Generative Adversarial Network (GAN)

Slides credit: Dr. Hung-Yi Lee

Updates:

- HW 4 out
  - Due August 15, 11:59 PM
  - No late submissions!

All Kinds of GAN …

https://github.com/hindupuravinash/the-gan-zoo

GAN

ACGAN

BGAN

CGAN

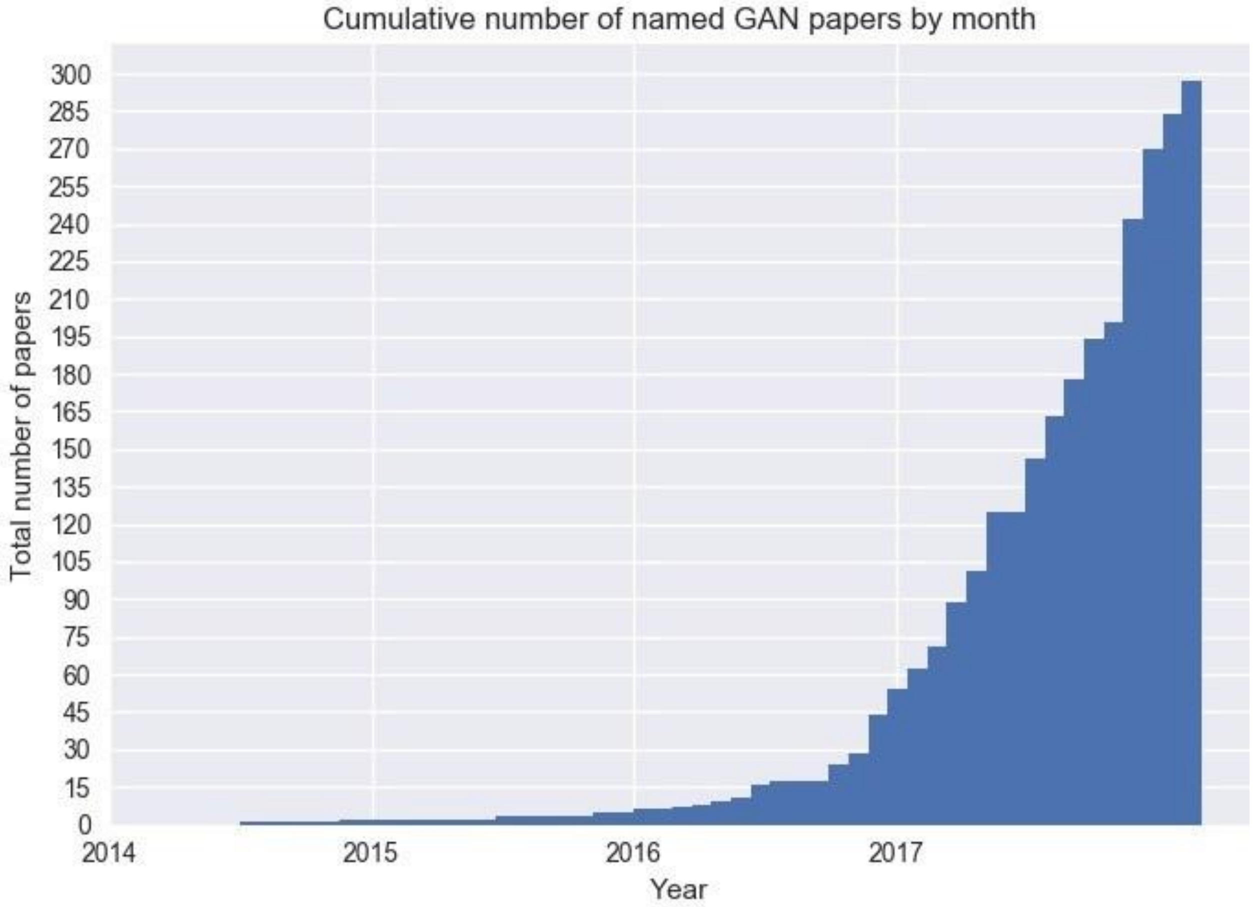
DCGAN

EBGAN

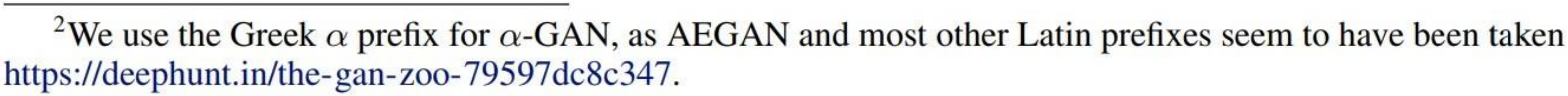
fGAN

GoGAN

⋮

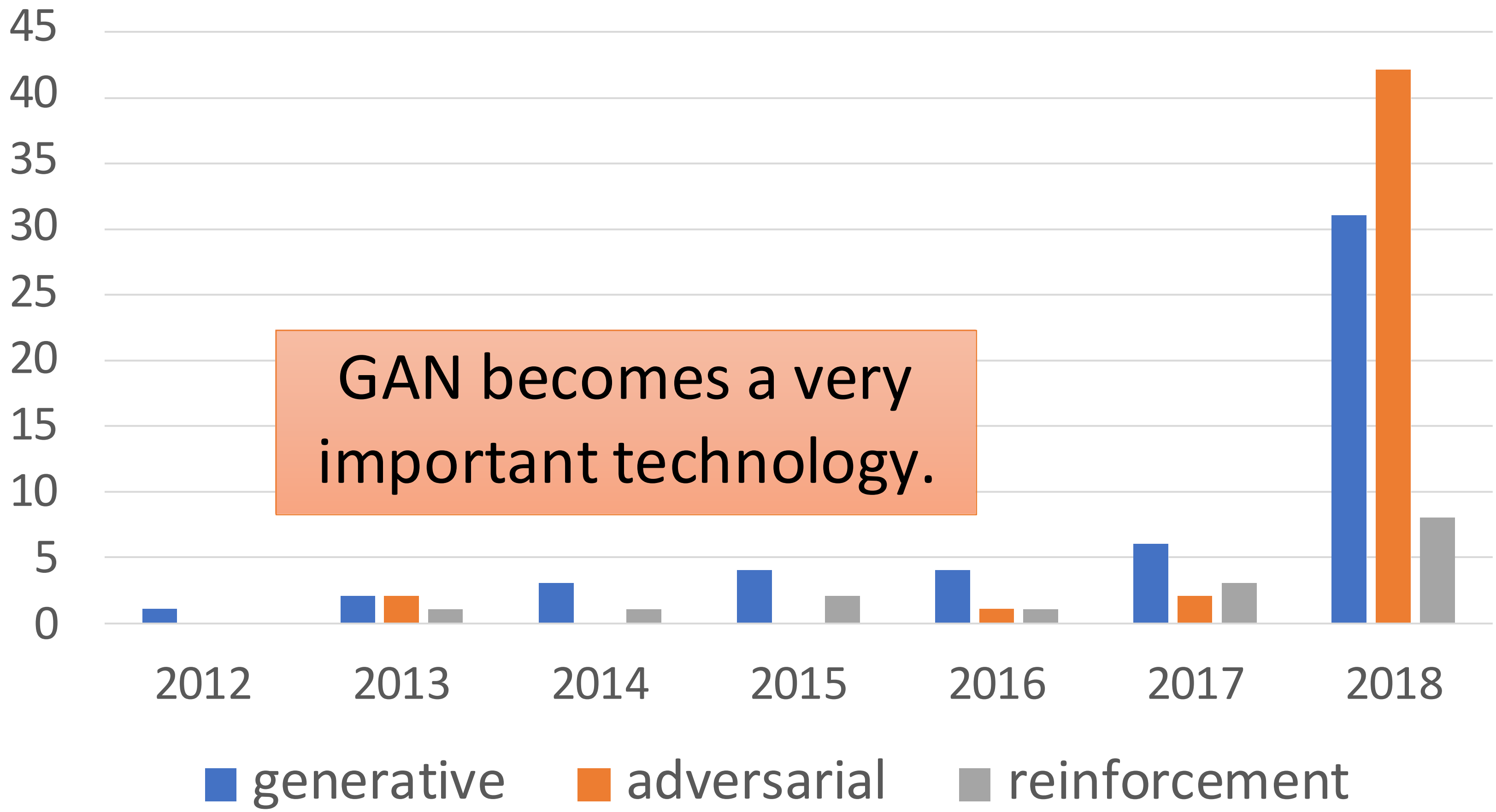


Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, Shakir Mohamed, "Variational Approaches for Auto-Encoding Generative Adversarial Networks", arXiv, 2017



[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.

Number of papers whose titles include the keyword



GAN becomes a very important technology.

Outline

Basic Idea of GAN

GAN as structured learning

Can Generator learn by itself?

Can Discriminator generate?

A little bit theory

Generation

Not very useful?
We will control what to generate latter
→ Conditional Generation

Image Generation

$$\begin{bmatrix}0.3\\-0.1\\\vdots\\-0.7\end{bmatrix}\begin{bmatrix}0.1\\-0.1\\\vdots\\0.7\end{bmatrix}\begin{bmatrix}-0.3\\0.1\\\vdots\\0.9\end{bmatrix} \rightarrow \boxed{\text{NN Generator}} \rightarrow$$

In a specific range

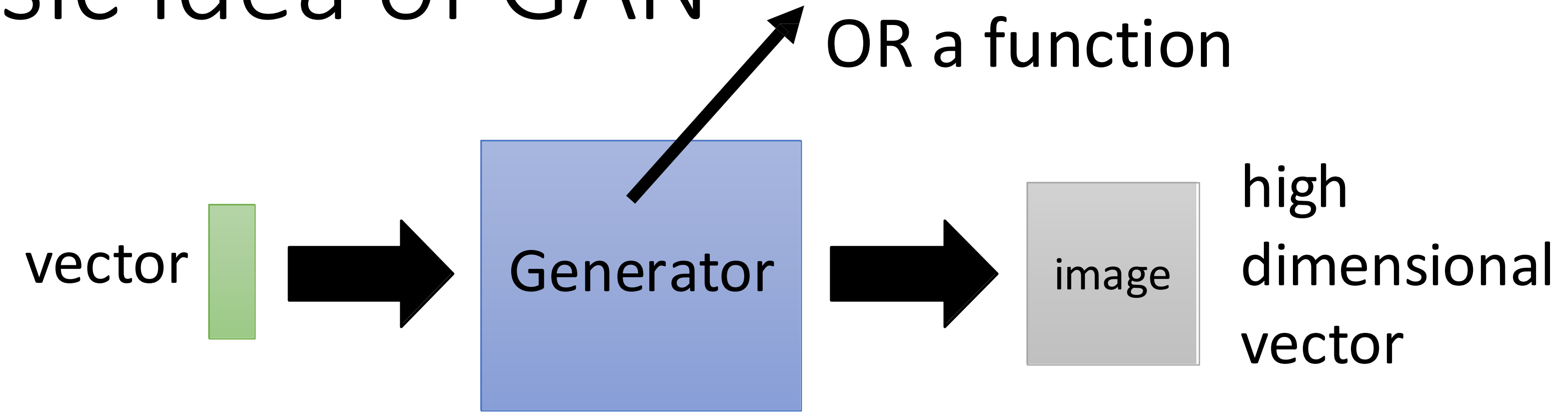
Sentence Generation

$$\begin{bmatrix}0.3\\-0.1\\\vdots\\-0.7\end{bmatrix}\begin{bmatrix}0.1\\-0.1\\\vdots\\0.2\end{bmatrix}\begin{bmatrix}-0.3\\0.1\\\vdots\\0.5\end{bmatrix} \rightarrow \boxed{\text{NN Generator}} \rightarrow$$

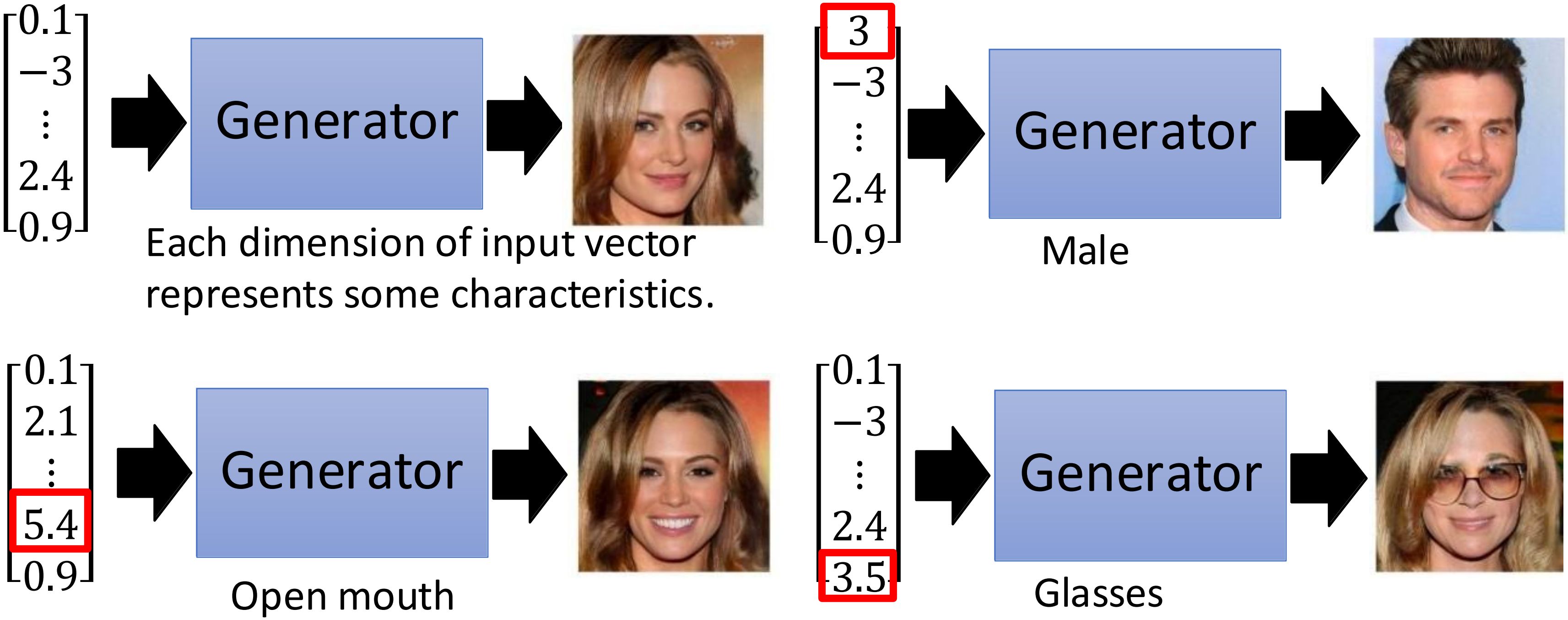
How are you?
Good morning.
Good afternoon.

Powered by: http://mattya.github.io/chainer-DCGAN/

Basic Idea of GAN



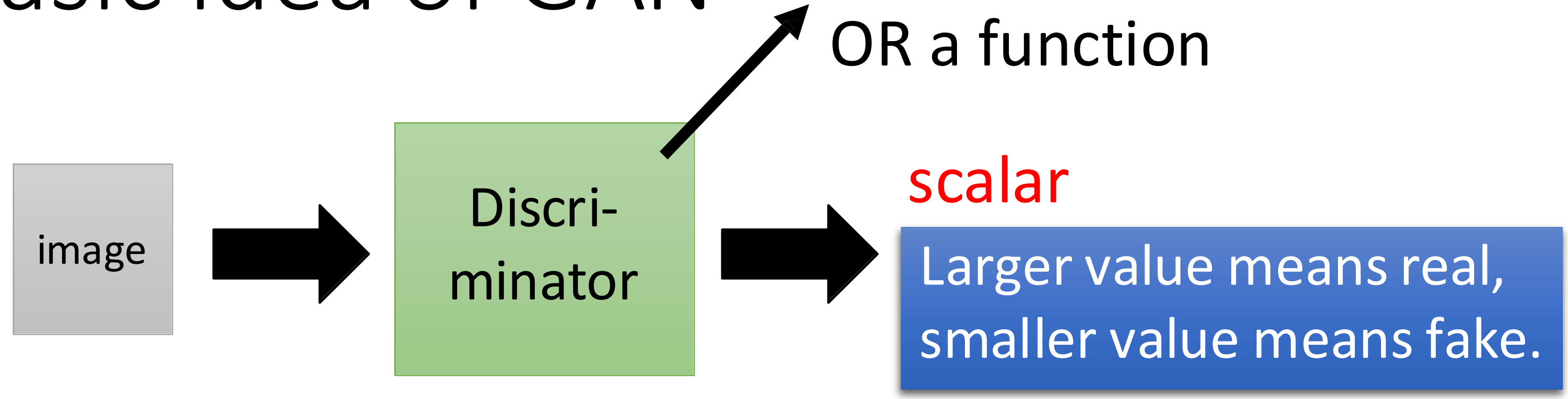
It is a neural network (NN) OR a function



Each dimension of input vector represents some characteristics.

Basic Idea of GAN

It is a neural network (NN)
OR a function

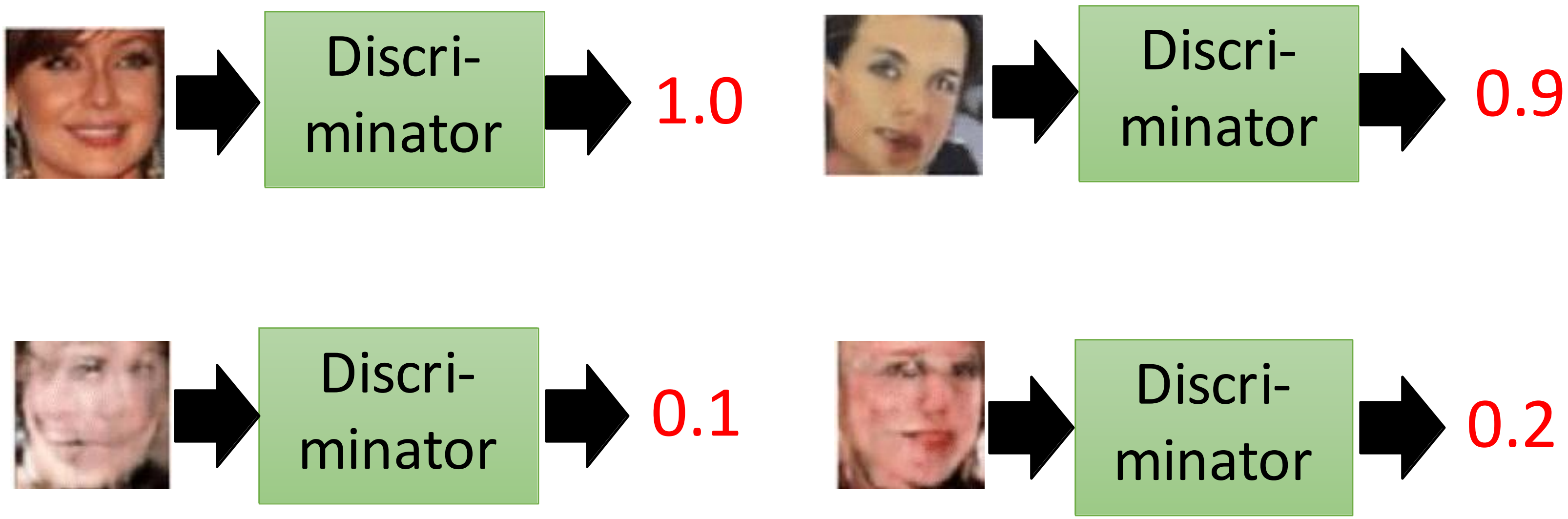


image

Discriminator

scalar

Larger value means real,
smaller value means fake.



Discriminator

1.0

Discriminator

0.9

Discriminator

0.1

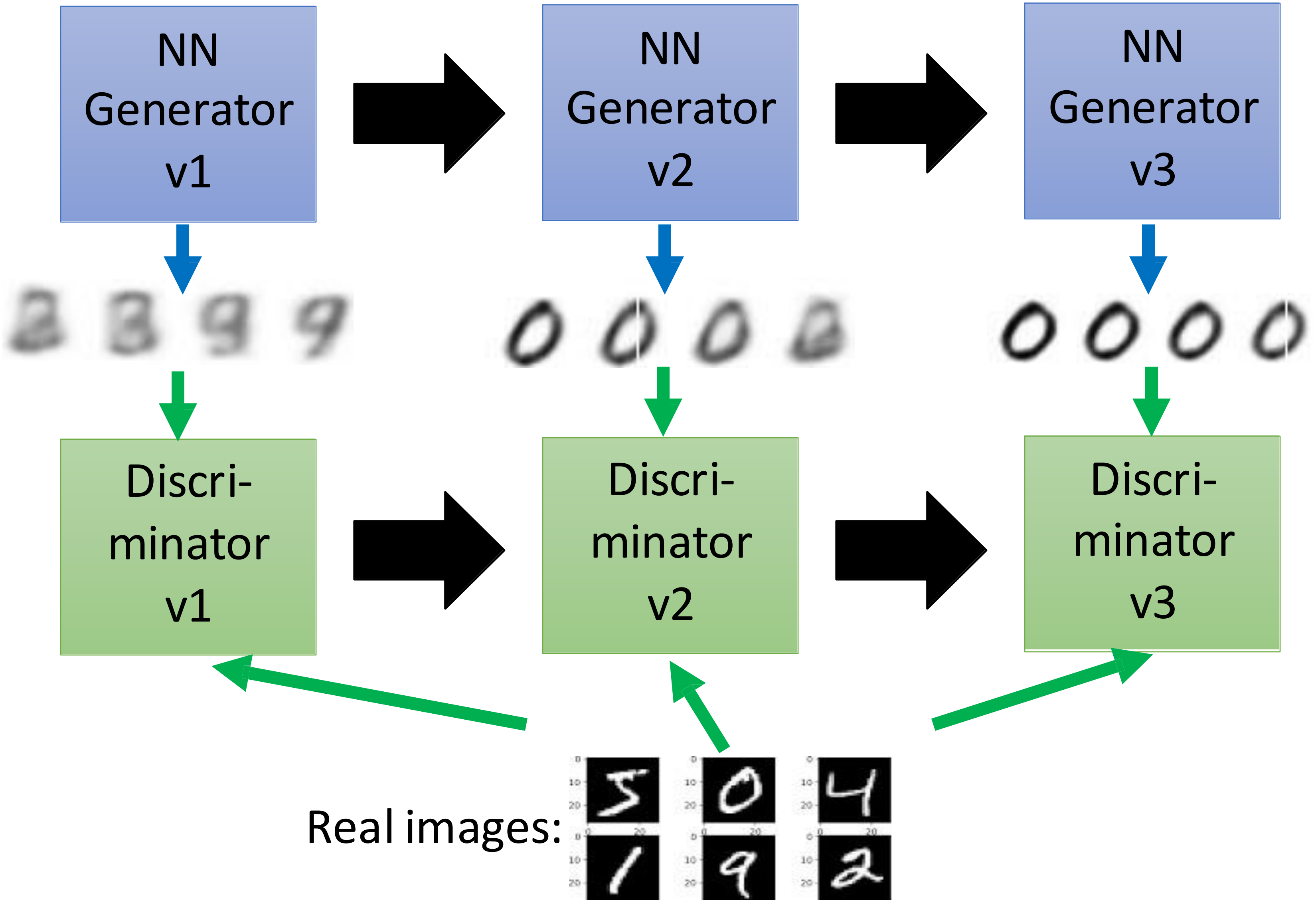
Discriminator

0.2

Basic Idea of GAN

This is where the term “***adversarial***” comes from.

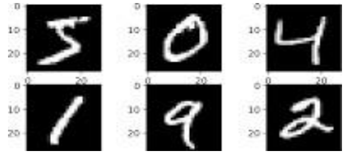
Prey ←→ Predator



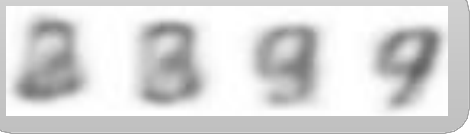
Basic Idea of GAN

Generator (student)

Discriminator (teacher)



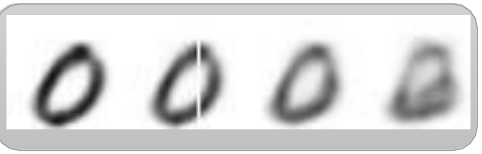
Generator v1



Discriminator v1

All are blur

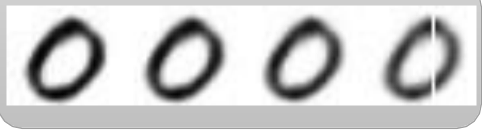
Generator v2



Discriminator v2

Some are blur

Generator v3



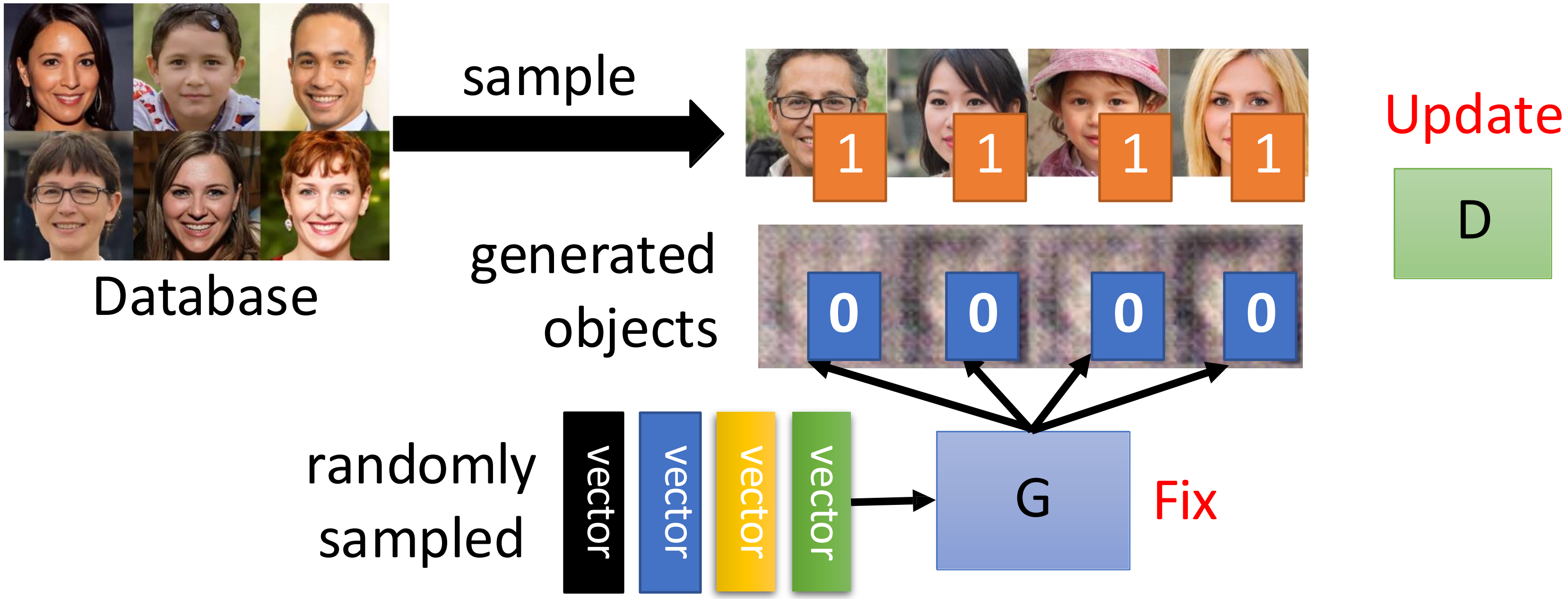
Why not learn directly?

Why not generate directly?

Algorithm

- Initialize generator and discriminator G D
- In each training iteration:

Step 1: Fix generator G, and update discriminator D



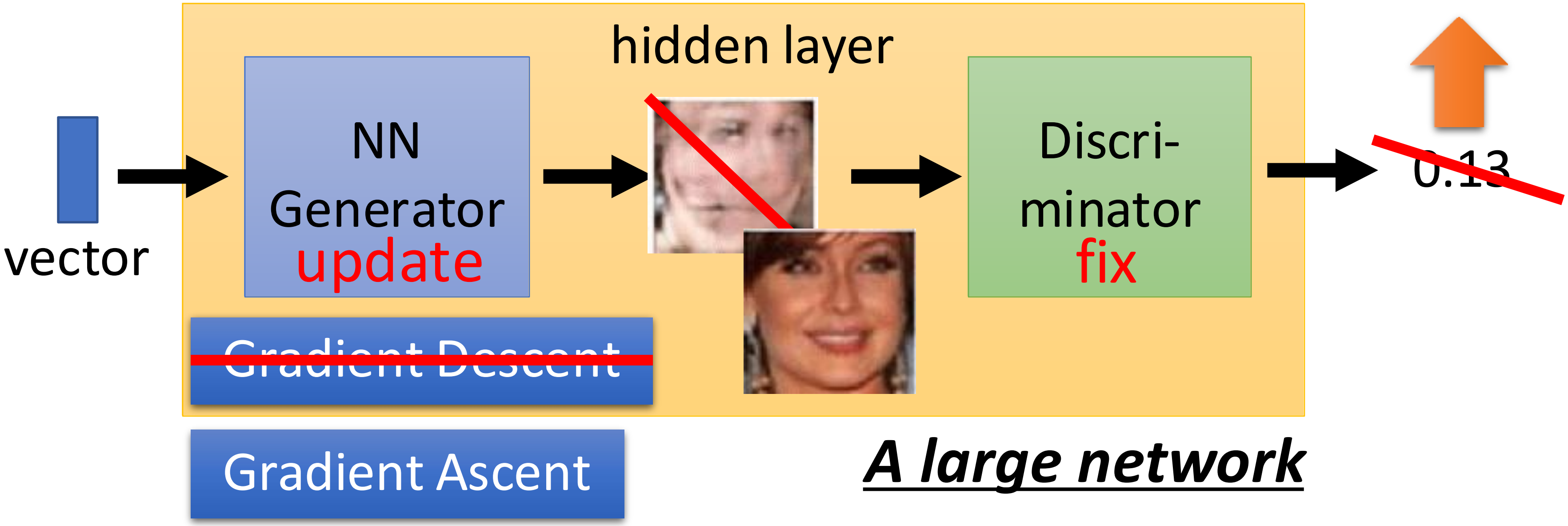
Discriminator learns to assign high scores to real objects and low scores to generated objects.

Algorithm

- Initialize generator and discriminator G D
- In each training iteration:

Step 2: Fix discriminator D, and update generator G

Generator learns to "fool" the discriminator



A large network

We want the output of the large network to be?

Algorithm Initialize θ_d for D and θ_g for G

- In each training iteration:

Learning
D

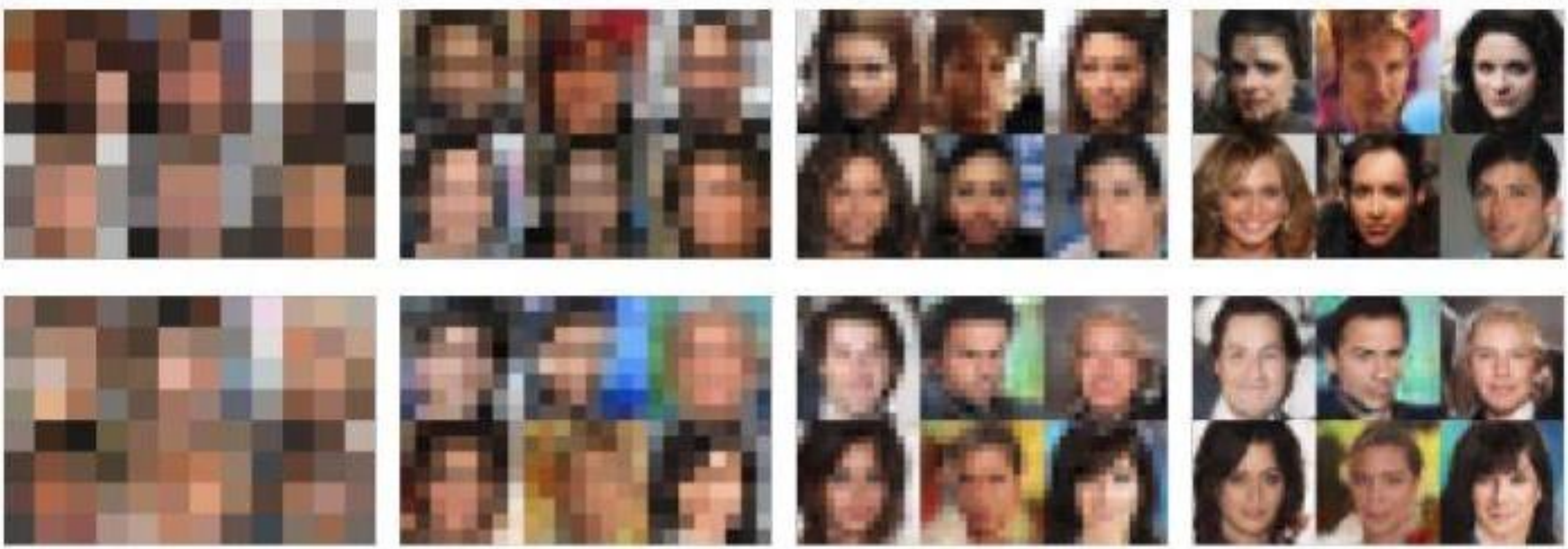
- Sample m examples $\{x^1, x^2, \dots, x^m\}$ from database
- Sample m noise samples $\{z^1, z^2, \dots, z^m\}$ from a distribution
- Obtaining generated data $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$, $\tilde{x}^i = G\left(z^i\right)$
- Update discriminator parameters θ_d to maximize
  - $\tilde{V}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

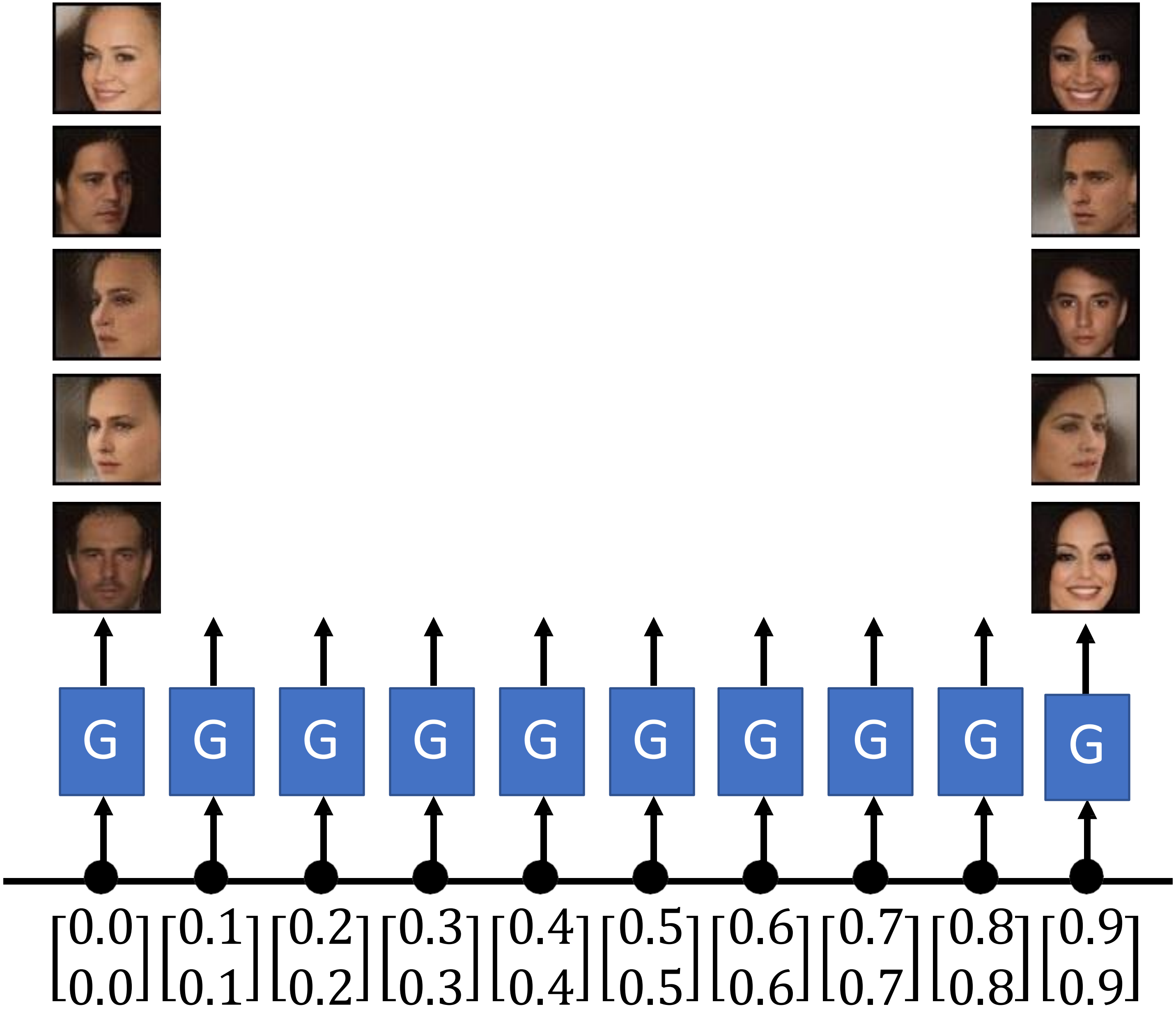
Learning
G

- Sample m noise samples$\{z^1, z^2, \dots, z^m\}$ from a distribution
- Update generator parameters θ_g to maximize
  - $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m} log\left(D\left(G\left(z^i\right)\right)\right)$
  - $\theta_g \leftarrow \theta_g + \eta\nabla\tilde{V}\left(\theta_g\right)$

Training iterations

1,000 2,000 4,000 8,000




G
G
G
G
G
G
G
G
G
G
$\begin{bmatrix}0.0\\0.0\end{bmatrix}$
$\begin{bmatrix}0.1\\0.1\end{bmatrix}$
$\begin{bmatrix}0.2\\0.2\end{bmatrix}$
$\begin{bmatrix}0.3\\0.3\end{bmatrix}$
$\begin{bmatrix}0.4\\0.4\end{bmatrix}$
$\begin{bmatrix}0.5\\0.5\end{bmatrix}$
$\begin{bmatrix}0.6\\0.6\end{bmatrix}$
$\begin{bmatrix}0.7\\0.7\end{bmatrix}$
$\begin{bmatrix}0.8\\0.8\end{bmatrix}$
$\begin{bmatrix}0.9\\0.9\end{bmatrix}$

We can control the generation.

Outline

Basic Idea of GAN

GAN as structured learning

Can Generator learn by itself?

Can Discriminator generate?

A little bit theory

Structured Learning

Machine learning is to find a function f

$$f : X \rightarrow Y$$

Regression: output a scalar

Classification: output a “class”   (one-hot vector)

1	0	0

Class 1

0	1	0

Class 2

0	0	1

Class 3

Structured Learning/Prediction: output a sequence, a matrix, a graph, a tree ……

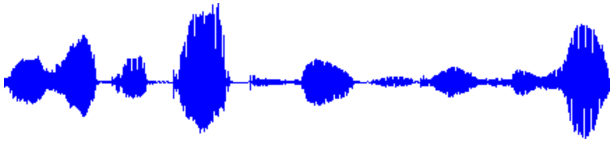
Output is composed of components with dependency

Output Sequence $f : X \to Y$

Machine Translation

X : (sentence of language 1) Y : (sentence of language 2)

Speech Recognition

X : 

(speech)

Y : “Good morning!”

(transcription)

Chat-bot

X : “How are you?”

(what a user says)

Y : “I’m fine.”

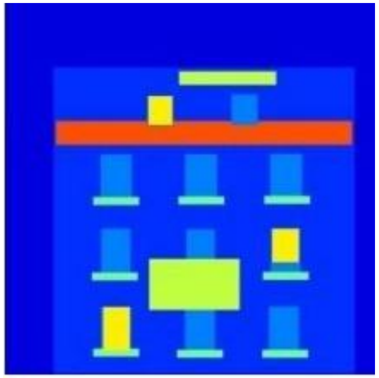
(response of machine)

Output Matrix

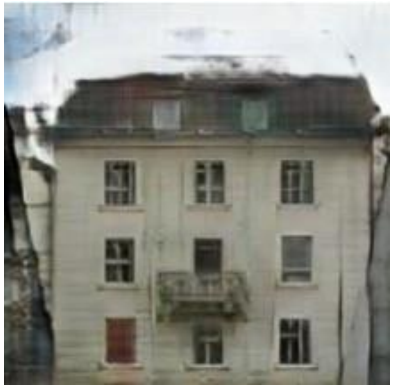
$$f : X \to Y$$

Image to Image

X :



Y :



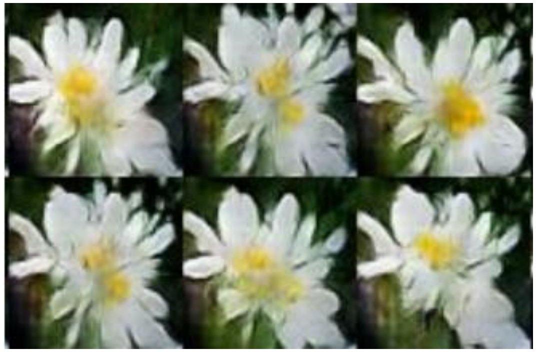
Colorization:

Ref: https://arxiv.org/pdf/1611.07004v1.pdf

Text to Image

X : "this white and yellow flower have thin white petals and a round yellow stamen"

Y :



ref: https://arxiv.org/pdf/1605.05396.pdf

Why Structured Learning Challenging?

- **One-shot/Zero-shot Learning**:
  - In classification, each class has some examples.
  - In structured learning,
    - If you consider each possible output as a “class”

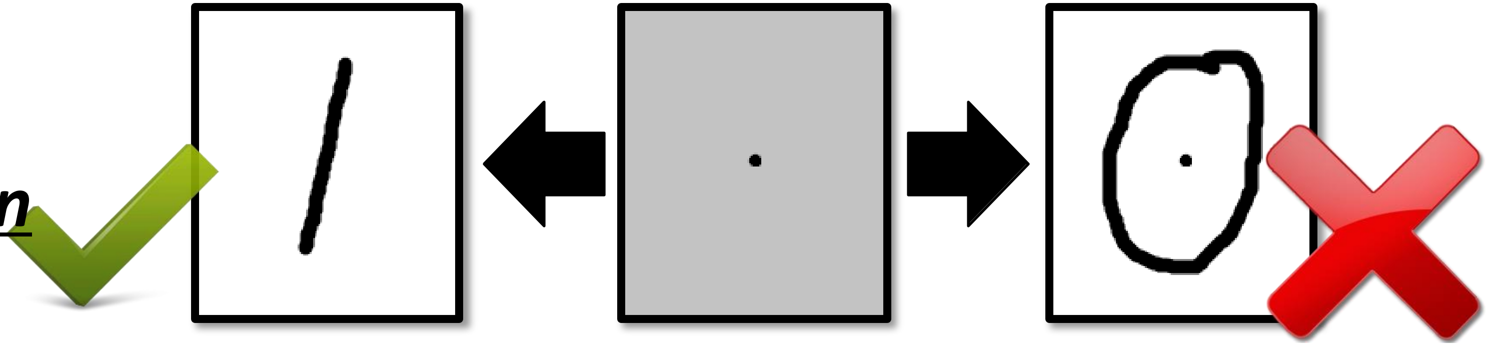
      → One-shot learning;
    - Since the output space is huge, most “classes” may not have any training data

      → Zero-shot learning;
    - Machine has to create new stuff during testing.
    - Need more intelligence

Why Structured Learning Challenging?

- Machine has to learn to do ***planning***
  - Machine generates objects component-by-component, but it should have a big picture in its mind.
  - Since the output components have dependency, they should be considered globally.

<u>Image Generation</u>



Do not know if a pixel is a good or bad generation.

Structured Learning Approach

Generator

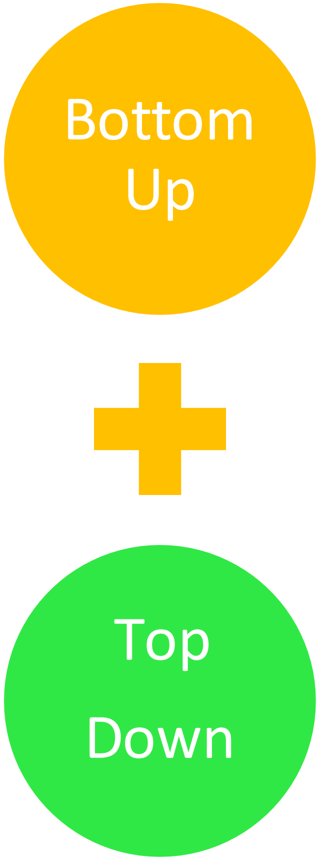
Learn to generate the object at the component level.

Cons: missing global sense.

Discriminator

Evaluating the whole object, and find the best one

Cons: missing local details.



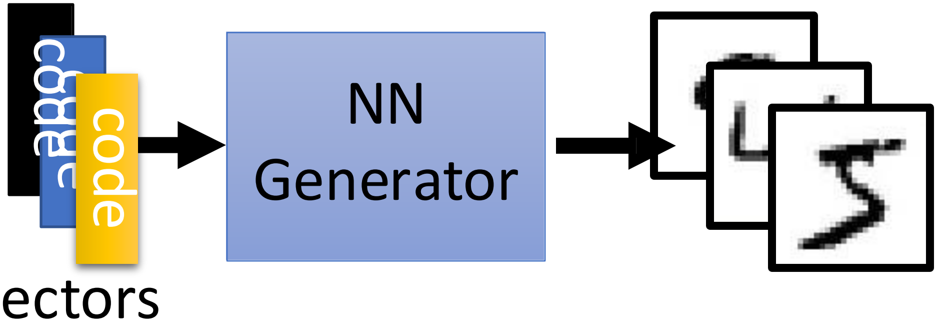
Bottom Up

Top Down

Outline

- Basic Idea of GAN
- GAN as structured learning
- Can Generator learn by itself?
- Can Discriminator generate?
- A little bit theory

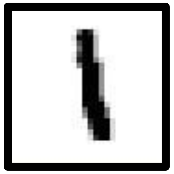
Generator

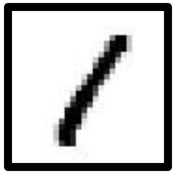


code: $\begin{bmatrix}0.1\\-0.5\end{bmatrix}$ $\begin{bmatrix}0.1\\0.9\end{bmatrix}$ $\begin{bmatrix}0.2\\-0.1\end{bmatrix}$ $\begin{bmatrix}0.3\\0.2\end{bmatrix}$

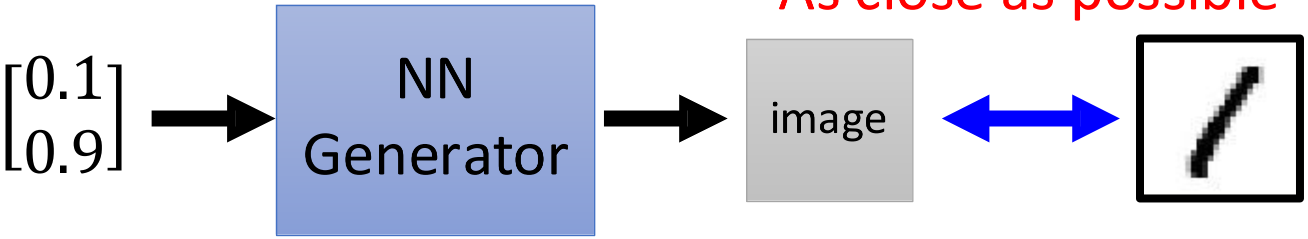
(where does they come from?)

Image:





As close as possible

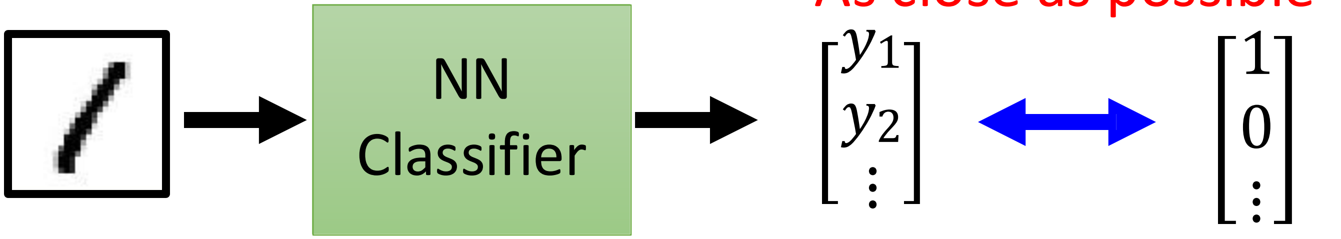

$\begin{bmatrix}0.1\\0.9\end{bmatrix}$

NN Generator

image

As close as possible

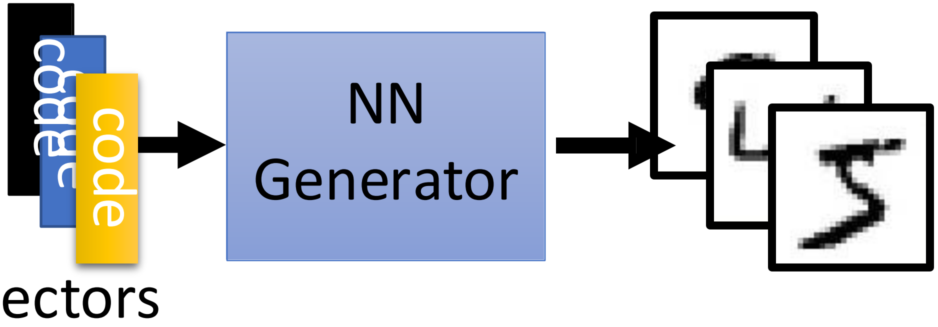
Classification:


NN Classifier

$\begin{bmatrix}y_1\\y_2\\\vdots\end{bmatrix}$

$\begin{bmatrix}1\\0\\\vdots\end{bmatrix}$

Generator

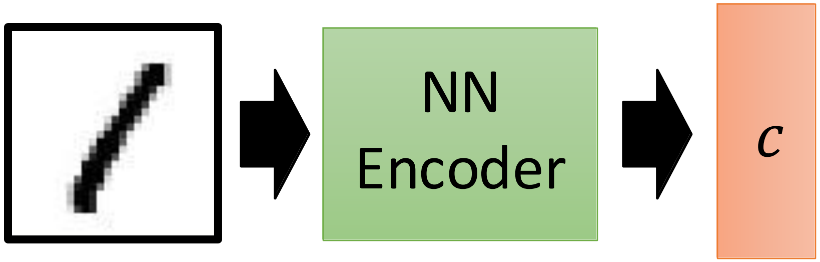


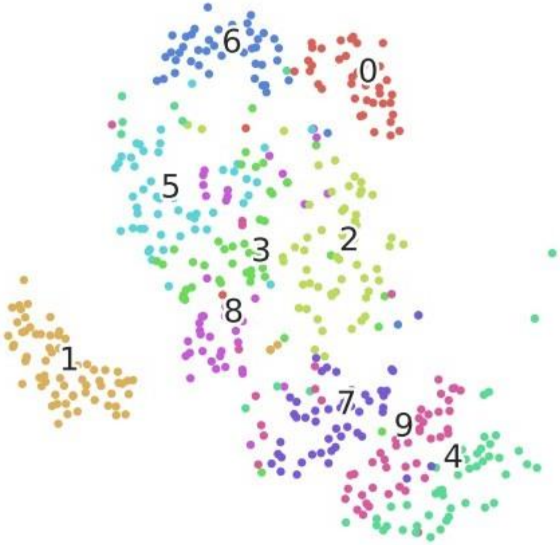
code: $\begin{bmatrix} 0.1 \\ -0.5 \end{bmatrix}$ $\begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$ $\begin{bmatrix} 0.2 \\ -0.1 \end{bmatrix}$ $\begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix}$

(where does they come from?)

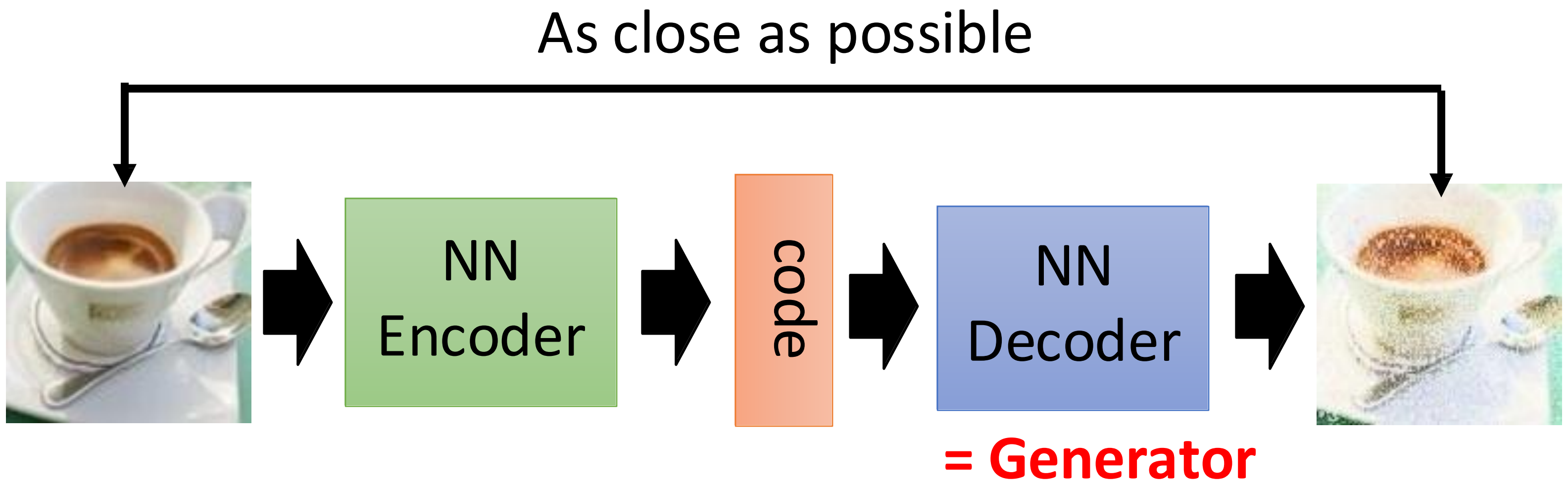
Image:

Encoder in auto-encoder provides the code ☺



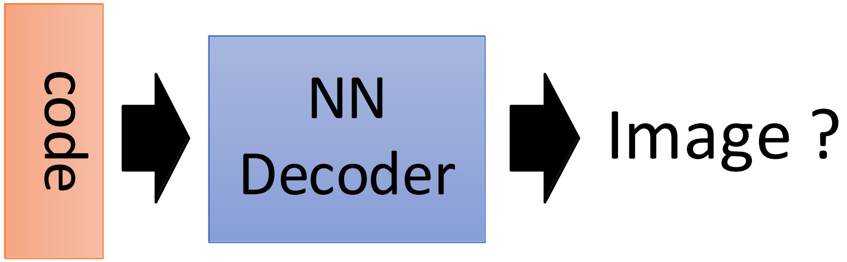


Auto-encoder



= Generator

Randomly generate
a vector as code



= Generator

Outline

Basic Idea of GAN

GAN as structured learning

Can Generator learn by itself?

Can Discriminator generate?

A little bit theory

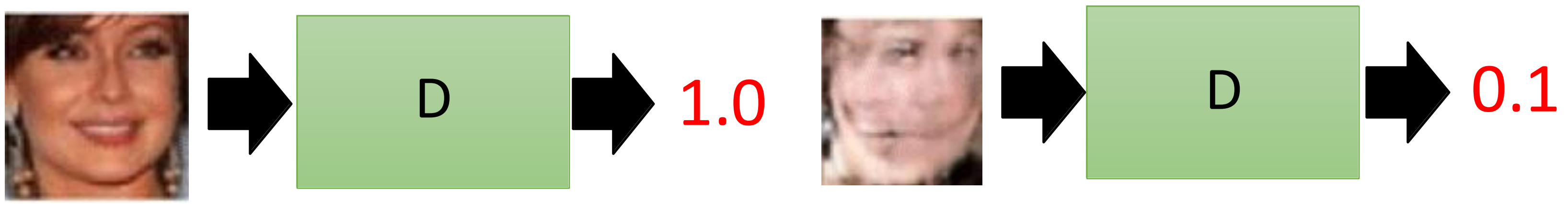
Discriminator

Evaluation function, Potential Function, Energy Function …

- Discriminator is a function D (network, can deep)

$$\mathrm{D} : X \rightarrow \mathrm{R}$$

  - Input x: an object x (e.g. an image)
  - Output D(x): a scalar which represents how “good” x is



D

1.0

D

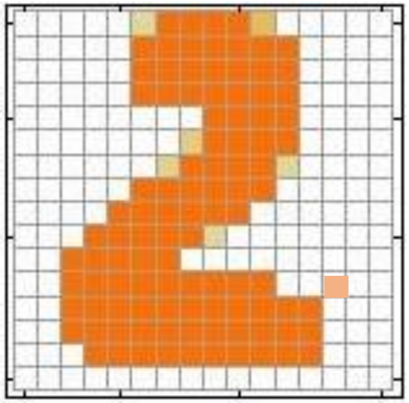
0.1

Can we use the discriminator to generate objects?

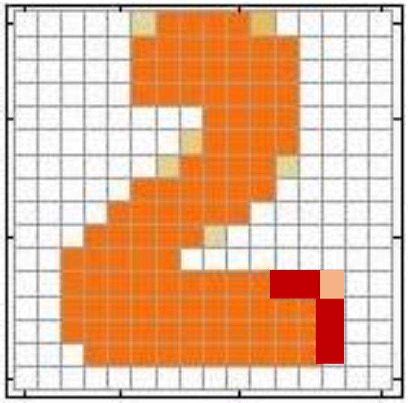
Yes.

Discriminator

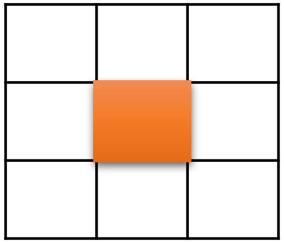
- It is easier to catch the relation between the components by top-down evaluation.



“unrealistic”



“realistic”



This CNN filter is good enough.

Discriminator

- Suppose we already have a good discriminator D(x) …

Inference

- Generate object $\tilde{x}$ that

$$\widetilde{x} = \arg\max_{x \in X} D(x)$$

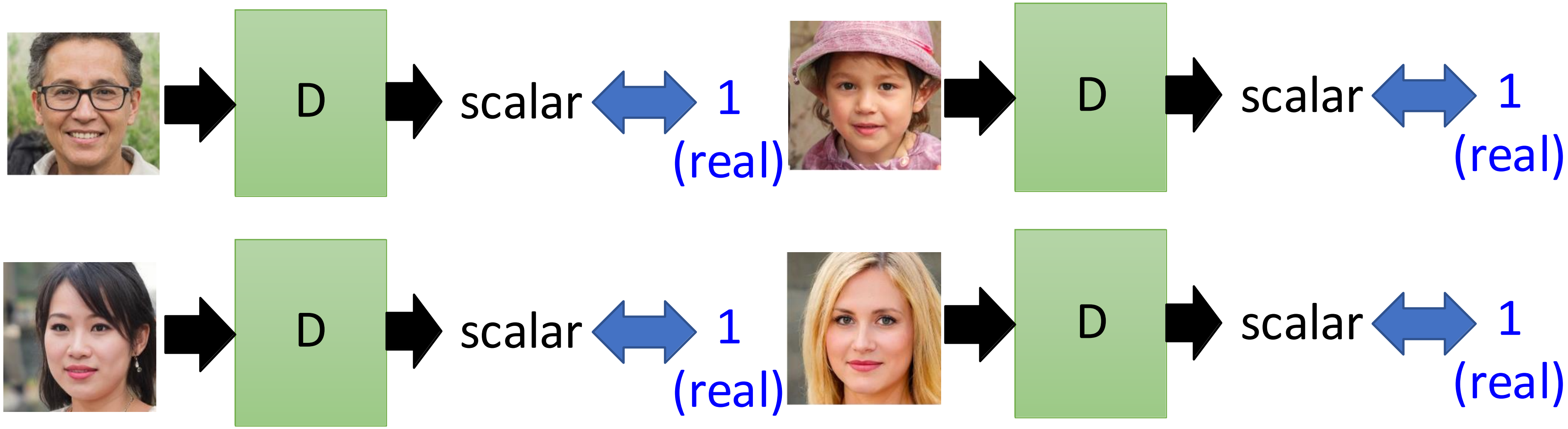
Enumerate all possible x !!!

It is feasible ???

How to learn the discriminator?

Discriminator - Training

- I have some real images



D scalar 1 (real)

D scalar 1 (real)

D scalar 1 (real)

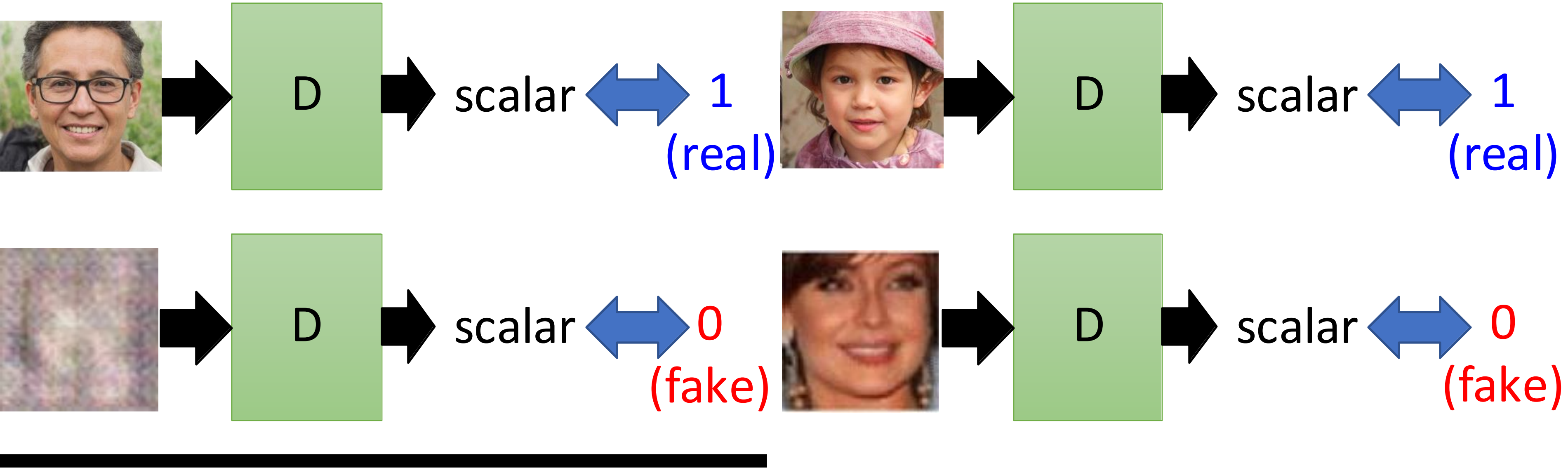
D scalar 1 (real)

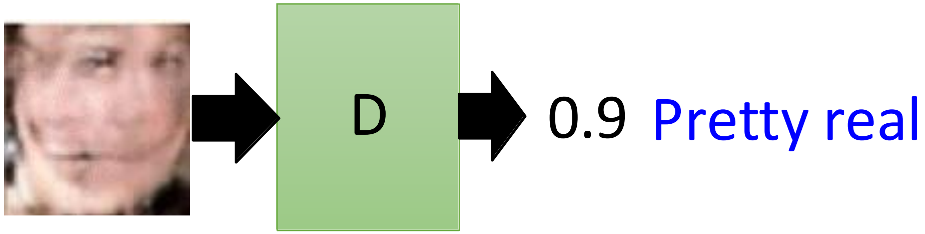
Discriminator only learns to output “1” (real).

Discriminator training needs some negative examples.

Discriminator - Training

- Negative examples are critical.





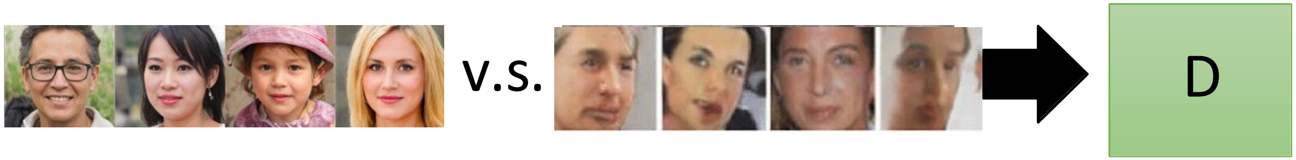
How to generate realistic negative examples?

Discriminator - Training

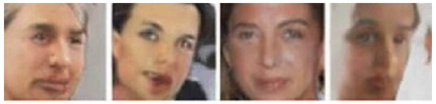
- **General Algorithm**

  - Given a set of positive examples, randomly generate a set of negative examples.

  - In each iteration
    - Learn a discriminator D that can discriminate positive and negative examples.

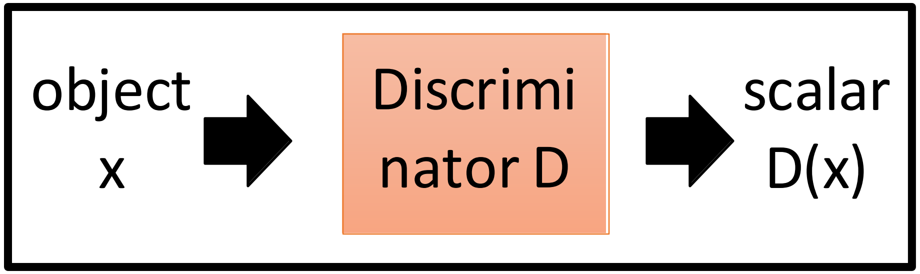


    - Generate negative examples by discriminator D



$$\tilde{x} = \arg\max_{x \in X} D(x)$$

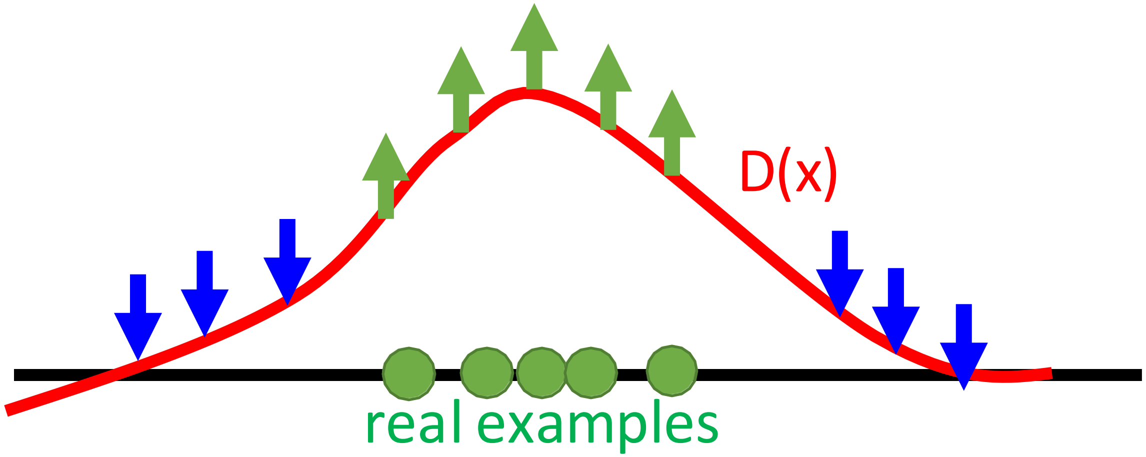
Discriminator - Training



object x

Discriminator D

scalar D(x)

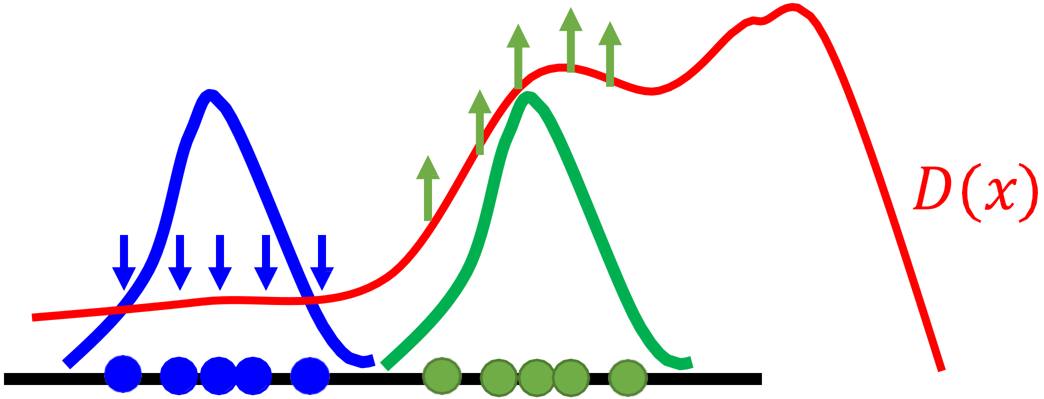


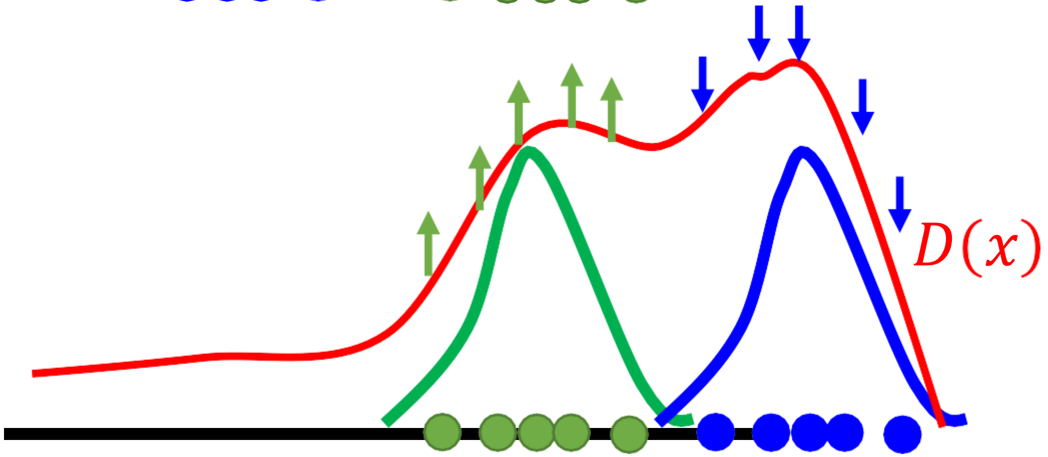
In practice, you cannot decrease all the x other than real examples.

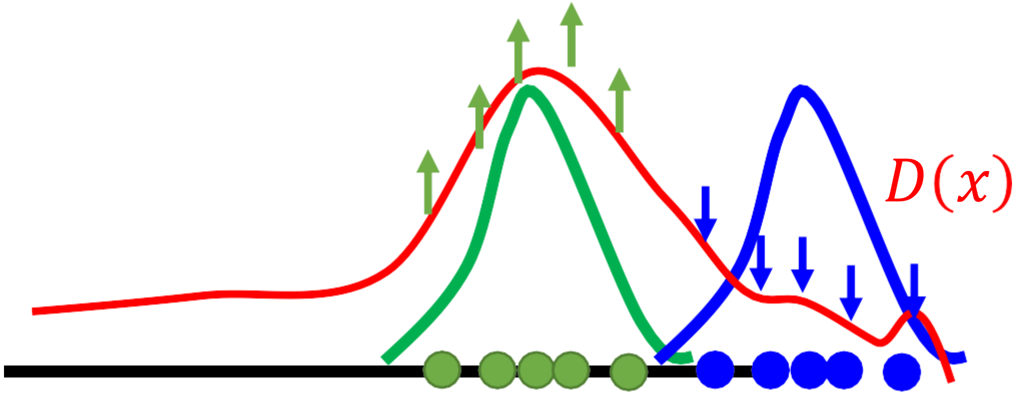
Discriminator - Training

🟢 real

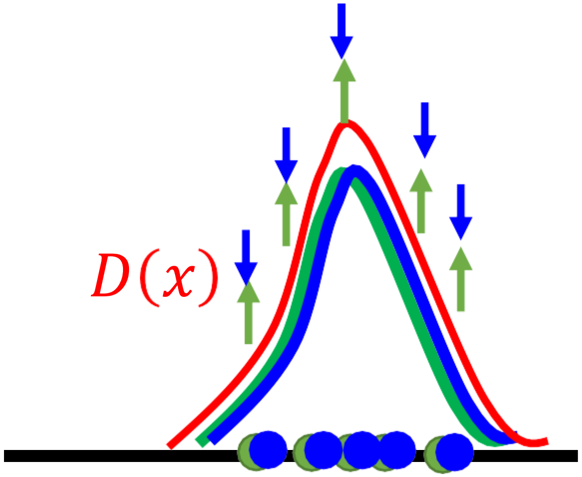
🔵 generated







In the end ……



Generator v.s. Discriminator

- ***Generator***
- Pros:
  - Easy to generate even with deep model
- Cons:
  - Imitate the appearance
  - Hard to learn the correlation between components

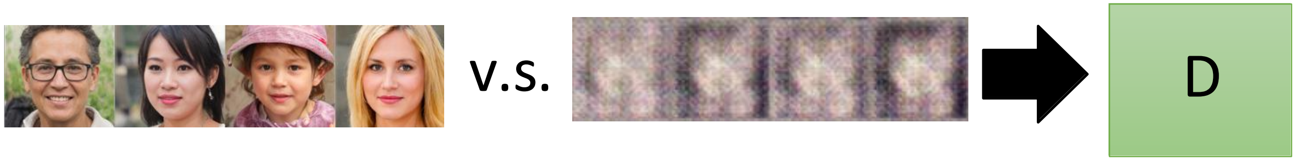
- ***Discriminator***
- Pros:
  - Considering the big picture
- Cons:
  - Generation is not always feasible
    - Especially when your model is deep
  - How to do negative sampling?

Generator + Discriminator

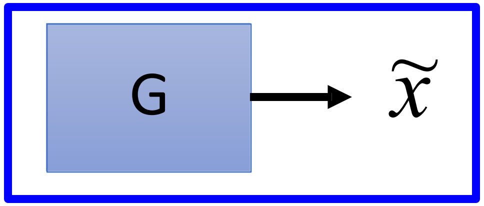
- **General Algorithm**

  - Given a set of positive examples, randomly generate a set of negative examples.

  - In each iteration
    - Learn a discriminator D that can discriminate positive and negative examples.



    - Generate negative examples by discriminator D

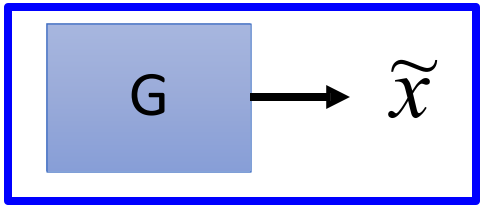


=

$$\tilde{x} = \arg\max_{x \in X} D(x)$$

Benefit of GAN

- From Discriminator’s point of view
  - Using generator to generate negative samples



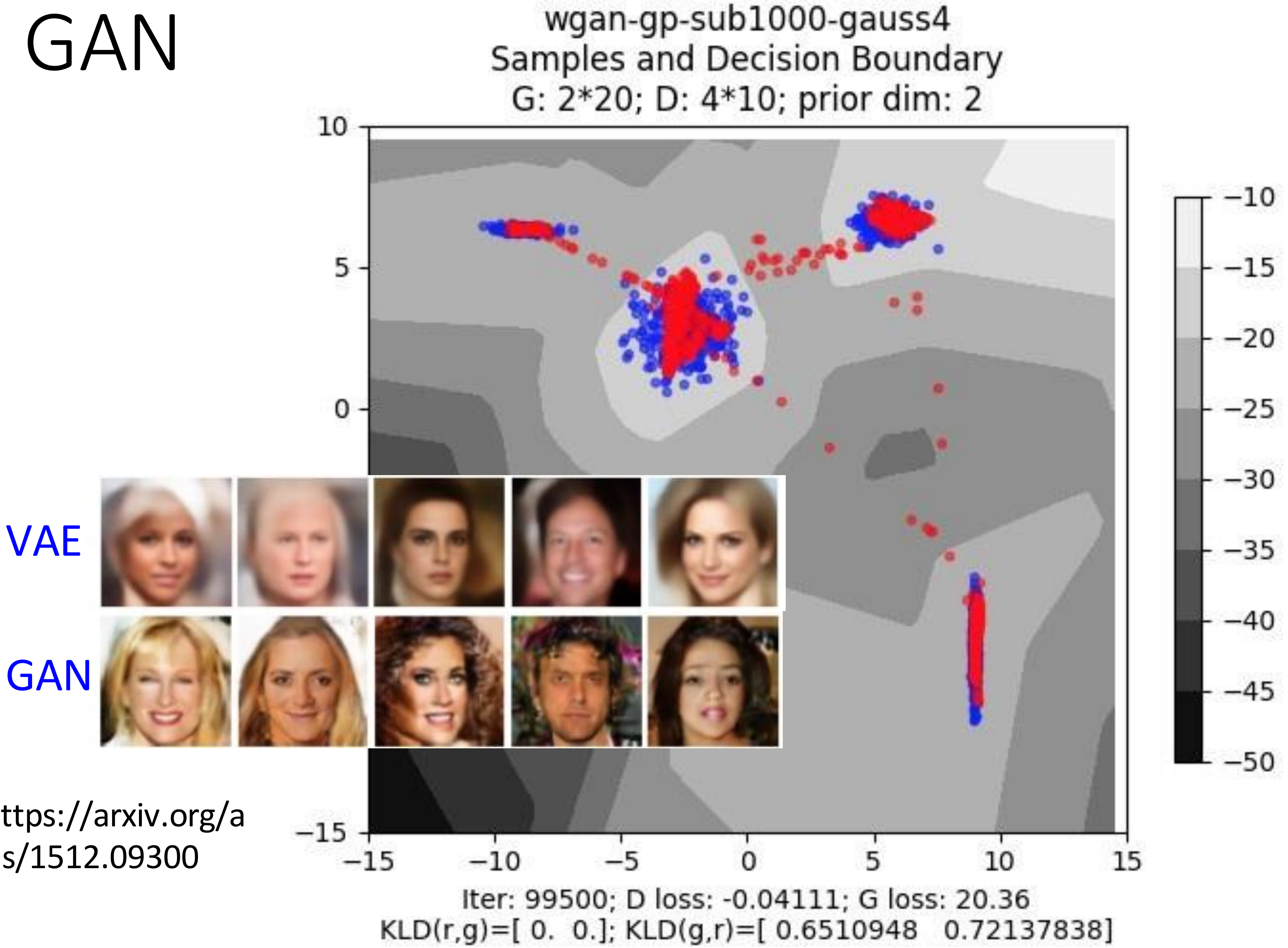
=

$$\tilde{x} = \arg\max_{x \in X} D(x)$$

efficient

- From Generator’s point of view
  - Still generate the object component-by-component
  - But it is learned from the discriminator with global view.

GAN

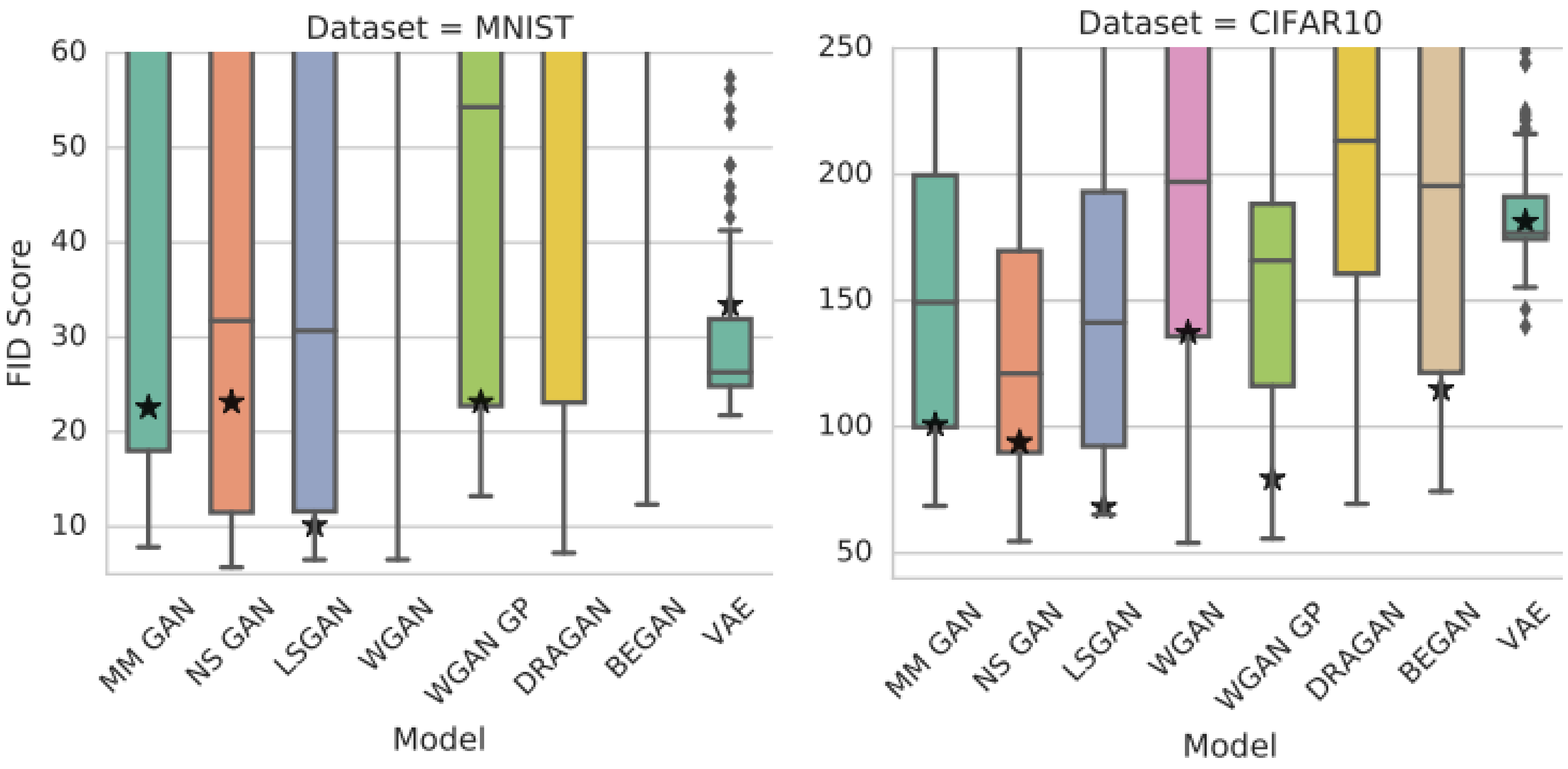


wgan-gp-sub1000-gauss4
Samples and Decision Boundary
G: 2*20; D: 4*10; prior dim: 2

Iter: 99500; D loss: -0.04111; G loss: 20.36
KLD(r,g)=[0. 0.]; KLD(g,r)=[0.6510948   0.72137838]

https://arxiv.org/abs/1512.09300

[Mario Lucic, et al. arXiv, 2017]



FID[Martin Heusel, et al., NIPS, 2017]: Smaller is better

Outline

Basic Idea of GAN

GAN as structured learning

Can Generator learn by itself?

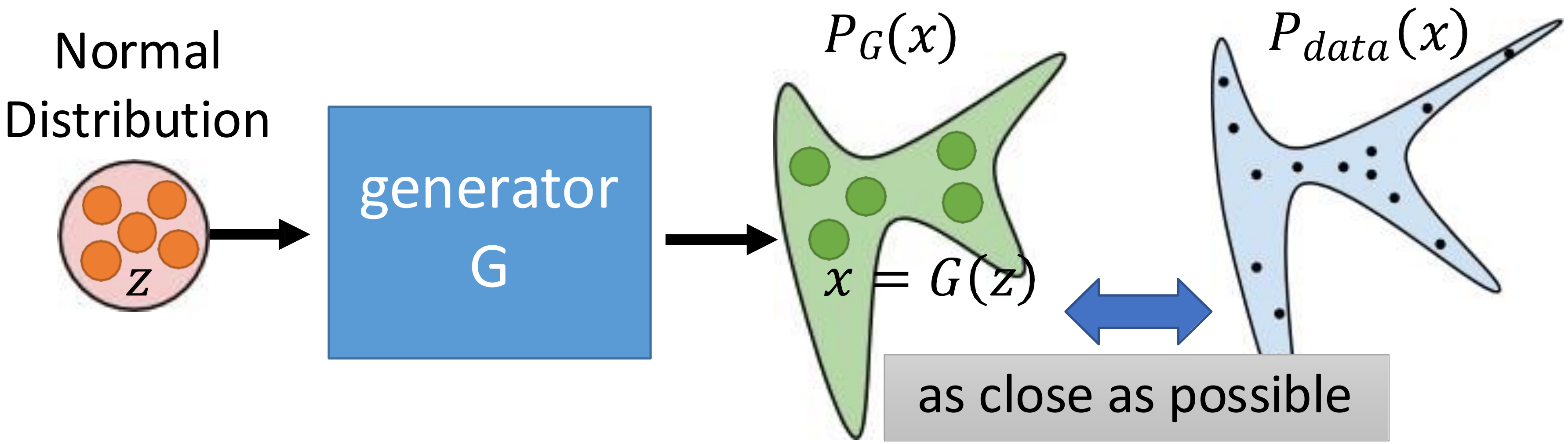
Can Discriminator generate?

A little bit theory

Generator

x: an image (a high-dimensional vector)

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



Normal Distribution

z

generator G

$P_G(x)$

$x = G(z)$

$P_{data}(x)$

as close as possible

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

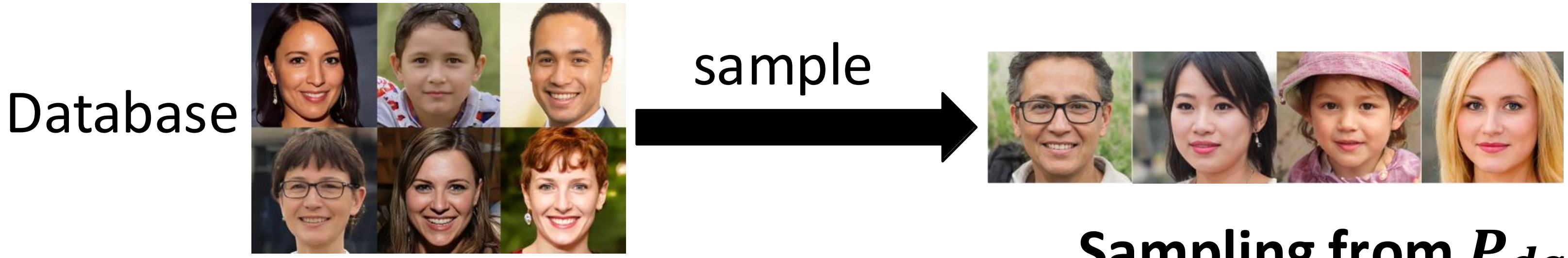
Divergence between distributions P_G and P_{data}

How to compute the divergence?

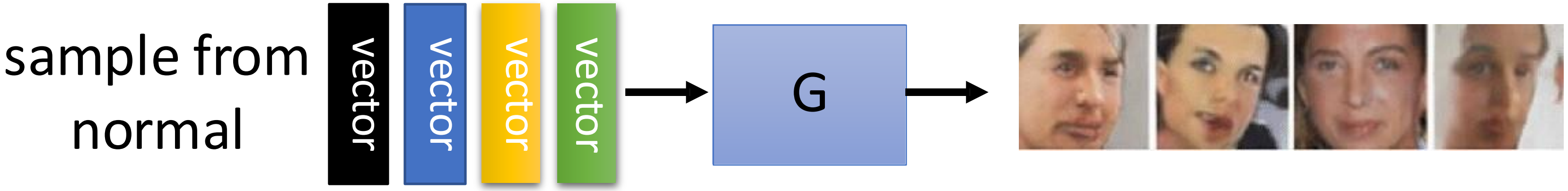
Discriminator

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

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



Sampling from P_{data}



Sampling from P_G

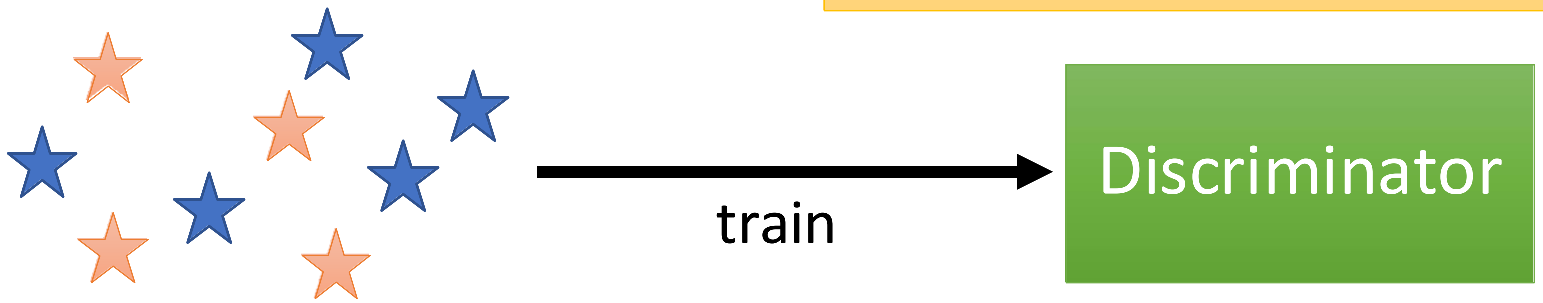
Discriminator

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

★ : data sampled from P_{data}

☆ : data sampled from P_G

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



Example Objective Function for D

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

(G is fixed)

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

The maximum objective value is related to JS divergence.

Discriminator

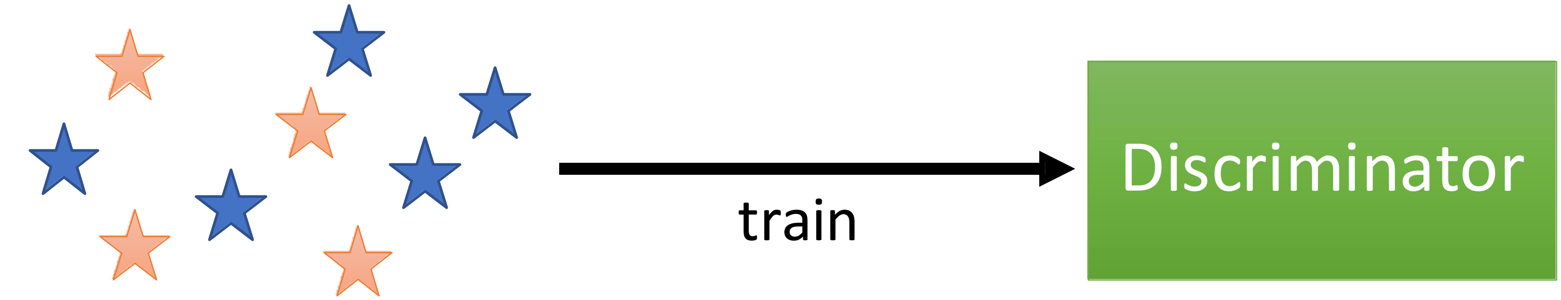
$$G^* = arg\min_G Div(P_G, P_{data})$$

★ : data sampled from P_{data}

★ : data sampled from P_G

Training:

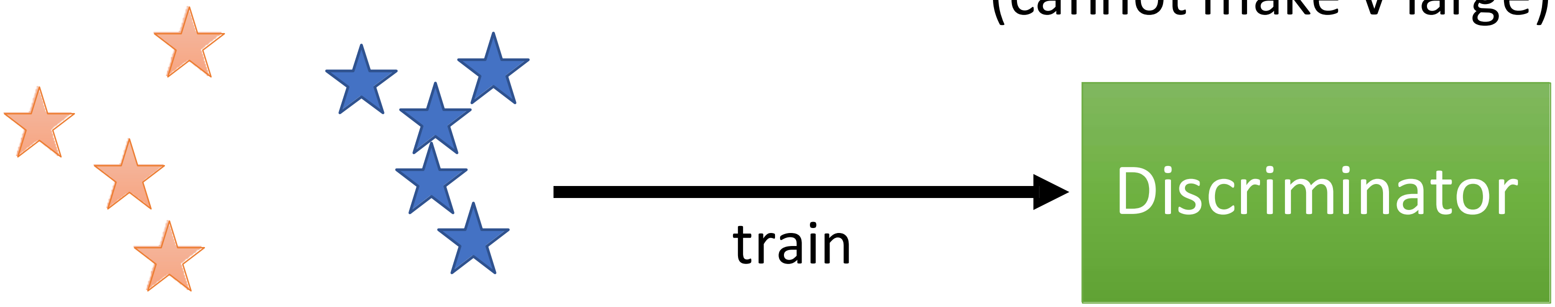
$$D^* = arg \boxed{\max_D V(D,G)}$$



small divergence

hard to discriminate

(cannot make V large)



large divergence

easy to discriminate

[Goodfellow, et al., NIPS, 2014]

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

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

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

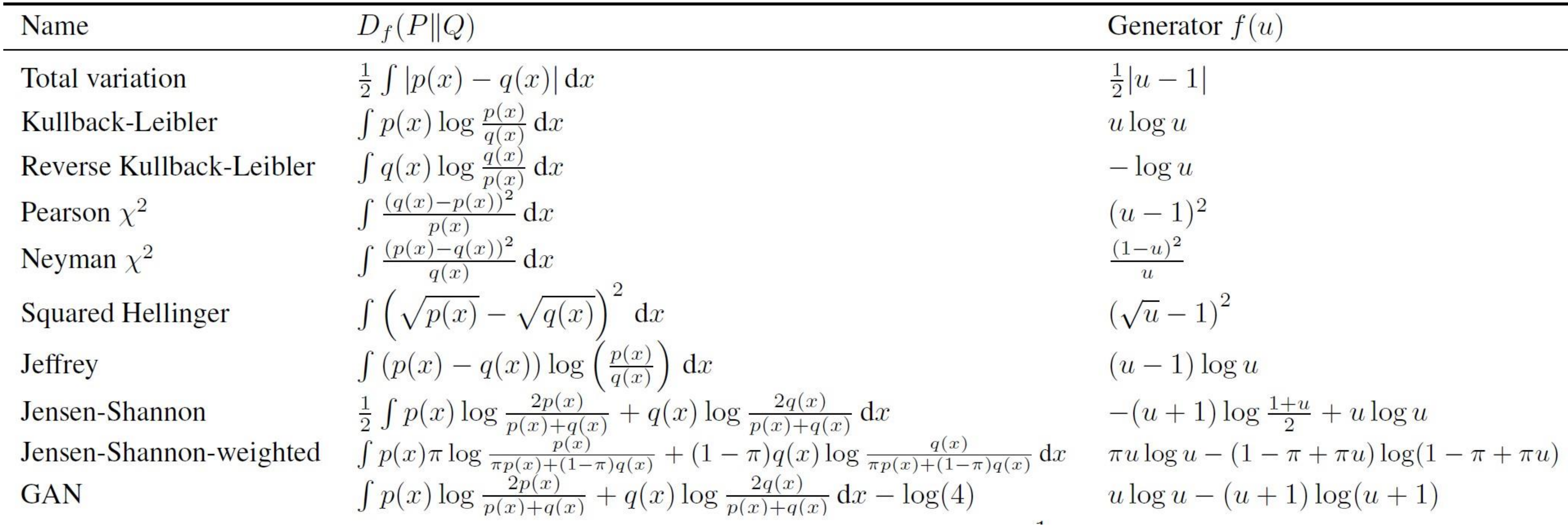
The maximum objective value is related to JS divergence.

- Initialize generator and discriminator
- In each training iteration:

  Step 1: Fix generator G, and update discriminator D

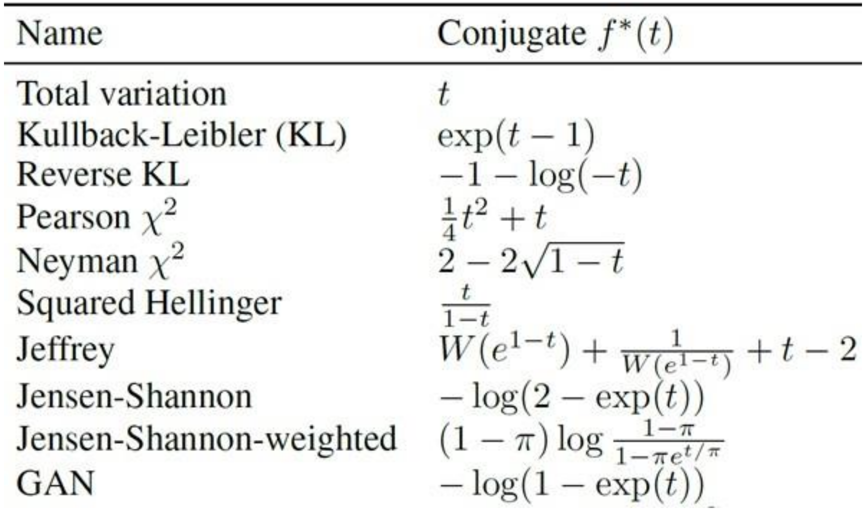
  Step 2: Fix discriminator D, and update generator G

Can we use other divergence?



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	$\int\left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2\,\mathrm{d}x$	$\left(\sqrt{u}-1\right)^2$
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	$\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x-\log(4)$	$u\log u-(u+1)\log(u+1)$

Using the divergence you like ☺



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	$-\log(1-\exp(t))$