Generative Adversarial Network (GAN)

Outlook:

Restricted Boltzmann Machine:
http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/RBM%20(v2).ecm.mp4/index.html

Gibbs Sampling:
http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/MRF%20(v2).ecm.mp4/index.html

NIPS 2016 Tutorial: Generative Adversarial Networks

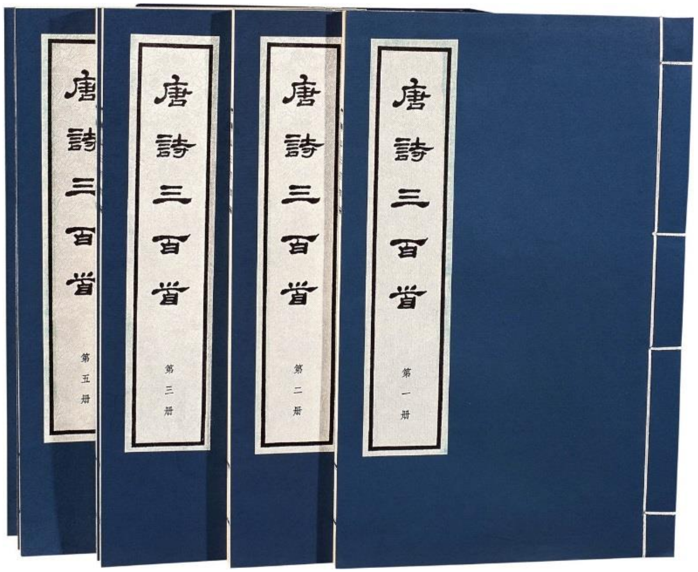
- Author: Ian Goodfellow
- Paper: https://arxiv.org/abs/1701.00160
- Video: https://channel9.msdn.com/Events/Neural-Information-Processing-Systems-Conference/Neural-Information-Processing-Systems-Conference-NIPS-2016/Generative-Adversarial-Networks

You can find tips for training GAN here:
https://github.com/soumith/ganhacks

Review

http://www.rb139.com/index.php?s=/Lot/44547

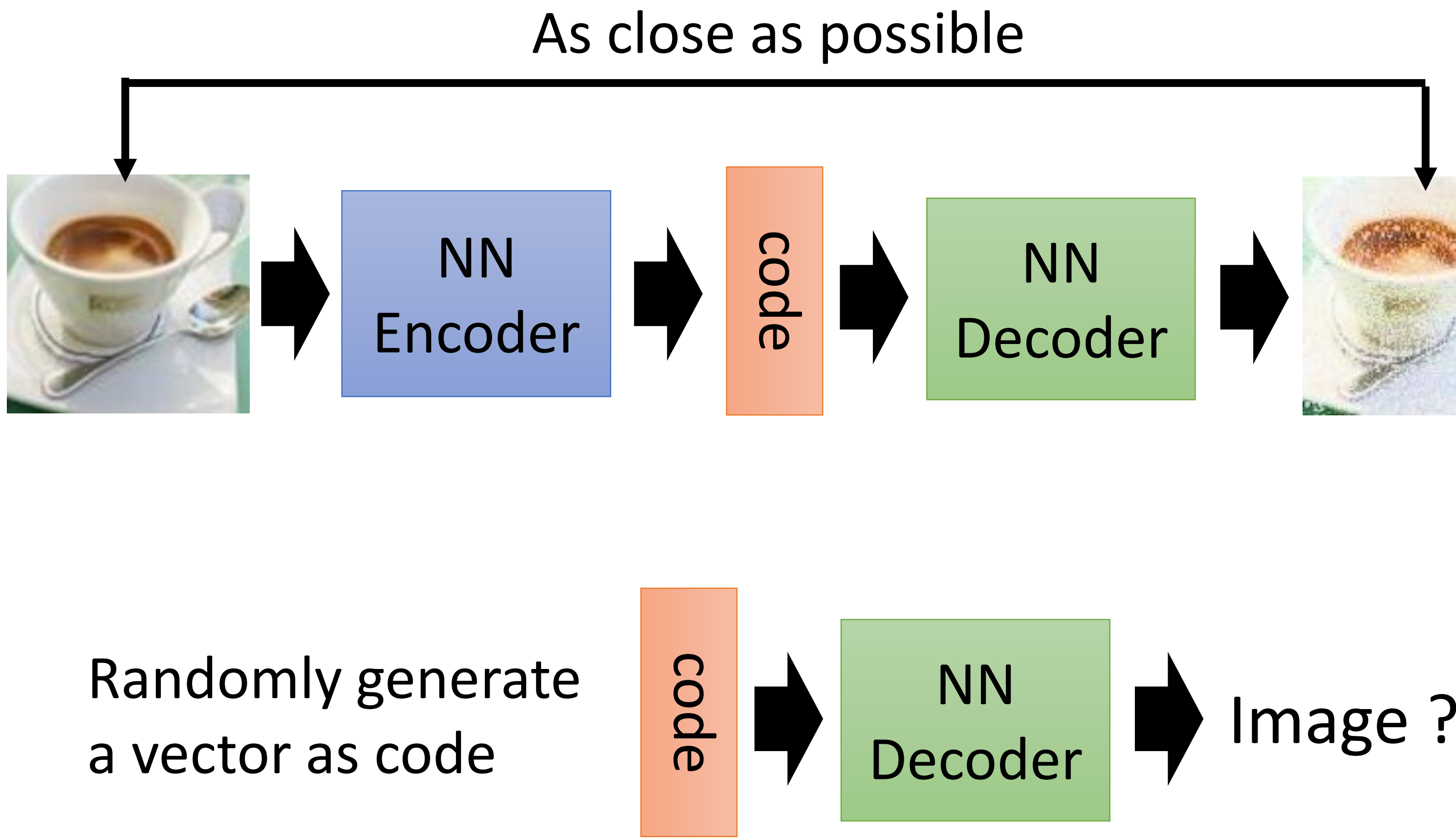
Generation



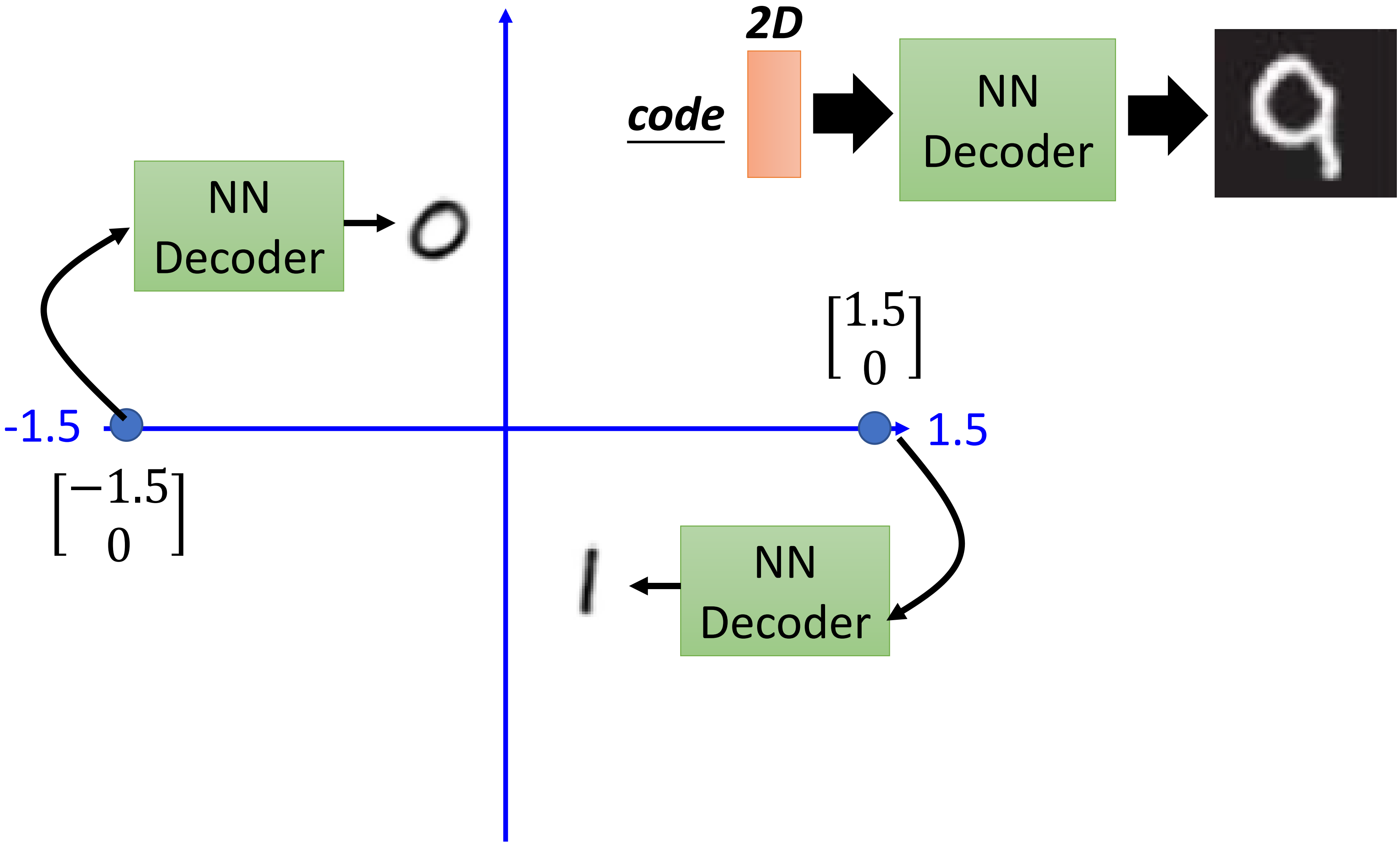
Writing Poems?

Drawing?

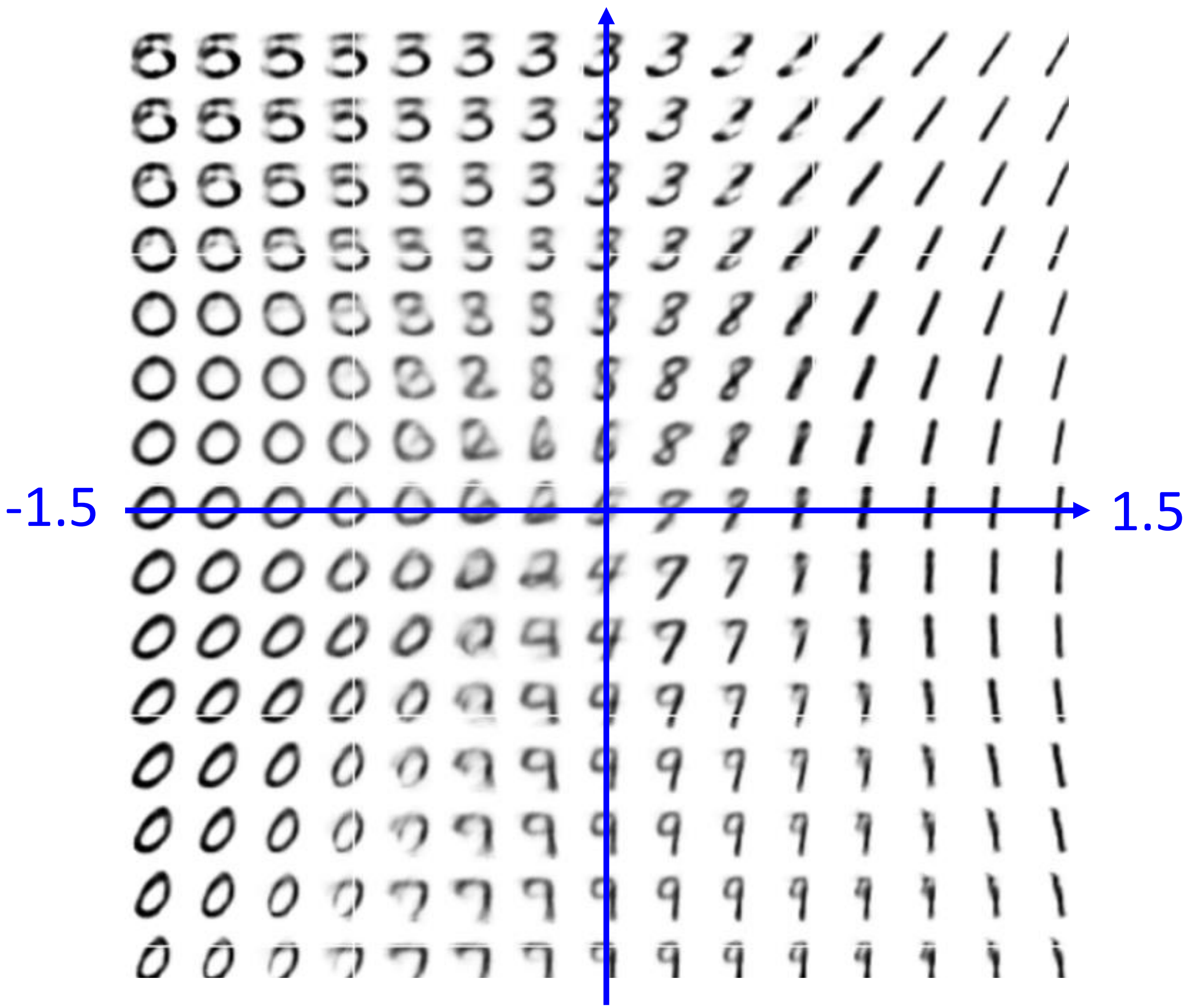
Review: Auto-encoder



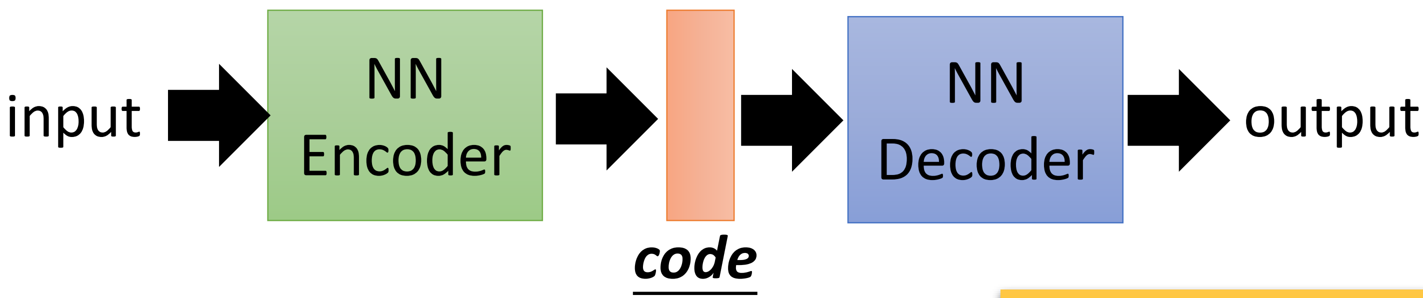
Review: Auto-encoder



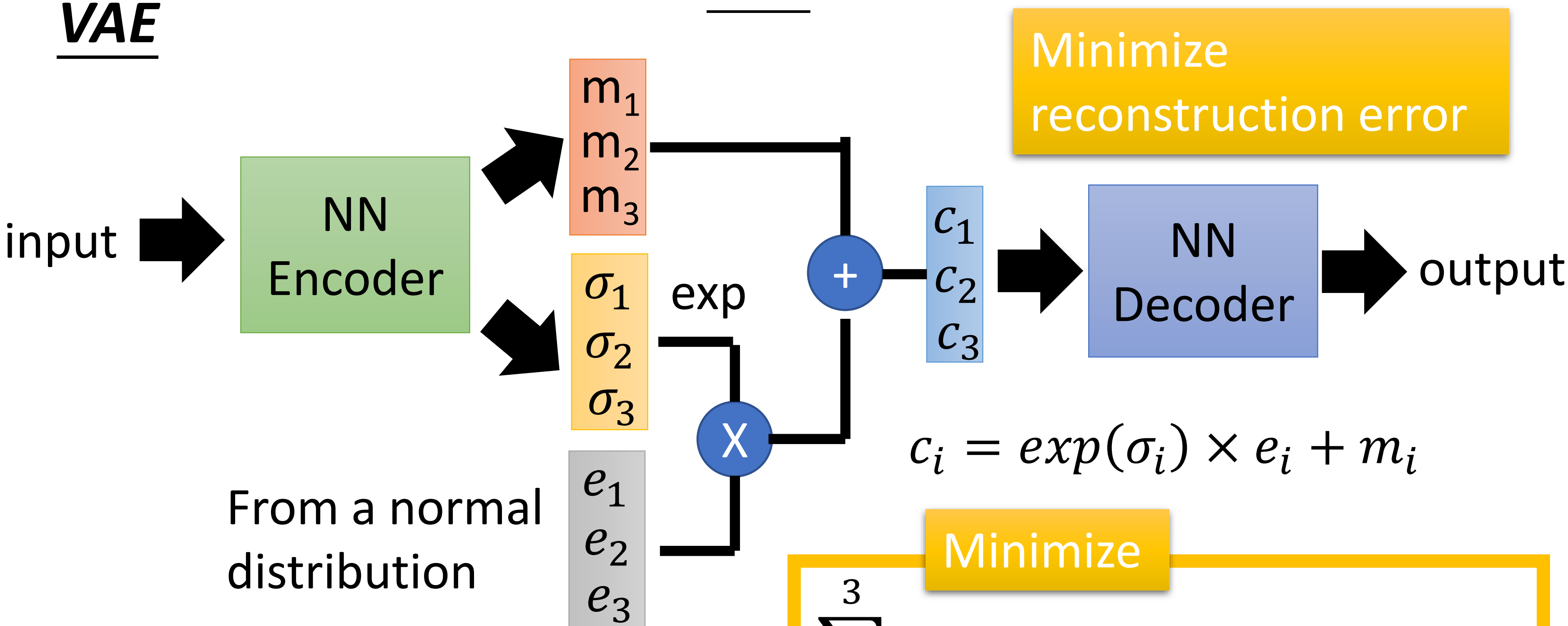
Review: Auto-encoder



Auto-encoder



VAE



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

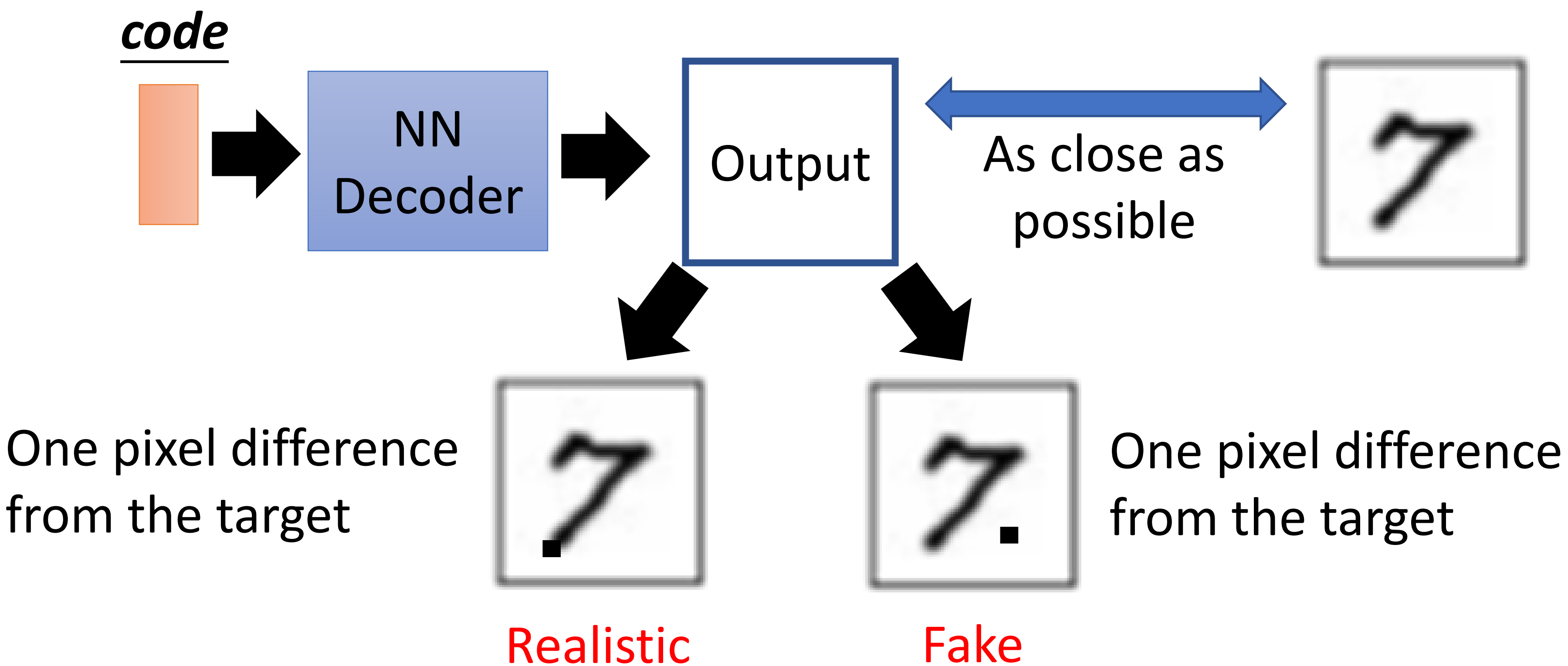
Minimize

$$\sum_{i=1}^{3} \left(exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2\right)$$

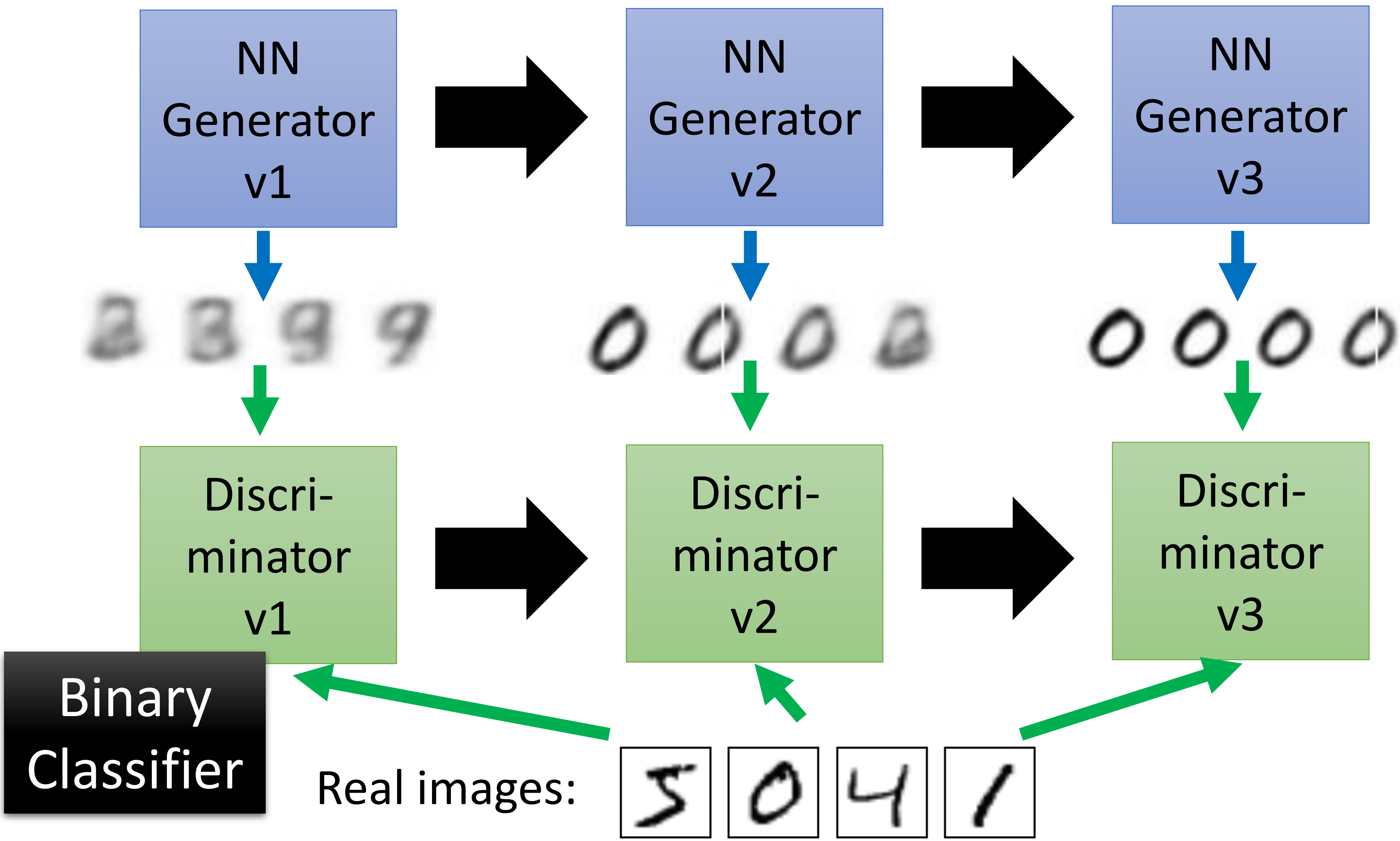
Auto-Encoding Variational Bayes,
https://arxiv.org/abs/1312.6114

Problems of VAE

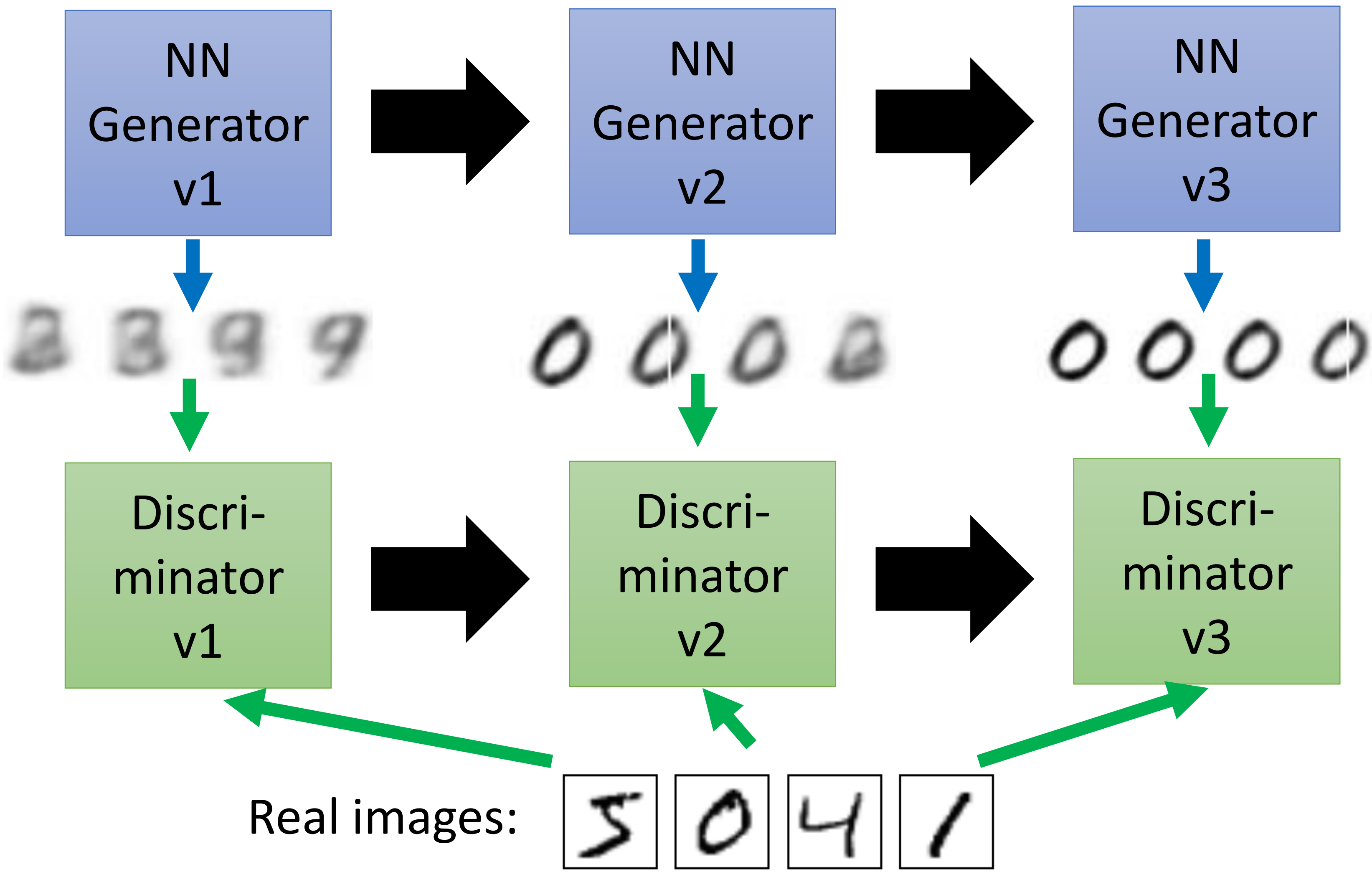
- It does not really try to simulate real images



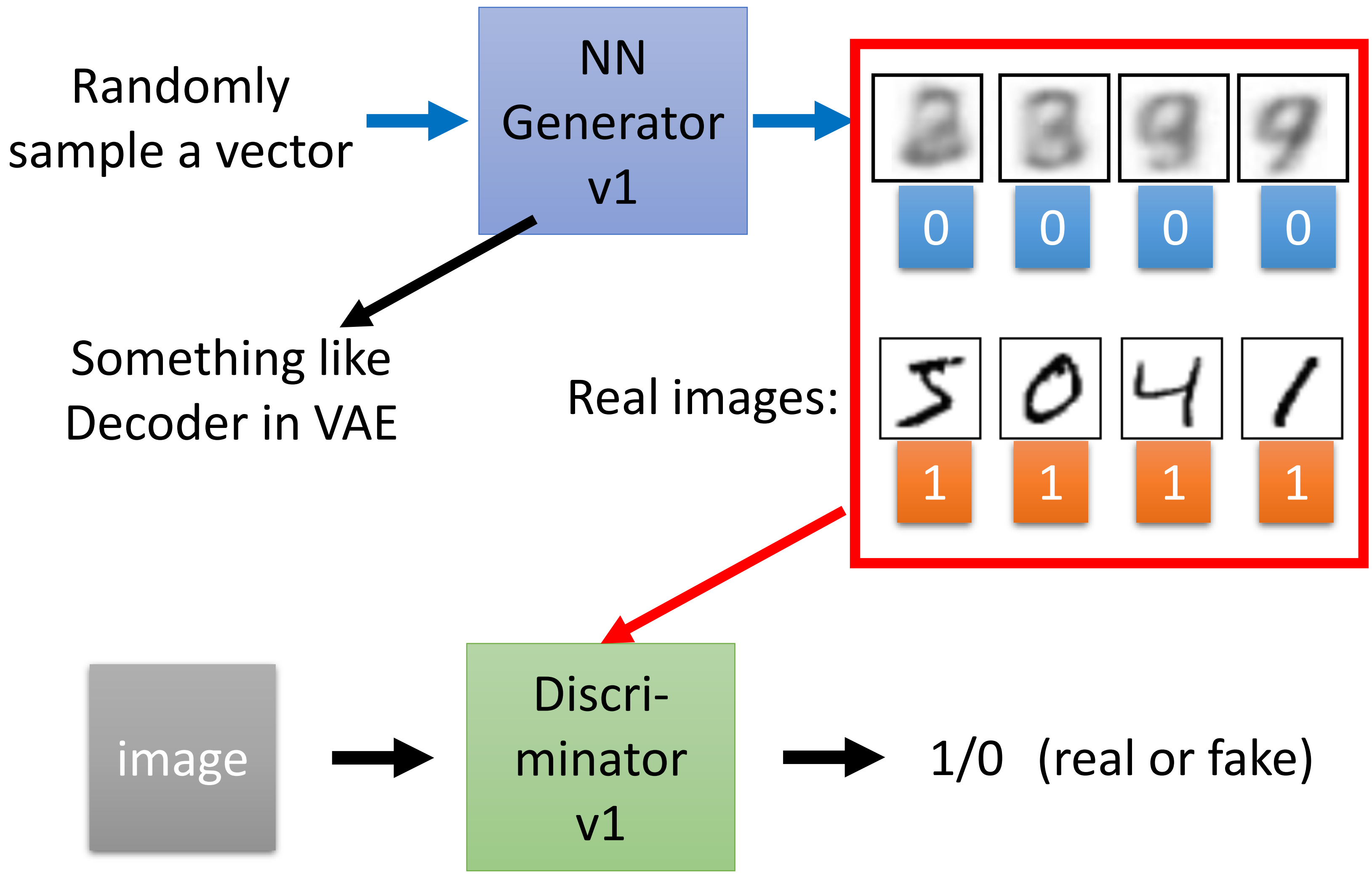
The evolution of generation



The evolution of generation



GAN - Discriminator



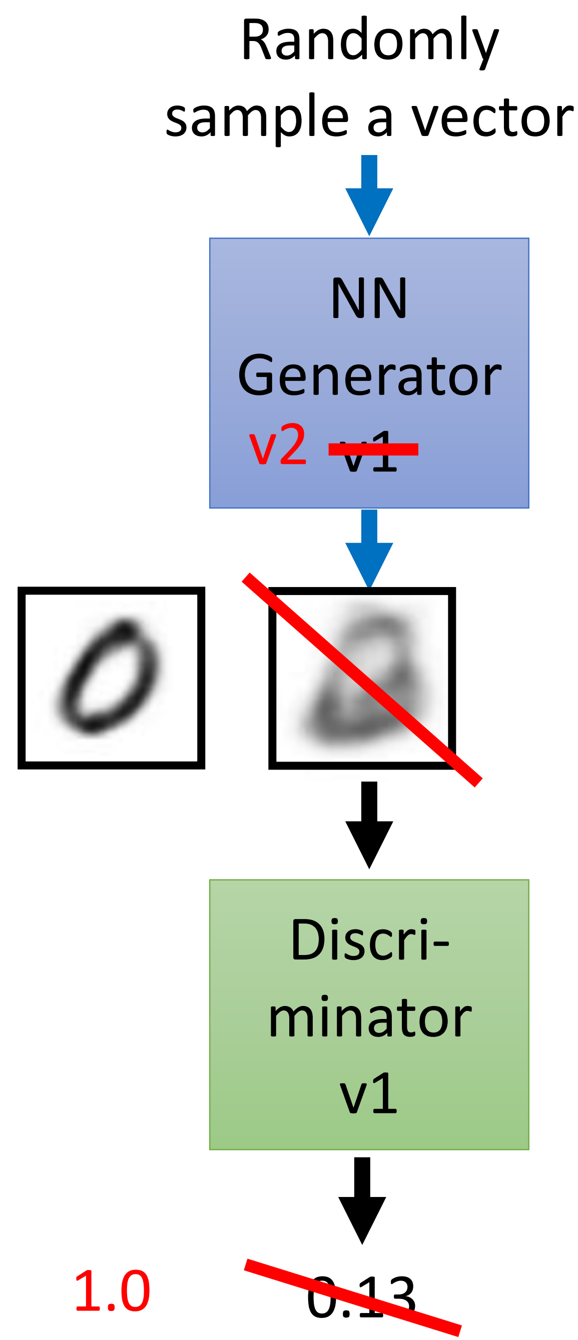
GAN - Generator

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

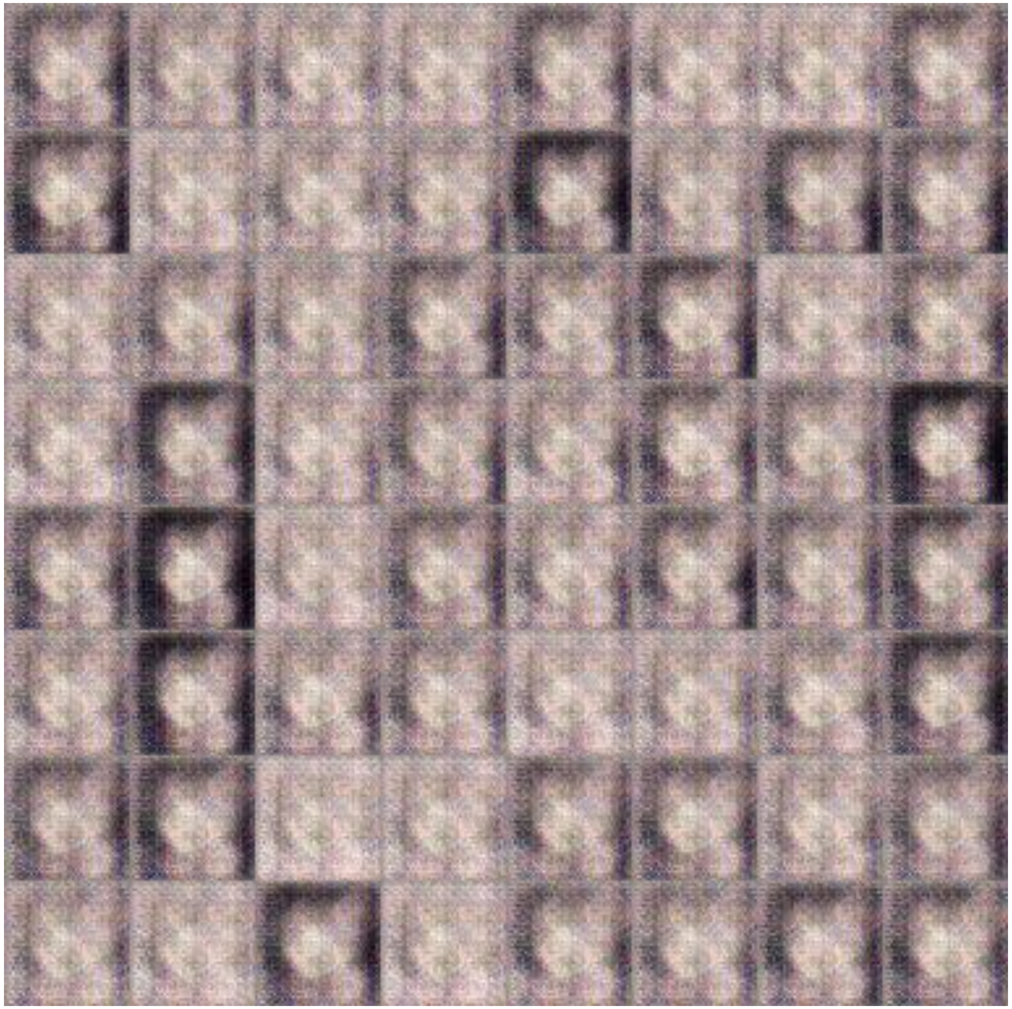


GAN – 二次元人物頭像鍊成

Source of images: https://zhuanlan.zhihu.com/p/24767059

DCGAN: https://github.com/carpedm20/DCGAN-tensorflow

GAN – 二次元人物頭像鍊成



100 rounds

GAN – 二次元人物頭像鍊成

1000 rounds

GAN – 二次元人物頭像鍊成

2000 rounds

GAN – 二次元人物頭像鍊成

5000 rounds

GAN – 二次元人物頭像鍊成

10,000 rounds

GAN – 二次元人物頭像鍊成

20,000 rounds

GAN – 二次元人物頭像鍊成

50,000 rounds

Basic Idea of GAN

Maximum Likelihood Estimation

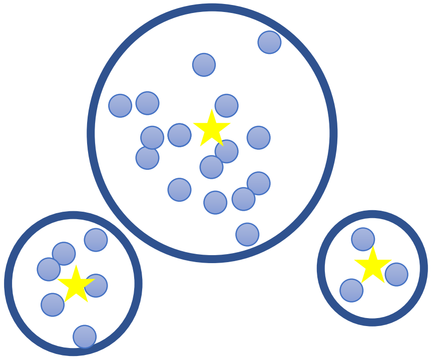
- Given a data distribution $P_{data}(x)$
- We have a distribution $P_G(x;\theta)$ parameterized by θ
  - E.g. $P_G(x;\theta)$ is a Gaussian Mixture Model, θ are means and variances of the Gaussians
  - We want to find θ such that $P_G(x;\theta)$ close to $P_{data}(x)$

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

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

Likelihood of generating the samples

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



Find θ^* maximizing the likelihood

Maximum Likelihood Estimation

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

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

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

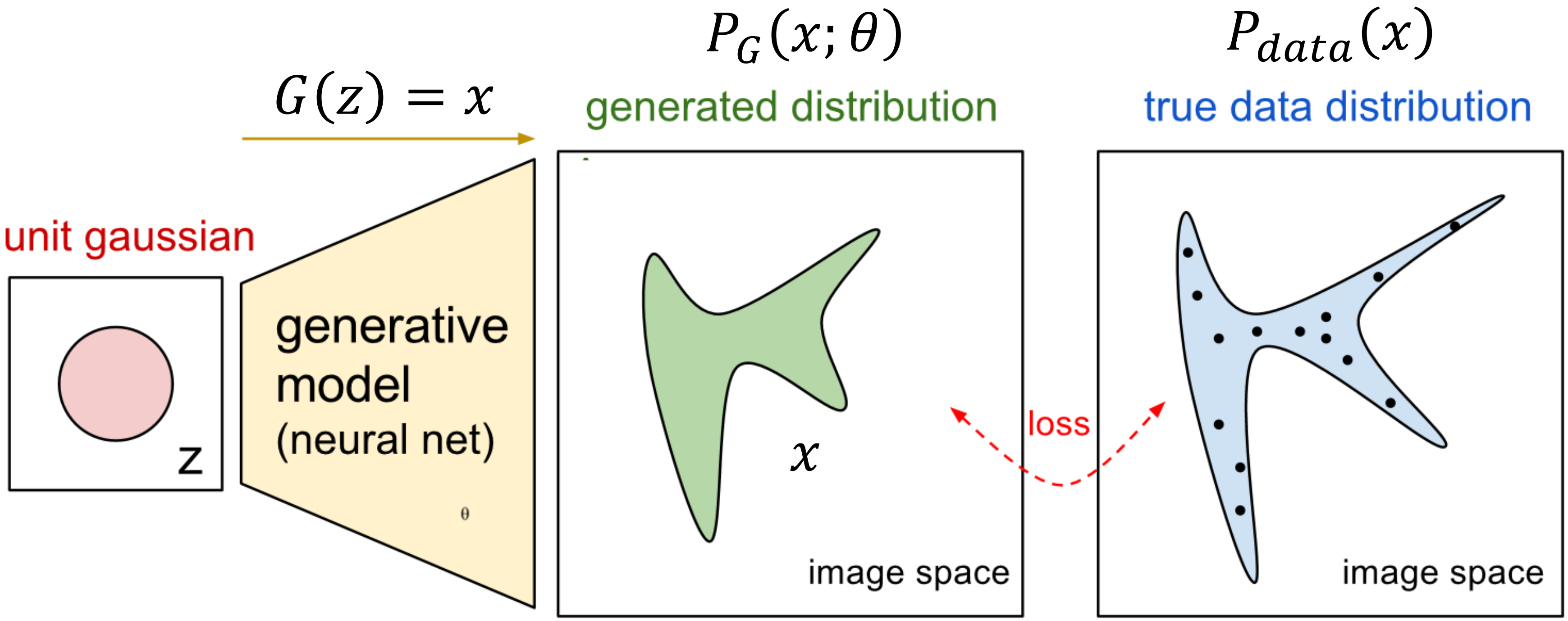
$$\approx arg\max_{\theta} E_{x\sim P_{data}}[logP_G(x;\theta)]$$

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

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

How to have a very general $P_G(x;\theta)$?

Now $P_G(x;\theta)$ is a NN



$$P_G(x) = \int_z P_{prior}(z) I_{[G(z)=x]} dz$$

It is difficult to compute the likelihood.

https://blog.openai.com/generative-models/

Basic Idea of GAN

- 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 $P_{data}(x)$
- There is a function V(G,D).

Hard to learn by maximum likelihood

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

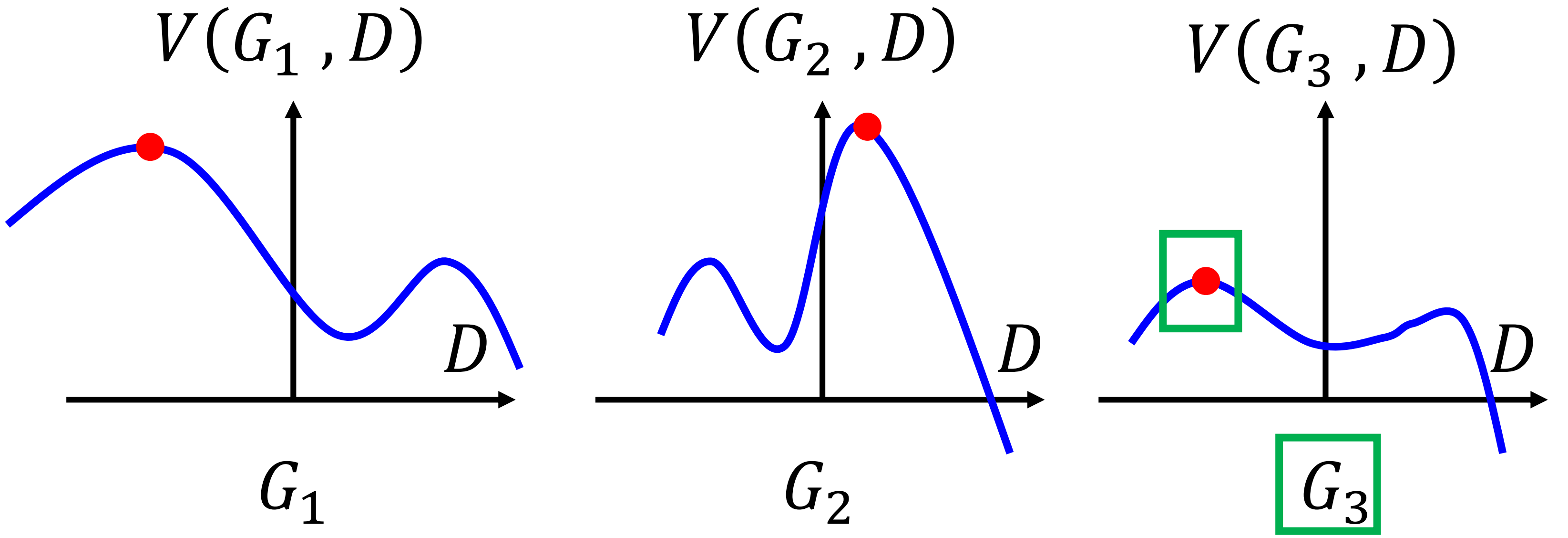
Basic Idea

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

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

Given a generator G, $\max_D V(G, D)$ evaluate the "difference" between P_G and P_{data}

Pick the G defining P_G most similar to P_{data}



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

- Given G, what is the optimal D* maximizing

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

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

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

Assume that D(x) can have any value here

- Given x, the optimal D* maximizing

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

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

- Given x, the optimal D* maximizing

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

a D b D

- Find D* maximizing: $\mathrm{f}(D) = a log(D) + b log(1 - D)$

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

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

$$a - aD^* = bD^*$$

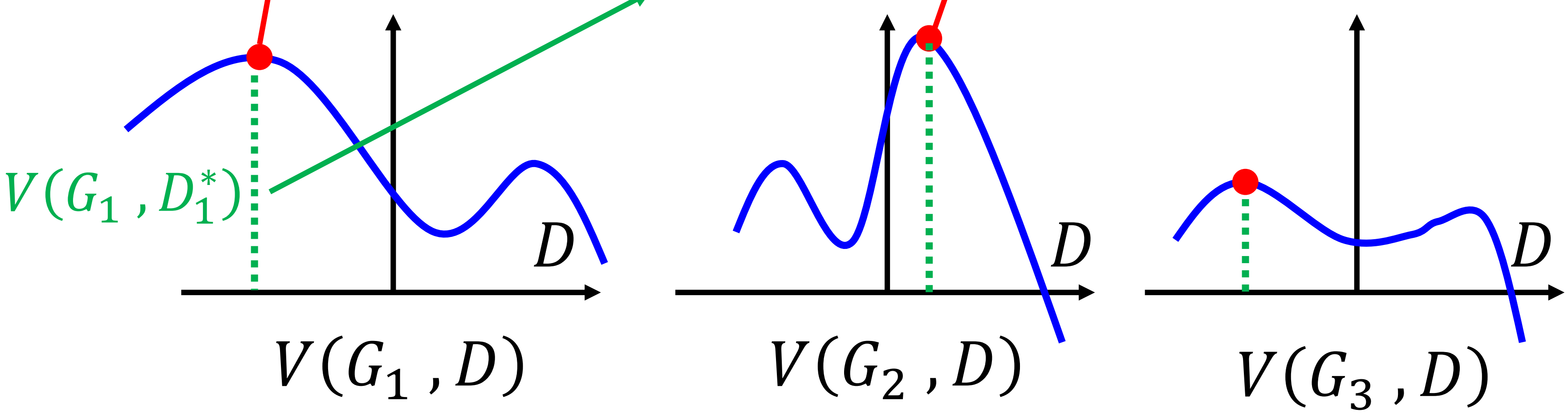
$$D^* = \frac{a}{a+b} \qquad D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

$0 <$ < 1

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

$$D_1^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_{G_1}(x)} \qquad D_2^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_{G_2}(x)}$$

"difference" between P_{G_1} and P_{data}



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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$= -2log2 + 2JSD\big(P_{data}(x)||P_G(x)\big)$$ Jensen-Shannon divergence

In the end ……

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

- Generator G, Discriminator D
- Looking for G* such that

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

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

0 < < log 2

$$= -2log2 + 2JSD\left(P_{data}(x)||P_G(x)\right)$$

- What is the optimal G?

$$P_G(x) = P_{data}(x)$$

Algorithm

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

$L(G)$

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

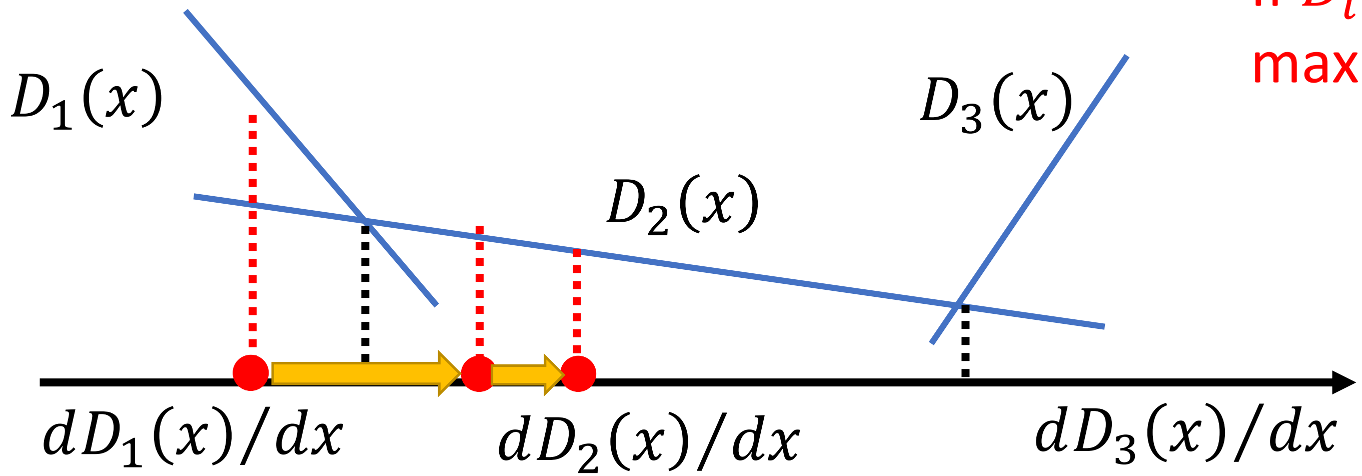
$$\theta_G \leftarrow \theta_G - \eta\, \partial L(G)/\partial \theta_G$$

θ_G defines G

$$f(x) = \max\{D_1(x), D_2(x), D_3(x)\}$$

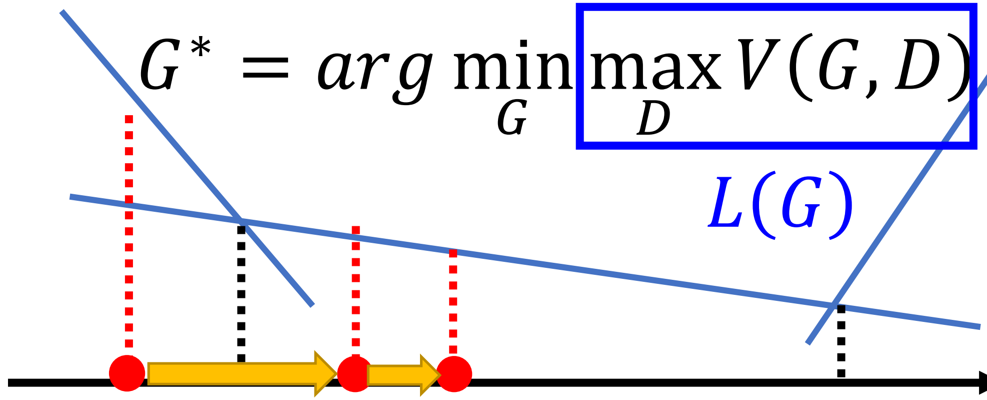
$$\frac{df(x)}{dx} =? \; dD_i(x)/dx$$

If $D_i(x)$ is the max one



Algorithm

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



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

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

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

Decrease JS divergence(?)

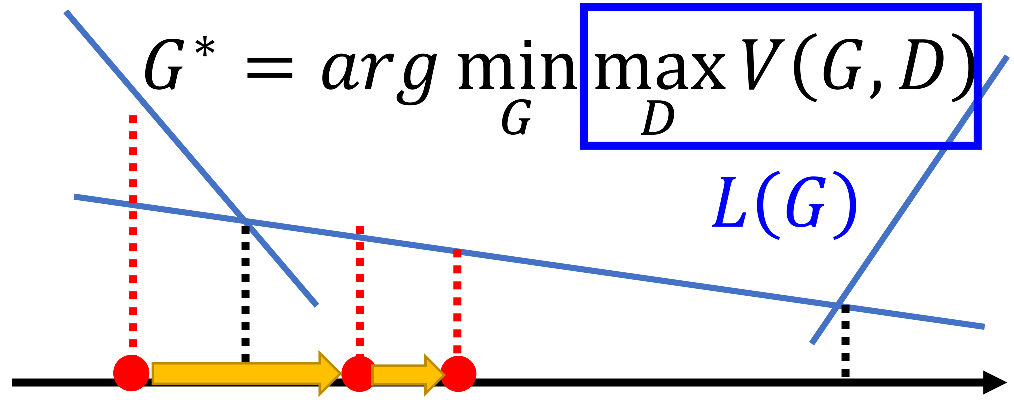
$V(G_1, D_1^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_1}(x)$

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

Decrease JS divergence(?)

Algorithm

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

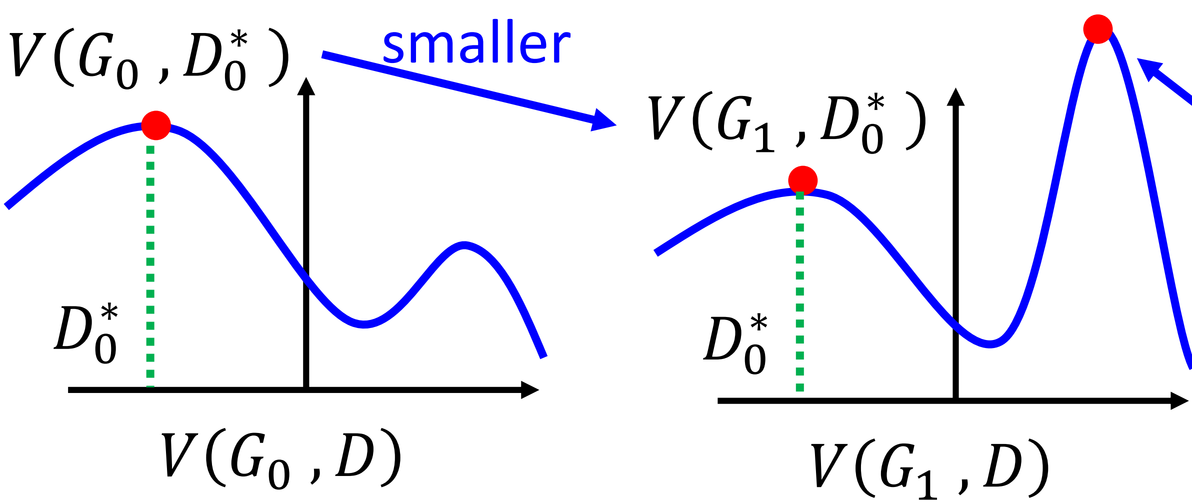


- Given G_0
- Find D_0^* maximizing $V(G_0, D)$

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

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

Decrease JS divergence(?)



$V(G_1, D_1^*)$ ……

Assume $D_0^* \approx D_1^*$

Don't update G too much

In practice …

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

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

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

Binary Classifier

Output is D(x) Minimize Cross-entropy

If x is a positive example → Minimize –log D(x)

If x is a negative example → Minimize –log(1-D(x))

Binary Classifier

Output is f(x) Minimize Cross-entropy

If x is a positive example → Minimize –log f(x)

If x is a negative example → Minimize –log(1-f(x))

D is a binary classifier (can be deep) with parameters θ_d

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

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

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

||

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

Algorithm

Initialize θ_d for D and θ_g for G

- In each training iteration:

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

Learning D

Repeat k times

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

Learning G

Only Once

  - Sample another m noise samples $\{z^1, z^2, \dots, z^m\}$ from the prior $P_{prior}(z)$
  - Update generator parameters θ_g to minimize
    - $\tilde{V} = \cancel{\frac{1}{m}\sum_{i=1}^{m} logD(x^i)} + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D\left(G(z^i)\right)\right)$
    - $\theta_g \leftarrow \theta_g - \eta\nabla\tilde{V}(\theta_g)$

Objective Function for Generator in Real Implementation

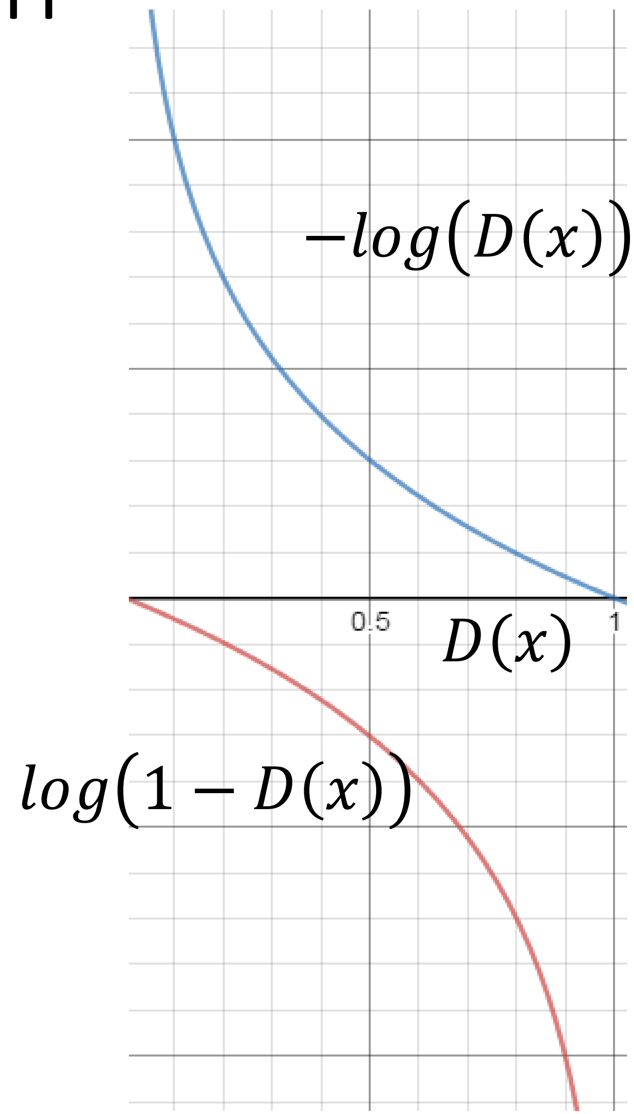
$$V = \cancel{E_{x\sim P_{data}}\left[logD(x)\right]}$$

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

Slow at the beginning

$$V = E_{x\sim P_G}\left[-log\left(D(x)\right)\right]$$

Real implementation:
label x from P_G as positive

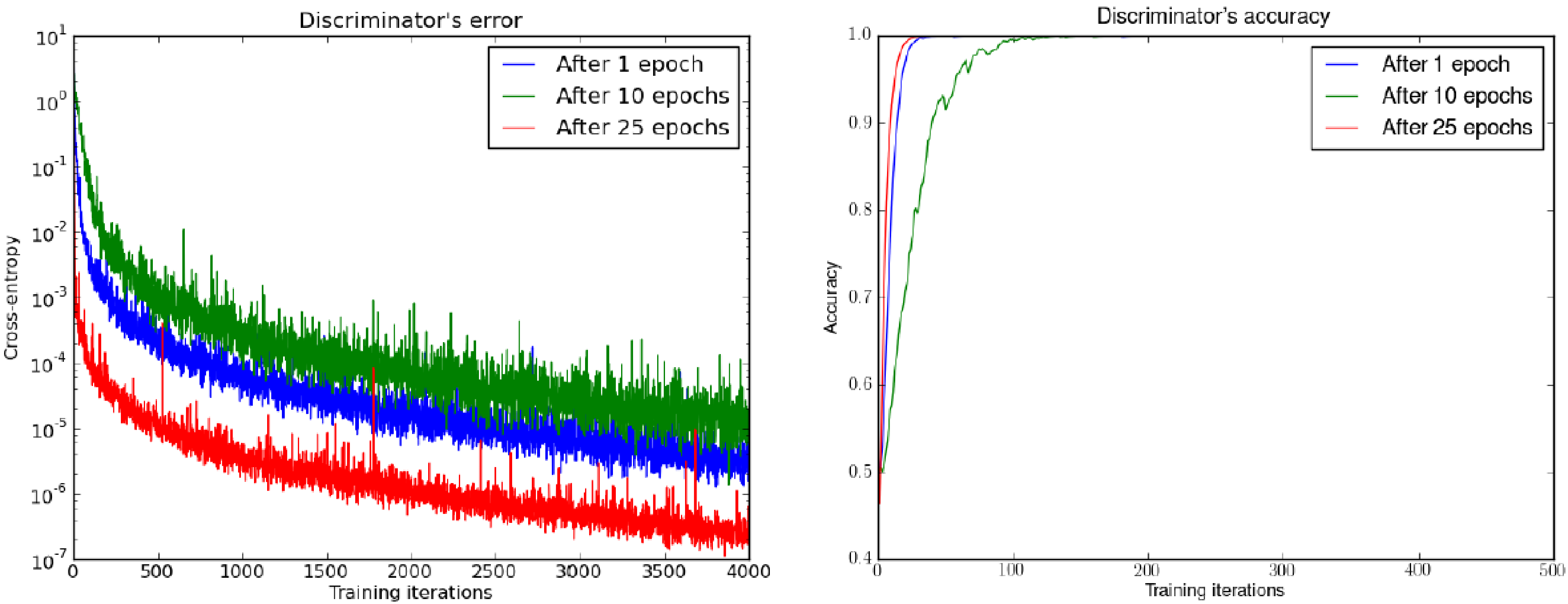


Demo

- The code used in demo from:
  - https://github.com/osh/KerasGAN/blob/master/MNIST_CNN_GAN_v2.ipynb

Issue about Evaluating the Divergence

Evaluating JS divergence

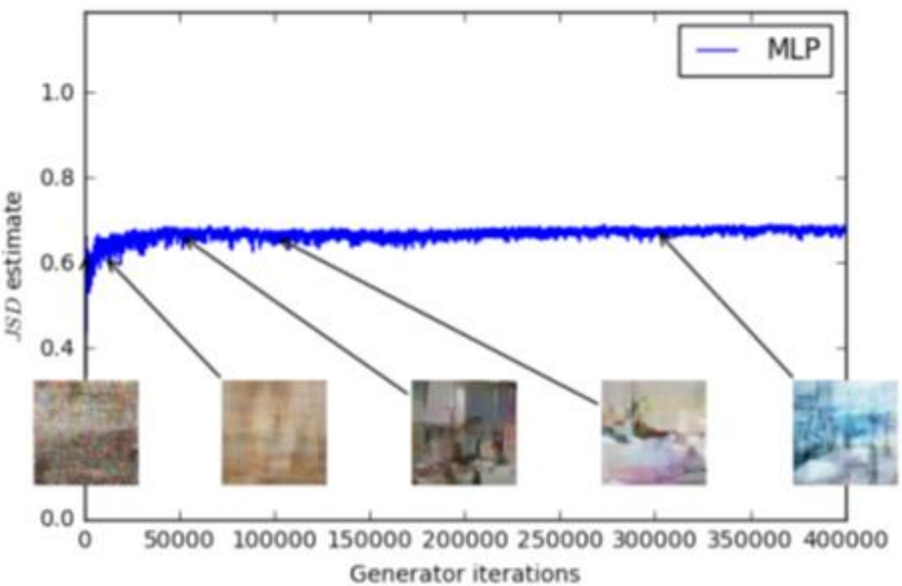


Martin Arjovsky, Léon Bottou, Towards Principled Methods for Training Generative Adversarial Networks, 2017, arXiv preprint

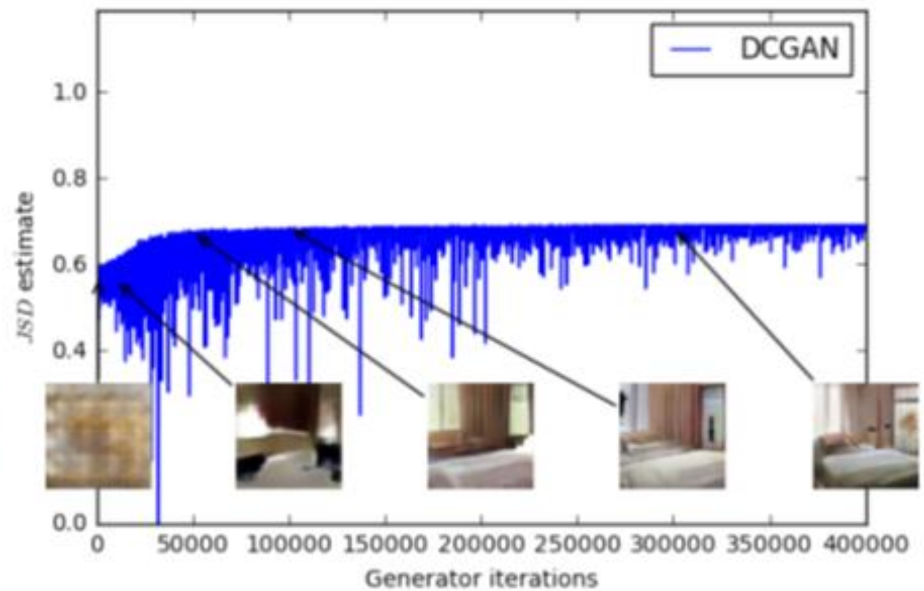
Evaluating JS divergence

https://arxiv.org/abs/1701.07875

- JS divergence estimated by discriminator telling little information



Weak Generator



Strong Generator

Discriminator

$$V = E_{x\sim P_{data}}\left[log\overset{1}{D(x)}\right] + E_{x\sim P_G}\left[log\left(1-\overset{0}{D(x)}\right)\right]$$

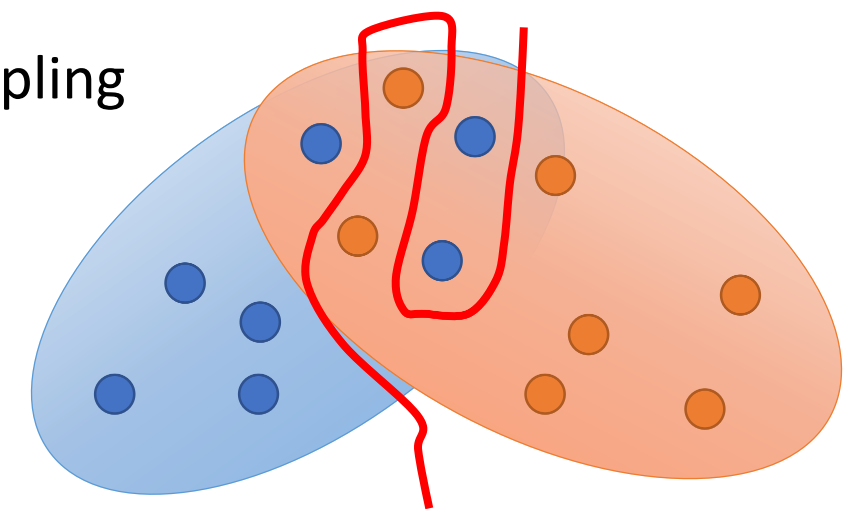
$$\approx \frac{1}{m}\sum_{i=1}^{m} logD\left(x^i\right) + \frac{1}{m}\sum_{i=1}^{m} log\left(1-D\left(\tilde{x}^i\right)\right)$$

$$\max_D V(G,D) = -2log2 + 2\underbrace{JSD\left(P_{data}(x)||P_G(x)\right)}_{log2} \quad = 0$$

Reason 1. Approximate by sampling

Weaken your discriminator?

Can weak discriminator compute JS divergence?



Discriminator

$$V = E_{x\sim P_{data}}[logD(\overset{1}{x})] + E_{x\sim P_G}[log(1 - \overset{0}{D(x)})]$$

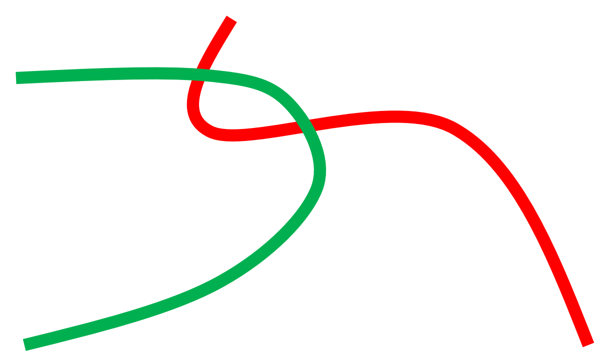
$$\approx \frac{1}{m}\sum_{i=1}^{m} logD(x^i) + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D(\tilde{x}^i)\right)$$

$$\max_D V(G,D) = -2log2 + 2\underbrace{JSD\left(P_{data}(x)||P_G(x)\right)}_{\log 2} \quad \boxed{= 0}$$

Reason 2. the nature of data

Both $P_{data}(x)$ and $P_G(x)$ are low-dim manifold in high-dim space

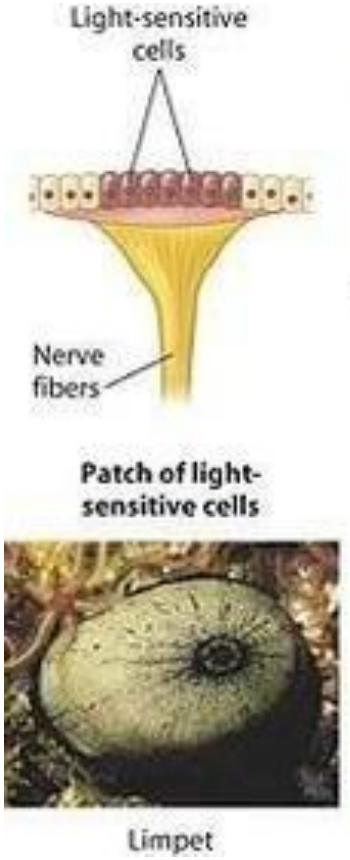
Usually they do not have any overlap

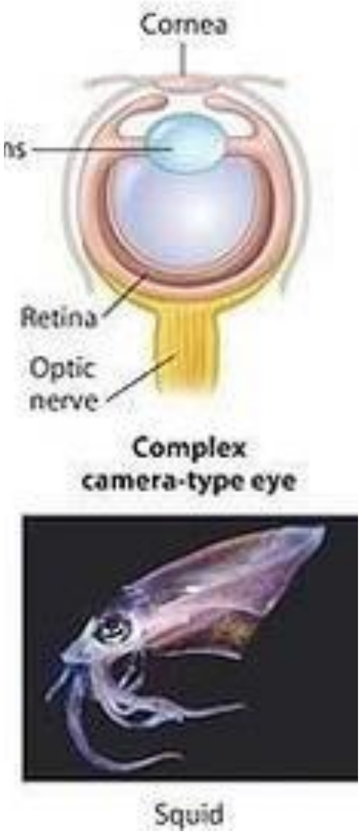


Evaluation

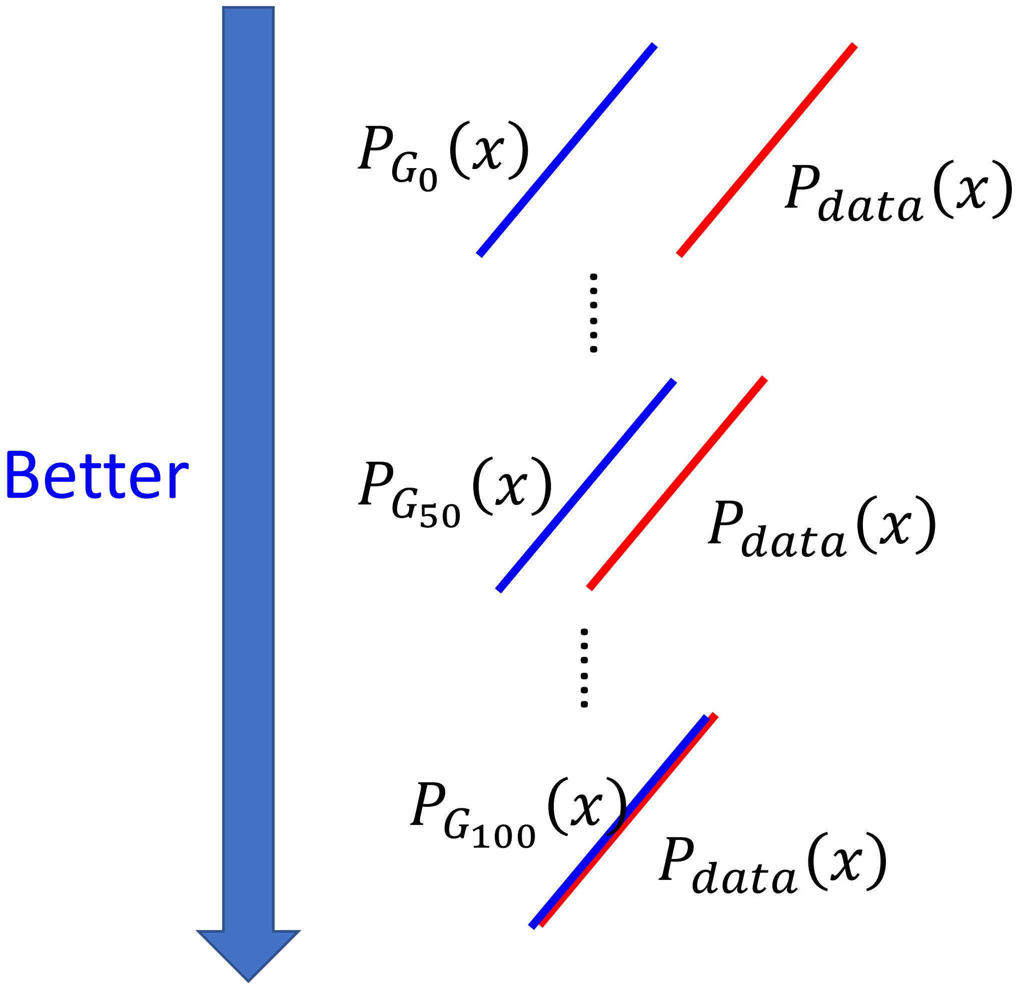
http://www.guokr.com/post/773890/

Better





Evaluation



$$JS\left(P_{G_1}||P_{data}\right) = log2$$

$$JS\left(P_{G_2}||P_{data}\right) = log2$$

Not really better ......

$$JS\left(P_{G_2}||P_{data}\right) = 0$$

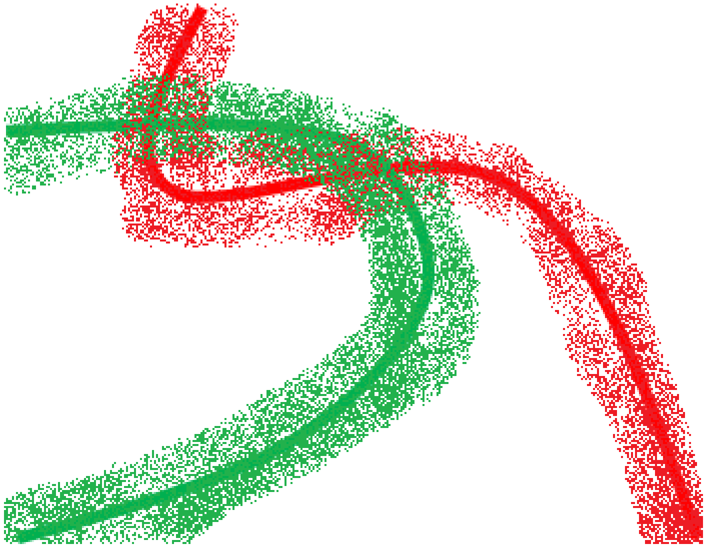
Add Noise

- Add some artificial noise to the inputs of discriminator
- Make the labels noisy for the discriminator

Discriminator cannot perfectly separate real and generated data

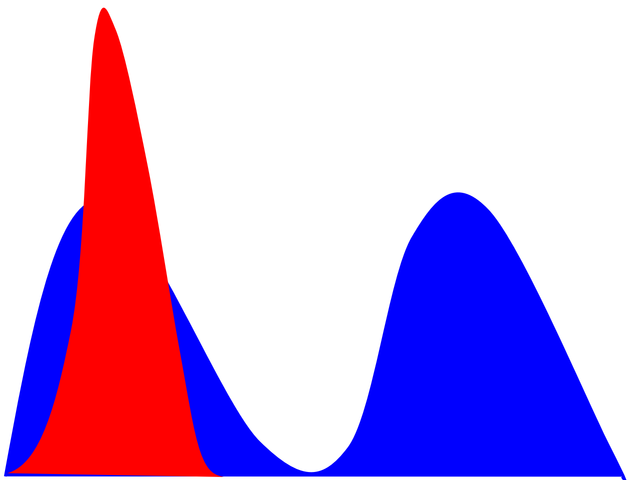
$P_{data}(x)$ and $P_G(x)$ have some overlap

Noises decay over time



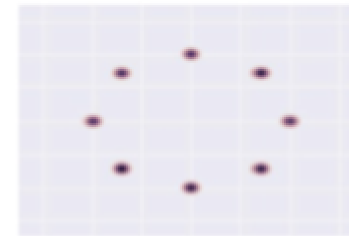
Mode Collapse

Mode Collapse



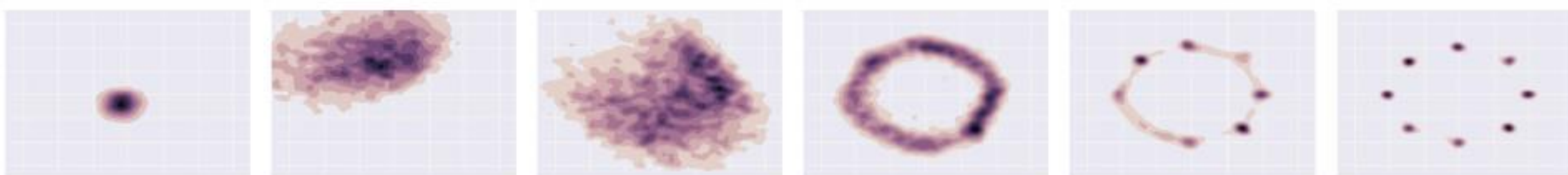
Mode Collapse

P_{data}



Target

What we want …



Step 0 | Step 5k | Step 10k | Step 15k | Step 20k | Step 25k

In reality …

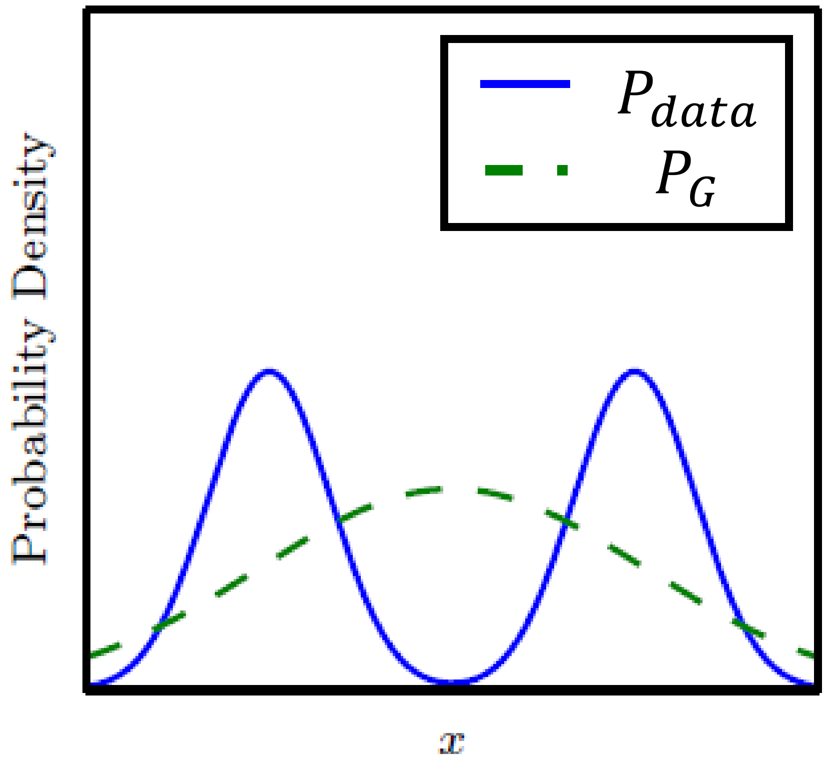
Step 0 | Step 5k | Step 10k | Step 15k | Step 20k | Step 25k

Modified from Ian Goodfellow's tutorial

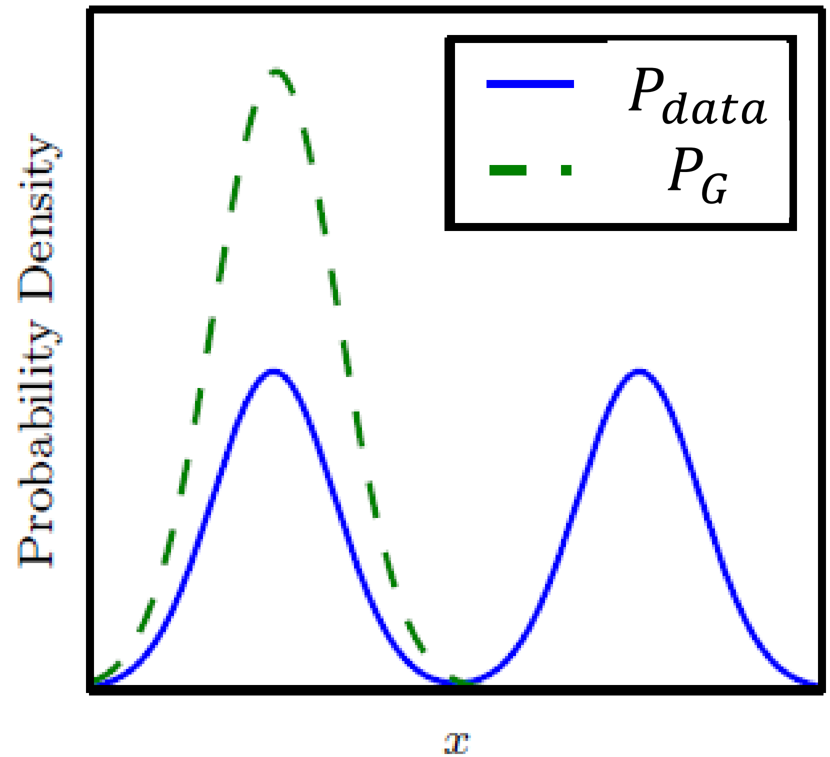
Flaw in Optimization?

$$KL = \int P_{data} log \frac{P_{data}}{P_G} dx$$

Reverse $KL = \int P_G log \frac{P_G}{P_{data}} dx$



Maximum likelihood
(minimize KL($P_{data}||P_G$))



Minimize KL($P_G||P_{data}$)
(reverse KL)

This may not be the reason (based on Ian Goodfellow's tutorial)

So many GANs ……

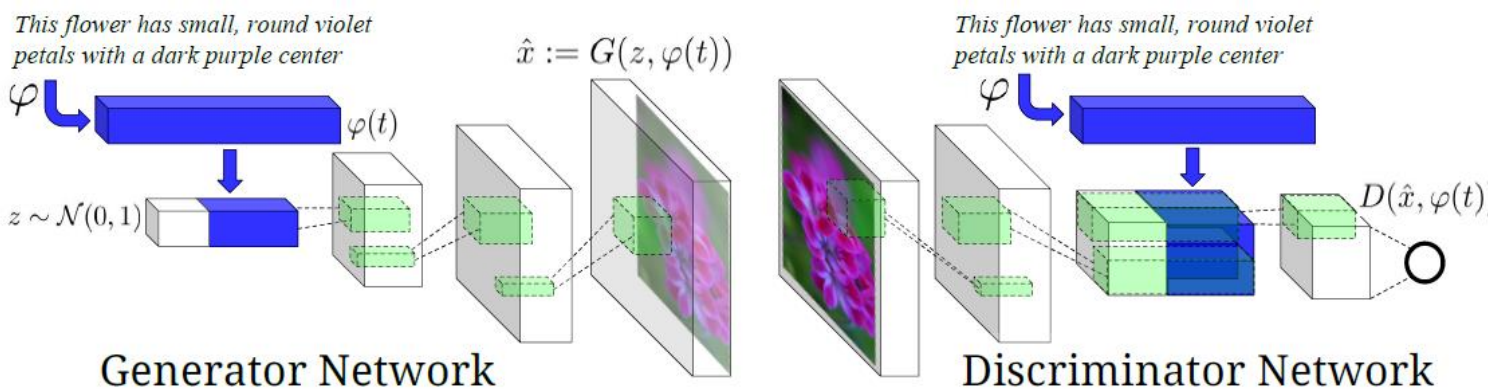
Modifying the Optimization of GAN
fGAN
WGAN
Least-square GAN
Loss Sensitive GAN
Energy-based GAN
Boundary-seeking GAN
Unroll GAN
……

Different Structure from the Original GAN
Conditional GAN
Semi-supervised GAN
InfoGAN
BiGAN
Cycle GAN
Disco GAN
VAE-GAN
……

Conditional GAN

Motivation

Text → Generator → Image



Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, Honglak Lee, "Generative Adversarial Text-to-Image Synthesis", ICML 2016

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, Dimitris Metaxas, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks", arXiv prepring, 2016

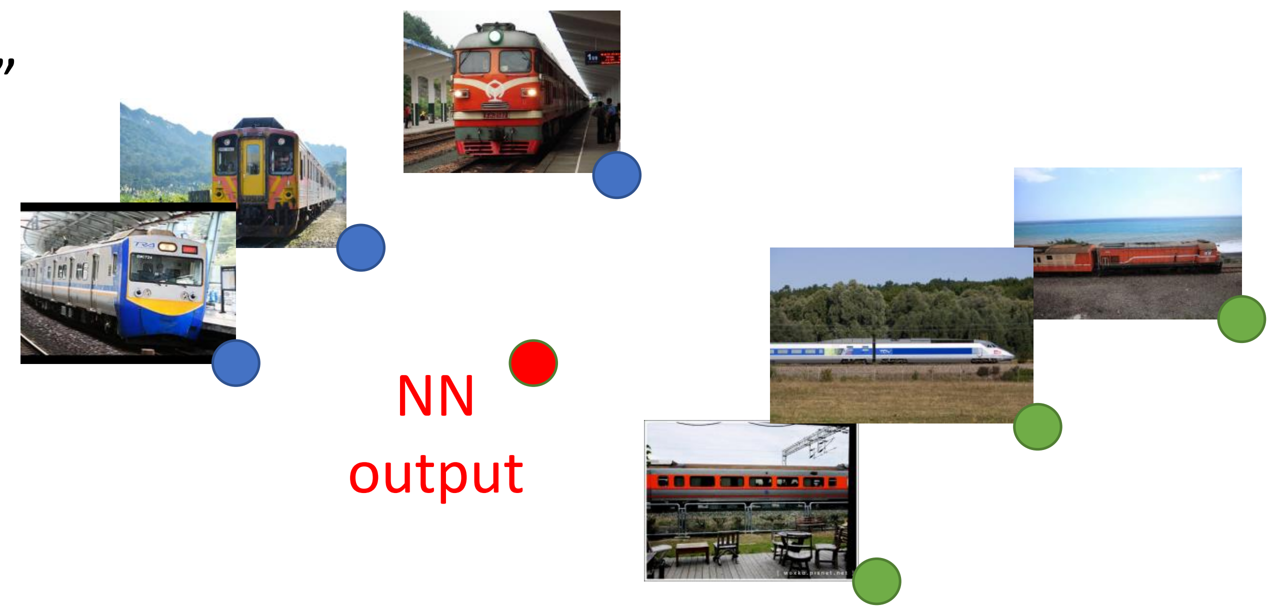
Scott Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, Honglak Lee, "Learning What and Where to Draw", NIPS 2016

Motivation

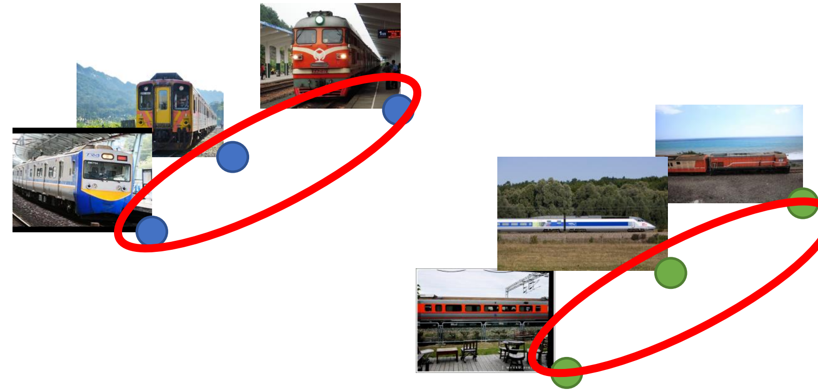
- Challenge

Text c → NN → Image x (a point, not a distribution)

Text: "train"



Conditional GAN



Training data: $(\hat{c}, \hat{x})$

condition c → G → x

~~Prior distribution z~~

Learn to ignore this term …

dropout

Learn to approximate P(x|c)

$(\hat{c}, x = G(\hat{c}))$ classified as positive

x → D (v1) → scalar

Can generated x not related to c

c, x → D (v2) → scalar

Positive example: $(\hat{c}, \hat{x})$

Negative example: $(\hat{c}, G(\hat{c})), (\widehat{c'}, \hat{x})$

Text to Image
- Results

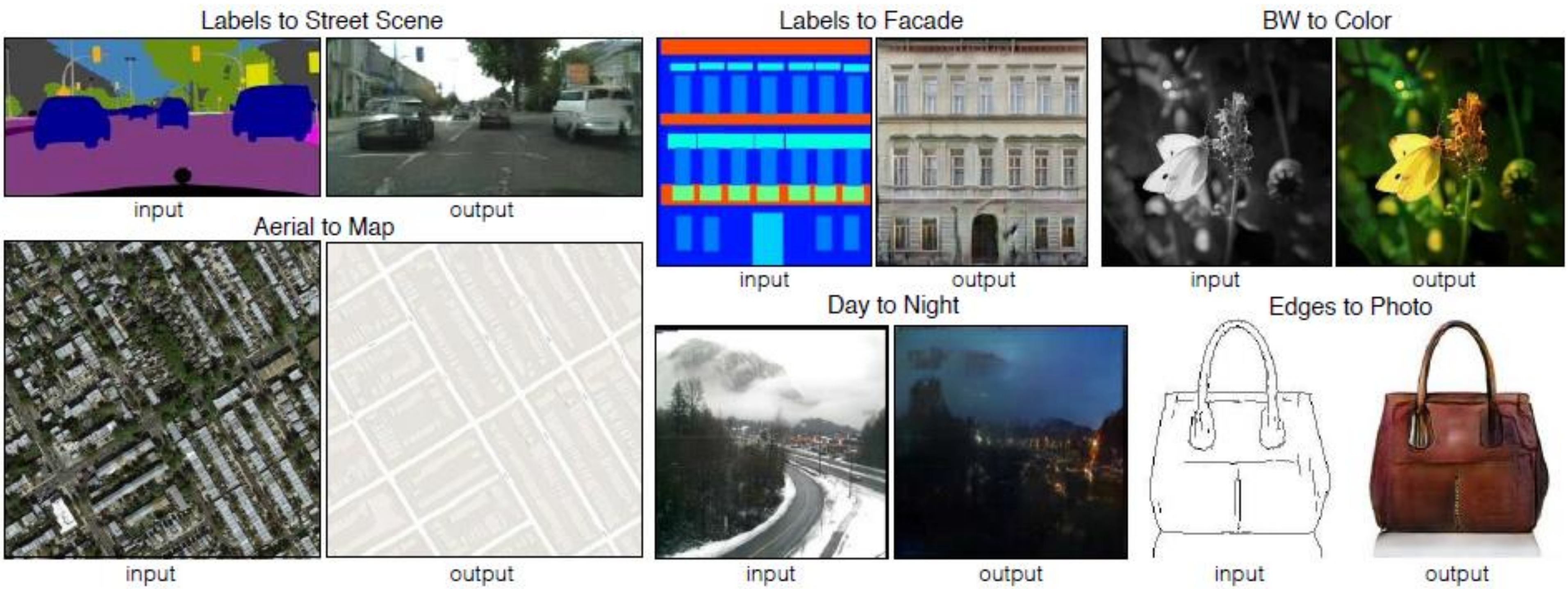
Caption	Image
a pitcher is about to throw the ball to the batter	
a group of people on skis stand in the snow	
a man in a wet suit riding a surfboard on a wave	

Text to Image - Results

"red flower with black center"

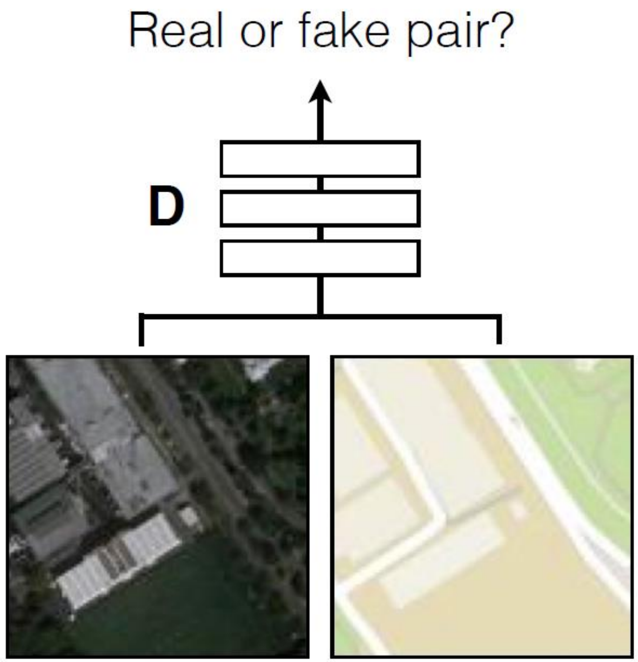
Caption	Image
this flower has white petals and a yellow stamen	
the center is yellow surrounded by wavy dark purple petals	
this flower has lots of small round pink petals	

Image-to-image Translation

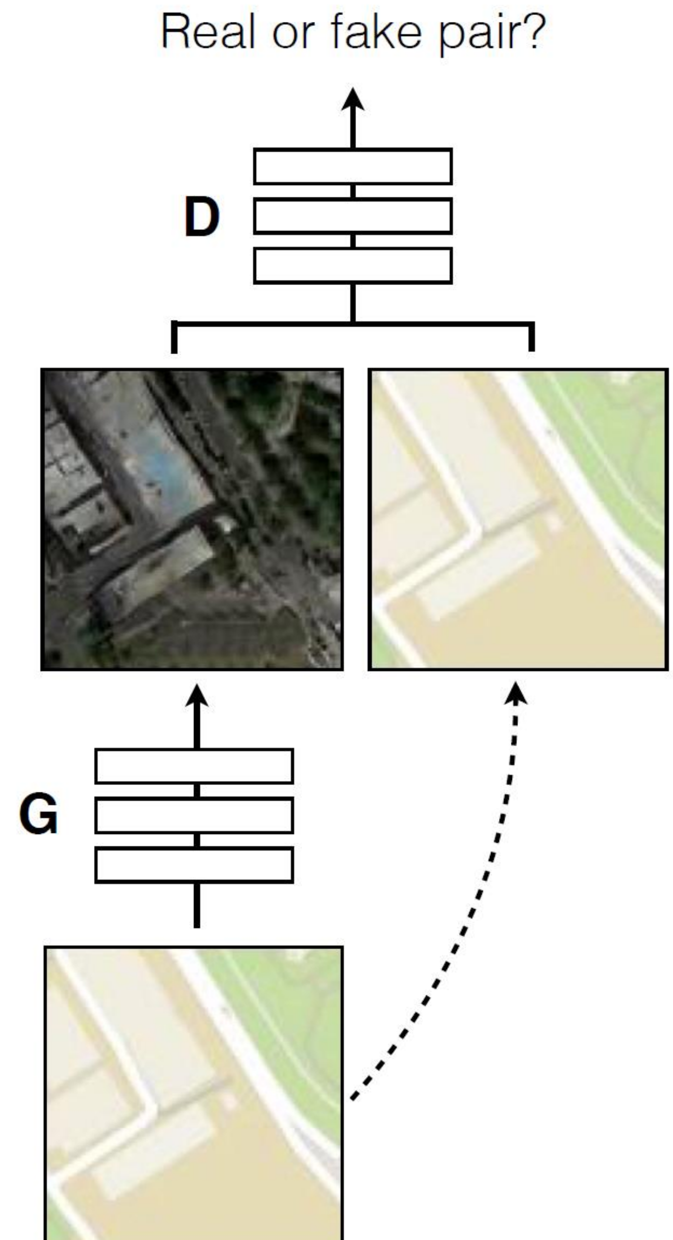


Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks”, arXiv preprint, 2016

Positive examples



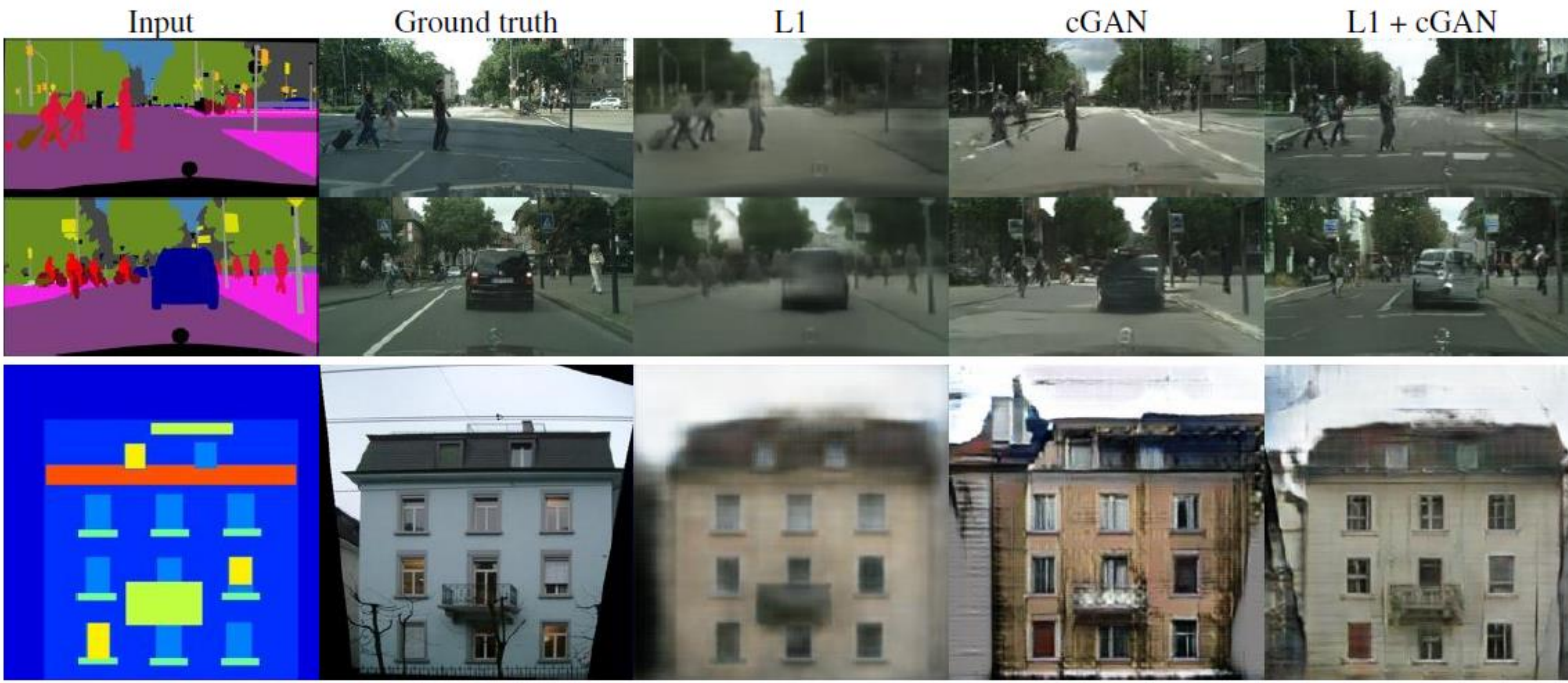
Negative examples



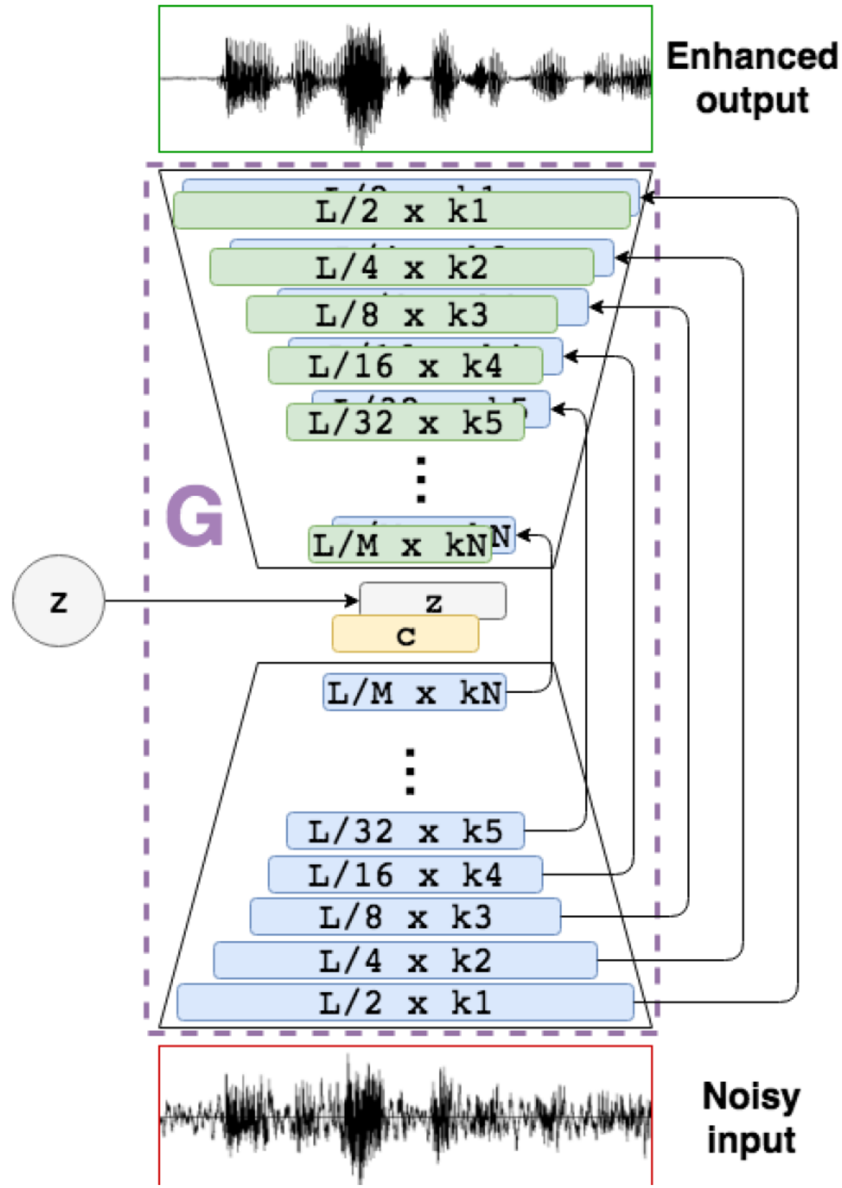
G tries to synthesize fake images that fool **D**

D tries to identify the fakes

Image-to-image Translation
- Results

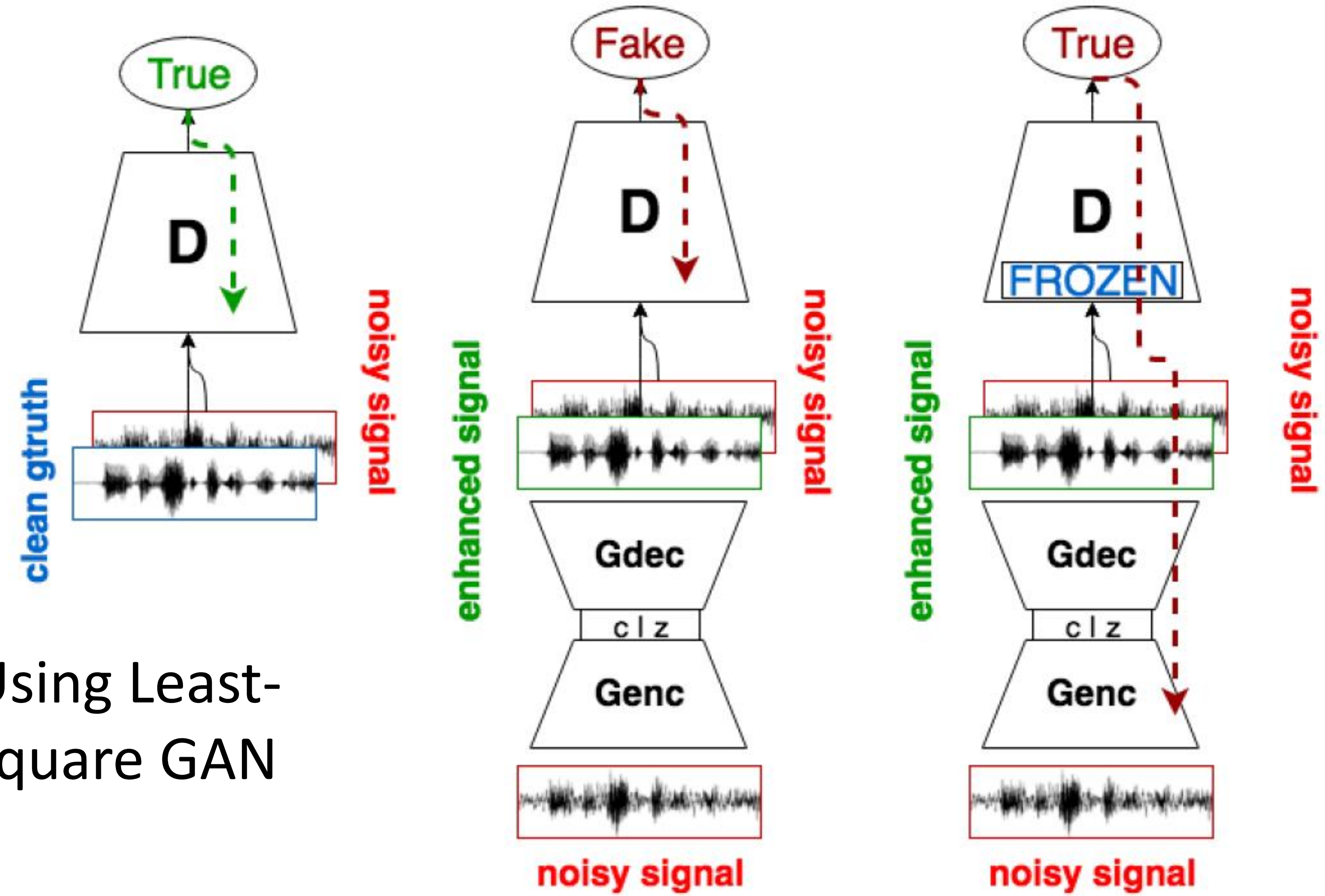


Speech Enhancement GAN



https://arxiv.org/abs/1703.09452

Speech Enhancement GAN



Using Least-square GAN

Least-square GAN

- For discriminator

D has linear output

$$\min_D \frac{1}{2} E_{x\sim P_{data}}[(D(x) - \underbrace{b}_{1})^2] + \frac{1}{2} E_{x\sim P_G}[(D(x) - \underbrace{a}_{0})^2]$$

- For Generator

$$\min_D \frac{1}{2} E_{z\sim P_{data}}\left[\left(D\left(G(z)\right) - \underbrace{c}_{1}\right)^2\right]$$

Least-square GAN

- The code used in demo from:
  - https://github.com/osh/KerasGAN/blob/master/MNIST_CNN_GAN_v2.ipynb