

What is this? Gum? It's GAN.

: Intuition & Mathematical proof

ISL Lab Seminar

Hansol Kang

Contents

Introduction

Paper review

Configuration

Experiment

Summary

Introduction

- Ian Goodfellow



GAN

"Generative adversarial nets."

DCGAN

LSGAN

F-GAN

BEGAN

...

InfoGAN

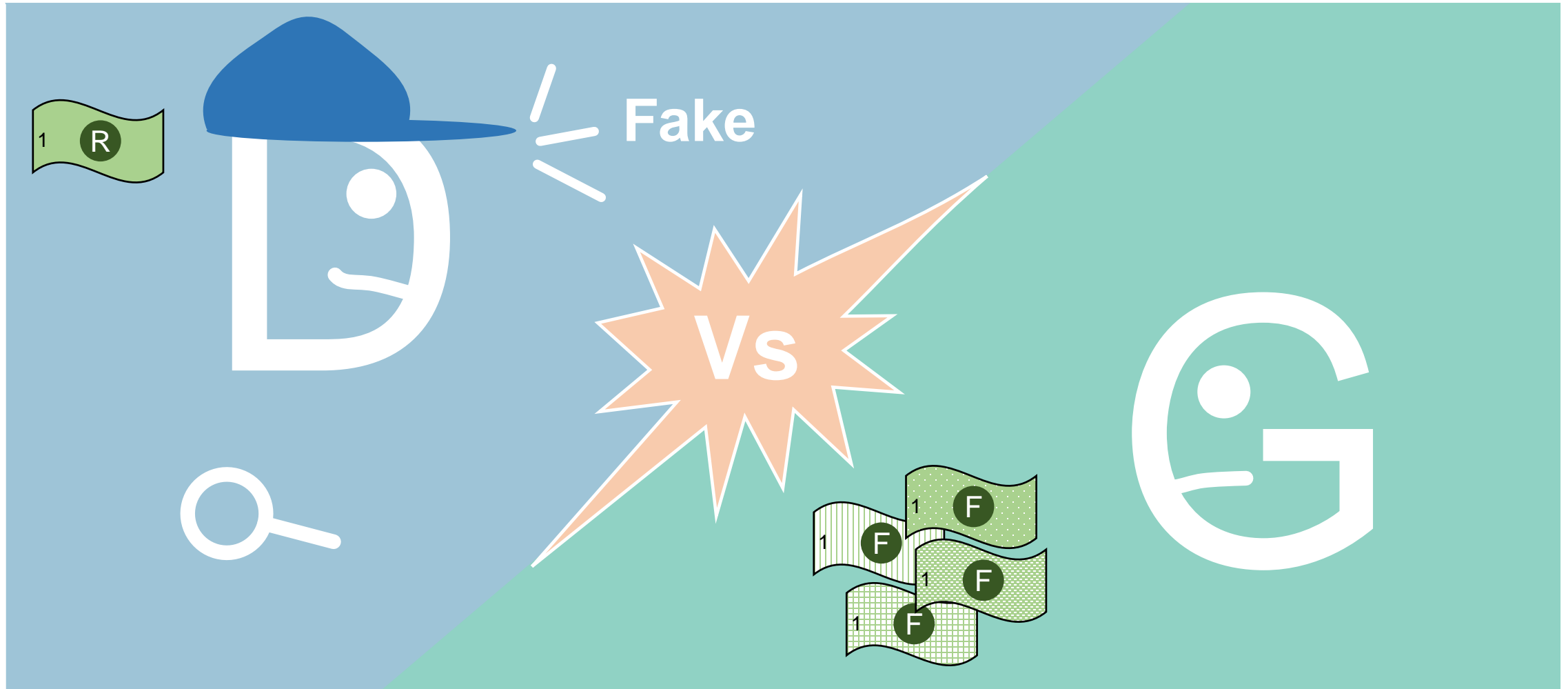
DiscoGAN

Unrolled
GAN

CycleGAN

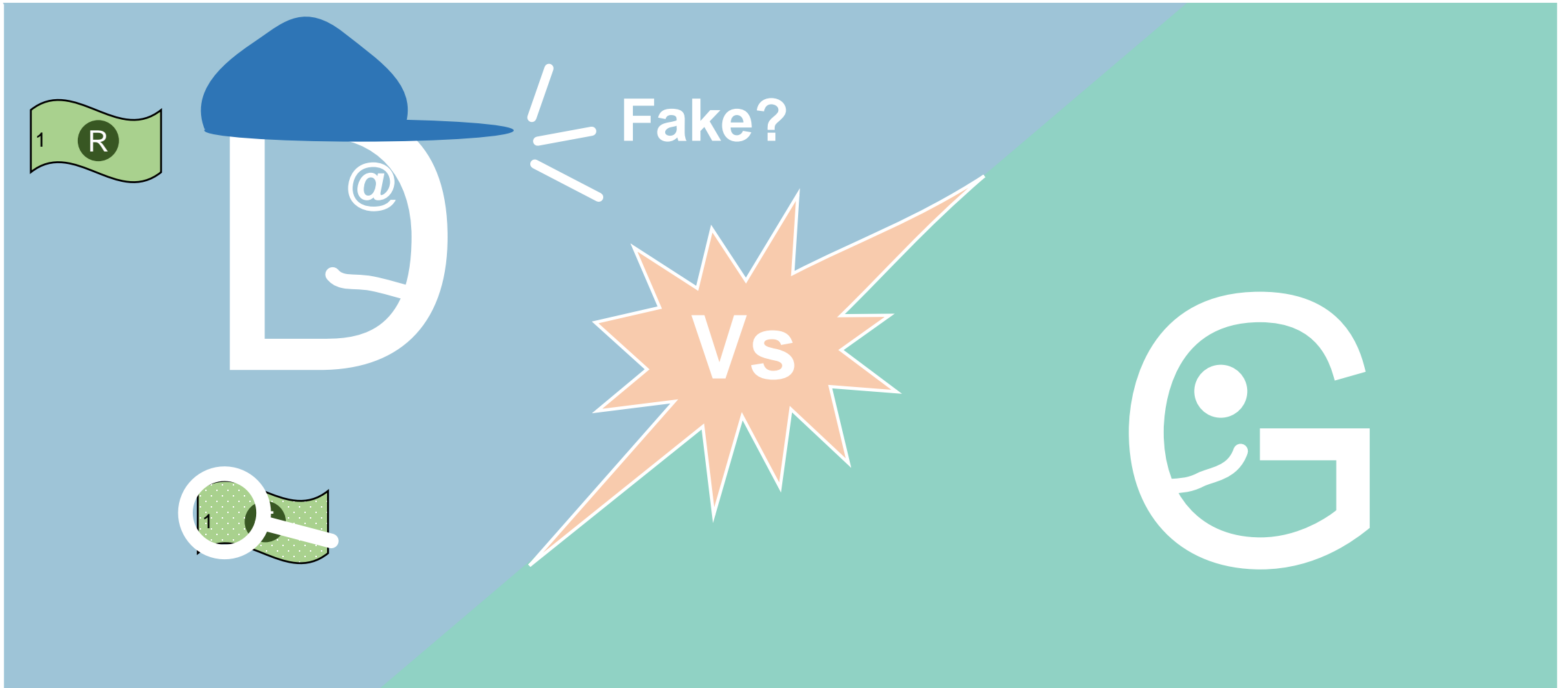
Introduction

- Concept of GAN



Introduction

- Concept of GAN

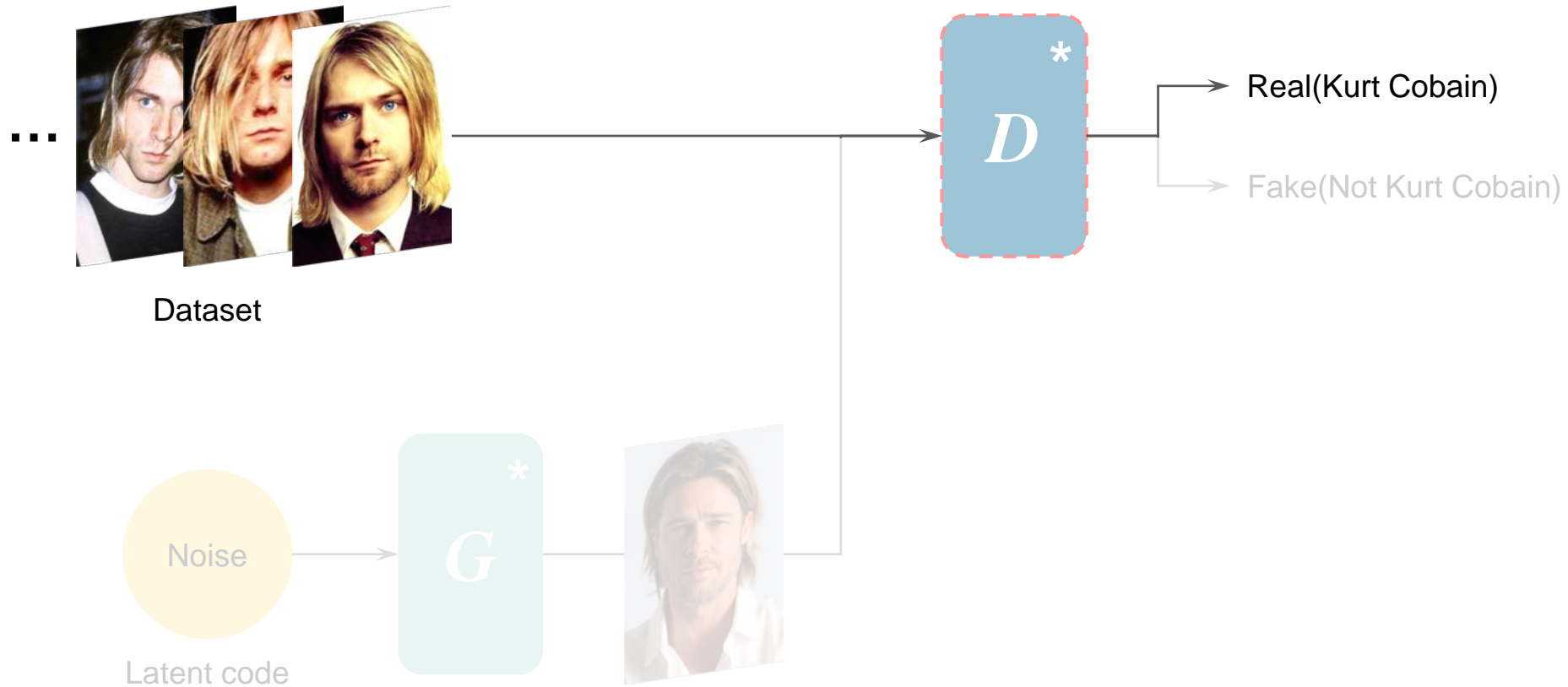


Introduction

- Process of Training

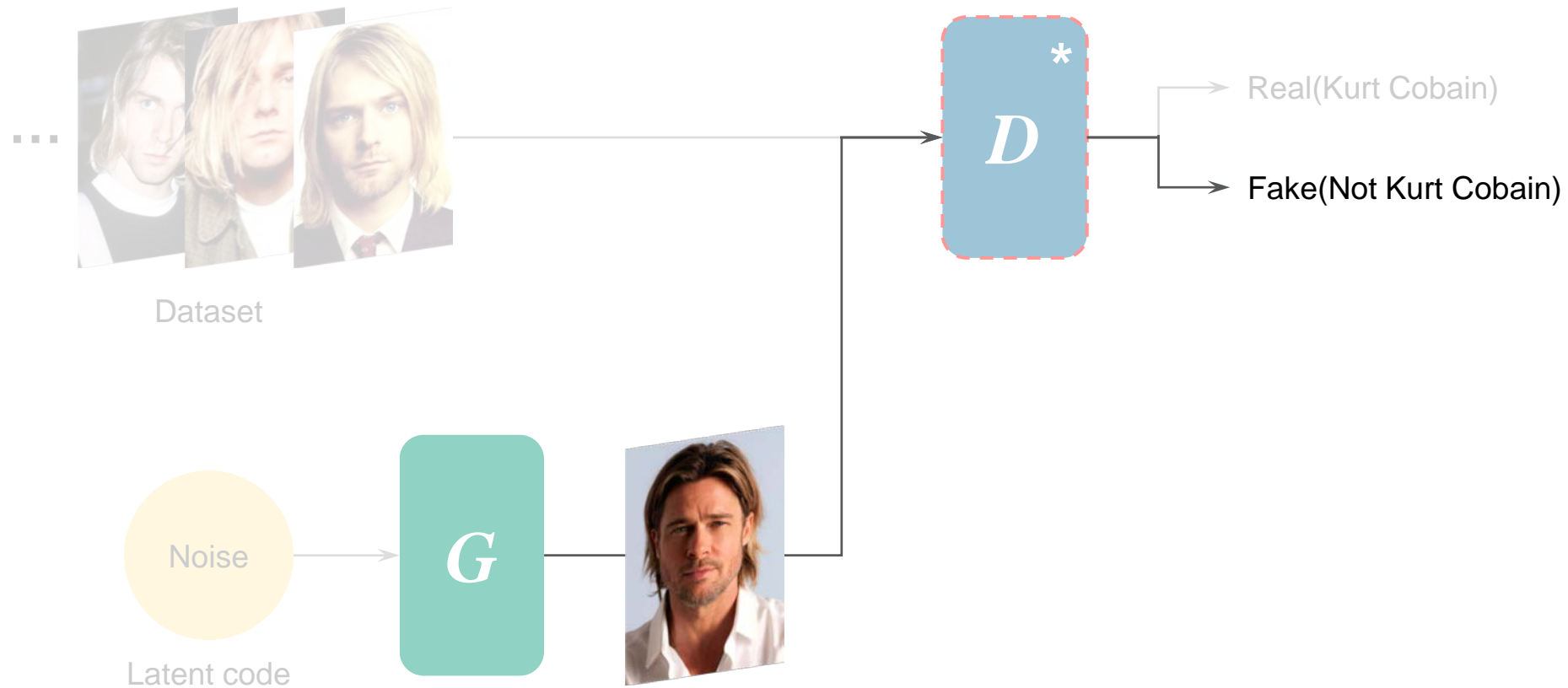
Train D

Train G



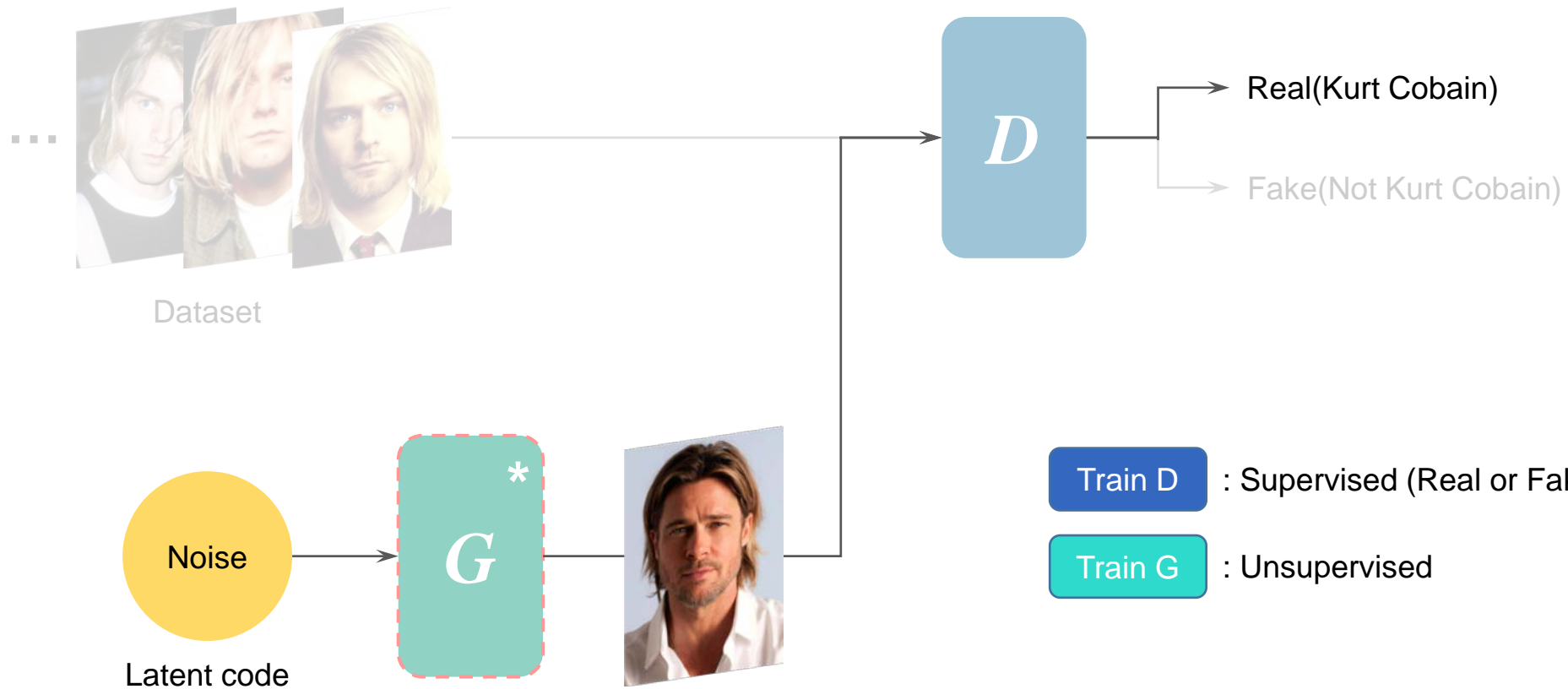
Introduction

- Process of Training



Introduction

- Process of Training



Train D

Train G

Train D : Supervised (Real or Fake)

Train G : Unsupervised

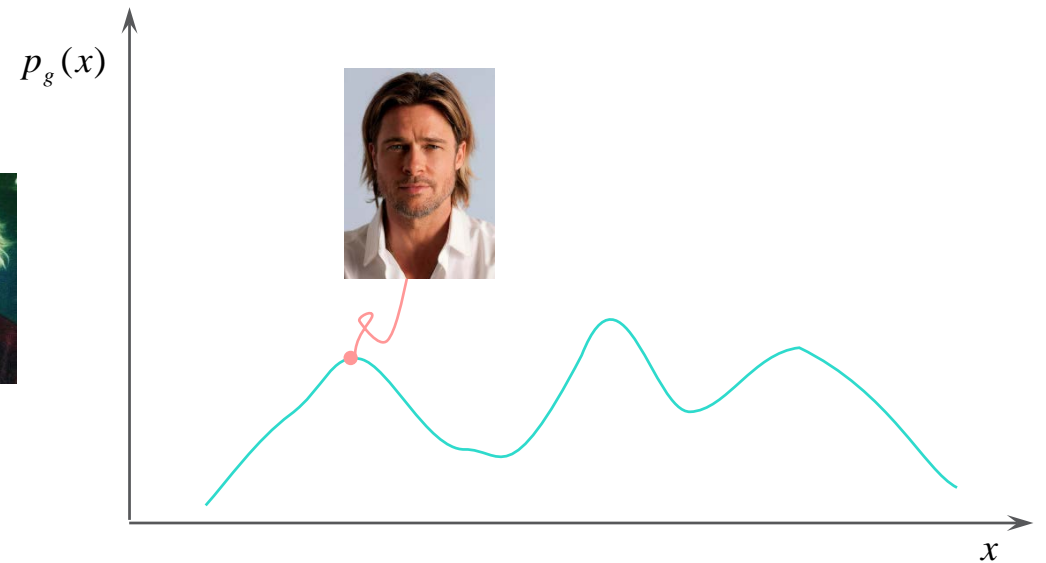
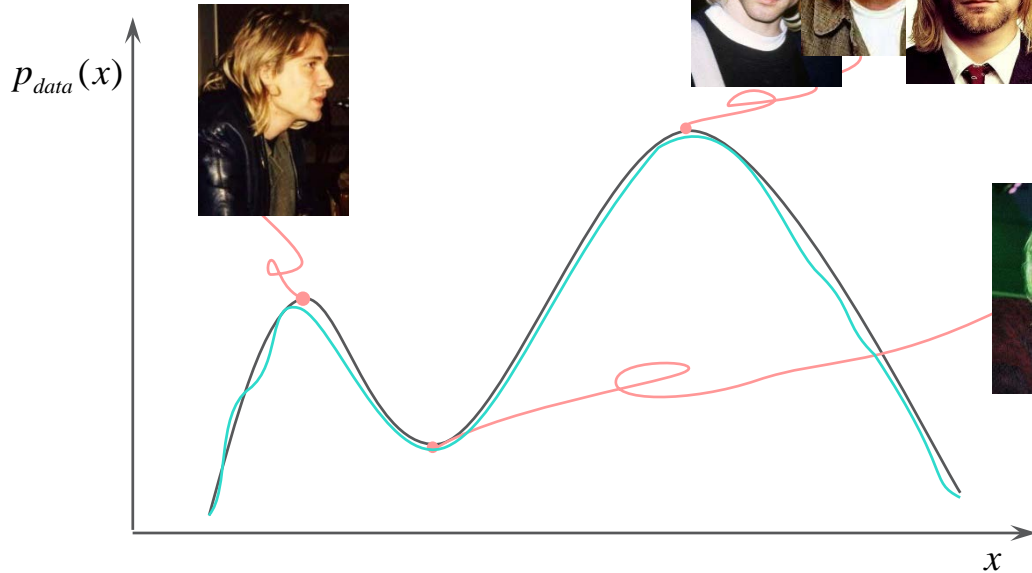
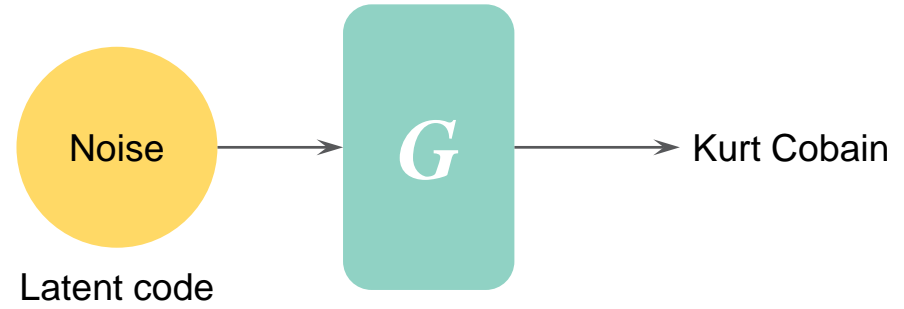
Introduction

- Final Goal

“Generative Adversarial Networks”

Goal

Method



Paper review

- Adversarial nets

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

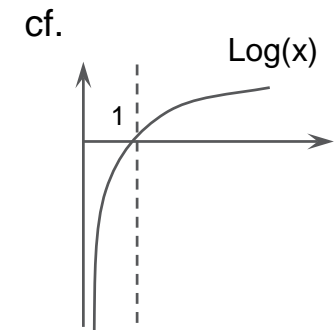
Smart D

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

↻ 1

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

↻ 0



Stupid D

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

↻ 0

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

↻ 1



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

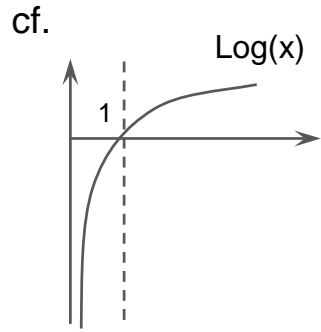
$$E_{x \sim p_{data}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

should be negative infinity

Stupid G

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

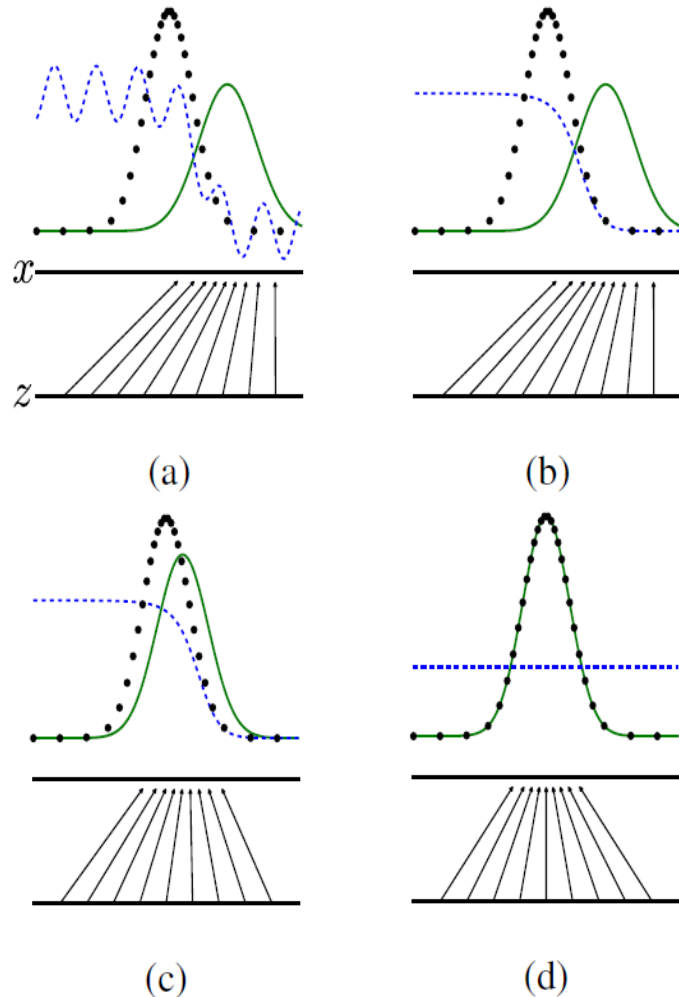
should be 0



**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 $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
- Update the discriminator by ascending its stochastic gradient:

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

end for

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

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

end for

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

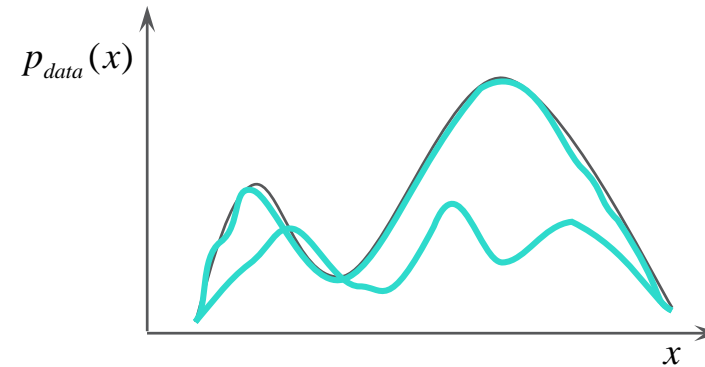
Paper review

- 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 $V(G, D)$

$$\begin{aligned} V(G, D) &= E_{x \sim p_{data}} [\log(D(x))] + E_{z \sim p_z} [\log(1 - D(G(z)))] \\ &= \int_x p_{data}(x) \log(D(x)) dx + \int_z p_z(z) \log(1 - D(G(z))) dz \\ &= \int_x p_{data}(x) \log(D(x)) dx + \int_x p_g(x) \log(1 - D(x)) dx \quad \left. \vphantom{\int_x p_g(x)} \right\} x = G(z) \\ &= \int_x p_{data}(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx \end{aligned}$$

Paper review

- Theoretical Results cont.

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

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

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

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

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

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



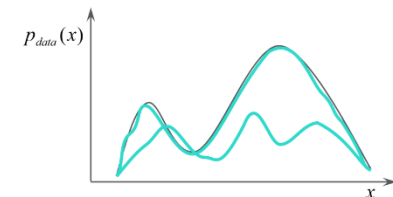
Paper review

- Theoretical Results

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

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

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



Paper review

- Theoretical Results cont.

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

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

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

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

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

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 $-\log 4$.

Proof.

$$\begin{aligned} C(G) &= E_{x \sim p_{data}} \left[\log D_G^*(x) \right] + E_{x \sim p_g} \left[\log(1 - D_G^*(x)) \right] \\ &= E \left[\log \frac{1}{2} \right] + E \left[\log \left(1 - \frac{1}{2} \right) \right] = \log \frac{1}{4} = -\log 4 \end{aligned}$$

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

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

Paper review

- Theoretical Results cont.

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

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

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

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

$$= -\log 4 + 2JSD(p_{data}(x) \parallel p_g(x)) \quad \text{if } JSD = 0, \text{ then } -\log 4$$

cf.

Kullback–Leibler divergence

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

Jensen–Shannon divergence

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

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

$$E_{x \sim P_{data}} [\log D_G^*(x)] + E_{x \sim P_g} [\log(1 - D_G^*(x))]$$

then P_g converges to P_{data}

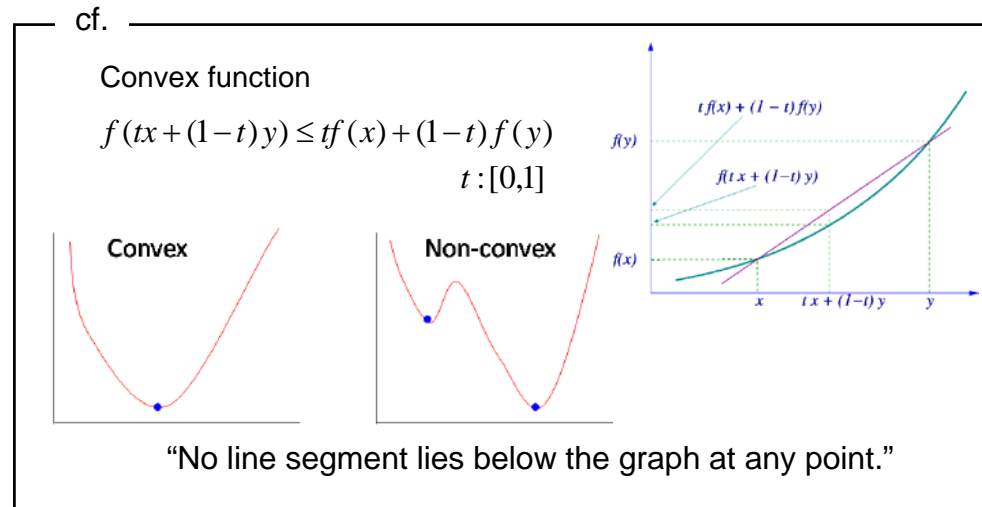
Paper review

- Theoretical Results cont.

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

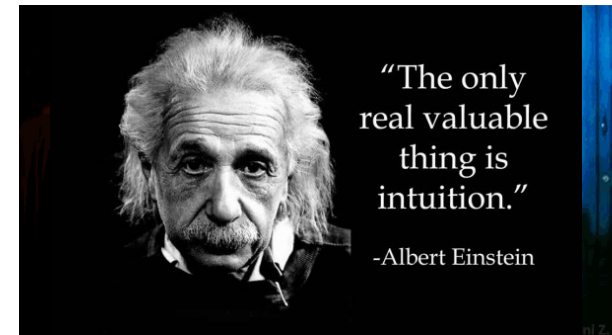
$\sup_D U(p_g, D)$ is convex in p_g with a unique global optima as proven in Thm 1, therefore with sufficiently small updates of p_g , p_g converges to p_{data} , concluding proof.

cf.

supremum

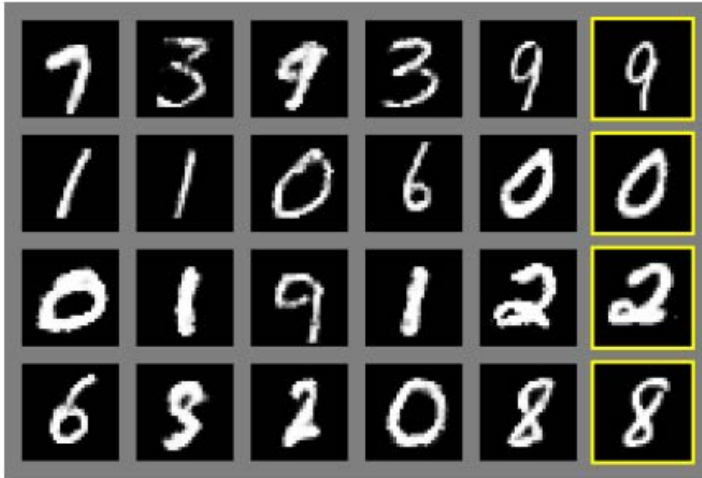
- 실수 b 가 집합 A 의 모든 원소들보다 크거나 같을때
 즉, $a \leq b$ 이면 b 를 집합 A 의 상계(Upper bound)라 한다. 단, $a \in A$
- 집합 A 의 상계들의 집합을 U 라 하면 U 의 최소원소를 집합 A 의 상한이라 하고 $\sup A$ 라 쓴다.

(<http://mathmath.tistory.com/27>)



Paper review

- Experiment



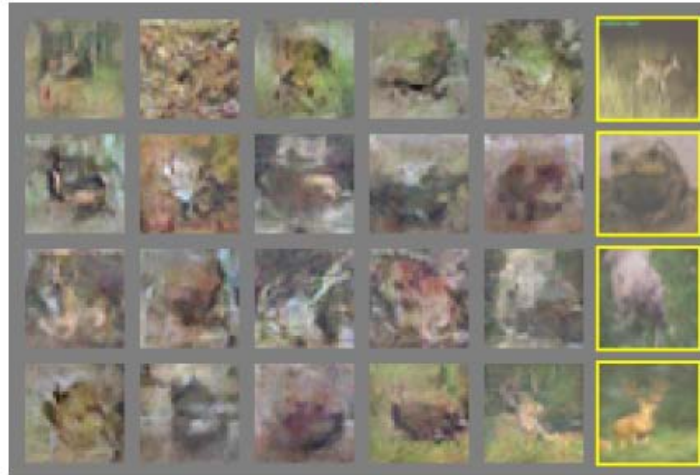
a)



b)



c)



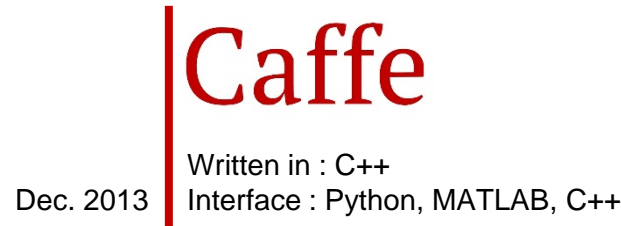
d)

Configuration

- Deep Learning Framework



Nov. 2010
Written in : Python
Interface : Python



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



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

 <p>Mar. 2015 Written in : Python Interface : Python, R</p>	 <p>Nov. 2015 Written in : C++, Python, CUDA Interface : Python, C/C++, Java, Go, R, Julia</p>	 <p>Oct. 2016 Written in : Python, C, CUDA Interface : Python</p>
--	---	--

Recommend to choose these framework

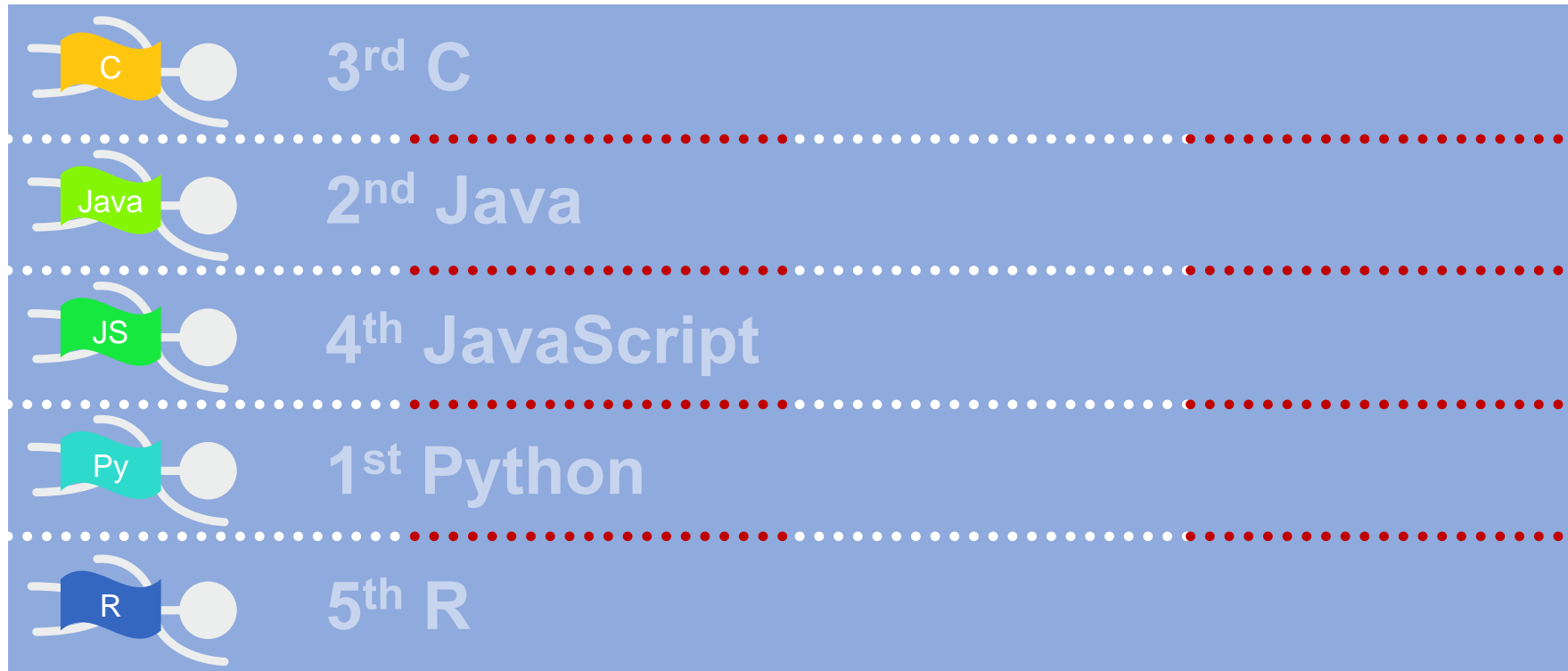


Apr. 2017
Written in :
Interface : Python, C++

- 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 :



Configuration

- Development Tool



: **Intellisense**

: **Cell based execution**



: **Extension program**

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



Visual Studio

: **Startup file**

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



VS Code

: **Intellisense**
: **Environment setting**

: **Insane extension program**

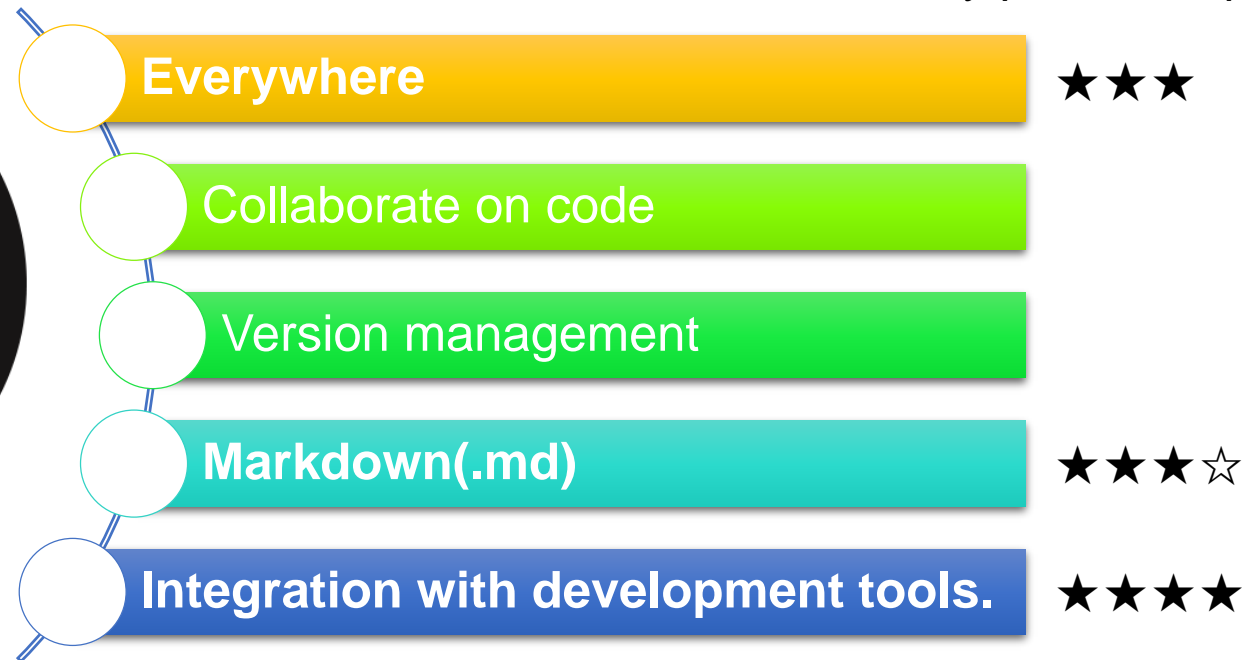
Configuration

- CM(Configuration Management) Tool

Very personal opinion

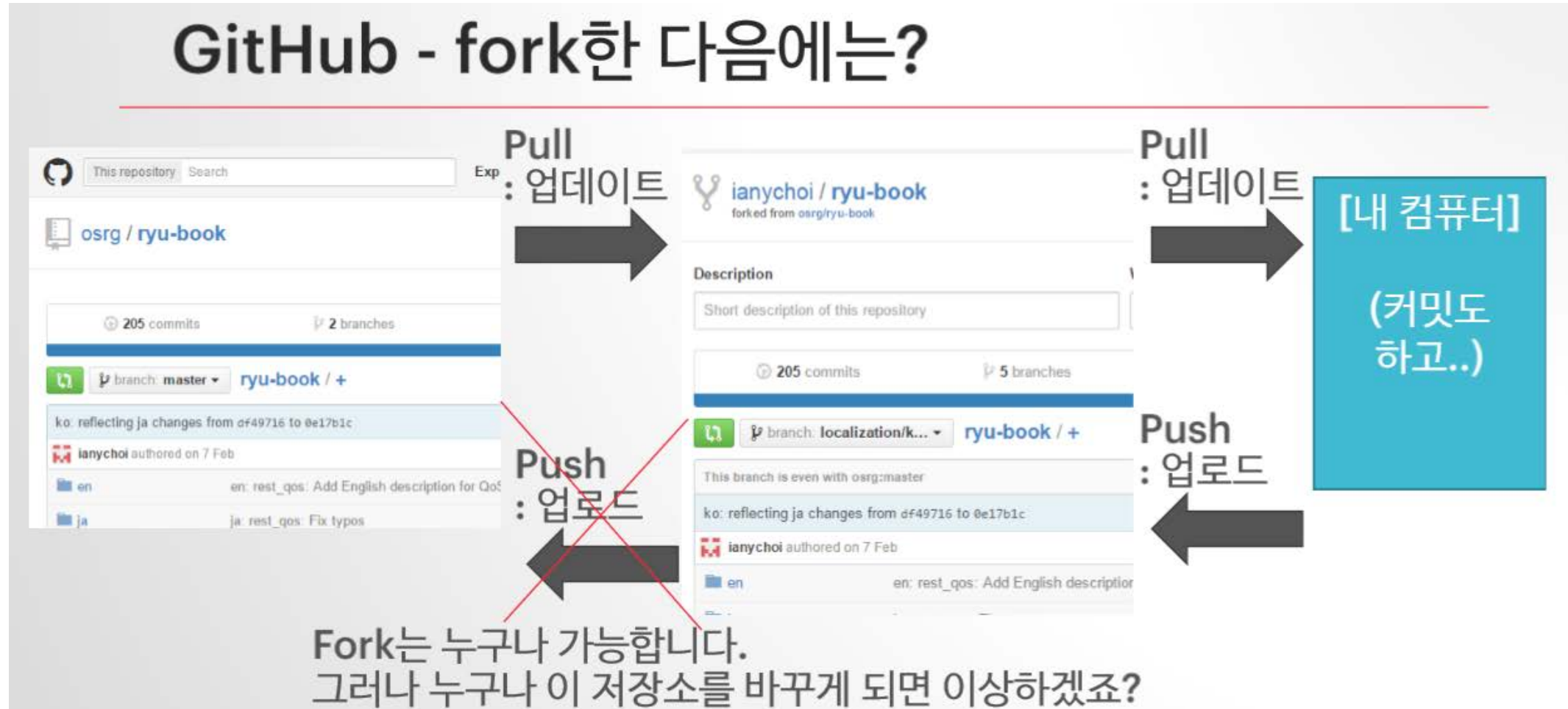


GitHub



Configuration

- CM(Configuration Management) Tool



Configuration

- CM(Configuration Management) Tool

Commits on Jul 5, 2018

다시 수정
rivergang committed a day ago

start change
rivergang committed a day ago

regression 추가
kurt committed 2 days ago

그림수정
kurt committed 2 days ago

그림수정
kurt committed 2 days ago

인스틀 작성완료
kurt committed 2 days ago

Commits on Jul 4, 2018

Tutorial test & md test
rivergang committed 2 days ago

Commits on Jul 3, 2018

Test1
rivergang committed 3 days ago

<- Revision history

The screenshot shows the Visual Studio 2017 interface. The main window displays a Python file named 'lab01.py' with the following code:

```

1 import torch
2
3 #torch.empty(r,c), torch.rand(r,c), torch.zeros(r,c,dtype)
4 #r : row, c: column
5
6 a = torch.tensor([2.3, 4]) #1차원 텐서 생성
7 b = torch.tensor([[2.2, 4.5], [5.3, 1.5]]) #2차원 텐서 생성
8
9 print("a : ", a)
10 print("size of a : ", a.size(), "\n")
11
12 print("b : ", b)
13 print("size of b : ", b.size(), "\n")
14
15 x = a.new_ones(5,3,dtype=torch.double) #텐서 재사용
16 print(x)
17
18 #텐서를 넘겨 받음. dtype override
19 x = torch.randn_like(x, dtype=torch.float)
20 print(x)
21
22 #y.add_(x), 언더바 붙이면 in-place 자기 자신
23 c = torch.rand(2,2)
24 c.add_(b)
25 print(c)
26

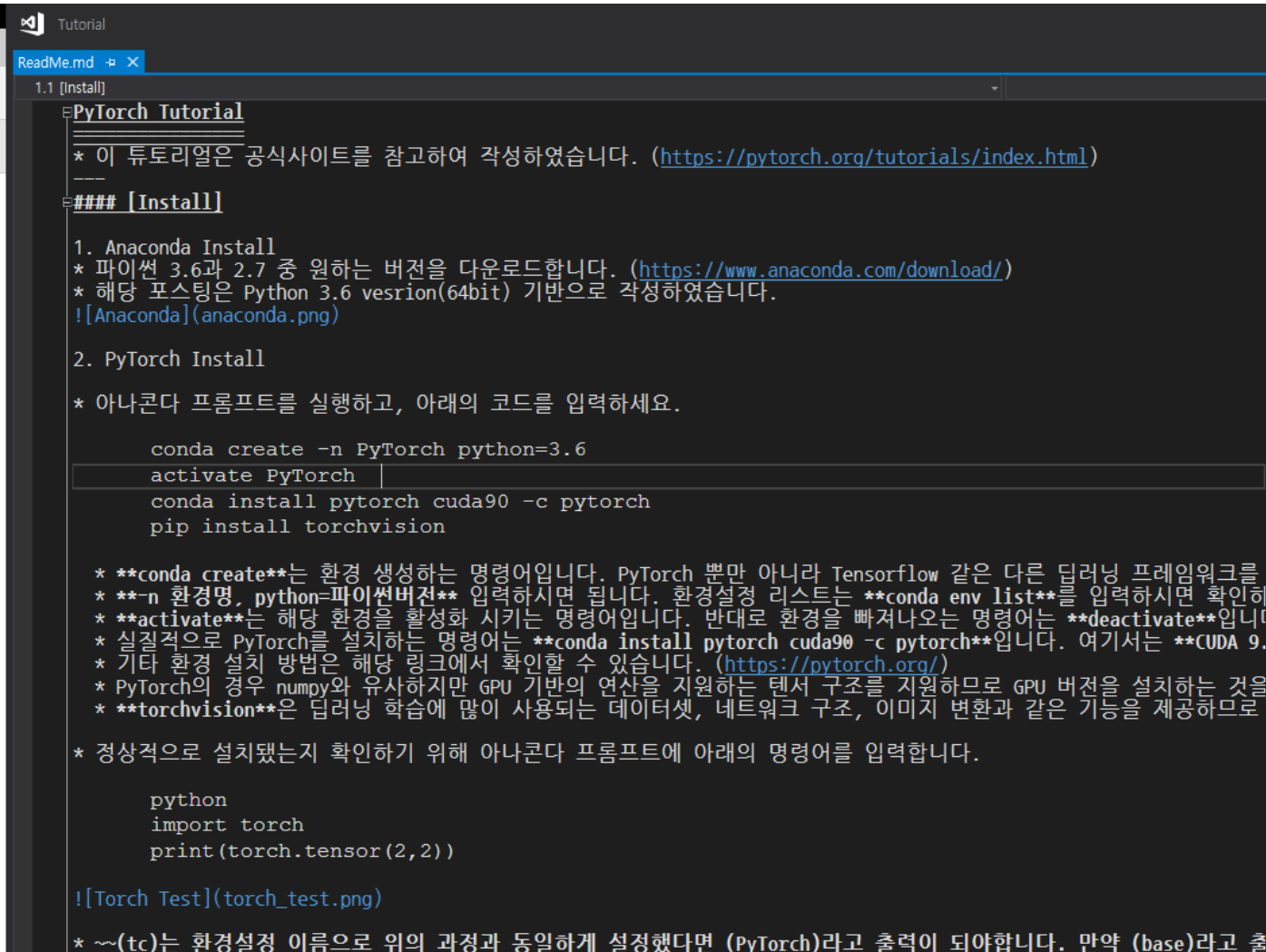
```

The sidebar on the left shows the 'GitHub' section with the repository 'rivergang/PyTorch' and a list of project actions: 변경 내용, 분기, Pull Requests, 동기화, 태그, Pulse, Graphs, Issues, Wiki, and 설정. The bottom status bar shows '출력 보기 선택(S): 빌드'.

VS 2017 ->

Configuration

- CM(Configuration Management) Tool



PyTorch Tutorial

* 이 튜토리얼은 공식사이트를 참고하여 작성하였습니다. (<https://pytorch.org/tutorials/index.html>)

[Install]

1. Anaconda Install

* 파이썬 3.6과 2.7 중 원하는 버전을 다운로드합니다. (<https://www.anaconda.com/download/>)

* 해당 포스팅은 Python 3.6 vesrion(64bit) 기반으로 작성하였습니다.

![Anaconda](anaconda.png)

2. PyTorch Install

* 아나콘다 프롬프트를 실행하고, 아래의 코드를 입력하세요.

```
conda create -n PyTorch python=3.6
activate PyTorch
conda install pytorch cuda90 -c pytorch
pip install torchvision
```


* ****conda create****는 환경 생성하는 명령어입니다. PyTorch 뿐만 아니라 Tensorflow 같은 다른 딥러닝 프레임워크를
* ****n 환경명, python=파이썬버전**** 입력하시면 됩니다. 환경설정 리스트는 ****conda env list****를 입력하시면 확인하
* ****activate****는 해당 환경을 활성화 시키는 명령어입니다. 반대로 환경을 빠져나오는 명령어는 ****deactivate****입니
* 실질적으로 PyTorch를 설치하는 명령어는 ****conda install pytorch cuda90 -c pytorch****입니다. 여기서는 ****CUDA 9.0****
* 기타 환경 설치 방법은 해당 링크에서 확인할 수 있습니다. (<https://pytorch.org/>)
* PyTorch의 경우 numpy와 유사하지만 GPU 기반의 연산을 지원하는 텐서 구조를 지원하므로 GPU 버전을 설치하는 것을
* ****torchvision****은 딥러닝 학습에 많이 사용되는 데이터셋, 네트워크 구조, 이미지 변환과 같은 기능을 제공하므로

* 정상적으로 설치됐는지 확인하기 위해 아나콘다 프롬프트에 아래의 명령어를 입력합니다.

```
python
import torch
print(torch.tensor(2,2))
```

![Torch Test](torch_test.png)

* **~(tc)**는 환경설정 이름으로 위의 과정과 동일하게 설정했다면 (PyTorch)라고 출력이 되어합니다. 만약 (base)라고 출



PyTorch Tutorial

• 이 튜토리얼은 공식사이트를 참고하여 작성하였습니다.
(<https://pytorch.org/tutorials/index.html>)

[Install]

1. Anaconda Install

• 파이썬 3.6과 2.7 중 원하는 버전을 다운로드합니다.
(<https://www.anaconda.com/download/>)

• 해당 포스팅은 Python 3.6 vesrion(64bit) 기반으로 작성하였습니다.

Anaconda 5.2 For Windows Installer

Python 3.6 version * Python 2.7 version *

or

Download Download

64-Bit Graphical Installer (631 MB) 64-Bit Graphical Installer (564 MB)

32-Bit Graphical Installer (506 MB) 32-Bit Graphical Installer (443 MB)

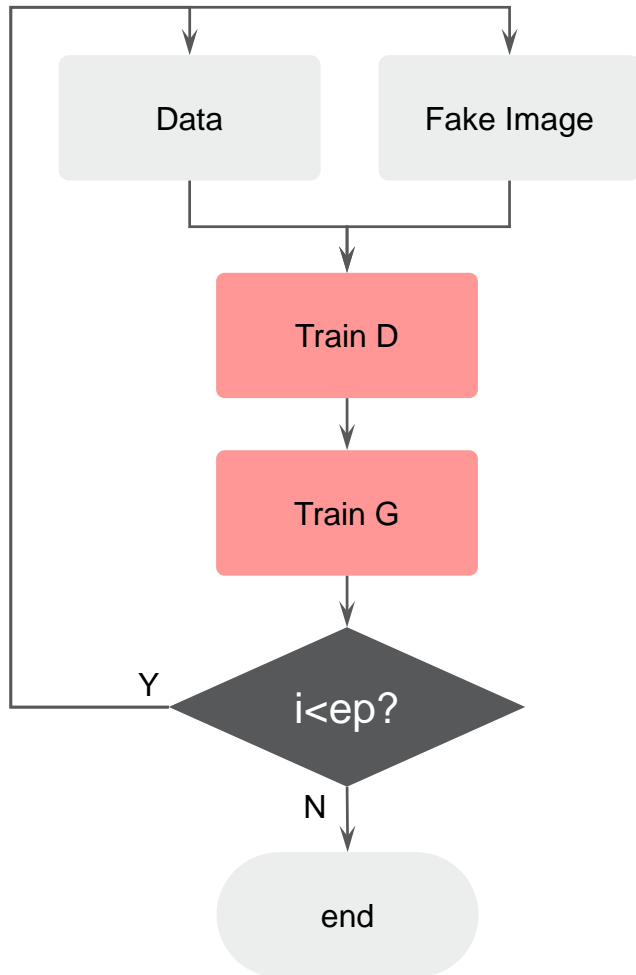
2. PyTorch Install

• 아나콘다 프롬프트를 실행하고, 아래의 코드를 입력하세요.

```
conda create -n PyTorch python=3.6
activate PyTorch
conda install pytorch cuda90 -c pytorch
pip install torchvision
```

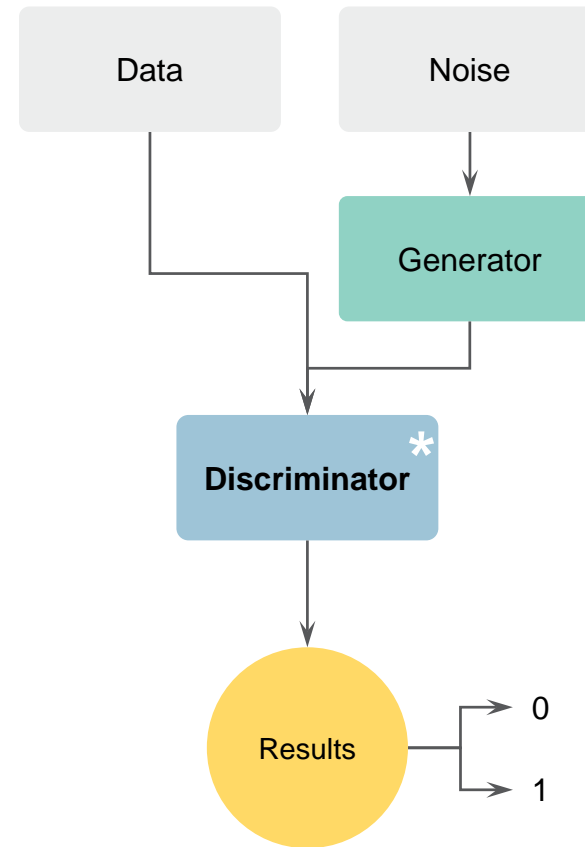
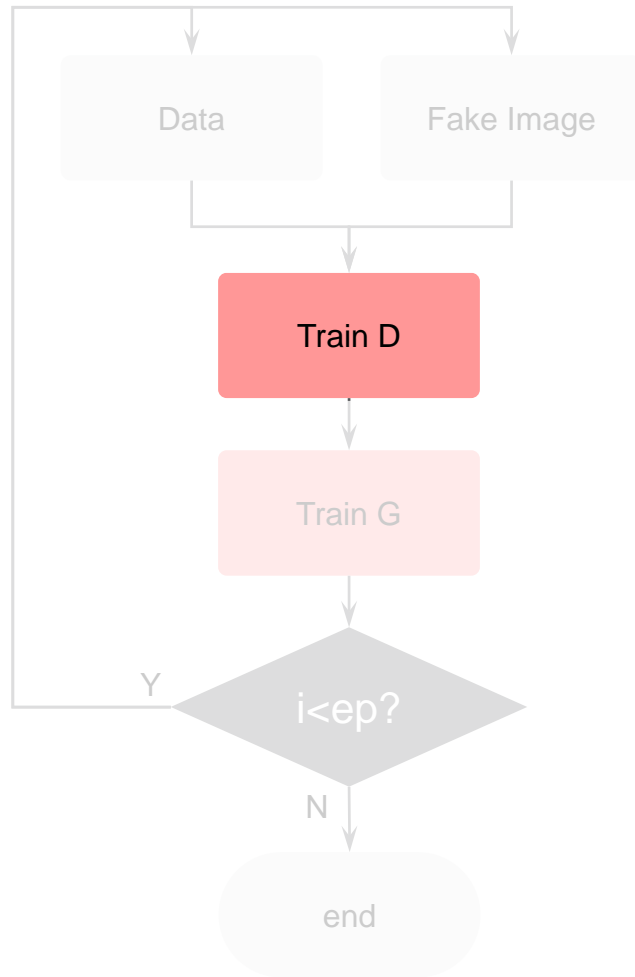
Experiment

- Flowchart



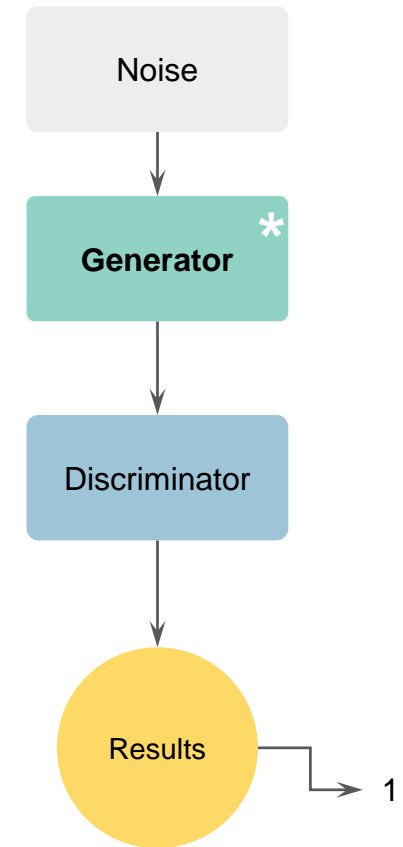
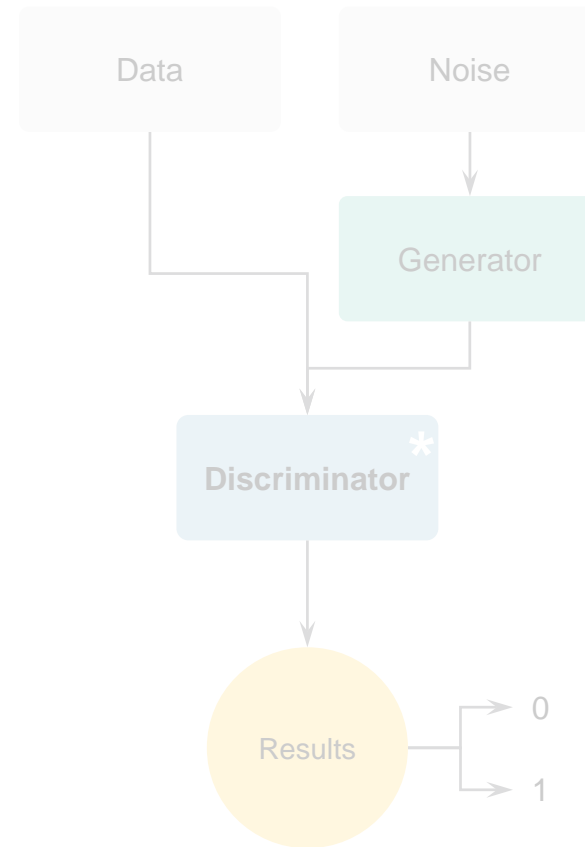
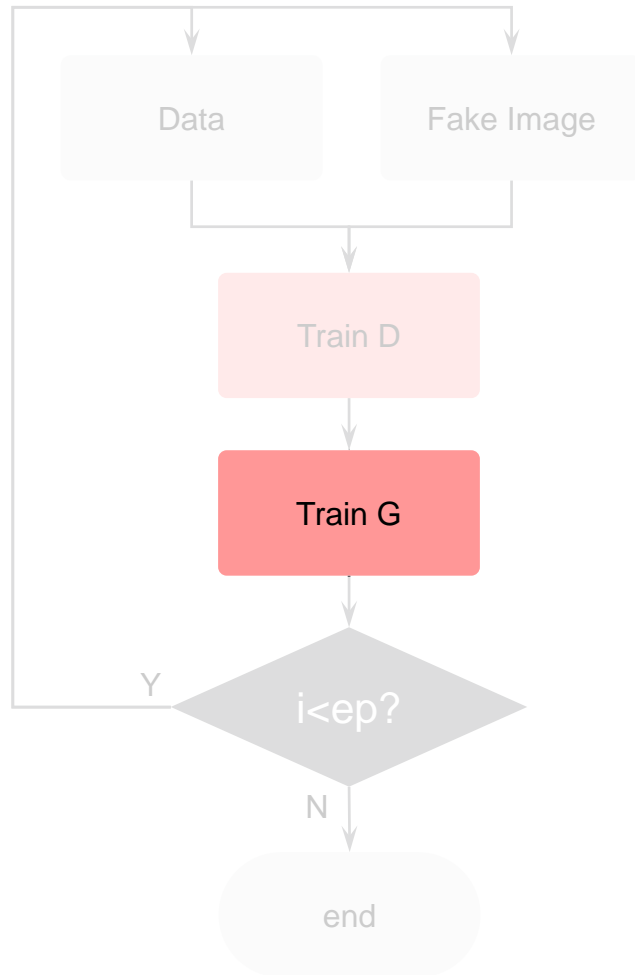
Experiment

- Flowchart



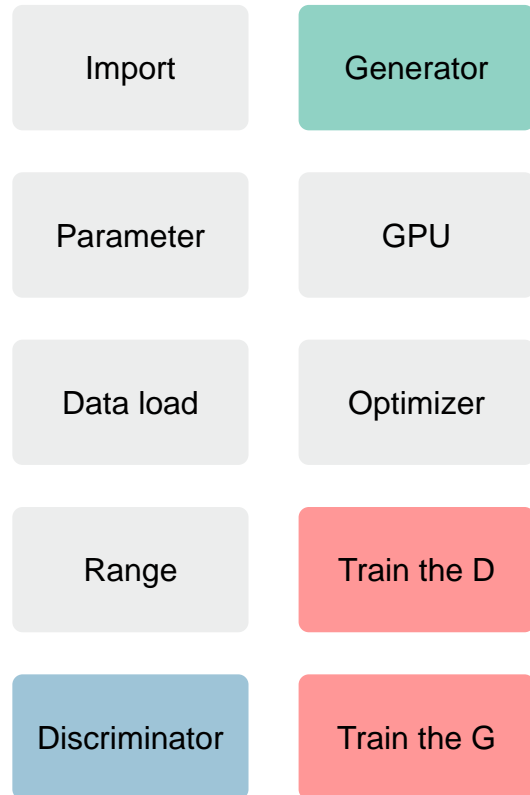
Experiment

- Flowchart



Experiment

- Implementation



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
```

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)

```

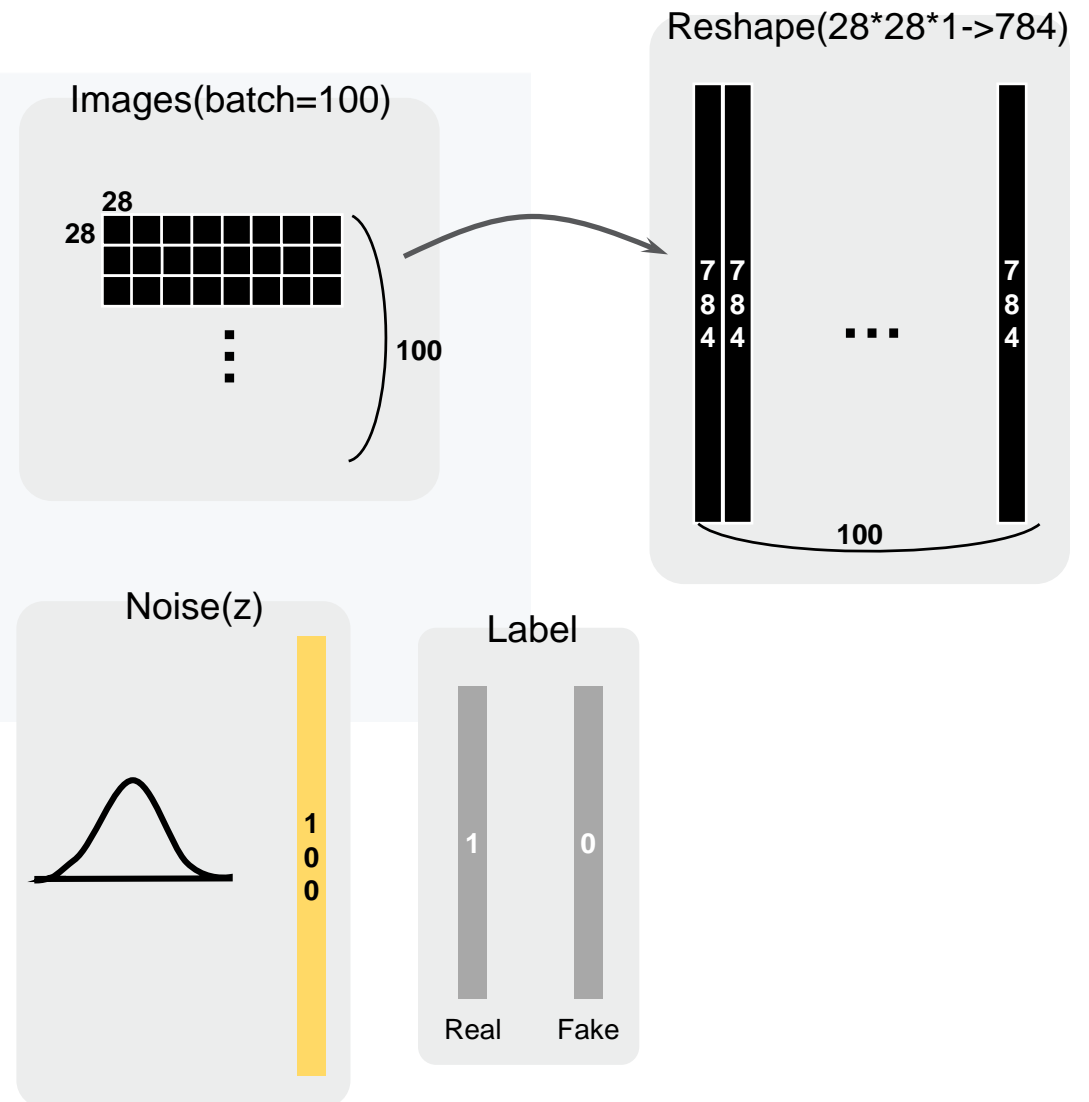
for ep in range(nEpoch):
    for step, (images, ) in enumerate(dataloader):
        images = images.reshape(batch_sz, -1).to(device)
        z = tc.randn(batch_sz, noise_sz).to(device)

        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.

```

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)

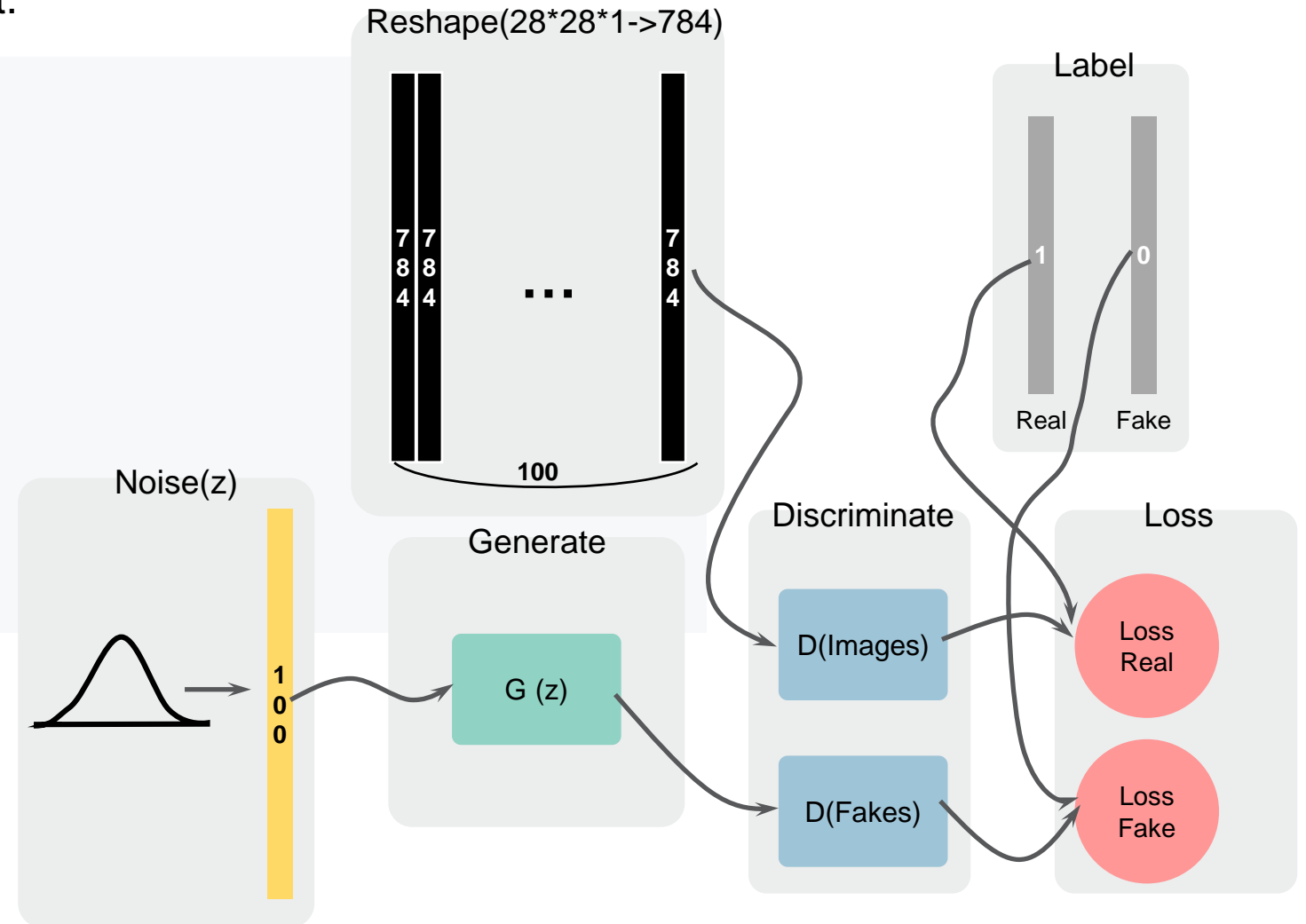
        real_label = tc.ones(batch_sz, 1).to(device)
        fake_label = tc.zeros(batch_sz, 1).to(device)

        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.

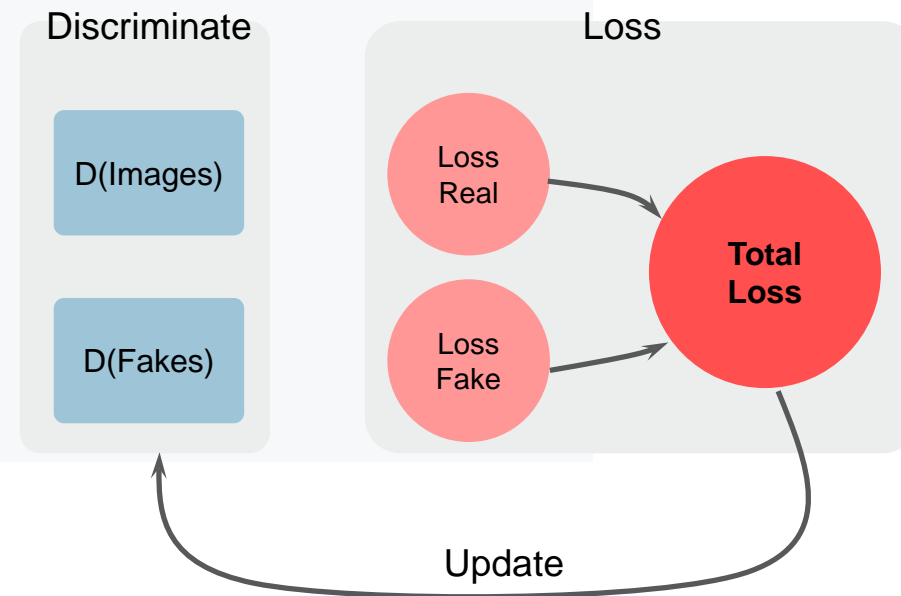
```
for ep in range(nEpoch):
    for step, (images, _) in enumerate(dataloader):
        images = images.reshape(batch_sz, -1).to(device)
        z = tc.randn(batch_sz, noise_sz).to(device)

        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()
```

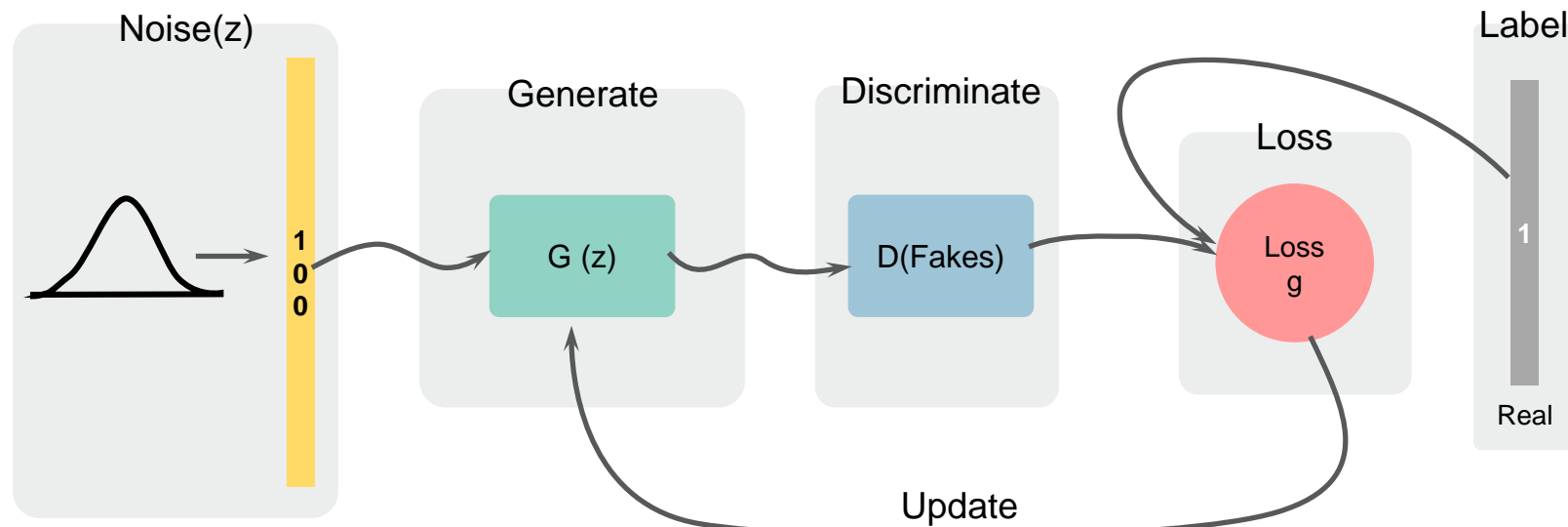


Experiment

- 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()
```

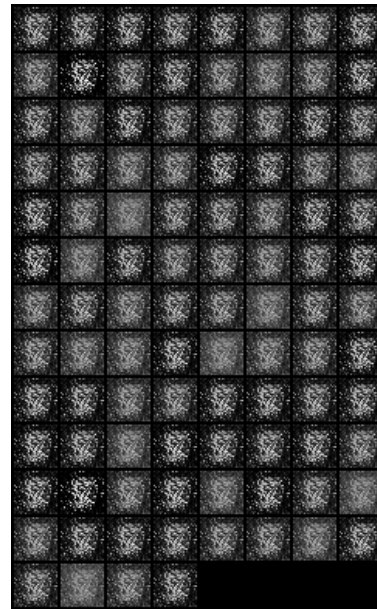


Experiment

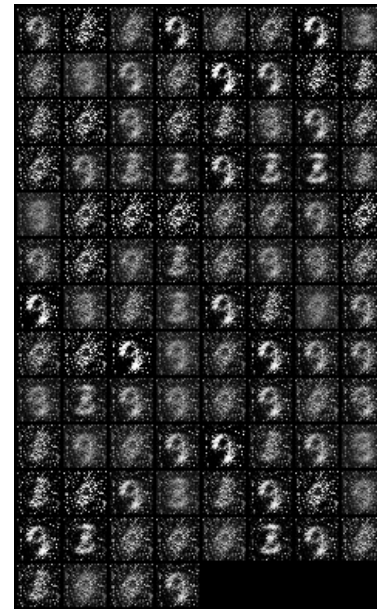
- Result#1 (MNIST)



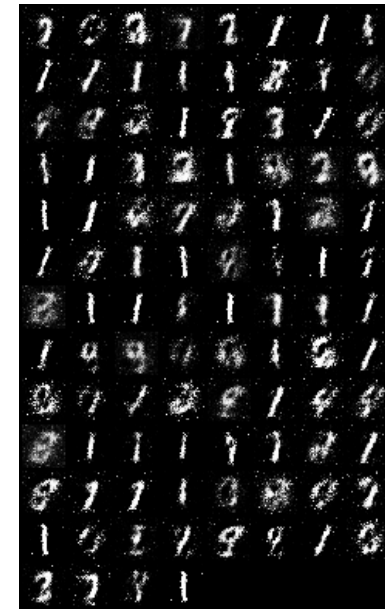
Real



1



10



30



100



200



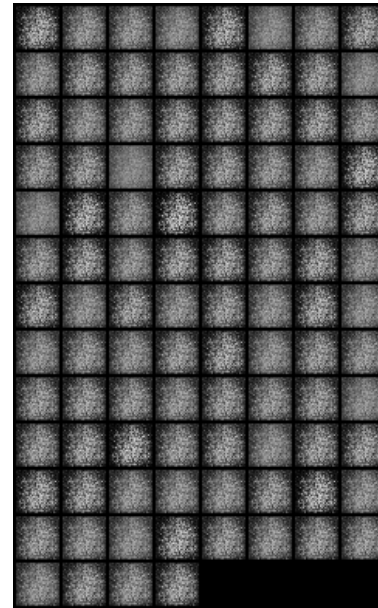
400

Experiment

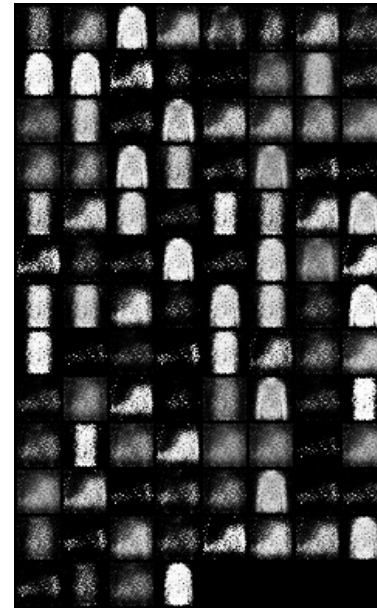
- Result#2 (FashionMNIST)



Real



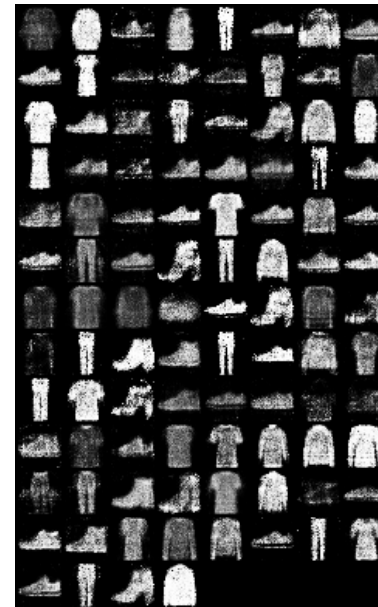
1



10



30



100



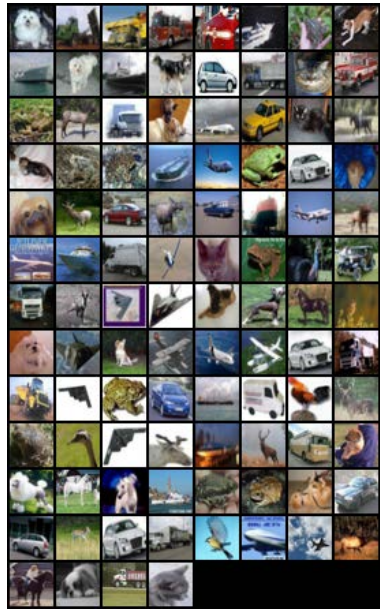
200



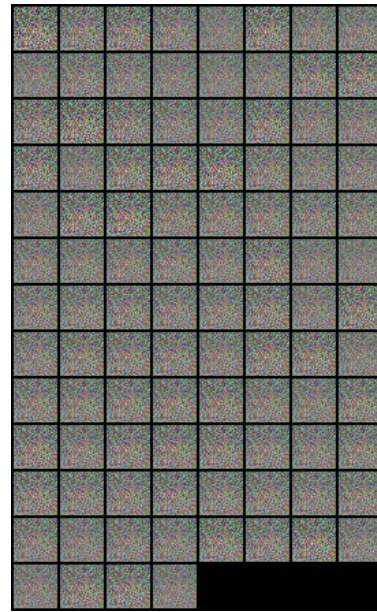
400

Experiment

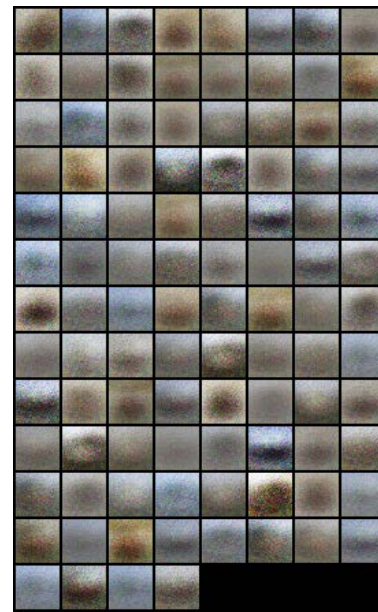
- Result#3 (CIFAR10)



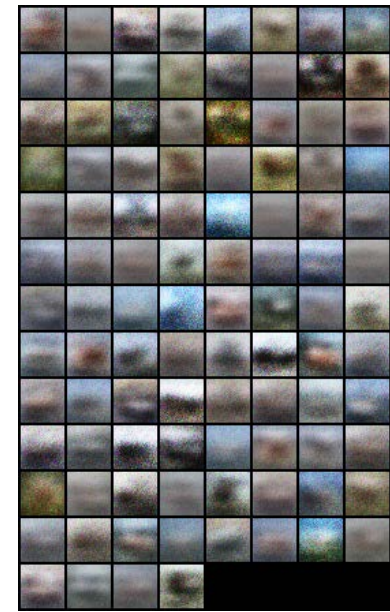
Real



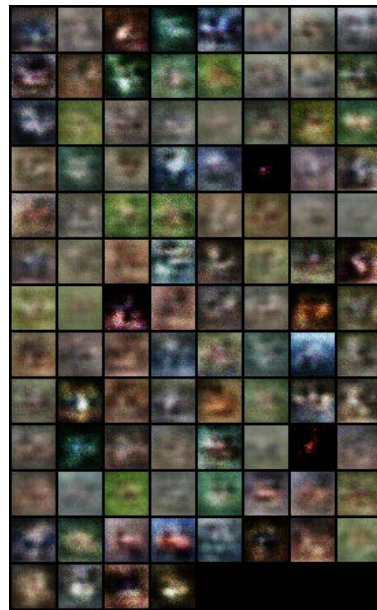
1



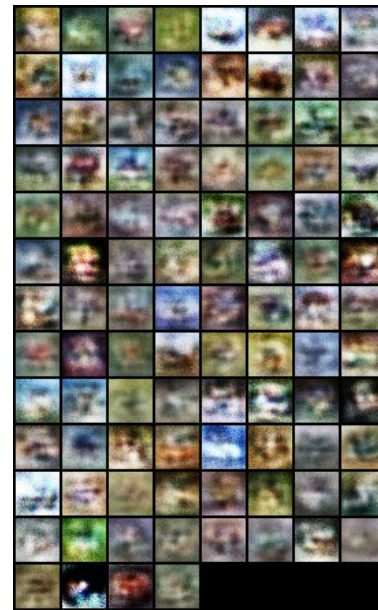
10



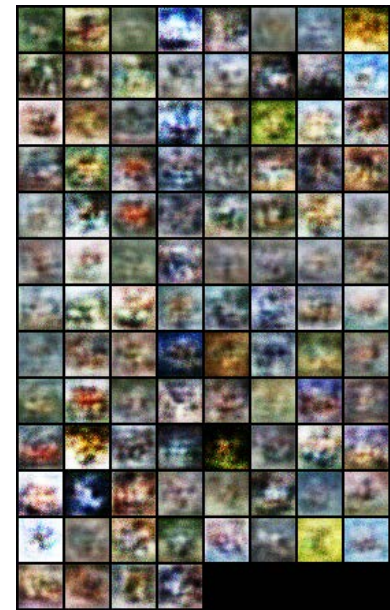
30



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 개발에 AI 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

- [gæɪn] or [gʌn]

↑ [stirling_archer](#) 2 points · 5 days ago
 ↓ I've only ever heard people pronounce it [gæɪn] 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 [gæɪn] 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](#)