Introduction of Generative Adversarial Network (GAN)

Yann LeCun's comment

What are some recent and potentially upcoming breakthroughs in unsupervised learning?

Yann LeCun, Director of AI Research at Facebook and Professor at NYU

Written Jul 29 · Upvoted by Joaquin Quiñonero Candela, Director Applied Machine Learning at Facebook and Huang Xiao

Adversarial training is the coolest thing since sliced bread.

I've listed a bunch of relevant papers in a previous answer.

Expect more impressive results with this technique in the coming years.

What's missing at the moment is a good understanding of it so we can make it work reliably. It's very finicky. Sort of like ConvNet were in the 1990s, when I had the reputation of being the only person who could make them work (which wasn't true).

https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-unsupervised-learning

Yann LeCun’s comment

What are some recent and potentially upcoming breakthroughs in deep learning?

Yann LeCun, Director of AI Research at Facebook and Professor at NYU

Written Jul 29 · Upvoted by Joaquin Quiñonero Candela, Director Applied Machine Learning at Facebook and Nikhil Garg, I lead a team of Quora engineers working on ML/NLP problems

……

The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal (he since moved to Google Brain and recently to OpenAI).

This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.

https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-deep-learning

Outline

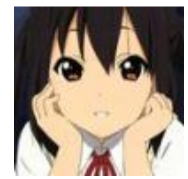
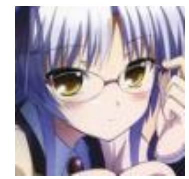
- Basic Idea of GAN
- GAN as structured learning
- Can Generator learn by itself?
- Can Discriminator generate?
- A little bit theory

Generation

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/

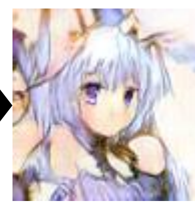
Basic Idea of GAN

It is a neural network (NN), or a function.

vector → Generator → image

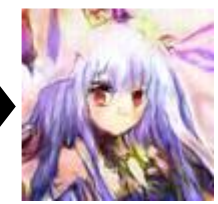
high dimensional vector

$$\begin{bmatrix} 0.1 \\ -3 \\ \vdots \\ 2.4 \\ 0.9 \end{bmatrix} \rightarrow \text{Generator} \rightarrow$$



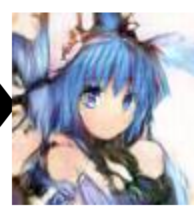
Each dimension of input vector represents some characteristics.

$$\begin{bmatrix} \boxed{3} \\ -3 \\ \vdots \\ 2.4 \\ 0.9 \end{bmatrix} \rightarrow \text{Generator} \rightarrow$$



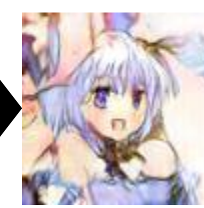
Longer hair

$$\begin{bmatrix} 0.1 \\ 2.1 \\ \vdots \\ \boxed{5.4} \\ 0.9 \end{bmatrix} \rightarrow \text{Generator} \rightarrow$$



blue hair

$$\begin{bmatrix} 0.1 \\ -3 \\ \vdots \\ 2.4 \\ \boxed{3.5} \end{bmatrix} \rightarrow \text{Generator} \rightarrow$$



Open mouth

Basic Idea of GAN

It is a neural network (NN), or a function.

image → Discri-minator → scalar

Larger value means real, smaller value means fake.

 → Discri-minator → 1.0

 → Discri-minator → 1.0

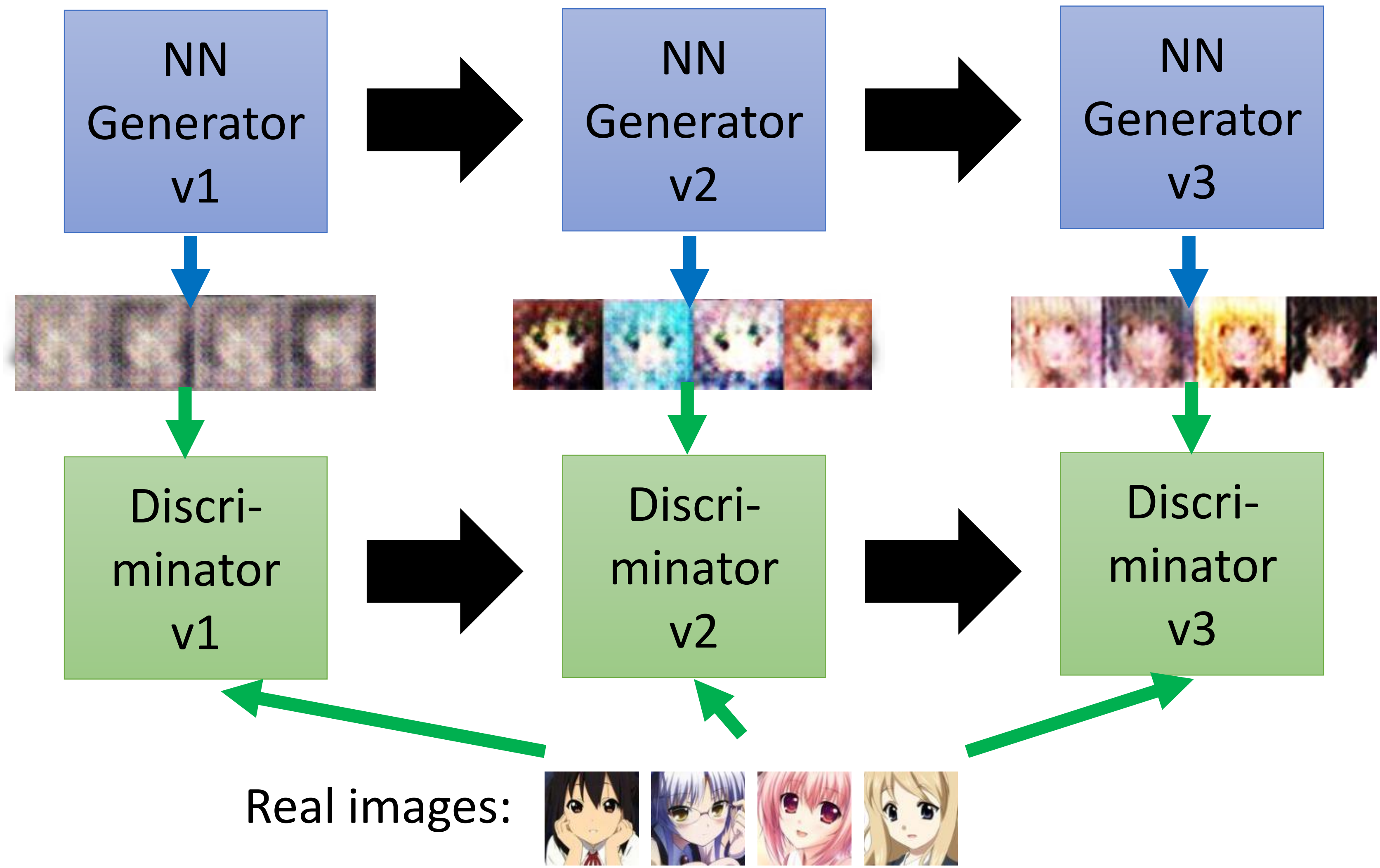
 → Discri-minator → 0.1

 → Discri-minator → 0.1

Basic Idea of GAN

This is where the term "***adversarial***" comes from.

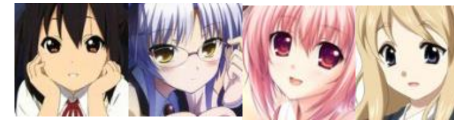
You can explain the process in different ways…….



Basic Idea of GAN

Generator (student)

Discriminator (teacher)

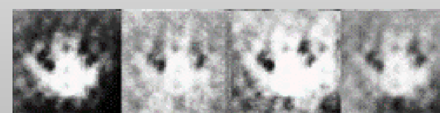


Generator v1

Discriminator v1

没有两个圈

Generator v2



Discriminator v2

没有颜色

Generator v3

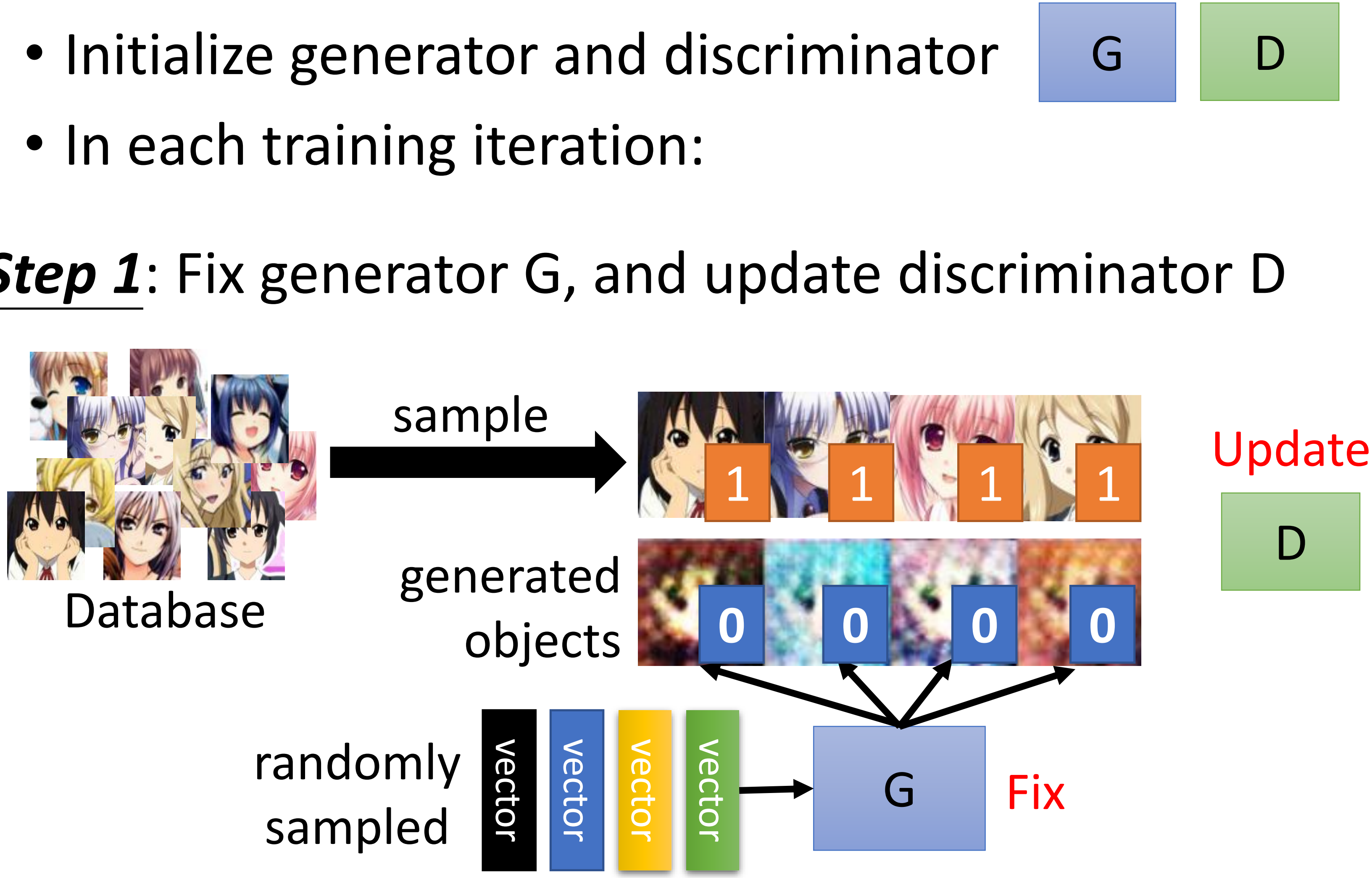
为什么不自己学？

为什么不自己做？

Algorithm

- Initialize generator and discriminator
- In each training iteration:

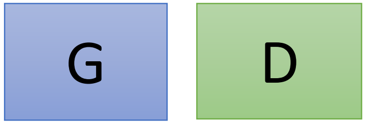
Step 1: Fix generator G, and update discriminator D



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

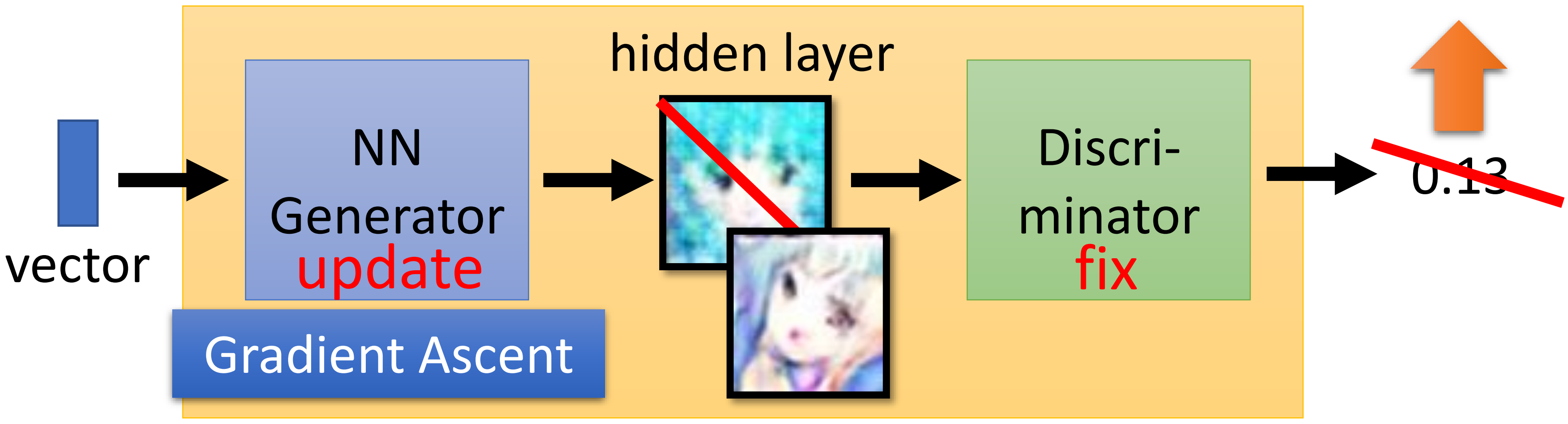
Algorithm

- Initialize generator and discriminator
- In each training iteration:



Step 2: Fix discriminator D, and update generator G

Generator learns to "fool" the discriminator



large network

Algorithm Initialize θ_d for D and θ_g for G

- In each training iteration:

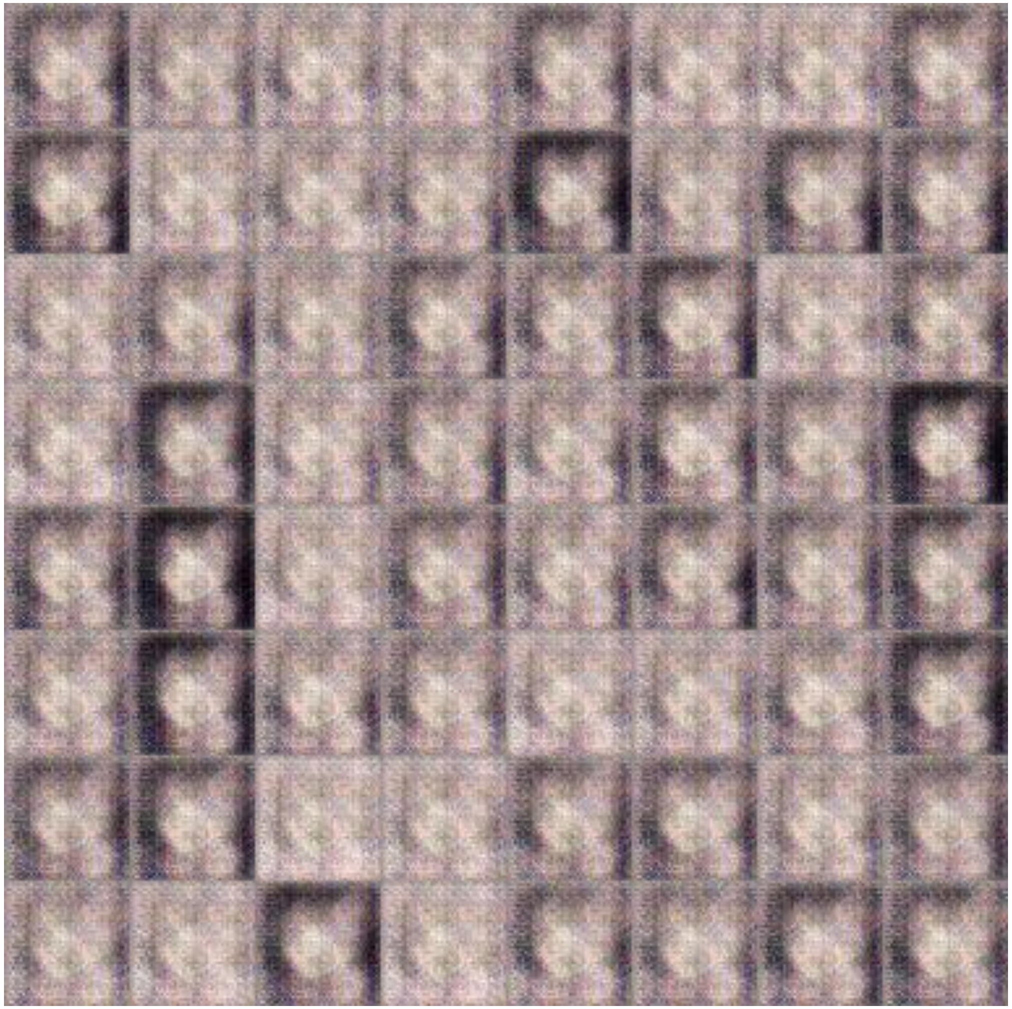
Learning D

- 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 $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$, $\tilde{x}^i = G(z^i)$
- Update discriminator parameters θ_d to maximize
  - $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m} logD(x^i) + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D\left(\tilde{x}^i\right)\right)$
  - $\theta_d \leftarrow \theta_d + \eta\nabla\tilde{V}(\theta_d)$

Learning G

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

Anime Face Generation



100 updates

Source of training data: https://zhuanlan.zhihu.com/p/24767059

Anime Face Generation

1000 updates

Anime Face Generation

2000 updates

Anime Face Generation

5000 updates

Anime Face Generation

10,000 updates

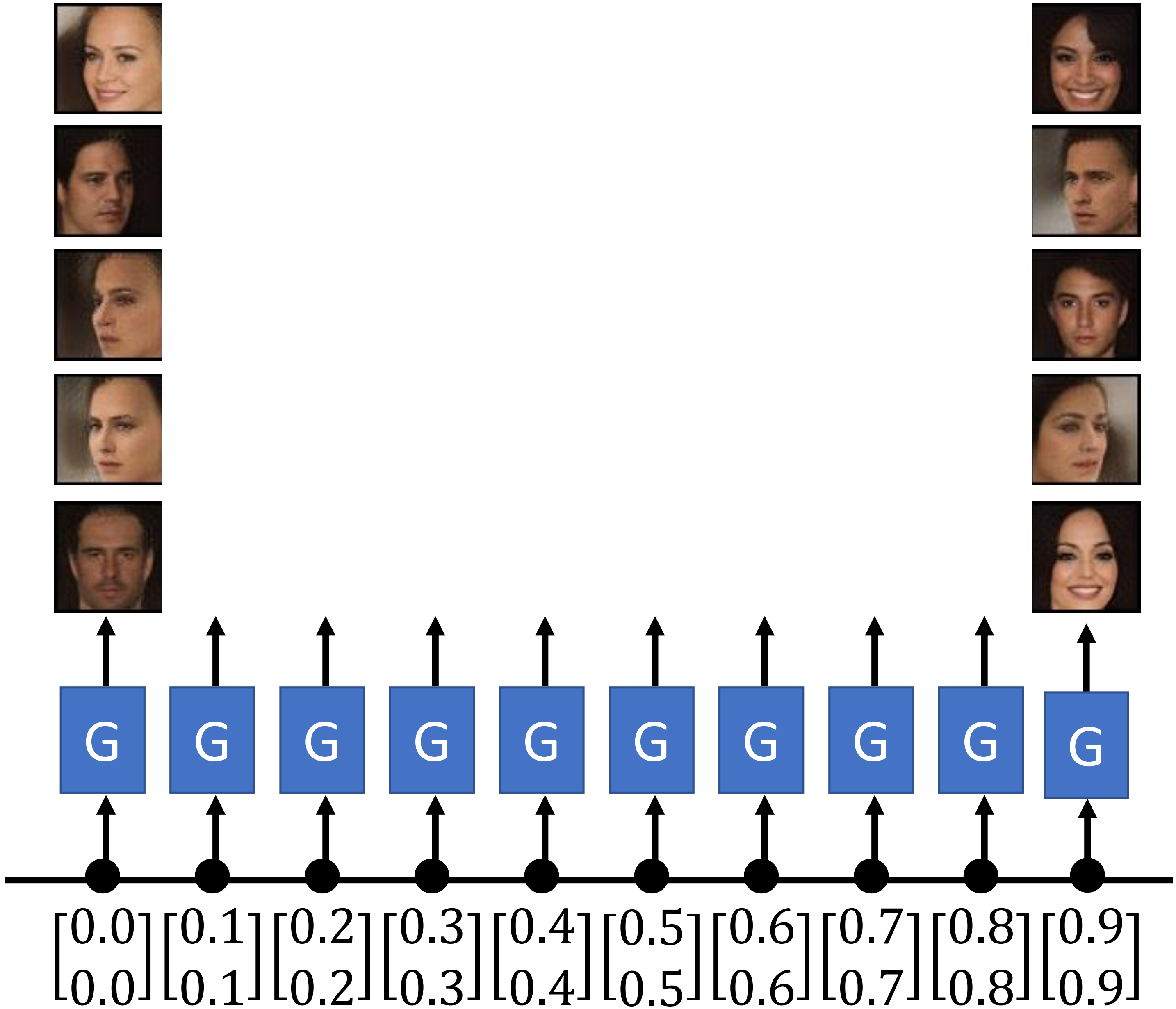
Anime Face Generation

20,000 updates

Anime Face Generation

50,000 updates

The faces generated by machine.


G
[0.0, 0.0]
[0.1, 0.1]
[0.2, 0.2]
[0.3, 0.3]
[0.4, 0.4]
[0.5, 0.5]
[0.6, 0.6]
[0.7, 0.7]
[0.8, 0.8]
[0.9, 0.9]

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 $f: X \rightarrow Y$

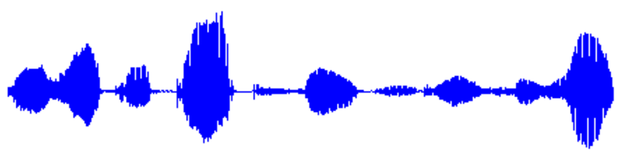
Machine Translation

X : “机器学习及其深层与结构化”
(sentence of language 1)

Y : “Machine learning and having it deep and structured”
(sentence of language 2)

Speech Recognition

X :



(speech)

Y : “感谢大家来上课”
(transcription)

Chat-bot

X : “How are you?”
(what a user says)

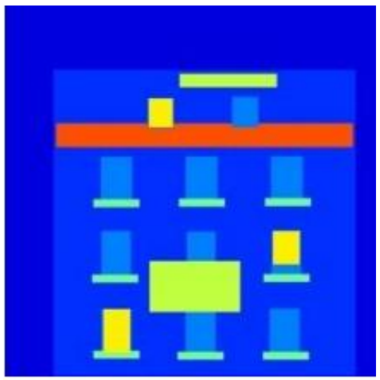
Y : “I’m fine.”
(response of machine)

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" ……
    - Since the output space is huge, most "classes" do not have any training data.
    - Machine has to create new stuff during testing.
    - Need more intelligence

Structured Learning Approach

<u>Generator</u>

Learn to generate the object at the component level

Bottom Up

+

<u>Discriminator</u>

Evaluating the whole object, and find the best one

Top Down

Outline

Basic Idea of GAN

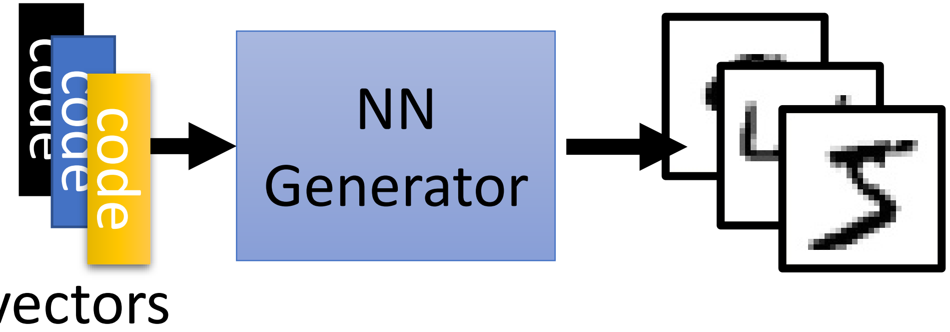
GAN as structured learning

Can Generator learn by itself?

Can Discriminator generate?

A little bit theory

Generator

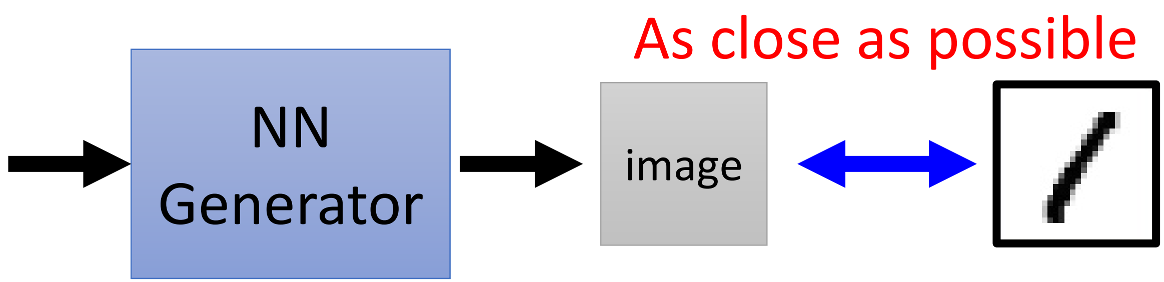


code: $\begin{bmatrix}0.1\\-0.5\end{bmatrix}$ $\begin{bmatrix}0.1\\0.9\end{bmatrix}$ $\begin{bmatrix}0.2\\-0.1\end{bmatrix}$ $\begin{bmatrix}0.3\\0.2\end{bmatrix}$

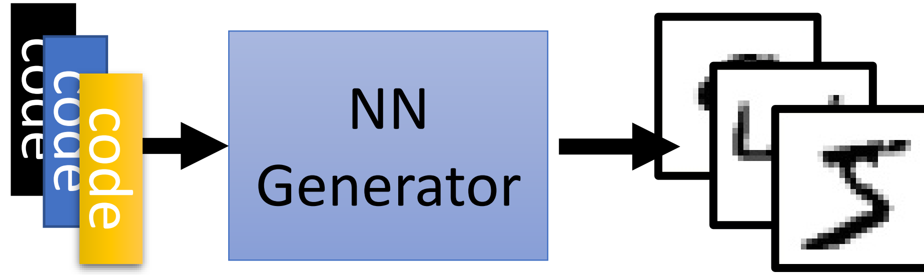
(where does they come from?)

Image:

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



Generator



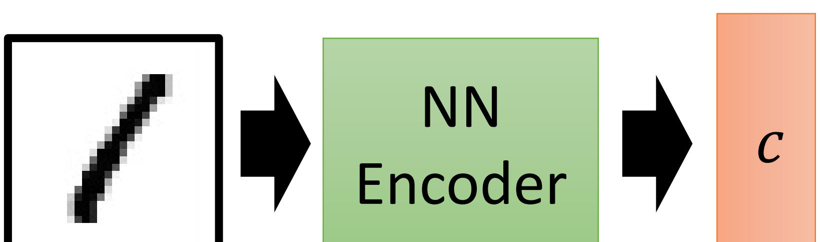
vectors

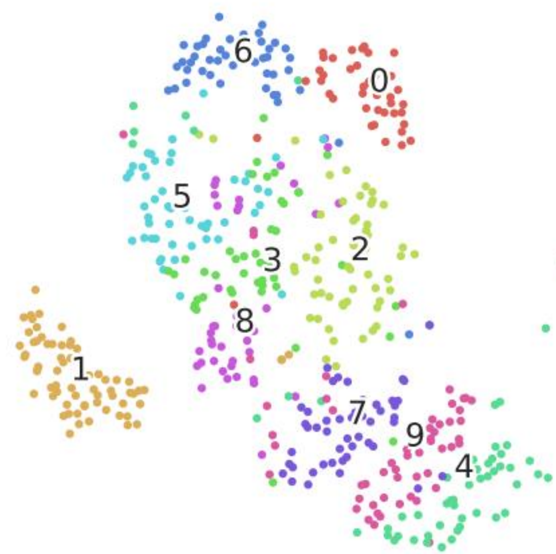
code: $\begin{bmatrix} 0.1 \\ -0.5 \end{bmatrix}$ $\begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$ $\begin{bmatrix} 0.2 \\ -0.1 \end{bmatrix}$ $\begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix}$

(where does they come from?)

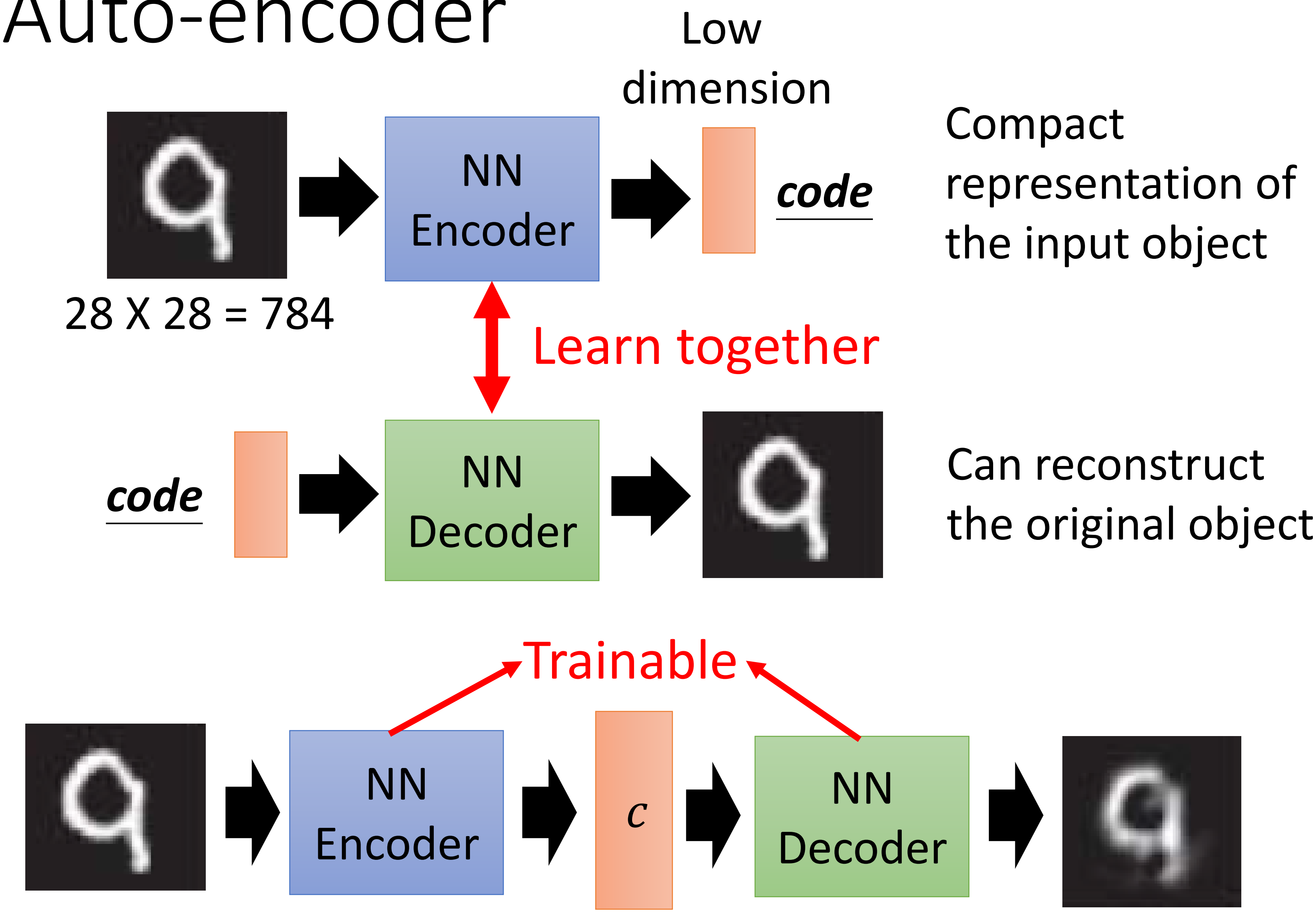
Image:

Encoder in auto-encoder provides the code ☺

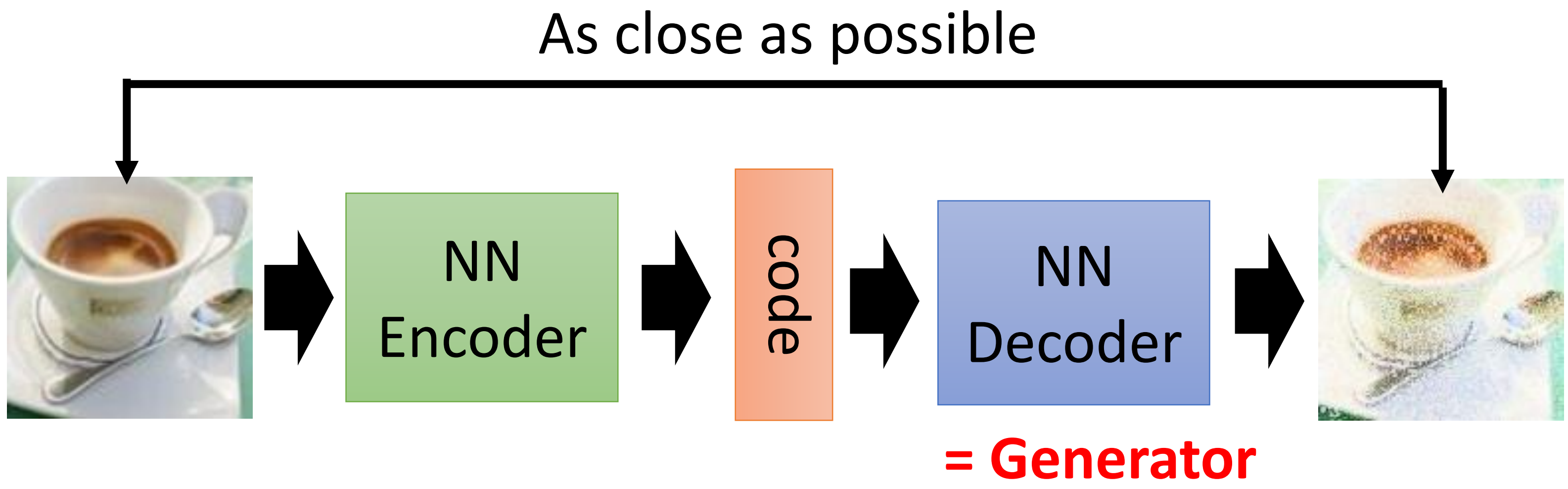


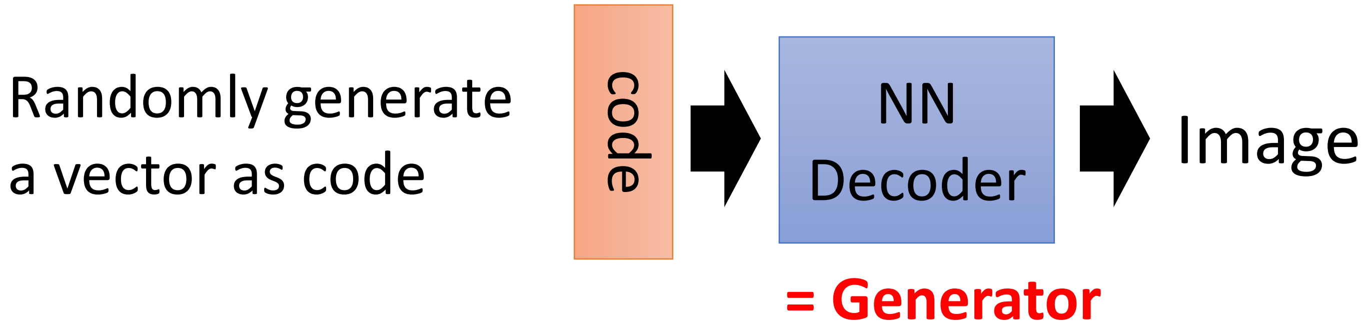


Auto-encoder

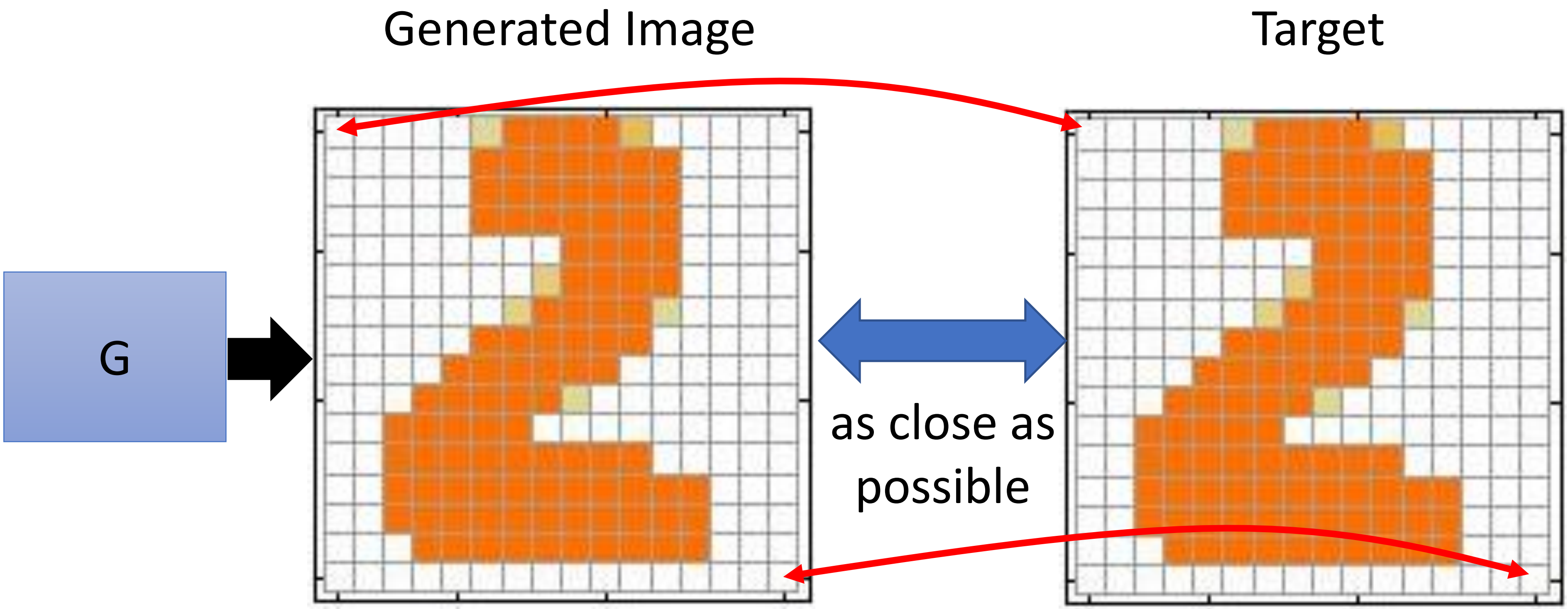


Auto-encoder





What do we miss?



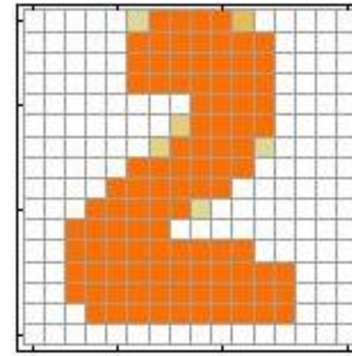
It will be fine if the generator can truly copy the target image.

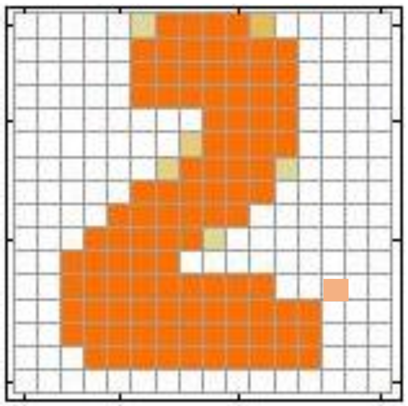
What if the generator makes some mistakes …….

Some mistakes are serious, while some are fine.

What do we miss?

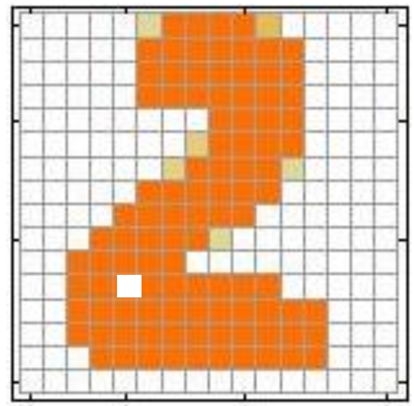
Target





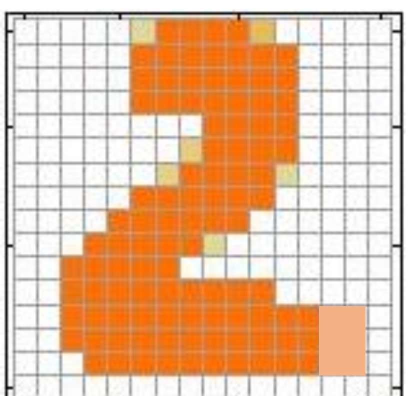
1 pixel error

我觉得不行



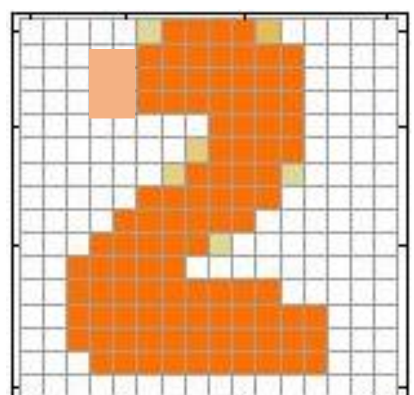
1 pixel error

我觉得不行



6 pixel errors

我觉得可以

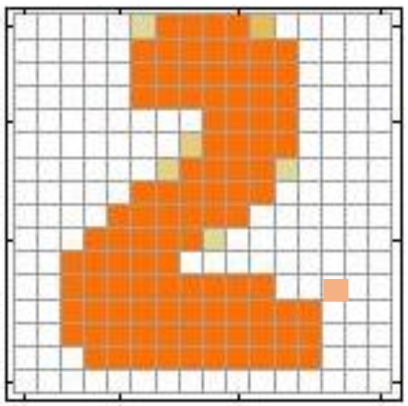


6 pixel errors

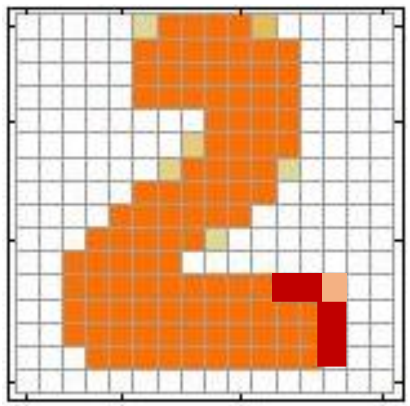
我觉得可以

What do we miss?

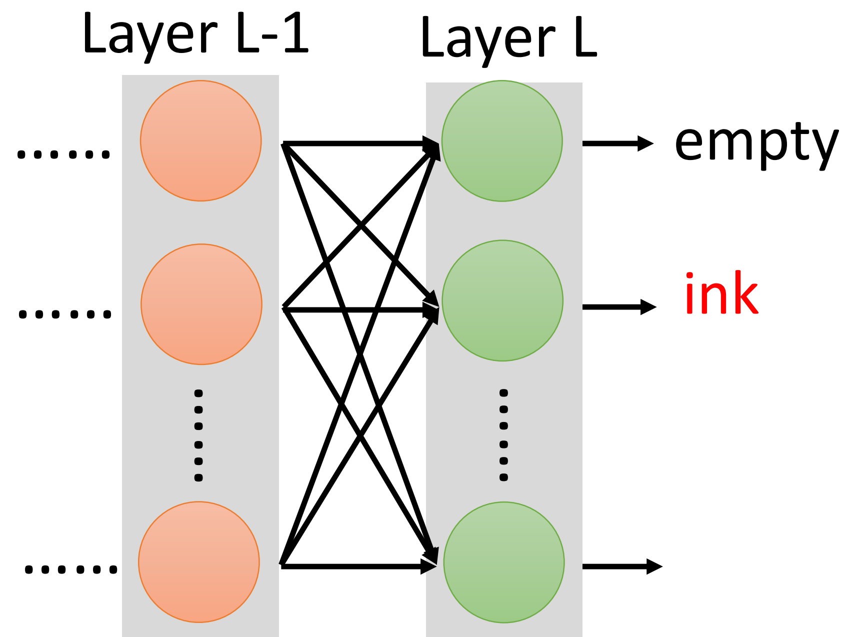
Each neural in output layer corresponds to a pixel.



我觉得不行



我觉得可以



The relation between the components are critical.

Although highly correlated, they cannot influence each other.

Need deep structure to catch the relation between components.

Outline

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

Discriminator

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

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

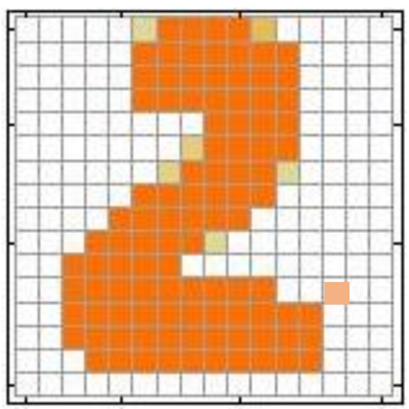
  - Input x: an object x (e.g. an image)
  - Output D(x): scalar which represents how "good" an object x is

Can we use the discriminator to generate objects?

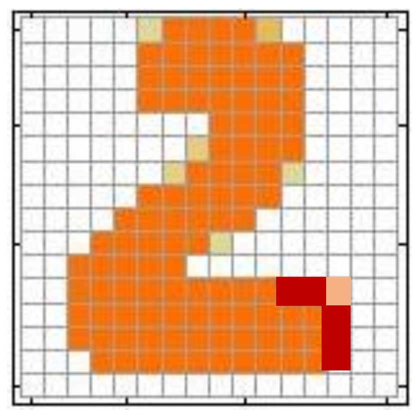
Yes.

Discriminator

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



我觉得不行



我觉得可以

Discriminator

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

Inference

- Generate object $\tilde{x}$ that

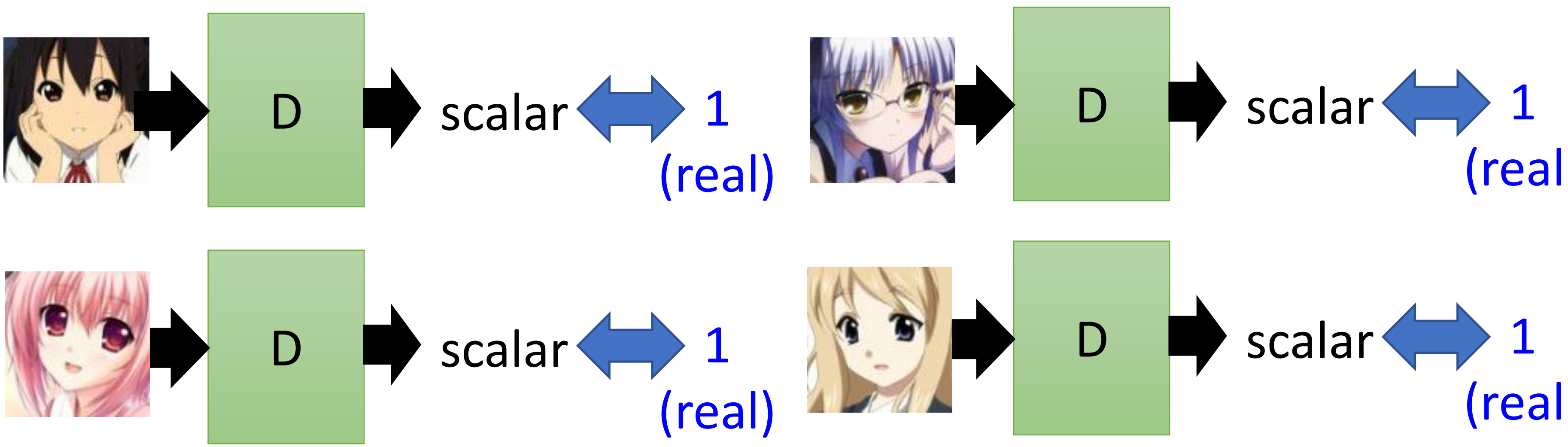
$$\tilde{x} = \arg\max_{x \in X} D(x)$$

Enumerate all possible x !!!

How to learn the discriminator?

Discriminator - Training

- I have some real images

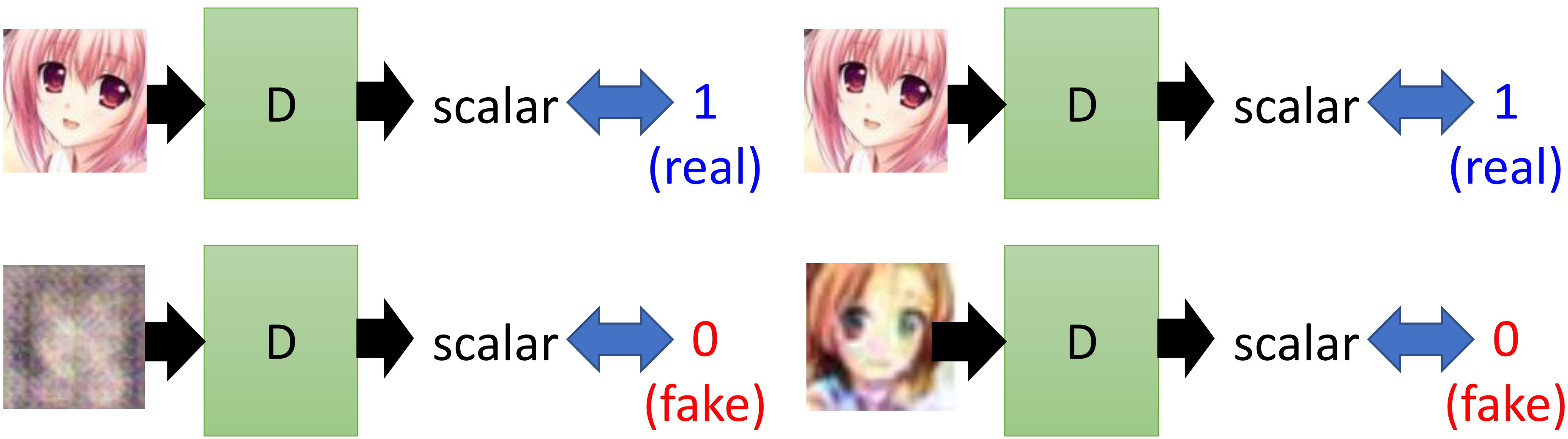


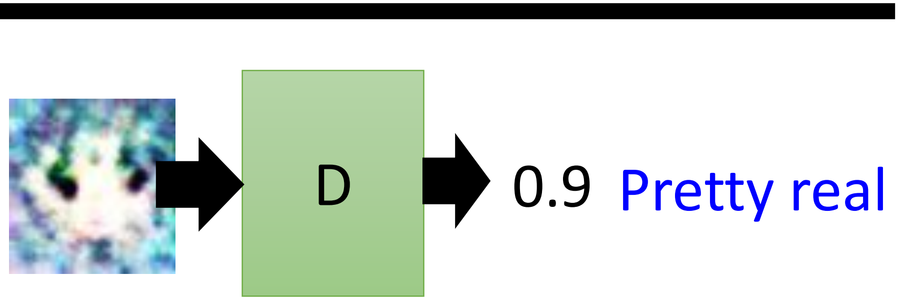
Discriminator only learns to output “1” (real).

Discriminator training needs some negative examples.

Discriminator - Training

- Negative examples are critical.



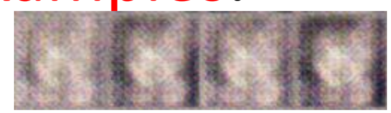


How to generate realistic negative examples?

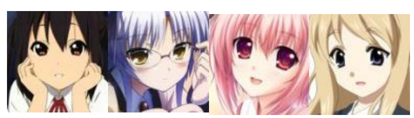
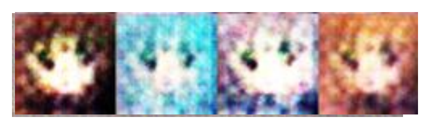
Discriminator - Training

- **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.

 v.s.  → D

    - Generate negative examples by discriminator D

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

Generator v.s. Discriminator

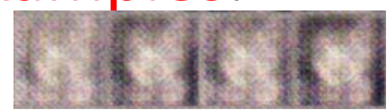
- ***Generator***
- Pros:
  - Easy to generate even with deep model
- Cons:
  - Imitate the appearance
  - Hard to learn the correlation between components

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

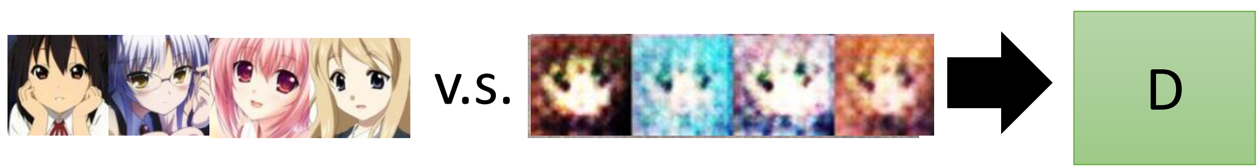
Generator + Discriminator

- **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

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

Benefit of GAN

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

$$\boxed{G \longrightarrow \tilde{x}} \quad = \quad \boxed{\tilde{x} = \arg\max_{x \in X} D(x)}$$

efficient

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

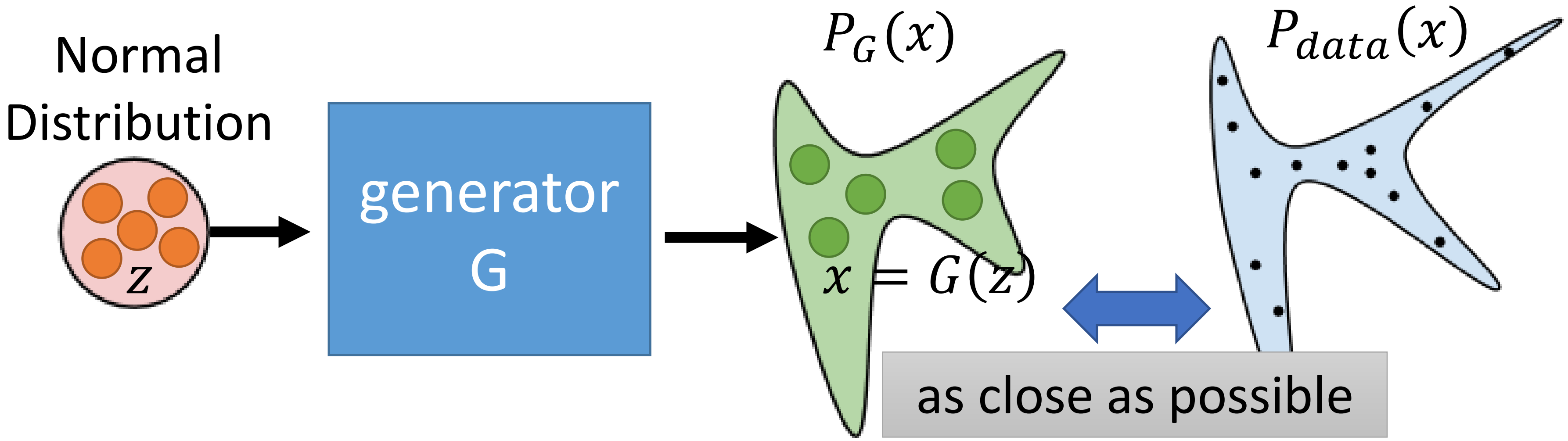
Outline

- 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



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

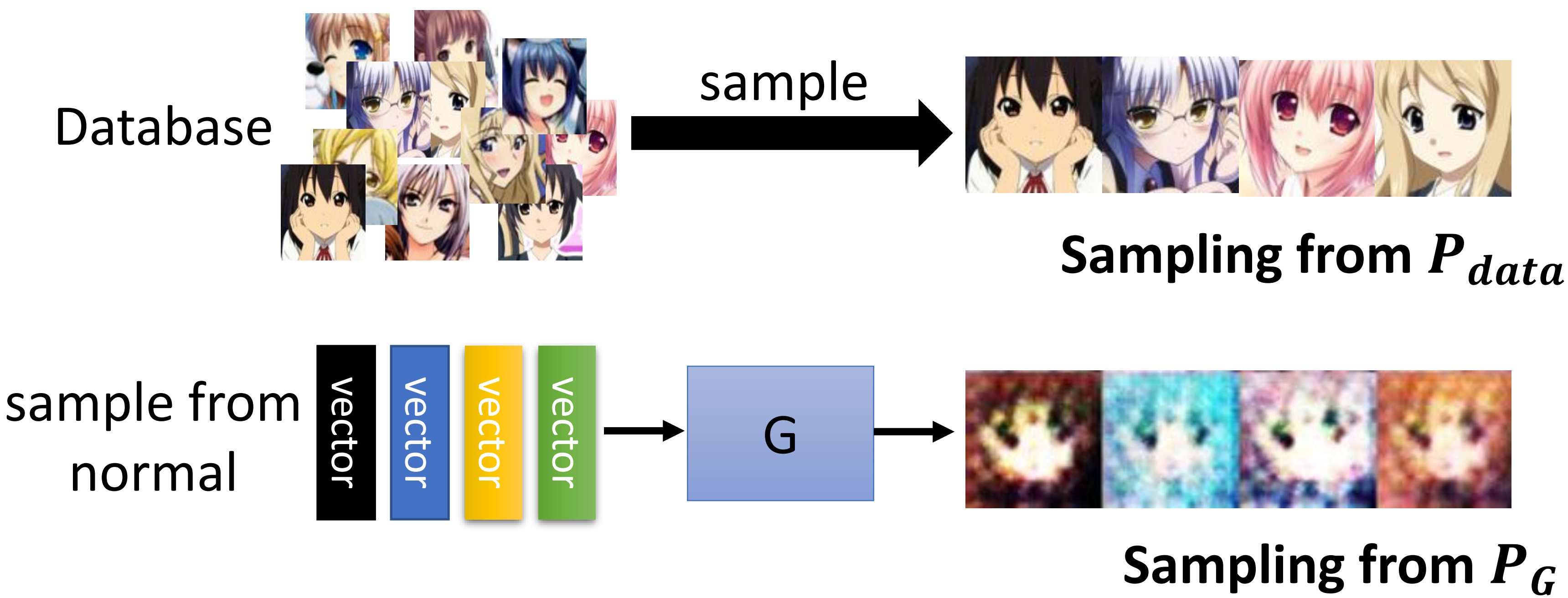
Divergence between distributions P_G and P_{data}

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.



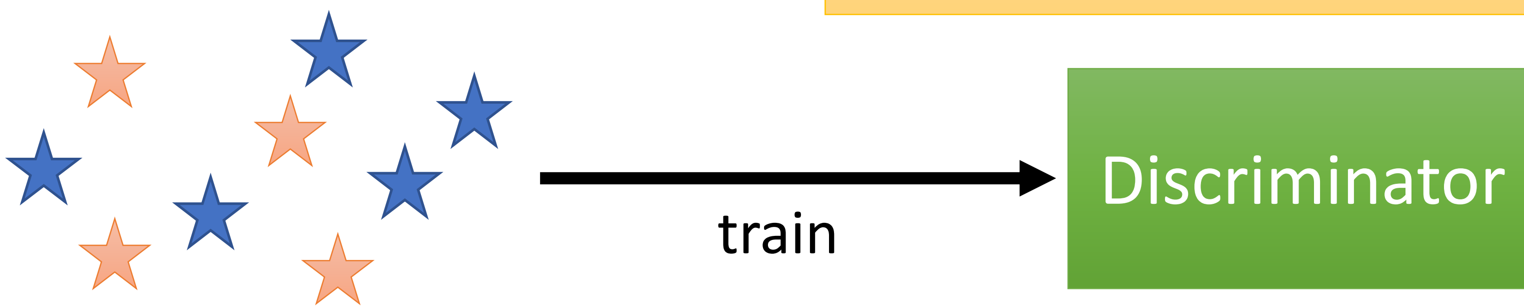
Discriminator

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

★ : 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

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

(G is fixed)

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

Discriminator

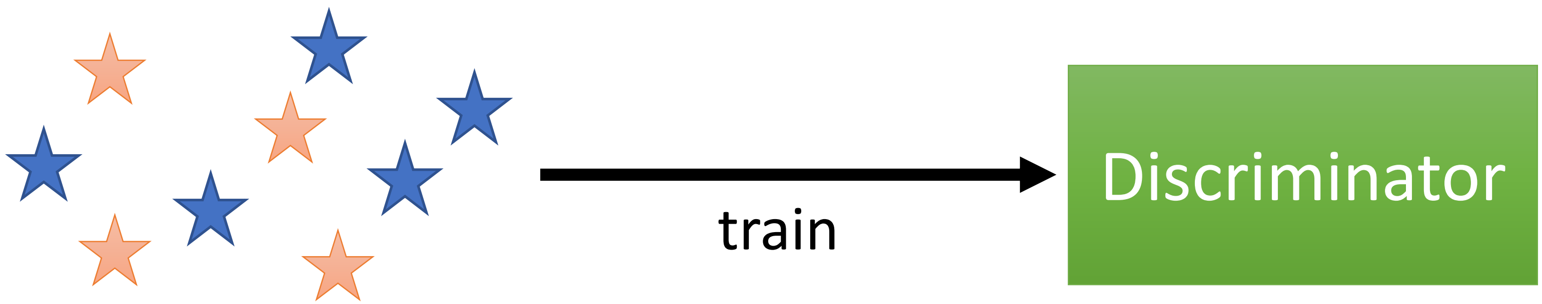
$$G^* = arg\min_G Div(P_G, P_{data})$$

: data sampled from P_{data}

: data sampled from P_G

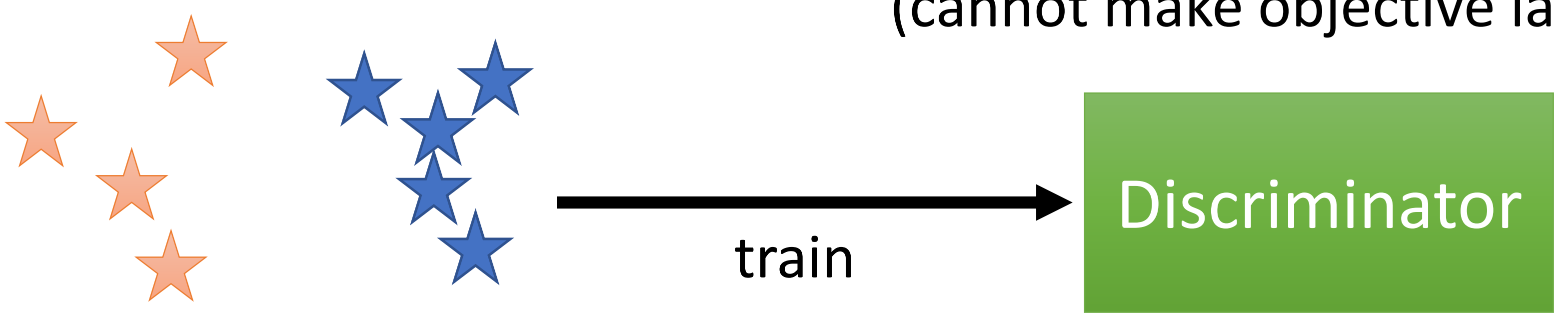
Training:

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



small divergence

hard to discriminate
(cannot make objective large)



large divergence

easy to discriminate

[Goodfellow, et al., NIPS, 2014]

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

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

- 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

The End!