人工智能技术及应用
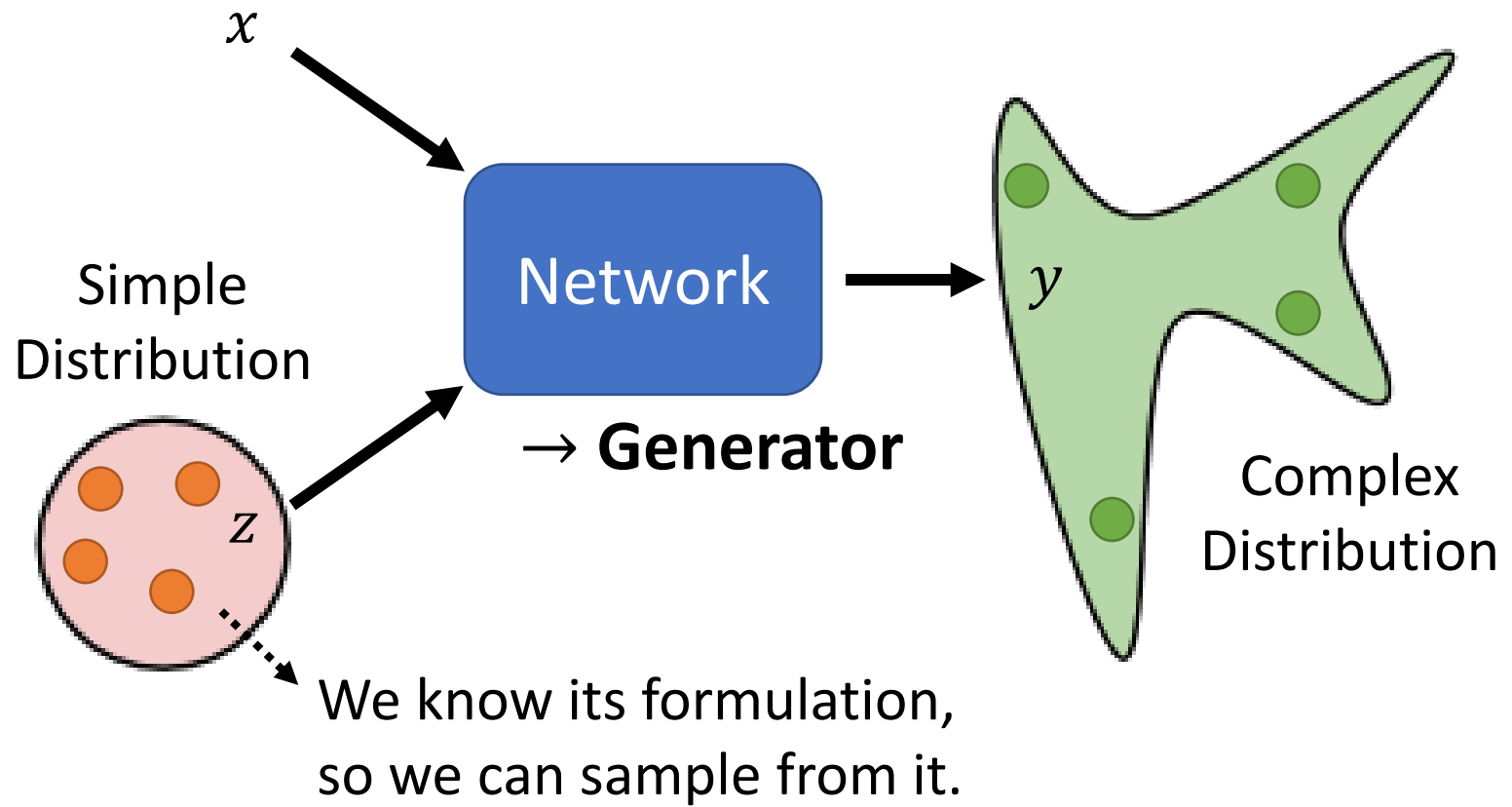Artificial Intelligence and Application
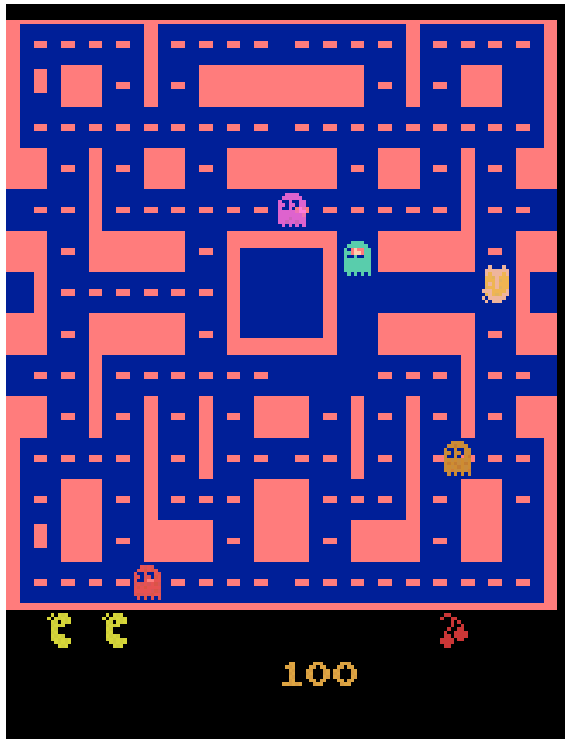
Generation

Network as Generator
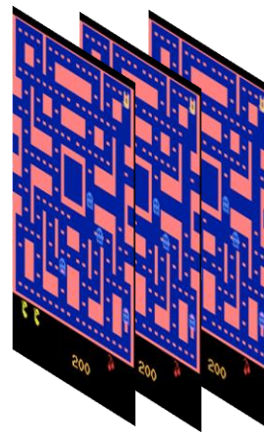
x

Network

→ **Generator**

y

Simple
Distribution

z

We know its formulation,
so we can sample from it.

Complex
Distribution

Why distribution?

Video Prediction
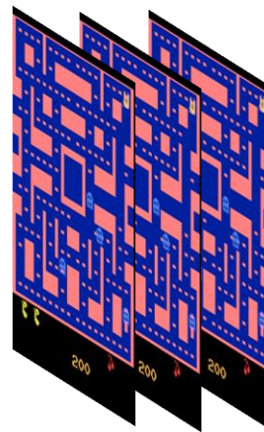
Real Video

Previous frames

Network → y

next frame

Why distribution?

Prediction

Video Prediction

Previous frames

Network → y **???**
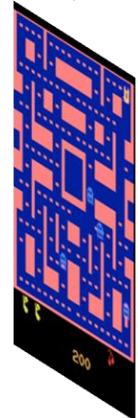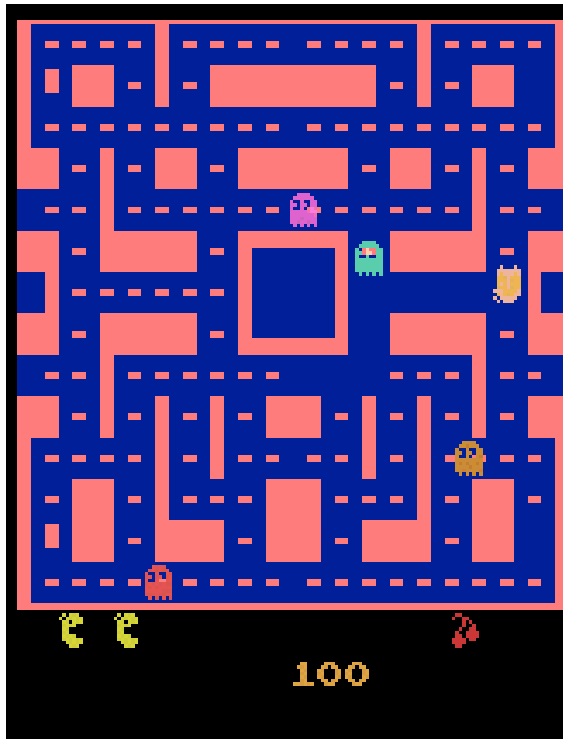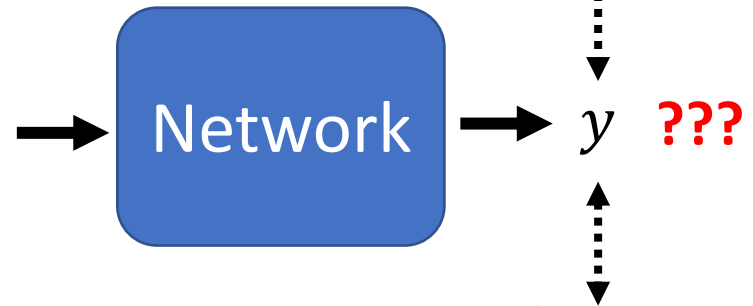
turn right
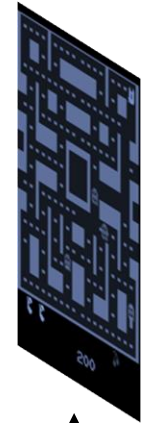
turn left

Why distribution?

Prediction

Video Prediction

Previous frames
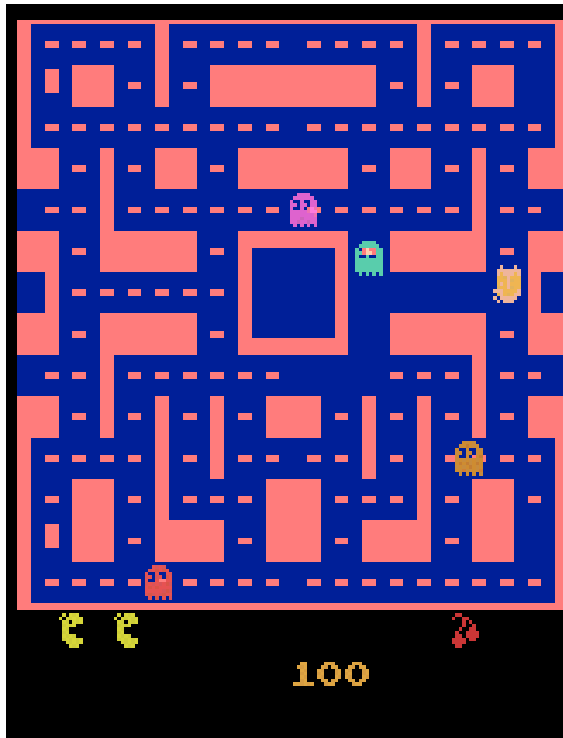
Network

z
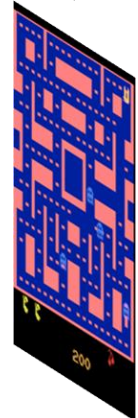
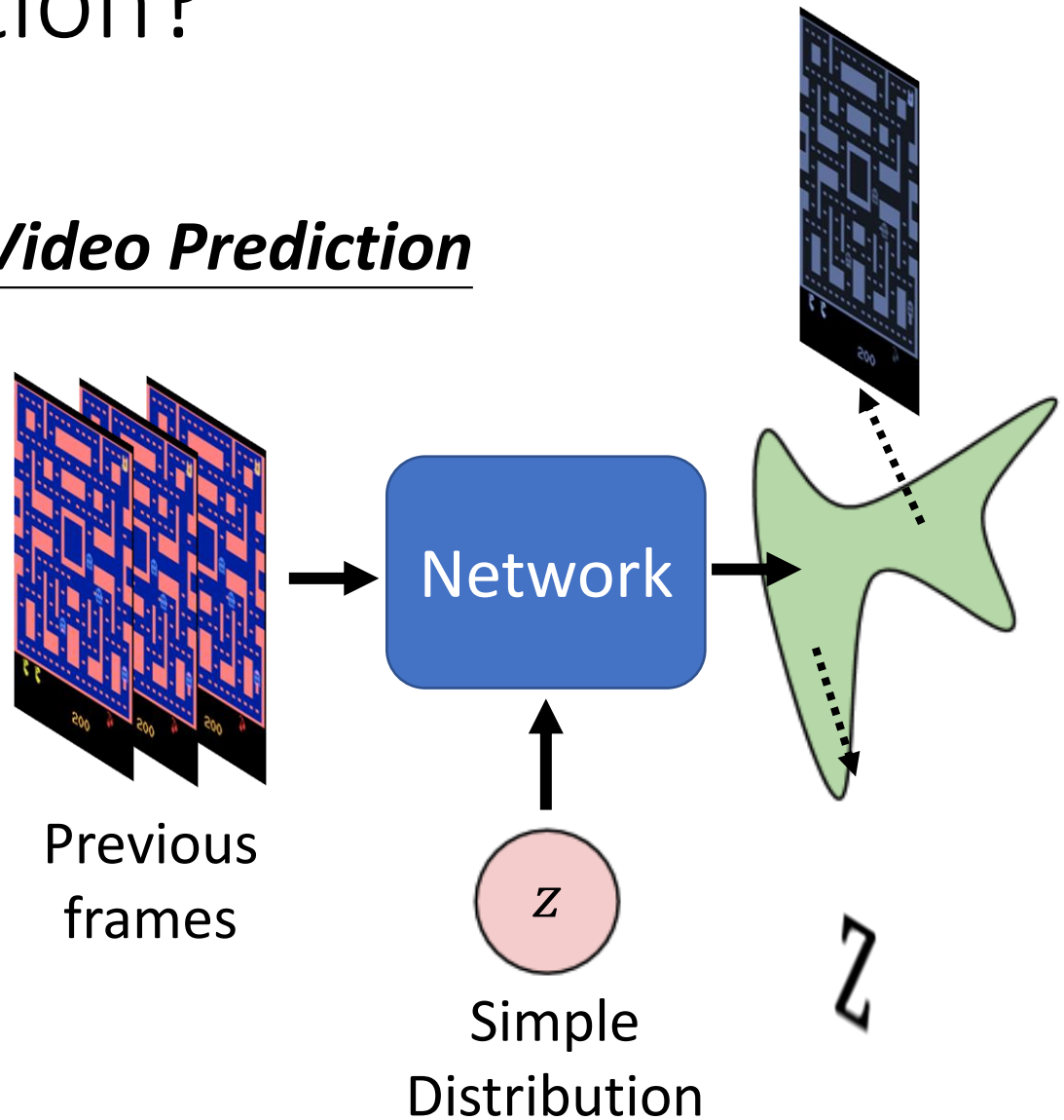Simple Distribution

Why distribution?

(The same input has different outputs.)

- Especially for the tasks needs "*creativity*"

Drawing

Character
with red eyes → Network →

Chatbot

你知道鲁班吗？ → Network → 古代发明家

团战可以输，鲁班必须死

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).

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

All Kinds of GAN ...

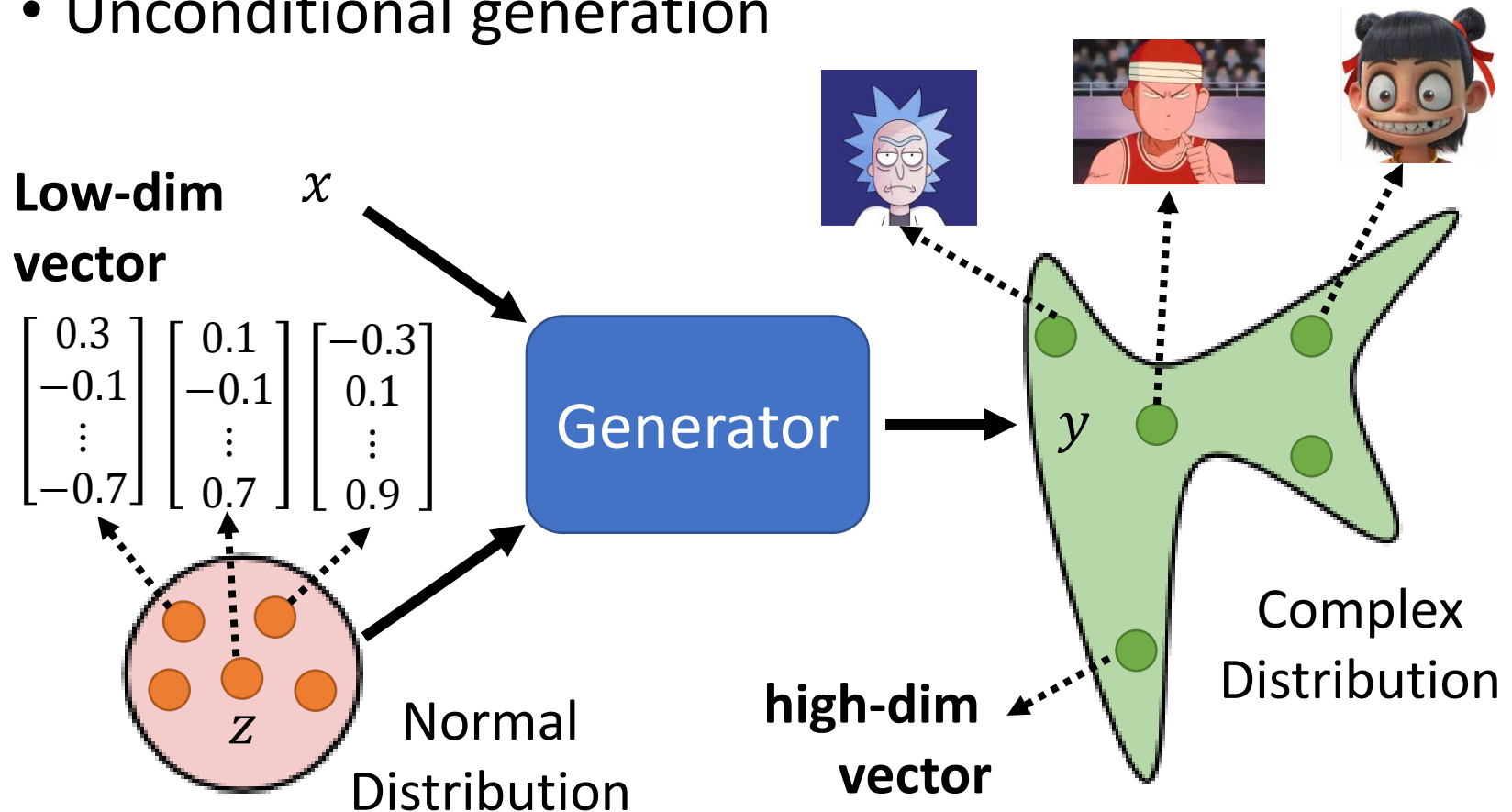https://github.com/hindupuravinash/the-gan-zoo

GAN

ACGAN

BGAN

CGAN

DCGAN

EBGAN

fGAN

GoGAN

⋮

- SeUDA - Semantic-Aware Generative Adversarial Nets for Unsupervised Domain Adapt Segmentation
- SG-GAN - Semantic-aware Grad-GAN for Virtual-to-Real Urban Scene Adaption (githu
- SG-GAN - Sparsely Grouped Multi-task Generative Adversarial Networks for Facial Attr
- SGAN - Texture Synthesis with Spatial Generative Adversarial Networks
- SGAN - Stacked Generative Adversarial Networks (github)
- SGAN - Steganographic Generative Adversarial Networks
- SGAN - SGAN: An Alternative Training of Generative Adversarial Networks
- SGAN - CT Image Enhancement Using Stacked Generative Adversarial Networks and T Segmentation Improvement
- sGAN - Generative Adversarial Training for MRA Image Synthesis Using Multi-Contrast
- SiftingGAN - SiftingGAN: Generating and Sifting Labeled Samples to Improve the Rem Classification Baseline in vitro
- SiGAN - SiGAN: Siamese Generative Adversarial Network for Identity-Preserving Face
- SimGAN - Learning from Simulated and Unsupervised Images through Adversarial Trai
- SisGAN - Semantic Image Synthesis via Adversarial Learning

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

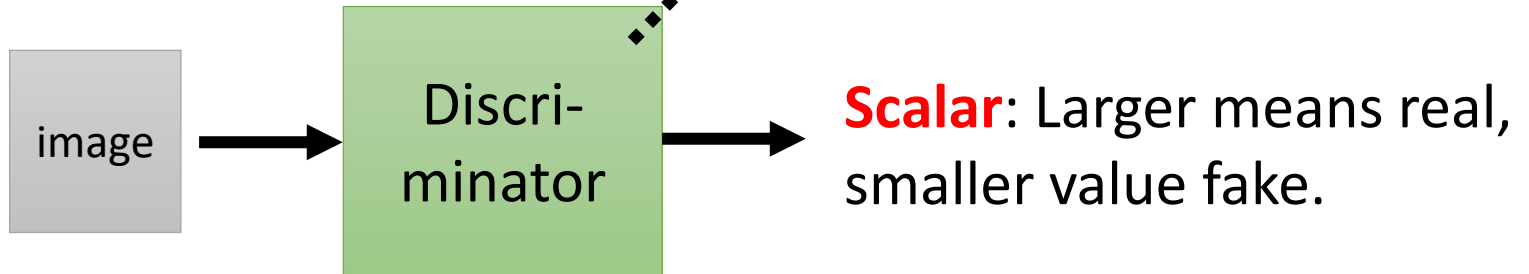[2] We use the Greek α prefix for α-GAN, as AEGAN and most other Latin prefixes seem to have been taken https://deephunt.in/the-gan-zoo-79597dc8c347.

Anime Face Generation

• Unconditional generation

Low-dim vector

x

$$\begin{bmatrix} 0.3 \\ -0.1 \\ \vdots \\ -0.7 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.1 \\ \vdots \\ 0.7 \end{bmatrix} \begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$

Generator

z

Normal Distribution

y

high-dim vector

Complex Distribution

Discriminator

It is a neural network (that is, a function).

image → **Discri-minator** → **Scalar**: Larger means real, smaller value fake.

 → **Discri-minator** → 1.0

 → **Discri-minator** → 1.0

 → **Discri-minator** → 0.1

 → **Discri-minator** → 0.1

Basic Idea of GAN

Generator

Brown

veins

Butterflies are not brown

Butterflies do not have veins

....
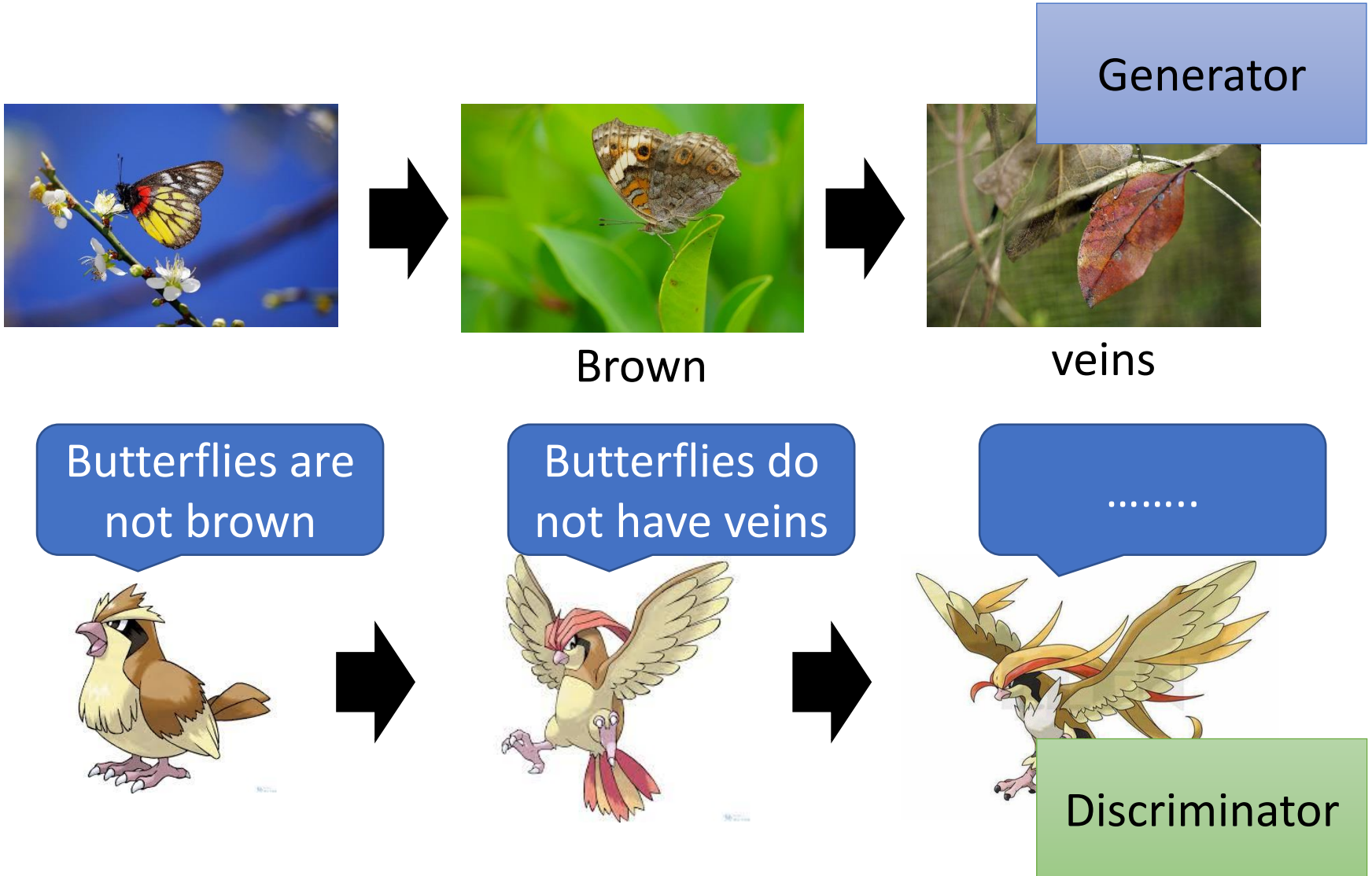
Discriminator

Basic Idea of GAN

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

Real images:
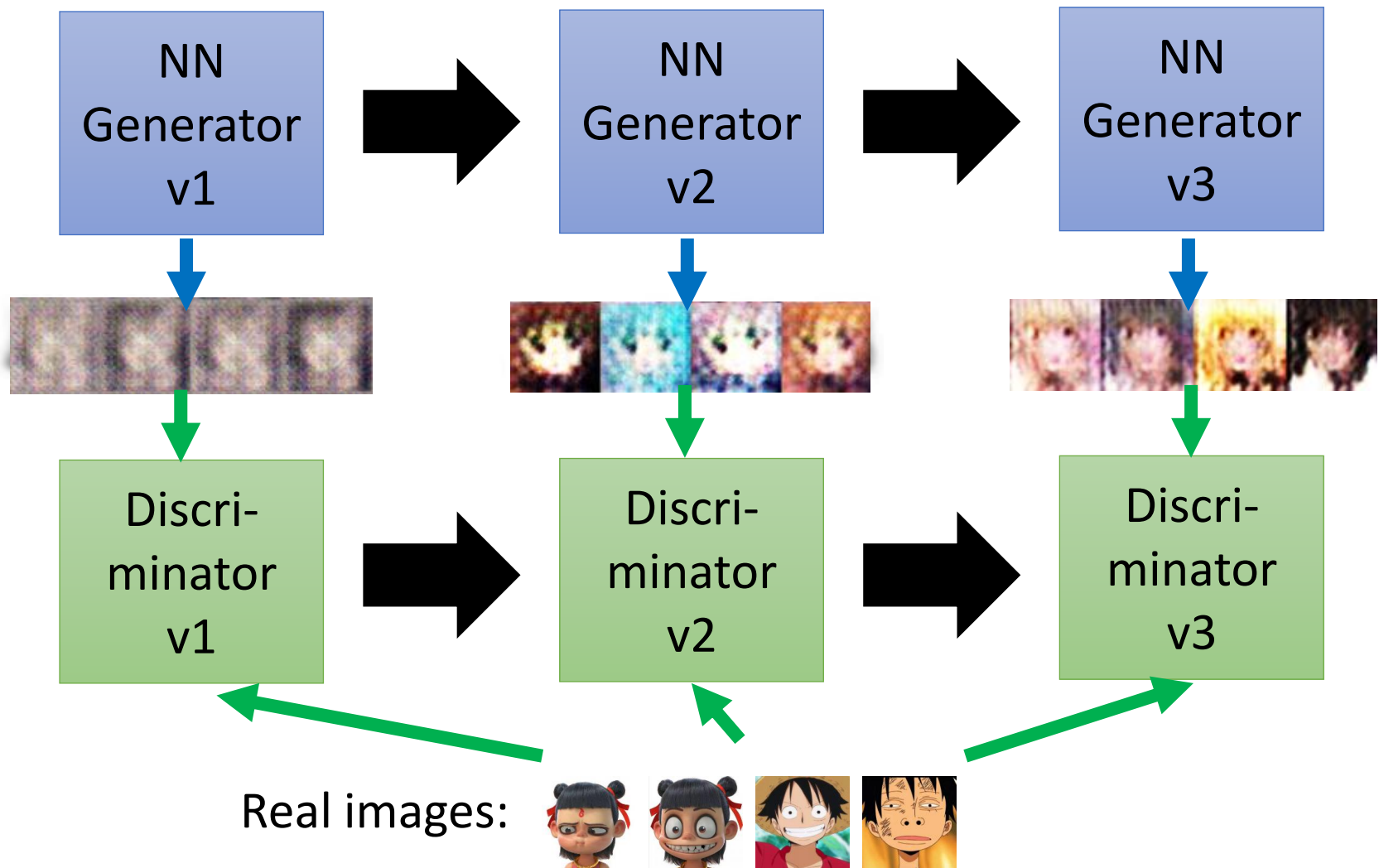
Basic Idea of GAN

Generator (student)   Discriminator (teacher)

Generator v1

Discriminator v1

没有两个圈

Generator v2

Discriminator v2

没有彩色

Generator v3

为什么不自己学？

为什么不自己做？

Basic Idea of GAN

- 写作敌人，念作朋友

Algorithm

- Initialize generator and discriminator   G   D

- In each training iteration:

Step 1: Fix generator G, and update discriminator D
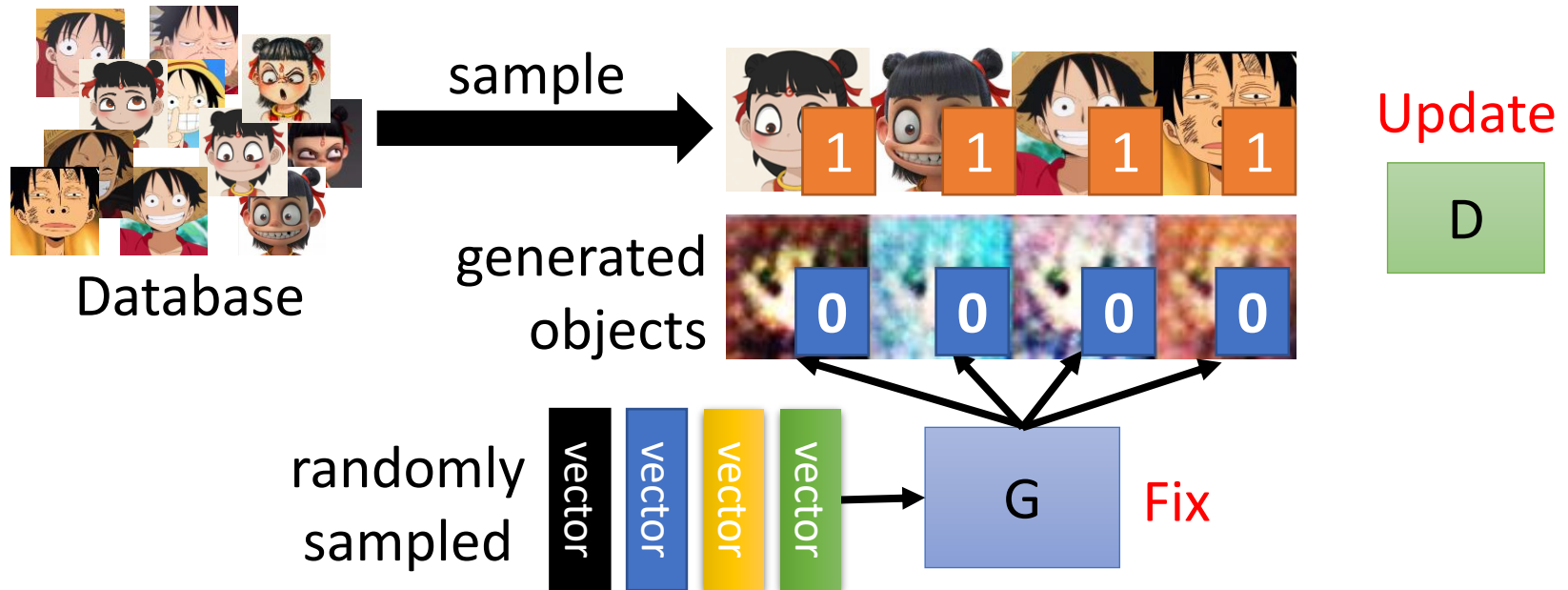
Discriminator learns to assign high scores to real objects and low scores to generated objects.

Algorithm

- Initialize generator and discriminator $\quad$ G $\quad$ D

- In each training iteration:

Step 2: Fix discriminator D, and update generator G

Generator learns to "fool" the discriminator

vector → NN Generator update → hidden layer → Discri-minator fix → 0.13

large network
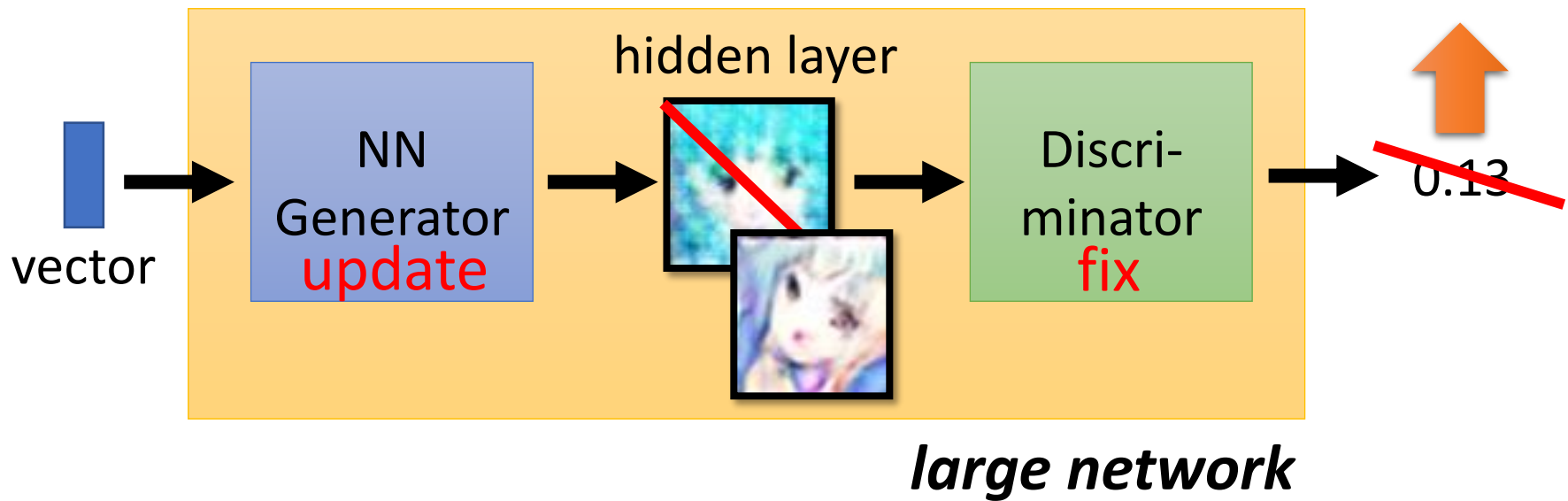
Algorithm

- Initialize generator and discriminator   G   D

- In each training iteration:

Learning D

Sample some real objects: 1 1 1 1

Generate some fake objects: 0 0 0 0

vector vector vector vector → G fix

Update   D

Learning G

vector vector vector vector → G update → i i i image → D fix → 1

Progressive GAN

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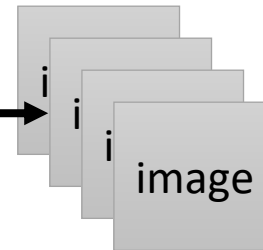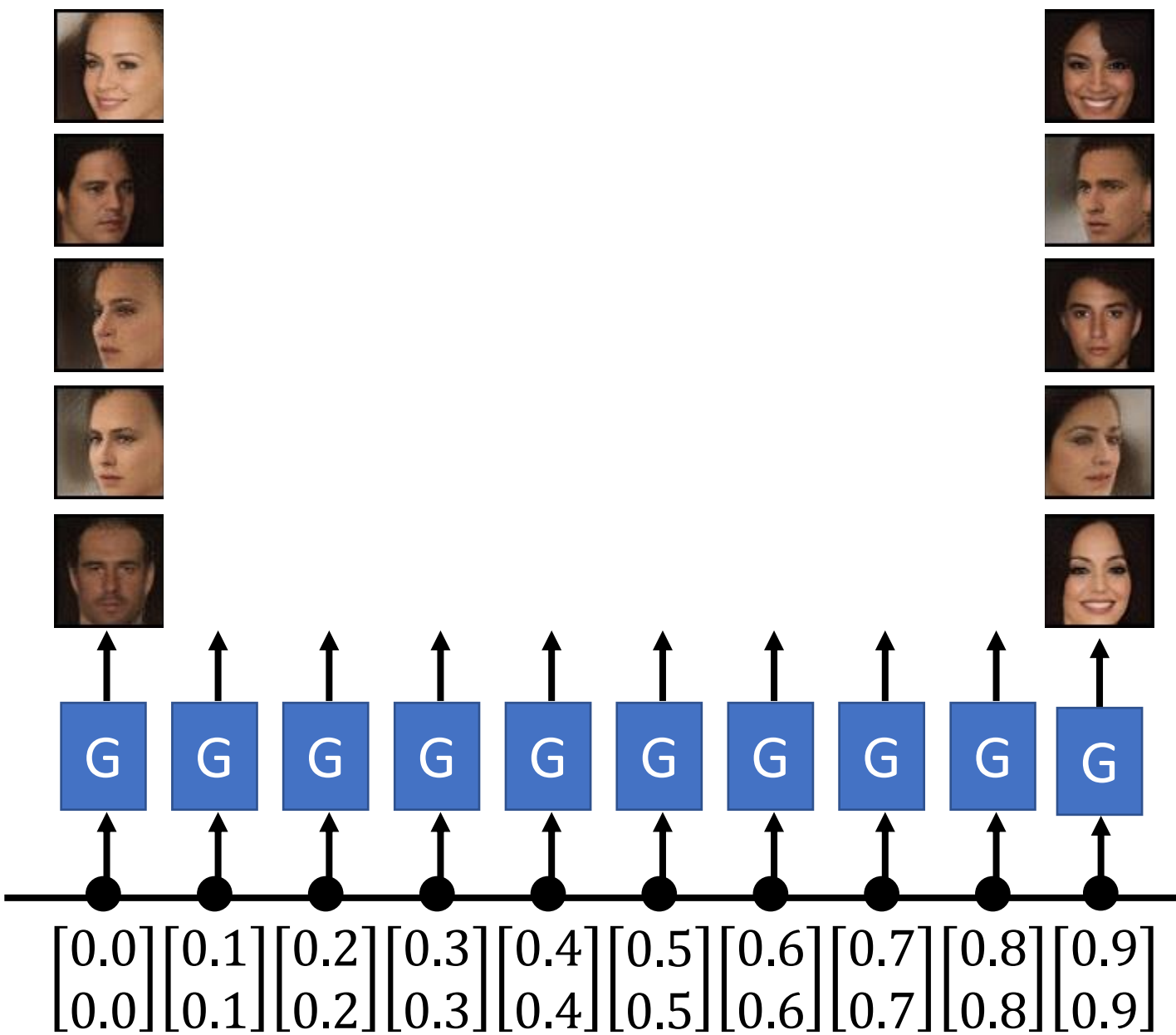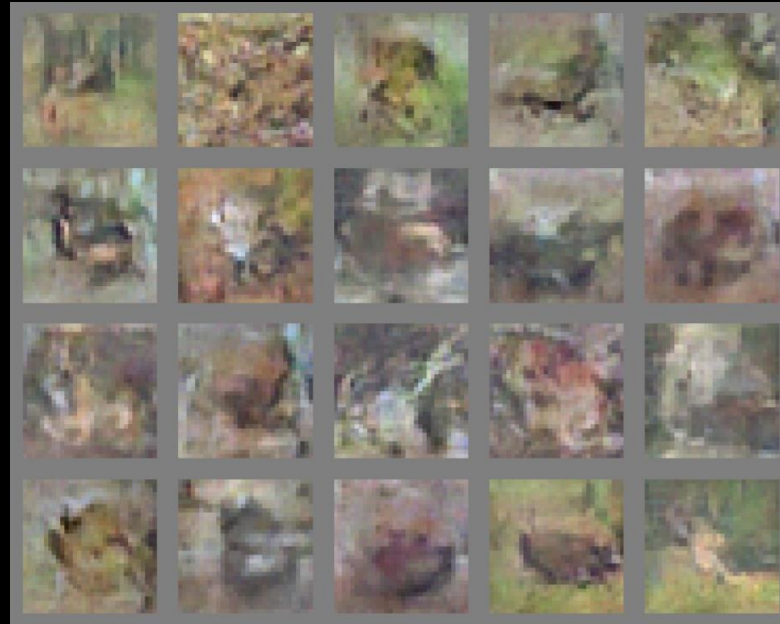
The first GAN

Today ...... BigGAN

GAN as structured learning

Structured Learning

Machine learning is to find a function f

$$f : X \longrightarrow Y$$

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
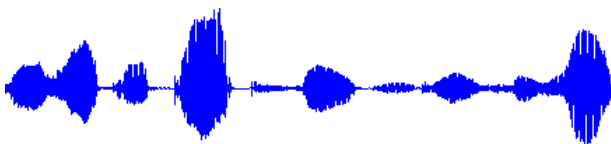
Output Sequence $f : X \rightarrow Y$

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

$$f : X \rightarrow Y$$

Image to Image

Colorization:

X : Y :

Ref: https://arxiv.org/pdf/1611.07004v1.pdf
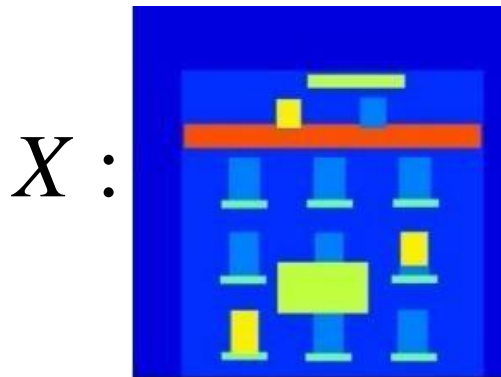
Text to Image

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

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.
  - Because the output components have dependency, they should be considered globally.

Image Generation
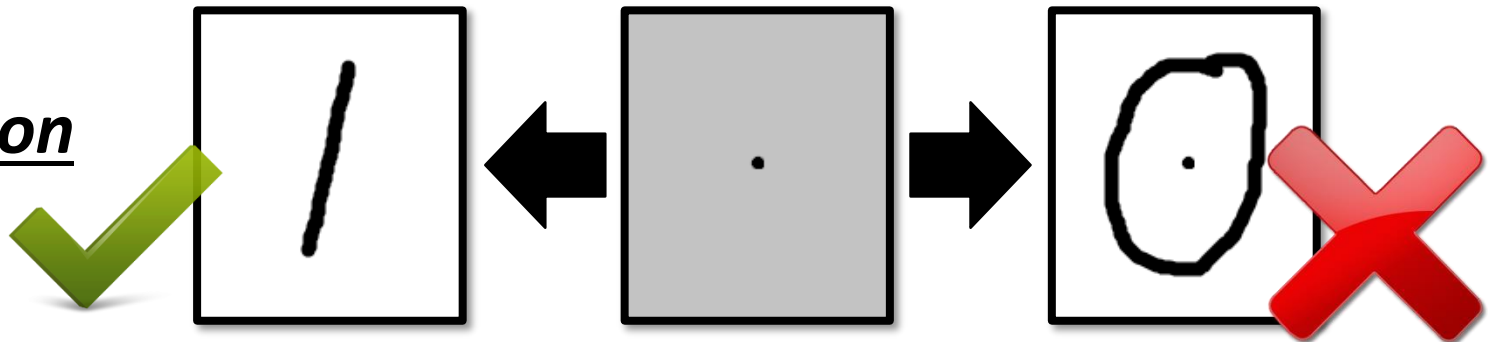
Structured Learning Approach

Generator

Learn to generate the object at the component level

Bottom Up

+

Top Down

→
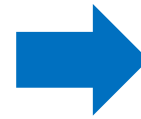
Generative Adversarial Network (GAN)

Discriminator

Evaluating the whole object, and find the best one

Can Generator learn by itself?
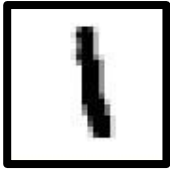
Generator

vectors

code:
(where does they come from?)

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

Image:

As close as possible

$$\begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix} \rightarrow \boxed{\text{NN Generator}} \rightarrow \boxed{\text{image}} \longleftrightarrow$$

As close as possible

c.f. $\rightarrow \boxed{\text{NN Classifier}} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix} \longleftrightarrow \begin{bmatrix} 1 \\ 0 \\ \vdots \end{bmatrix}$

Generator

vectors

code:
(where does they come from?)

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

Image:

Encoder in auto-encoder provides the code ☺

NN Encoder → c

Auto-encoder
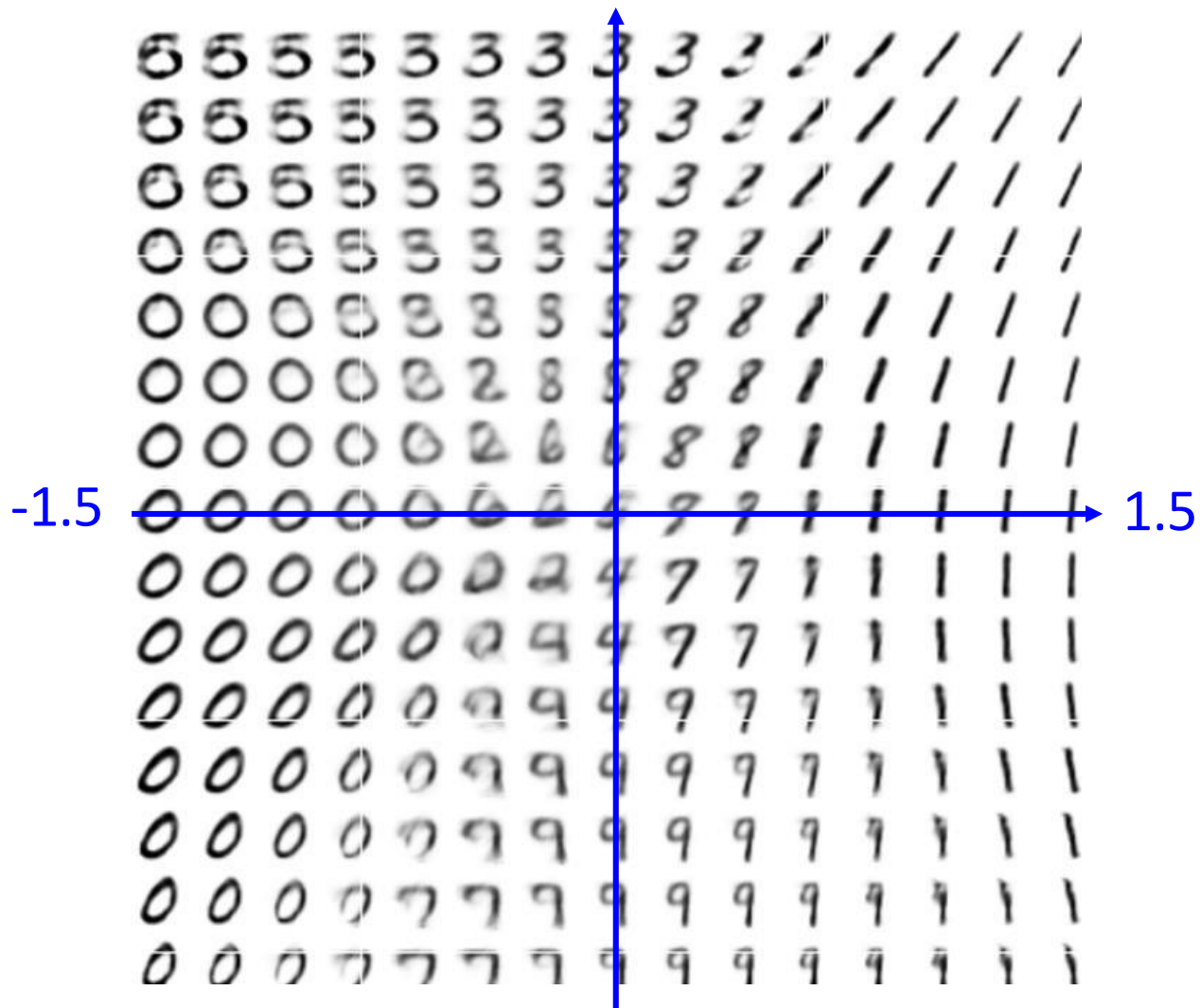

Low dimension

Compact representation of the input object

28 X 28 = 784

Learn together

Can reconstruct the original object

Trainable

Auto-encoder

As close as possible

NN Encoder → code → NN Decoder
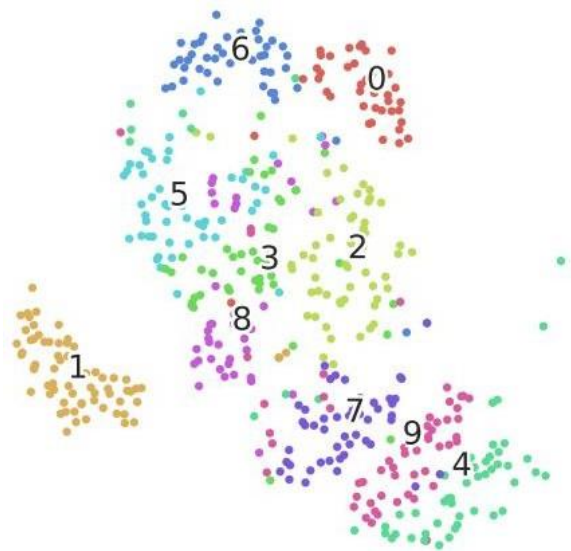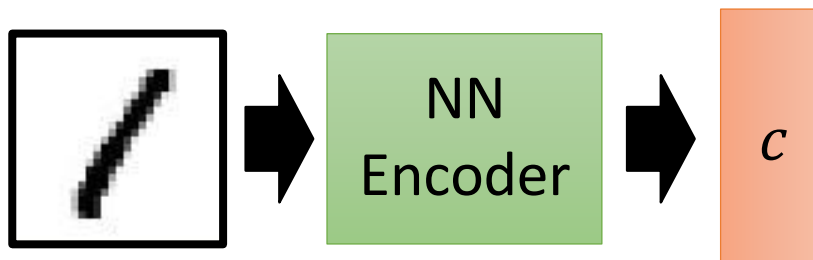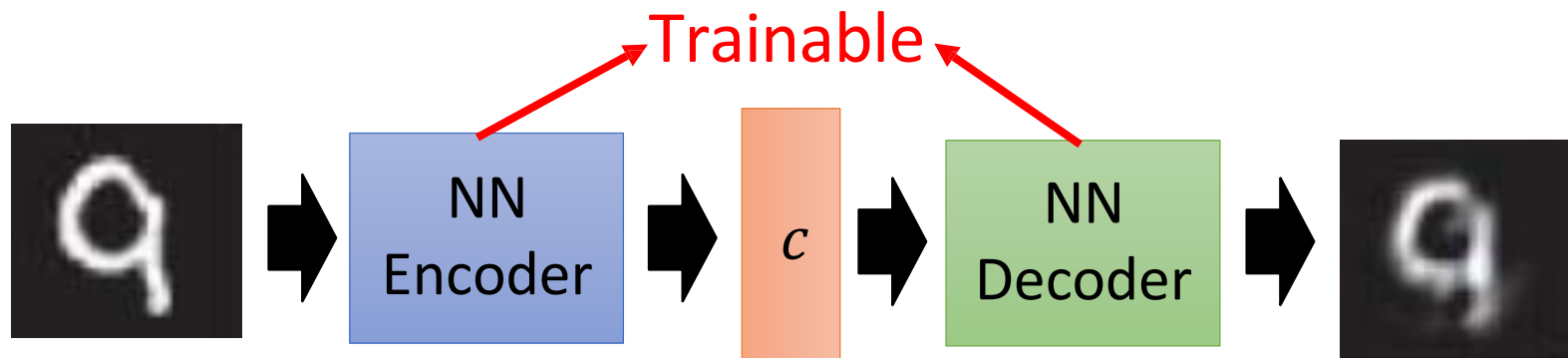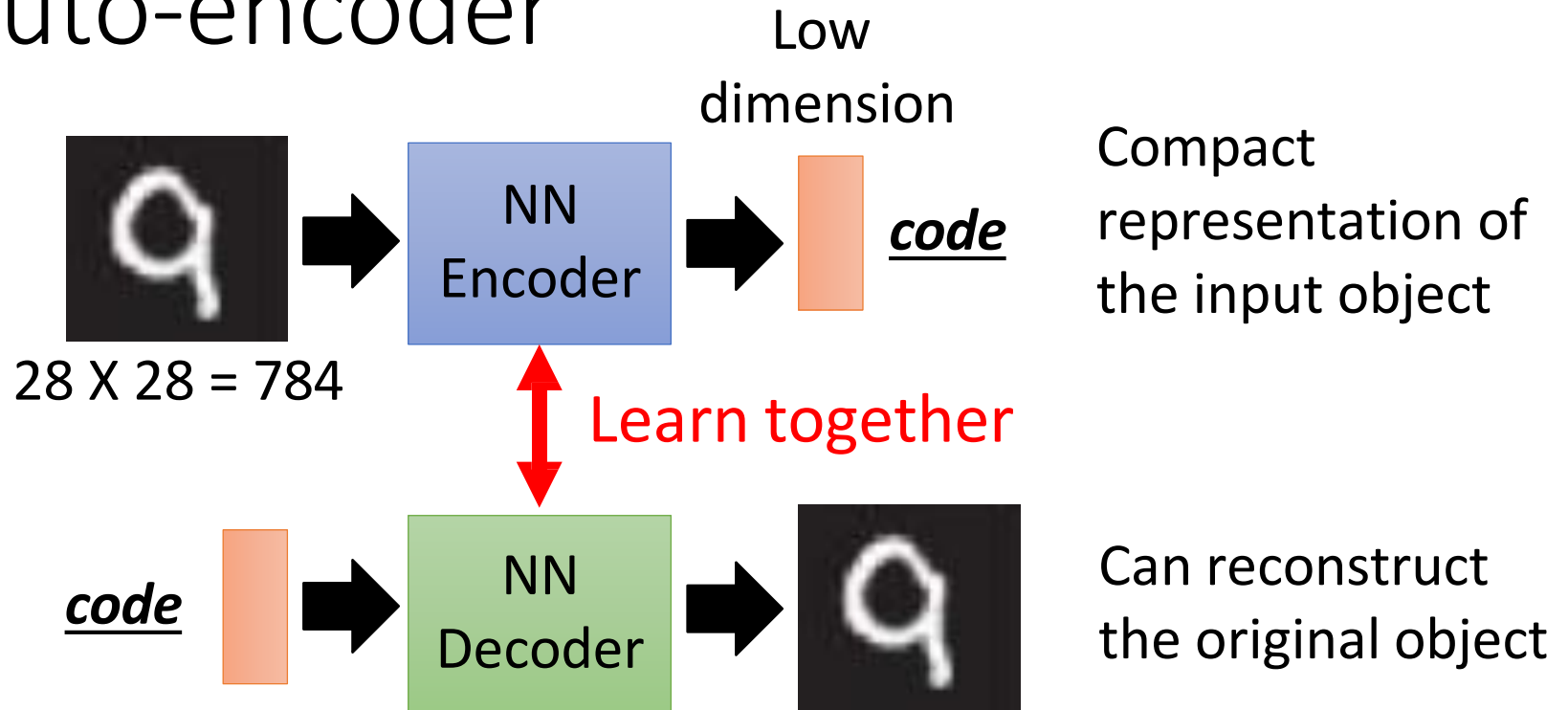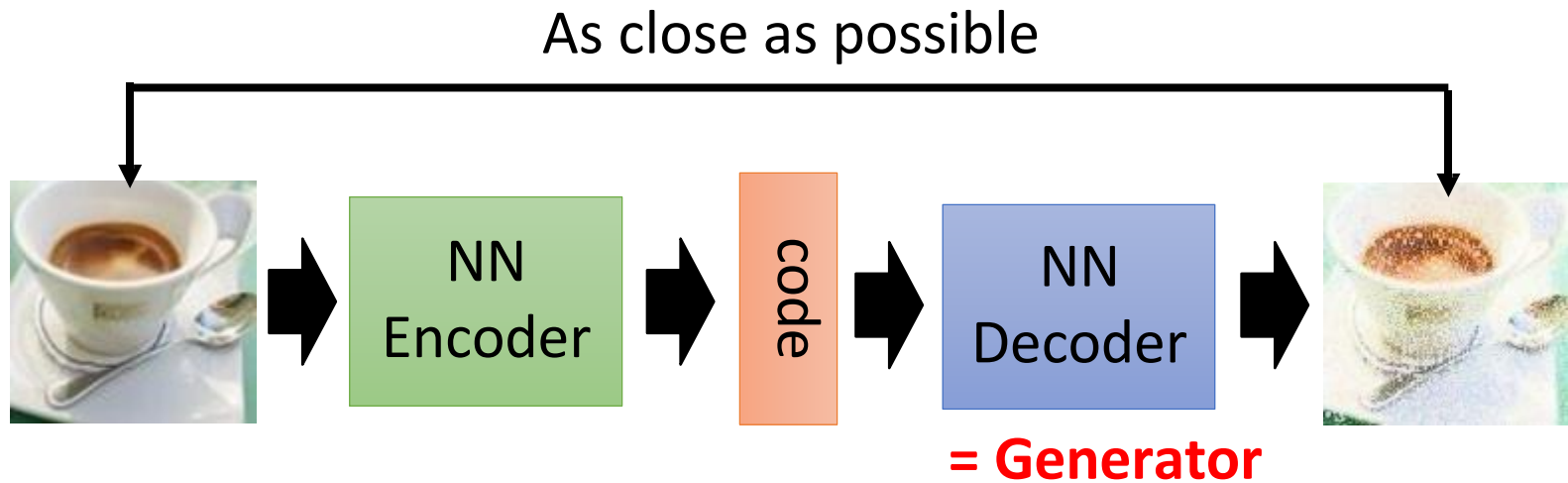
= Generator

Randomly generate a vector as code

code → NN Decoder → Image ?

= Generator

Auto-encoder

2D

code

NN Decoder → image

NN Decoder → O

$$\begin{bmatrix} 1.5 \\ 0 \end{bmatrix}$$

-1.5 • ———————————————— •→ 1.5

$$\begin{bmatrix} -1.5 \\ 0 \end{bmatrix}$$

NN Decoder → l

Auto-encoder

-1.5   1.5

What do we miss?

Generated Image   Target

G

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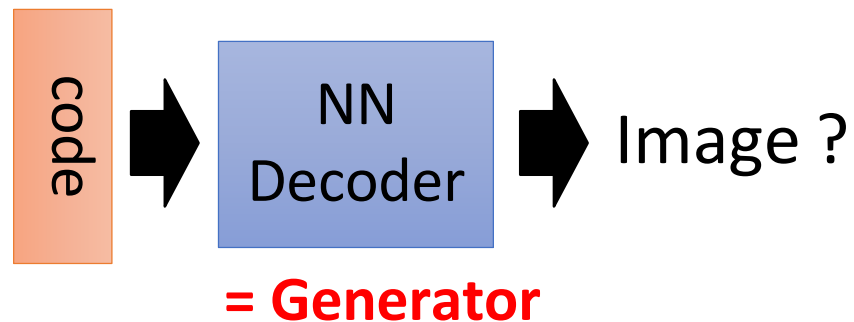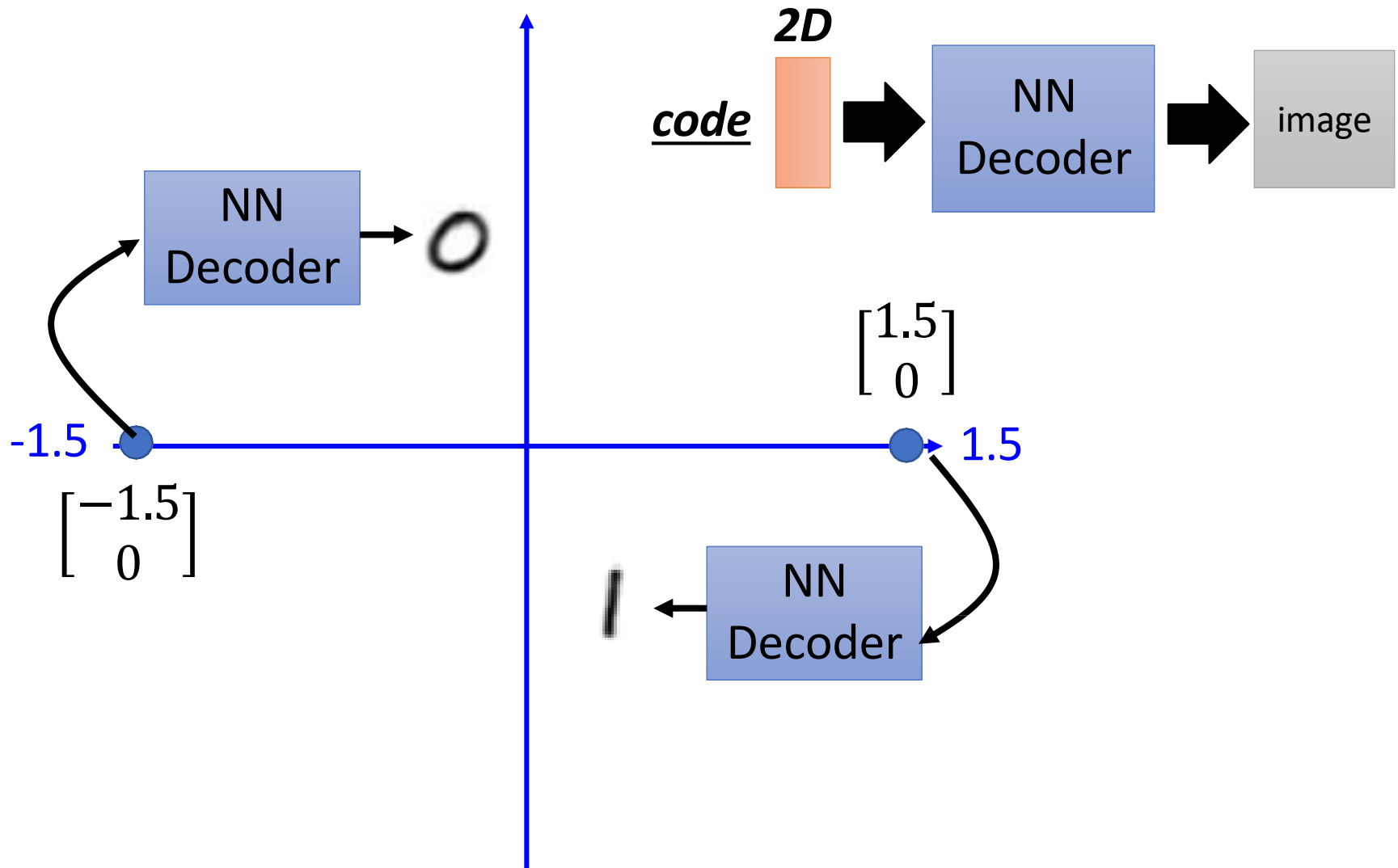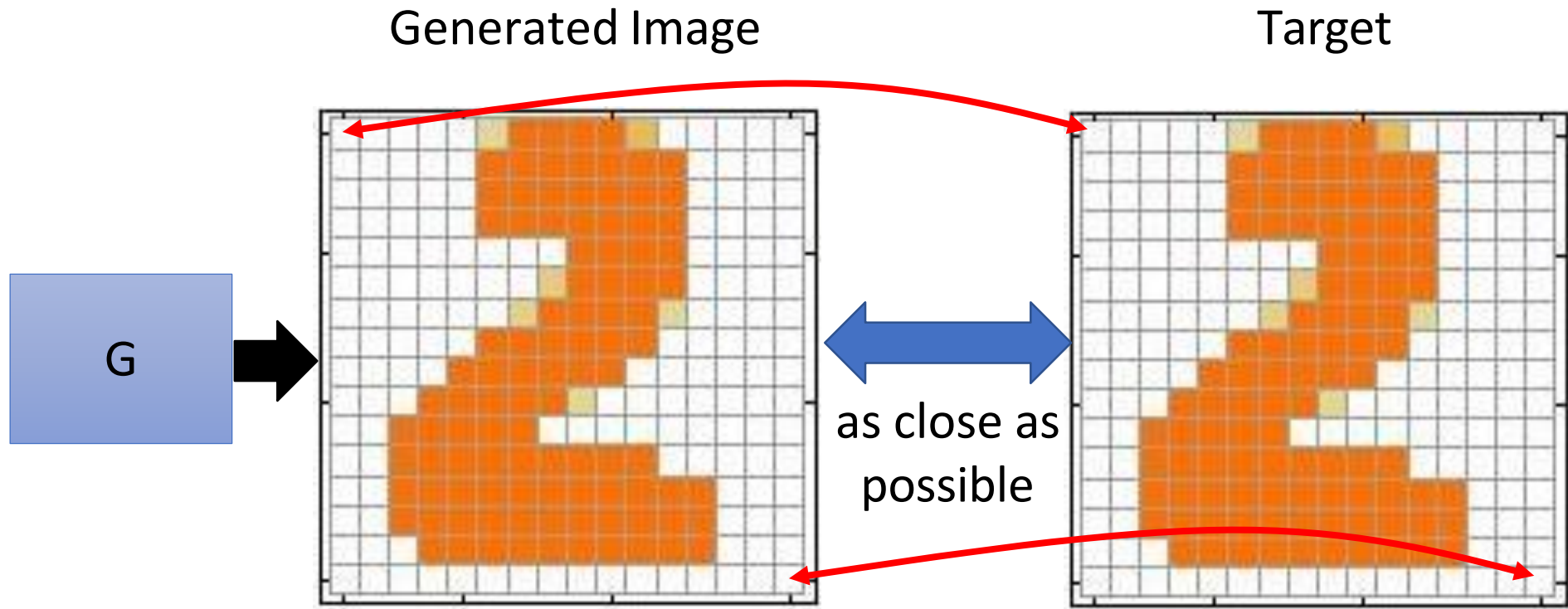
It will be fine if the generator can truly copy the target image.

What if the generator makes some mistakes ....

Some mistakes are serious, while some are fine.

What do we miss?

Target
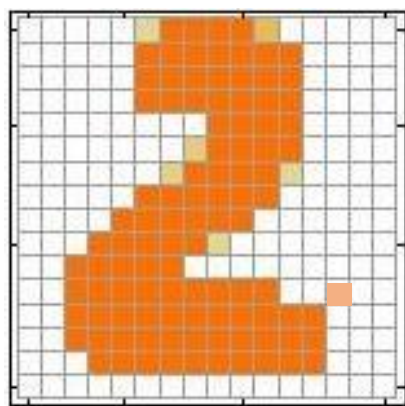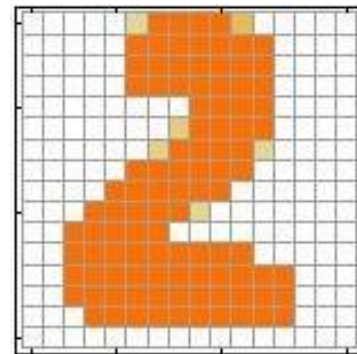
1 pixel error

我觉得不行
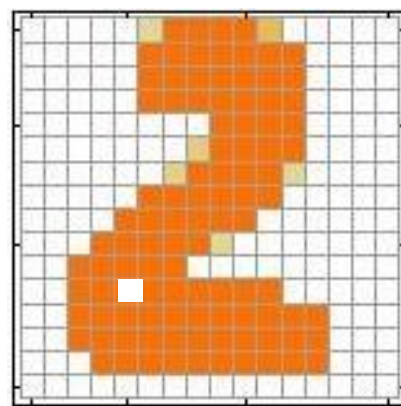
1 pixel error

我觉得不行
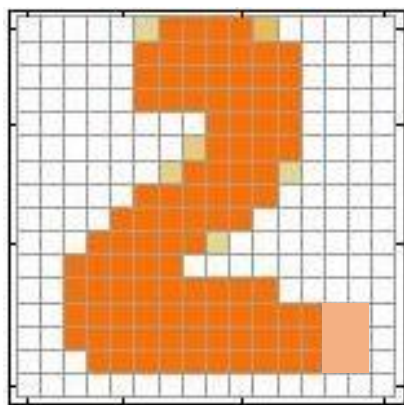
6 pixel errors
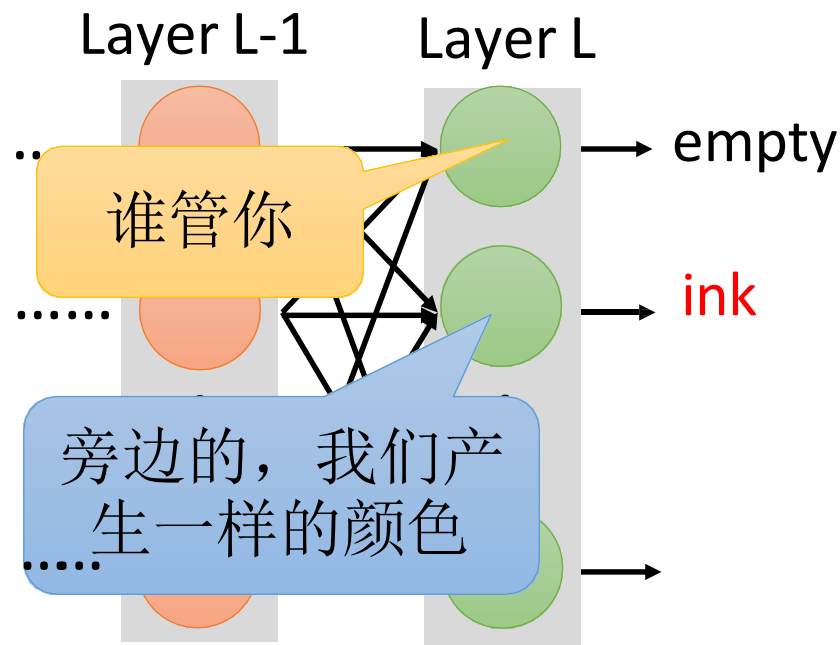
我觉得其实
可以
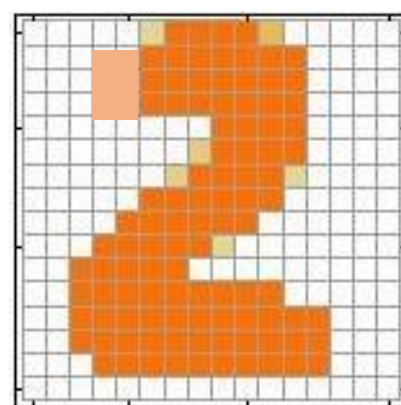
6 pixel errors

我觉得其实
可以

What do we miss?

Each neural in output layer corresponds to a pixel.

我觉得不行　　我觉得其实可以

Layer L-1　　Layer L

谁管你

旁边的，我们产生一样的颜色

empty

ink
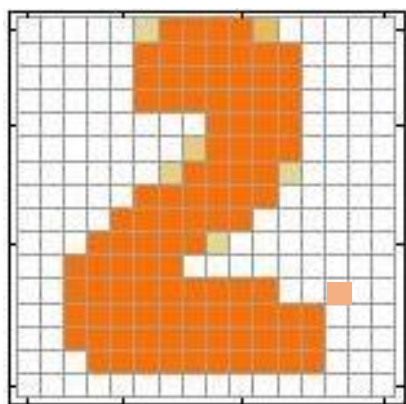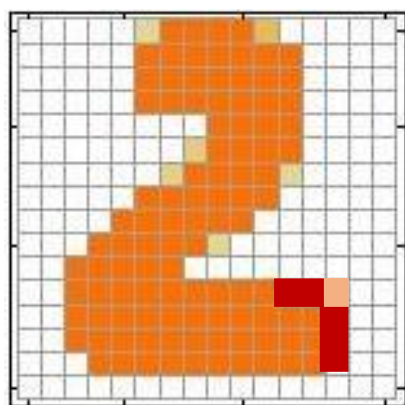
The relation between the components are critical.

Although highly correlated, they cannot influence each other.

Need deep structure to catch the relation between components.

(Variational) Auto-encoder

$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ ➡ G ➡ $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
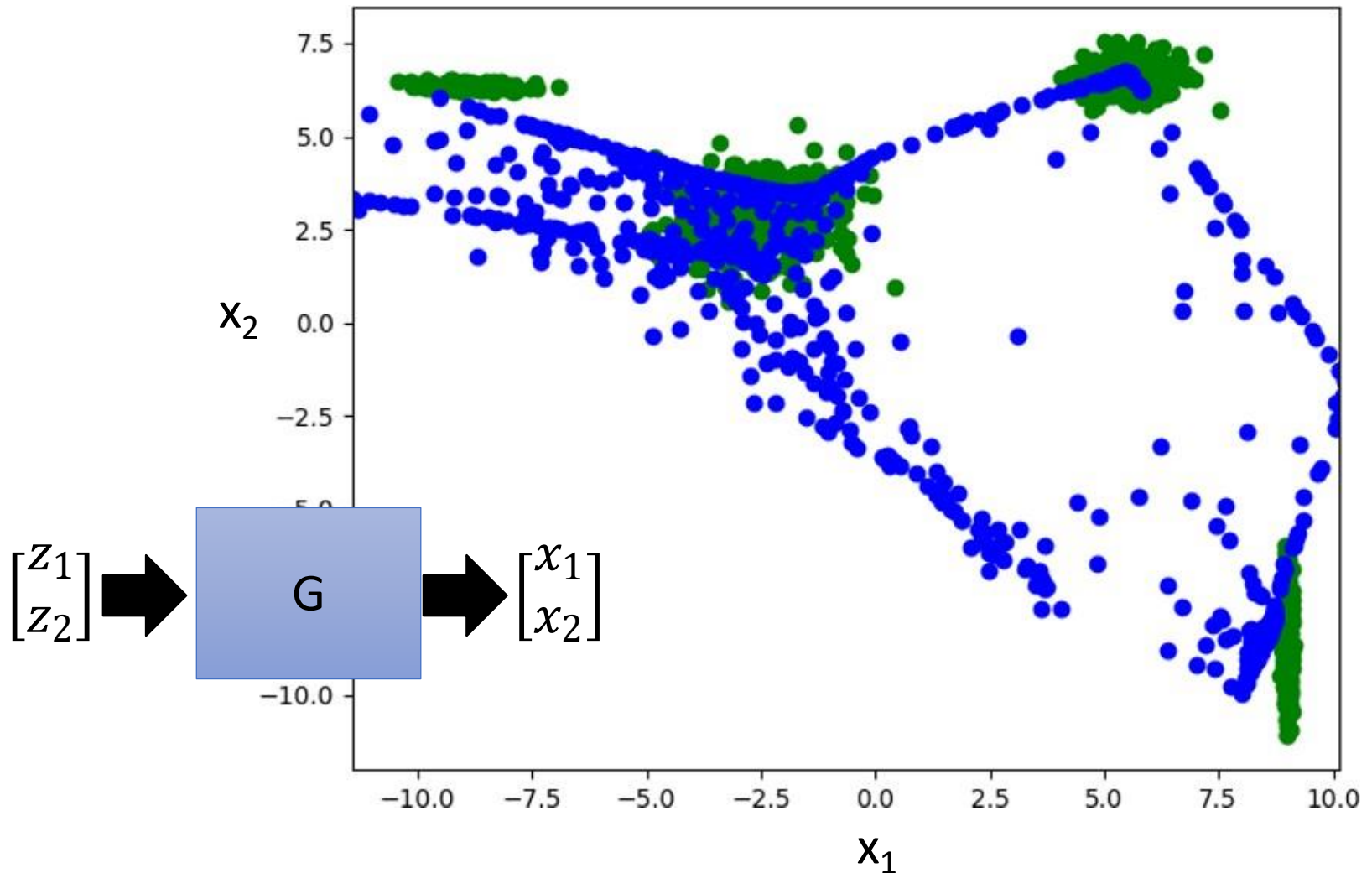
Can Discriminator generate?

Discriminator

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

$$D : X \rightarrow R$$

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

 D → 1.0
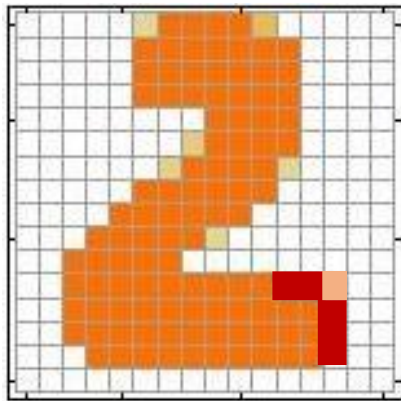
 D → 0.1

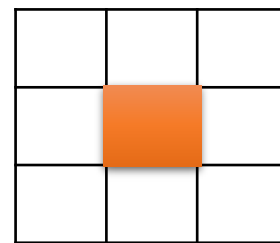Can we use the discriminator to generate objects?

Yes.

Discriminator

- It is easier to catch the relation between the components by top-down evaluation.
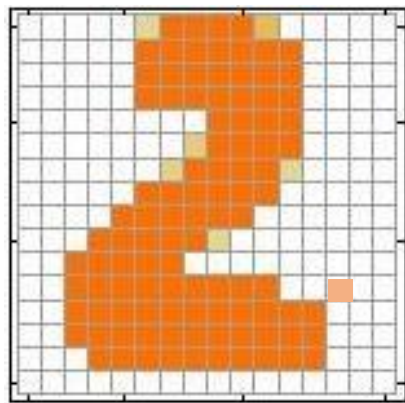
我觉得不行　　我觉得其实OK

This CNN filter is good enough.

Discriminator

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

Inference

- Generate object $\tilde{x}$ that

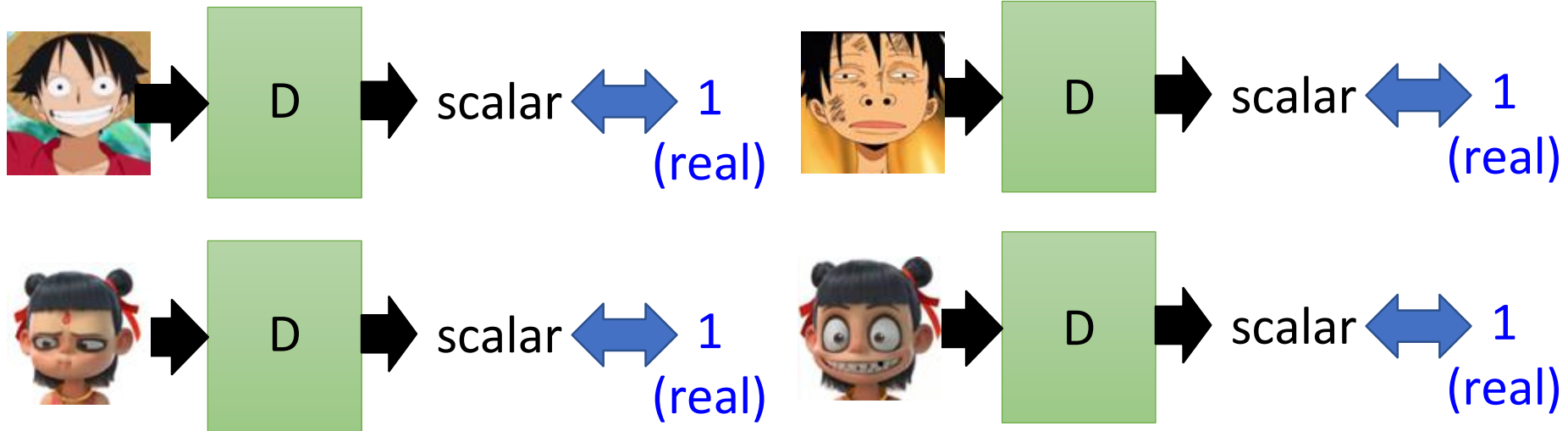$$\tilde{x} = \arg\max_{x \in X} D(x)$$

Enumerate all possible x !!!

It is feasible ???

How to learn the discriminator?
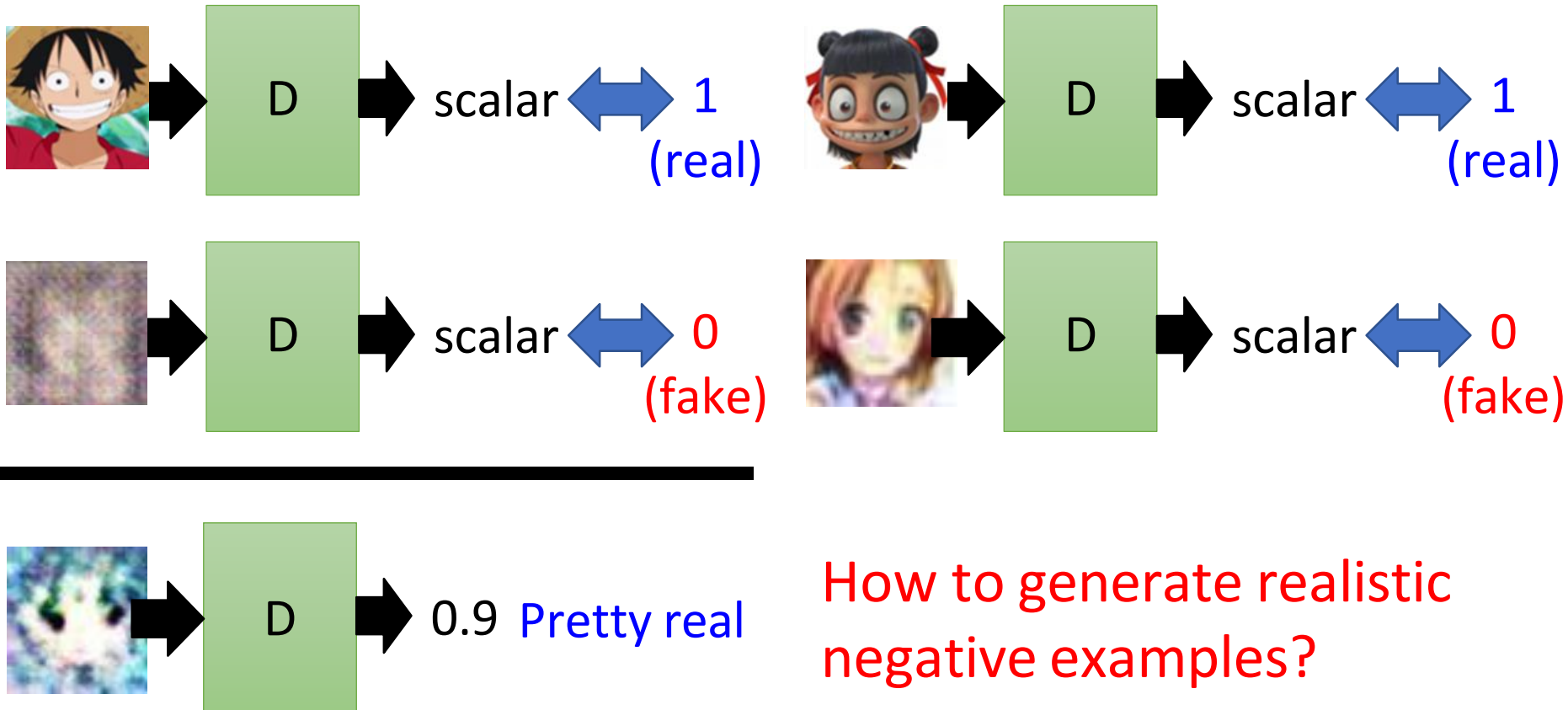
Discriminator - Training

- I have some real images

Discriminator only learns to output "1" (real).

Discriminator training needs some negative examples.

Discriminator - Training

- Negative examples are critical.

scalar ⟷ 1 (real)

scalar ⟷ 1 (real)

scalar ⟷ 0 (fake)

scalar ⟷ 0 (fake)

0.9 Pretty real

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

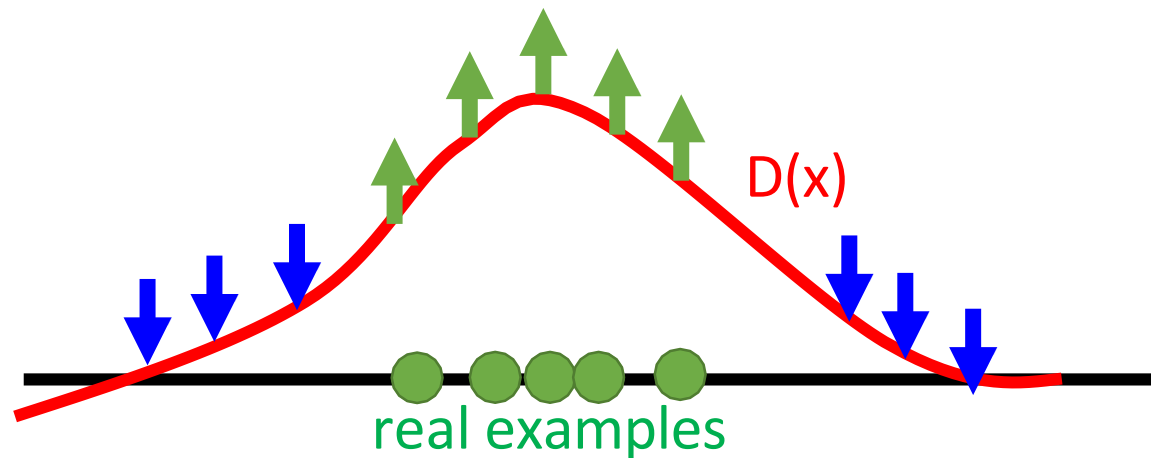   - Learn a discriminator D that can discriminate positive and negative examples.

     v.s. ➡️ D
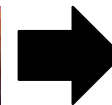
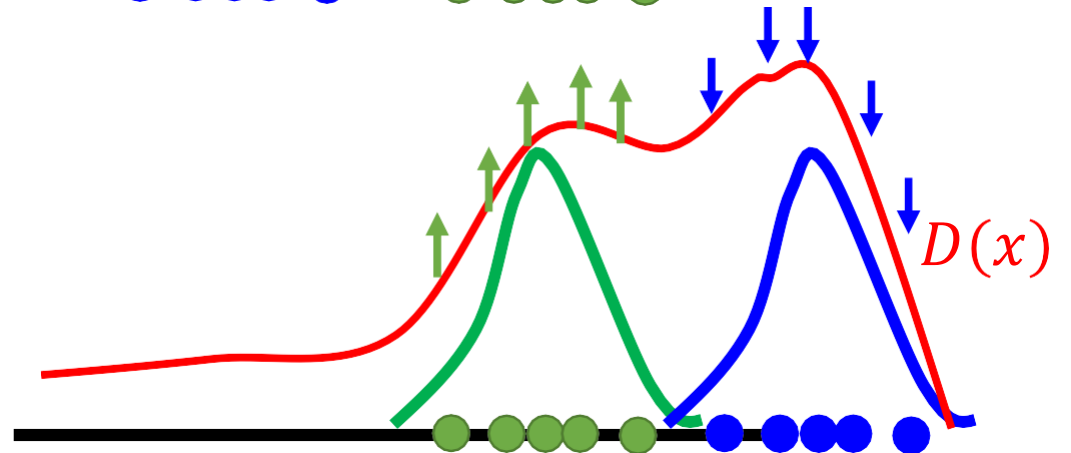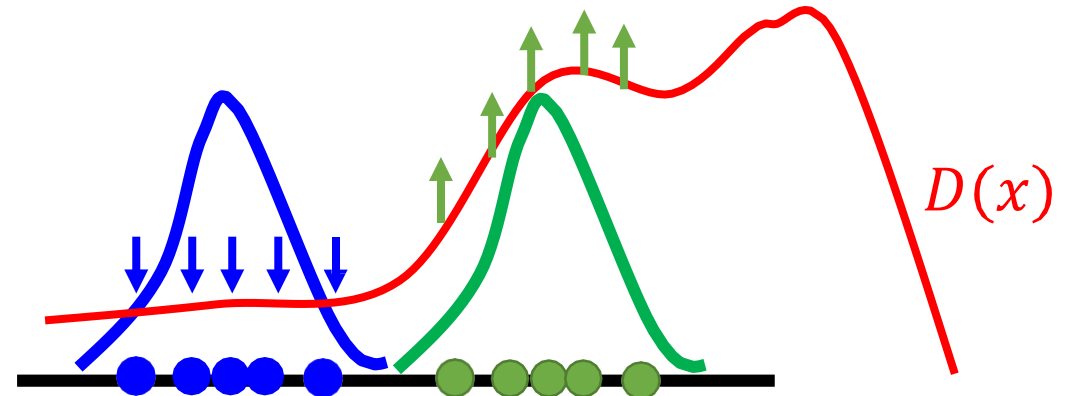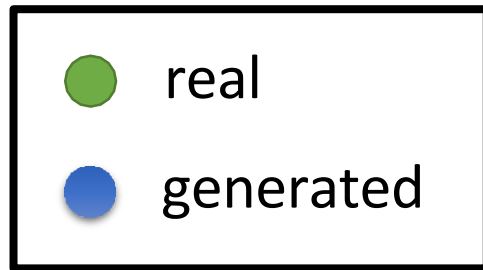   - Generate negative examples by discriminator D

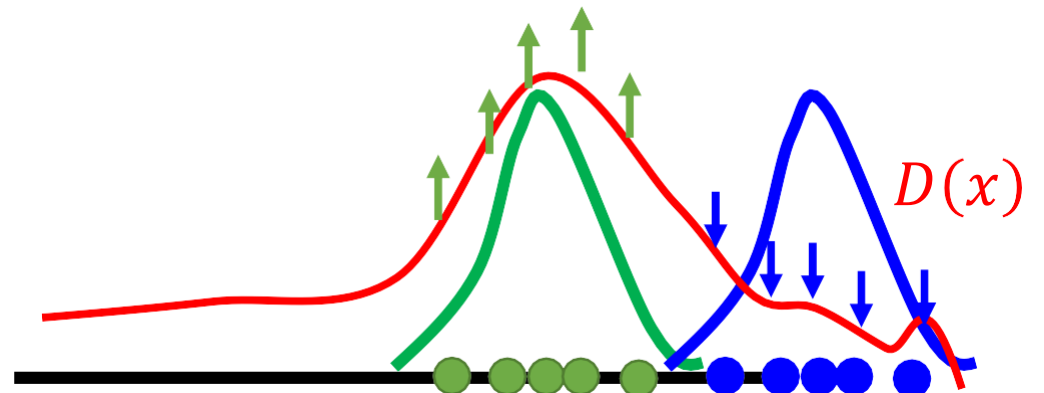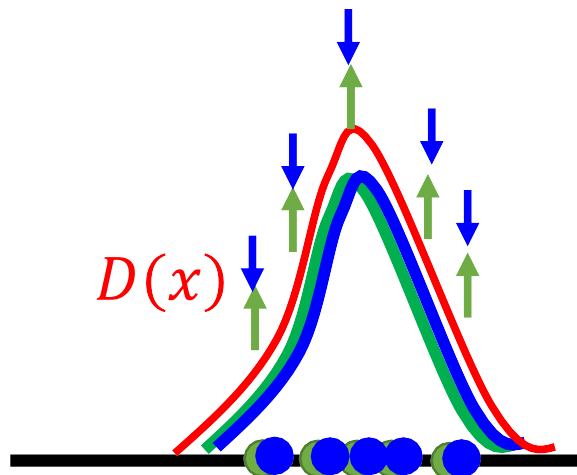 $$\tilde{x} = \arg \max_{x \in X} D(x)$$

Discriminator - Training

object x → Discriminator D → scalar D(x)

D(x)

real examples

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

Discriminator - Training

real — (green circle)

generated — (blue circle)

In the end ......

$D(x)$

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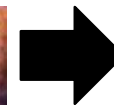 - In each iteration
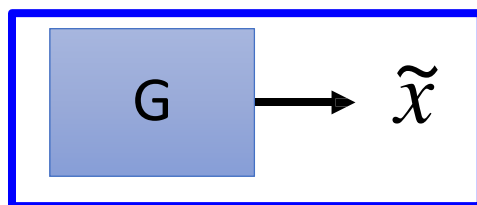   - Learn a discriminator D that can discriminate positive and negative examples.

       v.s. ➡ D
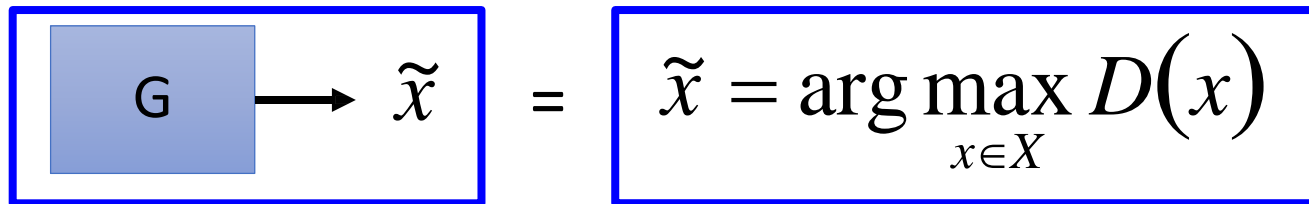
    - Generate negative examples by discriminator D

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

Benefit of GAN

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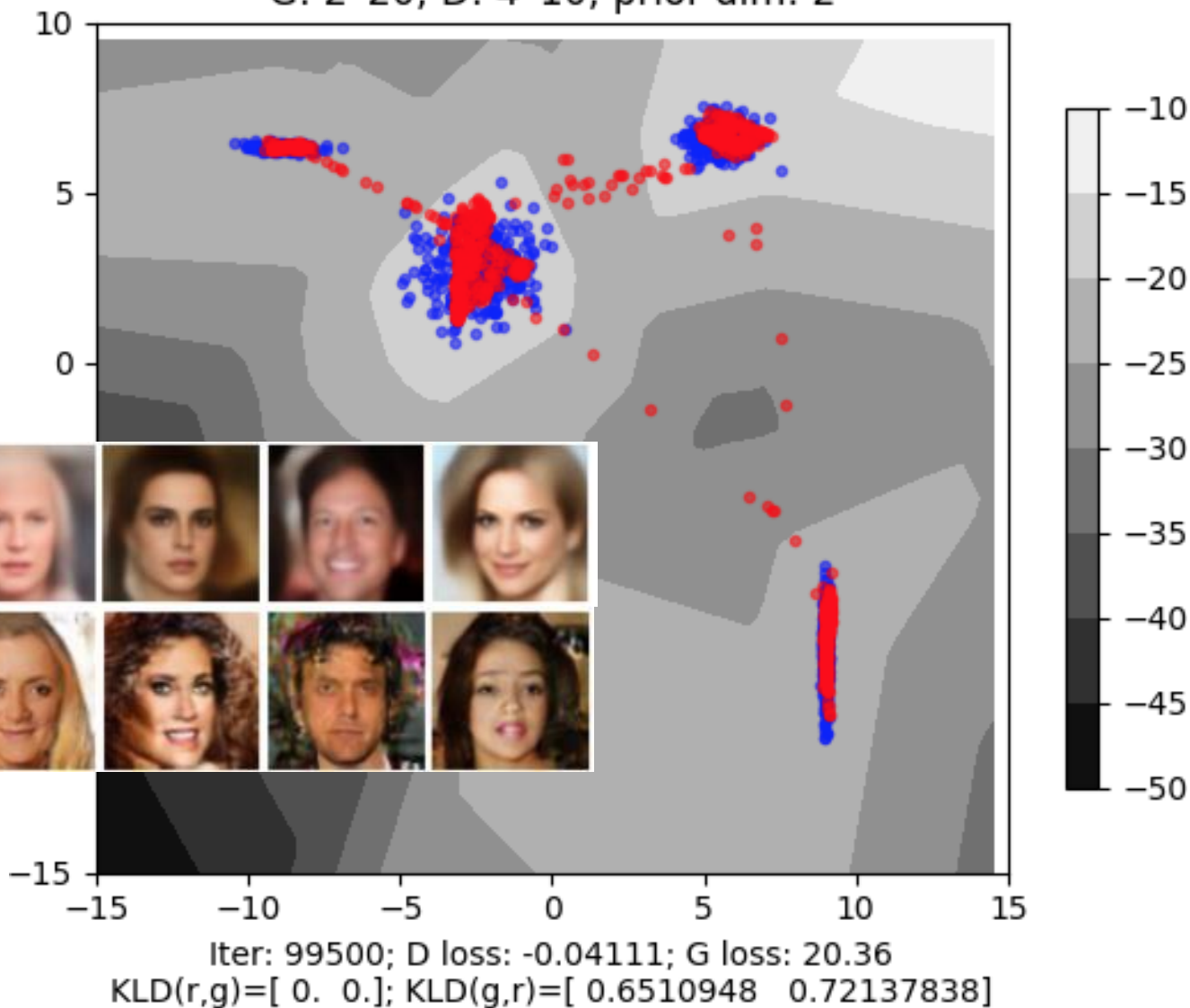 - Using generator to generate negative samples

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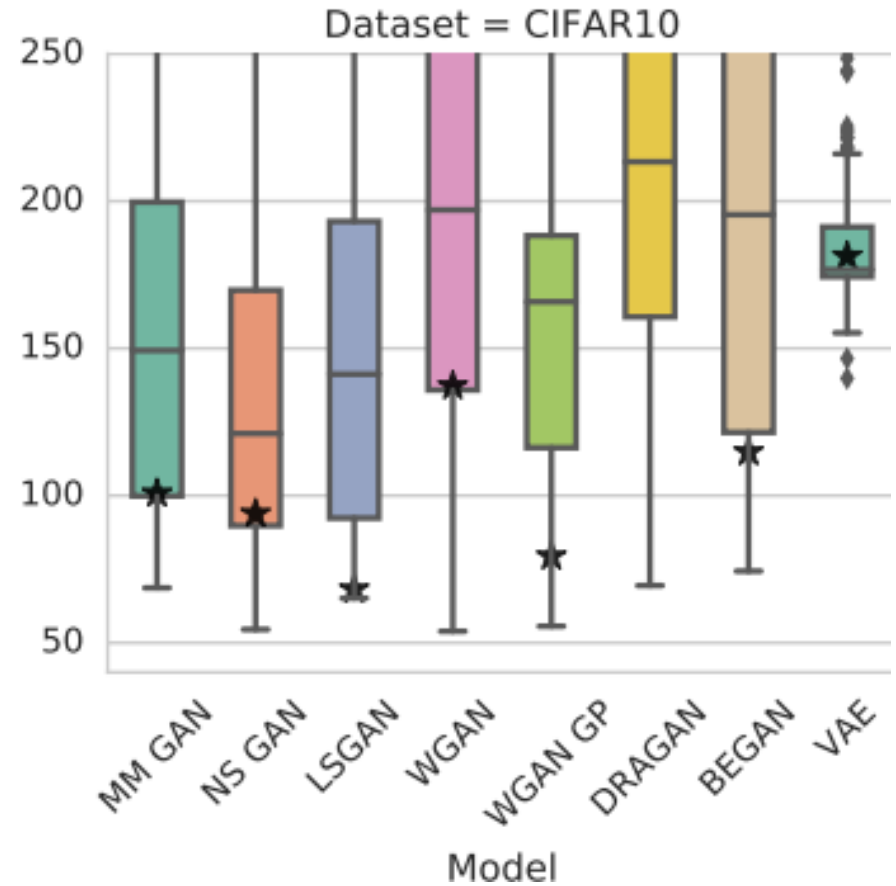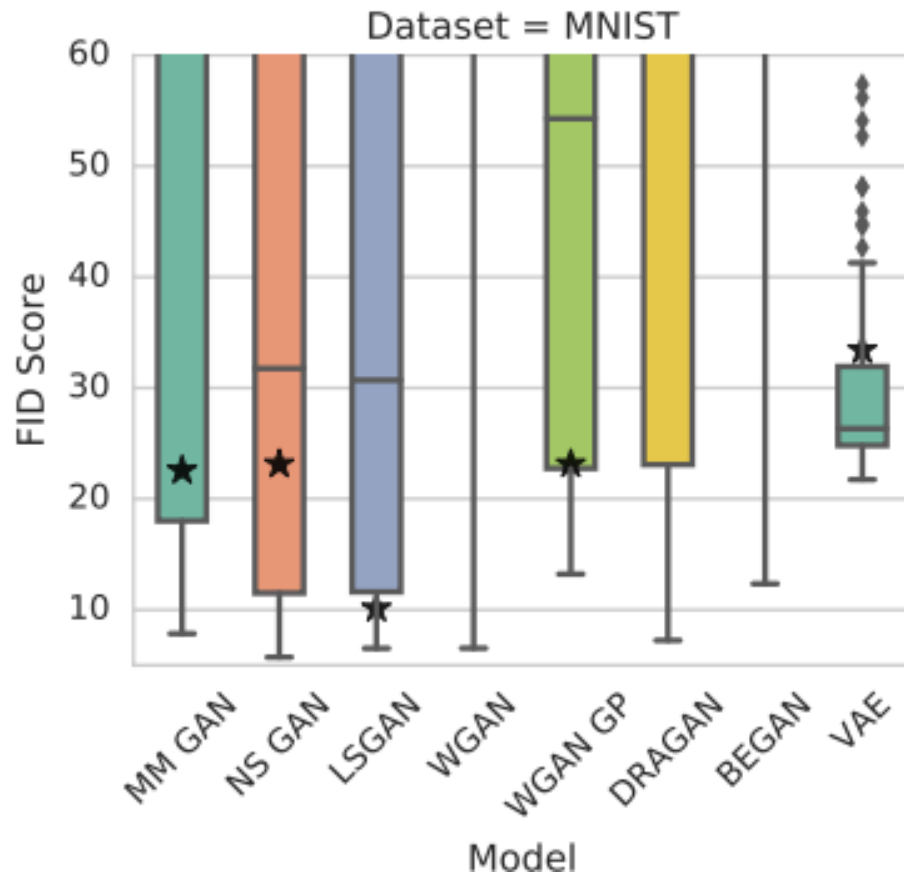
efficient

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

GAN

VAE

GAN

https://arxiv.org/abs/1512.09300

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