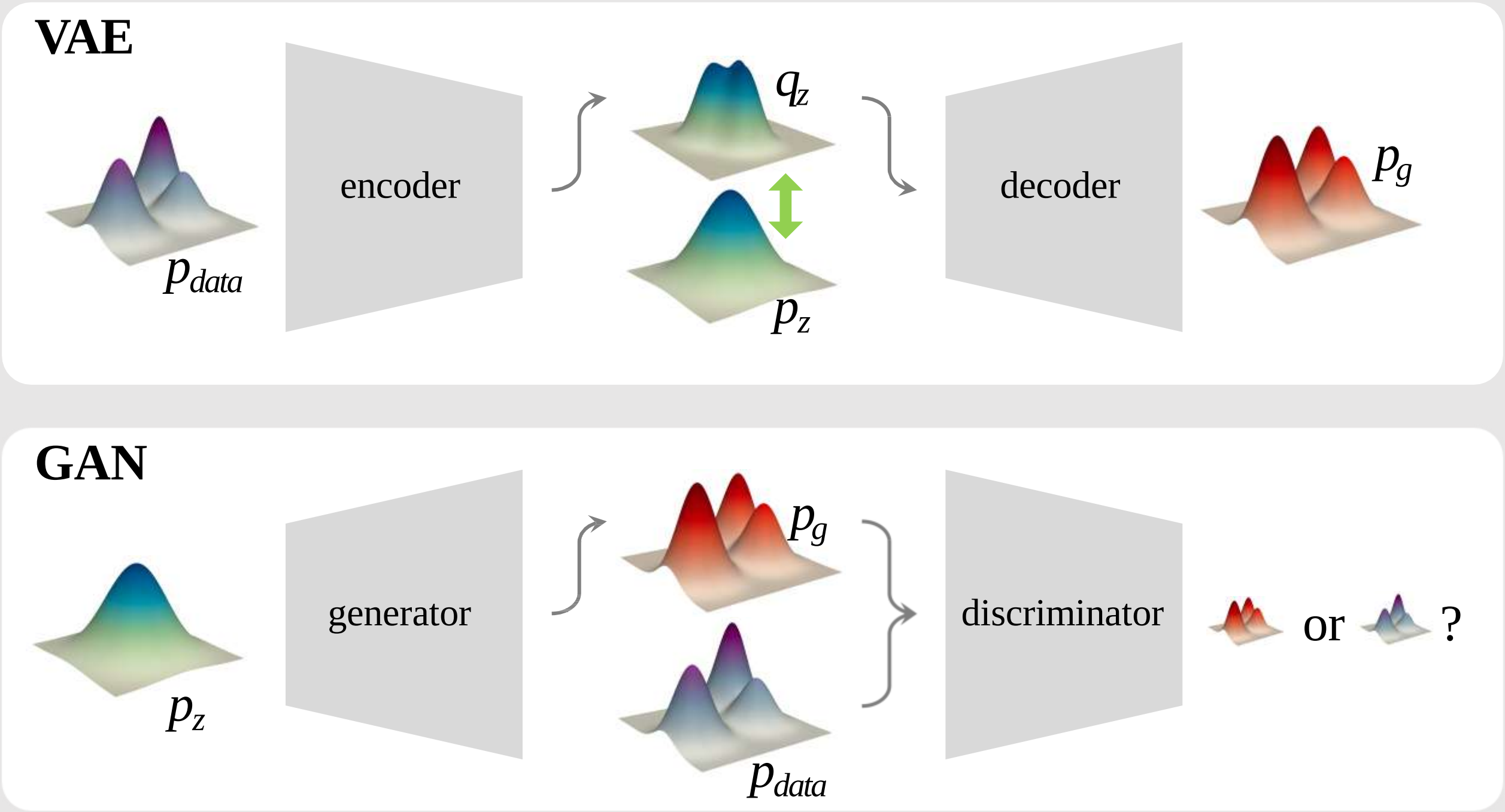
Recap.

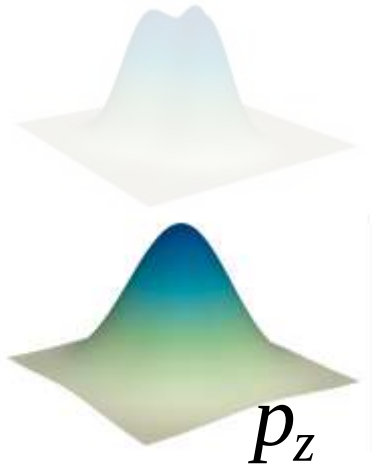

VAE
p_{data}
encoder
q_z
p_z
decoder
p_g
GAN
p_z
generator
p_g
p_{data}
discriminator
or
?

VAE

generation

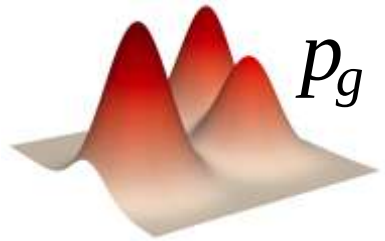
p_{data}

encoder



p_z

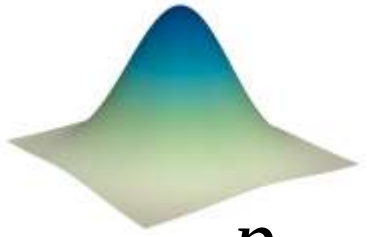
decoder



p_g

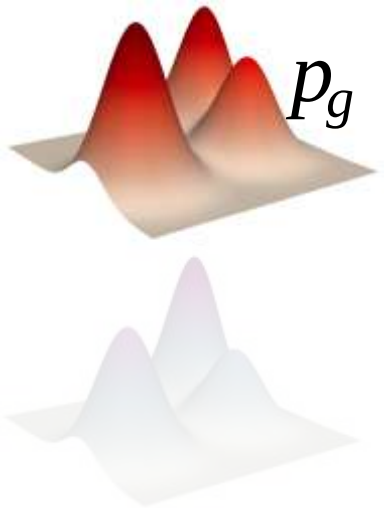
GAN

generation



p_z

generator



p_g

discriminator

or

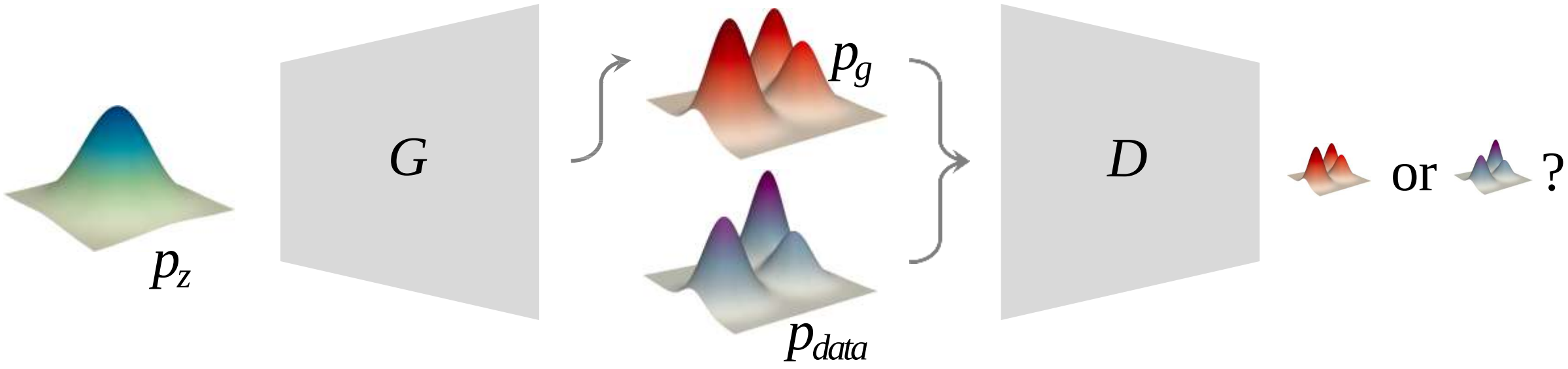
?

Adversarial Objective

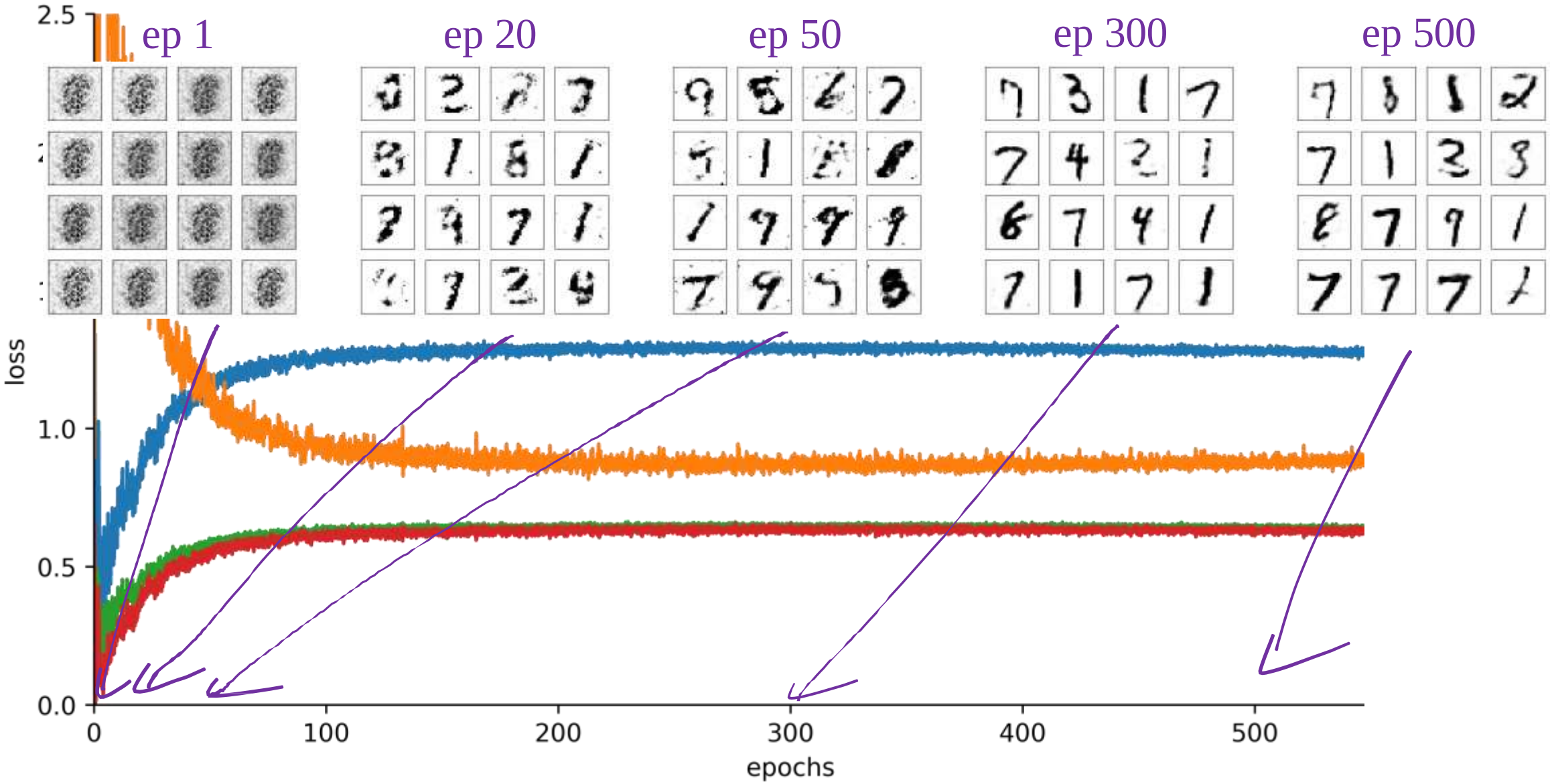
$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

min-max process

(vs. EM's max-max process)



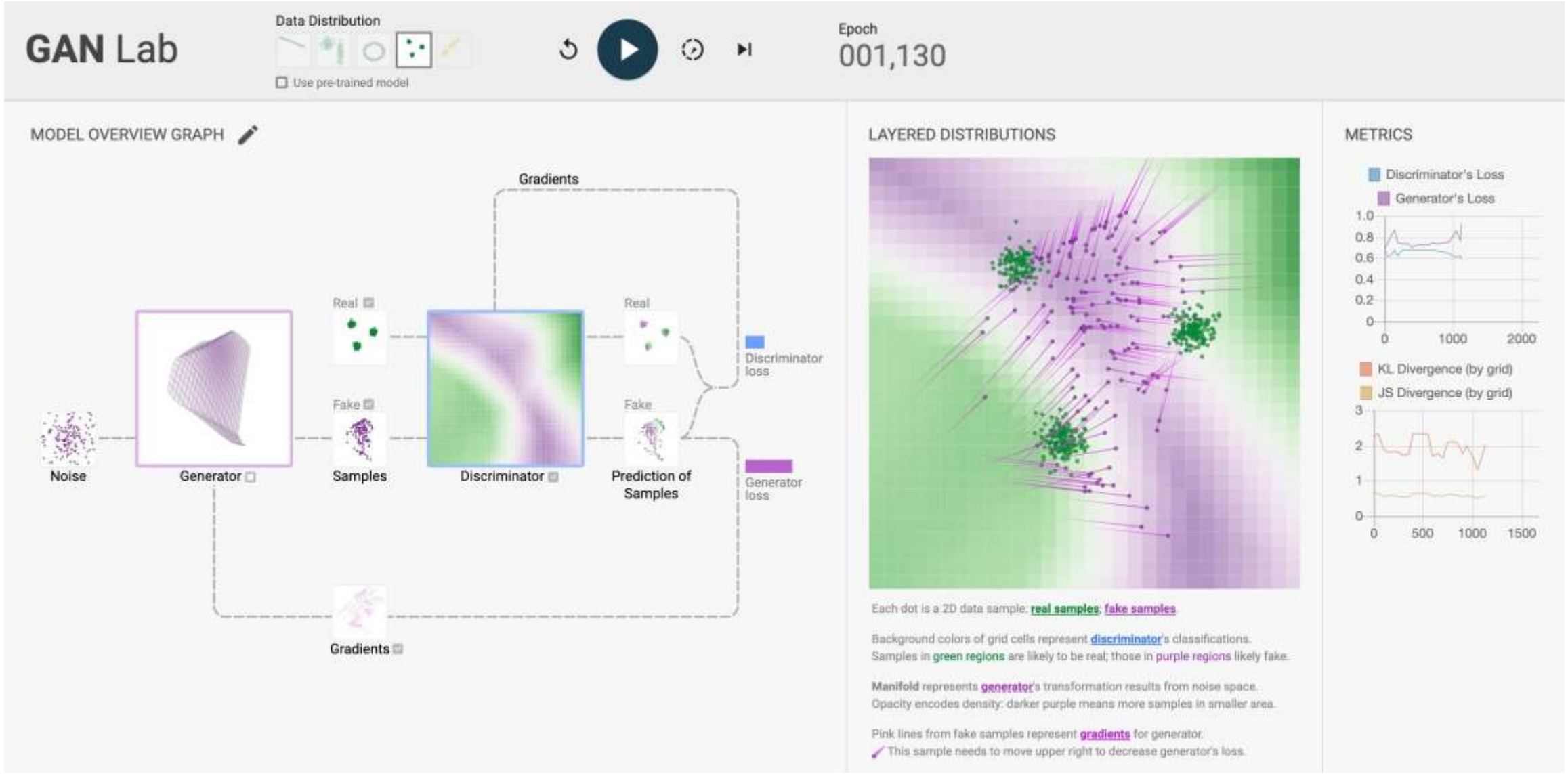
Running example: MNIST



*All objectives are negative of their original form

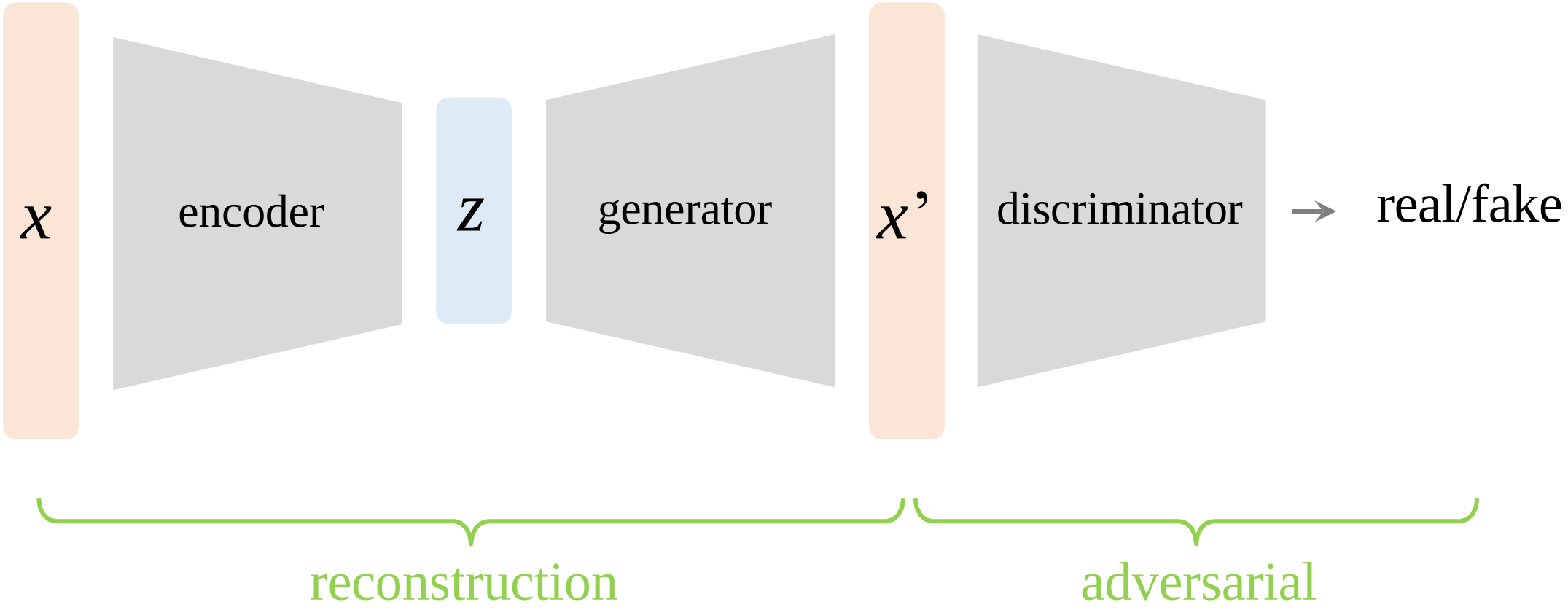
Code adapted from: https://github.com/prcastro/pytorch-gan/tree/master

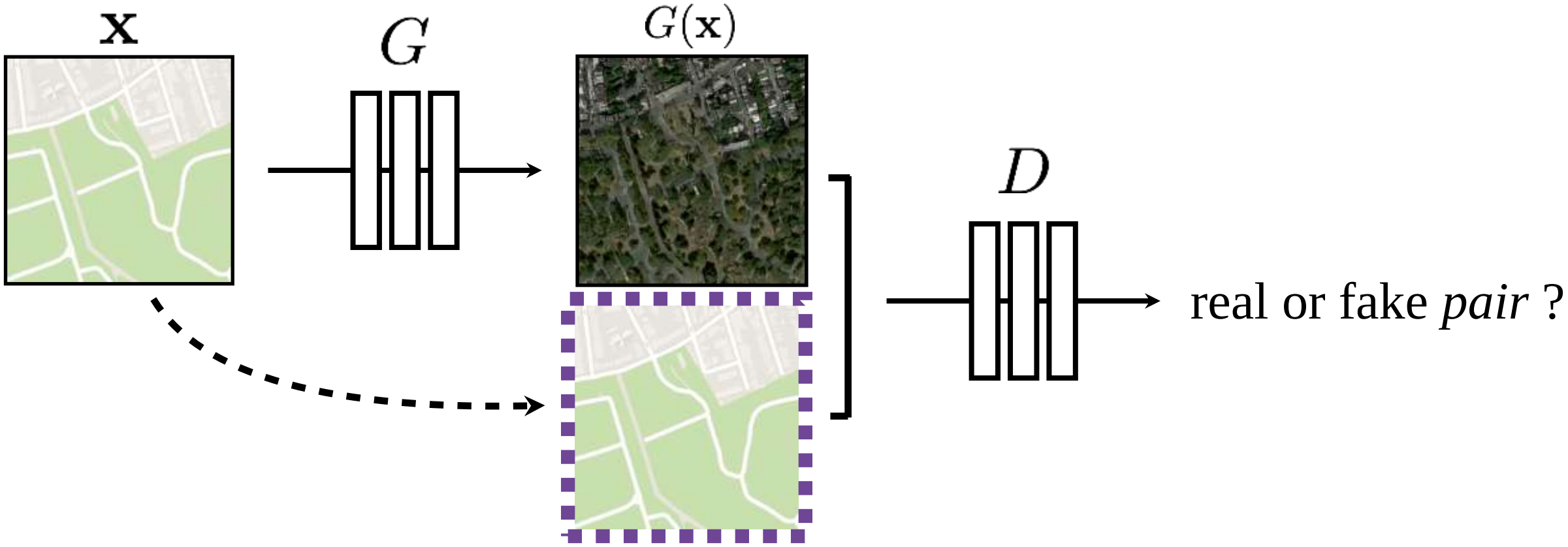
Running example: GAN Lab https://poloclub.github.io/ganlab/



Adversary as a Loss Function

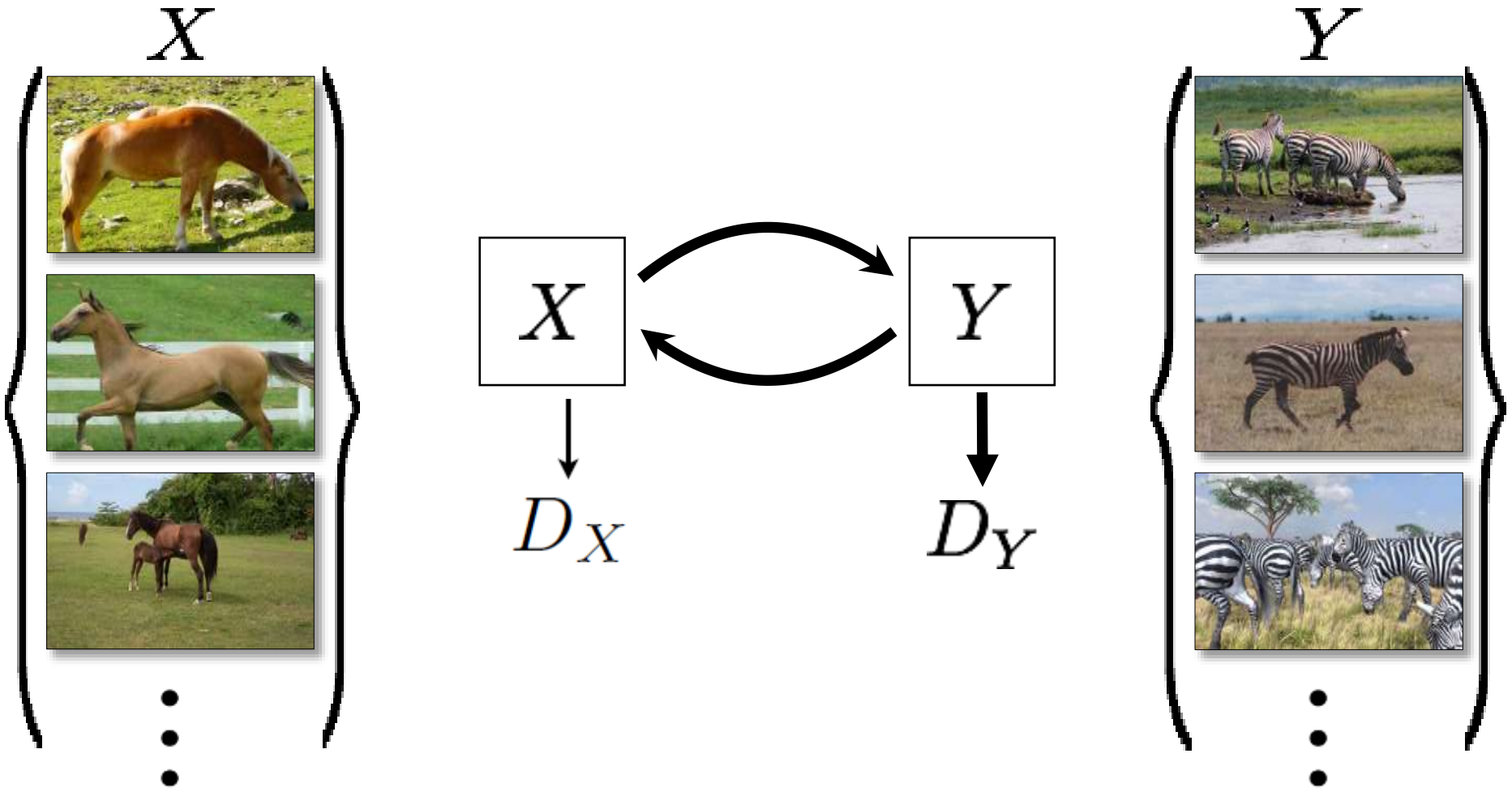
- Input can be from another source





$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y})) \;]$$

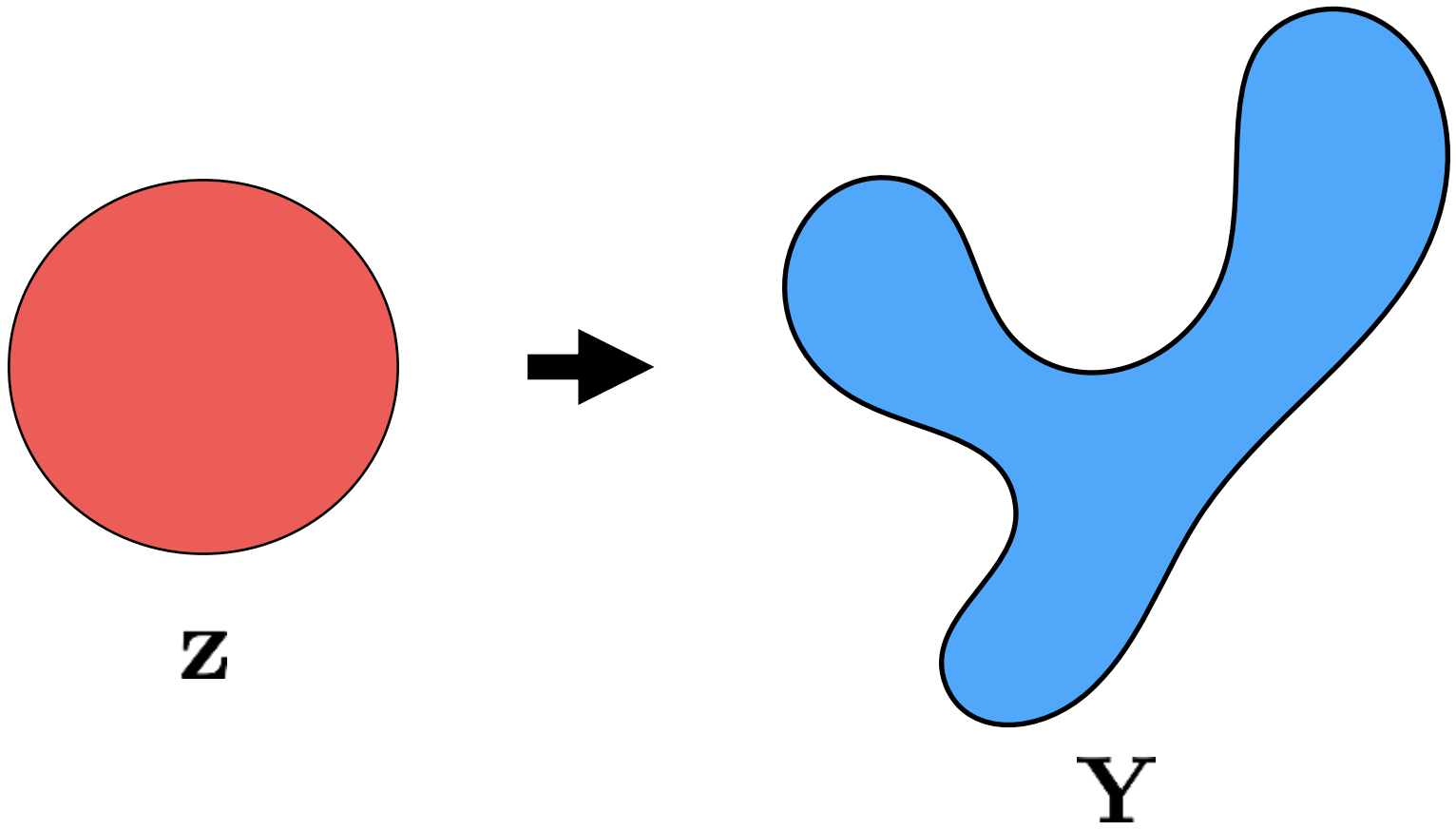
CycleGAN, or there and back aGAN



GANs

Gaussian

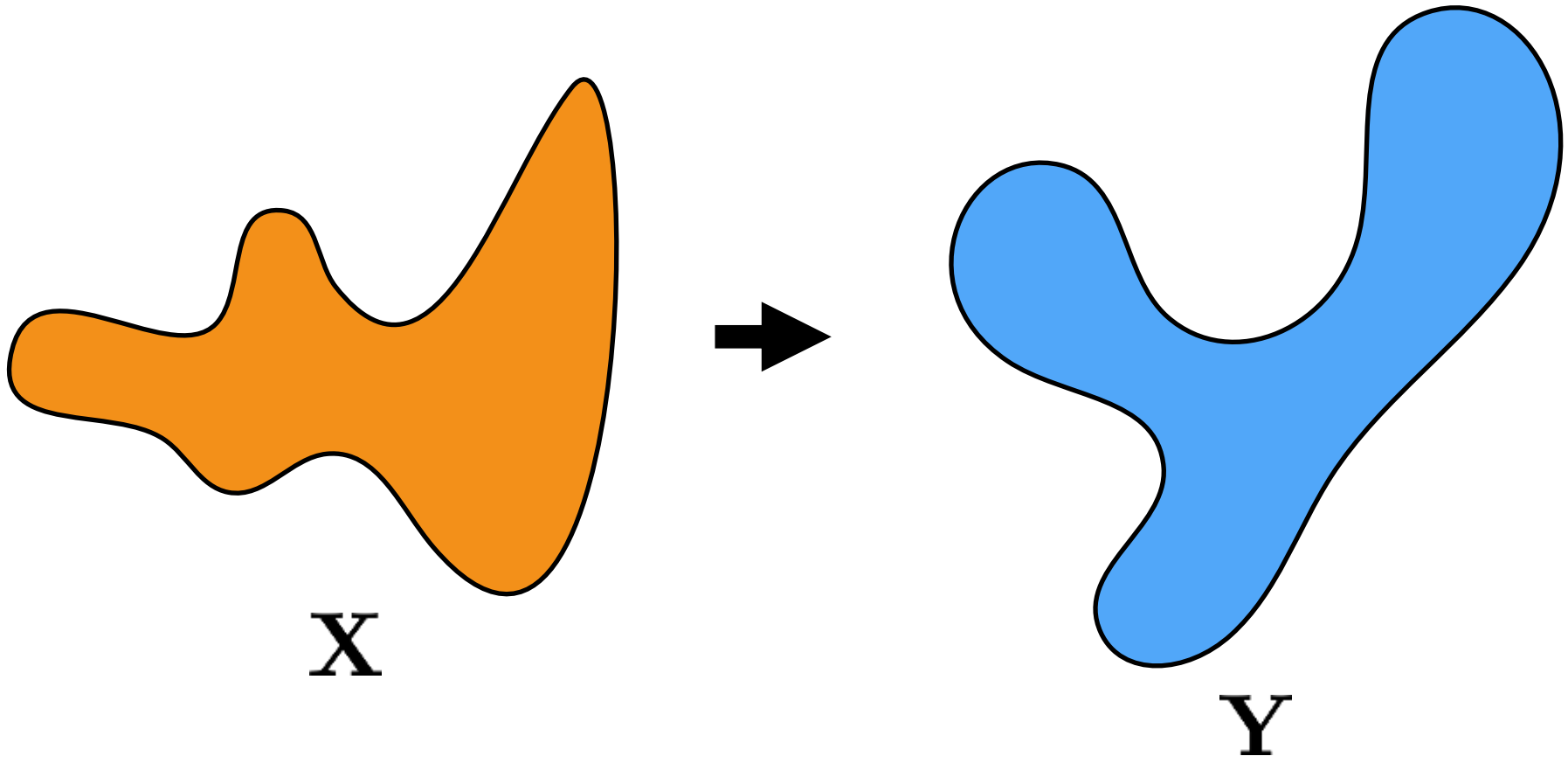
Target distribution



CycleGAN

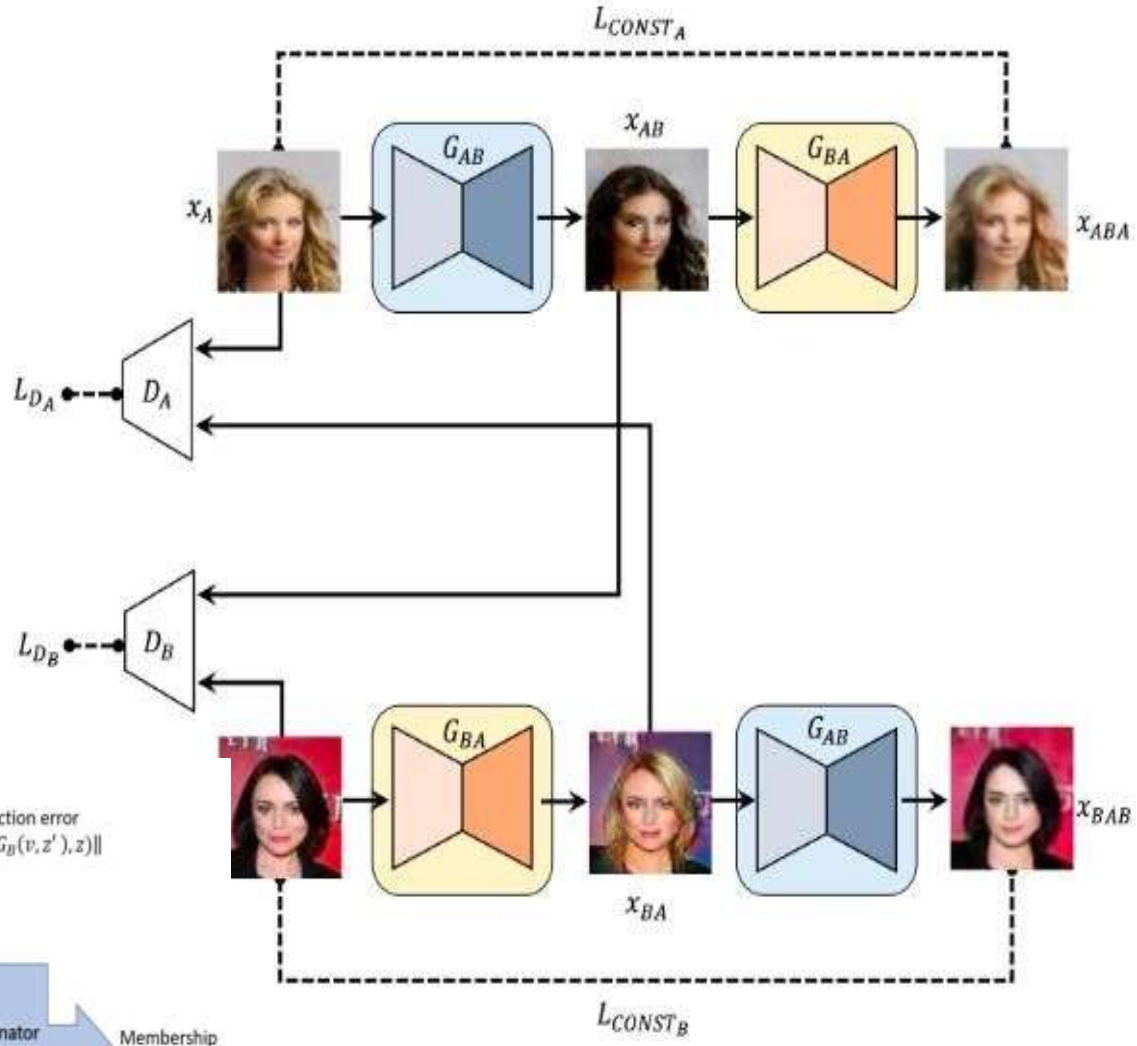
Horses

Zebras



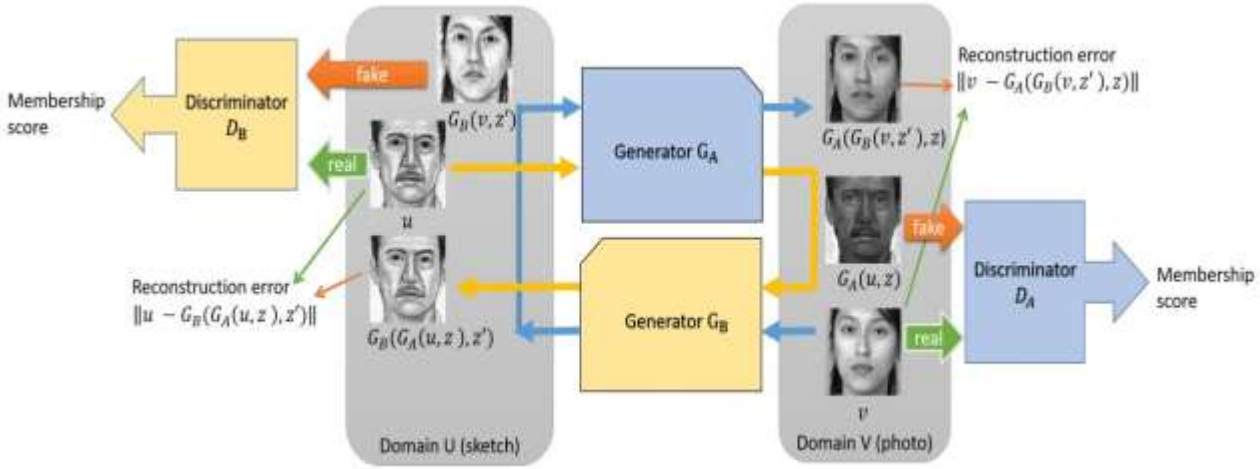
Disco GAN

https://arxiv.org/abs/1703.05192



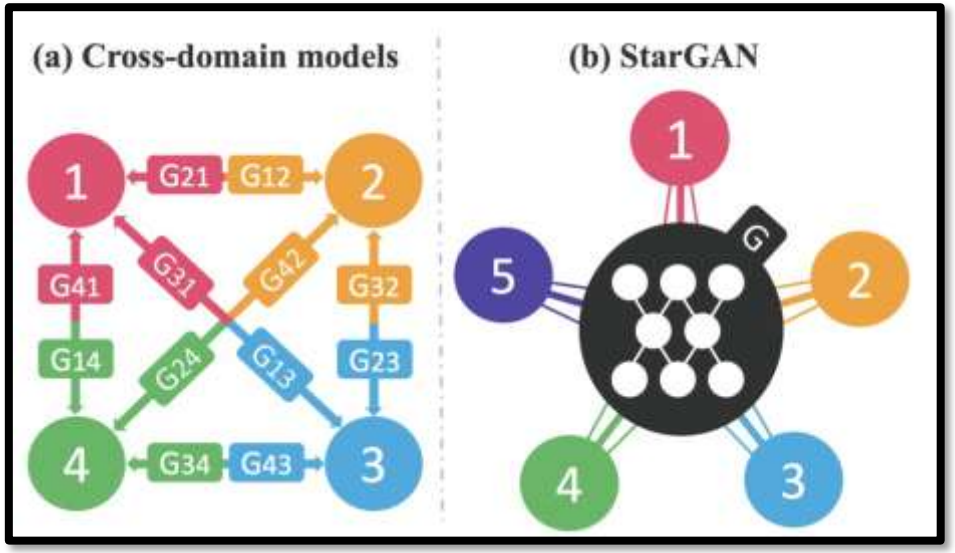
Dual GAN

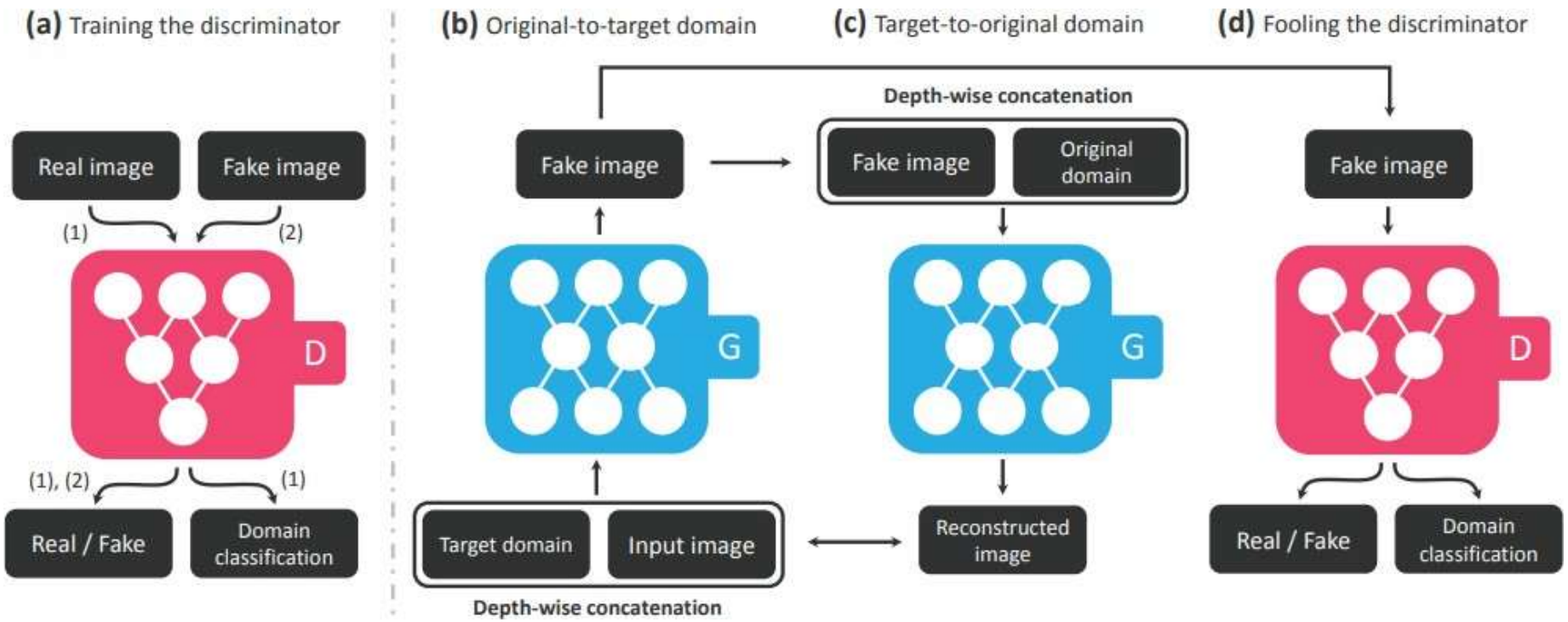
https://arxiv.org/abs/1704.02510



StarGAN

https://arxiv.org/abs/1711.09020

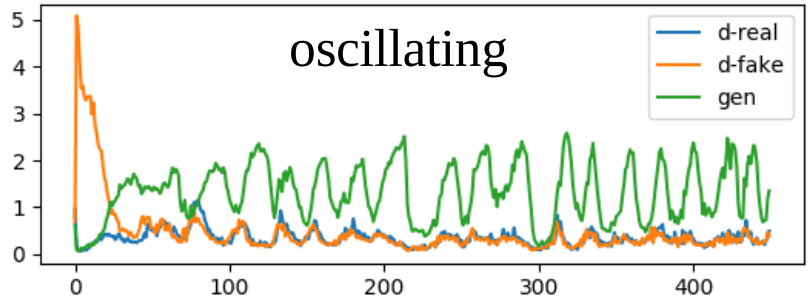




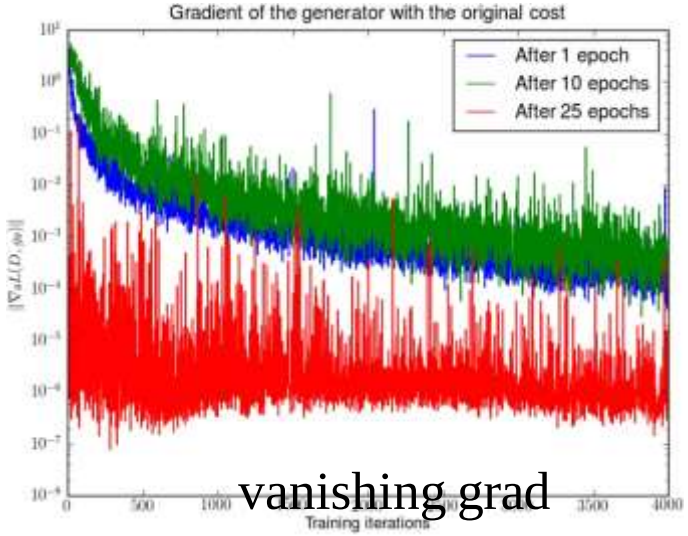
Problems of GAN

Difficult to train/converge

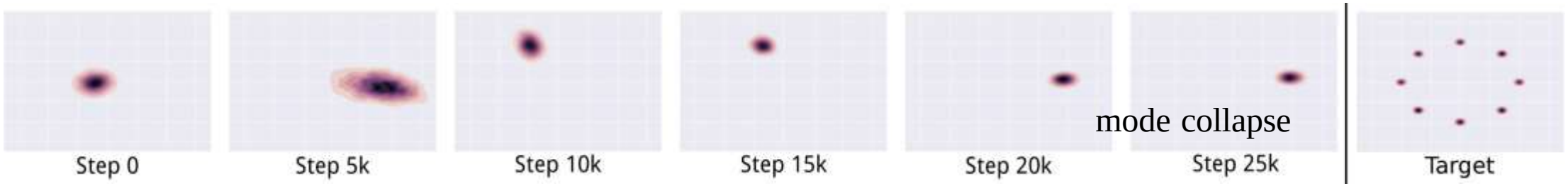
- Hard to achieve equilibrium
- Vanishing gradients
- Mode collapse



J. Brownlee, “How to Identify and Diagnose GAN Failure Modes”



Arjovsky & Bottou, “Towards Principled Methods for Training GANs”



L. Metz, “Unrolled Generative Adversarial Networks”

判别难度：如何有效地区分两个分布？

- 怎么判别人脸是真实的？
  - 这并不容易…
  - 判别器无法有效为生成器提供优化信息

真实！

生成！

快速判断

A

B

Which face is real?

快速判断

A

B

Which face is real?

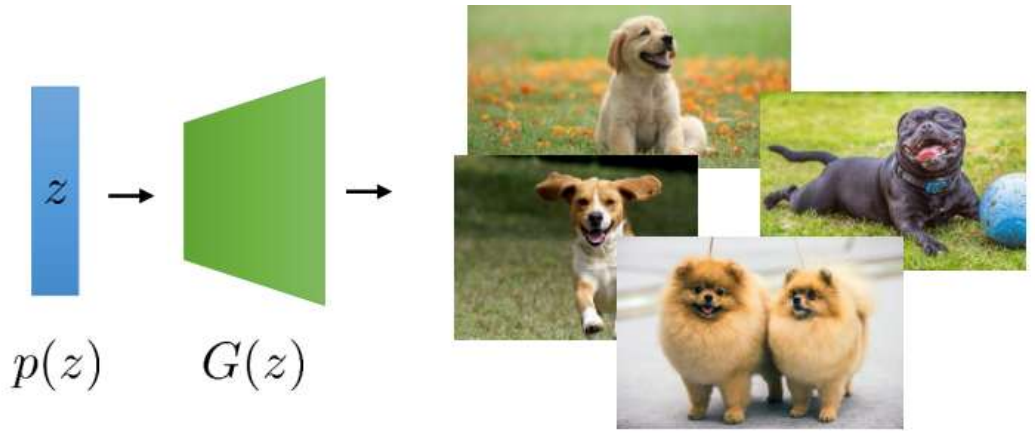
Mode Collapse

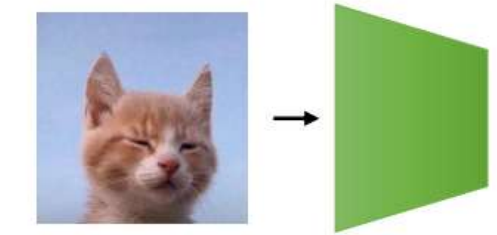
https://arxiv.org/pdf/1701.07875.pdf

模式崩溃（Mode Collapse）

- 对于生成器而言，不仅要生成真实图像，也要生成**全部真实图像**

虽然很真实，但是只生成狗没有猫！





→ True = 1.0!

损失为**0**，没有梯度回传给生成器！

无法指导生成猫！

→ True = 0.25!

损失较大，生成器对狗的生成倾向更严重

Wasserstein GAN

W-GAN in Short

For mathematicians:

- Wasserstein distance, instead of JS divergence

For engineers:

- remove logarithms
- clip weights

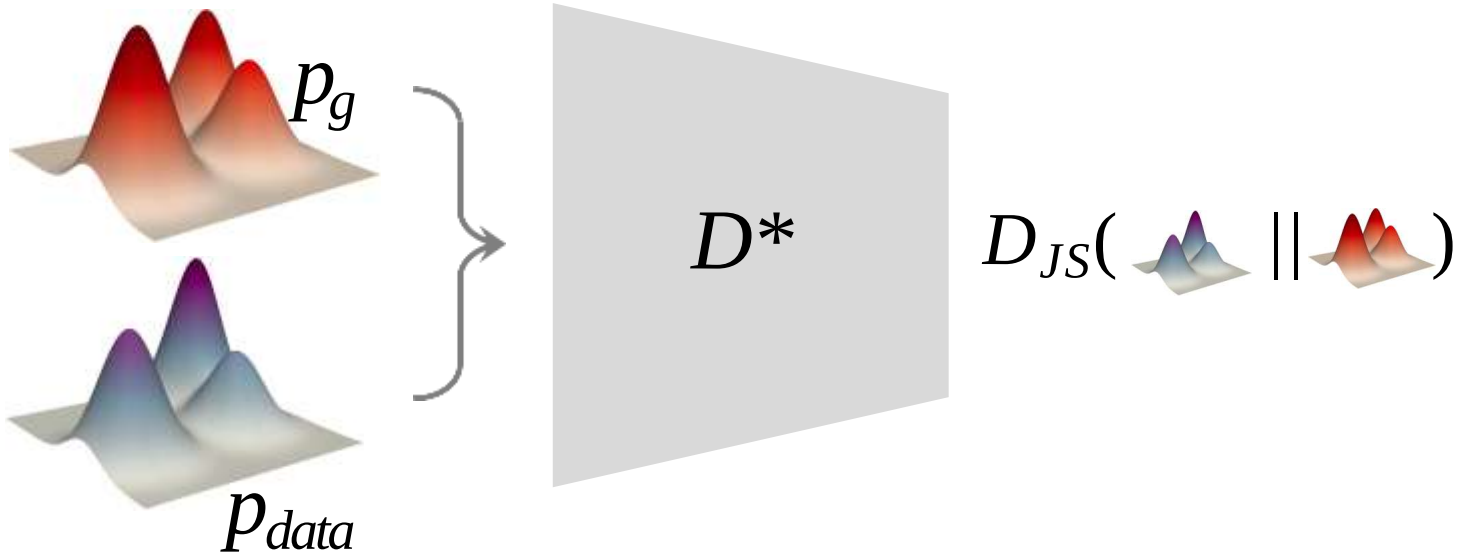
For laymen:

- art critic, instead of forgery expert

Summarized by Reddit user danielvarga: https://www.reddit.com/r/MachineLearning/comments/5qxoaz/comment/dd7aomb/

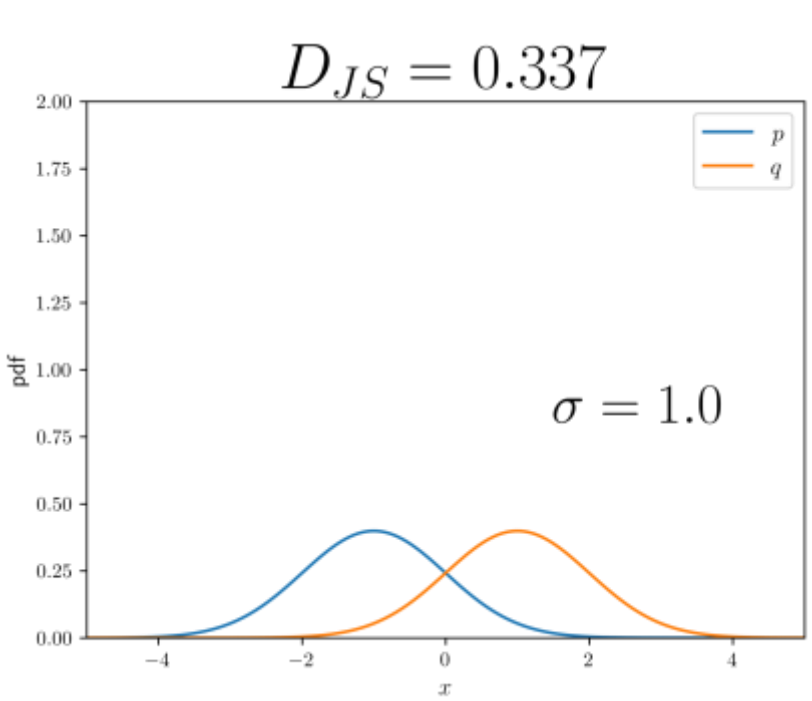
Recap: GAN optimizes for D_{JS}

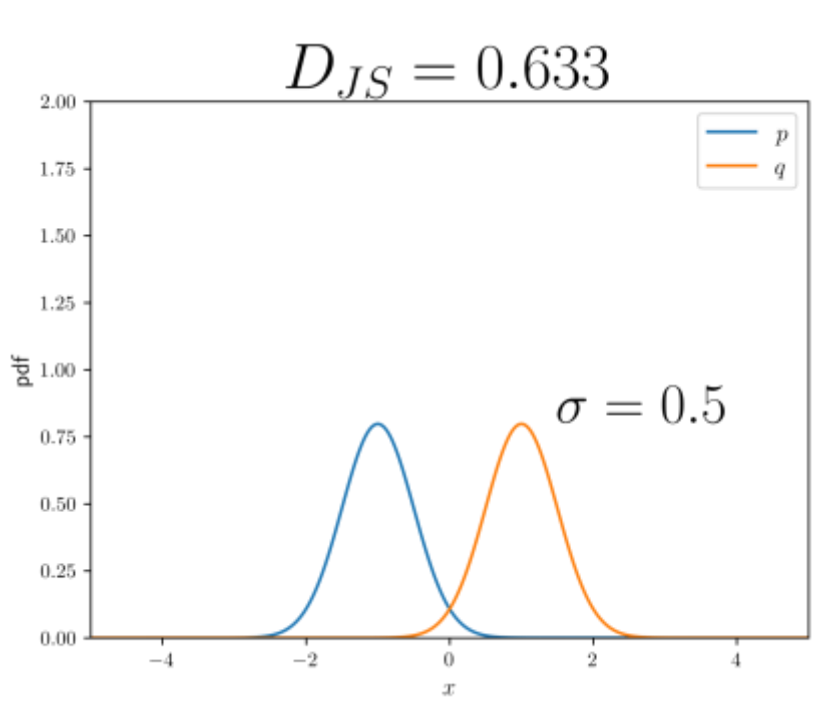
$$\mathcal{L}(D^*, G) = 2D_{JS}(p_{\text{data}} \| p_g) - 2\log 2$$



Problems of D_{JS}

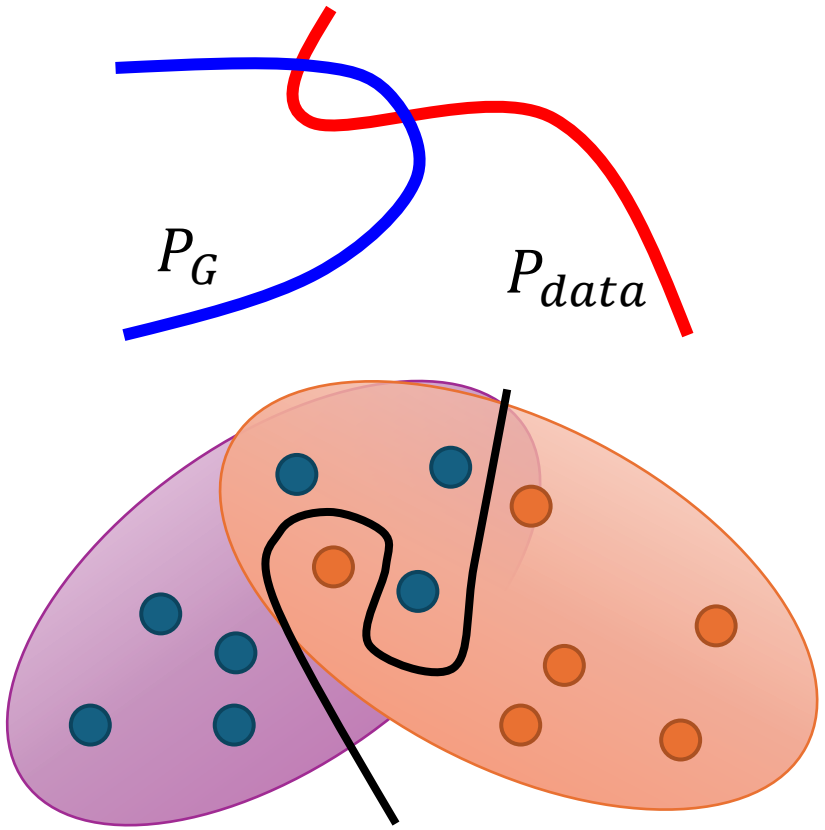
If p and q don't overlap, D_{JS} is a constant (log2), i.e., no gradient





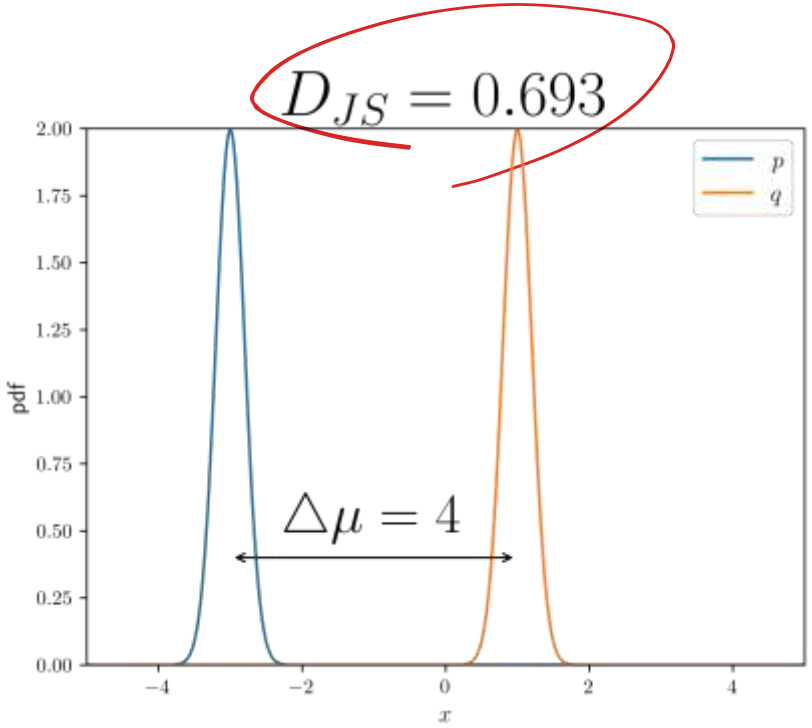
Problems of D_{JS}

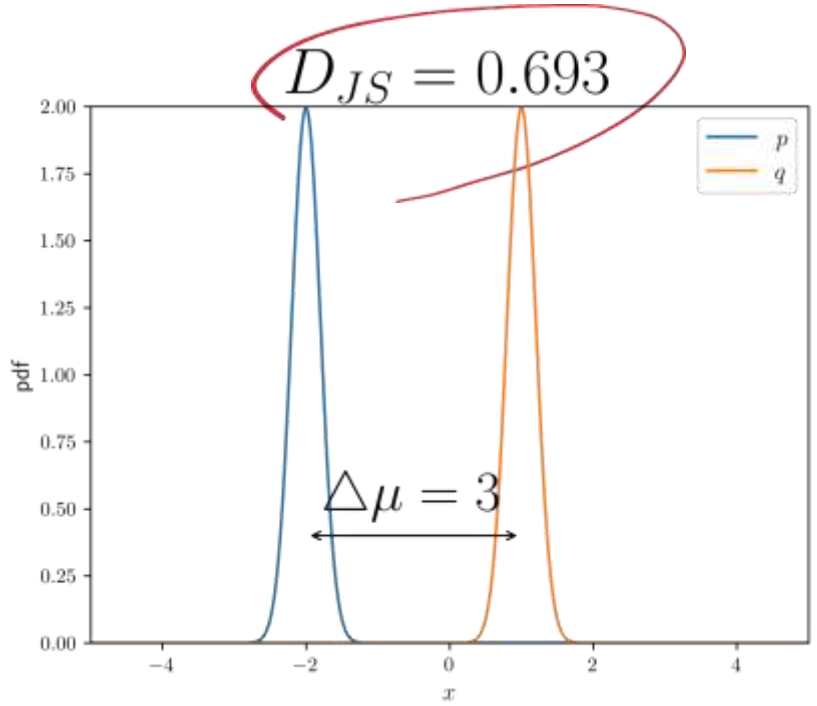
Why it is common p and q don't overlap?

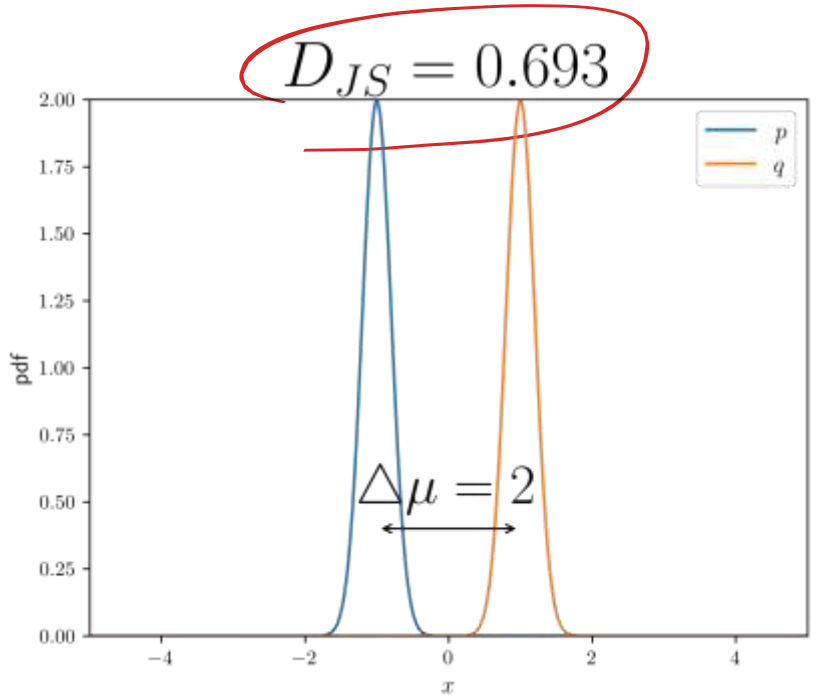


Problems of D_{JS}

If p and q don't overlap, D_{JS} is a constant (log2), i.e., no gradient

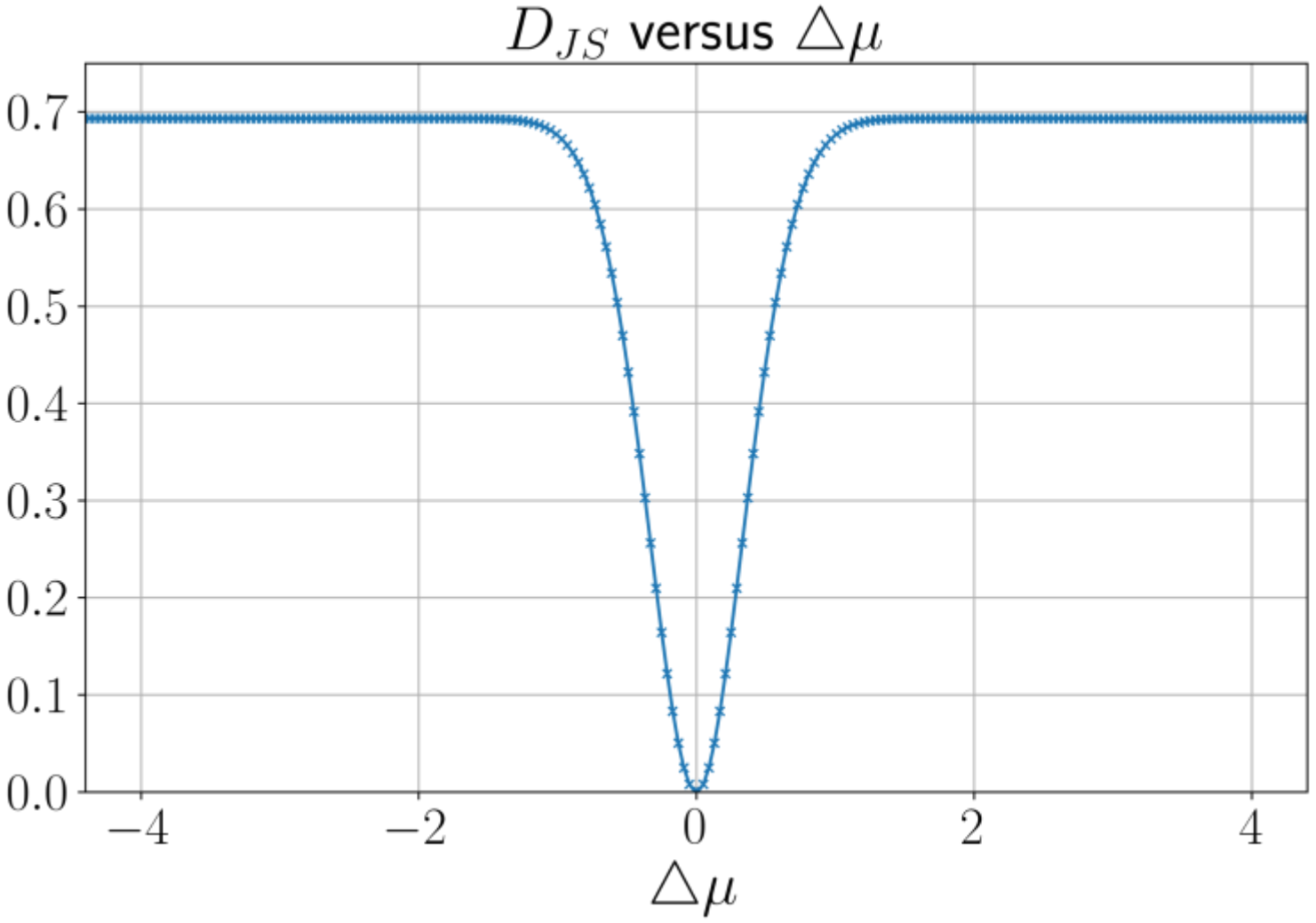


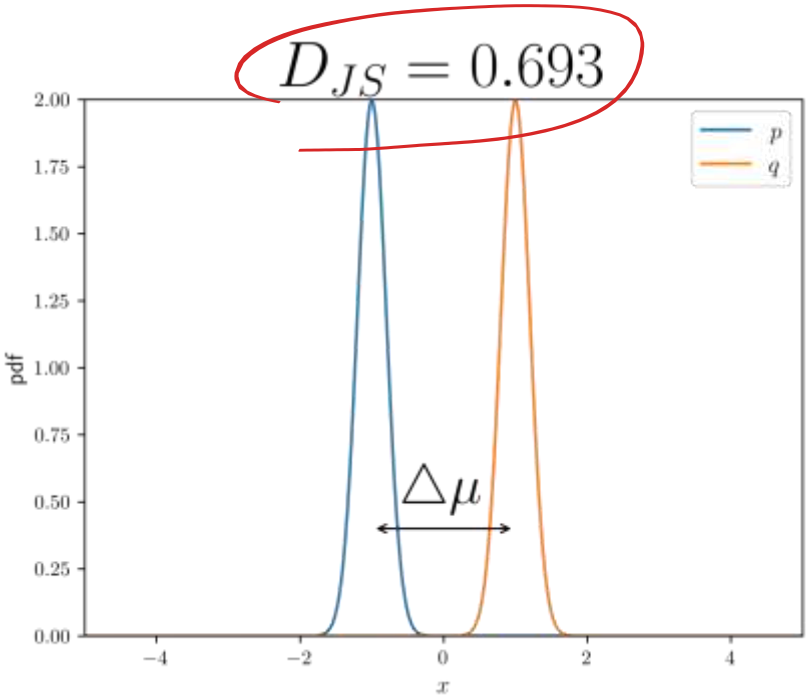




Problems of D_{JS}

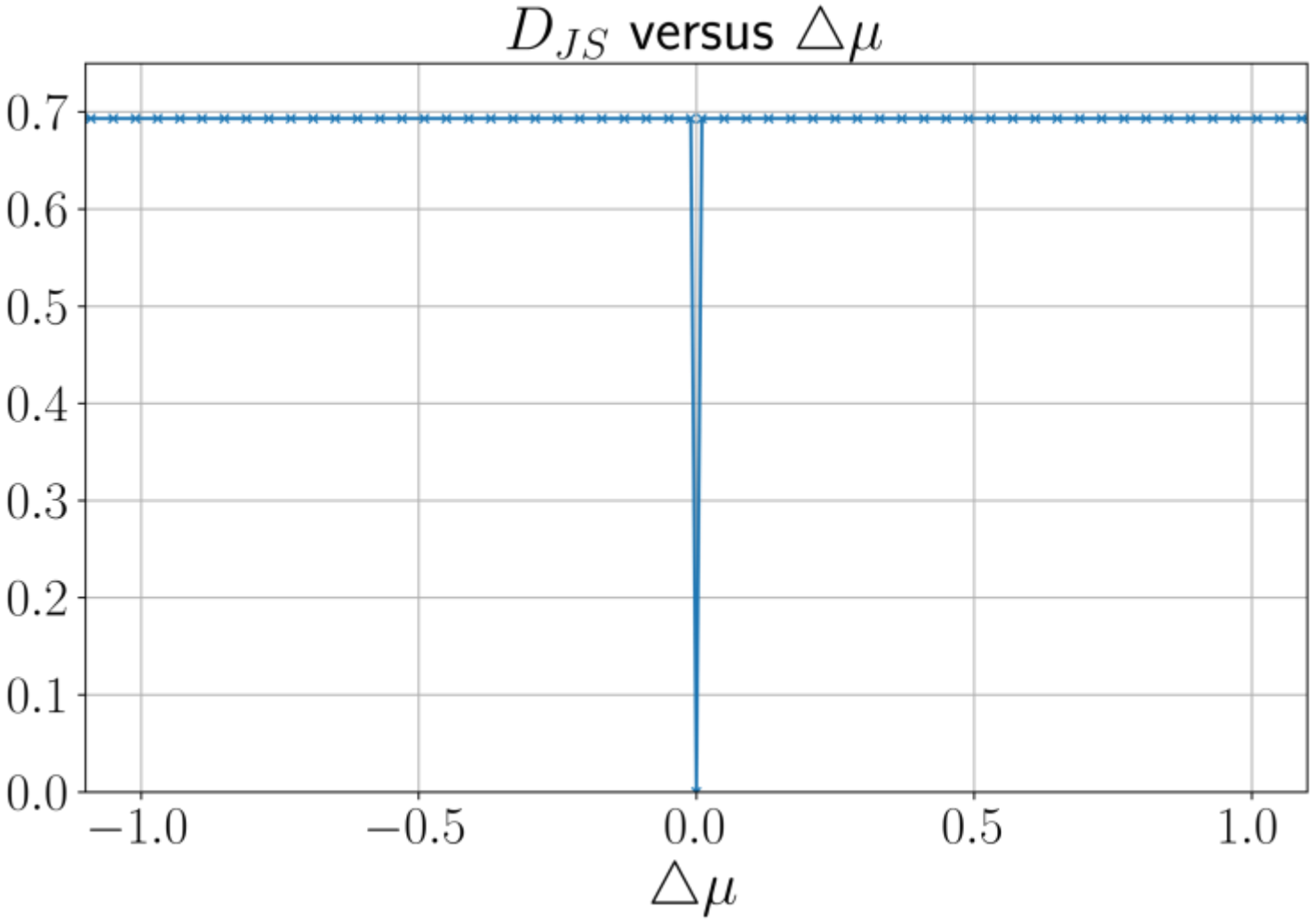
- D_{JS} is useful only if p and q are close

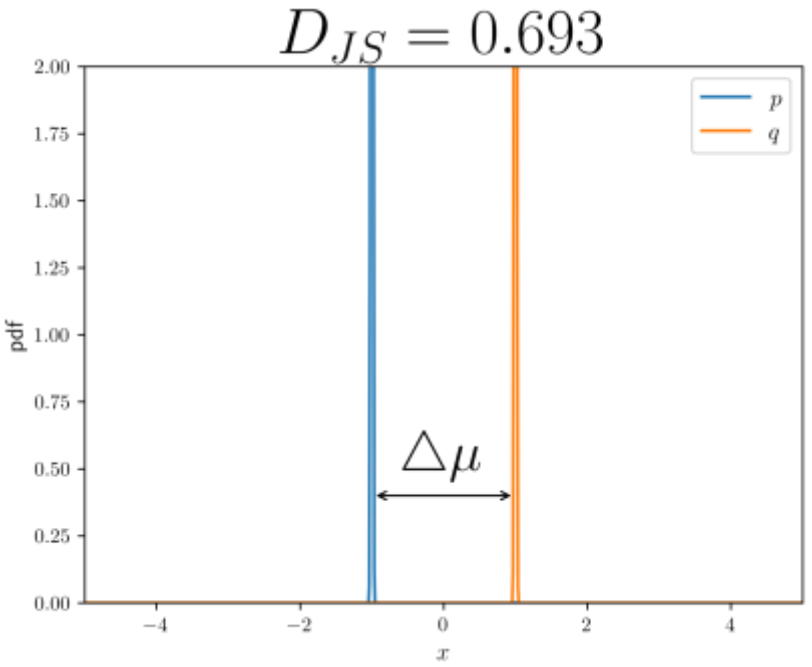




Problems of D_{JS}

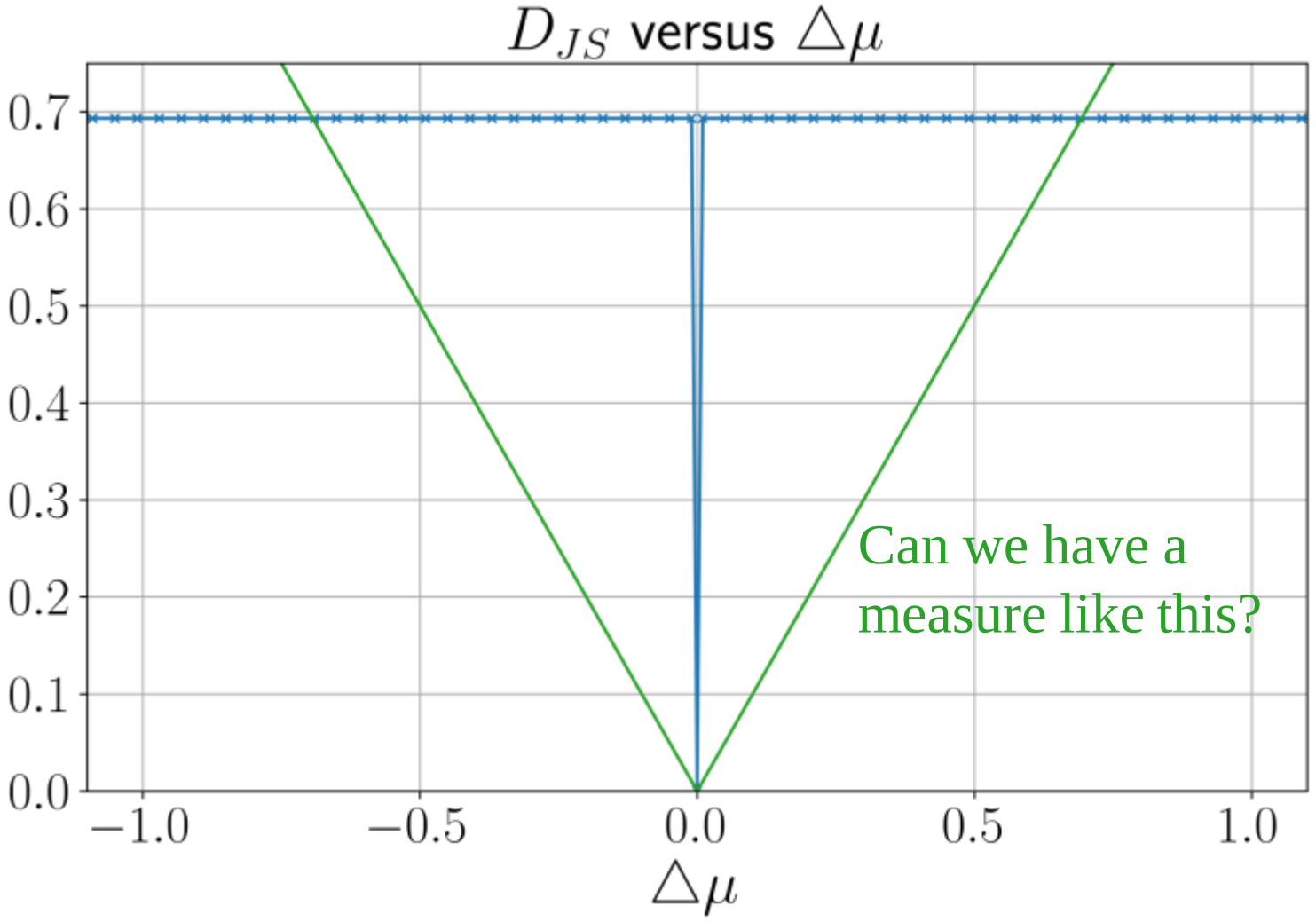
- D_{JS} is a delta function when p and q are delta functions

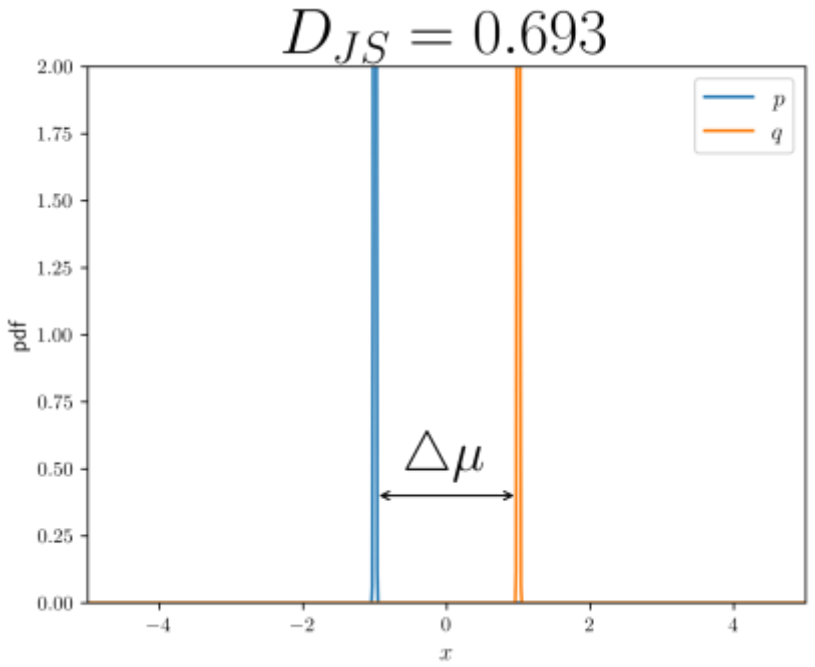




Problems of D_{JS}

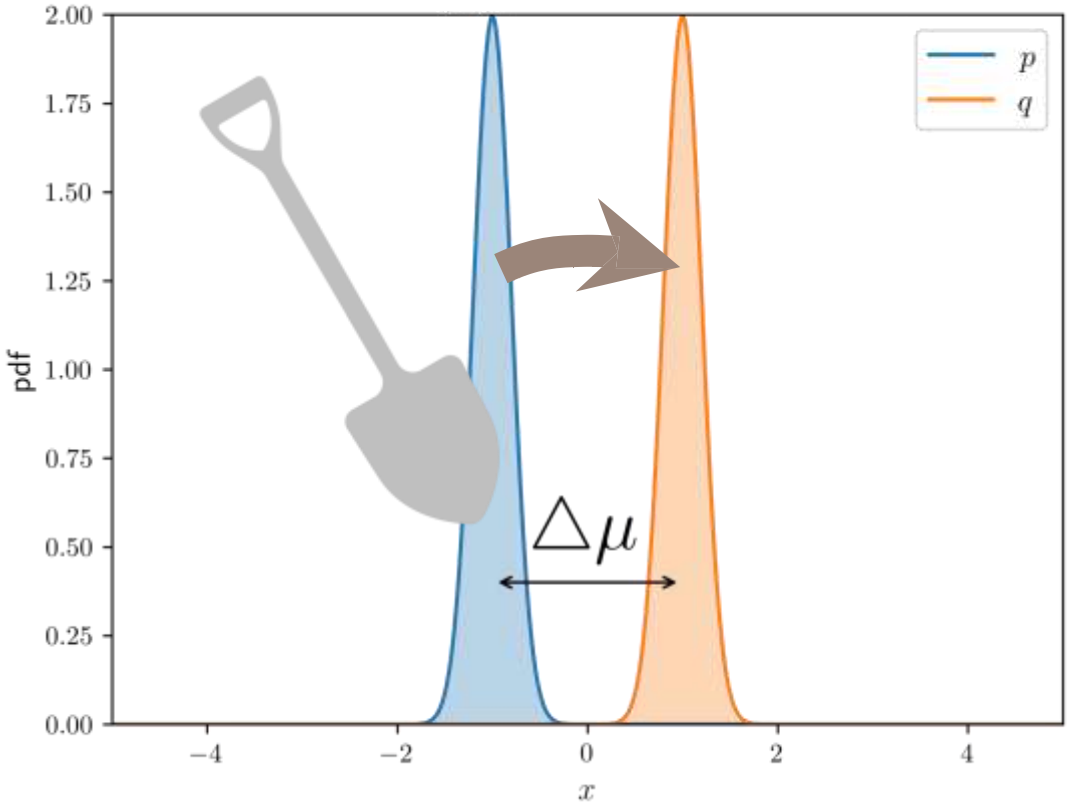
- D_{JS} is a delta function when p and q are delta functions

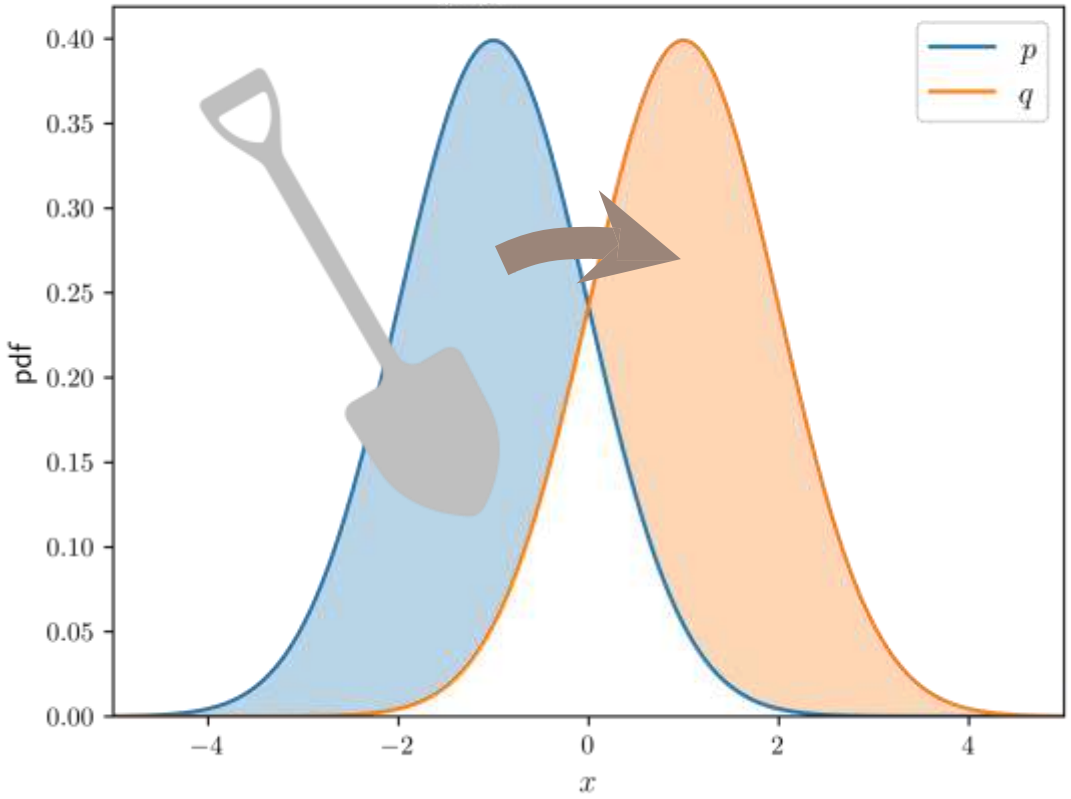




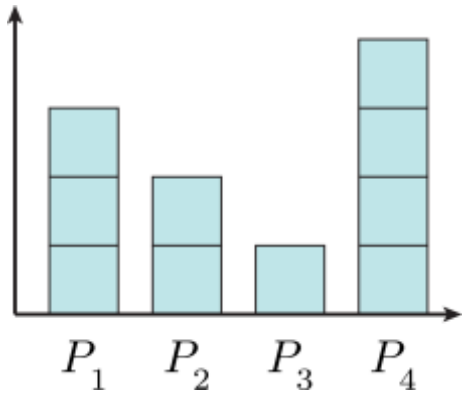
Wasserstein Distance

"Earth Mover's Distance"





Running example: Wasserstein Distance



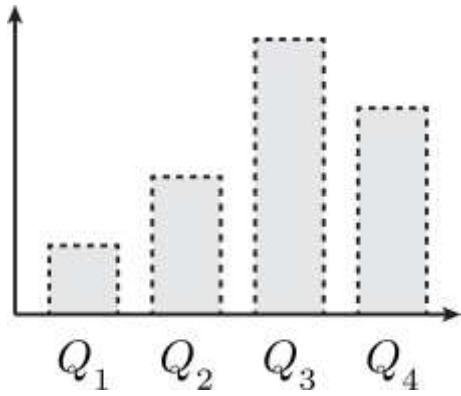
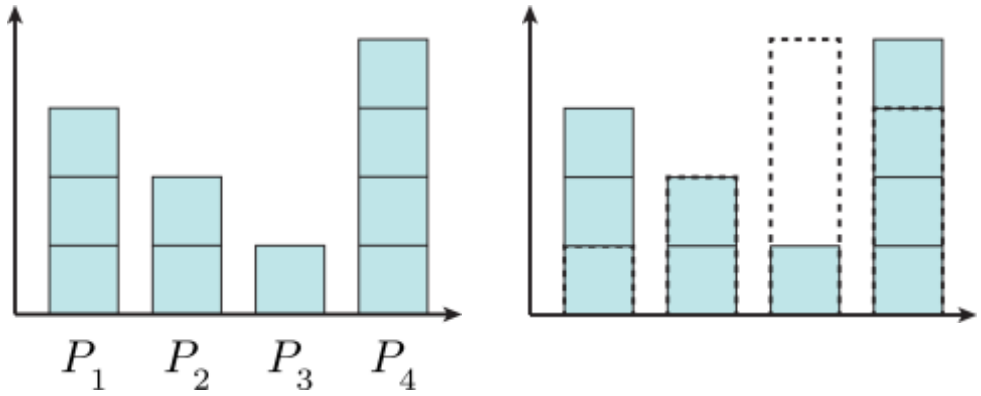


Figure inspired by: Lilian Weng, “From GAN to WGAN”, arXiv:1904.08994

Running example: Wasserstein Distance



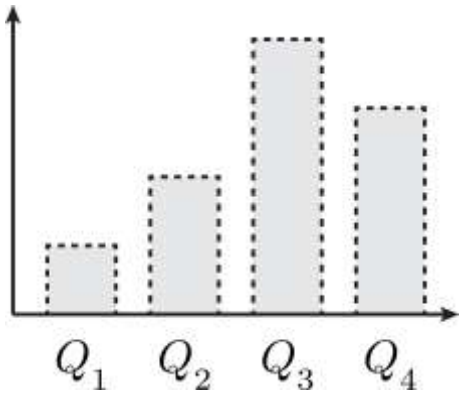
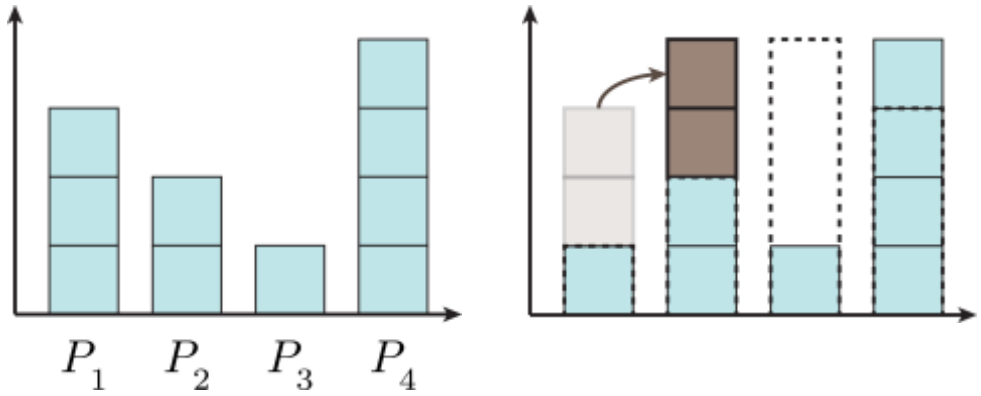


Figure inspired by: Lilian Weng, “From GAN to WGAN”, arXiv:1904.08994

Running example: Wasserstein Distance

2 shovelfuls



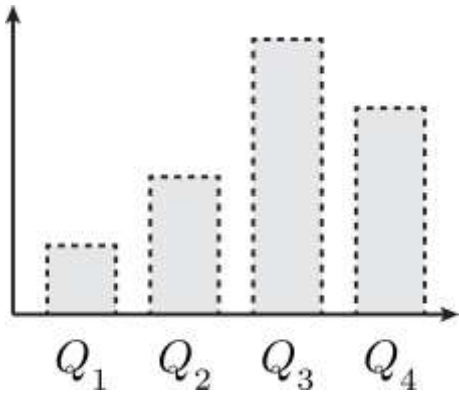


Figure inspired by: Lilian Weng, “From GAN to WGAN”, arXiv:1904.08994

Running example: Wasserstein Distance

2 shovelfuls

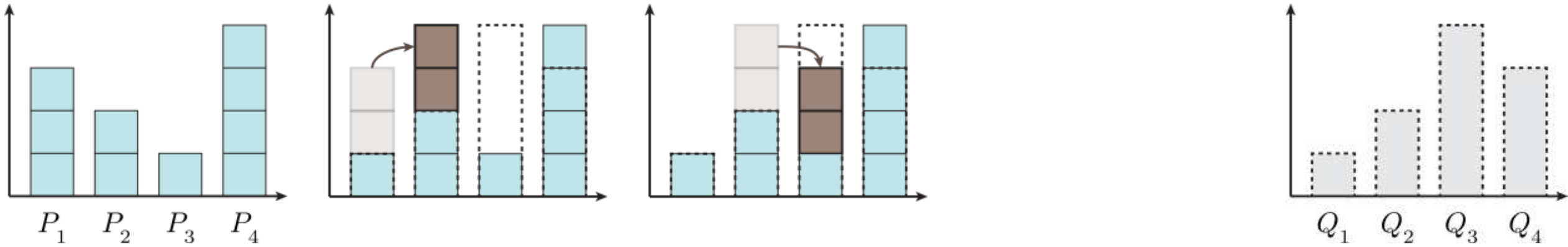


Figure inspired by: Lilian Weng, “From GAN to WGAN”, arXiv:1904.08994

Running example: Wasserstein Distance

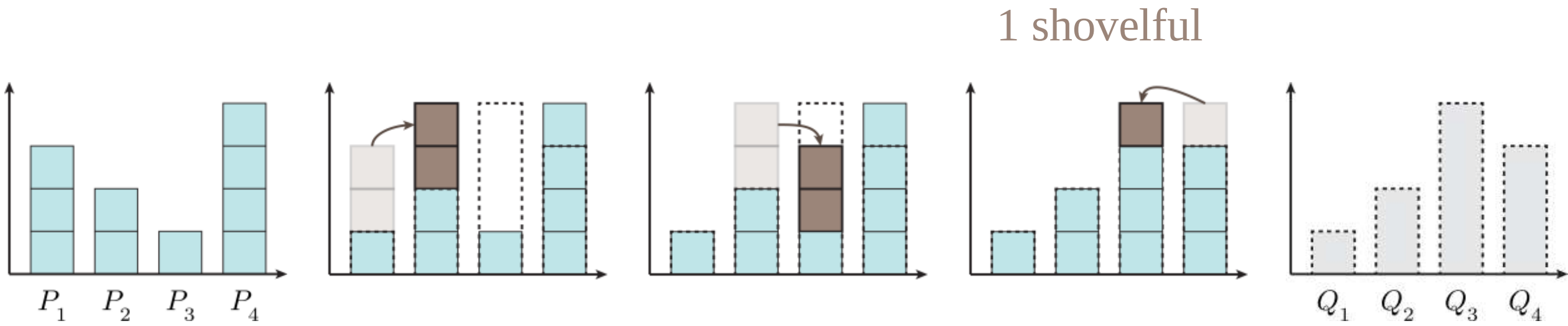


Figure inspired by: Lilian Weng, “From GAN to WGAN”, arXiv:1904.08994

Running example: Wasserstein Distance

$$W(P, Q) = 5 \times \blacksquare$$

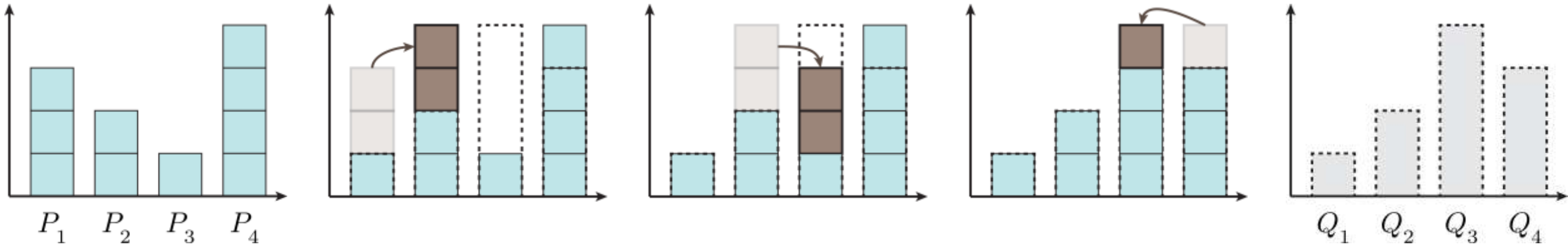
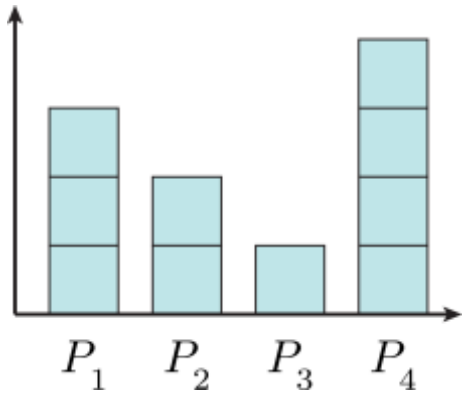


Figure inspired by: Lilian Weng, “From GAN to WGAN”, arXiv:1904.08994

Running example: Wasserstein Distance



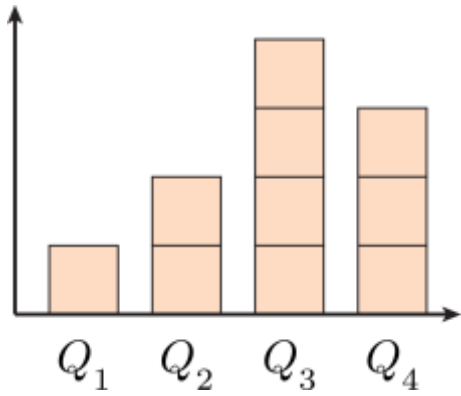
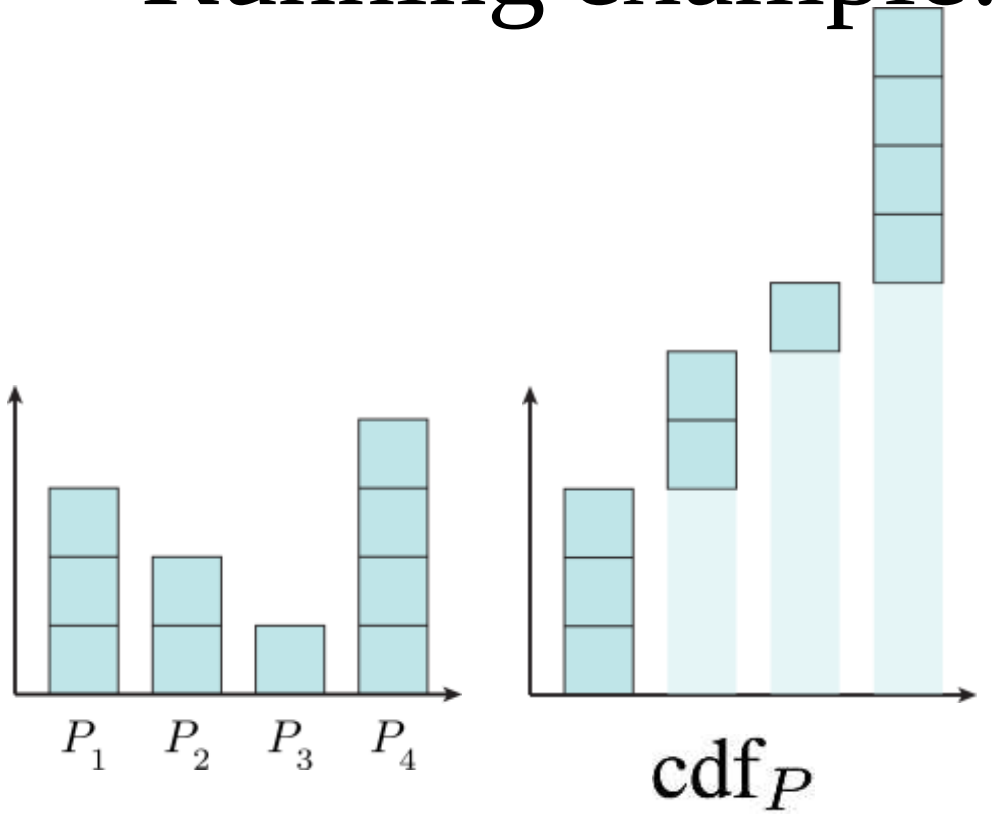
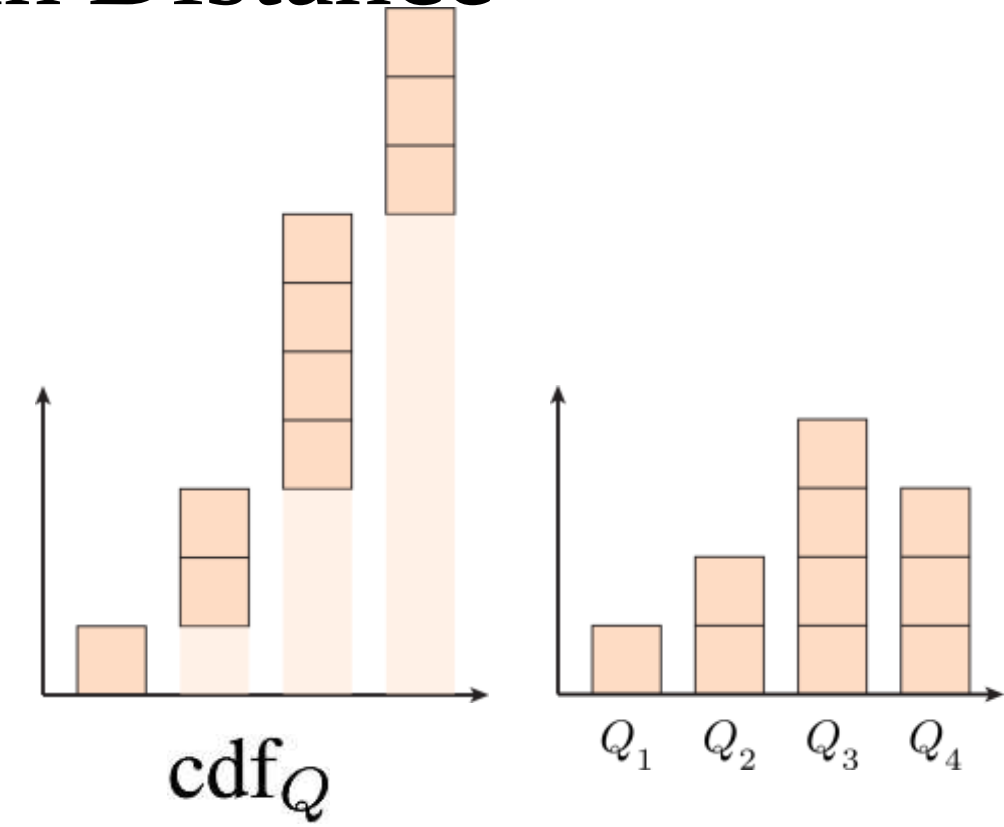


Figure inspired by: Lilian Weng, “From GAN to WGAN”, arXiv:1904.08994

Running example: Wasserstein Distance

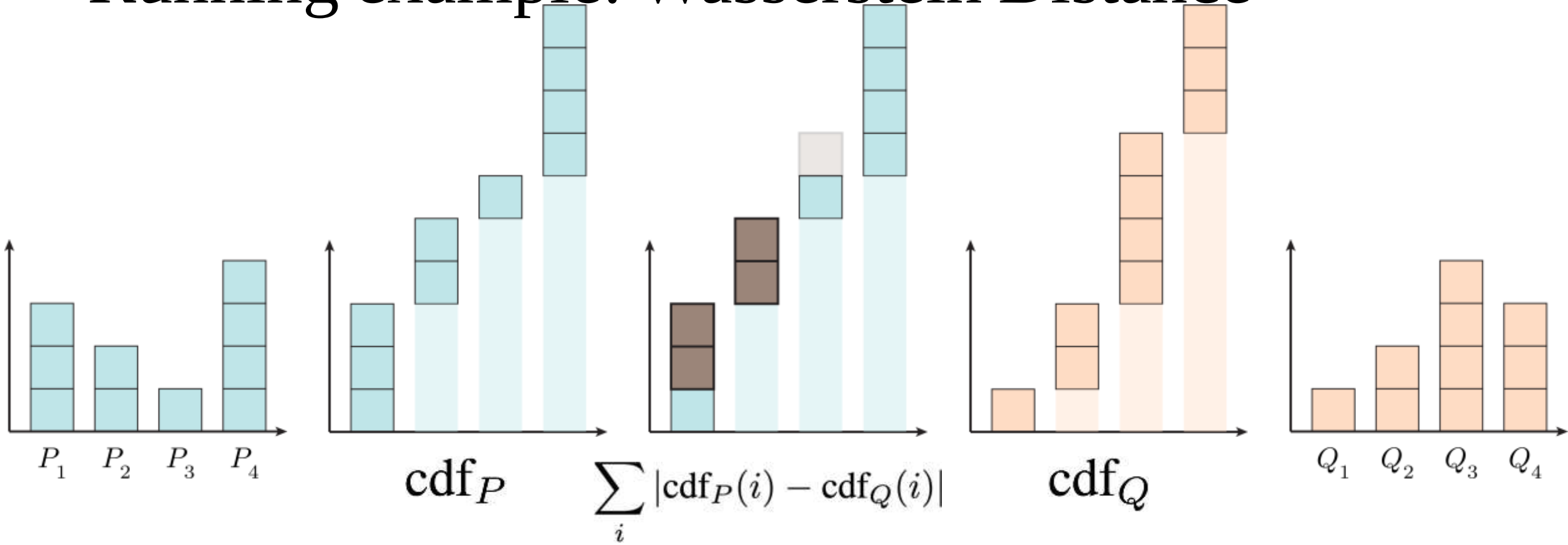




- cdf: cumulative distribution function

Figure inspired by: Lilian Weng, “From GAN to WGAN”, arXiv:1904.08994

Running example: Wasserstein Distance



$$W(P, Q) = 5 \times \blacksquare$$

Figure inspired by: Lilian Weng, “From GAN to WGAN”, arXiv:1904.08994

Wasserstein Distance

- 1-Wasserstein Distance (1-d, discrete)

l_1-norm

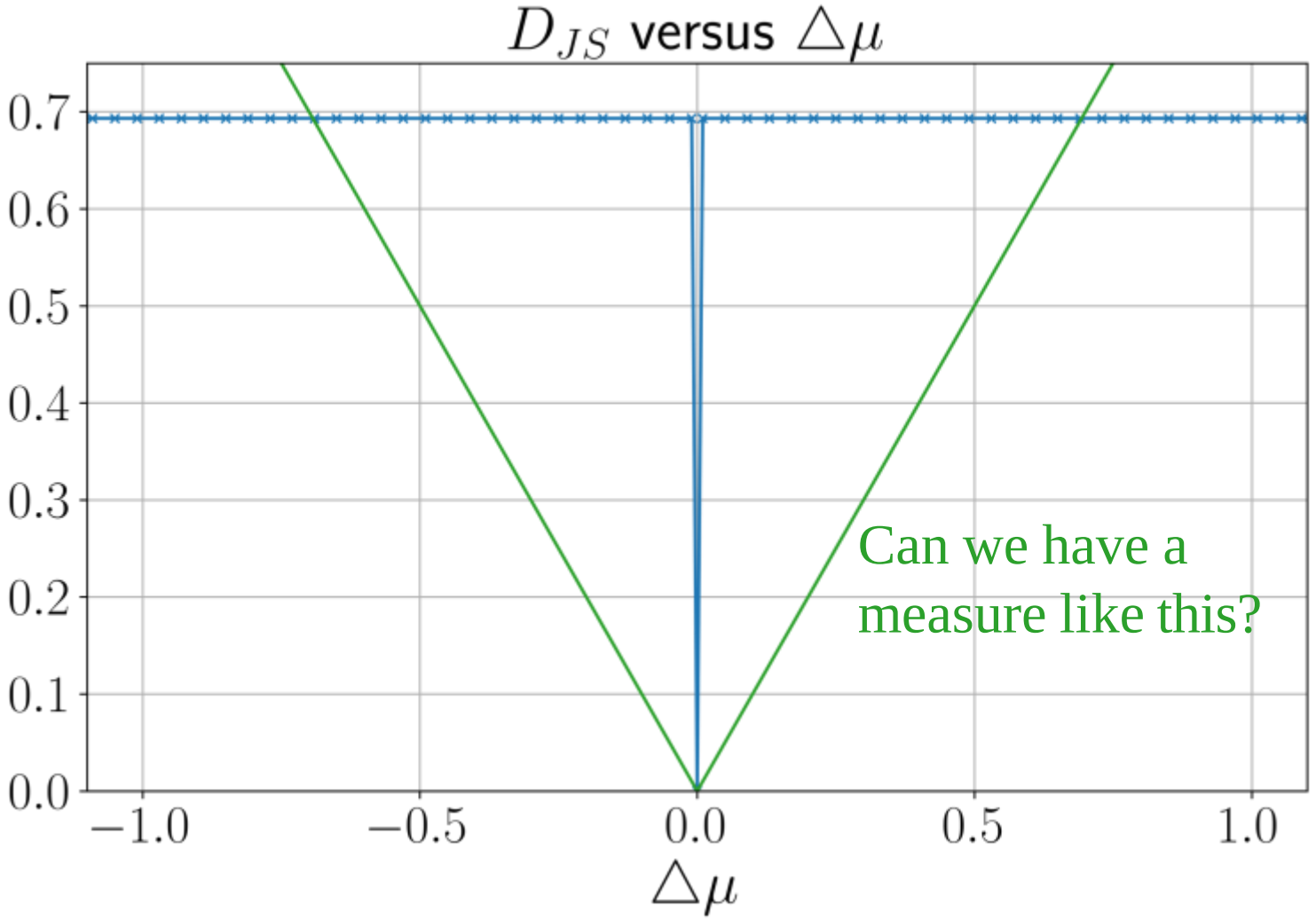
$$W_1(P, Q) = \sum_i |\text{cdf}_P(i) - \text{cdf}_Q(i)|$$

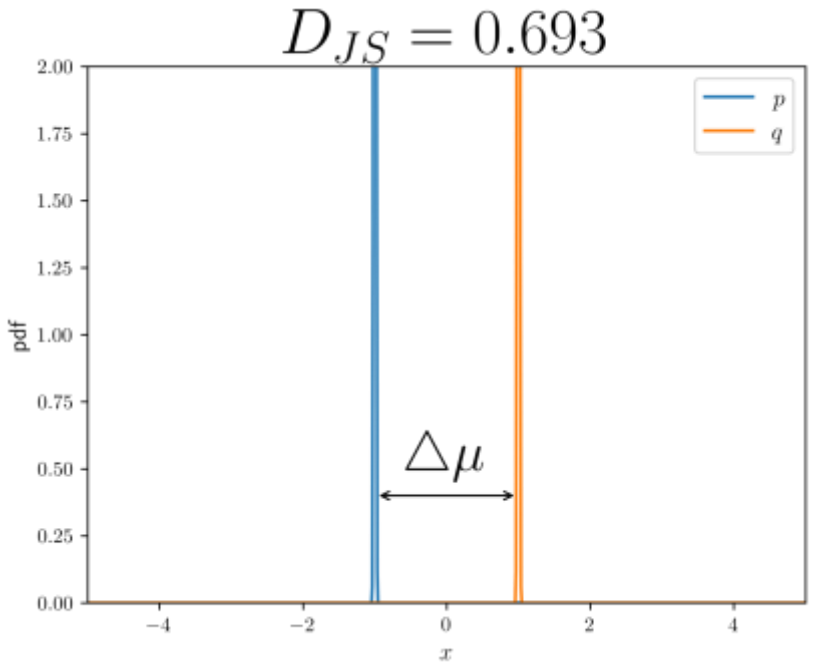
- 1-Wasserstein Distance (1-d, continuous)

$$W_1(p, q) = \int_x |\text{cdf}_p(x) - \text{cdf}_q(x)| dx$$

Recap: D_{JS}

- D_{JS} is a delta function when p and q are delta functions

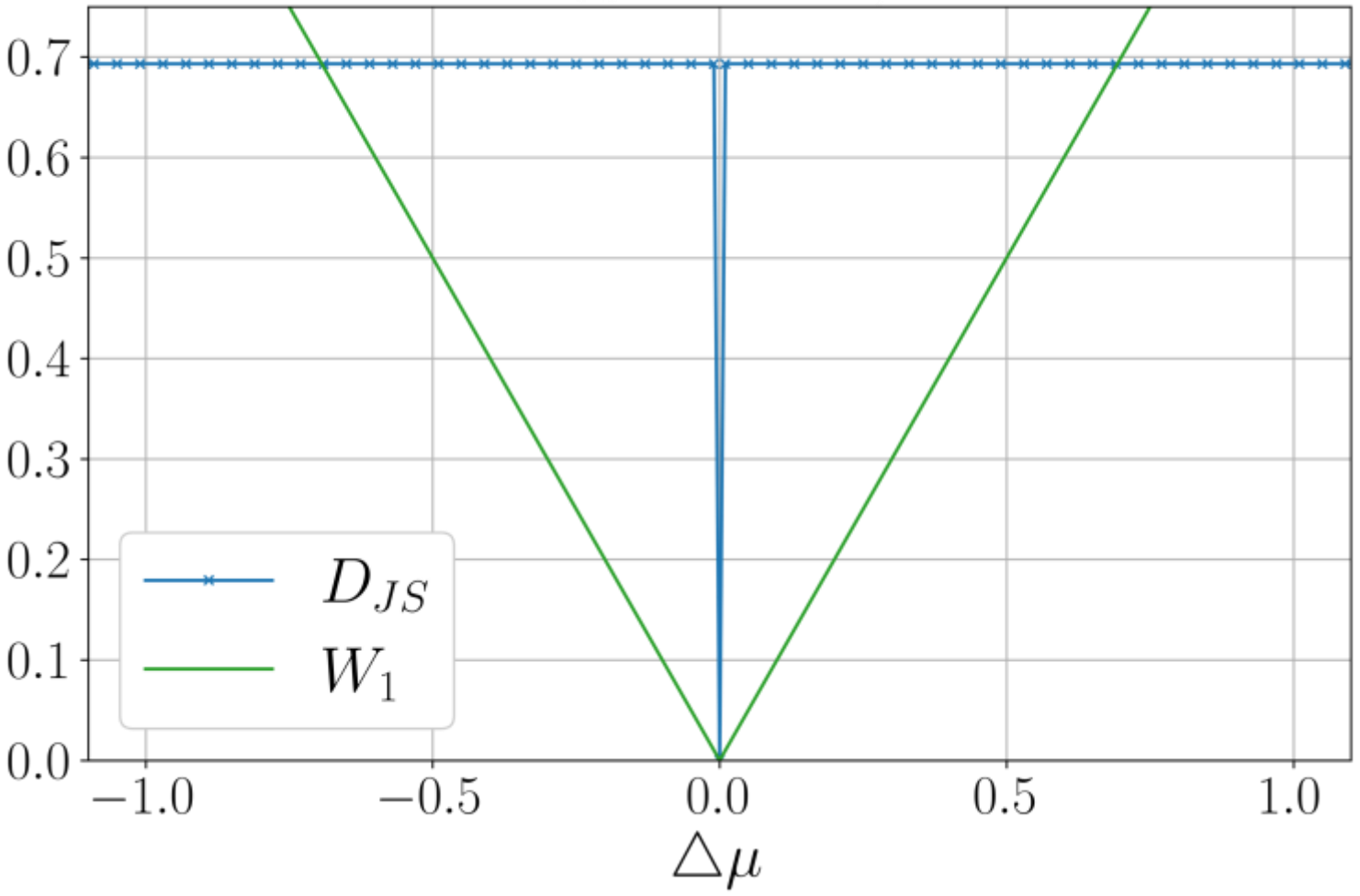


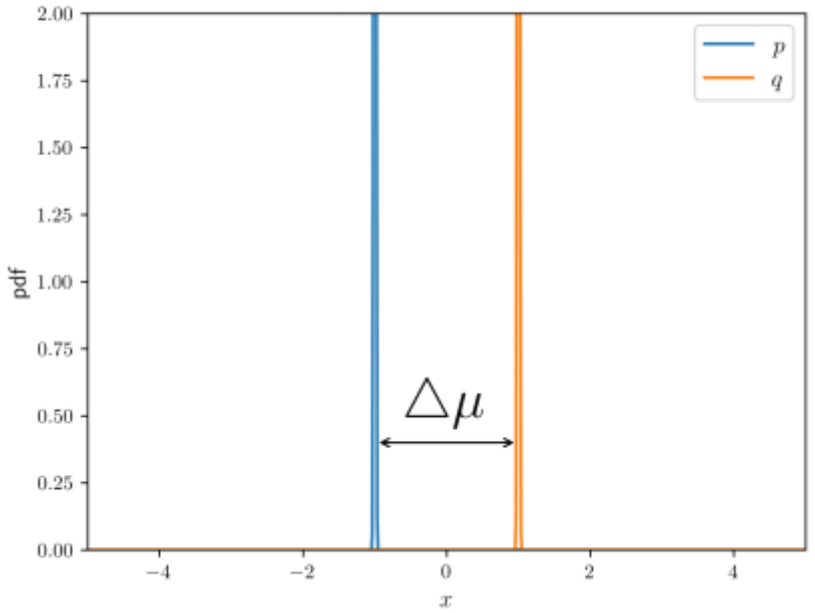


Wasserstein Distance

- when p and q are delta functions:

$$W_1(p, q) = |\mu_p - \mu_q|$$





Q x_q

P

x_p

moving plan γ

All possible plan Π

A “moving plan” is a matrix

The value of the element is the amount of earth from one position to another.

Average distance of a plan γ:

$$B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

Earth Mover’s Distance:

$$W(P, Q) = \min_{\gamma \in \Pi} B(\gamma)$$

The best plan

Wasserstein Distance

- 1-Wasserstein Distance (high-dim, continuous)

$$W_1(p,q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[|x-y|]$$

- all joint distributions $\gamma(x, y)$ whose marginals are p and q

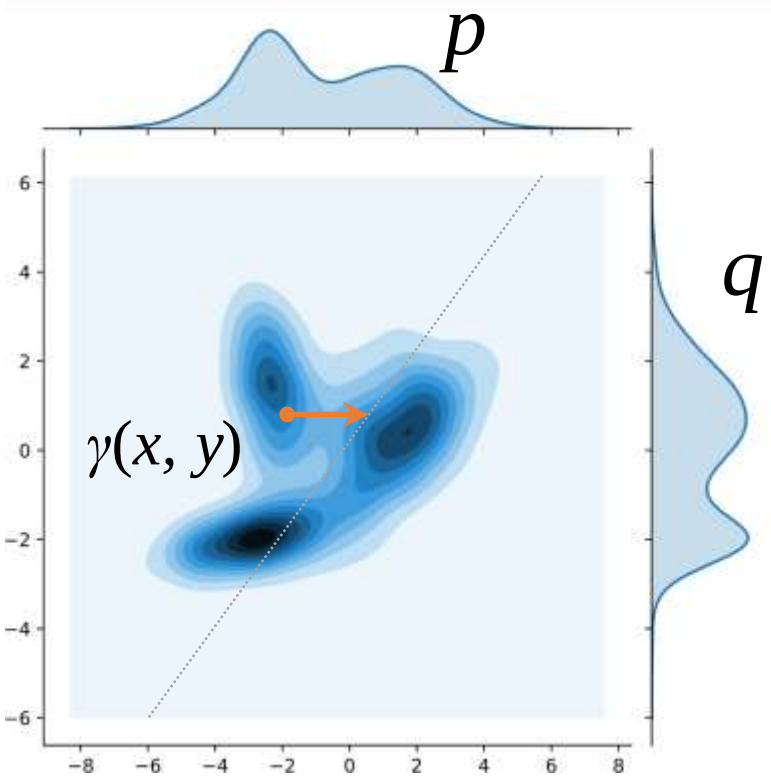
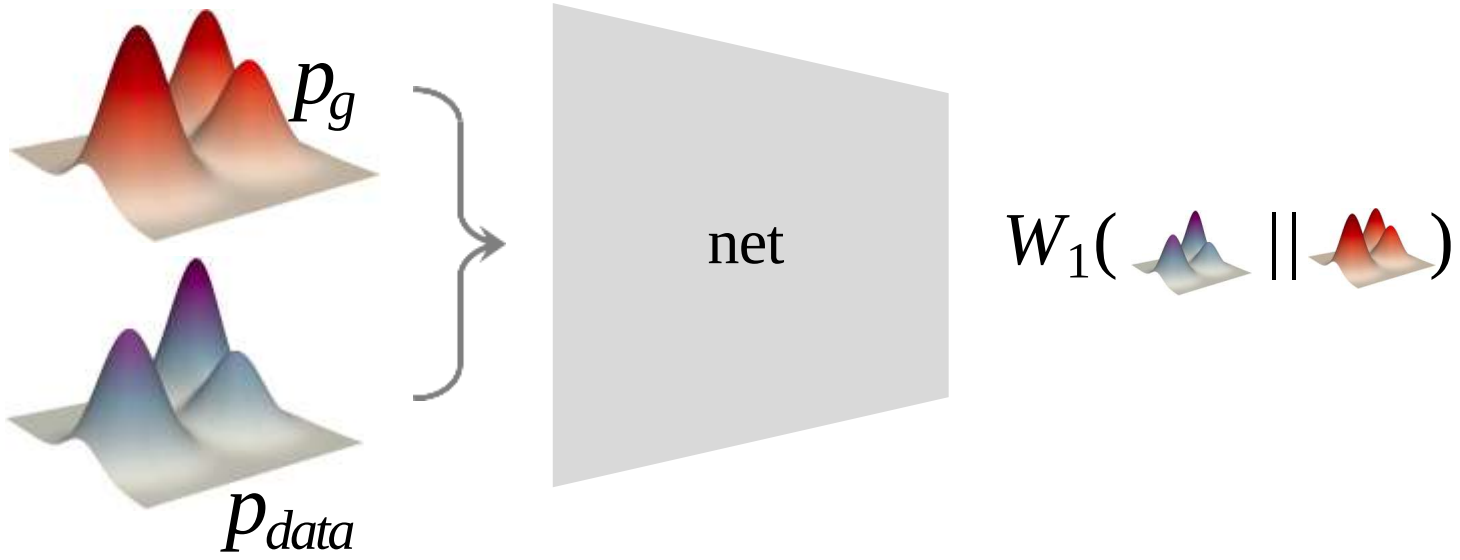


Figure: https://en.wikipedia.org/wiki/Wasserstein_metric

W-GAN optimizes for Wasserstein Distance

$$W_1(p,q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[|x-y|]$$



W-GAN optimizes for Wasserstein Distance

- Kantorovich-Rubinstein duality:

$$W_1(p, q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{x \sim q}[f(x)]$$

  - all 1-Lipschitz functions

W-GAN optimizes for Wasserstein Distance

- Kantorovich-Rubinstein duality:

$$W_1(p,q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{x\sim q}[f(x)]$$

  - all 1-Lipschitz functions

K-Lipschitz continuity:

$$|f(x) - f(y)| \leq K|x-y|, \quad \forall x, y$$

gradient is bounded:

$$\frac{|f(x) - f(y)|}{|x-y|} \leq K$$

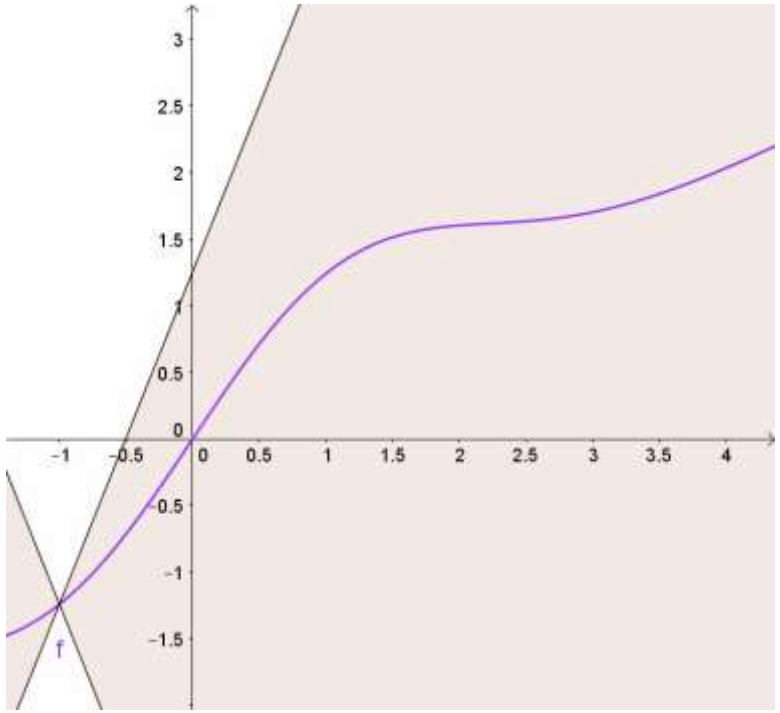


Figure from: https://en.wikipedia.org/wiki/Lipschitz_continuity

W-GAN optimizes for Wasserstein Distance

- Kantorovich-Rubinstein duality:

$$W_1(p, q) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{x \sim q}[f(x)]$$

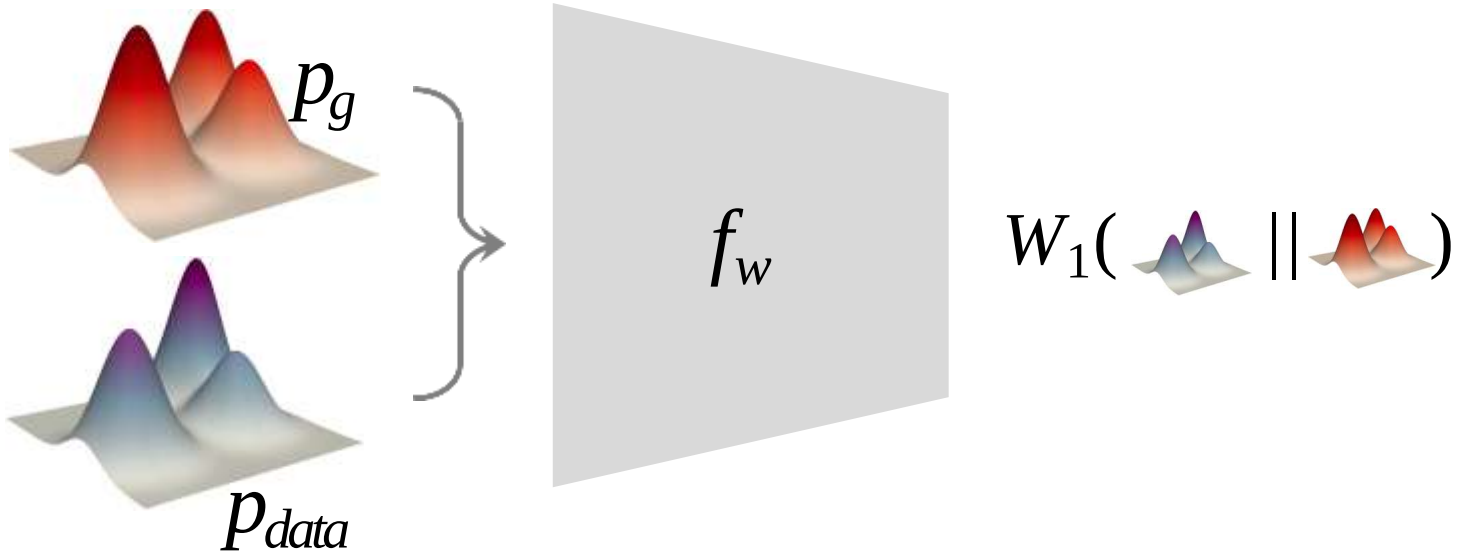
K-Lipschitz continuity:

$$|f(x) - f(y)| \leq K|x - y|, \quad \forall x, y$$

W-GAN optimizes for Wasserstein Distance

- W-GAN's objective function:

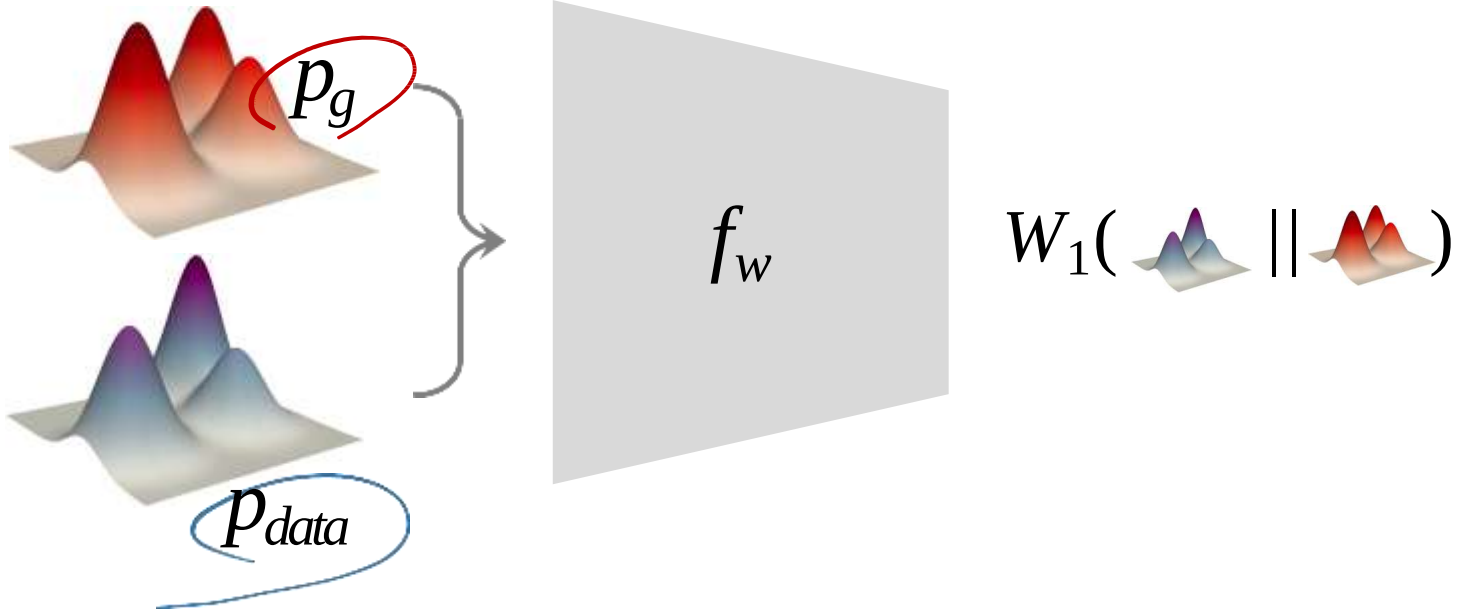
$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim p_{\text{data}}}[f_w(x)] - \mathbb{E}_{x \sim p_g}[f_w(x)]$$



W-GAN optimizes for Wasserstein Distance

- W-GAN's objective function:

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim p_{\text{data}}}[f_w(x)] - \mathbb{E}_{x \sim p_g}[f_w(x)]$$

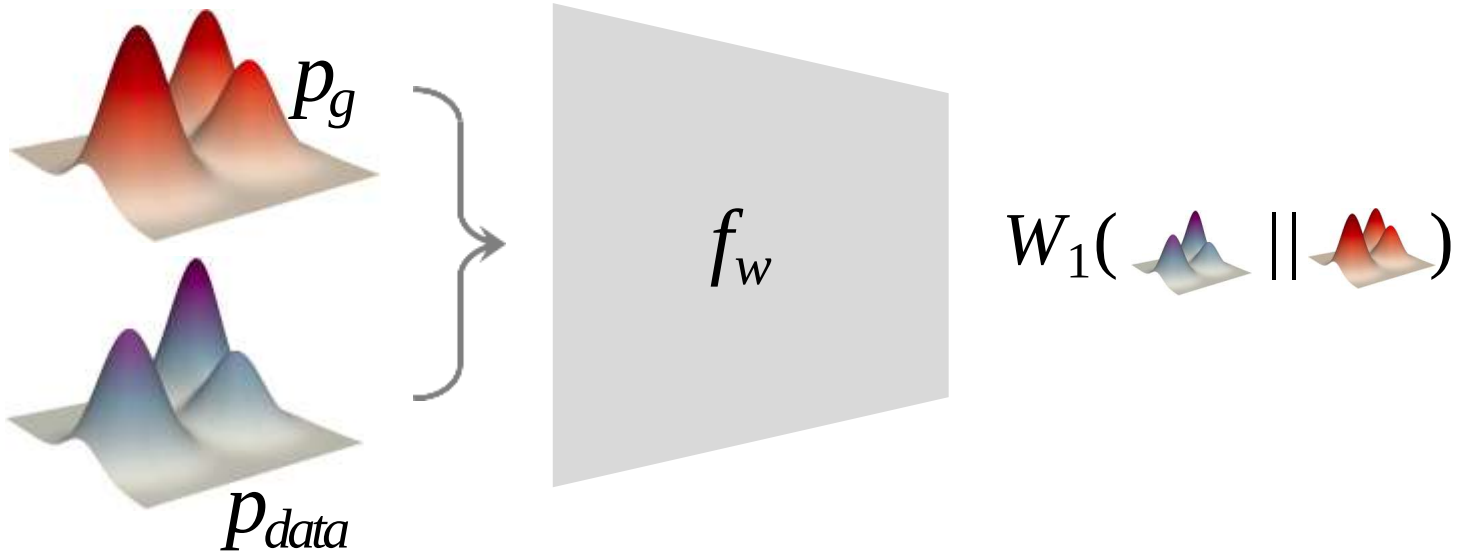


W-GAN optimizes for Wasserstein Distance

- W-GAN's objective function:

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim p_{\text{data}}}[f_w(x)] - \mathbb{E}_{x \sim p_g}[f_w(x)]$$

  - weights are bounded: in practice, clipped [-0.01, 0.01]



W-GAN vs. original GAN

- W-GAN's objective function:

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim p_{\text{data}}}[f_w(x)] - \mathbb{E}_{x \sim p_g}[f_w(x)]$$

  - clip weights
  - remove logarithms

- original GAN's objective function (D-step):

$$\max_{D} \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{x \sim p_g}[\log(1 - D(x))]$$

W-GAN vs. original GAN

"art critic"
- value/merit/quality/...
- direction to improve (gradients)

- W-GAN's objective function:

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim p_{\text{data}}}[f_w(x)] - \mathbb{E}_{x \sim p_g}[f_w(x)]$$

- original GAN's objective function (D-step):

$$\max_{D} \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{x \sim p_g}[\log(1 - D(x))]$$

"forgery expert"
- real/fake

W-GAN algorithm annotated

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

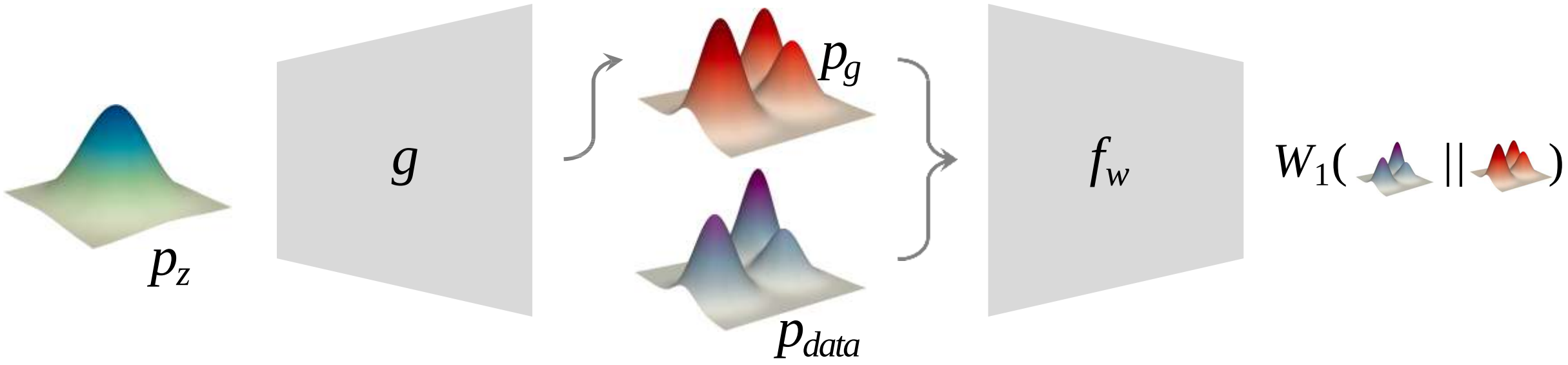
Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

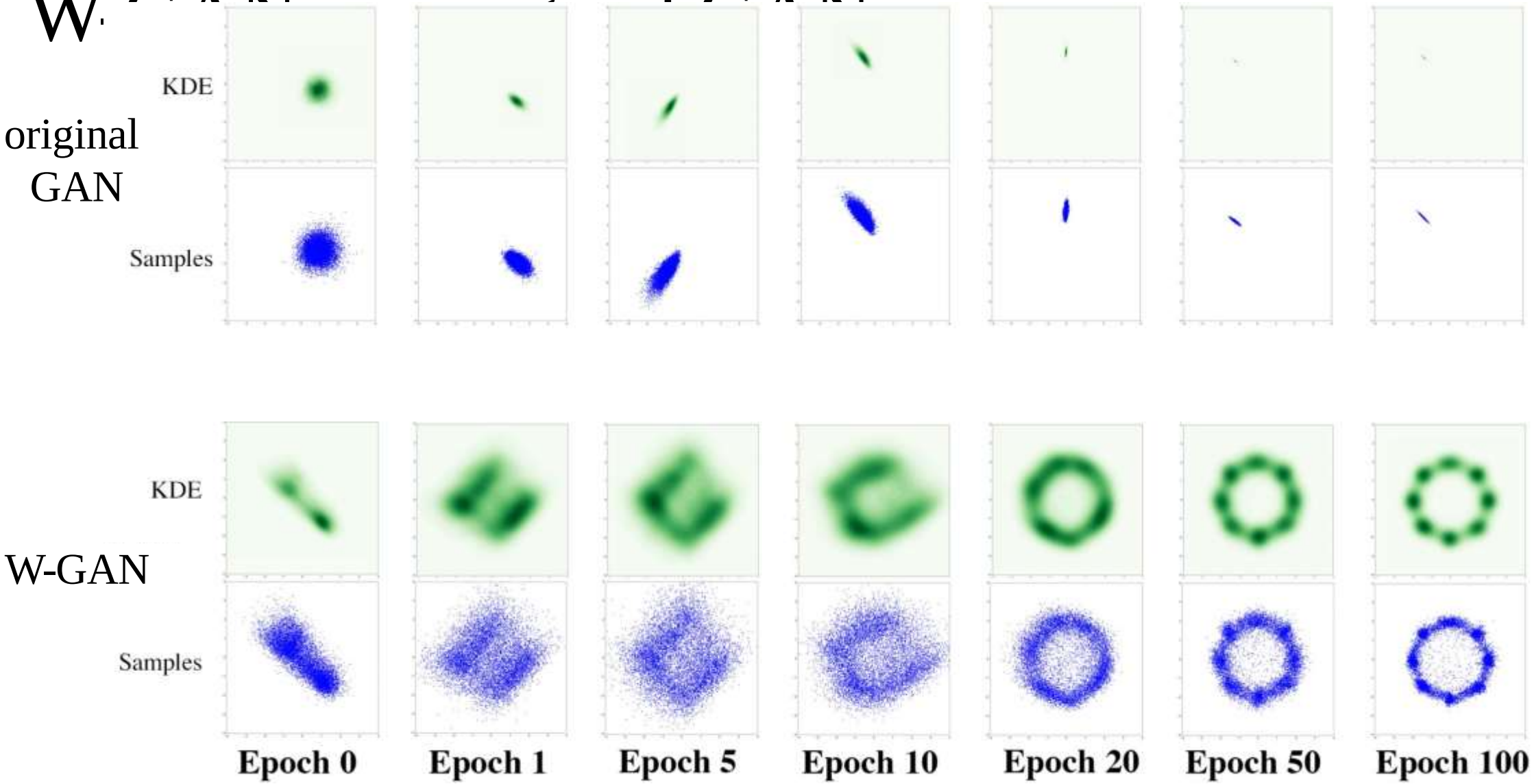
while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)})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

remove logarithms

clip weights



W- GAN vs original GAN



[Martin Arjovsky, et al., arXiv, 2017]

WGAN

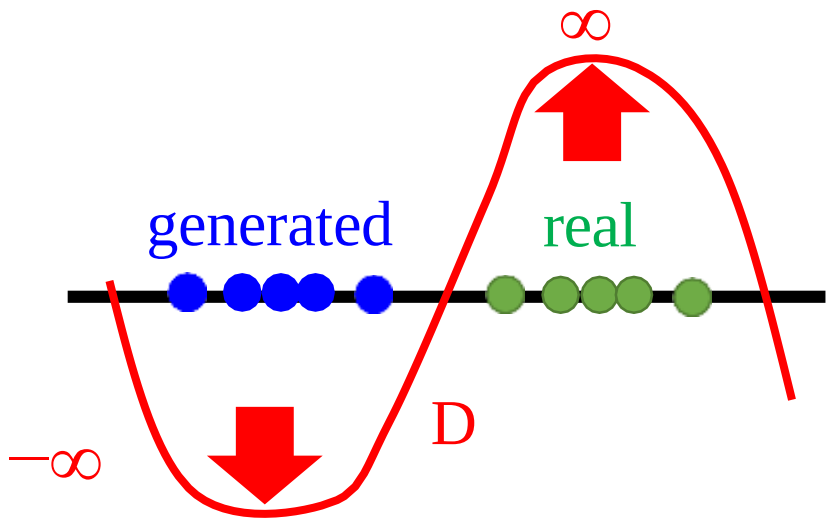
Evaluate wasserstein distance between P_{data} and P_G

$$V(G, D) = \max_{D \in 1\text{-}Lipschitz} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\}$$

D has to be smooth enough.

Without the constraint, the training of D will not converge.

Keeping the D smooth forces D(x) become ∞ and $-\infty$



WGAN

Weight Clipping [Martin Arjovsky, et al., arXiv, 2017]

Force the parameters w between c and -c After parameter update, if w > c, w = c;
if w < -c, w = -c

Evaluate wasserstein distance between P_{data} and P_G

$$V(G,D) = \max_{D \in 1-Lipschitz} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\}$$

D has to be smooth enough.

How to fulfill this constraint?

Lipschitz Function

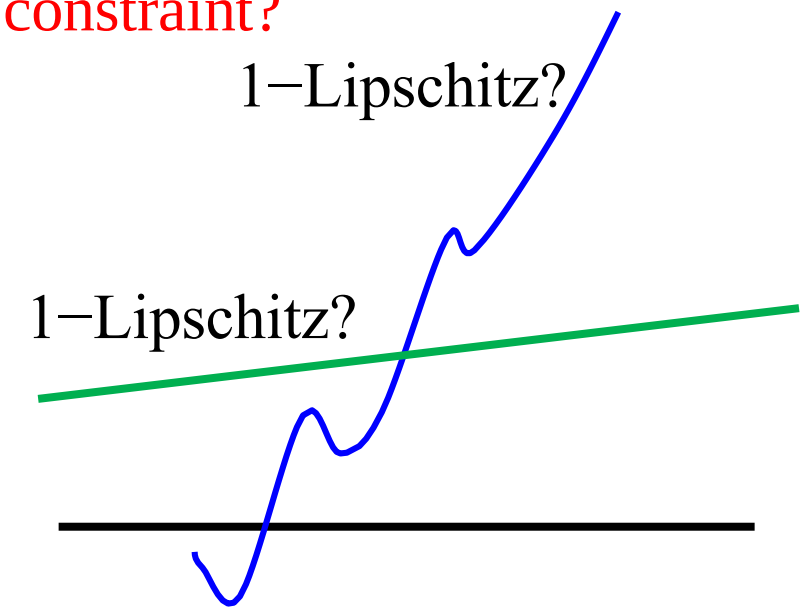
$$\|f(x_1) - f(x_2)\| \leq K \|x_1 - x_2\|$$

Output change

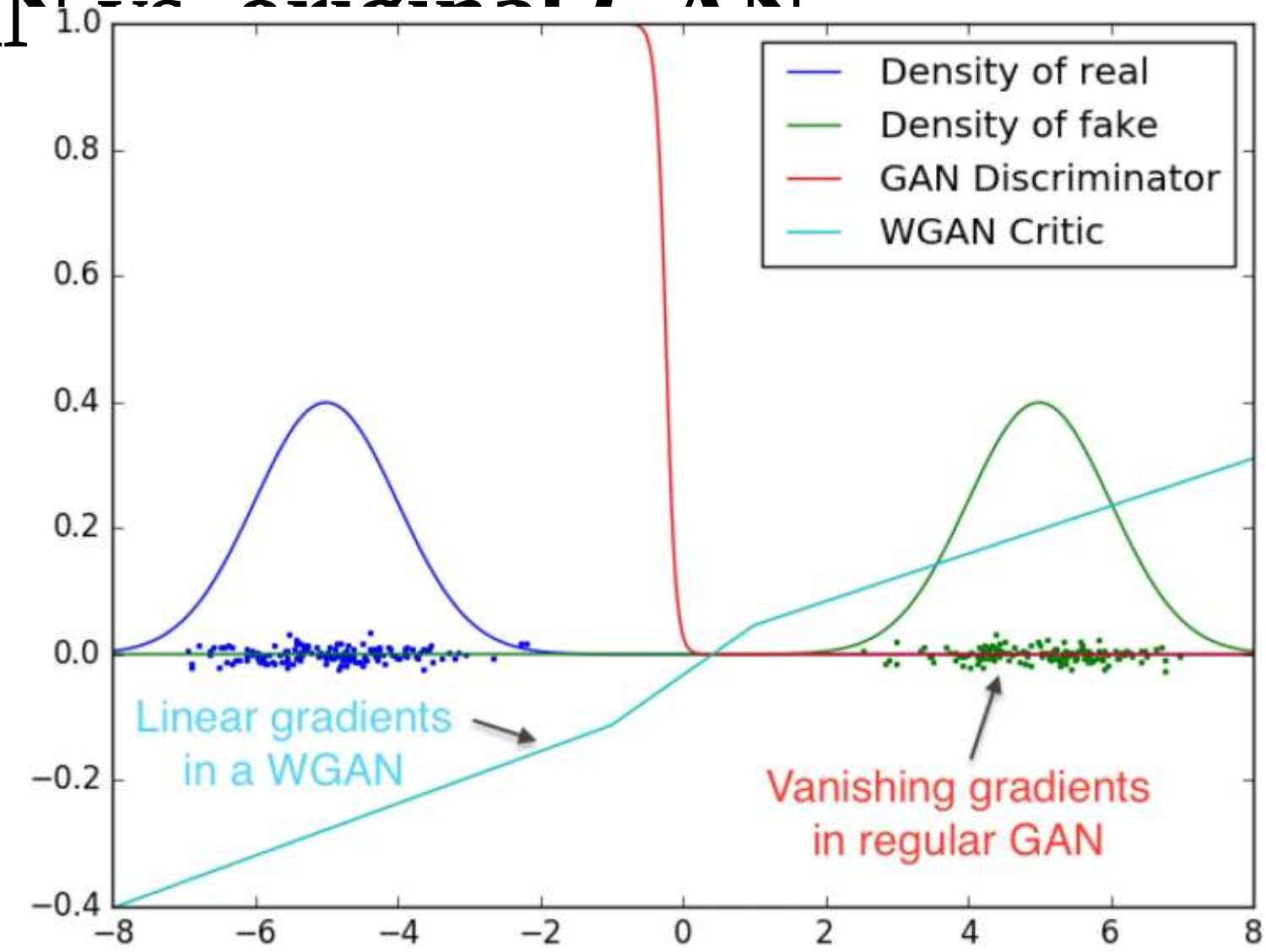
Input change

K=1 for "$1-Lipschitz$"

Do not change fast



W-GAN vs original GAN



Improved WGAN (WGAN-GP)

$$V(G,D) = \max_{D \in 1-Lipschitz} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\}$$

A differentiable function is 1-Lipschitz if and only if it has gradients with norm less than or equal to 1 everywhere.

$$D \in 1-Lipschitz \iff \|\nabla_x D(x)\| \leq 1 \text{ for all x}$$

$$V(G,D) \approx \max_D \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]$$

~~$-\lambda \int_x max(0, \|\nabla_x D(x)\| - 1)dx\}$~~

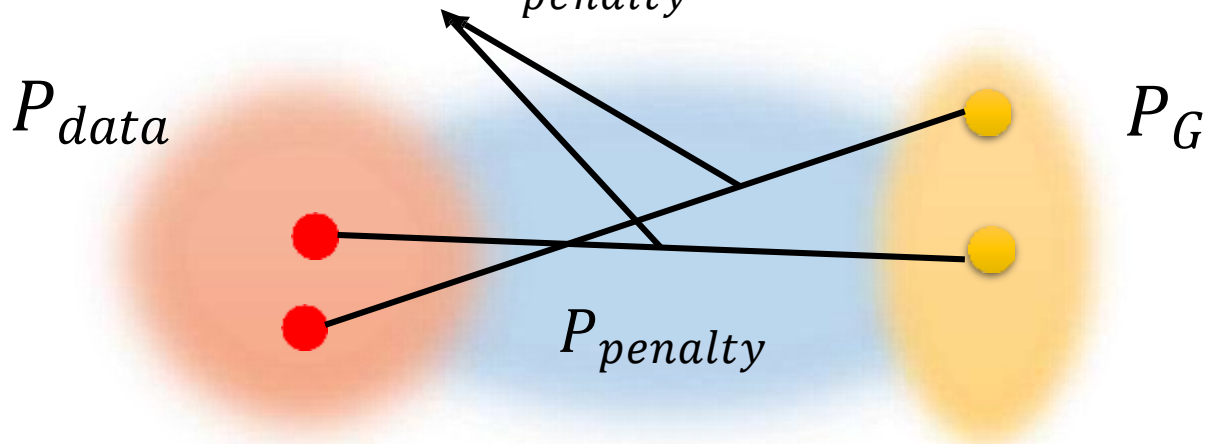
Prefer $\|\nabla_x D(x)\| \leq 1$ for all x

$$-\lambda E_{x \sim P_{penalty}}[max(0, \|\nabla_x D(x)\| - 1)]\}$$

Prefer $\|\nabla_x D(x)\| \leq 1$ for x sampling from $x \sim P_{penalty}$

Improved WGAN (WGAN-GP)

$$V(G,D) \approx \max_{D}\{E_{x\sim P_{data}}\ [D(x)] - E_{x\sim P_G}\ [D(x)]$$
$$-\lambda E_{x\sim P_{penalty}}\ [max(0, \|\nabla_x D(x)\| - 1)]\}$$



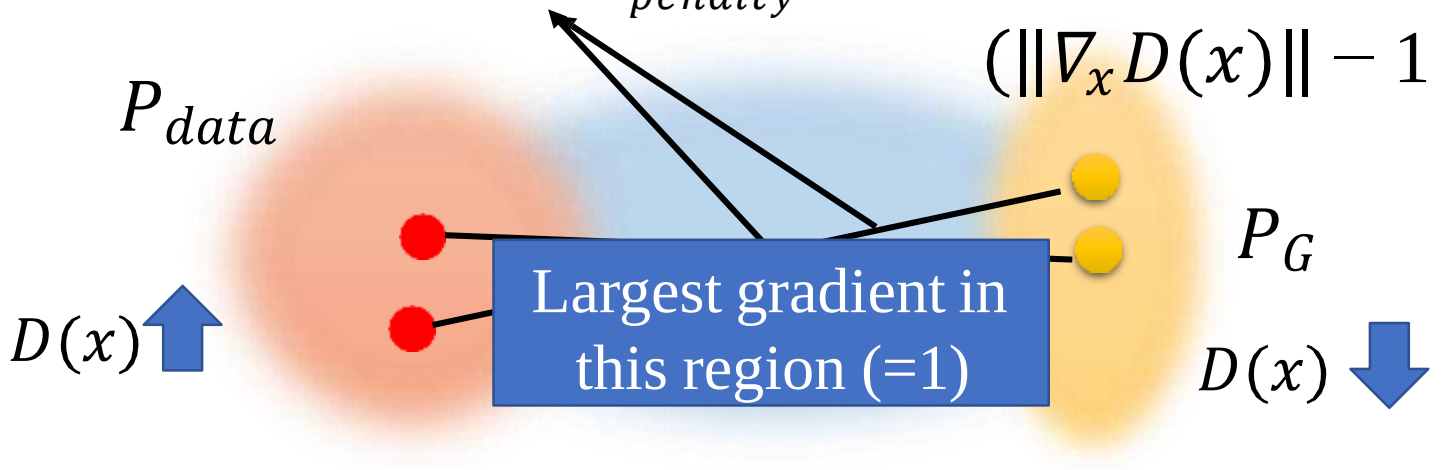
"Given that enforcing the Lipschitz constraint everywhere is intractable, enforcing it ***only along these straight lines*** seems sufficient and experimentally results in good performance."

Only give gradient constraint to the region between P_{data} and P_G because they influence how P_G moves to P_{data}

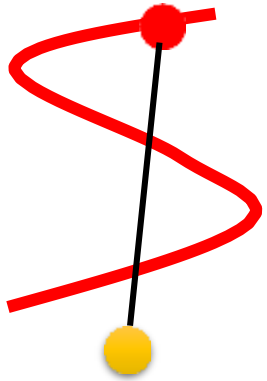
Improved WGAN (WGAN-GP)

$$V(G, D) \approx \max_{D}\{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)] - \lambda E_{x\sim P_{penalty}}[\cancel{max(0, \|\nabla_x D(x)\| - 1)}]\}$$

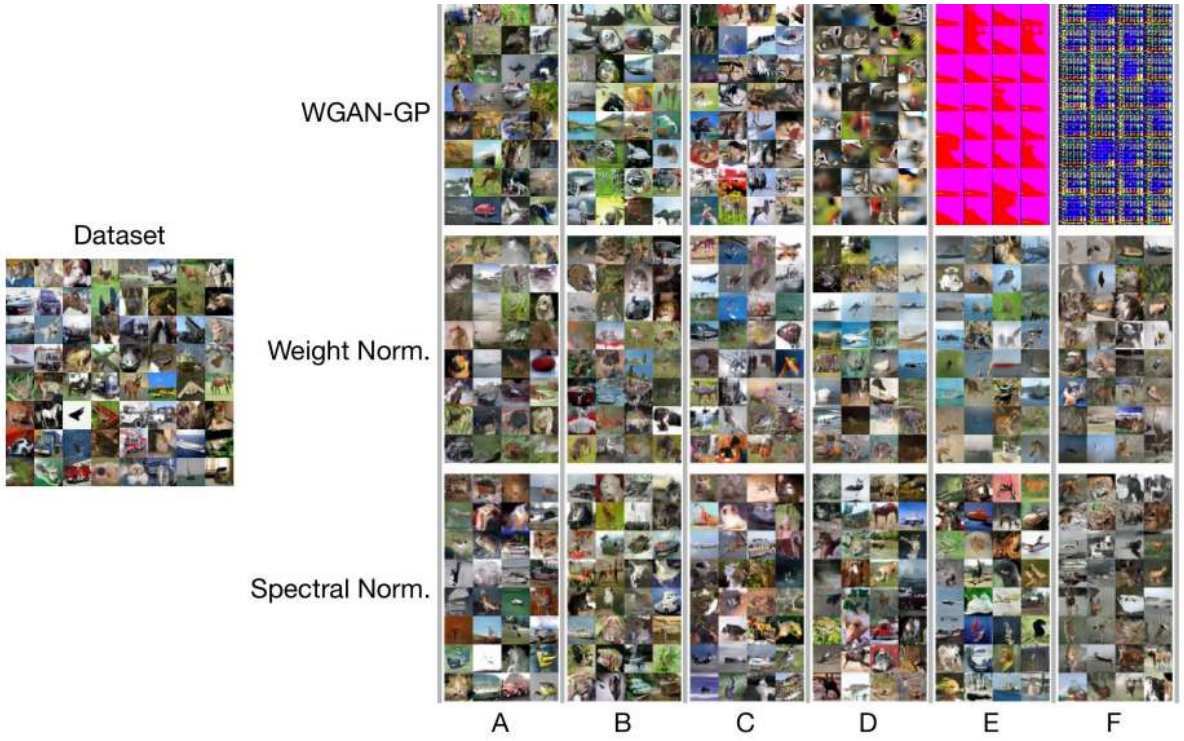
$$(\|\nabla_x D(x)\| - 1)^2$$



"Simply penalizing overly large gradients also works in theory, but experimentally we found that this approach converged faster and to better optima."



Spectrum Norm



Spectral Normalization → Keep gradient norm smaller than 1 everywhere [Miyato, et al., ICLR, 2018]

$$\|f\|_{\text{Lip}} \leq \|(\boldsymbol{h}_L \mapsto W^{L+1}\boldsymbol{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_{L-1} \mapsto W^{L}\boldsymbol{h}_{L-1})\|_{\text{Lip}}$$

$$\cdots \|a_1\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_0 \mapsto W^{1}\boldsymbol{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\boldsymbol{h}_{l-1} \mapsto W^{l}\boldsymbol{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l).$$

W-GAN in Short

For mathematicians:

- Wasserstein distance, instead of JS divergence

For engineers:

- remove logarithms
- clip weights

Wasserstein distance

Lipschitz continuity

For laymen:

- art critic instead of a forgery expert

gradients

Summarized by Reddit user danielvarga: https://www.reddit.com/r/MachineLearning/comments/5qxoaz/comment/dd7aomb/

Brief: LSGAN, EBGAN

- Least Square (LS) GAN:

$$\mathbb{E}_{x\sim p_{\text{data}}}(D(x)-b)^2 \quad + \quad \mathbb{E}_{x\sim p_g}(D(x)-a)^2$$

- Energy-based (EB) GAN:

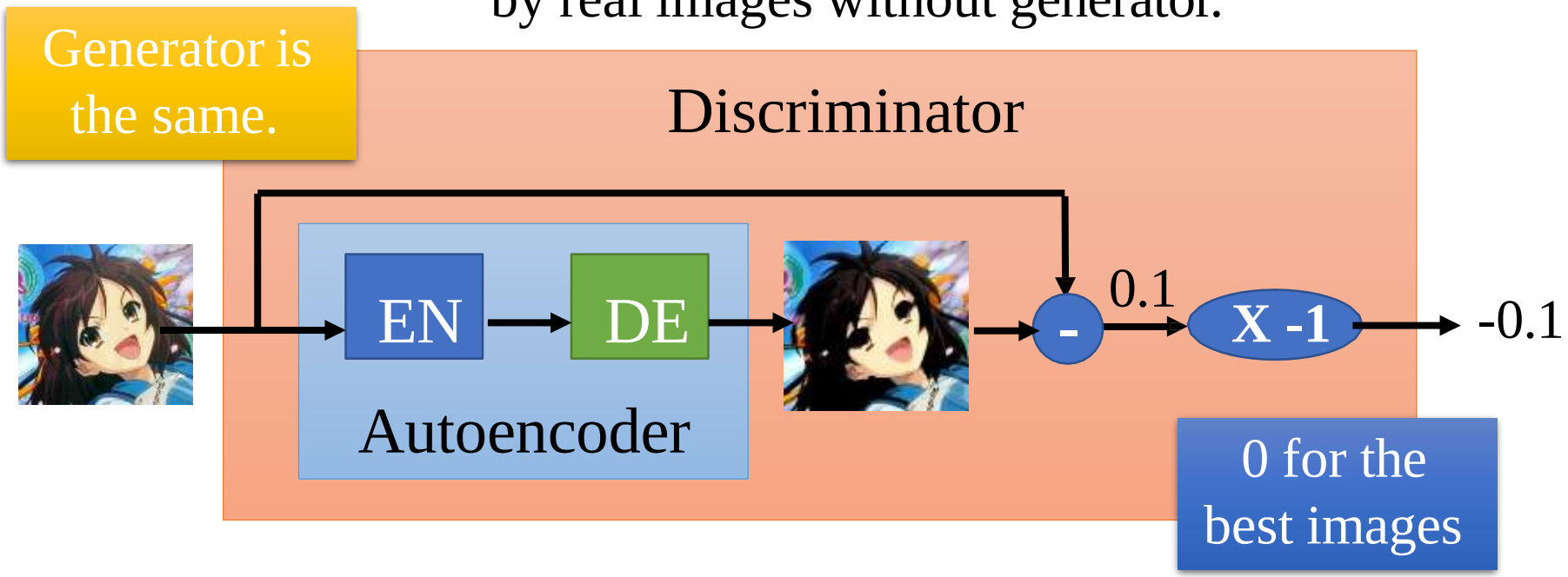
$$\mathbb{E}_{x\sim p_{\text{data}}}D(x) \quad + \quad \mathbb{E}_{x\sim p_g}[m-D(x)]^+$$

Mao, et al., “Least Squares Generative Adversarial Networks”, ICCV 2017
Zhao, et al., “Energy-based Generative Adversarial Networks”, ICLR 2017

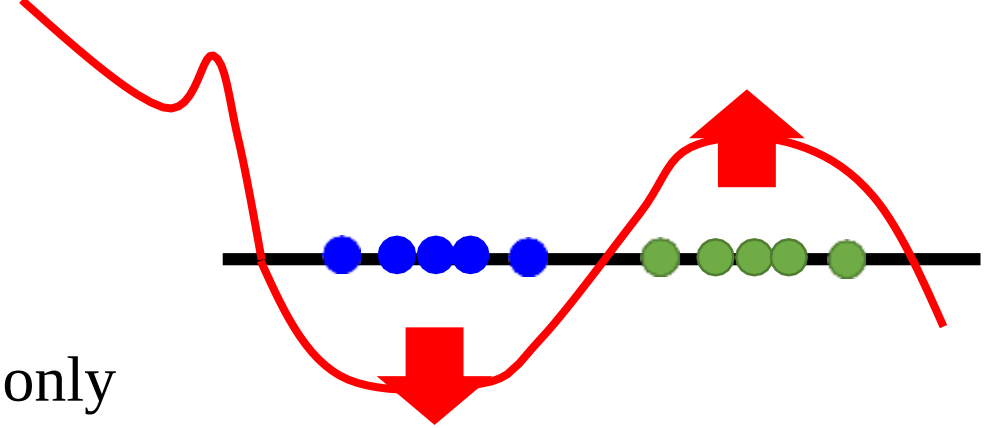
[Junbo Zhao, et al., arXiv, 2016]

Energy-based GAN (EBGAN)

- Using an autoencoder as discriminator D
  - ➢ Using the negative reconstruction error of auto-encoder to determine the goodness
  - ➢ **Benefit**: The auto-encoder can be pre-train by real images without generator.

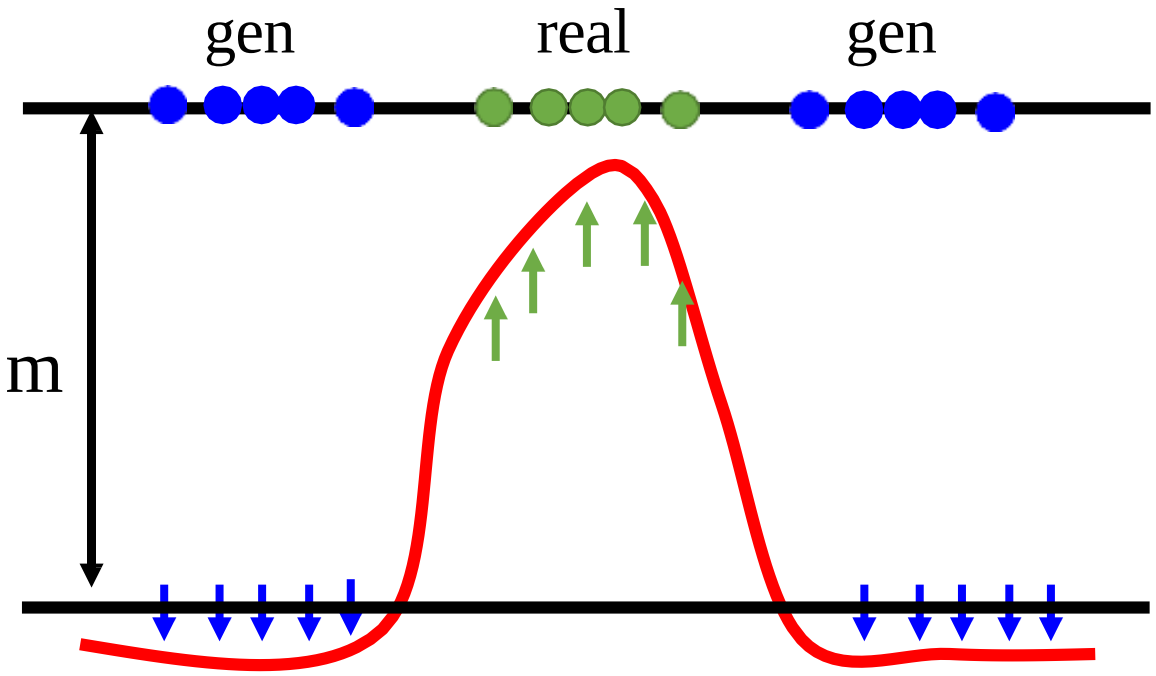


EBGAN



Auto-encoder based discriminator only gives limited region large value.

0 is for the best.

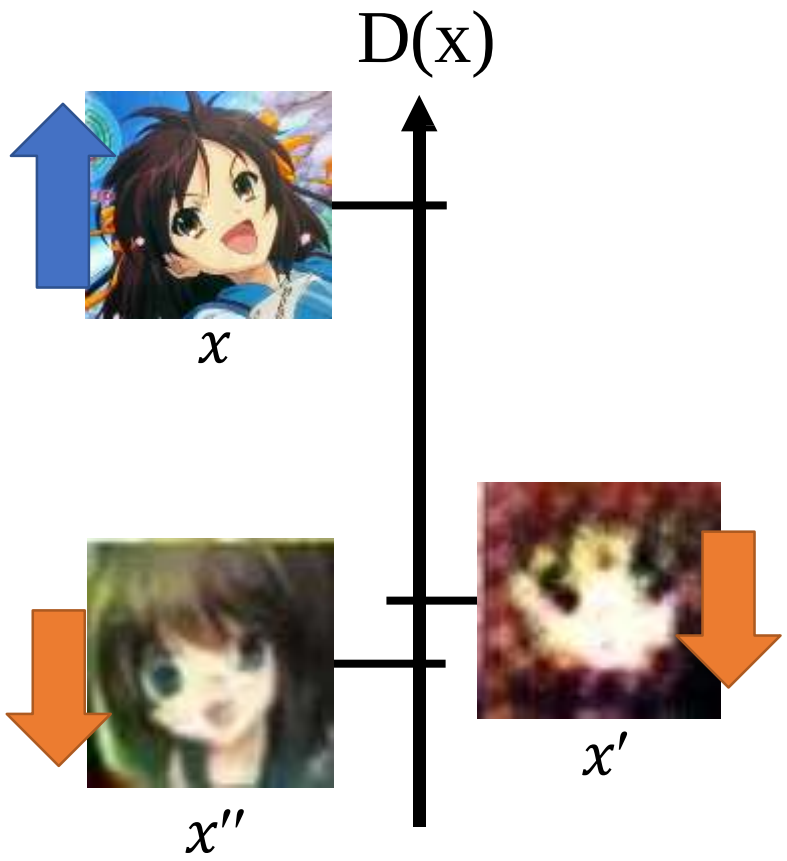


Do not have to be very negative

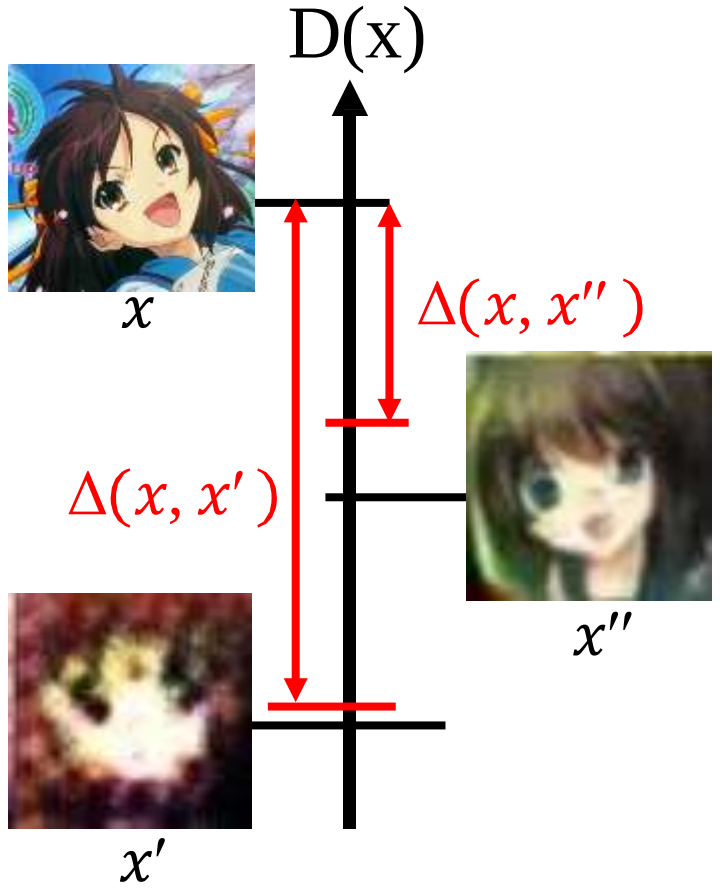
Hard to reconstruct, easy to destroy

Loss-sensitive GAN (LSGAN)

WGAN



LSGAN



About GANs

- Generative Adversarial Networks (GAN)
- Adversary as a Loss Function
- Wasserstein GAN (W-GAN)

Main References

- Goodfellow et al. “Generative Adversarial Nets”, NeurIPS 2014
- Arjovsky et al. “Wasserstein Generative Adversarial Networks”, ICML 2017
- Zhuowen Tu. “Learning Generative Models via Discriminative Approaches”, CVPR 2007