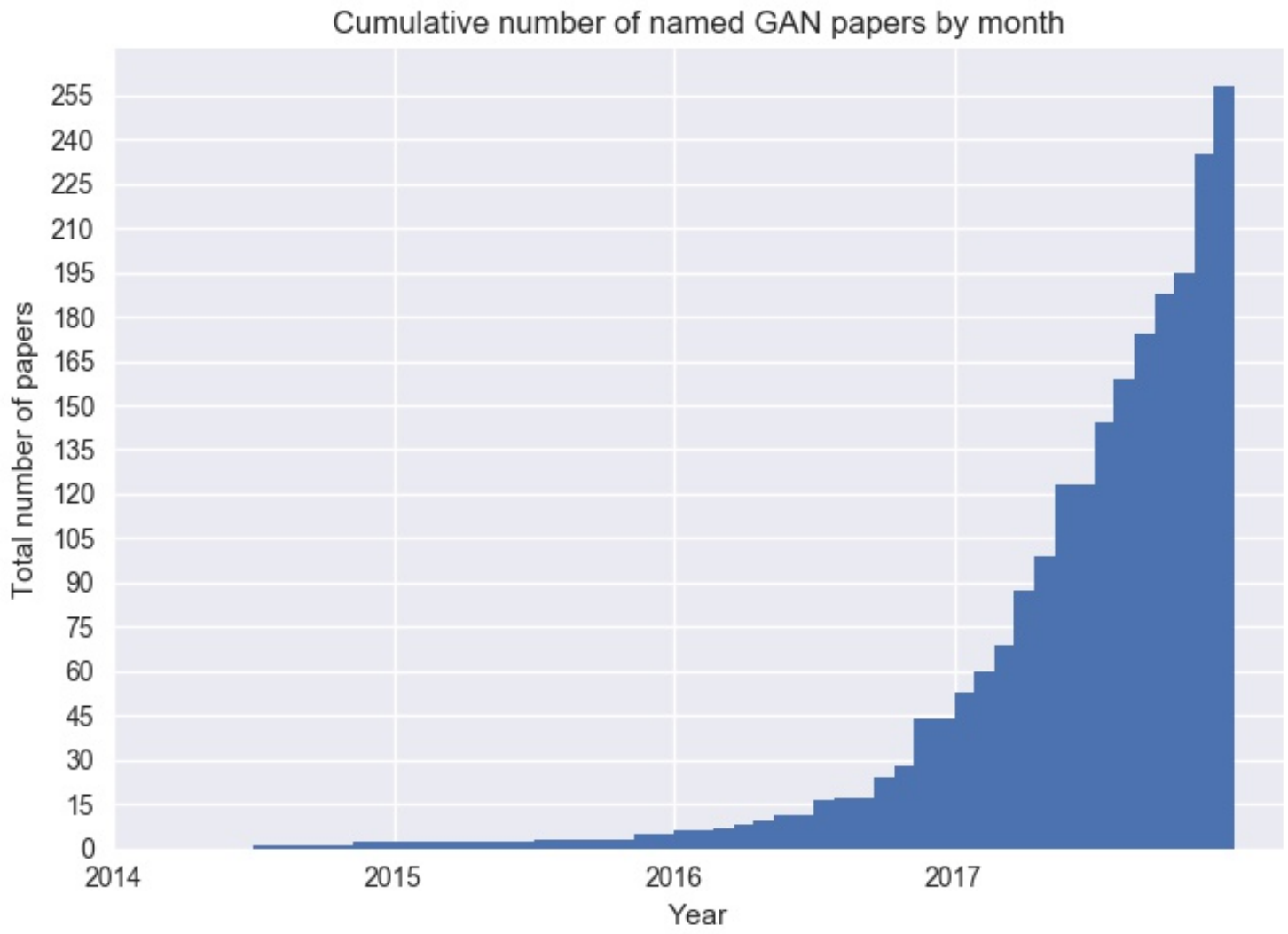
All Kinds of GAN

Hung-yi Lee



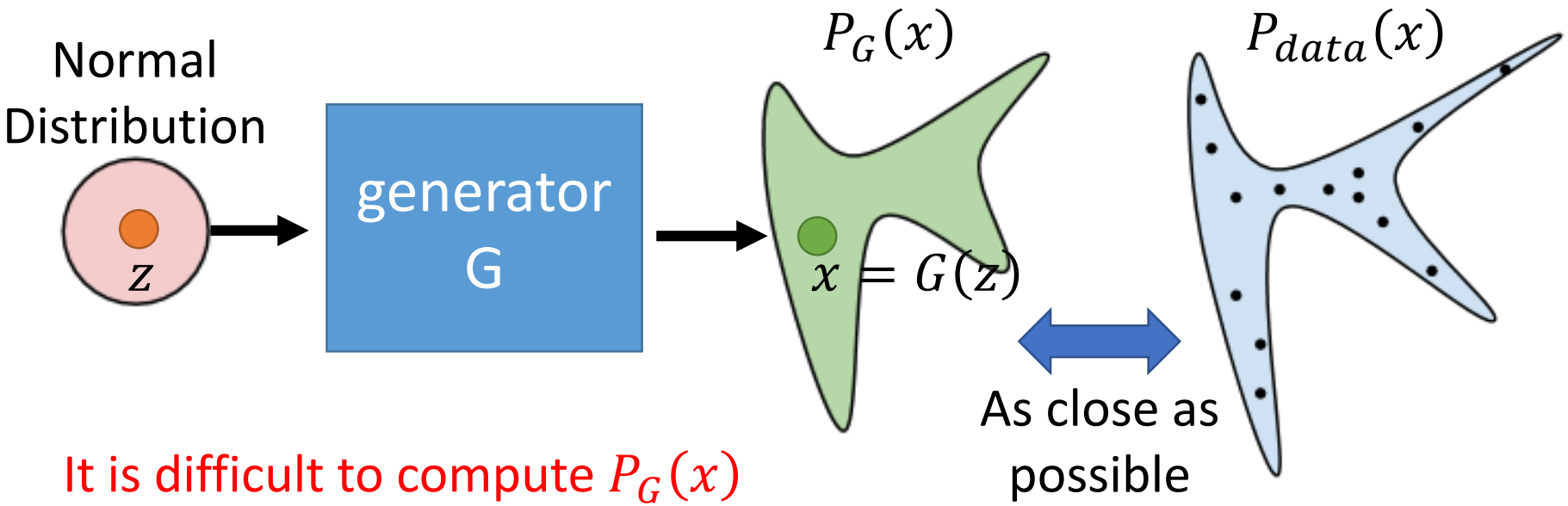
Zoo of GAN: https://github.com/hindupuravinash/the-gan-zoo

General Idea of GAN

Sebastian Nowozin, Botond Cseke, Ryota Tomioka, "***f-GAN***: Training Generative Neural Samplers using Variational Divergence Minimization", NIPS, 2016

Basic Idea of GAN

- A generator G is a network. The network defines a probability distribution.



It is difficult to compute $P_G(x)$

We can only sample from the distribution.

https://blog.openai.com/generative-models/

Basic Idea of GAN

- Generator G

  Hard to learn by maximum likelihood

  - G is a function, input z, output x
  - Given a prior distribution $P_{prior}(z)$, a probability distribution $P_G(x)$ is defined by function G
- Discriminator D
  - D is a function, input x, output scalar
  - Evaluate the "difference" between $P_G(x)$ and $P_{data}(x)$
- There is a function V(G,D).

$$G^* = arg \min_G \max_D V(G, D)$$

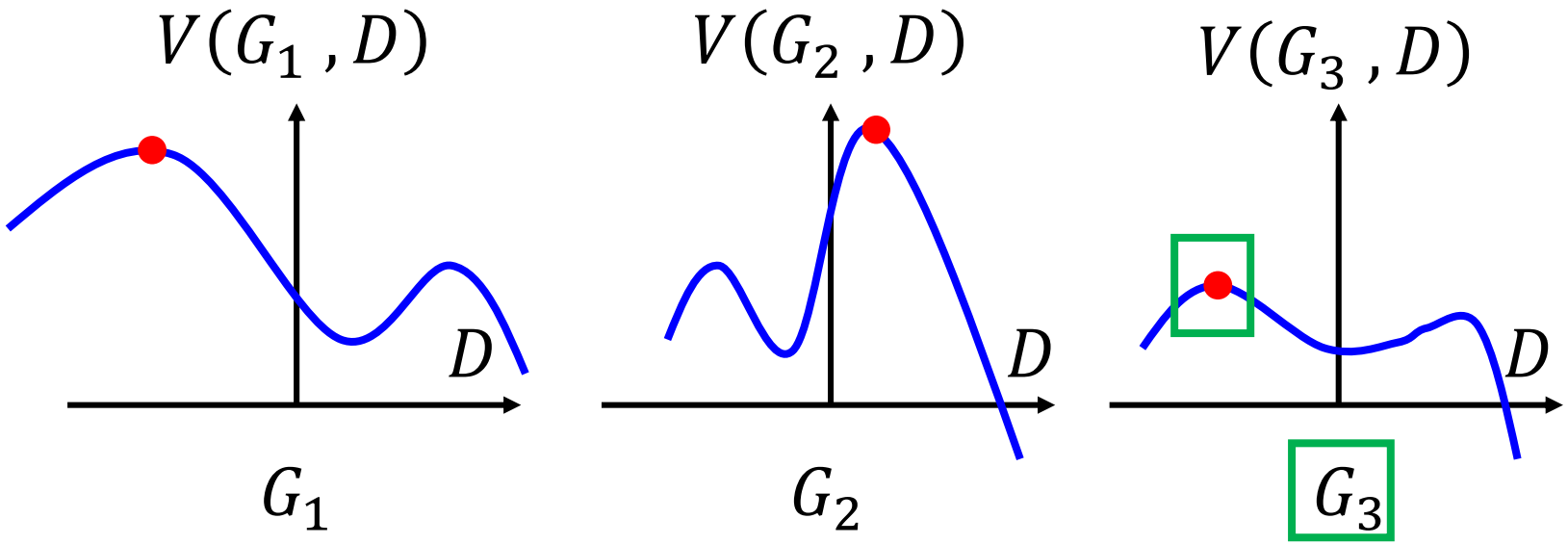
Basic Idea

$$G^* = arg \min_G \max_D V(G,D)$$

$$V = E_{x \sim P_{data}}[logD(x)] + E_{x \sim P_G}[log(1 - D(x))]$$

Given a generator G, $\max_D V(G,D)$ evaluate the JS divergence between P_G and P_{data}

Pick the G defining P_G most similar to P_{data}



Algorithm Initialize θ_d for D and θ_g for G

- In each training iteration:

Learning D

Repeat k times

- Sample m examples $\{x^1, x^2, \dots, x^m\}$ from data distribution $P_{data}(x)$
- Sample m noise samples $\{z^1, z^2, \dots, z^m\}$ from the prior $P_{prior}(z)$
- Obtaining generated data $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$, $\tilde{x}^i = G(z^i)$
- Update discriminator parameters θ_d to maximize
  - $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m} logD(x^i) + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D(\tilde{x}^i)\right)$
  - $\theta_d \leftarrow \theta_d + \eta\nabla\tilde{V}(\theta_d)$

Learning G

Only Once

- Sample another m noise samples $\{z^1, z^2, \dots, z^m\}$ from the prior $P_{prior}(z)$
- Update generator parameters θ_g to minimize
  - $\tilde{V} = \cancel{\frac{1}{m}\sum_{i=1}^{m} logD(x^i)} + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D\left(G(z^i)\right)\right)$
  - $\theta_g \leftarrow \theta_g - \eta\nabla\tilde{V}(\theta_g)$

Objective Function for Generator in Real Implementation

$$V = E_{x \sim P_{data}}[logD(x)]$$

$$+E_{x \sim P_G}[log(1 - D(x))]$$

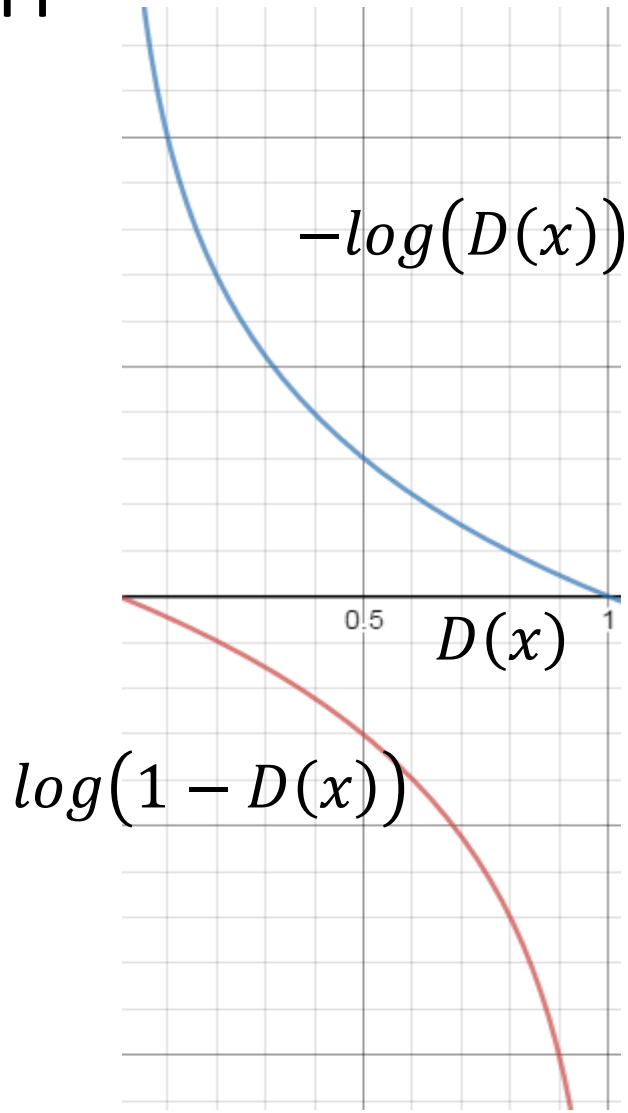
Slow at the beginning

Minimax GAN (MMGAN)

$$V = E_{x \sim P_G}[-log(D(x))]$$

Real implementation:
label x from P_G as positive

Non-saturating GAN (NSGAN)



f-divergence

P and Q are two distributions. $p(x)$ and $q(x)$ are the probability of sampling x.

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

f is convex

f(1) = 0

$D_f(P||Q)$ evaluates the difference of P and Q

If $p(x) = q(x)$ for all x

smallest $\quad D_f(P||Q) = \int_x q(x) f\left(\underbrace{\frac{p(x)}{q(x)}}_{=1}\right) dx = \underbrace{0}_{=0}$

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

Because f is convex

$$\geq f\left(\int_x \cancel{q(x)} \frac{p(x)}{\cancel{q(x)}} dx\right)$$

$$= f(1) = 0$$

If P and Q are the same distributions,

$D_f(P||Q)$ has the smallest value, which is 0

f-divergence

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

f is convex

f(1) = 0

$f(x) = xlogx$

KL

$$D_f(P||Q) = \int_x q(x)\frac{p(x)}{q(x)} log\left(\frac{p(x)}{q(x)}\right) dx = \int_x p(x) log\left(\frac{p(x)}{q(x)}\right) dx$$

$f(x) = -logx$

Reverse KL

$$D_f(P||Q) = \int_x q(x)\left(-log\left(\frac{p(x)}{q(x)}\right)\right) dx = \int_x q(x) log\left(\frac{q(x)}{p(x)}\right) dx$$

$f(x) = (x-1)^2$

Chi Square

$$D_f(P||Q) = \int_x q(x)\left(\frac{p(x)}{q(x)} - 1\right)^2 dx = \int_x \frac{\left(p(x) - q(x)\right)^2}{q(x)} dx$$

Fenchel Conjugate

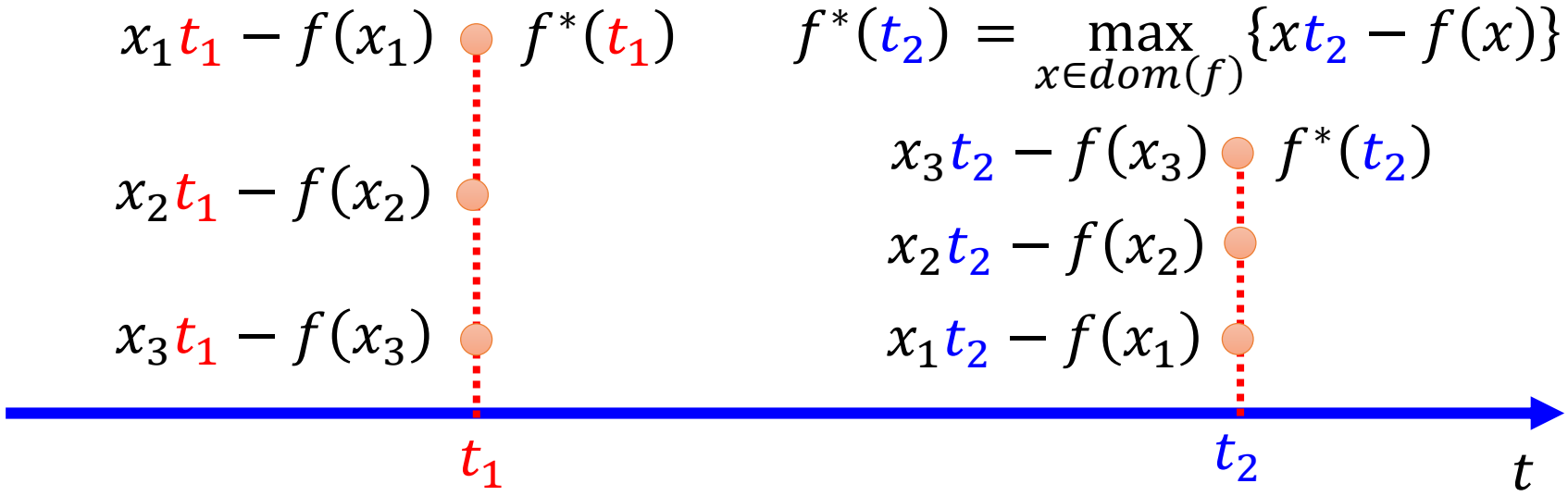
$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

f is convex, f(1) = 0

- Every convex function f has a conjugate function f*

$$f^*(t) = \max_{x \in dom(f)} \{xt - f(x)\}$$

$$f^*(t_1) = \max_{x \in dom(f)} \{xt_1 - f(x)\}$$



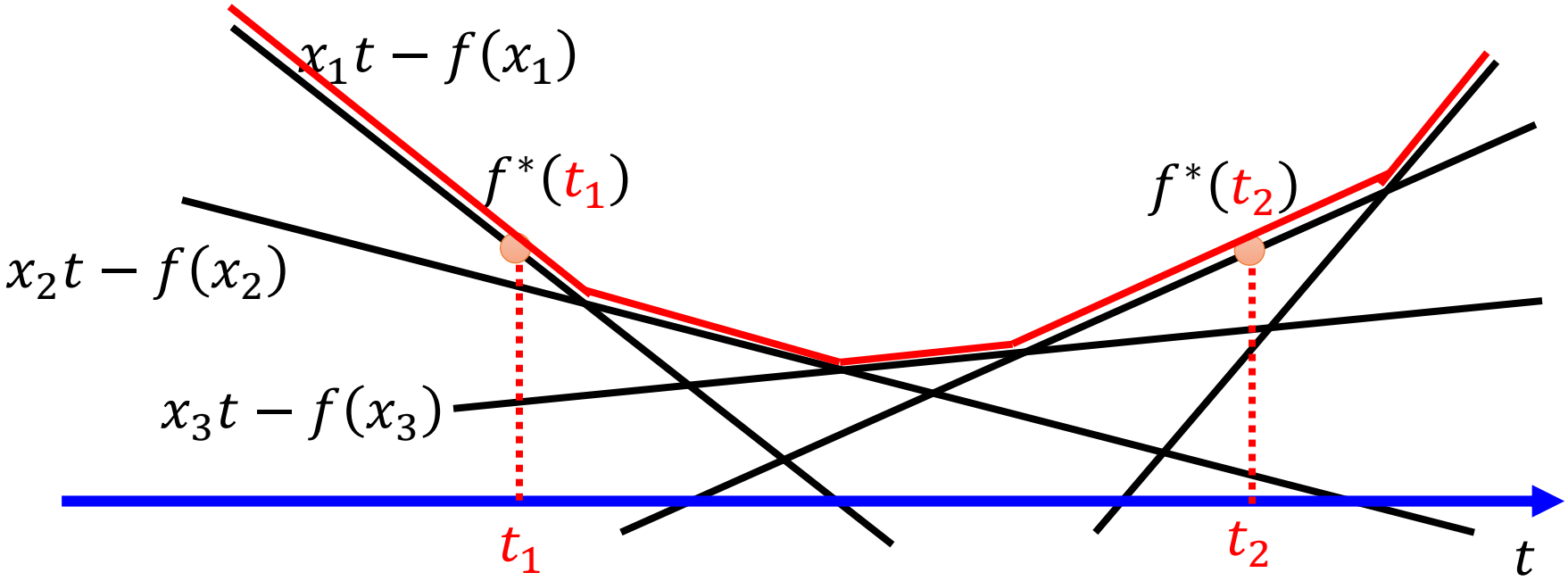
Fenchel Conjugate

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

f is convex, f(1) = 0

- Every convex function f has a conjugate function f*

$$f^*(t) = \max_{x \in dom(f)} \{xt - f(x)\}$$



Connection with GAN

$$f^*(t) = \max_{x \in dom(f)} \{xt - f(x)\} \longleftrightarrow f(x) = \max_{t \in dom(f^*)} \{xt - f^*(t)\}$$

$\frac{p(x)}{q(x)}$ $\frac{p(x)}{q(x)}$

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

$$= \int_x q(x) \left(\max_{t \in dom(f^*)} \left\{ \frac{p(x)}{q(x)} t - f^*(t) \right\} \right) dx$$

$$\approx \max_D \int_x p(x)D(x)dx - \int_x q(x) f^*\big(D(x)\big)dx$$

D is a function whose input is x, and output is t

$$D_f(P||Q) \geq \int_x q(x) \left(\frac{p(x)}{q(x)} D(x) - f^*\big(D(x)\big) \right) dx$$

$$= \int_x p(x)D(x)dx - \int_x q(x) f^*\big(D(x)\big)dx$$

Connection with GAN

$$D_f(P||Q) \approx \max_D \int_x p(x)D(x)dx - \int_x q(x)f^*\big(D(x)\big)dx$$

$$= \max_D \{E_{x\sim P}[D(x)] - E_{x\sim Q}[f^*\big(D(x)\big)]\}$$

Samples from P Samples from Q

$$D_f(P_{data}||P_G) = \max_D \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[f^*\big(D(x)\big)]\}$$

$$G^* = arg\min_G D_f(P_{data}||P_G)$$

Original GAN has different V(G,D)

$$= arg\min_G \max_D \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[f^*\big(D(x)\big)]\}$$

$$= arg\min_G \max_D V(G,D)$$ familiar? ☺

$$D_f(P_{data}||P_G) = \max_{\mathrm{D}}\{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[f^*(D(x))]\}$$

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

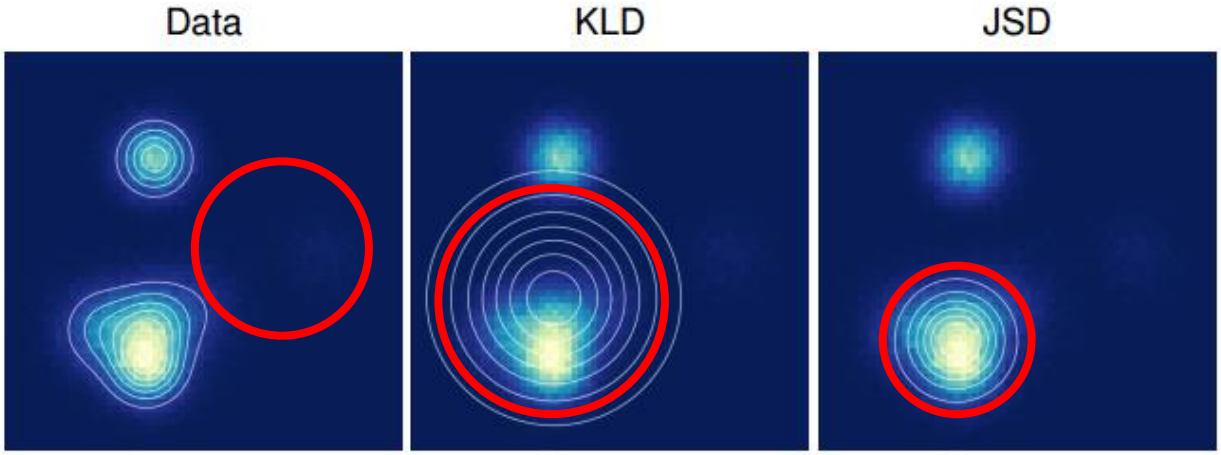
Name	Conjugate $f^*(t)$
Total variation	t
Kullback-Leibler (KL)	$\exp(t-1)$
Reverse KL	$-1-\log(-t)$
Pearson χ^2	$\frac{1}{4}t^2+t$
Neyman χ^2	$2-2\sqrt{1-t}$
Squared Hellinger	$\frac{t}{1-t}$
Jeffrey	$W(e^{1-t}) + \frac{1}{W(e^{1-t})} + t - 2$
Jensen-Shannon	$-\log(2-\exp(t))$
Jensen-Shannon-weighted	$(1-\pi)\log\frac{1-\pi}{1-\pi e^{t/\pi}}$
GAN	$-\log(1-\exp(t))$

Using the f-divergence you like ☺

https://arxiv.org/pdf/1606.00709.pdf

Experimental Results

- Approximate a mixture of Gaussians by single mixture



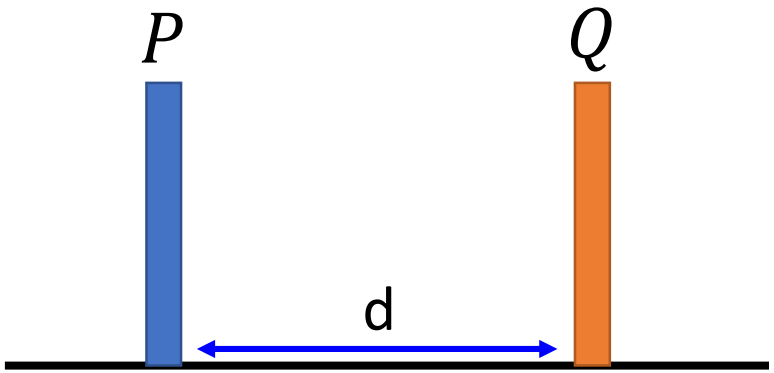
train \ test	KL	KL-rev	JS	Jeffrey	Pearson
KL	**0.2808**	0.3423	0.1314	0.5447	0.7345
KL-rev	0.3518	**0.2414**	0.1228	0.5794	1.3974
JS	0.2871	0.2760	**0.1210**	0.5260	0.92160
Jeffrey	0.2869	0.2975	0.1247	**0.5236**	0.8849
Pearson	0.2970	0.5466	0.1665	0.7085	**0.648**

WGAN

Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, arXiv prepring, 2017

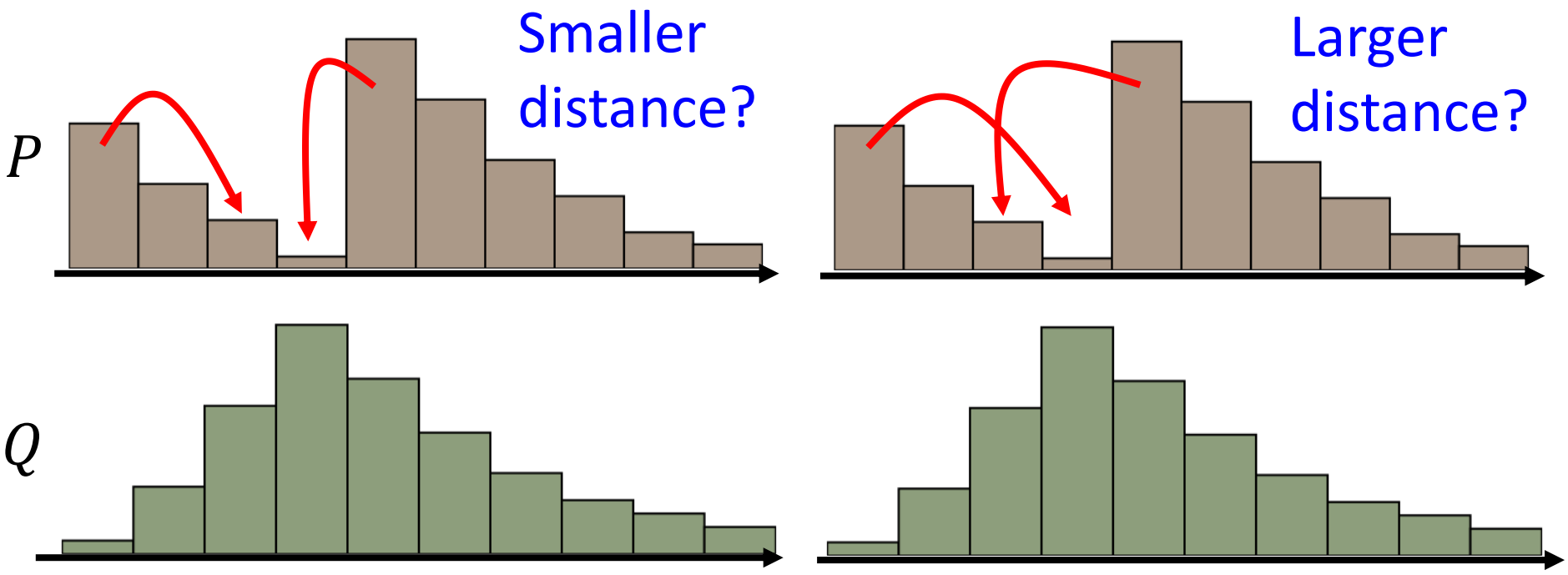
Earth Mover's Distance

- Considering one distribution P as a pile of earth, and another distribution Q as the target
- The average distance the earth mover has to move the earth.



$$W(P, Q) = d$$

Earth Mover's Distance



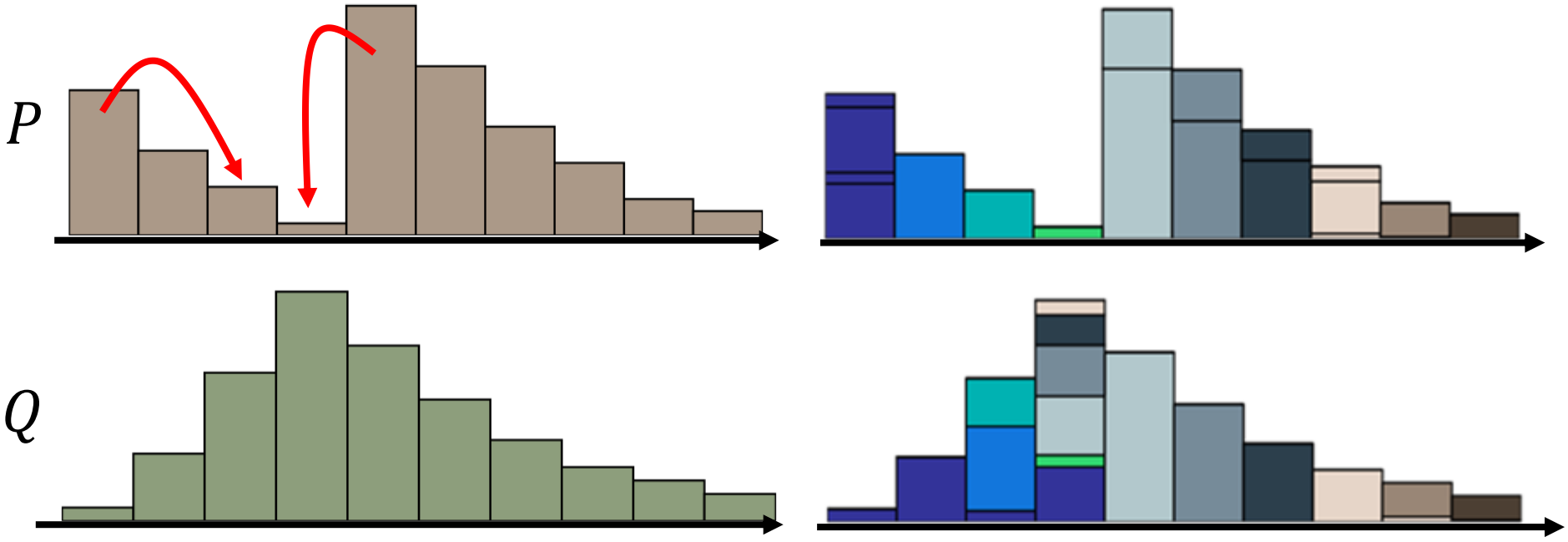
There many possible "moving plans".

Using the "moving plan" with the smallest average distance to define the earth mover's distance.

Source of image: https://vincentherrmann.github.io/blog/wasserstein/

Earth Mover's Distance

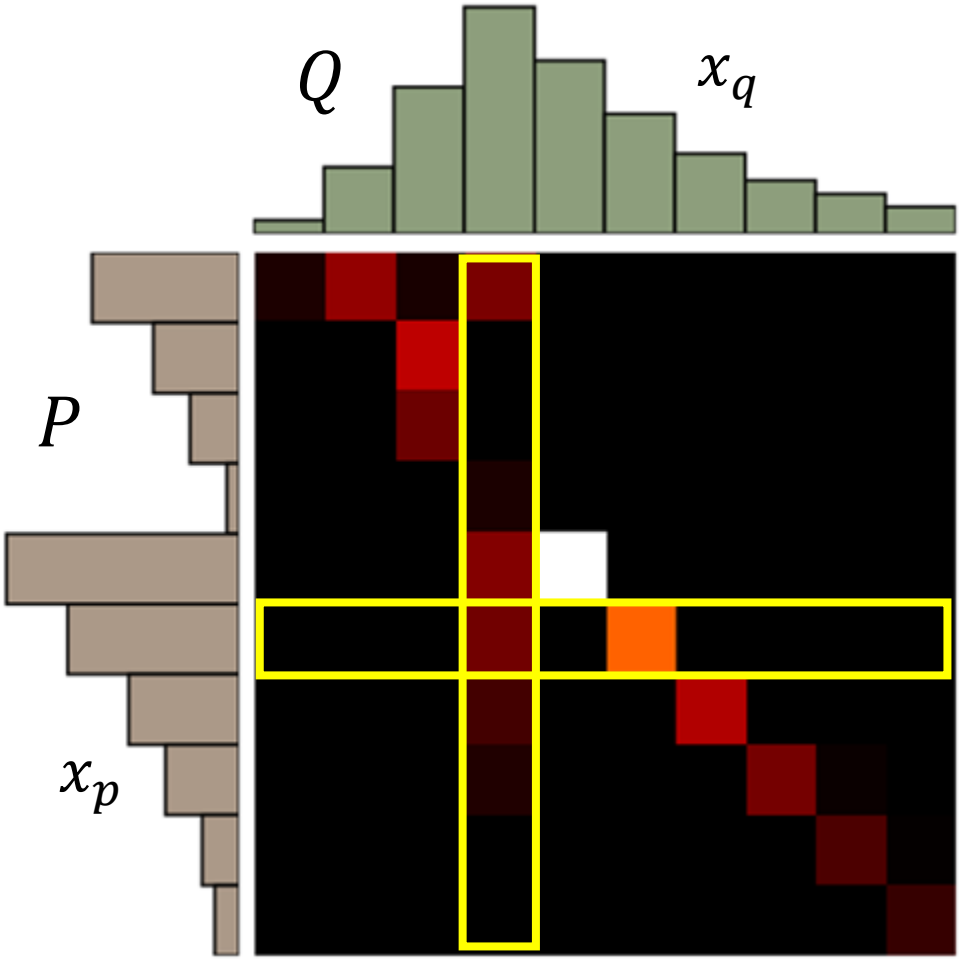
Best "moving plans" of this example



There many possible "moving plans".

Using the "moving plan" with the smallest average distance to define the earth mover's distance.

Source of image: https://vincentherrmann.github.io/blog/wasserstein/



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.

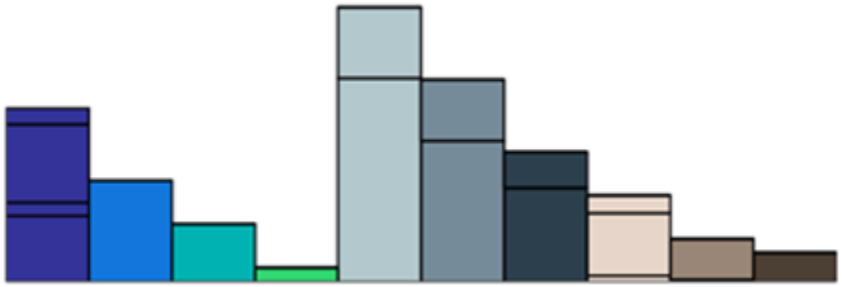
Average distance of a plan γ:

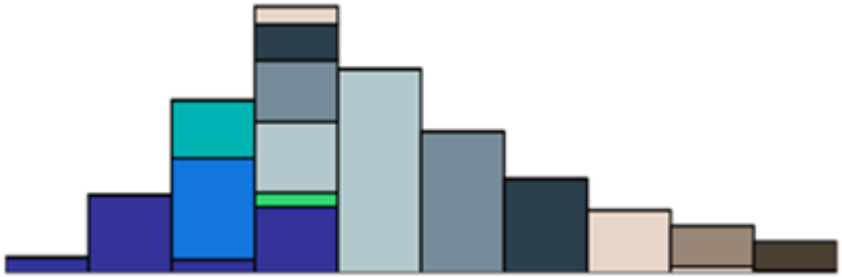
$$B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

Earth Mover’s Distance:

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

The best plan

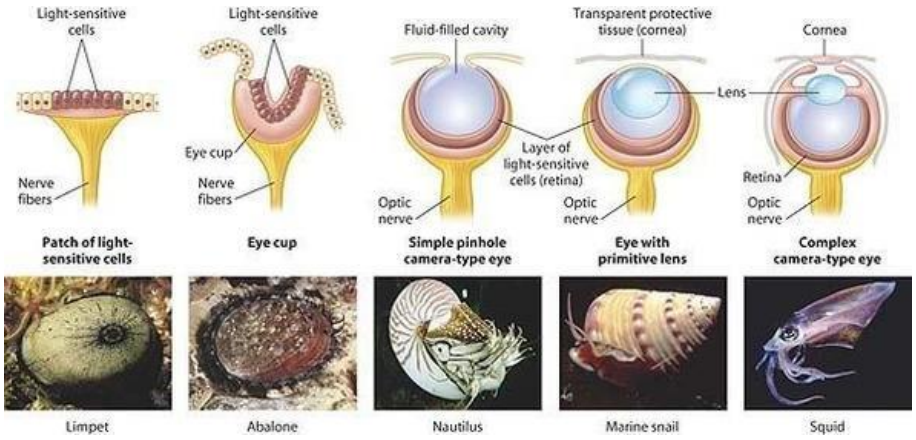


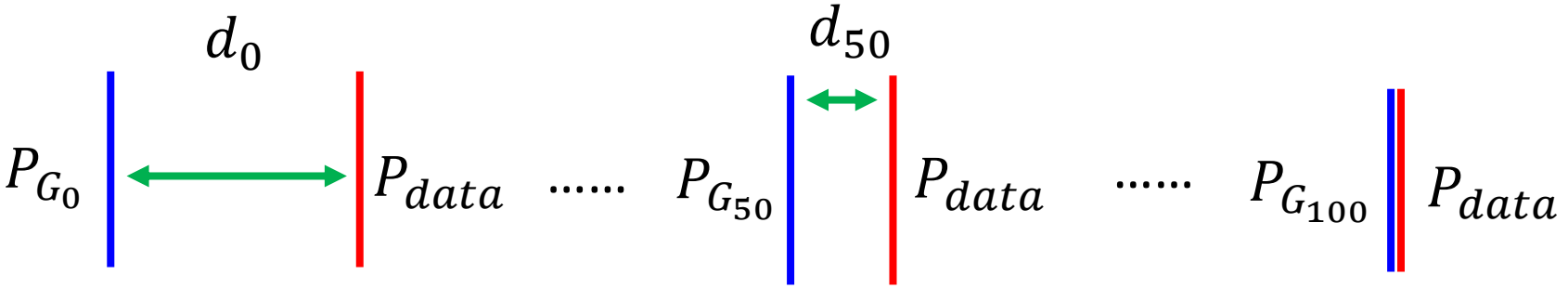


Why Earth Mover's Distance?

$$D_f(P_{data}||P_G)$$

$$W(P_{data}, P_G)$$





$$JS(P_{G_0}, P_{data}) = log2$$

$$JS(P_{G_{50}}, P_{data}) = log2$$

$$JS(P_{G_{100}}, P_{data}) = 0$$

$$W(P_{G_0}, P_{data}) = d_0$$

$$W(P_{G_{50}}, P_{data}) = d_{50}$$

$$W(P_{G_{100}}, P_{data}) = 0$$

Back to the GAN framework

$$D_f(P_{data}||P_G) \Rightarrow W(P_{data}, P_G)$$

$$= \max_D \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[f^*(D(x))]\}$$

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

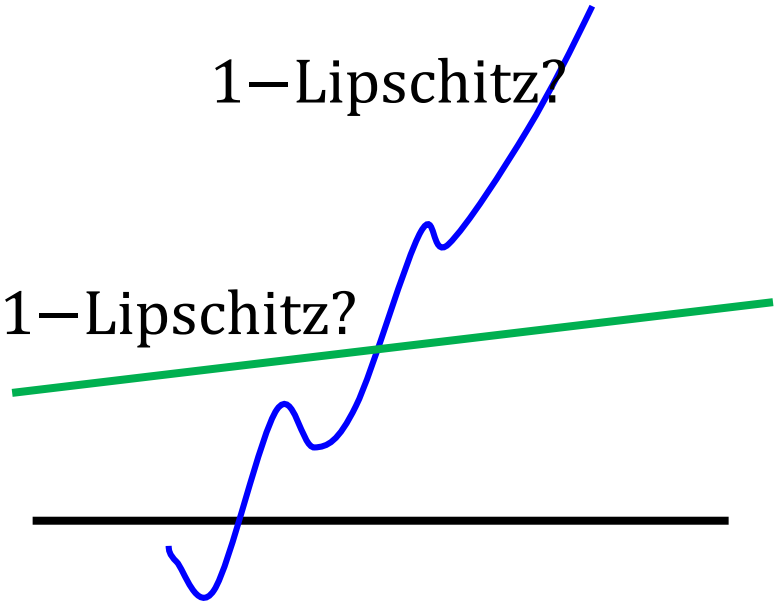
Lipschitz Function

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

Output change / Input change

K=1 for "$1-Lipschitz$"

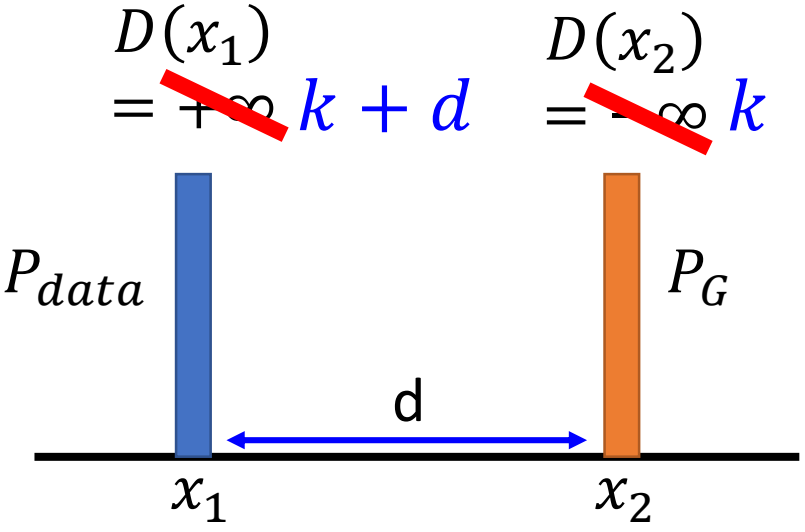
Do not change fast



Back to the GAN framework

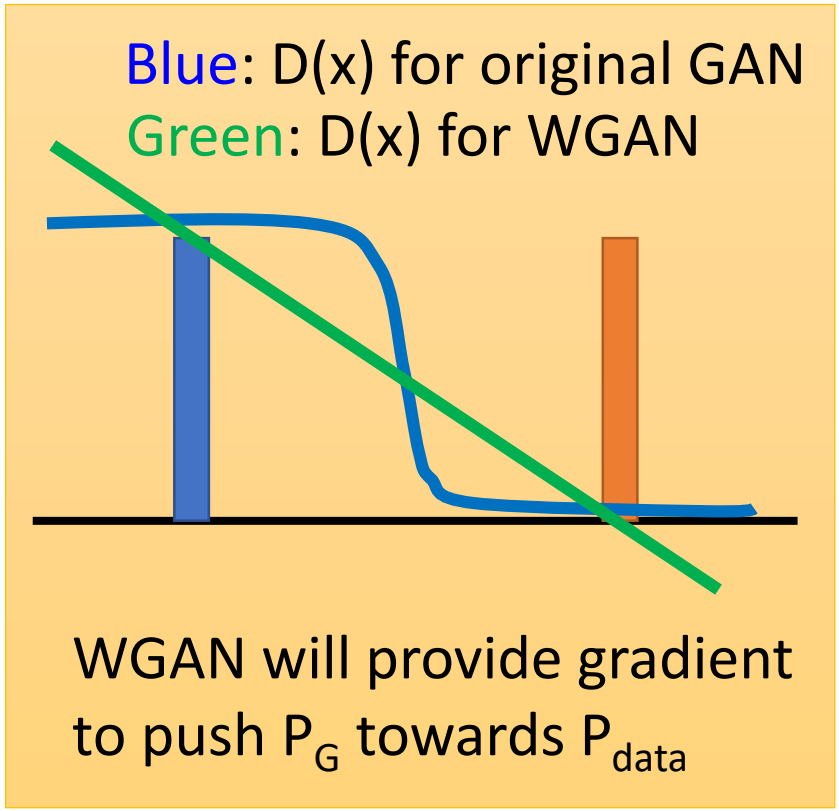
$$W(P_{data}, P_G) = \max_{D \in 1-Lipschitz} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\}$$

$k + d$ k



$$W(P_{data}, P_G) = d$$

$$\|D(x_1) - D(x_2)\| \leq \|x_1 - x_2\|$$



Back to the GAN framework

K $W(P_{data}, P_G)$

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

K

How to use gradient descent to optimize?

Weight clipping:

Force the weights w between c and -c

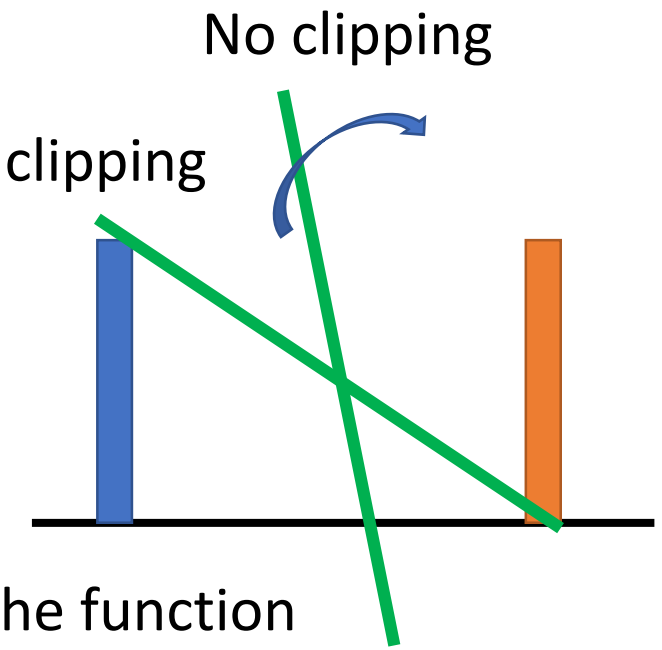
After parameter update,

if w > c, then w=c; if w<-**c**, then w=-c

We only ensure that

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

For some K



Do not truly find function D maximizing the function

Algorithm of WGAN

- In each training iteration:

No sigmoid for the output of D

Learning D

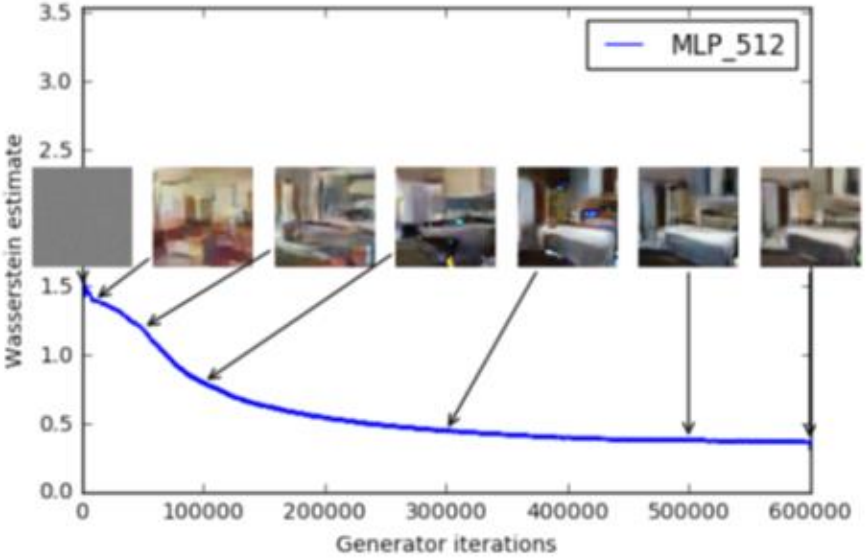
Repeat k times

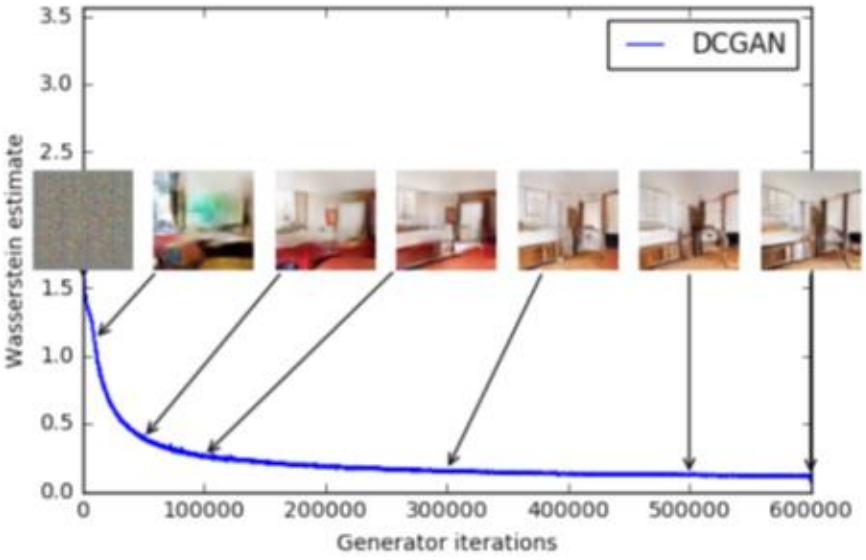
- Sample m examples $\{x^1, x^2, \dots, x^m\}$ from data distribution $P_{data}(x)$
- Sample m noise samples $\{z^1, z^2, \dots, z^m\}$ from the prior $P_{prior}(z)$
- Obtaining generated data $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$, $\tilde{x}^i = G(z^i)$
- Update discriminator parameters θ_d to maximize
  - $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m} D(x^i) - \frac{1}{m}\sum_{i=1}^{m} D(\tilde{x}^i)$
  - $\theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$ Weight clipping

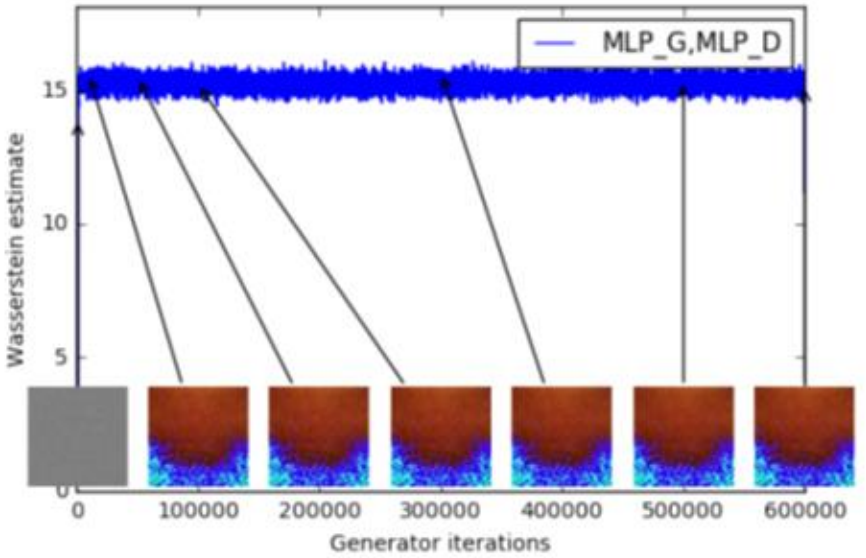
Learning G

Only Once

- Sample another m noise samples $\{z^1, z^2, \dots, z^m\}$ from the prior $P_{prior}(z)$
- Update generator parameters θ_g to minimize
  - $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m} \log D(x^i) - \frac{1}{m}\sum_{i=1}^{m} D\left(G(z^i)\right)$
  - $\theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$







Vertical

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

https://arxiv.org/abs/1701.07875

Improved WGAN

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville, “Improved Training of Wasserstein GANs”, arXiv prepring, 2017

Improved WGAN

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

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

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

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

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

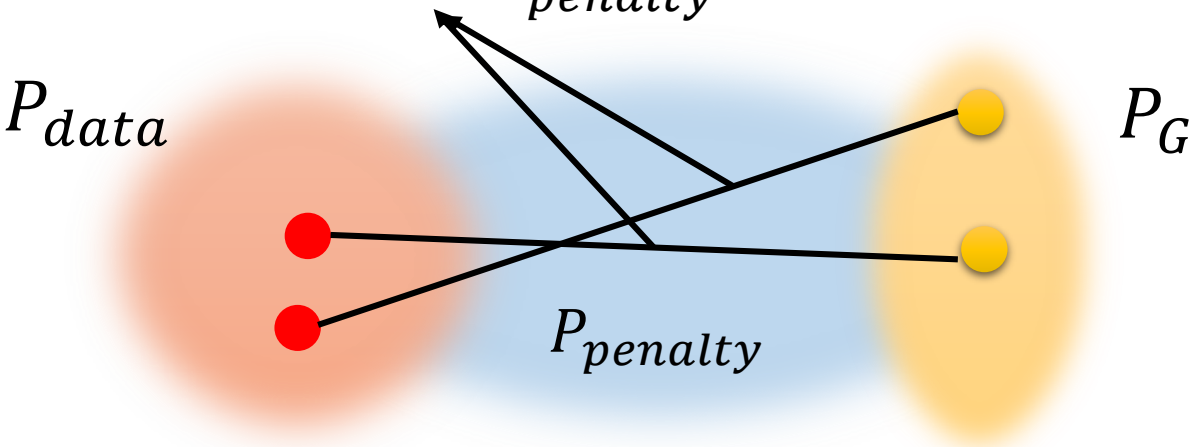
Prefer $\|\nabla_x D(x)\| \leq 1$ for all x

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

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

Improved WGAN

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

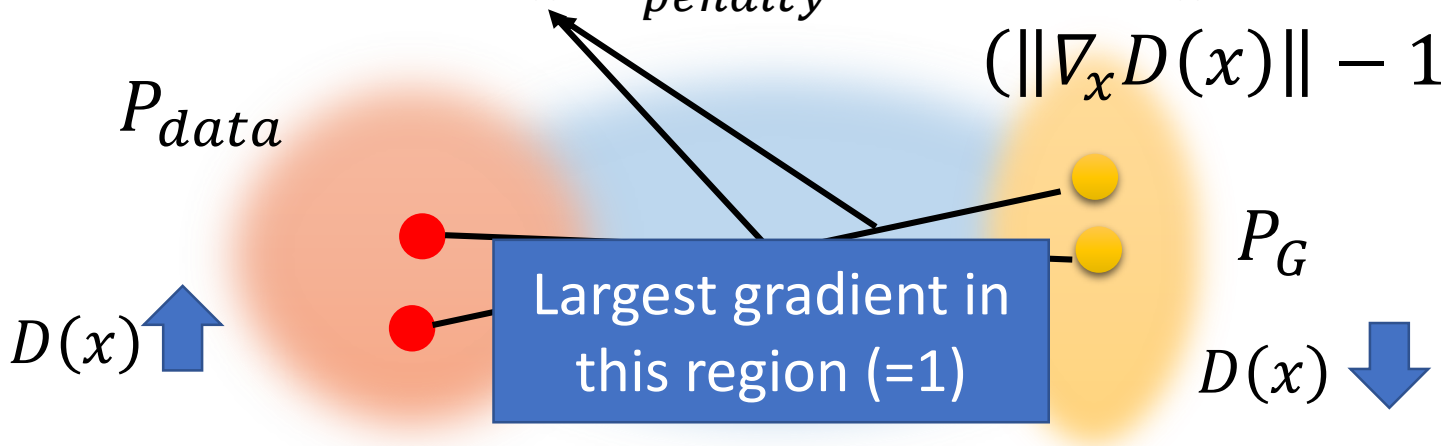


"Given that enforcing the Lipschitz constraint everywhere is intractable, enforcing it ***only along these straight lines*** seems sufficient and experimentally results in good performance."

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

$$W(P_{data}, P_G) \approx \max_D \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)]$$
$$-\lambda E_{x\sim P_{penalty}}[\cancel{max(0, \|\nabla_x D(x)\| - 1)}]\}$$
$$(\|\nabla_x D(x)\| - 1)^2$$



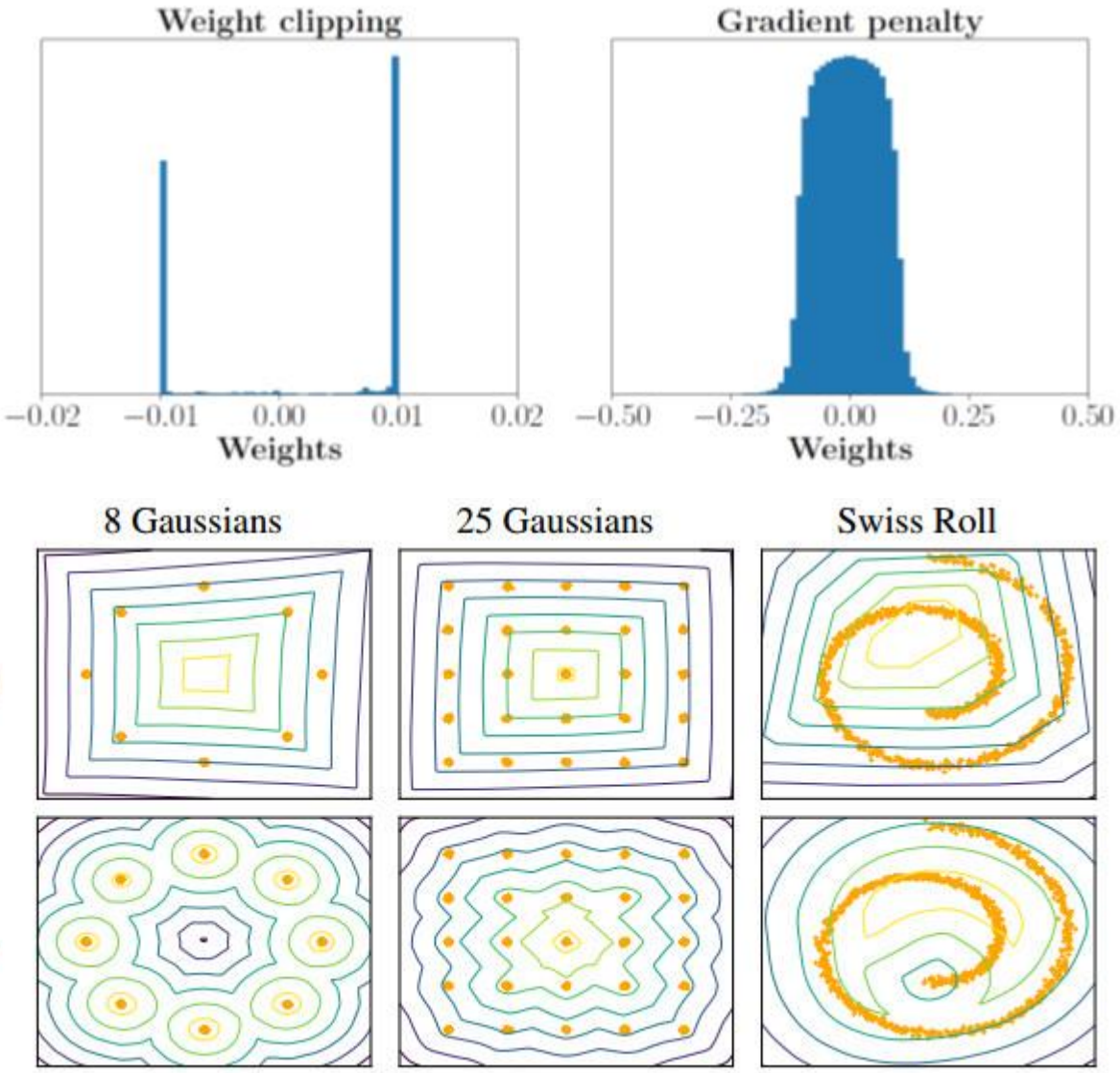
"One may wonder why we penalize the norm of the gradient for differing from 1, instead of just penalizing large gradients. The reason is that the optimal critic … actually has gradients with norm 1 almost everywhere under Pr and Pg"

(check the proof in the appendix)

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

https://arxiv.org/abs/1704.00028

Improved WGAN



DCGAN

LSGAN

Original WGAN

Improved WGAN

G: CNN, D: CNN




G: CNN (no normalization), D: CNN (no normalization)




G: CNN (tanh), D: CNN(tanh)

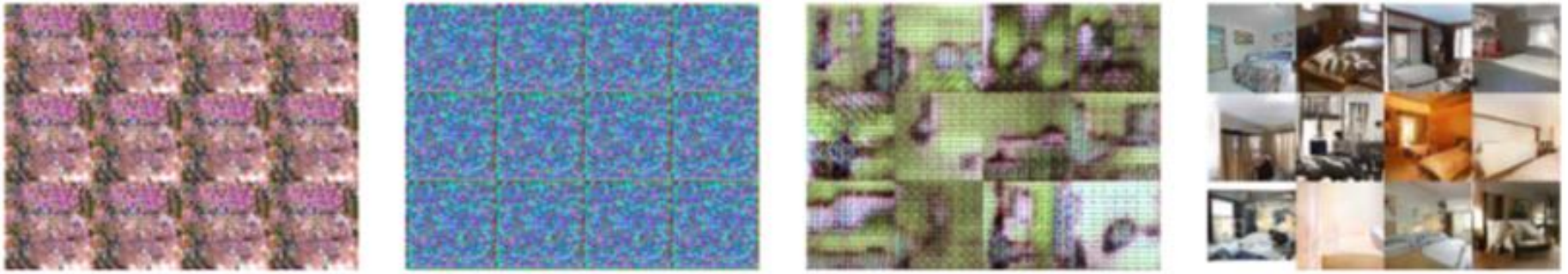



DCGAN | LSGAN | Original WGAN | Improved WGAN

G: MLP, D: CNN

G: CNN (bad structure), D: CNN

G: 101 layer, D: 101 layer



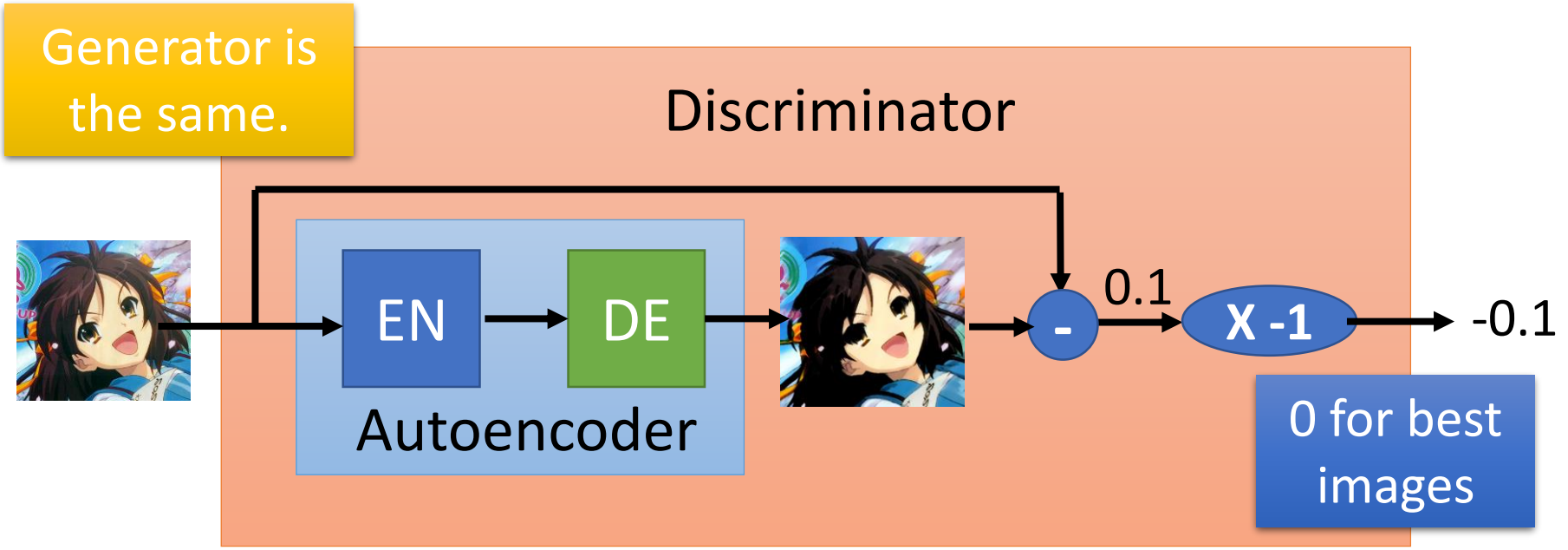
Energy-based GAN

Ref: Junbo Zhao, Michael Mathieu, Yann LeCun, Energy-based Generative Adversarial Network, ICRL 2017

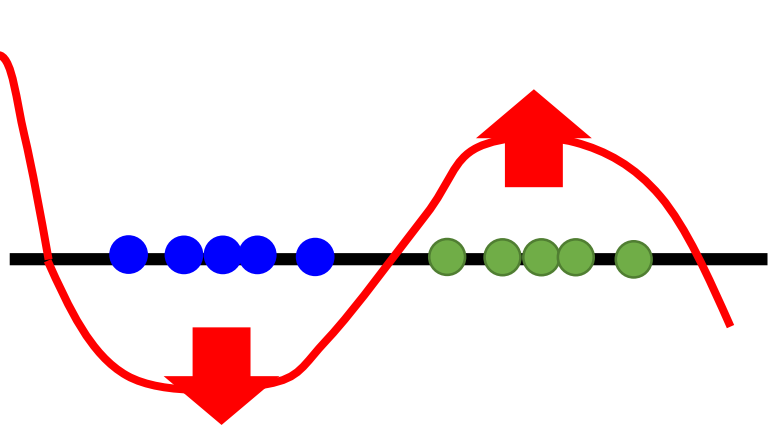
Energy-based GAN (EBGAN)

- Using an autoencoder as discriminator D

An image is good. = It can be reconstructed by autoencoder.

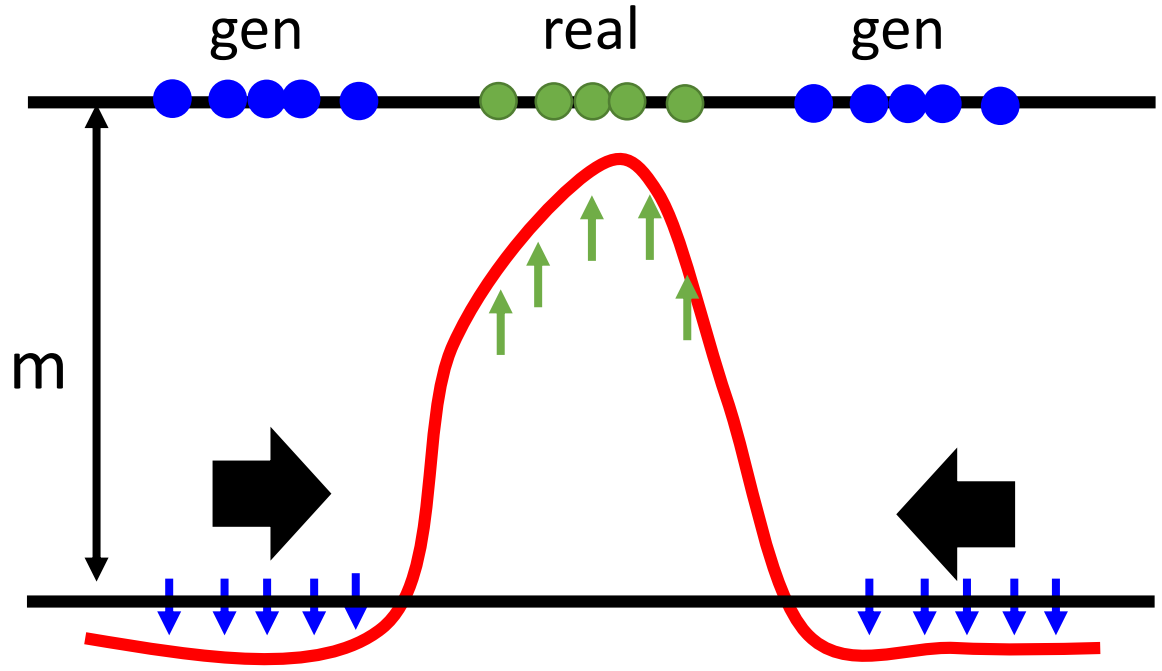


EBGAN



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

0 is for
the best.



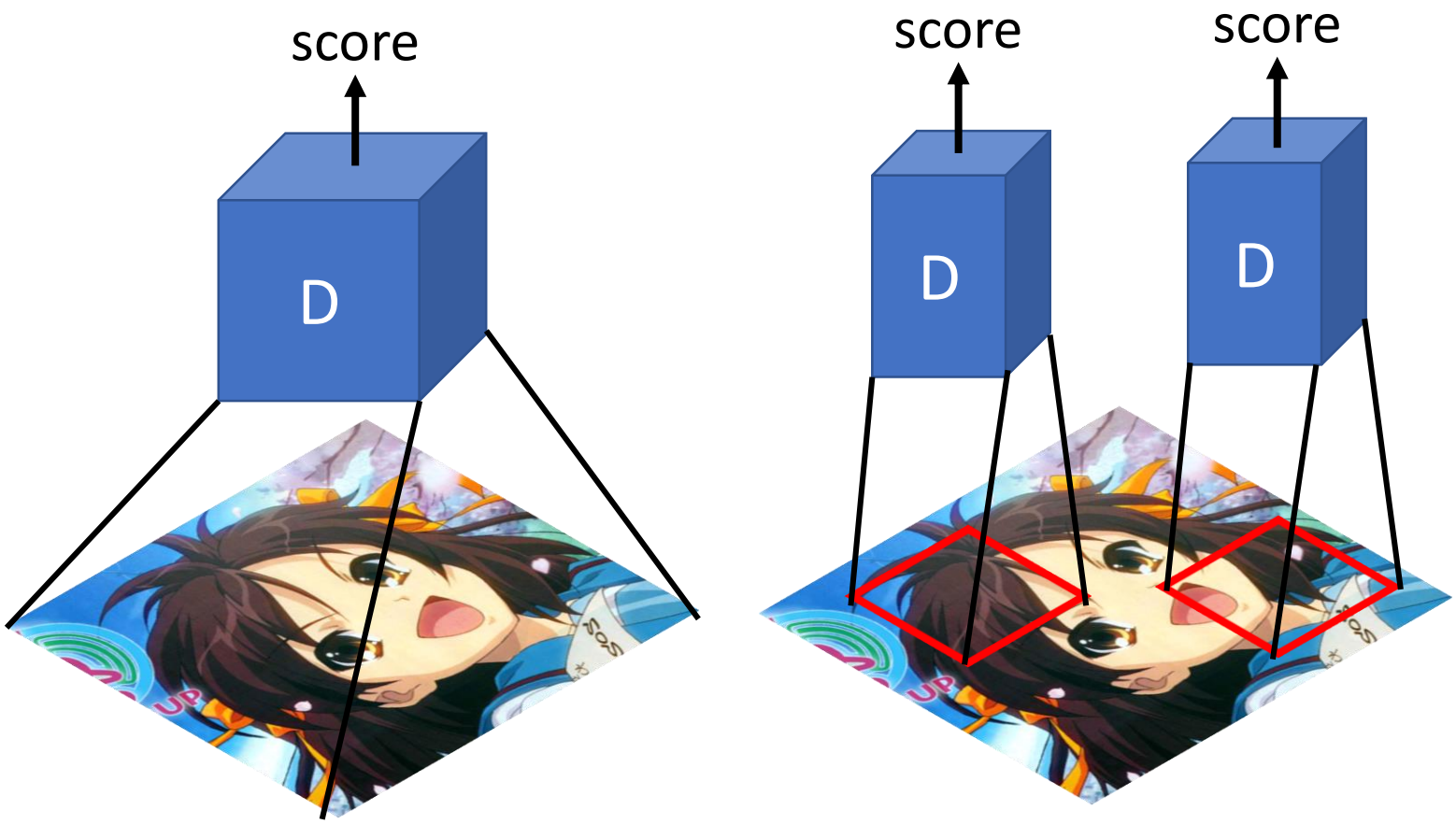
Do not have to
be very negative

Hard to reconstruct, easy to destroy

Stack and Progressive GAN

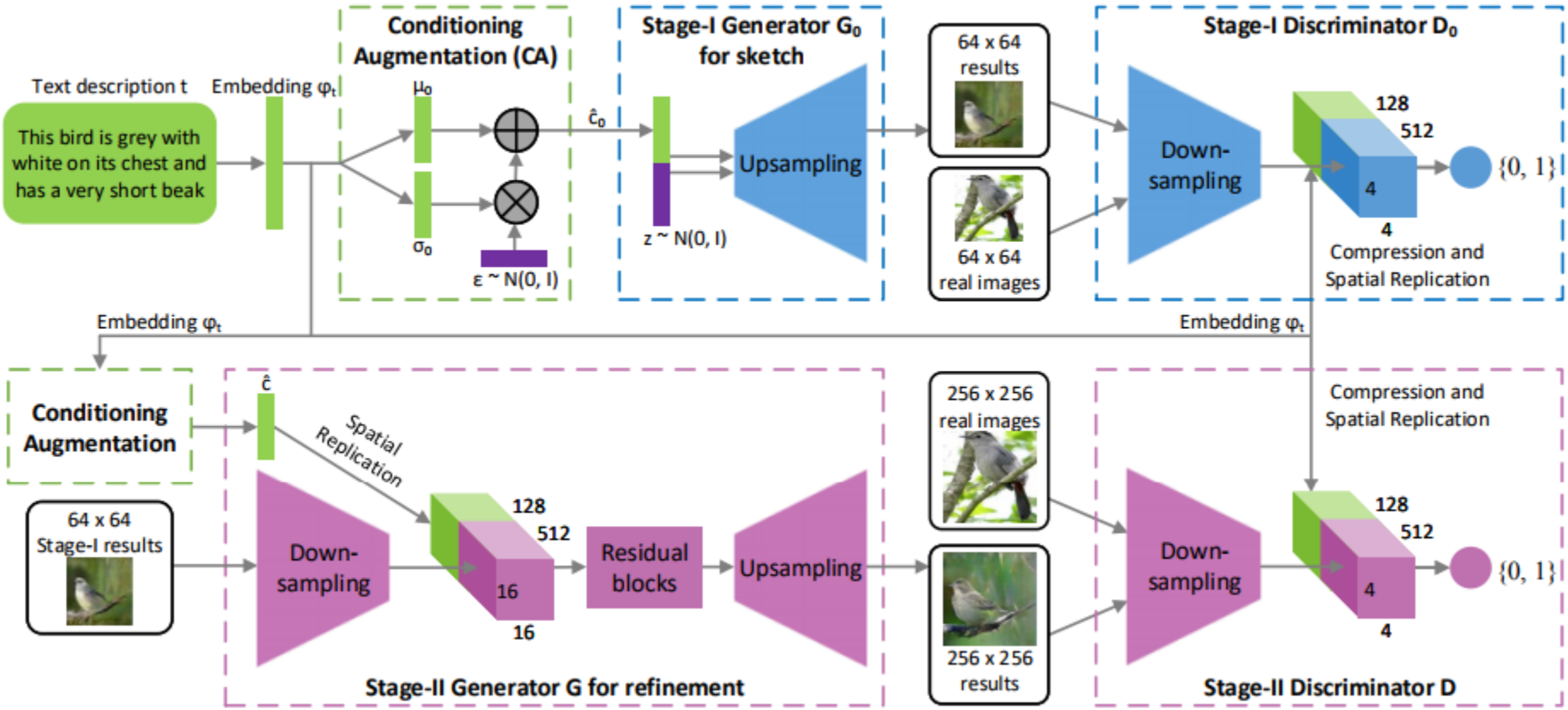
Patch GAN

https://arxiv.org/pdf/1611.07004.pdf



Stack GAN

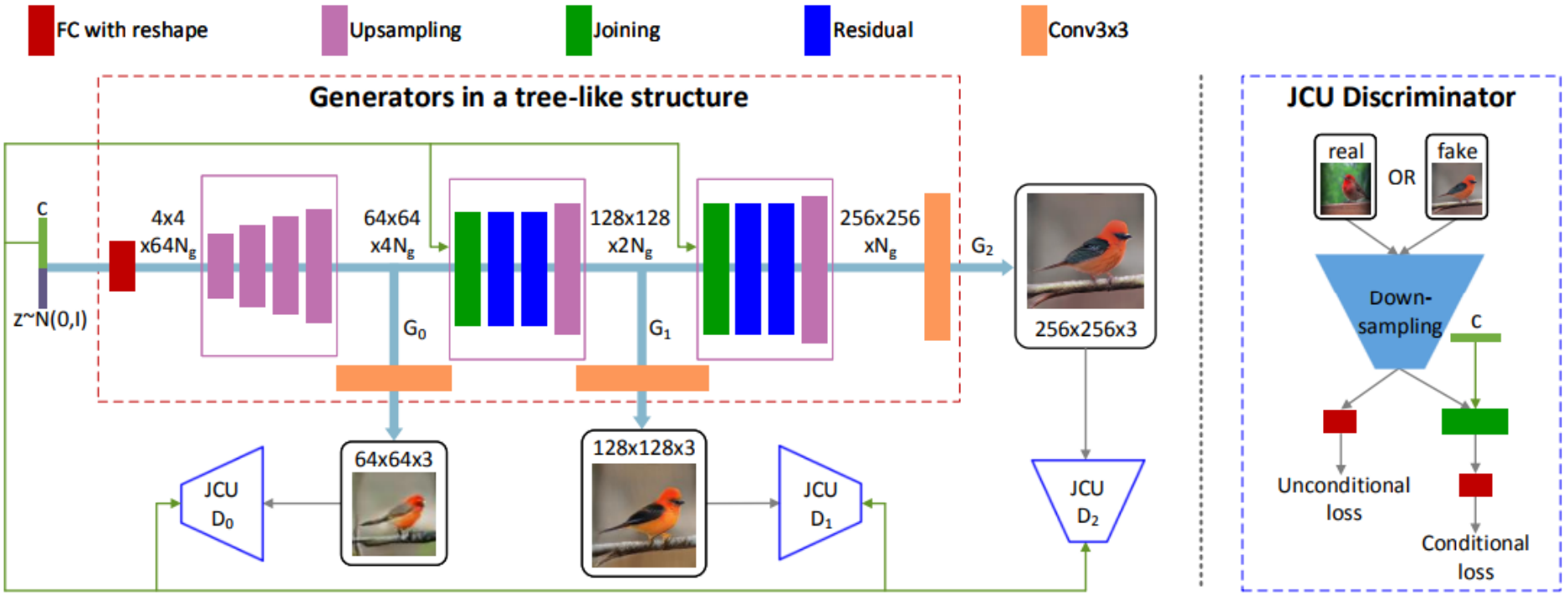
Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks", ICCV, 2017



Stack GAN ++

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas, "StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks", arXiv 2017

- Tree-like structure

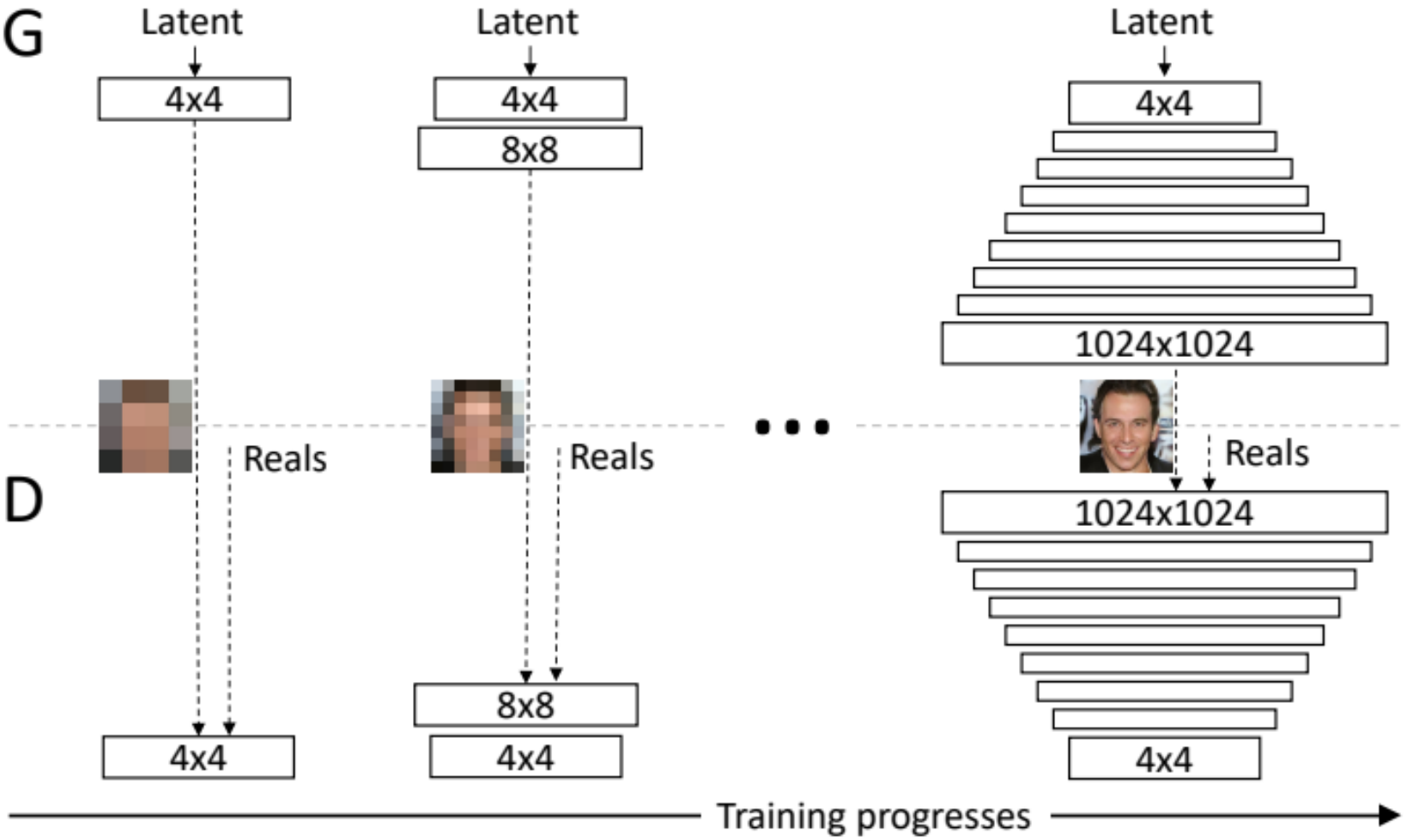


c.f. ACGAN

https://arxiv.org/pdf/1610.09585.pdf

Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen, " Progressive Growing of GANs for Improved Quality, Stability, and Variation", arXiv 2017

Progressive Growing of GAN



Ensemble

Observation

陳柏文同學提供
實驗結果

Yaxing Wang, Lichao Zhang, Joost van de Weijer, “Ensembles of Generative Adversarial Networks”, NIPS workshop, 2016

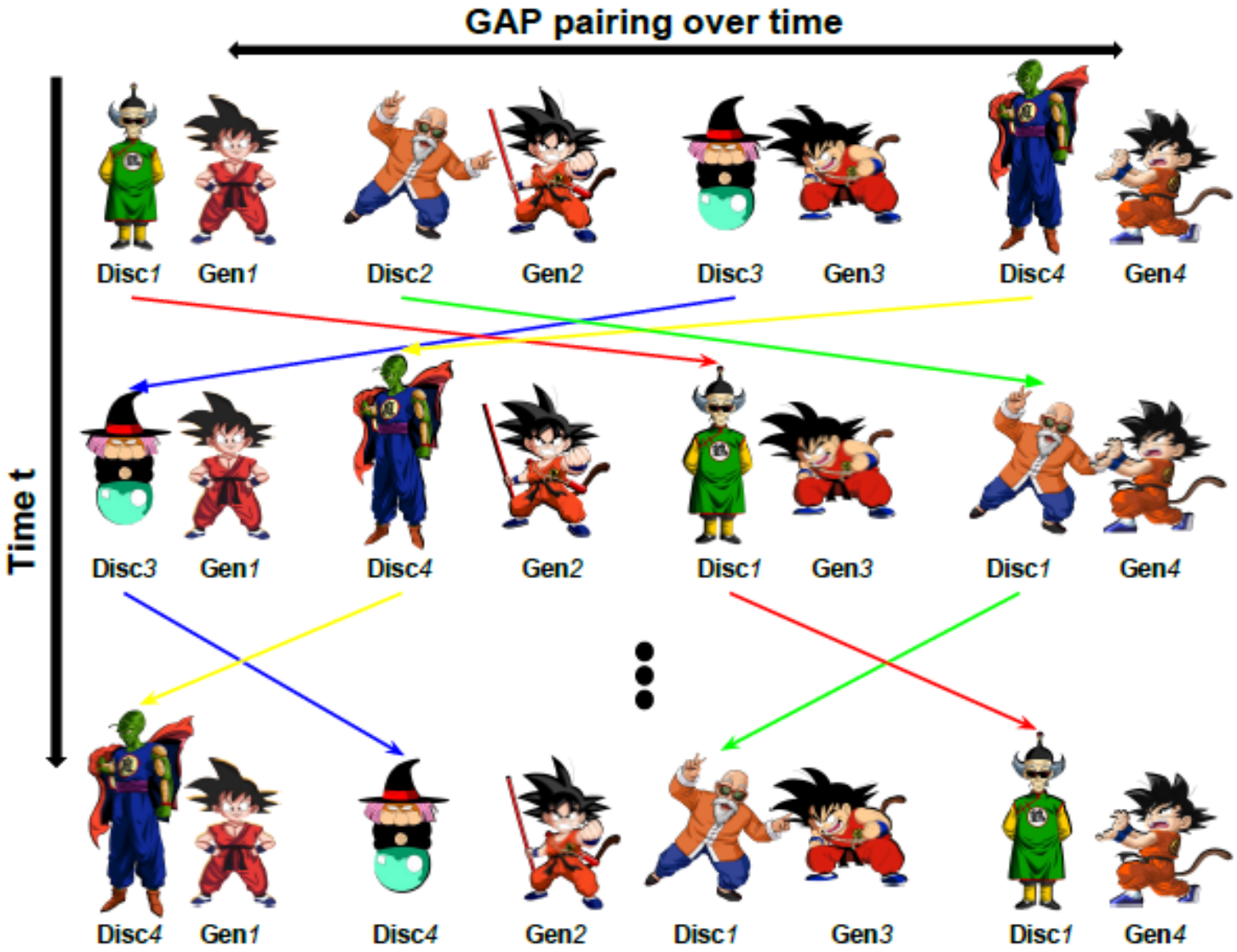
圖片來源 柯達方 同學

Ensemble

Generator 1

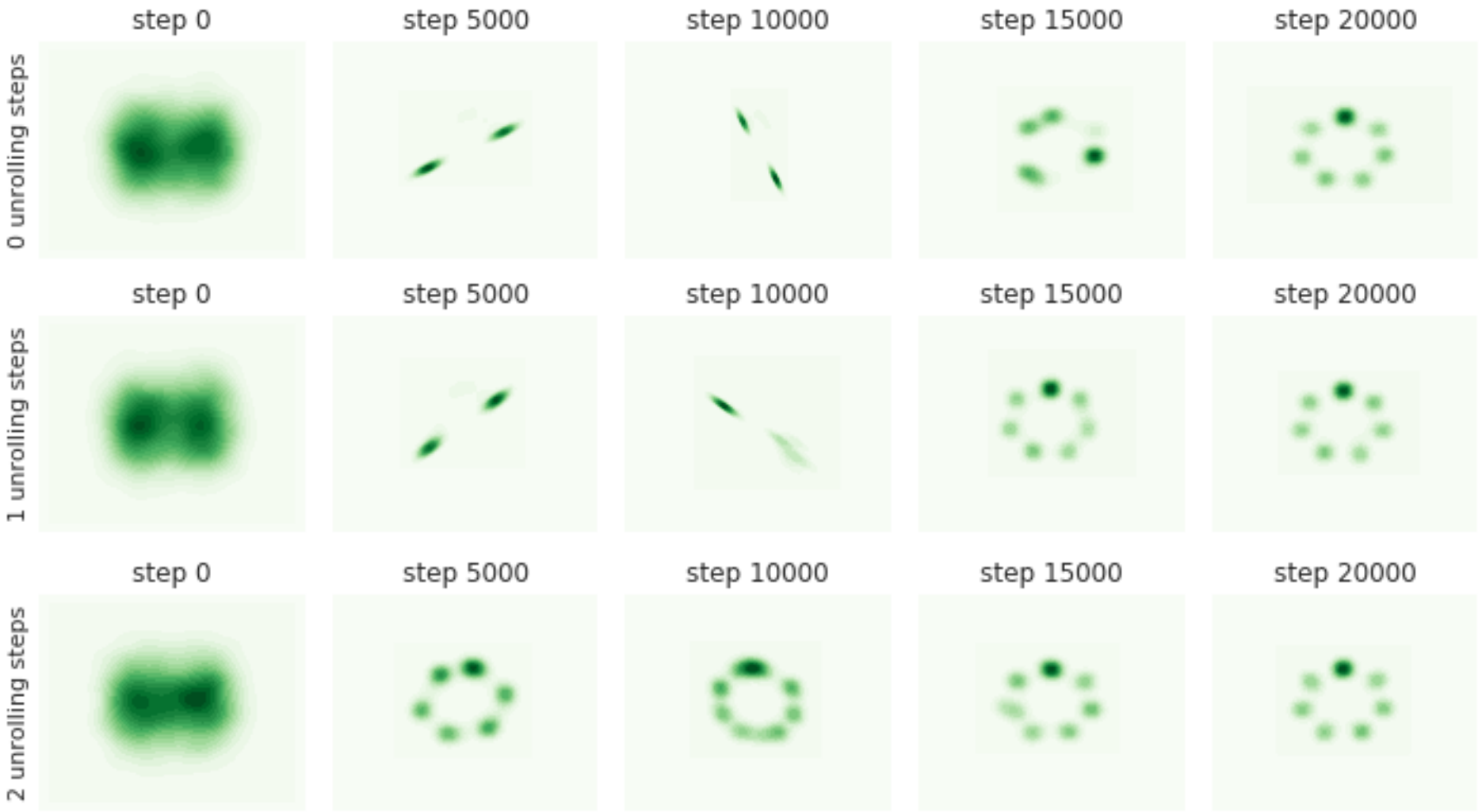
Generator 2

Generative Adversarial Parallelization



This figure is from the original paper: https://arxiv.org/abs/1612.04021

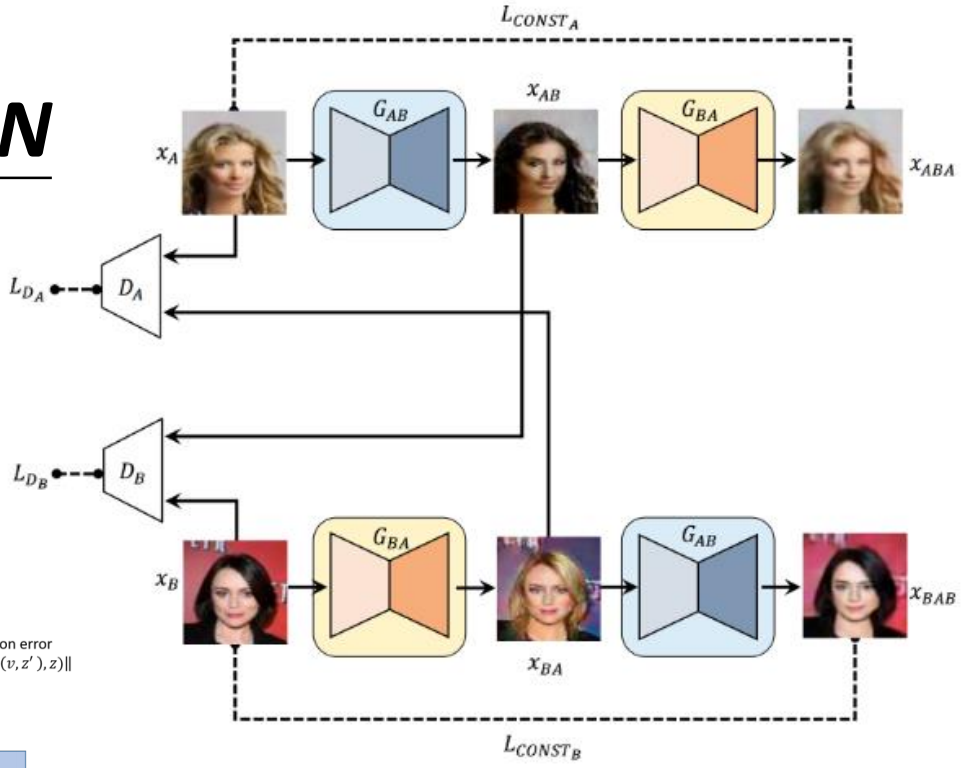
Unroll GAN – Experimental Results



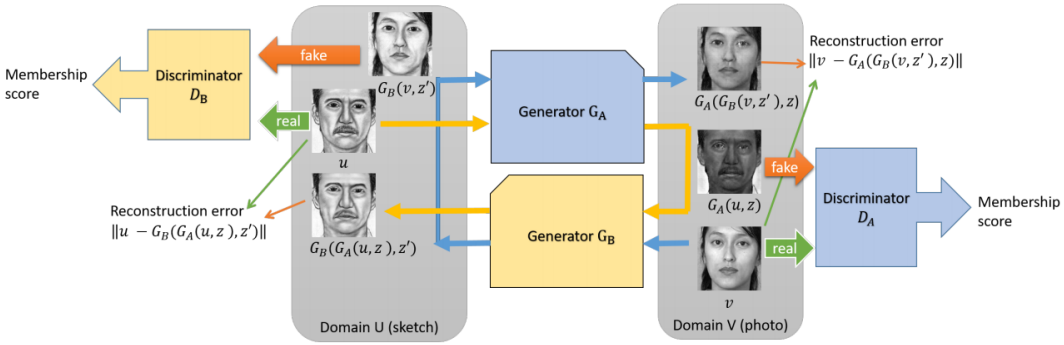
Code: https://github.com/poolio/unrolled_gan/blob/master/Unrolled%20GAN%20demo.ipynb

Style Transfer

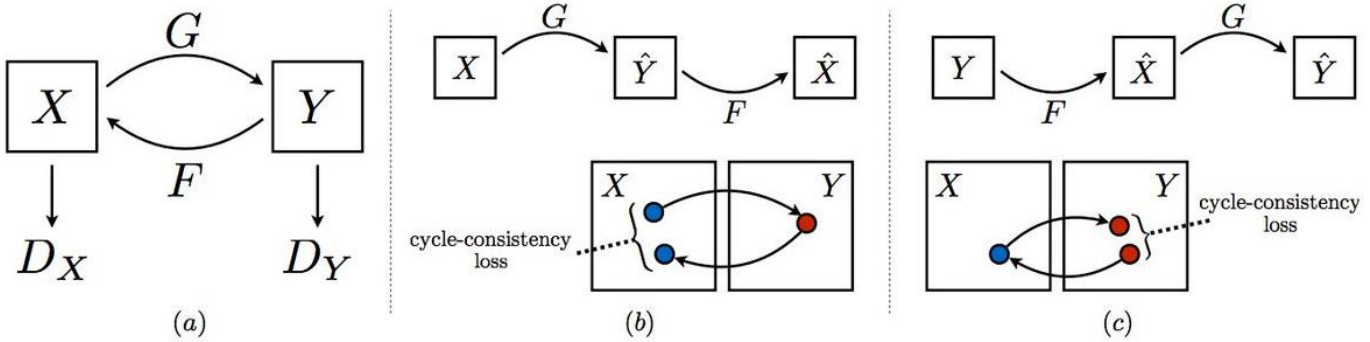
Disco GAN



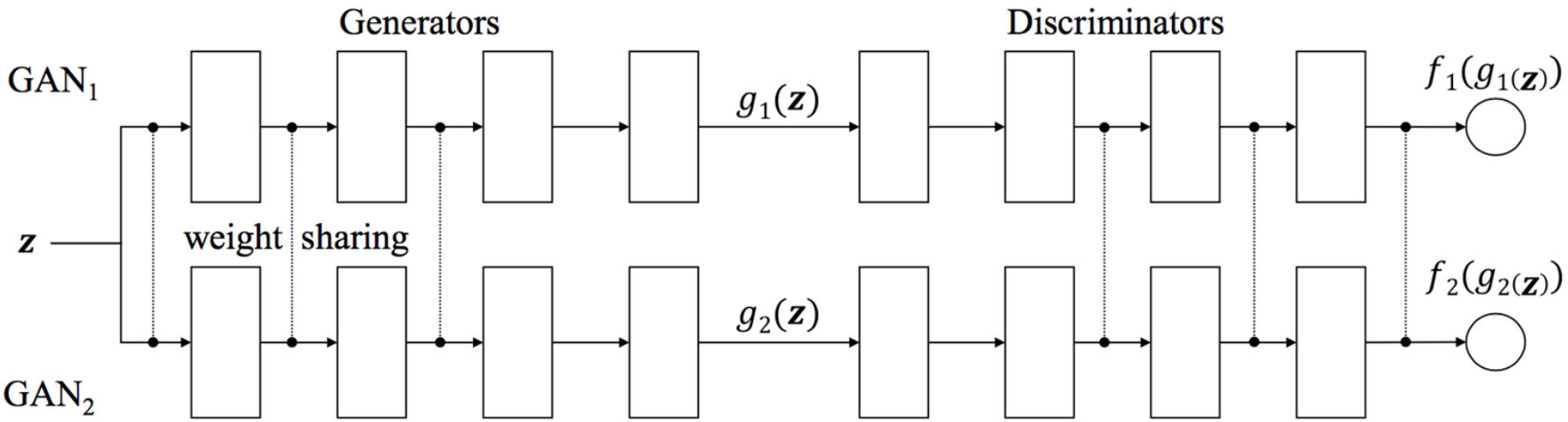
Dual GAN



Cycle GAN



Coupled GAN (CoGAN)



Ming-Yu Liu, Oncel Tuzel, “Coupled Generative Adversarial Networks”, NIPS, 2016

UNIT: Unsupervised Image-to-image Translation

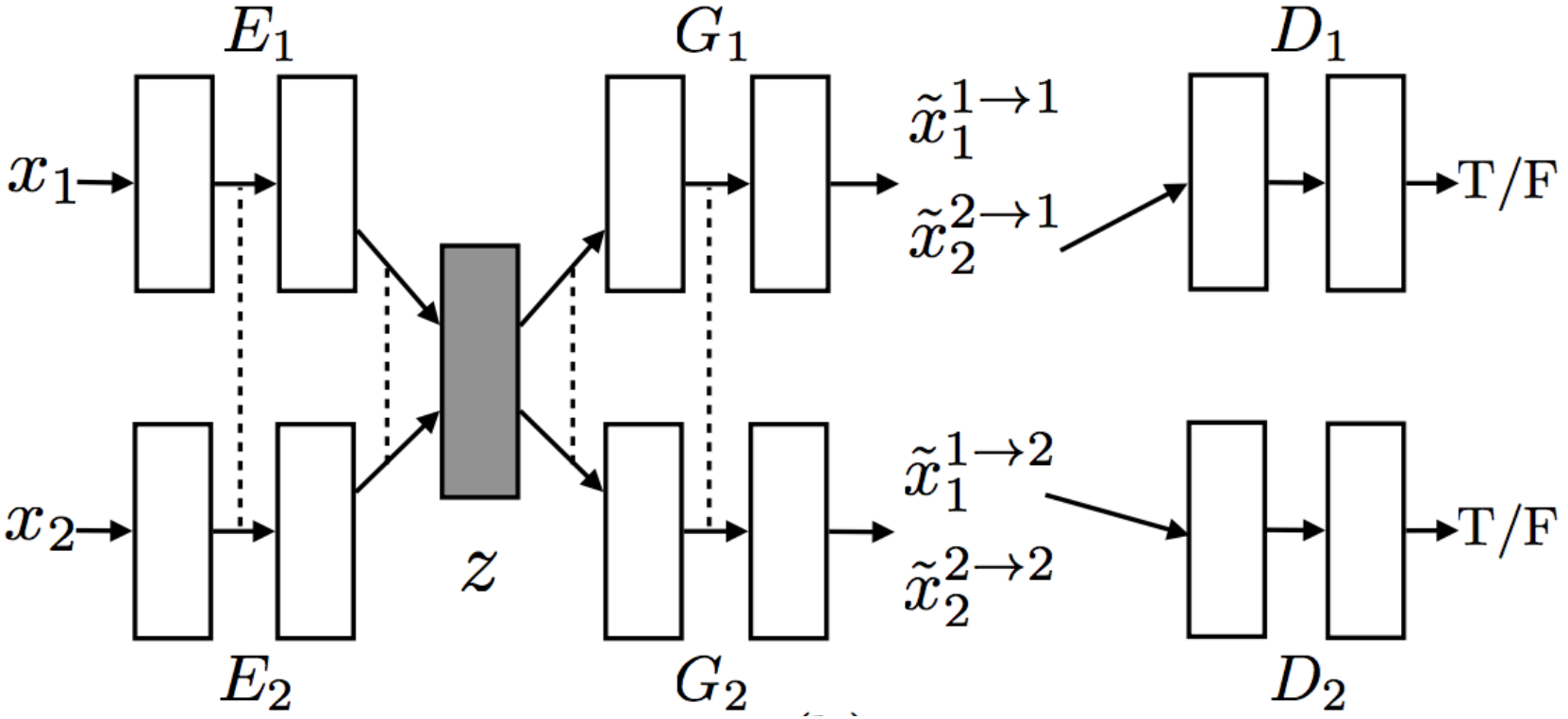


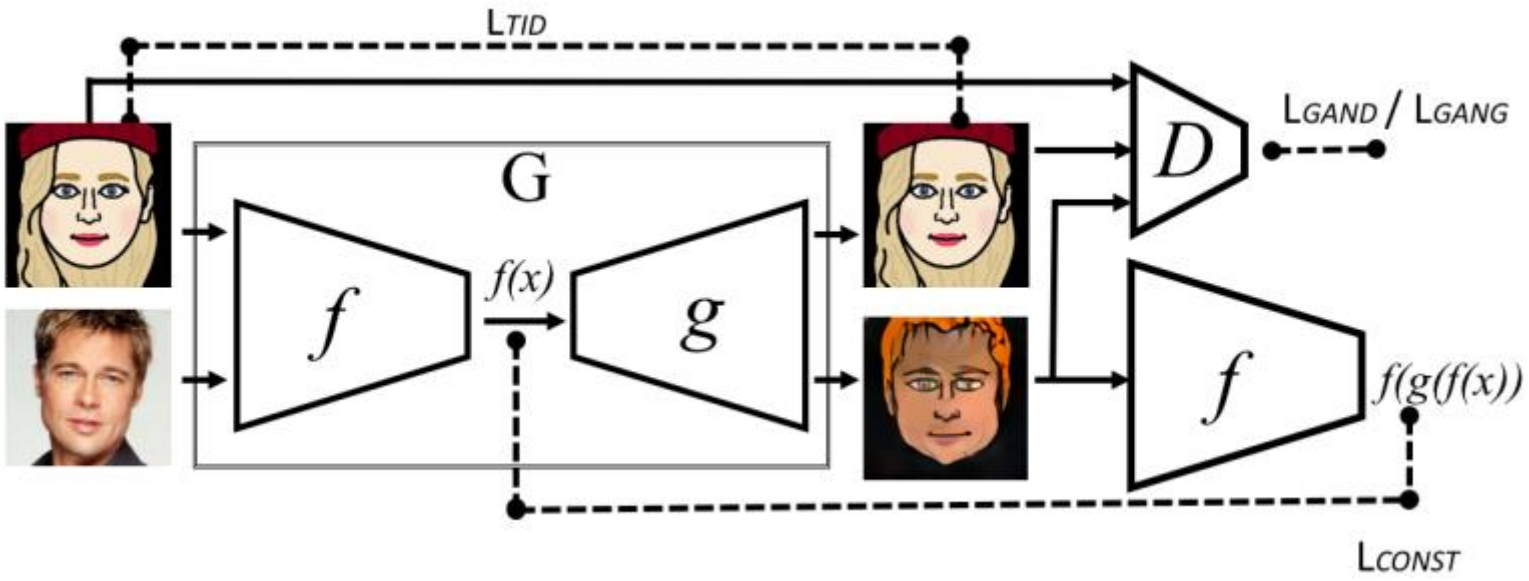
Table 1: Interpretation of the roles of the subnetworks in the proposed framework.

Networks	$\{E_1, G_1\}$	$\{E_1, G_2\}$	$\{G_1, D_1\}$	$\{E_1, G_1, D_1\}$	$\{G_1, G_2, D_1, D_2\}$
Roles	VAE for $\mathcal{X}_1$	Image Translator $\mathcal{X}_1 \to \mathcal{X}_2$	GAN for $\mathcal{X}_1$	VAE-GAN [14]	CoGAN [17]

https://arxiv.org/pdf/1703.00848.pdf
https://github.com/mingyuliutw/unit

https://arxiv.org/pdf/1611.02200.pdf

DTN: Domain Transfer Network



$$L_{\mathrm{D}} = -\mathbb{E}_{x\in\mathbf{s}} \log D_1(g(f(x))) - \mathbb{E}_{x\in\mathbf{t}} \log D_2(g(f(x))) - \mathbb{E}_{x\in\mathbf{t}} \log D_3(x)$$

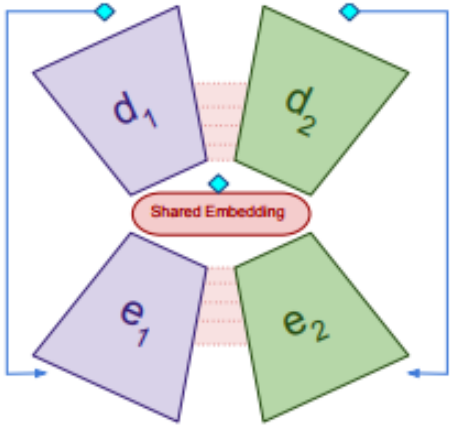
$$L_{\mathrm{GANG}} = -\mathbb{E}_{x\in\mathbf{s}} \log D_3(g(f(x))) - \mathbb{E}_{x\in\mathbf{t}} \log D_3(g(f(x)))$$

$$L_{\mathrm{CONST}} = \sum_{x\in\mathbf{s}} d(f(x), f(g(f(x))))$$

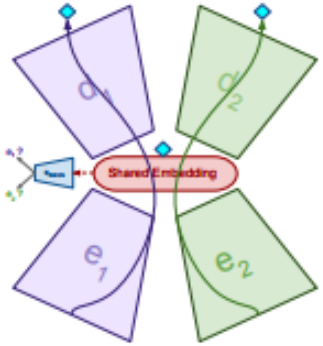
$$L_{\mathrm{TID}} = \sum_{x\in\mathbf{t}} d_2(x, G(x))$$

XGAN

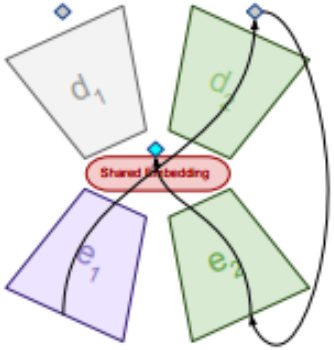
https://arxiv.org/pdf/1711.05139.pdf



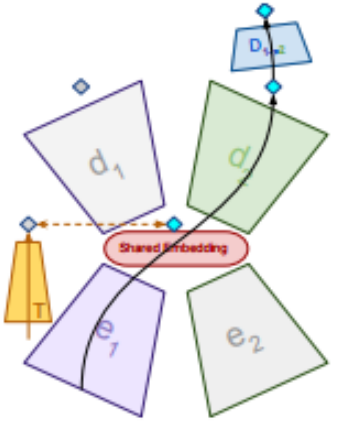
(A) High-level view of the XGAN architecture



(B1) Domain-adversarial autoencoder



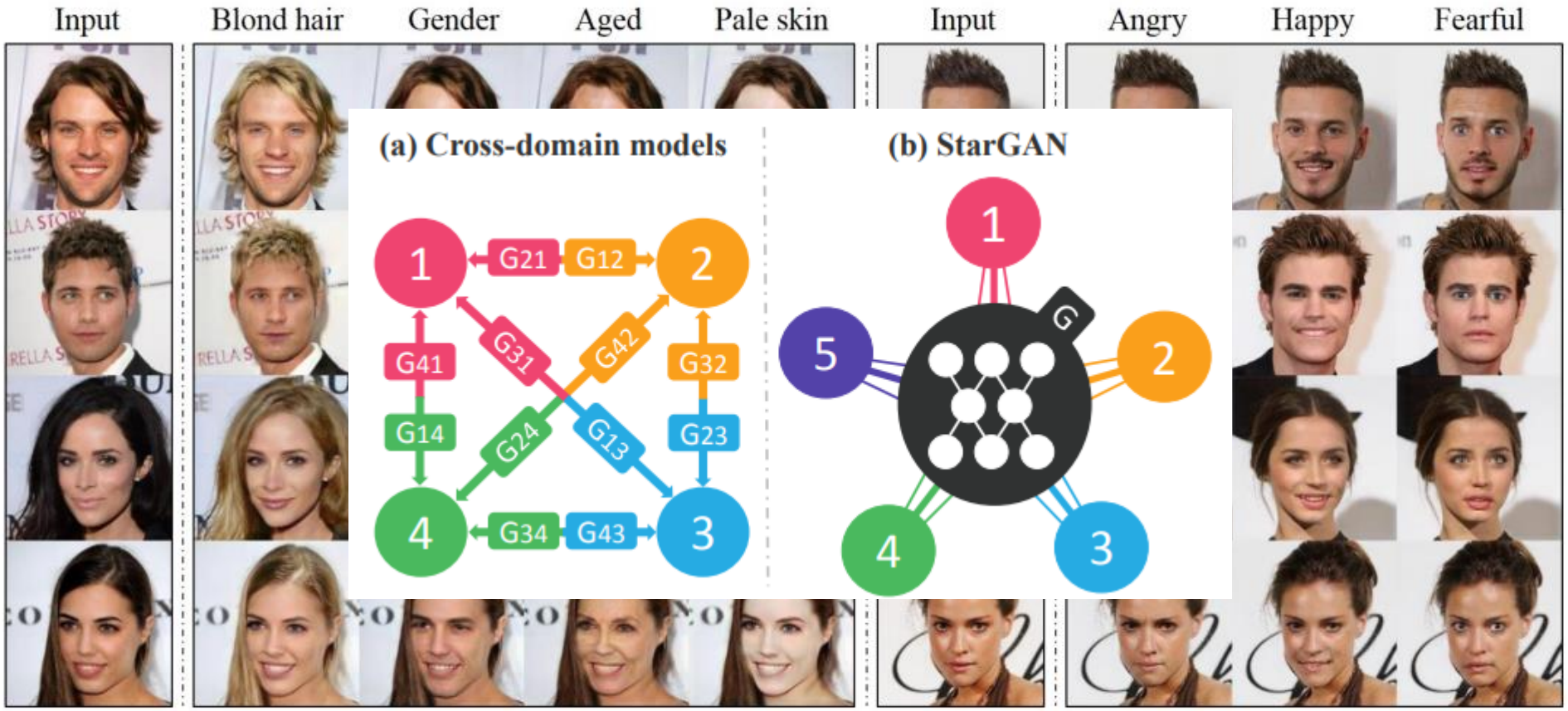
(B2) Semantic consistency



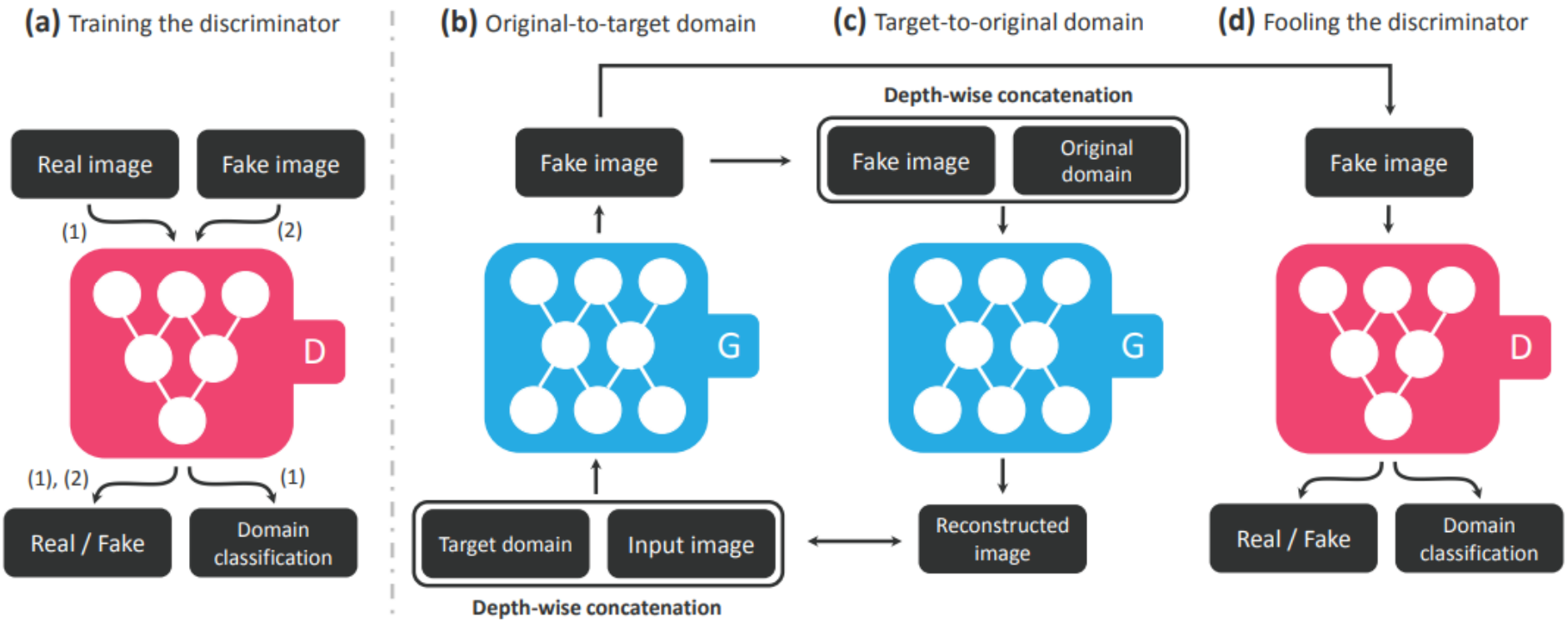
(B3) Refinements for sample quality

https://arxiv.org/pdf/1711.09020.pdf

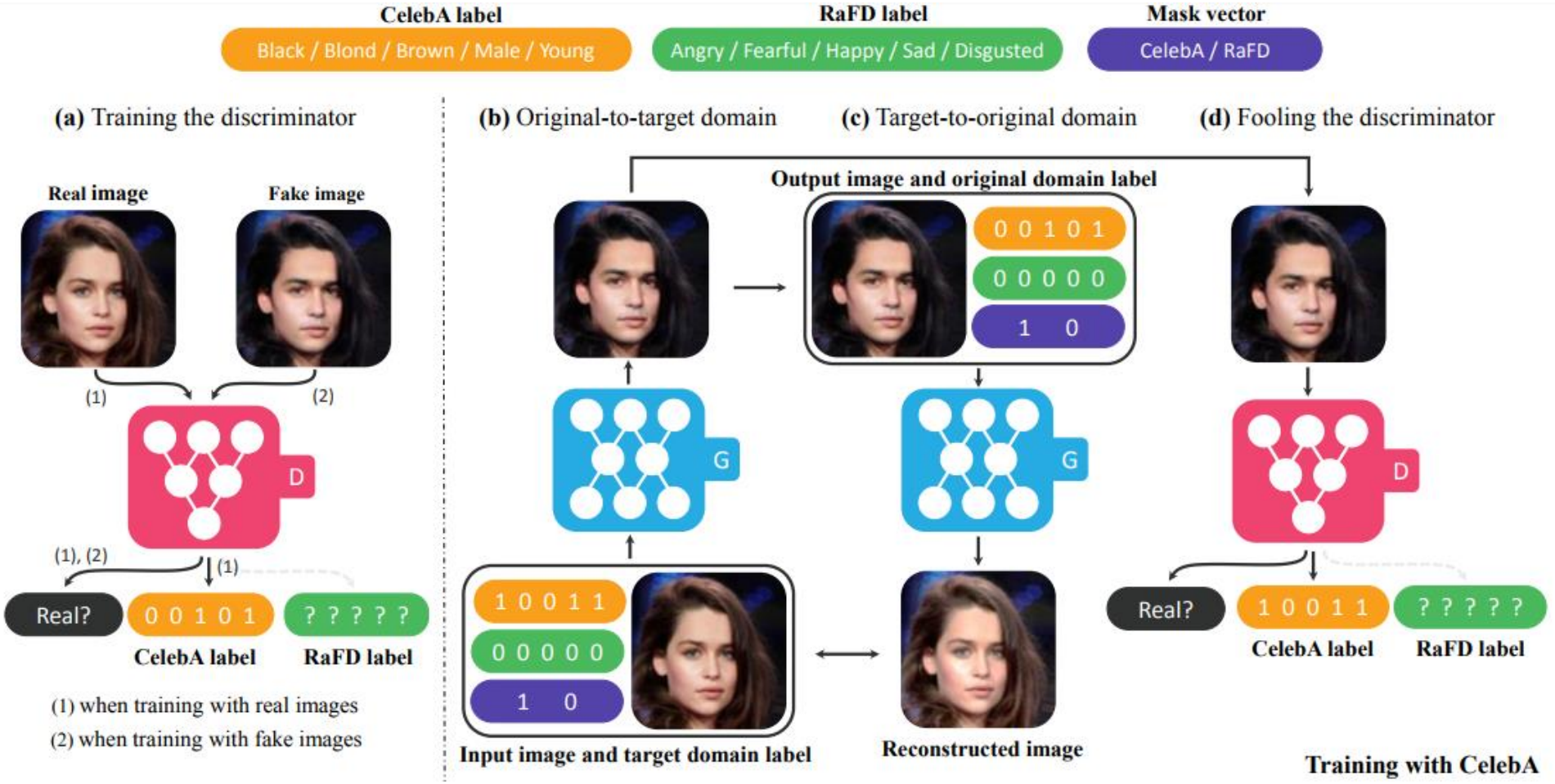
StarGAN



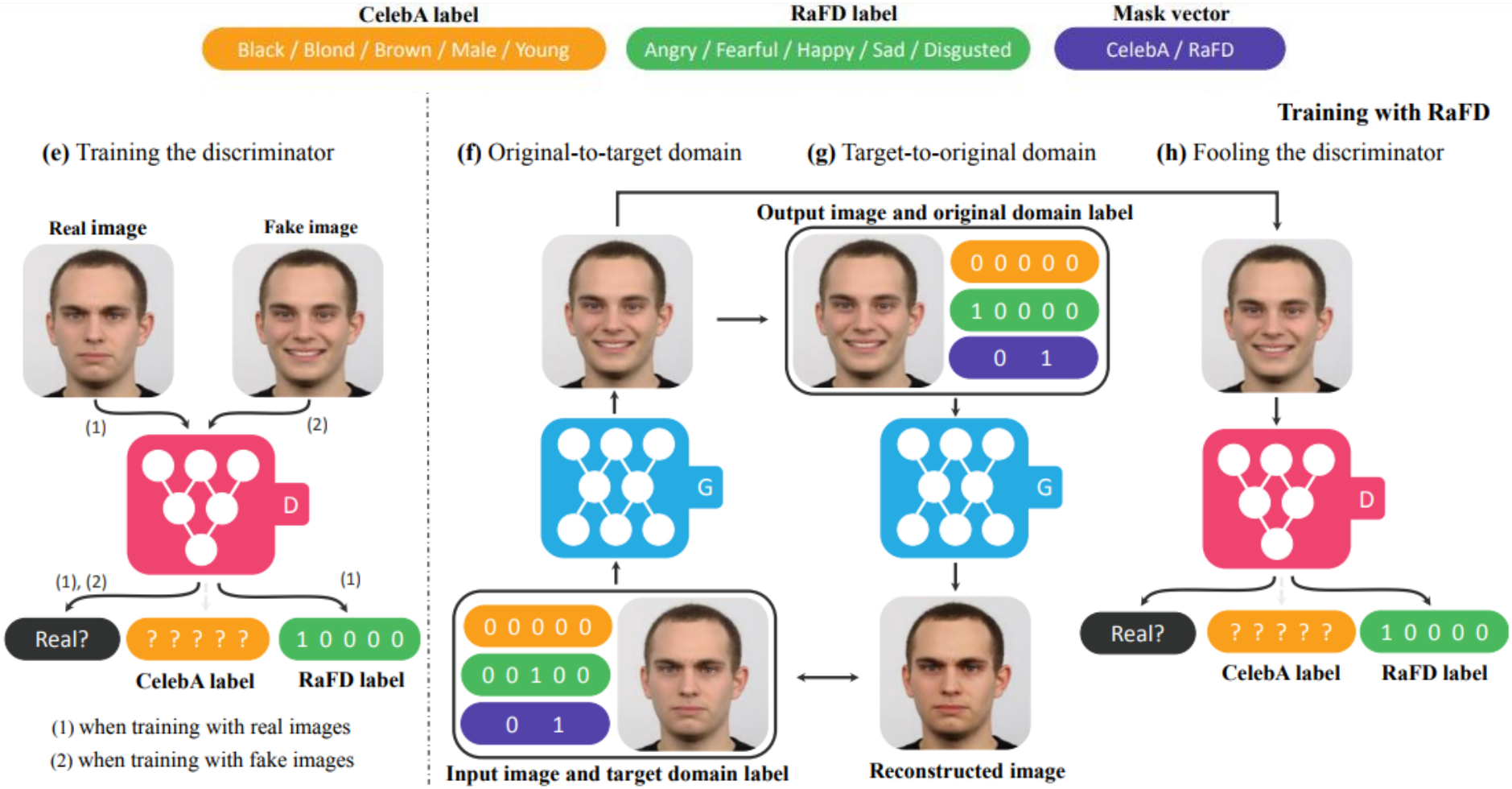
StarGAN


(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

StarGAN


CelebA label
Black / Blond / Brown / Male / Young
RaFD label
Angry / Fearful / Happy / Sad / Disgusted
Mask vector
CelebA / RaFD
(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?
0 0 1 0 1
? ? ? ? ?
CelebA label
RaFD label
(1) when training with real images
(2) when training with fake images
Output image and original domain label
0 0 1 0 1
0 0 0 0 0
1 0
G
G
1 0 0 1 1
0 0 0 0 0
1 0
Input image and target domain label
Reconstructed image
D
Real?
1 0 0 1 1
? ? ? ? ?
CelebA label
RaFD label
Training with CelebA

StarGAN


CelebA label
Black / Blond / Brown / Male / Young
RaFD label
Angry / Fearful / Happy / Sad / Disgusted
Mask vector
CelebA / RaFD
Training with RaFD
(e) Training the discriminator
(f) Original-to-target domain
(g) Target-to-original domain
(h) Fooling the discriminator
Real image
Fake image
(1)
(2)
D
(1), (2)
(1)
Real?
? ? ? ? ?
1 0 0 0 0
CelebA label
RaFD label
(1) when training with real images
(2) when training with fake images
Output image and original domain label
0 0 0 0 0
1 0 0 0 0
0 1
G
G
0 0 0 0 0
0 0 1 0 0
0 1
Input image and target domain label
Reconstructed image
D
Real?
? ? ? ? ?
1 0 0 0 0
CelebA label
RaFD label

Feature

Anders Boesen, Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, Ole Winther, “Autoencoding beyond pixels using a learned similarity metric”, ICML. 2016

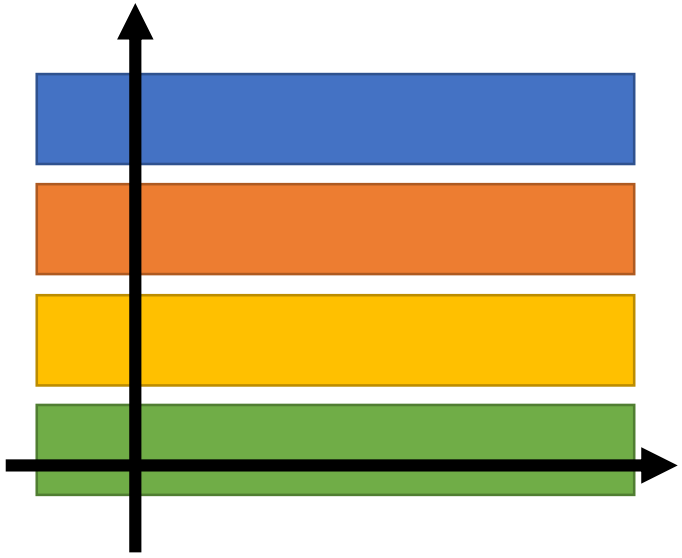
InfoGAN

(The colors represents the characteristics.)

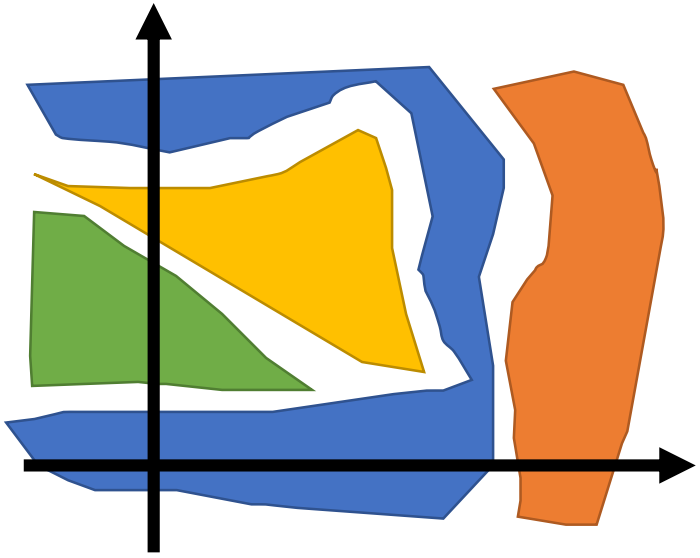
Regular GAN

Modifying a specific dimension, no clear meaning

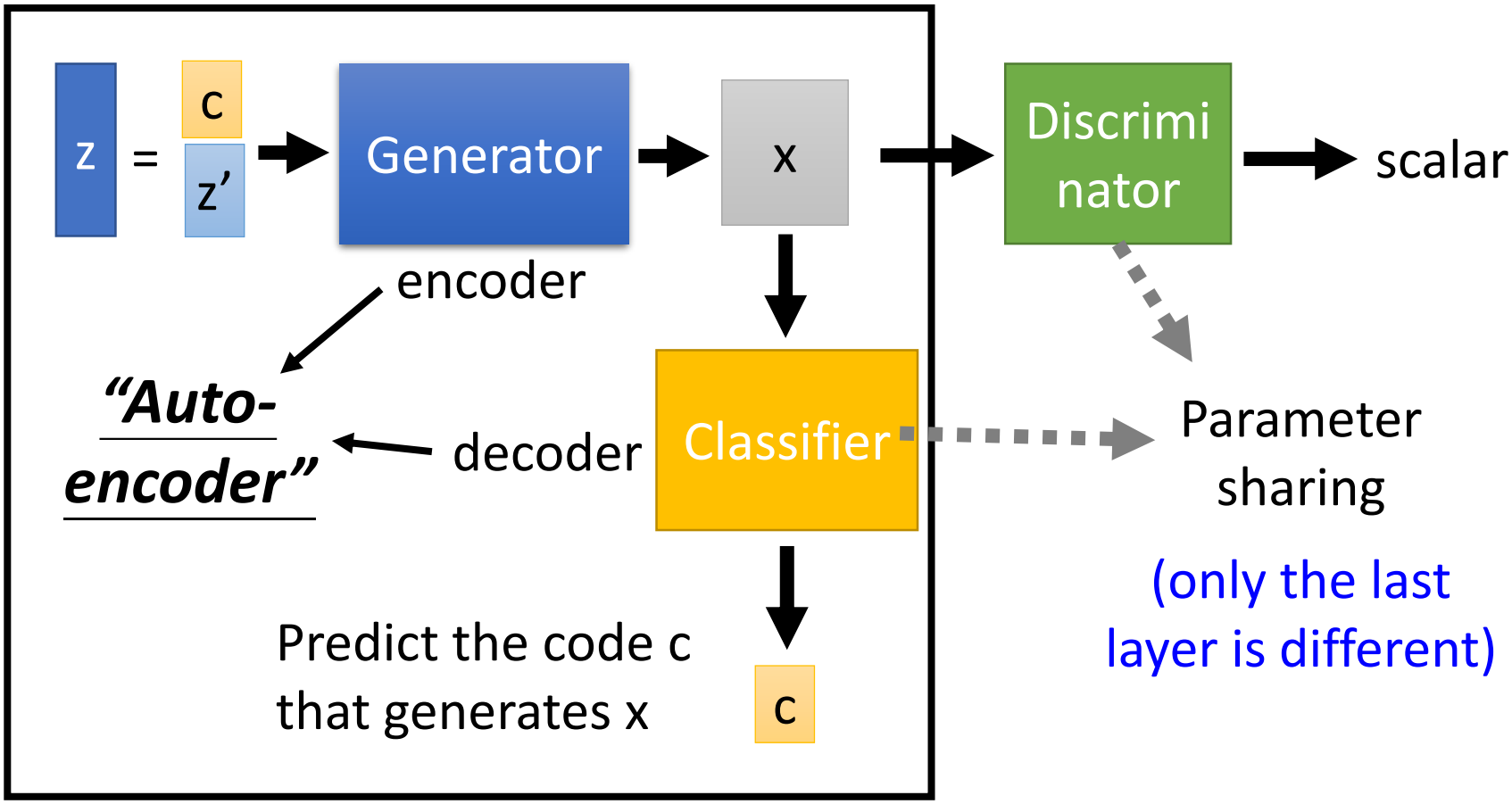
What we expect



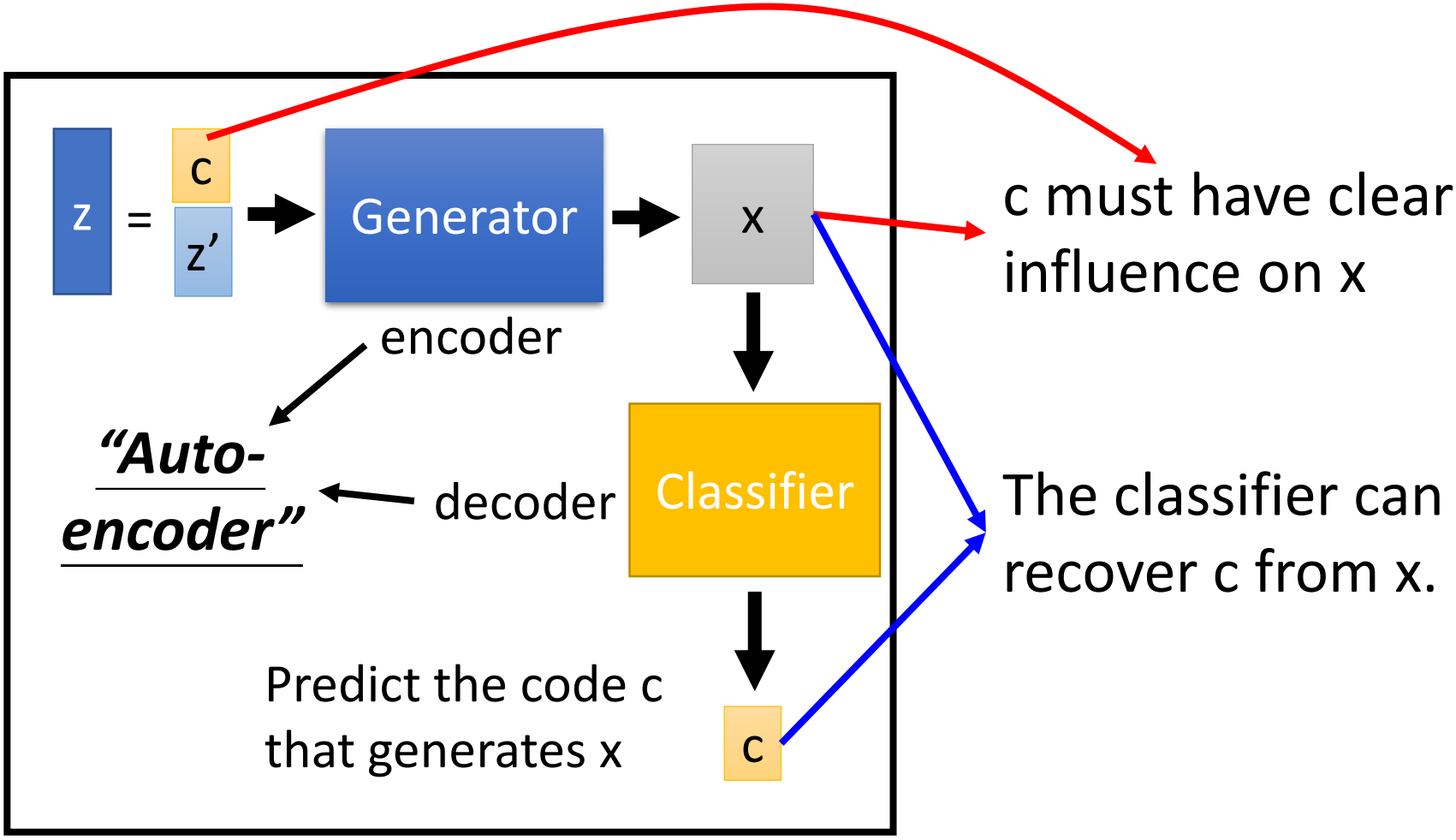
Actually …

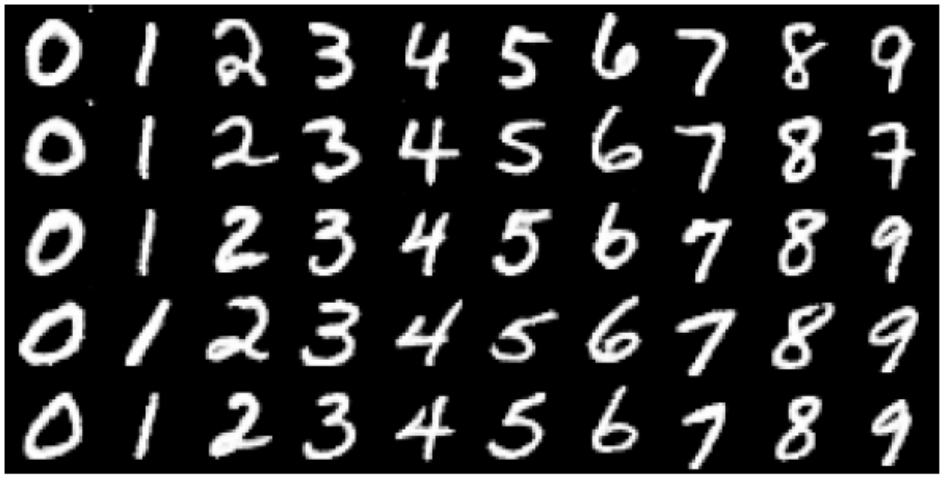


What is InfoGAN?

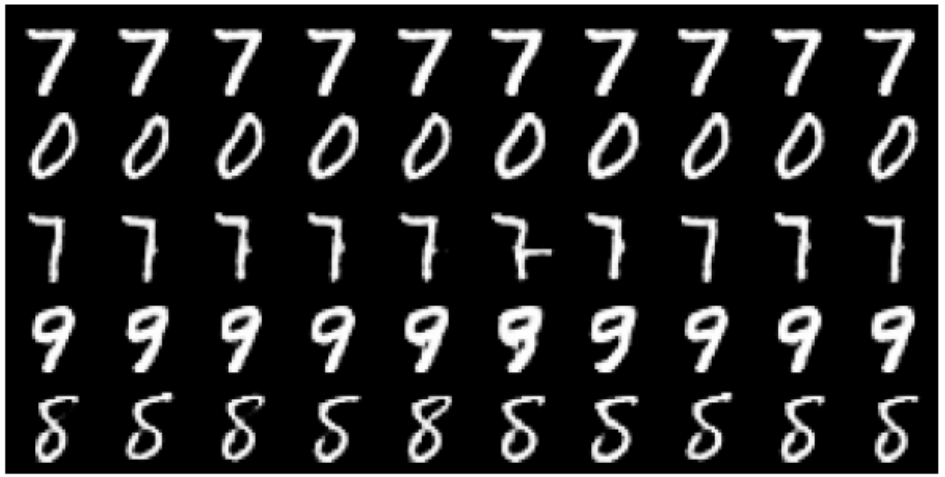


What is InfoGAN?

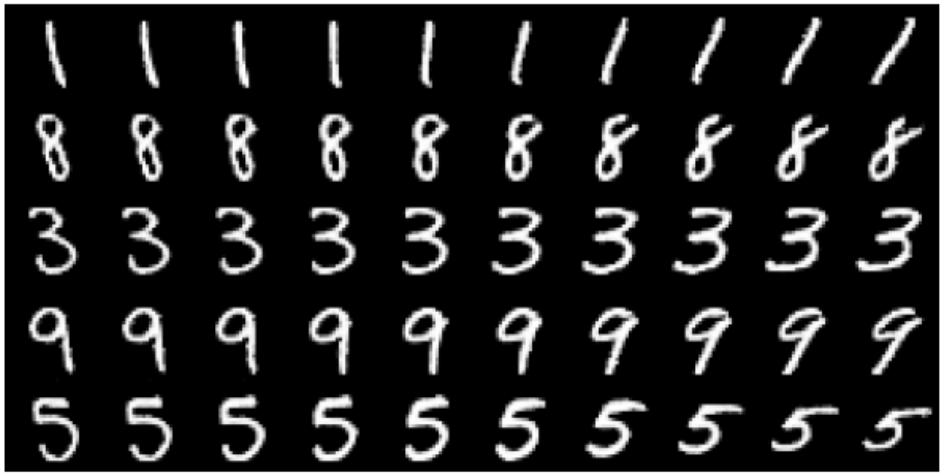




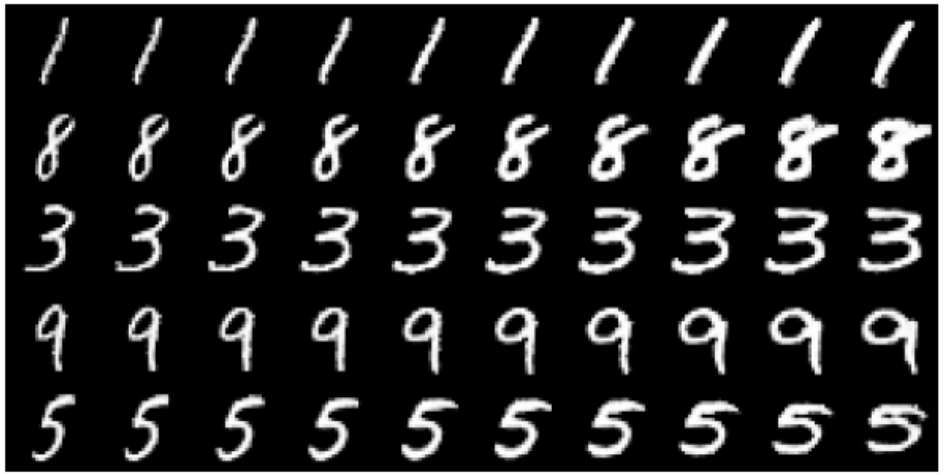
(a) Varying c_1 on InfoGAN (Digit type)



(b) Varying c_1 on regular GAN (No clear meaning)



(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)



(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

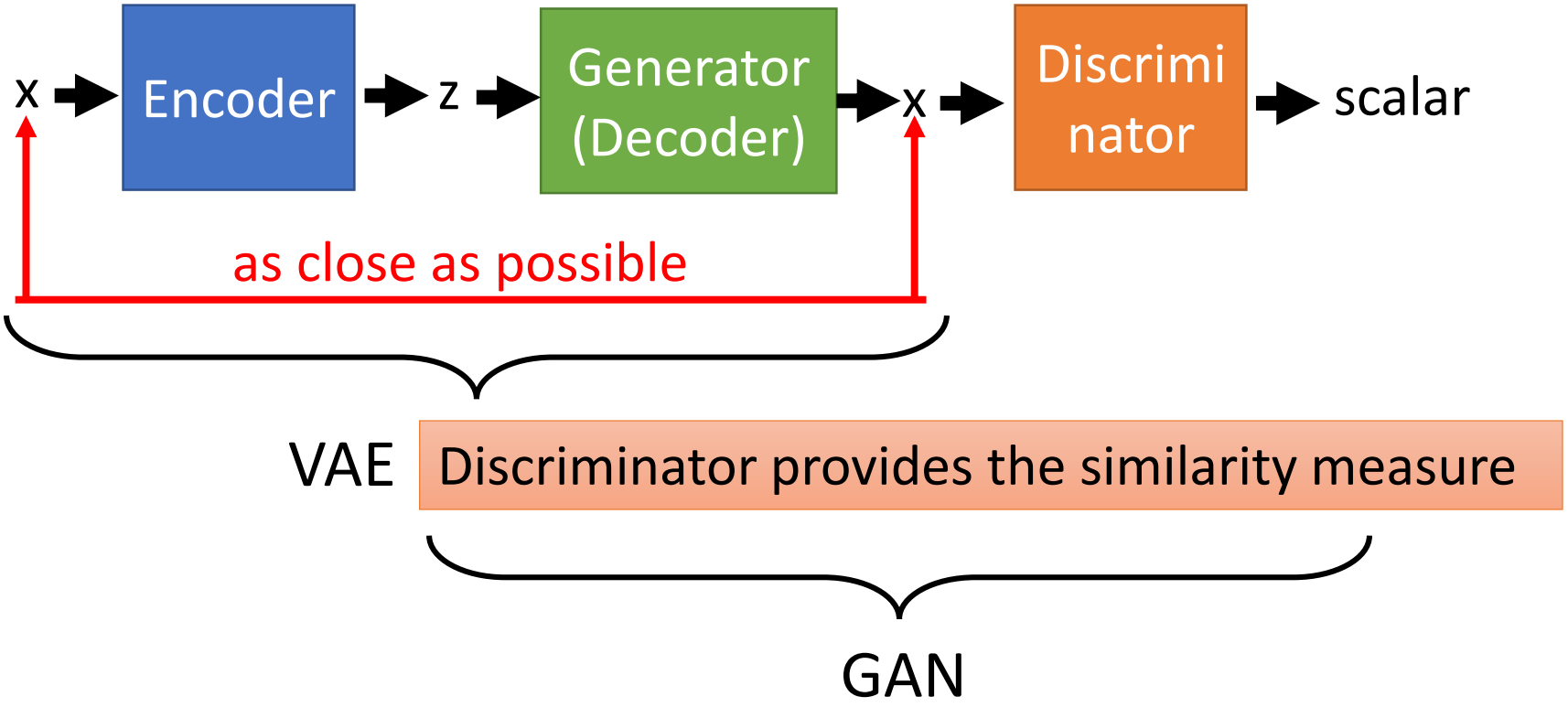
https://arxiv.org/abs/1606.03657

VAE-GAN

- Minimize reconstruction error
- z close to normal

- Minimize reconstruction error
- Cheat discriminator

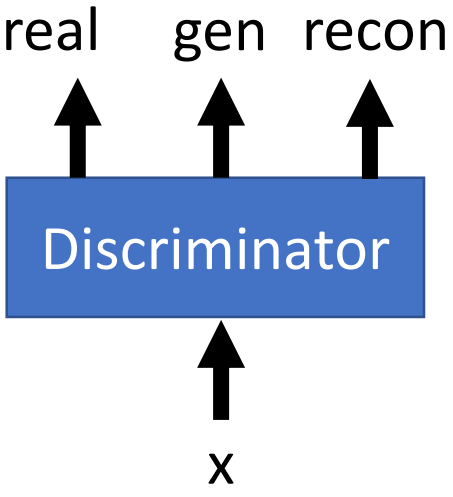
- Discriminate real, generated and reconstructed images



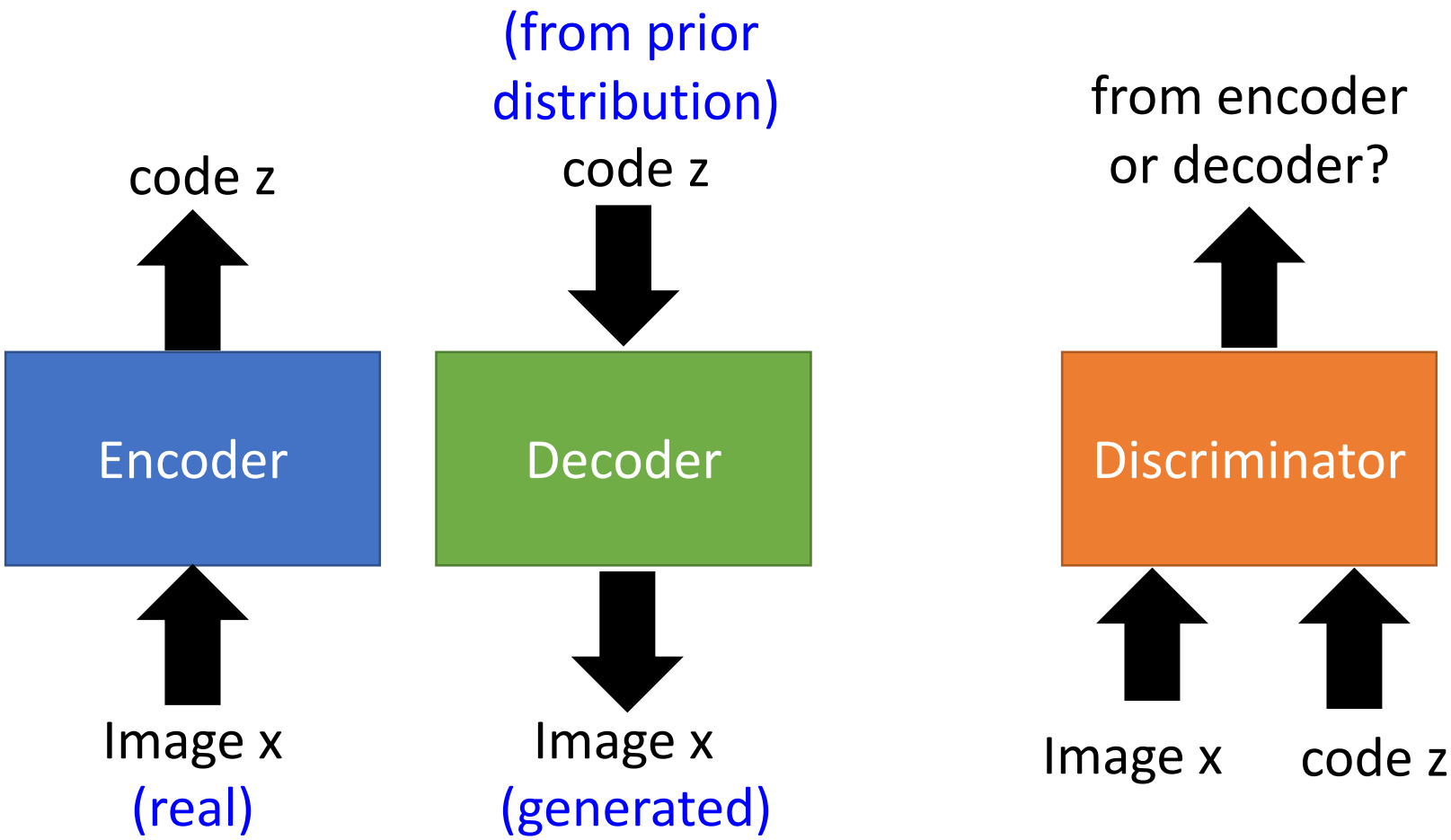
Algorithm

- Initialize En, De, Dis
- In each iteration:
  - Sample M images $x^1, x^2, \cdots, x^M$ from database
  - Generate M codes $\tilde{z}^1, \tilde{z}^2, \cdots, \tilde{z}^M$ from encoder
    - $\tilde{z}^i = En(x^i)$
  - Generate M images $\tilde{x}^1, \tilde{x}^2, \cdots, \tilde{x}^M$ from decoder
    - $\tilde{x}^i = En(\tilde{z}^i)$
  - Sample M codes $z^1, z^2, \cdots, z^M$ from prior P(z)
  - Generate M images $\hat{x}^1, \hat{x}^2, \cdots, \hat{x}^M$ from decoder
    - $\hat{x}^i = En(z^i)$
  - Update En to decrease $\|\tilde{x}^i - x^i\|$, decrease KL(P($\tilde{z}^i$|x^i)||P(z))
  - Update De to decrease $\|\tilde{x}^i - x^i\|$, increase $Dis(\tilde{x}^i)$ and $Dis(\hat{x}^i)$
  - Update Dis to increase $Dis(x^i)$, decrease $Dis(\tilde{x}^i)$ and $Dis(\hat{x}^i)$

Another kind of discriminator:

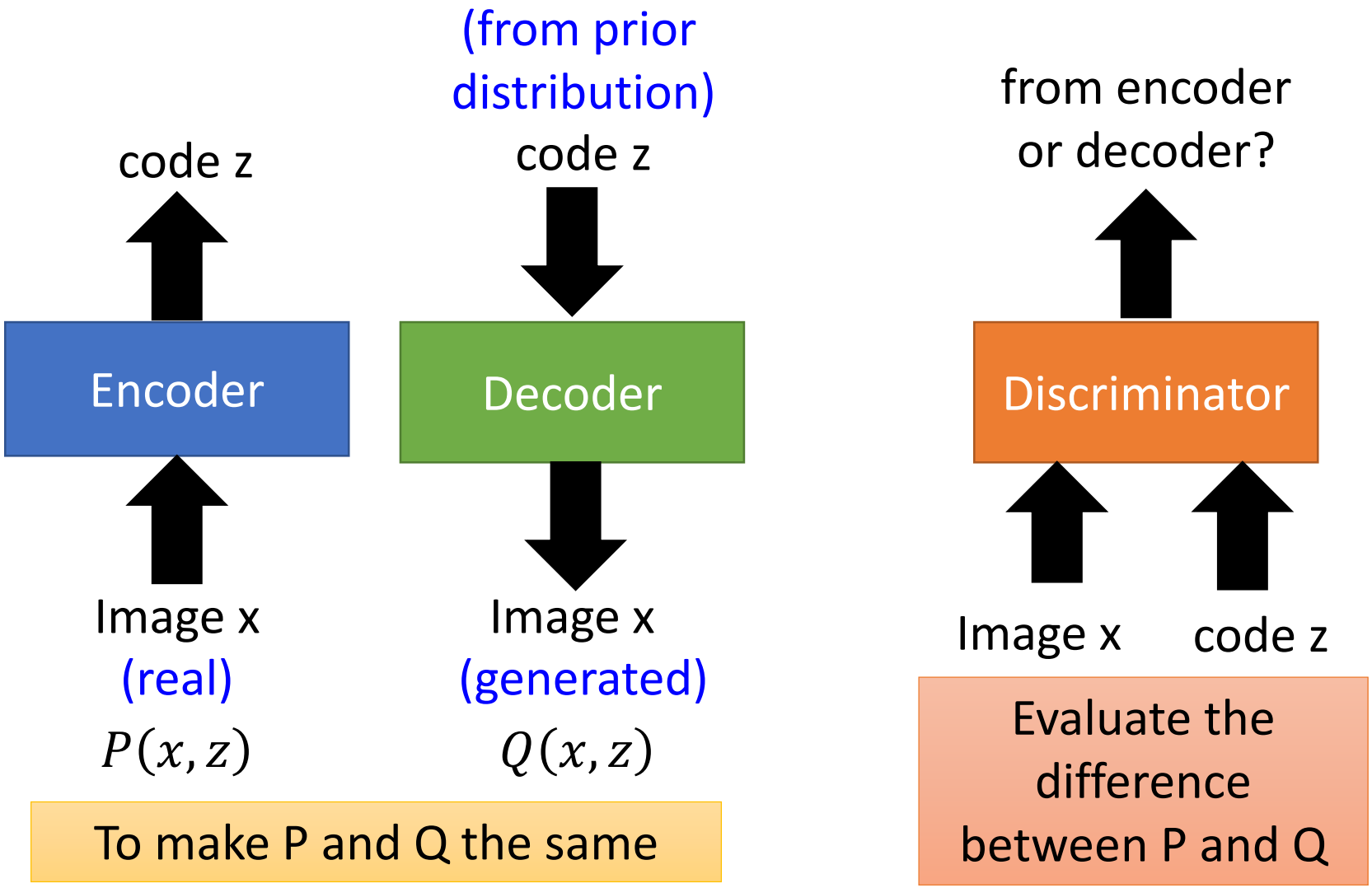


BiGAN


code z
Encoder
Image x
(real)
(from prior
distribution)
code z
Decoder
Image x
(generated)
from encoder
or decoder?
Discriminator
Image x
code z

Algorithm

- Initialize encoder En, decoder De, discriminator Dis
- In each iteration:
  - Sample M images $x^1, x^2, \cdots, x^M$ from database
  - Generate M codes $\tilde{z}^1, \tilde{z}^2, \cdots, \tilde{z}^M$ from encoder
    - $\tilde{z}^i = En(x^i)$
  - Sample M codes $z^1, z^2, \cdots, z^M$ from prior P(z)
  - Generate M codes $\tilde{x}^1, \tilde{x}^2, \cdots, \tilde{x}^M$ from decoder
    - $\tilde{x}^i = De(z^i)$
  - Update Dis to increase $Dis(x^i, \tilde{z}^i)$, decrease $Dis(\tilde{x}^i, z^i)$
  - Update En and De to decrease $Dis(x^i, \tilde{z}^i)$, increase $Dis(\tilde{x}^i, z^i)$



Optimal encoder and decoder:

En(x') = z' ⇒ De(z') = x' For all x'

De(z'') = x'' ⇒ En(x'') = z'' For all z''

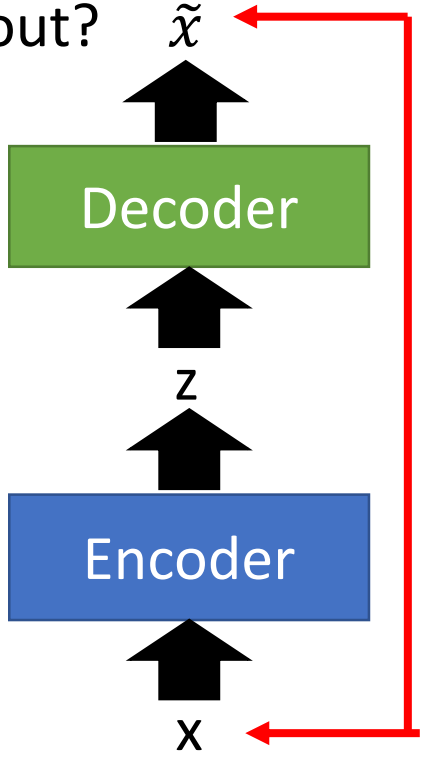
BiGAN

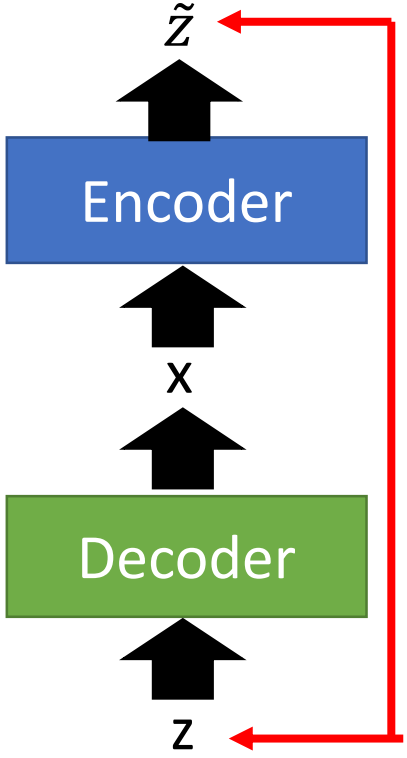
Optimal encoder and decoder:

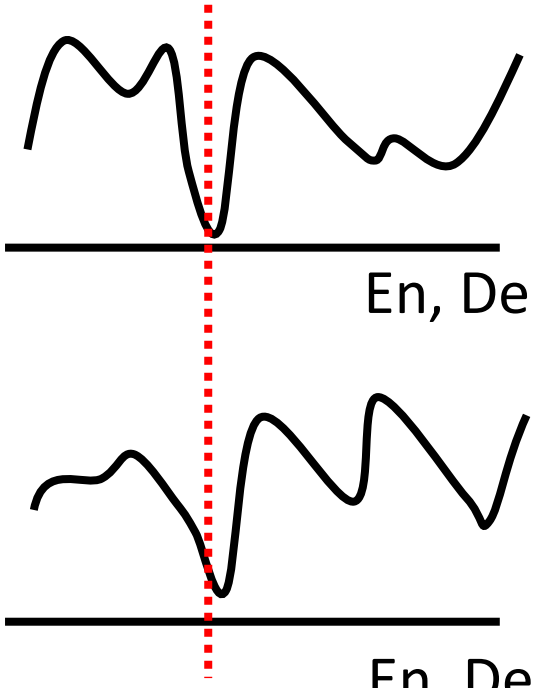
En(x') = z' → De(z') = x' For all x'

De(z'') = x'' → En(x'') = z'' For all z''

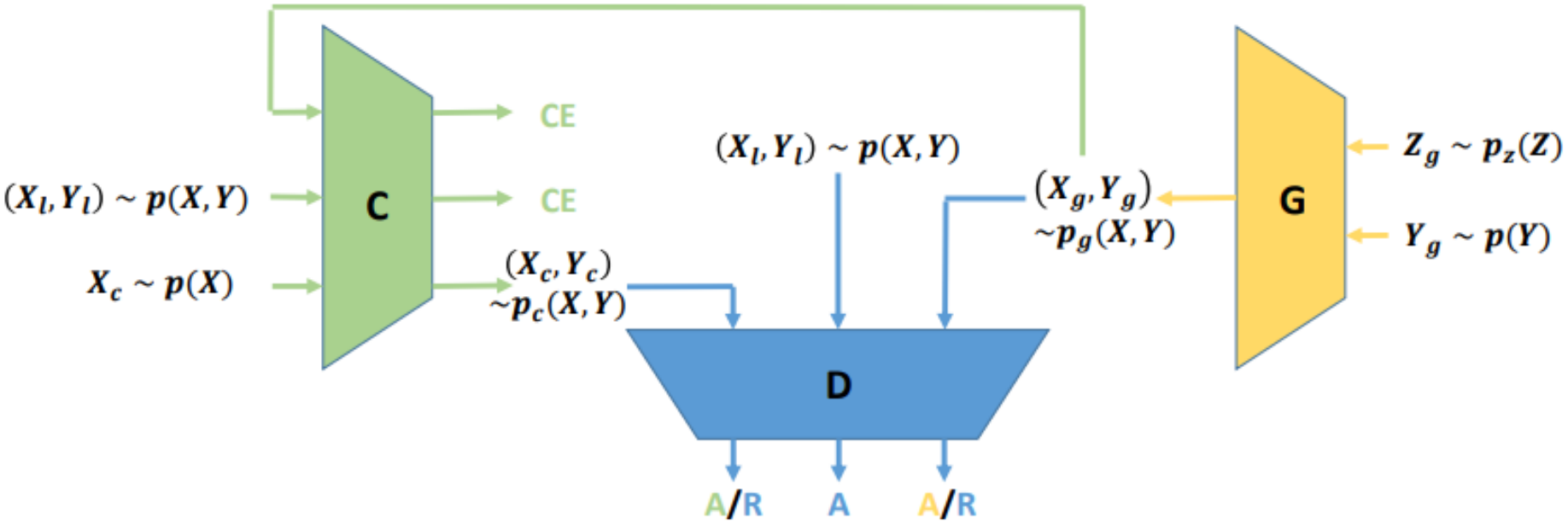
How about?







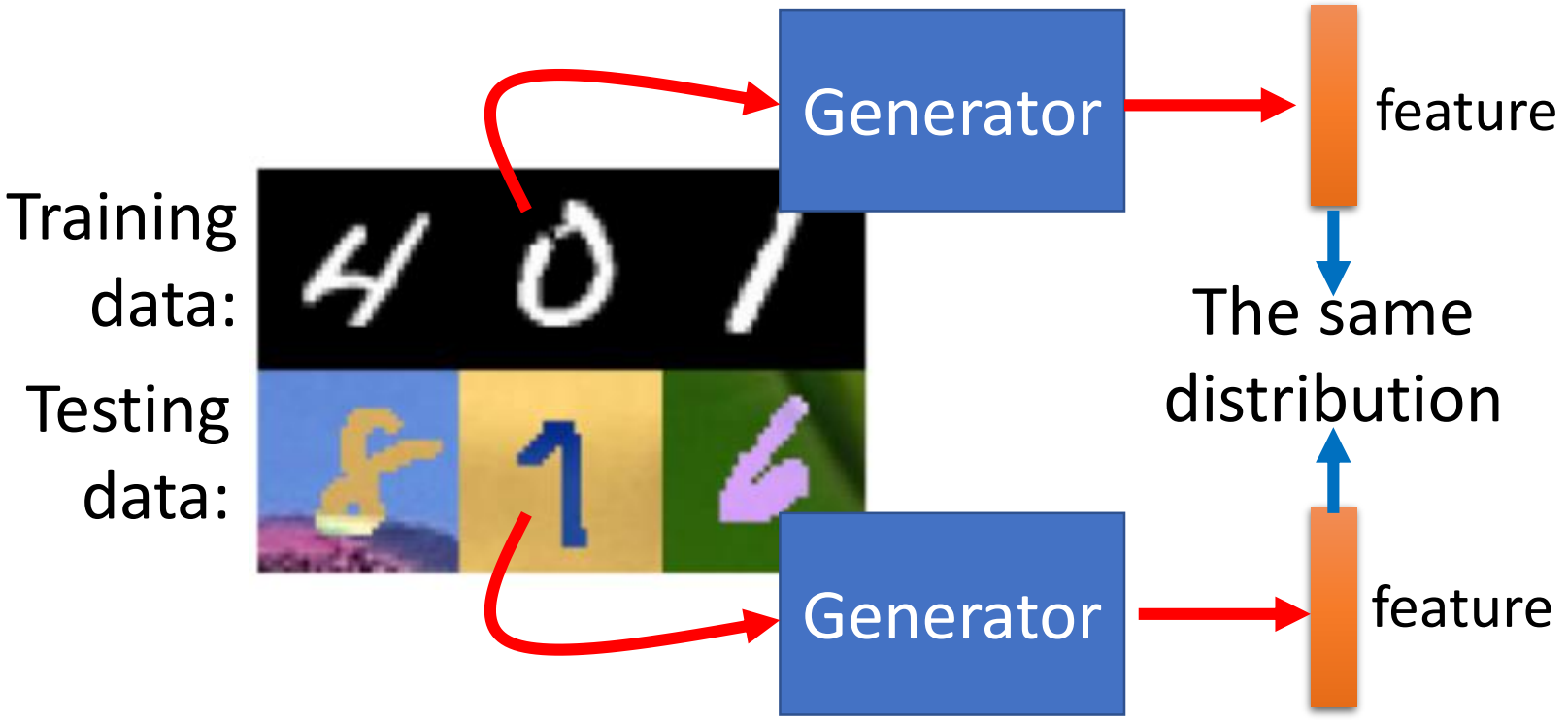
Triple GAN



Chongxuan Li, Kun Xu, Jun Zhu, Bo Zhang, “Triple Generative Adversarial Nets”, arXiv 2017

Domain-adversarial training

- Training and testing data are in different domains

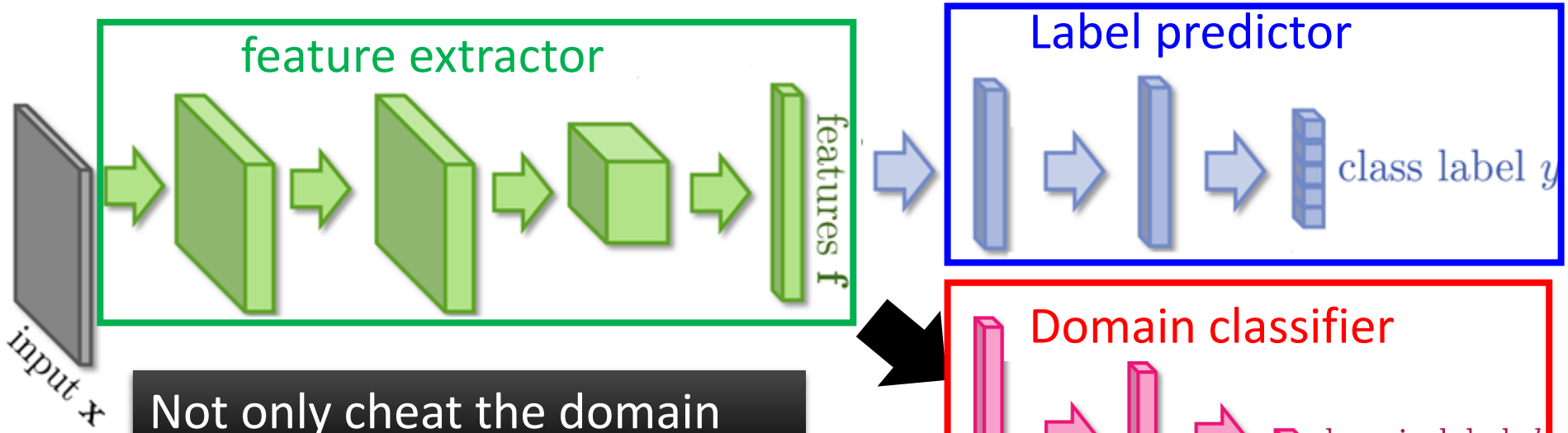


Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016

Domain-adversarial training

Maximize label classification accuracy + minimize domain classification accuracy

Maximize label classification accuracy



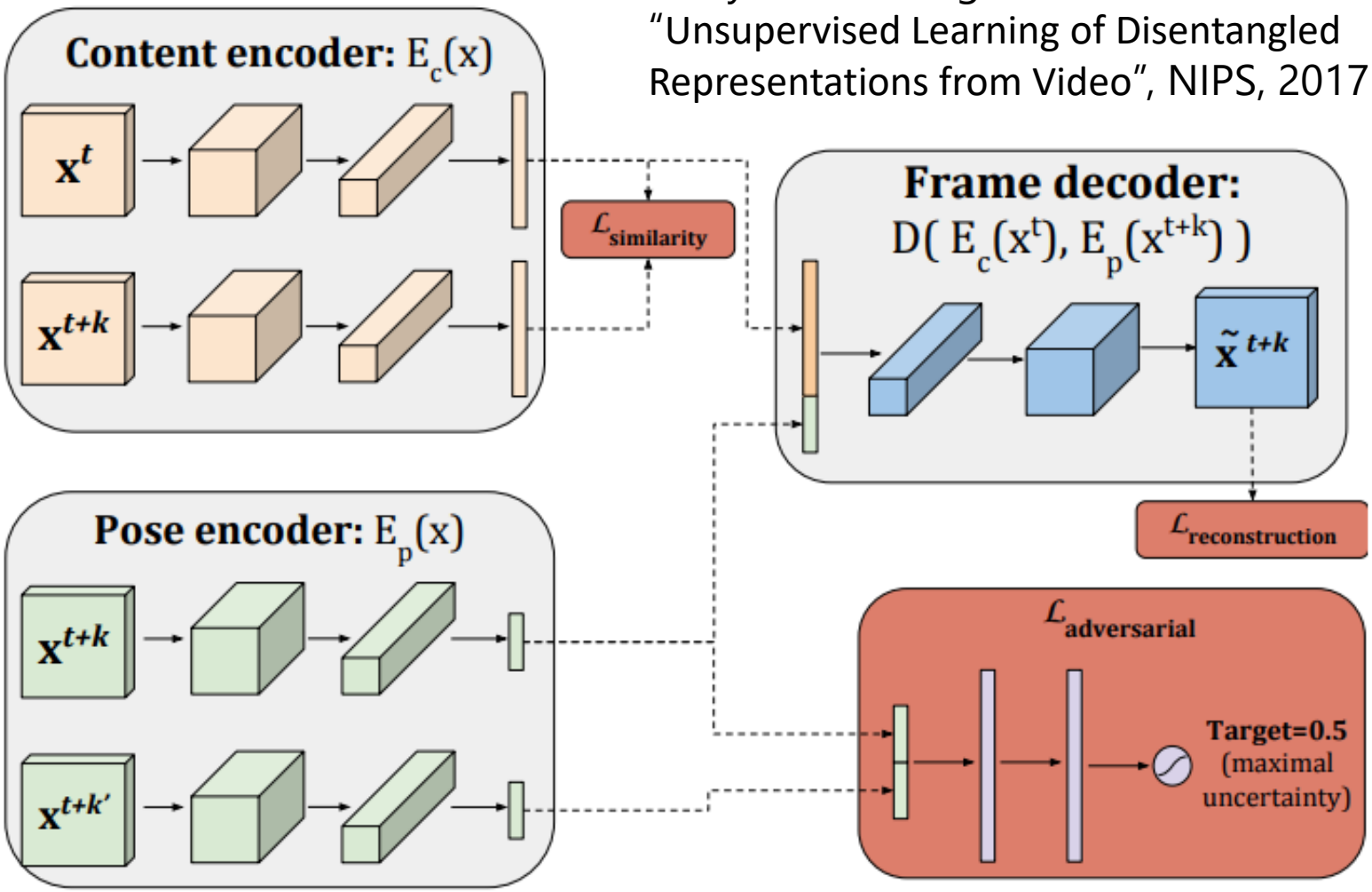
Not only cheat the domain classifier, but satisfying label classifier at the same time

Maximize domain classification accuracy

This is a big network, but different parts have different goals.

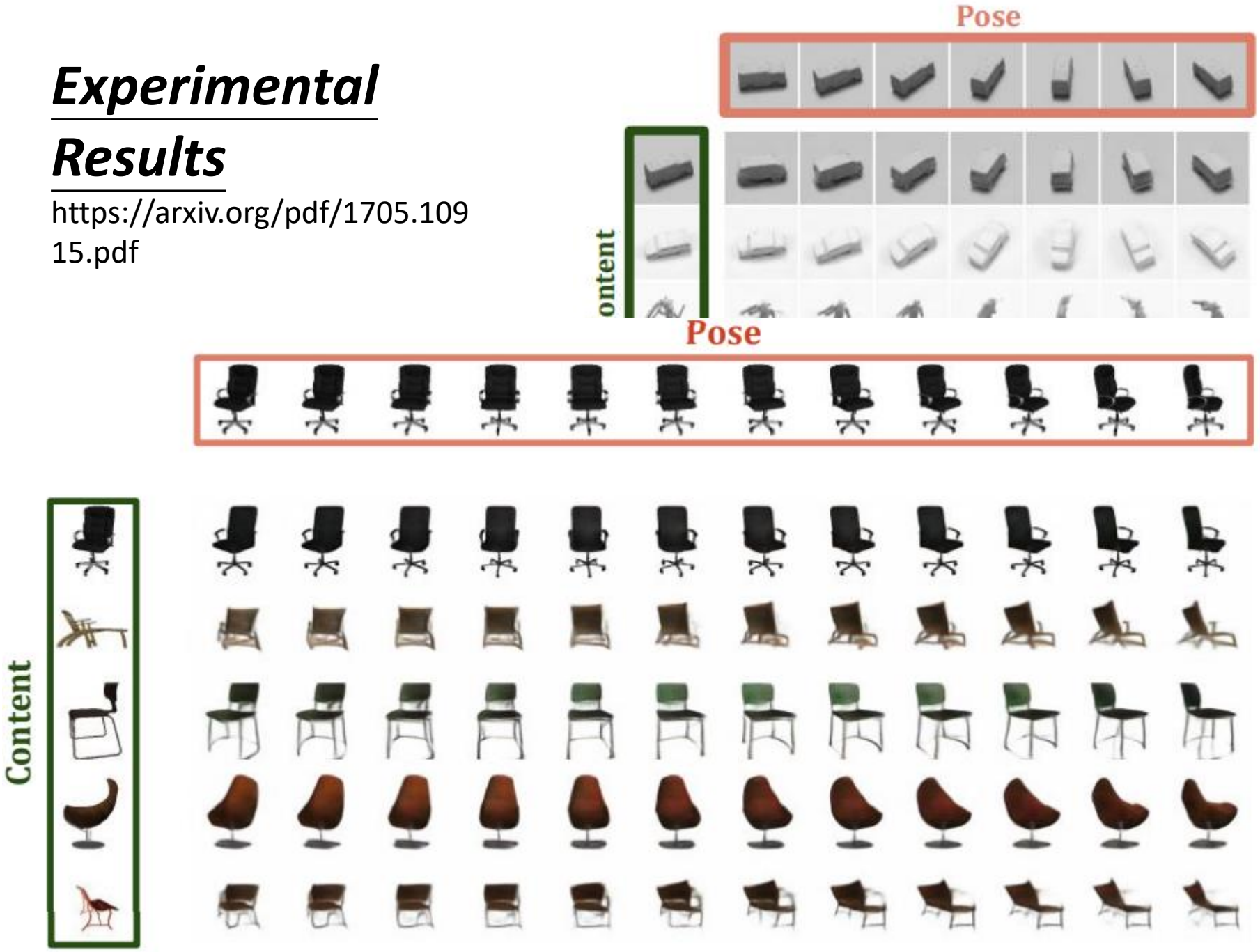
Feature Disentangle

Emily Denton, Vighnesh Birodkar, "Unsupervised Learning of Disentangled Representations from Video", NIPS, 2017



Experimental Results

https://arxiv.org/pdf/1705.10915.pdf



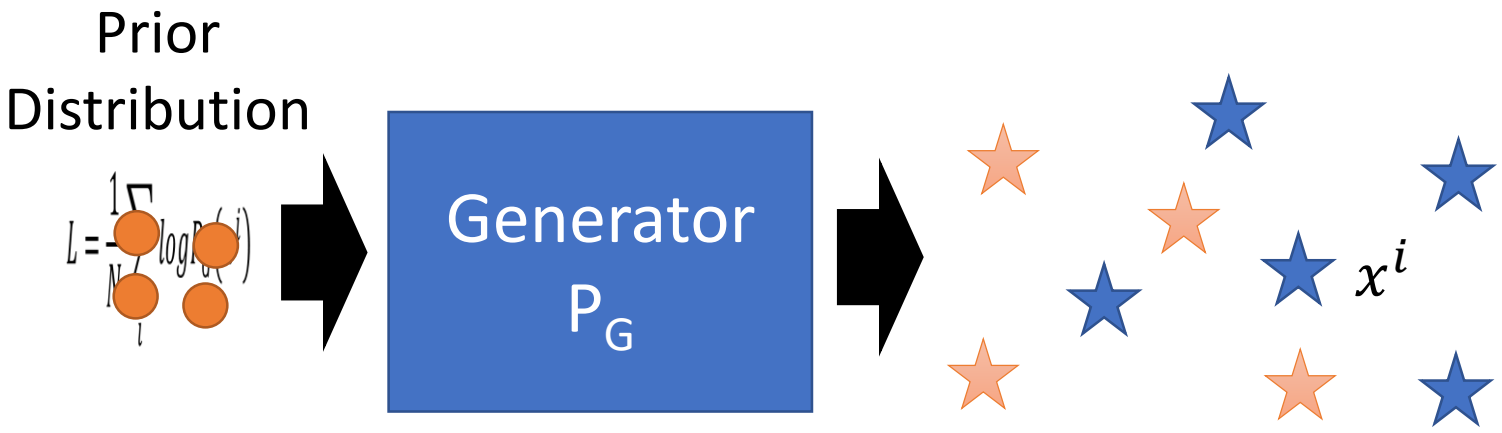
Evaluation

Ref: Lucas Theis, Aäron van den Oord, Matthias Bethge, “A note on the evaluation of generative models”, arXiv preprint, 2015

Likelihood

: real data (not observed during training)

: generated data

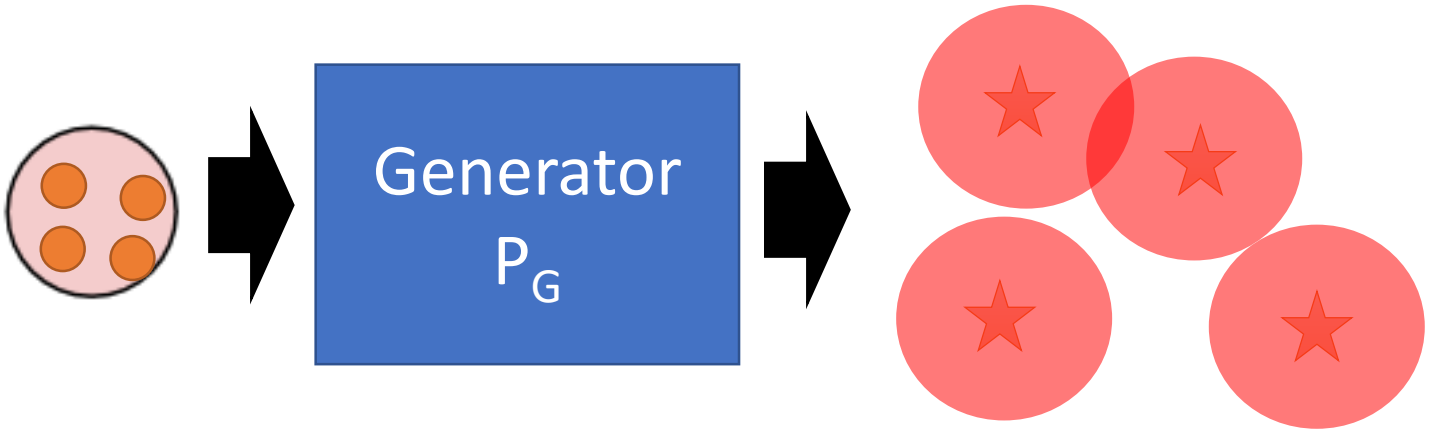


Log Likelihood: $L = \frac{1}{N}\sum_i log P_G(x^i)$

We cannot compute $P_G(x^i)$. We can only sample from P_G.

Likelihood
- Kernel Density Estimation

- Estimate the distribution of $P_G(x)$ from sampling



Each sample is the mean of a Gaussian with the same covariance.

Now we have an approximation of P_G, so we can compute $P_G(x^i)$ for each real data x^i

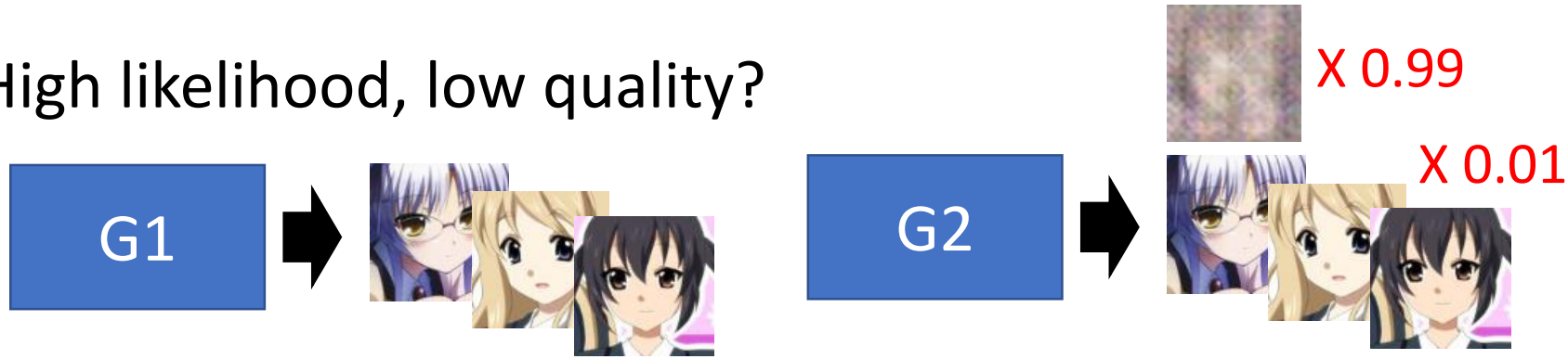
Then we can compute the likelihood.

Likelihood v.s. Quality

- Low likelihood, high quality?

  Considering a model generating good images (small variance)

- High likelihood, low quality?



$$L = \frac{1}{N}\sum_i log P_G(x^i) = -log100 + \frac{1}{N}\sum_i log P_G(x^i)$$

$\frac{P_G(x^i)}{100}$ 4.6

Evaluate by Other Networks

x → Well-trained CNN → $P(y|x)$

Lower entropy means higher visual quality

x^1 → CNN → $P(y^1|x^1)$

x^2 → CNN → $P(y^2|x^2)$

x^3 → CNN → $P(y^3|x^3)$

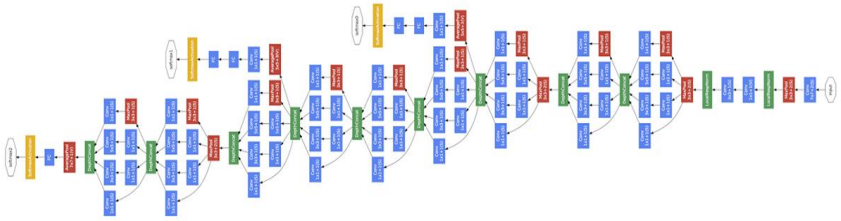
⋮

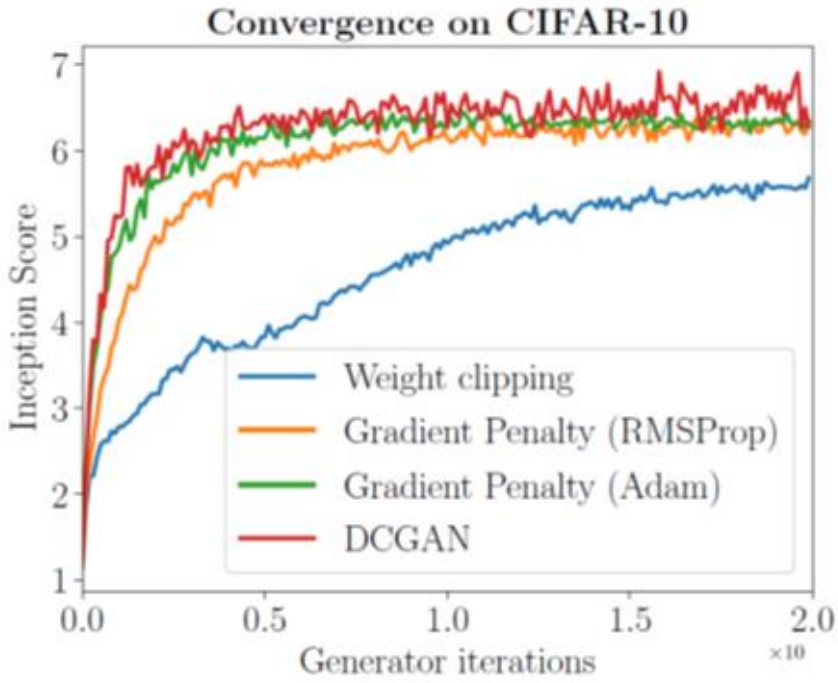
$$\frac{1}{N}\sum_{n} P(y^n|x^n)$$

Higher entropy means higher variety

https://arxiv.org/pdf/1606.03498.pdf

Evaluate by Other Networks - Inception Score



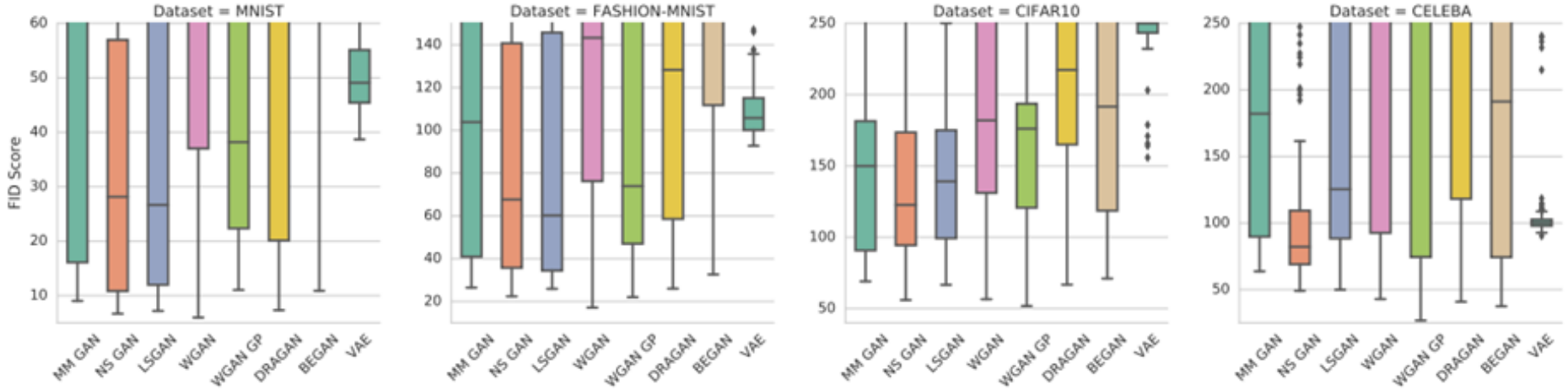


Inception Score

$$= \sum_x \sum_y P(y|x) log P(y|x)$$ Negative entropy of P(y|x)

$$- \sum_y P(y) log P(y)$$ Entropy of P(y)

GAN	Discriminator Loss	Generator Loss
MM GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] + \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{GAN}} = -\mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}}$
NS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{NSGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}}$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{NSGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} - = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}}$
WGAN GP	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\|\nabla D(\alpha x + (1-\alpha\hat{x})\|_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x} - 1)^2]$
DRAGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{DRAGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d+\mathcal{N}(0,c)}[(\|\nabla D(\hat{x})\|_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{DRAGAN}} = -\mathcal{L}_{\mathrm{D}}^{\mathrm{NS\ GAN}}$
BEGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{BEGAN}} = \mathbb{E}_{x\sim p_d}[\|\|x - \mathrm{AE}(x)\|\|_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \mathrm{AE}(\hat{x})\|\|_1]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \mathrm{AE}(\hat{x})\|\|_1]$



Smaller is better

FIT:
https://arxiv.org/pdf/1706.08500.pdf

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, Olivier Bousquet, "Are GANs Created Equal? A Large-Scale Study", arXiv, 2017

Mode Collapse

Missing Mode ?

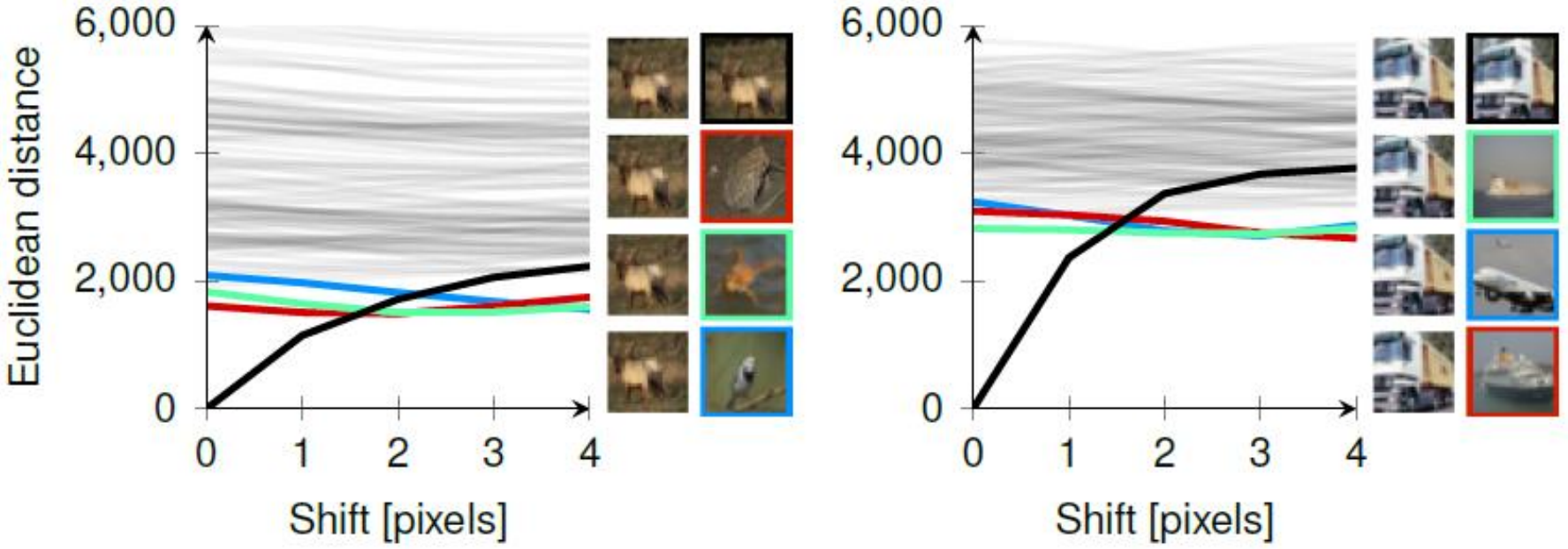
Mode collapse is easy to detect.

?

https://arxiv.org/pdf/1511.01844.pdf

We don’t want memory GAN.

- Using k-nearest neighbor to check whether the generator generates new objects



Concluding Remarks

from A to Z

[2]We use the Greek α prefix for α-GAN, as AEGAN and most other Latin prefixes seem to have been taken https://deephunt.in/the-gan-zoo-79597dc8c347.

Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, Shakir Mohamed, "Variational Approaches for Auto-Encoding Generative Adversarial Networks", arXiv, 2017

A	B	C	D	E	F
ACGAN	BiGAN	CycleGAN Couple GAN	DCGAN DuelGAN	EBGAN	fGAN

G	H	I	J	K	L
GAP	?	InfoGAN	?	?	LSGAN

(only list those mentioned in class)

M MMGAN

N NSGAN

O ?

P Progressive GAN

Q ?

R Rank GAN

S StackGAN StarGAN

T Triple GAN

U Unroll GAN

V VAEGAN

W WGAN

X ?

Y ?

Z ?