

지적 대화를 위한 깊고 넓은 딥러닝

김태훈
carpedm20

저는

carpe
dm20

UNIST 졸업

DEVSISTERS 병특

딥러닝,  TensorFlow

<http://carpedm20.github.io/>

지적 대화를 위한 깊고 넓은 딥러닝

지적 대화를 위한 깊고 넓은 딥러닝

Deep Learning



Google DeepMind

OpenAI



1980년 1997년

CNN, RNN과 같이 흔한 모델이 아니라, 흥미로운 최신 모델들

지적 대화를 위한

깊고 넓은 딥러닝

지적 대화를 위한 깊고 넓은 딥러닝

- 1) Deep Convolutional Generative Adversarial Network
(DCGAN)

지적 대화를 위한 깊고 넓은 딥러닝

2) NTM, 3) DQN, 4) Visual Analogy

DCGAN

DCGAN \notin Supervised Learning

내일의 주가를 예측하고, 평점을 예측하며 사진을 구분할 수 있는 모델들

DCGAN ∈ Unsupervised Learning

Clustering, Anomaly Detection, Generative Models

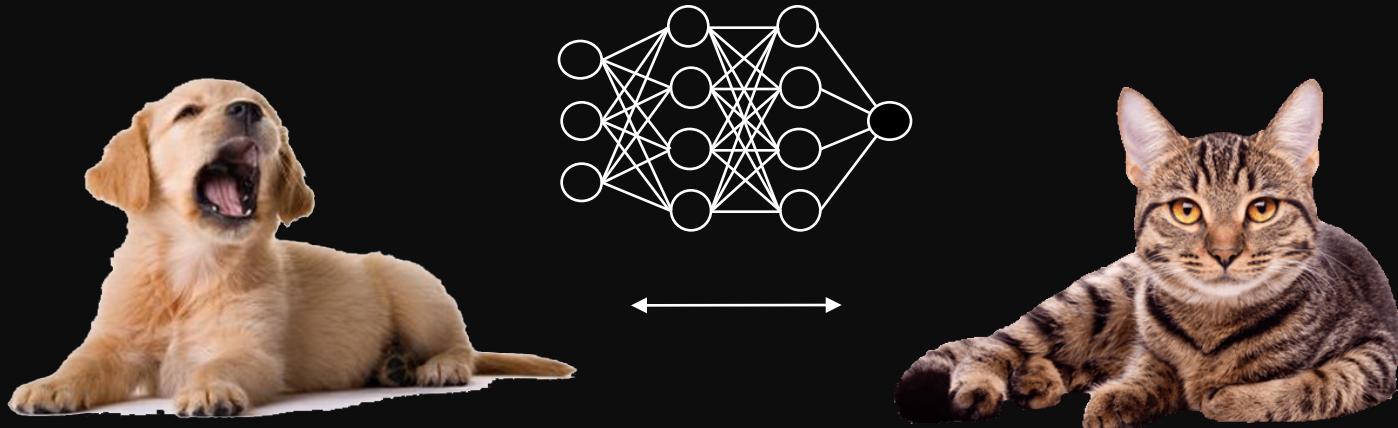
DCGAN ∈ Generative Model

“What I cannot create, I do not understand.”

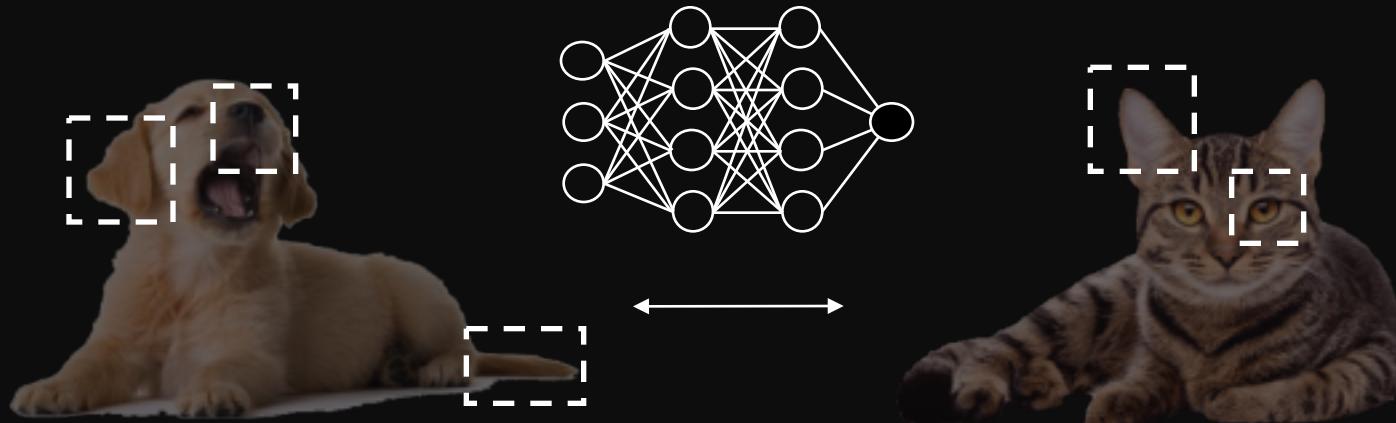
- Richard Feynman



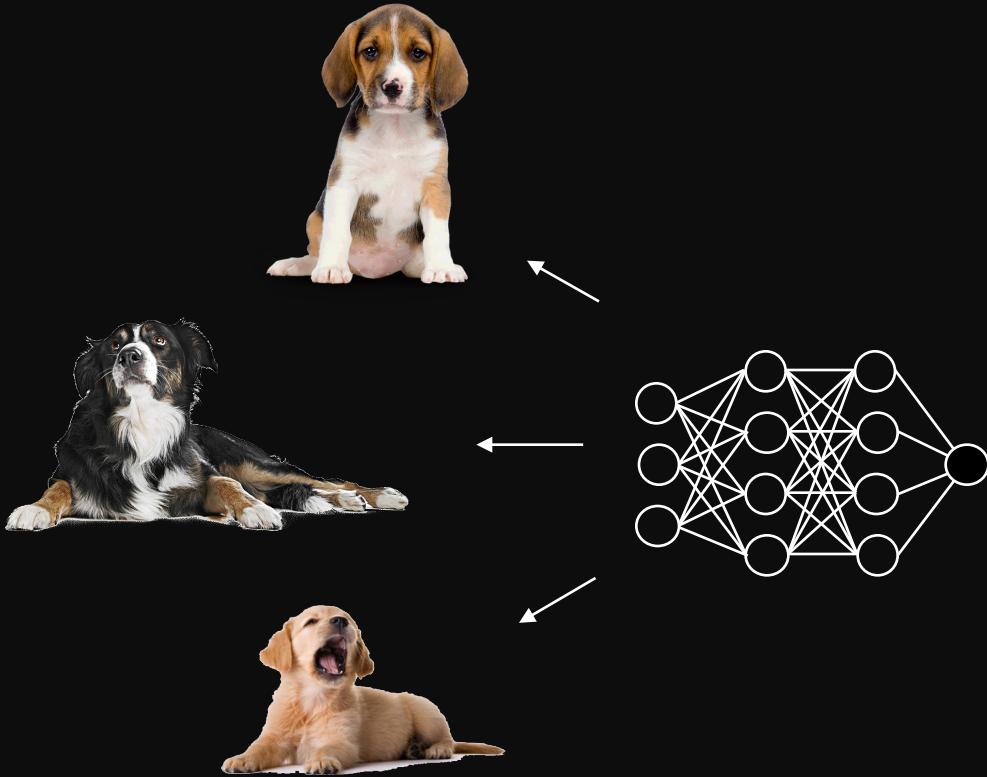
DEVSISTERS



개와 고양이를 구분하는 모델

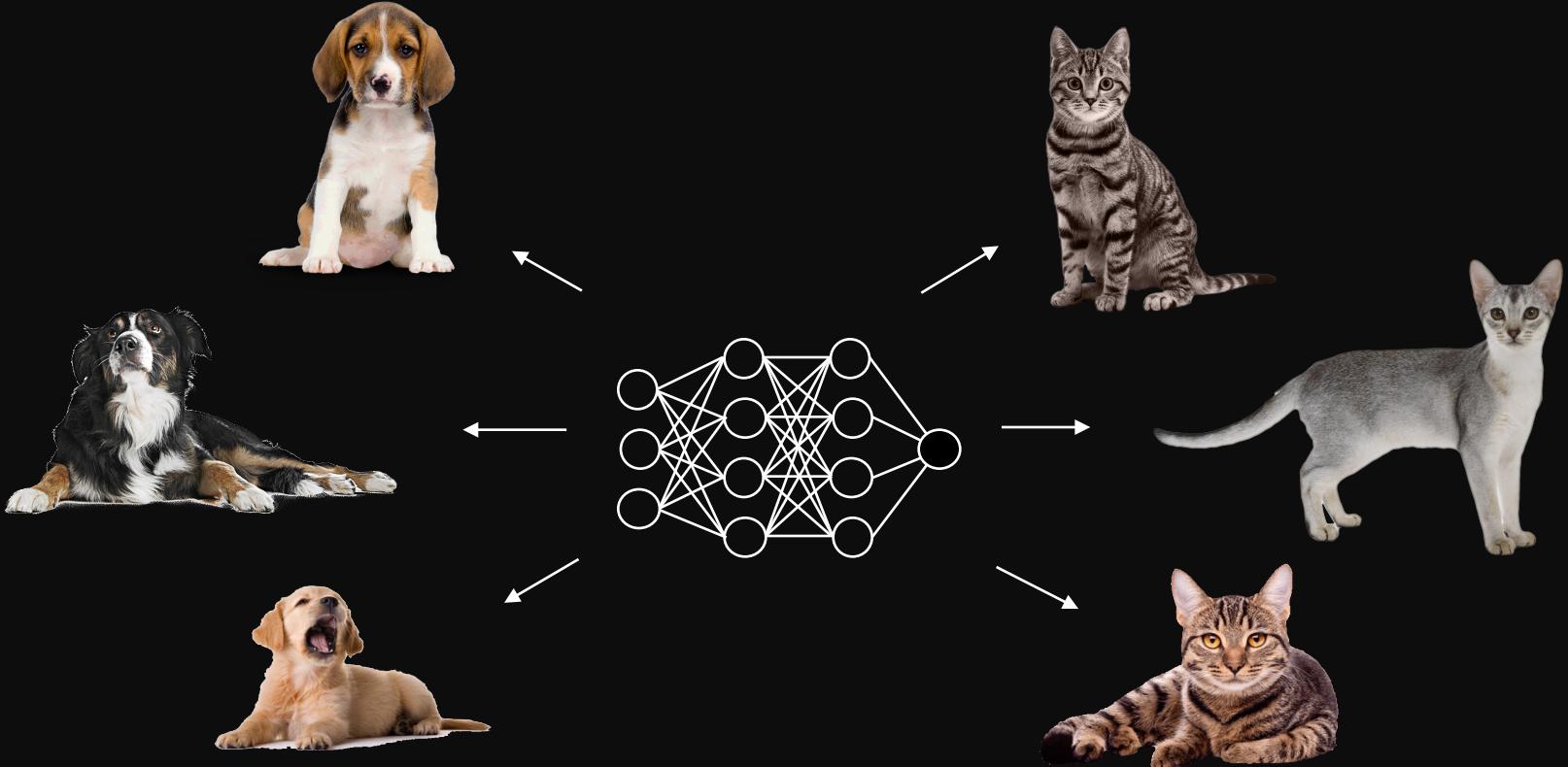


단순히 **특징**만을 배운 것에 불과

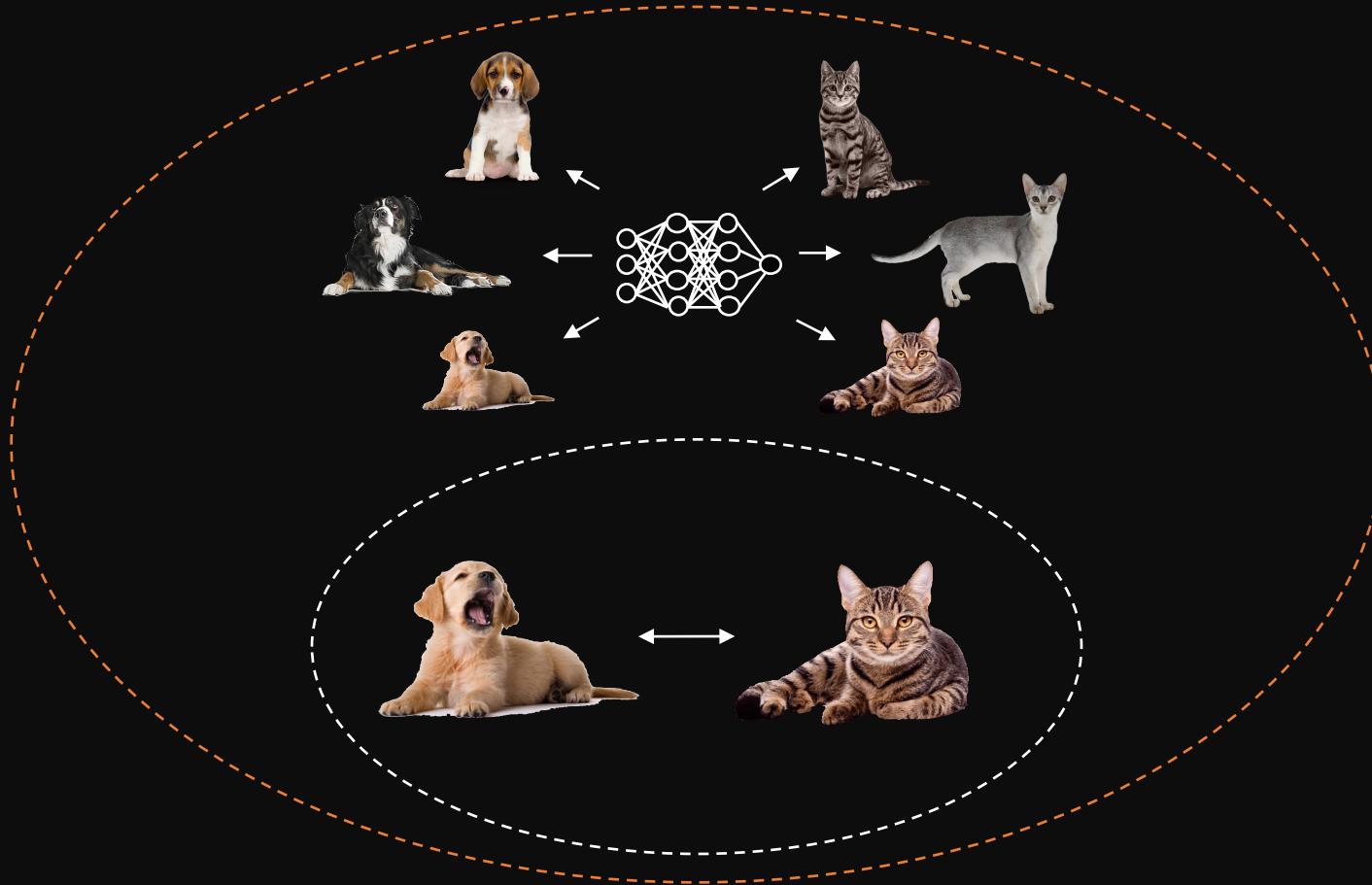


개와 고양이 사진을 만들 수 있는 모델?

Generative Model



분류 모델보다는 개와 고양이를 제대로 이해하고 있다



제대로 이해하고 있다면, 개, 고양이의 분류는 쉬운 문제

이것이 바로

Generative Model이 중요한 이유!



Google DeepMind

OpenAI





?





DCGAN



?

celebA

$f(z)$: 함수를 학습

$$z = \begin{bmatrix} 0.4 \\ -0.1 \\ 0.2 \end{bmatrix} \quad z = \begin{bmatrix} 0.1 \\ 0.4 \\ 0.9 \end{bmatrix} \quad z = \begin{bmatrix} -0.1 \\ 0.2 \\ -0.7 \end{bmatrix}$$

\searrow \swarrow

$f(z) : \text{함수}$

$$z = \begin{bmatrix} 0.4 \\ -0.1 \\ 0.2 \end{bmatrix}$$
$$z = \begin{bmatrix} 0.1 \\ 0.4 \\ 0.9 \end{bmatrix}$$
$$z = \begin{bmatrix} -0.1 \\ 0.2 \\ -0.7 \end{bmatrix}$$

$f(z)$: 함수



$f(z)$: 얼굴 사진을 만드는 궁극의 함수

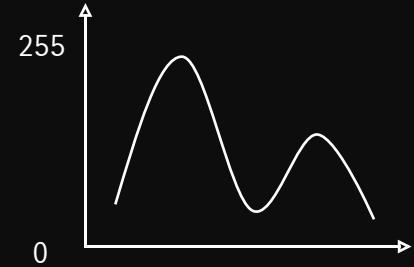


$f(z)$: 얼굴 사진을 만드는 궁극의 함수

72 34 187	69 15 187	80 34 187	23 57 187
44 44 187	83 64 187	64 88 187	44 43 187
70 74 187	24 46 187	36 64 187	54 87 187
78 44 187	74 44 187	35 44 187	67 97 187



RGB 이미지

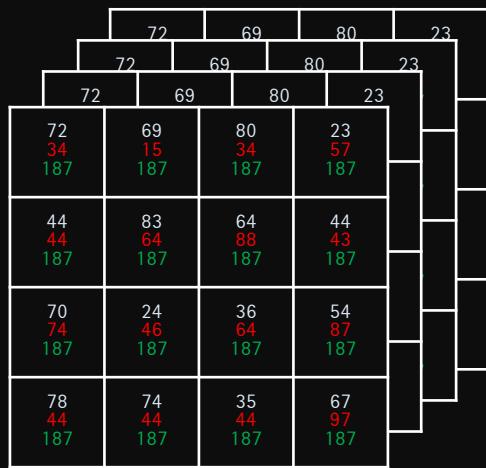


$f(z) = P(y)$: 얼굴 사진을 만드는 궁극의 확률 분포

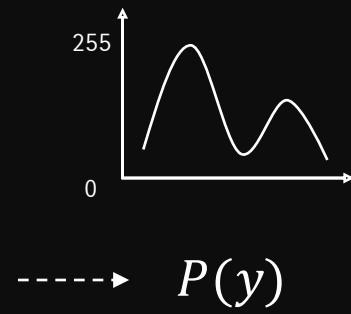
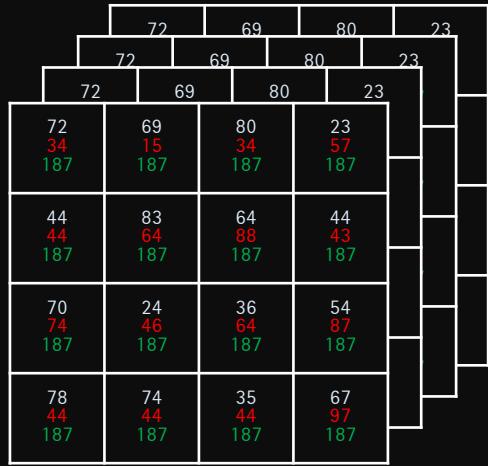
Probabilistic Distribution

72 34 187	69 15 187	80 34 187	23 57 187
44 44 187	83 64 187	64 88 187	44 43 187
70 74 187	24 46 187	36 64 187	54 87 187
78 44 187	74 44 187	35 44 187	67 97 187

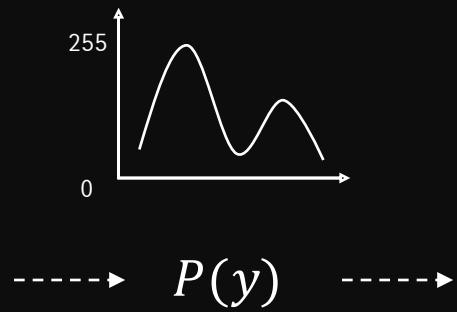
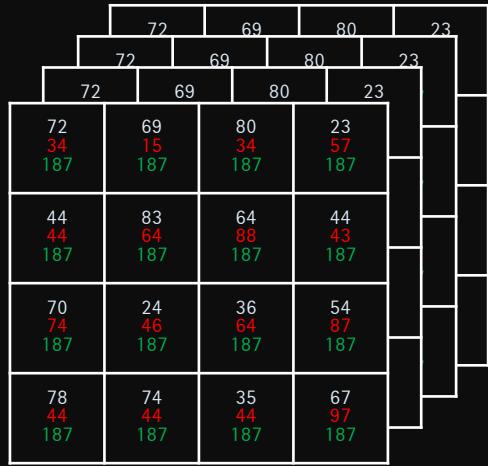
RGB 이미지

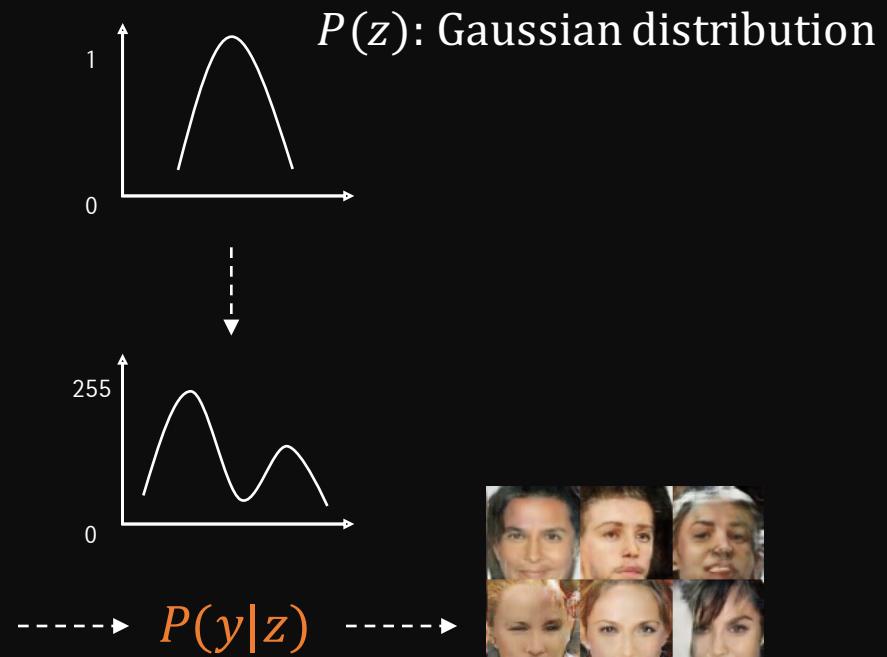
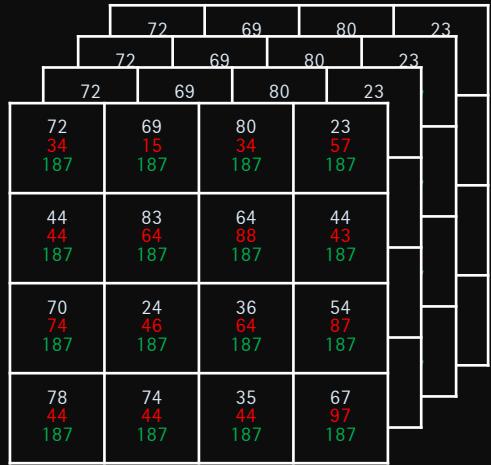


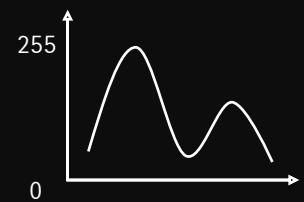
celebA



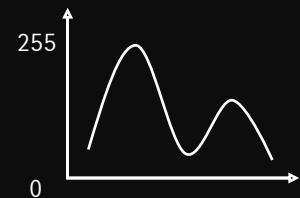
픽셀들의 확률 분포



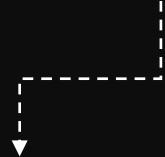




$$P(y|z)$$



$P(y|z)$



DCGAN

Deep Convolutional Generative Adversarial Networks

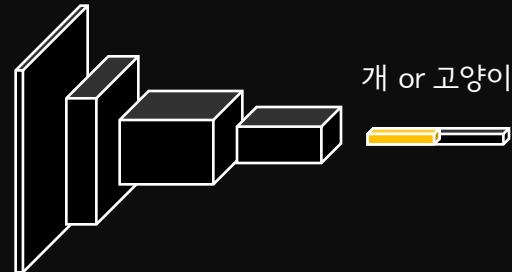
Deep Convolutional Generative Adversarial Networks

딥러닝 Deep Learning

Deep Convolutional Generative Adversarial Networks

Deep ConvNet Convolutional Generative Adversarial Networks

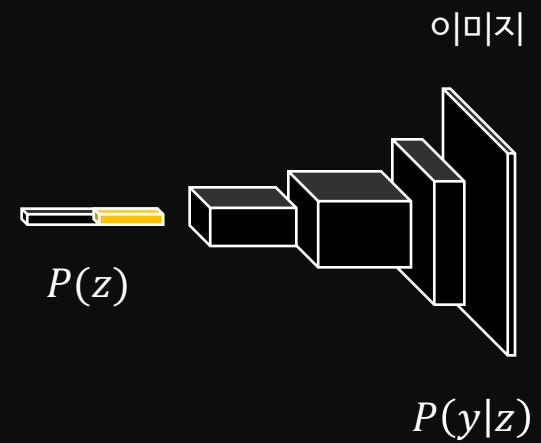
이미지



보통 CNN classification은 이게 전부

생성 모델
Generative Model

Deep Convolutional Generative Adversarial Networks



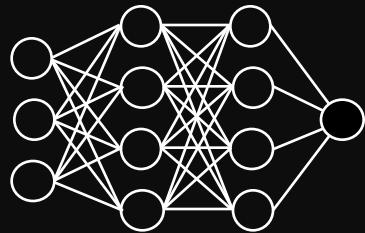
이미지

적대적 학습
Adversarial Learning

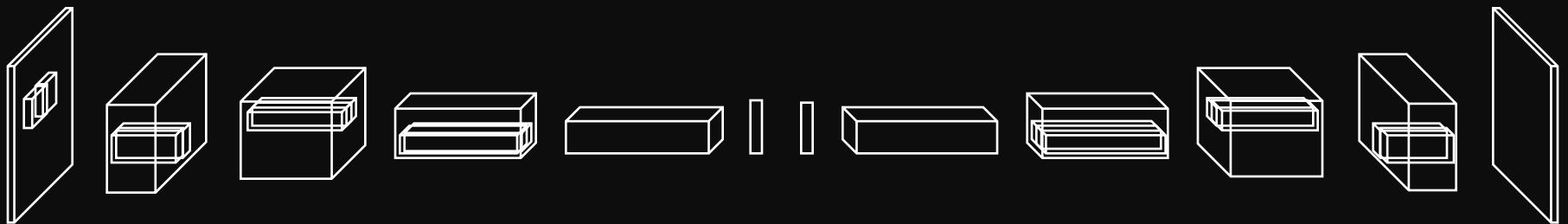
Deep Convolutional Generative Adversarial Networks

인공 신경망
Neural Network

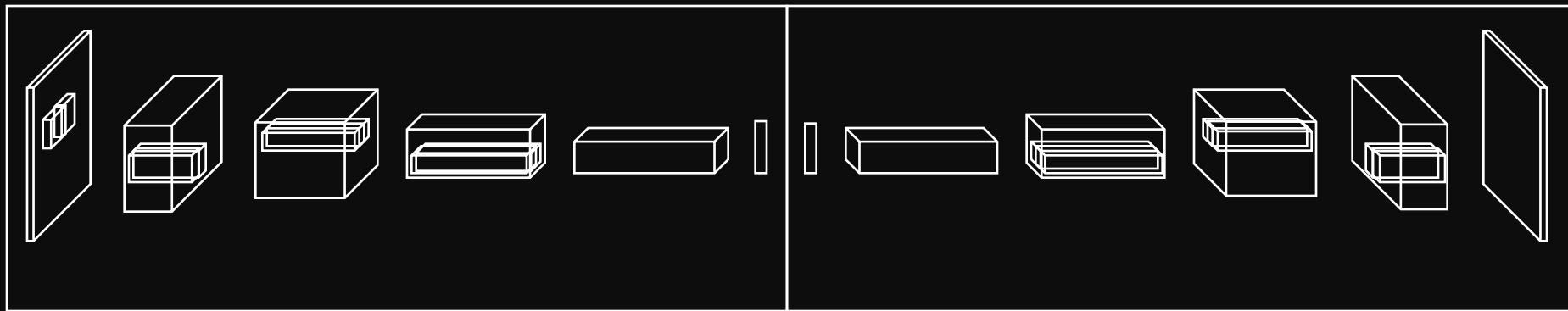
Deep Convolutional Generative Adversarial Networks



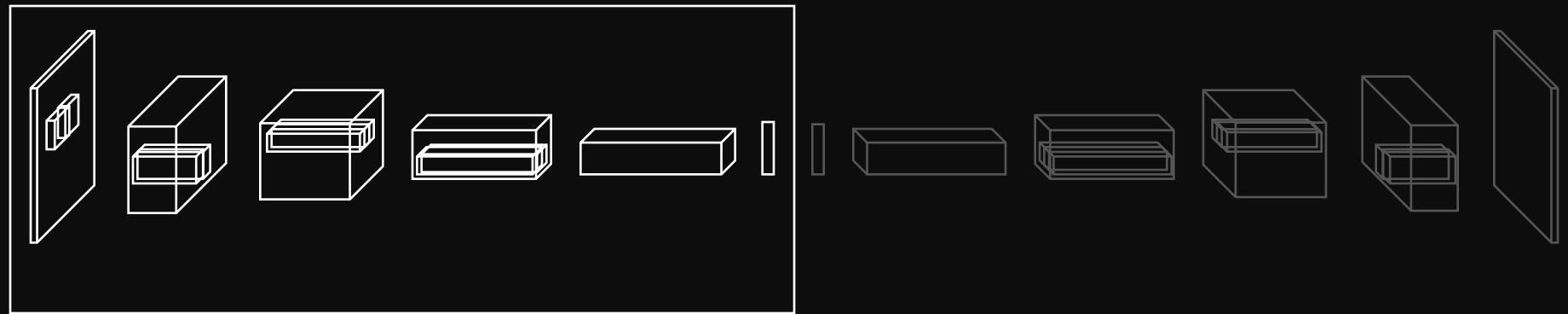
DCGAN



DCGAN



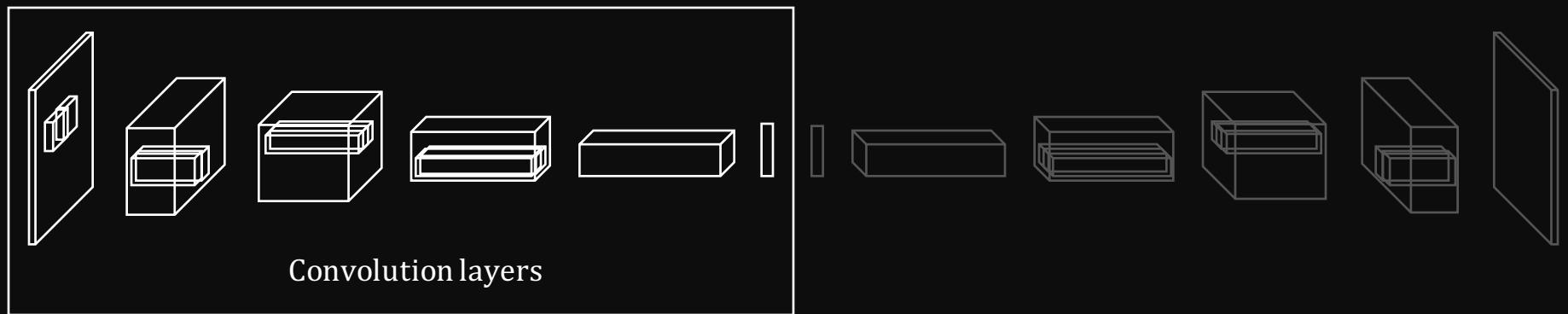
DCGAN



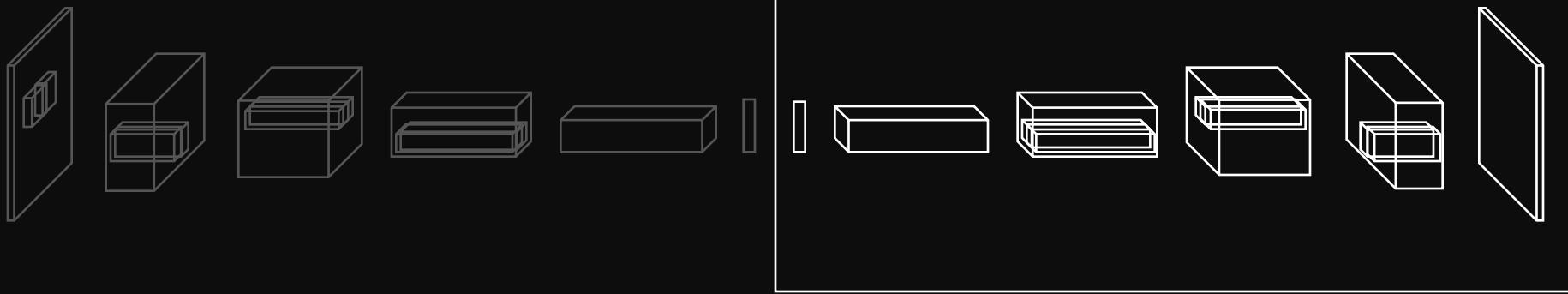
DCGAN

구분자

Discriminator

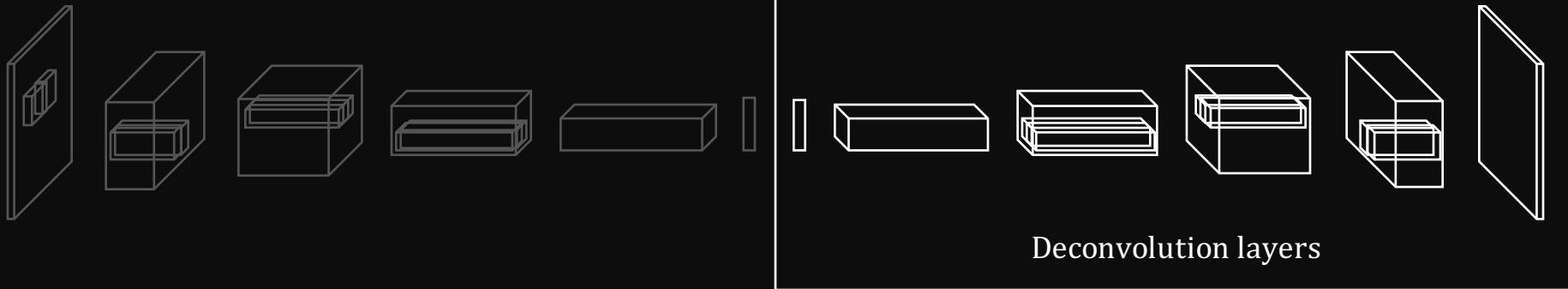


DCGAN



DCGAN

생성자
Generator



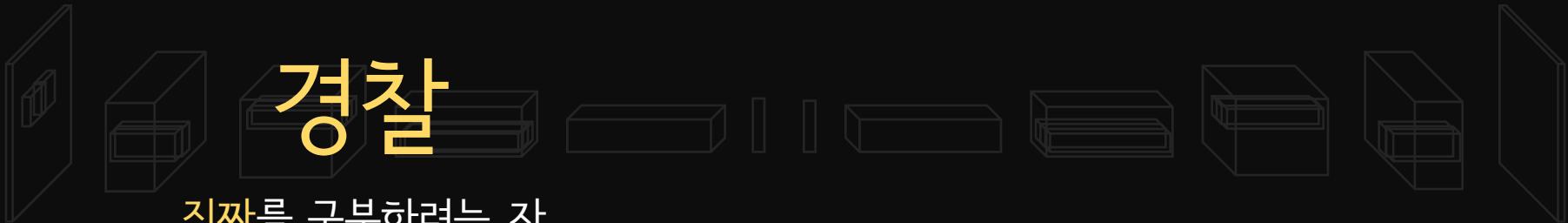
DCGAN

구분자

Discriminator

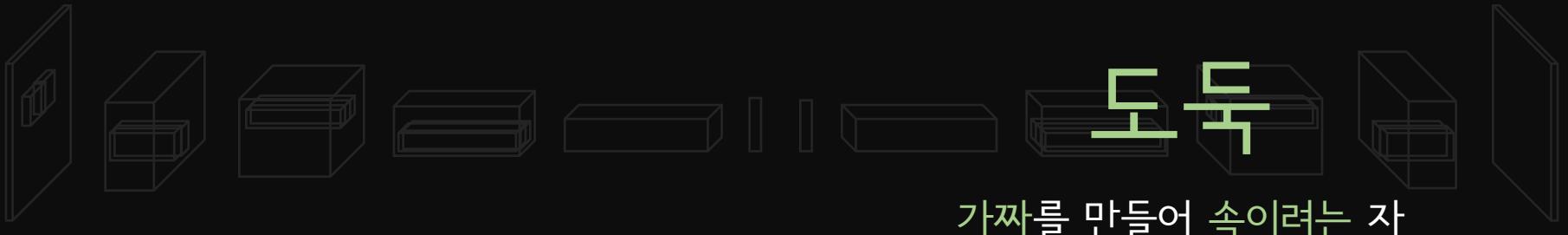
경찰

진짜를 구분하려는 자

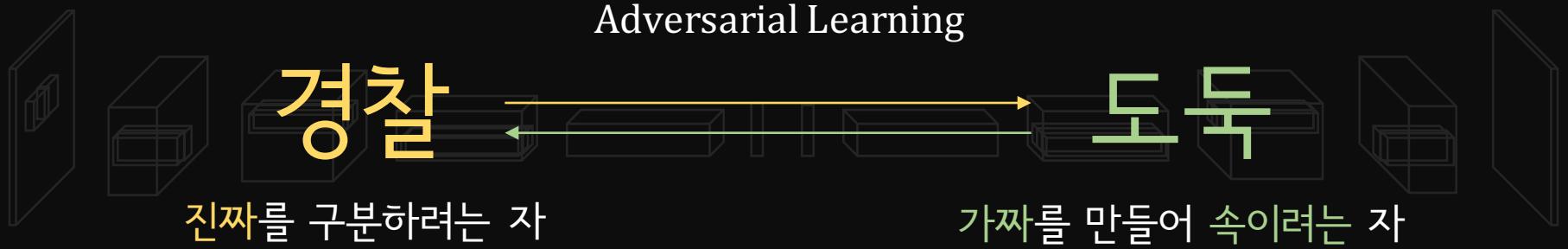


DCGAN

생성자
Generator



DCGAN



적대적 학습

Adversarial Learning

적대적 학습

Adversarial learning

경찰

도둑

적대적 학습

Adversarial learning



한번에 한번씩 번갈아가며 학습

1. 경찰의 학습



→ O? X?



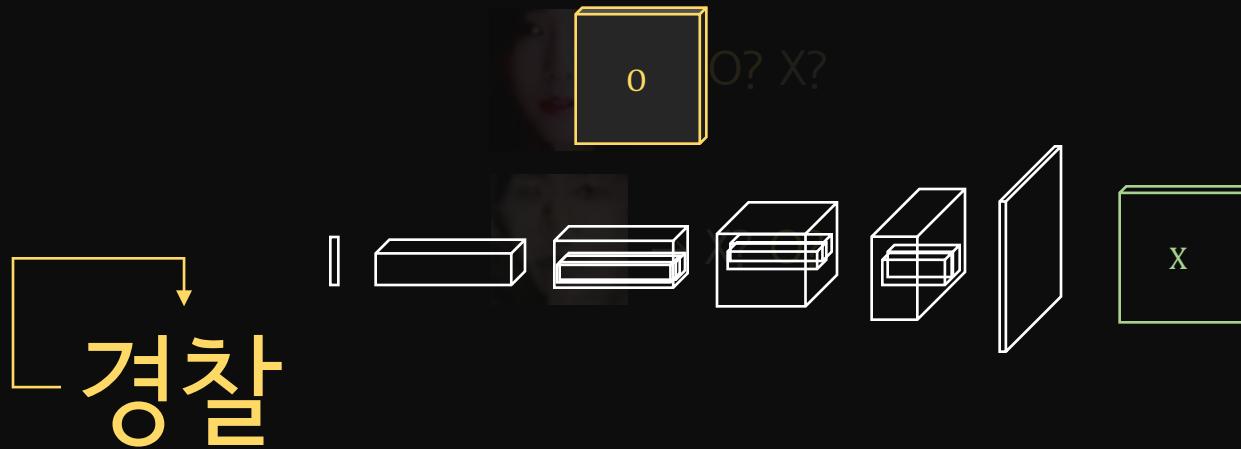
→ X? O?



경찰

진짜와 가짜를 구분하는 방법을 학습

1. 경찰의 학습



진짜와 가짜를 구분하는 방법을 학습

2. 도둑의 학습

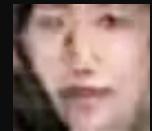


경찰을 속이는 사진을 만드는 방법을 학습

경찰



→ O? X?



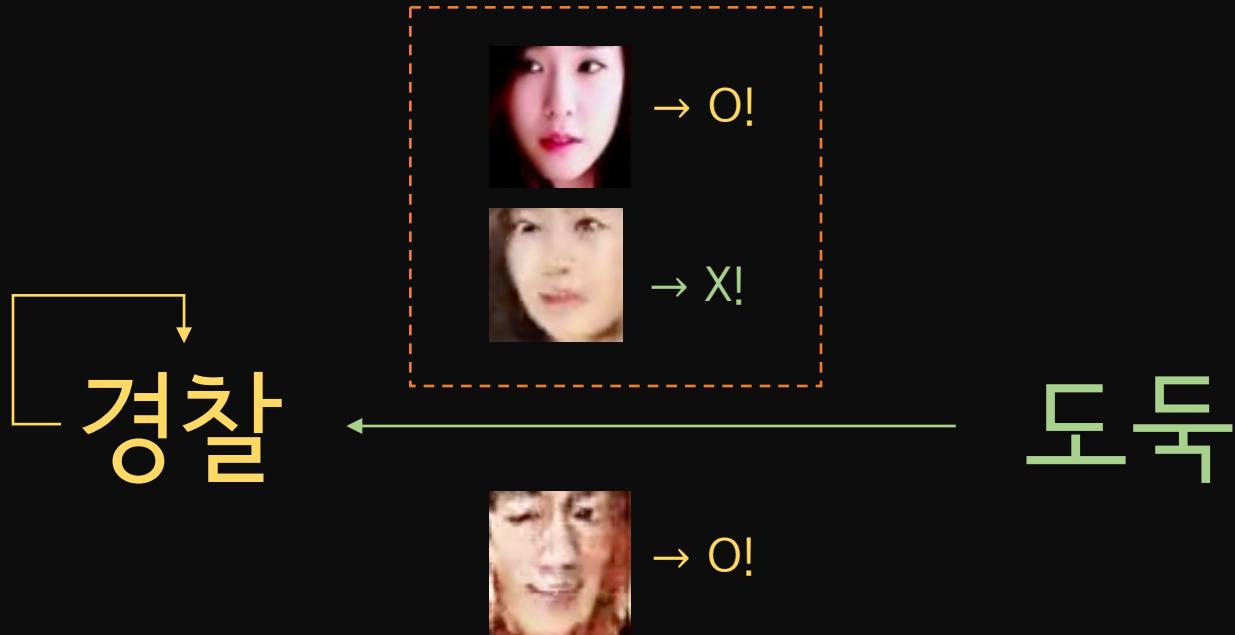
→ X? O?

도둑



→ O!

경찰은 점점 진짜와 가짜를 구분하게 되고





경찰



도둑



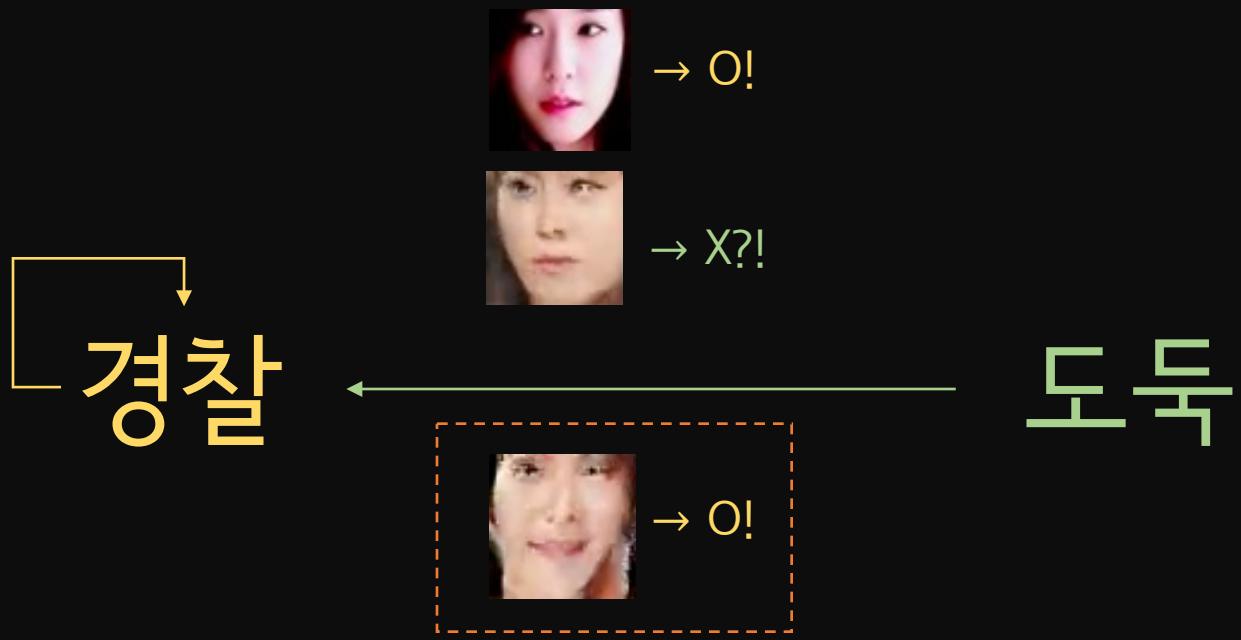
→ O!



→ X!



→ O!

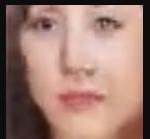


도둑은 점점 진짜같은 가짜를 잘 만들게 된다

경찰



→ O!



→ X?!

도둑

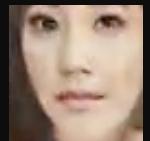


→ O!

경찰



→ O!



→ O?!

도둑



→ O!

경찰



도둑



→ O!



→ O?



→ O!

경찰



→ O!



→ O!

도둑



→ O!

경찰



→ O!



→ O!

도둑



→ O!

경찰



도둑



→ O!



→ O!

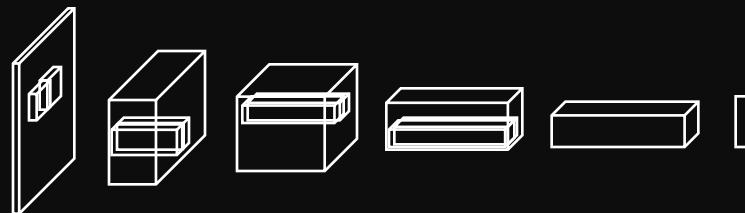


→ O!

네트워크와 함께 설명해 보겠습니다

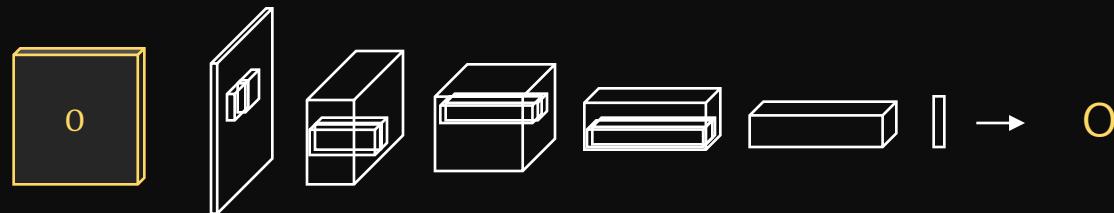
구분자 = 경찰

Descriptor



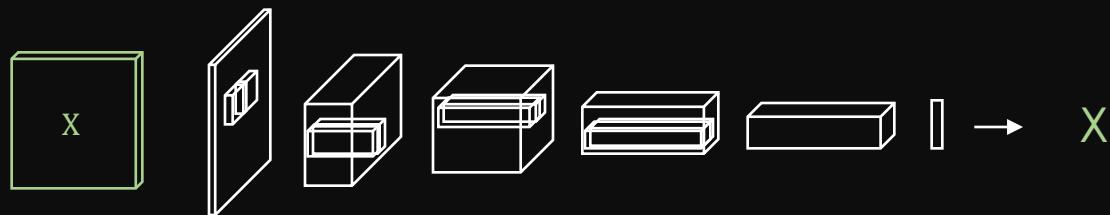
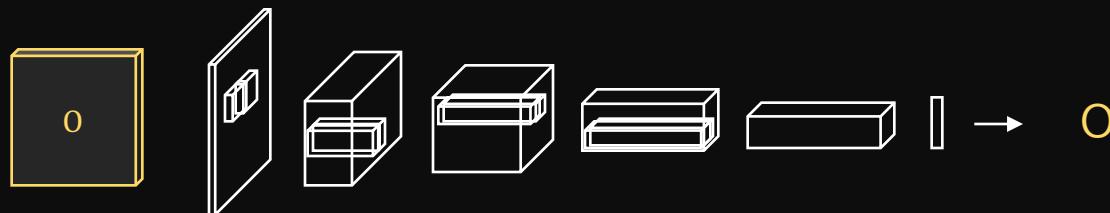
구분자 = 경찰

Descriptor



구분자 = 경찰

Descriptor



구분자 = 경찰

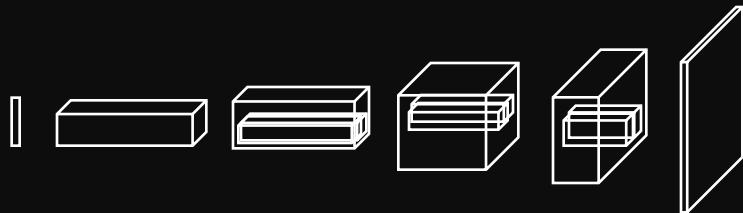
Descriptor



1, 0로 이루어진 벡터를 예측

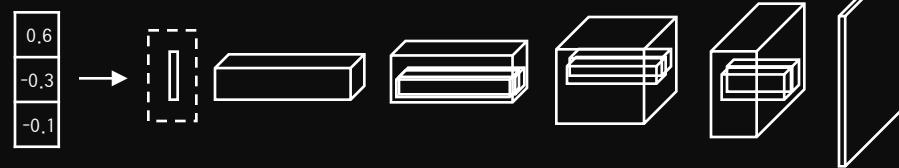
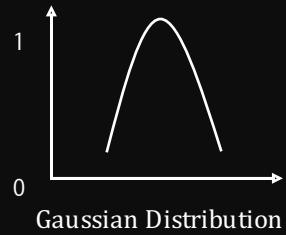
생성자 = 도둑

Generator



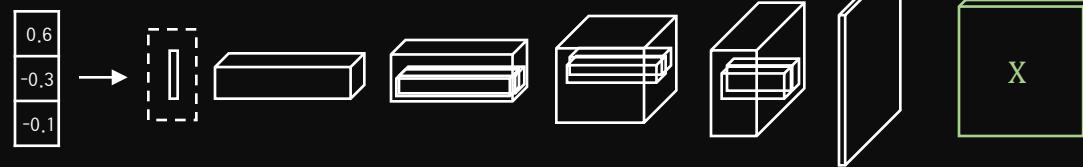
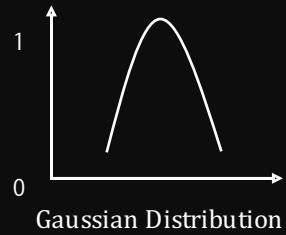
생성자 = 도둑

Generator



생성자 = 도둑

Generator

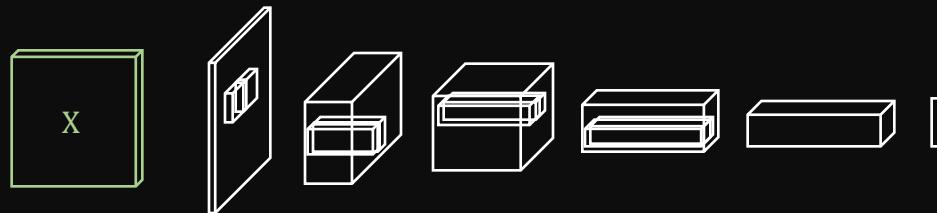
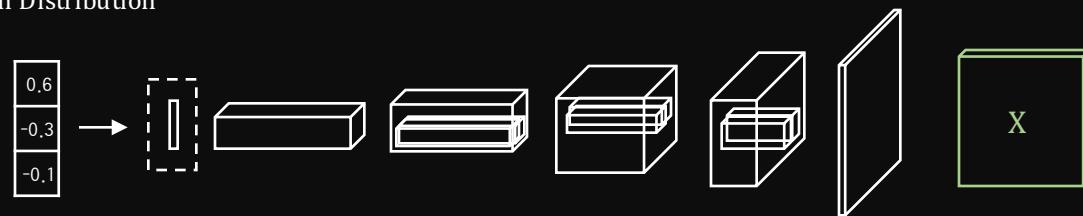


생성자 = 도둑

Generator

평균이 0인 랜덤한 숫자들

Gaussian Distribution

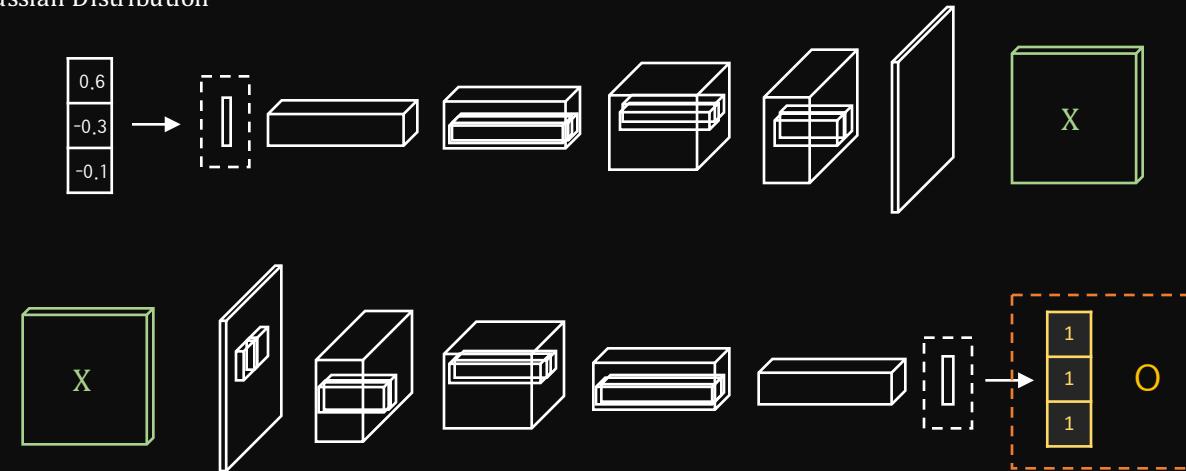


생성자 = 도둑

Generator

평균이 0인 랜덤한 숫자들

Gaussian Distribution



X를 O라고 하도록 학습

네트워크 구조

Convolution, Deconvolution, Batch Normalization

구분자

Discriminator

이미지



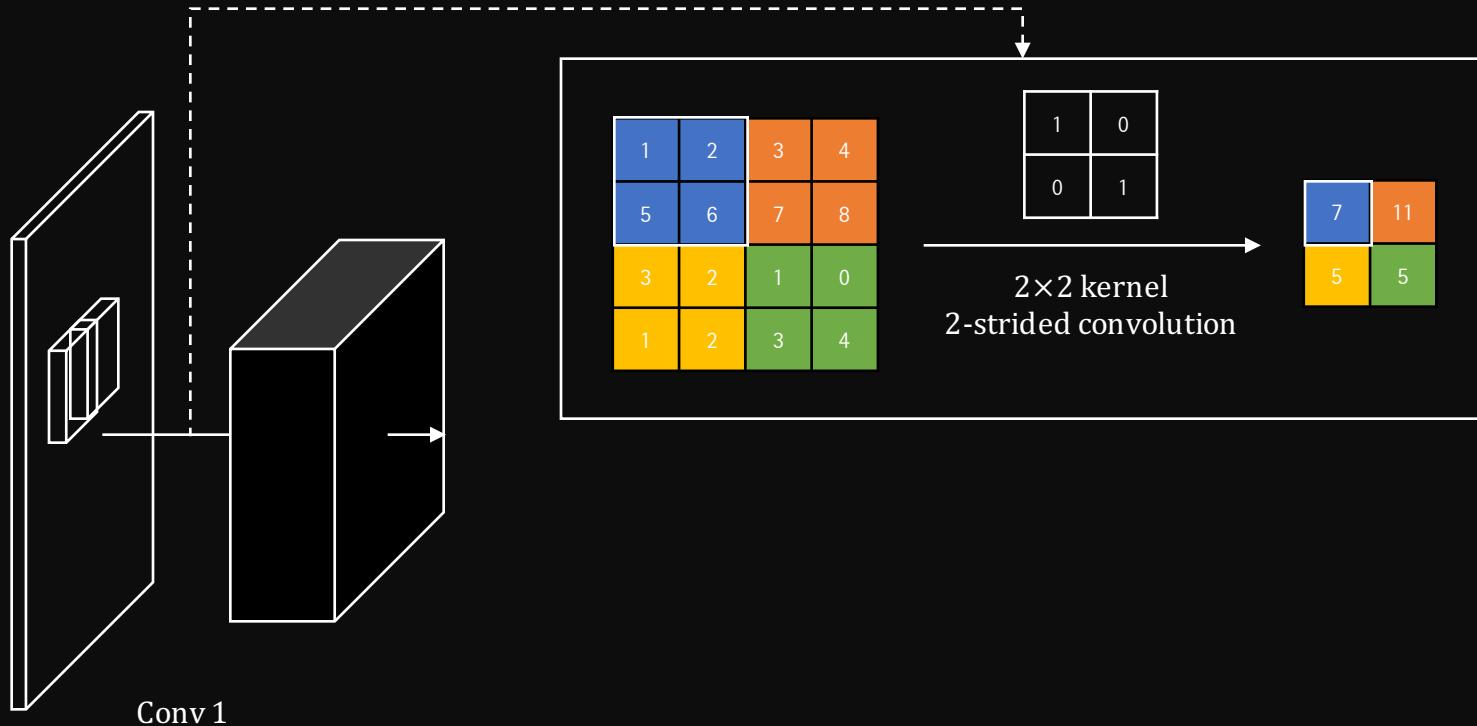
구분자

Discriminator



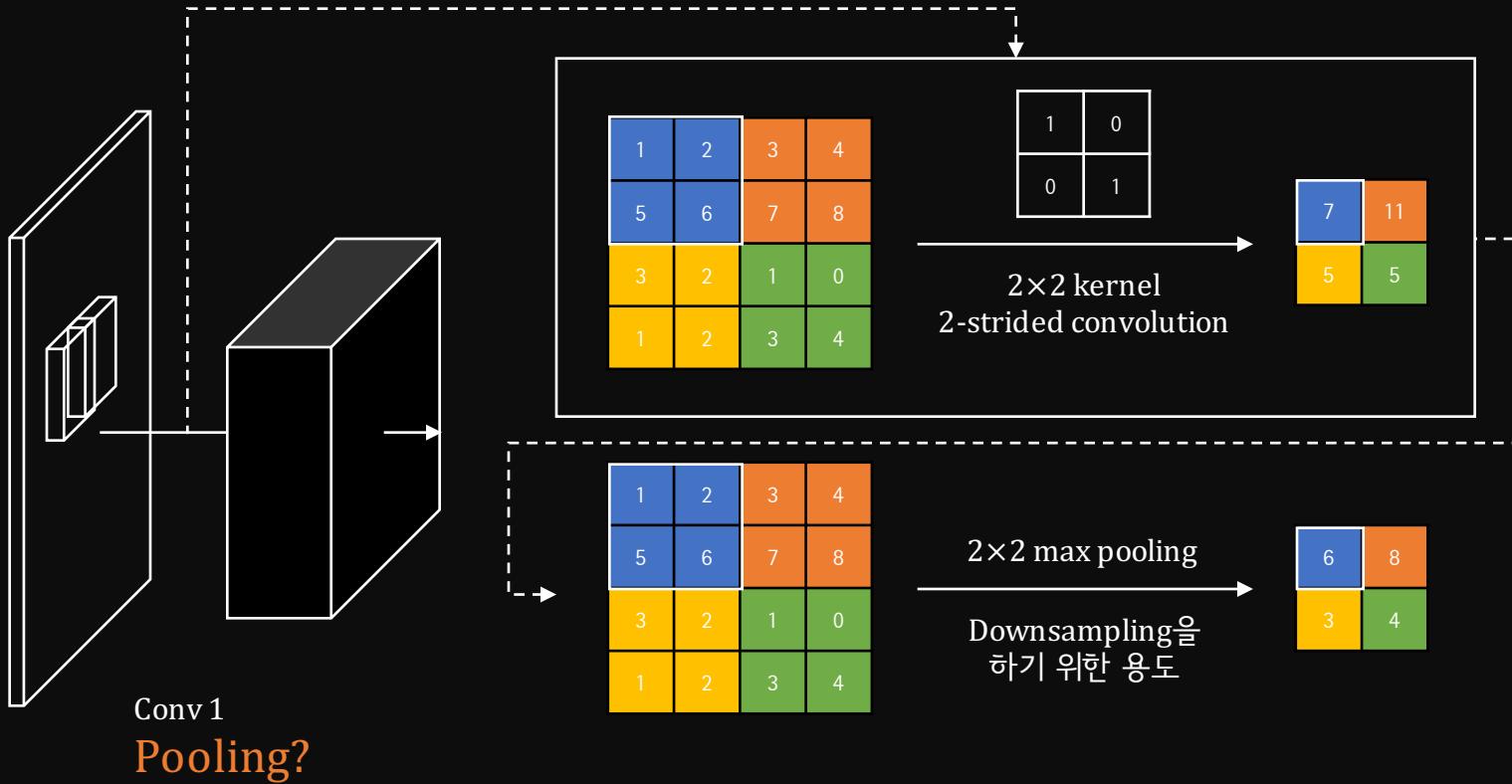
구분자

Discriminator



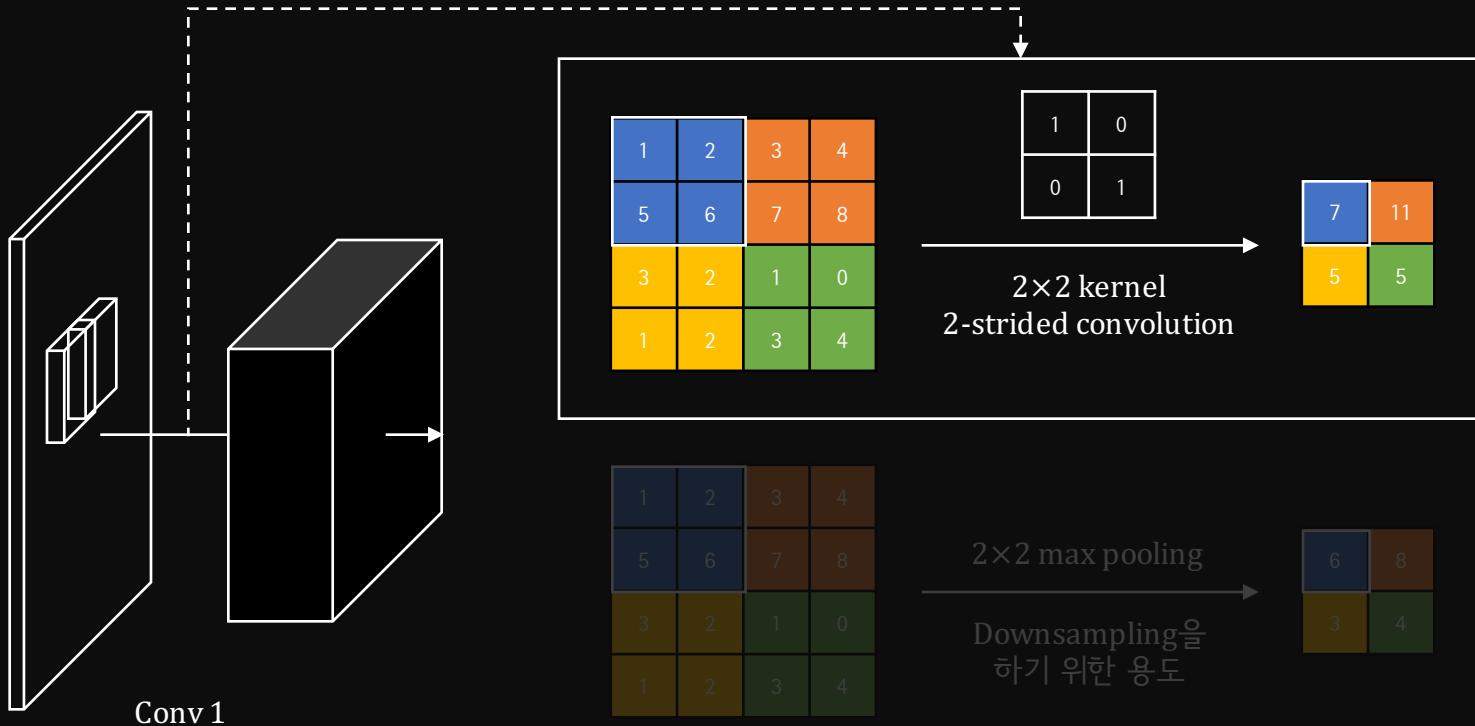
구분자

Discriminator



구분자

Discriminator



구분자

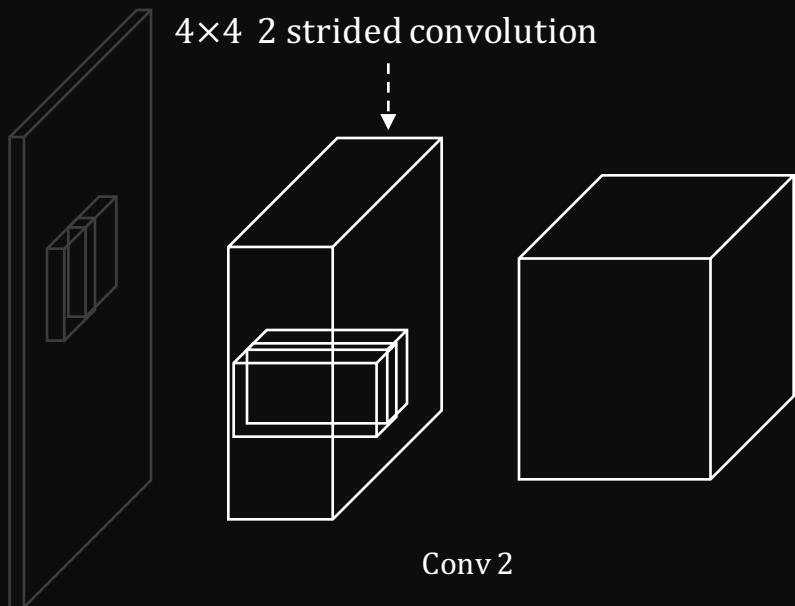
Discriminator

4×4 2 strided convolution



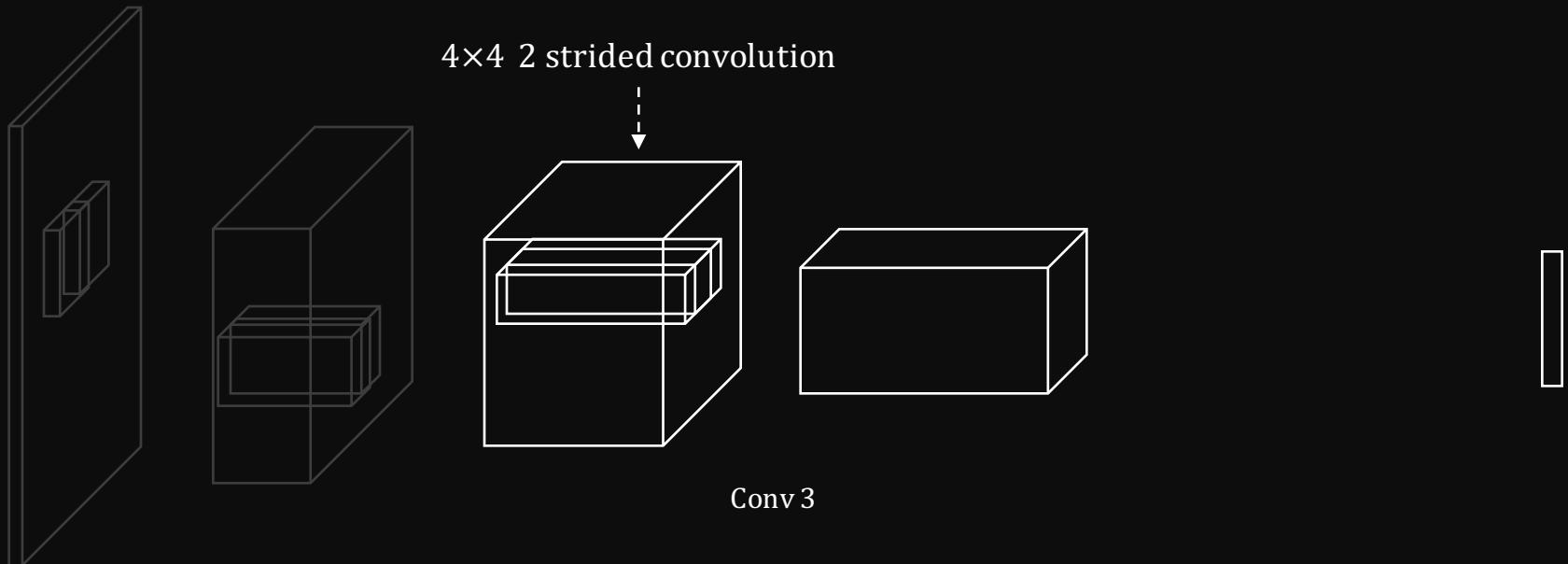
구분자

Discriminator



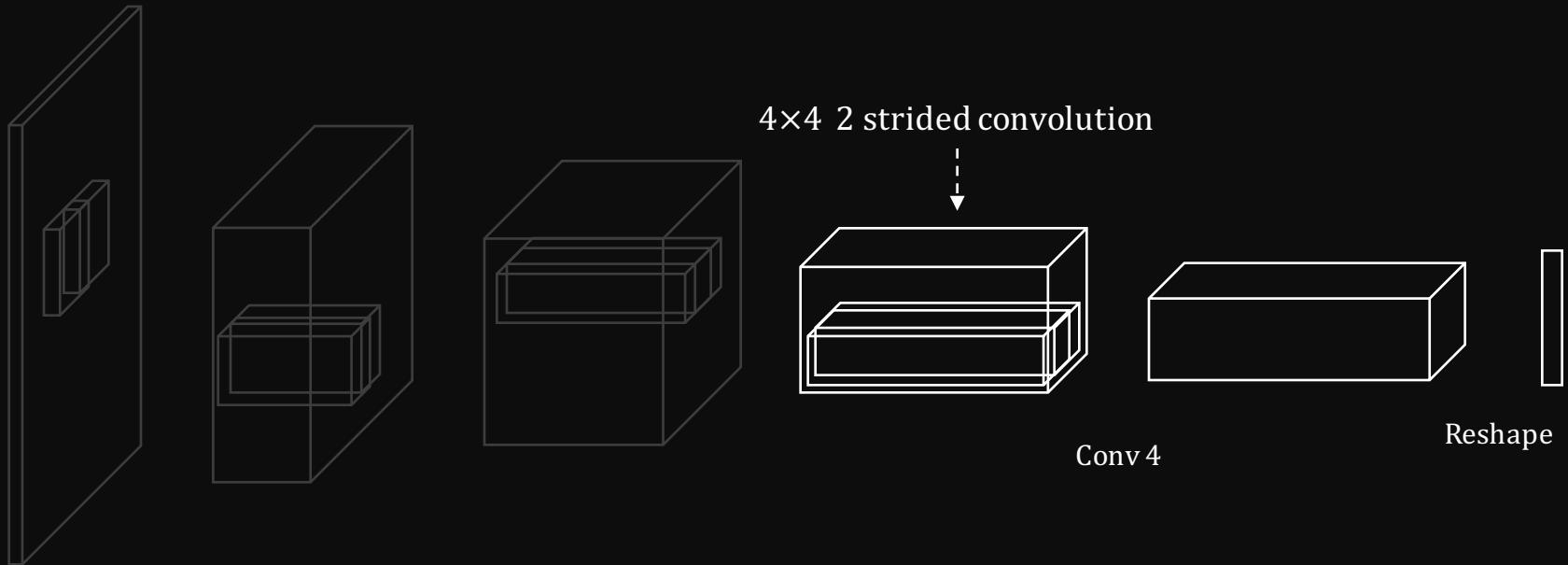
구분자

Discriminator



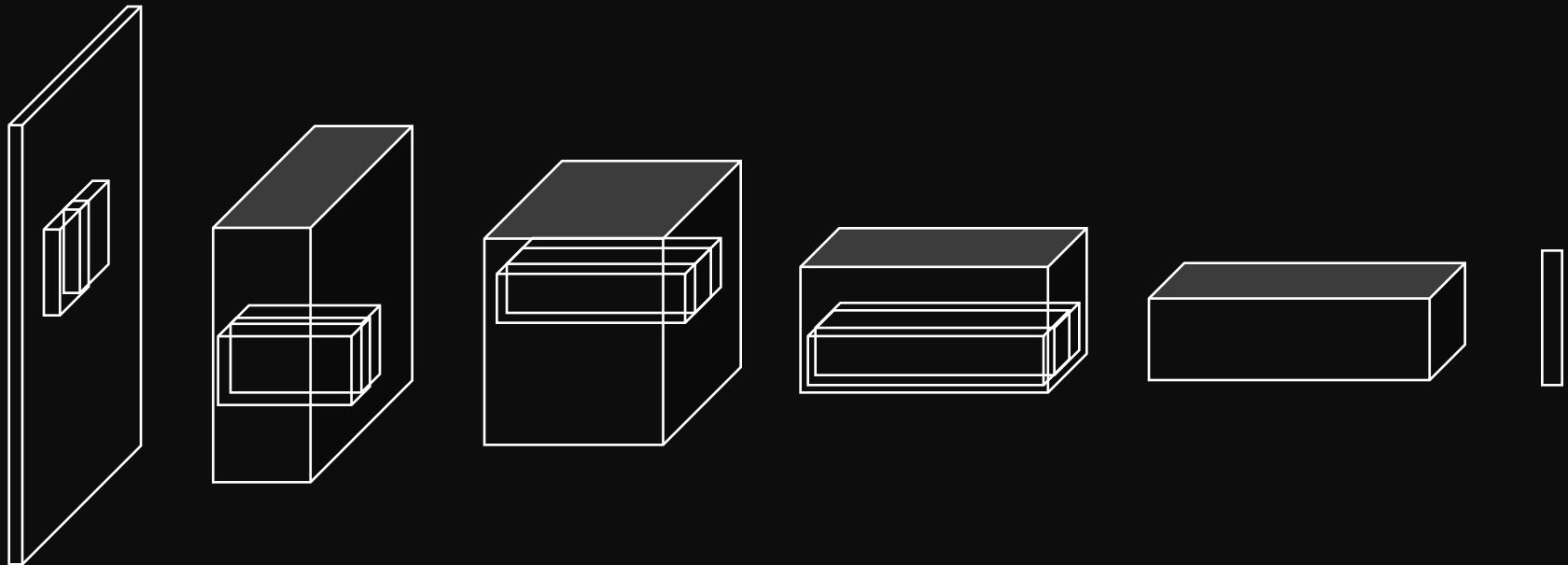
구분자

Discriminator



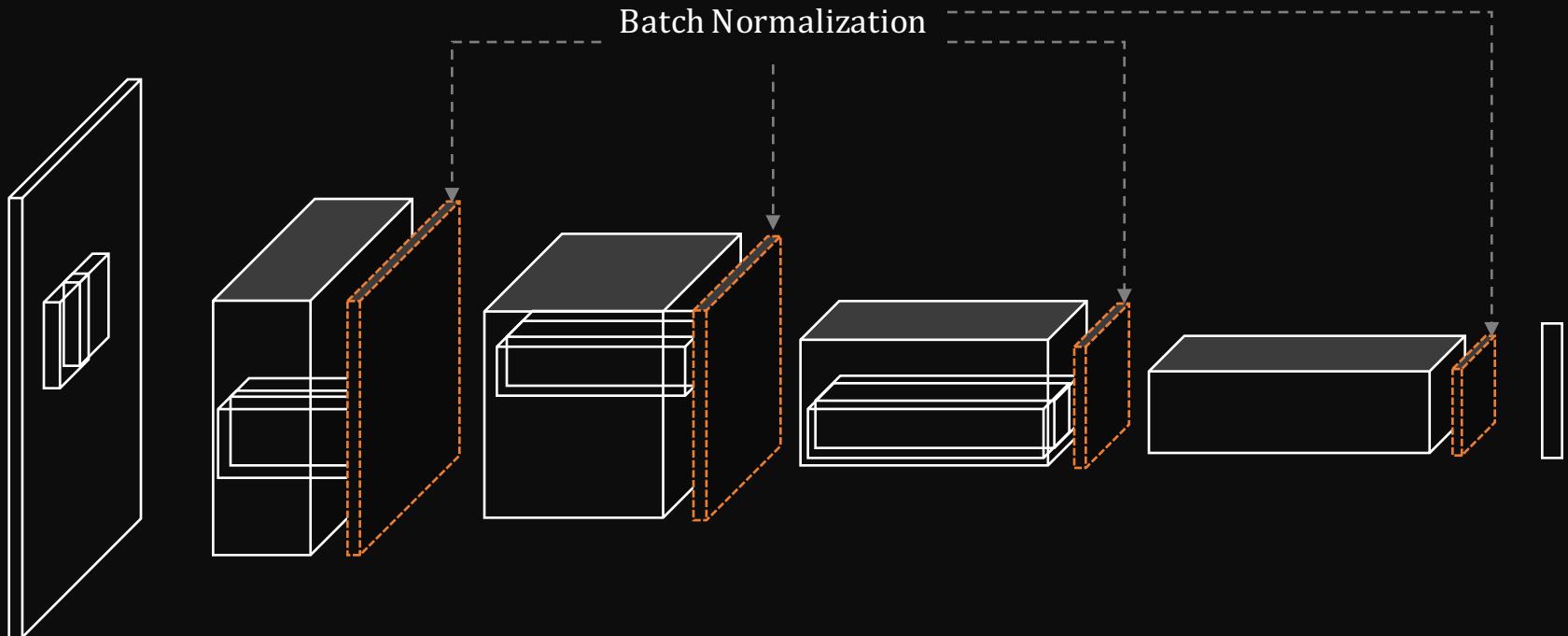
구분자

Discriminator



구분자

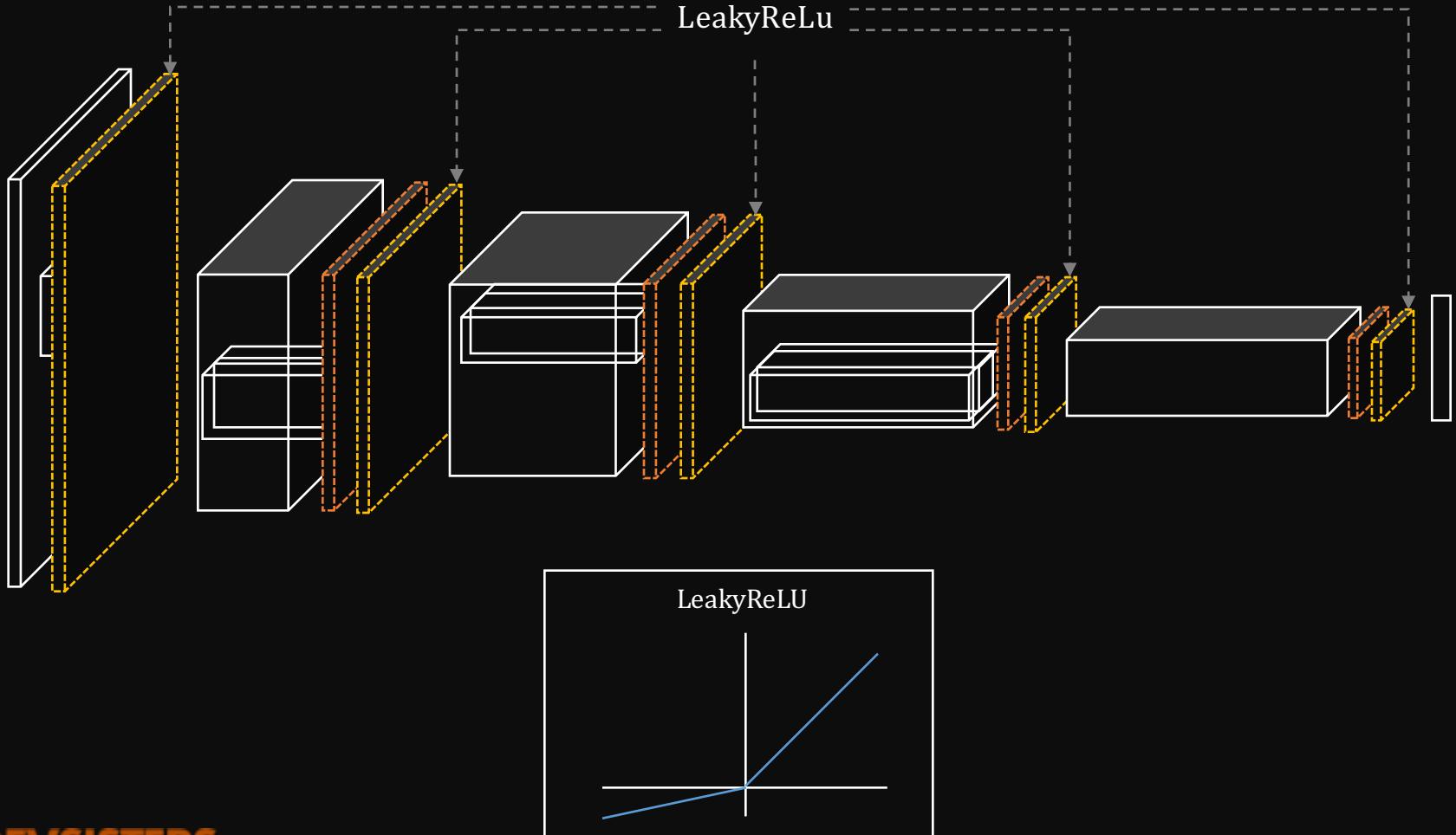
Discriminator



이전 layer의 output을 normalize시켜 학습을 가속시키는 방법

구분자

Discriminator

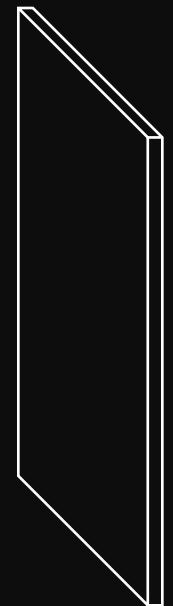


생성자

Generator

이미지

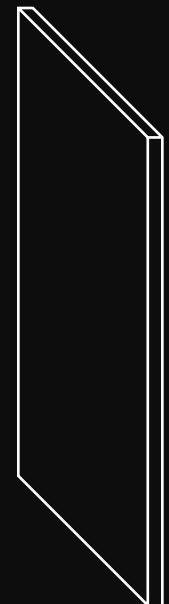
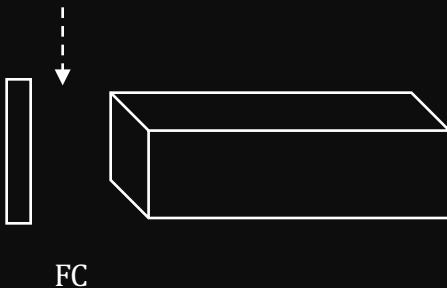
Z



생성자

Generator

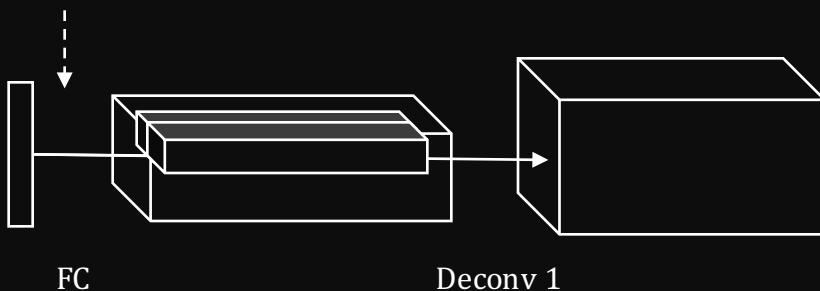
Matrix multiplication
(Fully connected)



생성자

Generator

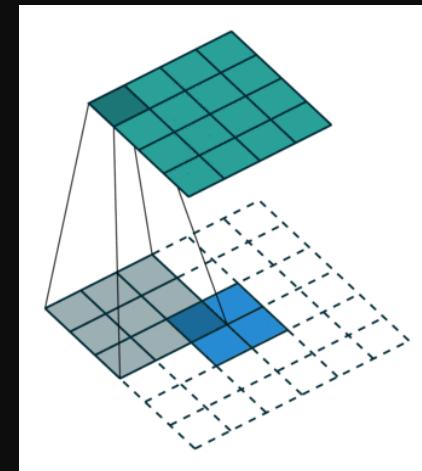
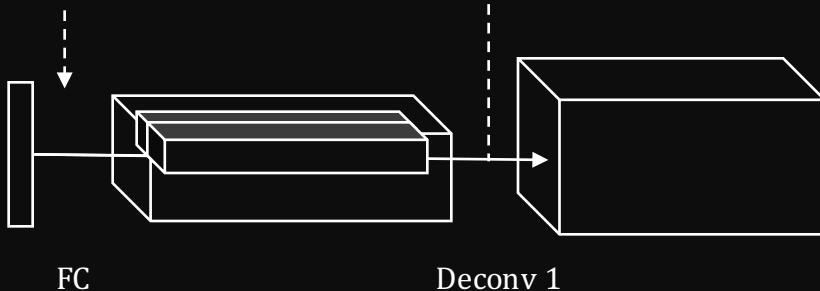
Matrix multiplication
(Fully connected)



생성자

Generator

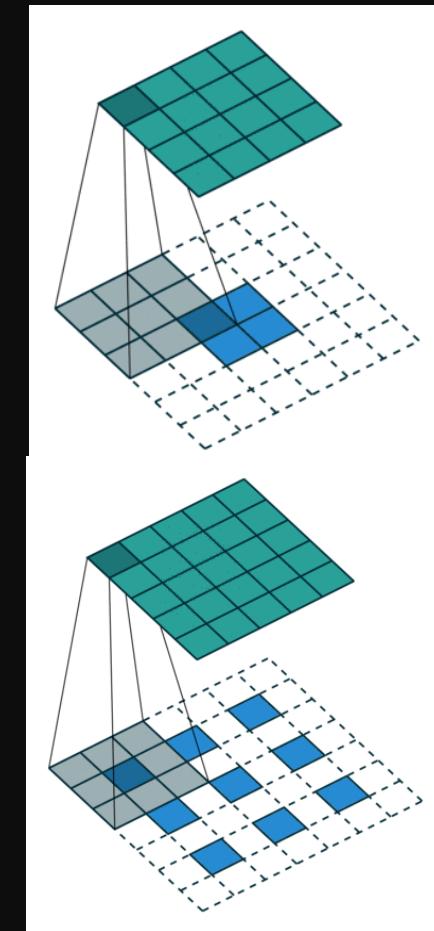
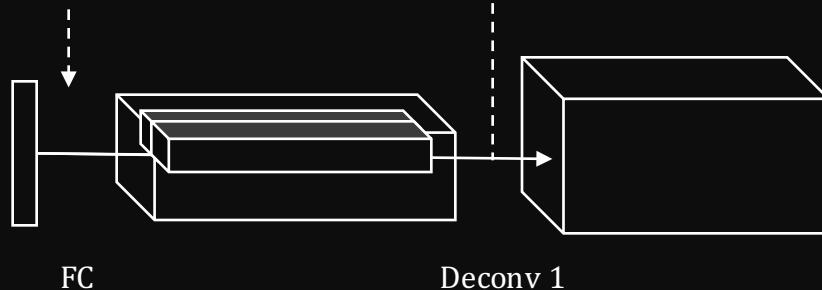
Matrix multiplication
(Fully connected)



생성자

Generator

Matrix multiplication
(Fully connected)

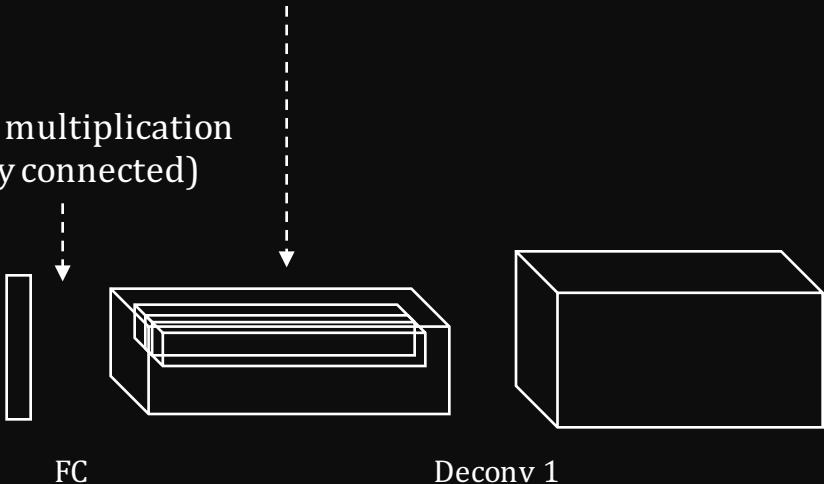


생성자

Generator

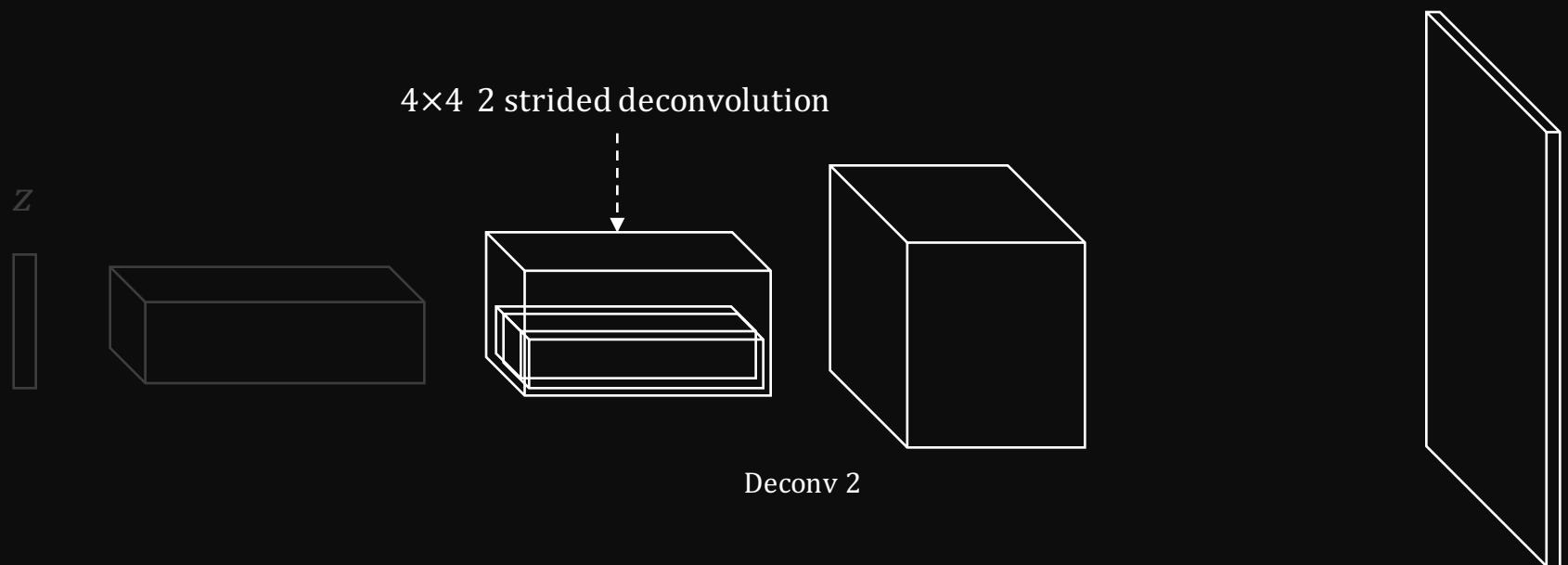
4×4 2 strided deconvolution

Matrix multiplication
(Fully connected)



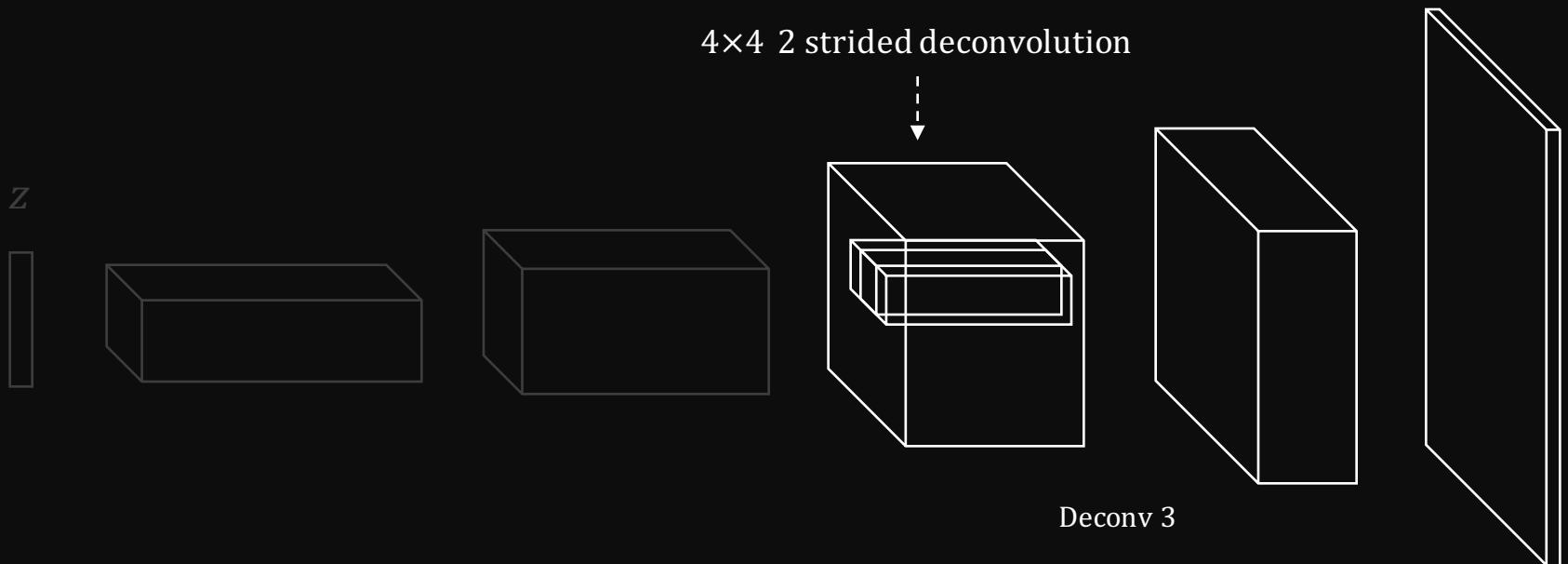
생성자

Generator



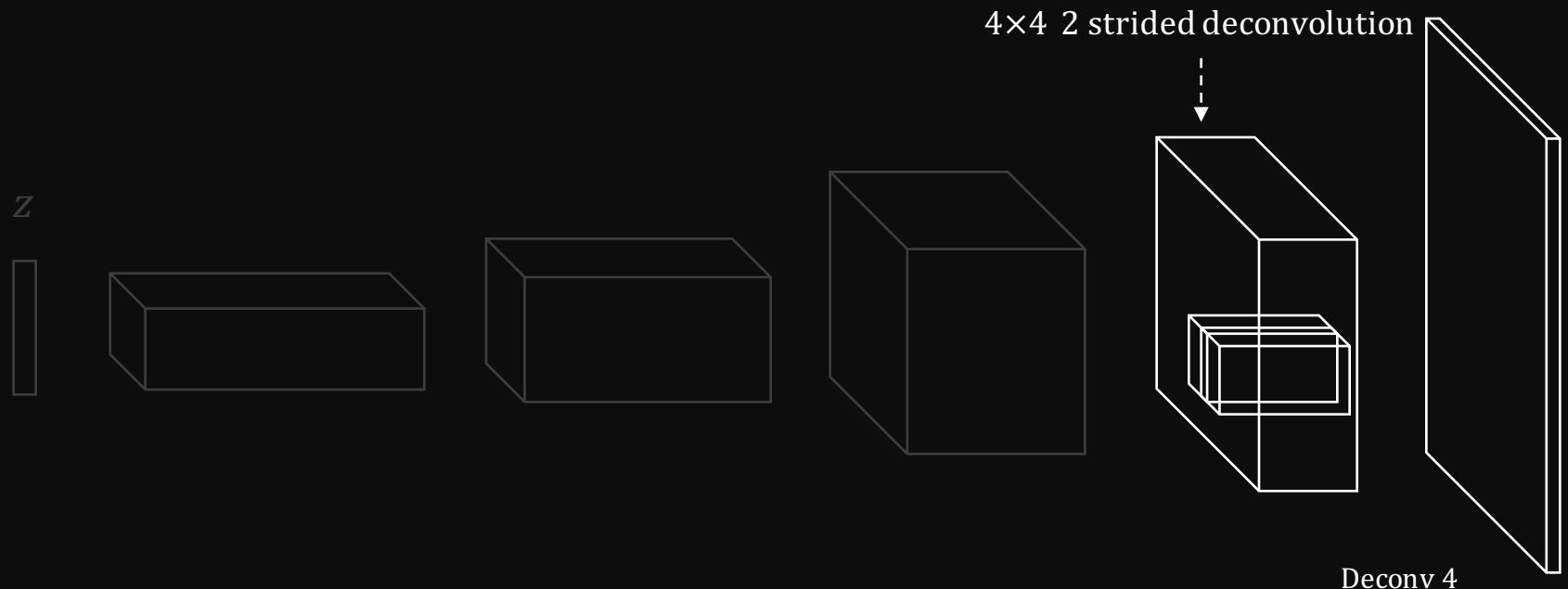
생성자

Generator



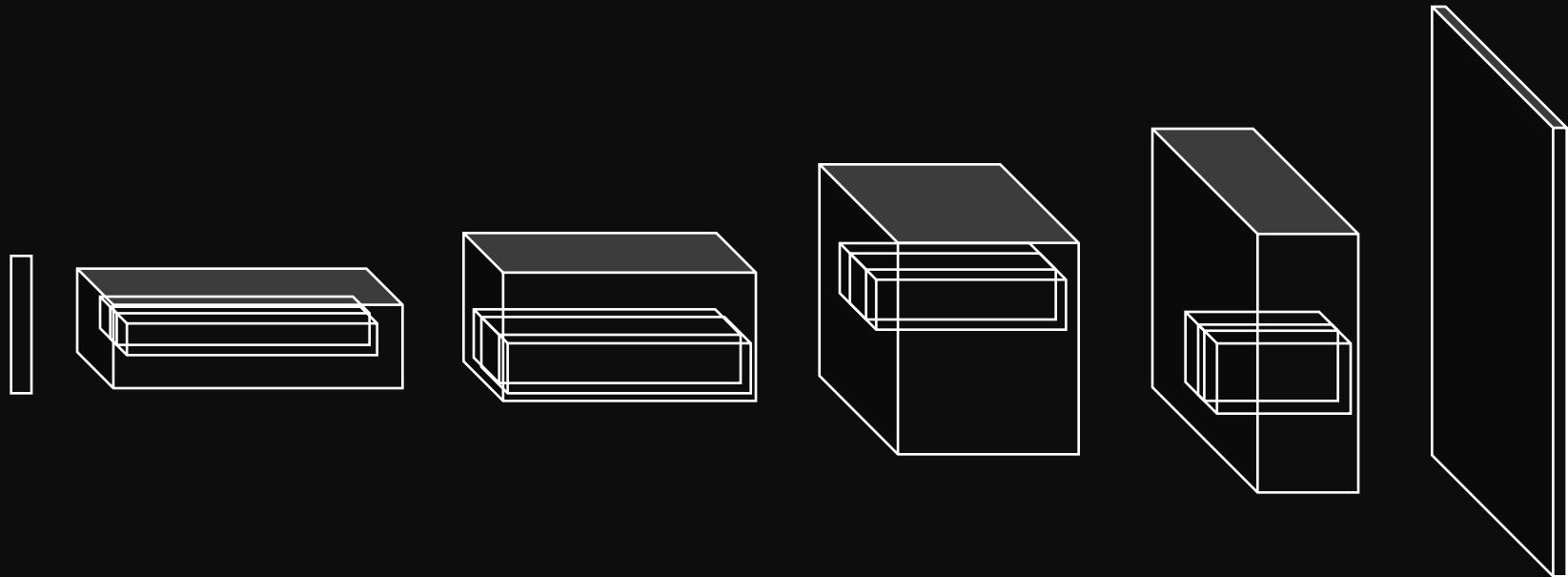
생성자

Generator



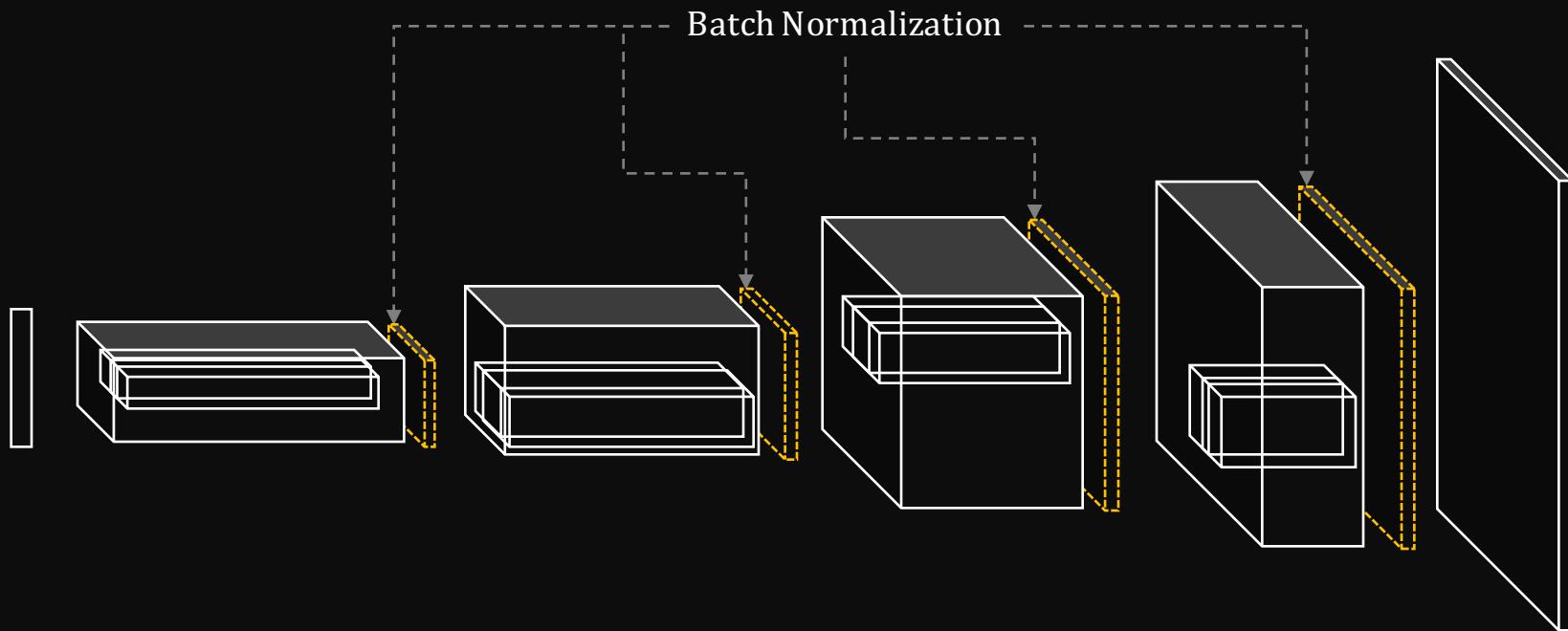
생성자

Generator



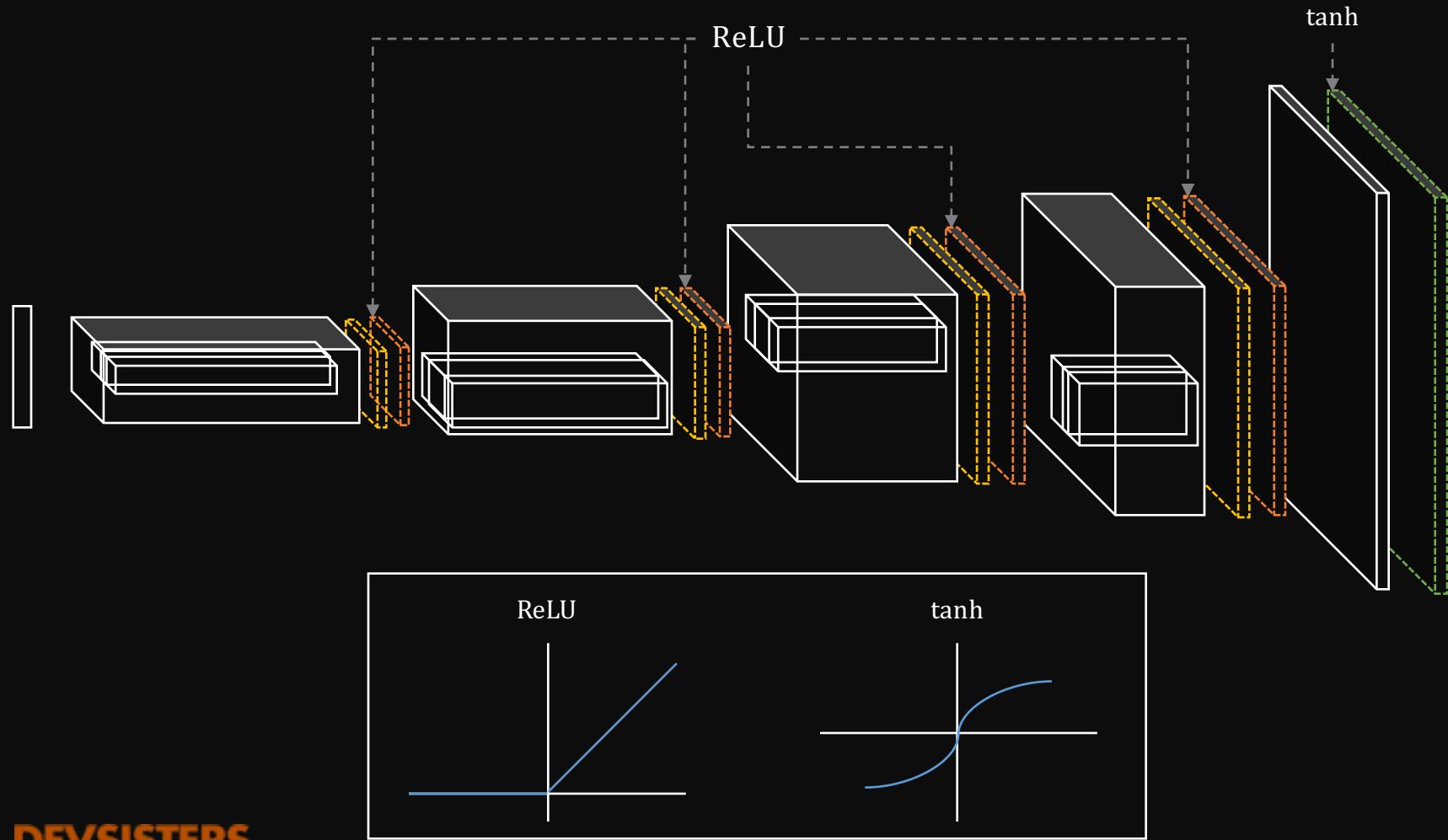
생성자

Generator



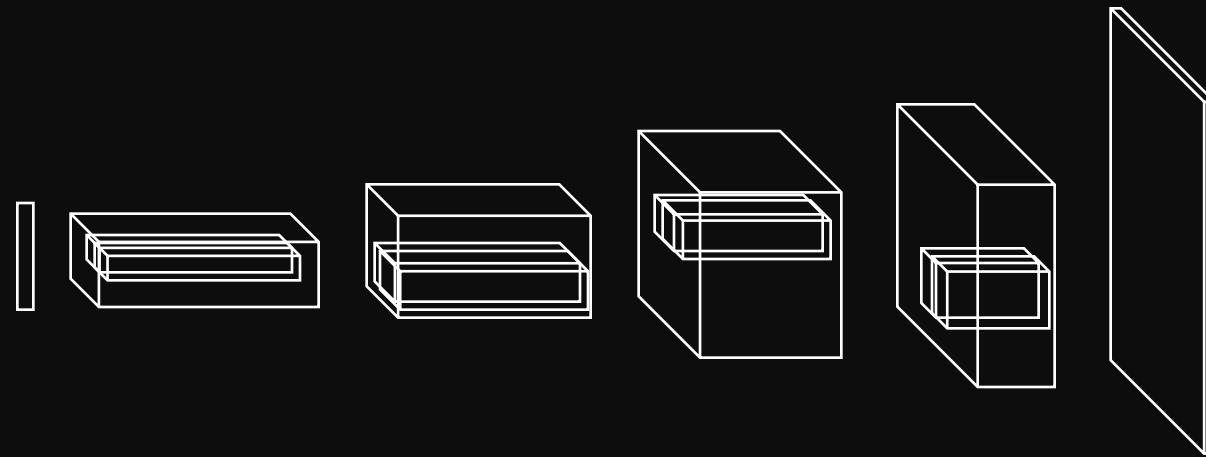
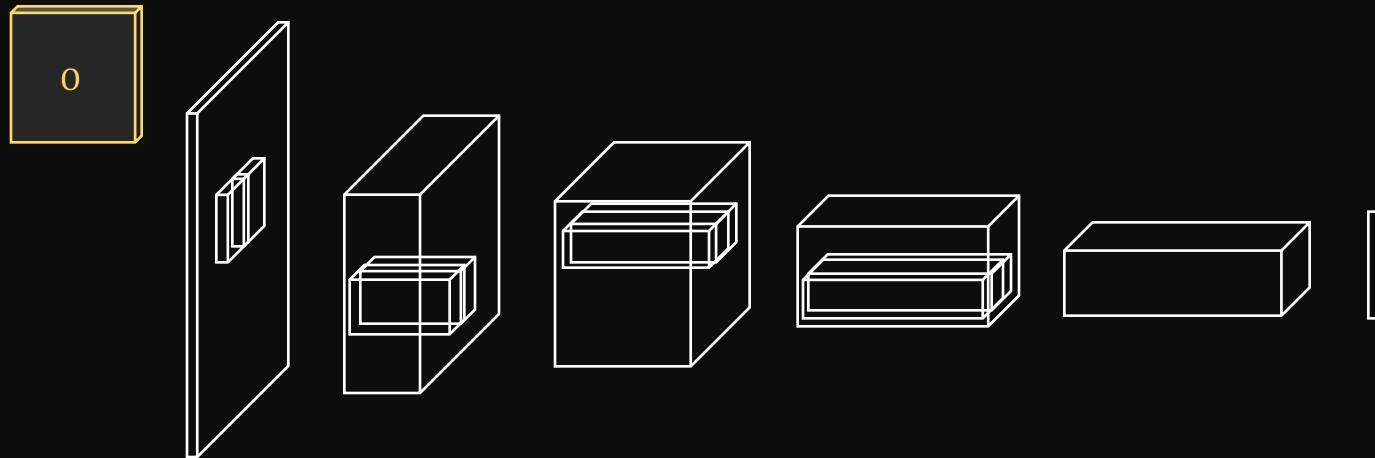
생성자

Generator

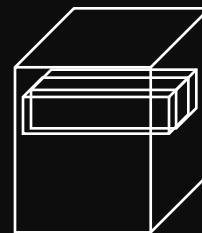
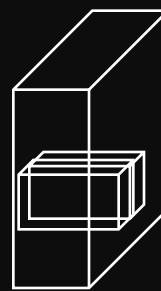


총 정리

(탕탕)

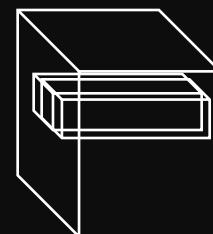
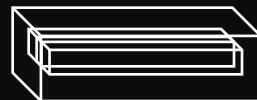


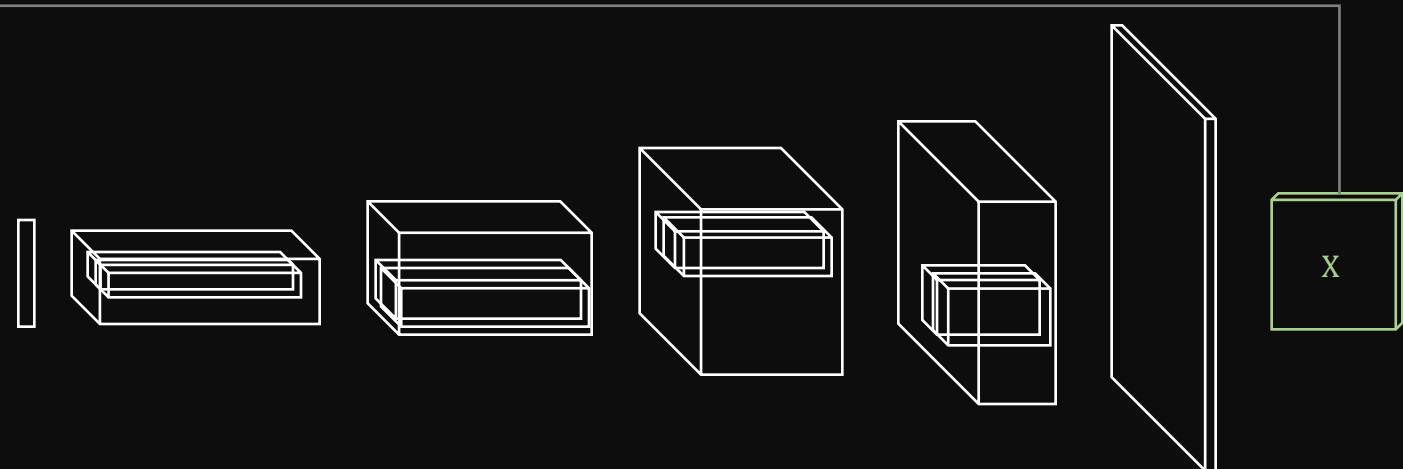
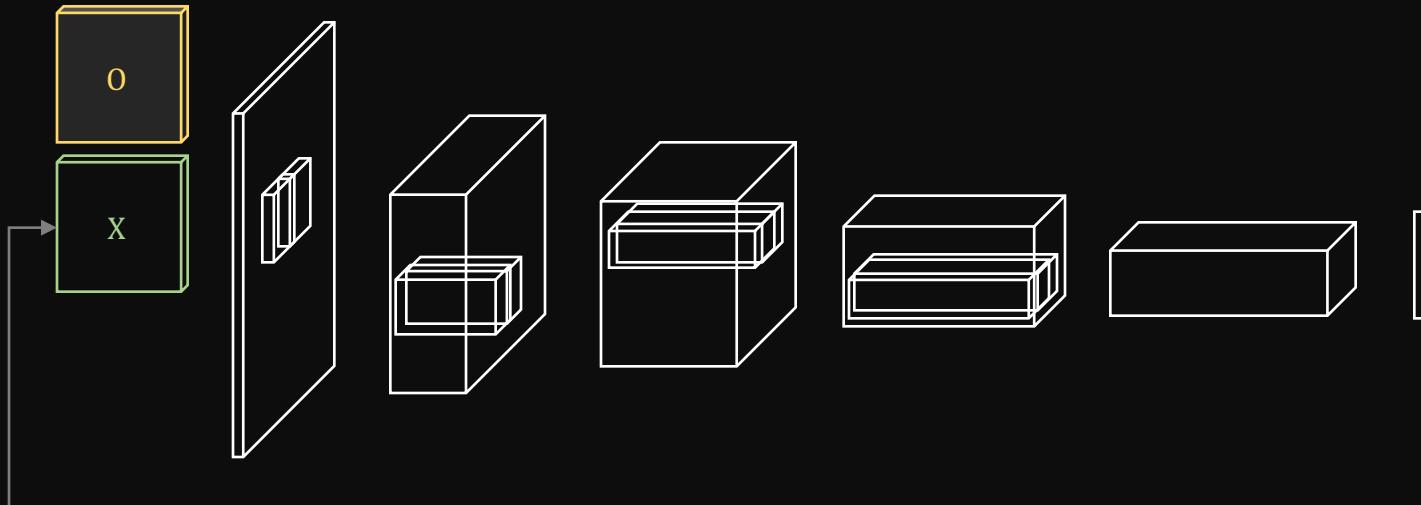
images

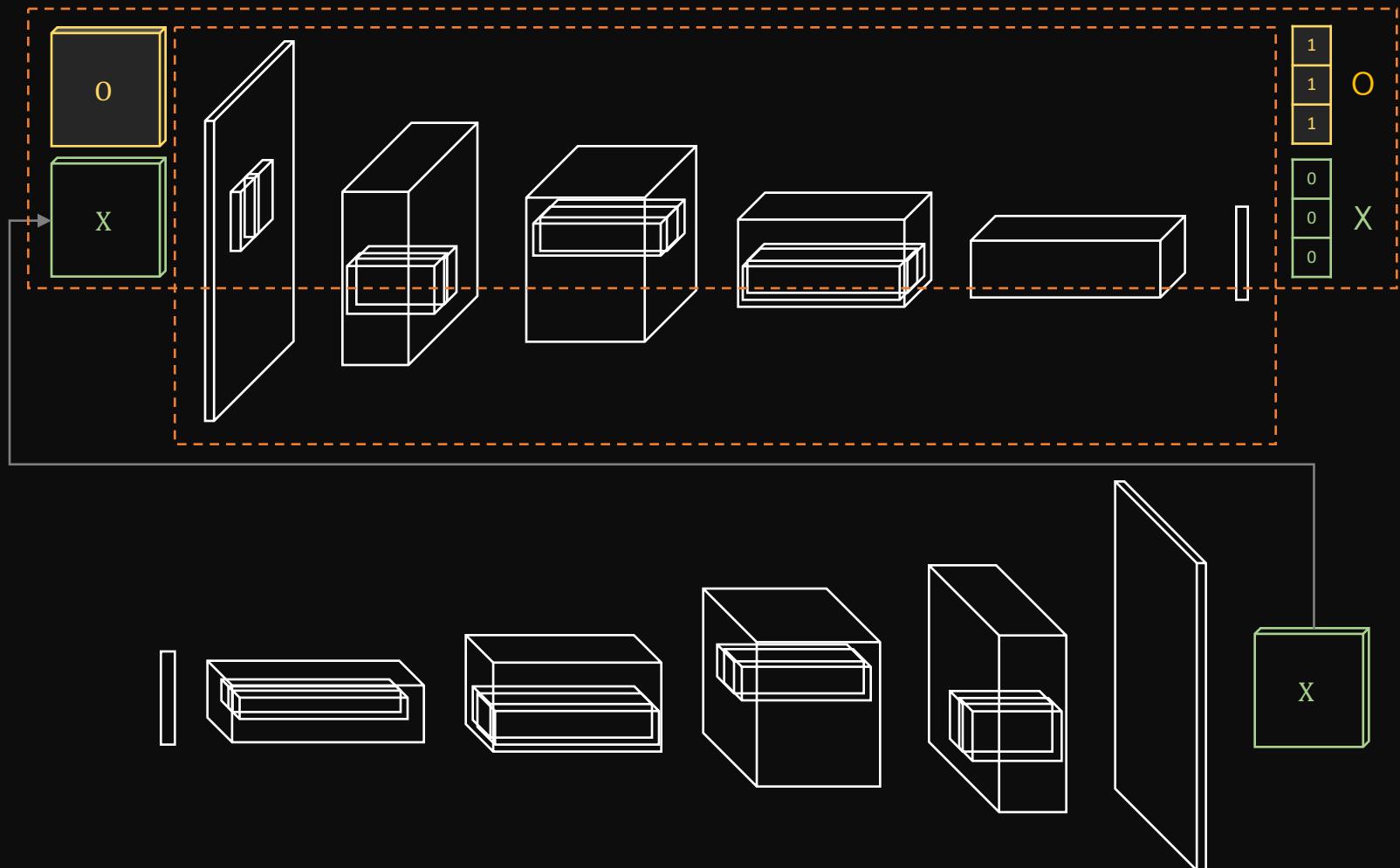


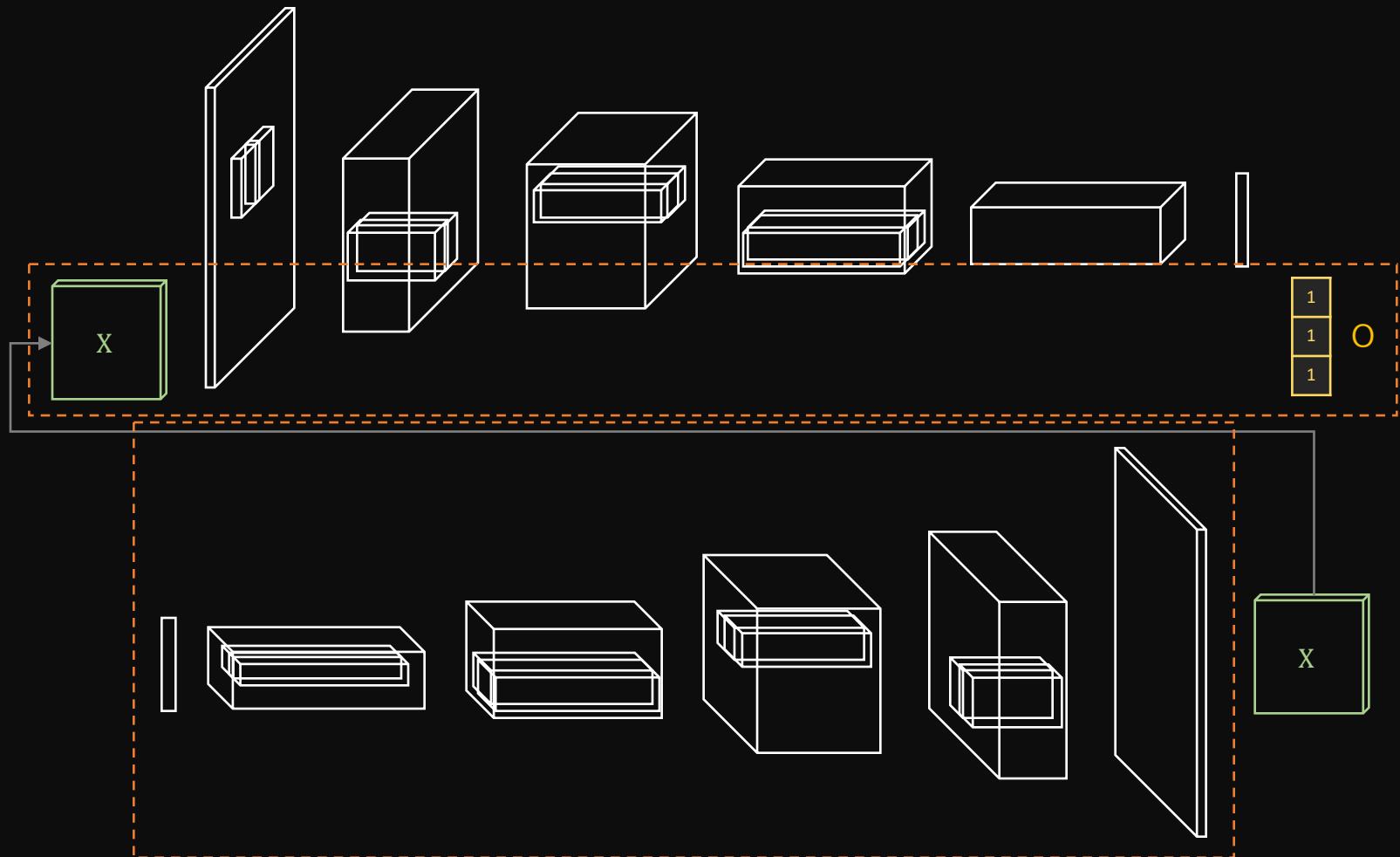
총 2개의 input

Z (Gaussian distribution)

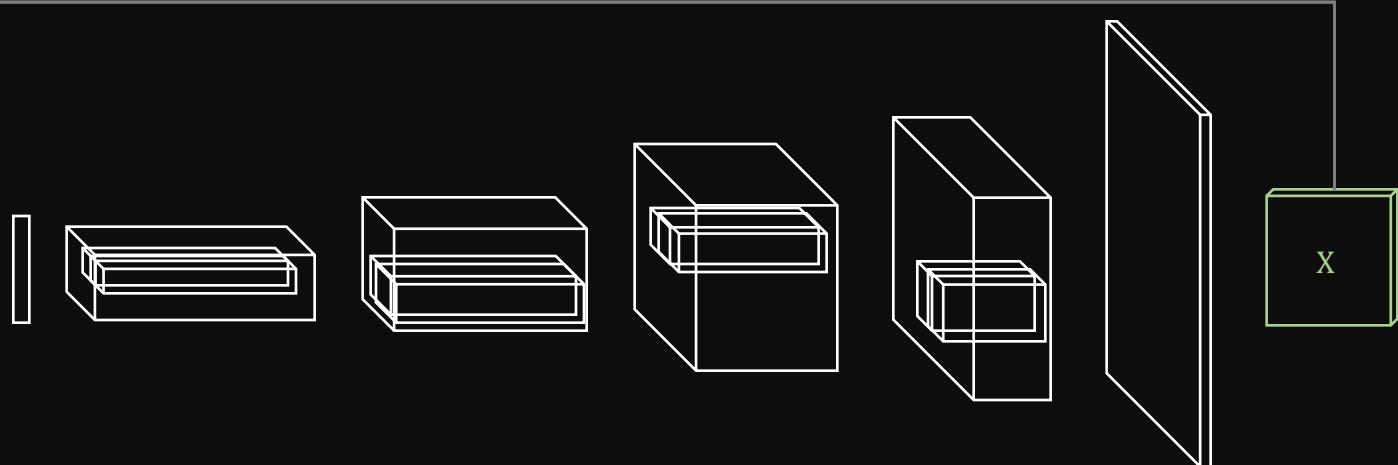
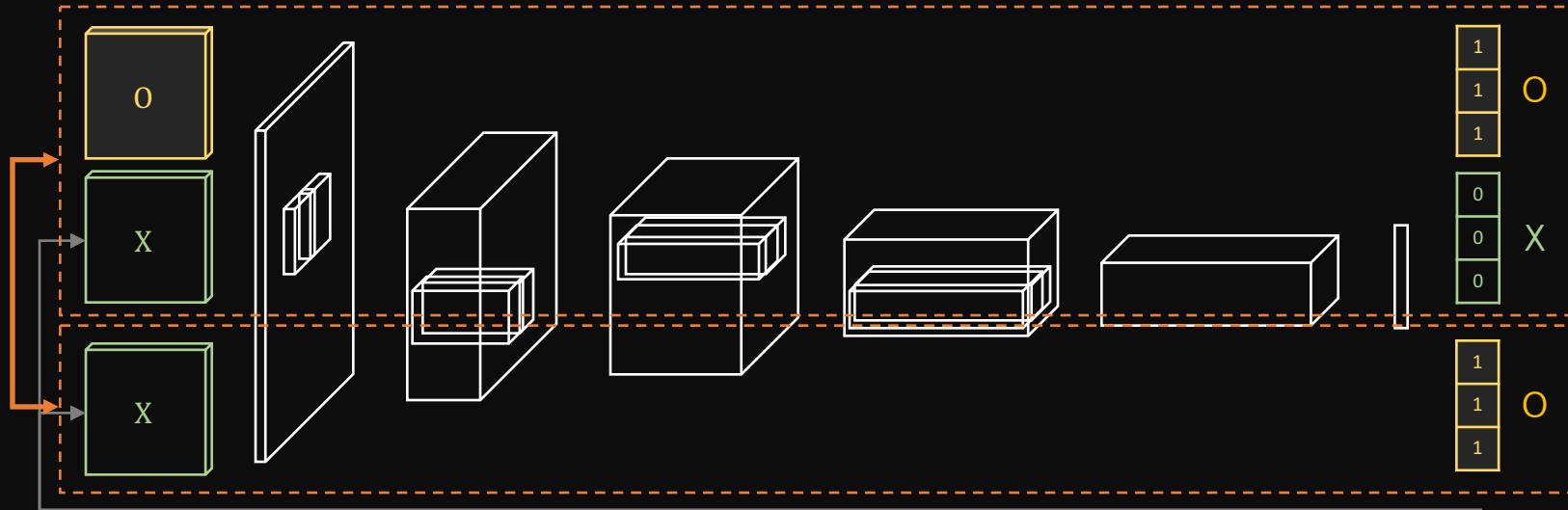








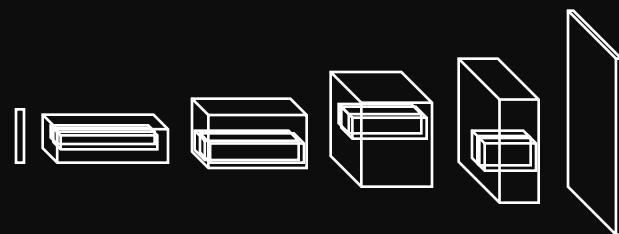
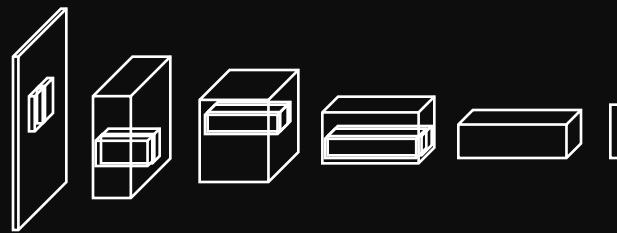
반복
학습



```
import tensorflow as tf
```

입력값

Inputs



입력값

Inputs

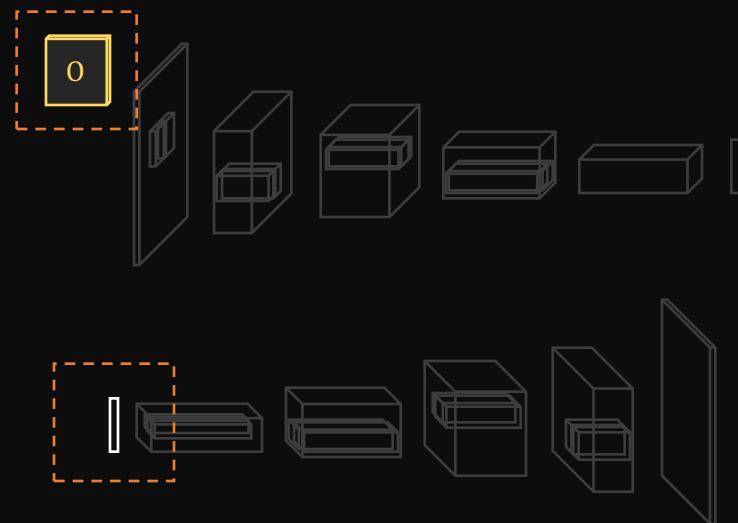
```
images = tf.placeholder(tf.float32, [32, 84, 84, 3])
```



입력값

Inputs

```
images = tf.placeholder(tf.float32, [32, 84, 84, 3])  
z = tf.placeholder(tf.float32, [32, 100], name='z')
```



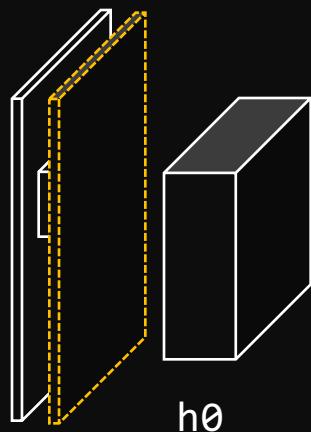
구분자

Discriminator

구분자

Discriminator

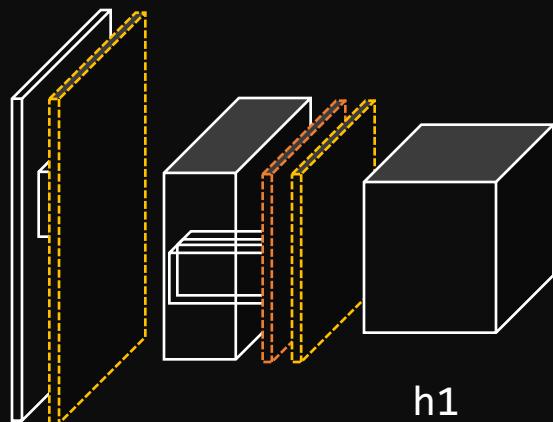
```
h0 = conv2d(images, 64, 4, 4, 2, 2)
```



구분자

Discriminator

```
h1_logits = batch_norm(conv2d(h0, 64 * 2, 4, 4, 2, 2))  
h1 = lrelu(h1_logits)
```



구분자

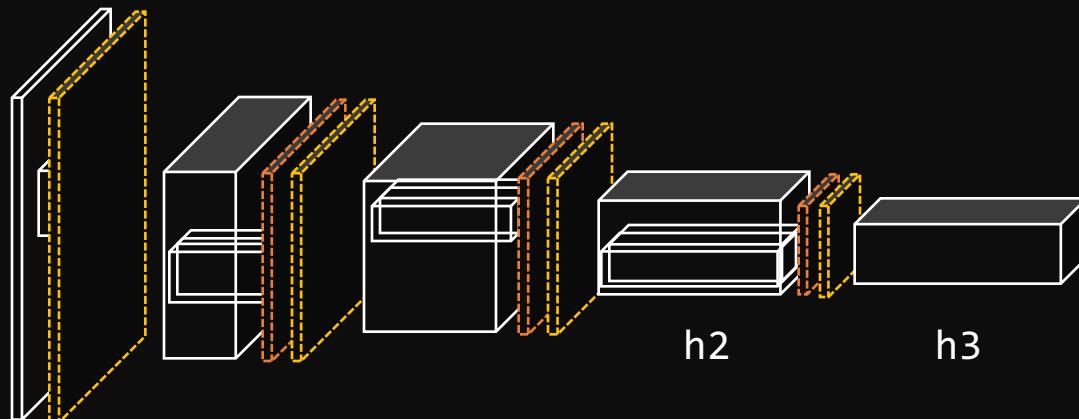
Discriminator

```
h2_logits = batch_norm(conv2d(h1, 64 * 4, 4, 4, 2, 2))
```

```
h2 = lrelu(h2_logits)
```

```
h3_logits = batch_norm(conv2d(h2, 64 * 8, 4, 4, 2, 2))
```

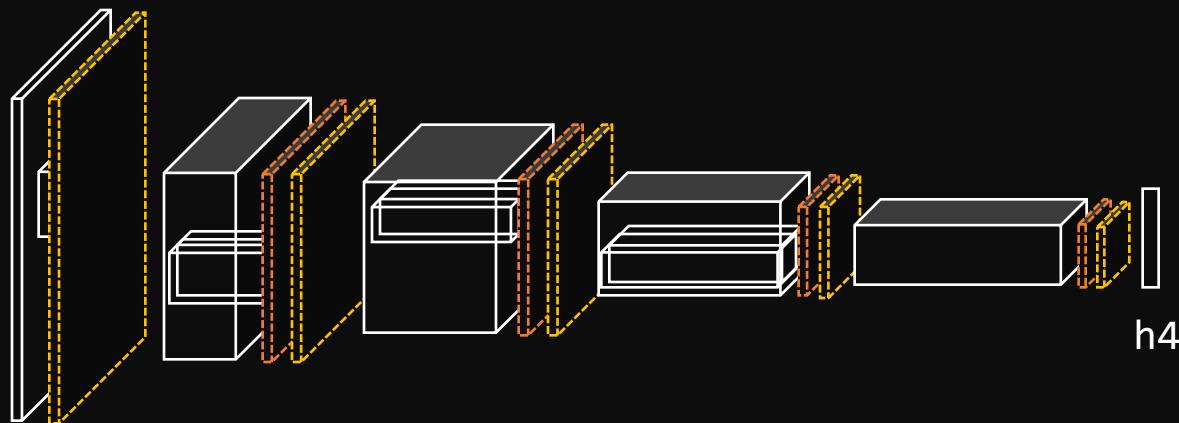
```
h3 = lrelu(h3_logits)
```



구분자

Discriminator

```
h4 = linear(tf.reshape(h3, [32, -1]))
```



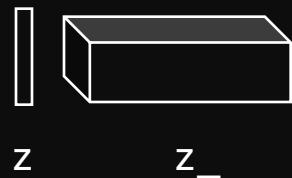
생성자

Geneartor

생성자

Geneartor

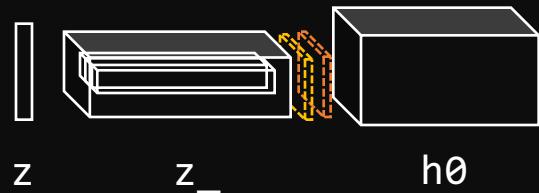
```
z_ = linear(z, 64*8*4*4)  
h0_logits = tf.reshape(z_, [-1, 4, 4, 64 * 8])
```



생성자

Geneartor

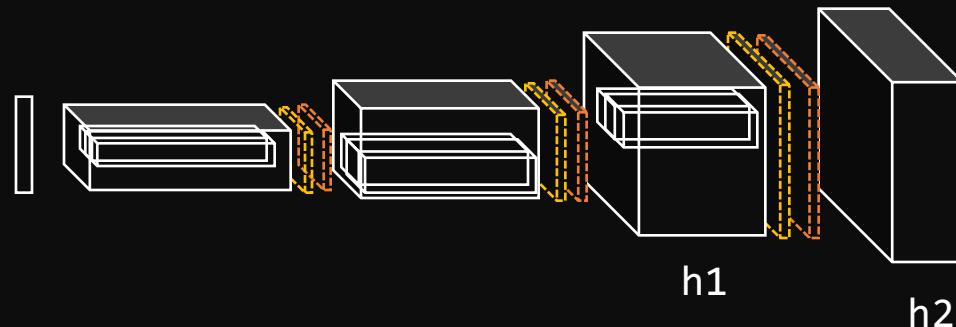
```
z_ = linear(z, 64*8*4*4)  
  
h0_logits = tf.reshape(z_, [-1, 4, 4, 64 * 8])  
  
h0 = tf.nn.relu(batch_norm(h0_logits))
```



생성자

Geneartor

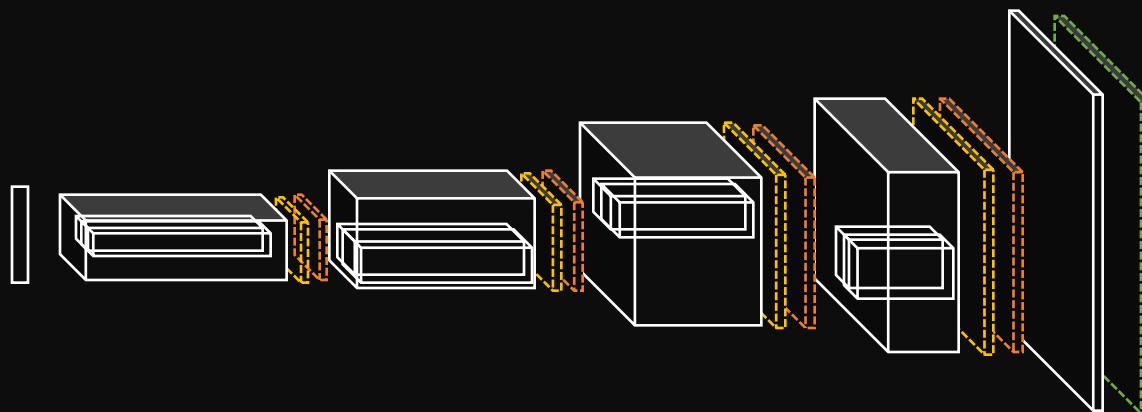
```
h1_logits = tf.nn.relu(deconv2d(h0, [32, 8, 8, 64*4])  
  
h1 = batch_norm(h1_logits))  
  
h2_logits = tf.nn.relu(deconv2d(h1, [32, 8, 8, 64*4]))  
  
h2 = batch_norm(h2_logits))
```



생성자

Geneartor

```
h3 = tf.nn.tanh(deconv2d(h2, [32, 8, 8, 64*4]))
```



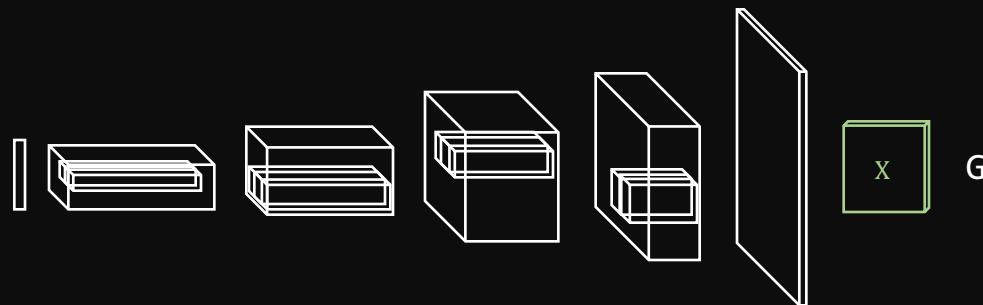
적대적 학습

Adversarial Learning

적대적 학습

Adversarial Learning

$G = \text{generator}(z)$

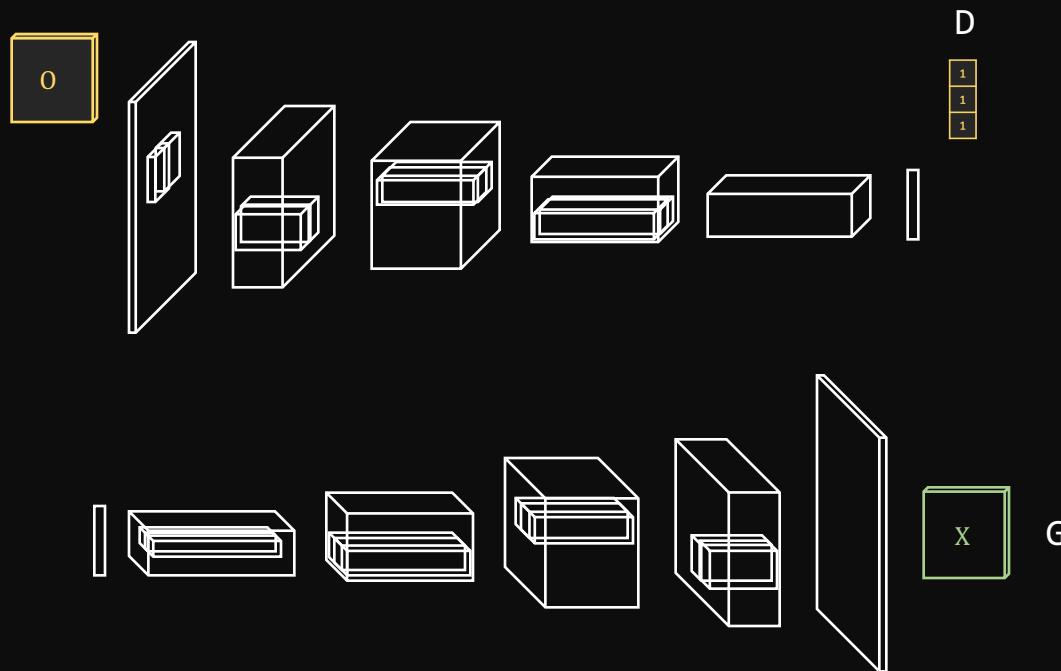


적대적 학습

Adversarial Learning

$G = \text{generator}(z)$

$D = \text{discriminator}(\text{images})$



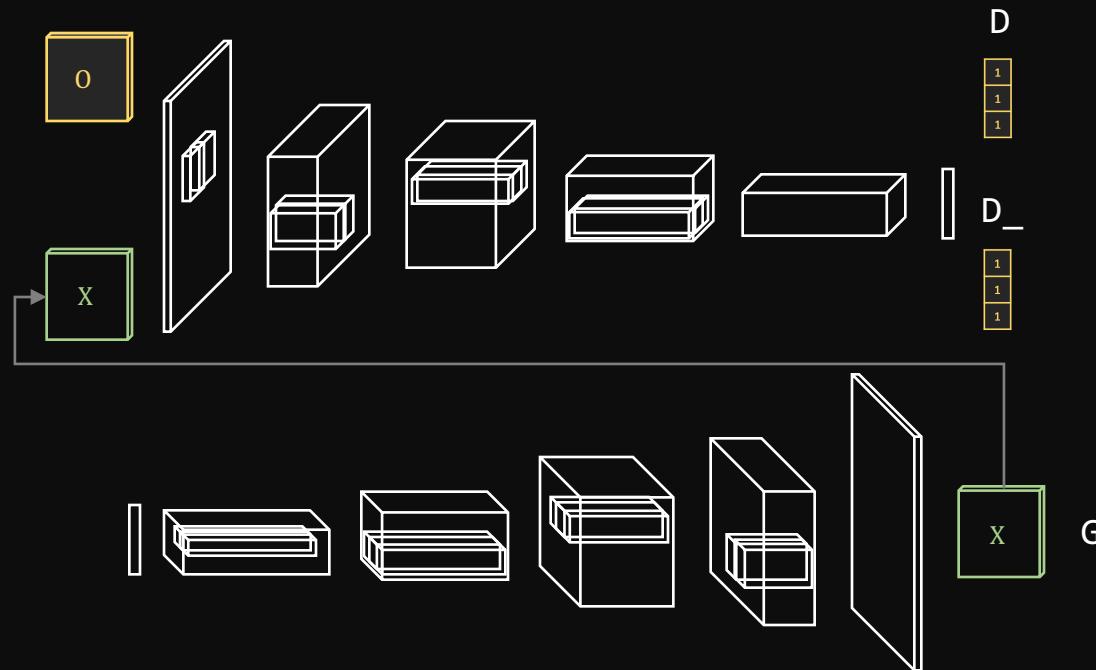
적대적 학습

Adversarial Learning

$G = \text{generator}(z)$

$D = \text{discriminator}(\text{images})$

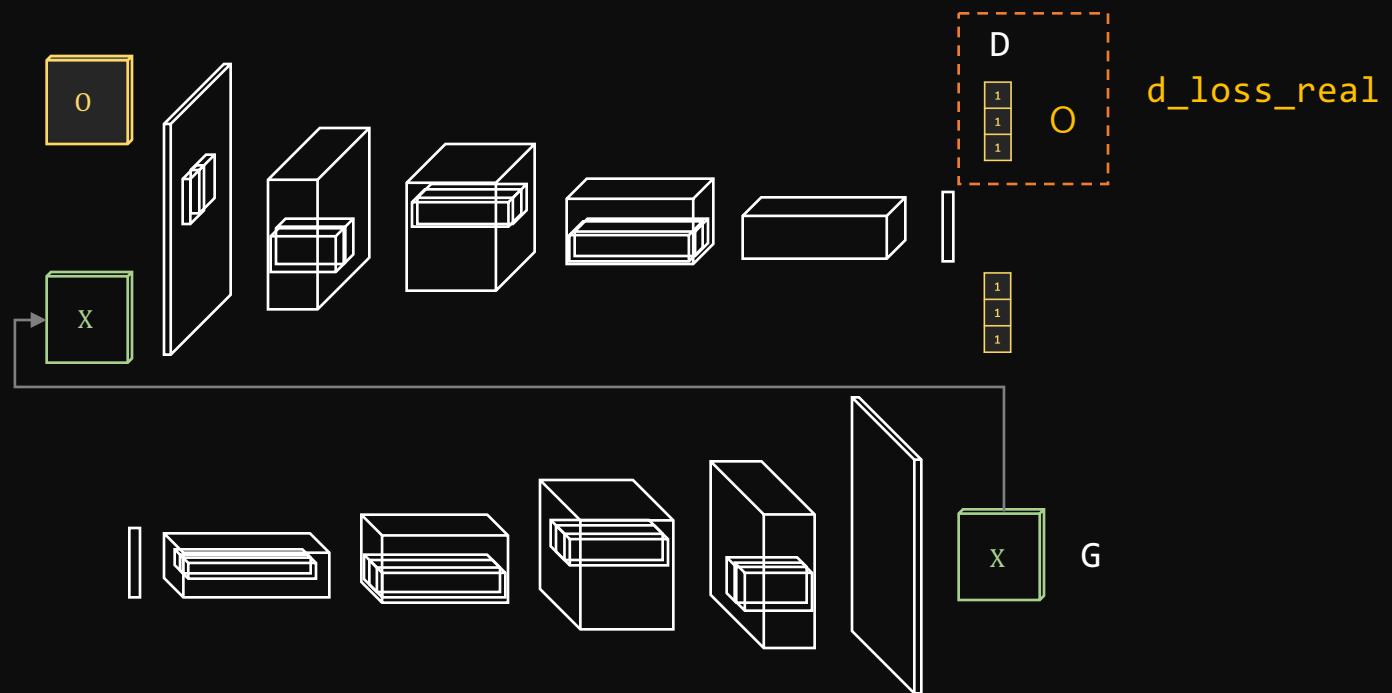
$D_ = \text{discriminator}(G, \text{reuse=True})$



적대적 학습

Adversarial Learning

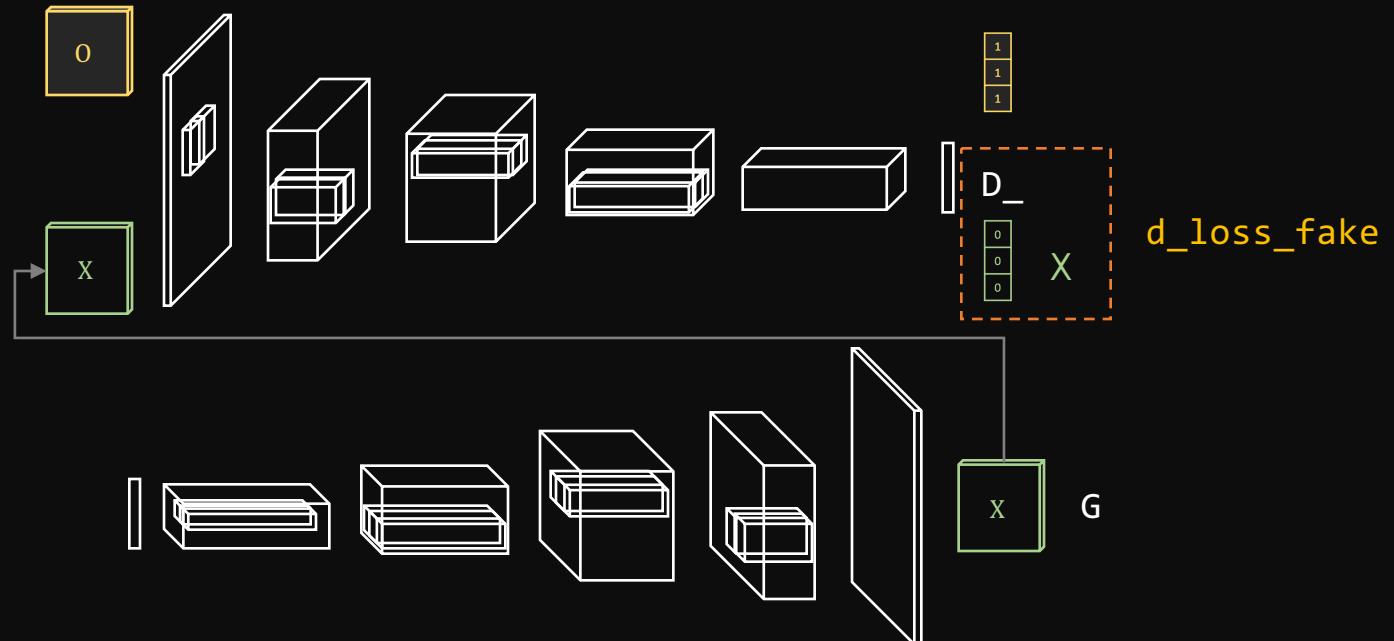
```
d_loss_real = tf.reduce_mean(  
    tf.nn.sigmoid_cross_entropy_with_logits(D, tf.ones_like(D)))
```



적대적 학습

Adversarial Learning

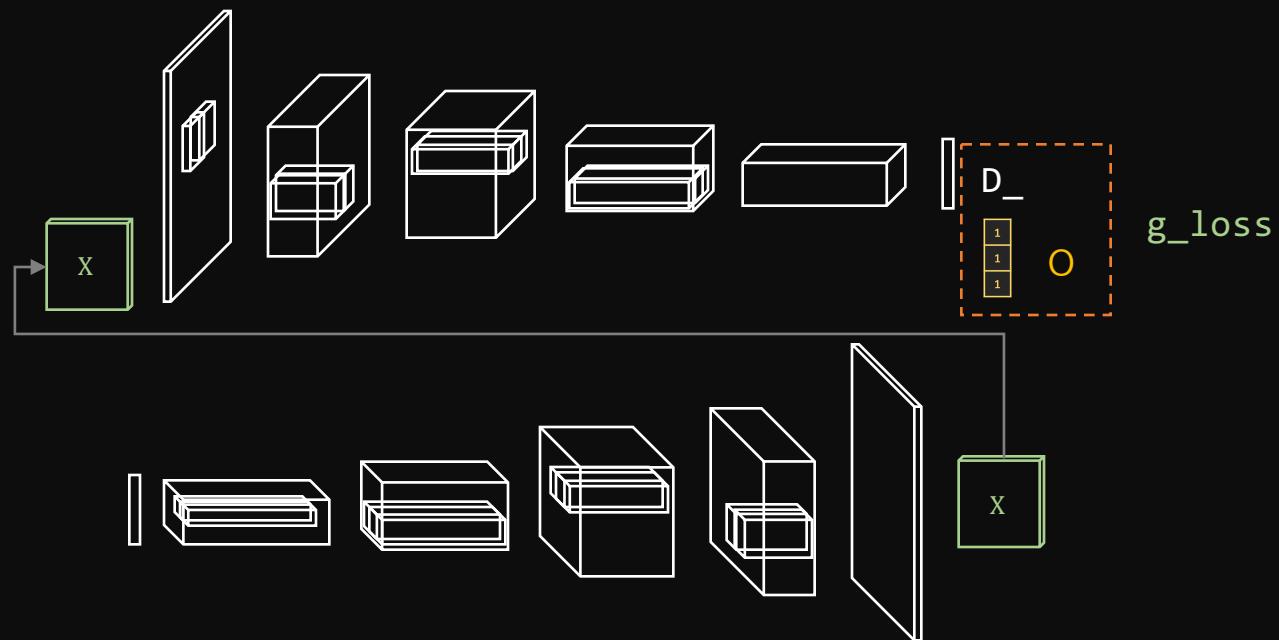
```
d_loss_real = tf.reduce_mean(  
    tf.nn.sigmoid_cross_entropy_with_logits(D, tf.ones_like(D)))  
  
d_loss_fake = tf.reduce_mean(  
    tf.nn.sigmoid_cross_entropy_with_logits(D_, tf.zeros_like(D_)))
```



적대적 학습

Adversarial Learning

```
g_loss = tf.reduce_mean(  
    tf.nn.sigmoid_cross_entropy_with_logits(D_logits_, tf.ones_like(D_)))
```



적대적 학습

Adversarial Learning

```
d_loss = d_loss_real + d_loss_fake
```

적대적 학습

Adversarial Learning

```
d_loss = d_loss_real + d_loss_fake
```

```
d_optim = tf.train.AdamOptimizer(0.0002).minimize(d_loss)
```

```
g_optim = tf.train.AdamOptimizer(0.0002).minimize(g_loss)
```

적대적 학습

Adversarial Learning

```
d_loss = d_loss_real + d_loss_fake
```

```
d_optim = tf.train.AdamOptimizer(0.0002).minimize(d_loss)
```

```
g_optim = tf.train.AdamOptimizer(0.0002).minimize(g_loss)
```

```
while True:
```

```
    sess.run(d_optim, { images: batch_images, z: batch_z })
```

```
    sess.run(g_optim, { z: batch_z })
```



학습 결과

0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9

MNIST
(Real)

1 3 9 6 9 5 3 2
4 2 9 8 2 1 6 5
0 2 3 9 5 1 4 4
8 3 1 8 3 4 1 2
9 7 2 9 3 3 9 2
8 3 6 3 5 0 7 7
3 3 6 8 2 1 2 8
6 4 7 5 5 2 3 6

MNIST
(Fake)

0 9 6 5 3 3 6 5
5 3 6 4 4 9 0 0
7 0 1 3 5 0 0 9
6 3 4 2 3 9 4 3
7 7 1 1 6 1 0 7
3 9 9 0 2 5 9 8
8 4 8 6 6 3 9 8
3 7 2 3 5 2 4 1

MNIST
(Fake)

2 3 5 6 5 2 9 3
3 9 4 0 8 4 4 2
0 3 8 4 3 0 1 7
8 4 4 7 5 1 0 0
6 2 3 3 5 0 9 7
6 3 8 5 2 1 7 9
1 2 3 1 5 3 0 3
9 8 4 2 0 9 0 8

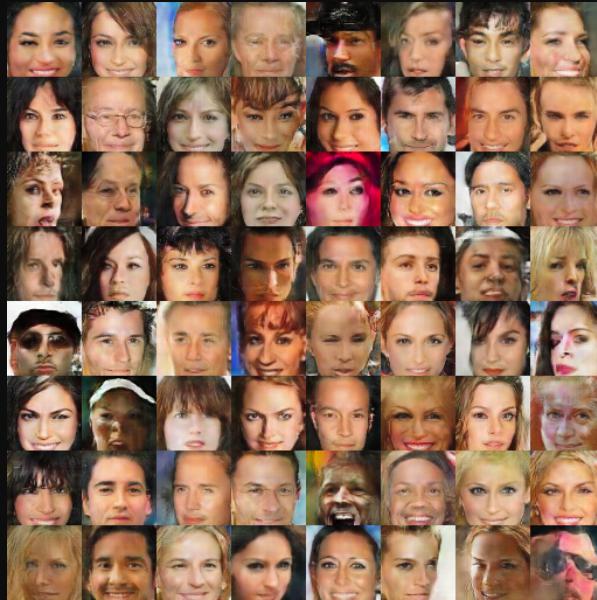
MNIST
(Fake)



CelebA
(Real)



CelebA
(Fake)

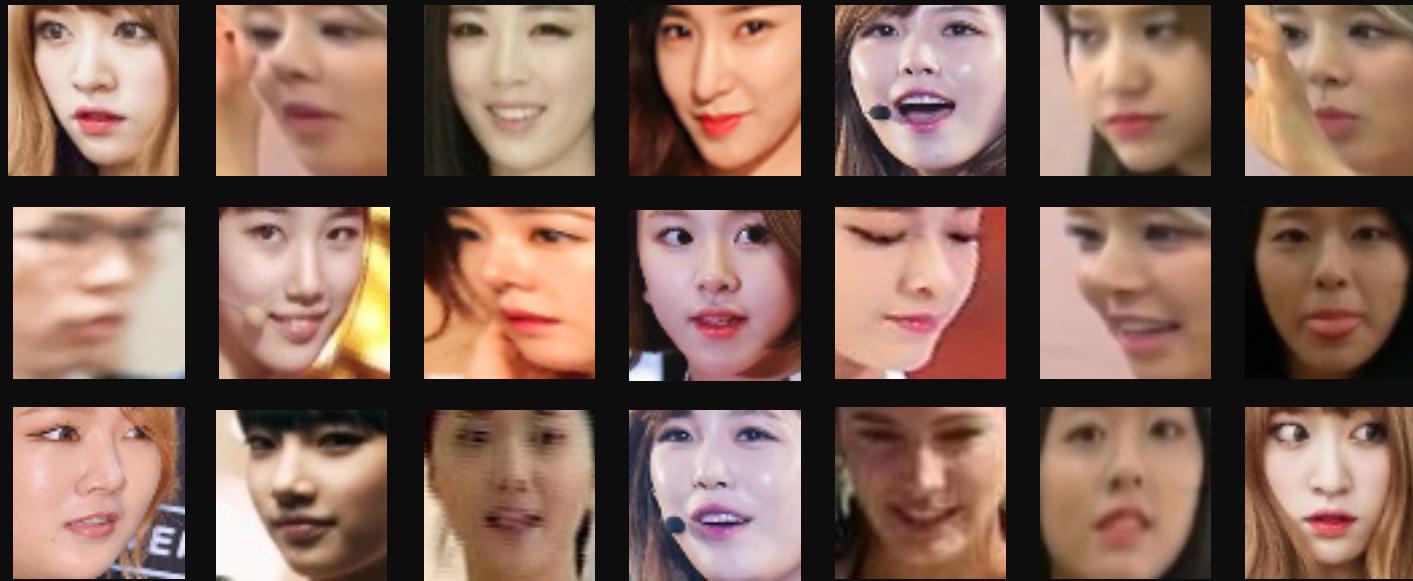


CelebA
(Fake)

한국인 얼굴 데이터



한국인 얼굴 데이터
(Real)



한국인 얼굴 데이터
(Real)

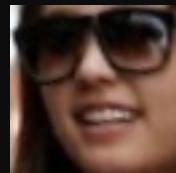
마이크



안경



흑백



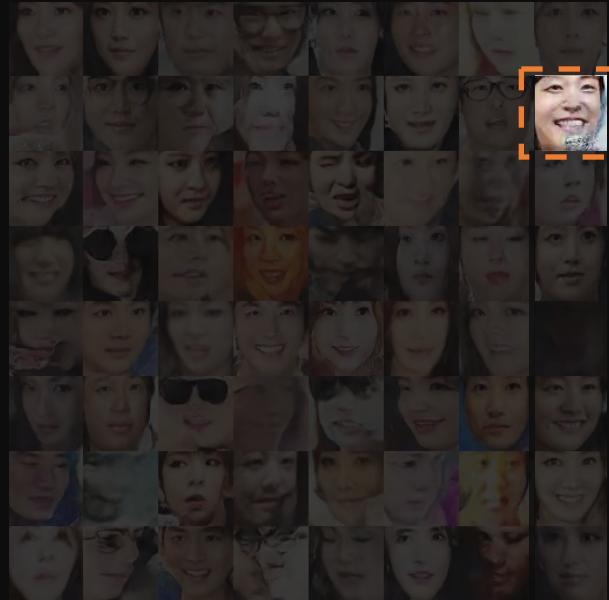
한국인 얼굴 데이터
(Real)



한국인 얼굴 데이터
(Fake)



한국인 얼굴 데이터
(Fake)



한국인 얼굴 데이터
(Fake)



한국인 얼굴 데이터
(Fake)



한국인 얼굴 데이터
(Fake)



한국인 얼굴 데이터
(Fake)



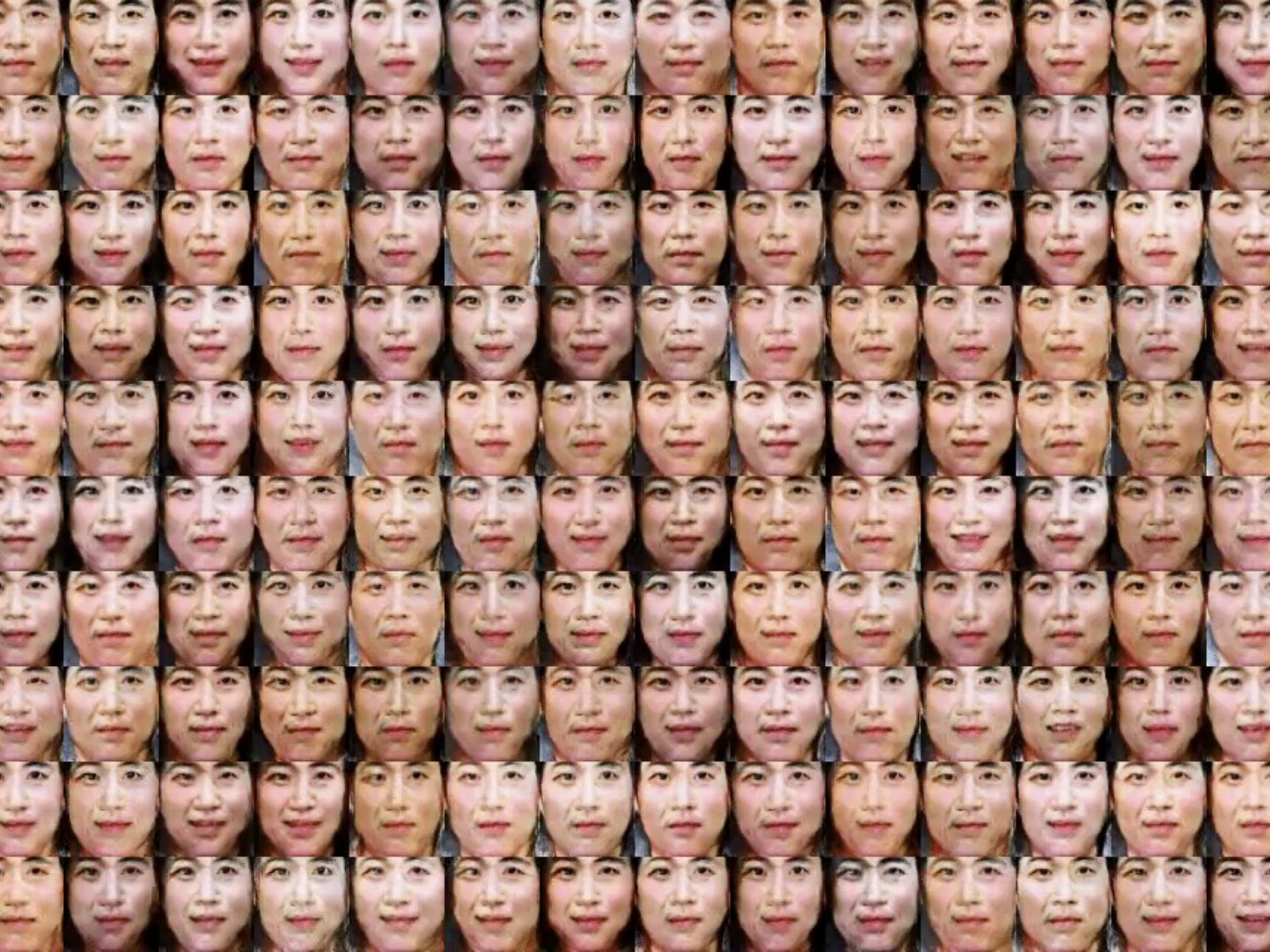
한국인 얼굴 데이터
(Fake)



한국인 얼굴 데이터
(Fake)



한국인 얼굴 데이터
(Fake)



프사 뉴럴

인공지능이 만든 얼굴들



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

<http://carpedm20.github.io/faces/>

Resources

- OpenAI's “Generative Models” blog post
- Image Completion with Deep Learning in TensorFlow
- A Beginner's Guide to Variational Methods: Mean-Field Approximation

DEVSISTERS

Algorithm Learning

Algorithm Learning

알고리즘의 입력과 출력을 보고 알고리즘을 배울 수 있을까?

3, 1, 2, 5 → 1, 2, 3, 5

Algorithm Learning

알고리즘의 입력과 출력을 보고 알고리즘을 배울 수 있을까?

3, 1, 2, 5 → 1, 2, 3, 5

9, 2, 4, 1 → 1, 2, 4, 9

7, 2, 6, 9 → 2, 6, 7, 9

9, 2, 4, 0 → 0, 2, 4, 9

...

Neural Turing Machine

Neural Turing Machine

뉴럴 네트워크로 투링 머신을 만들 수 있다면?

튜링 머신

Turing machine

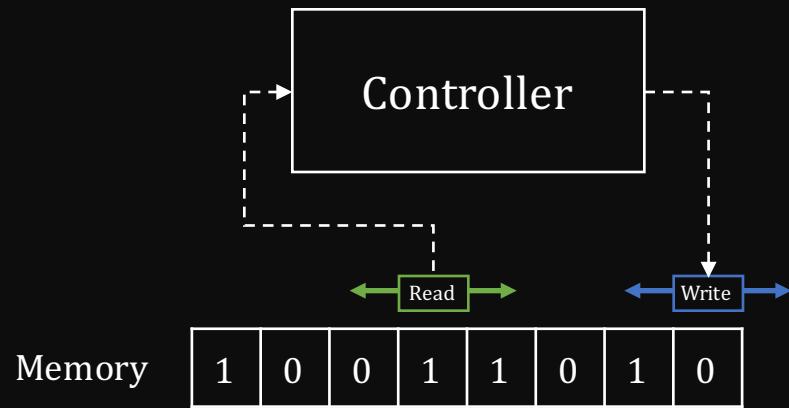
Controller

Memory

1	0	0	1	1	0	1	0
---	---	---	---	---	---	---	---

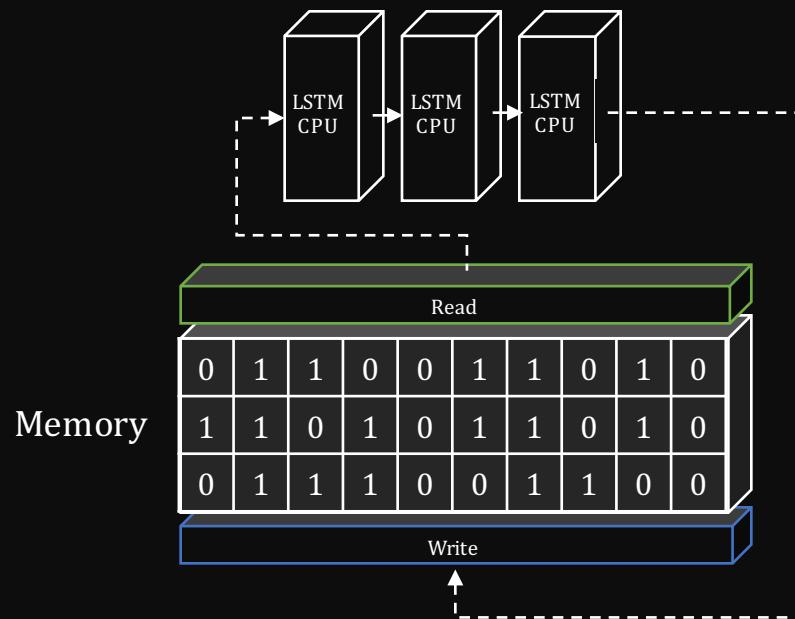
튜링 머신

Turing machine

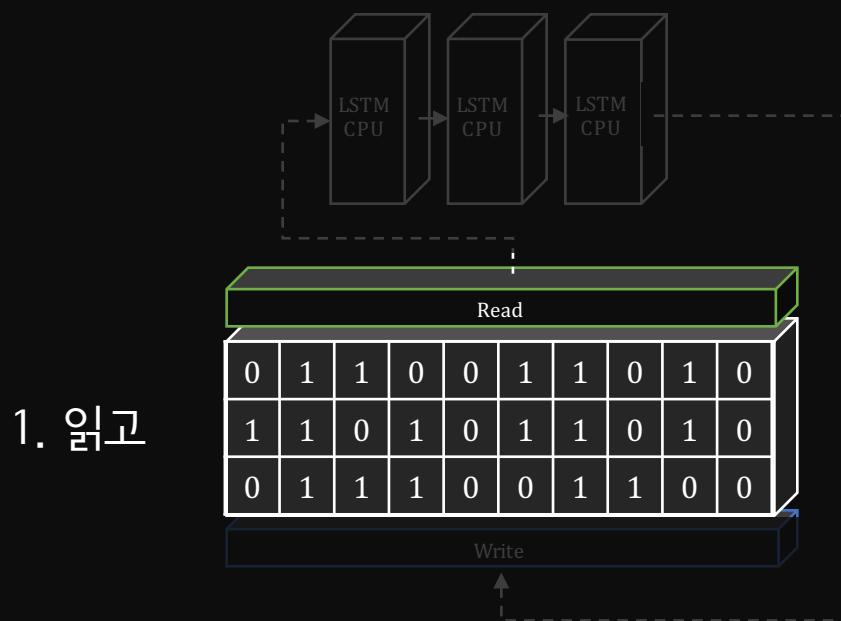


뉴럴 투링 머신

Neural Turing machine

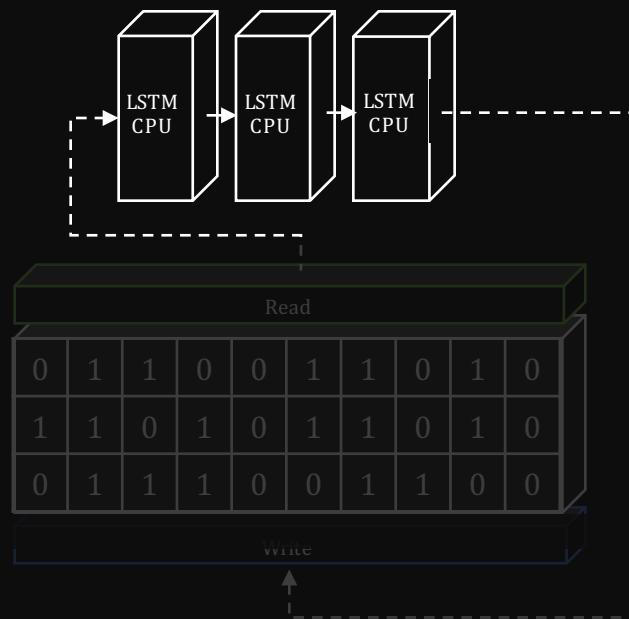


실행 과정

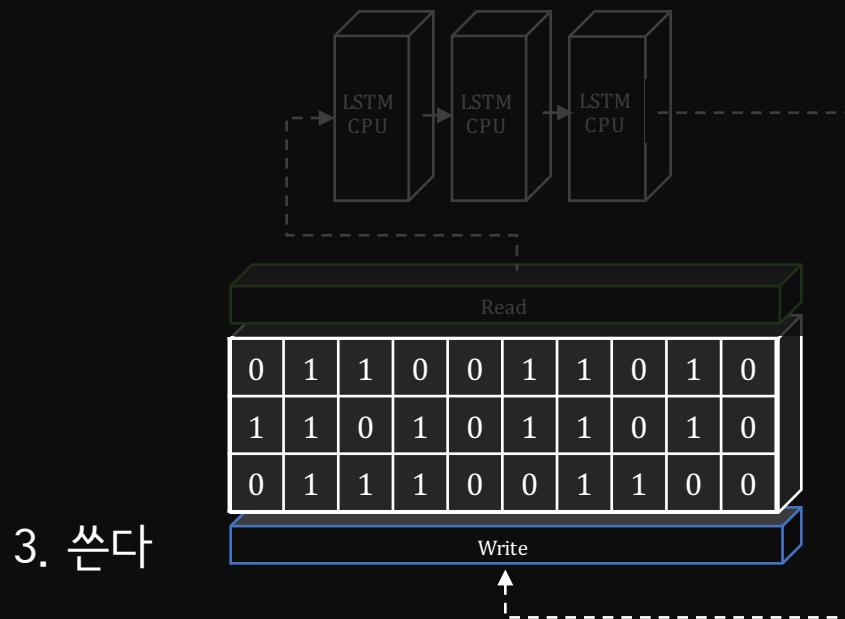


실행 과정

2. 처리하고



실행 과정



$t=1$



Start symbol

t=2

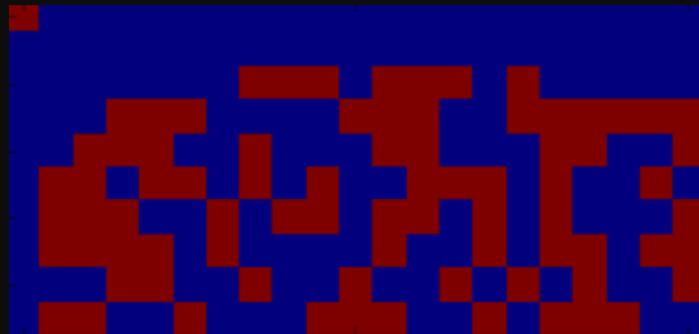


하나씩 읽어 메모리에 저장

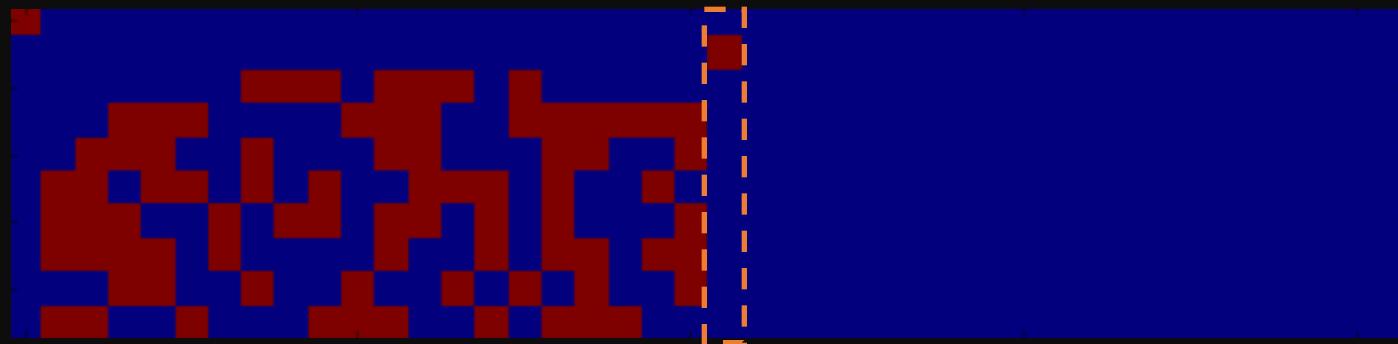
t=3



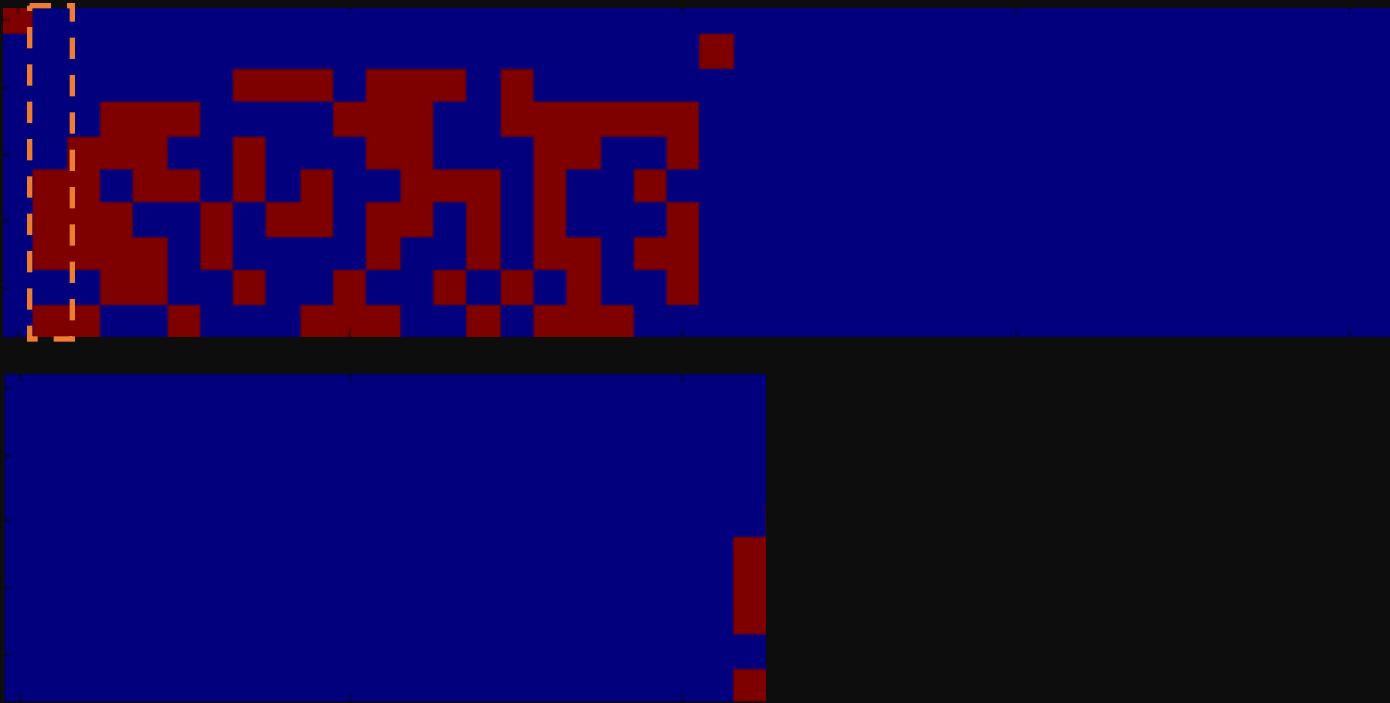
하나씩 읽어 메모리에 저장



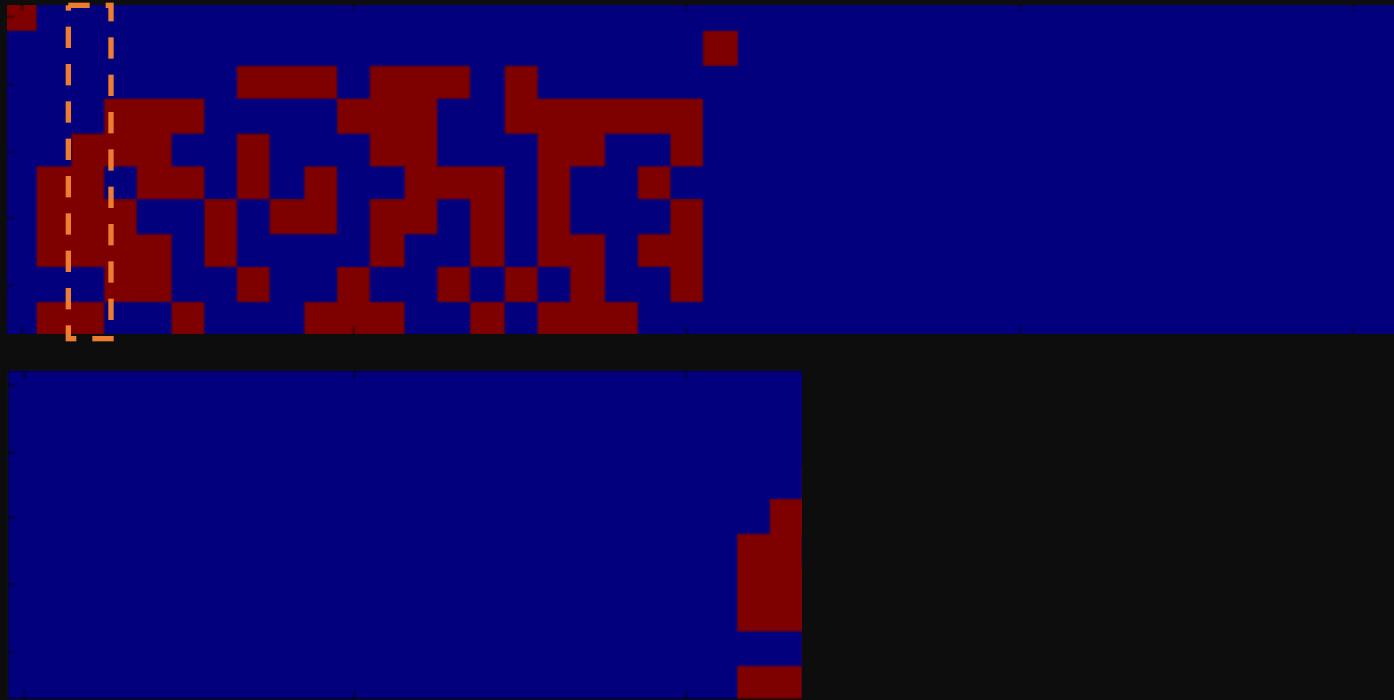
DEVSISTERS



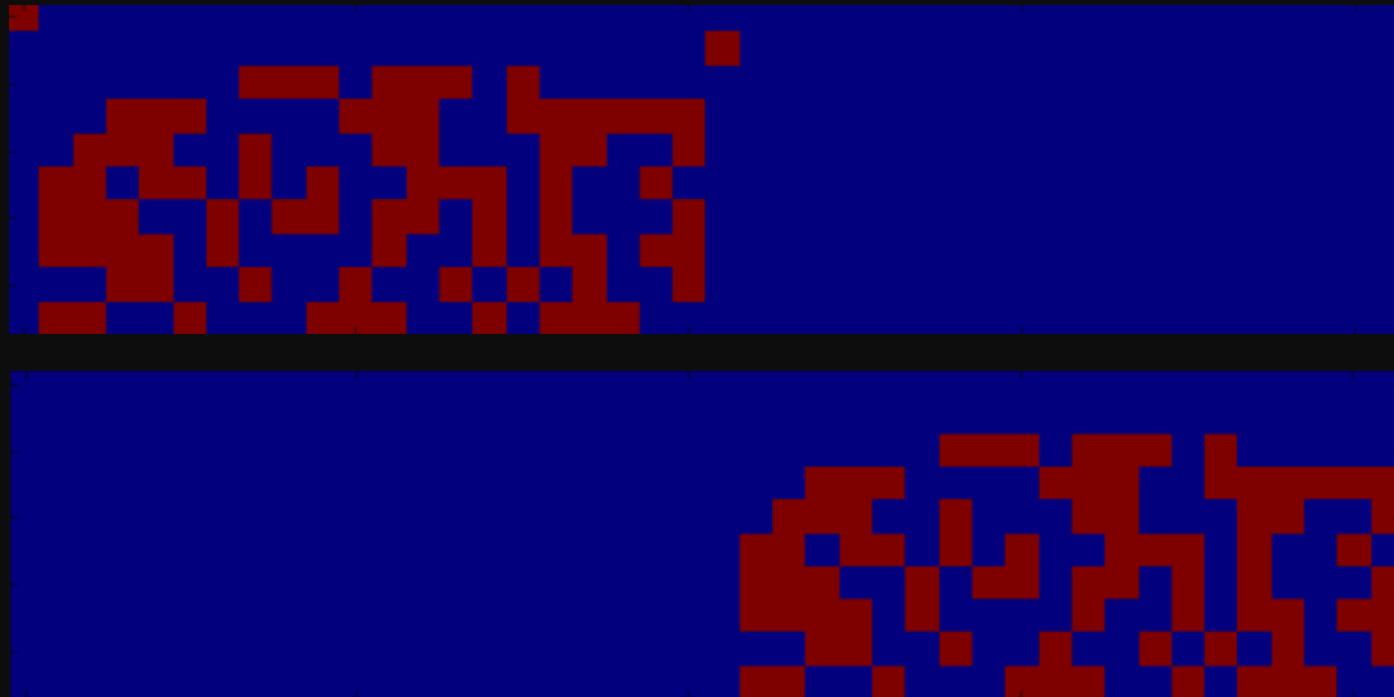
End symbol

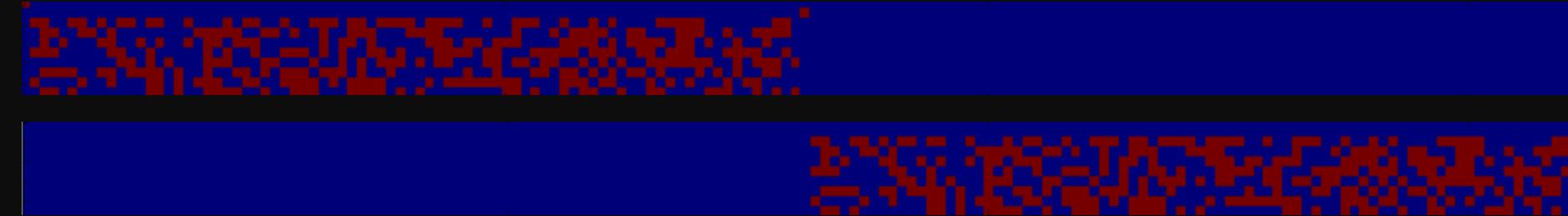


메모리에서 하나씩 읽어서 출력

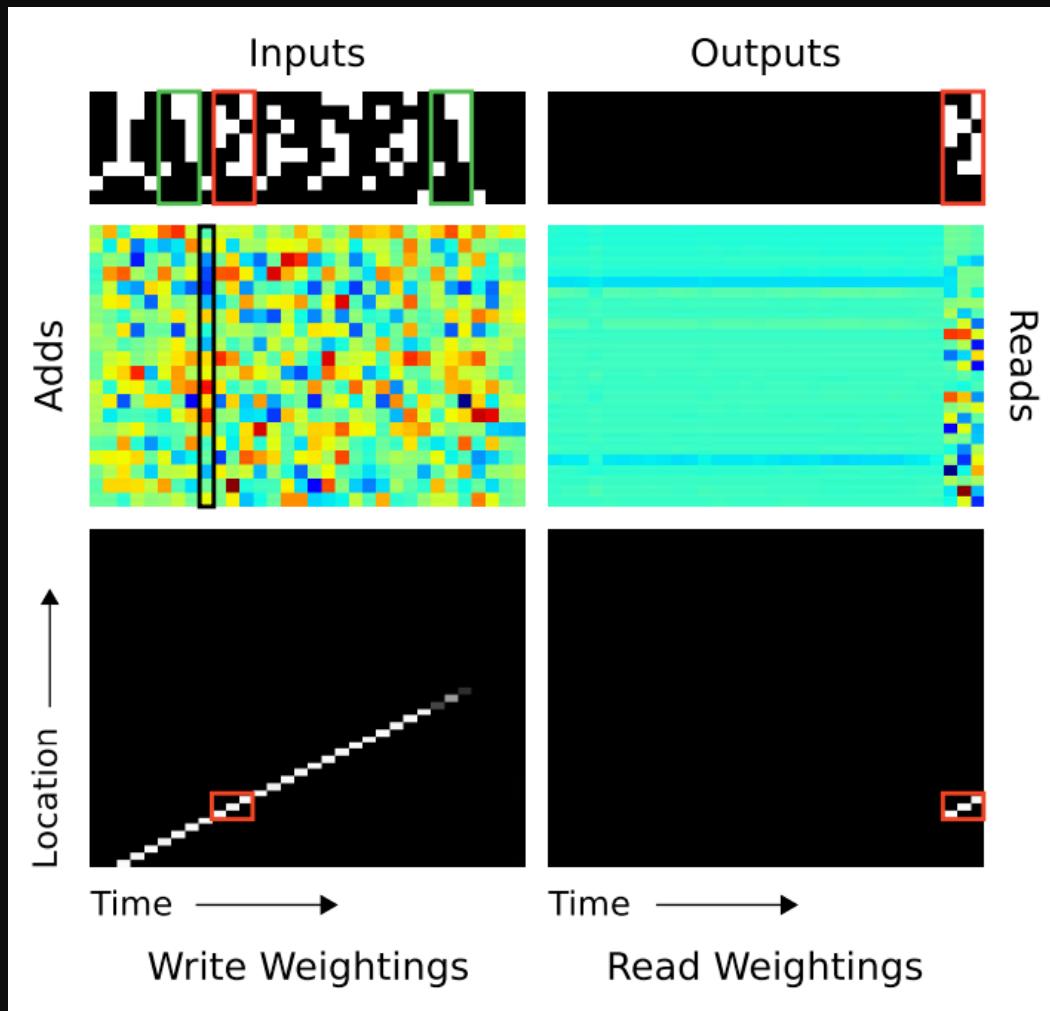


메모리에서 하나씩 읽어서 출력





학습에 보지 않았던 긴 input



<https://github.com/carpedm20/NTM-tensorflow>

Resources

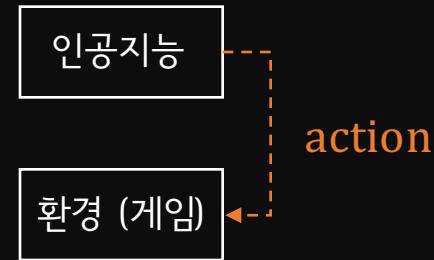
- [Neural Turing Machine \(in Chinese\)](#)
- [Fancy Addressing for Neural Turing Machines](#)
- [Neural Turing Machines implementation](#)
- [Neural Turing Machines - A First Look](#)

DEVSISTERS

Deep Q Learning



Deep Q Learning ∈ Reinforcement Learning



Deep Q Learning ∈ Reinforcement Learning



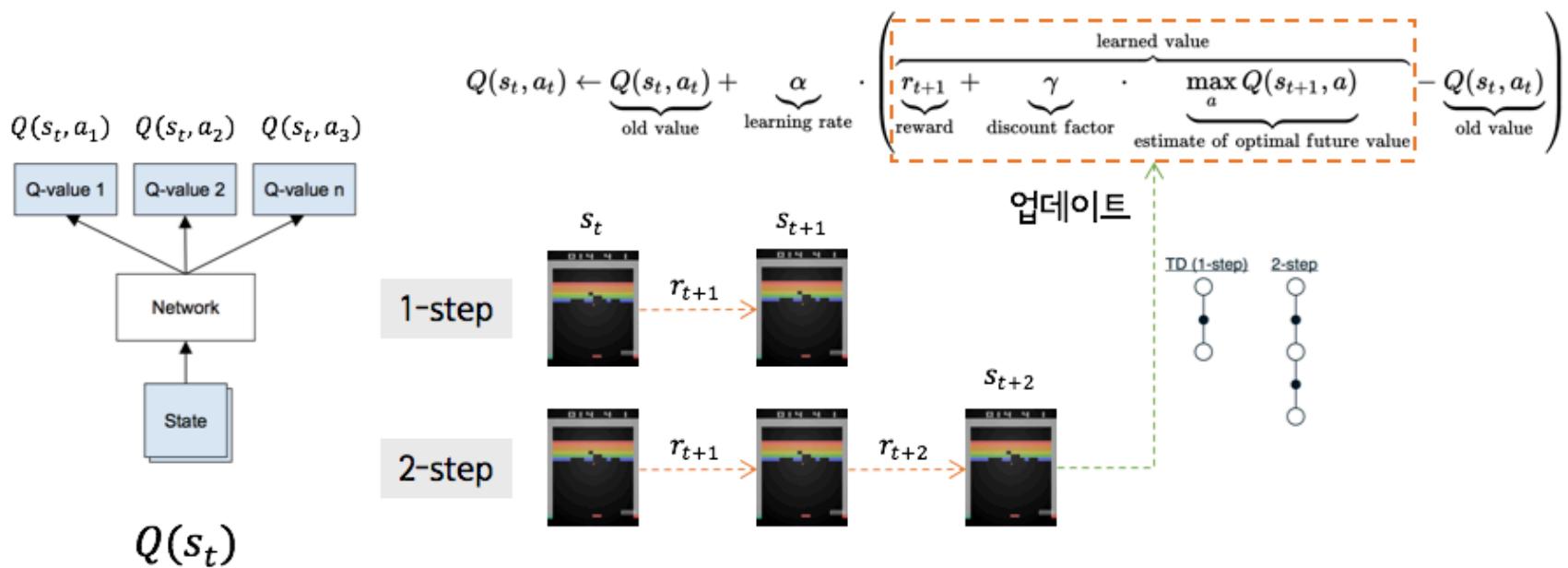
Deep Q Learning ∈ Reinforcement Learning

“AI = Reinforcement Learning + Deep Learning”

- David Silver



n-step Q-learning



See [David Silver's course](#) & [Sutton's book](#) for more information

더 자세한건...

텐서플로우 설치도 했고
튜토리얼도 봤고
기초 예제도 짜봤다면

김태훈
carpedm20

<http://www.slideshare.net/carpedm20/ss-63116251>

더 자세한건...

Reinforcement Learning: An Introduction

Richard S. Sutton and Andrew G. Barto

김태훈

<http://www.slideshare.net/carpedm20/reinforcement-learning-an-introduction-64037079>

<https://github.com/carpedm20/deep-rl-tensorflow>

<https://github.com/devsisters/DQN-tensorflow/>

Resources

- David Silver's ICML “Reinforcement Learning tutorial”
- Demystifying Deep Reinforcement Learning
- Playing Atari With Deep Reinforcement Learning

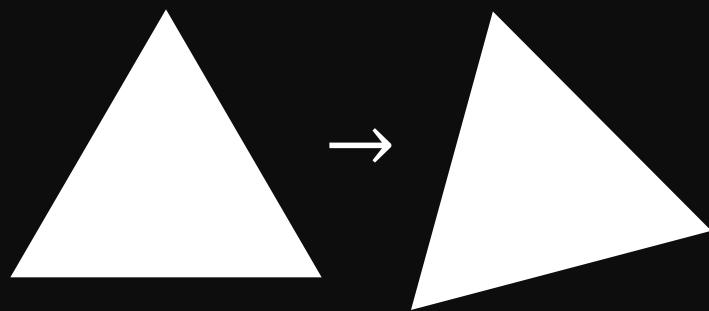
DEVSISTERS

Deep Visual Analogy-Making

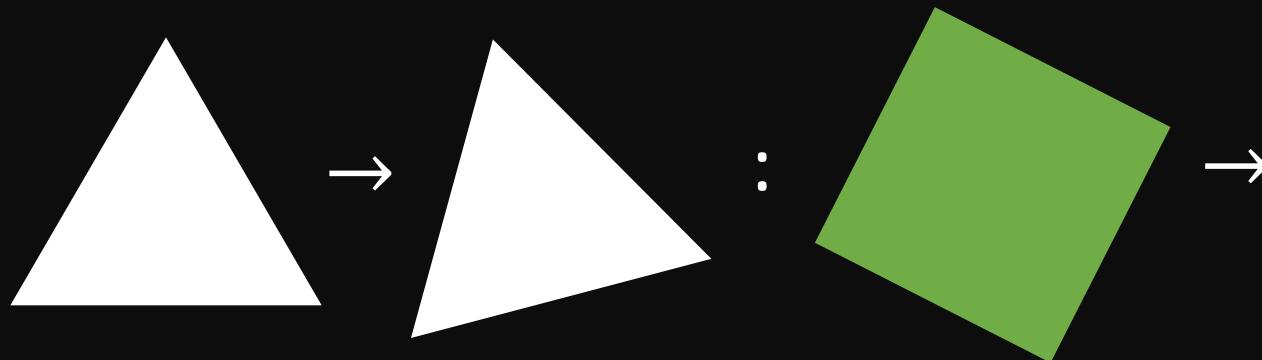
Deep Visual Analogy-Making

이미지 버전의 '왕 - 남자 + 여자 = 여왕' (word2vec)

A → B

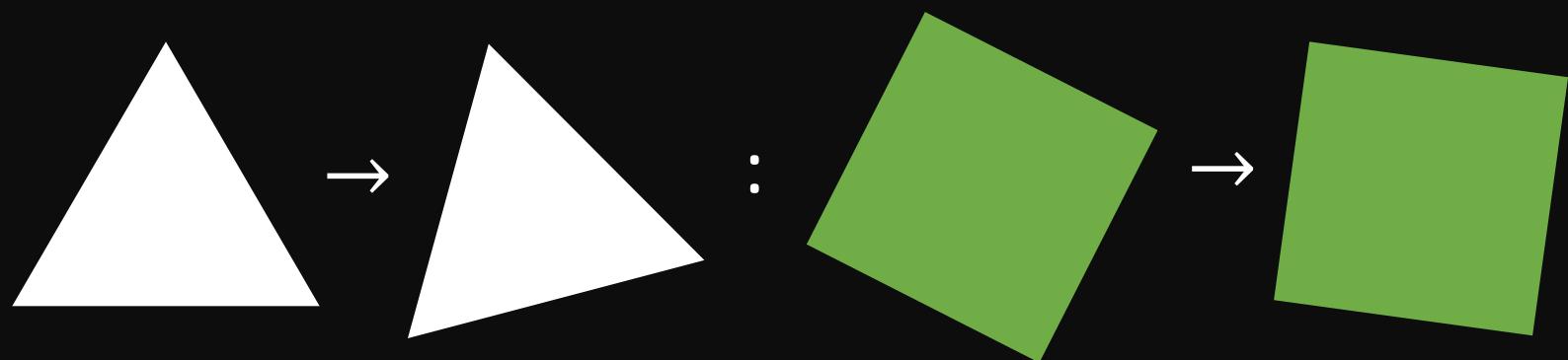


A → B : C → ?



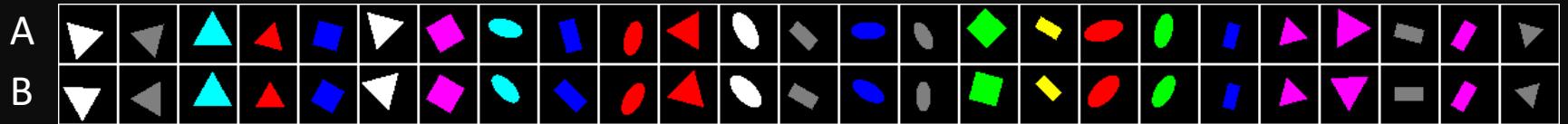
A에서 B의 변화를 C에 적용하면?

A → B : C → ?

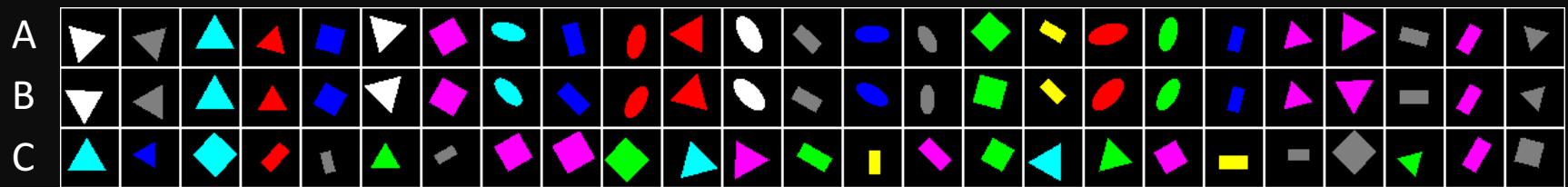


A에서 B의 변화를 C에 적용하면?

A → B : C → ?

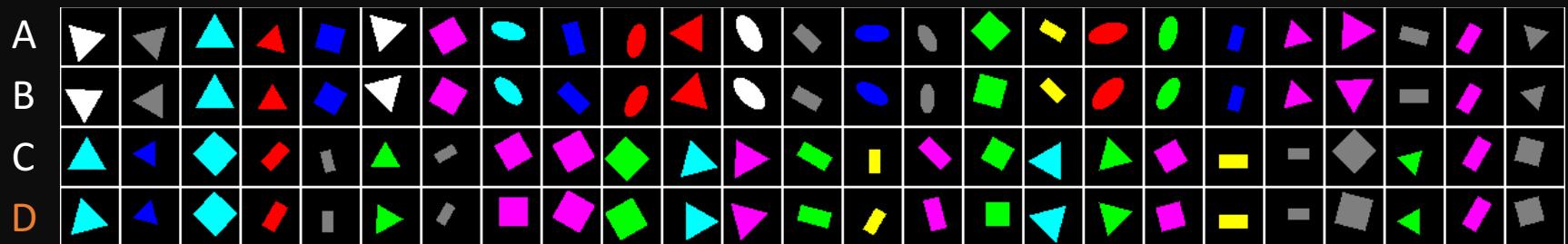


A → B : C → ?



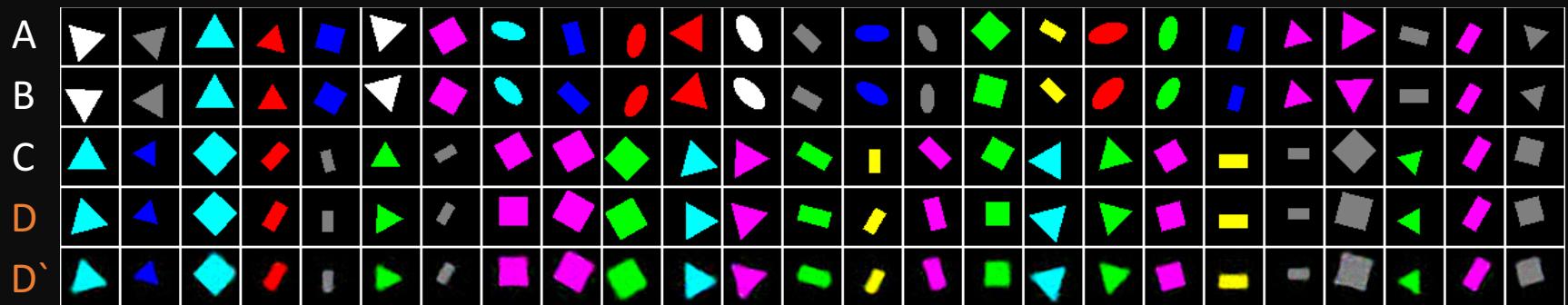
회전한 이미지를 보고

A → B : C → ?



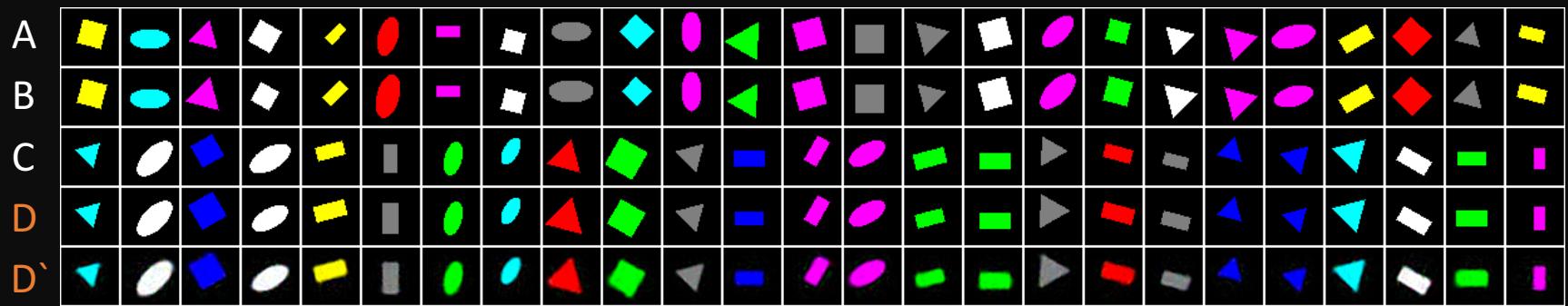
회전 1번

A → B : C → ?



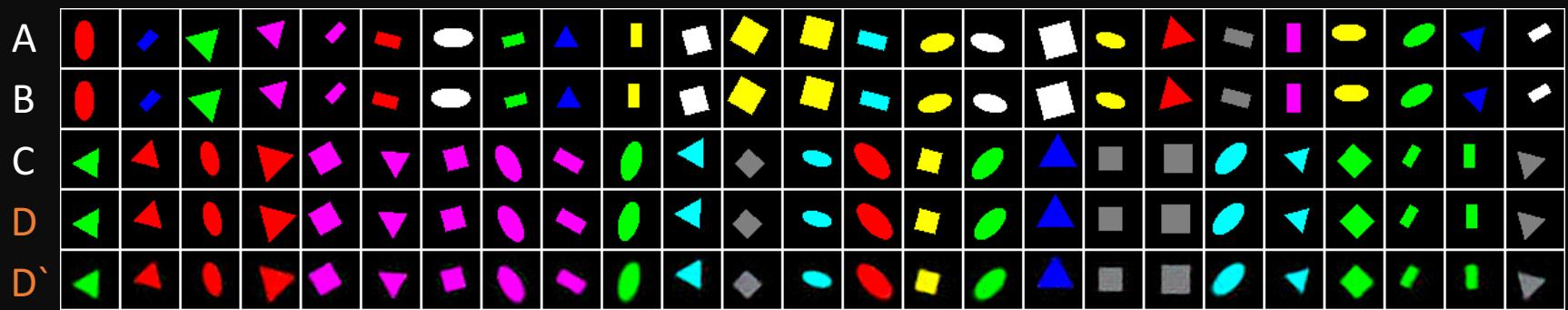
회전 2번

A → B : C → ?



확대, 축소

A → B : C → ?



x, y 축 이동

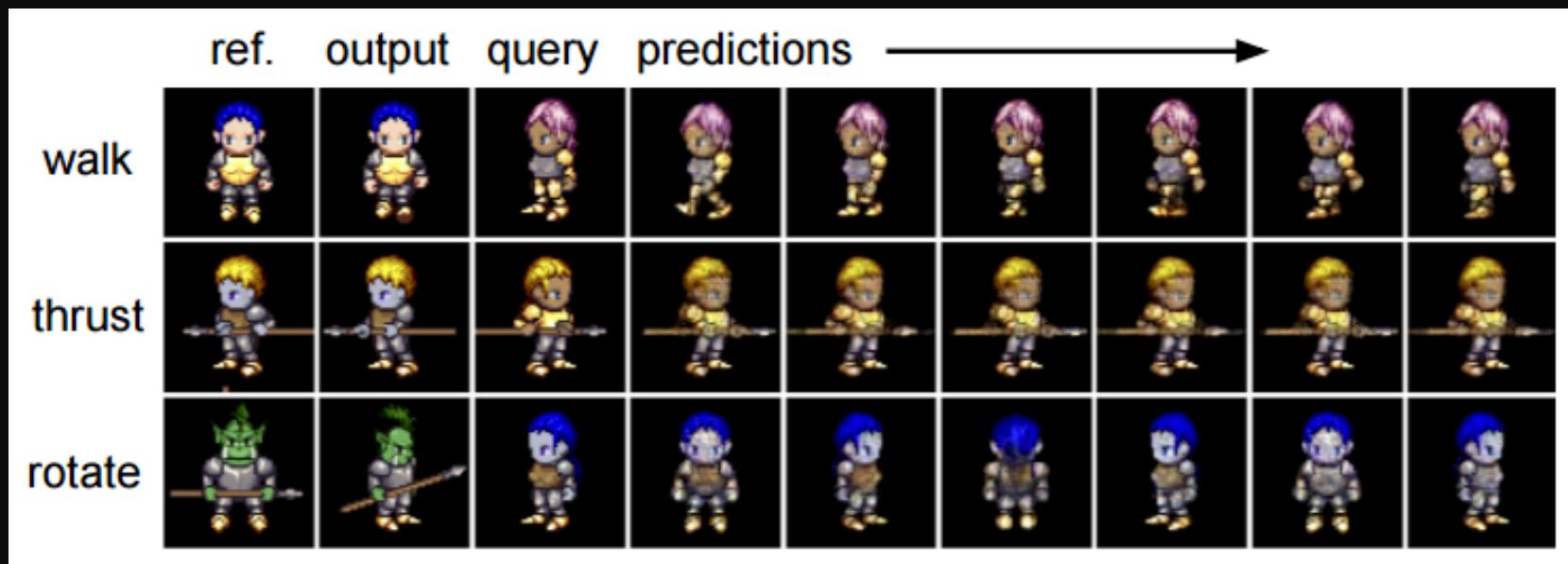
A → B : C → ?

	ref.	output
walk		
thrust		
rotate		

A → B : C → ?

	ref.	output	query
walk			
thrust			
rotate			

A → B : C → ?



<https://github.com/carpedm20/visual-analogy-tensorflow>

DEVSISTERS

지금까지 제가 구현한 논문은

- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- Deep Visual Analogy-Making
- Pixel Recurrent Neural Networks

Computer Vision

지금까지 제가 구현한 논문은

- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
 - Deep Visual Analogy-Making
 - Pixel Recurrent Neural Networks
-
- End-To-End Memory Networks
 - Neural Variational Inference for Text Processing
 - Character-Aware Neural Language Models
 - Teaching Machines to Read and Comprehend
- Computer Vision
- Natural Language Processing

지금까지 제가 구현한 논문은

- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
 - Deep Visual Analogy-Making
 - Pixel Recurrent Neural Networks
-
- End-To-End Memory Networks
 - Neural Variational Inference for Text Processing
 - Character-Aware Neural Language Models
 - Teaching Machines to Read and Comprehend
-
- Playing Atari with Deep Reinforcement Learning
 - Human-level control through deep reinforcement learning
 - Deep Reinforcement Learning with Double Q-learning
 - Dueling Network Architectures for Deep Reinforcement Learning
 - Language Understanding for Text-based Games Using Deep Reinforcement Learning
 - Asynchronous Methods for Deep Reinforcement Learning
 - Continuous Deep q-Learning with Model-based Acceleration
-
- Computer Vision
 - Natural Language Processing
 - Reinforcement Learning

지금까지 제가 구현한 논문은

- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
 - Deep Visual Analogy-Making
 - Pixel Recurrent Neural Networks
-
- End-To-End Memory Networks
 - Neural Variational Inference for Text Processing
 - Character-Aware Neural Language Models
 - Teaching Machines to Read and Comprehend
-
- Playing Atari with Deep Reinforcement Learning
 - Human-level control through deep reinforcement learning
 - Deep Reinforcement Learning with Double Q-learning
 - Dueling Network Architectures for Deep Reinforcement Learning
 - Language Understanding for Text-based Games Using Deep Reinforcement Learning
 - Asynchronous Methods for Deep Reinforcement Learning
 - Continuous Deep q-Learning with Model-based Acceleration
-
- Neural Turing Machines
- Computer Vision
- Natural Language Processing
- Reinforcement Learning
- Algorithm Learning

재미있었던 논문은

- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
 - Deep Visual Analogy-Making
 - Pixel Recurrent Neural Networks
- Computer Vision
-
- End-To-End Memory Networks
 - Neural Variational Inference for Text Processing
 - Character-Aware Neural Language Models
 - Teaching Machines to Read and Comprehend
- Natural Language Processing
-
- Playing Atari with Deep Reinforcement Learning
 - Human-level control through deep reinforcement learning
 - Deep Reinforcement Learning with Double Q-learning
 - Dueling Network Architectures for Deep Reinforcement Learning
 - Language Understanding for Text-based Games Using Deep Reinforcement Learning
 - Asynchronous Methods for Deep Reinforcement Learning
 - Continuous Deep q-Learning with Model-based Acceleration
- Reinforcement Learning
-
- Neural Turing Machines
- Algorithm Learning

각 모델들은

- 이미지(사람의 얼굴 사진)을 이해하고 스스로 만드는 모델
- 이미지 버전의 '왕 - 남자 + 여자 = 여왕'
- 이미지를 이해하고 픽셀 하나씩 만들어가는 모델

Computer Vision

- Question Answering, Language Model
- Neural Variational Inference for Text Processing
- Character-level Language Models with CNN
- Question Answering

Natural Language Processing

- Atari 게임을 화면의 픽셀만 보고 배우는 딥러닝 모델 (최초)
- Atari 게임을 화면의 픽셀만 보고 배우는 딥러닝 모델 (Nature 논문)
- Double Q-Learning로 Atari 게임을 더 잘 플레이하는 모델
- Dueling Network로 Atari 게임을 더 잘 플레이하는 모델
- 텍스트 게임을 강화학습으로 배우는 모델
- Asynchronous 학습으로 빠르게로 Atari 게임을 학습하는 모델
- Atari 게임과 같이 discrete한 action이 아닌 환경을 잘 학습하는 모델

Reinforcement Learning

- 뉴럴 튜링 머신

Algorithm Learning

모든 코드는 Github에

- <https://github.com/carpedm20/DCGAN-tensorflow> ★ 580+
- <https://github.com/carpedm20/visual-analogy-tensorflow> ★ 210+
- <https://github.com/carpedm20/pixel-rnn-tensorflow/> ★ 80+

- <https://github.com/carpedm20/MemN2N-tensorflow> ★ 180+
- <https://github.com/carpedm20/variational-text-tensorflow> ★ 330+
- <https://github.com/carpedm20/lstm-char-cnn-tensorflow> ★ 170+
- <https://github.com/carpedm20/attentive-reader-tensorflow> ★ 100+

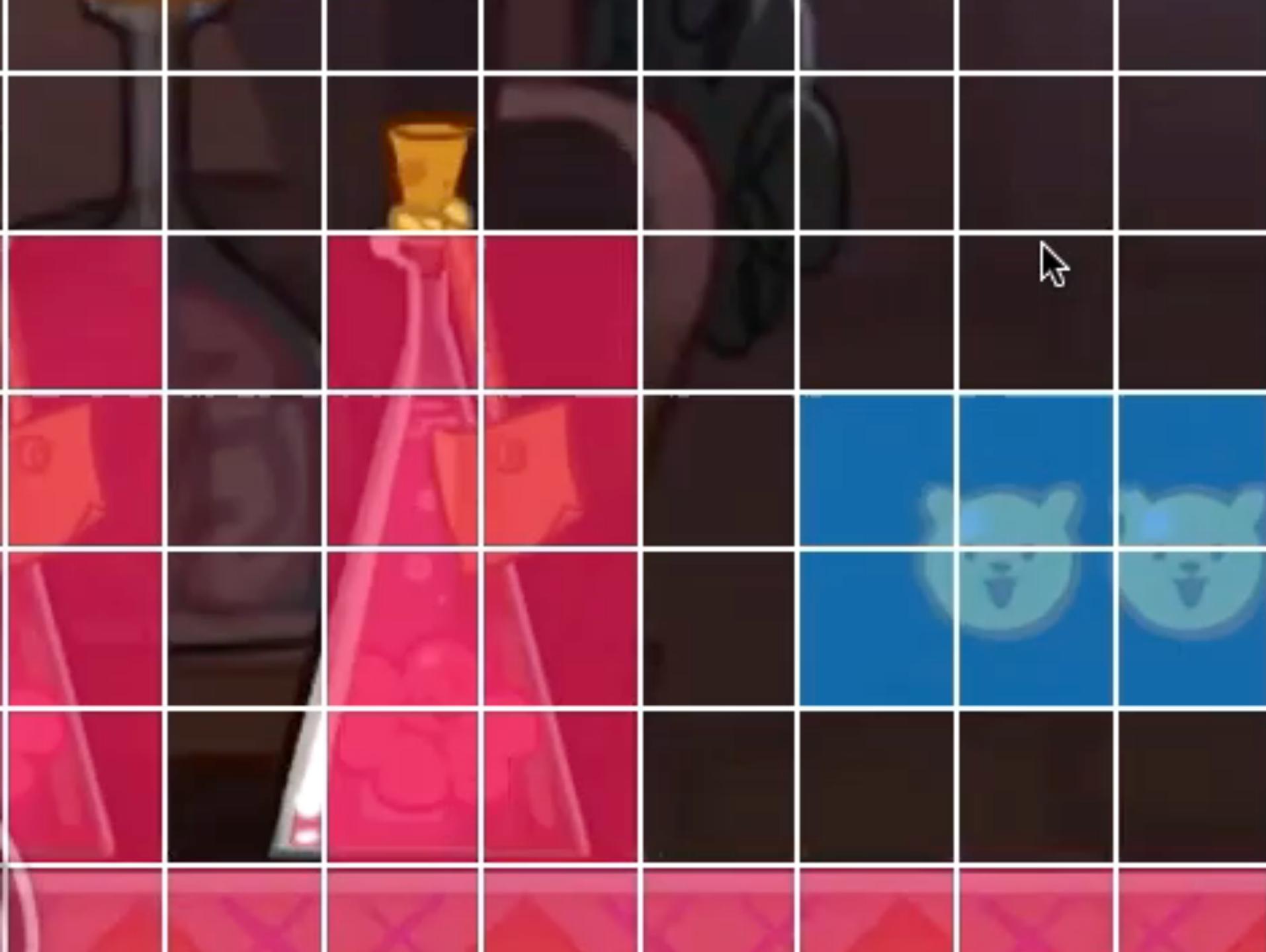
- <https://github.com/devsisters/DQN-tensorflow> ★ 410+
- <https://github.com/devsisters/DQN-tensorflow> ★ 410+
- <https://github.com/carpedm20/deep-rl-tensorflow> ★ 370+
- <https://github.com/carpedm20/deep-rl-tensorflow> ★ 370+
- <https://github.com/carpedm20/text-based-game-rl-tensorflow> ★ 20+
- <https://github.com/devsisters/async-rl-tensorflow> ★ 40+

- <https://github.com/carpedm20/NAF-tensorflow/> ★ 330+

DEVSISTERS

마지막으로..





DEVSISTERS 로 오세요 :)

<http://www.devsisters.com/jobs/>

감사합니다

<http://carpedm20.github.io/>