Applied Deep Learning

Guest Lecture by Hung-yi Lee

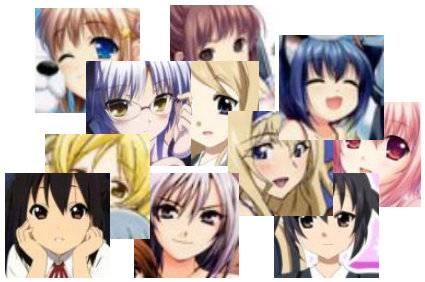
Generative Adversarial Network

May 19th, 2020 http://adl.miulab.tw

National Taiwan University

Three Categories of GAN

1. Typical GAN



$$\begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$

random vector

Generator

image

2. Conditional GAN

blue eyes,
red hair,
short hair

paired data

"Girl with red hair"

text

Generator

image

3. Unsupervised Conditional GAN

domain x

domain y

unpaired data

x

Photo

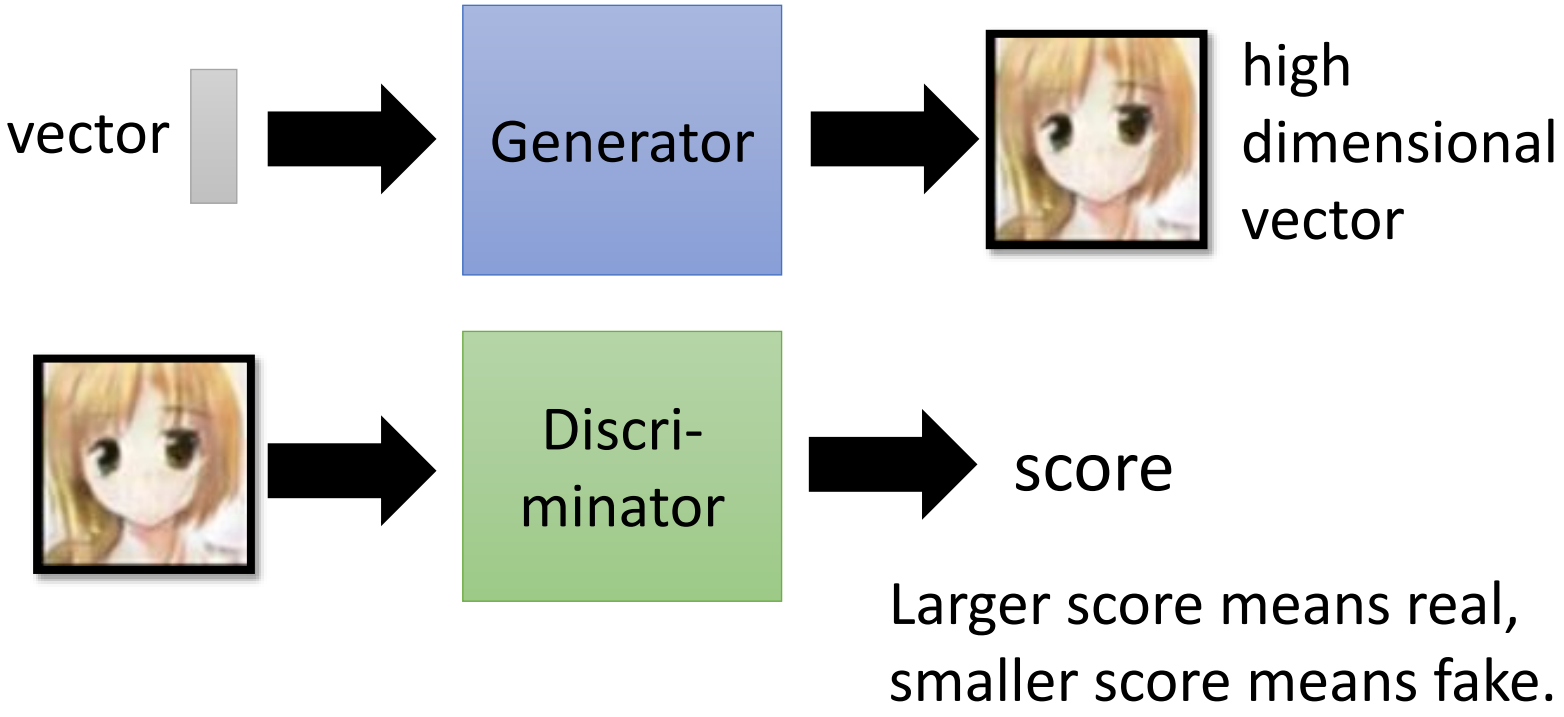
Generator

y

Vincent van Gogh's style

Generative Adversarial Network (GAN)

- Anime face generation as example

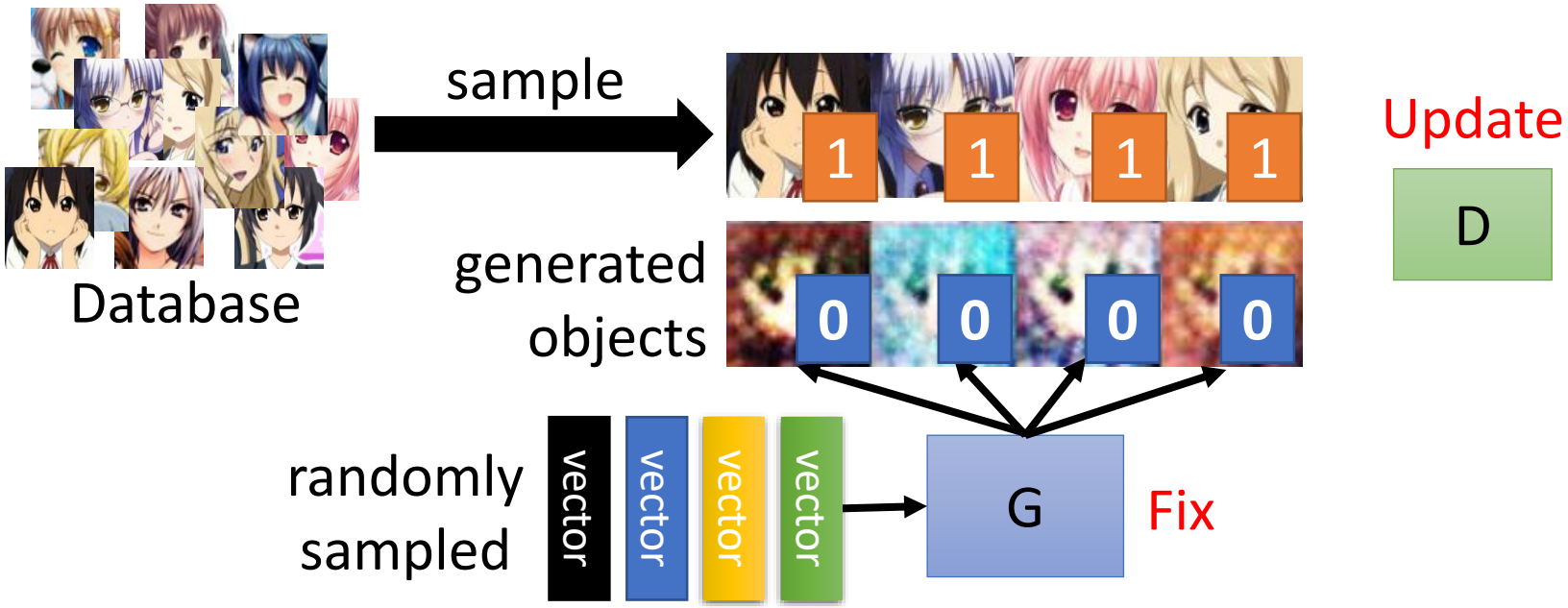


Algorithm

- Initialize generator and discriminator

- In each training iteration:

Step 1: Fix generator G, and update discriminator D



Discriminator learns to assign high scores to real objects and low scores to generated objects.

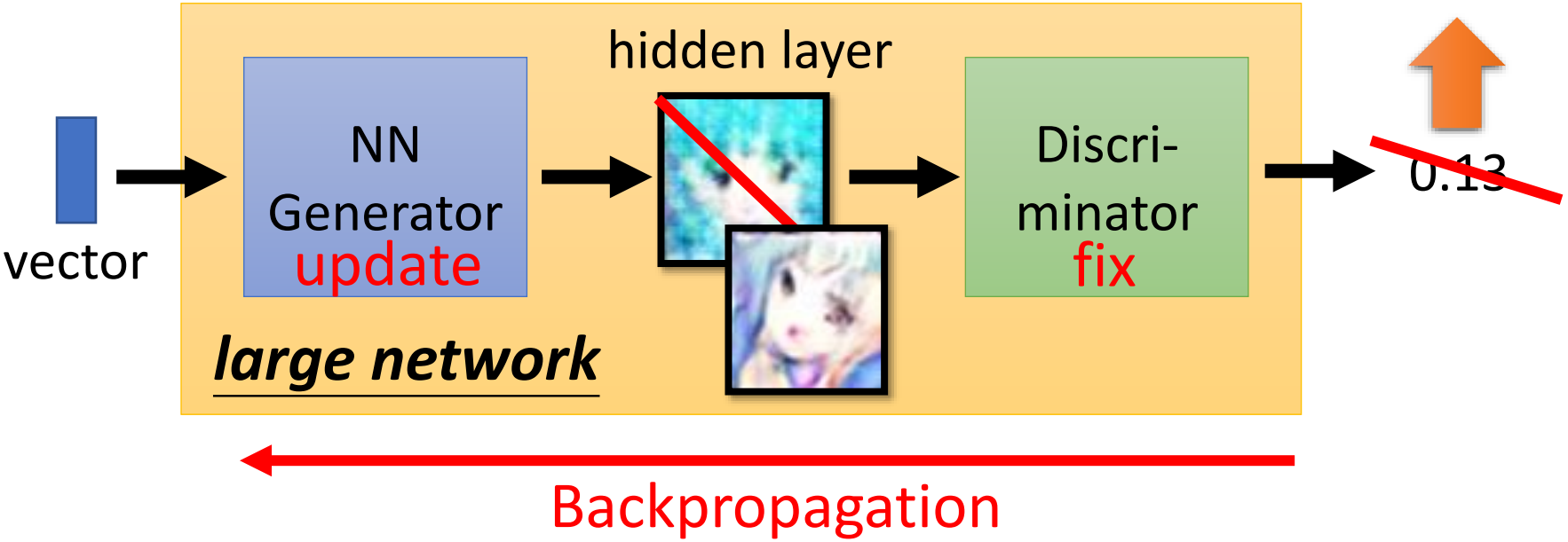
Algorithm

- Initialize generator and discriminator
- In each training iteration:

G D

Step 2: Fix discriminator D, and update generator G

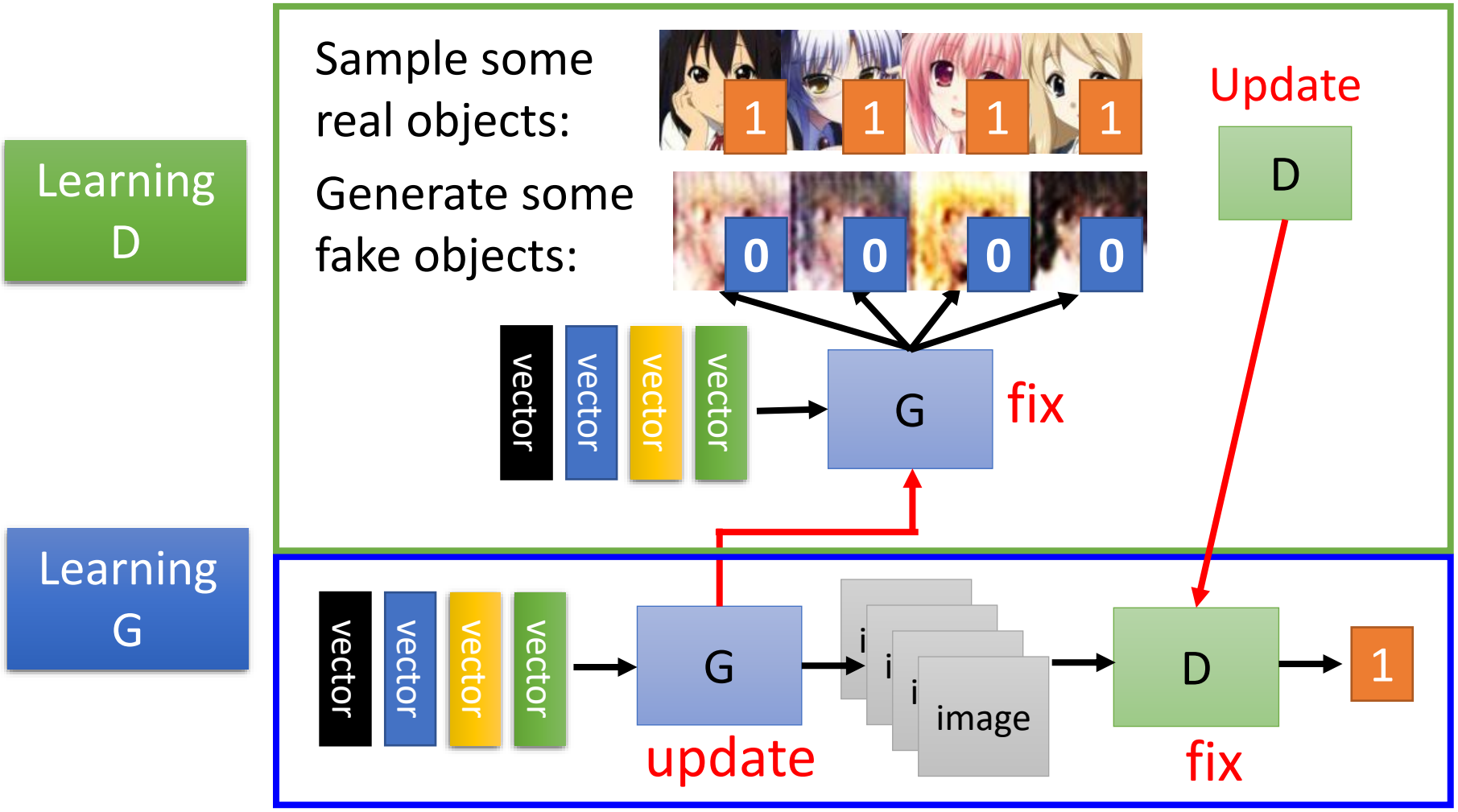
Generator learns to "*fool*" the discriminator



Algorithm

- Initialize generator and discriminator

- In each training iteration:



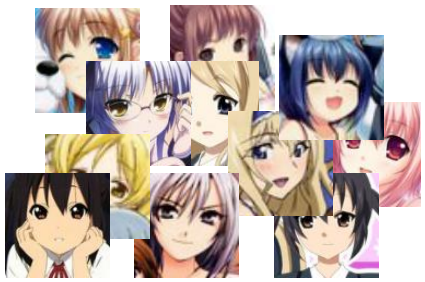
https://crypko.ai/#/

GAN is hard to train ……

(I found this joke from 陳柏文's facebook.)

Three Categories of GAN

1. Typical GAN



$$\begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$

random vector

→ Generator →

image

2. Conditional GAN

blue eyes,
red hair,
short hair

paired data

"Girl with red hair"

text

→ Generator →

image

3. Unsupervised Conditional GAN

domain x

domain y

unpaired data

x

Photo

→ Generator →

y

Vincent van Gogh's style

Text-to-Image

a dog is running

a bird is flying

- **Traditional supervised approach**

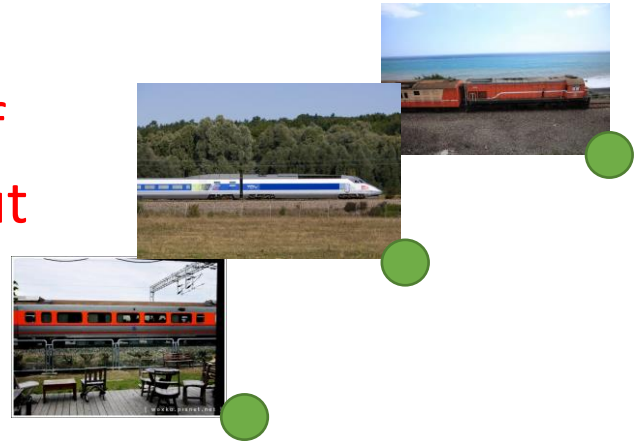
c^1: a dog is running → NN → Image ↔ 

as close as possible

Text: “train”

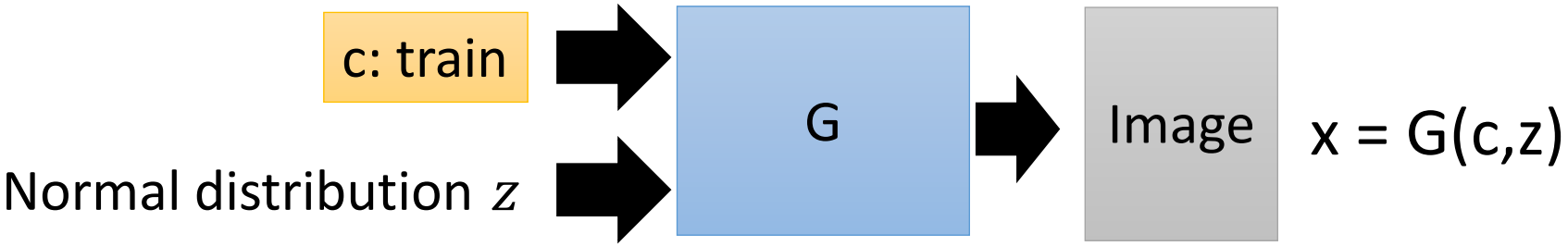
Target of NN output

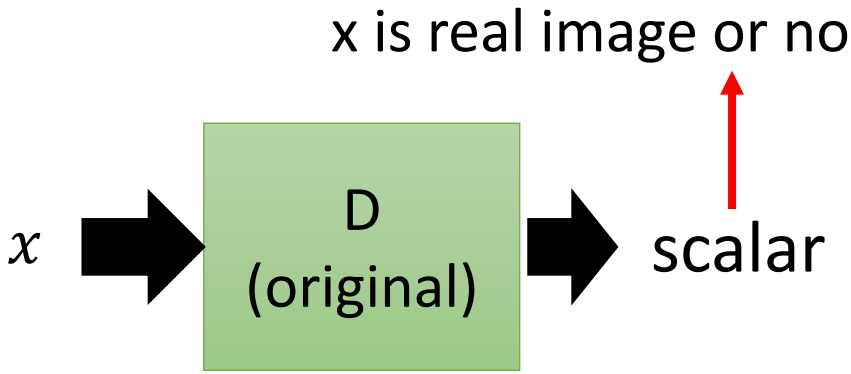
A blurry image!



[Scott Reed, et al, ICML, 2016]

Conditional GAN



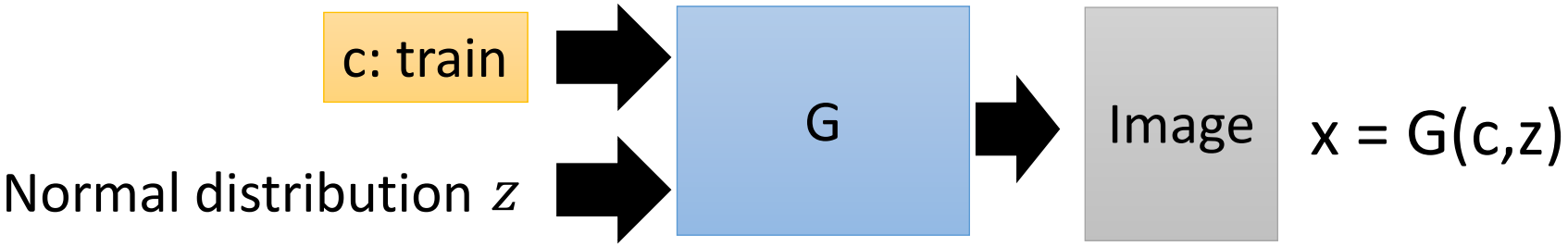


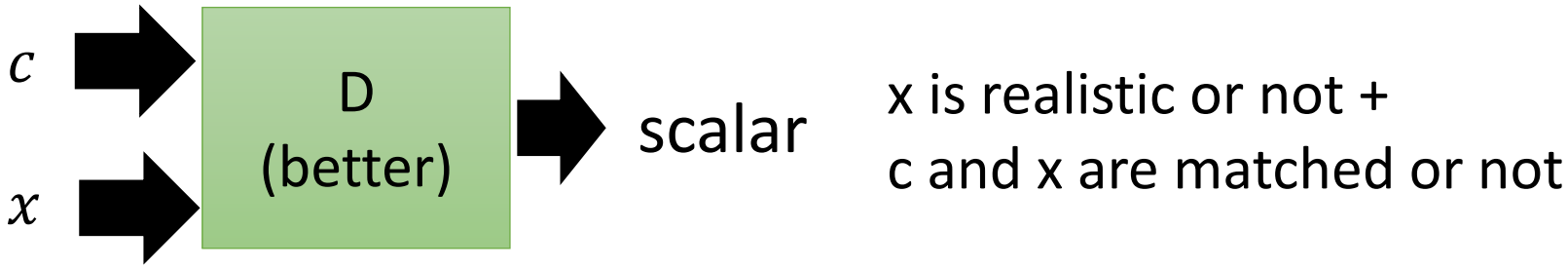
Generator will learn to generate realistic images ….

But completely ignore the input conditions.

[Scott Reed, et al, ICML, 2016]

Conditional GAN





True text-image pairs: (train , ) 1

(cat , ) 0

(train , Image) 0

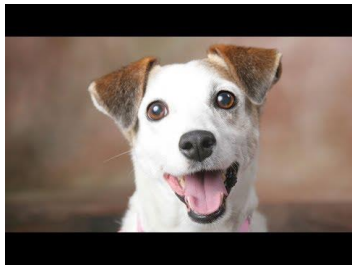
Conditional GAN - Sound-to-image

c: sound

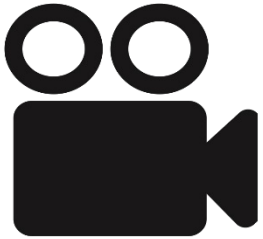
"a dog barking sound"

G

Image



Training Data Collection



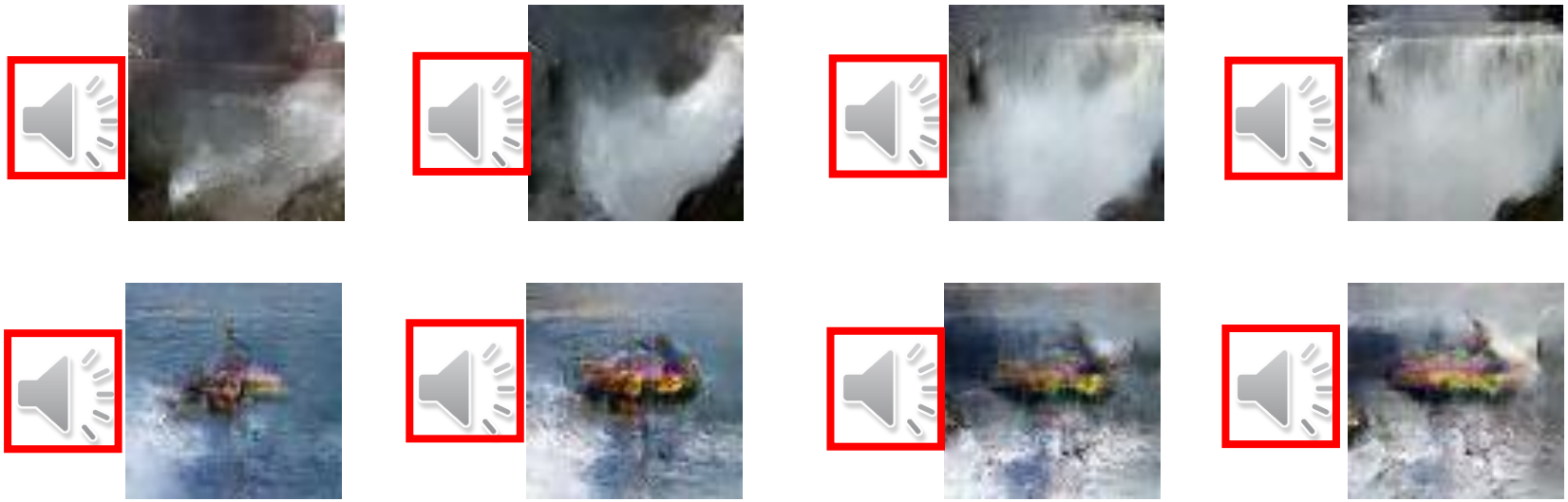
video

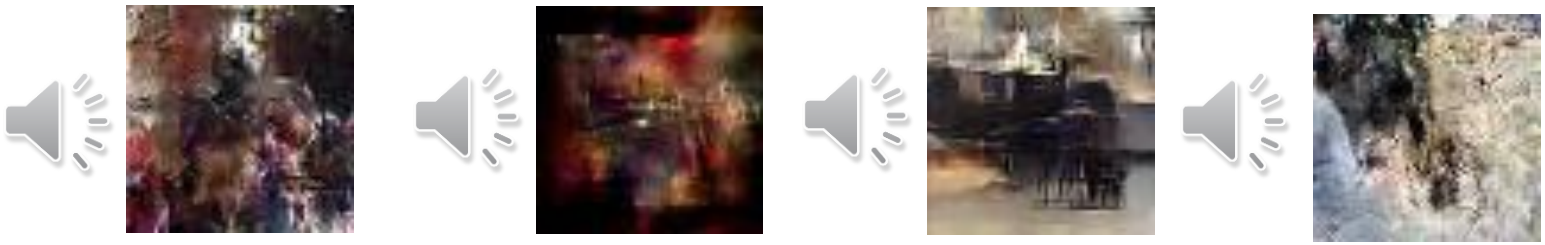
Conditional GAN - Sound-to-image

The images are generated by Chia-Hung Wan and Shun-Po Chuang.
https://wjohn1483.github.io/audio_to_scene/index.html

- Audio-to-image

Louder





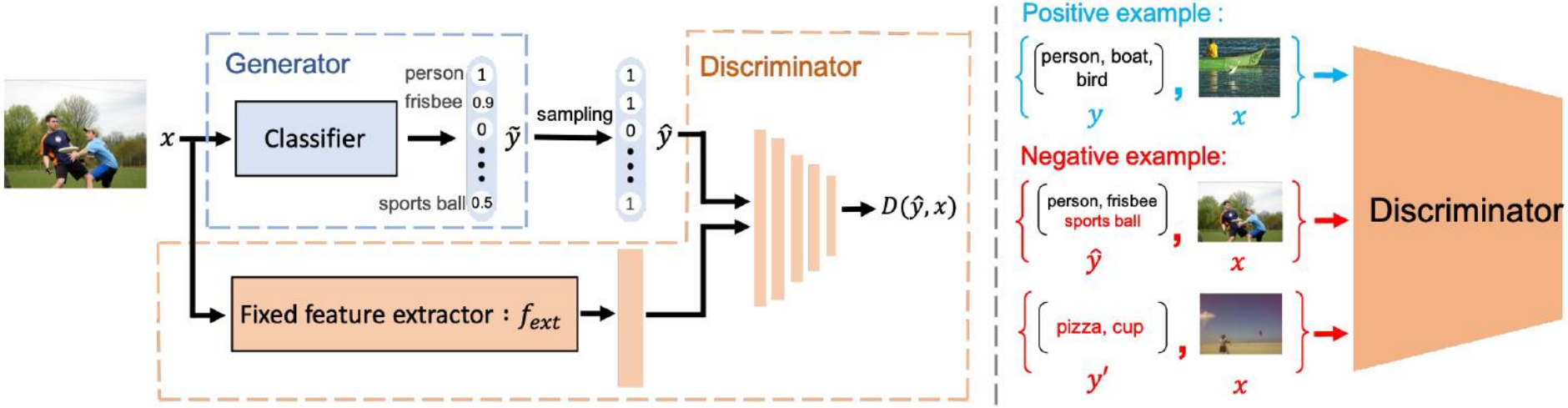
Conditional GAN - Image-to-label

Multi-label Image Classifier

person, sports ball,
baseball bat, baseball glove

Input condition

Generated output



Conditional GAN - Image-to-label

The classifiers can have different architectures.

The classifiers are trained as conditional GAN.

F1	MS-COCO	NUS-WIDE
VGG-16	56.0	33.9
+ GAN	60.4	41.2
Inception	62.4	53.5
+GAN	63.8	55.8
Resnet-101	62.8	53.1
+GAN	64.0	55.4
Resnet-152	63.3	52.1
+GAN	63.9	54.1
Att-RNN	62.1	54.7
RLSD	62.0	46.9

[Tsai, et al., submitted to ICASSP 2019]

Conditional GAN - Image-to-label

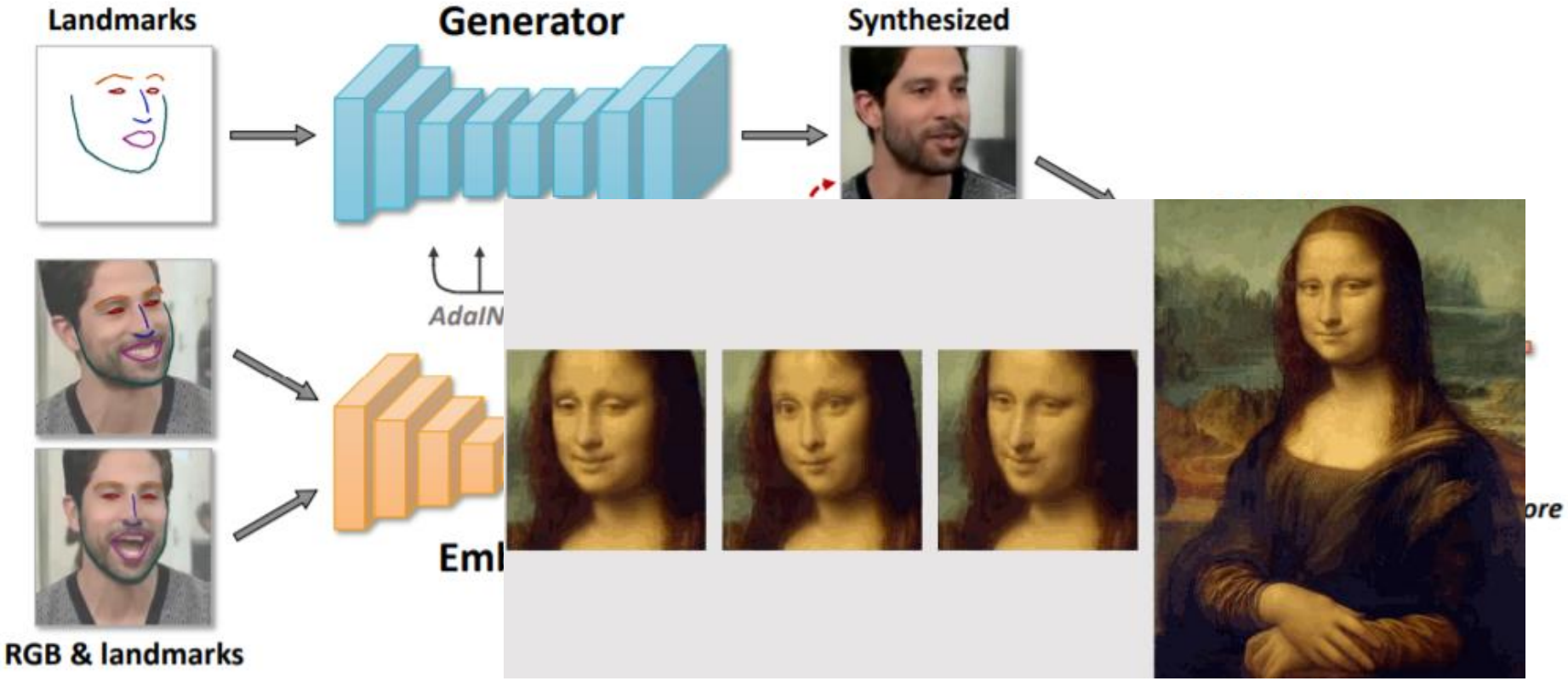
The classifiers can have different architectures.

The classifiers are trained as conditional GAN.

Conditional GAN outperforms other models designed for multi-label.

F1	MS-COCO	NUS-WIDE
VGG-16	56.0	33.9
+ GAN	60.4	41.2
Inception	62.4	53.5
+GAN	63.8	55.8
Resnet-101	62.8	53.1
+GAN	64.0	55.4
Resnet-152	63.3	52.1
+GAN	63.9	54.1
Att-RNN	62.1	54.7
RLSD	62.0	46.9

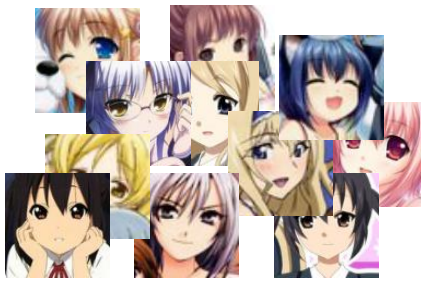
Talking Head



https://arxiv.org/abs/1905.08233

Three Categories of GAN

1. Typical GAN



$$\begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$

random vector

→ Generator →

image

2. Conditional GAN

blue eyes,
red hair,
short hair

paired data

"Girl with red hair"

text

→ Generator →

image

3. Unsupervised Conditional GAN

domain x

domain y

unpaired data

x

Photo

→ Generator →

y

Vincent van Gogh's style

Cycle GAN

Domain X

Domain Y

Domain X

$G_{X \to Y}$

Become similar to domain Y

D_Y

scalar

Input image belongs to domain Y or not

Domain Y

Cycle GAN

Domain X

Domain Y

Domain X

ignore input

$G_{X \to Y}$

Become similar to domain Y

Not what we want!

D_Y

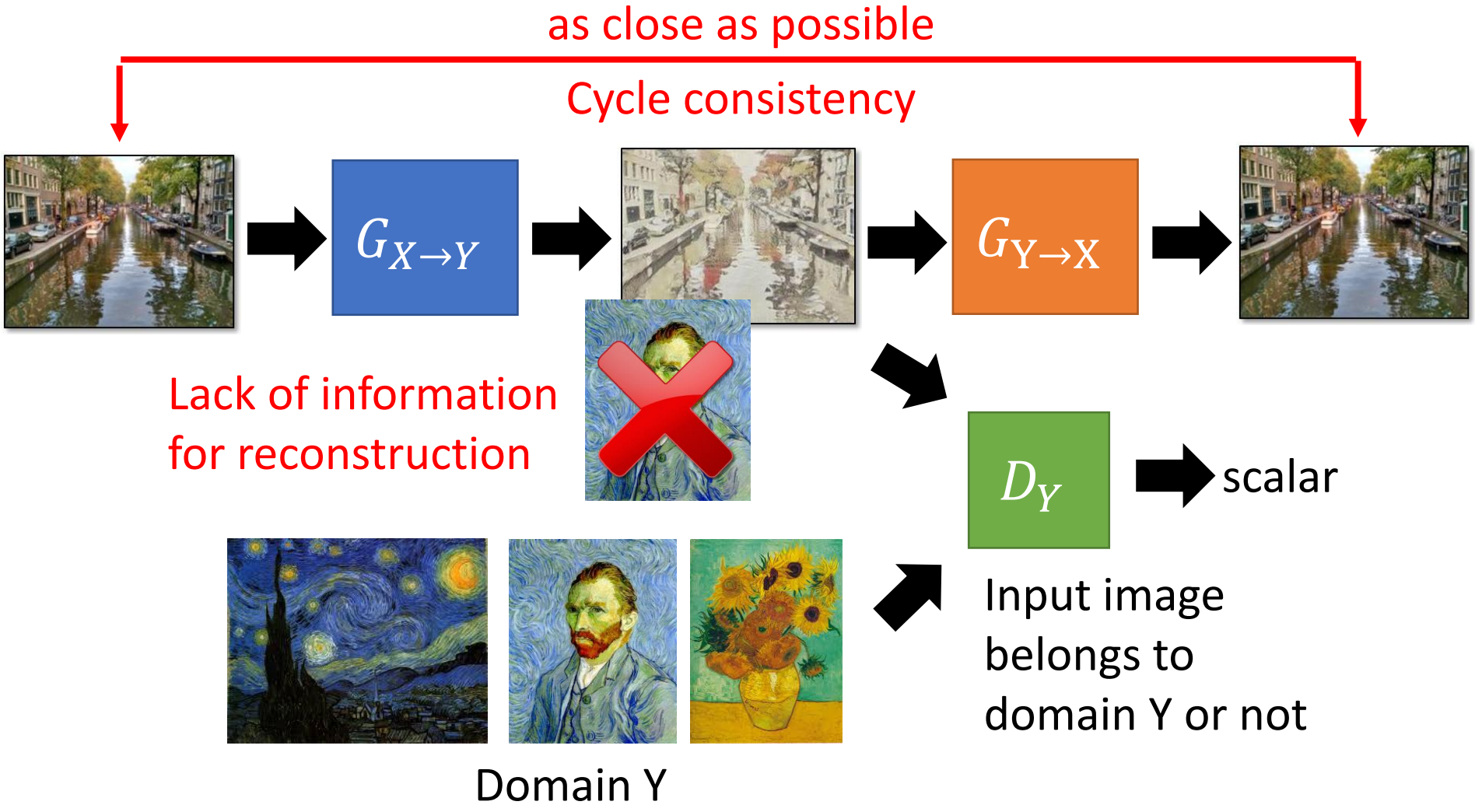
scalar

Input image belongs to domain Y or not

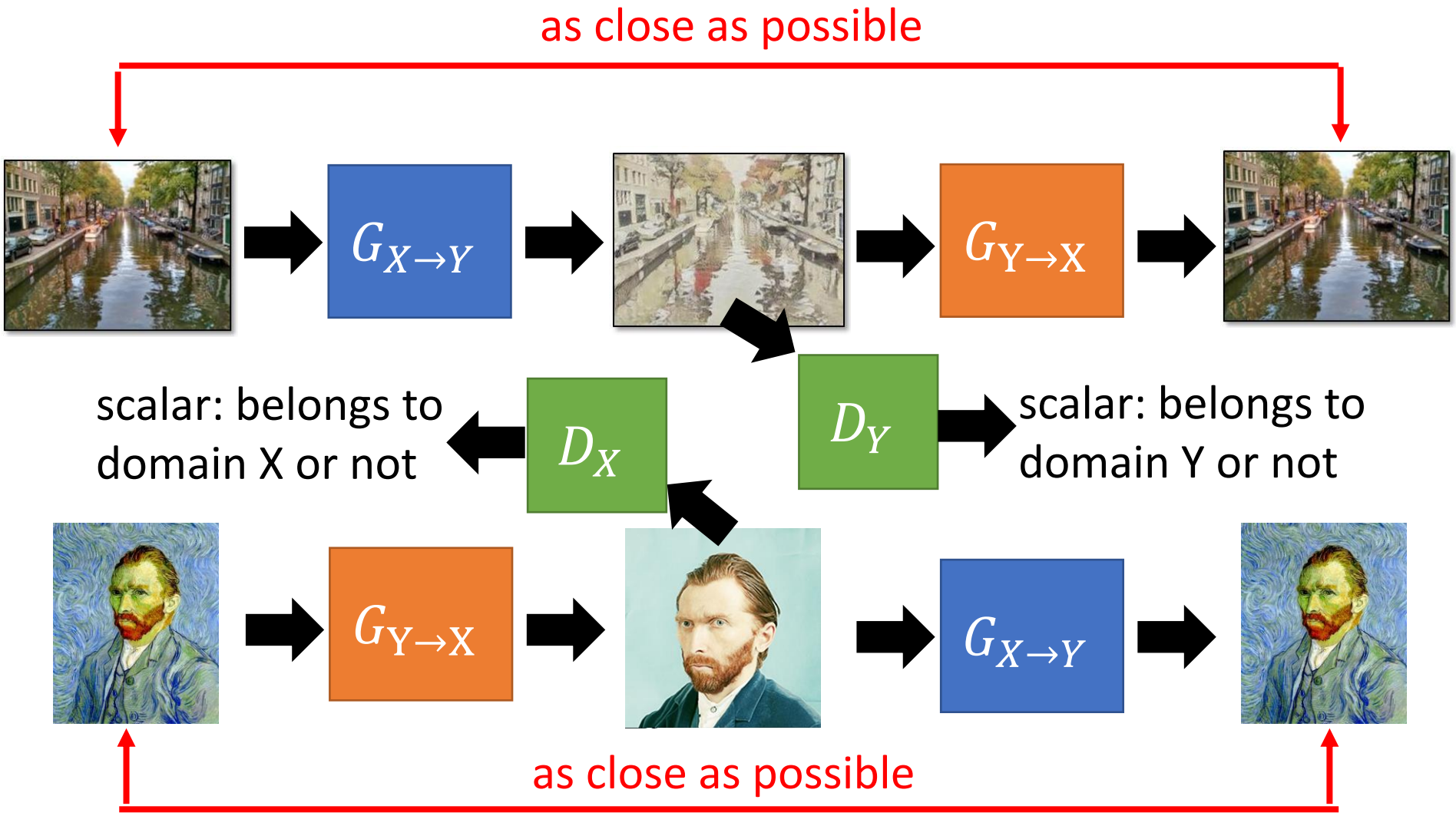
Domain Y

[Jun-Yan Zhu, et al., ICCV, 2017]

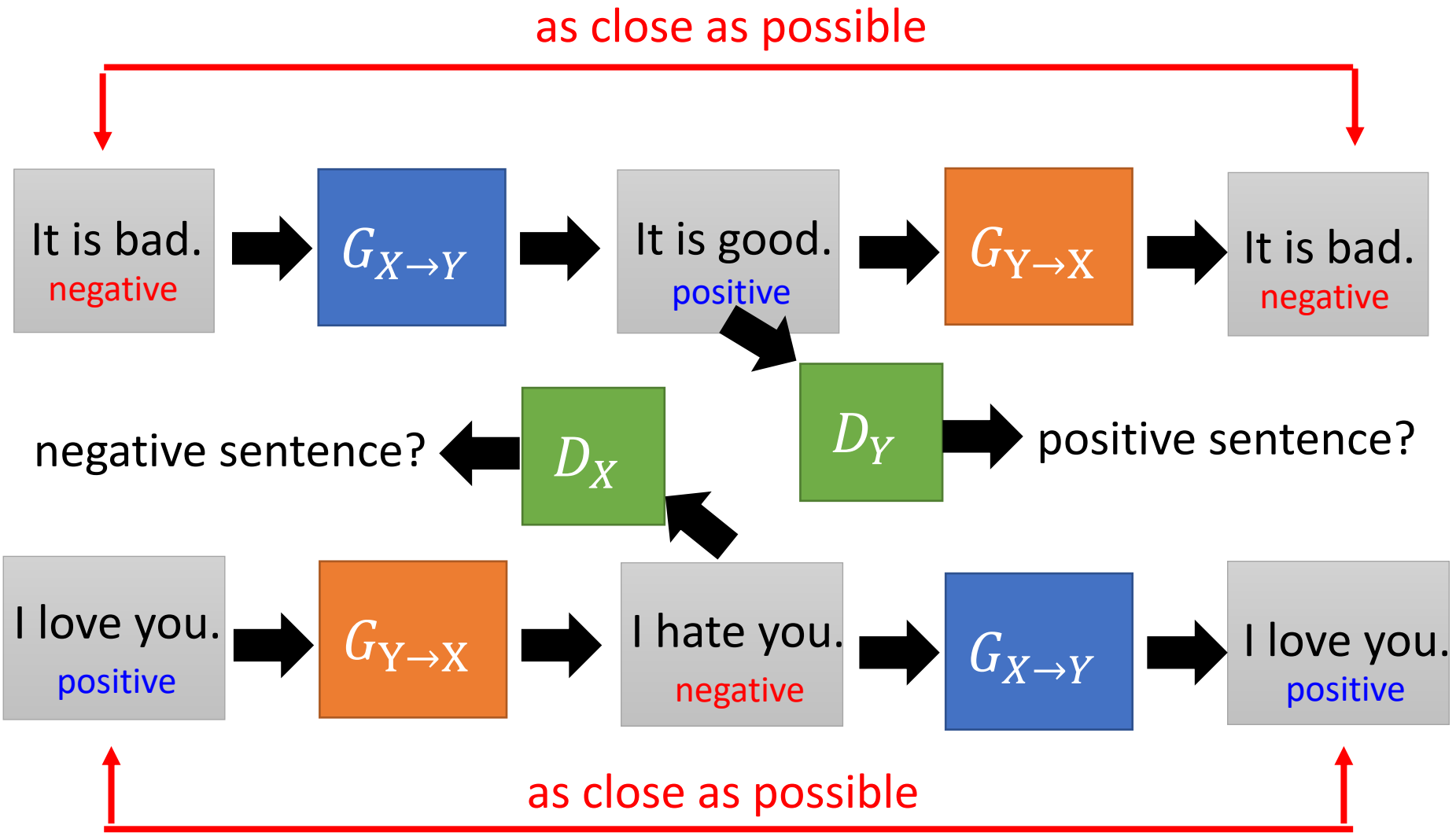
Cycle GAN



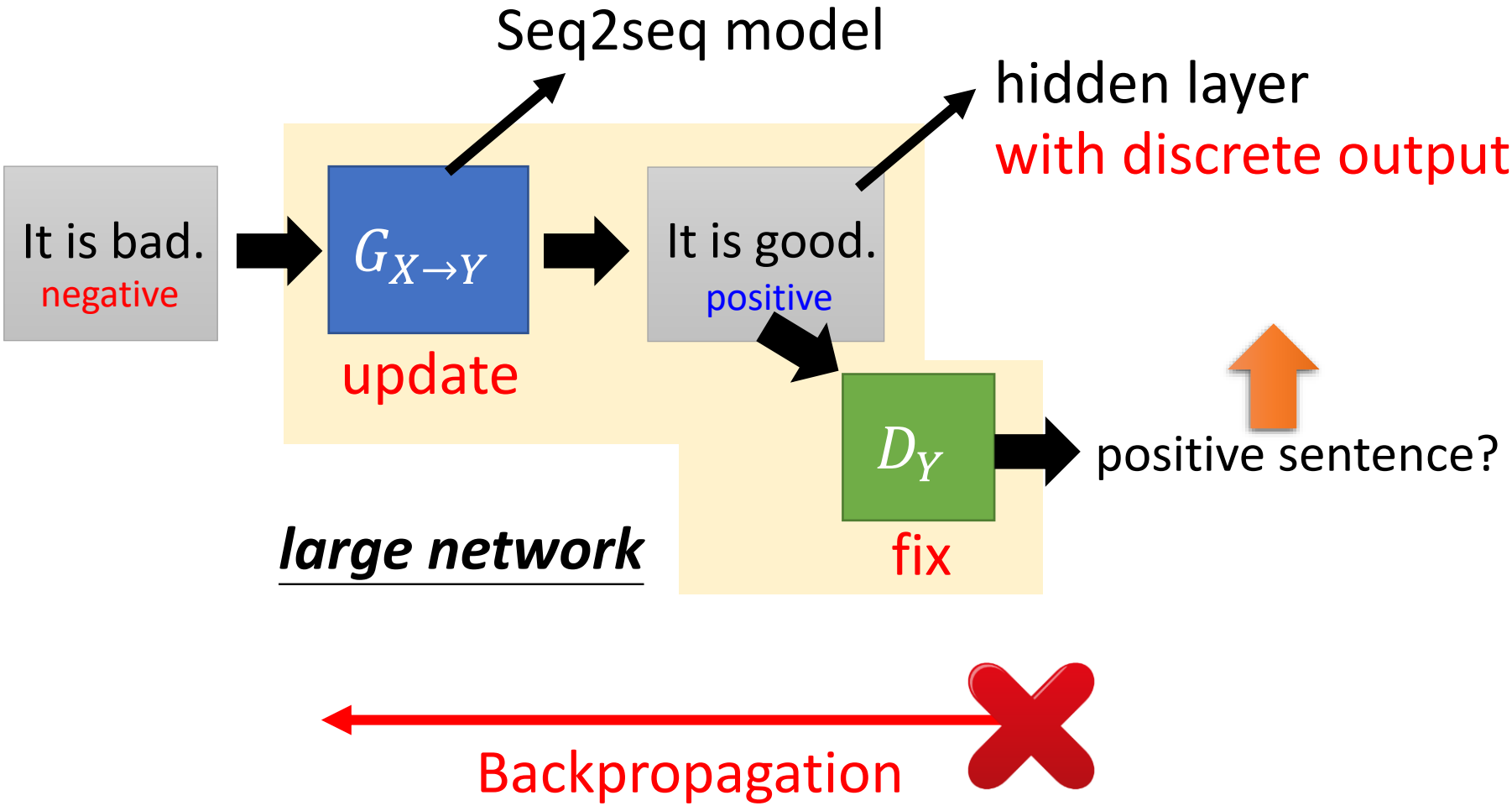
Cycle GAN


as close as possible
$G_{X\to Y}$
$G_{Y\to X}$
D_Y
D_X
scalar: belongs to domain X or not
scalar: belongs to domain Y or not
$G_{Y\to X}$
$G_{X\to Y}$
as close as possible

Cycle GAN


as close as possible
It is bad.
negative
$G_{X\to Y}$
It is good.
positive
$G_{Y\to X}$
It is bad.
negative
D_Y
positive sentence?
negative sentence?
D_X
I love you.
positive
$G_{Y\to X}$
I hate you.
negative
$G_{X\to Y}$
I love you.
positive
as close as possible

Discrete Issue



Three Categories of Solutions

Gumbel-softmax

- [Matt J. Kusner, et al, arXiv, 2016]

Continuous Input for Discriminator

- [Sai Rajeswar, et al., arXiv, 2017][Ofir Press, et al., ICML workshop, 2017][Zhen Xu, et al., EMNLP, 2017][Alex Lamb, et al., NIPS, 2016][Yizhe Zhang, et al., ICML, 2017]

"Reinforcement Learning"

- [Yu, et al., AAAI, 2017][Li, et al., EMNLP, 2017][Tong Che, et al, arXiv, 2017][Jiaxian Guo, et al., AAAI, 2018][Kevin Lin, et al, NIPS, 2017][William Fedus, et al., ICLR, 2018]

文句改寫

感謝 王耀賢 同學提供實驗結果

Negative sentence to positive sentence:

it's a crappy day -> it's a great day
i wish you could be here -> you could be here
it's not a good idea -> it's good idea
i miss you -> i love you
i don't love you -> i love you
i can't do that -> i can do that
i feel so sad -> i happy
it's a bad day -> it's a good day
it's a dummy day -> it's a great day
sorry for doing such a horrible thing -> thanks for doing a great thing
my doggy is sick -> my doggy is my doggy
my little doggy is sick -> my little doggy is my little doggy

文句改寫

感謝 張瓊之 同學提供實驗結果

Negative sentence to positive sentence:

胃疼，沒睡醒，各種不舒服 -> 生日快樂，睡醒，超級舒服

我都想去上班了，真夠賤的! -> 我都想去睡了，真帥的！

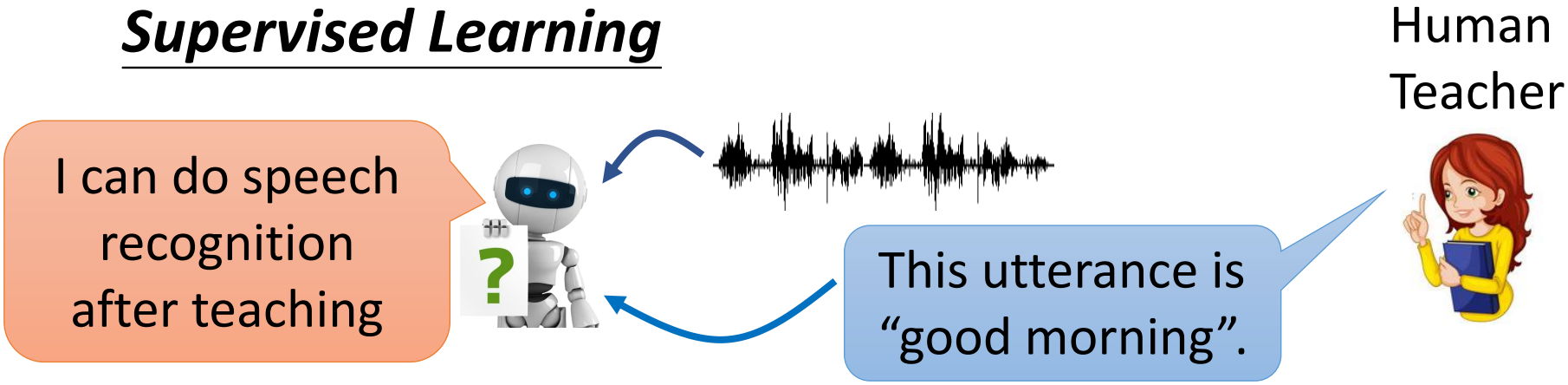
暈死了，吃燒烤、竟然遇到個變態狂 -> 哈哈好～，吃燒烤～竟然遇到帥狂

我肚子痛的厲害 -> 我生日快樂厲害

感冒了，難受的說不出話來了！-> 感冒了，開心的說不出話來！

Speech Recognition

Supervised Learning



- Supervised learning needs lots of annotated speech.
- However, most of the languages are low resourced.

Speech Recognition

Supervised Learning

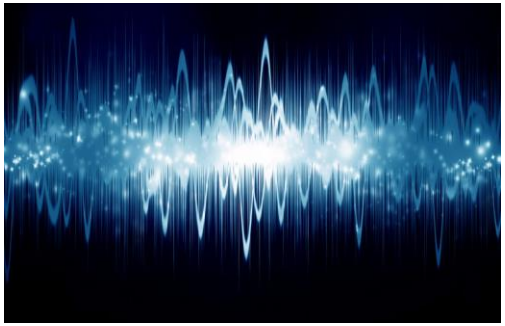
Human Teacher

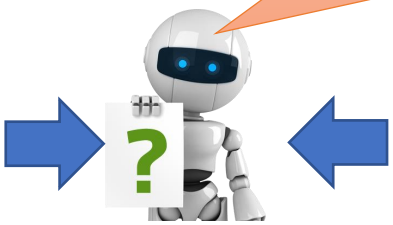
I can do speech recognition after teaching

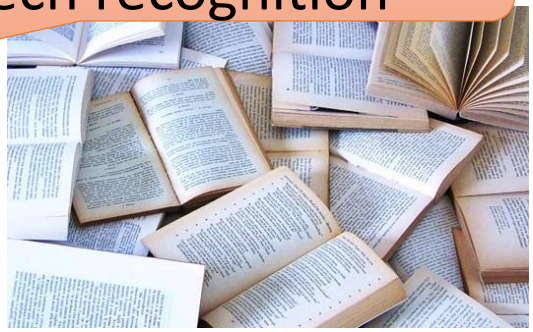
This utterance is “good morning”.

Unsupervised Learning

I can automatically learn speech recognition



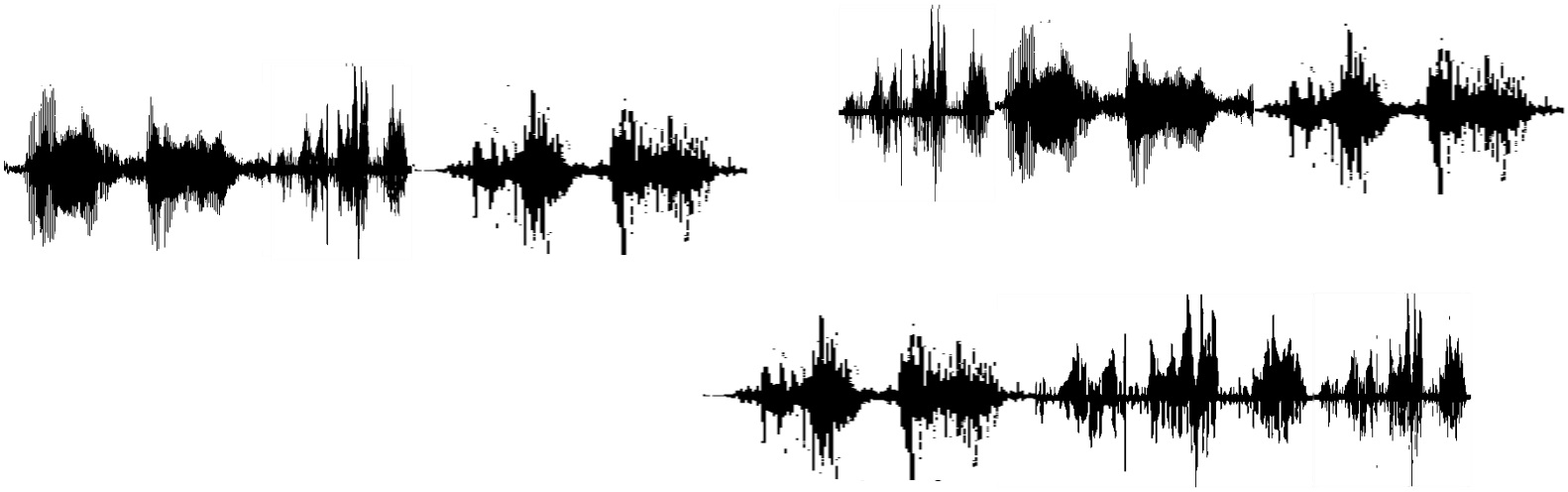




Listening to human talking

Reading text on the Internet

Acoustic Token Discovery

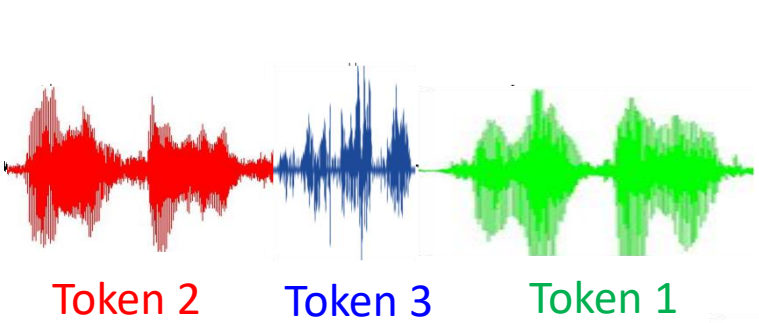


Acoustic tokens can be discovered from audio collection without text annotation.

Acoustic tokens: chunks of acoustically similar audio segments with token IDs

[Zhang & Glass, ASRU 09]
[Huijbregts, ICASSP 11]
[Chan & Lee, Interspeech 11]

Acoustic Token Discovery



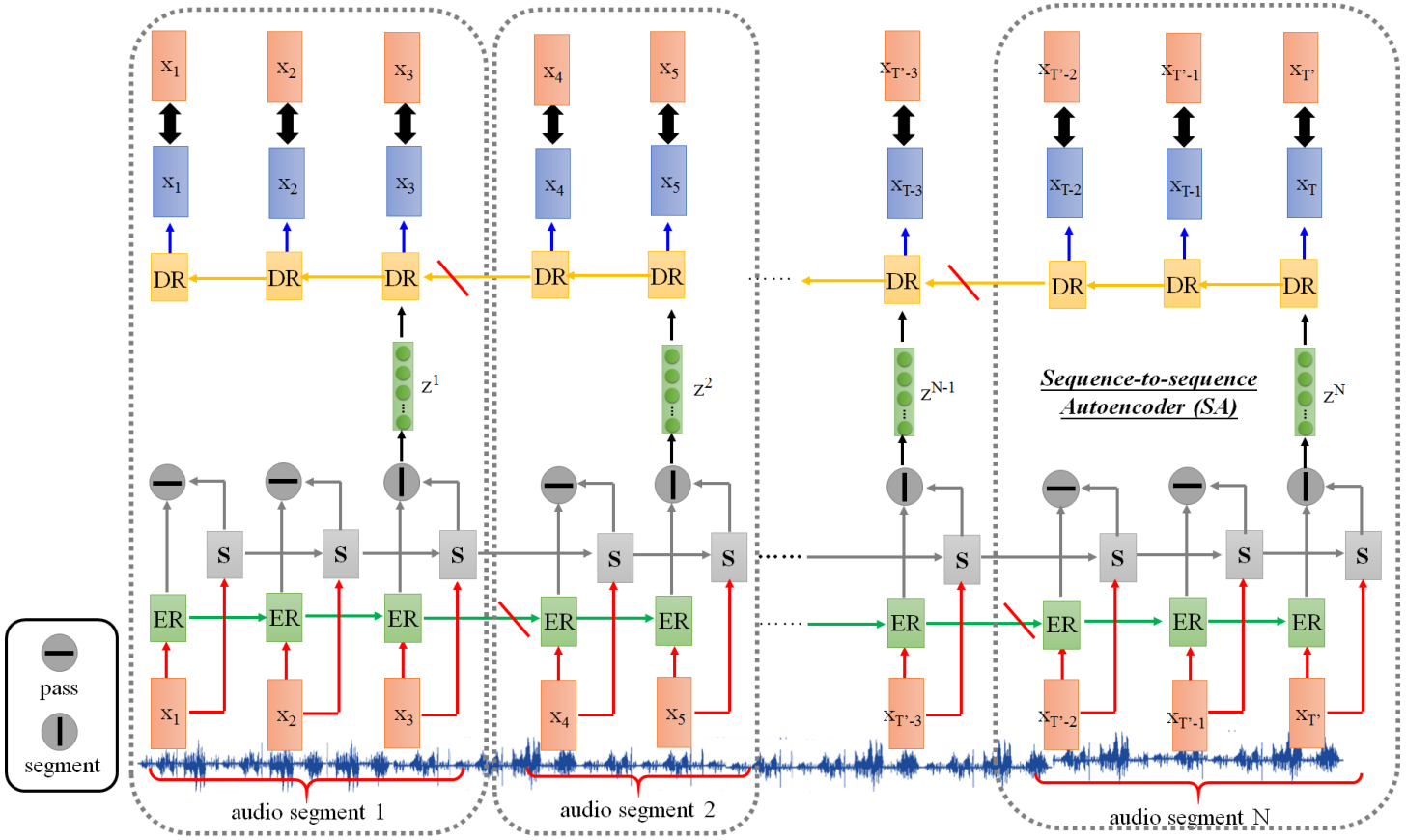
Acoustic tokens can be discovered from audio collection without text annotation.

Acoustic tokens: chunks of acoustically similar audio segments with token IDs

[Zhang & Glass, ASRU 09]
[Huijbregts, ICASSP 11]
[Chan & Lee, Interspeech 11]

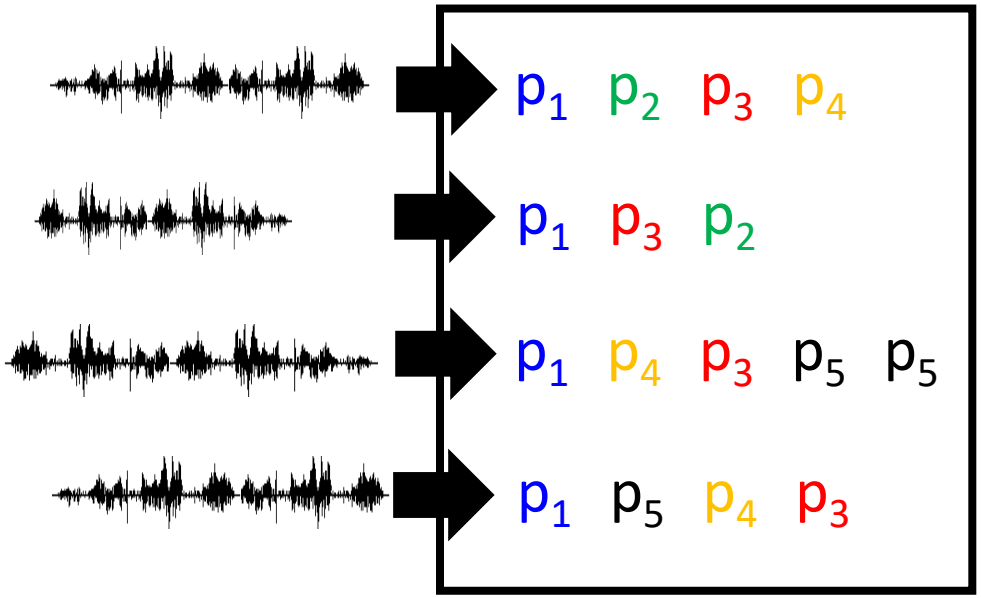
[Wang, et al., ICASSP, 2018]

Acoustic Token Discovery



Phonetic-level acoustic tokens are obtained by segmental sequence-to-sequence autoencoder.

Unsupervised Speech Recognition



Cycle GAN

AY L AH V Y UW

G UH D B AY

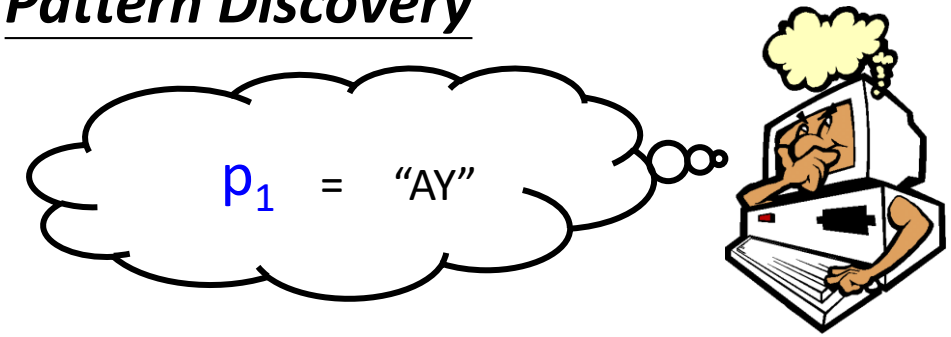
HH AW AA R Y UW

AY M F AY N

T AY W AA N

Phone-level Acoustic Pattern Discovery

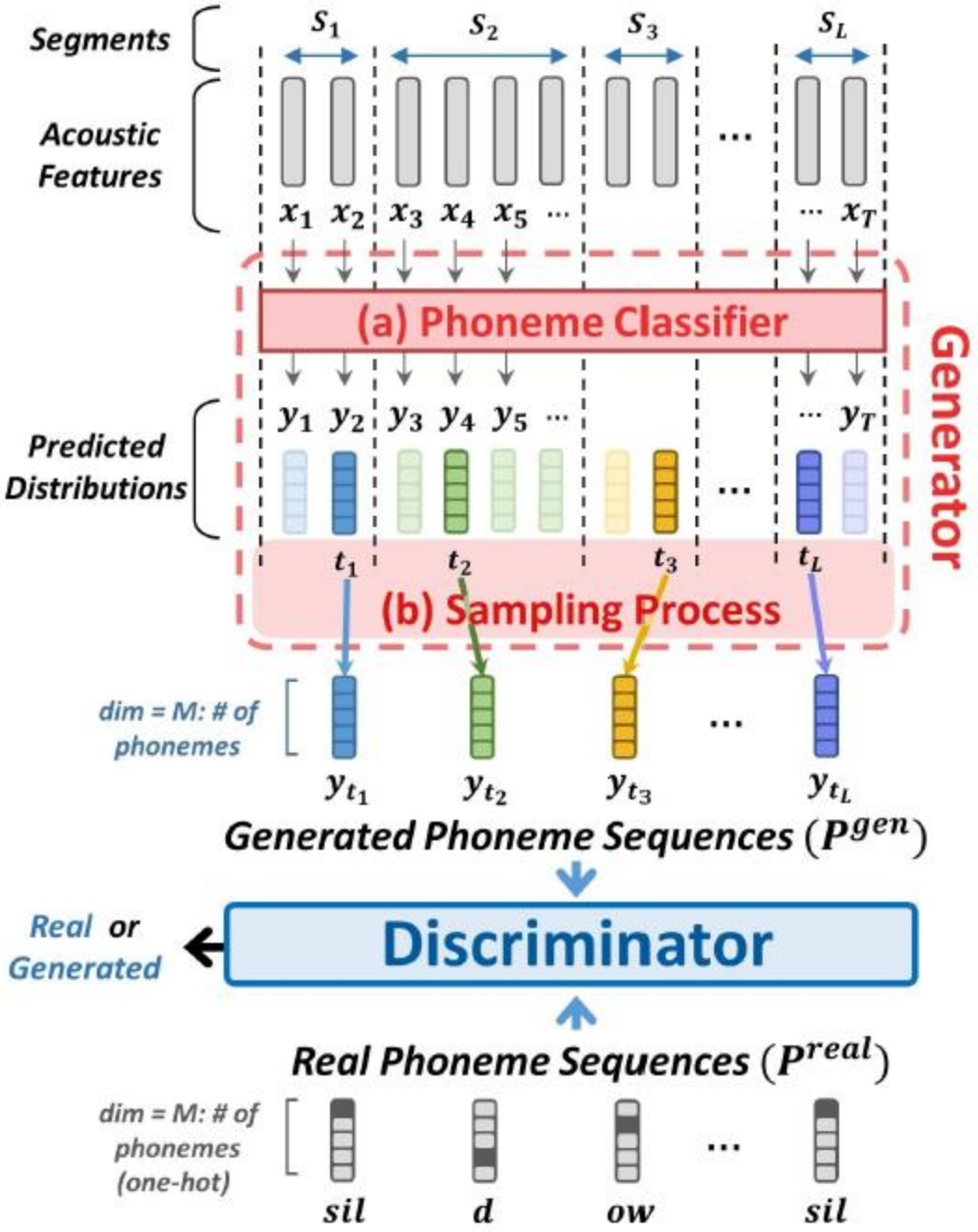
Phoneme sequences from Text



[Liu, et al., INTERSPEECH, 2018]

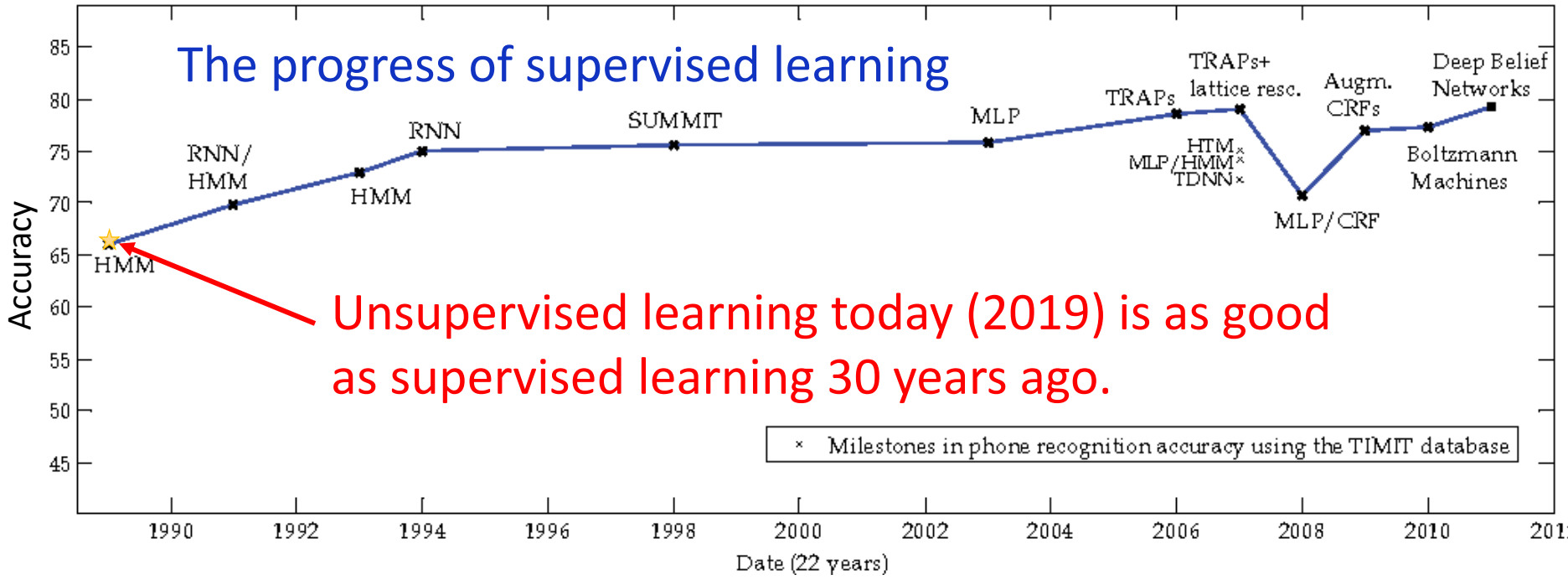
[Chen, et al., arXiv, 2018]

Model


Segments
Acoustic Features
Predicted Distributions
(a) Phoneme Classifier
(b) Sampling Process
Generator
dim = M: # of phonemes
Generated Phoneme Sequences (P^{gen})
Real or Generated
Discriminator
Real Phoneme Sequences (P^{real})
dim = M: # of phonemes (one-hot)
sil
d
ow
sil

Experimental Results

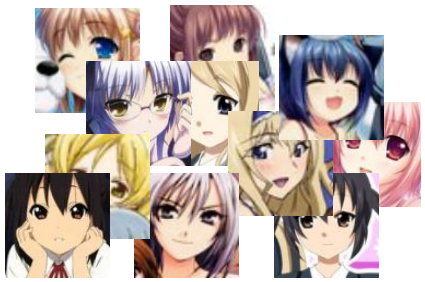
Approaches		Matched (all 4000)		Nonmatched (3000/1000)	
		FER	PER	FER	PER
(I) Supervised (labeled)					
(a) RNN Transducer [23]		-	17.7	-	-
(b) standard HMMs		-	21.5	-	-
(c) Phoneme classifier		27.0	28.9	-	-
(II) Unsupervised (with oracle boundaries)					
(d) Relationship mapping GAN [22]		40.5	40.2	43.6	43.4
(e) Segmental Emperical-ODM [23]		33.3	32.5	40.0	40.1
(f) Proposed: GAN		27.6	28.5	32.7	34.3
(III) Completely unsupervised (no label at all)					
(g) Segmental Emperical-ODM [23]		-	36.5	-	41.6
Proposed	iteration 1 (h) GAN	48.3	48.6	50.3	50.0
	iteration 1 (i) GAN/HMM	-	30.7	-	39.5
	iteration 2 (j) GAN	41.0	41.0	44.3	44.3
	iteration 2 (k) GAN/HMM	-	27.0	-	35.5
	iteration 3 (l) GAN	39.7	38.4	45.0	44.2
	iteration 3 (m) GAN/HMM	-	26.1	-	33.1



The image is modified from: Phone recognition on the TIMIT database Lopes, C. and Perdigão, F., 2011. Speech Technologies, Vol 1, pp. 285--302.

Three Categories of GAN

1. Typical GAN



$$\begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$

random vector

→ Generator →

image

2. Conditional GAN

blue eyes,
red hair,
short hair

paired data

"Girl with red hair"

text

→ Generator →

image

3. Unsupervised Conditional GAN

domain x

domain y

unpaired data

x

Photo

→ Generator →

y

Vincent van Gogh's style

Theory behind GAN

Generation

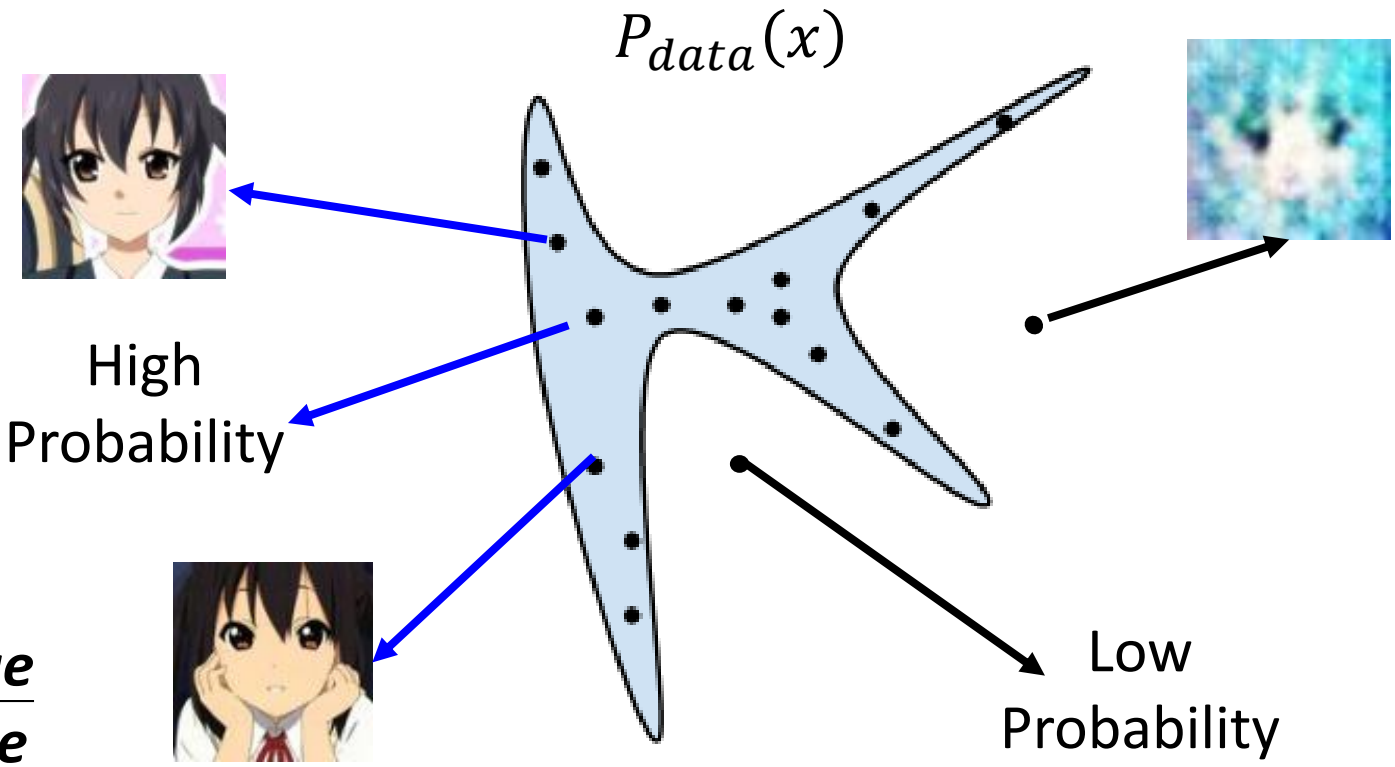
Using Generative Adversarial Network (GAN)

Drawing?

Generation

x: an image (a high-dimensional vector)

- We want to find data distribution $P_{data}(x)$



Maximum Likelihood Estimation

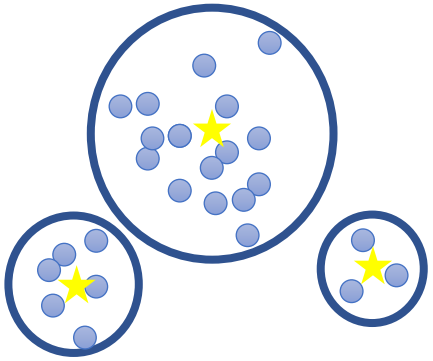
- Given a data distribution $P_{data}(x)$ (We can sample from it.)
- We have a distribution $P_G(x;\theta)$ parameterized by θ
  - We want to find θ such that $P_G(x;\theta)$ close to $P_{data}(x)$
  - E.g. $P_G(x;\theta)$ is a Gaussian Mixture Model, θ are means and variances of the Gaussians

Sample $\{x^1, x^2, \dots, x^m\}$ from $P_{data}(x)$

We can compute $P_G(x^i;\theta)$

Likelihood of generating the samples

$$L = \prod_{i=1}^{m} P_G(x^i;\theta)$$



Find θ^* maximizing the likelihood

Maximum Likelihood Estimation = Minimize KL Divergence

$$\theta^* = arg\max_\theta \prod_{i=1}^{m} P_G(x^i;\theta) = arg\max_\theta log \prod_{i=1}^{m} P_G(x^i;\theta)$$

$$= arg\max_\theta \sum_{i=1}^{m} logP_G(x^i;\theta)$$

$\{x^1, x^2, \ldots, x^m\}$ from $P_{data}(x)$

$$\approx arg\max_\theta E_{x\sim P_{data}}[logP_G(x;\theta)]$$

$$= arg\max_\theta \int_x P_{data}(x)logP_G(x;\theta)dx - \int_x P_{data}(x)logP_{data}(x)dx$$

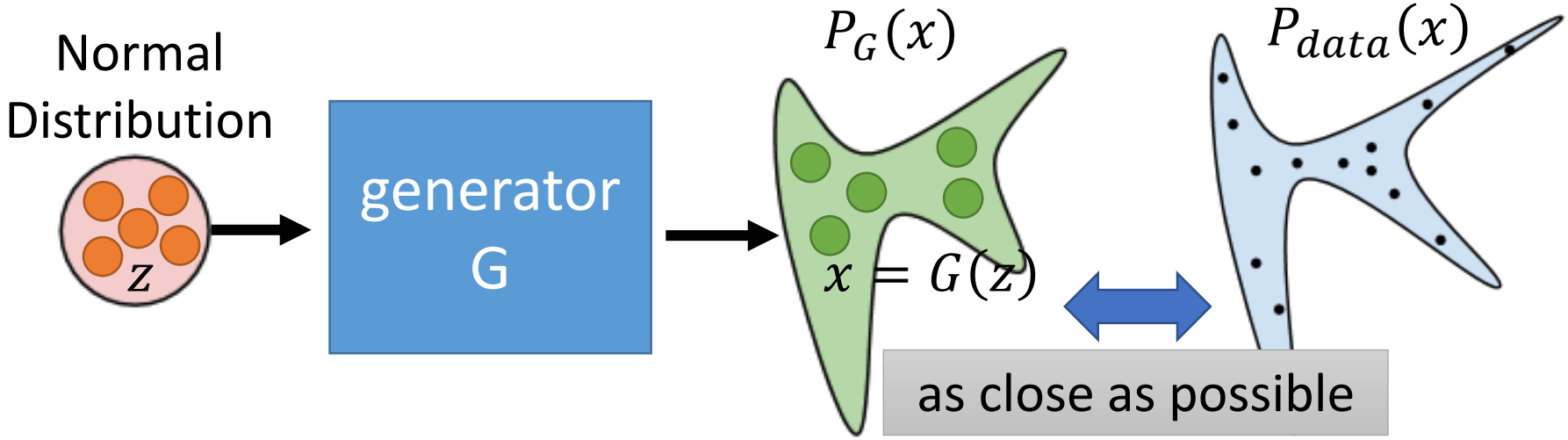
$$= arg\min_\theta KL(P_{data}||P_G)$$

How to define a general P_G?

Generator

x: an image (a high-dimensional vector)

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



$$G^* = arg \min_G Div(P_G, P_{data})$$

Divergence between distributions P_G and P_{data}

How to compute the divergence?

Discriminator

$$G^* = arg \min_G Div(P_G, P_{data})$$

Although we do not know the distributions of P_G and P_{data}, we can sample from them.

Database

sample

Sampling from P_{data}

sample from normal

G

Sampling from P_G

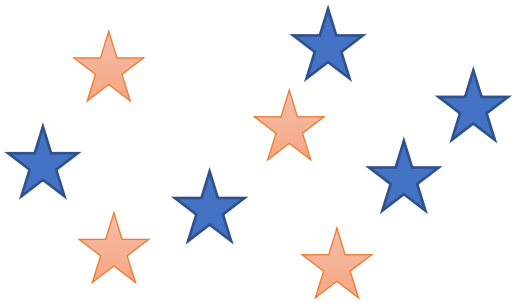
Discriminator

$$G^* = arg \min_G Div(P_G, P_{data})$$

: data sampled from P_{data}

: data sampled from P_G

Using the example objective function is exactly the same as training a binary classifier.



train

Discriminator

Sigmoid Output

Example Objective Function for D

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

(G is fixed)

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

The maximum objective value is related to JS divergence.

[Goodfellow, et al., NIPS, 2014]

Discriminator

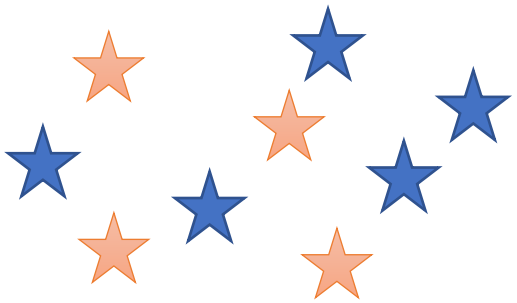
$$G^* = arg \min_G Div(P_G, P_{data})$$

: data sampled from P_{data}

: data sampled from P_G

Training:

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

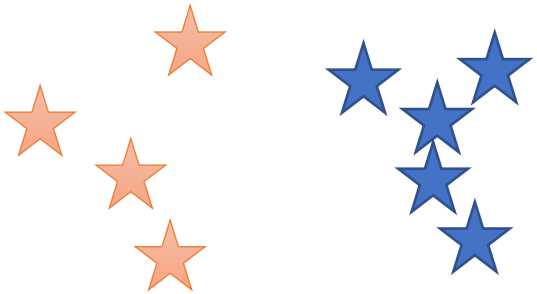


small divergence

train

Discriminator

hard to discriminate
(cannot make objective large)



large divergence

train

Discriminator

easy to discriminate

$$\max_D V(G,D)$$

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

- Given G, what is the optimal D* maximizing

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

$$= \int_x P_{data}(x)logD(x)\,dx + \int_x P_G(x)log\left(1-D(x)\right)dx$$

$$= \int_x \left[P_{data}(x)logD(x) + P_G(x)log\left(1-D(x)\right)\right]dx$$

Assume that D(x) can be any function

- Given x, the optimal D* maximizing

$$P_{data}(x)logD(x) + P_G(x)log\left(1-D(x)\right)$$

$$\max_D V(G,D)$$

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

- Given x, the optimal D* maximizing

$$P_{data}(x)logD(x) + P_G(x)log(1-D(x))$$

a D b D

- Find D* maximizing: $f(D) = alog(D) + blog(1-D)$

$$\frac{df(D)}{dD} = a\times\frac{1}{D} + b\times\frac{1}{1-D}\times(-1) = 0$$

$$a\times\frac{1}{D^*} = b\times\frac{1}{1-D^*} \qquad a\times(1-D^*) = b\times D^*$$

$$a - aD^* = bD^* \qquad a = (a+b)D^*$$

$$D^* = \frac{a}{a+b} \Rightarrow D^*(x) = \frac{P_{data}(x)}{P_{data}(x)+P_G(x)}$$

$0 <$ < 1

$$\max_D V(G,D)$$

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

$$\max_D V(G,D) = V(G,D^*) \qquad D^*(x) = \frac{P_{data}(x)}{P_{data}(x)+P_G(x)}$$

$$= E_{x\sim P_{data}}\left[\log \frac{P_{data}(x)}{P_{data}(x)+P_G(x)}\right] + E_{x\sim P_G}\left[\log \frac{P_G(x)}{P_{data}(x)+P_G(x)}\right]$$

$$= \int_x P_{data}(x)\log \frac{\frac{1}{2}P_{data}(x)}{\frac{P_{data}(x)+P_G(x)}{2}}dx + \int_x P_G(x)\log \frac{\frac{1}{2}P_G(x)}{\frac{P_{data}(x)+P_G(x)}{2}}dx$$

$$+2\log\frac{1}{2} \quad -2\log 2$$

$$\max_D V(G,D)$$

$$\mathrm{JSD}(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M)$$

$$M = \frac{1}{2}(P+Q)$$

$$\max_D V(G,D) = V(G,D^*) \qquad D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

$$= -2log2 + \int_x P_{data}(x) log \frac{P_{data}(x)}{\big(P_{data}(x) + P_G(x)\big)/2} dx$$

$$+ \int_x P_G(x) log \frac{P_G(x)}{\big(P_{data}(x) + P_G(x)\big)/2} dx$$

$$= -2\log 2 + \mathrm{KL}\left(\mathrm{P_{data}} || \frac{\mathrm{P_{data}} + \mathrm{P_G}}{2}\right) + \mathrm{KL}\left(\mathrm{P_G} || \frac{\mathrm{P_{data}} + \mathrm{P_G}}{2}\right)$$

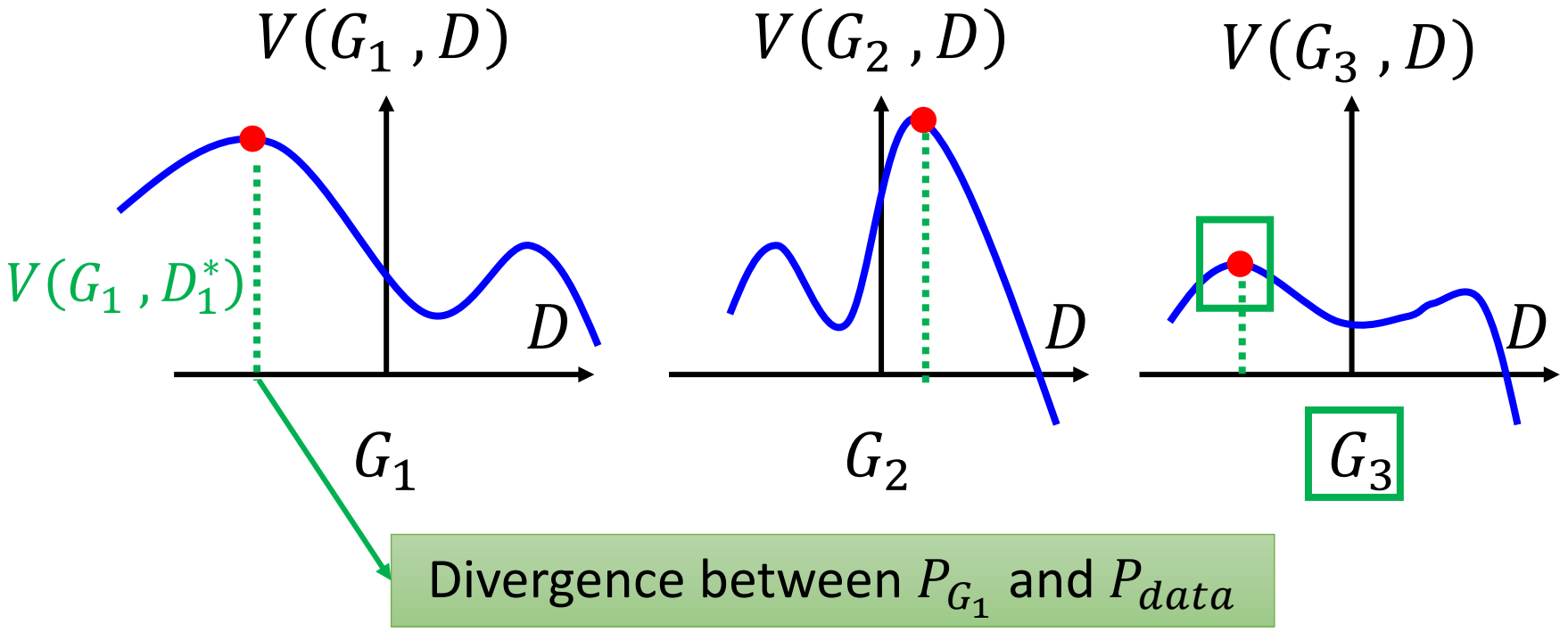
$$= -2log2 + 2JSD(P_{data}||P_G)$$

Jensen-Shannon divergence

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

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

The maximum objective value is related to JS divergence.



[Goodfellow, et al., NIPS, 2014]

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

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

The maximum objective value is related to JS divergence.

- Initialize generator and discriminator
- In each training iteration:

  Step 1: Fix generator G, and update discriminator D

  Step 2: Fix discriminator D, and update generator G

Algorithm

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

$L(G)$

- To find the best G minimizing the loss function $L(G)$,

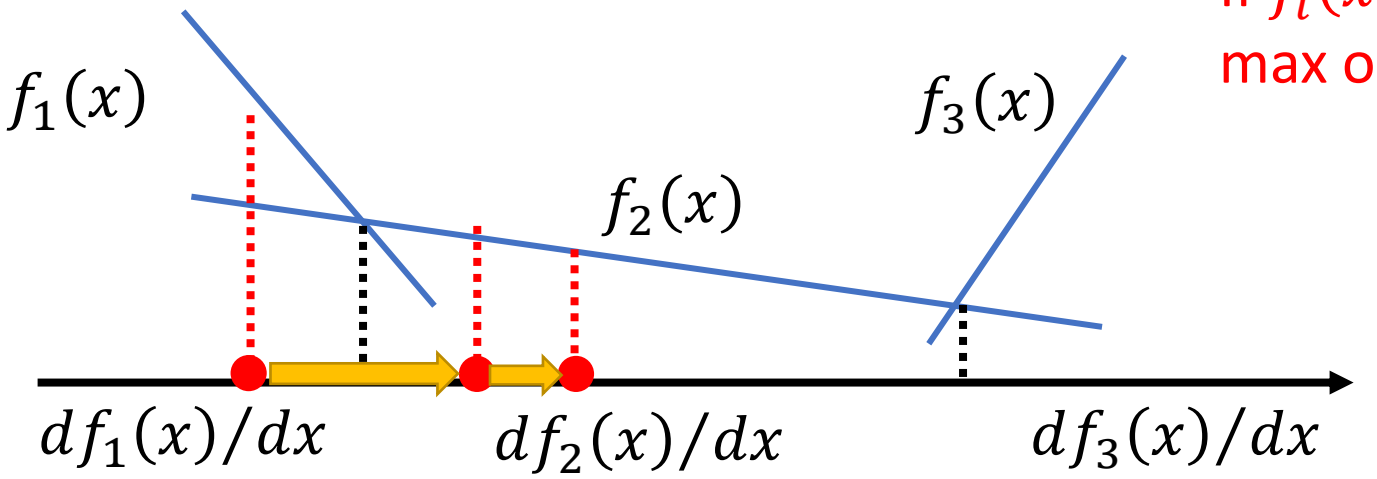
$$\theta_G \leftarrow \theta_G - \eta \, \partial L(G)/\partial \theta_G$$

θ_G defines G

$$f(x) = \max\{f_1(x), f_2(x), f_3(x)\}$$

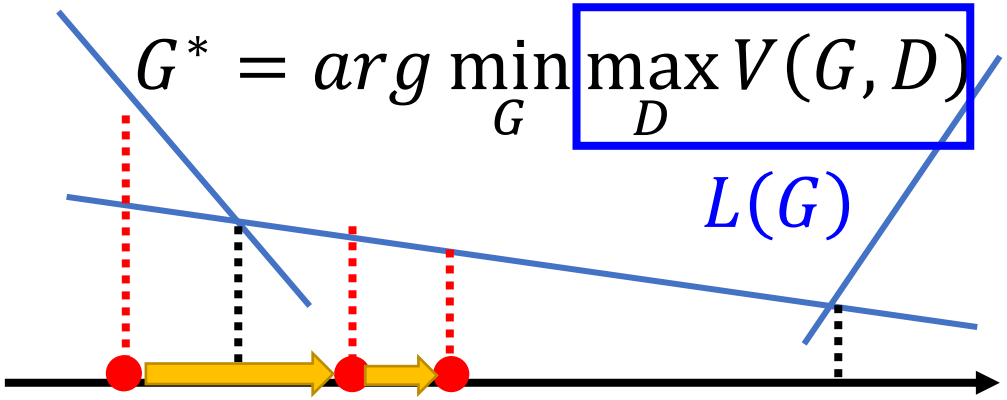
$$\frac{df(x)}{dx} =? \; df_i(x)/dx$$

If $f_i(x)$ is the max one



Algorithm

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



- Given G_0
- Find D_0^* maximizing $V(G_0, D)$ **Using Gradient Ascent**

$V(G_0, D_0^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_0}(x)$

- $\theta_G \leftarrow \theta_G - \eta \, \partial V(G, D_0^*)/\partial \theta_G$ → Obtain G_1
- Find D_1^* maximizing $V(G_1, D)$

Decrease JS divergence(?)

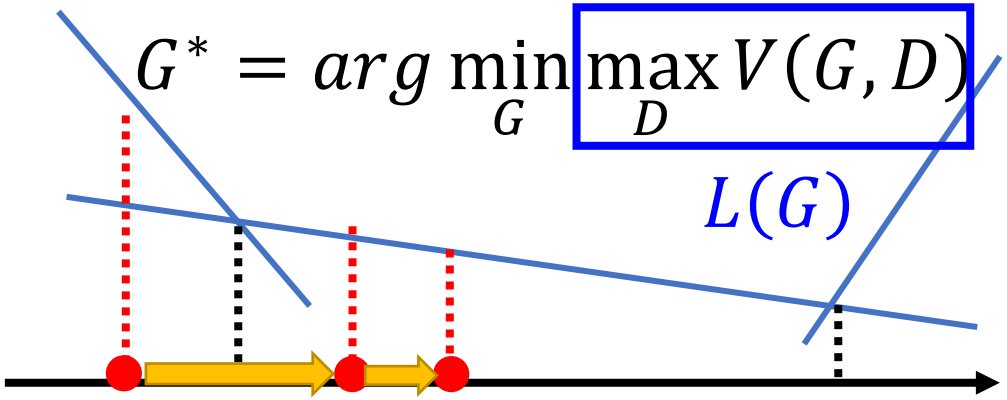
$V(G_1, D_1^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_1}(x)$

- $\theta_G \leftarrow \theta_G - \eta \, \partial V(G, D_1^*)/\partial \theta_G$ → Obtain G_2
- ……

Decrease JS divergence(?)

Algorithm

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

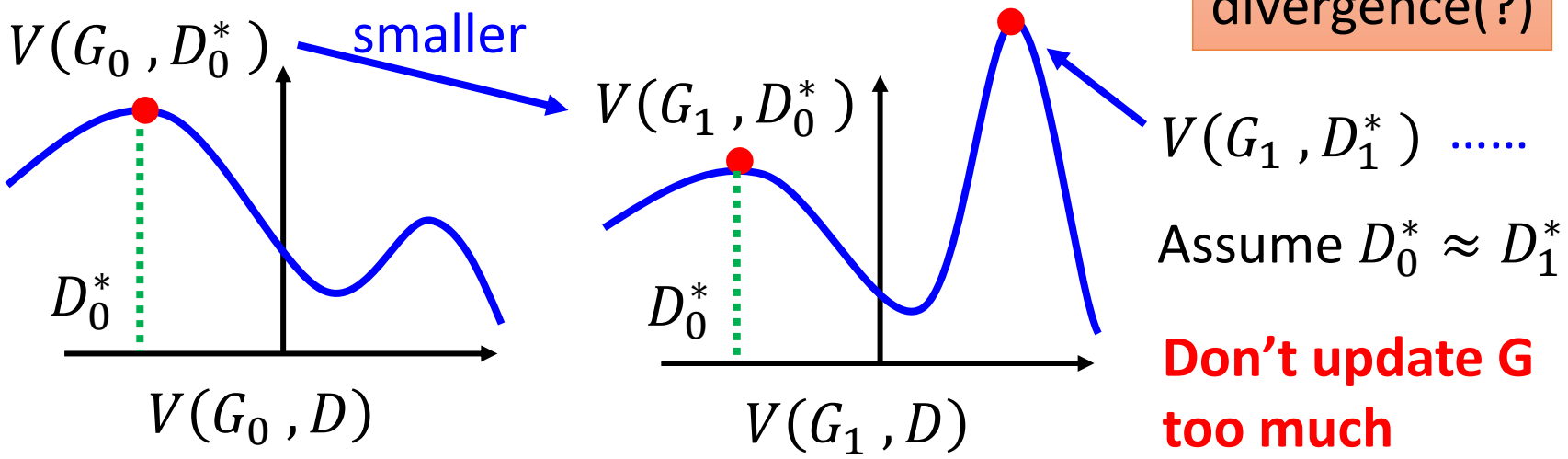


- Given G_0
- Find D_0^* maximizing $V(G_0, D)$

$V(G_0, D_0^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_0}(x)$

- $\theta_G \leftarrow \theta_G - \eta \, \partial V(G, D_0^*)/\partial \theta_G$ ➡ Obtain G_1

Decrease JS divergence(?)



In practice …

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

- Given G, how to compute $\max_D V(G, D)$
  - Sample $\{x^1, x^2, \ldots, x^m\}$ from $P_{data}(x)$, sample $\{\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^m\}$ from generator $P_G(x)$

Maximize $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m} \log D(x^i) + \frac{1}{m}\sum_{i=1}^{m} \log\left(1 - D(\tilde{x}^i)\right)$

Binary Classifier

D is a binary classifier with sigmoid output (can be deep)

$\{x^1, x^2, \ldots, x^m\}$ from $P_{data}(x)$ → Positive examples

$\{\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^m\}$ from $P_G(x)$ → Negative examples

Minimize Cross-entropy

Algorithm

Initialize θ_d for D and θ_g for G

Can only find lower bound of $\max_D V(G,D)$

- 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
  - $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m} log D(x^i) + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D(\tilde{x}^i)\right)$
  - $\theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$

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
  - $\tilde{V} = \cancel{\frac{1}{m}\sum_{i=1}^{m} log D(x^i)} + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D\left(G(z^i)\right)\right)$
  - $\theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$

Objective Function for Generator in Real Implementation

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

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

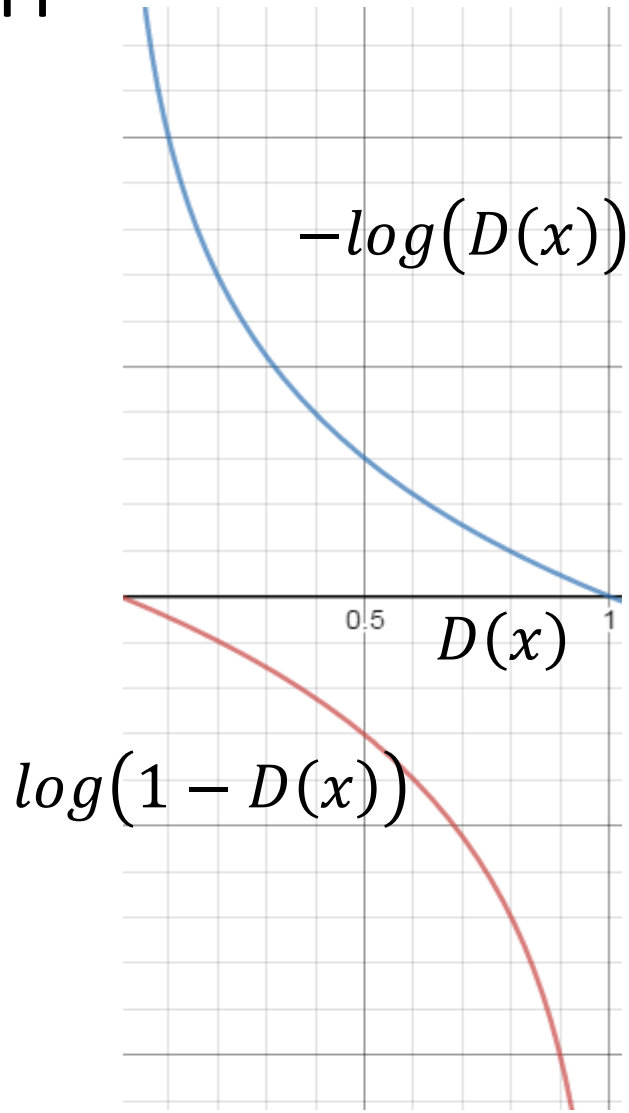
Slow at the beginning

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)



Tips for Improving GAN

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

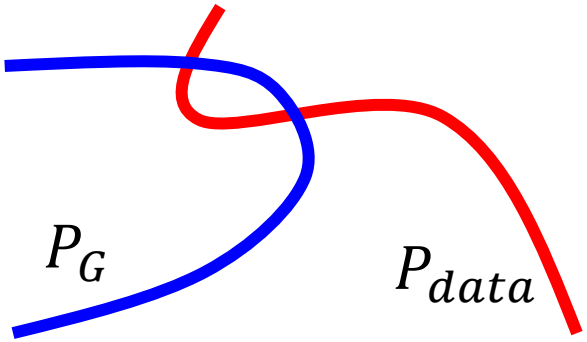
Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville, "Improved Training of Wasserstein GANs", arXiv prepring, 2017

JS divergence is not suitable

- In most cases, P_G and P_{data} are not overlapped.
- 1. The nature of data

  Both P_{data} and P_G are low-dim manifold in high-dim space.

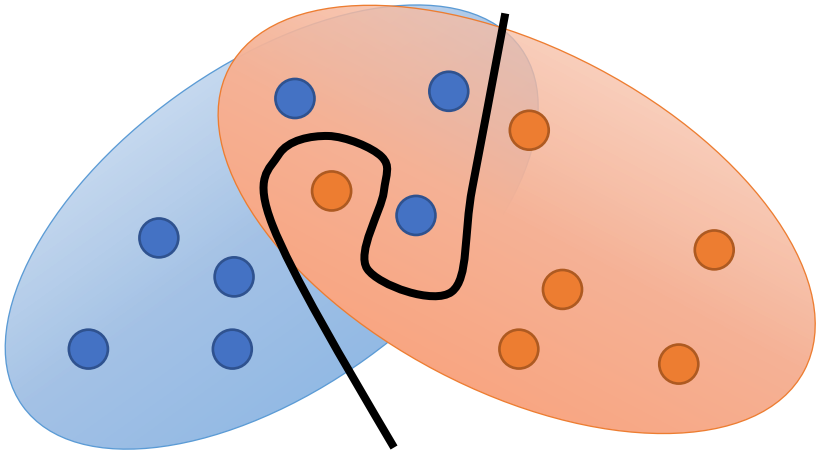
  The overlap can be ignored.



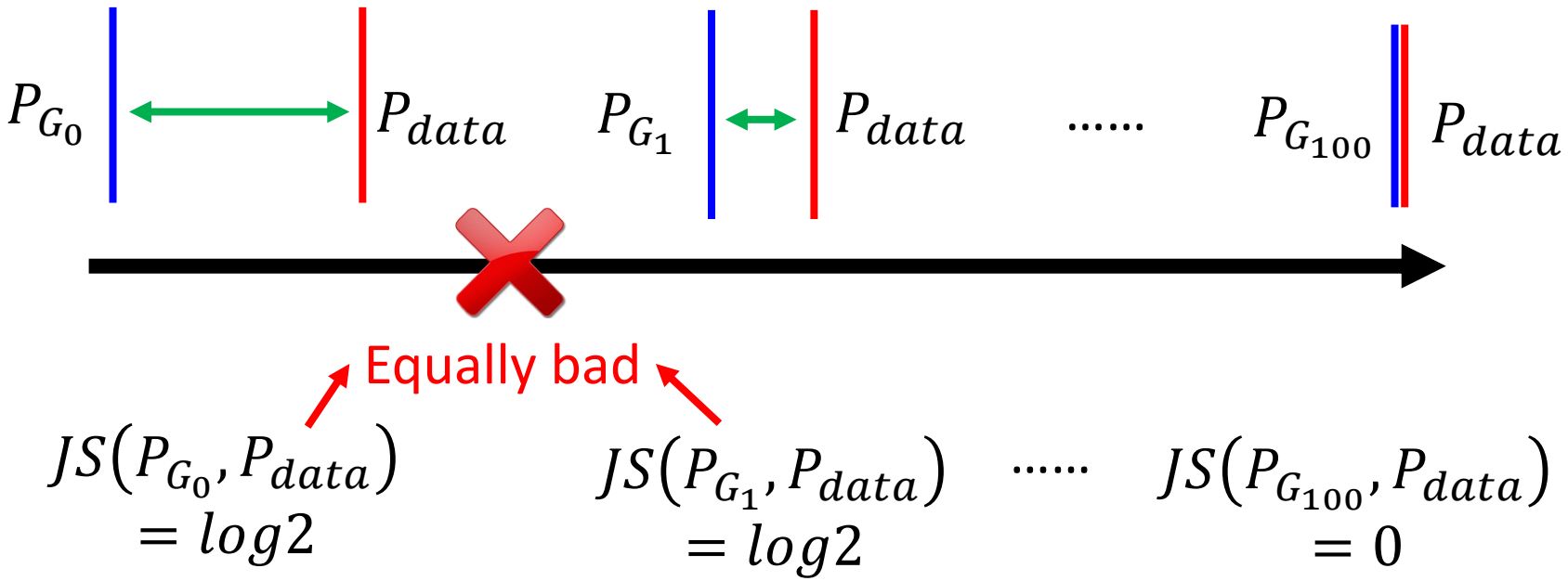
- 2. Sampling

  Even though P_{data} and P_G have overlap.

  If you do not have enough sampling ……



What is the problem of JS divergence?

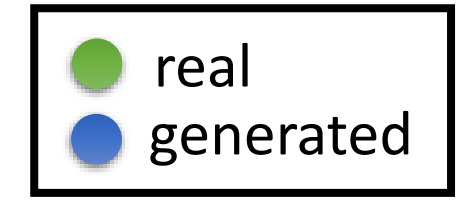


JS divergence is log2 if two distributions do not overlap.

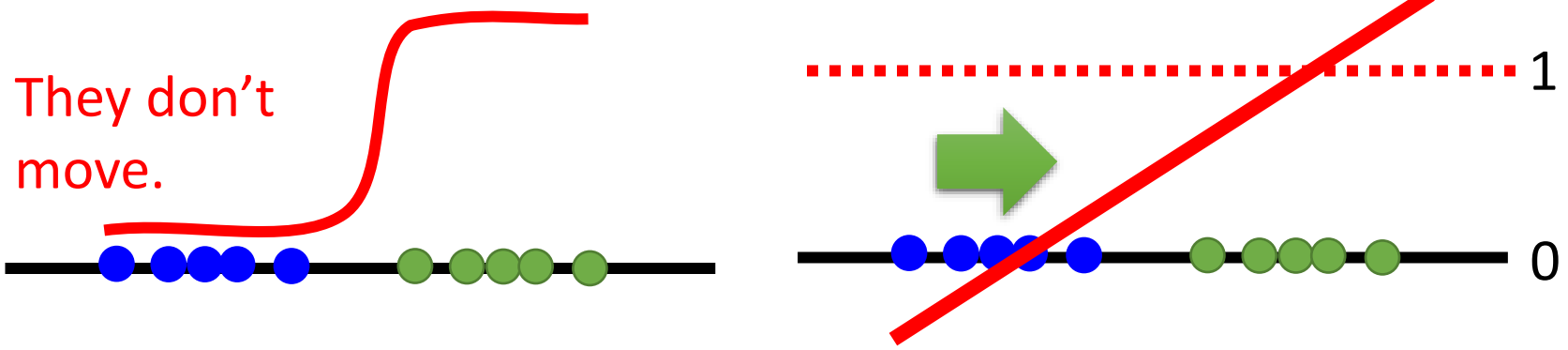
Intuition: If two distributions do not overlap, binary classifier achieves 100% accuracy

➡ Same objective value is obtained. ➡ Same divergence

Least Square GAN (LSGAN)

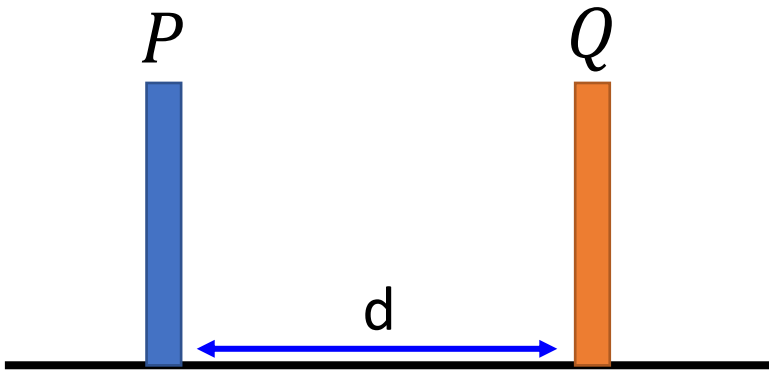


- Replace sigmoid with linear (replace classification with regression)



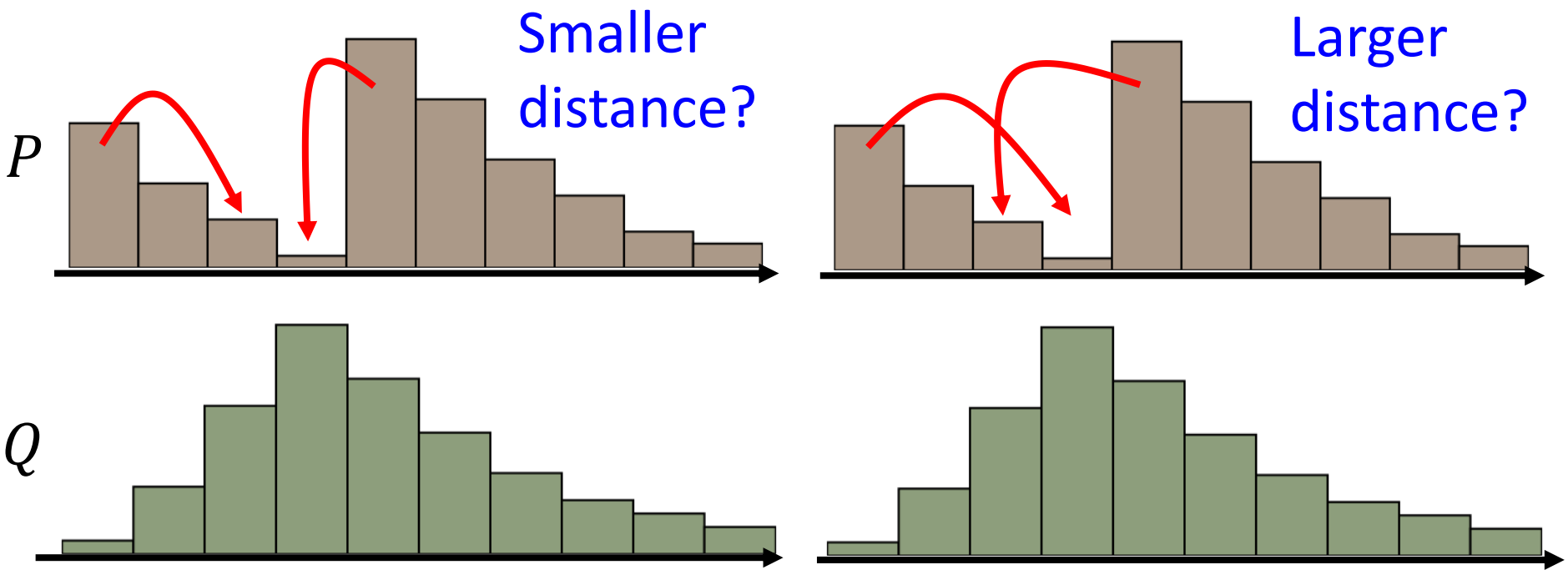
Wasserstein GAN (WGAN): Earth Mover's Distance

- 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.



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

WGAN: Earth Mover's Distance



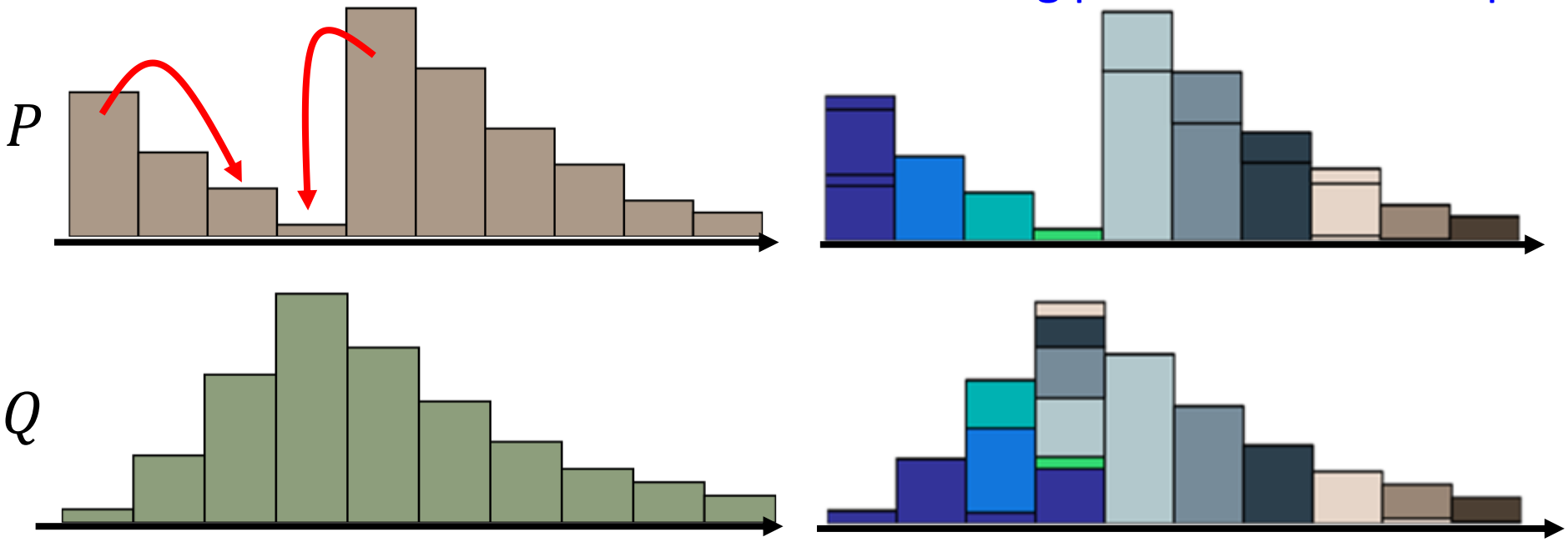
There many possible "moving plans".

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/

WGAN: Earth Mover's Distance

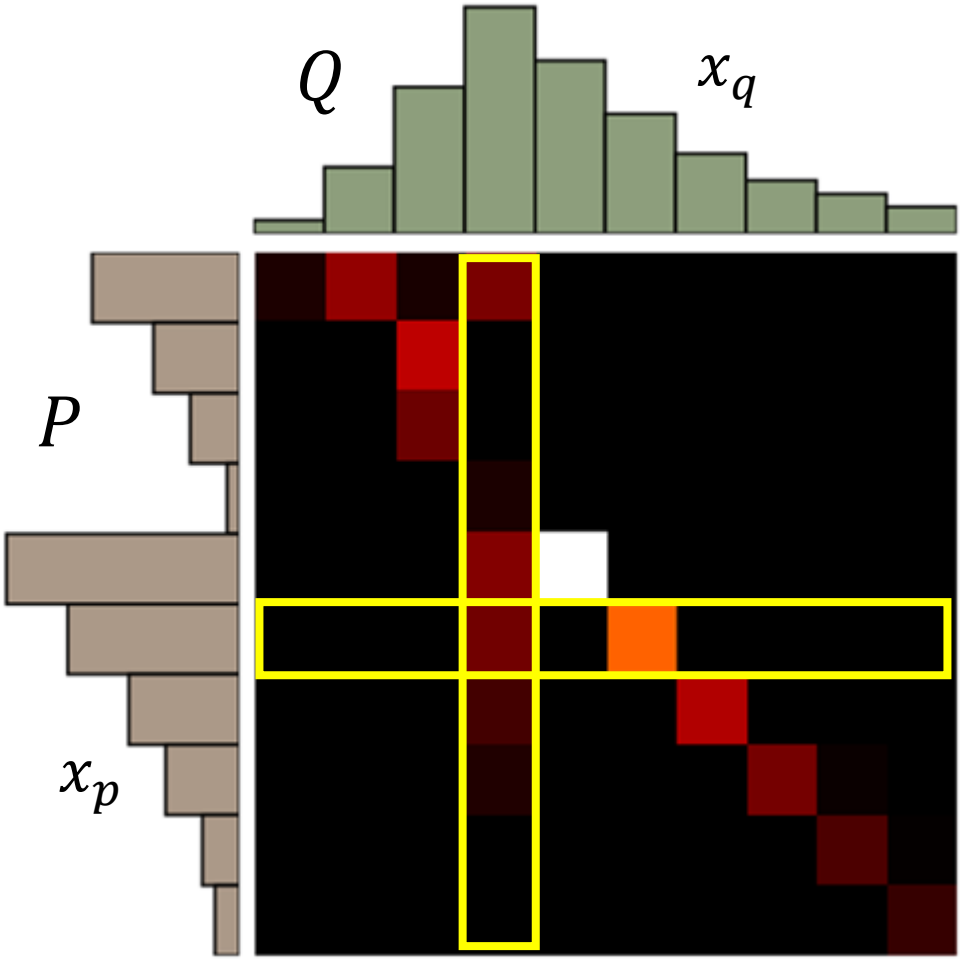
Best "moving plans" of this example



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.

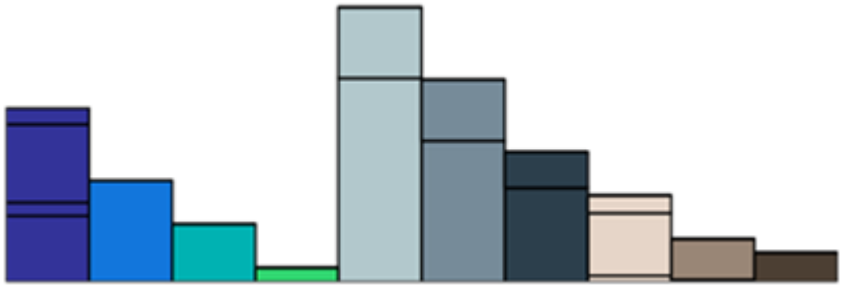
Average distance of a plan γ:

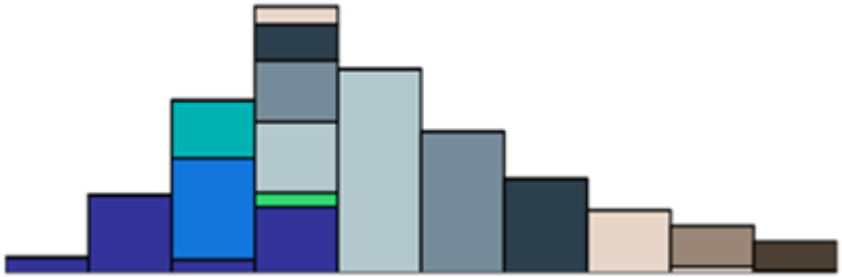
$$B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

Earth Mover’s Distance:

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

The best plan

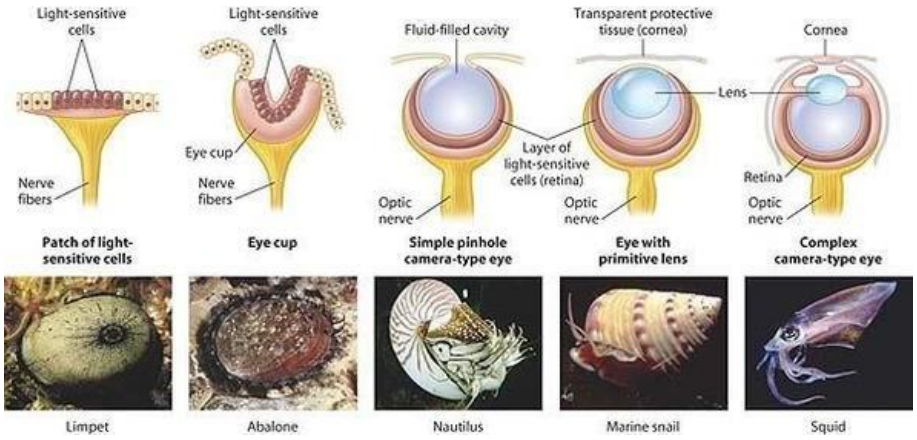


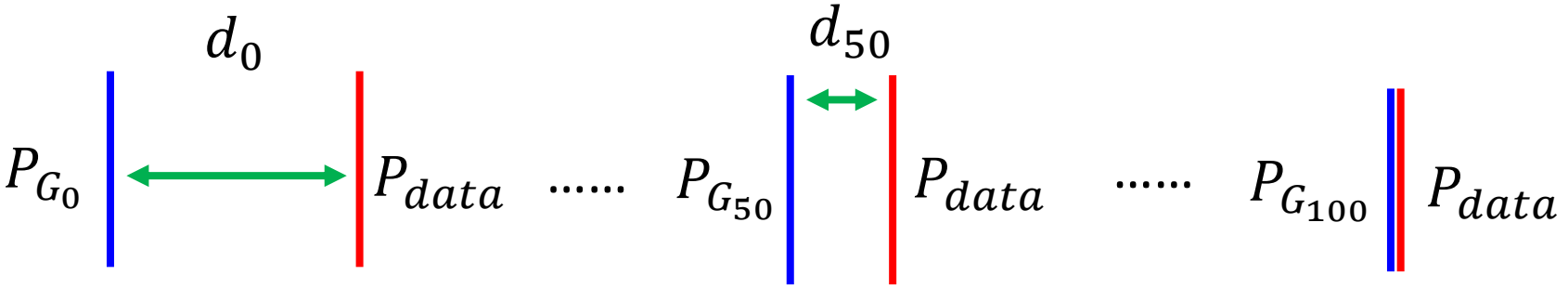


Why Earth Mover's Distance?

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

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





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

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

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

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

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

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

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

WGAN

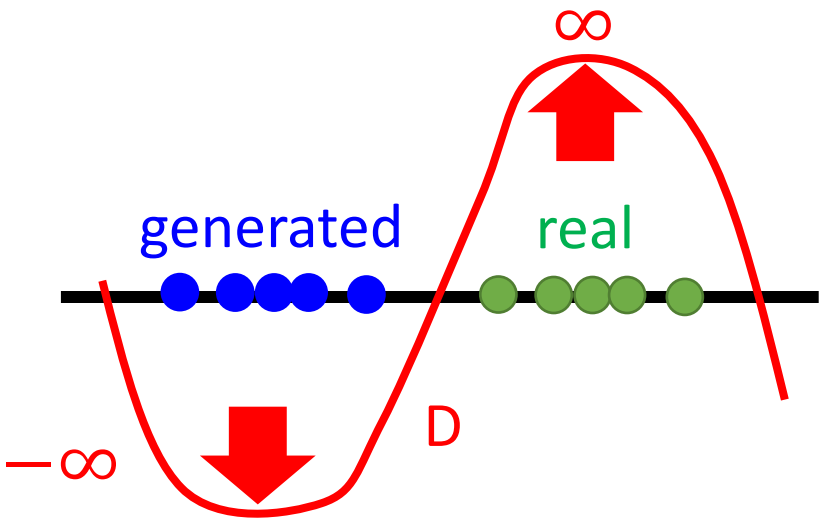
Evaluate wasserstein distance between P_{data} and P_G

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

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 $-\infty$



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

WGAN

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

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

D has to be smooth enough. How to fulfill this constraint?

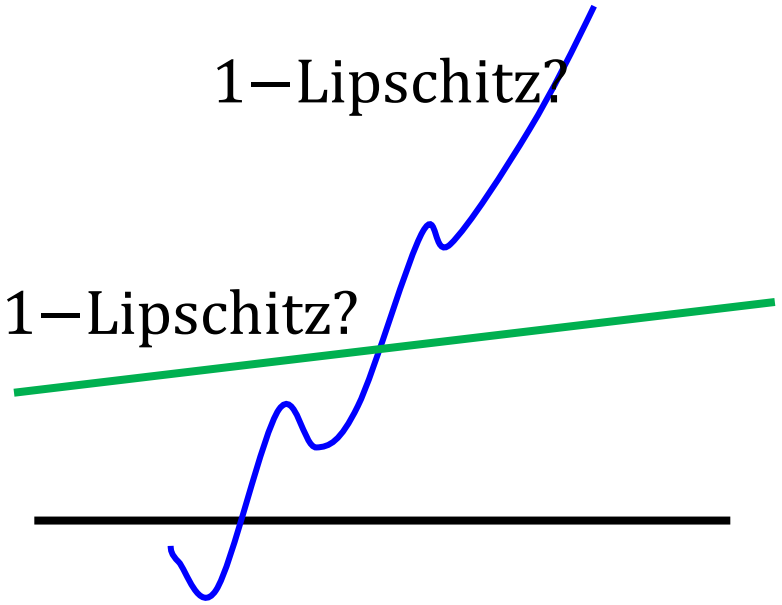
Lipschitz Function

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

Output change / Input change

K=1 for "$1-Lipschitz$"

Do not change fast



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.

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

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

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

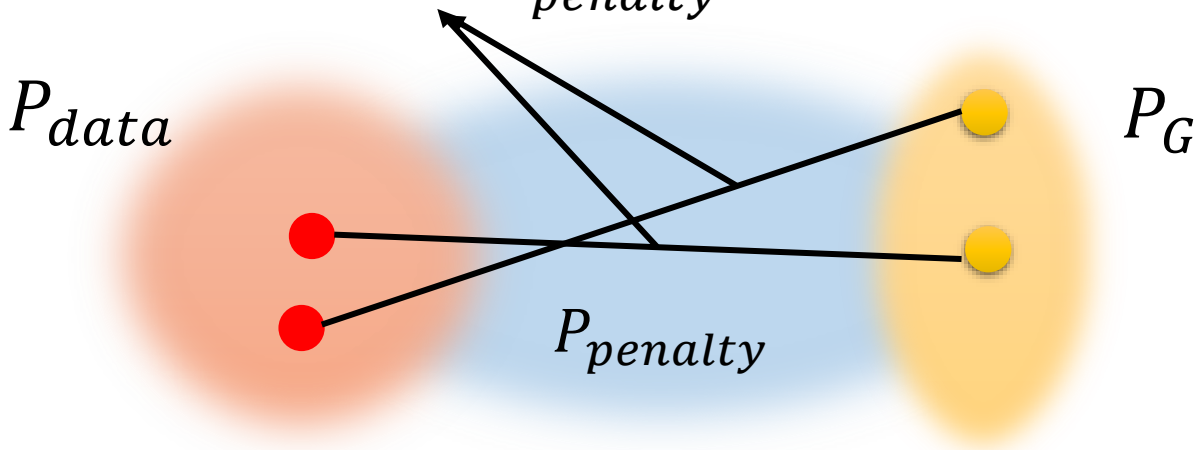
Prefer $\|\nabla_x D(x)\| \leq 1$ for all x

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

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

Improved WGAN (WGAN-GP)

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



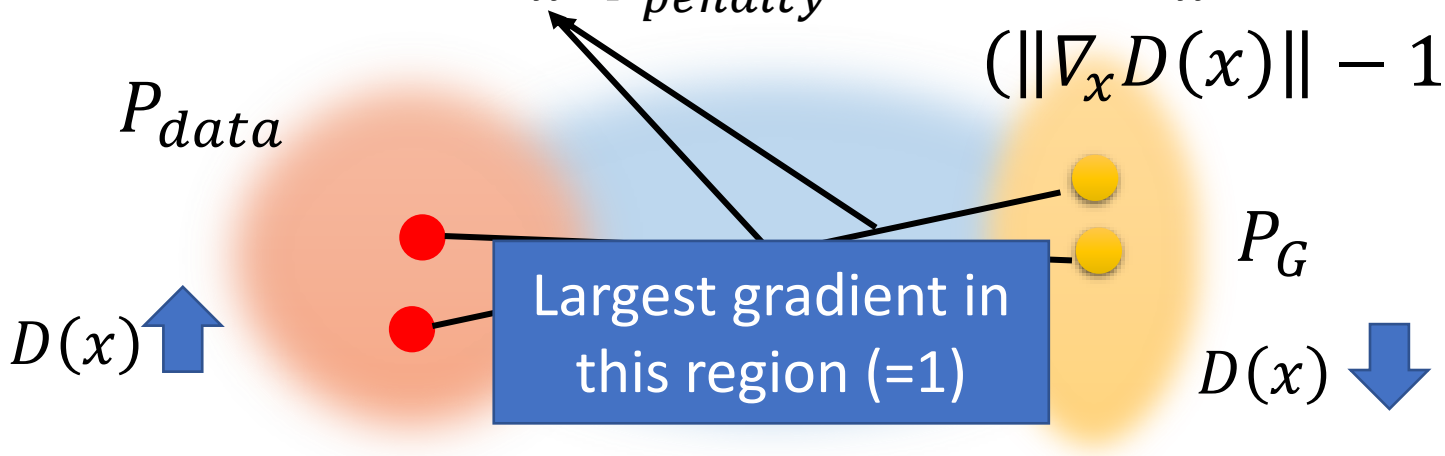
"Given that enforcing the Lipschitz constraint everywhere is intractable, enforcing it ***only along these straight lines*** seems sufficient and experimentally results in good performance."

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

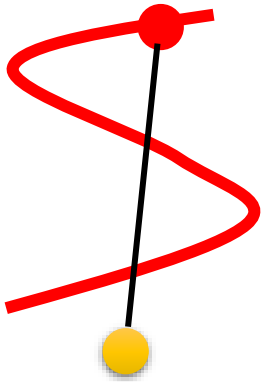
Improved WGAN (WGAN-GP)

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

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



"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]

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 / Gradient Penalty …

Learning G

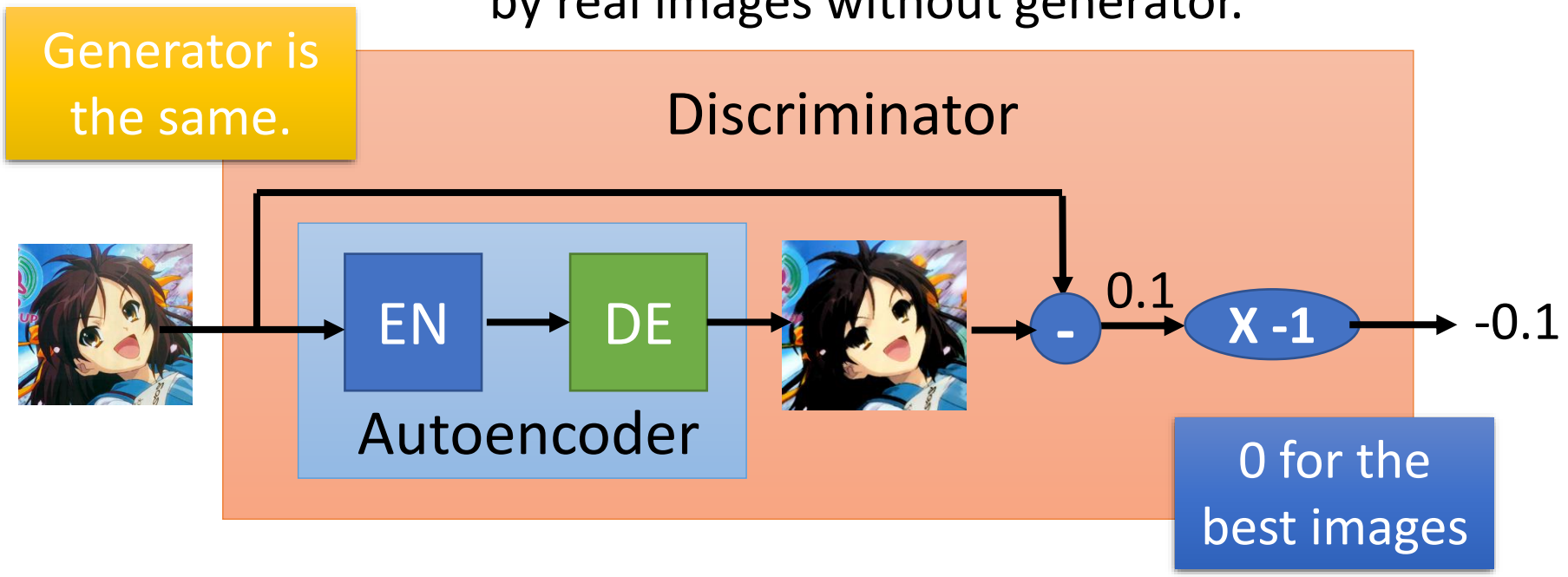
Only Once

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

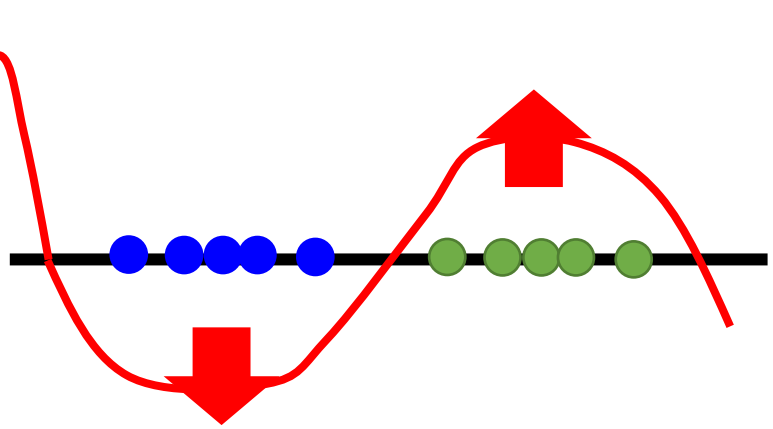
[Junbo Zhao, et al., arXiv, 2016]

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.

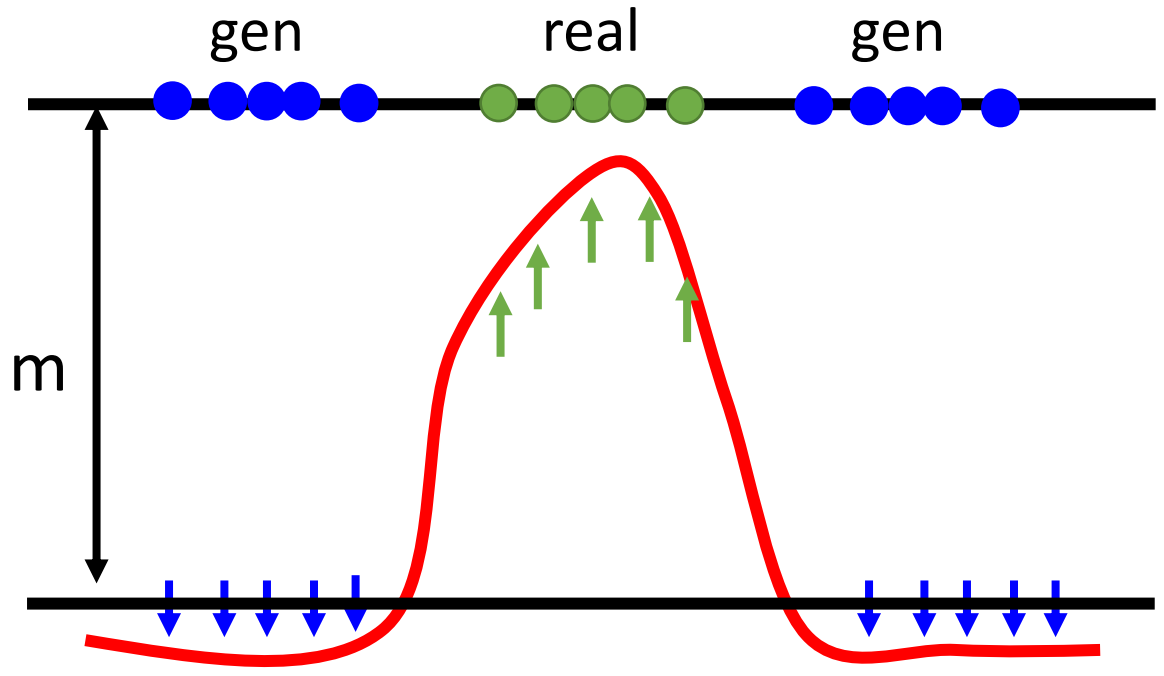


EBGAN



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

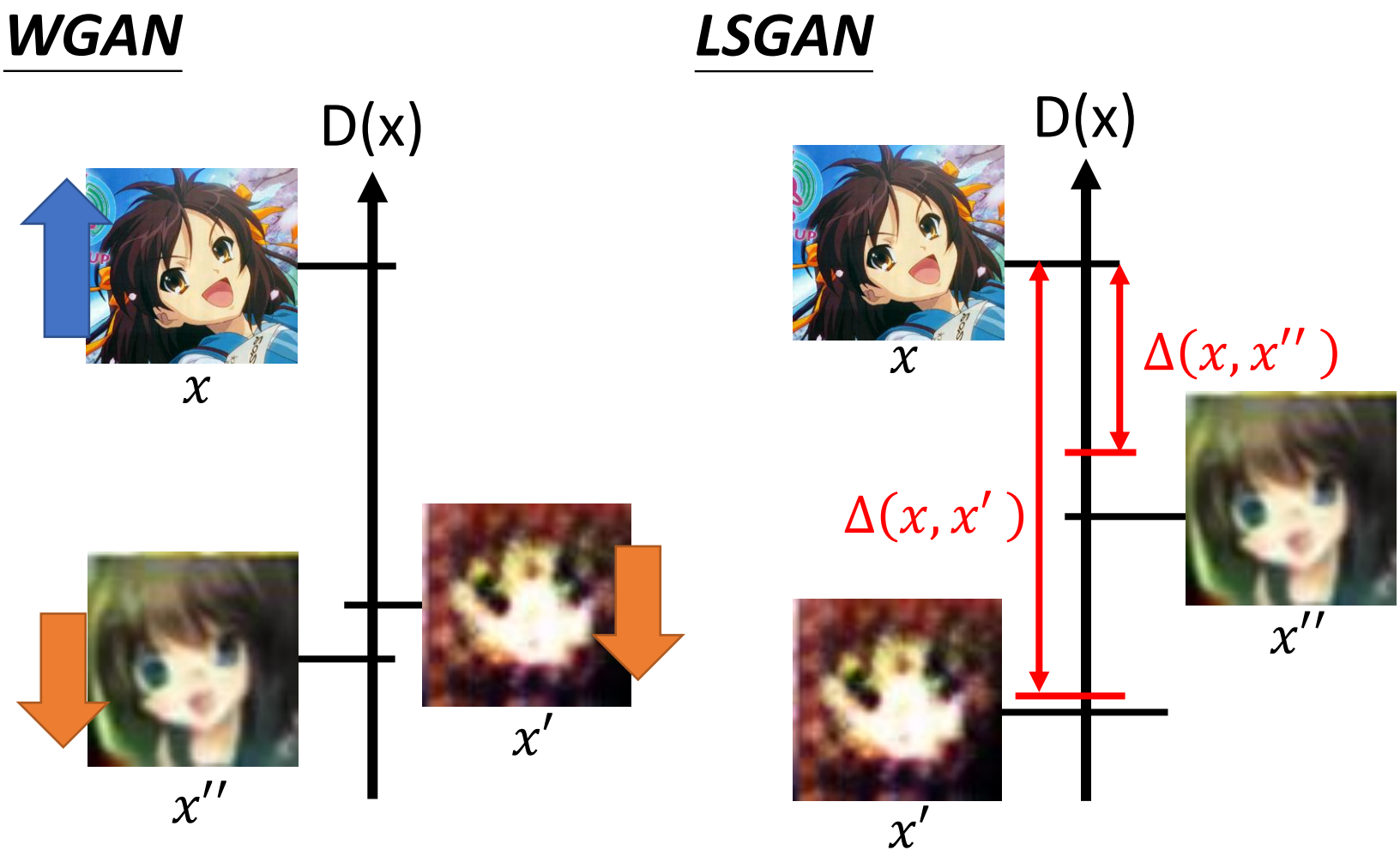
0 is for
the best.



Do not have to
be very negative

Hard to reconstruct, easy to destroy

Outlook:
Loss-sensitive GAN (LSGAN)



Reference

- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative Adversarial Networks, NIPS, 2014
- Sebastian Nowozin, Botond Cseke, Ryota Tomioka, “f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization”, NIPS, 2016
- Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, arXiv, 2017
- Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville, Improved Training of Wasserstein GANs, NIPS, 2017
- Junbo Zhao, Michael Mathieu, Yann LeCun, Energy-based Generative Adversarial Network, arXiv, 2016
- Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, Olivier Bousquet, “Are GANs Created Equal? A Large-Scale Study”, arXiv, 2017
- Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen Improved Techniques for Training GANs, NIPS, 2016

Reference

- Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter, GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, NIPS, 2017
- Naveen Kodali, Jacob Abernethy, James Hays, Zsolt Kira, “On Convergence and Stability of GANs”, arXiv, 2017
- Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, Liqiang Wang, Improving the Improved Training of Wasserstein GANs: A Consistency Term and Its Dual Effect, ICLR, 2018
- Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida, Spectral Normalization for Generative Adversarial Networks, ICLR, 2018