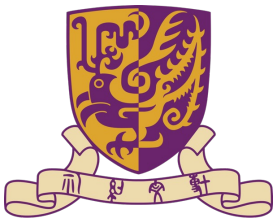


AIR5101 / CIE6021

Generative AI

Lecture 2: Generative Adversarial Networks

Instructor: Zhen Liu

Spring 2025, CUHK-Shenzhen

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie[*], Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair[†], Aaron Courville, Yoshua Bengio[‡]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

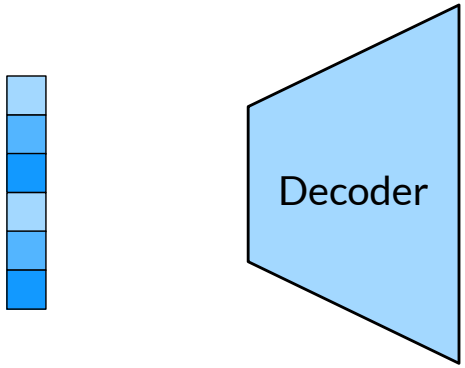

LES 3 BRASSEURS

Agenda

- Vanilla GAN
- Variants (W-GAN, etc.)
- Adversary as a Loss

Vanilla GAN

Recap: Latent Variable Model



$z \sim p(z)$

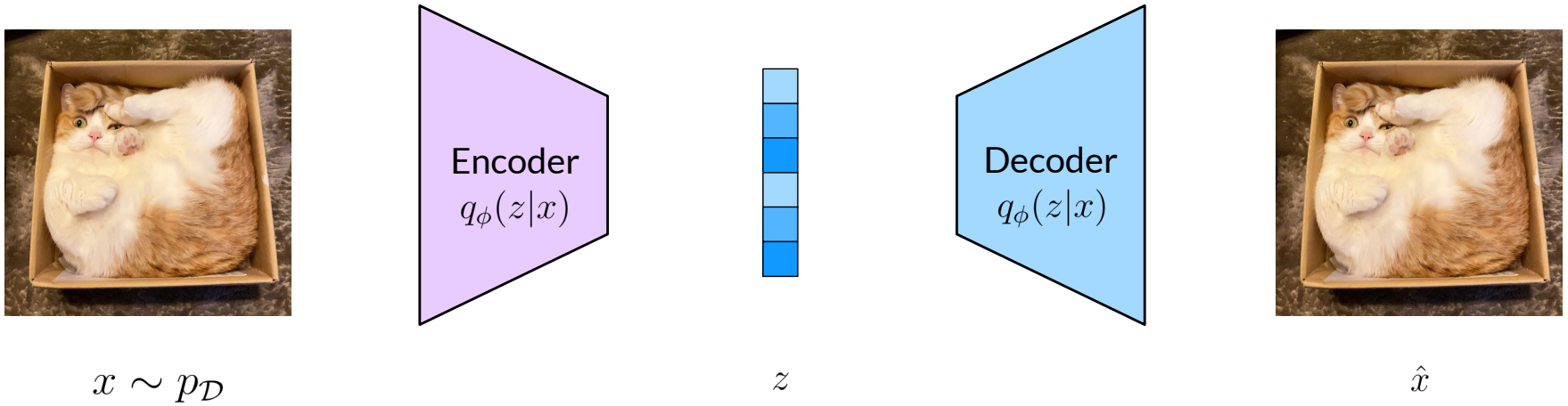
Latent features

Shape, Pose, …

$p_\theta(x)$

Target Distribution

Recap: Variational Autoencoder (VAE)



$x \sim p_{\mathcal{D}}$ z $\hat{x}$

Recall: Vanilla VAE on Images

Blurry reconstructed images even on simple datasets

- Canonical pose
- Canonical size
- Single-category objects
- Consistent lighting

Why VAEs fail in high-dimensional data?

Strong assumptions

- Gaussian encoder + Gaussian latent distribution

Why VAEs fail in high-dimensional data?

Strong assumptions

- Gaussian encoder + Gaussian latent distribution
- Gaussian decoder

Why VAEs fail in high-dimensional data?

Strong assumptions

- Gaussian encoder + Gaussian latent distribution
- Gaussian decoder

Any other metrics for measuring difference b/w distributions?

Why VAEs fail in high-dimensional data?

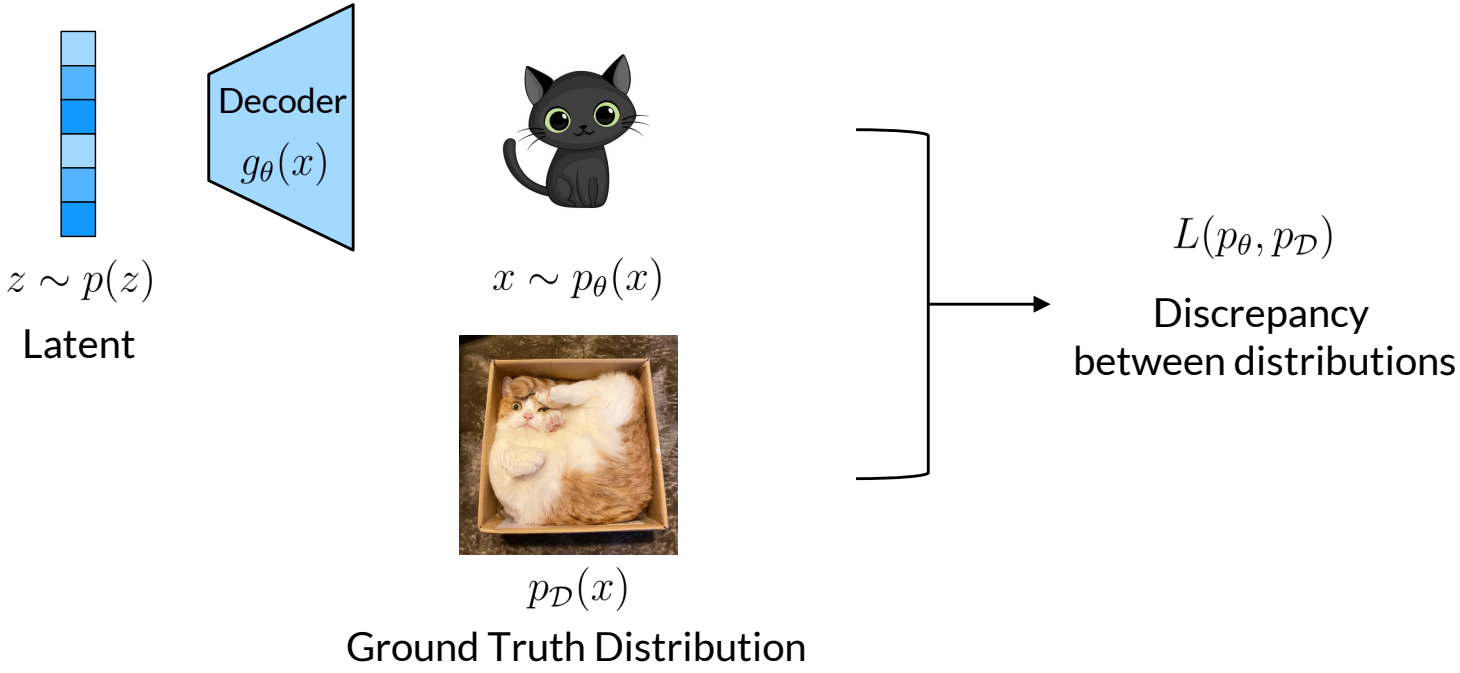
Strong assumptions

- Gaussian encoder + Gaussian latent distribution
- Gaussian decoder

Any other metrics for measuring difference b/w distributions?

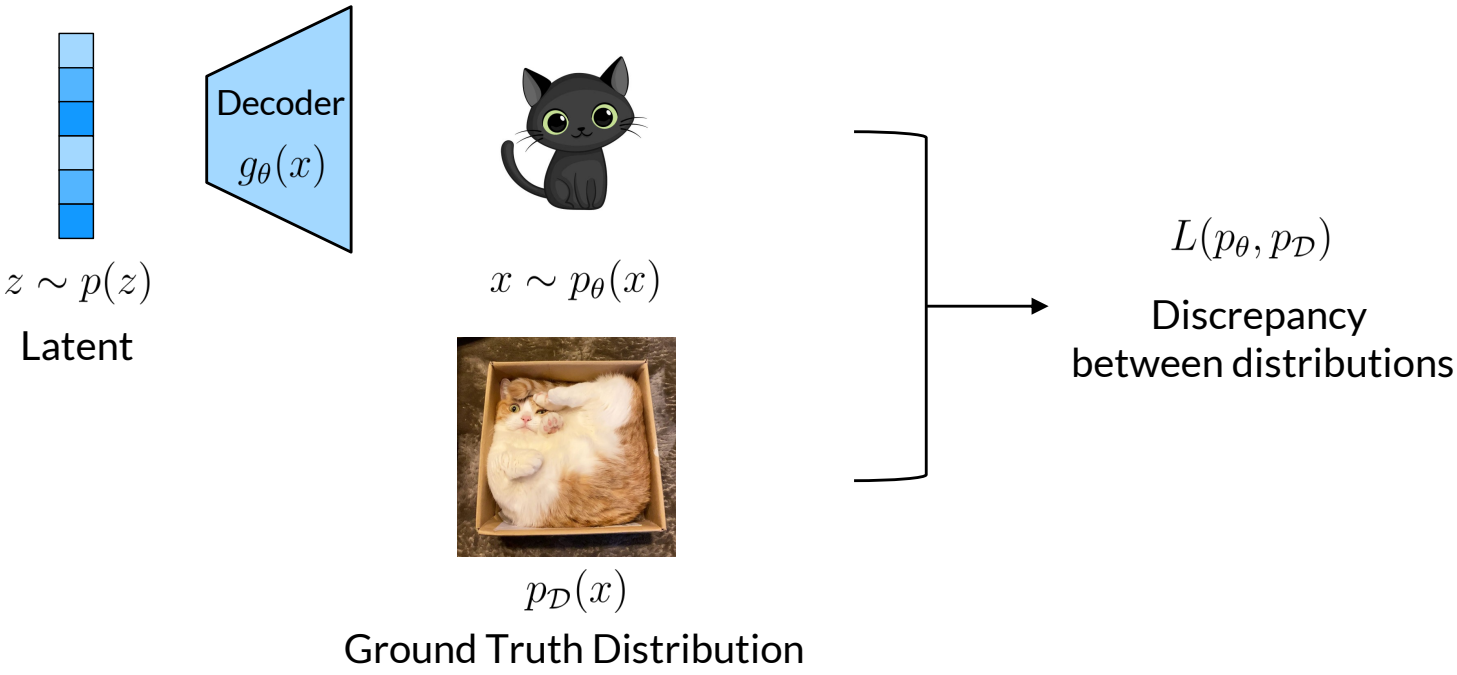
Generative Adversarial Networks

What should be a good discrepancy function?



Generative Adversarial Networks

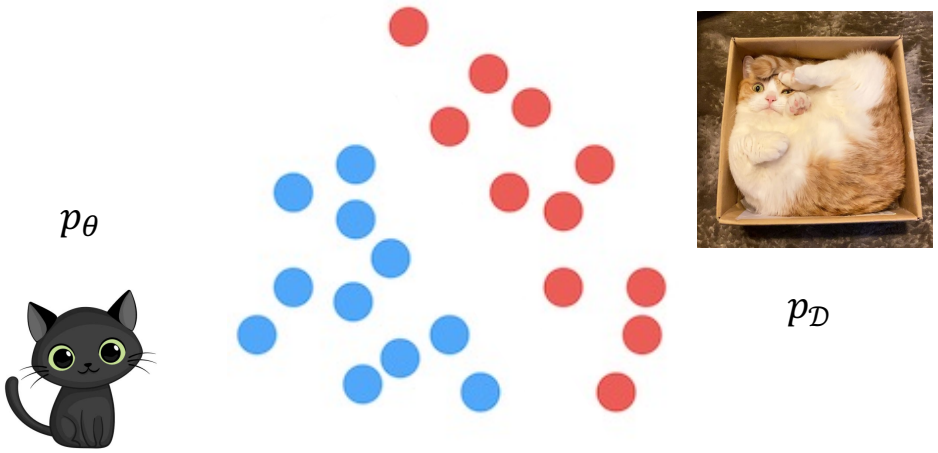
What should be a good discrepancy function?



Discriminator

Recall that $L(p_\theta, p_\mathcal{D}) = 0$ when $p_\theta = p_\mathcal{D}$.

What is the easy thing to distinguish two sets?

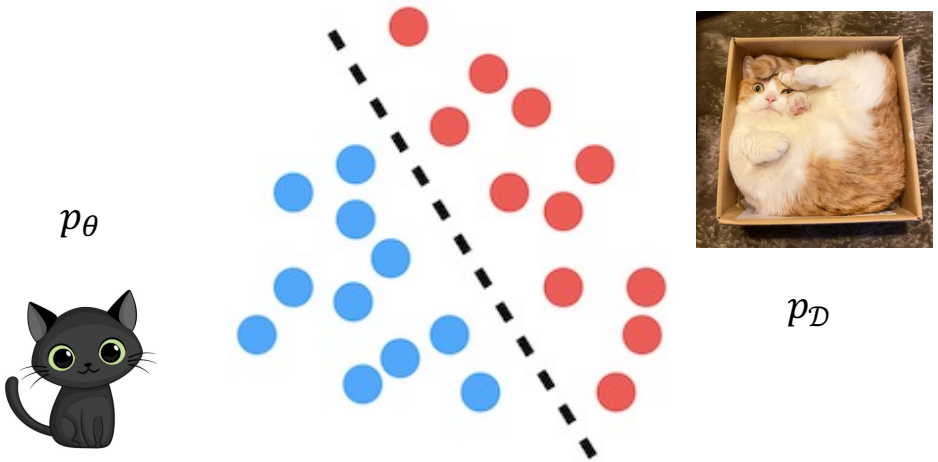


Discriminator

Recall that $L(p_\theta, p_\mathcal{D}) = 0$ when $p_\theta = p_\mathcal{D}$.

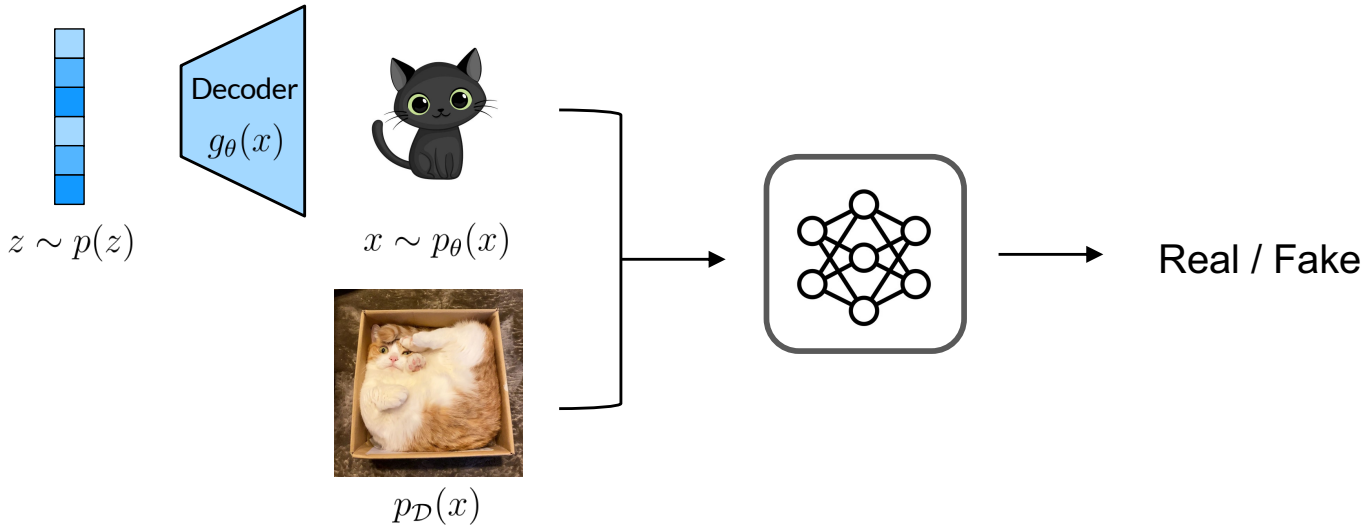
What is the easy thing to distinguish two sets?

A binary classifier!



Generative Adversarial Networks

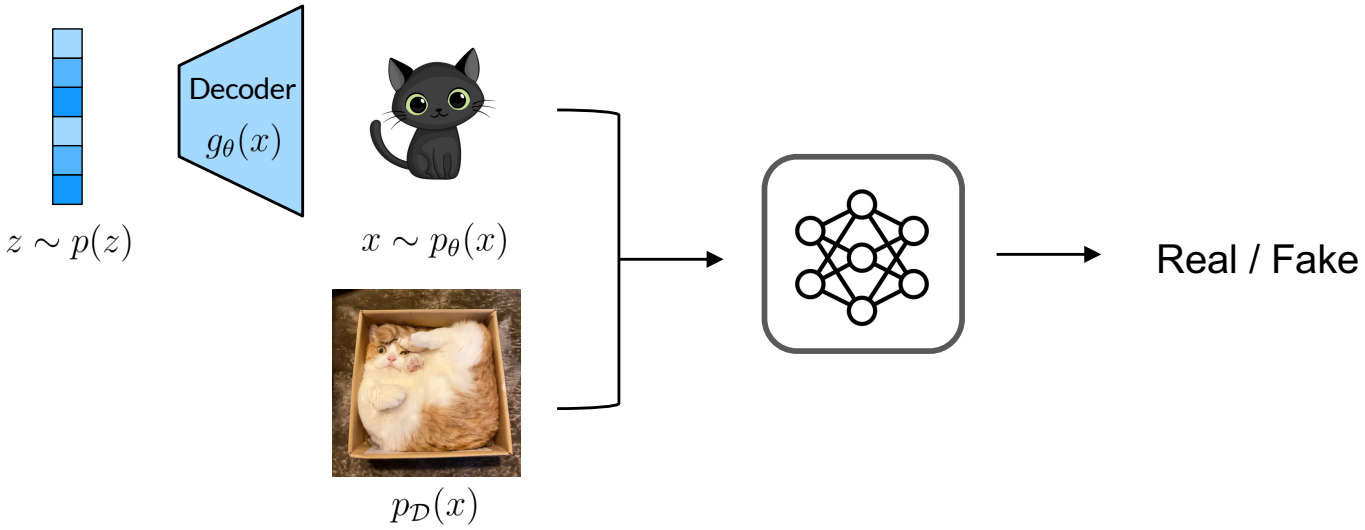
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$



Generative Adversarial Networks

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

Min-max optimization

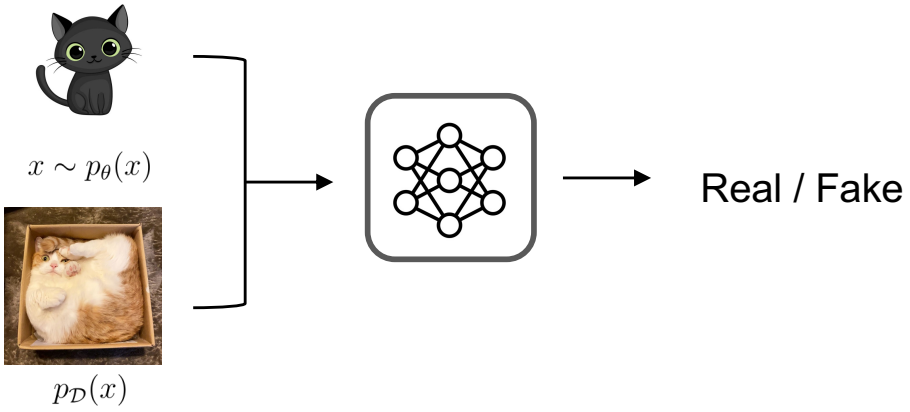


Generative Adversarial Networks

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

Optimizing D (G fixed): Data: label 1 Generated: label 0

Logistic regression $\mathcal{L} = -\mathbb{E}_{x\sim p_{\text{data}}}[\log f(x)] - \mathbb{E}_{x\sim p_g}[\log(1 - f(x))]$

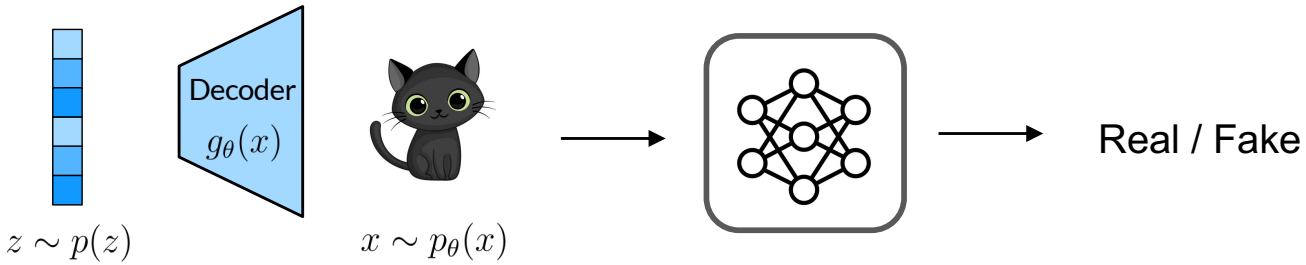


Generative Adversarial Networks

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

Optimizing G (D fixed)

Generator as adversary

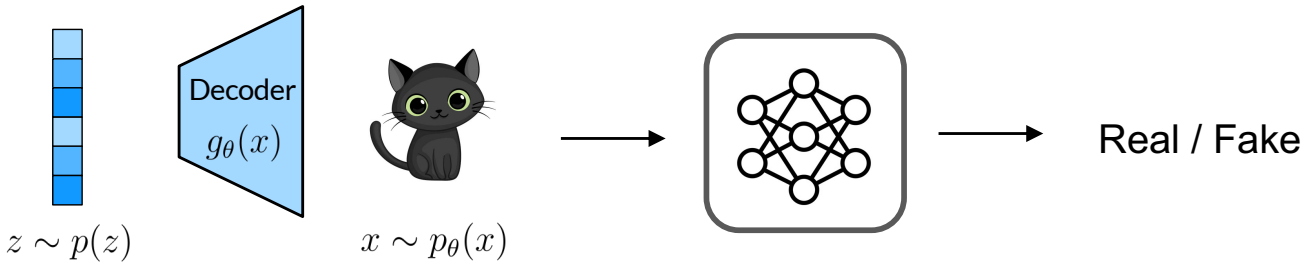


Generative Adversarial Networks

$$\min_G \; \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Deceive the discriminator
Push D(G(z)) to 1

Optimizing G (D fixed)
Generator as adversary

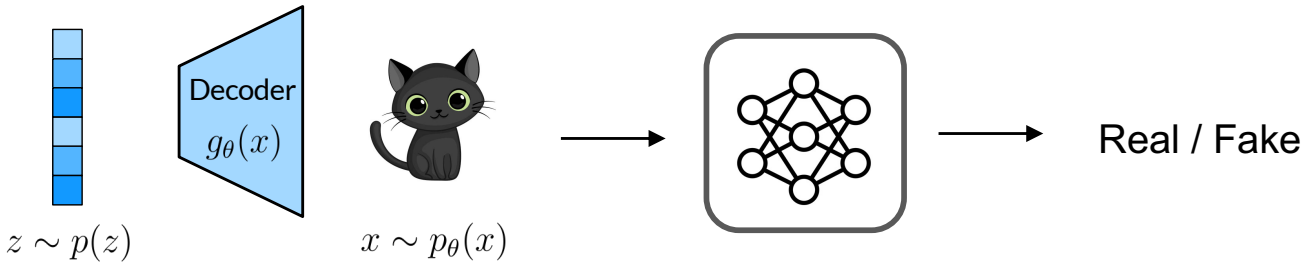


Generative Adversarial Networks

$$\min_G \ \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Optimizing G (D fixed)
Generator as adversary

Almost equivalent to $\max_G \ \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(D(G(\boldsymbol{z})))]$

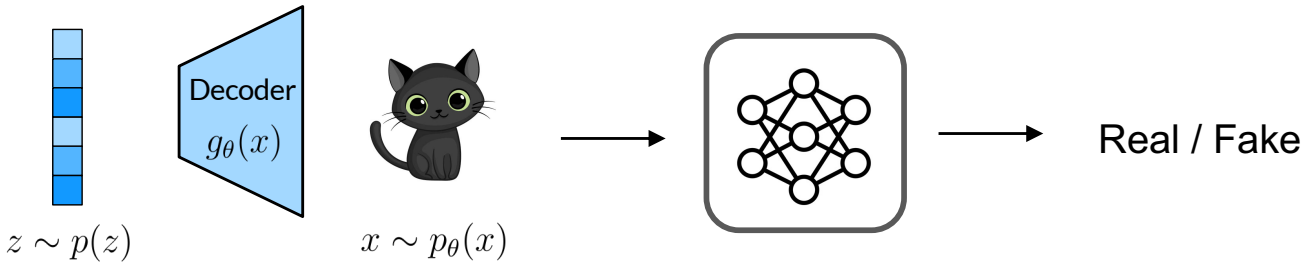


Generative Adversarial Networks

$$\min_G \ \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Optimizing G (D fixed)
Generator as adversary

Almost equivalent to $\max_G \ \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(D(G(\boldsymbol{z})))]$



Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

Critic Update

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

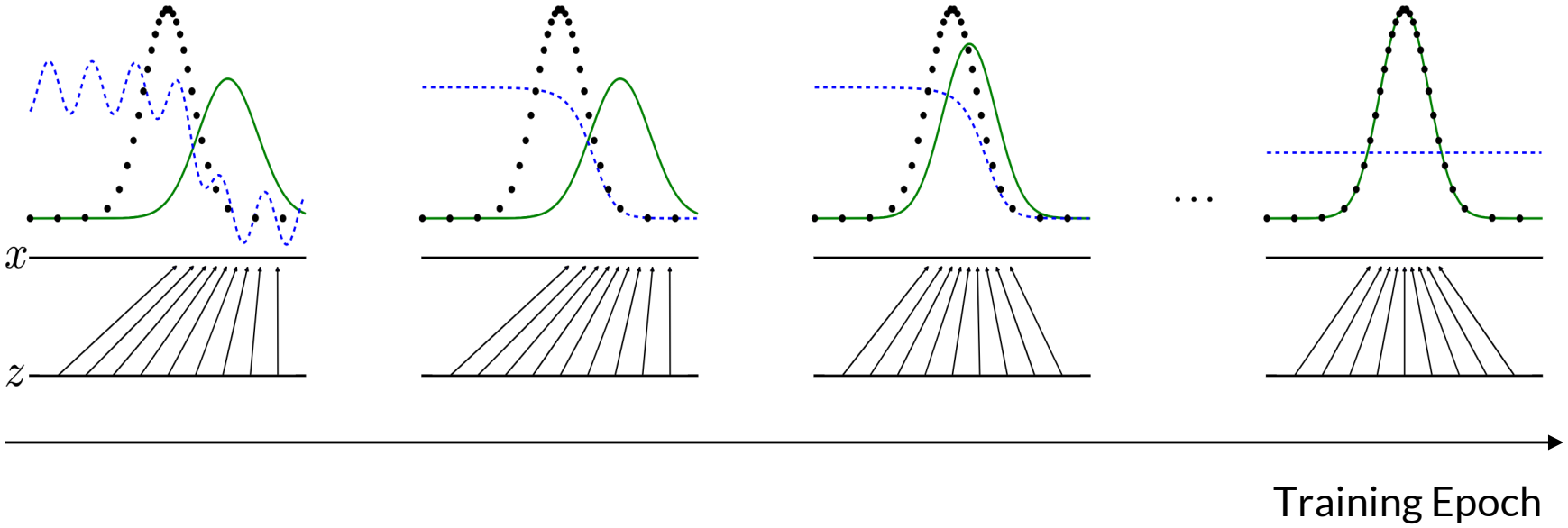
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

Generator Update

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Illustration

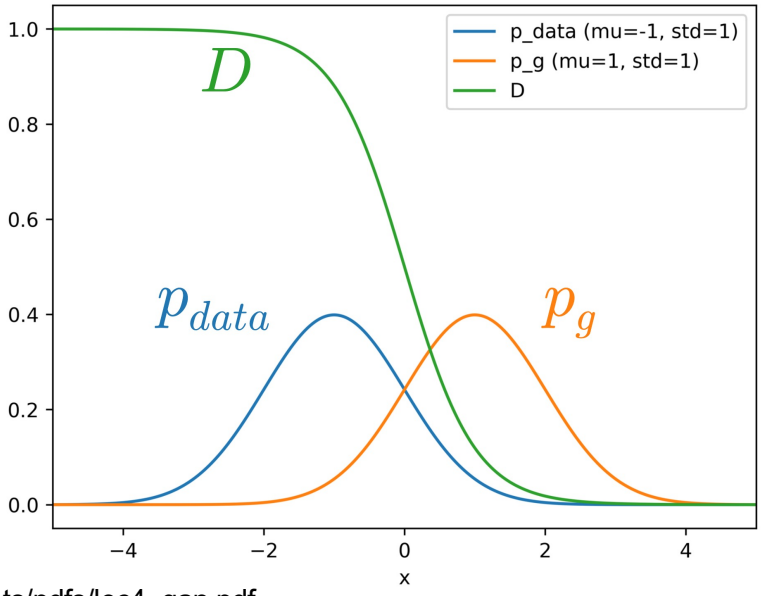


Black: data points
Green: modeled distribution
Blue: Discriminator

Theory

Proposition 1. *For G fixed, the optimal discriminator D is*

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



https://mit-6s978.github.io/assets/pdfs/lec4_gan.pdf

Theory

Proposition 1. *For G fixed, the optimal discriminator D is*

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

Indeed, $D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$ is a generative model itself.

Alternative interpretation: D = generative model, G = sampler

Will revisit briefly in the next lecture

https://mit-6s978.github.io/assets/pdfs/lec4_gan.pdf

Theory

What is the optimal discriminator given G?

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

Jensen-Shannon Divergence

JS Divergence

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

Symmetric difference to the “middle point”

Theory

JS Divergence

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

Properties:

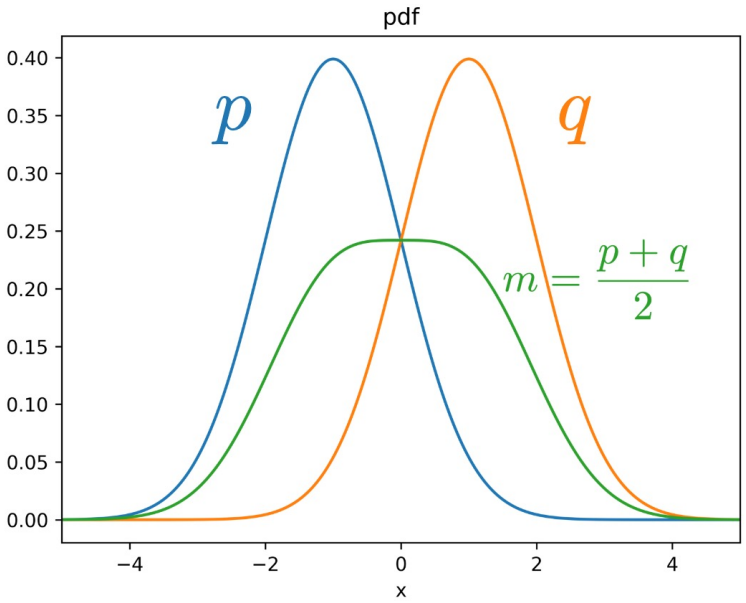
- Symmetric (KL divergence is not)
- Bounded in [0, log 2]
- Smoother gradients

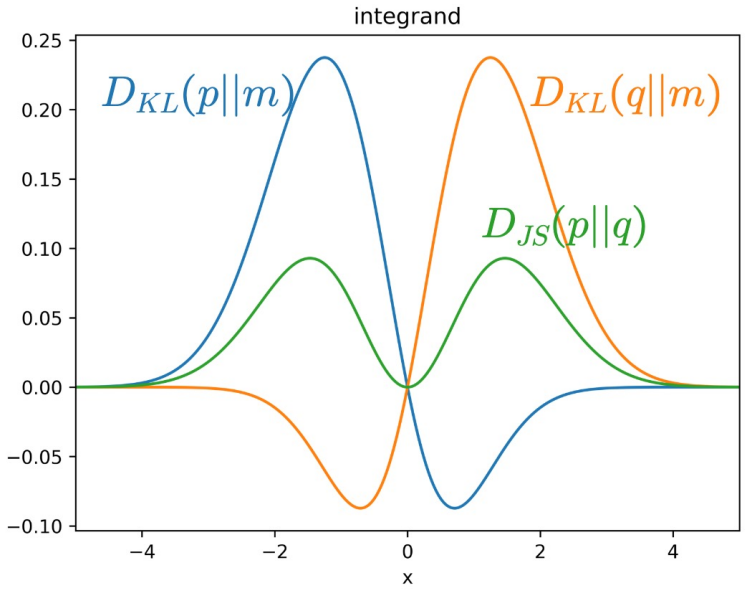
https://mit-6s978.github.io/assets/pdfs/lec4_gan.pdf

Theory

JS Divergence

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



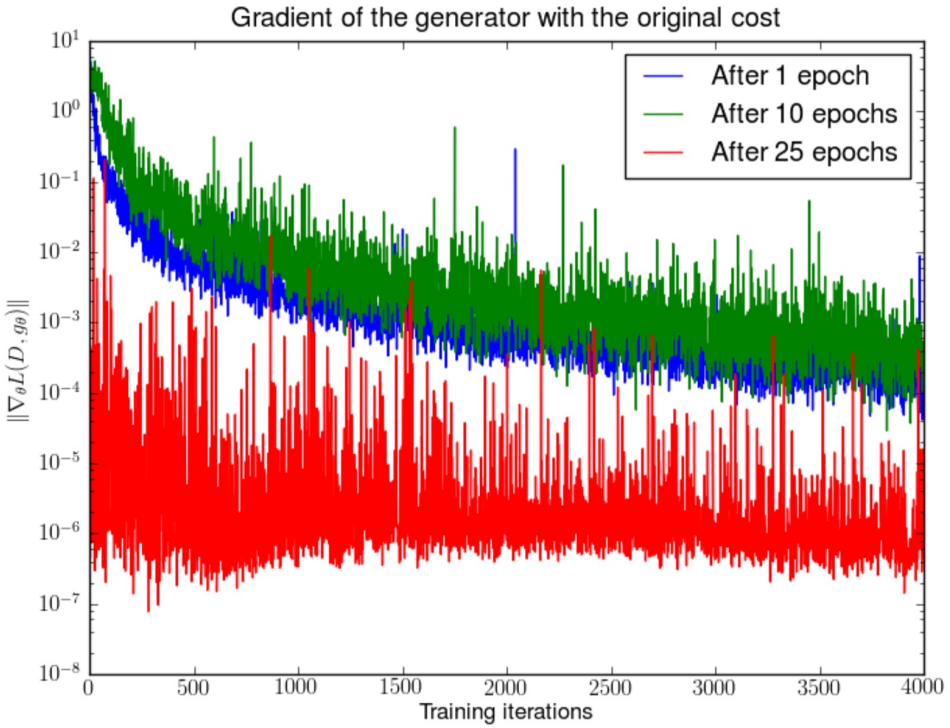


https://mit-6s978.github.io/assets/pdfs/lec4_gan.pdf

DCGAN

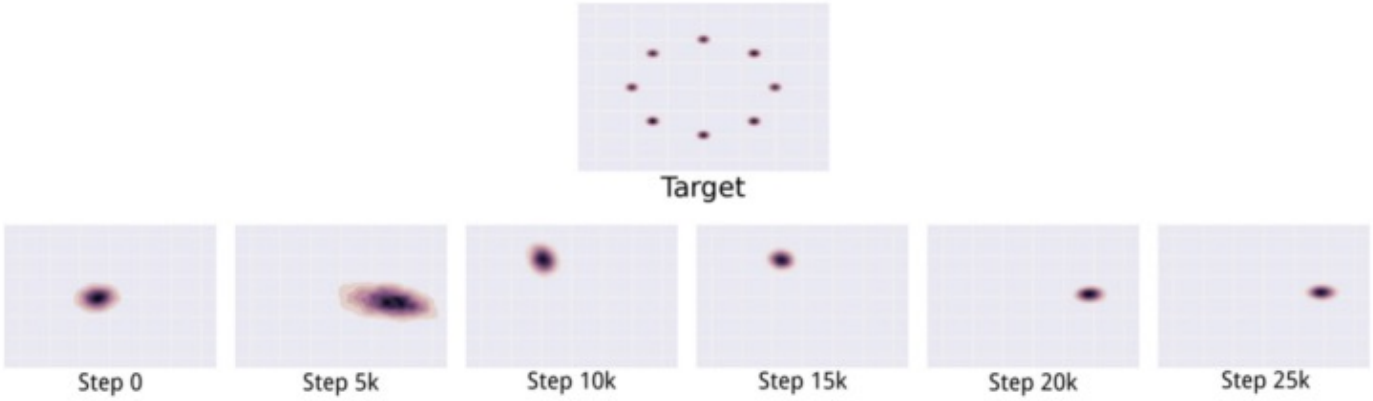
Results from the first GAN model with ConvNets

Vanishing Gradients



https://lilianweng.github.io/posts/2017-08-20-gan/

Mode Collapse



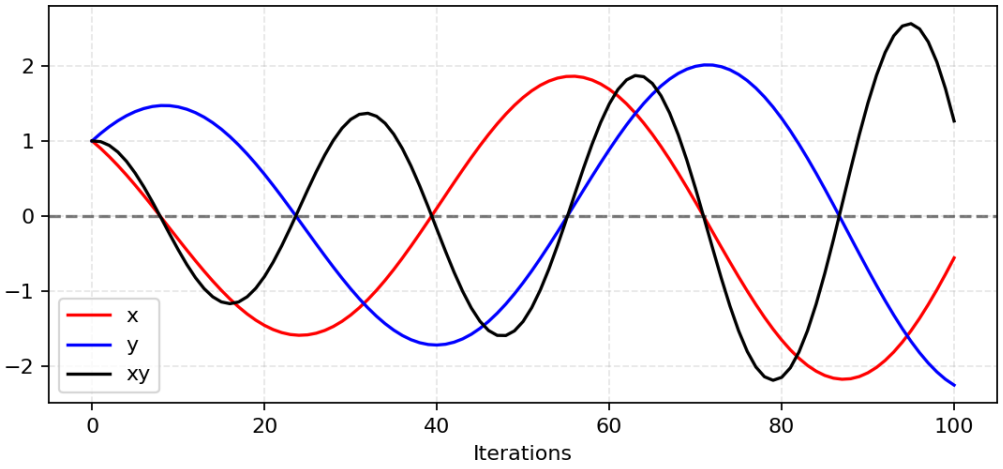
Many modes not captured

Training Instability

Min-max optimization = differential game

Example: finding a Nash equilibrium in a 1D differentiable game

$$\min_x \max_y xy$$

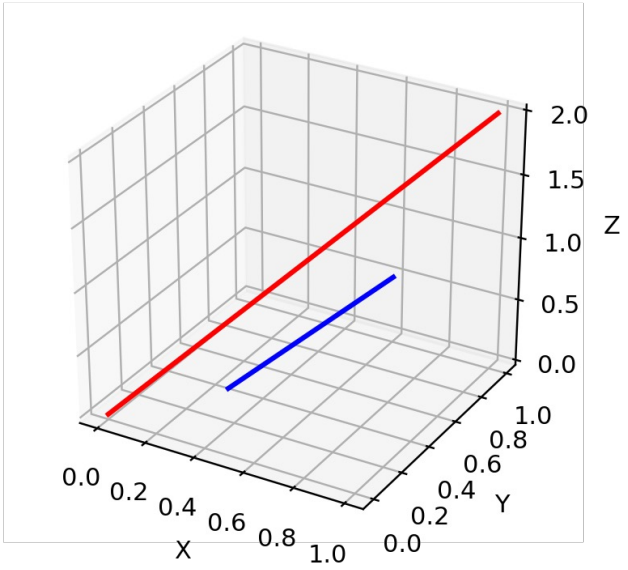


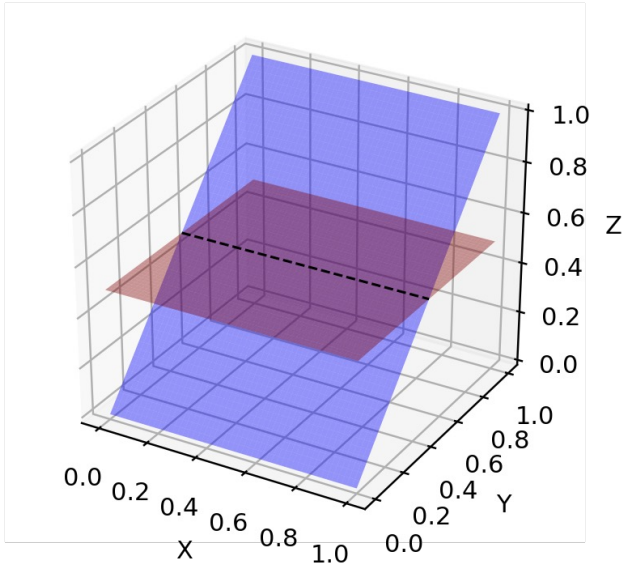
https://lilianweng.github.io/posts/2017-08-20-gan/

Trivial Solutions

Low dimensional support of both data and modeled distributions

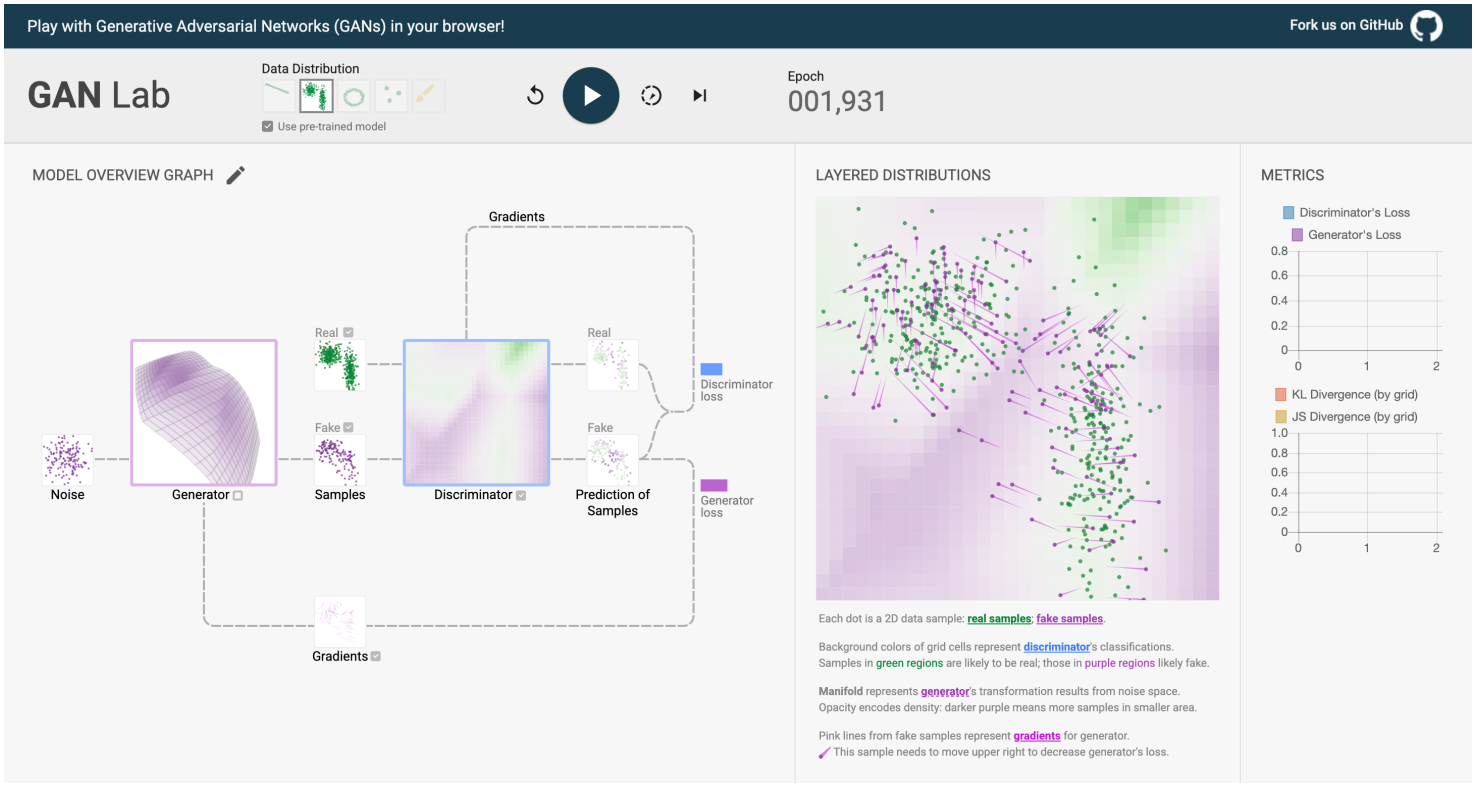
Need careful tuning to avoid trivial discriminator





Visualize GAN Training in Your Browser

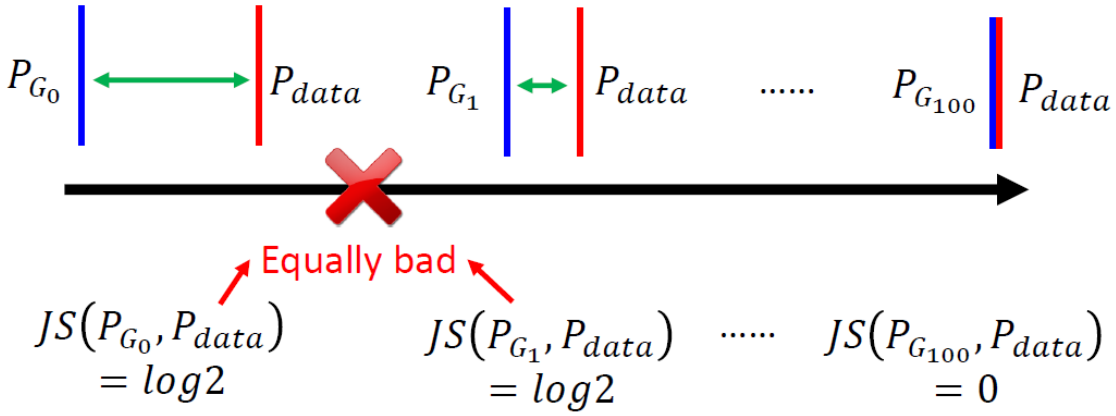
https://poloclub.github.io/ganlab/



GAN Variants

Issue with JS Divergence

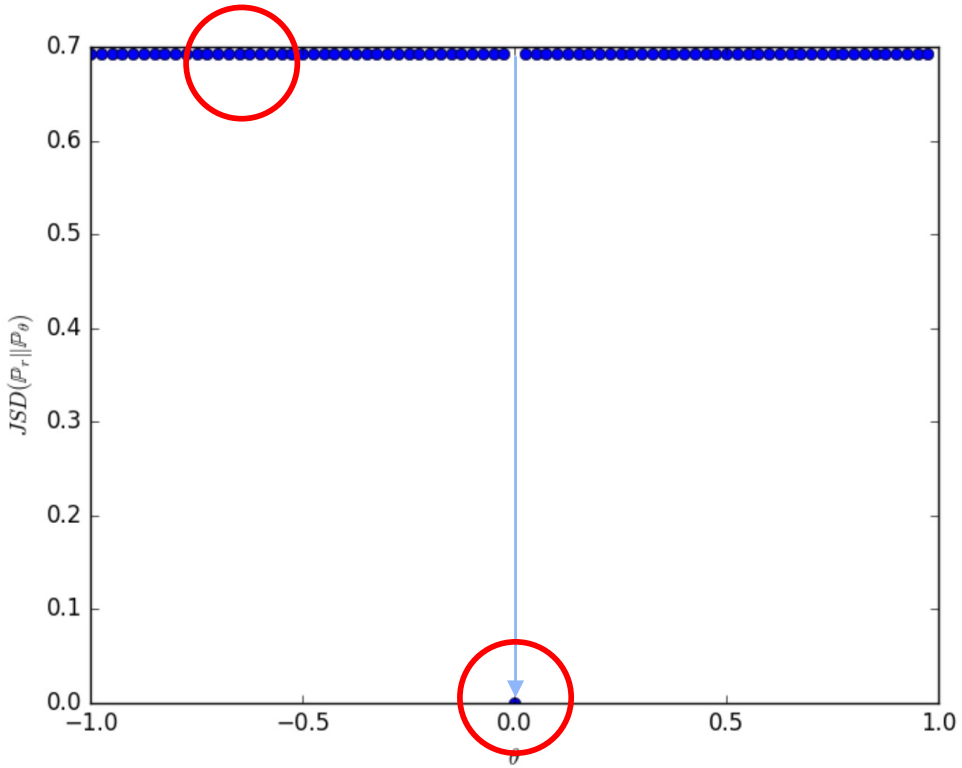
Suppose p_θ and $p_\mathcal{D}$ are parallel lines in 2D
We want to move p_θ parallelly to $p_\mathcal{D}$



JS divergence is log2 if two distributions do not overlap.

Issue with JS Divergence

Zero gradient elsewhere



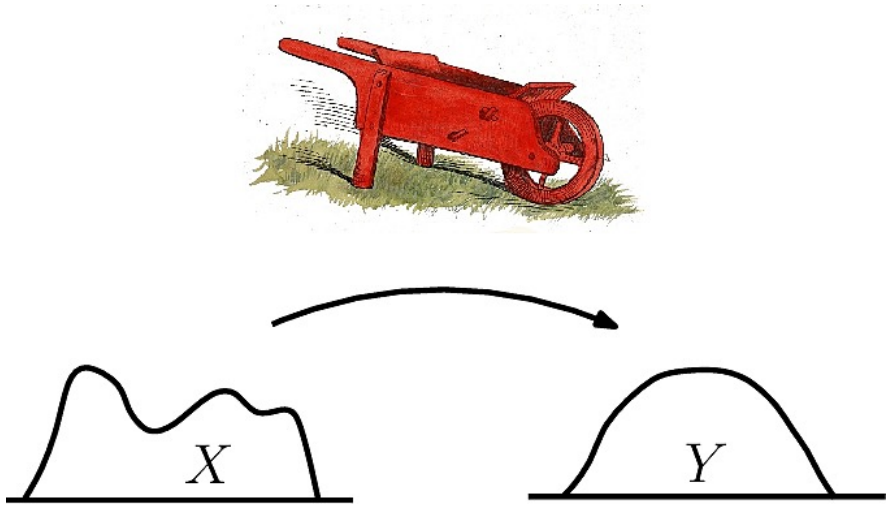
Any other distance function?

Infinite gradient when overlapped

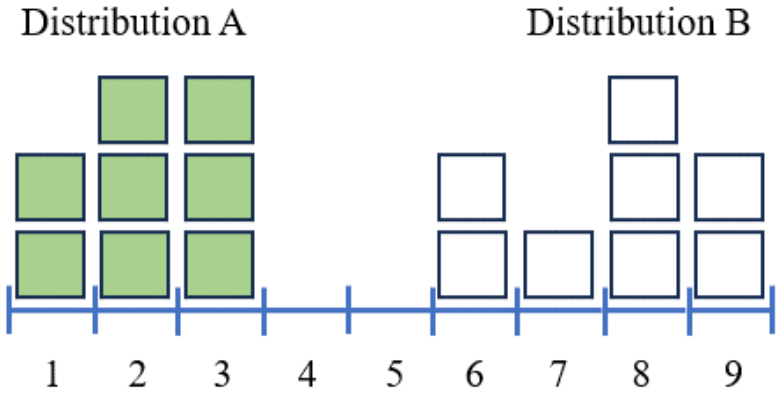
Wasserstein Distance

a.k.a. Earth Mover's Distance

How can you move one pile of earth to the other shape (at different location) with the least cost?



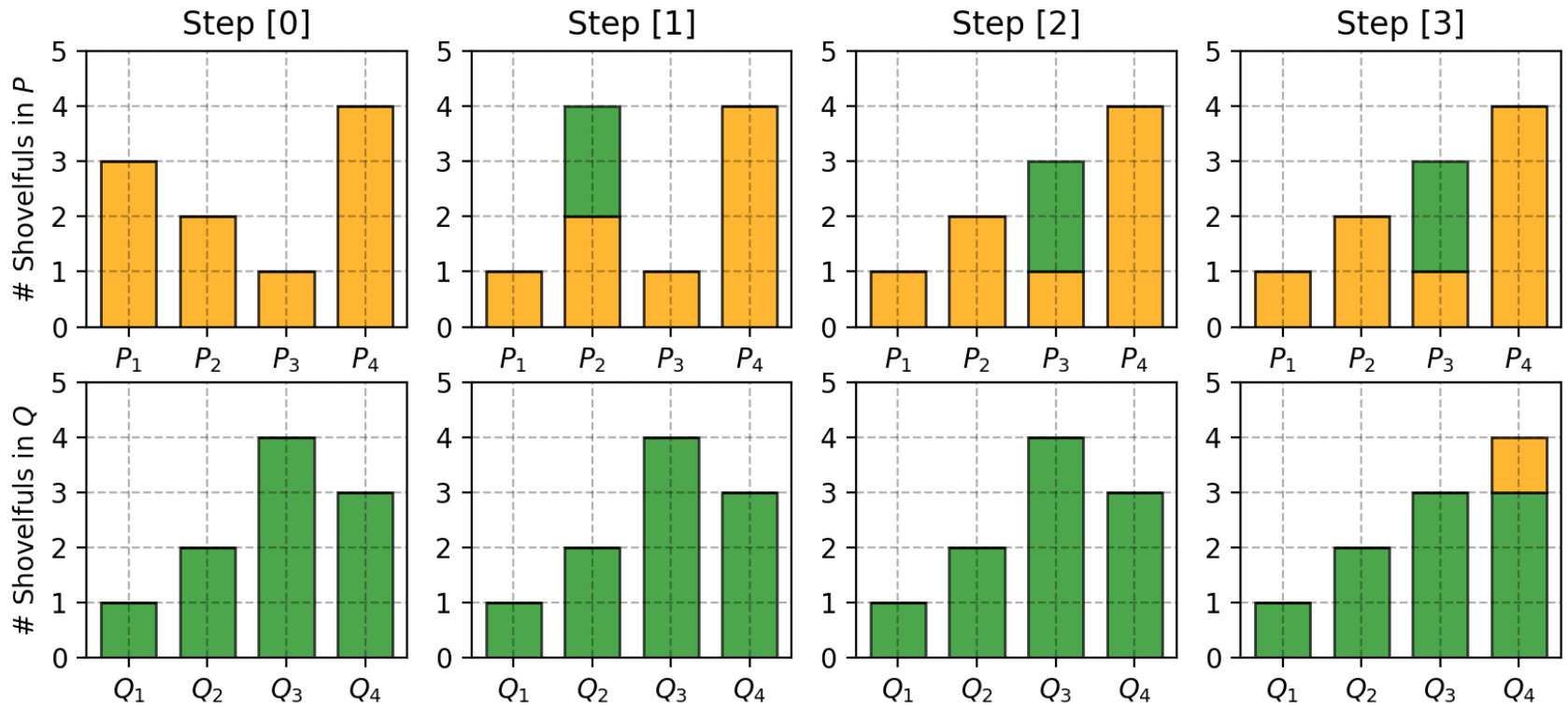
Wasserstein Distance



Work

https://towardsdatascience.com/comparison-of-distributions-with-earth-movers-distance-71f714440923?gi=3b6f5b39cceb

Wasserstein Distance



https://lilianweng.github.io/posts/2017-08-20-gan/

Wasserstein Distance

Formal Definition

$$W_p(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \left(\mathbf{E}_{(x,y)\sim\gamma} d(x,y)^p \right)^{\frac{1}{p}},$$

$\Gamma(\mu, \nu)$	Set of all possible couplings (continuous transportation plans)
$d(x, y)$	Distance function between two points
$\inf_{\gamma \in \Gamma(\mu,\nu)}$	Find the cheapest transportation plan

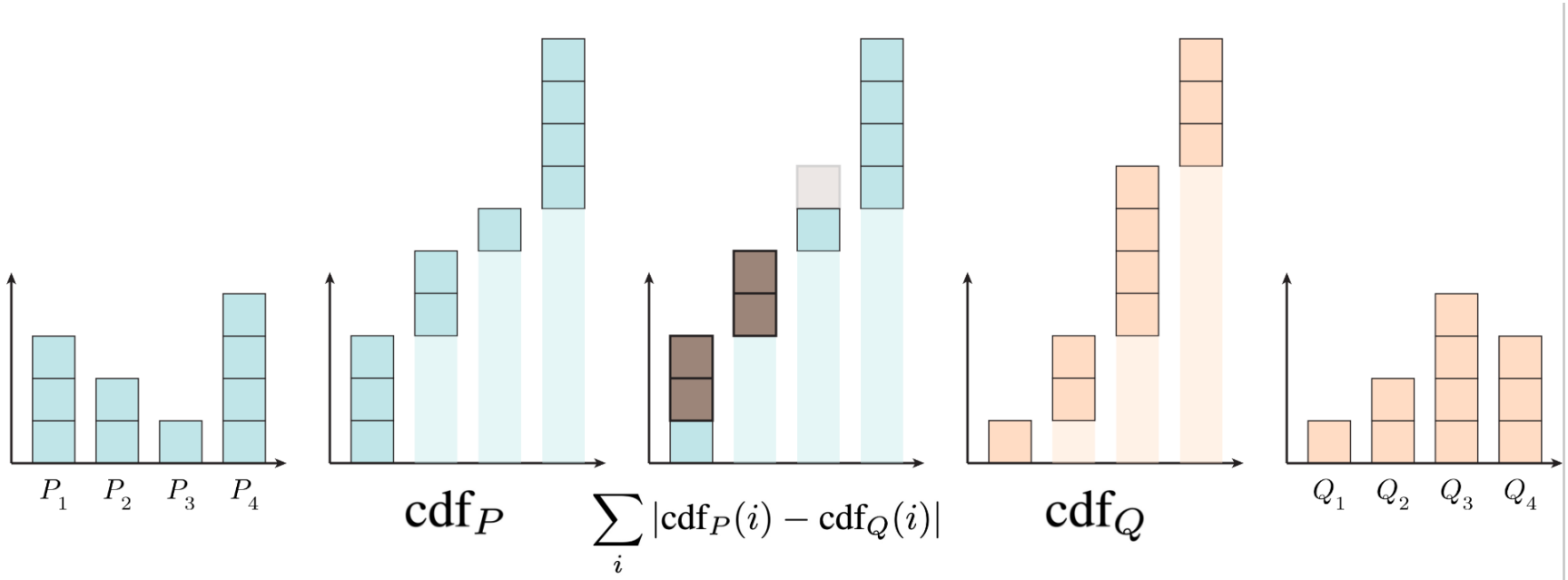
Wasserstein Distance in 1D

If we have the CDF of distributions for 1D distributions p and q

1-Wasserstein distance:

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

Wasserstein Distance in 1D



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

https://mit-6s978.github.io/assets/pdfs/lec4_gan.pdf

Wasserstein GAN

In a nutshell, to replace JS divergence with Wasserstein distance

Do not need to exactly compute Wasserstein distance

Critic update:

Logits for binary classification

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

Any real value

W-GAN $$\max_{w\in\mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z)]$$

Same formulation for discriminator

Key Difference: (asymmetric) logits to symmetric critic values

Wasserstein GAN

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.

  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$

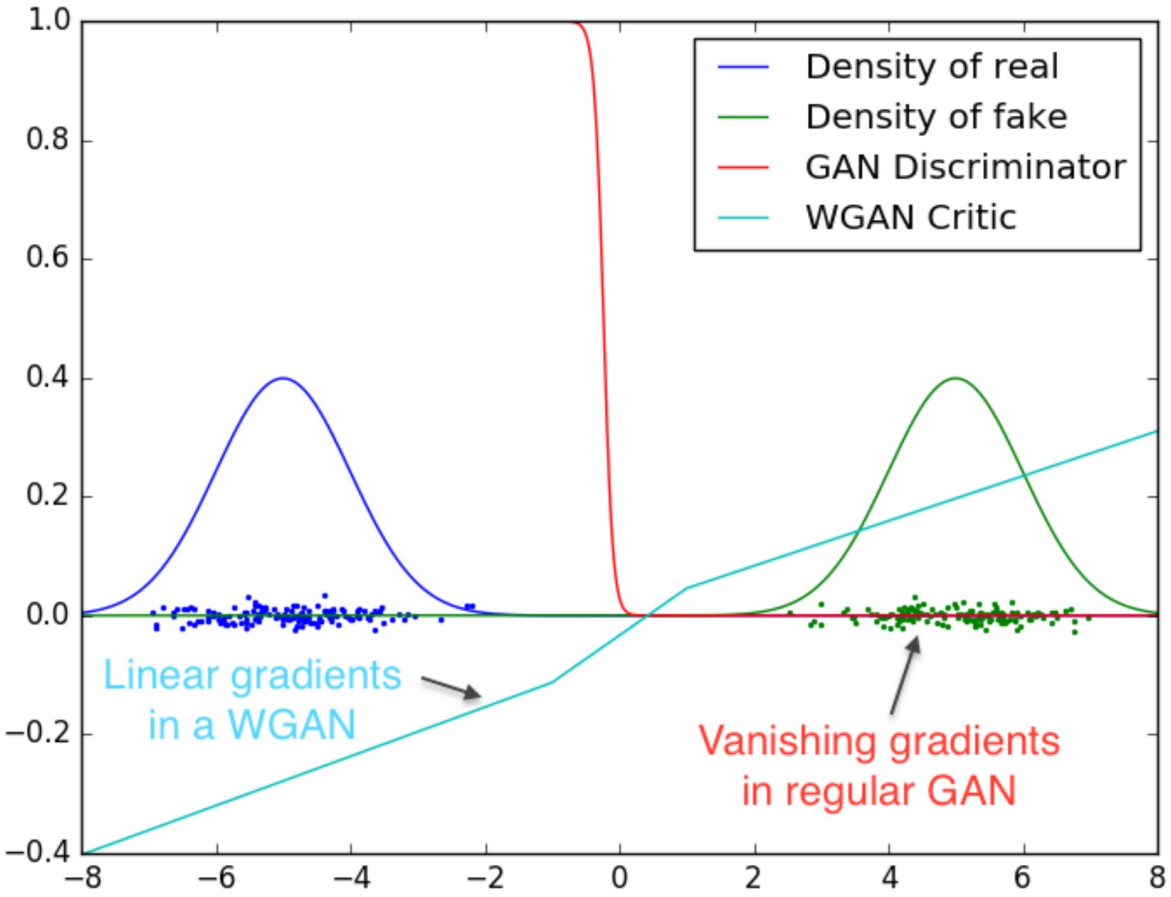
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$

end while

Critic Update

Generator Update

W-GAN vs. Vanilla GAN



WGAN-GP

Use gradient penalty instead of weight clipping to stabilize training

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

f-GAN

Instead of JS divergence, use the more general f-divergence

Name	$D_f(P\|Q)$	Generator $f(u)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x$	$u \log u$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)} \, \mathrm{d}x$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} \, \mathrm{d}x$	$(u-1)^2$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \, \mathrm{d}x$	$\left(\sqrt{u}-1\right)^2$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \, \mathrm{d}x$	$-(u+1) \log \frac{1+u}{2} + u \log u$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \, \mathrm{d}x - \log(4)$	$u \log u - (u+1) \log(u+1)$

f-GAN

Instead of JS divergence, use the more general f-divergence

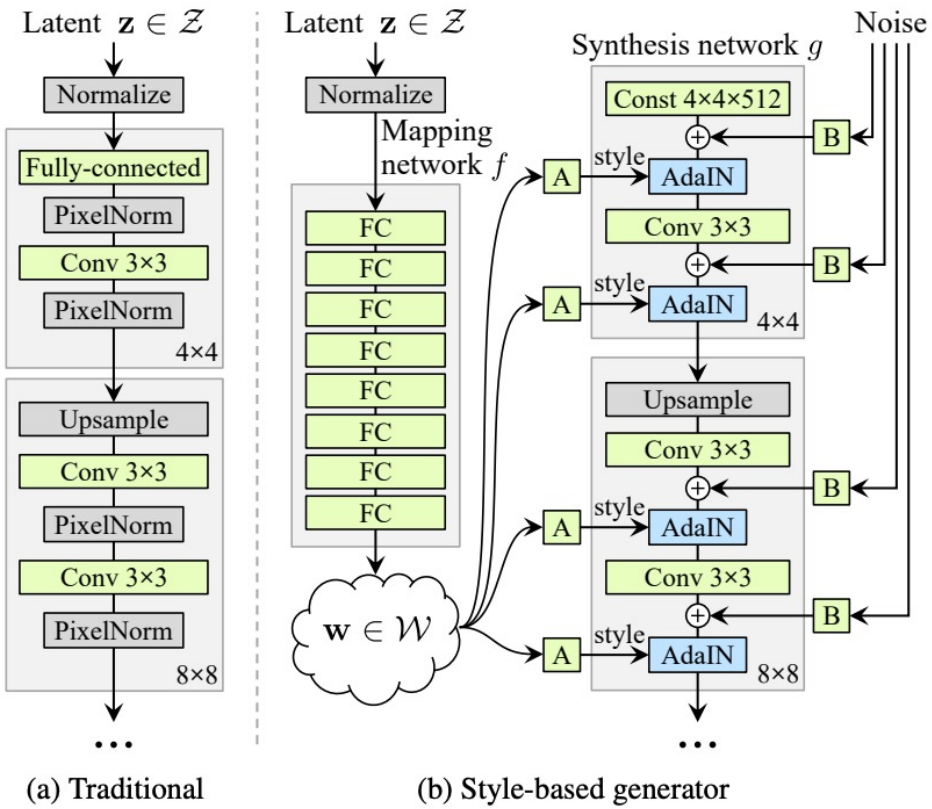
$$F(\theta, \omega) = \underbrace{\mathbb{E}_{x \sim P}\left[g_f(V_\omega(x))\right]}_{\text{Real}} + \underbrace{\mathbb{E}_{x \sim Q_\theta}\left[-f^*(g_f(V_\omega(x)))\right]}_{\text{Fake}}$$

Name	Output activation g_f	dom_{f^*}	Conjugate $f^*(t)$	$f'(1)$
Kullback-Leibler (KL)	v	$\mathbb{R}$	$\exp(t-1)$	1
Reverse KL	$-\exp(-v)$	$\mathbb{R}_-$	$-1-\log(-t)$	-1
Pearson χ^2	v	$\mathbb{R}$	$\frac{1}{4}t^2+t$	0
Squared Hellinger	$1-\exp(-v)$	$t<1$	$\frac{t}{1-t}$	0
Jensen-Shannon	$\log(2)-\log(1+\exp(-v))$	$t<\log(2)$	$-\log(2-\exp(t))$	0
GAN	$-\log(1+\exp(-v))$	$\mathbb{R}_-$	$-\log(1-\exp(t))$	$-\log(2)$

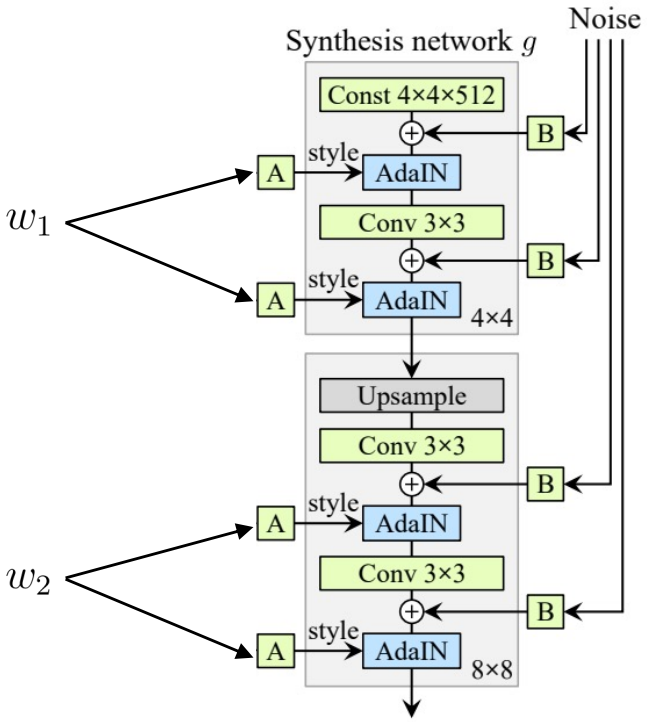
StyleGAN

- Latent as style condition
- Multi-resolution noise
- Blending with AdaIN (module for style transfer)

$$\text{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$



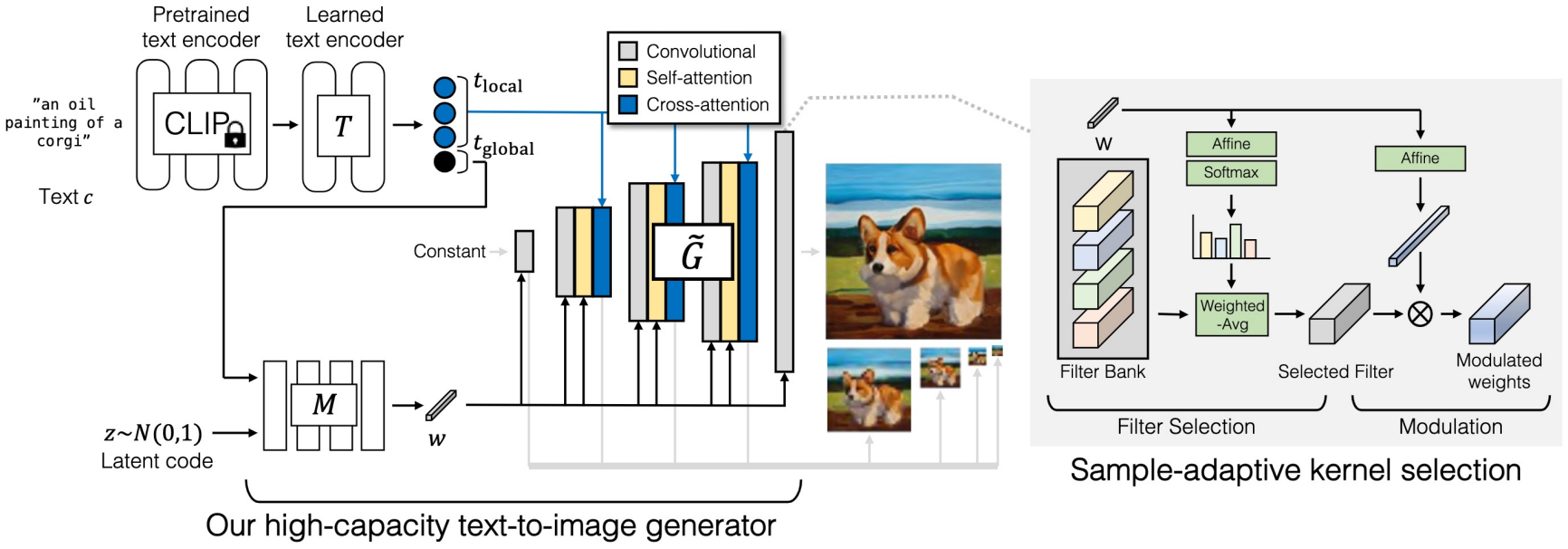
Style Mixing with StyleGAN



StyleGAN Families

GigaGAN

Heavily engineered architecture and losses for large-scale text-conditioned models



Latest: Relativistic GANs + Regularizations

$$\mathcal{L}(\theta, \psi) = \mathbb{E}_{\substack{z \sim p_z \\ x \sim p_{\mathcal{D}}}} \left[f\left(D_\psi(G_\theta(z)) - D_\psi(x) \right) \right]$$

(typically with a saturating f function)

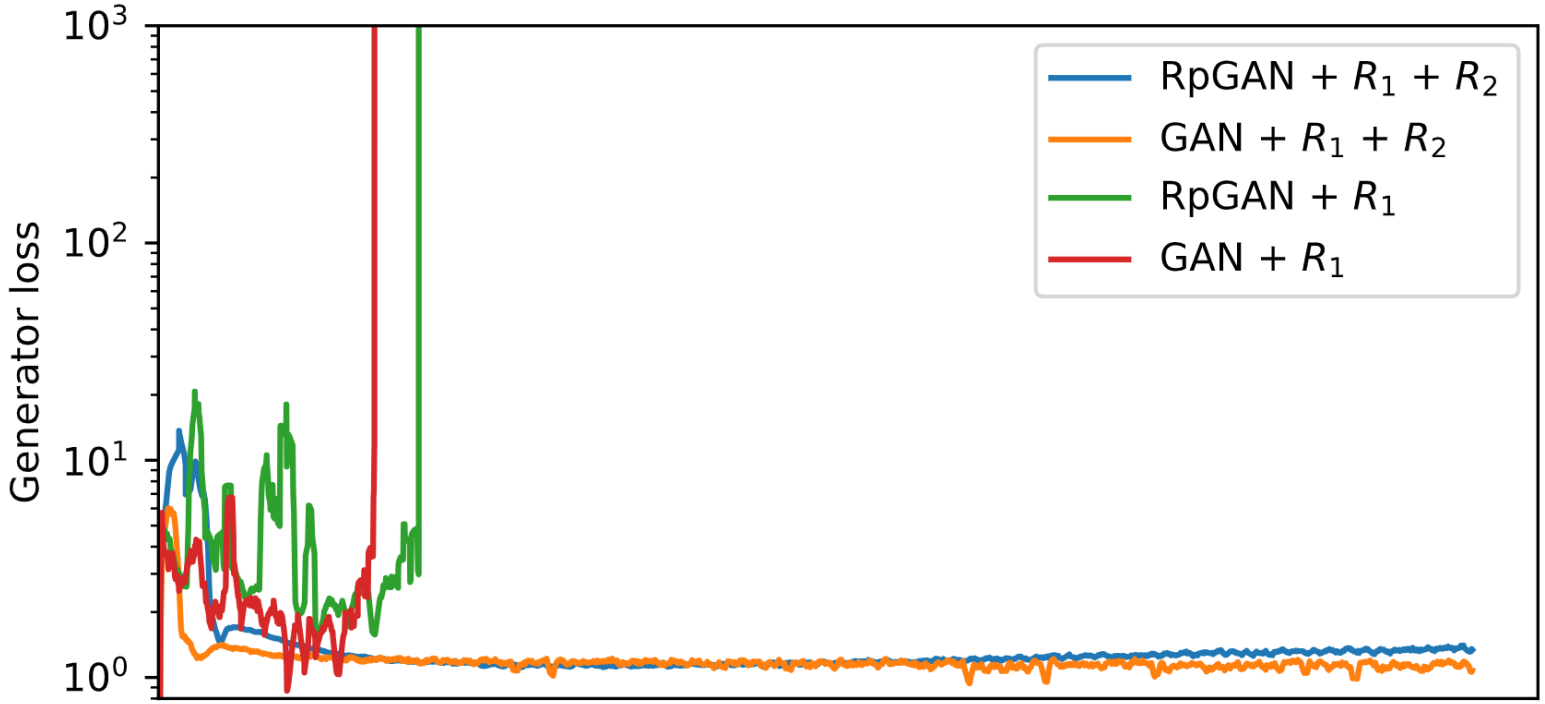
$$R_1(\psi) = \frac{\gamma}{2} \mathbb{E}_{x \sim p_{\mathcal{D}}} \left[\|\nabla_x D_\psi\|^2 \right]$$

$$R_2(\theta, \psi) = \frac{\gamma}{2} \mathbb{E}_{x \sim p_\theta} \left[\|\nabla_x D_\psi\|^2 \right]$$

Zero-centered Gradient Penalty

https://arxiv.org/abs/2501.05441

Latest: Relativistic GANs + Regularizations



https://arxiv.org/abs/2501.05441

Latest: Relativistic GANs + Regularizations

Competitive performance

Remains to be verified on larger models

Model	NFE↓	FID↓
BigGAN-deep [3]	1	4.06
DDPM [20]	250	11.0
DDIM [76]	50	13.7
ADM [7]	§250	2.91
EDM [33]	79	2.23
CT [79]	2	11.1
CD [79]	3	4.32
iCT-deep [77]	2	2.77
DMD [96]	1	2.62
Ours—Config E	1	2.09
With ImageNet feature leakage [41]:		
StyleGAN-XL* [69]	1	1.52

Table 8: ImageNet-64. §deterministic sampling.

https://arxiv.org/abs/2501.05441

Adversary as a Loss

Adversarial loss

Not necessary to have a generator g(z)

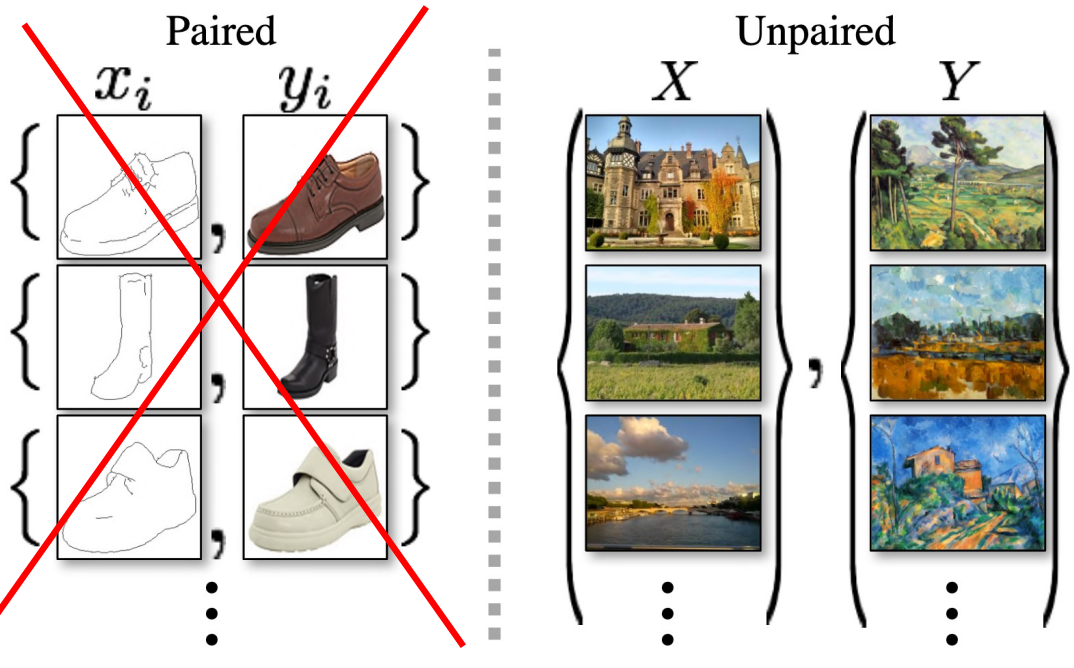
- Can be used with VAEs, diffusion models and other generative models

Can use discriminator to refine results

- Supplement to reconstruction losses for better visual quality
- Leverage implicit priors in neural nets (and adversarial training)

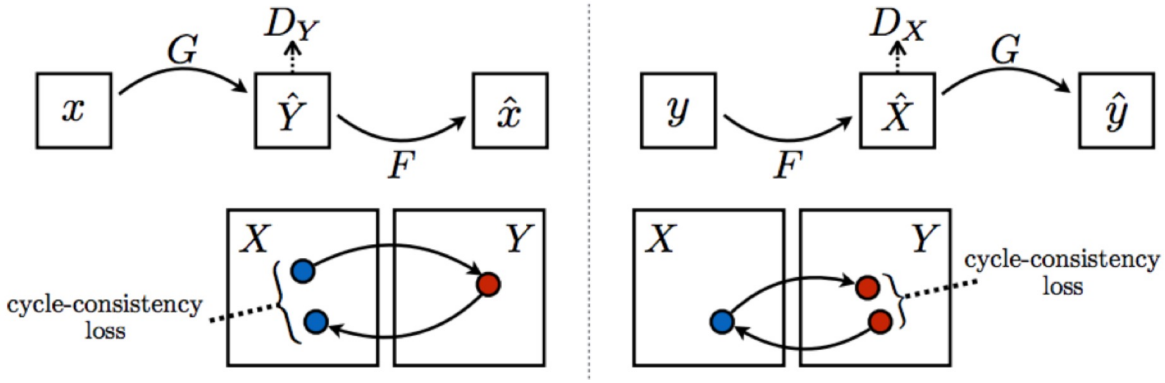
CycleGAN

How to learn style transfer from unpaired data?



https://arxiv.org/abs/1703.10593

CycleGAN



$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

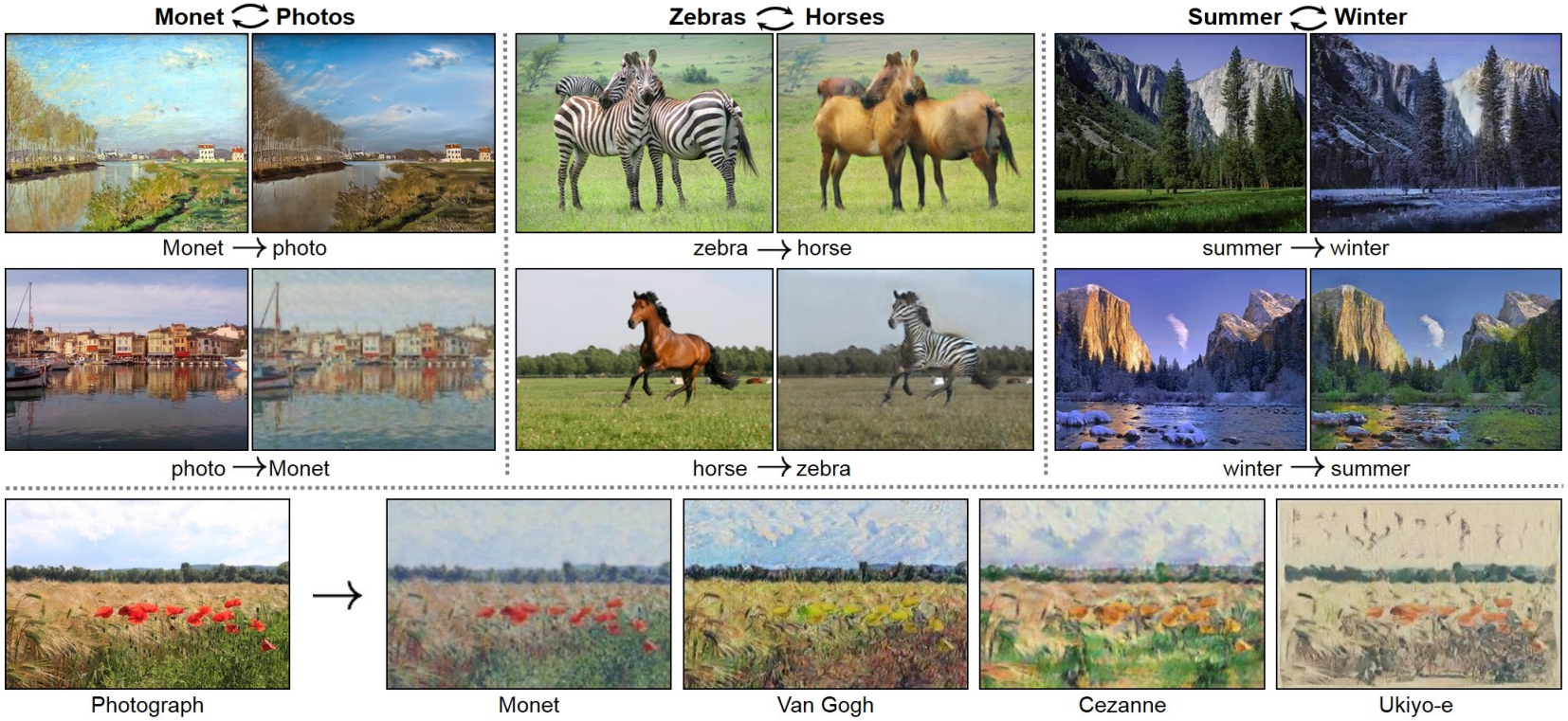
Cycle Consistency

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

Full loss = GAN losses + Cycle consistency

https://junyanz.github.io/CycleGAN/

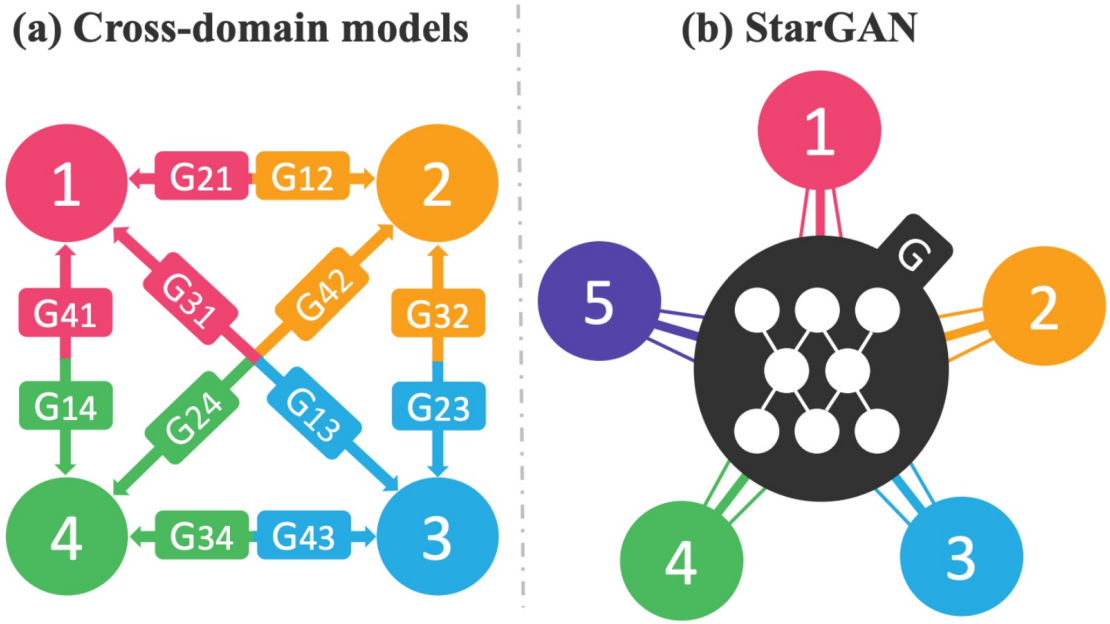
CycleGAN



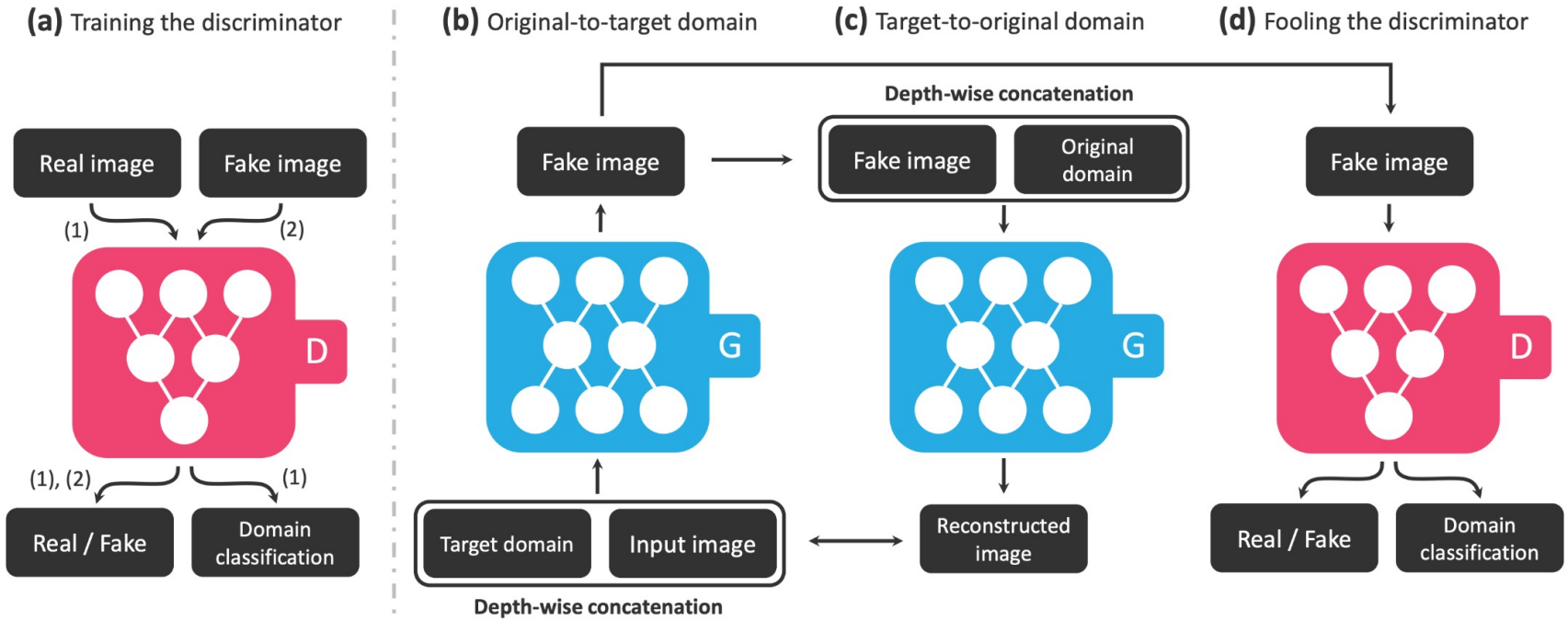
https://junyanz.github.io/CycleGAN/

StarGAN

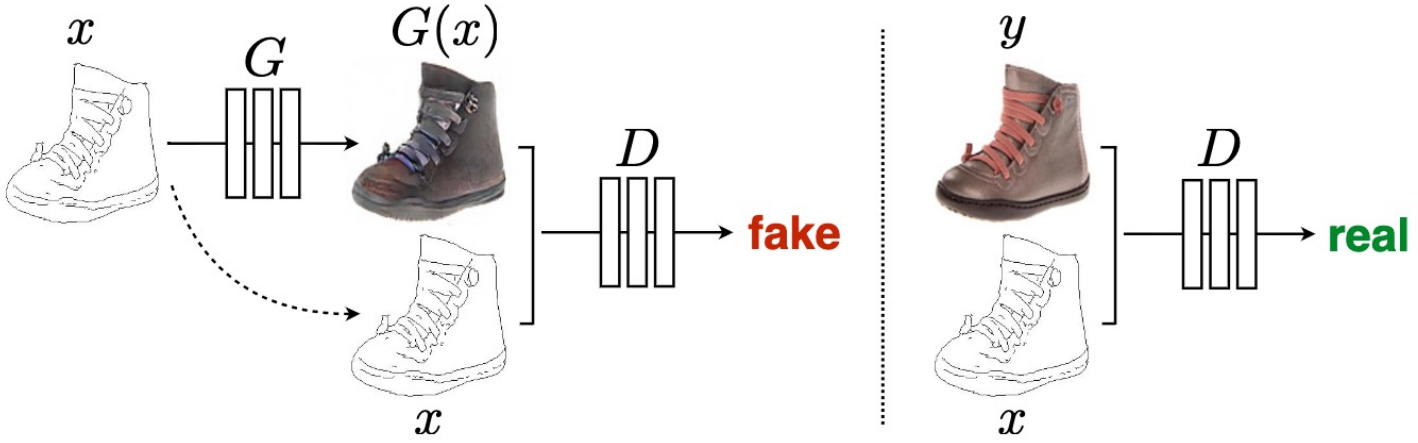
More than one domain?



StarGAN


(a) Training the discriminator
(b) Original-to-target domain
(c) Target-to-original domain
(d) Fooling the discriminator
Real image
Fake image
(1)
(2)
D
(1), (2)
(1)
Real / Fake
Domain classification
Depth-wise concatenation
Fake image
G
Target domain
Input image
Depth-wise concatenation
Fake image
Original domain
G
Reconstructed image
Fake image
D
Real / Fake
Domain classification

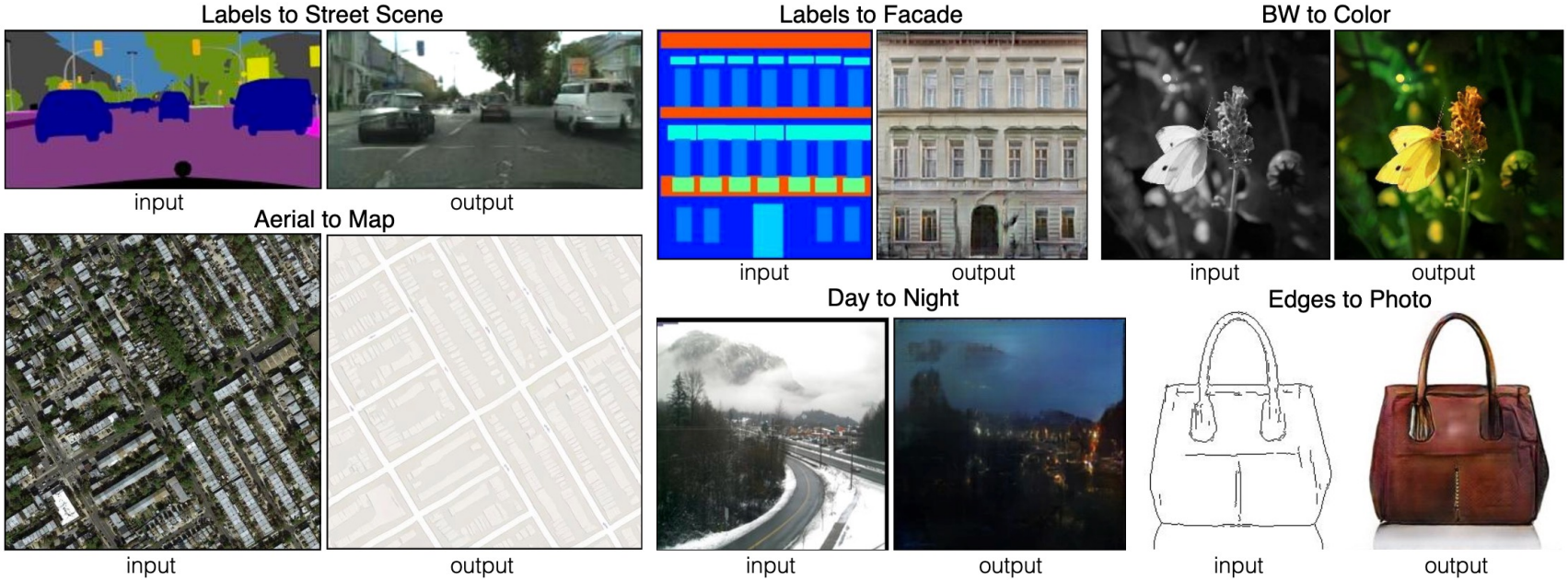
Pix2Pix



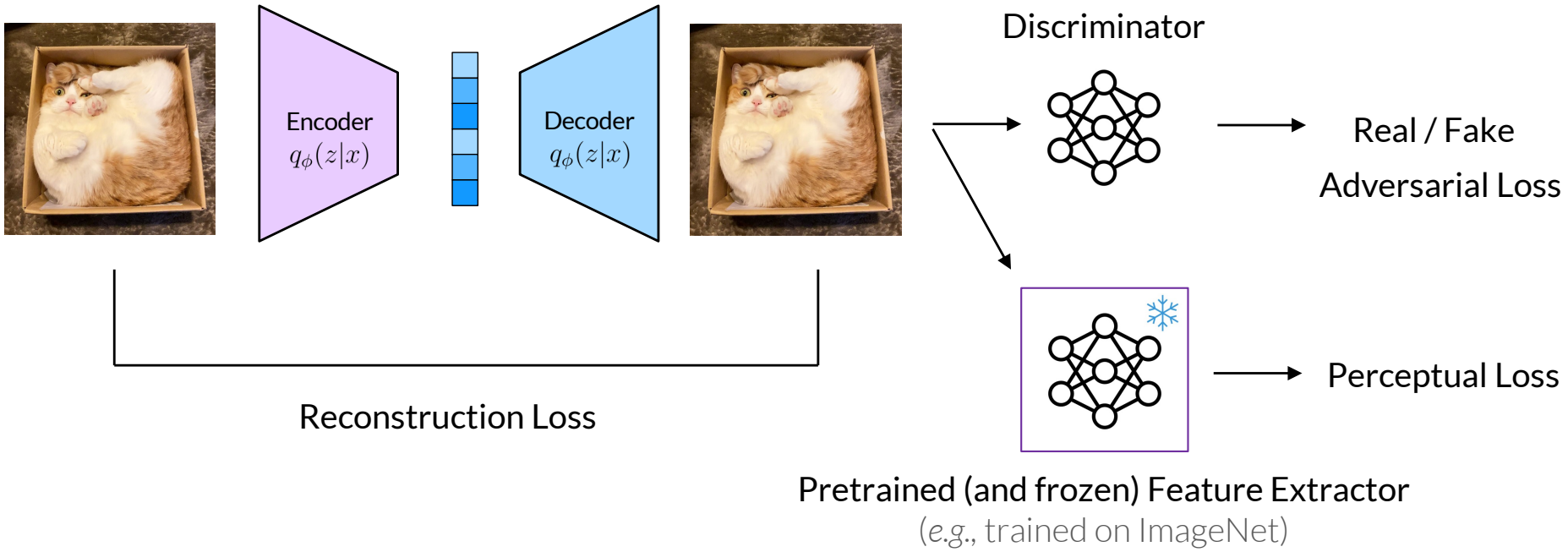
$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$

Pix2Pix


Labels to Street Scene
input
output
Labels to Facade
input
output
BW to Color
input
output
Aerial to Map
input
output
Day to Night
input
output
Edges to Photo
input
output

VAE + GAN + Perceptual Loss



Used for modern image encoders / tokenizers for images
(StableDiffusion, Sora, etc.)

Summary

- Vanilla GAN
- Variants (W-GAN, etc.)
- Adversary as a Loss