Machine Learning, Spring 2020

Generative Adversarial Networks (GAN)

Instructor: Prof. Yi Fang

yfang@nyu.edu

Python tutorial: http://learnpython.org/

TensorFlow tutorial: https://www.tensorflow.org/tutorials/

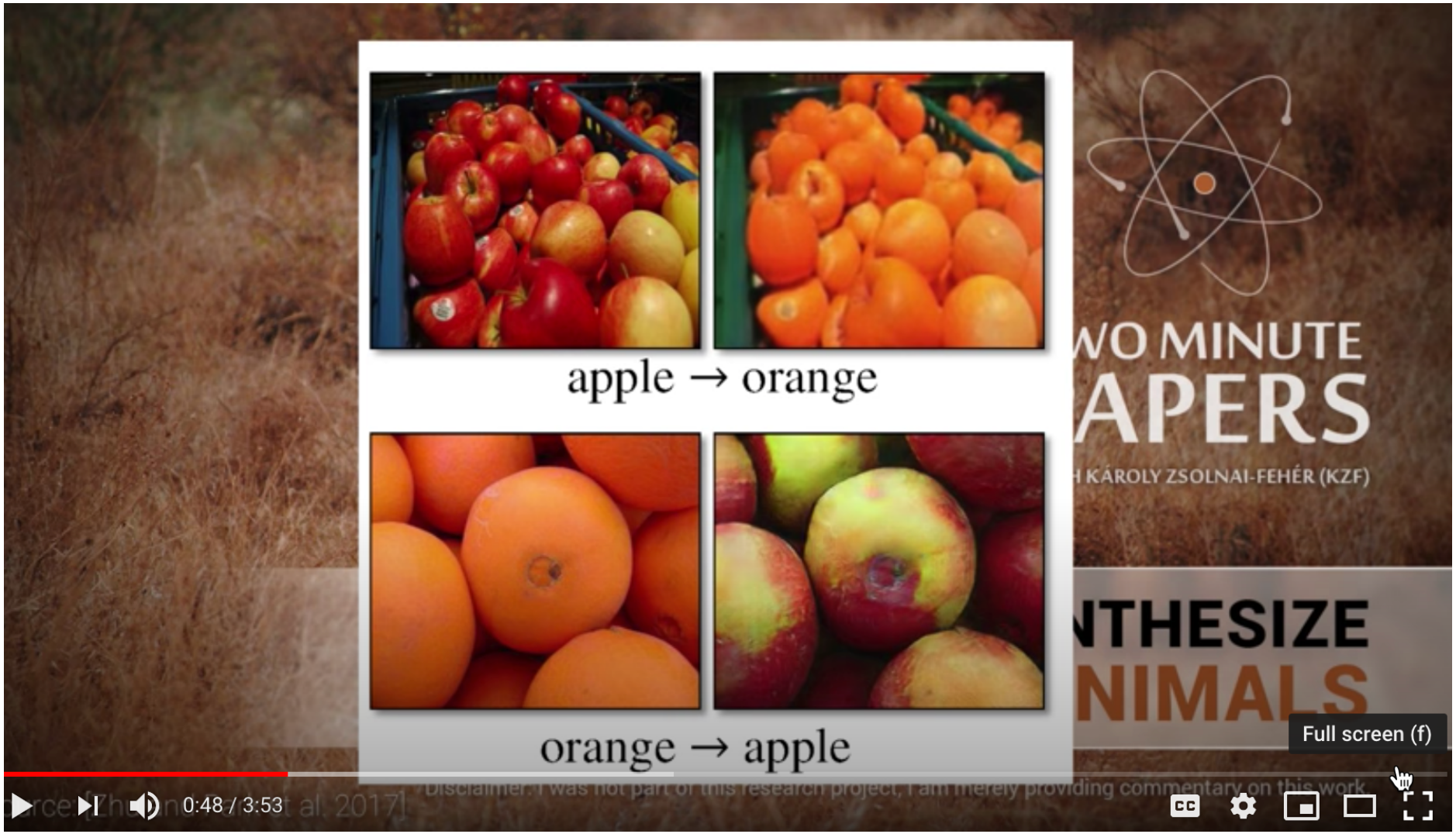
PyTorch tutorial: https://pytorch.org/tutorials/

Acknowledge: The slides are partially referred to the online materials by Taegyun Joen, https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection and online YOLO paper and other materials (from ECS289g by Prof. Lee)

Magic of GANs

- Music Generation

Image Style Transfer



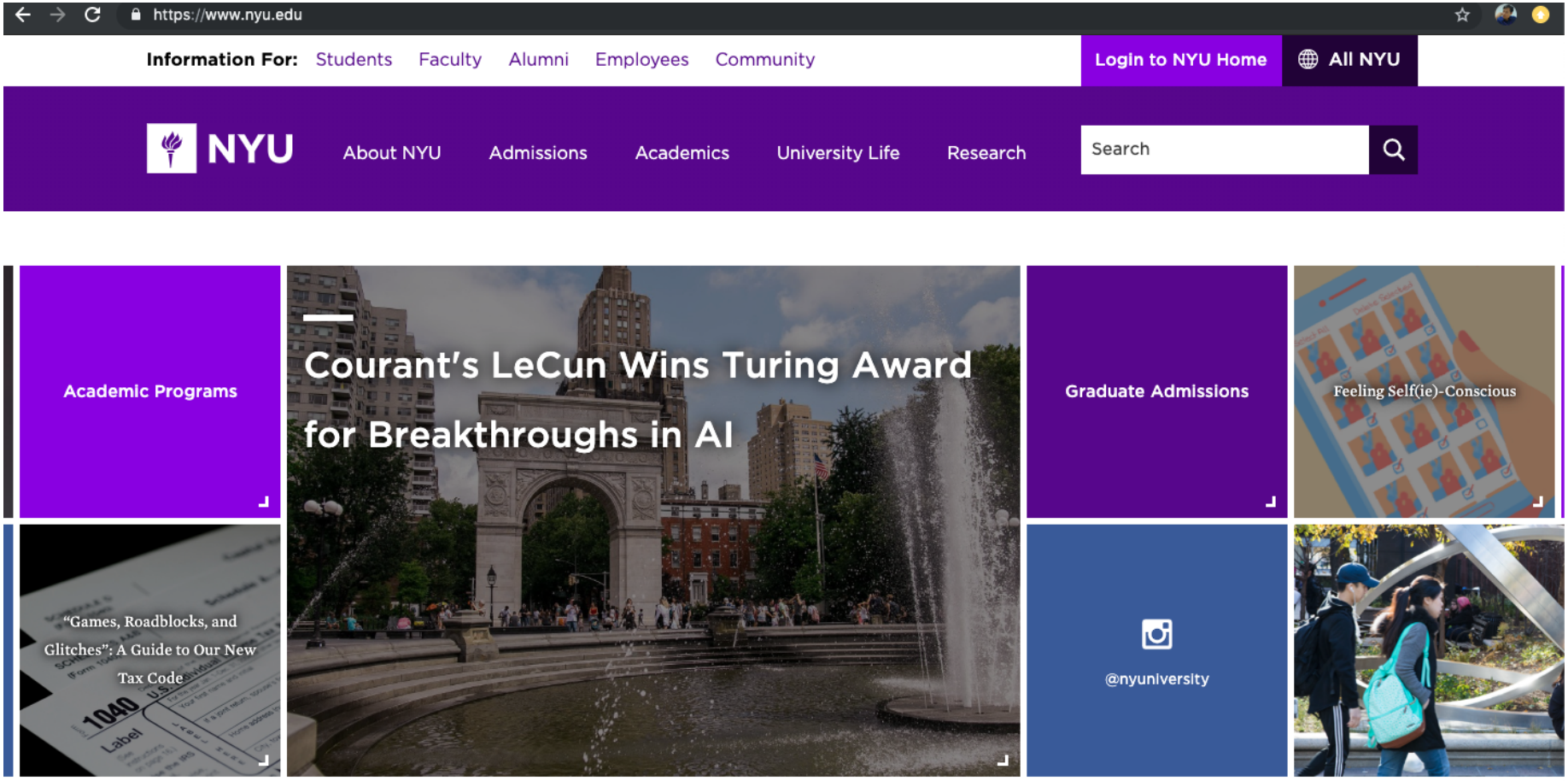
Magic of GANs

- Which one is Computer generated?

Src: Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." arXiv preprint arXiv:1609.04802 (2016).

ARTISTIC STYLE TRANSFER

Src: https://neurohive.io/en/state-of-the-art/twin-gan-cross-domain-translation-of-human-portraits/


https://www.nyu.edu
Information For: Students Faculty Alumni Employees Community
Login to NYU Home
All NYU
NYU
About NYU
Admissions
Academics
University Life
Research
Search
Academic Programs
Courant's LeCun Wins Turing Award for Breakthroughs in AI
Graduate Admissions
Feeling Self(ie)-Conscious
"Games, Roadblocks, and Glitches": A Guide to Our New Tax Code
@nyuniversity

May 5, 2019

LeCun, NIPS 2016

- Reinforcement learning (cherry)
- Supervised learning (Chocolate)
- Unsupervised/Predictive learning (Cake)
  - Generative adversarial nets (GAN)

"GANs are the most interesting idea in the last 10 years in ML"

Yann LeCun.

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

.....

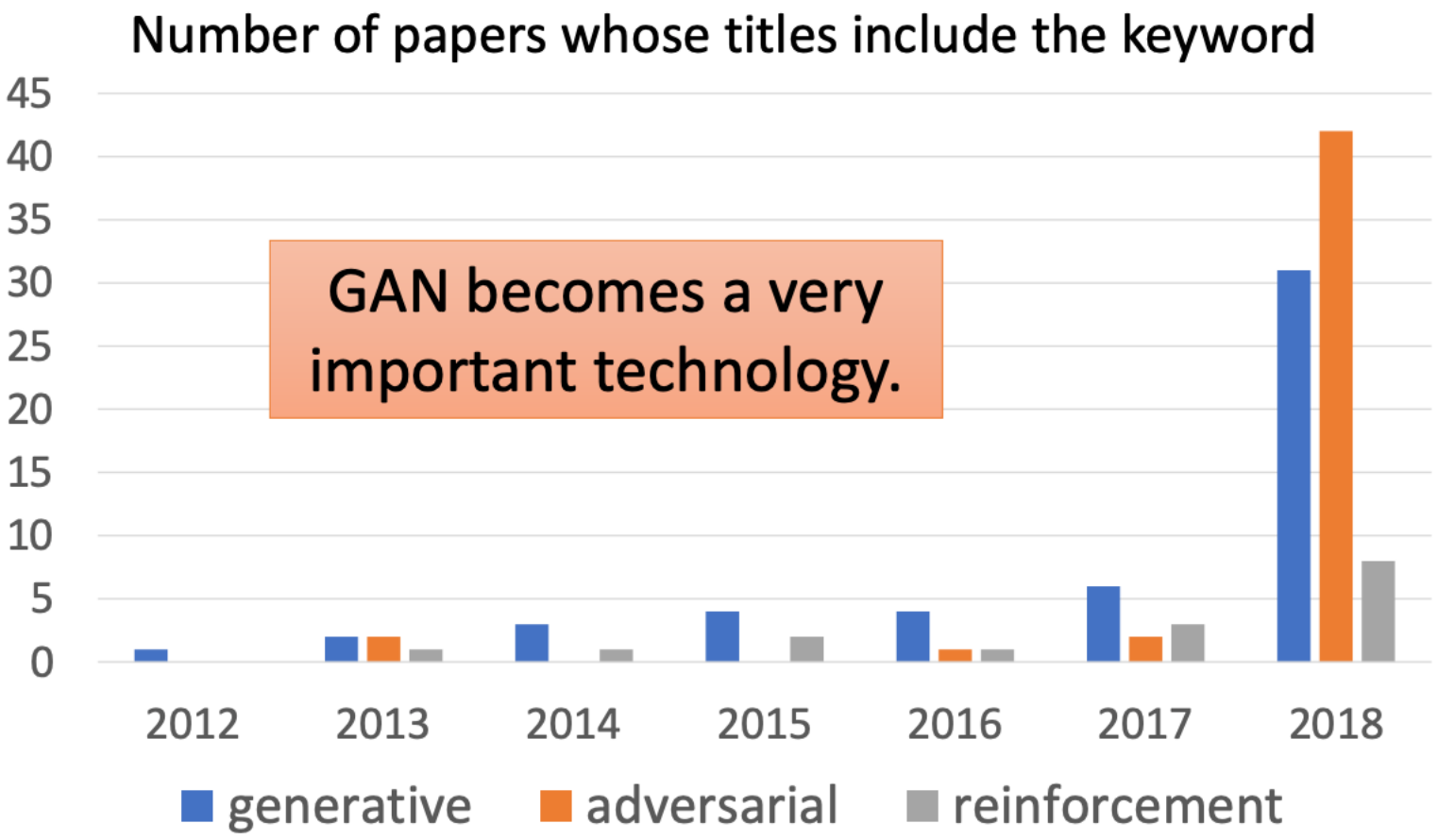
The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal (he since moved to Google Brain and recently to OpenAI).

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

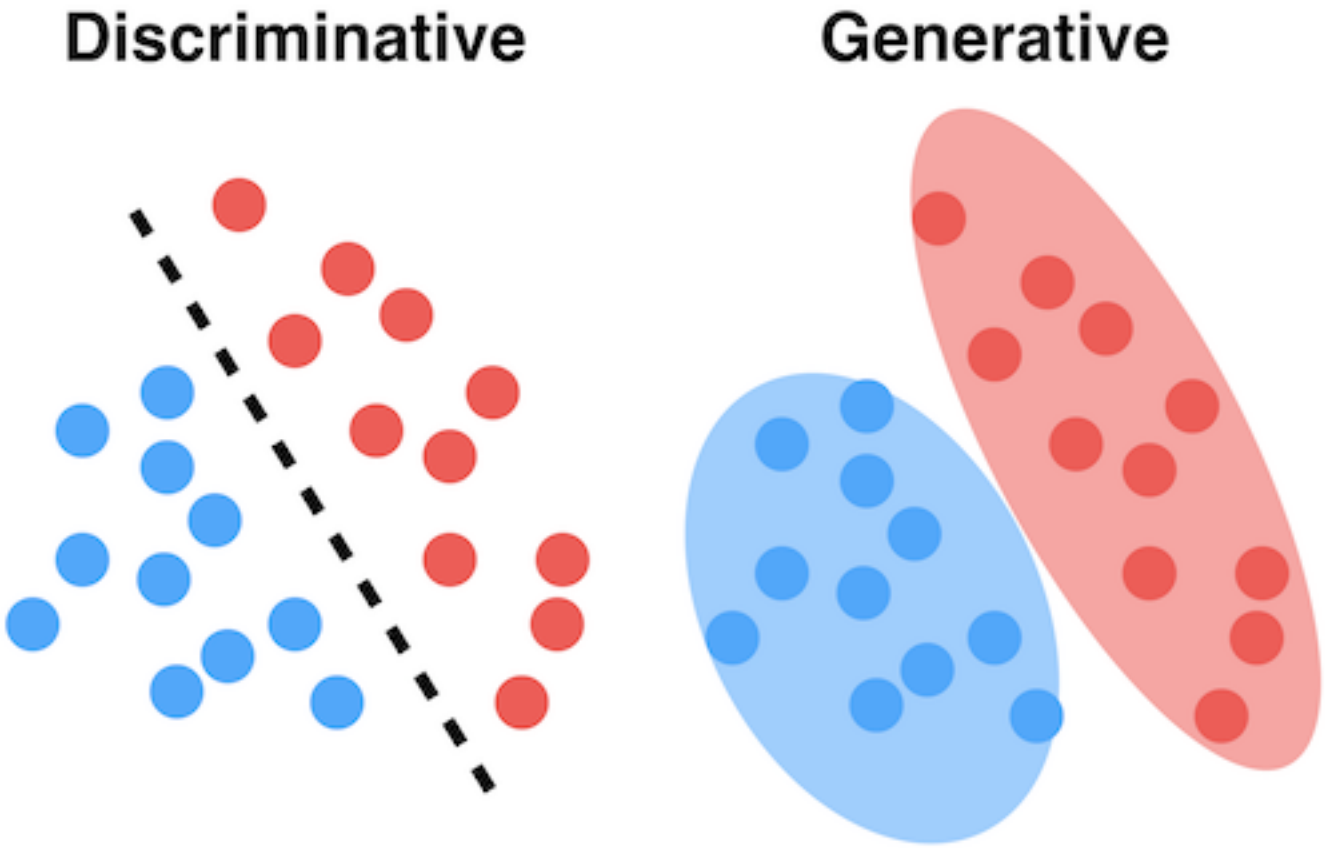
https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-deep-learning

ICASSP

Keyword search on session index page, so session names are included.



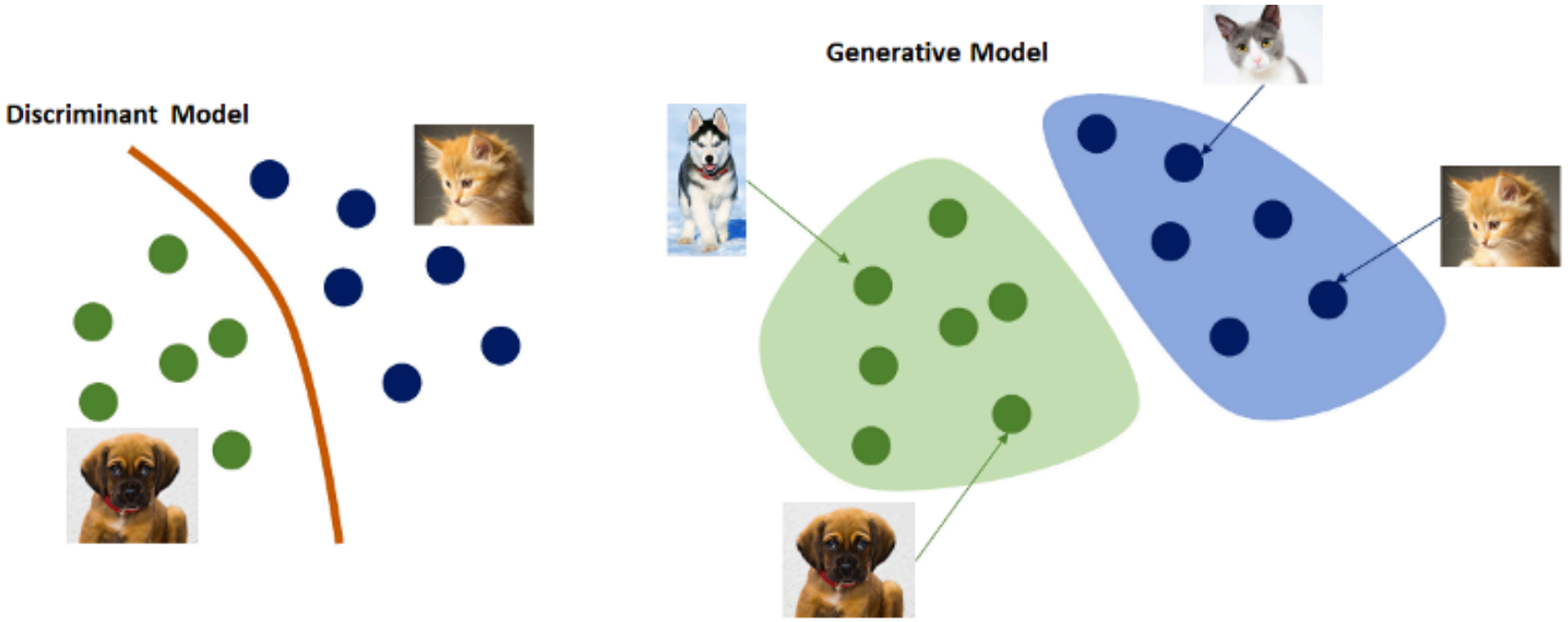
Generative and Discriminative models



Src: https://medium.com/@jordi299/about-generative-and-discriminative-models-d8958b67ad32

- In supervised learning, we have data x and response (label) y and the goal is to learn a function to map x to y e.g. regression, classification, object detection;

- In unsupervised learning, there are no labels and the goal is to find some underlying hidden structure of the data e.g. clustering, dimensionality reduction, feature learning. The goal of generative models is to generate new samples of data from a distribution. These models are used in problems such as density estimation, a problem of unsupervised learning.



Src: https://kharshit.github.io/blog/2018/09/28/generative-models-and-generative-adversarial-networks

Generation

We will control what to generate latter. → Conditional Generation

Image Generation

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

In a specific range

→ NN Generator →

Sentence Generation

$\begin{bmatrix} 0.3 \\ -0.1 \\ \vdots \\ -0.7 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.1 \\ \vdots \\ 0.2 \end{bmatrix} \begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.5 \end{bmatrix}$

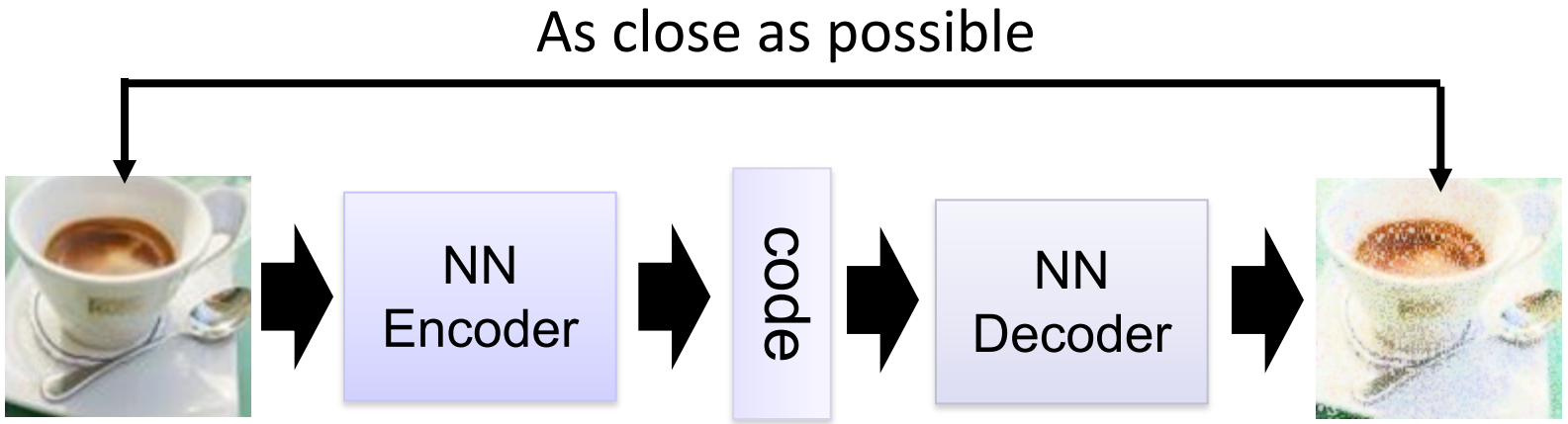
→ NN Generator →

How are you?

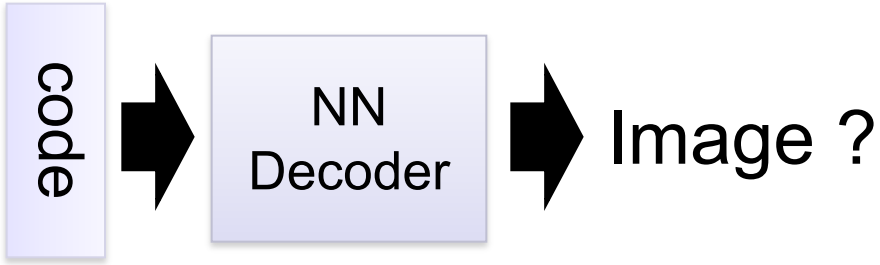
Good morning.

Good afternoon.

Autoencoder



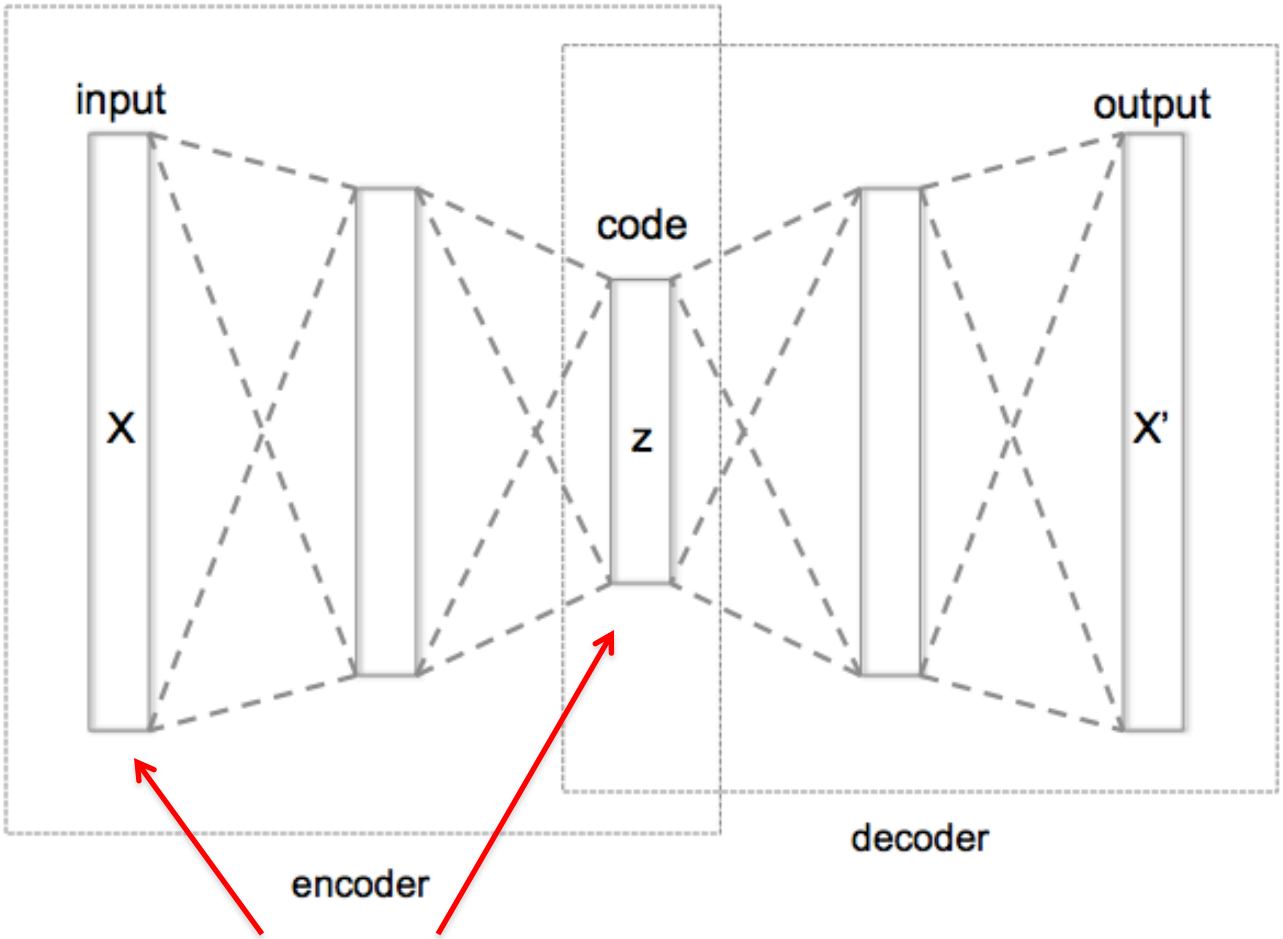
Randomly generate a vector as code



Src: https://cs.uwaterloo.ca/~mli/Deep-Learning-2017-Lecture7GAN.ppt

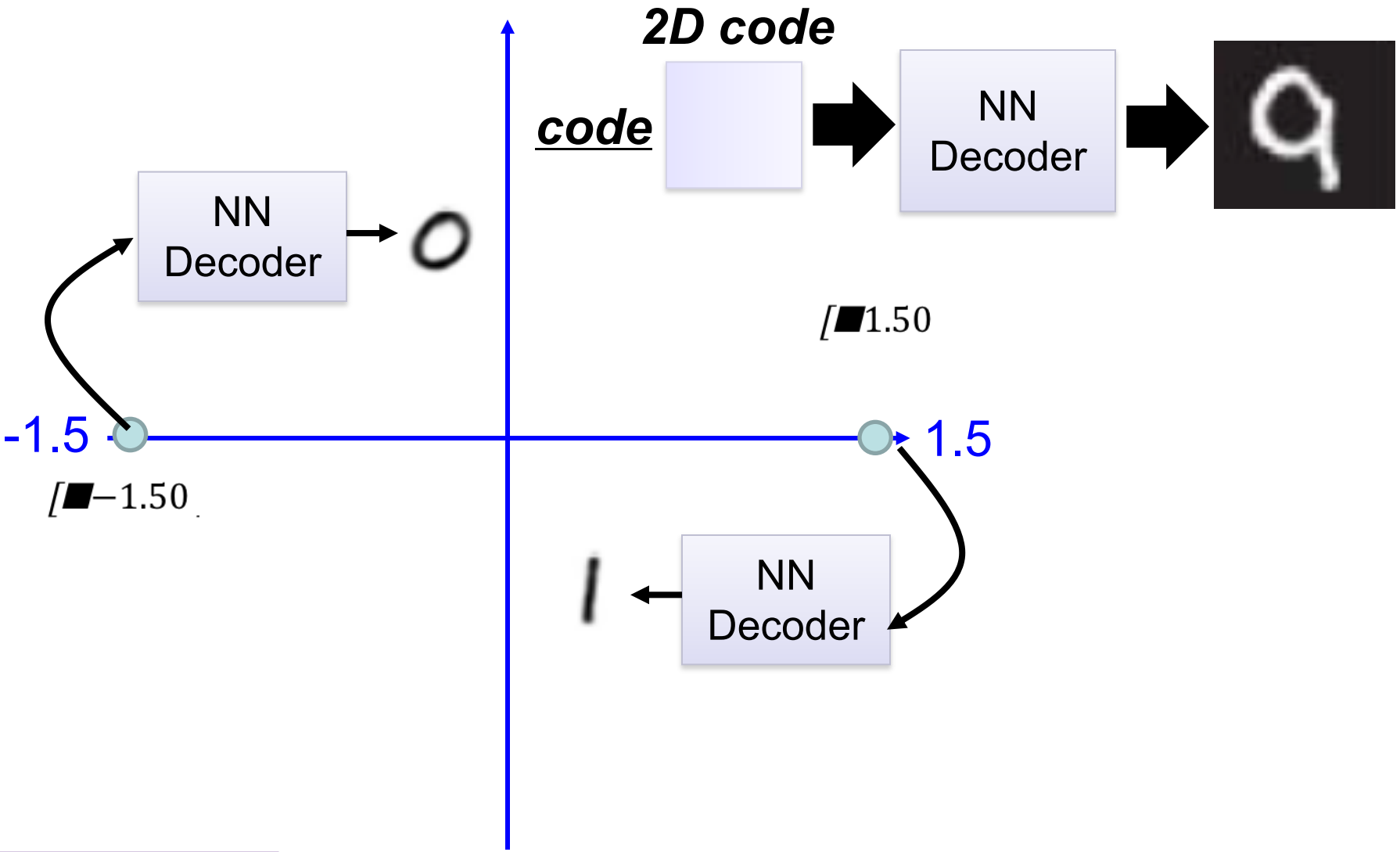
Autoencoder with 3 fully connected layers

Training: model.fit(X,X)
Cost function: $\Sigma_{k=1..N} (x_k - x'_k)^2$

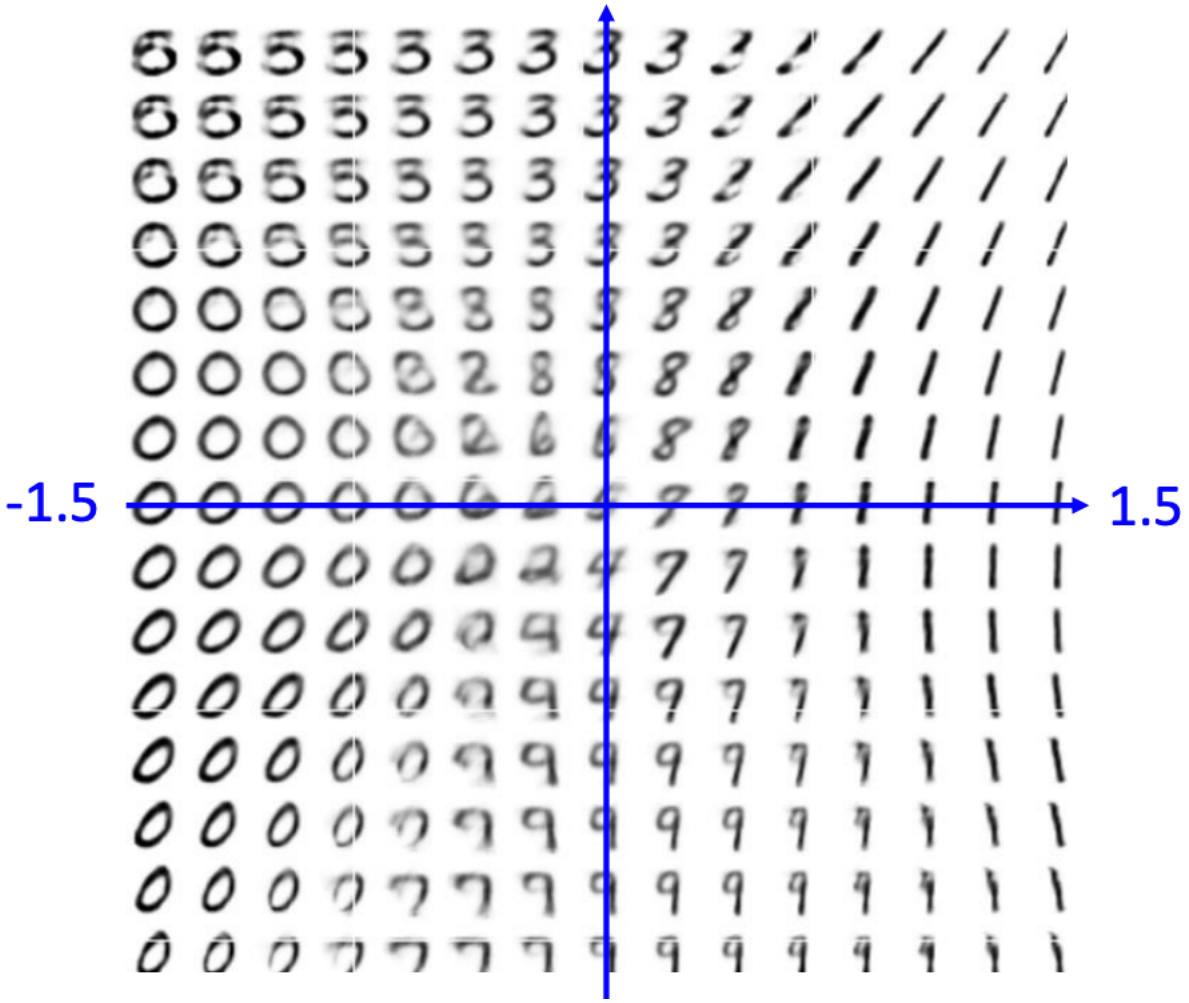


Large → small, learn to compress

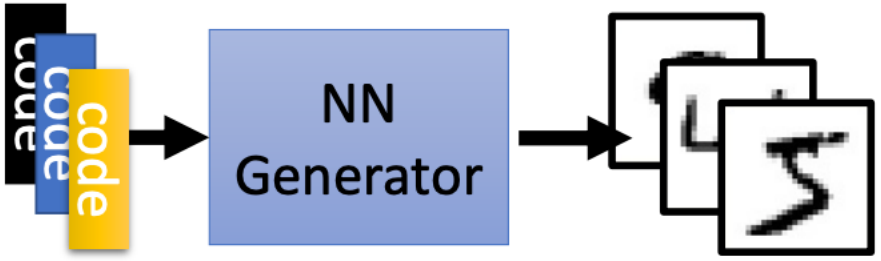
Auto-encoder



Auto-encoder



Auto-encoder

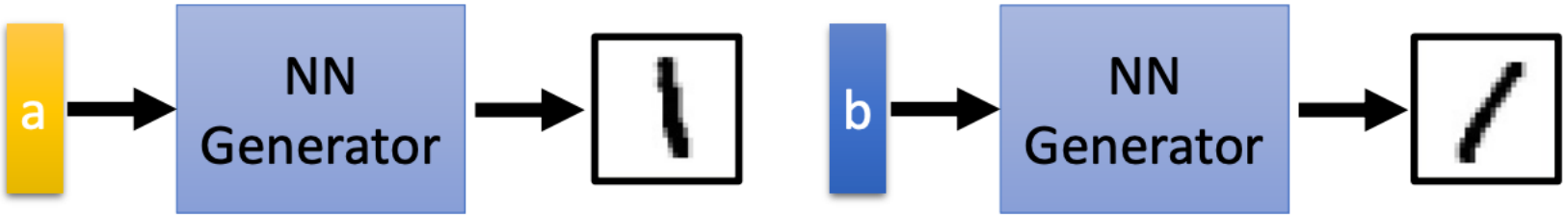


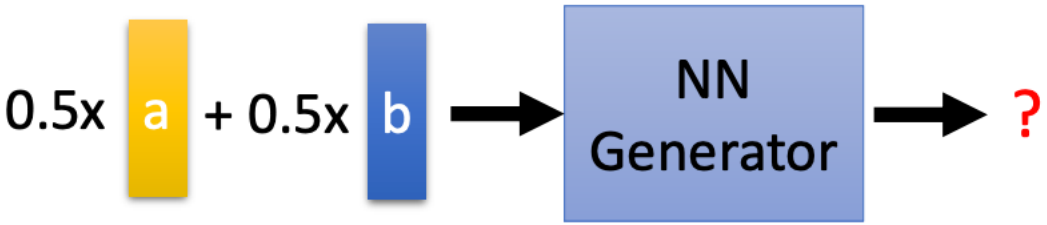
vectors

code:
(where does them come from?)

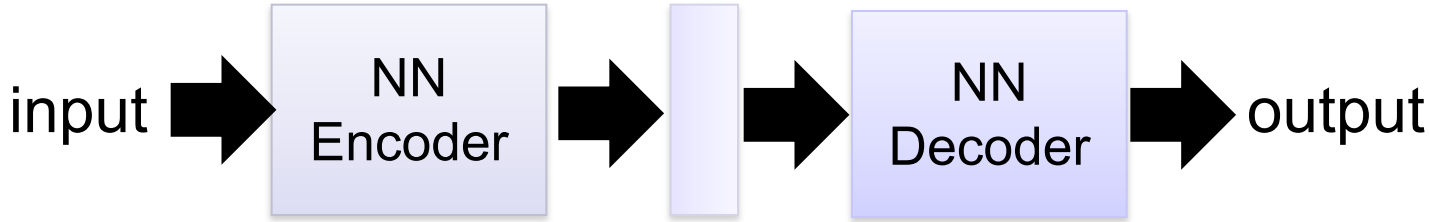
$\begin{bmatrix} 0.1 \\ -0.5 \end{bmatrix}$ $\begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$ $\begin{bmatrix} 0.2 \\ -0.1 \end{bmatrix}$ $\begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix}$

Image:





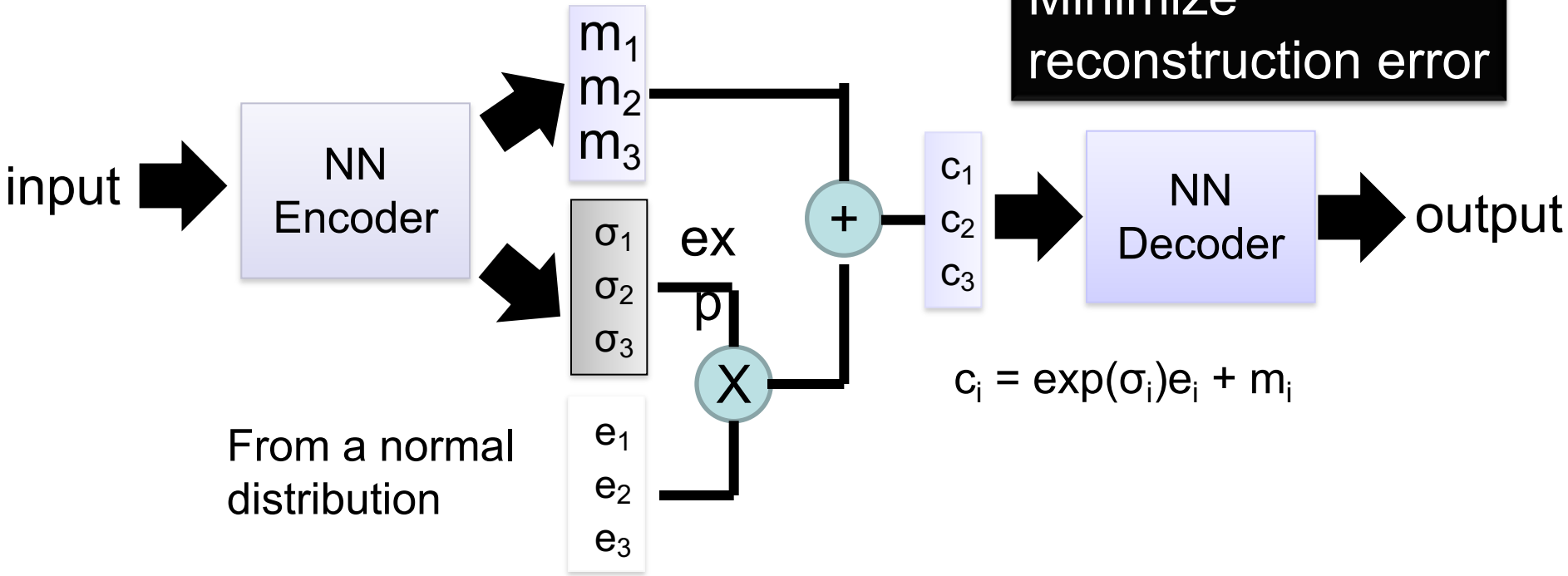
Auto-encoder



code

VAE

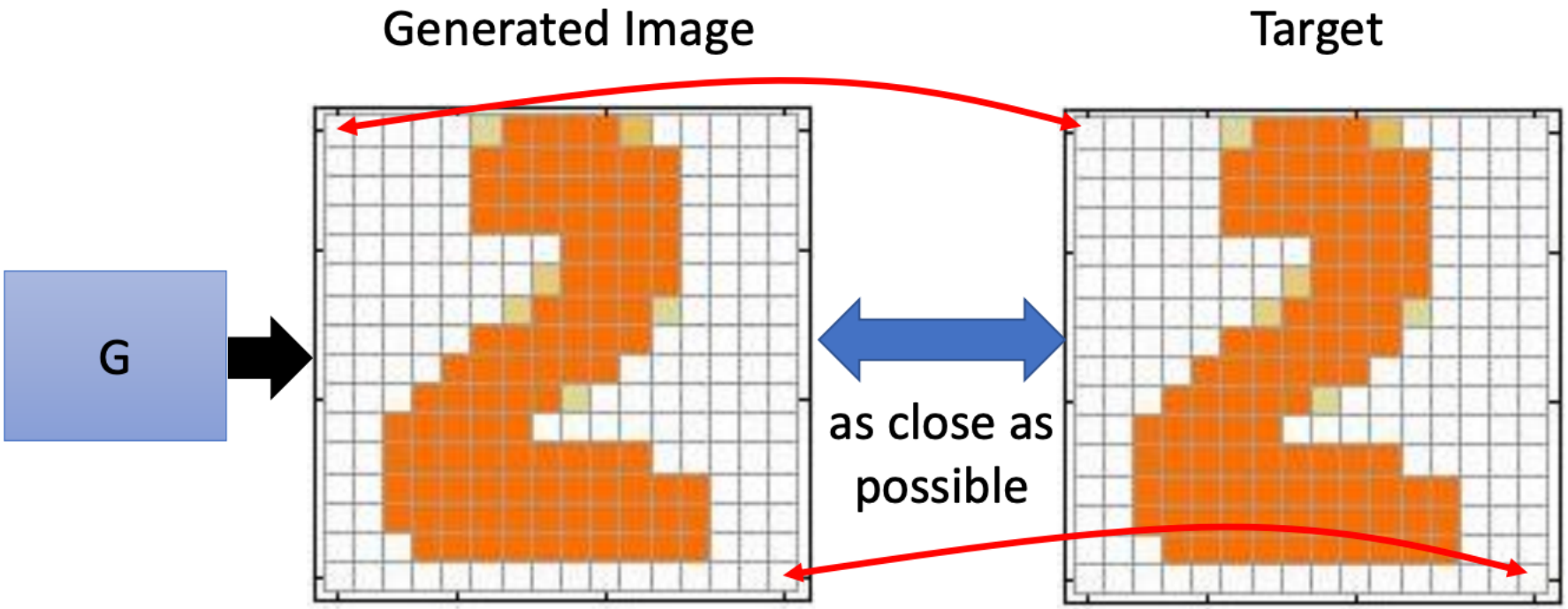
Minimize reconstruction error



$c_i = \exp(\sigma_i)e_i + m_i$

Auto-Encoding Variational Bayes, https://arxiv.org/abs/1312.6114

What do we miss?



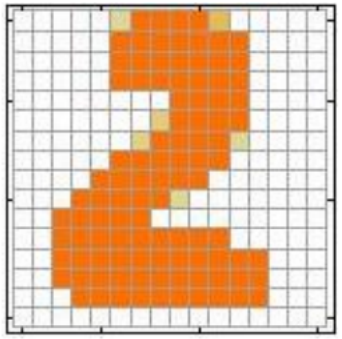
It will be fine if the generator can truly copy the target image.

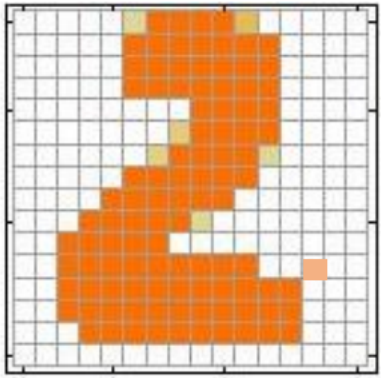
What if the generator makes some mistakes …….

Some mistakes are serious, while some are fine.

What do we miss?

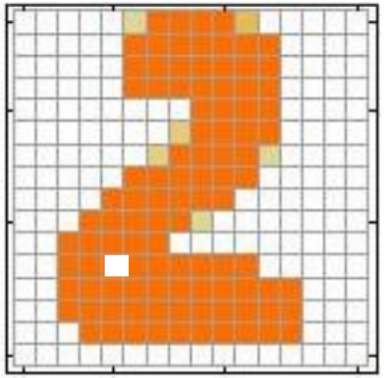
Target





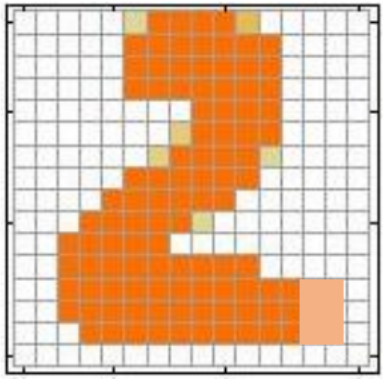
1 pixel error

I think it is bad



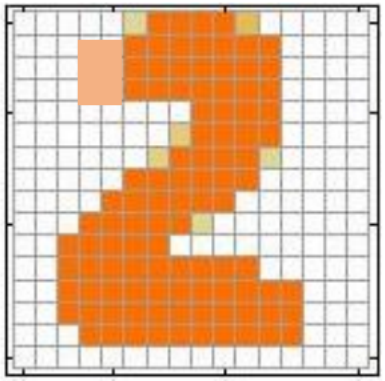
1 pixel error

I think it is bad



6 pixel errors

I think it is good

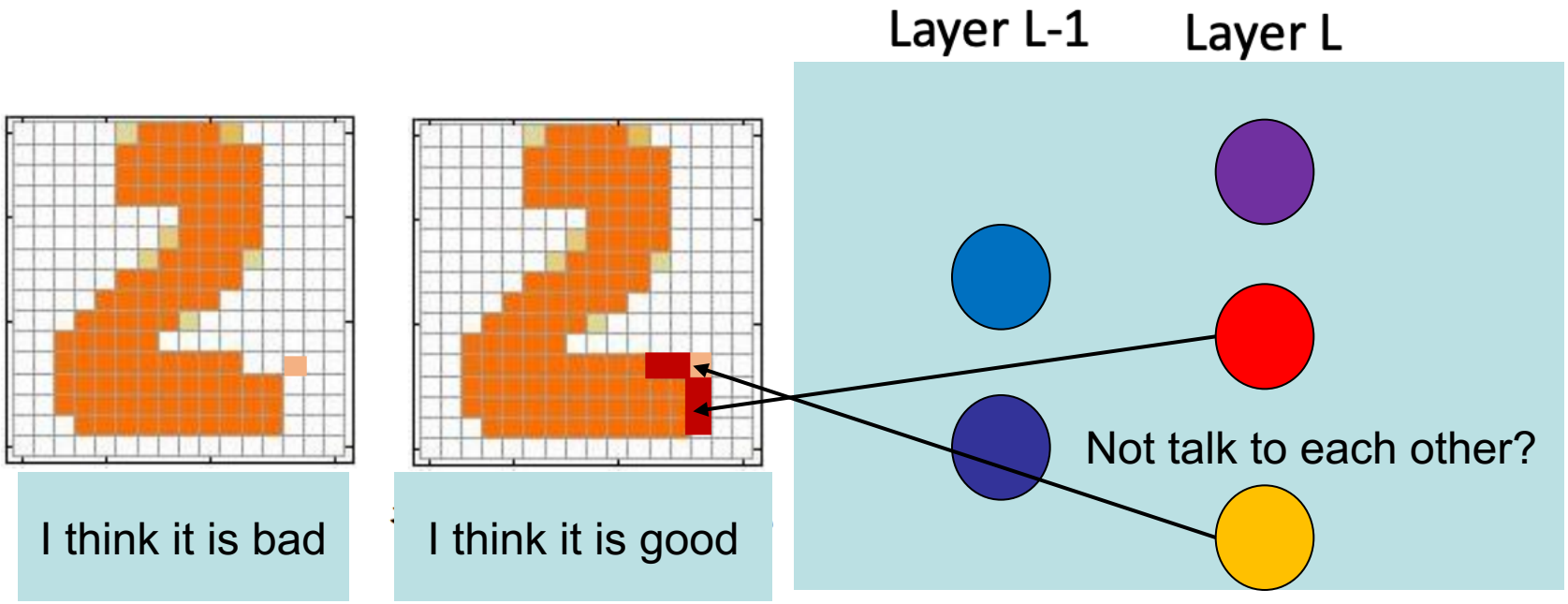


6 pixel errors

I think it is good

What do we miss?

Each neural in output layer corresponds to a pixel.

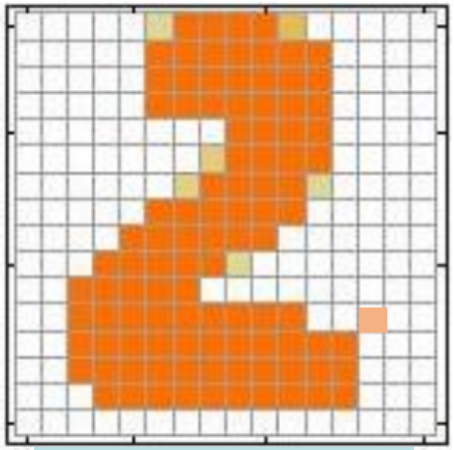


The relation between the components are critical.

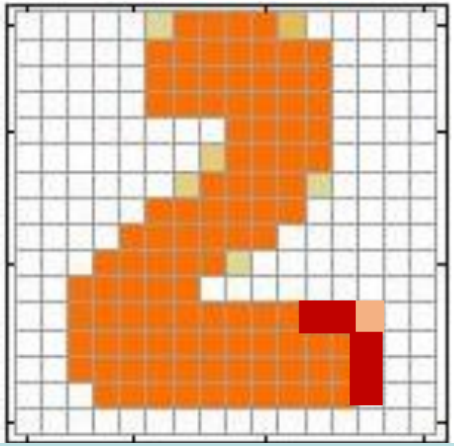
Although highly correlated, they cannot influence each other.

Need deep structure to catch the relation between components.

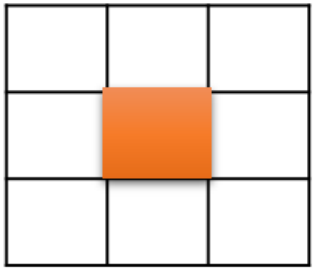
Better Loss Design



I think it is bad



I think it is good



This CNN filter is good enough.

Discriminator

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

Inference

- Generate object $\tilde{x}$ that

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

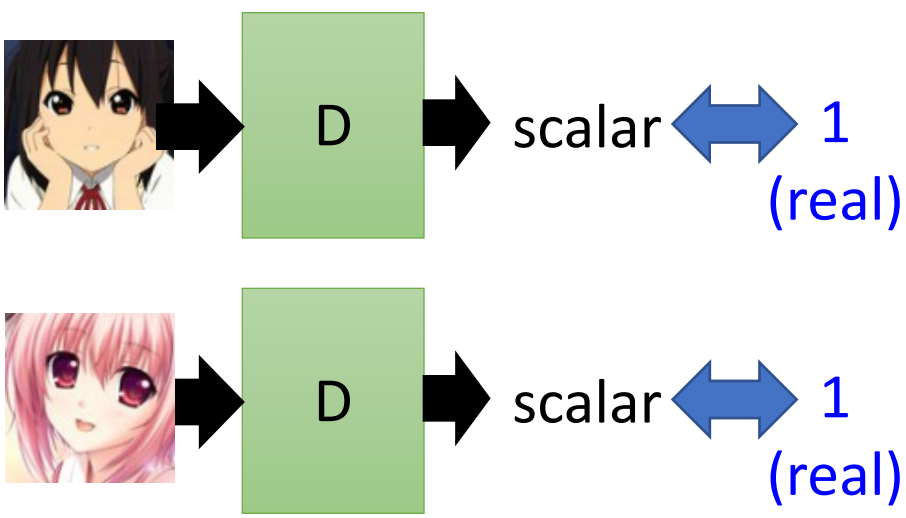
Enumerate all possible x !!!

It is feasible ???

How to learn the discriminator?

Discriminator - Training

- I have some real images

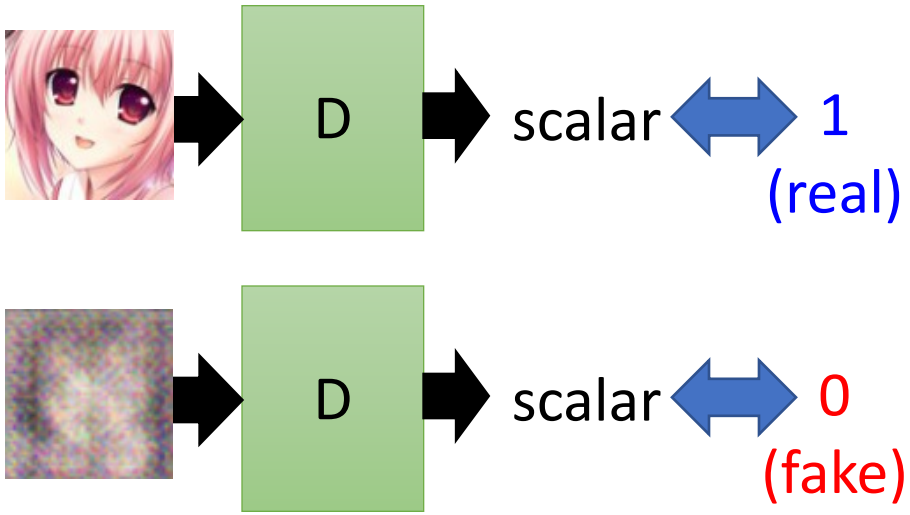


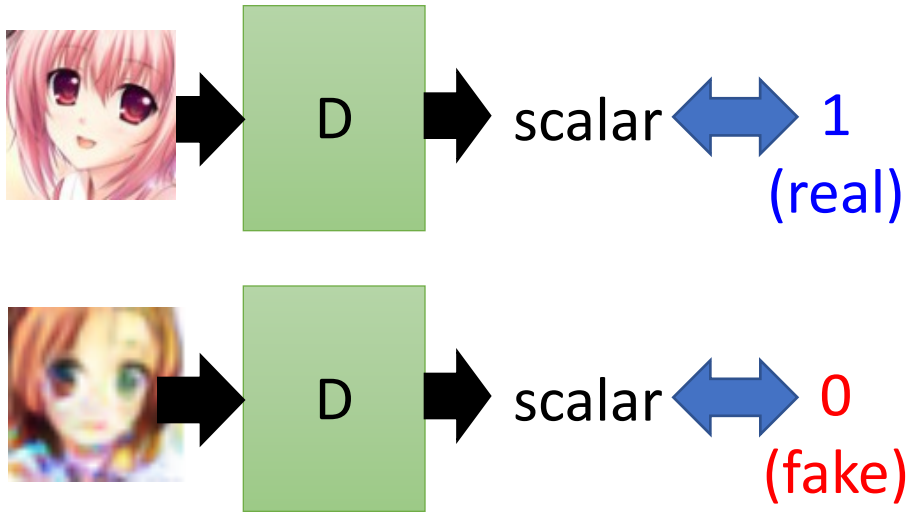
Discriminator only learns to output “1” (real).

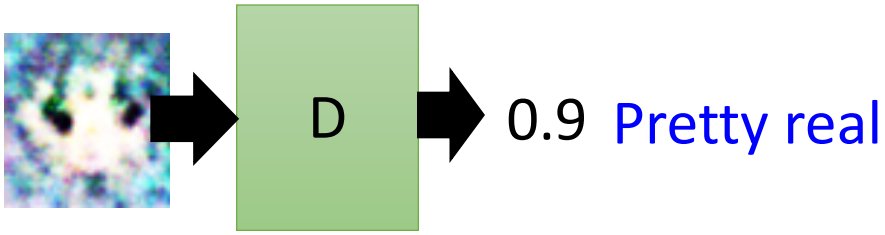
Discriminator training needs some negative examples.

Discriminator - Training

- Negative examples are critical.





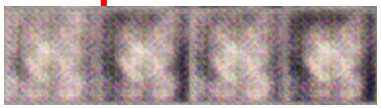


How to generate realistic negative examples?

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.

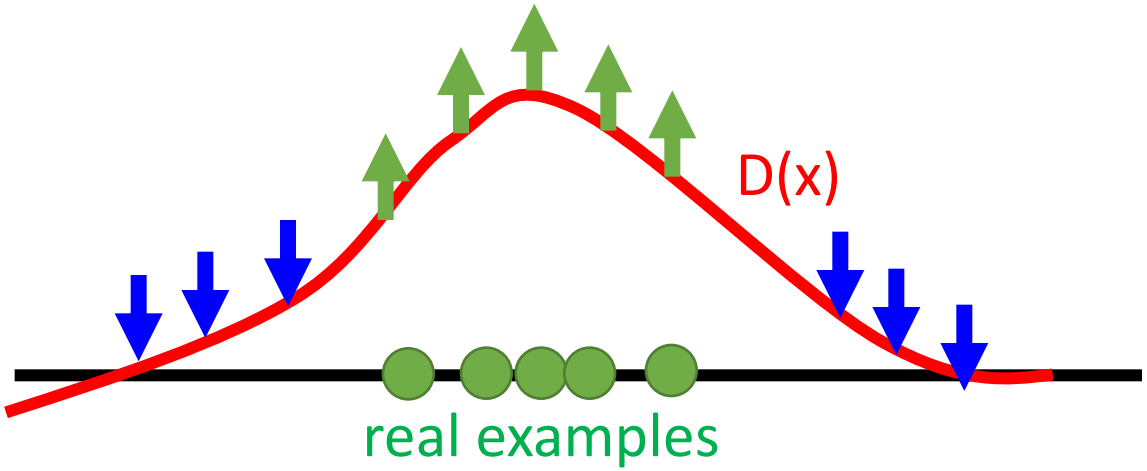

v.s.

D

    - Generate negative examples by discriminator D

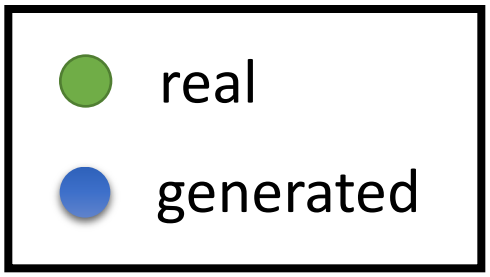
$$\tilde{x} = \arg\max_{x \in X} D(x)$$

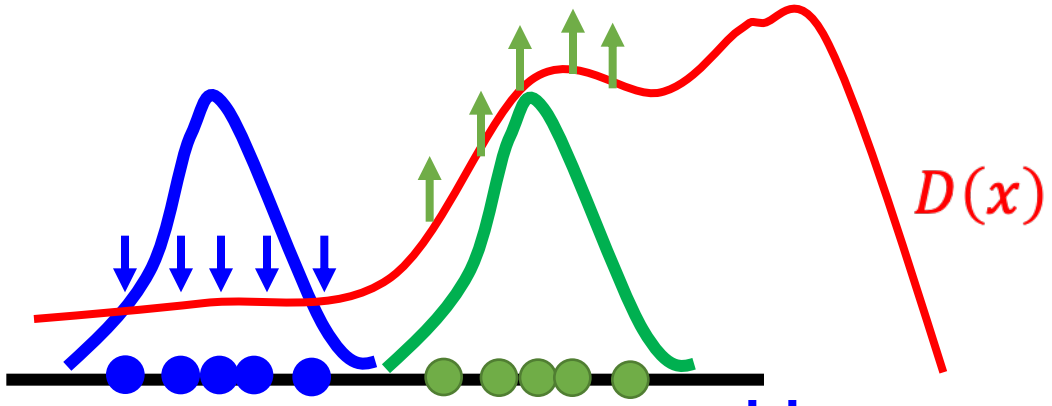
Discriminator - Training

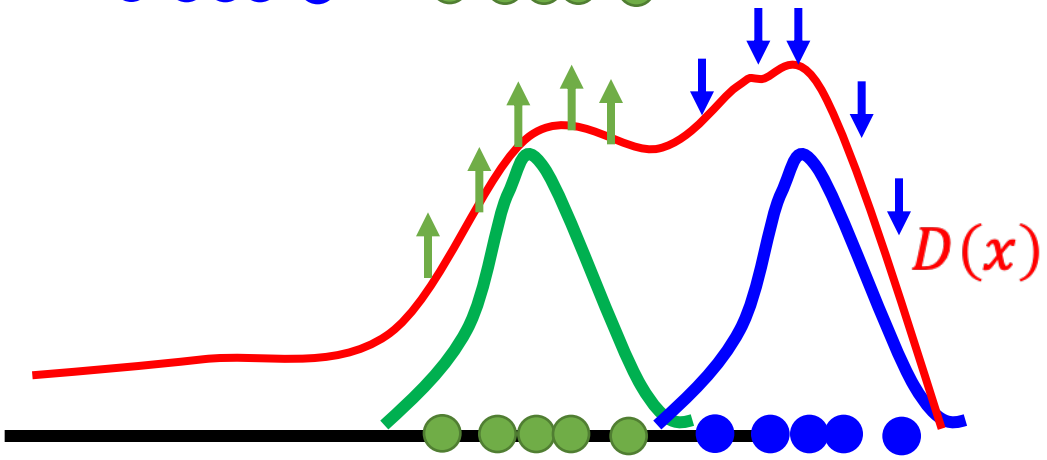


In practice, you cannot decrease all the x other than real examples.

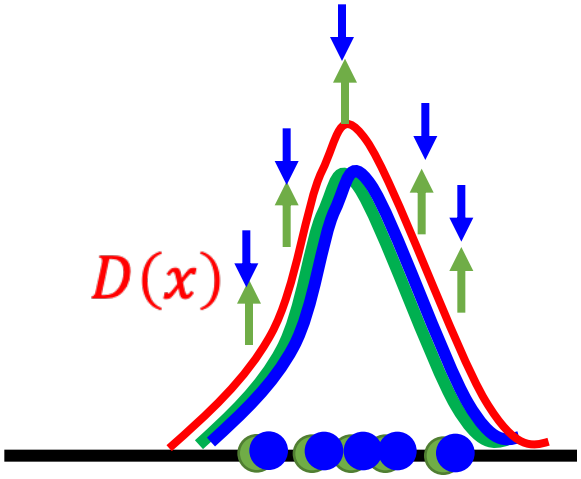
Discriminator - Training

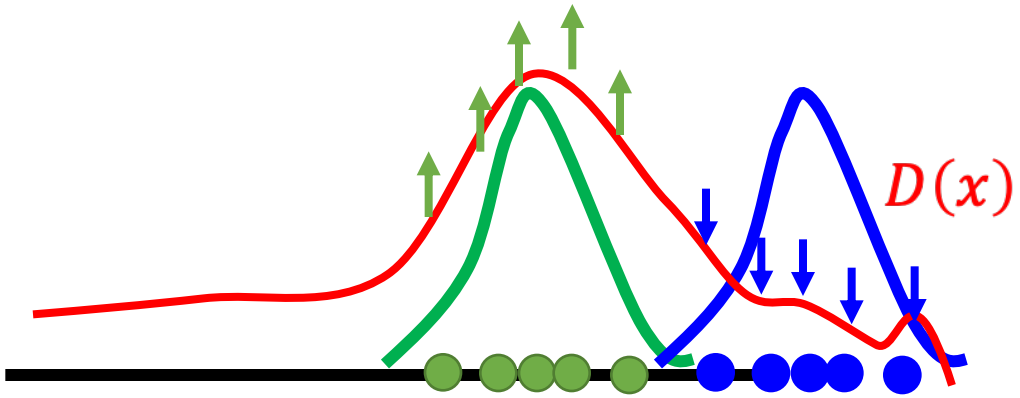






In the end ……





Re-visit 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.

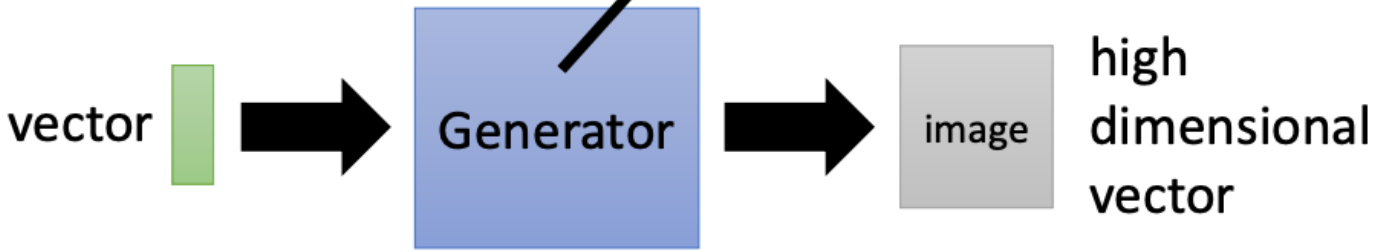
    - Generate negative examples by discriminator D

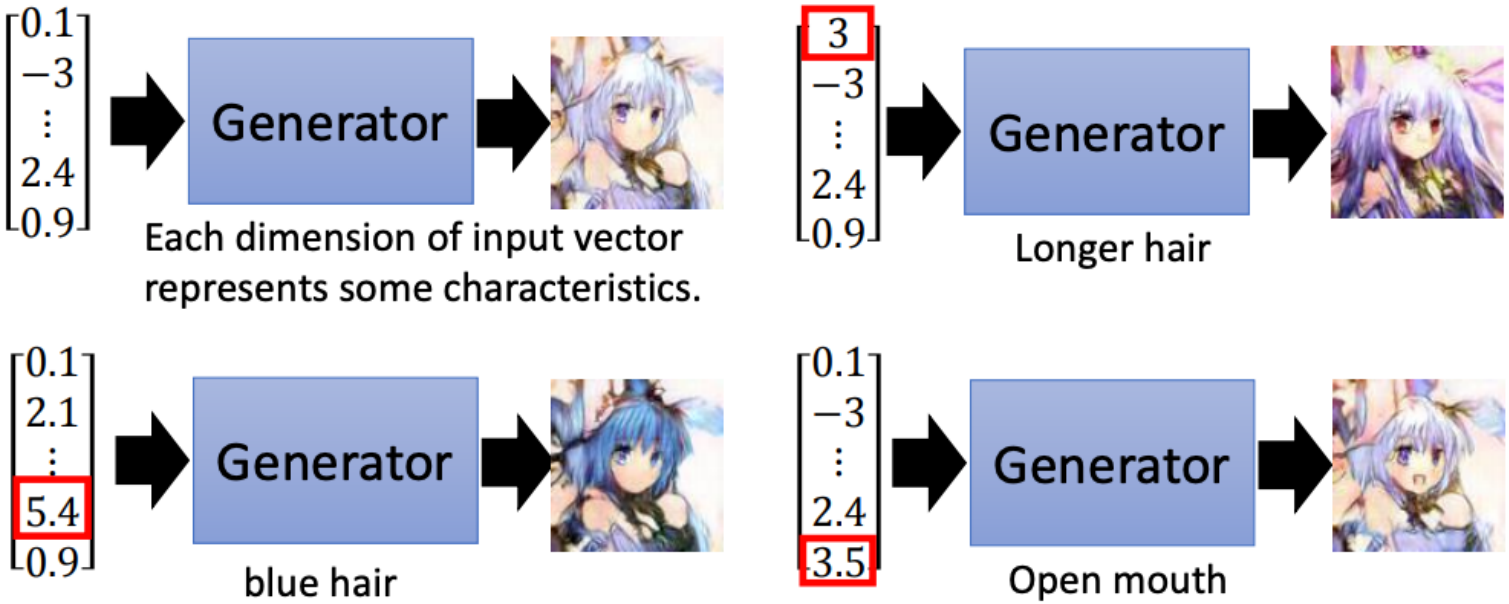
Replace with Deep Neural Network

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

Basic Idea of GAN

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





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

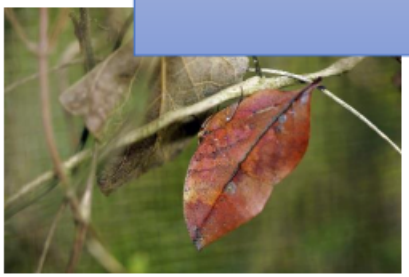
 → Discri-minator → 0.1

 → Discri-minator → 0.1

Basic Idea of GAN

Brown

Generator

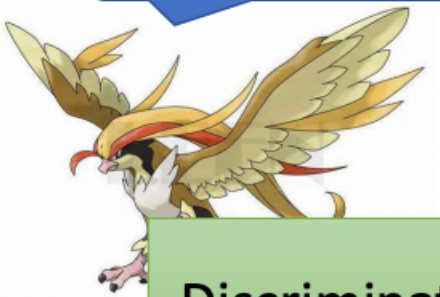


veins

Butterflies are not brown

Butterflies do not have veins

.....

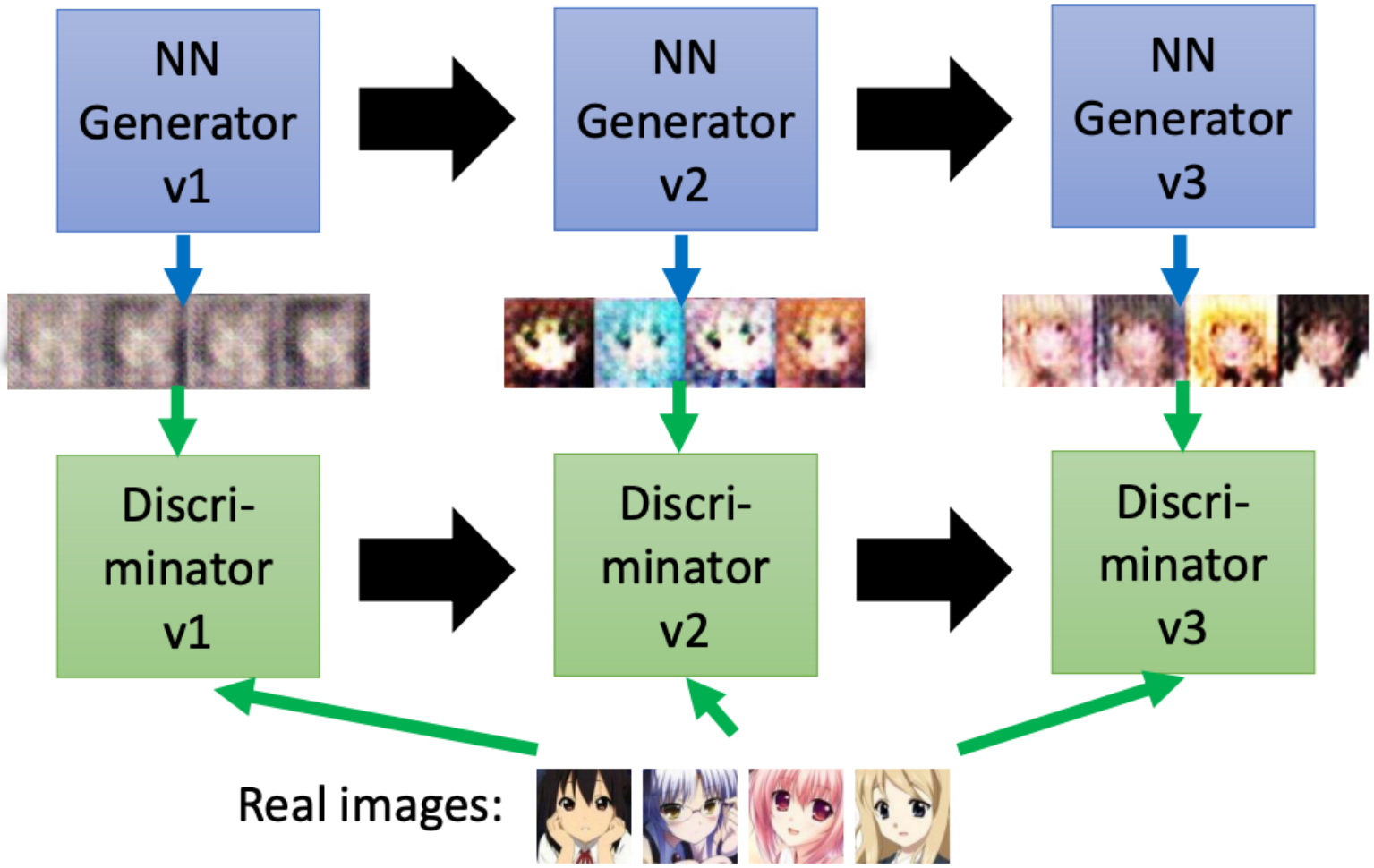


Discriminator

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)

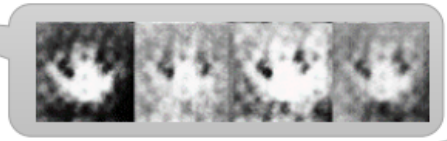
Discriminator (teacher)

Generator v1

Discriminator v1

No two circles?

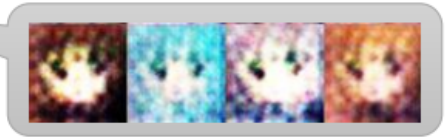
Generator v2



Discriminator v2

No colors?

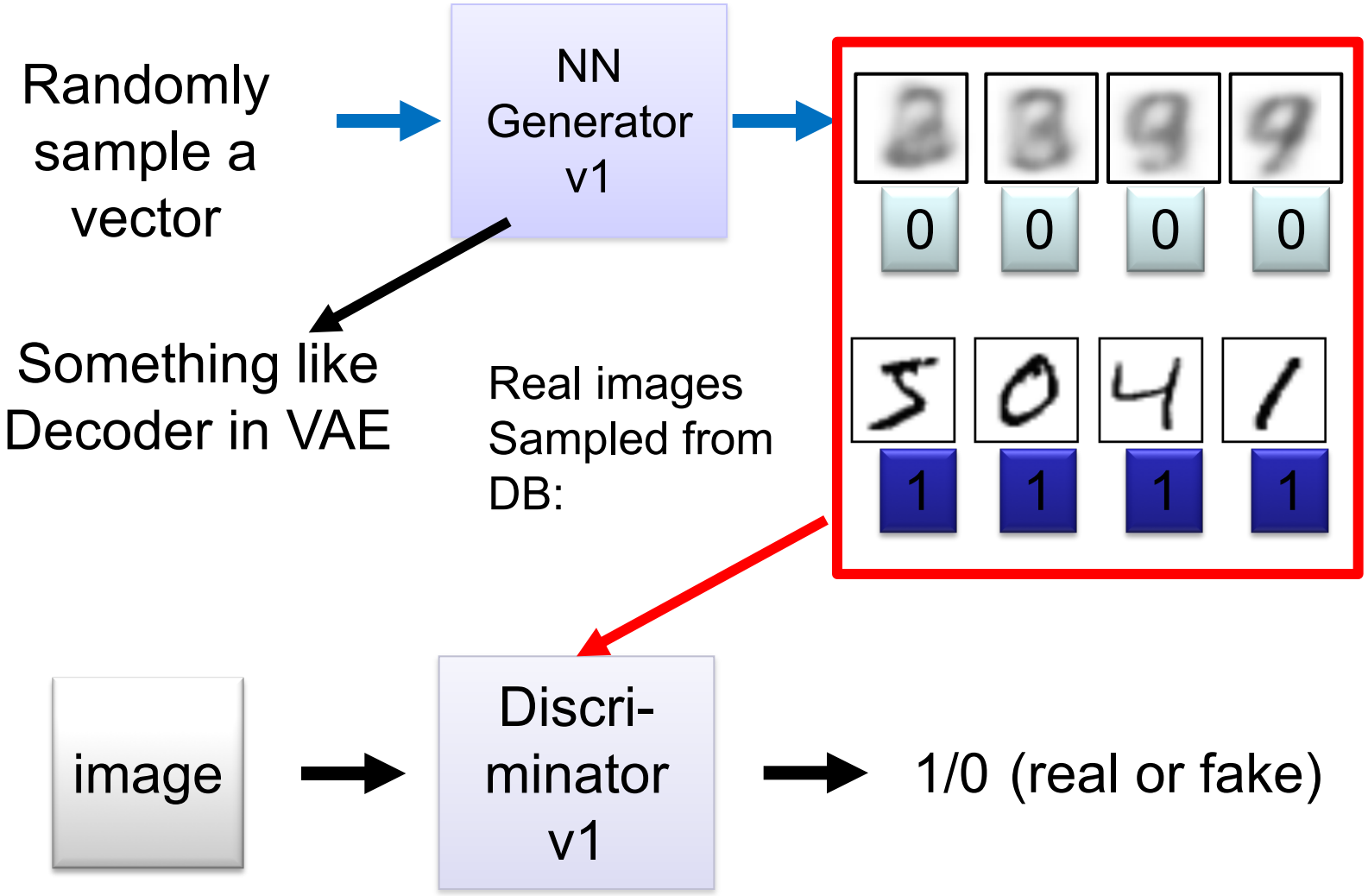
Generator v3



Why NOT learn by myself?

Why Not do it by yourself?

GAN – Learn a discriminator



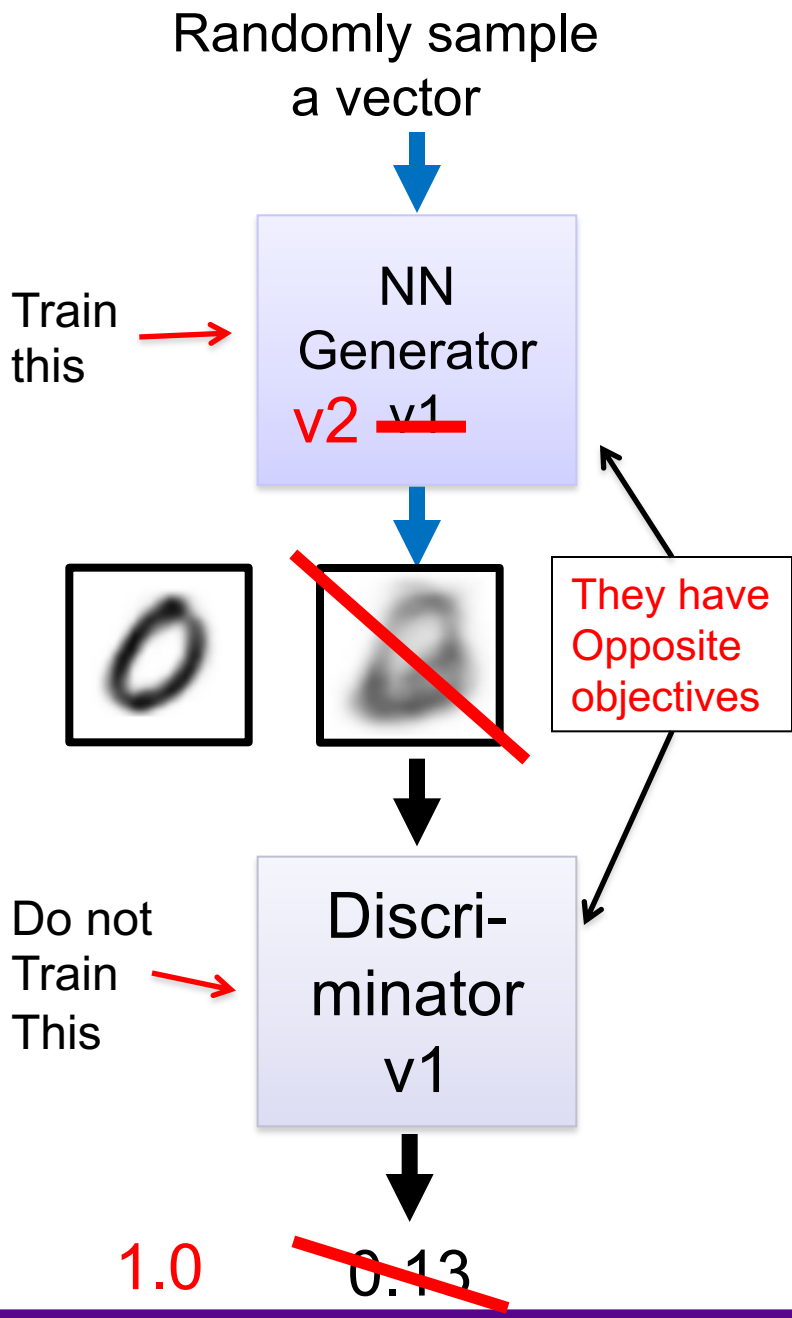
GAN – Learn a generator

Updating the parameters of generator

The output be classified as "real" (as close to 1 as possible)

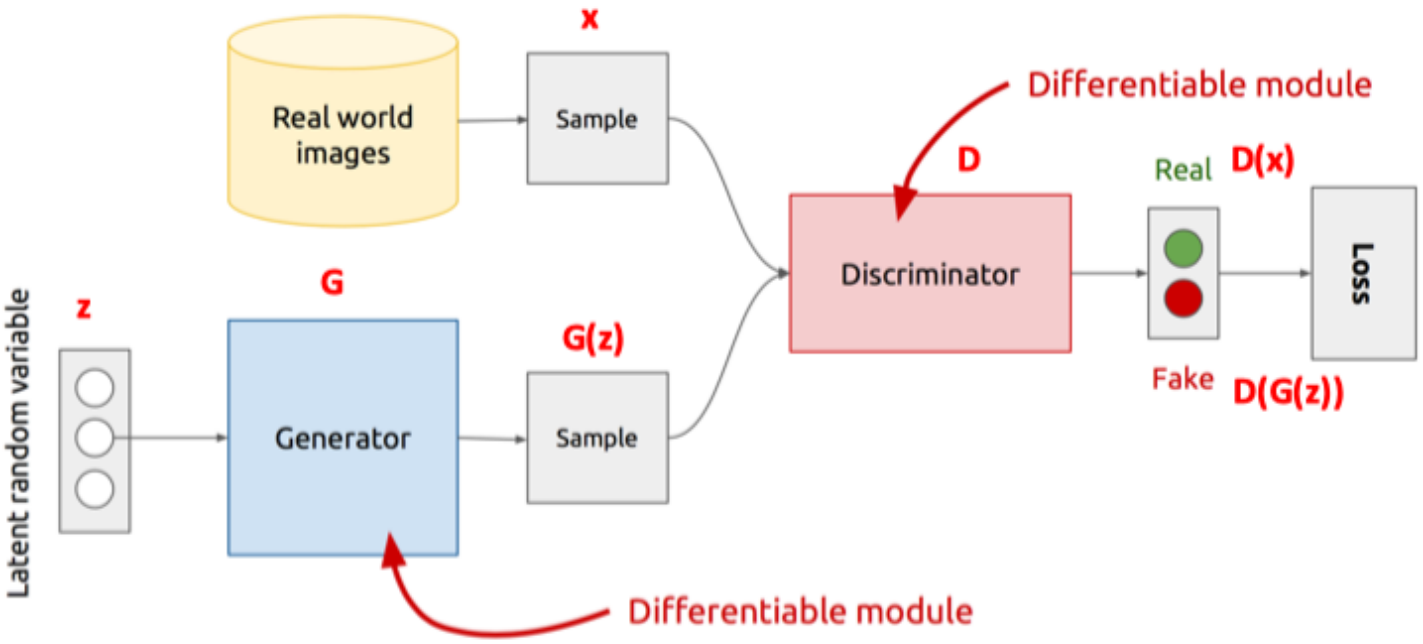
Generator + Discriminator = a network

Using gradient descent to update the parameters in the generator, but fix the discriminator



Generative Adversarial Network

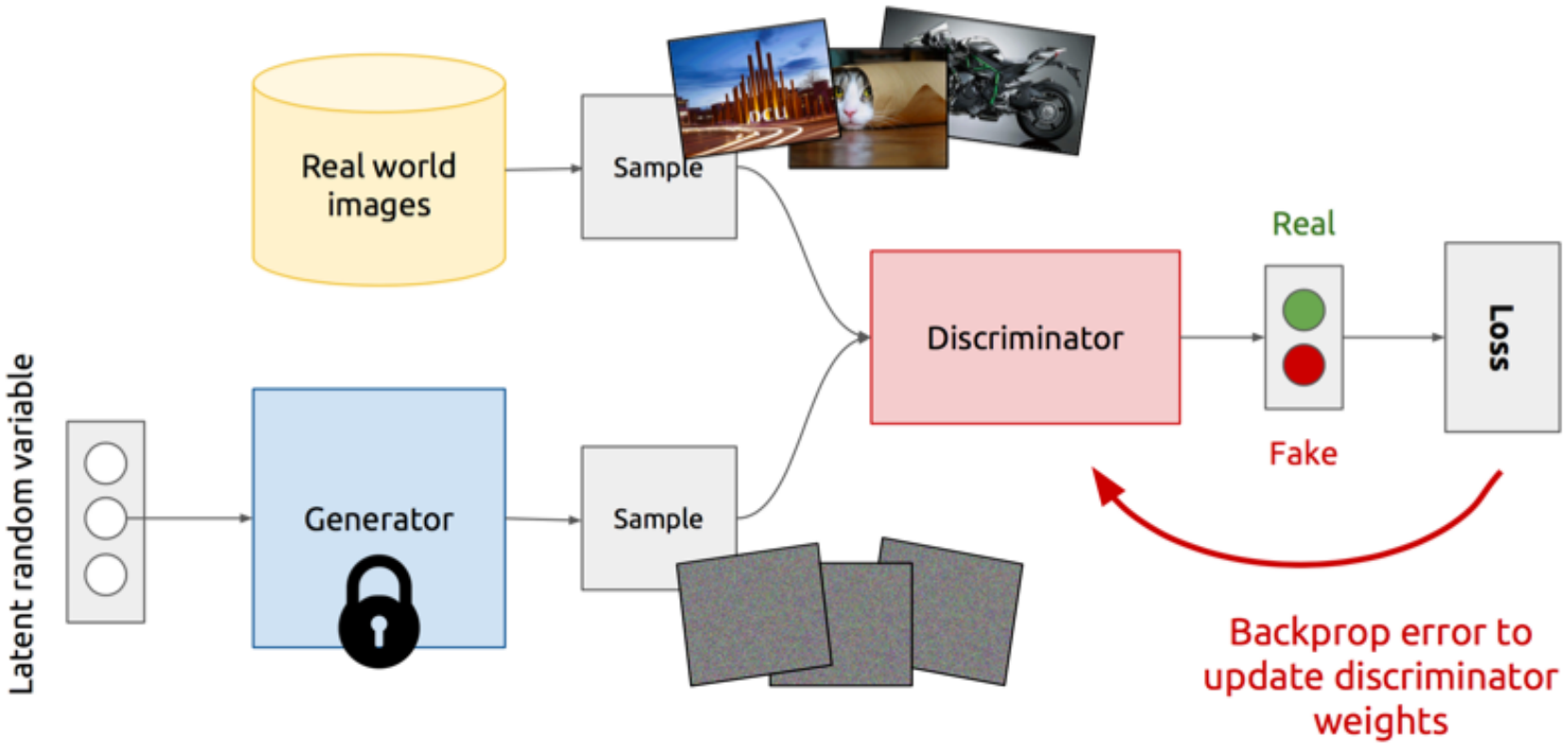
GAN's Architecture



- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

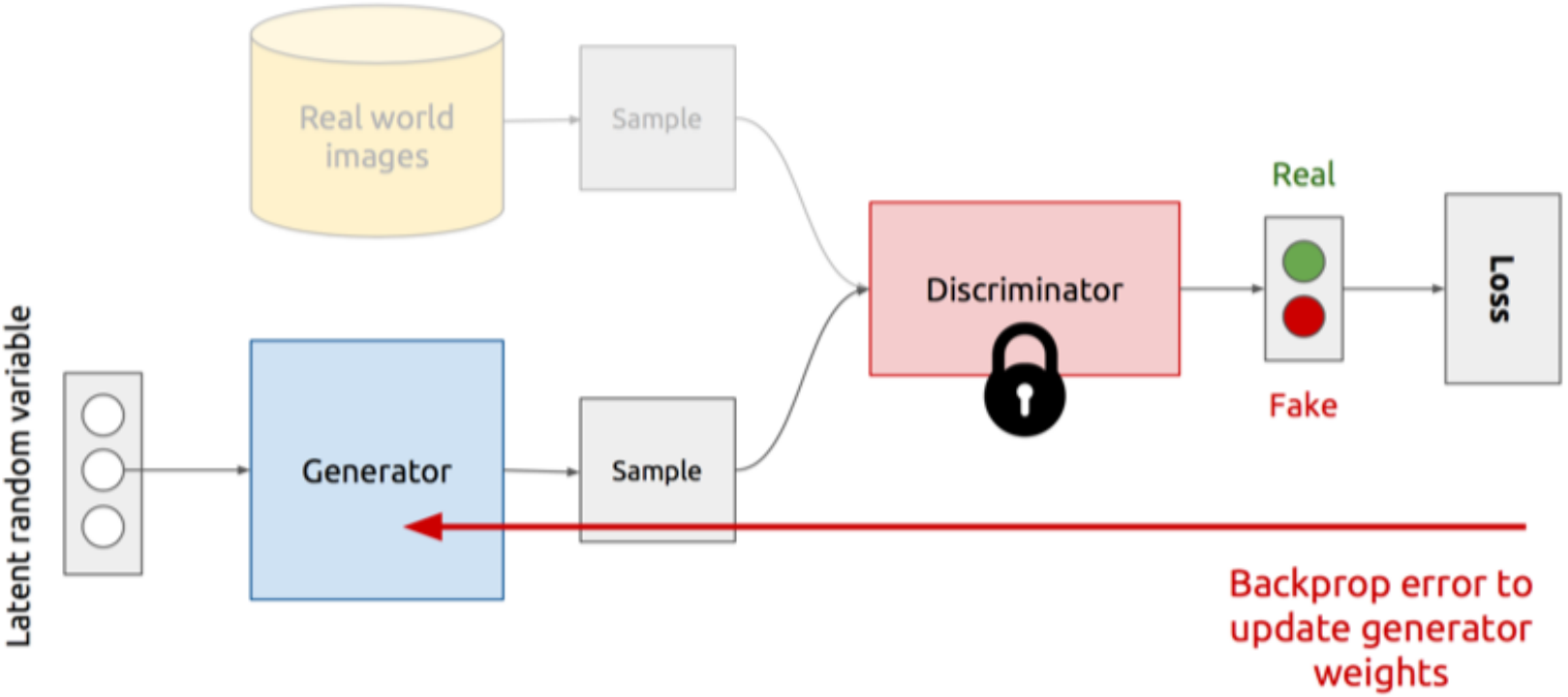
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Discriminator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Generator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN Algorithms

- Generator G
  - G is a function, input z, output x
  - Given a prior distribution $P_{prior}(z)$, a probability distribution $P_G(x)$ is defined by function G
- Discriminator D
  - D is a function, input x, output scalar
  - Evaluate the "difference" between $P_G(x)$ and $P_{data}(x)$
- In order for D to find difference between P_{data} from P_G, we need a cost function V(G,D):

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

Note, we are changing distribution G, not just update its parameters (as in the max likelihood case).

Hard to learn P_G by maximum likelihood

GAN's formulation

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

- It is formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize its reward $\boldsymbol{V(D,G)}$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}[\log(1 - D(G(z)))]$$

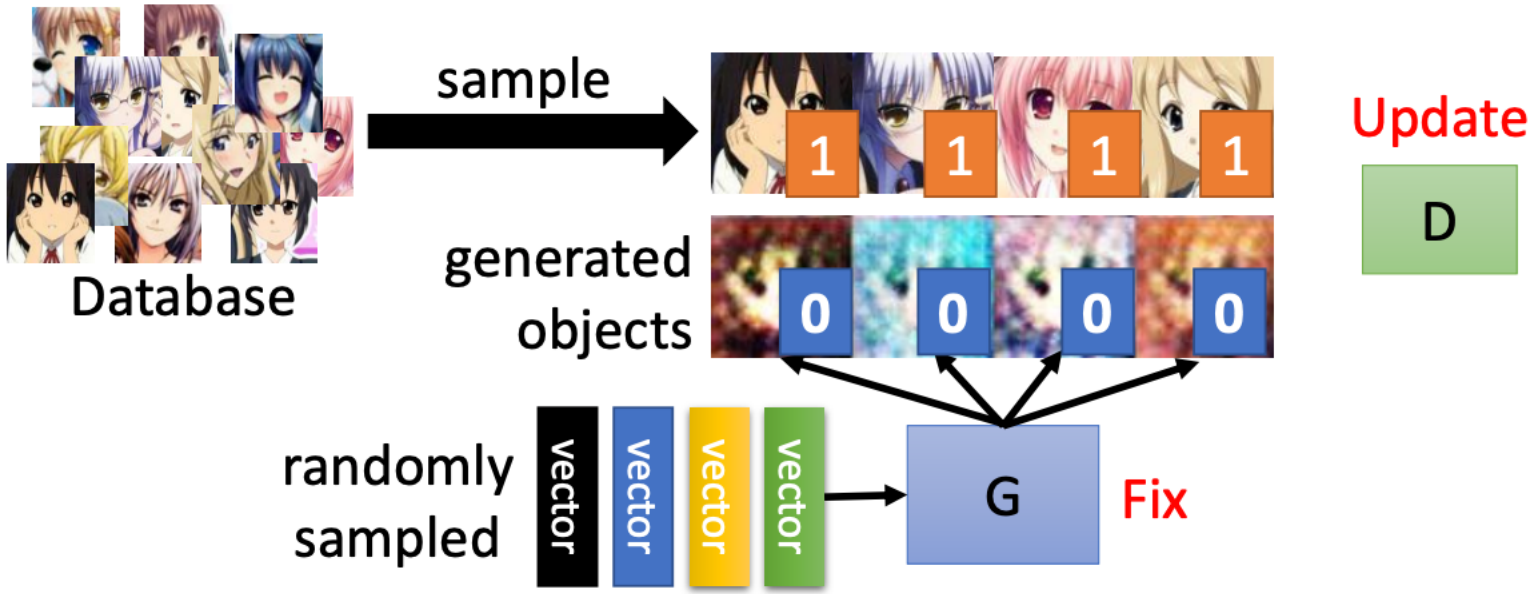
- The Nash equilibrium of this particular game is achieved at:
  - $P_{data}(x) = P_{gen}(x) \ \forall x$
  - $\mathrm{D}(x) = \frac{1}{2} \ \forall x$

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.

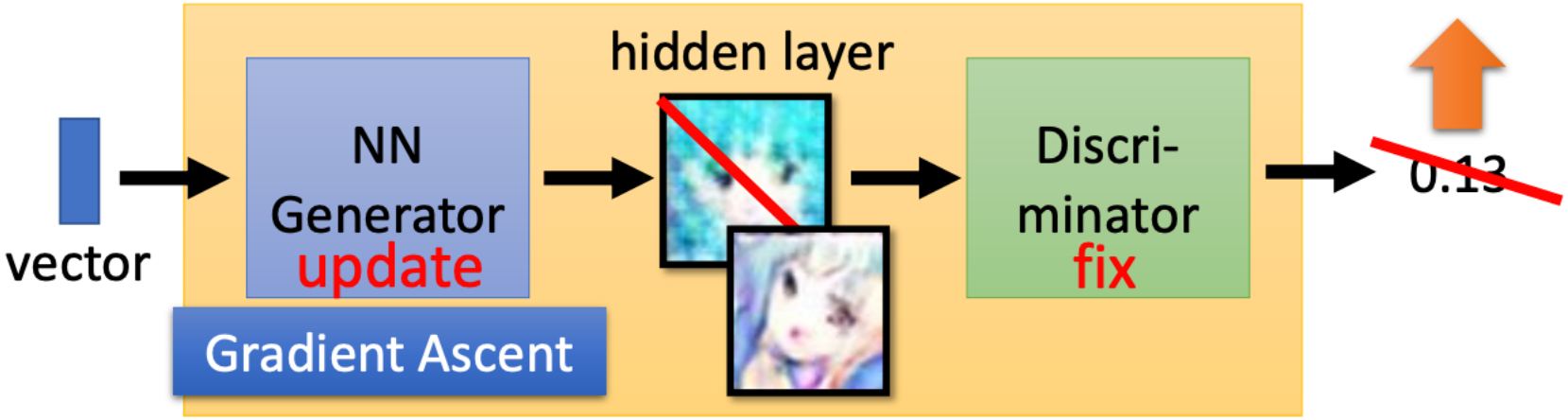
Algorithm

- Initialize generator and discriminator

- In each training iteration:

Step 2: Fix discriminator D, and update generator G

Generator learns to "fool" the discriminator



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

Learning G

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

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.

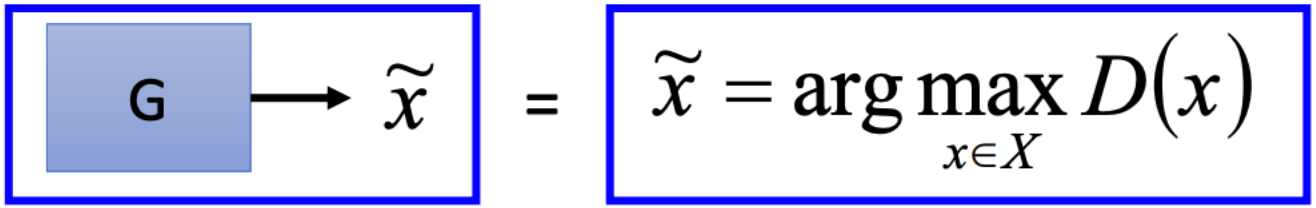
D

    - 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



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

https://arxiv.org/abs/1512.09300

References

- http://slazebni.cs.illinois.edu/spring17/
- https://cs.uwaterloo.ca/~mli/Deep-Learning-2017-Lecture7GAN.ppt