What is this? Gum? It's GAN.

: Intuition & Mathematical proof

ISL Lab Seminar

Hansol Kang

ISL Image System Laboratory

Contents

Introduction

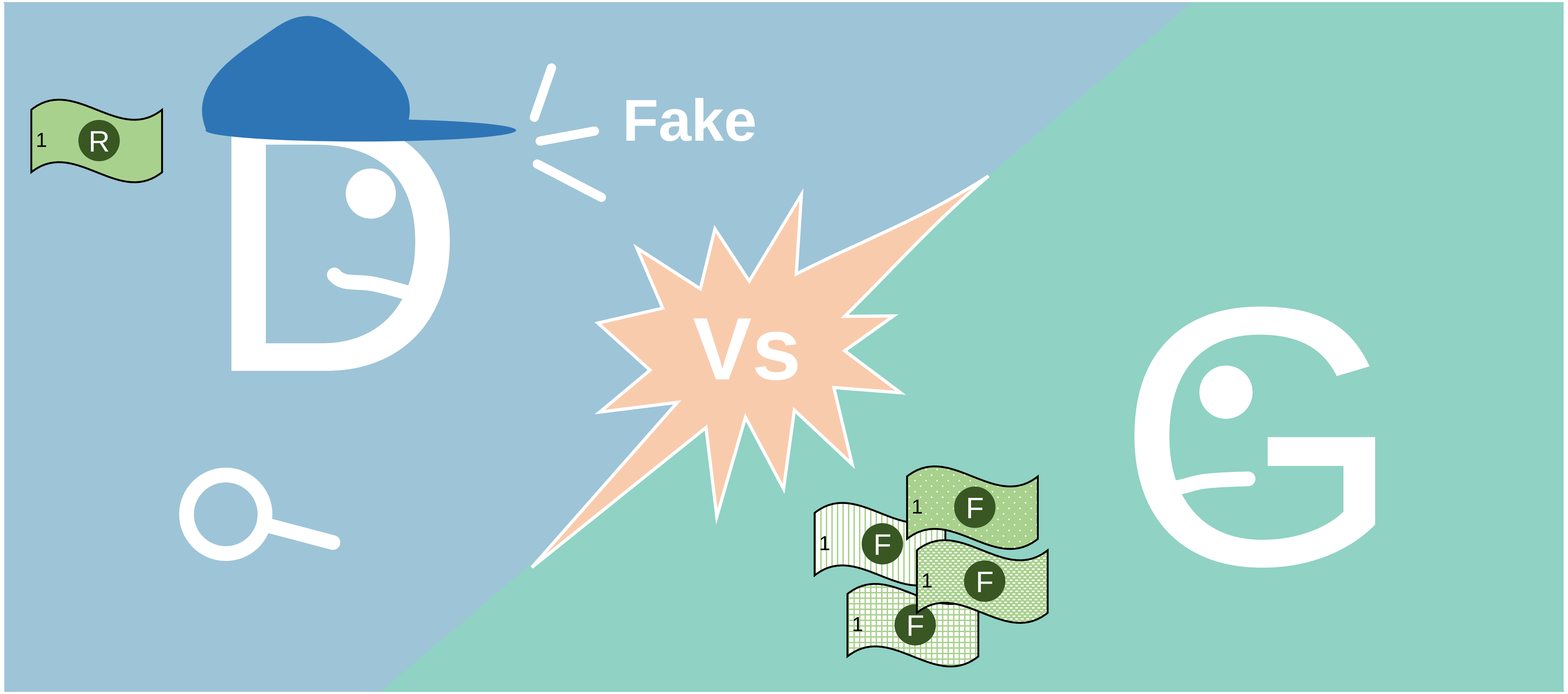
- Ian Goodfellow

DCGAN | LSGAN
F-GAN | BEGAN
InfoGAN | DiscoGAN
Unrolled GAN | CycleGAN
…

GAN "Generative adversarial nets."

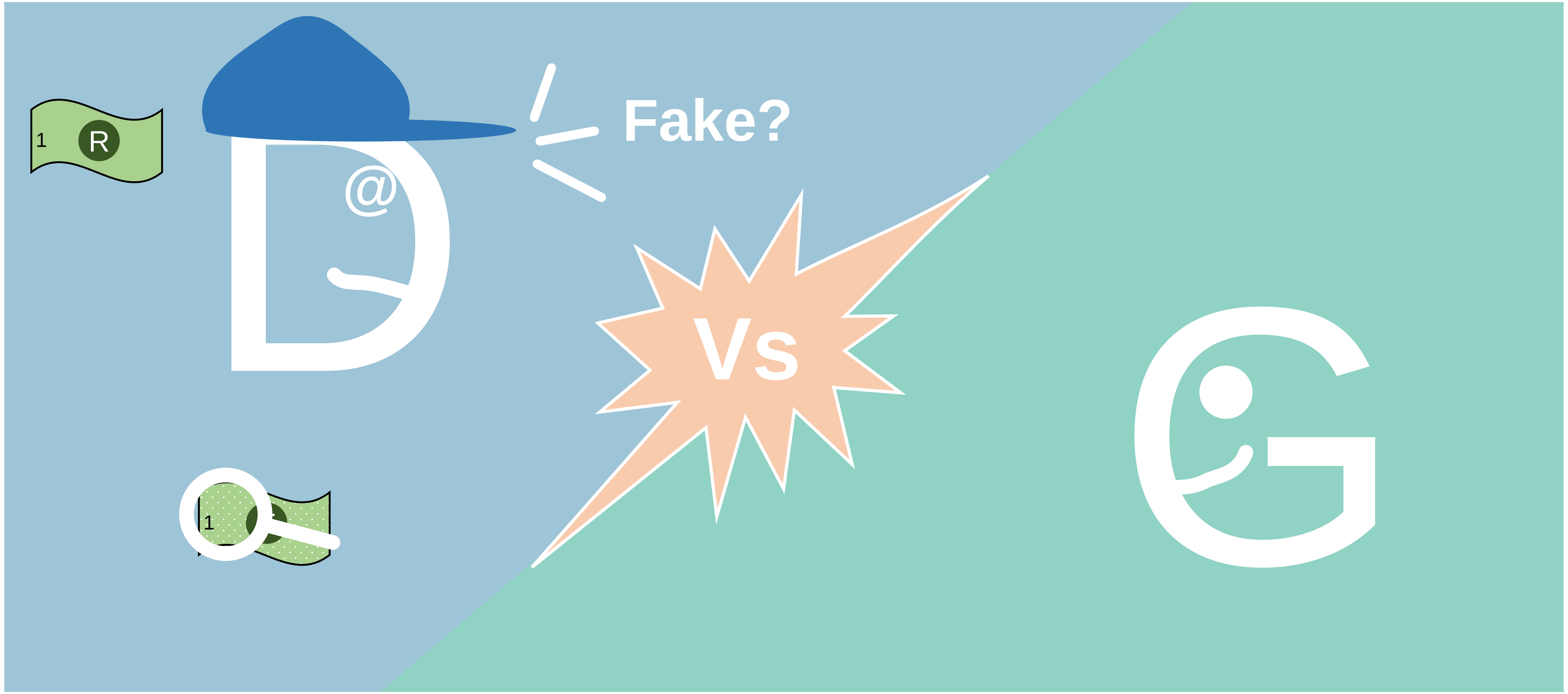
Introduction

- Concept of GAN



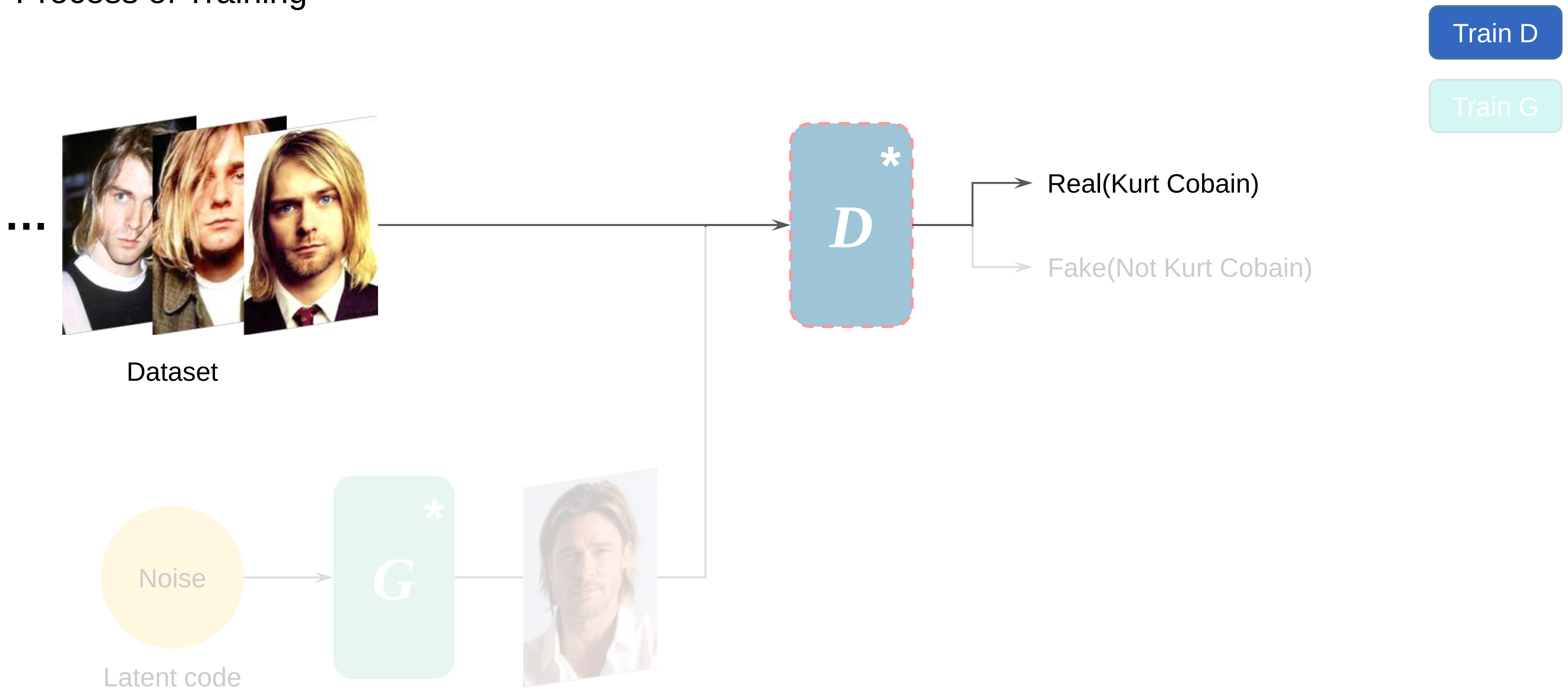
Introduction

- Concept of GAN



Introduction

- Process of Training

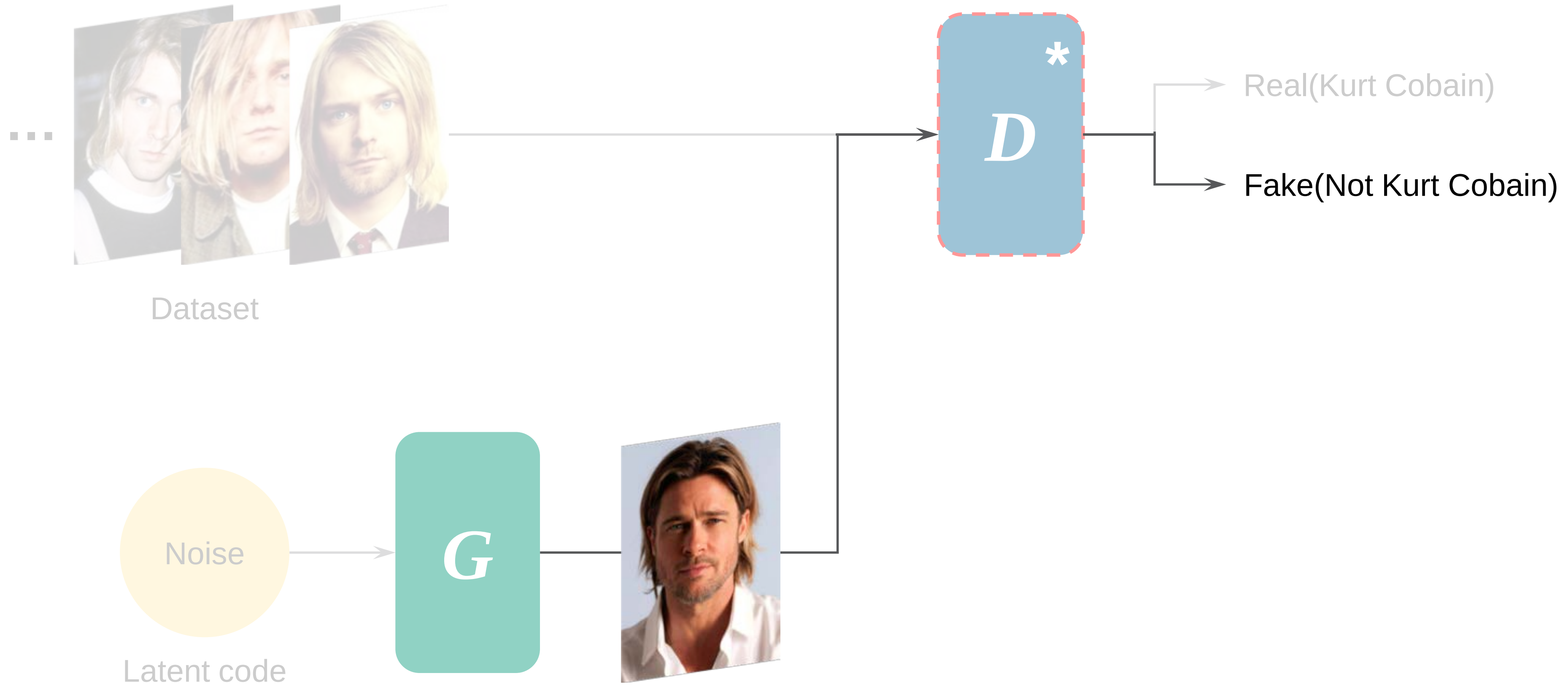


Introduction

- Process of Training

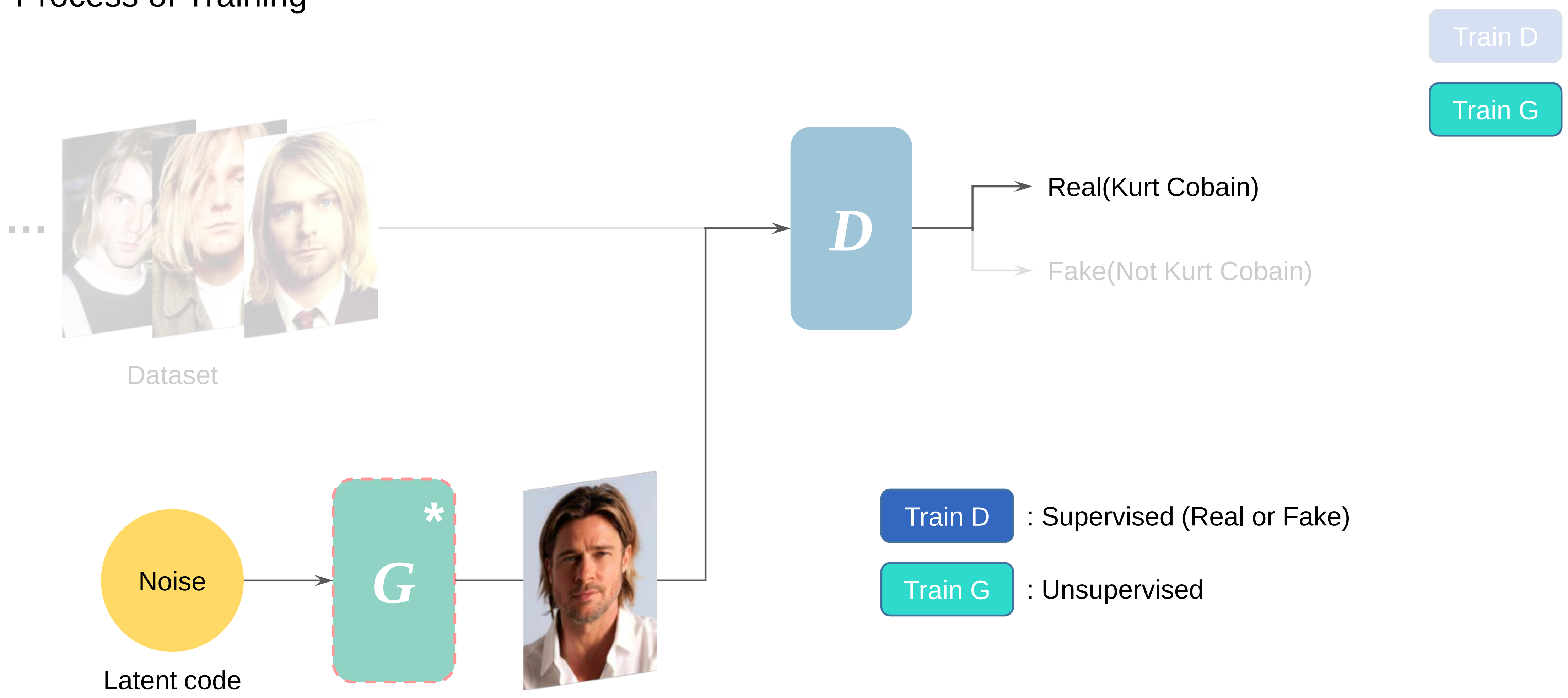
Train D

Train G



Introduction

- Process of Training



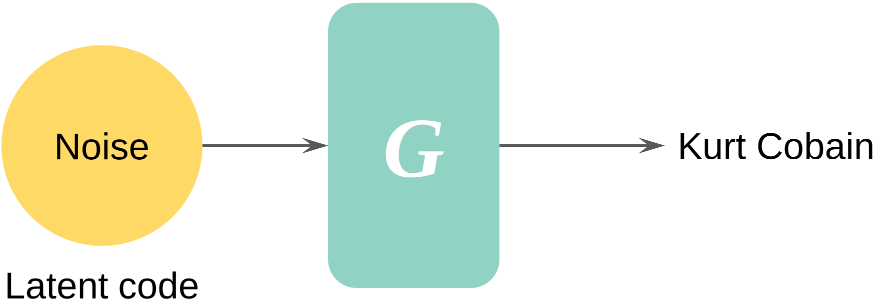
Introduction

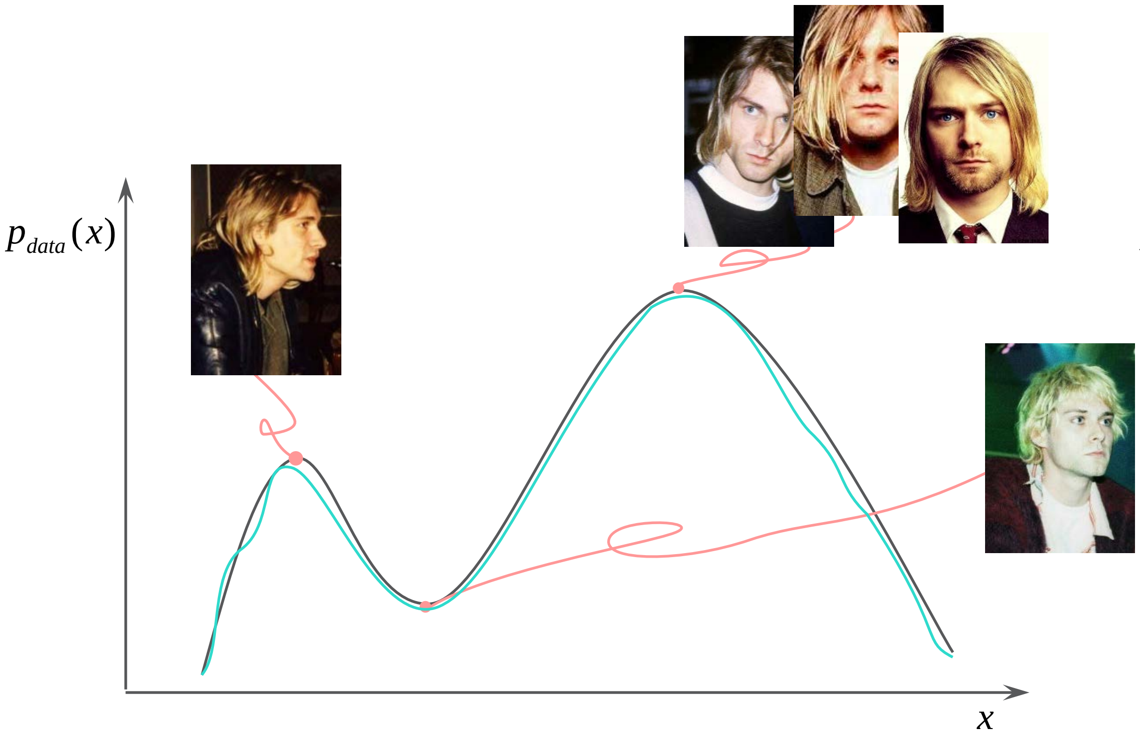
- Final Goal

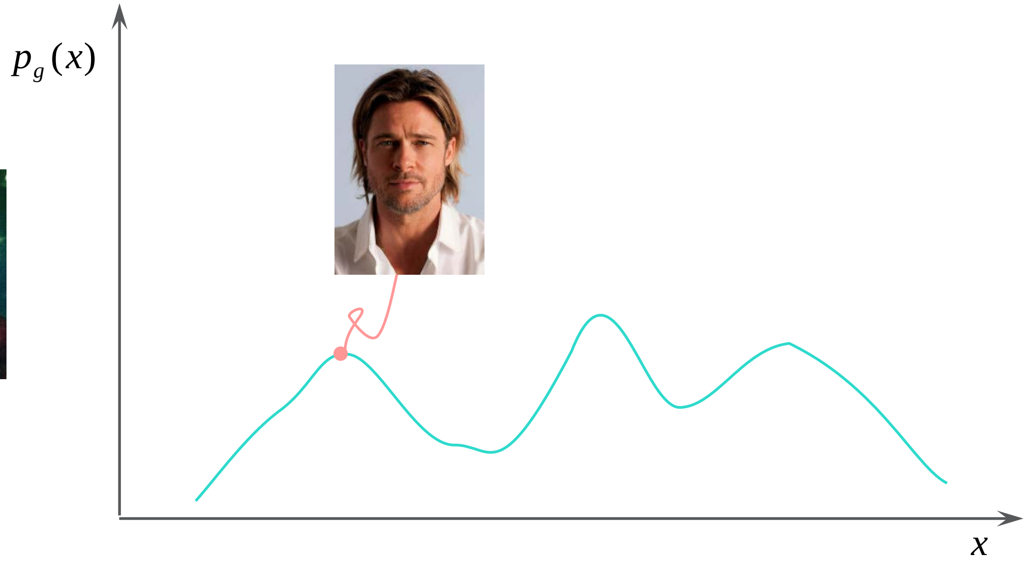
“Generative Adversarial Networks”

Goal

Method







Paper review

- Adversarial nets

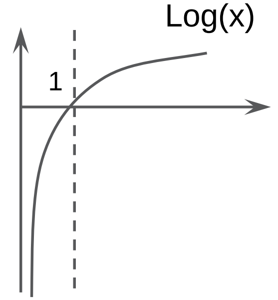
$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

Smart D

Real case $E_{x\sim p_{data}(x)}[\log \underset{\searrow 1}{D(x)}] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$ should be 0

Fake case $E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-\underset{\searrow 0}{D(G(z))})]$ should be 0

cf.



Stupid D

Real case $E_{x\sim p_{data}(x)}[\log \underset{\searrow 0}{D(x)}] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$ should be negative infinity

Fake case $E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-\underset{\searrow 1}{D(G(z))})]$ should be negative infinity

D perspective,
it should be maximum.

Paper review

- Adversarial nets

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

Generator

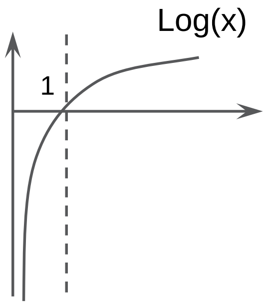
Smart G $\quad E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$ should be negative infinity

1

Stupid G $\quad E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$ should be 0

0

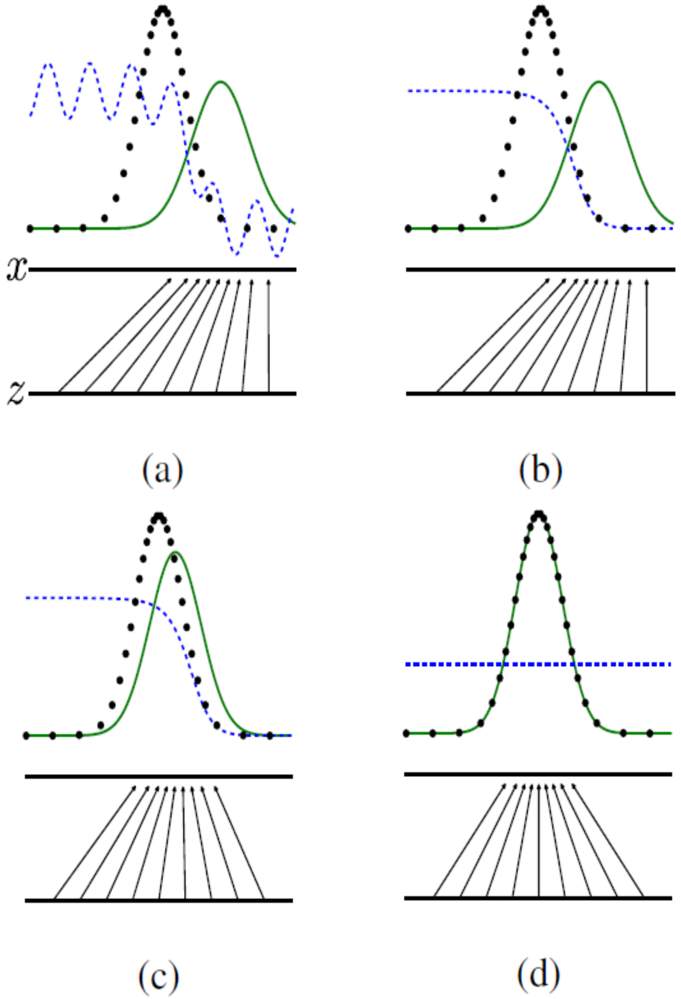
cf.



**G perspective,
it should be minimum.**

Paper review

- Adversarial nets



Algorithm 1

for number of training iterations **do**
  for k steps **do**
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

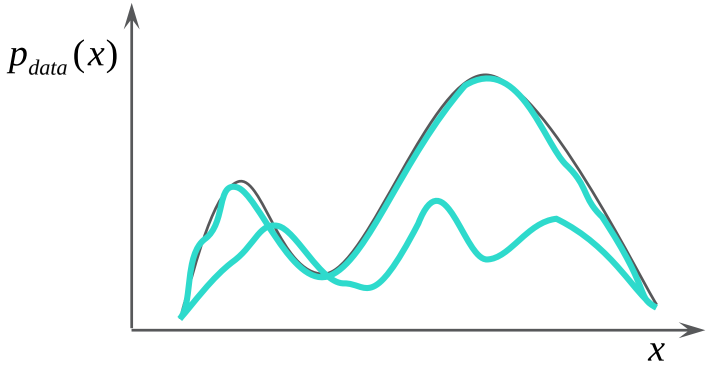
Paper review

- Theoretical Results

1) Global Optimality of $p_g = p_{data}$

Proposition 1. For G fixed, the optimal discriminator D is

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$



Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$

$$V(G,D) = E_{x \sim p_{data}}\left[\log\left(D\left(x\right)\right)\right] + E_{z \sim p_z}\left[\log\left(1 - D\left(G\left(z\right)\right)\right)\right]$$

$$= \int_x p_{data}(x)\log(D(x))dx + \int_z p_z(z)\log(1 - D(G(z)))dz$$

$$x = G(z)$$

$$= \int_x p_{data}(x)\log(D(x))dx + \int_x p_g(x)\log(1 - D(x))dx$$

$$= \int_x p_{data}(x)\log(D(x)) + p_g(x)\log(1 - D(x))dx$$

Paper review

- Theoretical Results cont.

1) Global Optimality of $p_g = p_{data}$

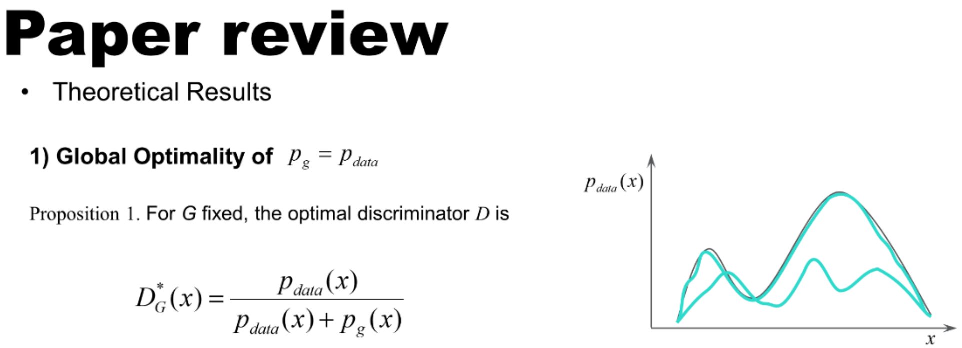
$$= \int_x p_{data}(x)\log(D(x)) + p_g(x)\log(1-D(x))dx$$

$$p_{data}(x)\log(D(x)) + p_g(x)\log(1-D(x)) \longrightarrow \text{Maximize}$$

Substitute $p_{data}(x) = a,\ p_g(x) = b,\ D(x) = y$

$$a\log y + b\log(1-y)$$

$$y = \frac{a}{a+b} \qquad D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$



Paper review

- Theoretical Results cont.

1) Global Optimality of $p_g = p_{data}$

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

$$= E_{x \sim p_{data}}\left[\log D_G^*(x)\right] + E_{z \sim p_z}\left[\log\left(1 - D_G^*(G(z))\right)\right]$$

$$= E_{x \sim p_{data}}\left[\log D_G^*(x)\right] + E_{x \sim p_g}\left[\log\left(1 - D_G^*(x)\right)\right]$$

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

Paper review

- Theoretical Results cont.

1) Global Optimality of $p_g = p_{data}$

Theorem 1. The global minimum of the virtual training criterion *C(G)* is achieved *if* and *only if* $p_g = p_{data}$ At that point, *C(G)* achieves the value –log4

Proof.

$$C(G) = E_{x\sim p_{data}}\left[\log D_G^*(x)\right] + E_{x\sim p_g}\left[\log(1 - D_G^*(x))\right]$$

$$= E\left[\log\frac{1}{2}\right] + E\left[\log(1-\frac{1}{2})\right] = \log\frac{1}{4} = -\log 4$$

$$C(G) = -\log 4 + \log 4 + E_{x\sim p_{data}}\left[\log\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}\right] + E_{x\sim p_g}\left[\log\frac{p_g(x)}{p_{data}(x) + p_g(x)}\right]$$

$$= -\log 4 + \log 2 + \log 2 + \sum_x p_{data}(x)\log\frac{p_{data}(x)}{p_{data}(x) + p_g(x)} + \sum_x p_g(x)\log\frac{p_g(x)}{p_{data}(x) + p_g(x)}$$

Paper review

- Theoretical Results cont.

1) Global Optimality of $p_g = p_{data}$

$$= -\log 4 + \log 2 + \log 2 + \sum_x p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} + \sum_x p_g(x) \log \frac{p_g(x)}{p_{data}(x) + p_g(x)}$$

$$= -\log 4 + \sum_x p_{data}(x) \log \frac{p_{data}(x)}{\frac{p_{data}(x) + p_g(x)}{2}} + \sum_x p_g(x) \log \frac{p_g(x)}{\frac{p_{data}(x) + p_g(x)}{2}}$$

$$= -\log 4 + KL\left(p_{data}(x) \,\|\, \frac{p_{data}(x) + p_g(x)}{2} \right) + KL\left(p_g(x) \,\|\, \frac{p_{data}(x) + p_g(x)}{2} \right)$$

$$= -\log 4 + 2JSD\left(p_{data}(x) \,\|\, p_g(x)\right) \quad if \;\; JSD = 0,\; then\; -\log 4$$

cf.

Kullback–Leibler divergence

$$KL(P \,\|\, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Jensen–Shannon divergence

$$JSD(P \,\|\, Q) = \frac{1}{2} KL(P \,\|\, M) + \frac{1}{2} KL(Q \,\|\, M)$$

Paper review

- Theoretical Results cont.

2) Convergence of Algorithm

Proposition 2. If *G* and *D* have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given *G*, and p_g is updated so as to improve the criterion

$$E_{x\sim p_{data}}\left[\log D_G^*(x)\right]+E_{x\sim p_g}\left[\log(1-D_G^*(x))\right]$$

then p_g converges to p_{data}

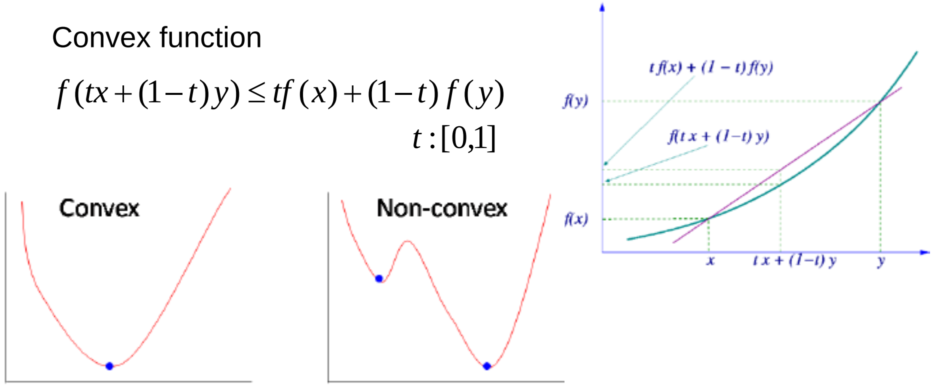
Paper review

- Theoretical Results cont.

cf.

Convex function

$$f(tx+(1-t)y) \le tf(x)+(1-t)f(y)$$
$$t:[0,1]$$



"No line segment lies below the graph at any point."

2) Convergence of Algorithm

Proof.

Consider $V(G,D)=U(p_g,D)$ as a functions of p_g as done in the above criterion. Note that $U(p_g,D)$ is convex in p_g

The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained.

if $f(x)=\sup_{\alpha\in A} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x)\in\partial f$ if $\beta=\arg\sup_{\alpha\in A} f_\alpha(x)$

g=data? g? $f(x)=\sup_{\alpha\in A} U_\alpha(p_g,D)$

This is equivalent to computing a gradient descent update for p_g at optimal D given the corresponding G

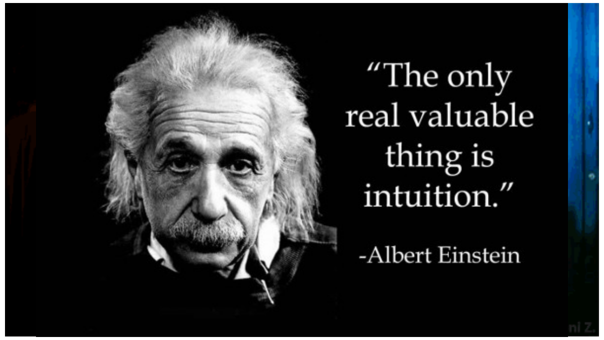
$\sup_D U(p_g, D)$ is convex in p_g with a unique global optima as proven in Thm 1, therefore with sufficiently small updates of p_g, p_g converges to p_{data}, concluding proof.

cf.

supremum

- 실수 *b*가 집합 *A*의 모든 원소들보다 크거나 같을때
  즉, *a*≤*b* 이면 *b*를 집합 *A*의 상계(Upper bound)라 한다. 단, *a*∈*A*
- 집합 *A*의 상계들의 집합을 *U*라 하면 *U*의 최소원소를 집합 *A*의 상한이라 하고 sup*A*라 쓴다.

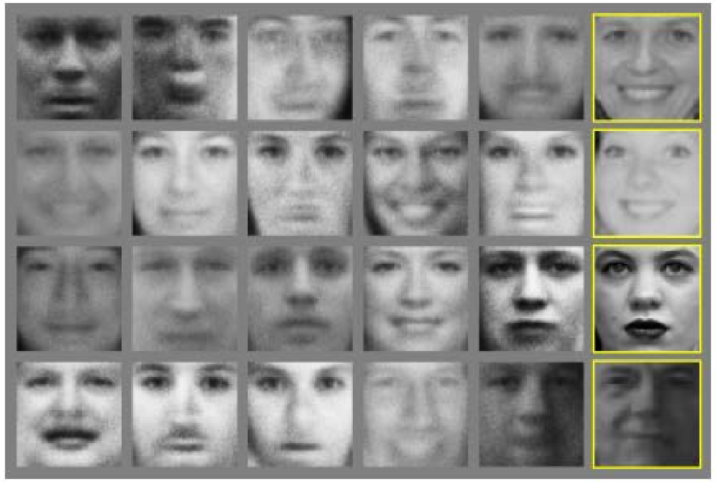
(http://mathnmath.tistory.com/27)



Paper review

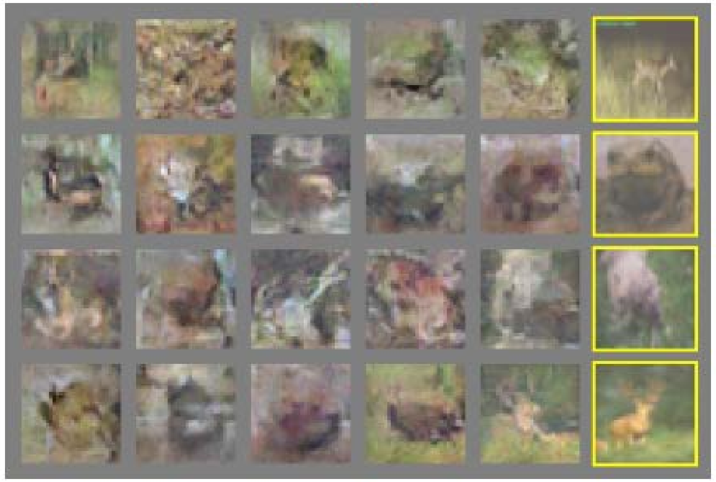
- Experiment

a)



b)

c)



d)

Configuration

- Deep Learning Framework

theano
Nov. 2010
Written in : Python
Interface : Python

Caffe
Dec. 2013
Written in : C++
Interface : Python, MATLAB, C++

torch
Jul. 2014
Written in : C, Lua
Interface : C, Lua

Keras
Mar. 2015
Written in : Python
Interface : Python, R

TensorFlow
Nov. 2015
Written in : C++, Python, CUDA
Interface : Python, C/C++, Java, Go, R, Julia

PYTORCH
Oct. 2016
Written in : Python, C, CUDA
Interface : Python

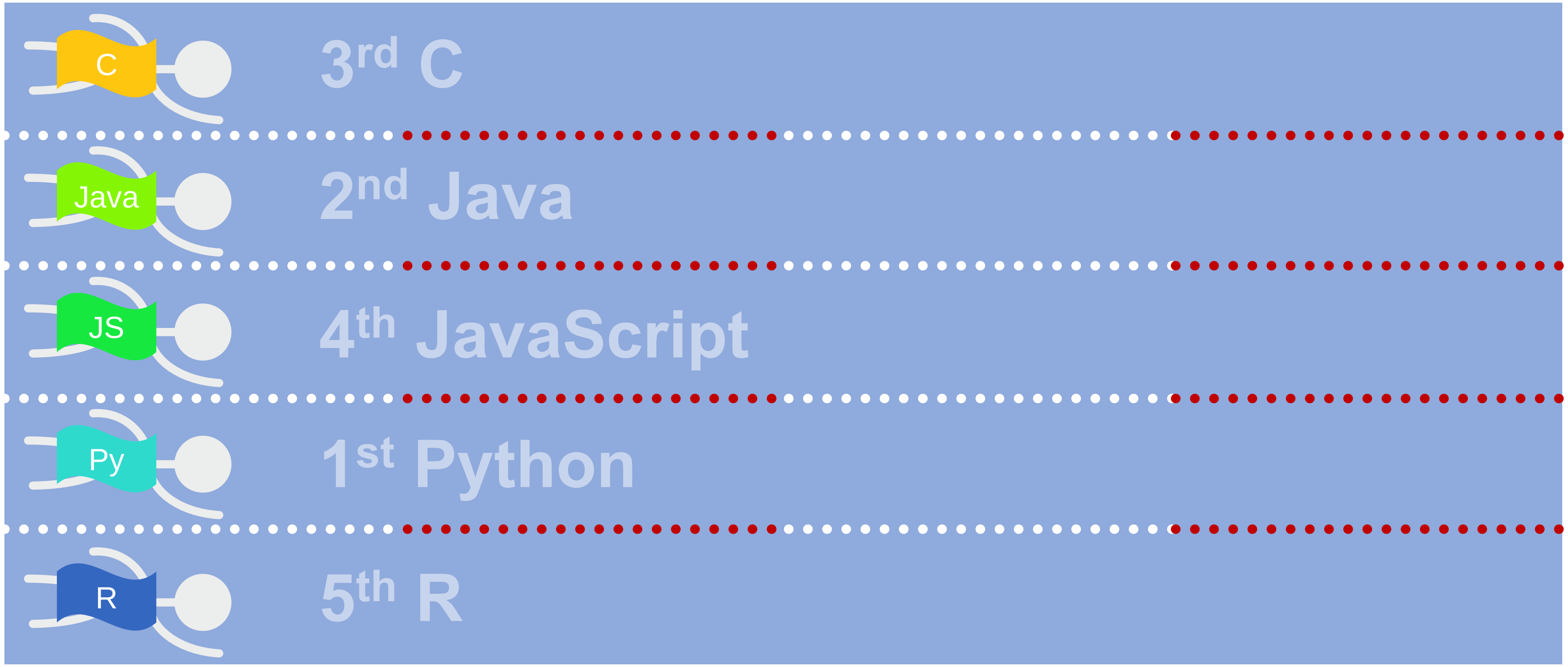
Recommend to choose these framework

Caffe2
Apr. 2017
Written in :
Interface : Python, C++

DL4J(Java)
Chainer(Python)
MXNet(C++, Python, Julia, MATLAB, JavaScript, Go, R, Scala, Perl)
CNTK(Python, C++),
TF Learn(Python)
TF-Slim(Python)
Etc.

Configuration

- Language



Elimination : Lua Julia Swift

Configuration

- Development Tool

: Intellisense

: Cell based execution

: Startup file

: Intellisense
: Management of python env.
: GitHub
~~: AI tool package~~

~~: Extension program~~

: Intellisense
: Cell based execution
: Management of python env.

: Intellisense
: Environment setting

: Insane extension program

Configuration

- CM(Configuration Management) Tool

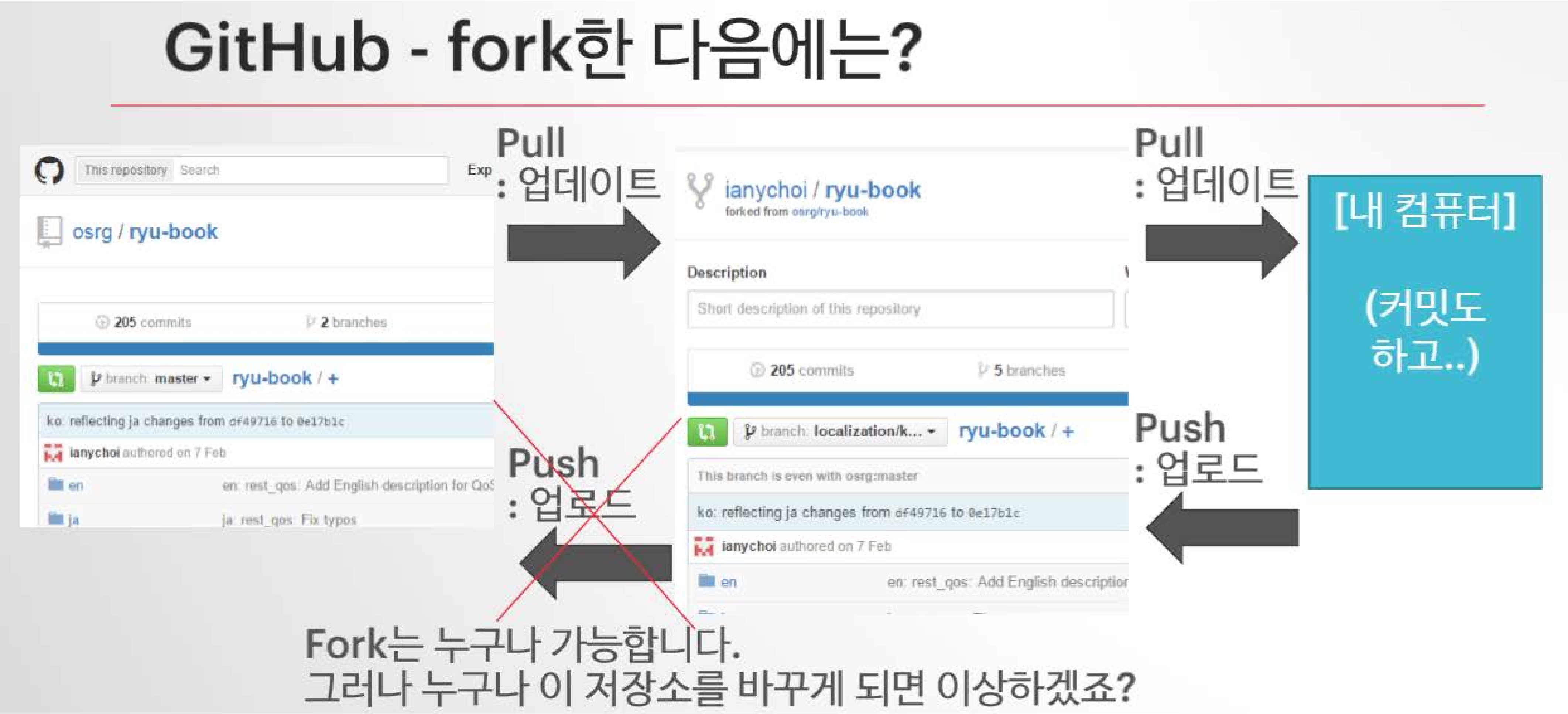
GitHub

Very personal opinion

- **Everywhere** ★★★
- Collaborate on code
- Version management
- **Markdown(.md)** ★★★☆
- **Integration with development tools.** ★★★★

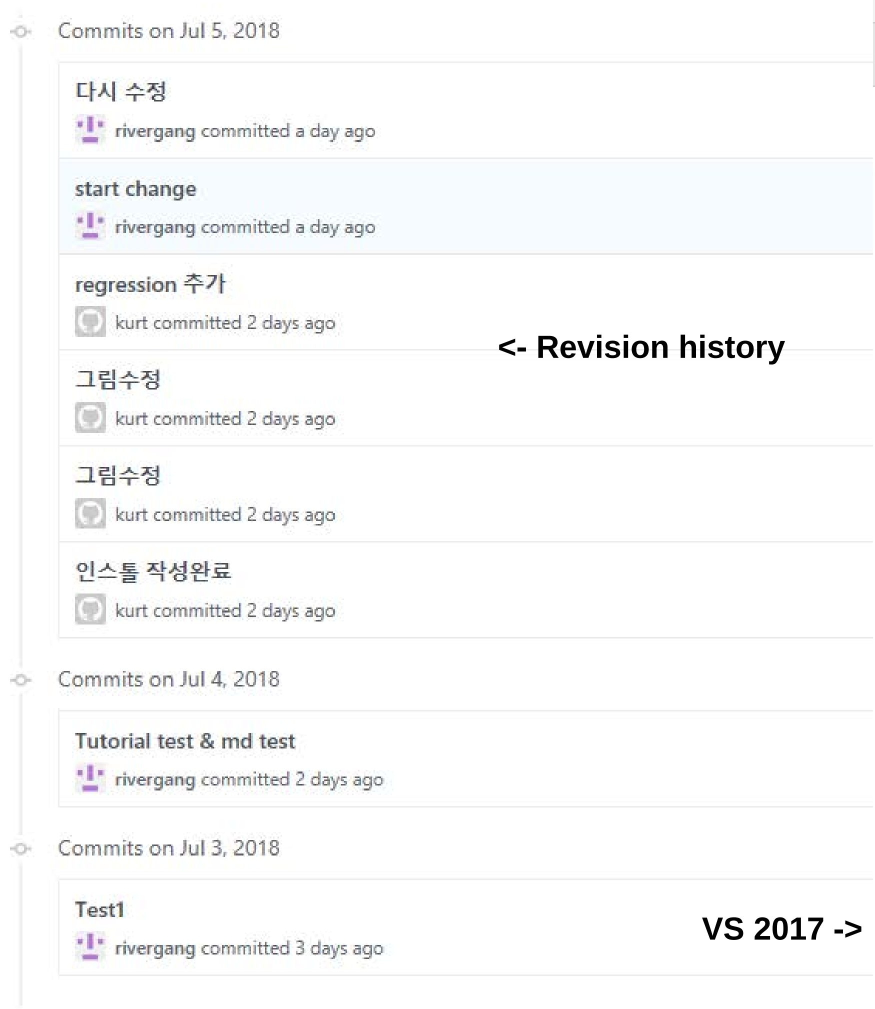
Configuration

- CM(Configuration Management) Tool



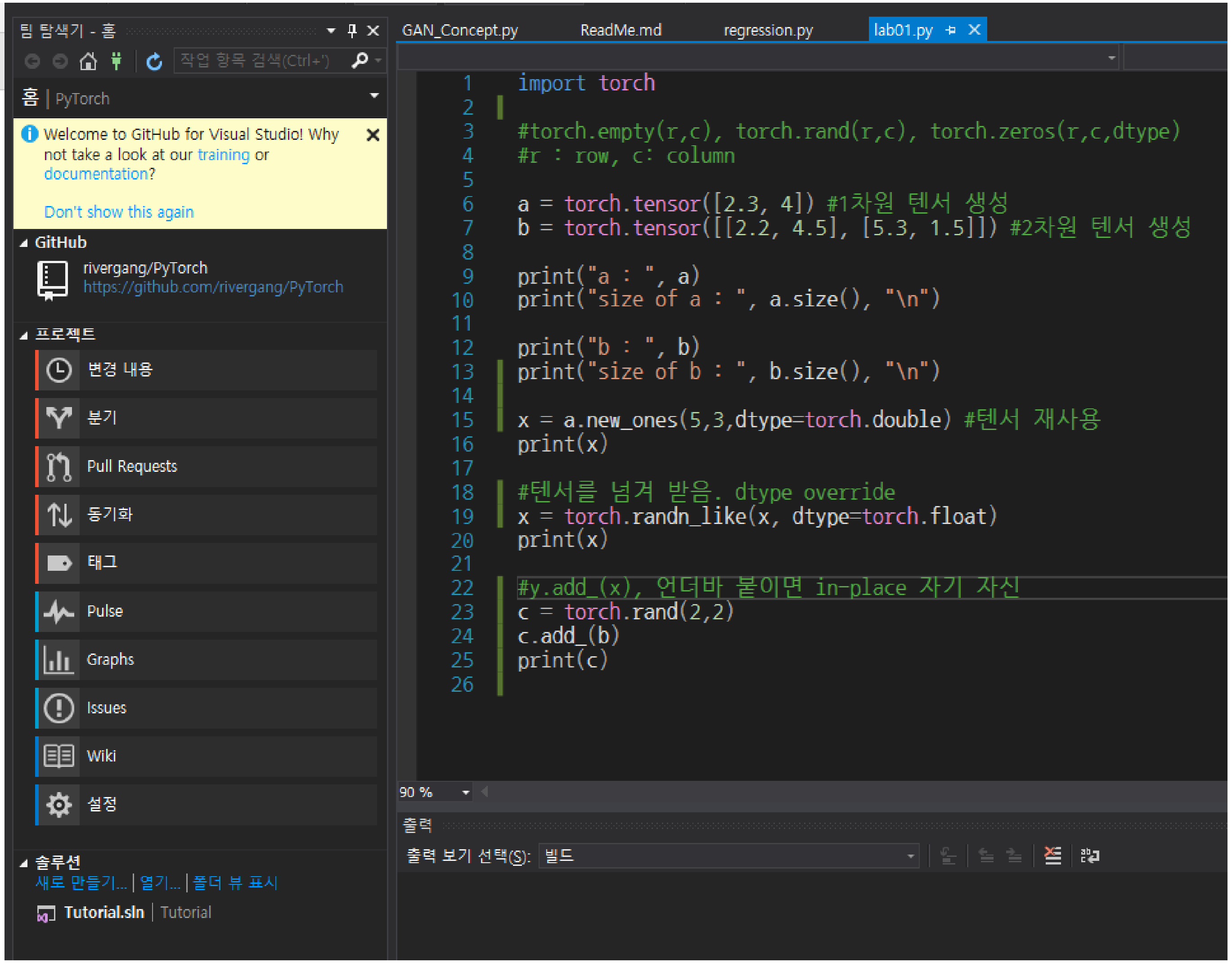
Configuration

- CM(Configuration Management) Tool



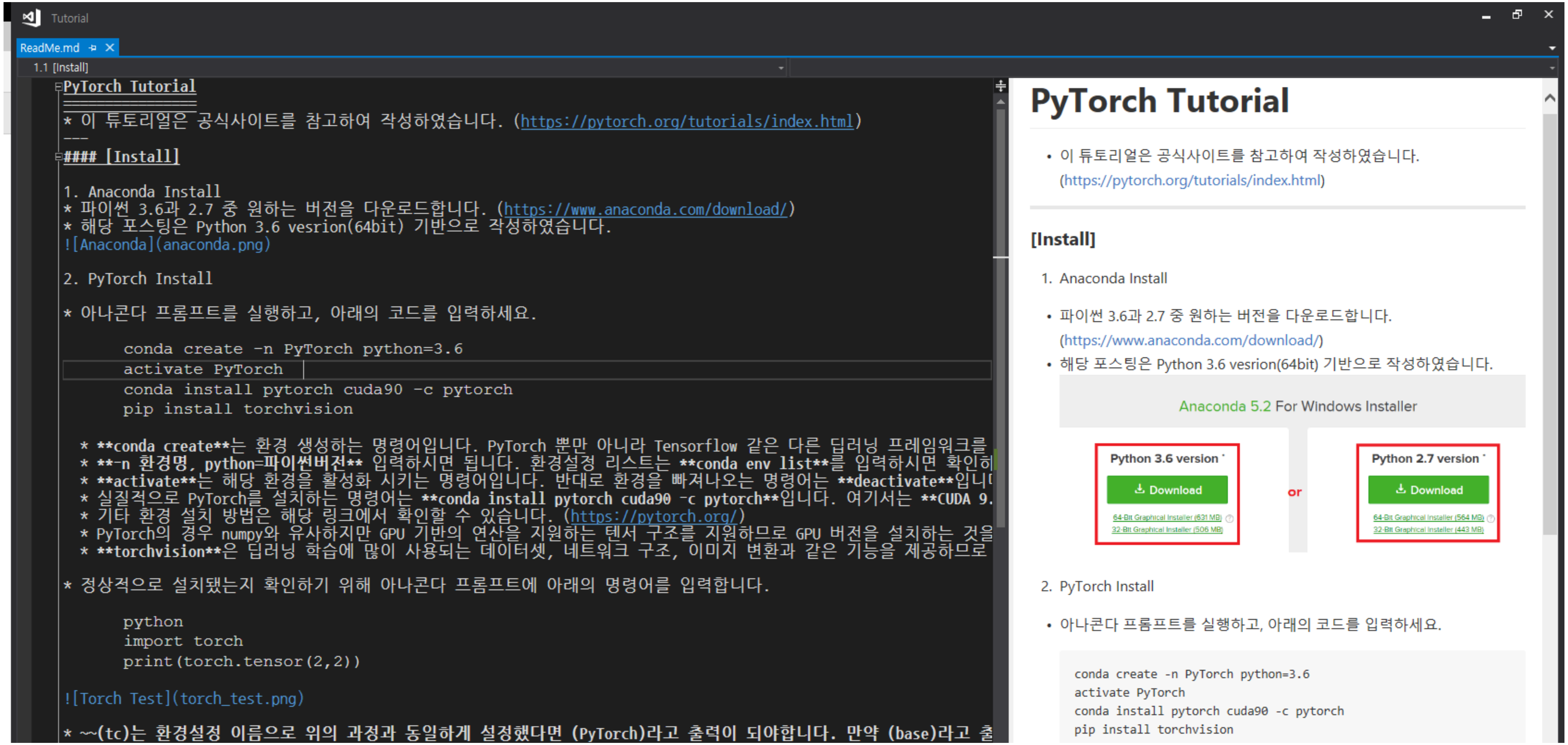
<- Revision history

VS 2017 ->



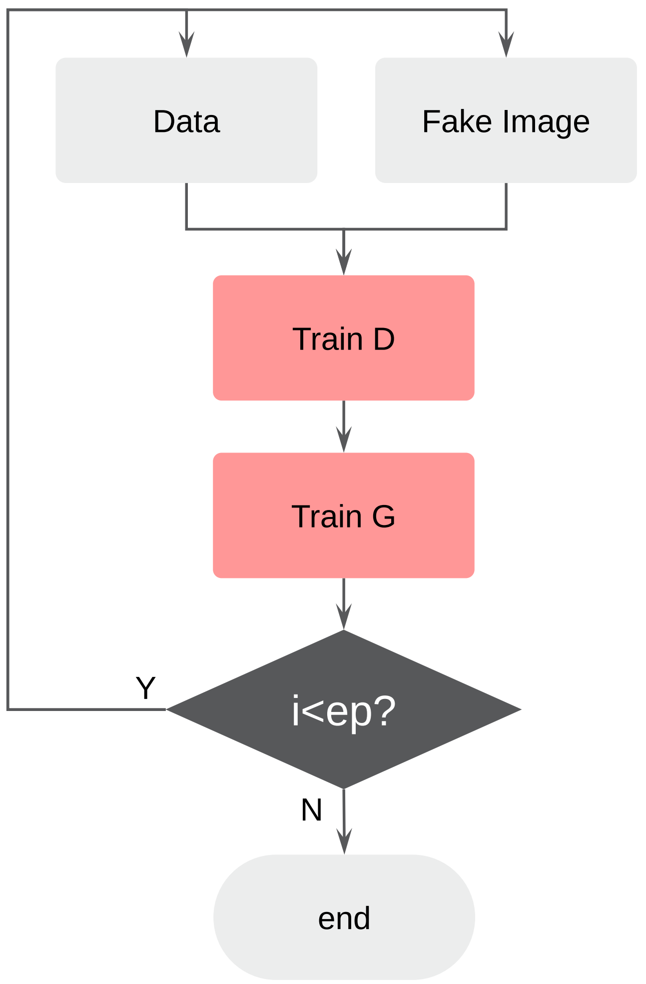
Configuration

- CM(Configuration Management) Tool



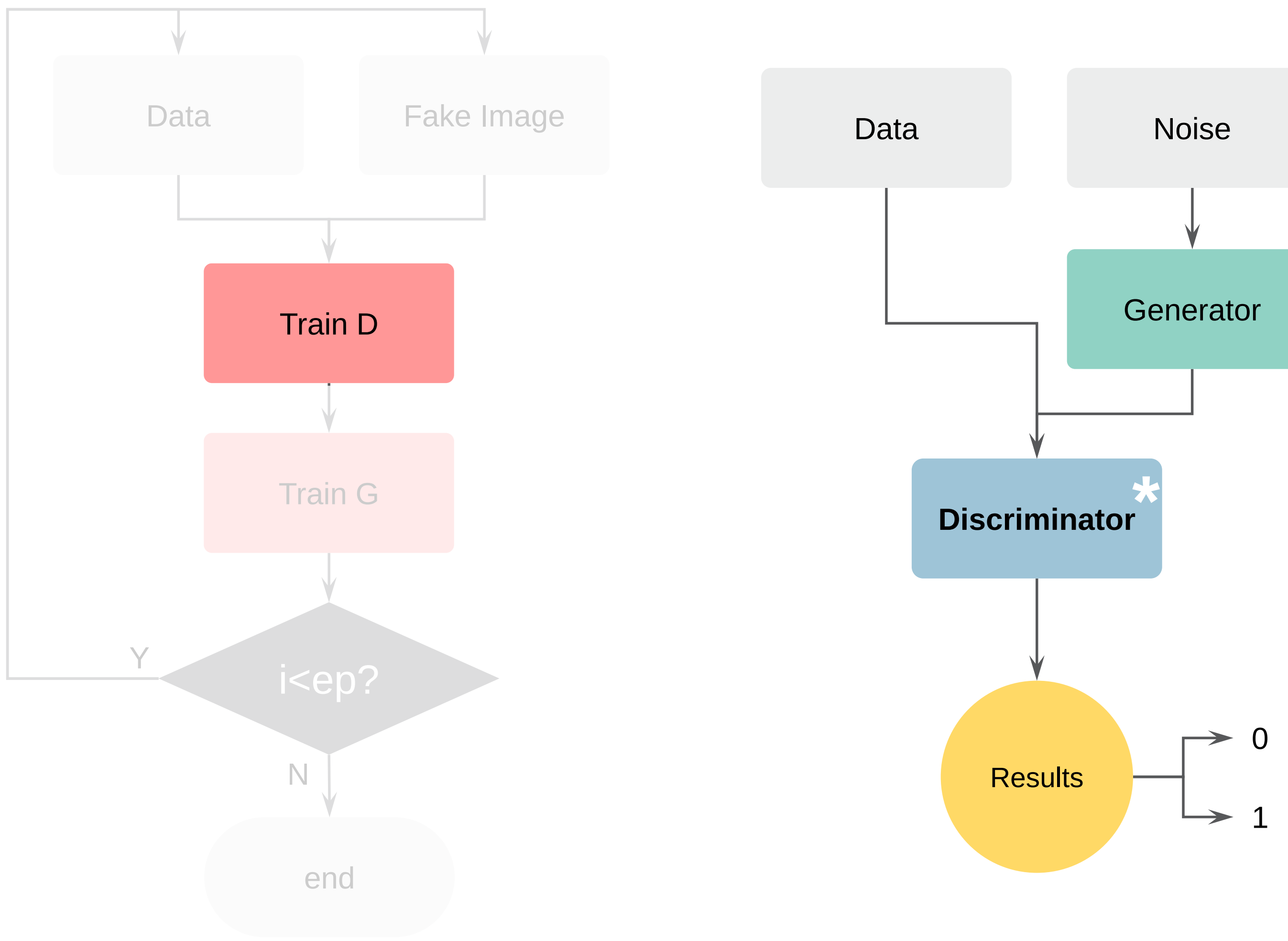
Experiment

- Flowchart



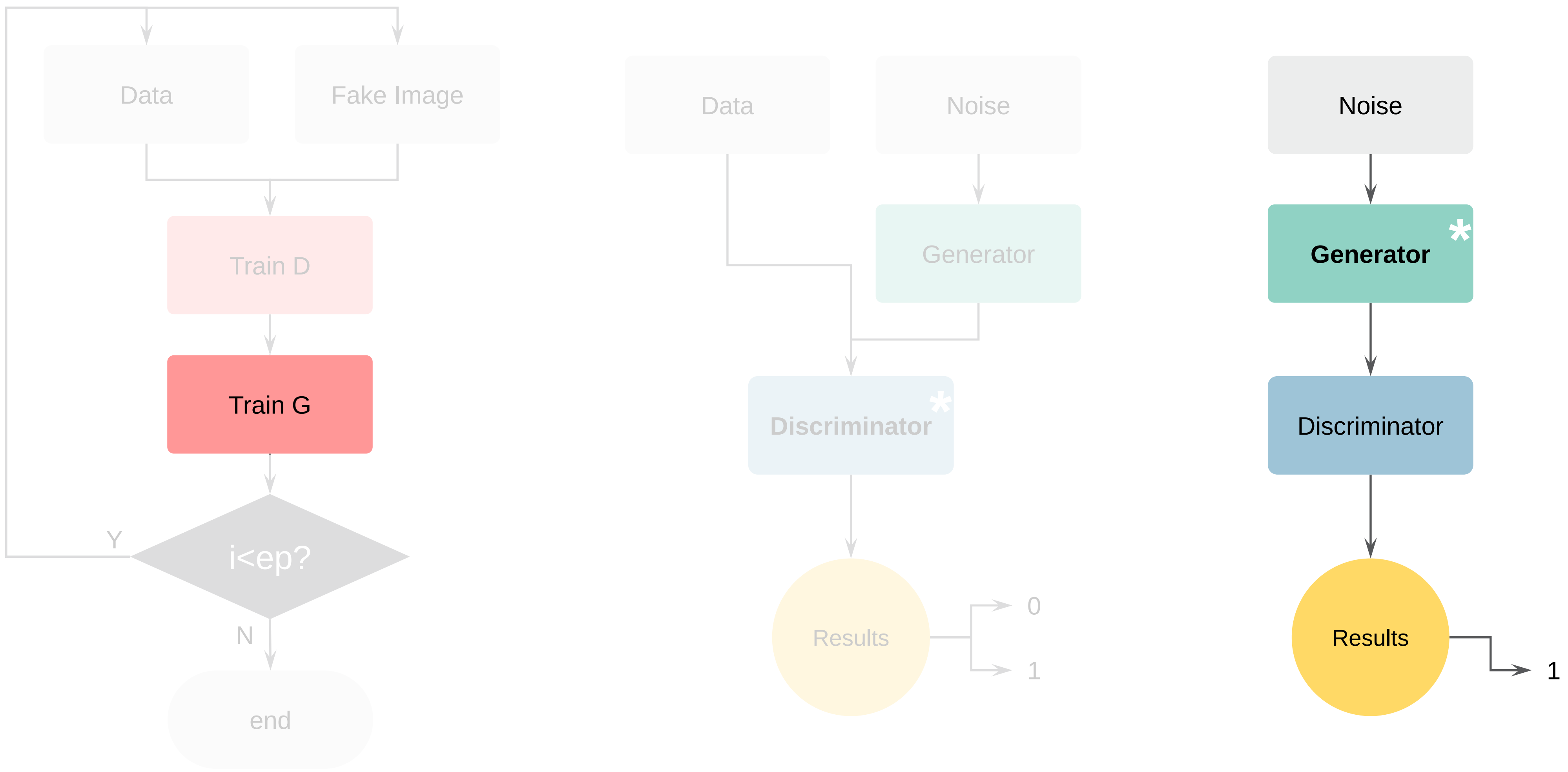
Experiment

- Flowchart



Experiment

- Flowchart



Experiment

- Implementation

Import	Generator
Parameter	GPU
Data load	Optimizer
Range	Train the D
Discriminator	Train the G

Import

필요한 라이브러리를 import한다.

- torch : tensor나 network 구조를 구현하기 위한 라이브러리
- torchvision : dataset을 관리하는데 필요한 라이브러리
- os : file path를 불러오기 위한 라이브러리

```
import torch as tc
import torch.nn as nn
import torchvision
import torchvision.transforms as transforms
from torchvision.utils import save_image
import os
```

Parameter

학습에 필요한 이미지 크기 및 저장 경로, hyper parameter를 설정한다.

- result_path : 저장되는 경로이다. 해당 파일인 'GAN_Simple.py'와 같은 경로에 'simple'이라는 폴더가 존재해야한다.**(조절 가능)**
- img_sz : 이미지 크기이다. 여기서는 MNIST이므로(28x28x1)의 크기를 사용한다.
- noise_sz : Generator의 입력으로 주어지는 latent code의 크기이다.**(조절 가능)**
- hidden_sz : Hidden Layer의 크기이다.**(조절 가능)**
- batch_sz : 배치 크기이다. **(조절 가능. MNIST의 총 데이터 개수가 60,000이므로 이에 나누어 떨어지게 조절해야한다.)**
- nEpoch : 에폭 횟수이다.**(조절 가능)**
- nChannel : 채널 크기이다. MNIST이므로 1이다.
- lr : 학습률(Learning Rate)이다. **(조절 가능)**

```
result_path = 'simple'
img_sz = 784
noise_sz = 100
hidden_sz = 512
batch_sz = 100
nEpoch = 300
nChannel = 1
```

(https://github.com/rivergang/PyTorch_GAN)

Experiment

- Implementation(Import, Parameter, Data load)

Experiment

- Implementation(Range, Discriminator, Generator)

Experiment

- Implementation (GPU, Optimizer)

```
loss_func = tc.nn.BCELoss()
d_opt = tc.optim.Adam(D.parameters(), lr=lr)
g_opt = tc.optim.Adam(G.parameters(), lr=lr)
```

Experiment

- Implementation (Train the D)

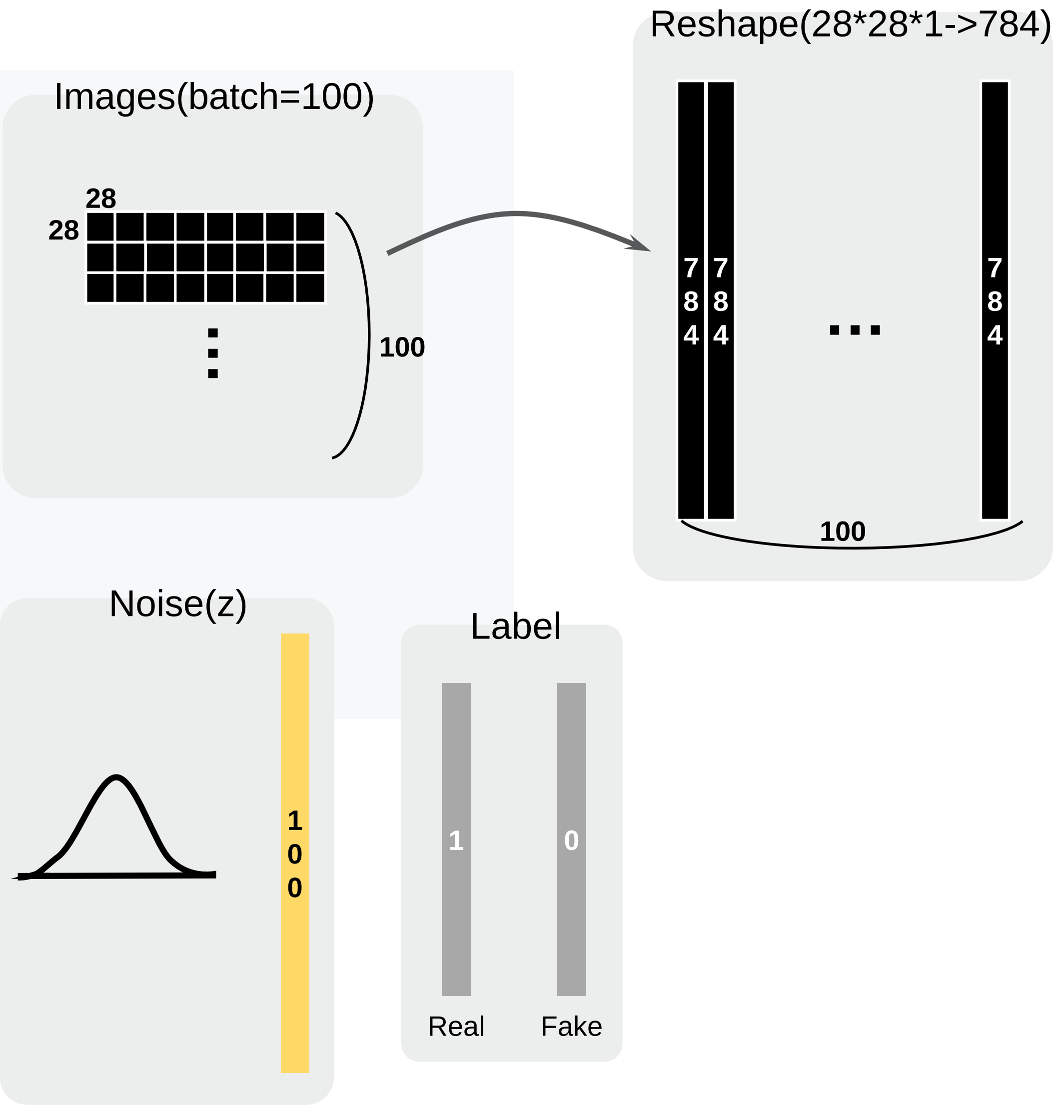
```
for ep in range(nEpoch):
    for step, (images, _) in enumerate(dataloader):
        images = images.reshape(batch_sz, -1).to(device)
        z = tc.randn(batch_sz, noise_sz).to(device)

        real_label = tc.ones(batch_sz, 1).to(device)
        fake_label = tc.zeros(batch_sz, 1).to(device)

        loss_real = loss_func(D(images), real_label)
        loss_fake = loss_func(D(G(z)), fake_label)

        d_loss = loss_real + loss_fake

        d_opt.zero_grad()
        d_loss.backward()
        d_opt.step()
```



Experiment

- Implementation (Train the D) - cont.

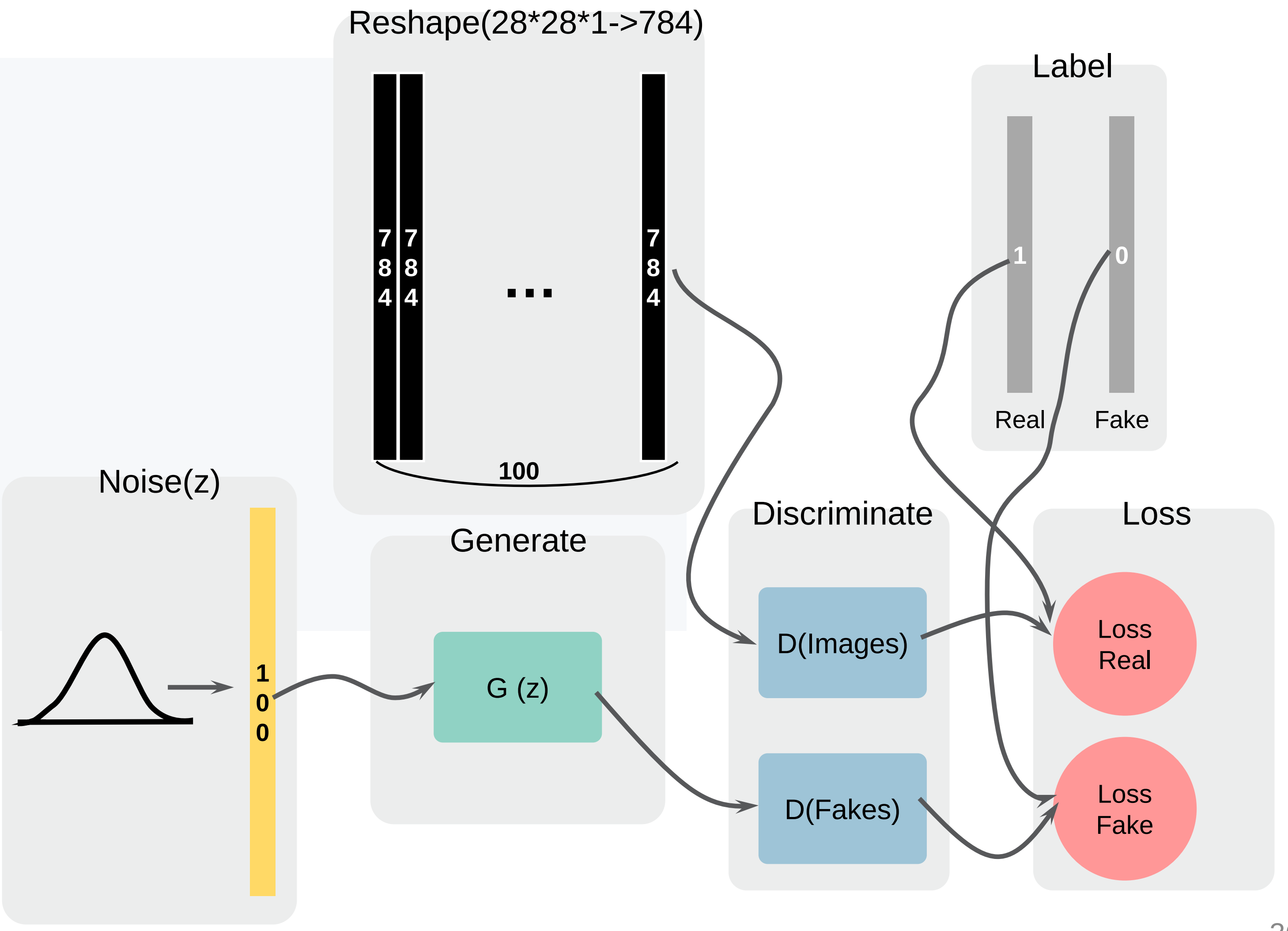
```python
for ep in range(nEpoch):
    for step, (images, _) in enumerate(dataloader):
        images = images.reshape(batch_sz, -1).to(device)
        z = tc.randn(batch_sz, noise_sz).to(device)

        real_label = tc.ones(batch_sz, 1).to(device)
        fake_label = tc.zeros(batch_sz, 1).to(device)

        loss_real = loss_func(D(images), real_label)
        loss_fake = loss_func(D(G(z)), fake_label)

        d_loss = loss_real + loss_fake

        d_opt.zero_grad()
        d_loss.backward()
        d_opt.step()
```



Experiment

- Implementation (Train the D) - cont.

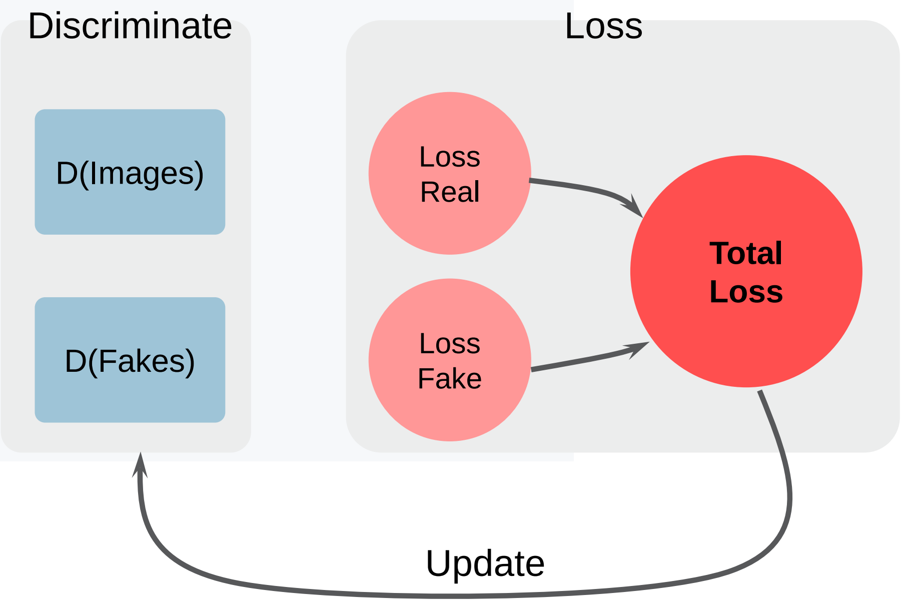
```
for ep in range(nEpoch):
    for step, (images, _) in enumerate(dataloader):
        images = images.reshape(batch_sz, -1).to(device)
        z = tc.randn(batch_sz, noise_sz).to(device)

        real_label = tc.ones(batch_sz, 1).to(device)
        fake_label = tc.zeros(batch_sz, 1).to(device)

        loss_real = loss_func(D(images), real_label)
        loss_fake = loss_func(D(G(z)), fake_label)

        d_loss = loss_real + loss_fake

        d_opt.zero_grad()
        d_loss.backward()
        d_opt.step()
```

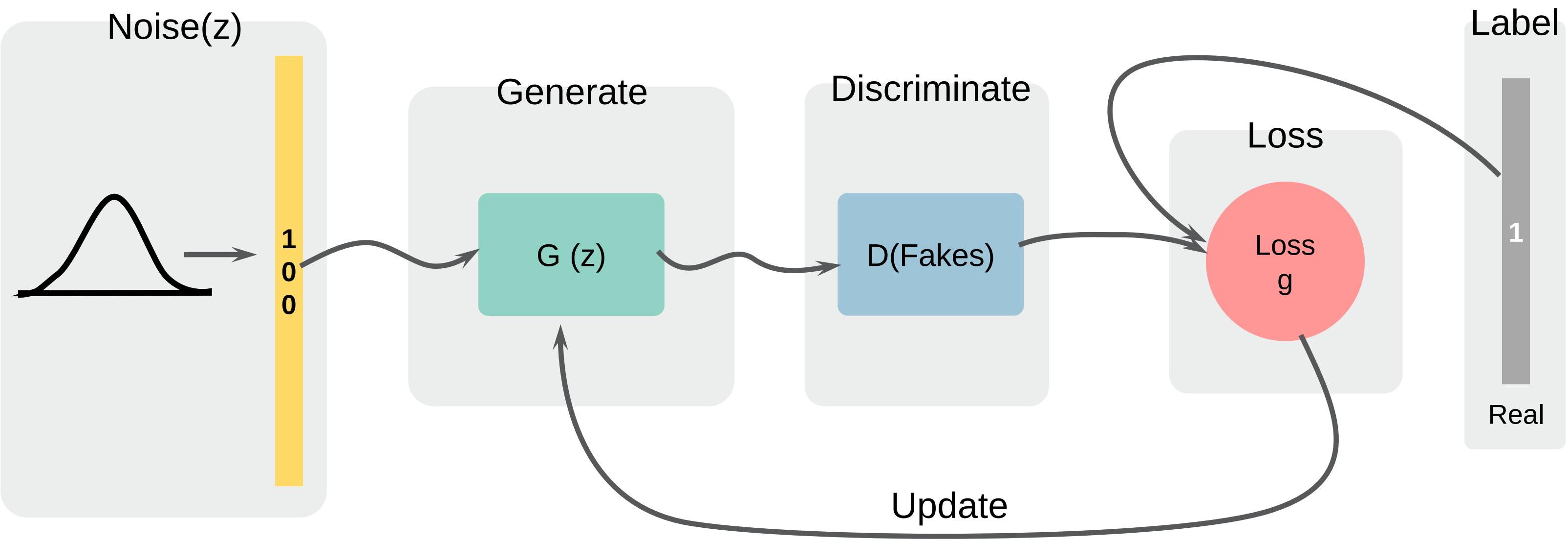


Experiment

- Implementation(Train the G)

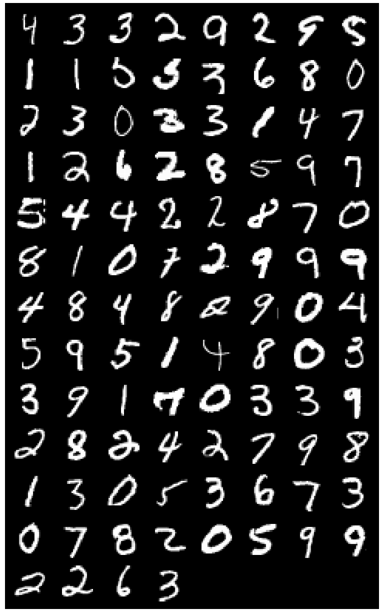
```
fake_images = G(z)
g_loss = loss_func(D(fake_images), real_label)

g_opt.zero_grad()
g_loss.backward()
g_opt.step()
```

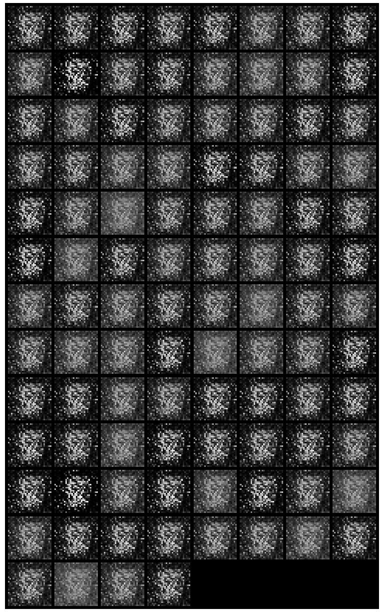


Experiment

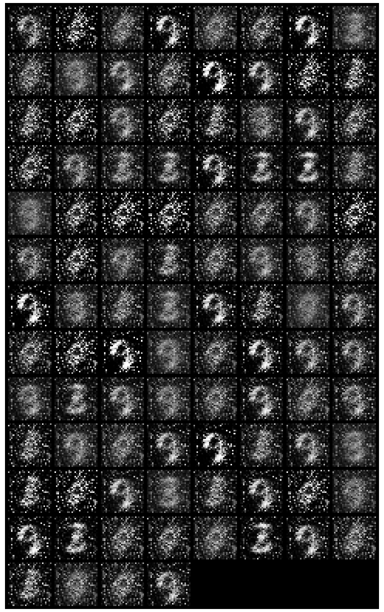
- Result#1 (MNIST)



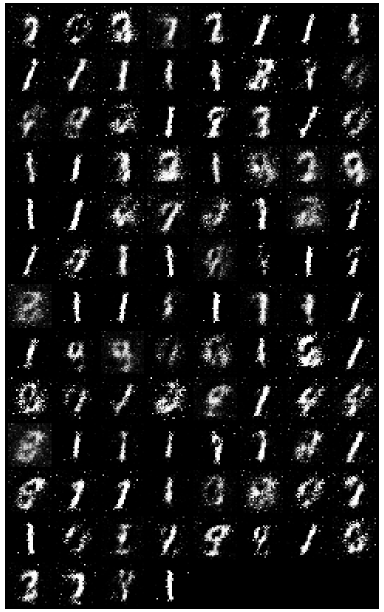
Real



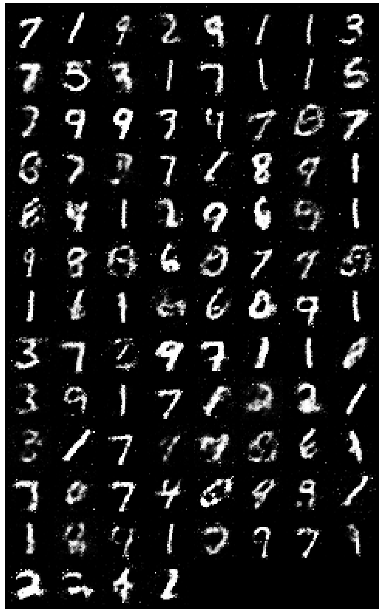
1



10



30



100

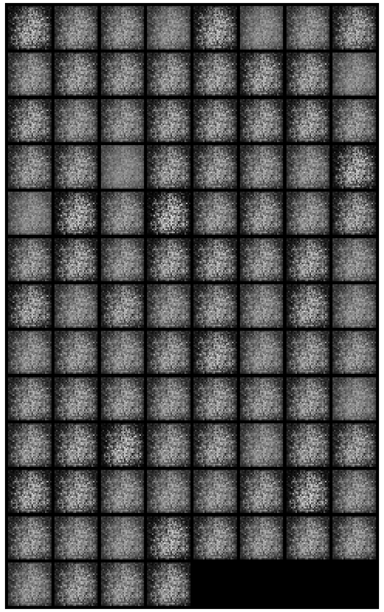
200

400

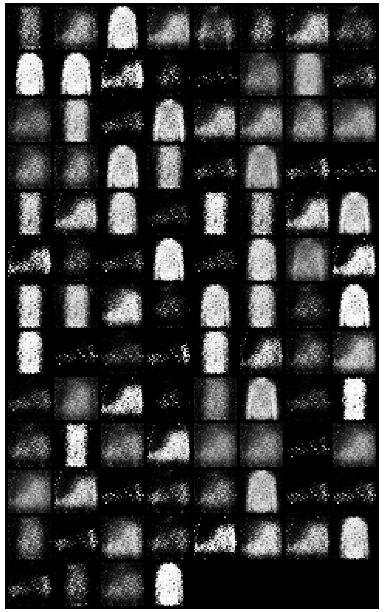
Experiment

- Result#2 (FashionMNIST)

Real

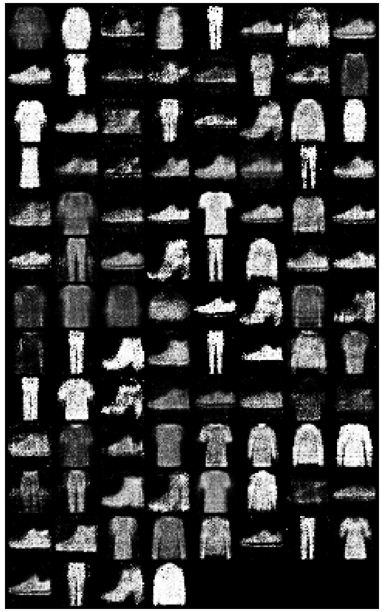


1

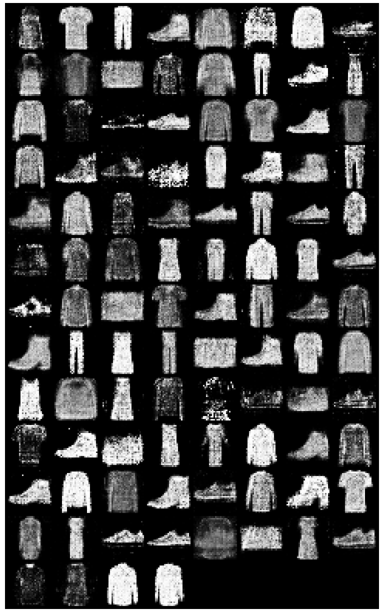


10

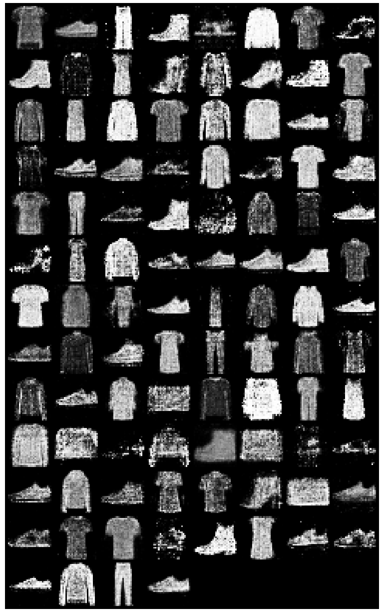
30



100



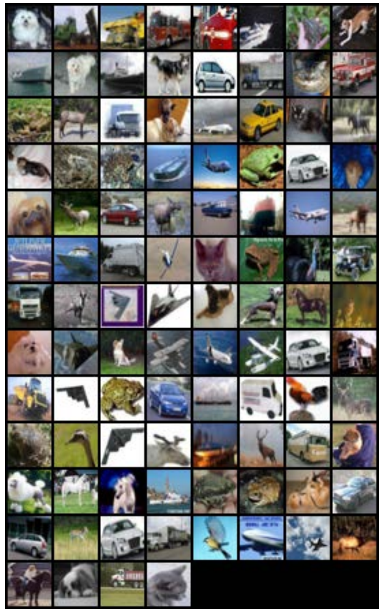
200



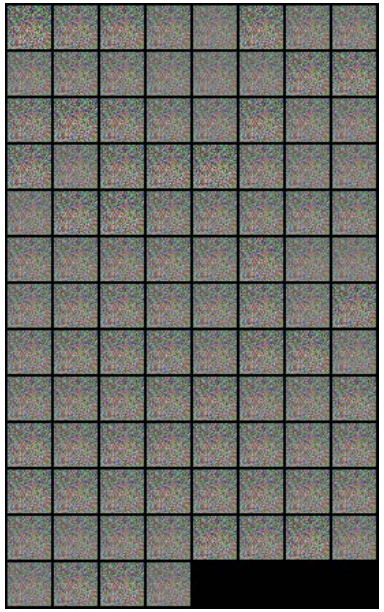
400

Experiment

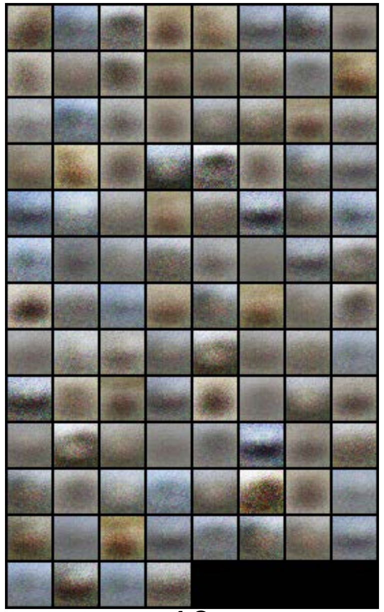
- Result#3 (CIFAR10)



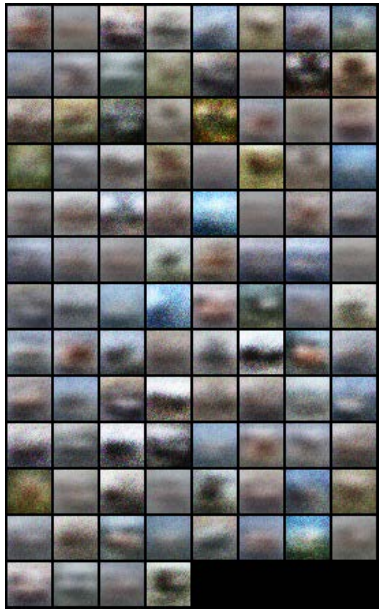
Real



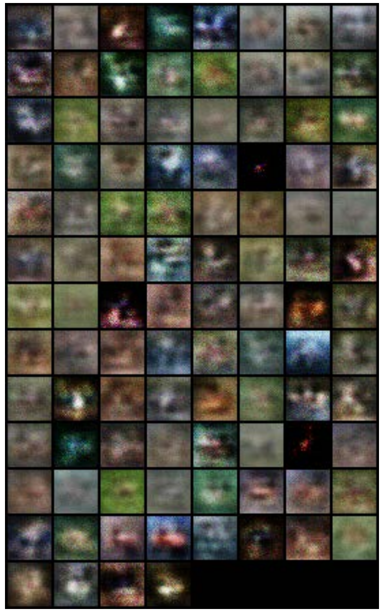
1



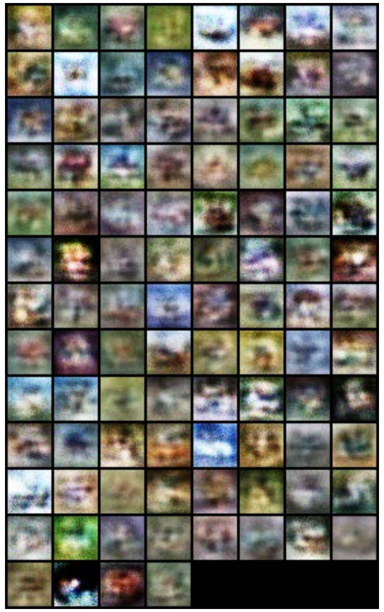
10



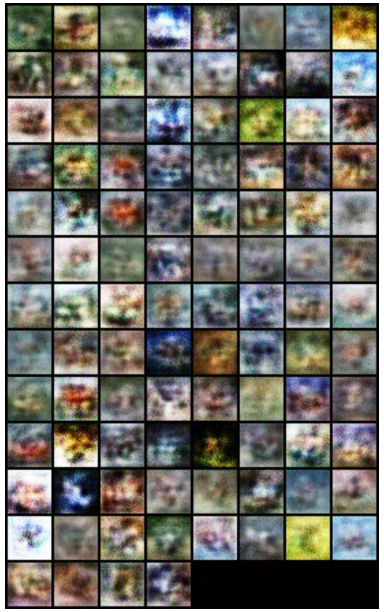
30



100



200



400

Summary

- Generative Adversarial Network is composed the generator model and the discriminator.
- When training the discriminator, the parameters of the generator should be fixed and vice versa.
- The global minimum of the training criterion is achieved if and only if $p_g = p_{data}$. **Global optimality**
- The generative distribution converges to the data distribution. **Convergence of algorithm**

Future work & Reference

- DCGAN *(based Conv. layer, optimal training network)*
- ~~AE, VAE *(encoder & decoder)*~~
- InfoGAN *(meaning of latent vector)*
- Unrolled GAN *(problem of instability)*
- LSGAN *(Loss)*
- Wasserstein GAN *(Wasserstein distance)*
- BEGAN *(equilibrium concept)*
- ~~Pix2Pix *(mapping)*~~
- Disco GAN *(cross domain relation)*
- Cycle GAN *(cross domain relation)*

- f-GAN
- Energy based GAN
- ~~U-Net~~
- ~~ResNet~~
- …

Q & A

Thank you for your attention

Reference

[1] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

[2] Wang, Su. "Generative Adversarial Networks (GAN) A Gentle Introduction."

[2] 초짜 대학원생의 입장에서 이해하는 Generative Adversarial Networks (https://jaejunyoo.blogspot.com/)

[3] 1시간만에 GAN(Generative Adversarial Network) 완전 정복하기 (https://www.slideshare.net/NaverEngineering/1-gangenerative-adversarial-network)

[4] 프레임워크 비교(https://deeplearning4j.org/kr/compare-dl4j-torch7-pylearn)

[5] AI 개발에AI 개발에 가장 적합한 5가지 프로그래밍 언어 (http://www.itworld.co.kr/news/109189#csidxf9226c7578dd101b41d03bfedfec05e)

[6] Git는 머꼬? GitHub는 또 머지?(https://www.slideshare.net/ianychoi/git-github-46020592)

[7] svn 능력자를 위한 git 개념 가이드(https://www.slideshare.net/einsub/svn-git-17386752)

Appendix

- What is Pythonic?

1. Collection이 있는 리스트에 대해 Loop을 돌 때:

Index 보다는 Element

Ok: Index

```
colors = ['red', 'green', 'blue', 'yellow']
for i in range(len(colors)):
    print(colors[i])
```

Good: Elements

```
for color in colors:
    print(color)
```

2. Loop을 거꾸로 돌 때:

Index 보다는 Reverse

Not Good: Index

```
colors = ['red', 'green', 'blue', 'yellow']
for i in range(len(colors)-1, -1):
    print(colors[i])
```

Good: Reverse

```
for color in reversed(colors):
    print(color)
```

(http://devdoggo.netlify.com/post/python/python_techniques/)

Appendix

- [gæn] or [gʌn]

stirling_archer 2 points · 5 days ago
I've only ever heard people pronounce it [gæn] across a few dialects of English.
Reply Share Report Save Give gold

visarga 2 points · 5 days ago
I pronounce it "gʌn" but I am not a native English speaker.
Reply Share Report Save Give gold

pumpkin105 1 point · 4 days ago
Like gun
Reply Share Report Save Give gold

CQQML 1 point · 1 day ago
most people read it as [gæn] in china
Reply Share Report Save Give gold

chisai_mikan 2 points · 5 days ago
JAN
Reply Share Report Save Give gold

swegmesterflex 1 point · 4 days ago
Jane
Reply Share Report Save Give gold

Surextra 14 points · 5 days ago
Hard G, rhymes with van. That's how Ian Goodfellow pronouces it anyway.
Reply Share Report Save Give gold

samclifford 19 points · 5 days ago
*Joodfellow
Reply Share Report Save Give gold

FutureIsMine 2 points · 4 days ago
Hey Jood......
Reply Share Report Save Give gold