# 찐개만
# Generative Dogs Images

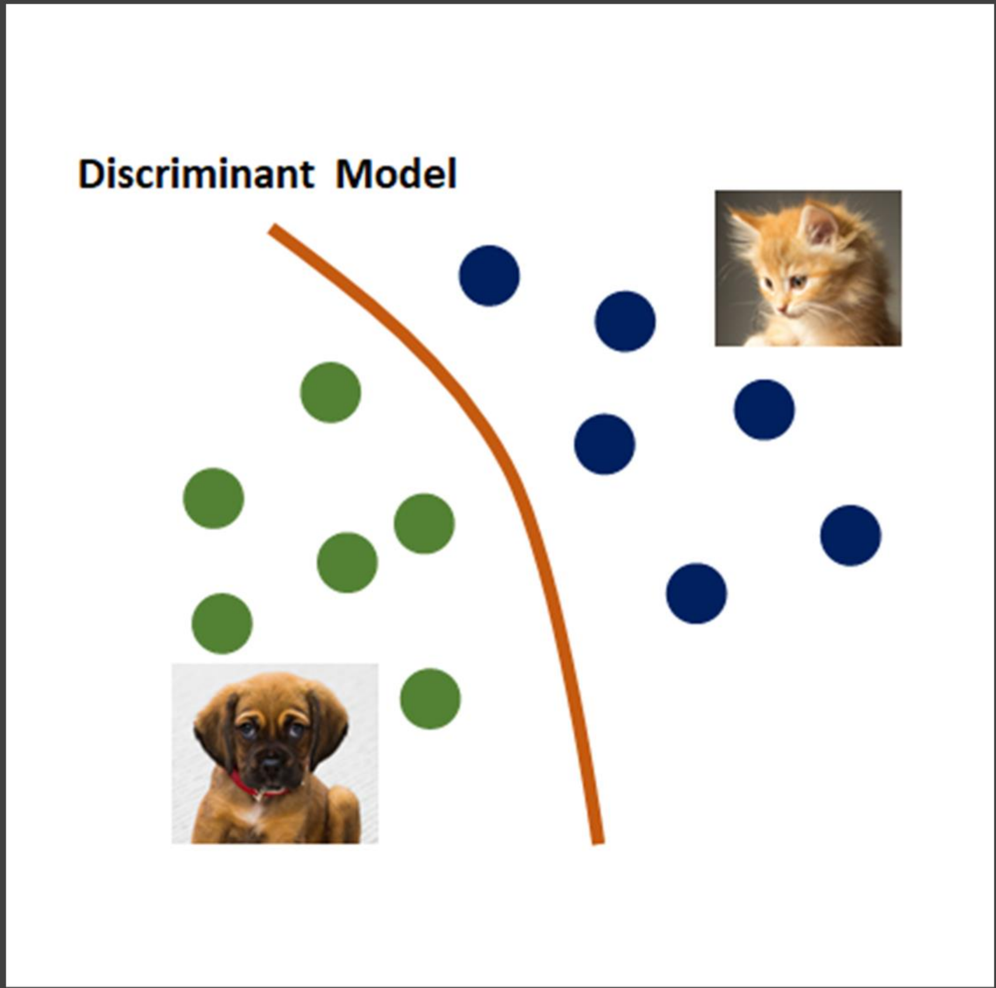
김보라 이경관

# 01
# Introduce

# Purpose of our Project

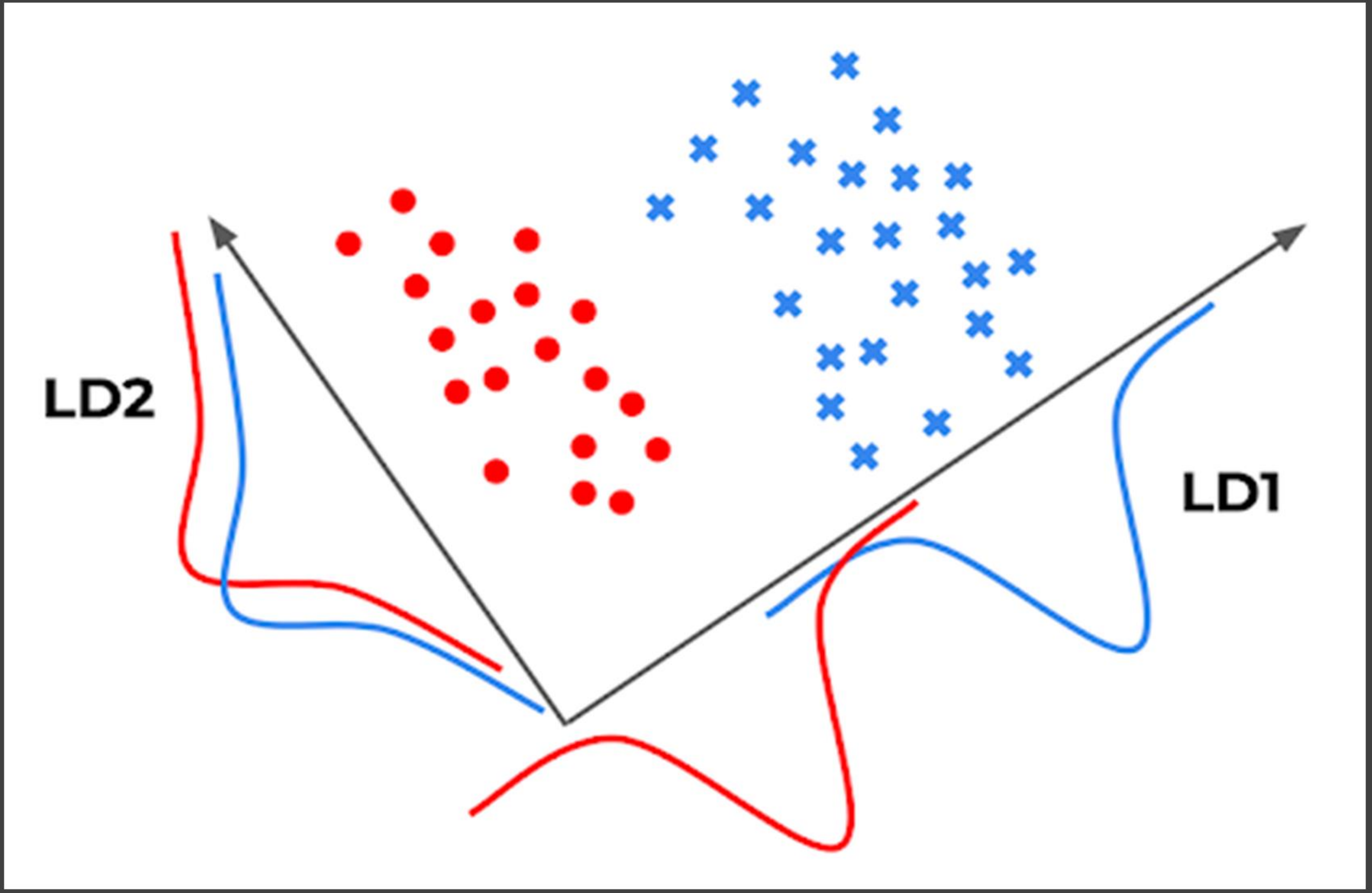
# Discriminative Model

무엇을 차이를 기준으로 어떻게 구별할 것인가?



# Generative Model

데이터가 어떻게 분포 되어 있는가?



# 02
# The beginning of GAN

# Origin of GAN

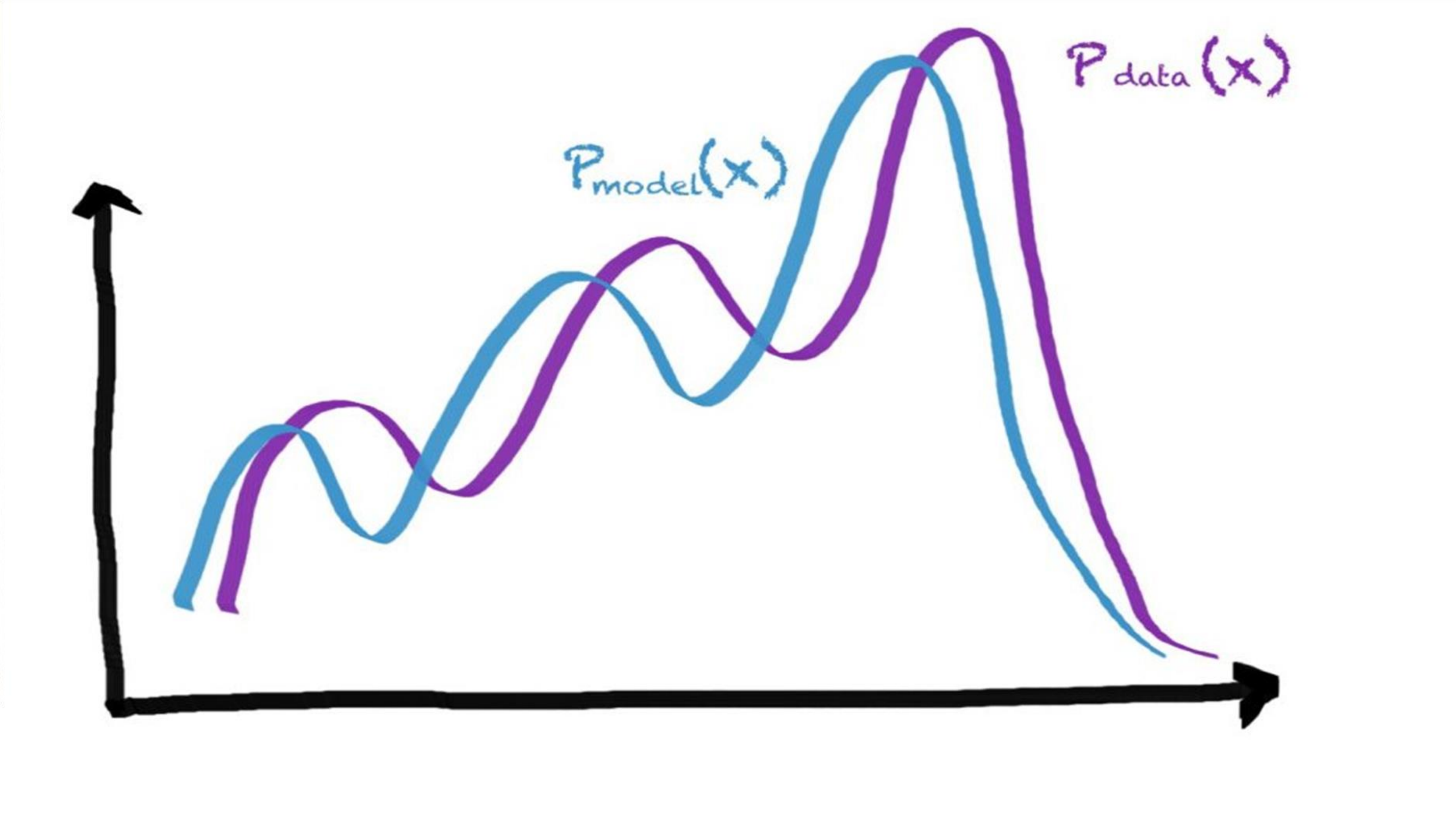
**학습을 위해 두 모델을 서로 견제하게 만들면 어떨까?**

**Generative Adversarial Networks(2014)**

**Ian, Goodfellow**

# What is GAN?

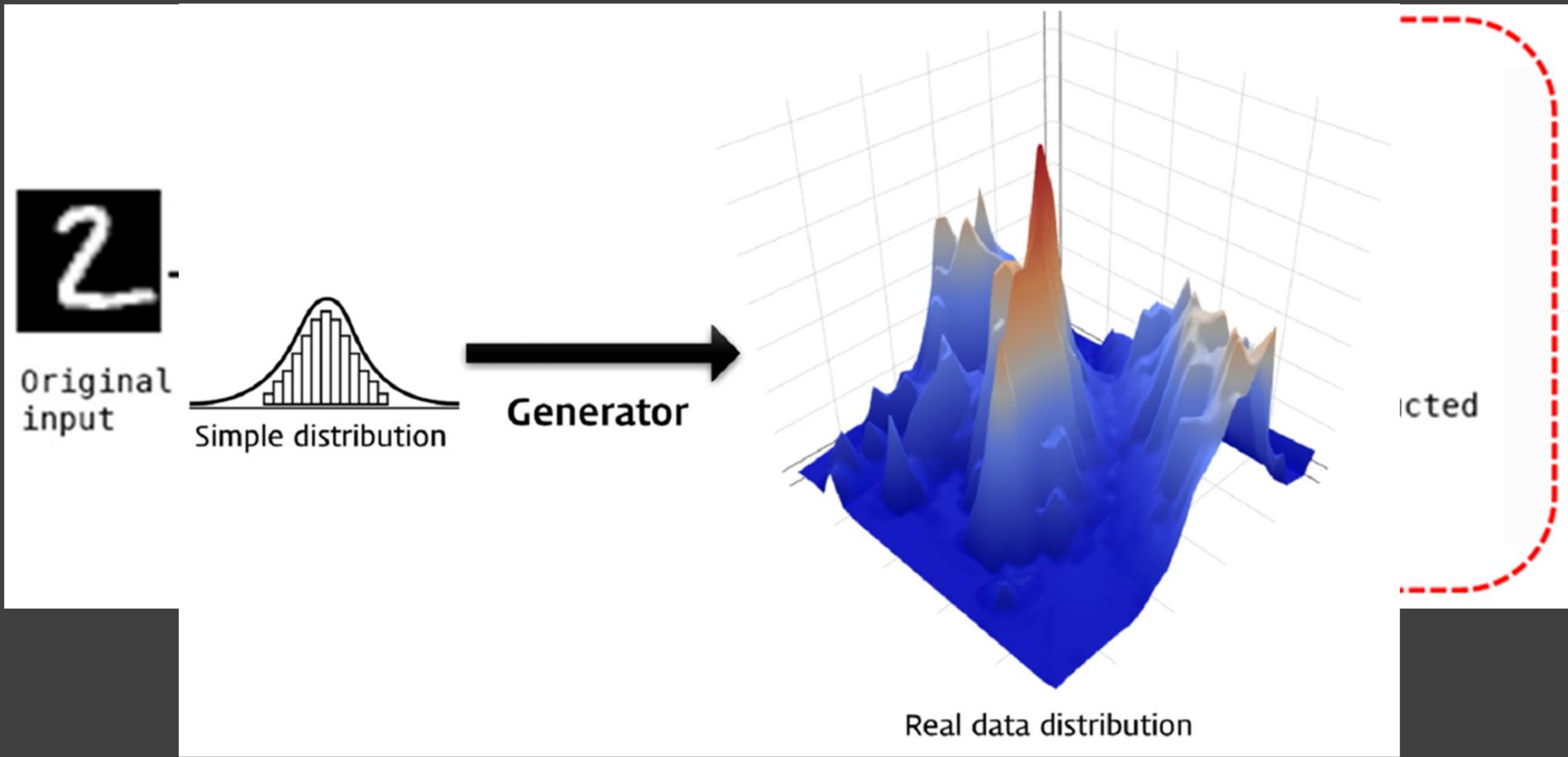
**판별자가 생성자의 분포를 실제 데이터의 분포를 모방하도록 유도하는 모델**



# Role of Generator

간단한(압축된) 정보를 복잡한 차원의 분포로 Mapping

오토인코더(GAN 이전의 Generation)



# Role of Discriminator

진짜일까, 가짜일까

# Object Function

문제의 목적을 나타내는, 최소화하거나 최대화할 함수

## Discriminator : 목적 함수를 최대화

Sample x from real data distribution

Sample latent code from Gaussian distribution

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

D should maximize V(D,G)

Maximum when D(x)=1, D(x)는 확률

Maximum when D(x)=0

## Generator : 목적 함수를 최소화

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

D should minimize V(D,G)

G is independent of this part

Maximum when D(G(z))=1

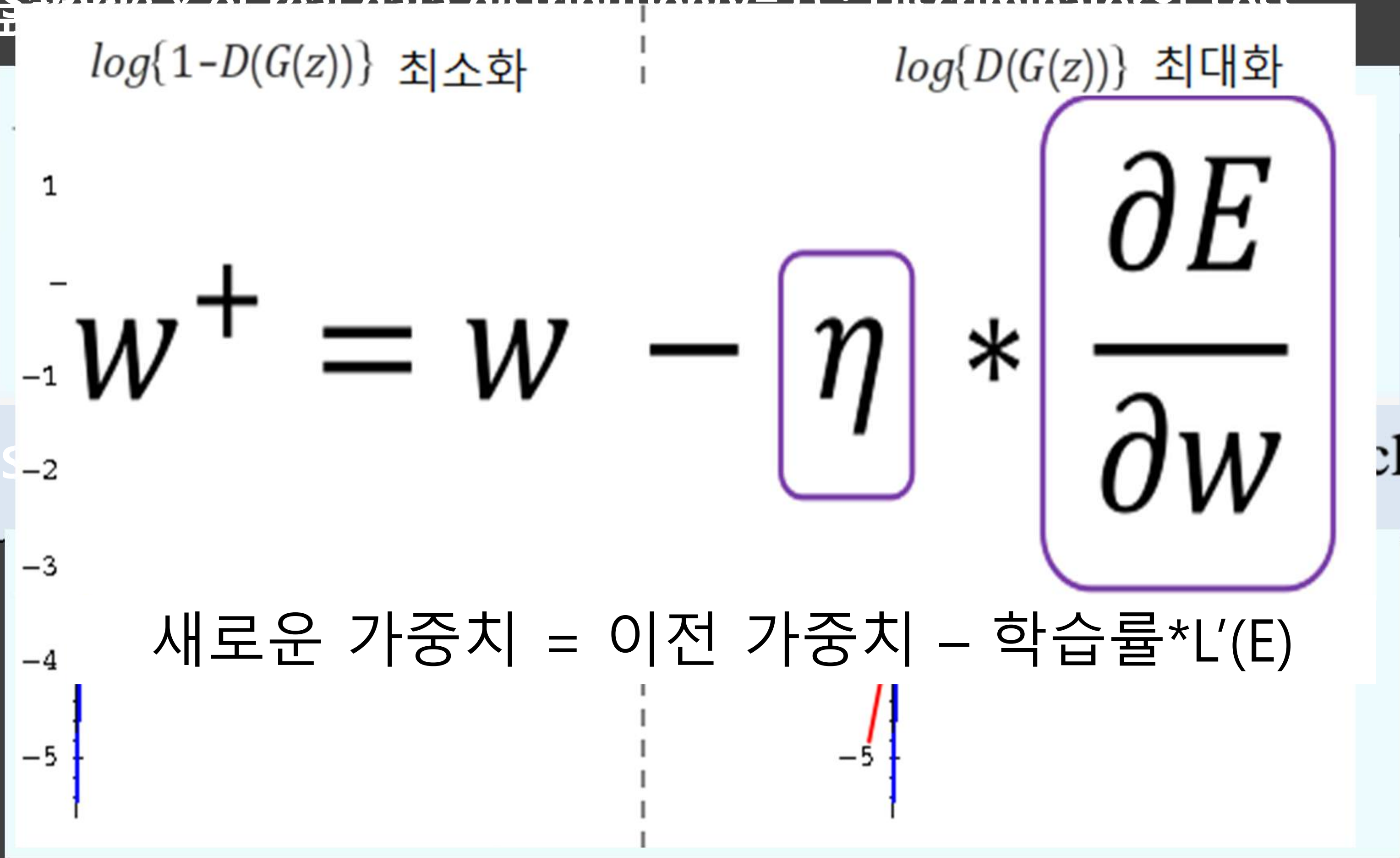
# Loss Function

**오차에 대해 페널티의 크기를 정하는 함수**

**Generator의 Loss function Graph**

**Sample x of real data distribution(y=1) : Discriminator의 Loss**

**Gan 모**




$$w^{+} = w - \boxed{\eta} * \boxed{\frac{\partial E}{\partial w}}$$

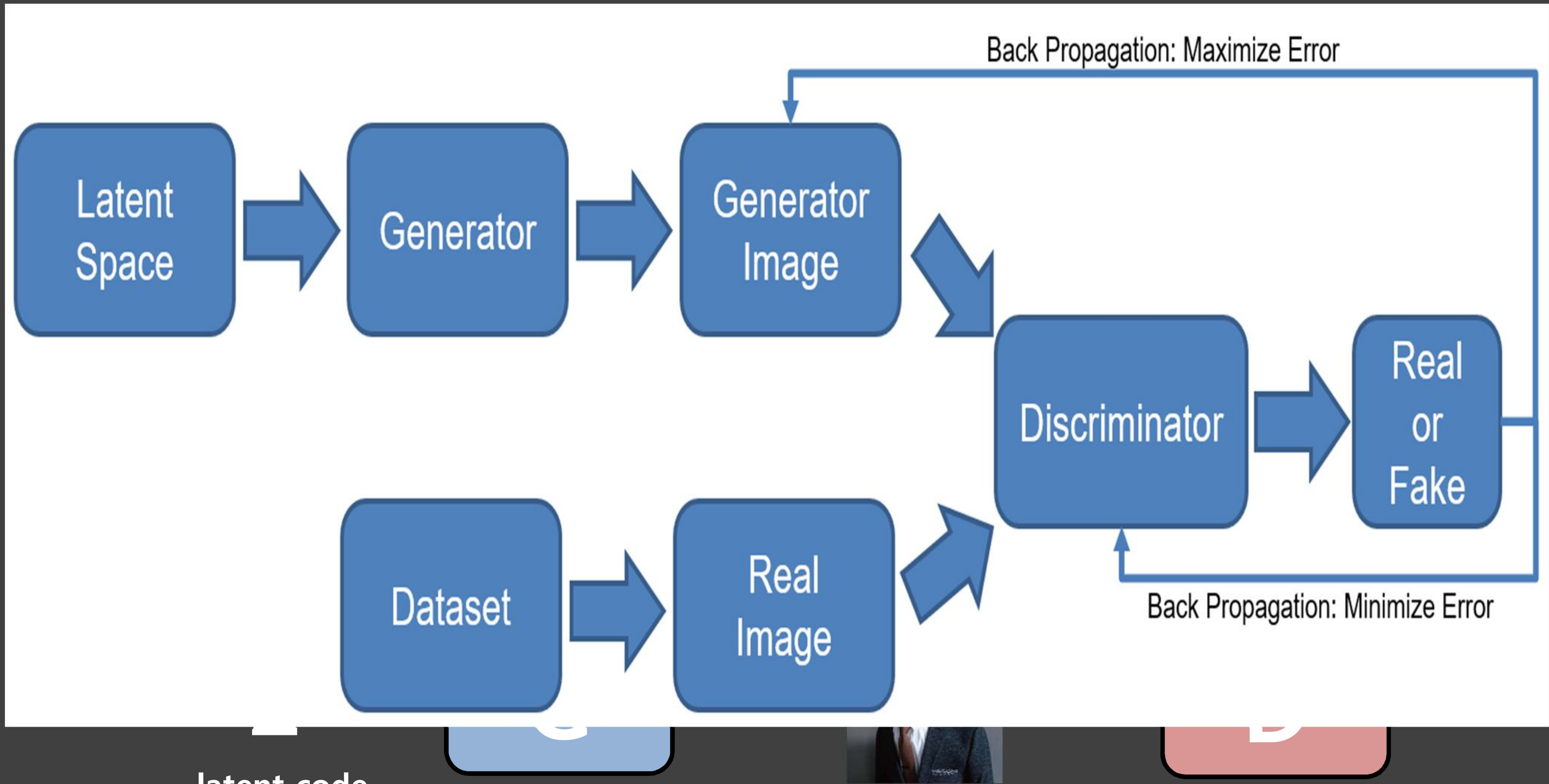
새로운 가중치 = 이전 가중치 – 학습률*L'(E)

whe

class

# Train of GAN

## 초기 GAN의 학습방법




latent code
(N개)

Fake Image

판별자의 정확도를 최소화

# 03
# Development of GAN

# Fake or Real?

**이제 인간은 실제와 가짜를 구분할 수 없다.**

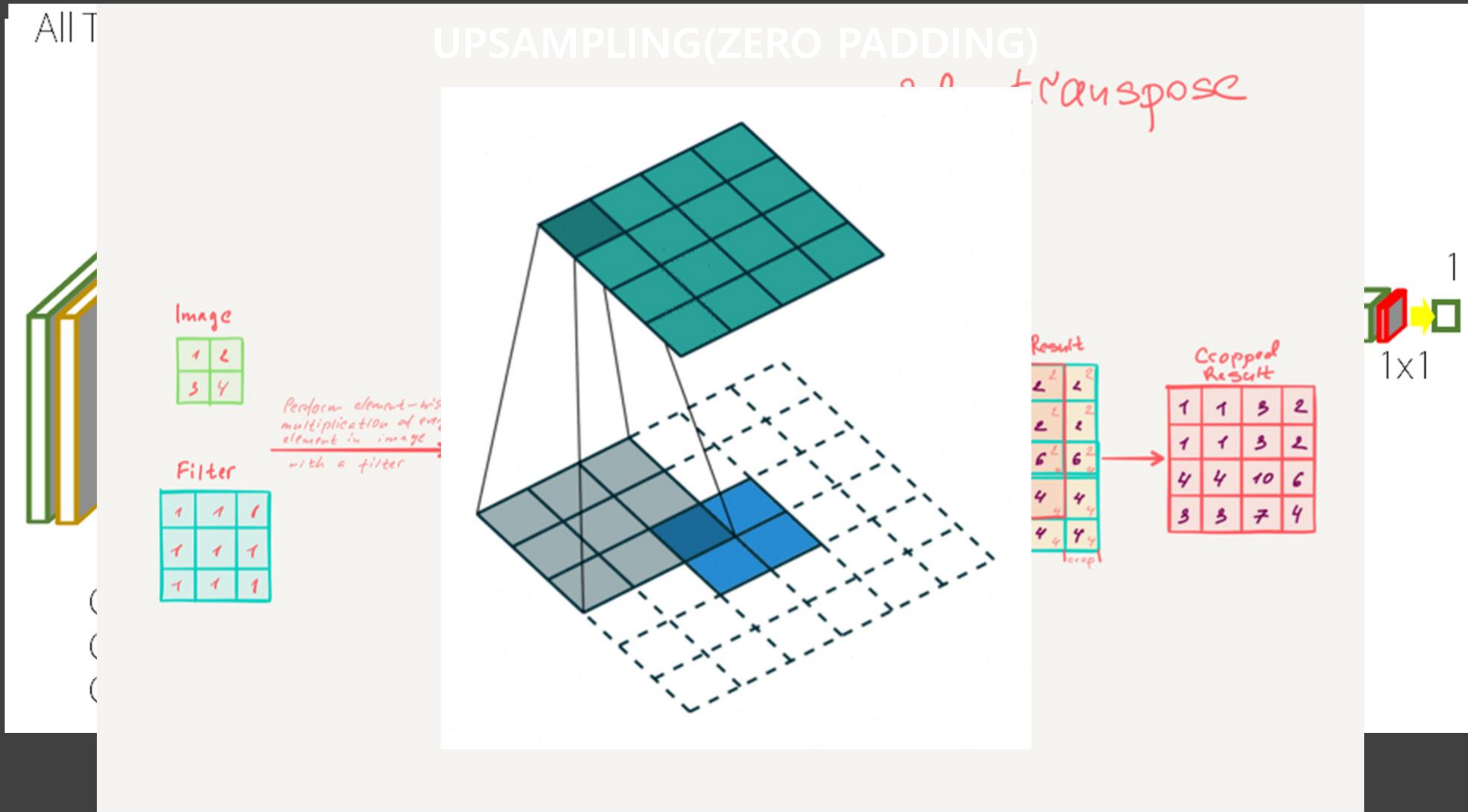
# DC GAN

## Deep-Convolution GAN

### DISCRIMINATOR

### GENERATOR

### 2D CONVOLUTION TRANSPOSE

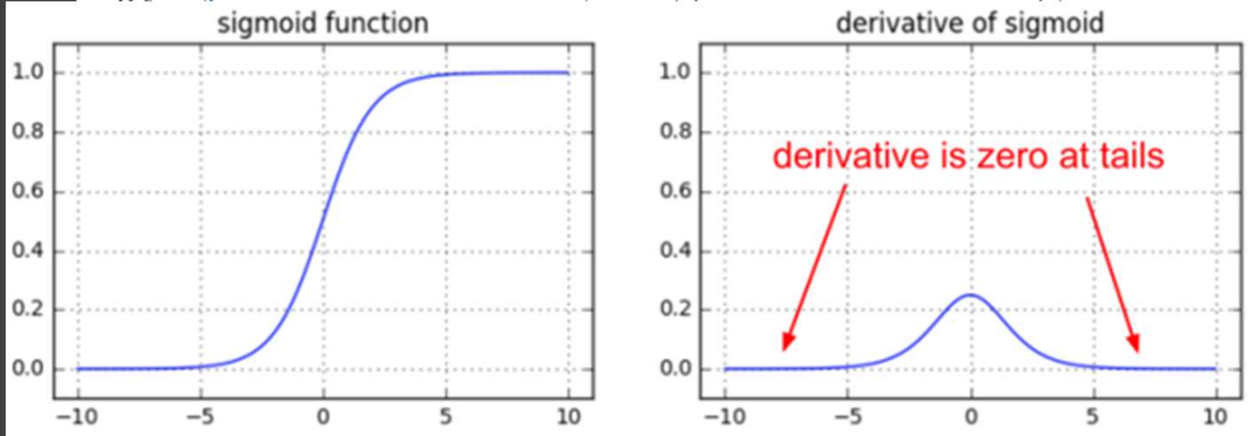


# Ls gan

## Least Square Gan

$$\max_{D} V(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

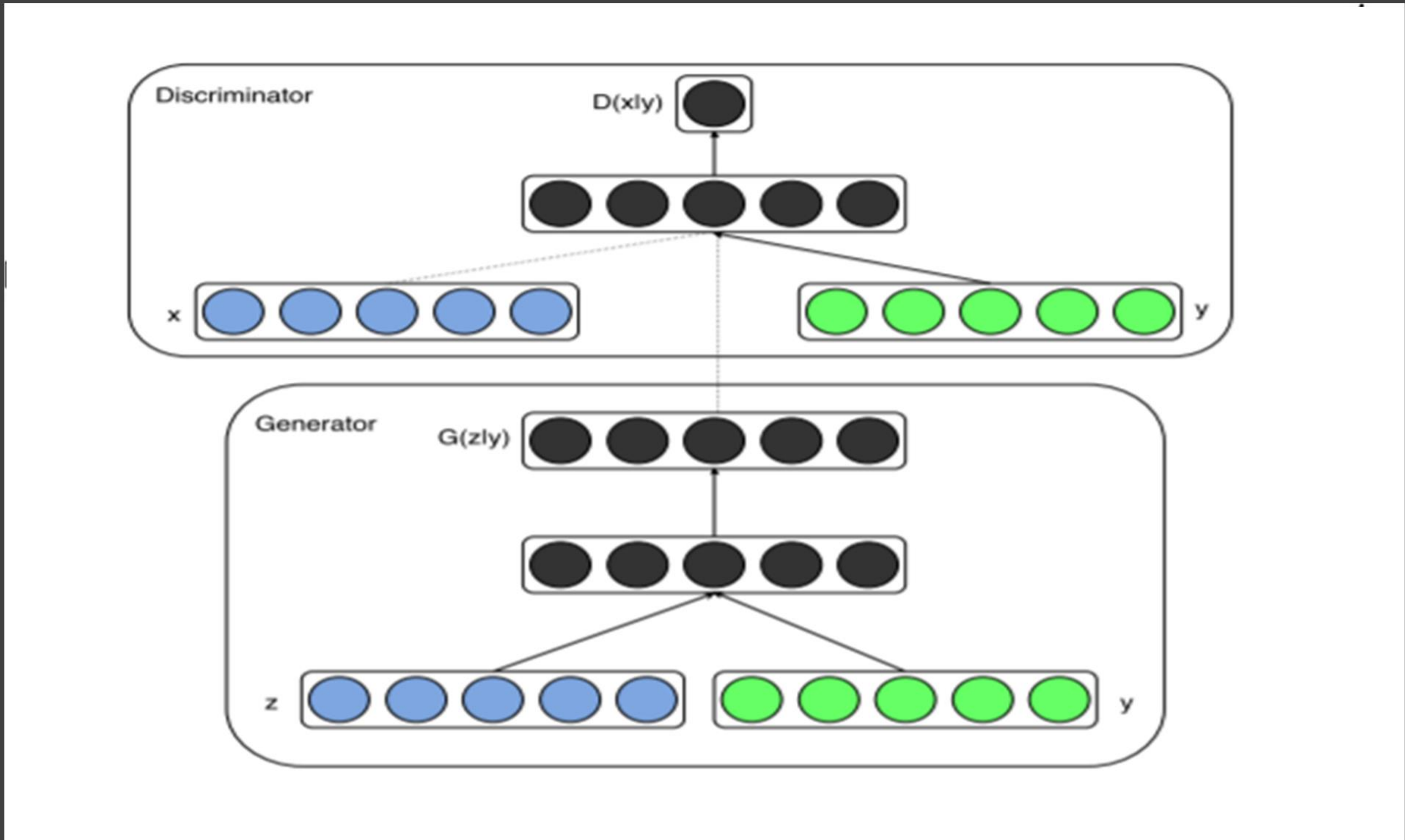
$$\min_{G} V(G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \textbf{ or } \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})} - [\log D(G(\boldsymbol{z}))]$$



# C-GAN

## Conditional GAN

Random한 출력에 Label을 추가하여 Output의 class를 선택



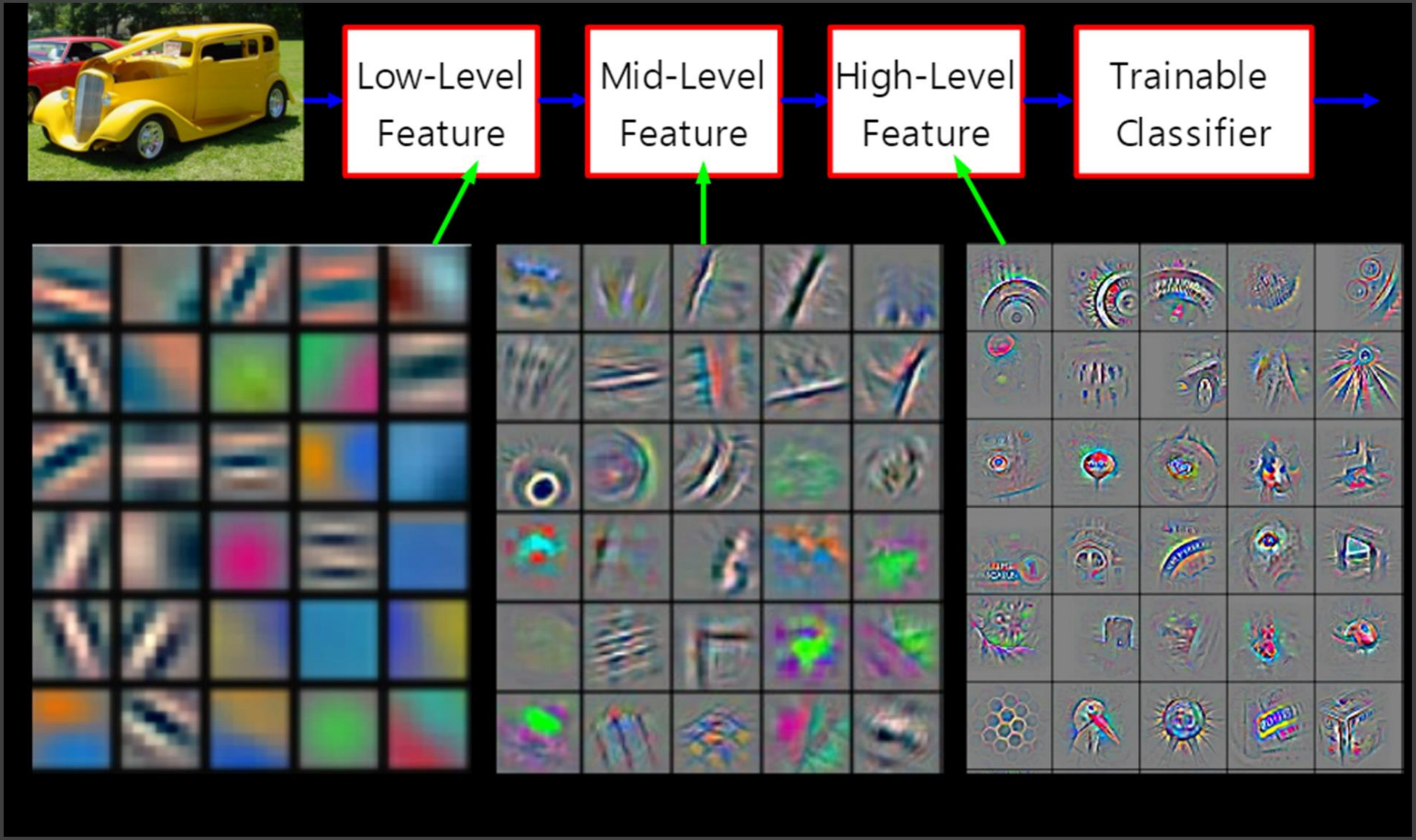
# Generative Adversarial Text to Image Synthesis(합성)

Text encoding 등

# SA-GAN

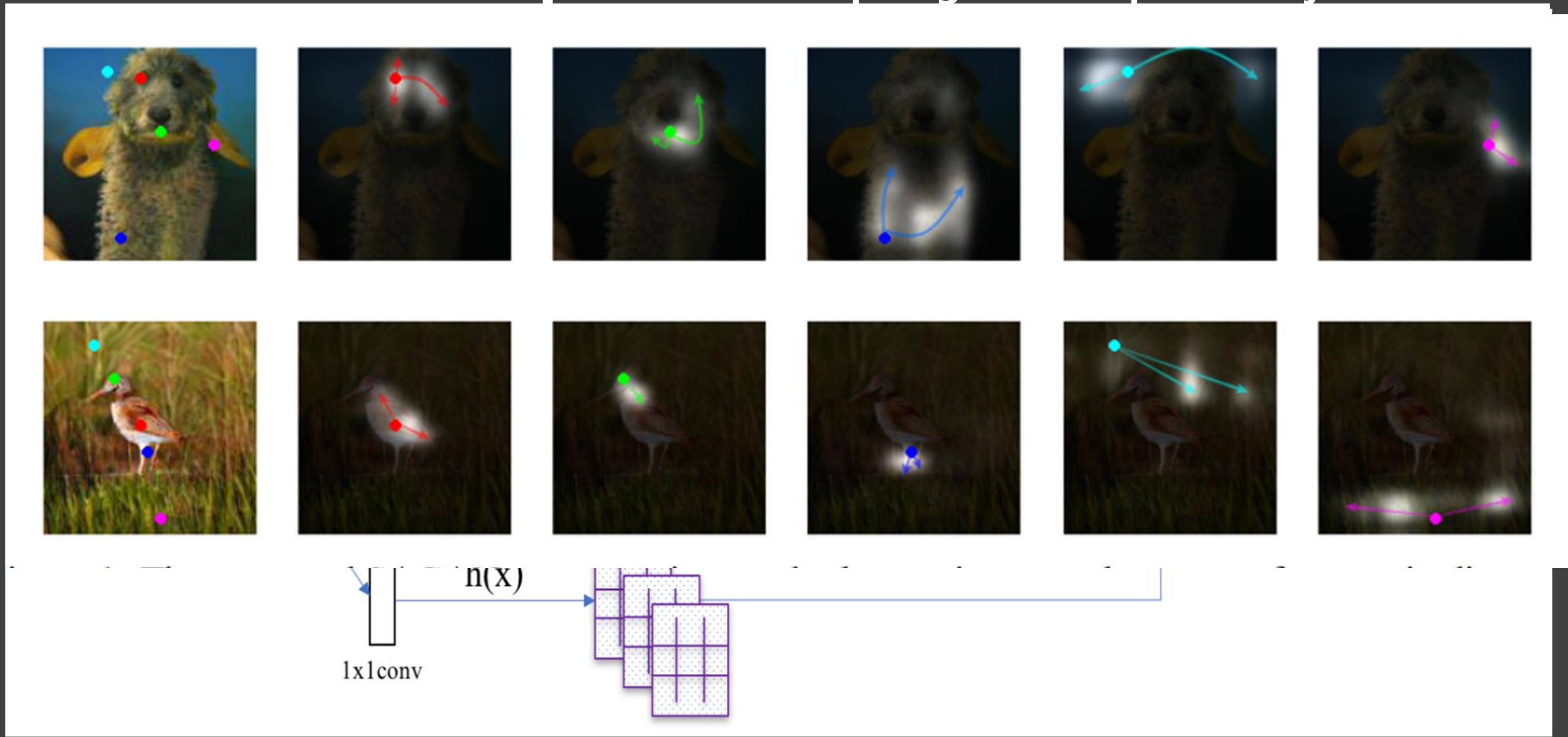
## Self-Attention GAN

Convolution layer는 local receptive field를 가진다. 하지만..



# SA-feature maps

Self-Attention feature map으로 생성된 feature map의 global dependency를 학습



# Latest GAN Application

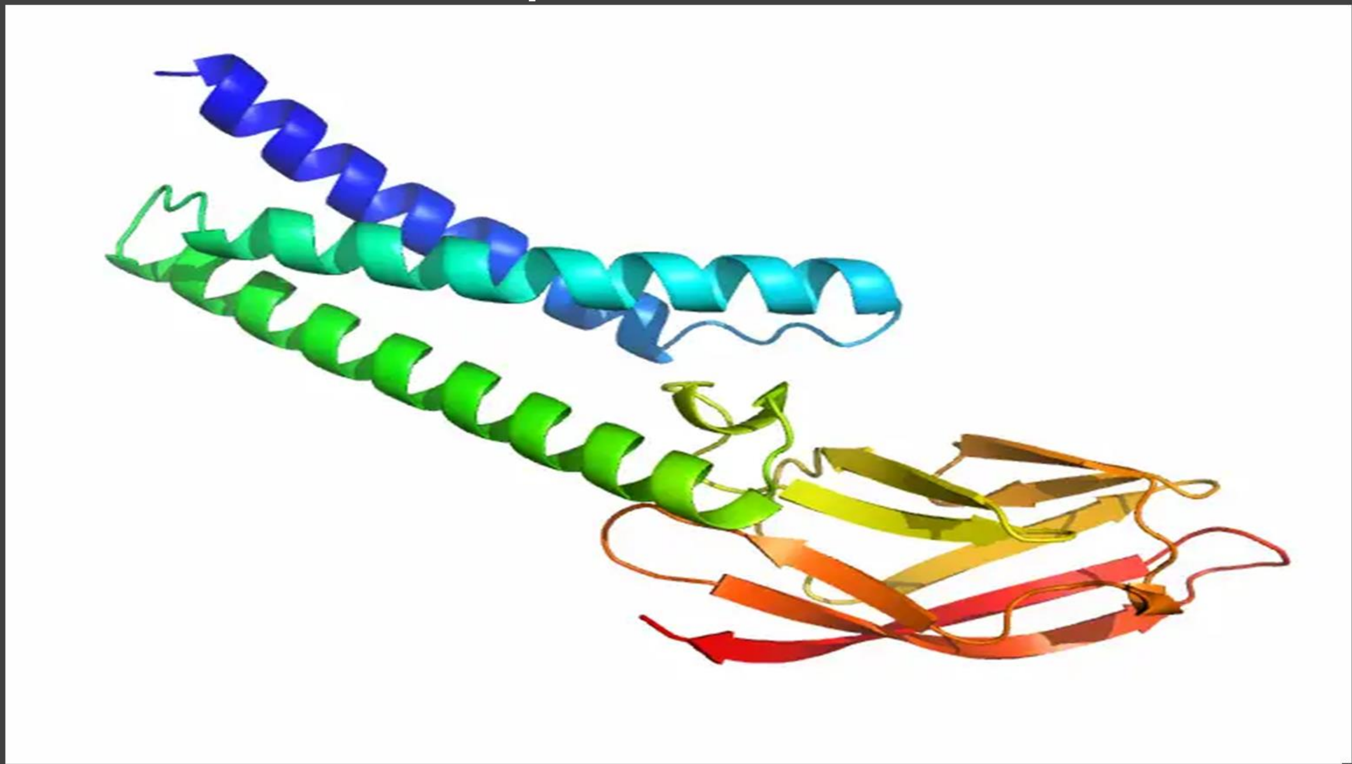
## GAN 개념이 적용된 최신 기술

가짜 오바마 연설
Nvidia S-motion

# Latest GAN Application

## GAN 개념이 적용된 최신 기술

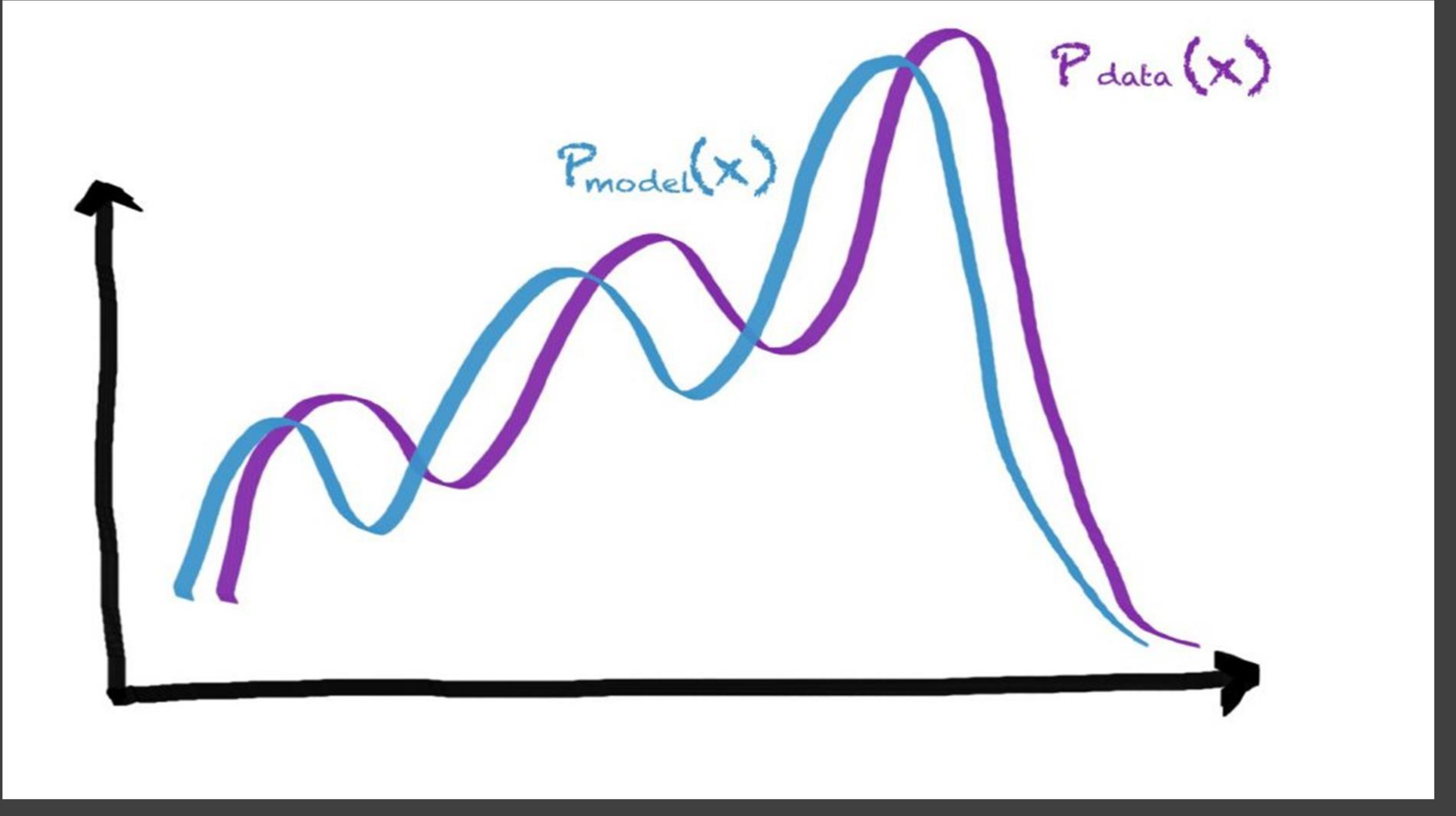
딥마인드의 알파폴드(AlphaFold)의 코로나 바이러스 형태 예측



# Final Goal of GAN

P(Generator) = P(Data)

어떻게 JSD를 최소화 할 것인가?

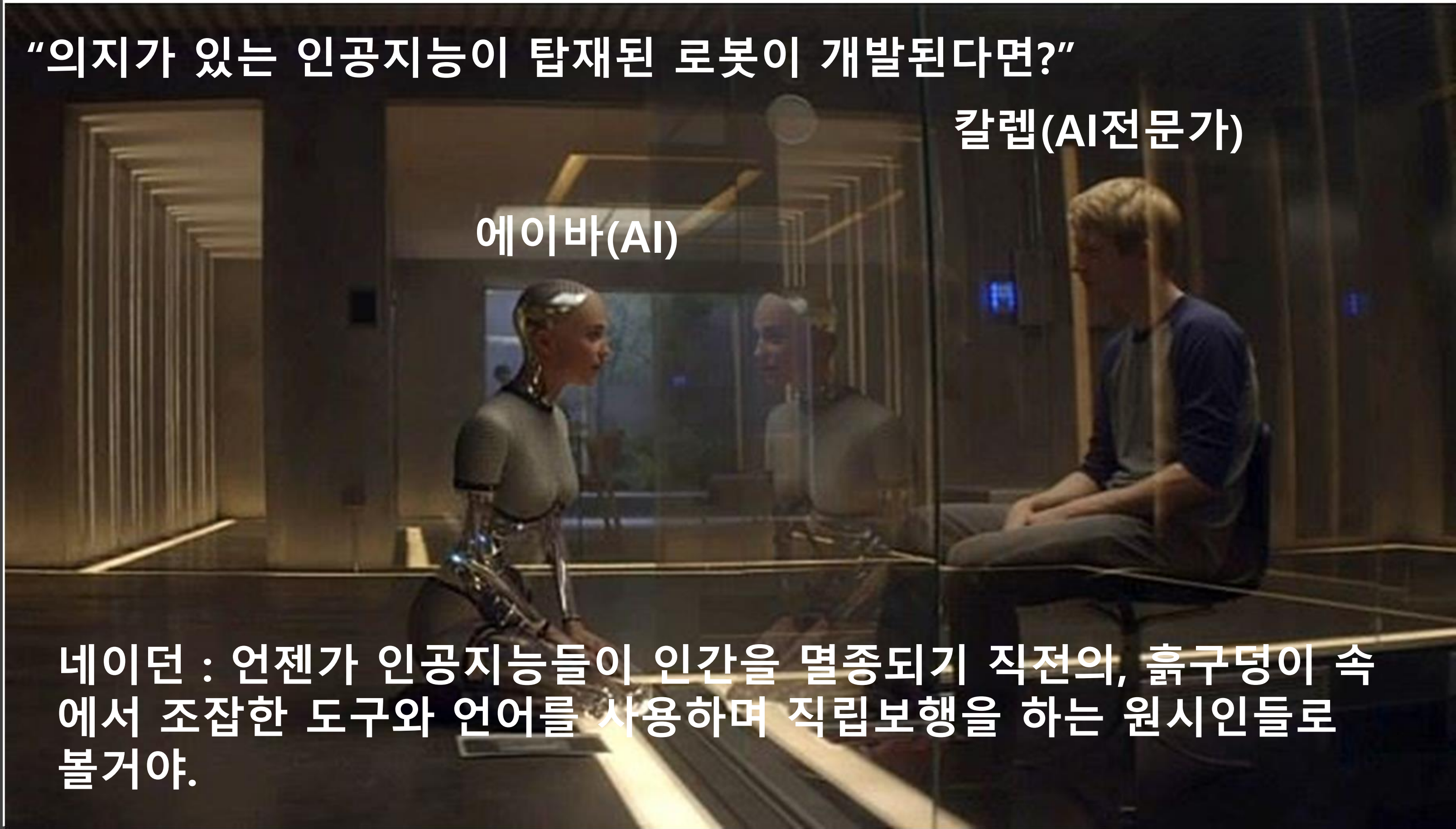


# 04
# Future of Generative Model

# Deus Ex Machina

(God from the machine)

[illegible] 일부분을 개척(약인공지능)



# 05
# 찐개만 with Kaggle

# 찐개만

## (진(찐)짜 개 같이 만들어보자)

Generative Dog Images of Kaggle-Competition

# 데이터 수집

Stanford Dogs Dataset
Number of categories : 120
Number of images : 20,580
Annotations : Class labels, Bounding boxes

# Data preprocessing

**Annotation의 x1,x2,x3,x4를 이용하여 개 이미지만 출력**

원본 이미지


이미지 크롭

32x32

64x64

# GAN의 성능 끌어올리기

## 어림짐작(Heuristics) for Training Stable GANs

- Downsample Using Strided Convolutions
- Upsample Using Strided Convolutions
- Use LeakyReLU
- Use Batch Normalization
- Use Gaussian Weight Initialization
- Use Adam Stochastic Gradient Descent
- Use a Gaussian Latent Space
- Separate Batches of Real and Fake Images

# Implementation with Pytorch

간단하게 설명하기 위한 모델

```python
import torch
import torch.nn as nn
```

Module import

```python
# Assum x be real images of shape(batch_size, 784)
# Assum z be random noize of shape(batch_size, 100)
```

Input에 대한 정보

```python
while True:
    # train D
    loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
    loss.backward()
    d_optimizer.step()
```

D의 학습

```python
    # train G
    loss = criterion(D(G(z)), 1)
    loss.backward()
    g_optimizer.step()
```

G의 학습

```python
d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)
```

최적화 모델 정의

# Dcgan 모델 구현(32x32)

**Discriminator**

**Input=(None, 32, 32, 3)**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 16, 16, 64) | 4864 |
| leaky_re_lu_1 (LeakyReLU) | (None, 16, 16, 64) | 0 |
| dropout_1 (Dropout) | (None, 16, 16, 64) | 0 |
| conv2d_2 (Conv2D) | (None, 8, 8, 128) | 204928 |
| leaky_re_lu_2 (LeakyReLU) | (None, 8, 8, 128) | 0 |
| dropout_2 (Dropout) | (None, 8, 8, 128) | 0 |
| conv2d_3 (Conv2D) | (None, 4, 4, 256) | 819456 |
| leaky_re_lu_3 (LeakyReLU) | (None, 4, 4, 256) | 0 |
| dropout_3 (Dropout) | (None, 4, 4, 256) | 0 |
| conv2d_4 (Conv2D) | (None, 4, 4, 512) | 3277312 |
| leaky_re_lu_4 (LeakyReLU) | (None, 4, 4, 512) | 0 |
| dropout_4 (Dropout) | (None, 4, 4, 512) | 0 |
| flatten_1 (Flatten) | (None, 8192) | 0 |
| dense_1 (Dense) | (None, 1) | 8193 |
| activation_1 (Activation) | (None, 1) | 0 |

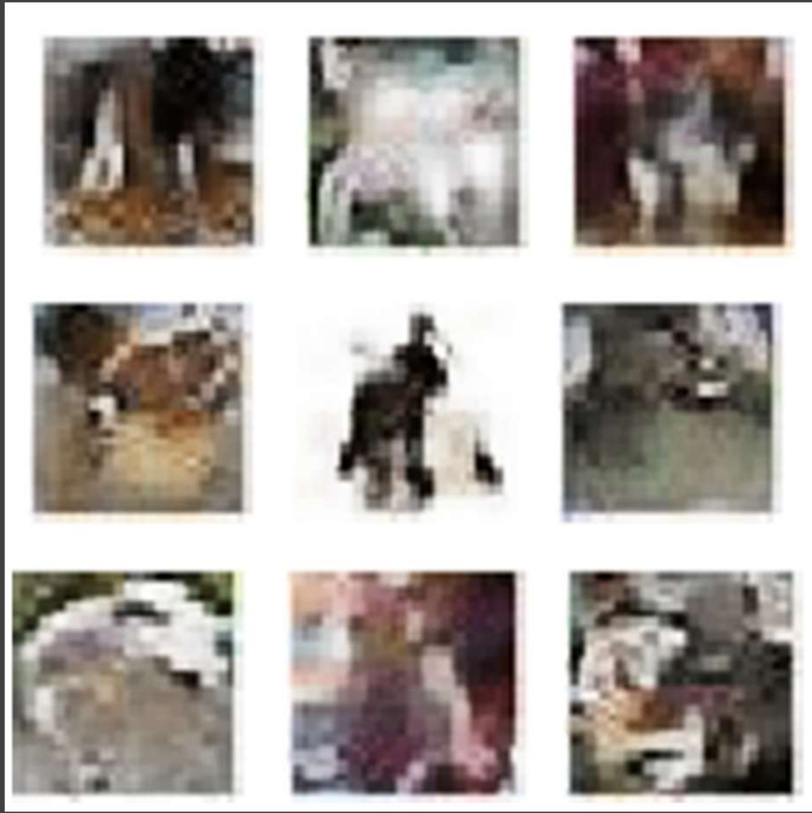
**Generator**

**Input=(None, 100)**

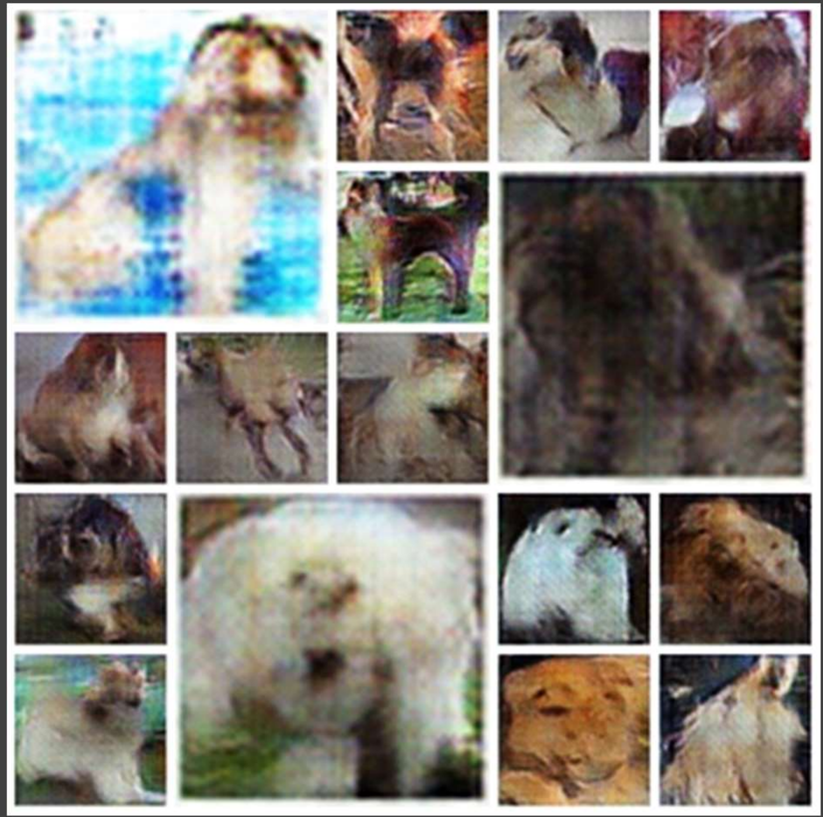
| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_2 (Dense) | (None, 16384) | 1654784 |
| batch_normalization_1 (Batch | (None, 16384) | 65536 |
| activation_2 (Activation) | (None, 16384) | 0 |
| reshape_1 (Reshape) | (None, 8, 8, 256) | 0 |
| dropout_5 (Dropout) | (None, 8, 8, 256) | 0 |
| up_sampling2d_1 (UpSampling2 | (None, 16, 16, 256) | 0 |
| conv2d_transpose_1 (Conv2DTr | (None, 16, 16, 128) | 819328 |
| batch_normalization_2 (Batch | (None, 16, 16, 128) | 512 |
| activation_3 (Activation) | (None, 16, 16, 128) | 0 |
| up_sampling2d_2 (UpSampling2 | (None, 32, 32, 128) | 0 |
| conv2d_transpose_2 (Conv2DTr | (None, 32, 32, 64) | 204864 |
| batch_normalization_3 (Batch | (None, 32, 32, 64) | 256 |
| activation_4 (Activation) | (None, 32, 32, 64) | 0 |
| conv2d_transpose_3 (Conv2DTr | (None, 32, 32, 32) | 51232 |
| batch_normalization_4 (Batch | (None, 32, 32, 32) | 128 |
| activation_5 (Activation) | (None, 32, 32, 32) | 0 |
| conv2d_transpose_4 (Conv2DTr | (None, 32, 32, 3) | 2403 |
| activation_6 (Activation) | (None, 32, 32, 3) | 0 |

# 모델 실행 결과

Dcgan 32x32

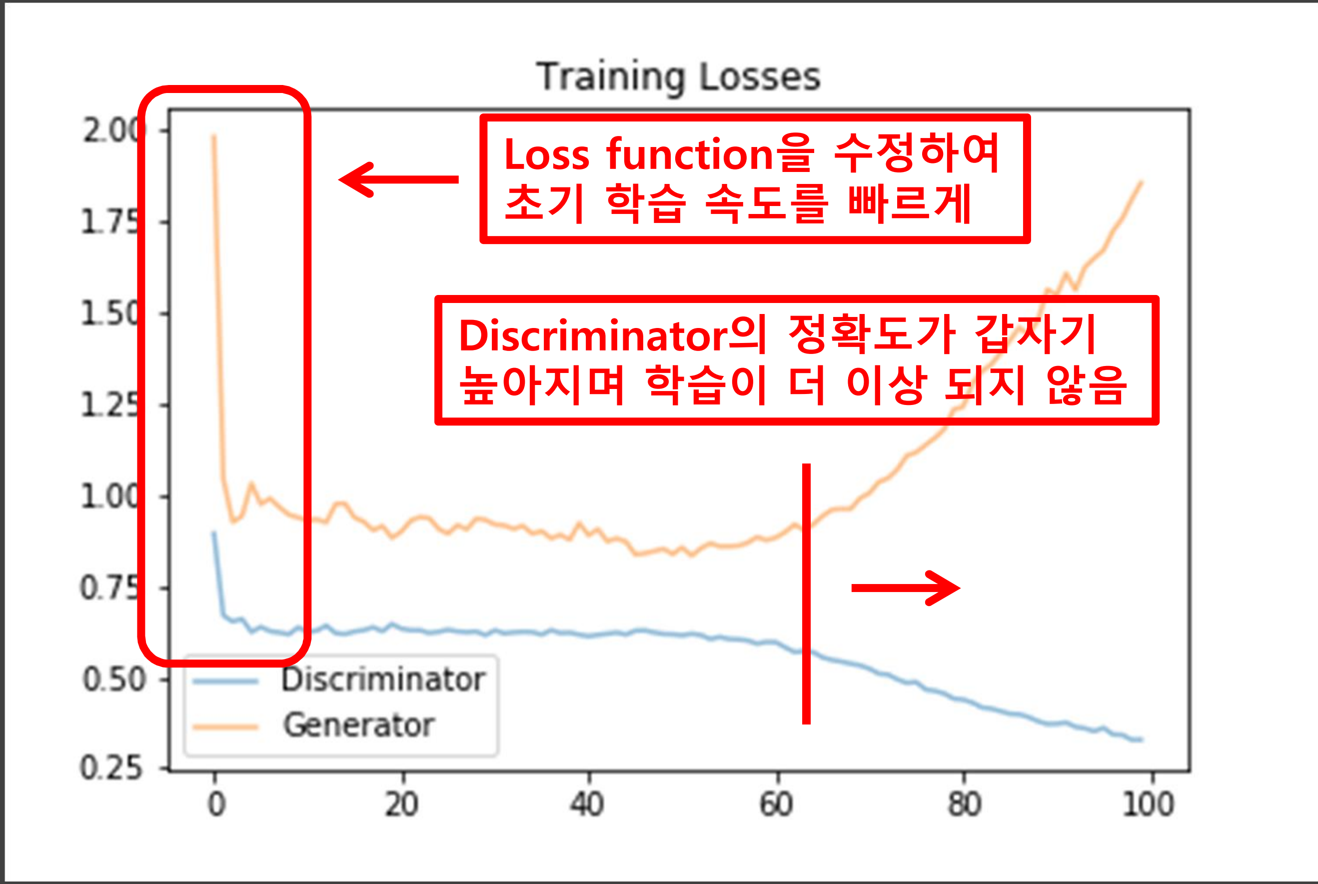


Dcgan 64x64



# 원인분석 with Loss

About 2000 Epoch



# 06
# 찐개만 with Better Code

# 최신의 GAN 모델로

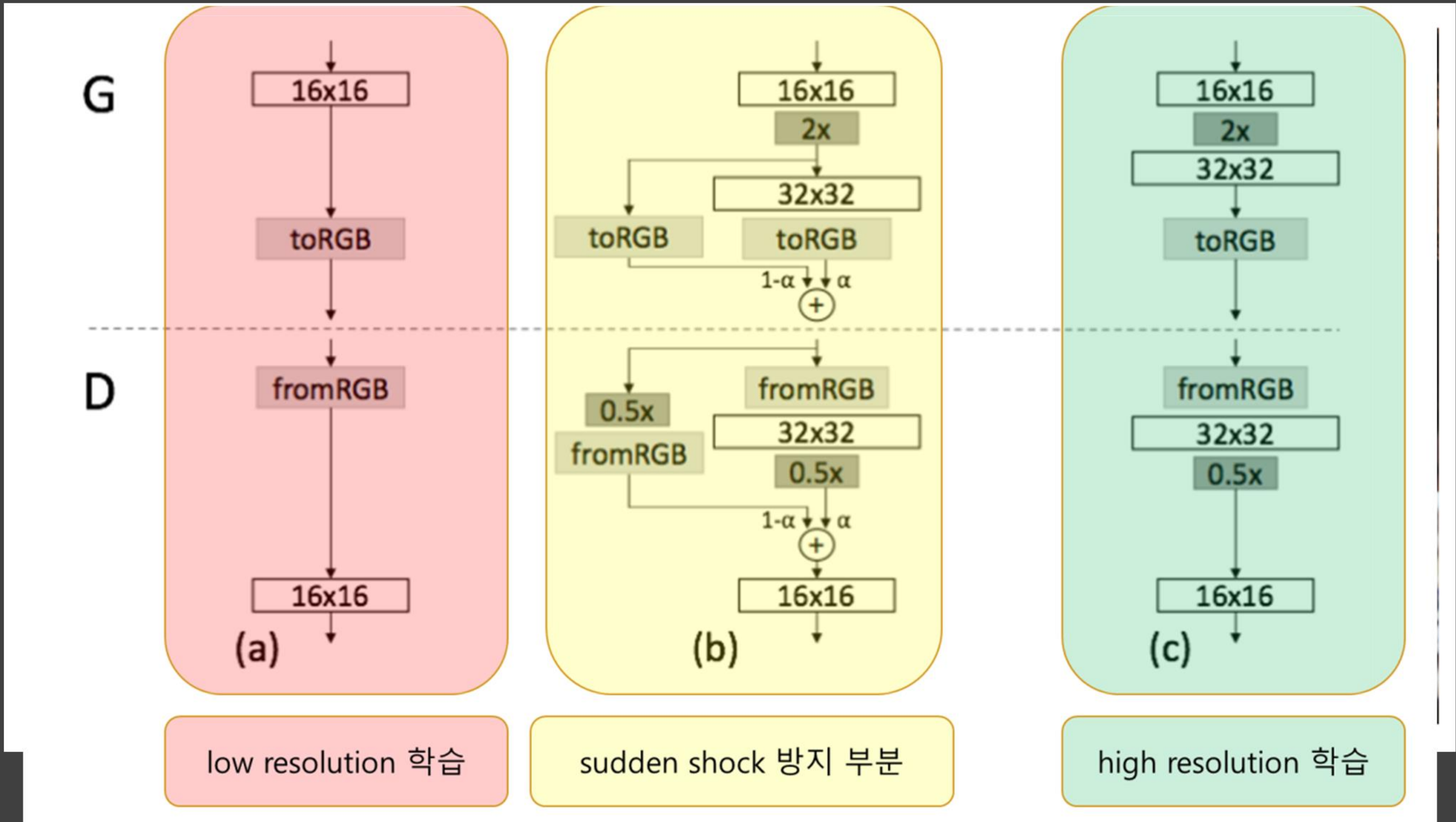
(무작정 돌려보기)

1. PRO C-GAN(1024x1024)
-> Resolution, batch_size, epoch 조정
2. BIG GAN(512x512)
-> Resolution, batch_size, epoch 조정

⇨ **64x64**

# Pro(PG)gan

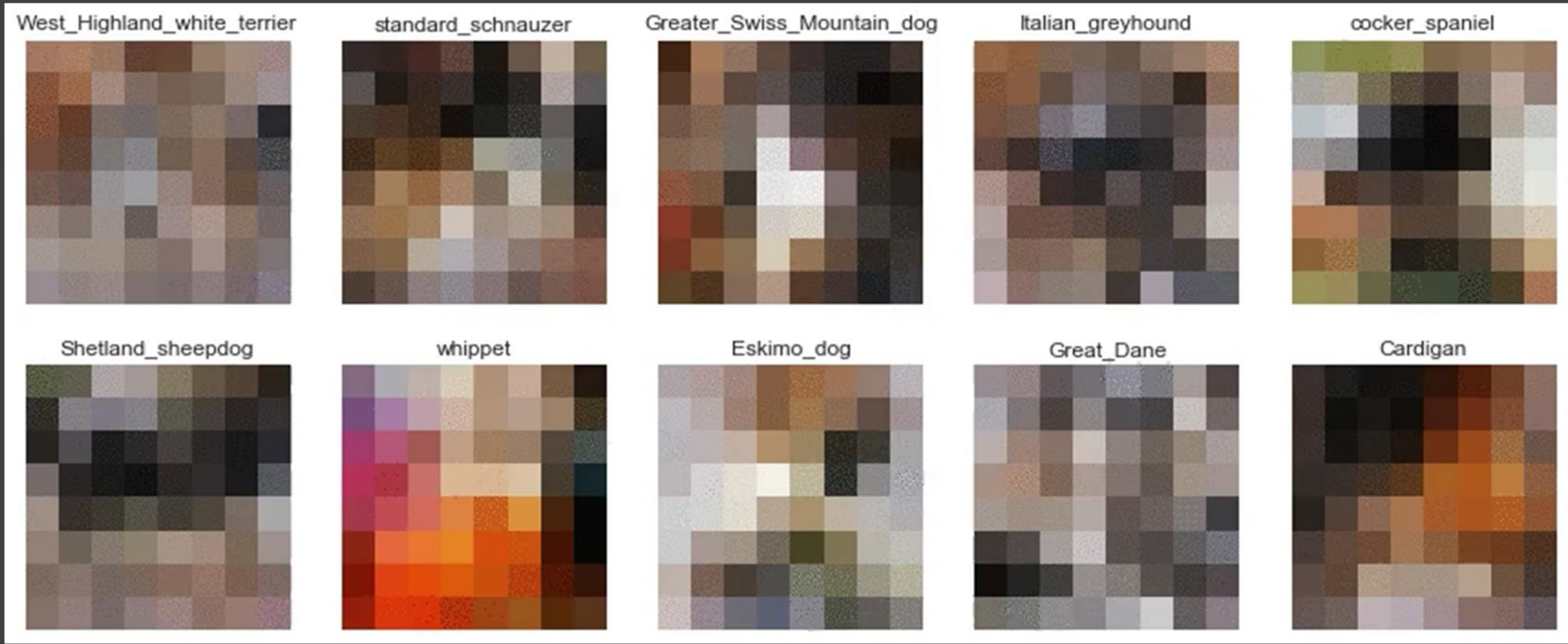
Progressive Gan(점진적인 Gan)



# Pro(PG)gan

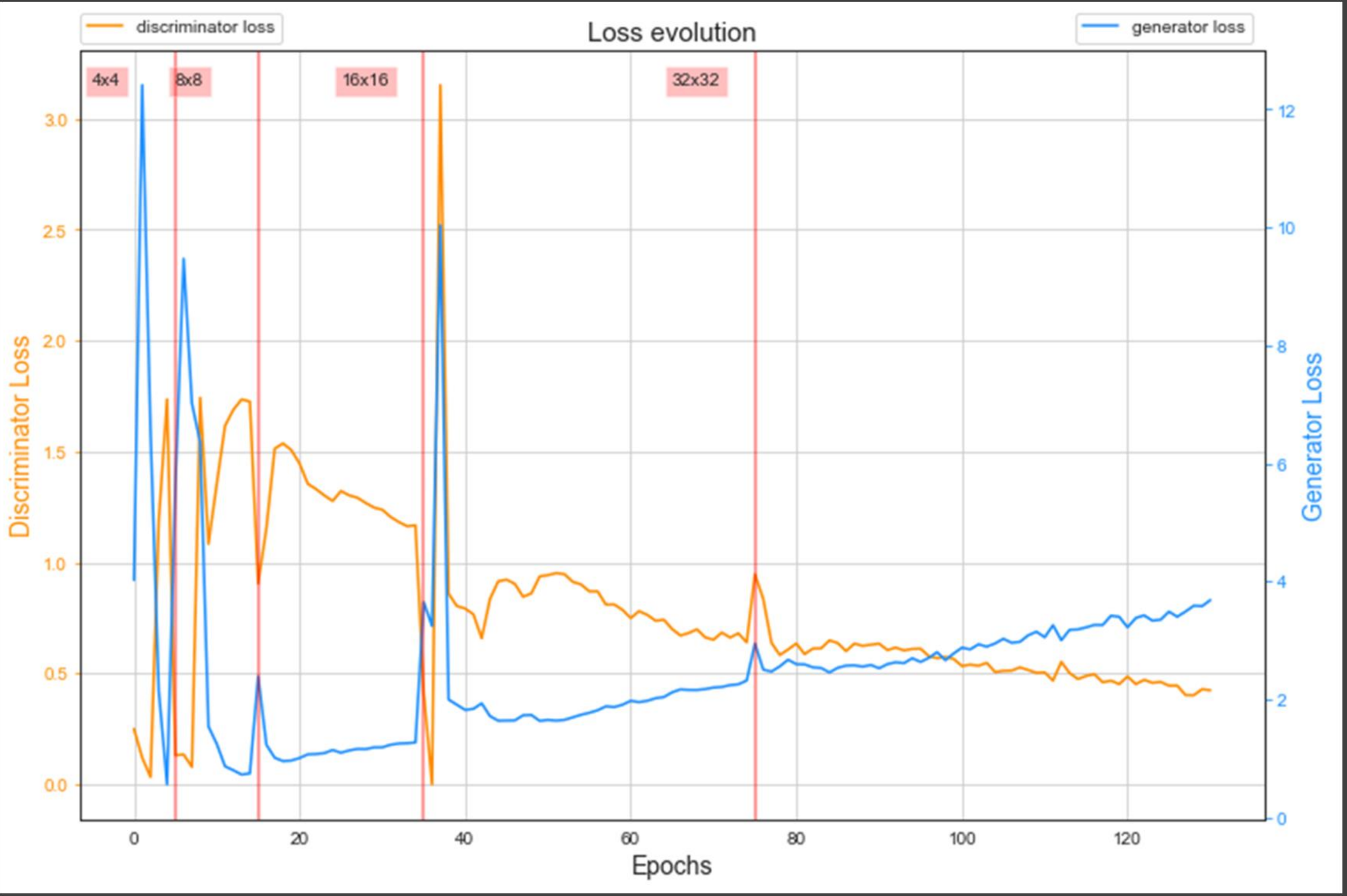
**Progressive Gan(점진적인 Gan)**

**Low Resolution -> High Resolution 으로 해상도를 높여가며 학습**



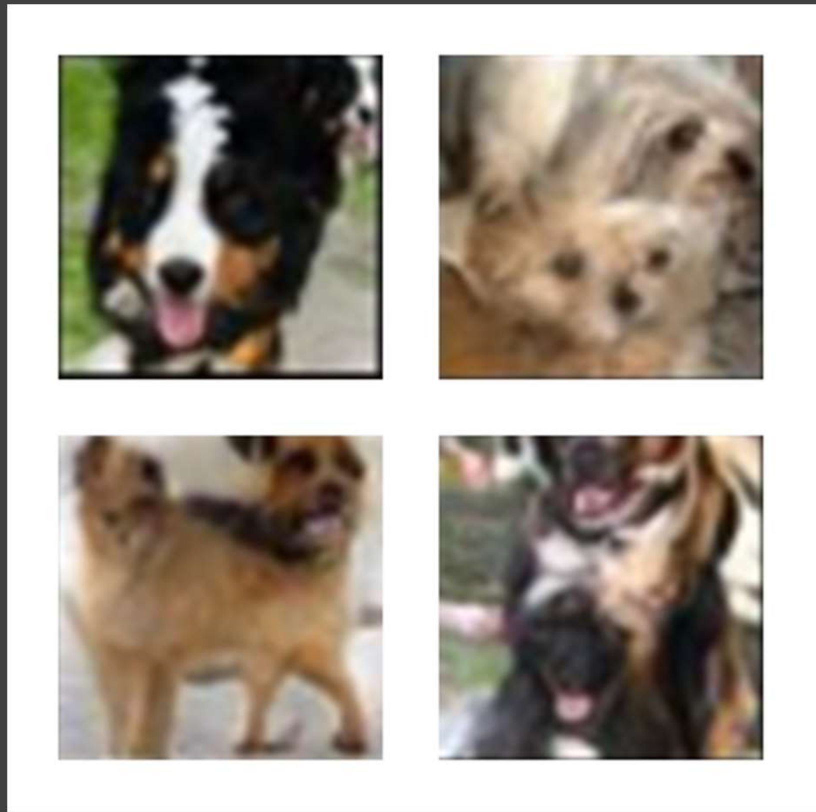
# Pro(PG)gan Loss

## Progressive Gan의 손실함수 분석



# Output

Bad example



Good example

# Big Gan

(Large scale Gan)

절단 트릭(Truncation Trick)

1. 구글의 딥

2. SA-GAN을

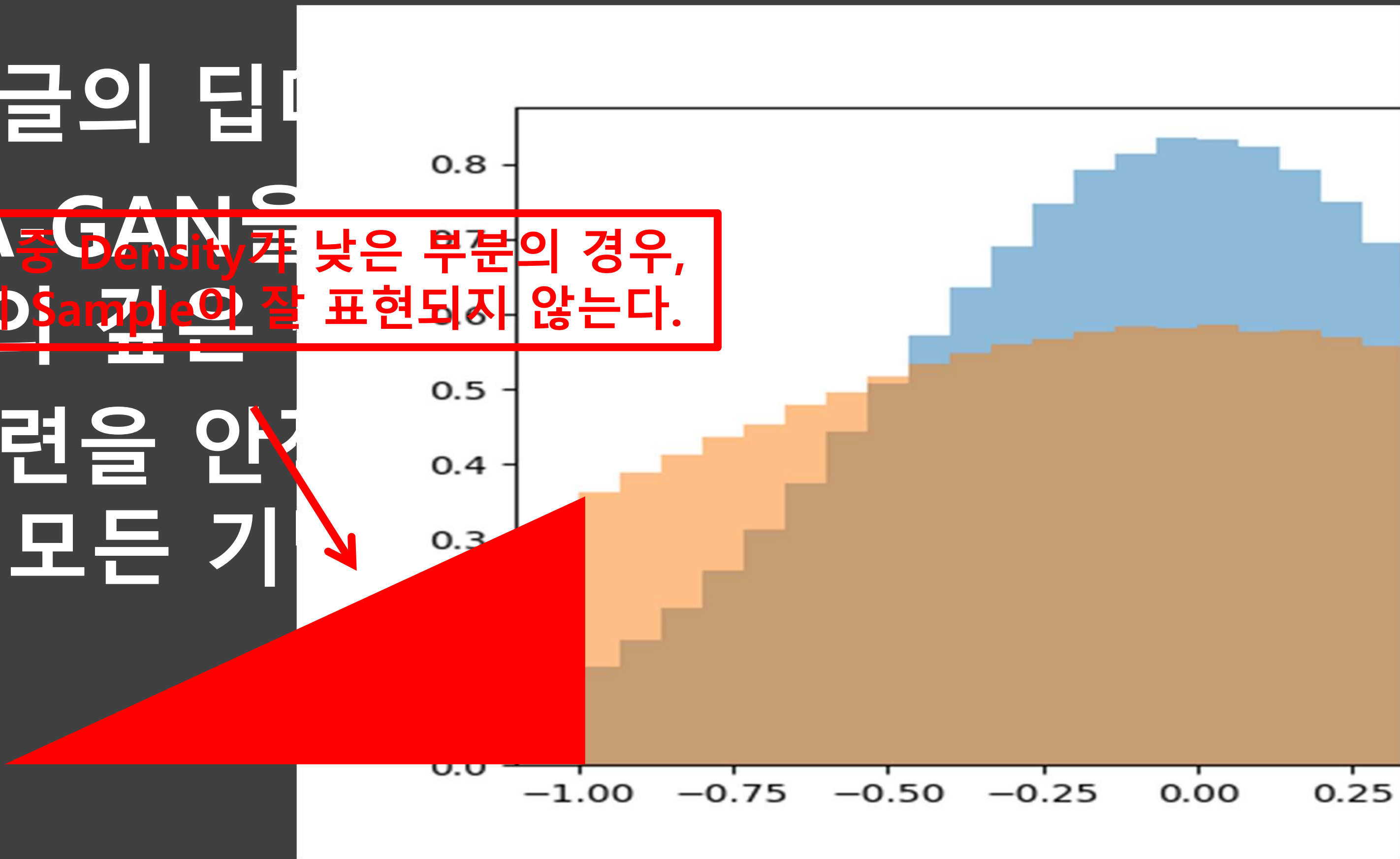
이상의 값은

3. 훈련을 안

능한 모든 기

확률분포 중 Density가 낮은 부분의 경우,

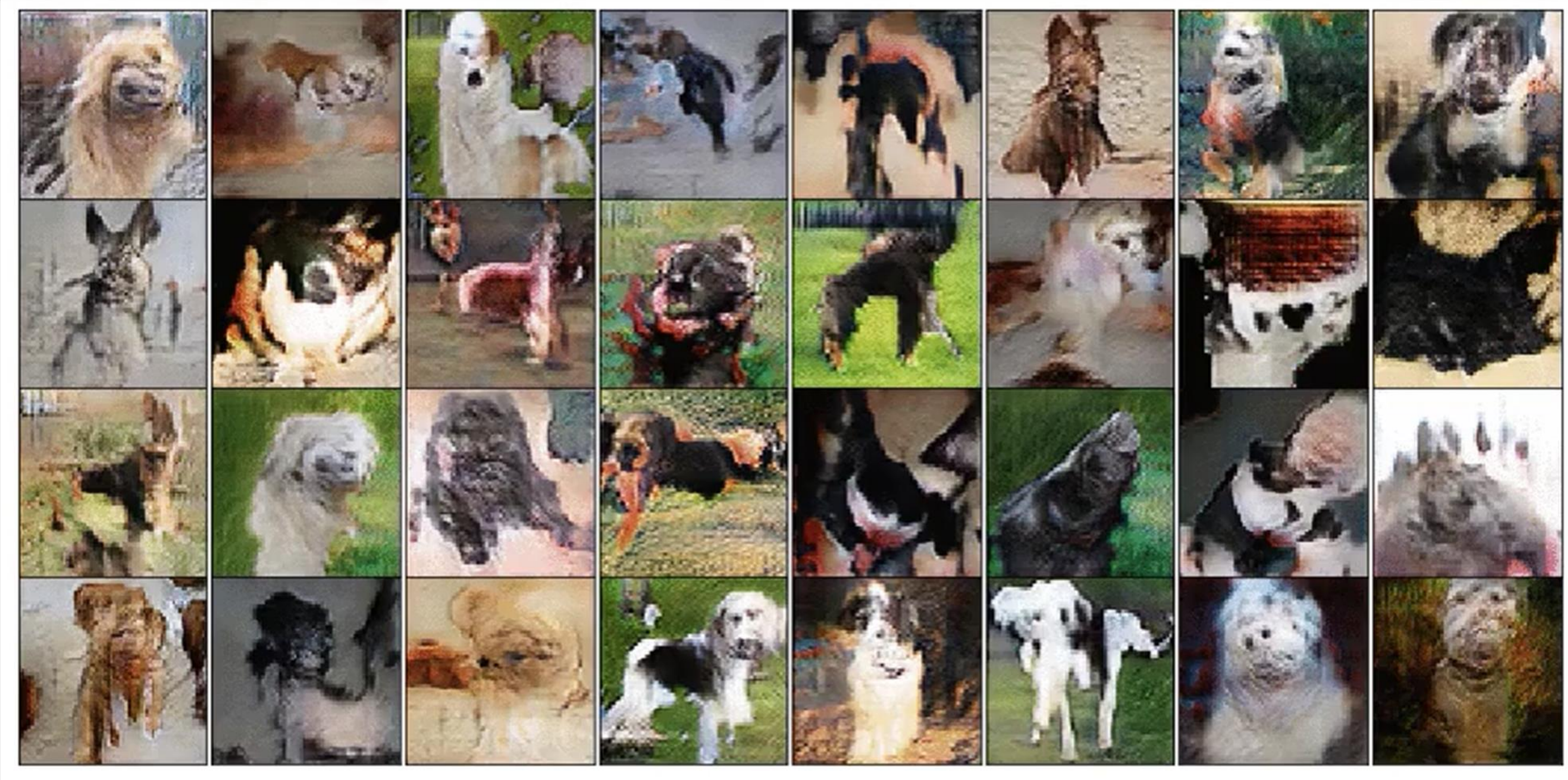
이 부분의 Sample이 잘 표현되지 않는다.



# Big Gan

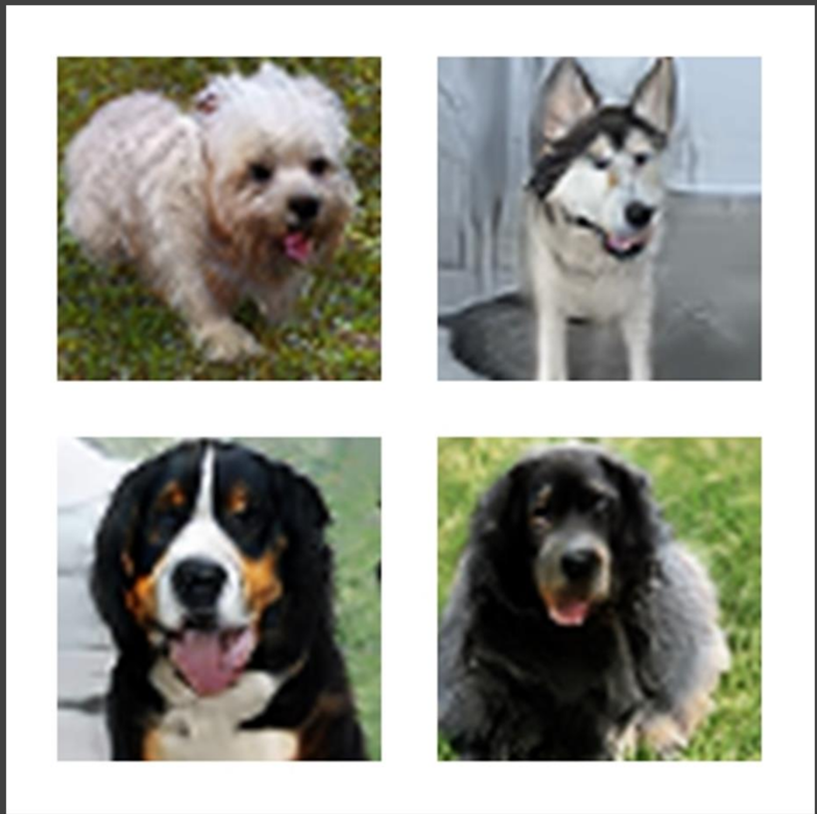
(Large scale Gan)

실제 모델의 훈련을 통해 Output의 품질을 개선하는 과정



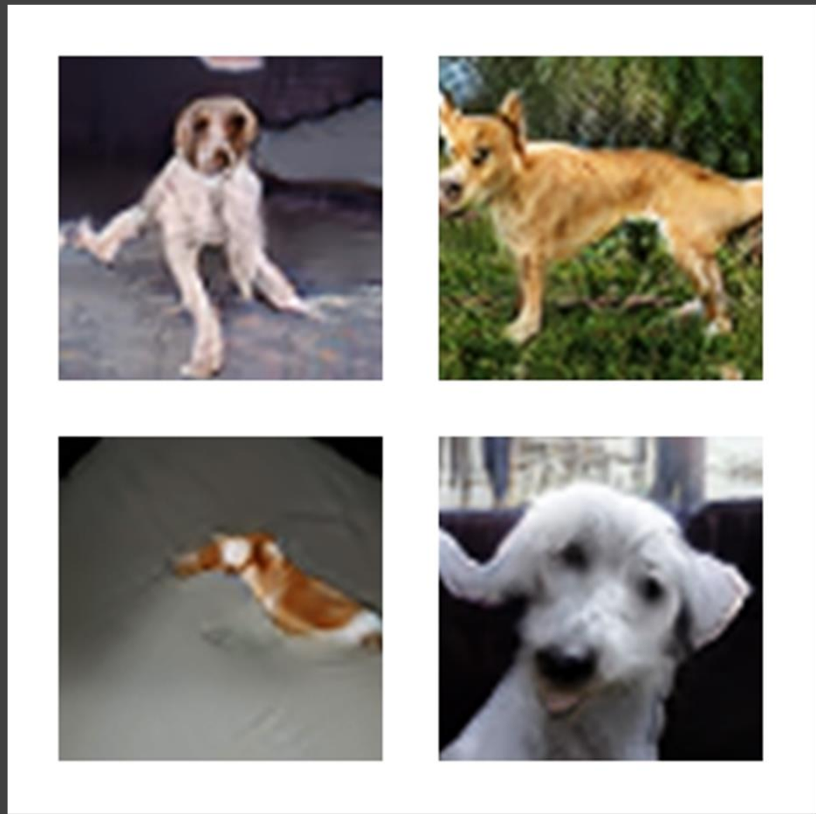
# Output

Good example



# Output

Bad example

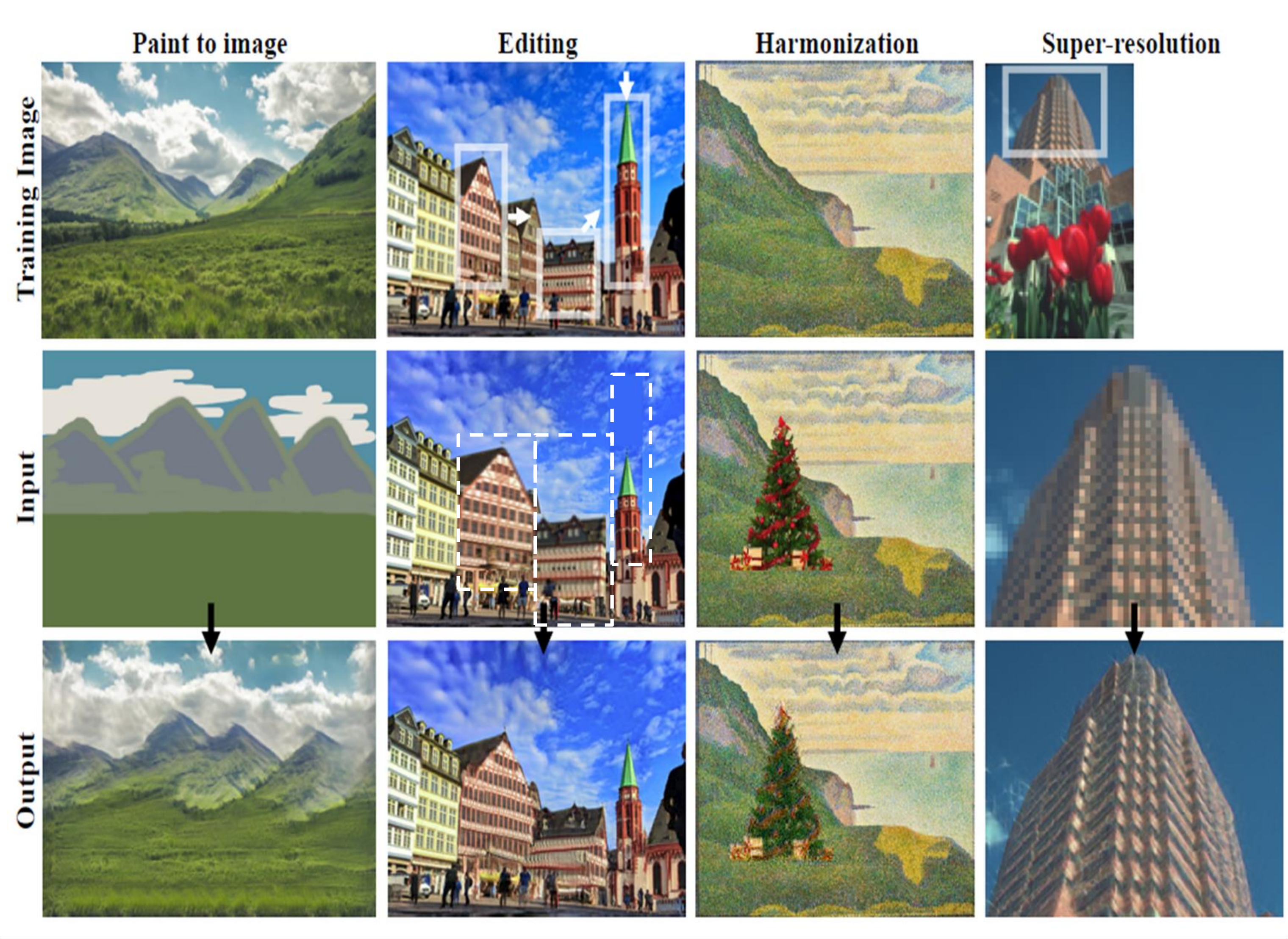


# 07
# Present of GAN
## (ICCV 2019 Seoul Best Award)
# Sin GAN

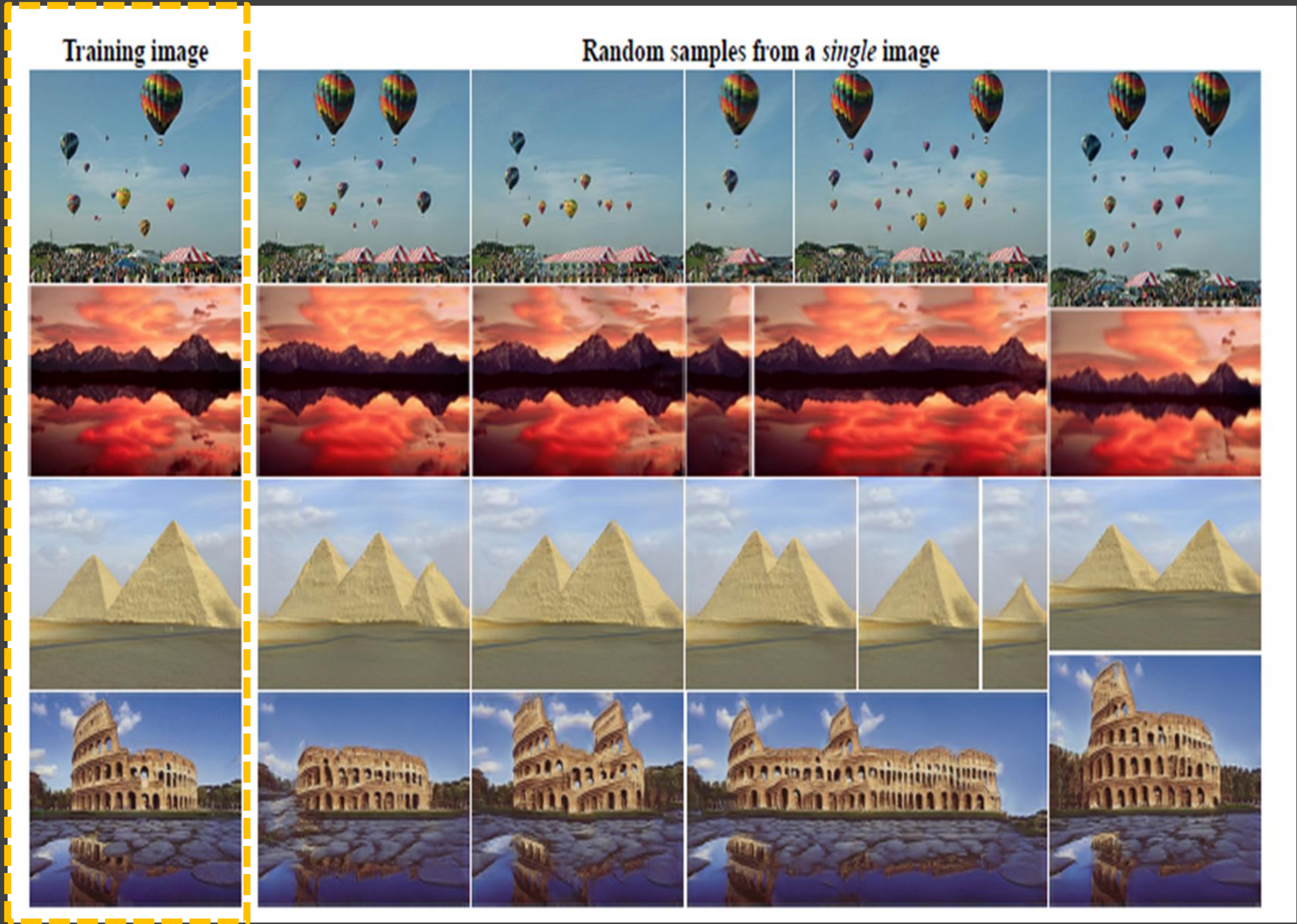
| GAN | SinGAN |
|---|---|
| Conditional (제한적)<br>e.g. 질감, 얼굴, 침실 등 한 가지 종류에 집중 | Unconditional<br>e.g. 다양한 활용 범위<br>- **Single Image Super Resolution**<br>- **Paint-to-Image Style Transfer**<br>- **Harmonization**<br>- **Editing**<br>- **Single Image Animation** |
| 수많은 데이터로 학습 | 한 장의 데이터로 학습 |

# SinGAN의 활용

①그림->사진 ②편집 ③어울리게 합성하기 ④고해상도로 변환



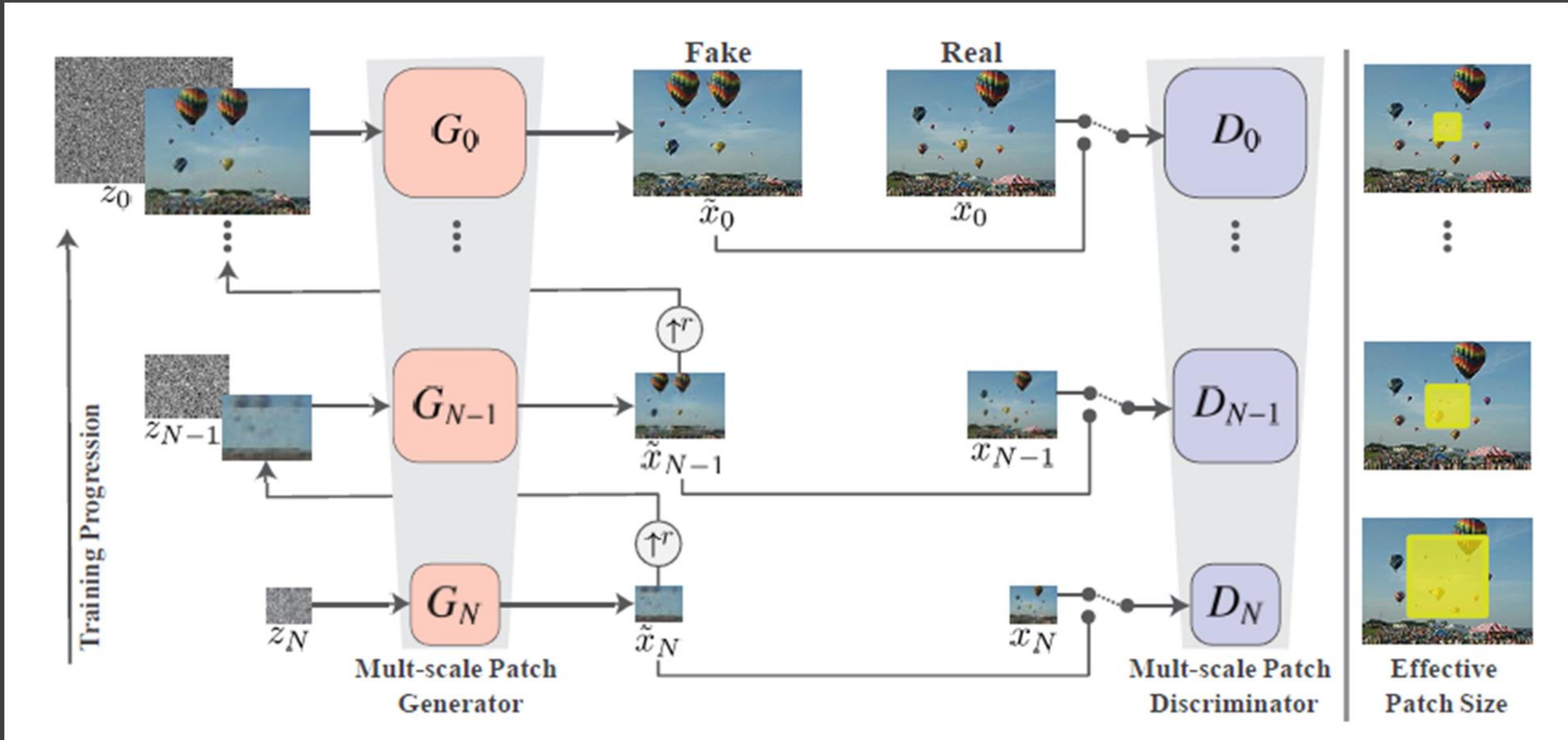
# SinGAN Generator: Fully-Convolutional 구조




Super-Resolution이 가능한 이유

# SinGAN의 활용

# SinGAN의 구조
## Multi-scale pipeline

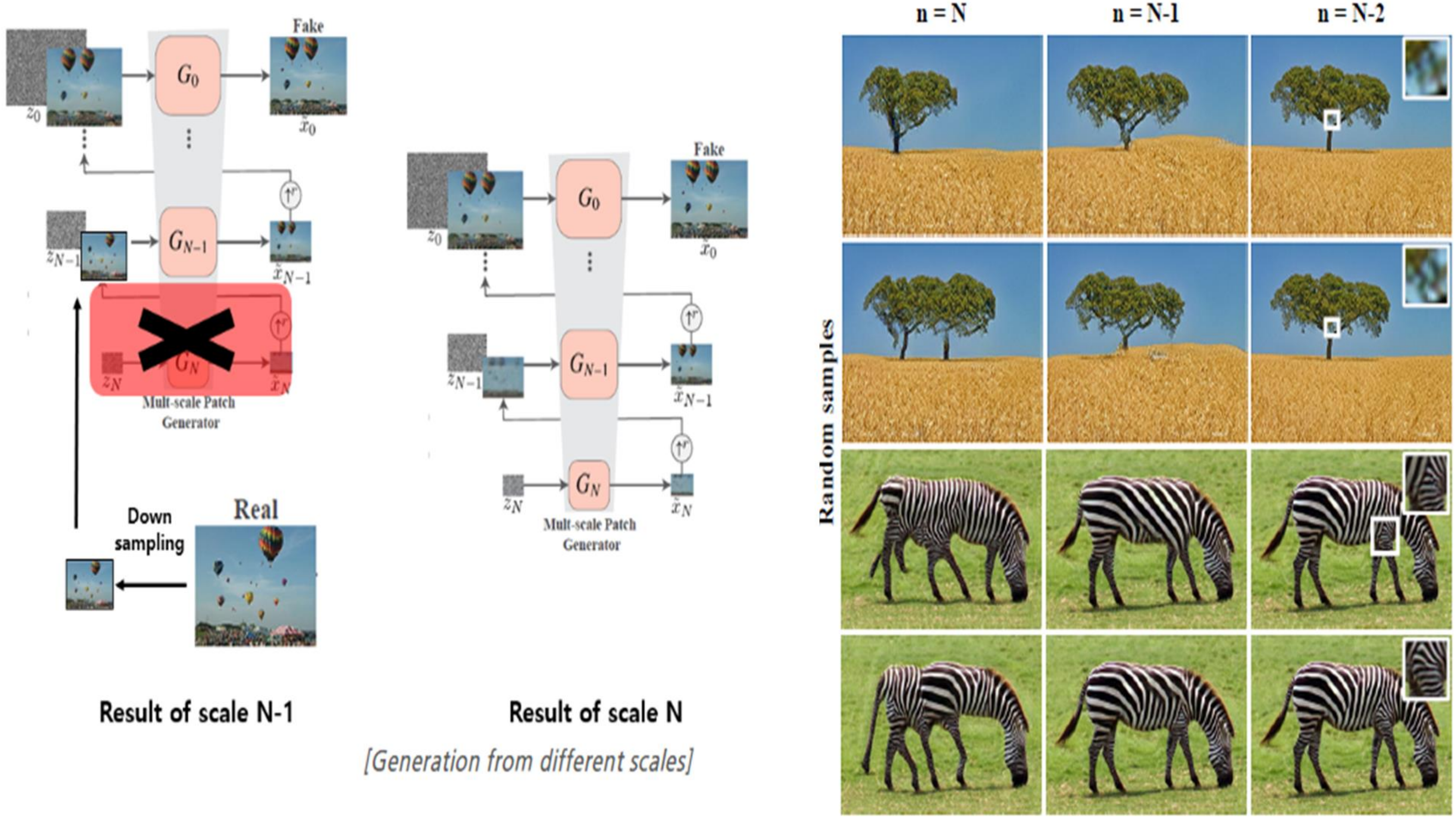



1. SinGAN의 목표: single training image -> <u>**unconditional**</u> generative model
2. 학습 요소: 전체적인 구조 + 세부사항

   (전체적인 특징, 이미지의 배열, 요소들의 모양, 요소들의 디테일, 질감)
3. 학습 방법: 점진적으로 세세하게 뽑아내는 것

# SinGAN의 구조

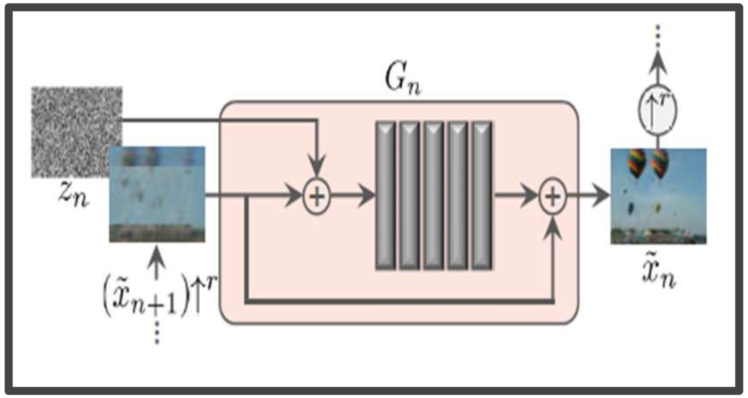
## Multi-scale pipeline



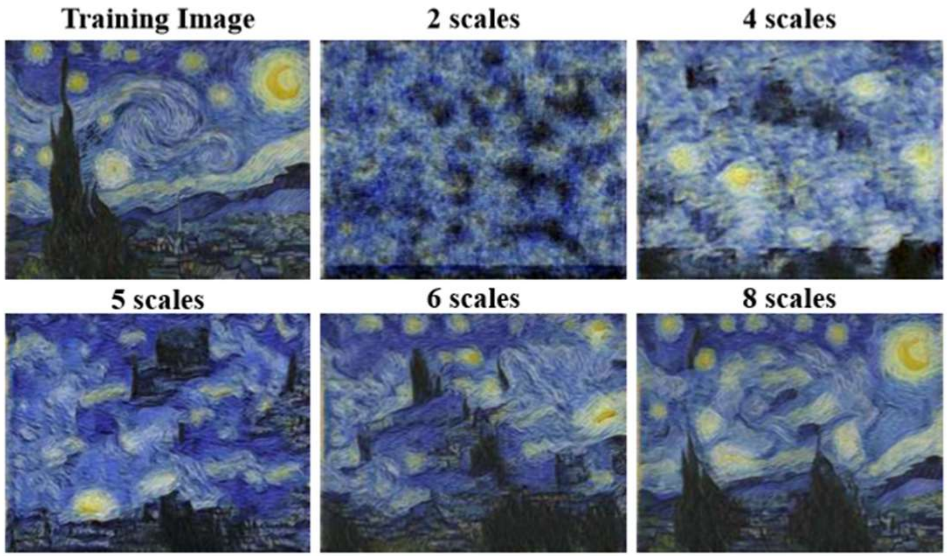

[Generation from different scales]

# SinGAN의 구조

## Single scale generation




Single scale generation

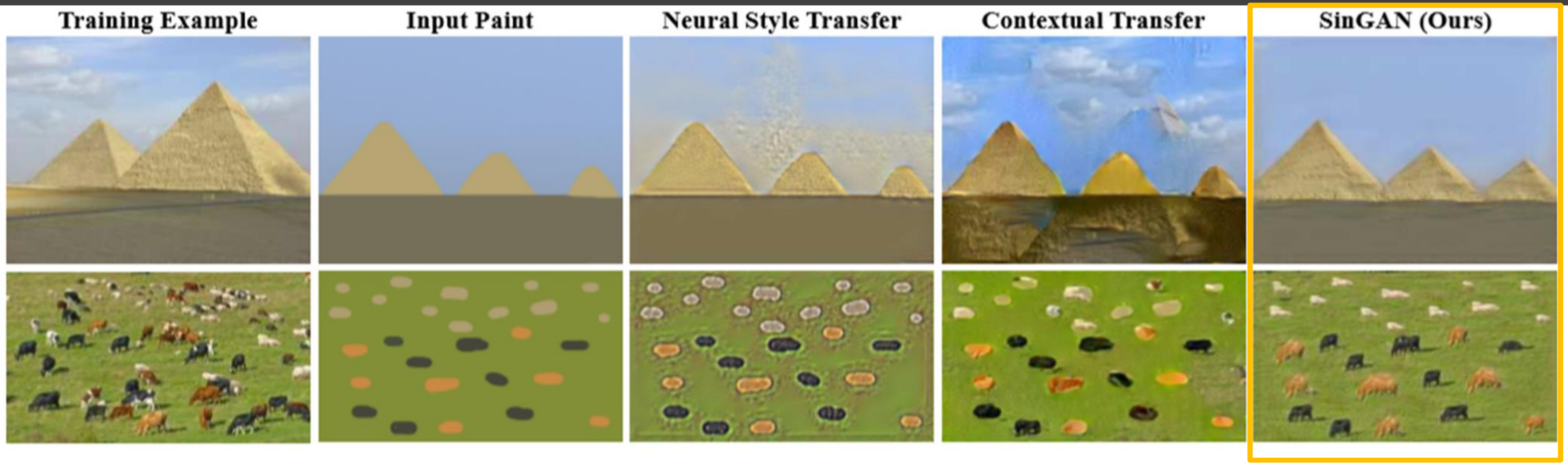



스케일의 숫자에 따른 효과

# SinGAN vs 다른 모델

Goal1: 고해상도로 사진 복구하기


Goal2: 그림을 사진처럼 만들기




| 학습 이미지 | Input 그림 | style transfer method 예시<br>결과값: 불합격! | SinGAN<br>결과값: 합격! |
| --- | --- | --- | --- |

## 08
## End of our Project

# What did the Project leave us?

- 초기 GAN 모델을 이해, 개선하는 아이디어와 기법에 대한 이해는 생각만큼 어렵지 않다.
- 신경망의 발전이 곧 GAN의 발전이다.
- 주로 쓰이는 프레임 워크는 케라스, 파이토치
- 전문가도 아직 이해하지 못한 부분이 많다.
- 선생님이 얼마나 많은 내용을 가르쳐 주셨는지 깨닫게 되었다.

# 레퍼런스

- 캐글 컴페티션 노트북
- https://www.kaggle.com/c/generative-dog-images
- 쉽게 씌어진 GAN
- https://dreamgonfly.github.io/2018/03/17/gan-explained.html
- Sualab Research Blog
- http://research.sualab.com/introduction/practice/2019/05/08/generative-adversarial-network.html
- Jaejun Yoo's Playground(초짜 대학원생의 입장에서 이해하는 Deep Convolutional Generative Adversarial Network (DCGAN))
- http://jaejunyoo.blogspot.com/2017/02/deep-convolutional-gan-dcgan-1.html
- Generative Adversarial Networks - The Story So Far
- https://blog.floydhub.com/gans-story-so-far/
- How to Implement GAN Hacks in Keras to Train Stable Models
- https://machinelearningmastery.com/how-to-code-generative-adversarial-network-hacks/

# Q&A

Danke für Ihre Aufmerksamkeit