

DCGAN

BabelSpeech 파트3 - 4회차

2018.05.14

anson@deephnatural.io

[**https://github.com/KaggleBreak/babelspeech**](https://github.com/KaggleBreak/babelspeech)

Generative Model

- 1. Intro to GANs**
- 2. Generative Modeling Review**
- 3. Variational Autoencoders**
- 4. DCGAN ***

Udacity DCGAN Lesson

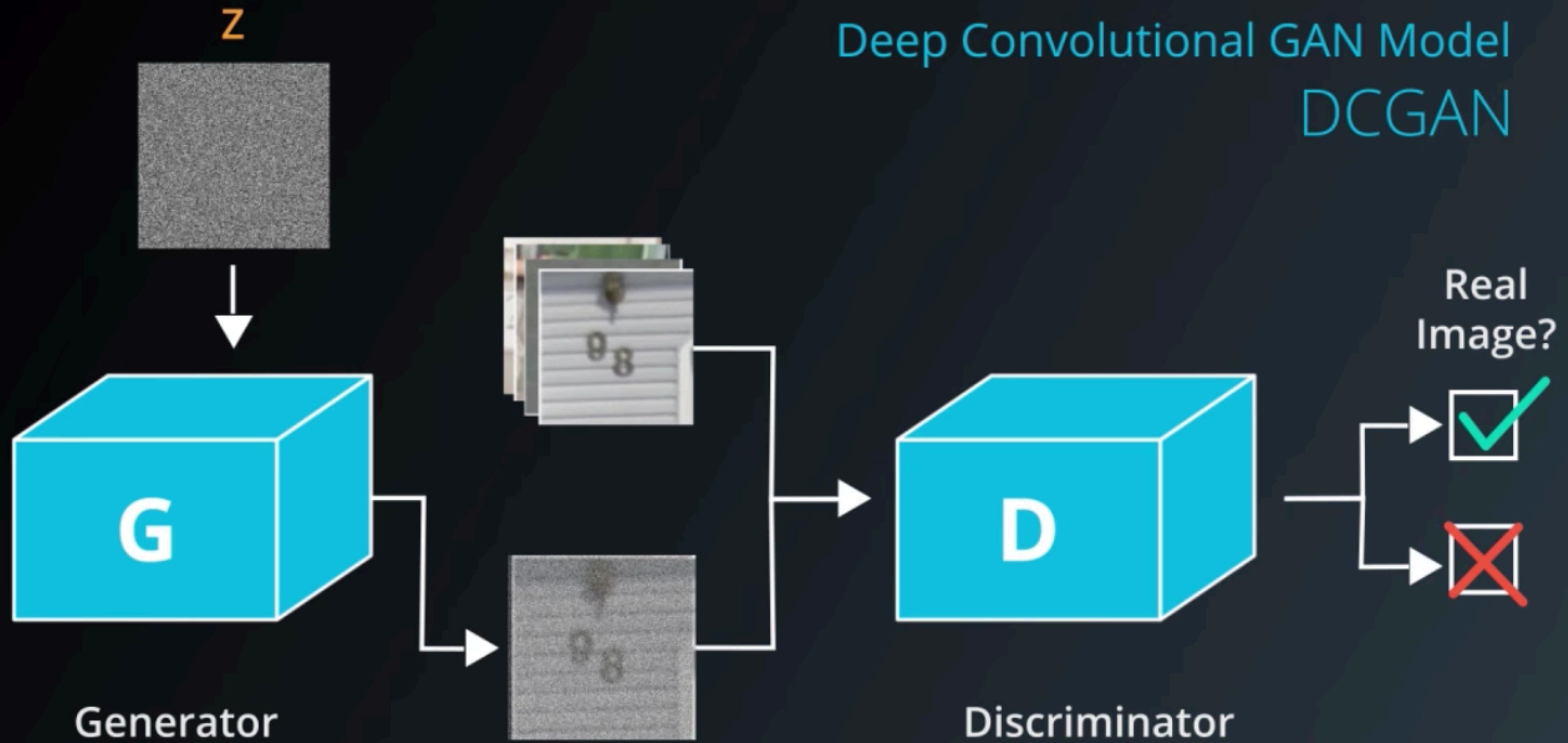
<https://www.nodalpoint.com/gans-udacity/>

Paper

Unsupervised Representation Learning with Deep
Convolutional Generative Adversarial Networks

<https://arxiv.org/pdf/1511.06434.pdf>

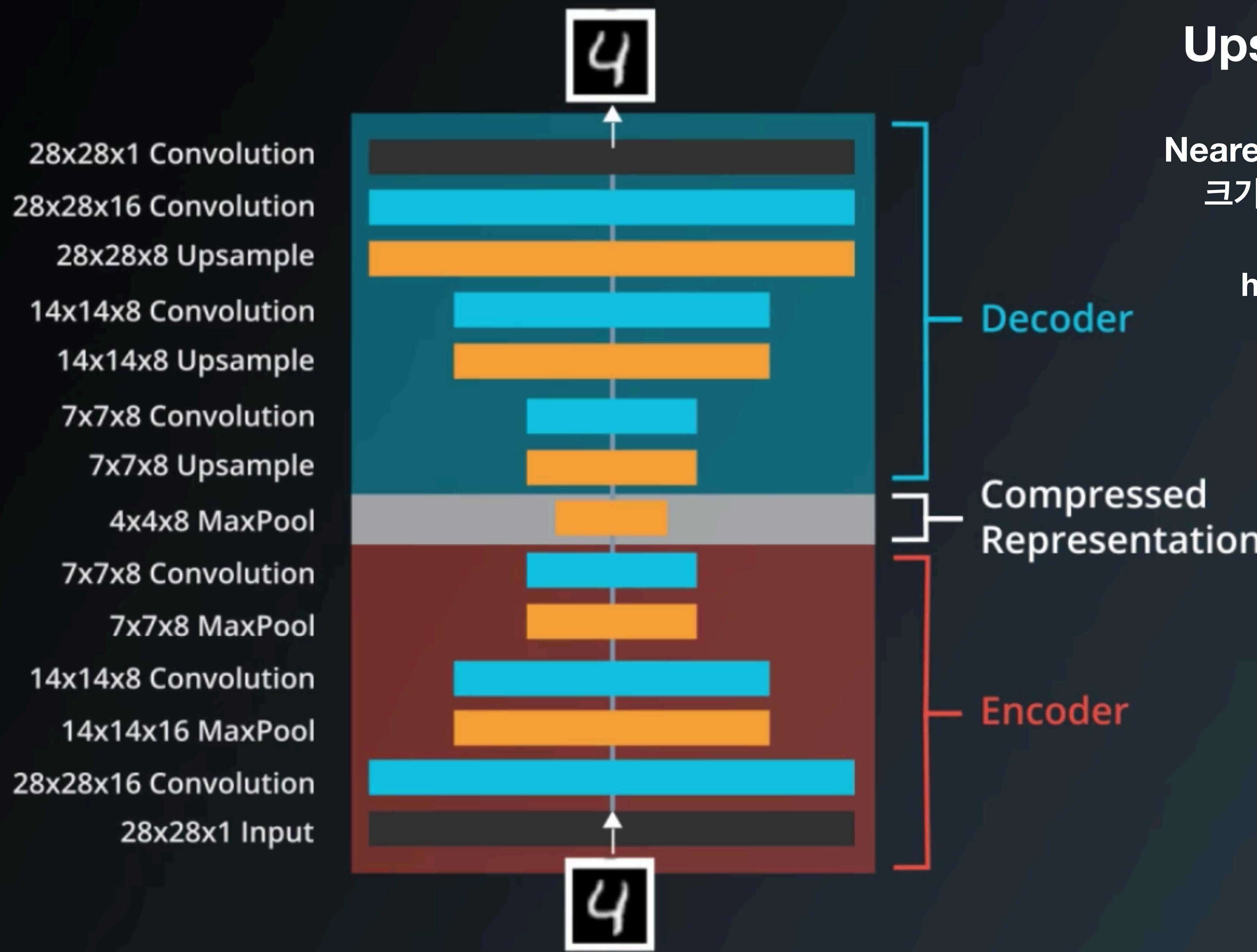
Deep Convolutional GAN Model DCGAN



Deep Convolutional Network를 이용해서 Generator와 Discriminator를 구성.

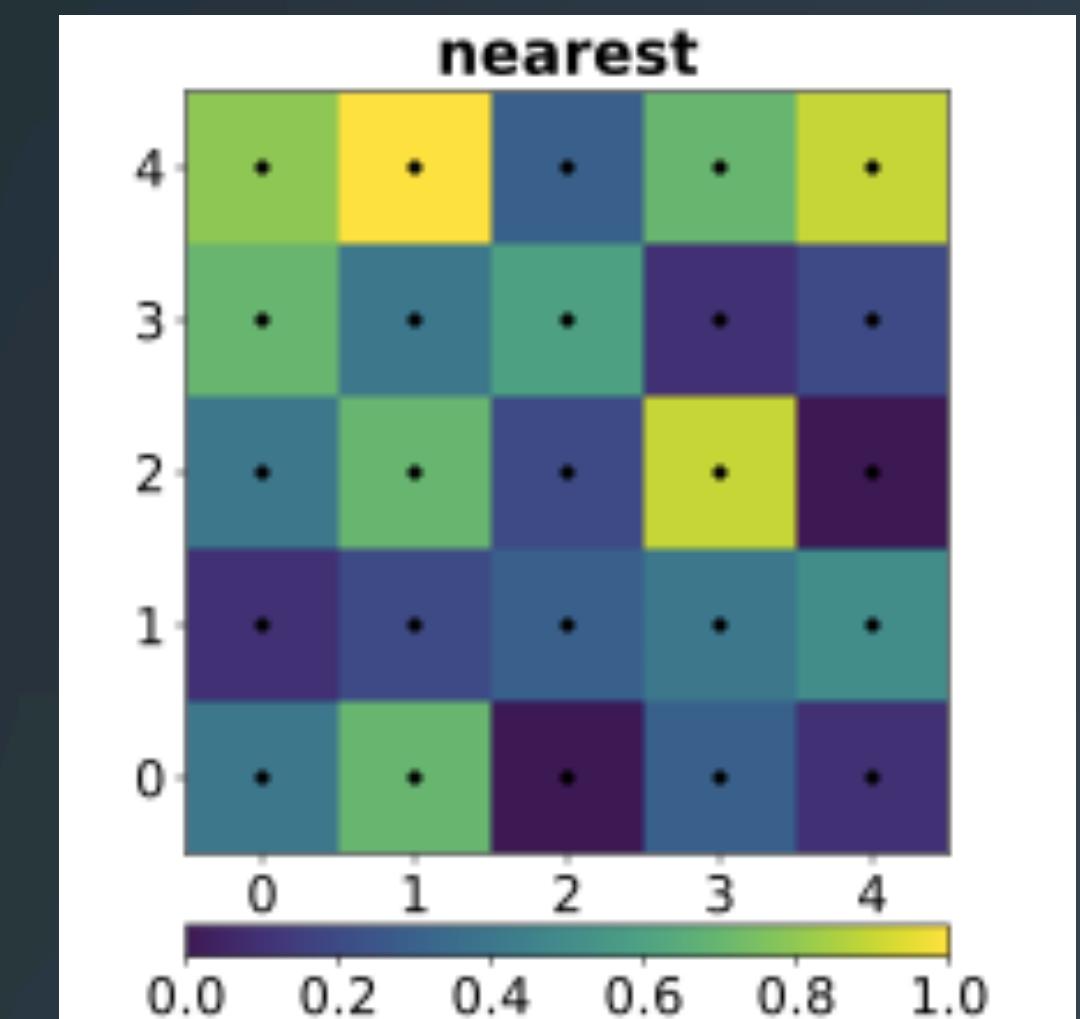
나머지는 GAN과 동일

Upsampling in Autoencoder

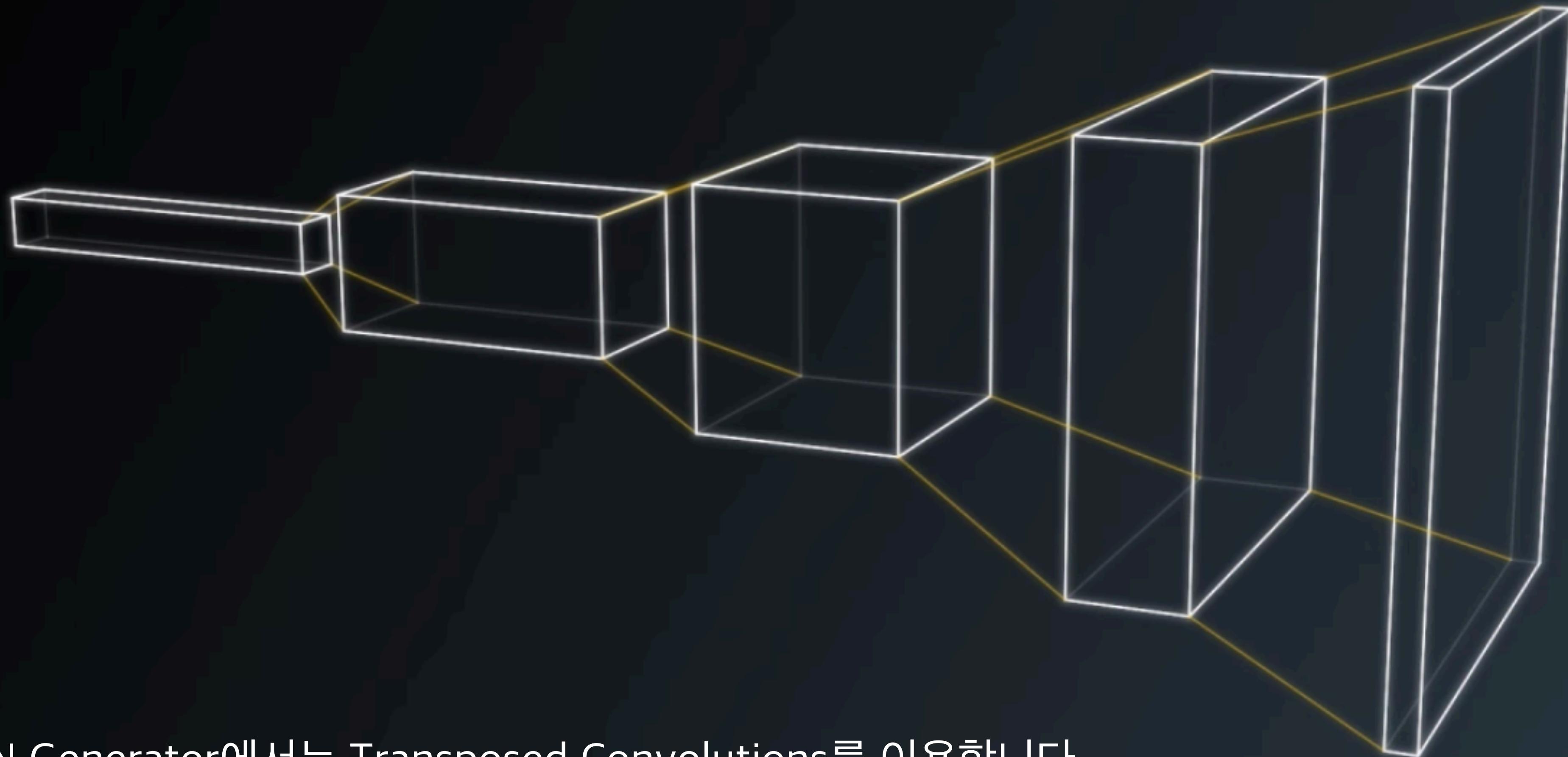


Nearest-neighbor interpolation으로 레이어 크기를 키워서 Convolutional Layer로 보냄

https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation

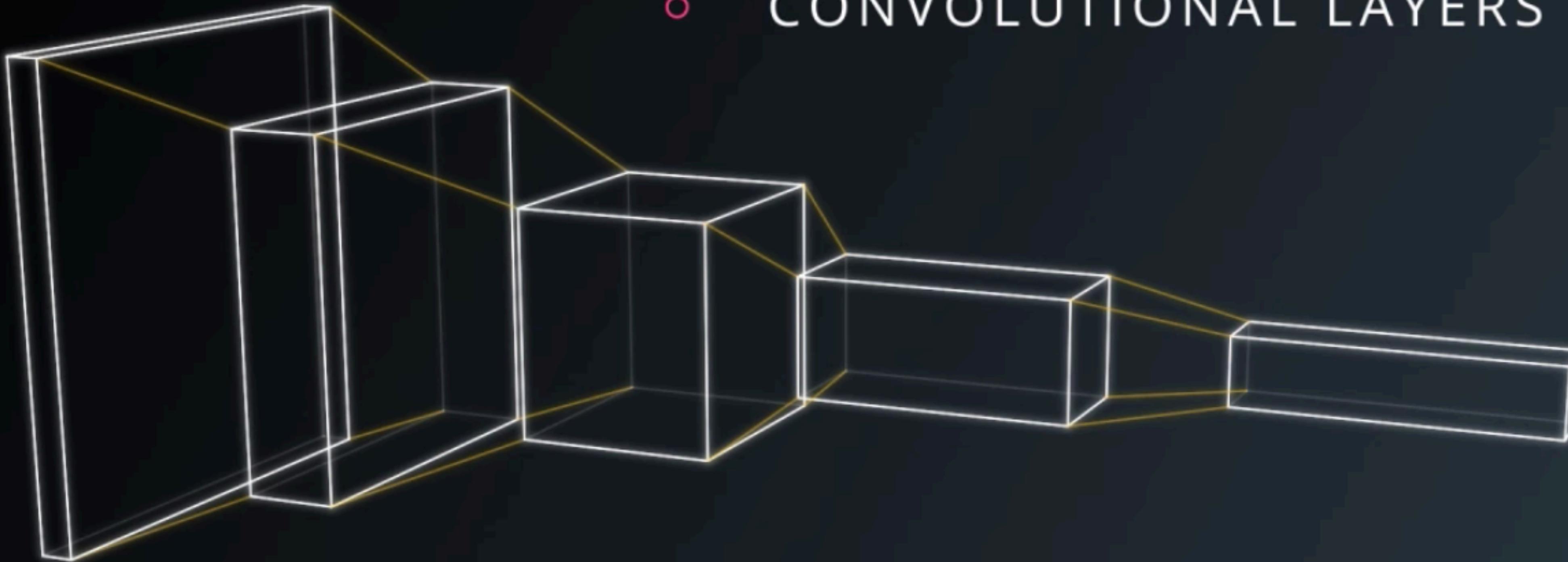


- TRANSPOSED CONVOLUTIONS



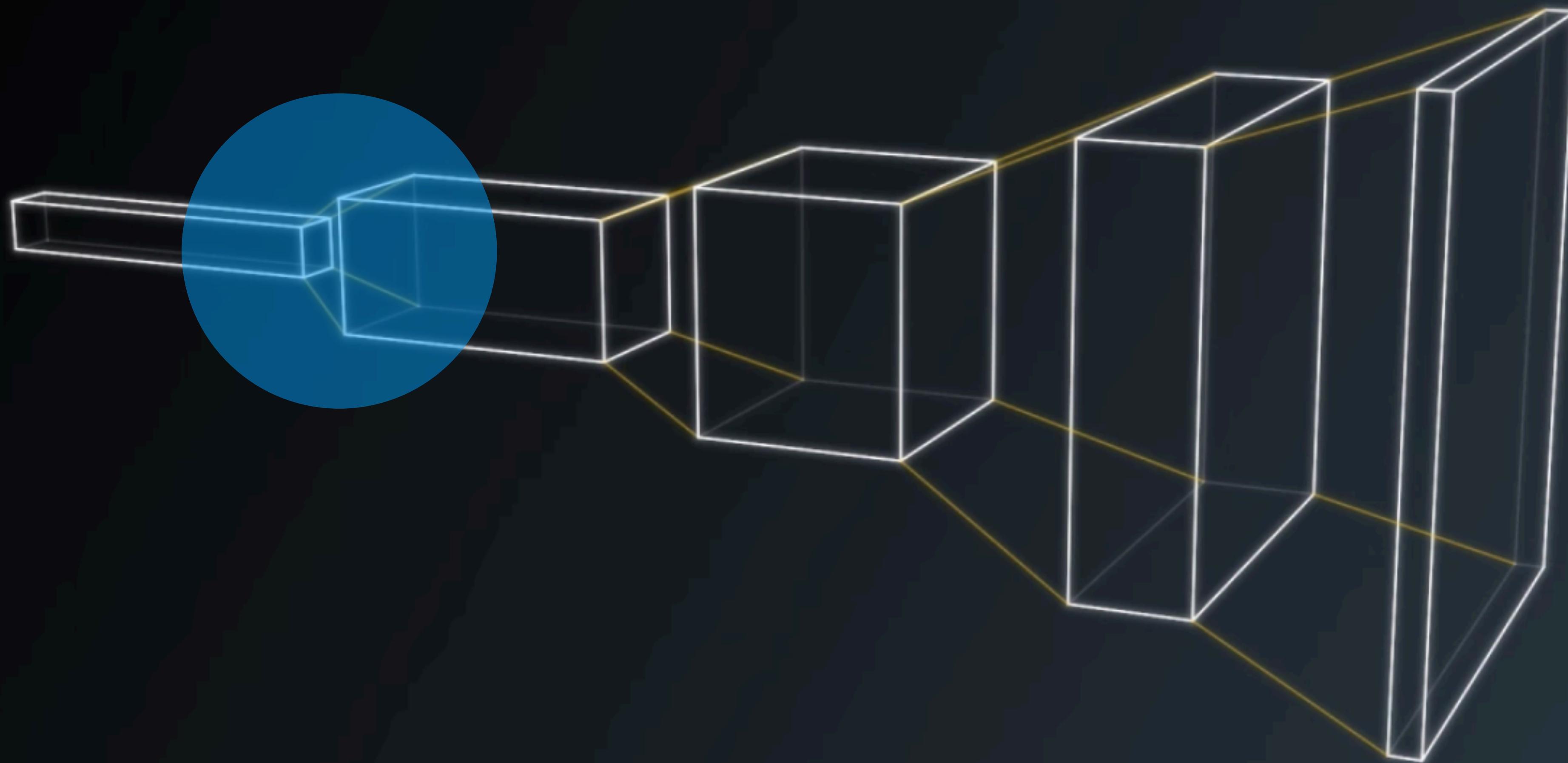
DCGAN Generator에서는 Transposed Convolutions를 이용합니다.
우리가 잘 알고 있는 Convolution과 같지만 반대로 뒤집혀 있는 구조입니다.

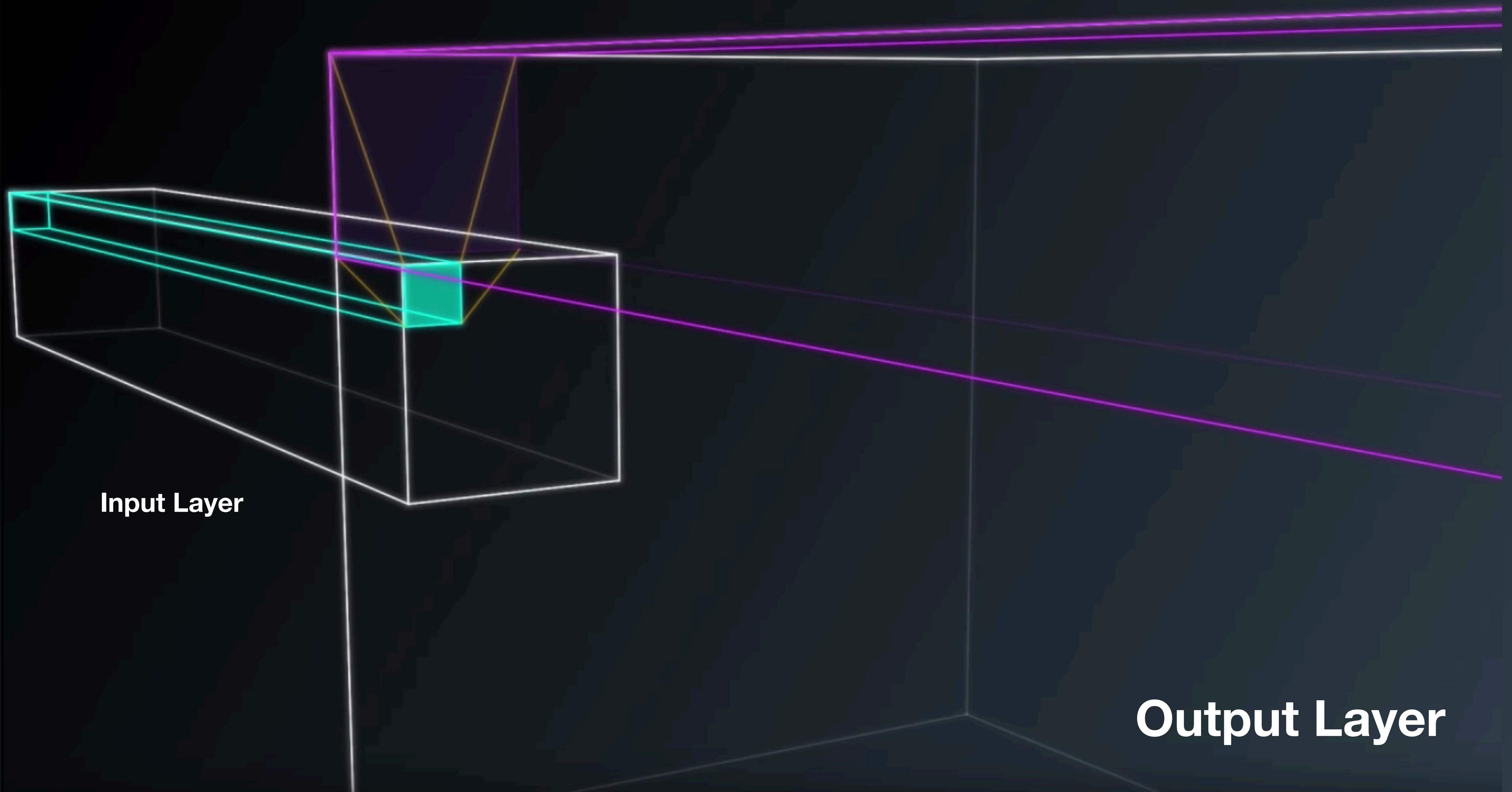
◦ CONVOLUTIONAL LAYERS

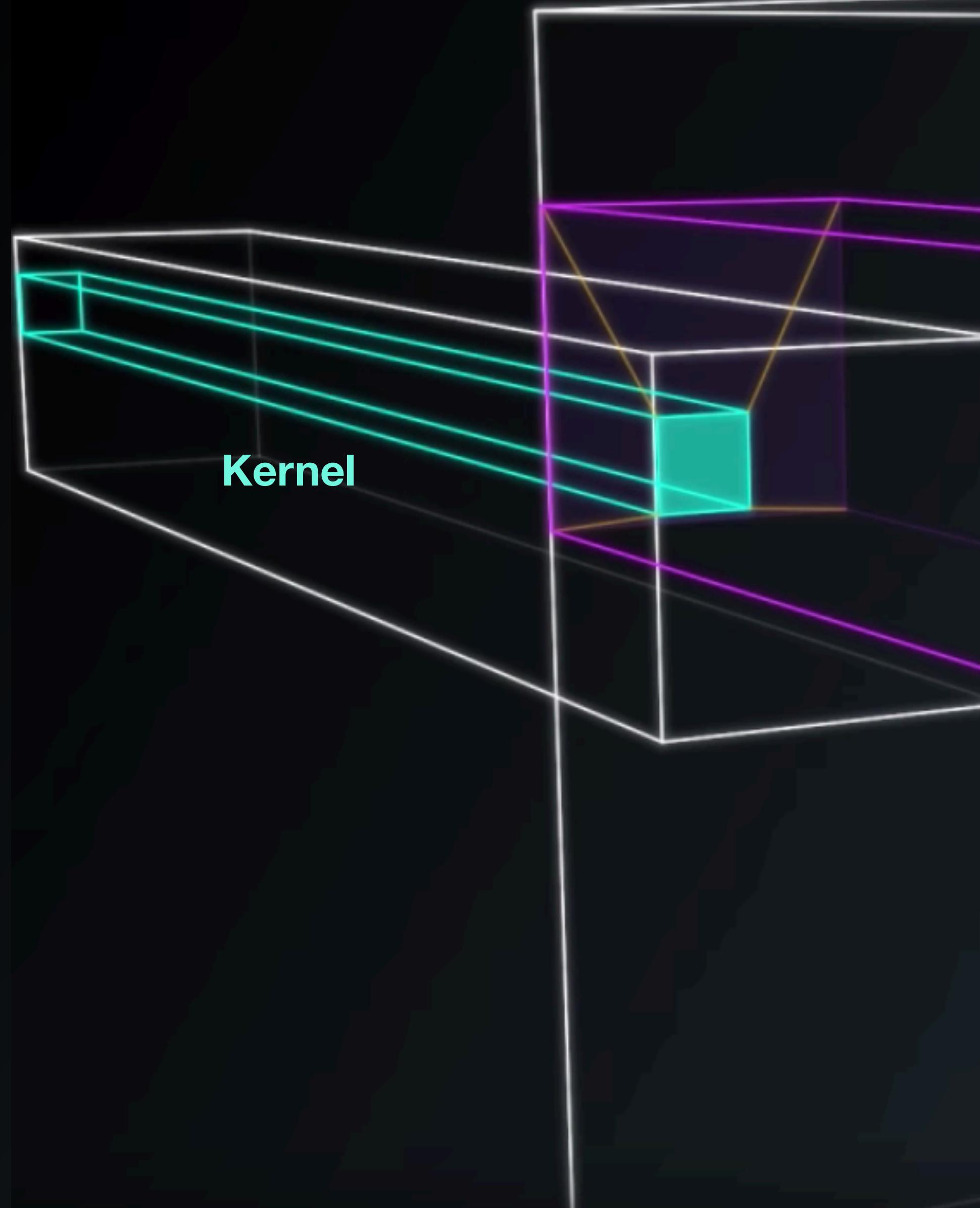


보통 Convolutional Layer는
얇고 넓은(shallow & wide) 레이어부터 시작해서 깊고 좁은(deep & narrow)
레이어로 진행됩니다.

- TRANPOSED CONVOLUTIONS







Patch

Kernel

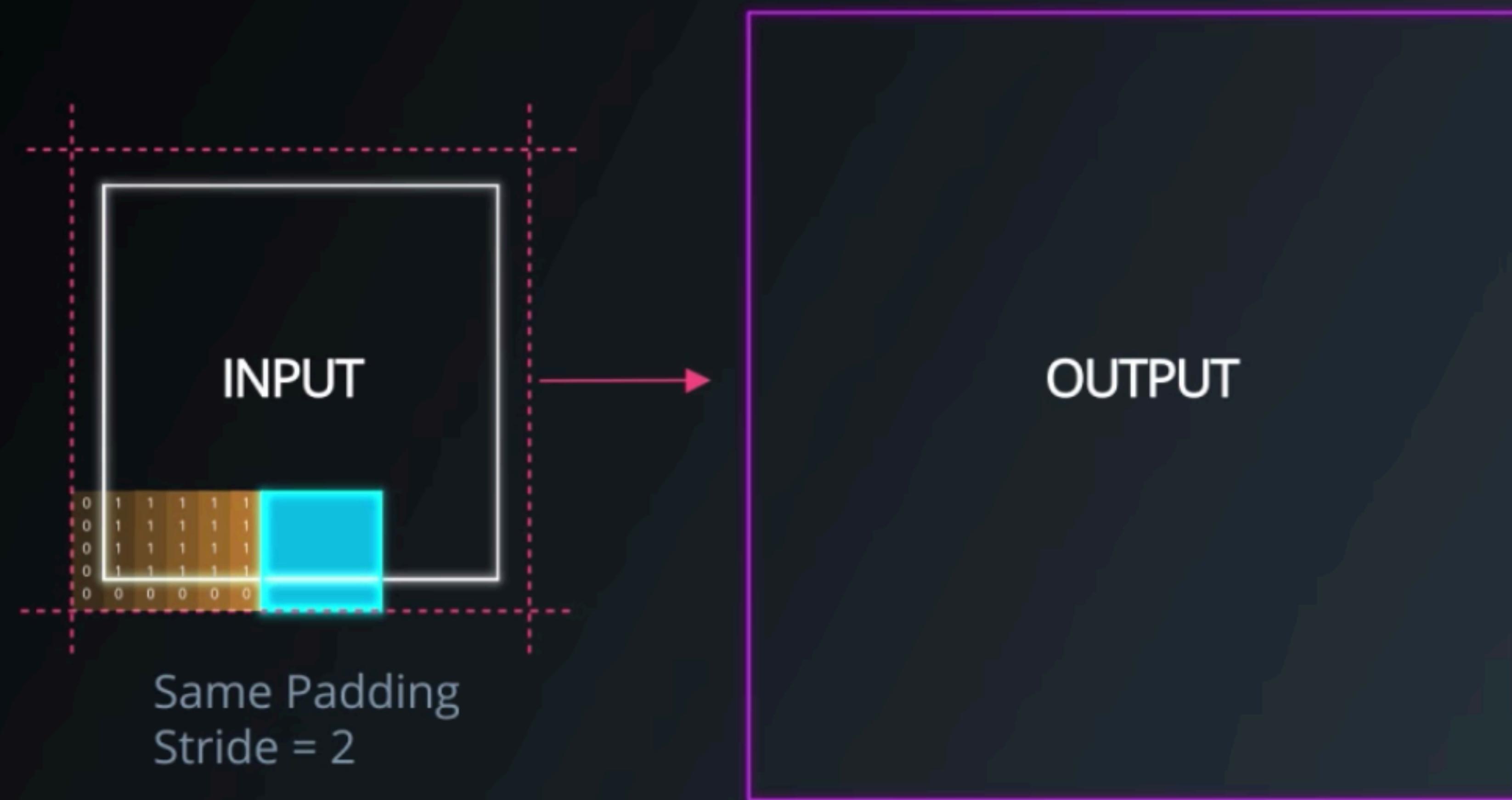
Upsampling

Stride = 2를 통해서
Input Layer의 kernel을 1 pixel 만큼 움직이고
Output Layer의 patch는 2 pixel 만큼 이동시킵니다.

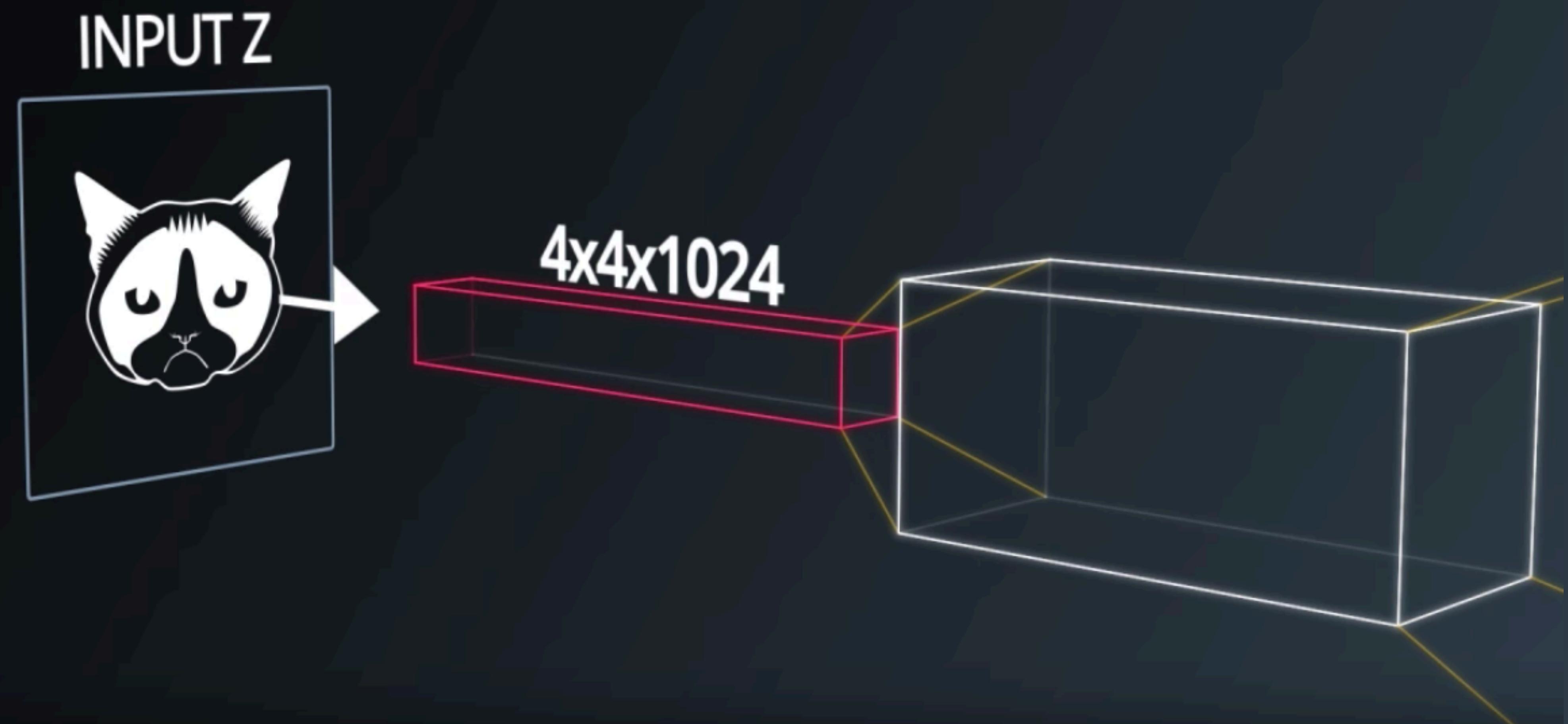
새로운 레이어의 크기는 Stride에 의해 결정됩니다.

Same Padding과 Stride = 2를 적용하면

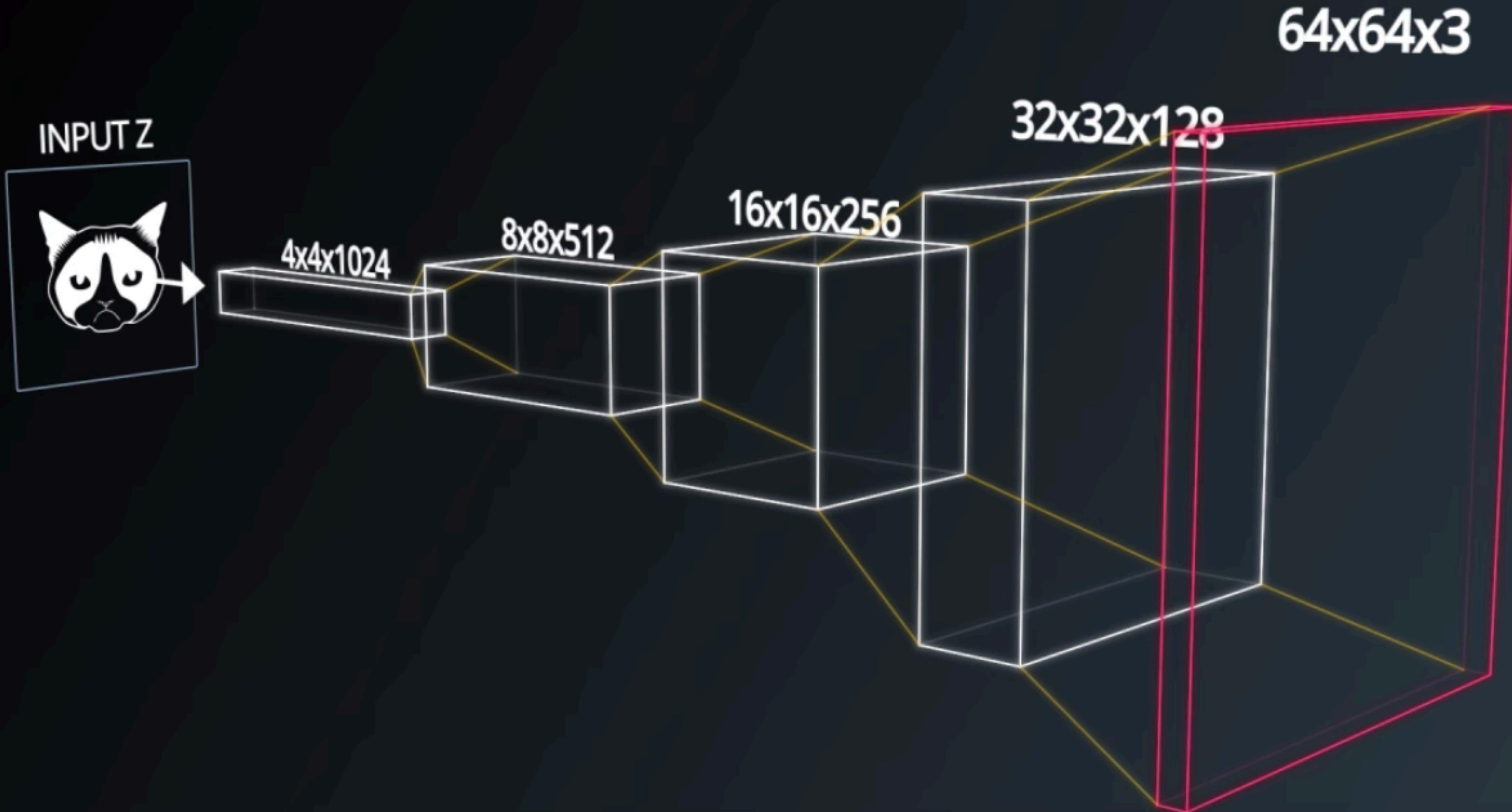
transposed convolutional layer는 input layer의 2배의 width, height을 가지게 됩니다.



DCGAN Generator의 첫번째 스텝은 input noise vector z 를 fully connected layer에 연결시키는 겁니다.
그리고는 fully connected layer를 $4 \times 4 \times 1024$ 로 shape을 변형시켜 줍니다.



이후 Transpose Convolution을 이용한 upsampling 레이어들을 차곡차곡 쌓아갑니다.
각각의 레이어는 Stride = 2를 통해 사이즈가 두배씩 증가하고 depth는 절반씩 감소하게 구성됩니다.
Generator의 마지막 레이어는 64x64x3 Convolutional Layer가 됩니다.

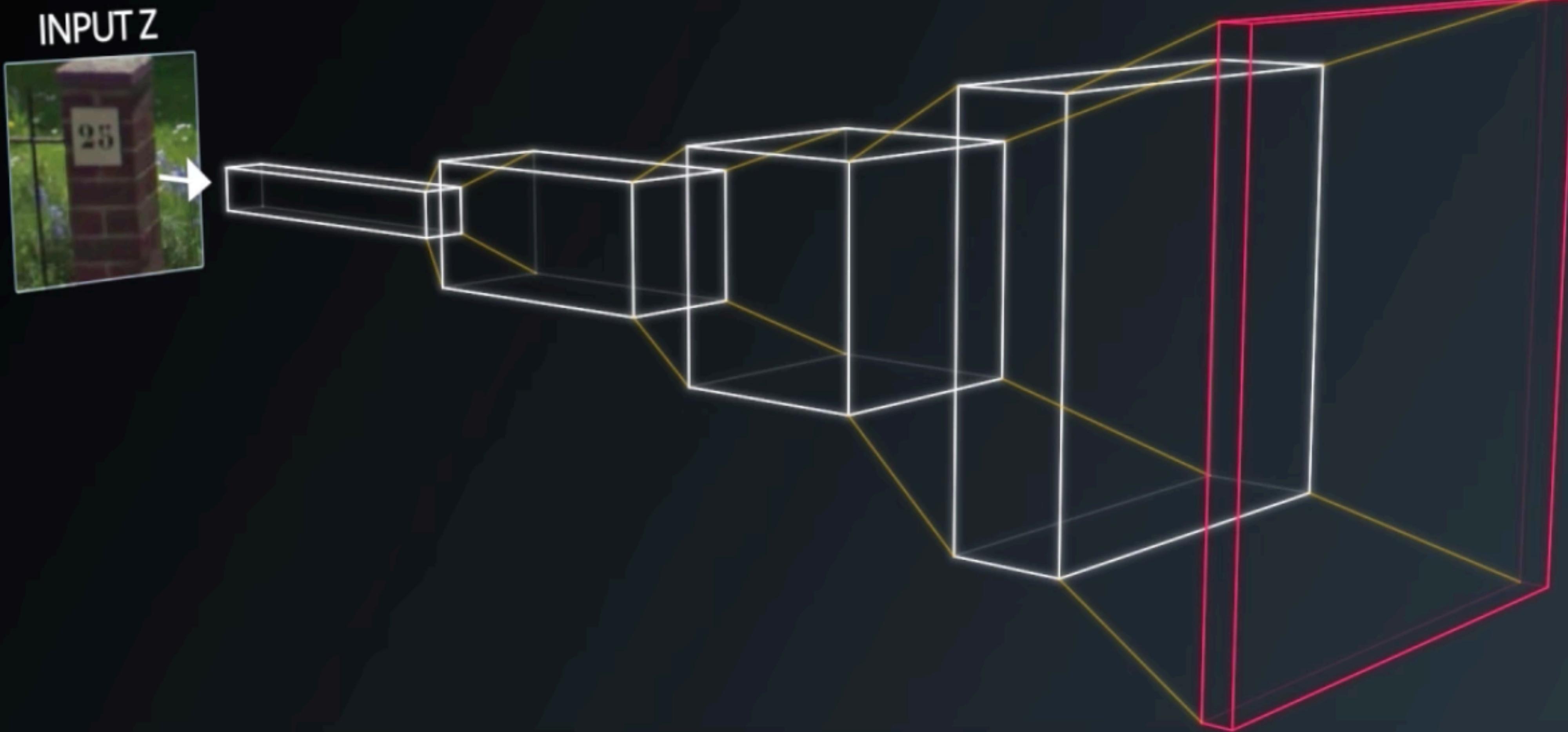


이 사이즈는 실제 이미지 사이즈와 동일해야죠.

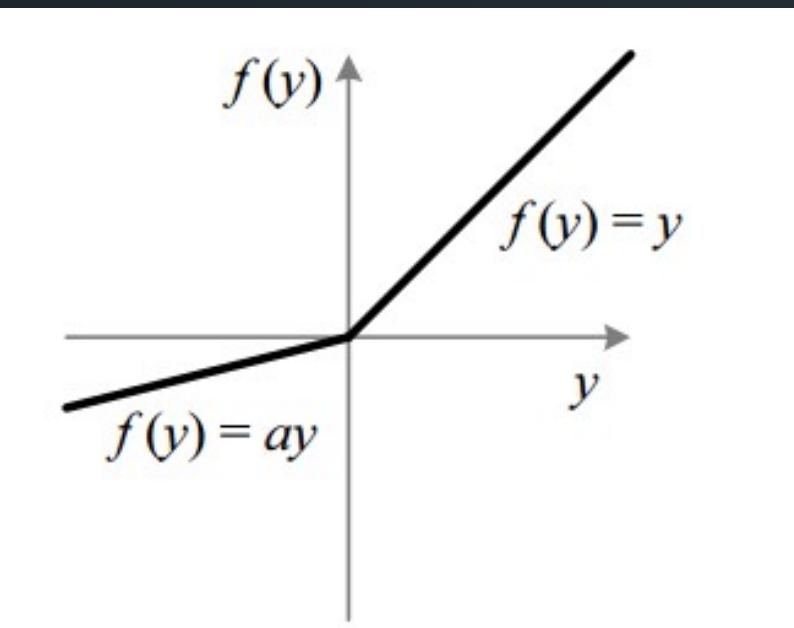
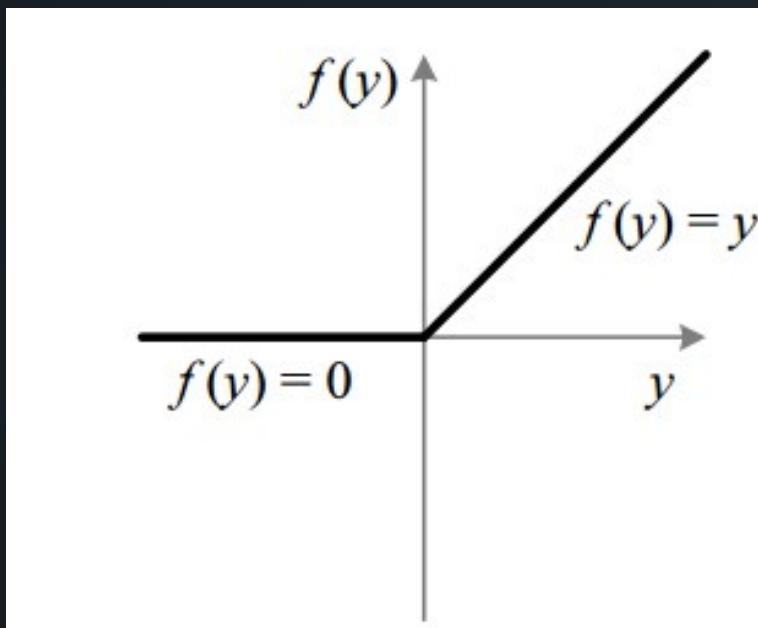
우리는 지금 Street View House Number 이미지를 이용하니 $32 \times 32 \times 3$ 가 되어야 합니다.

여기서 3은 세가지 컬러 채널 R, G, B를 위한 것입니다.

$32 \times 32 \times 3$

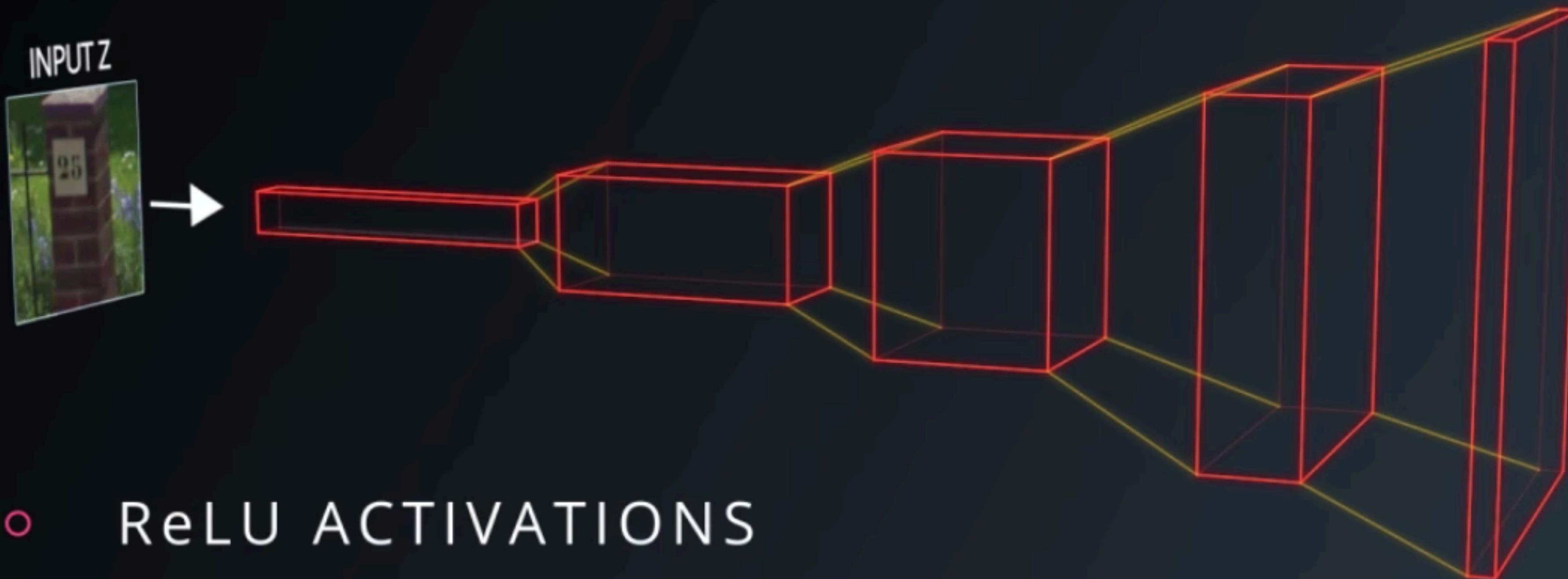


많이 보아왔던 Convolutional Network와 다르게
max pool과 fully connected layer 없이 convolution만 적용.
그리고 각각의 layer에 Leaky ReLU Activation을 이용합니다.



ReLU

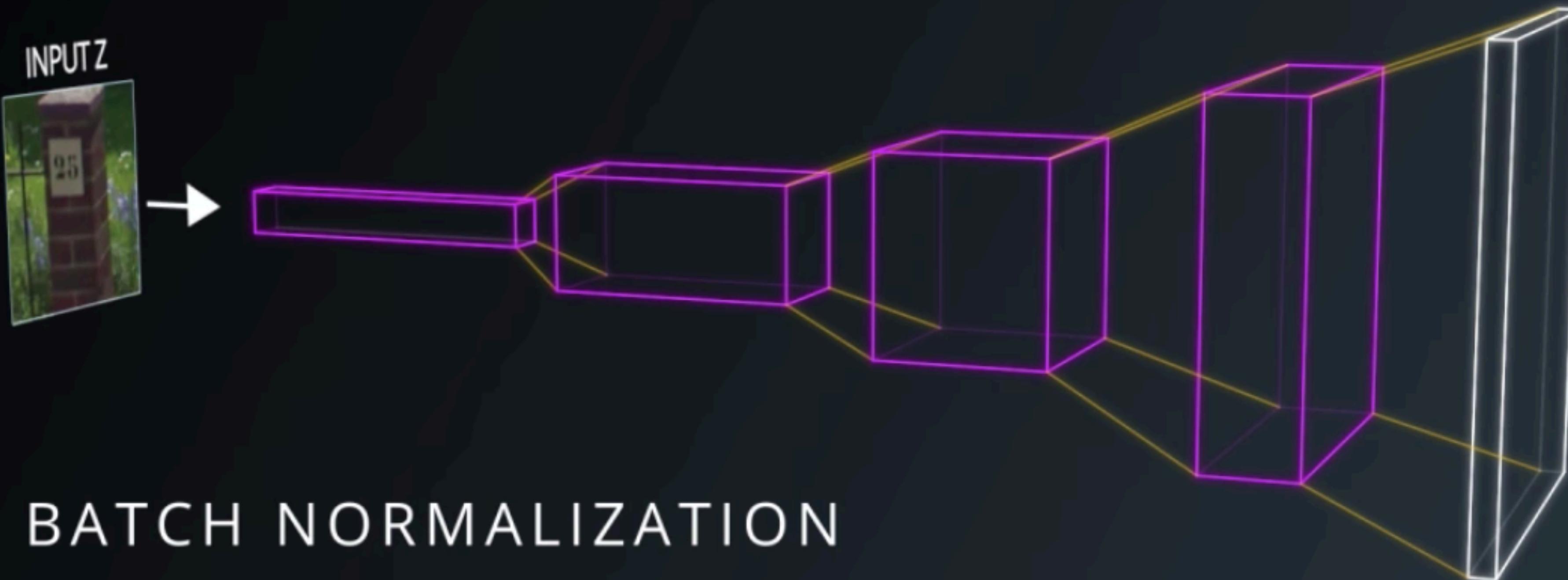
Leaky ReLU



Batch normalization은 레이어의 input을 $\text{mean} = 0$, $\text{variance} = 1$ 로 스케일을 조정합니다.
이는 학습 속도를 빠르게 해주고, poor parameter initialization 문제를 줄여줍니다.
Deeper network에서 학습이 잘 되도록 해주고 더 나은 결과물을 보여줍니다

<https://arxiv.org/abs/1502.03167>

<https://github.com/udacity/deep-learning/blob/master/batch-norm/Batch%20Normalization%20Solutions.ipynb>

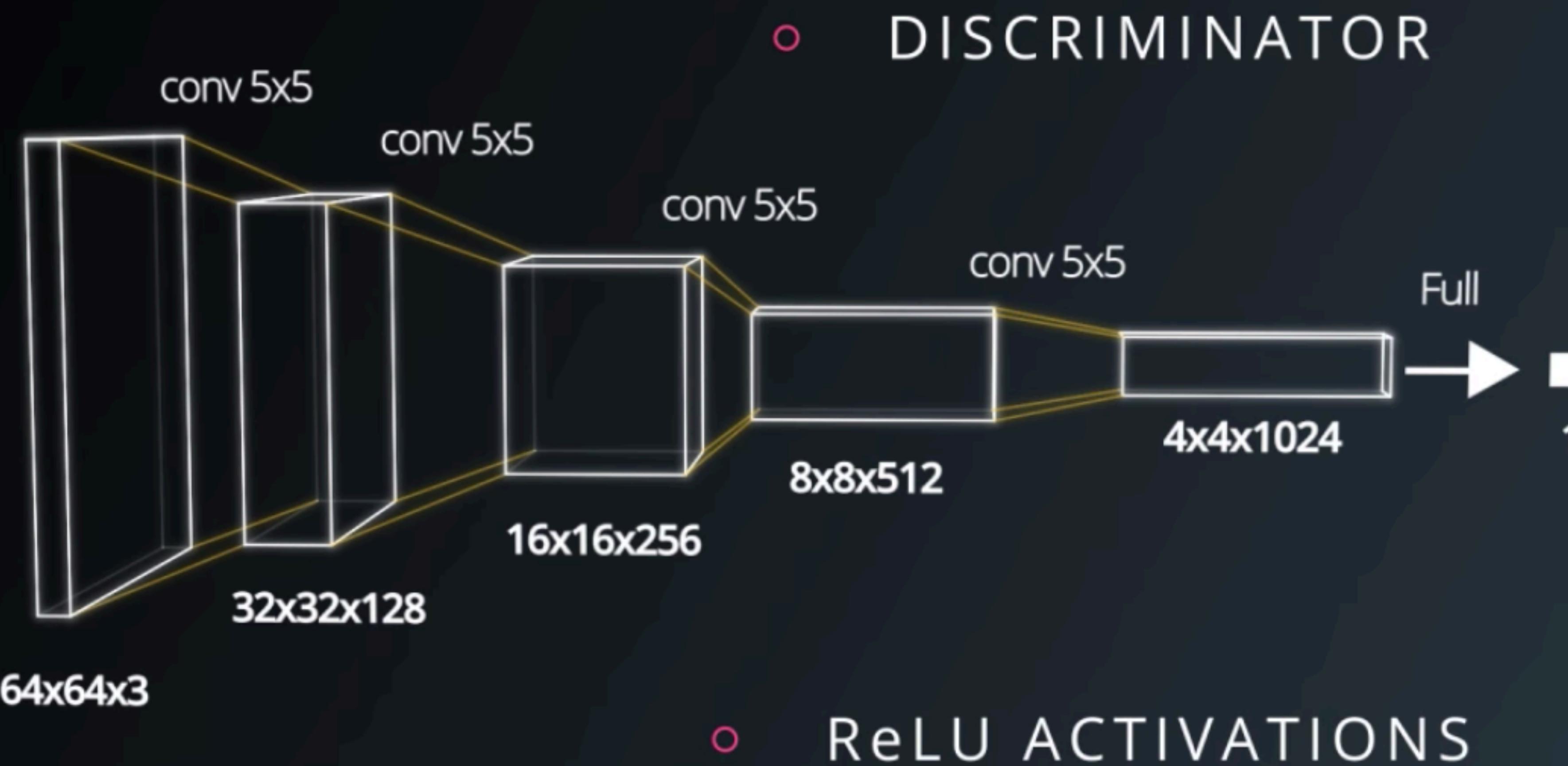


- BATCH NORMALIZATION

Mean = 0 | Variance = 1

가장 마지막 레이어는 Batch Normalization
적용하지 않음

Discriminator도 하나의 Convolutional Network입니다.
제일 마지막에 fully connected network을 두고 sigmoid 출력을 내도록 합니다.



첫번째 레이어에서는 Batch Normalization 생략
마찬가지로 max pool layer를 두지 않습니다.
Downsampling은 전적으로 Stride를 통해 진행됩니다.
제일 마지막 Convolutional Layer는 Flatten되고 Single Sigmoid Unit에 연결되어집니다.

DISCRIMINATOR



BATCH NORMALIZATION

이제 텐서플로 코드를 보러 가야죠.

Jupyter Notebook

Udacity Github은 아래 주소에 :

[https://github.com/udacity/deep-learning/blob/master/dcgan-svhn/
DCGAN.ipynb](https://github.com/udacity/deep-learning/blob/master/dcgan-svhn/DCGAN.ipynb)

제가 돌려본 결과는 여기에 :

[http://nbviewer.jupyter.org/github/anonswspark/babelspeech/blob/master/
part3/gm/gan/dcgan/dcgan-svhn/DCGAN.ipynb](http://nbviewer.jupyter.org/github/anonswspark/babelspeech/blob/master/part3/gm/gan/dcgan/dcgan-svhn/DCGAN.ipynb)

[https://github.com/anonswspark/babelspeech/blob/master/part3/gm/gan/
dcgan/dcgan-svhn/gif/dcgan_street_view_house_numbers.gif](https://github.com/anonswspark/babelspeech/blob/master/part3/gm/gan/dcgan/dcgan-svhn/gif/dcgan_street_view_house_numbers.gif)