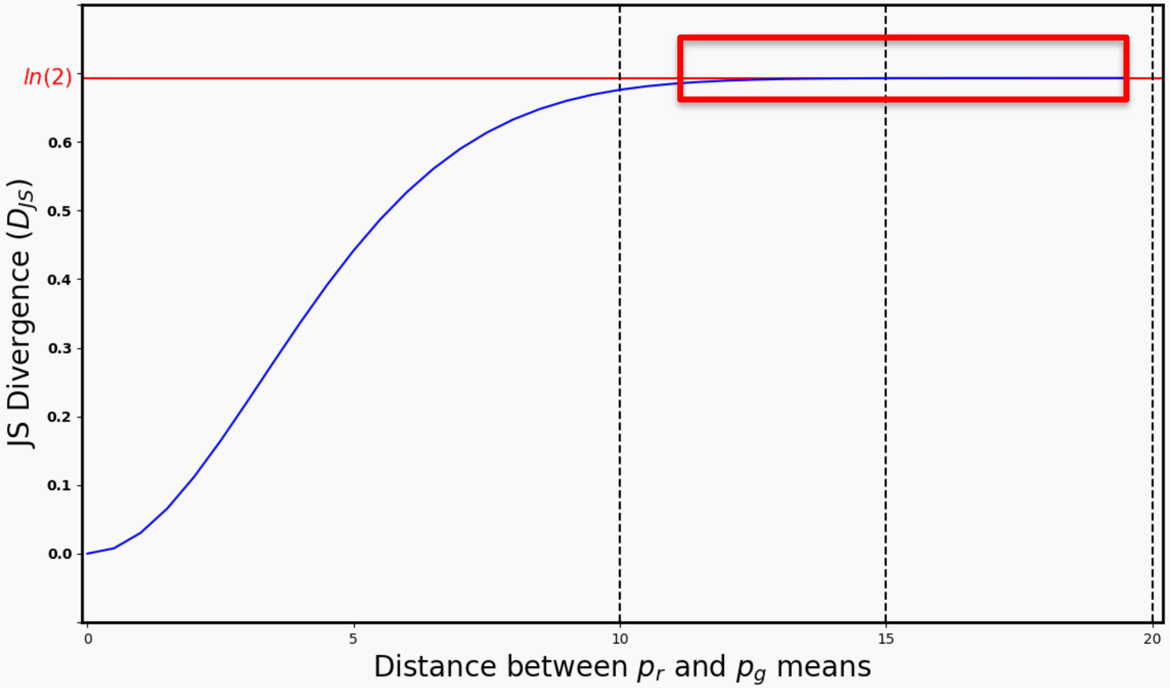
Remember: $D_{JS}\left(p_r \parallel p_g\right) = \frac{1}{2} D_{KL}\left(p_r \parallel \frac{p_r + p_g}{2}\right) + \frac{1}{2} D_{KL}\left(p_g \parallel \frac{p_r + p_g}{2}\right)$

Wasserstein GAN

If the generated image has distribution p_g far away from the ground truth p_r, the generator barely learns anything. Why?



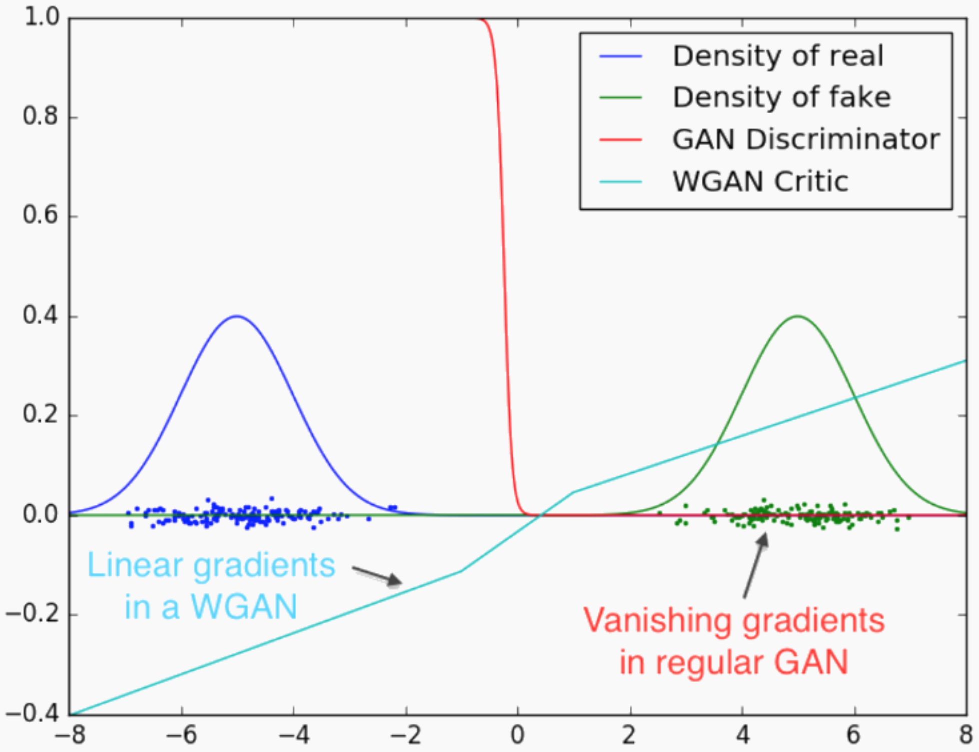
Vanishing Gradient



Wasserstein GAN

The WGAN's new cost function, Wasserstein distance, has a smoother gradient everywhere.

WGAN learns regardless of the generator's performance.



[Source]

Wasserstein GAN

The Wasserstein GAN not only:

- Learns faster because it does not suffer as much from vanishing gradients.

It also:

- Provides higher stability during training so there is less need for carefully balancing generator and discriminator.

- Has a meaningful loss metric that correlates well with sample quality.

- Reduces the occurrence of mode collapse.

Wasserstein GAN

Let $\Pi(p_r,\ p_g)$ be the set of all joint distributions γ whose marginal distributions are p_r and p_g.

Then W-distance is:

Greatest lower bound, or the cost for the cheapest plan.

$$W(p_r, p_g) = \inf_{\gamma \epsilon \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma} [\| x\ - y \|]$$

Wasserstein GAN

$$W\left(p_r, p_g\right) = \inf_{\gamma \epsilon \Pi\left(p_r, p_g\right)} \mathbb{E}_{(x,y)\sim\gamma}\left[\| x - y \|\right]$$

The above formula’s exact computation is intractable.

Instead, we use Kantorovich-Rubinstein duality to get:

$$W\left(p_r, p_g\right) = \sup_{\| f \|_L \leq 1} \mathbb{E}_{x\sim p_r}\left[f(x)\right] - \mathbb{E}_{x\sim p_g}\left[f(x)\right]$$

Least upper bound

f is a 1-Lipschitz function

Wasserstein GAN

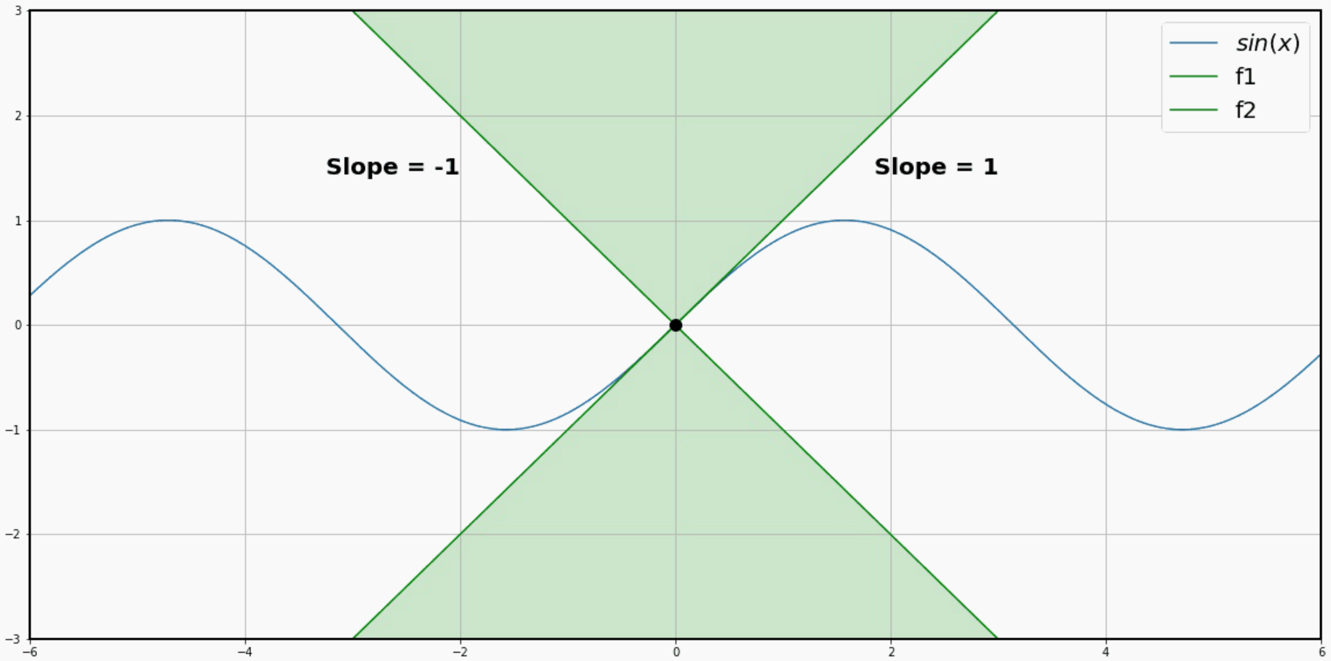
$$W\left(p_r, p_g\right) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_r}\left[f(x)\right] - \mathbb{E}_{x \sim p_g}\left[f(x)\right]$$

A 1-Lipschitz function satisfies:

$$\frac{| f(x_1) - f(x_2)|}{|x_1 - x_2|} \leq 1$$

Wasserstein GAN

In other words, a differentiable function f is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere.



Examples of a 1-Lipschitz function: sin(x), ReLU, LeakyReLU (for leak value less than one).

Wasserstein GAN

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

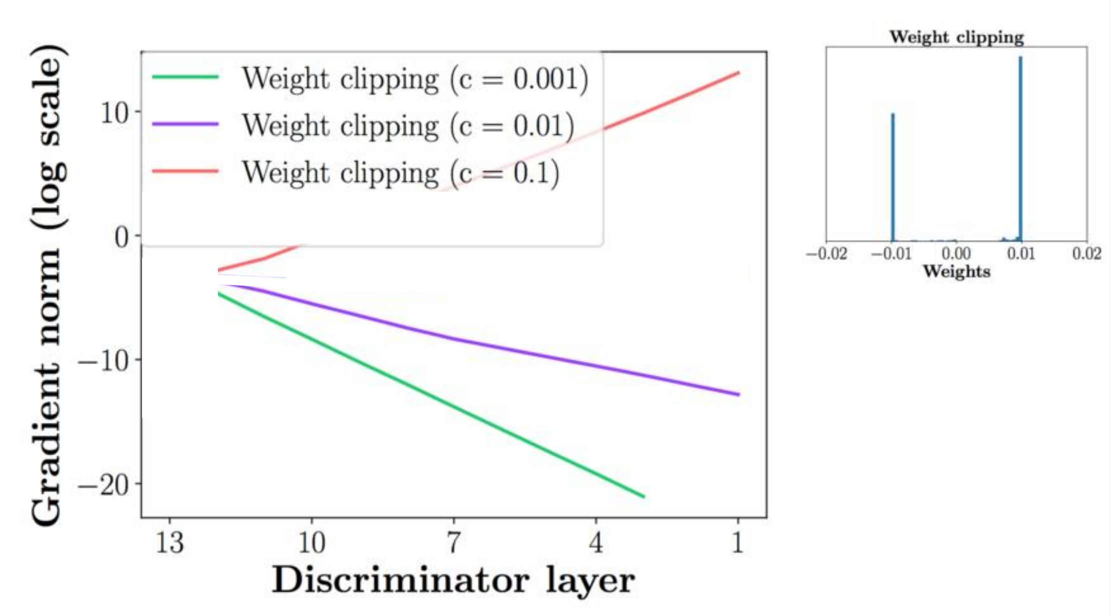
So, to calculate the Wasserstein distance, we just need to find a 1-Lipschitz function f(x). We can build a deep network to learn f(x) and use clipping to enforce the Lipschitz constraint on the critic's model.

This network is like the discriminator ***D,*** just without the sigmoid function and outputs a scalar. This score can be interpreted as how real the input images are.

Wasserstein GAN Issues

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."

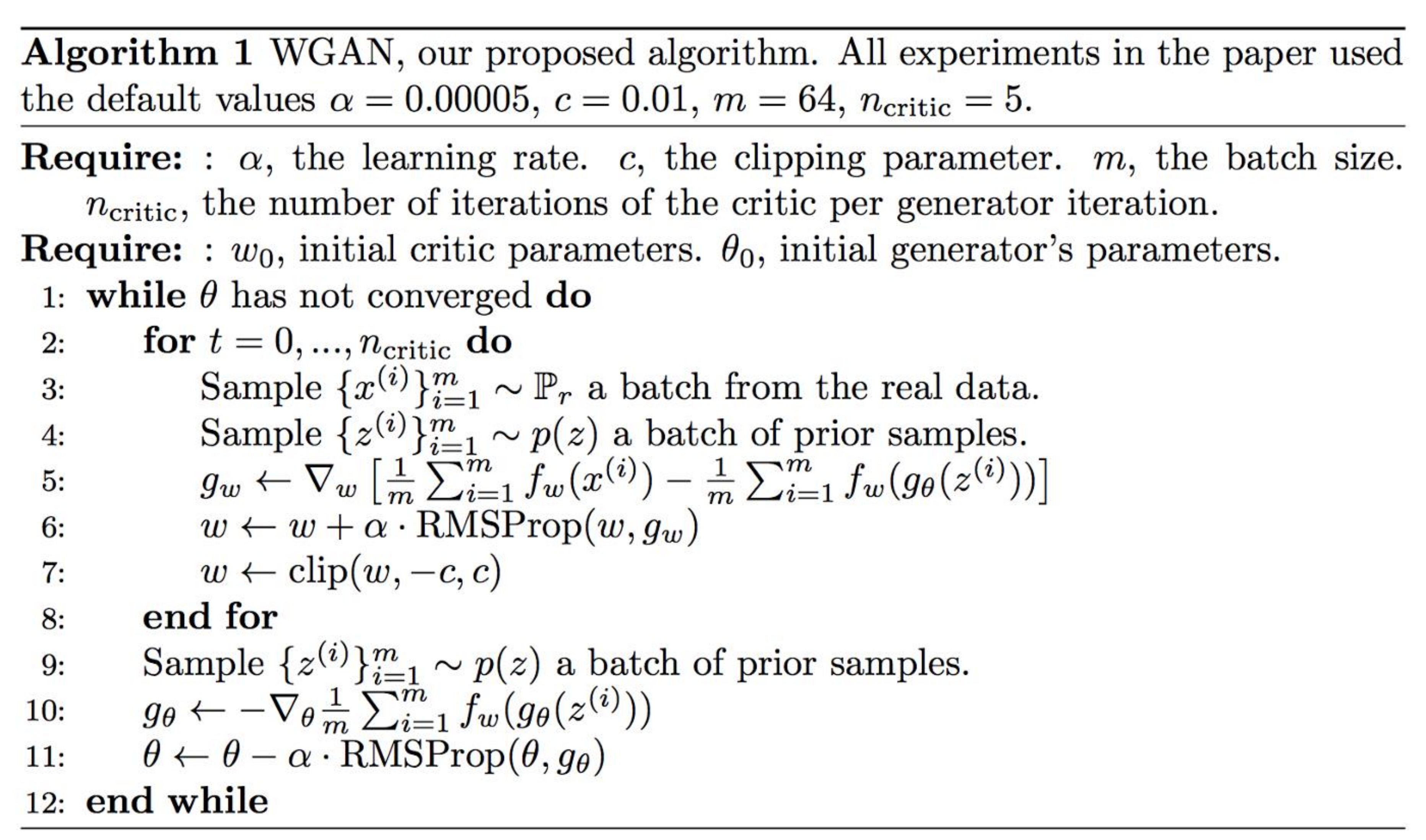


Vanilla GAN vs Wasserstein GAN

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(z^{(i)}\right)\right)\right)$
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, ***f*** has to be a 1-Lipschitz function.

Wasserstein GAN: Algorithm



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: : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size.
    $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
Require: : $w_0$, initial critic parameters. $\theta_0$, 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]

Outline

- Wasserstein GAN
- **Wasserstein-GP GAN**
- GAN Hacks

WGAN with gradient penalty (WGAN-GP)

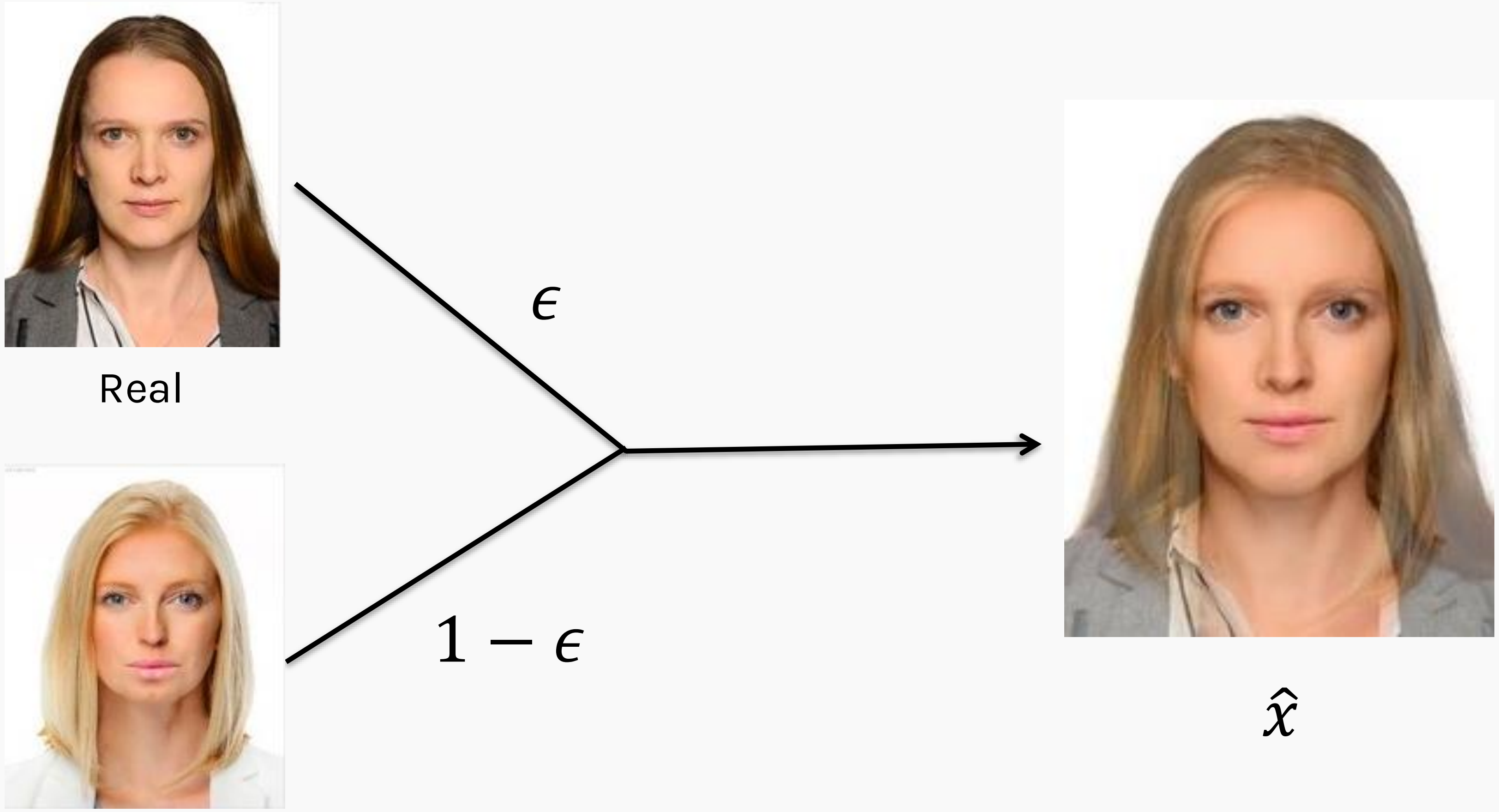
WGAN-GP uses gradient penalty instead of the weight clipping to enforce the Lipschitz constraint.

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

Interpolated Image

WGAN with gradient penalty (WGAN-GP)

Interpolated Image:



Real

ϵ

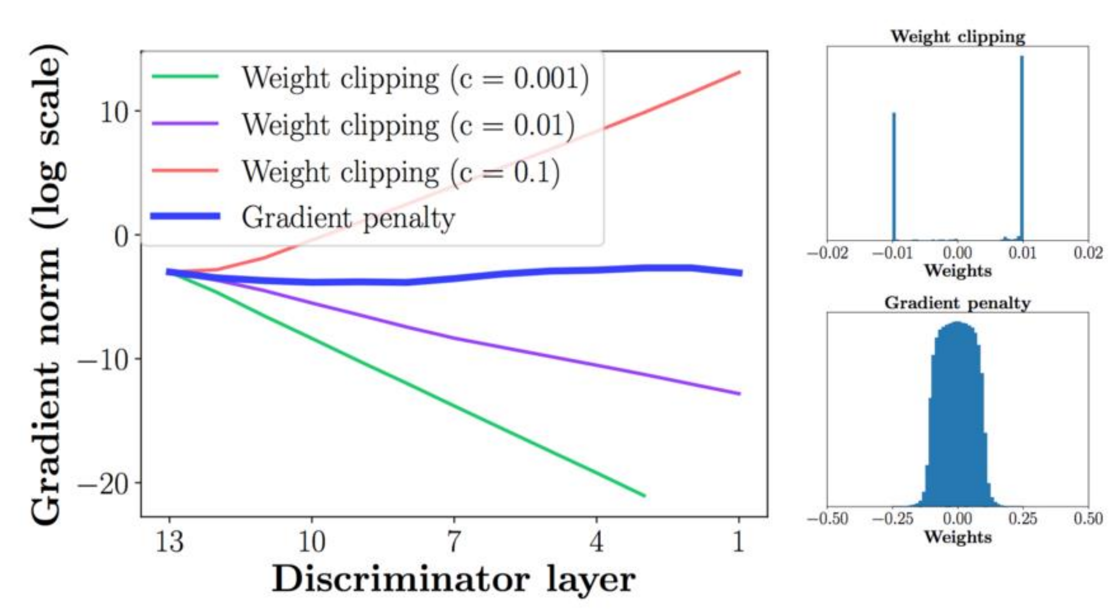
$1-\epsilon$

Fake

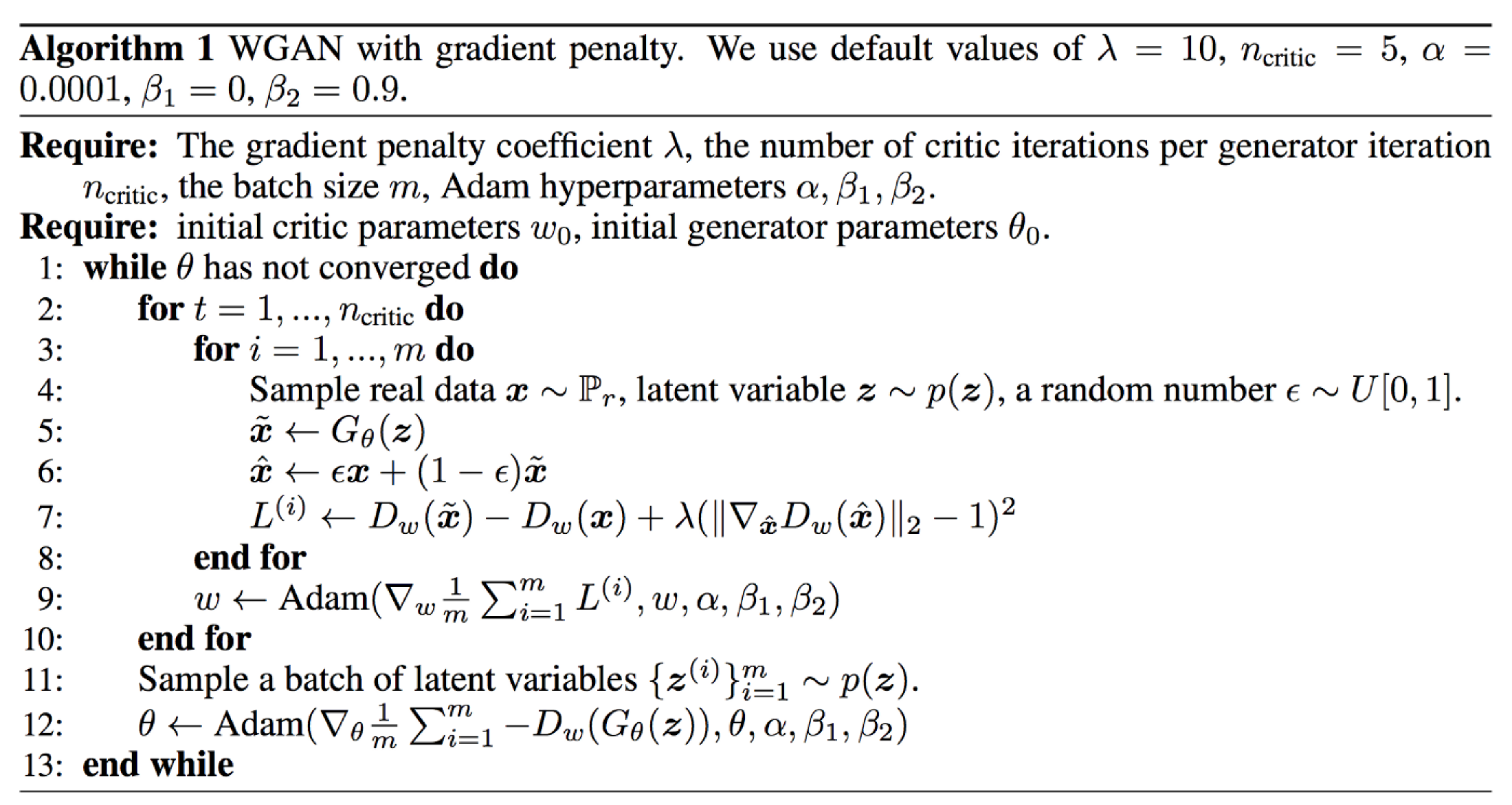
$\hat{x}$

WGAN with gradient penalty (WGAN-GP)

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



WGAN-GP: Algorithm



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 λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.

Require: initial critic parameters w_0, initial generator parameters θ_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]

Spectral Normalization

Method to stabilize the training of the discriminator and restrict its capacity using a novel weight normalization technique:

$$\boxed{\bar{W}_{SN} = W/\sigma(W)}$$

Singular value is very similar to PCA

where $\sigma(W)$ is equivalent to the largest singular value of W.

[Miyato et al. 2018]

Outline

- Wasserstein GAN
- Wasserstein-GP GAN
- **GAN Hacks**

GAN Rules of Thumb (GANHACKs)

Normalize the inputs

- Normalize the images between -1 and 1.
- Use tanh as the last layer of the generator output.

Use Spherical Z

- Don’t sample from a uniform distribution.
- When doing interpolations, do the interpolation via a great circle, rather than a straight line from point A to point B.
  Tom White's Sampling Generative Networks has more details

GAN Rules of Thumb (GANHACKs)

Batch Normalization

- Construct different mini-batches for real and fake, i.e. each mini-batch needs to contain only all real images or all generated images.
- When batch normalization is not an option, use instance normalization. For each sample, subtract mean and divide by standard deviation.

Avoid Sparse Gradients: ReLU, MaxPool

- The stability of the GAN game suffers if you have sparse gradients
- LeakyReLU = good (in both G and D)
- For Downsampling, use: Average Pooling, Conv2d + stride
- For Upsampling, use: ConvTranspose2d + stride

GAN Rules of Thumb (GANHACKs)

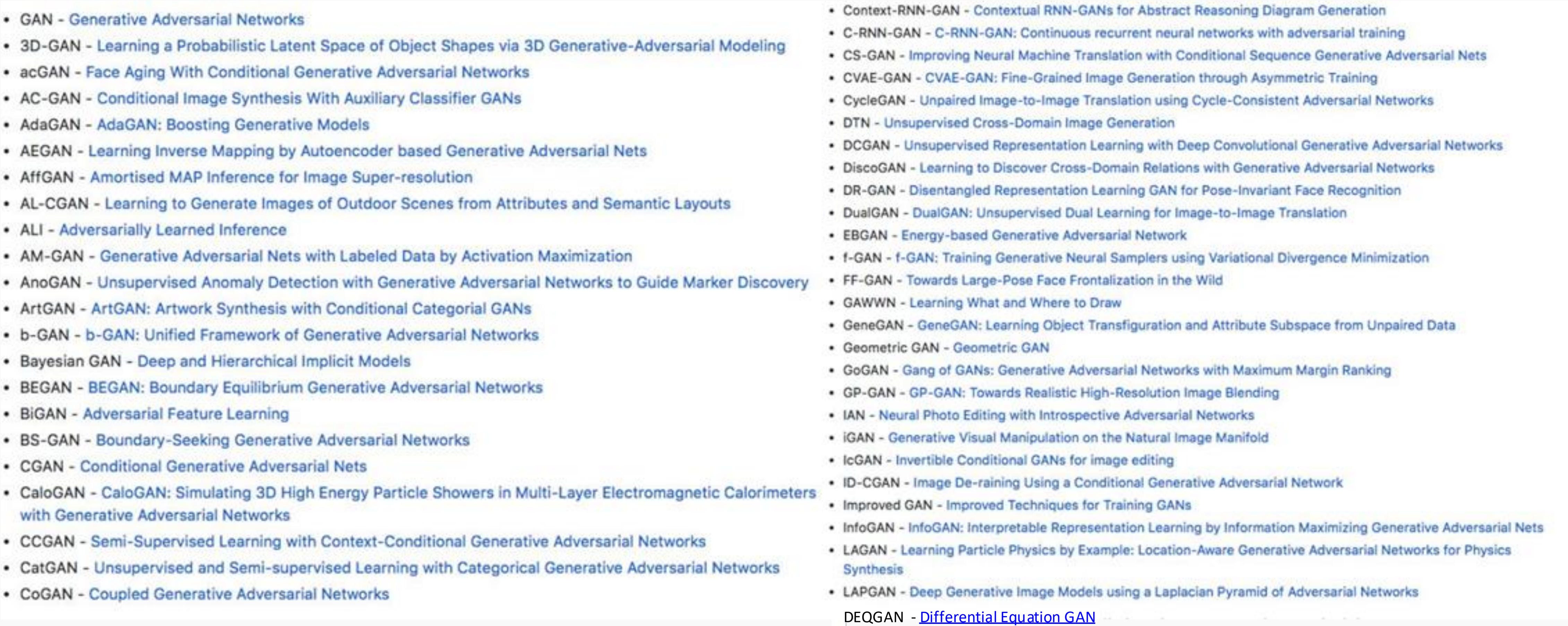
Use Soft and Noisy Labels

- Label Smoothing, i.e. if you have two target labels: Real=1 and Fake=0, then for each incoming sample, if it is real, then replace the label with a random number between 0.7 and 1.2, and if it is a fake sample, replace it with 0.0 and 0.3 (for example).
- Make the labels noisy for the discriminator: occasionally flip the labels when training the discriminator.

See GANHACKs (https://github.com/soumith/ganhacks) for more tips.

The explosion of GANs

The GAN Zoo



- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

DEQGAN - Differential Equation GAN
TCGAN – Time Conditional GAN

Thank you