SAI - S 2020

Motion and Deep learning

CONTENTS

01 Deep learning in CGI (Motion, Render)

02 Learning motion manifold with CAE

03 Deep learning framework for Motion synthesis

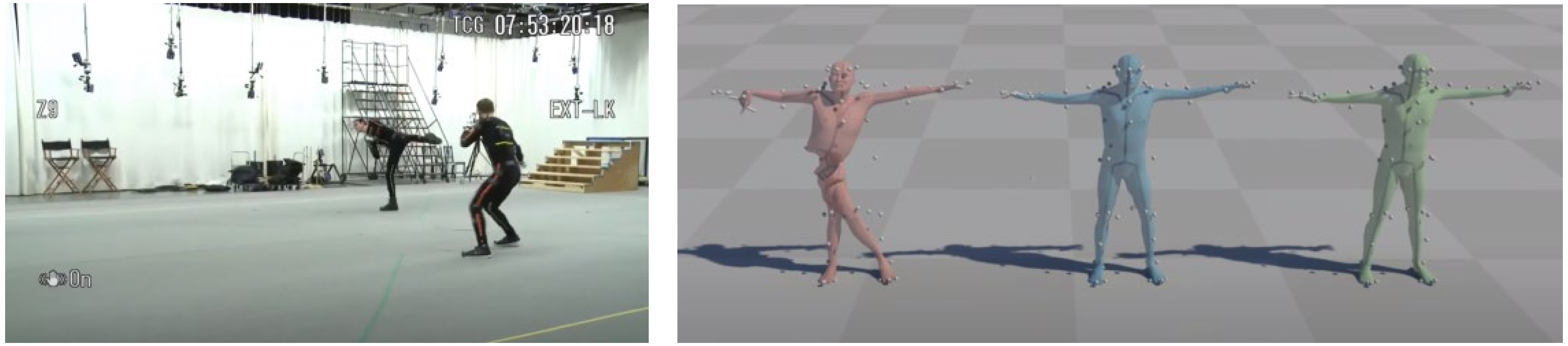
CHAPTER 01

Deep learning in CGI

Denoise

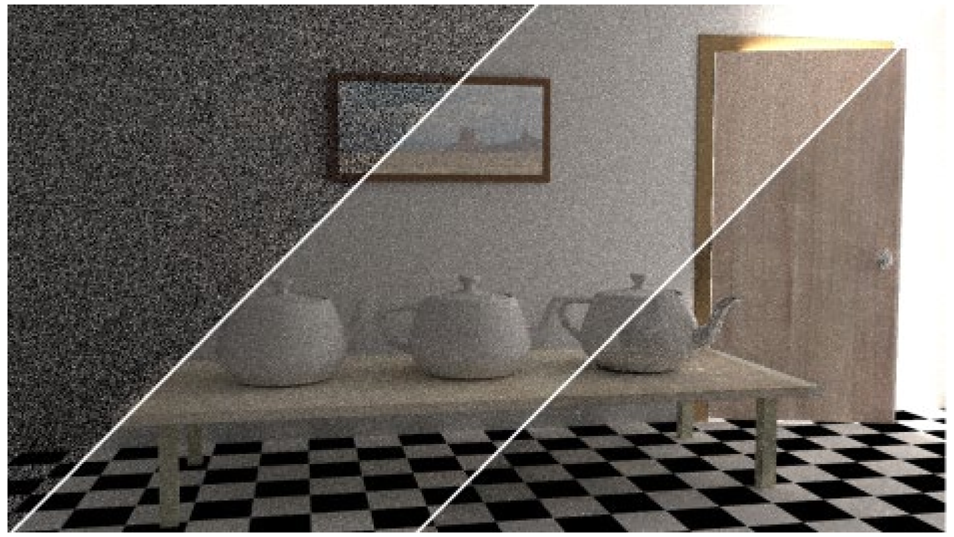
Improving corrupted data

01 모션 입력(좌), 왼쪽에서부터 오른쪽으로 순서대로 uncleaned data, denoised data, hand-cleaned data



출처: Ubisoft

02 좌우 그림에서 좌측은 간단한 path tracing으로 렌더된 이미지들, 중간은 RL/ML 개선된 이미지들



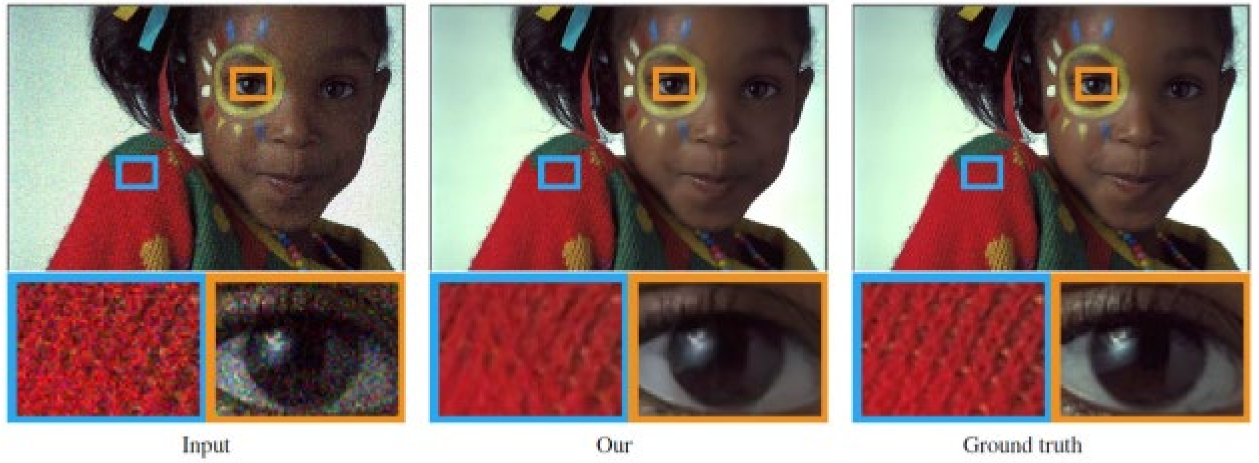


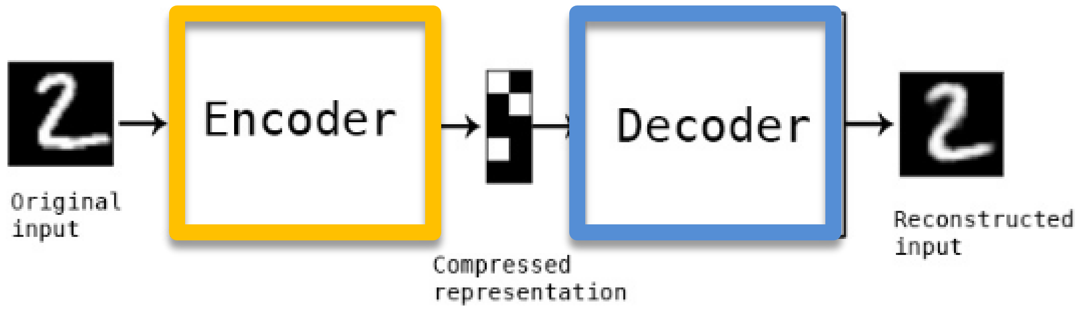
Figure 1. Example results for Poisson noise ($\lambda = 30$). Our result was computed by using noisy targets.

AN OVERVIEW

CHAPTER 02

Learning motion manifold with Convolutional AE

What is Convolutional AutoEncoder?



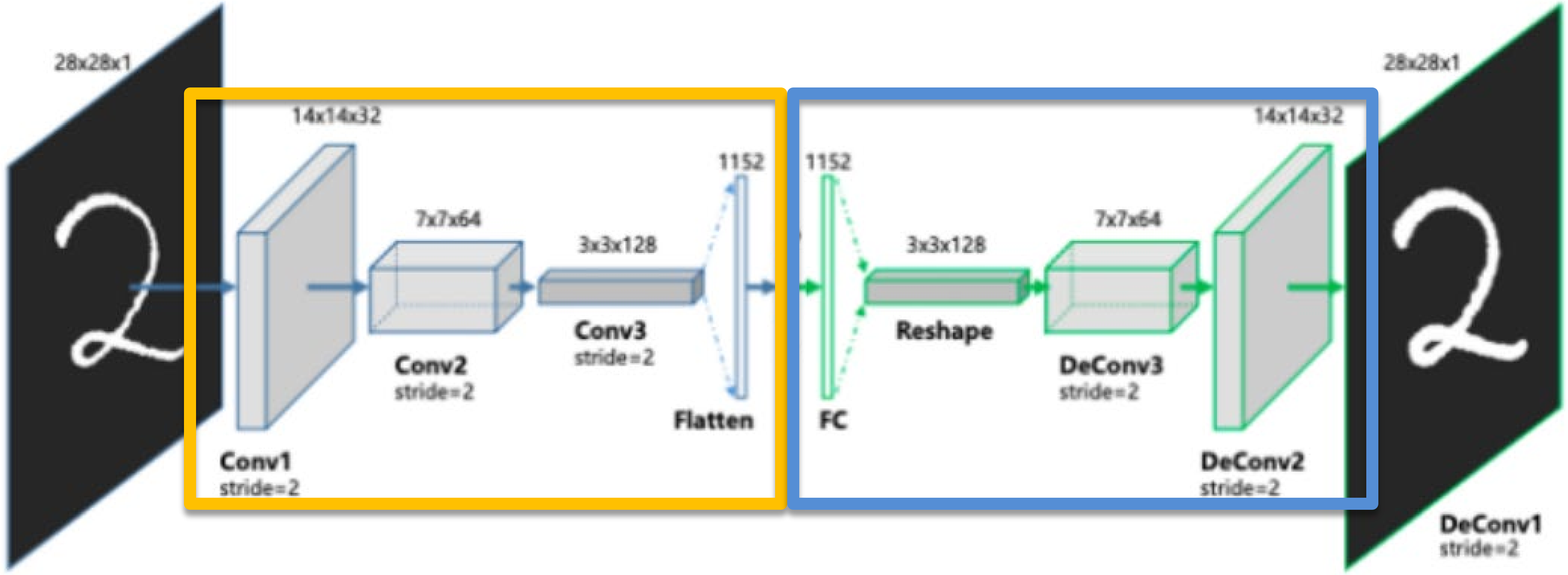
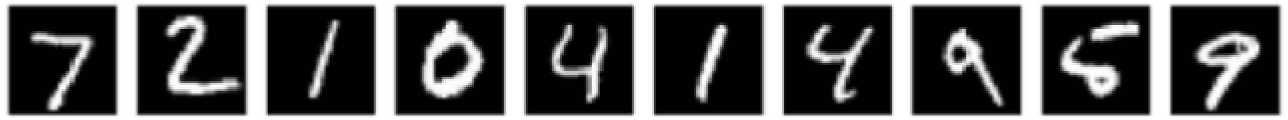
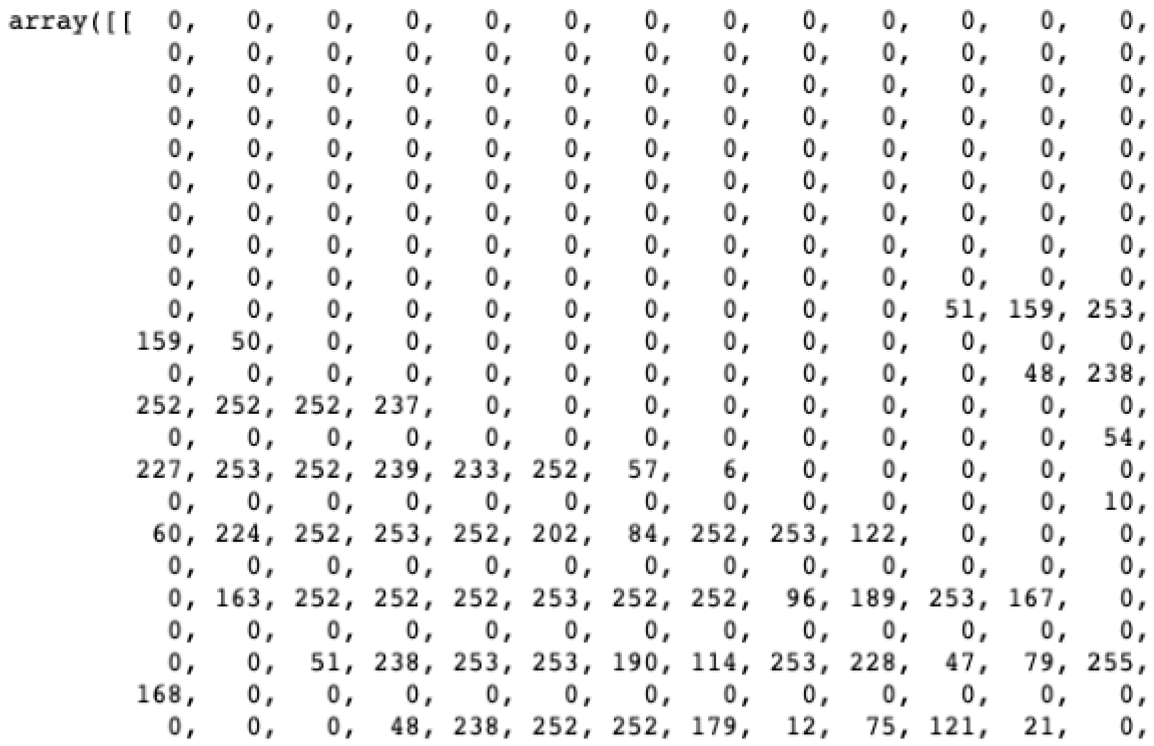


Figure (D)

Encoder: Filtering + MaxPooling



In order to fit a neural network framework for model training, we can stack all the 28 x 28 = 784 values in a column. The stacked column for the first record look like this: (using `x_train[1].reshape(1,784)`):



```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  51, 159, 253,
        159,  50,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  48, 238,
        252, 252, 252, 237,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  54,
        227, 253, 252, 239, 233, 252,  57,   6,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  10,
         60, 224, 252, 253, 252, 202,  84, 252, 253, 122,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0, 163, 252, 252, 252, 253, 252, 252,  96, 189, 253, 167,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,  51, 238, 253, 253, 190, 114, 253, 228,  47,  79, 255,
        168,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,  48, 238, 252, 252, 179,  12,  75, 121,  21,   0,
```

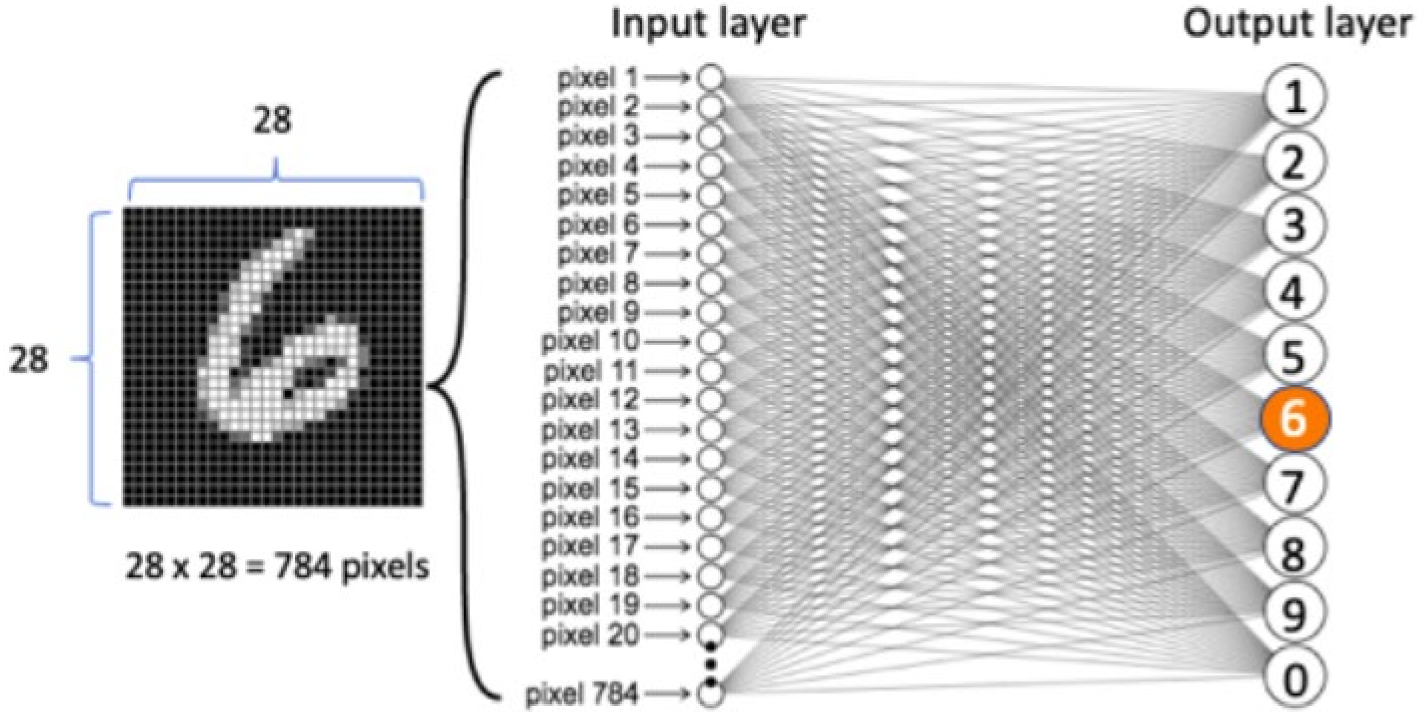


Figure (B)

Encoder: Filtering + MaxPooling

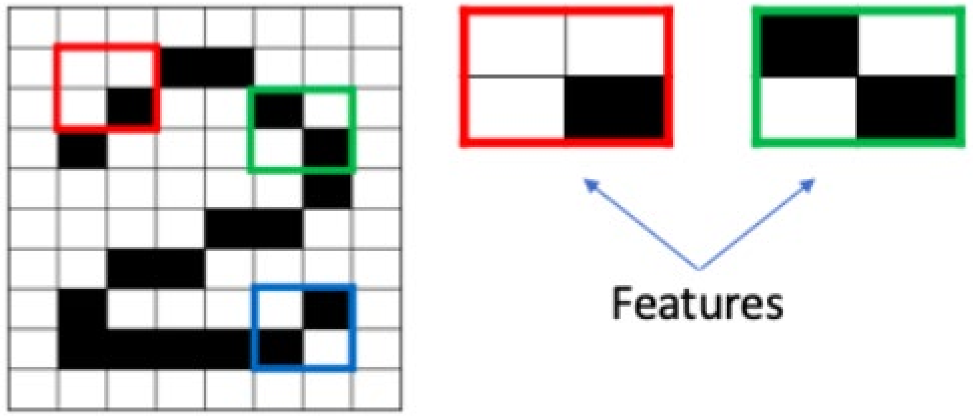
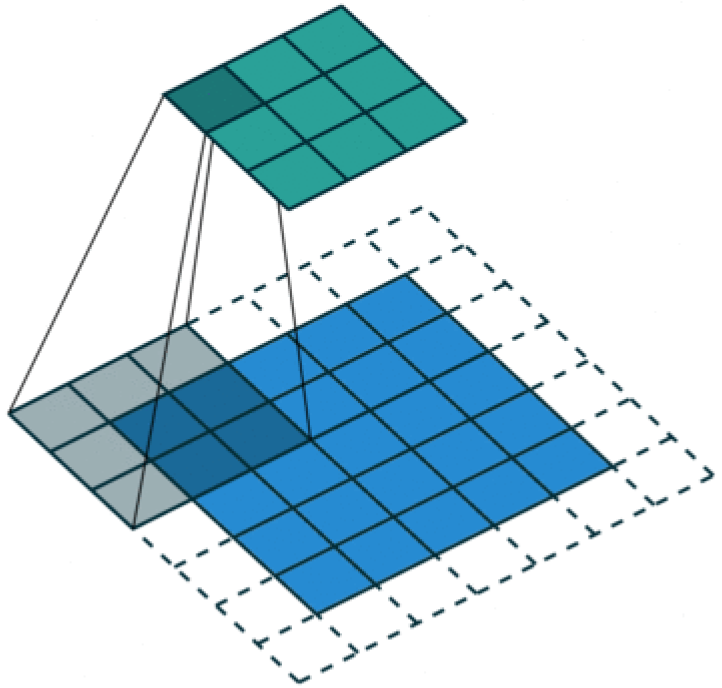


Figure (E): The Feature Maps



filtering

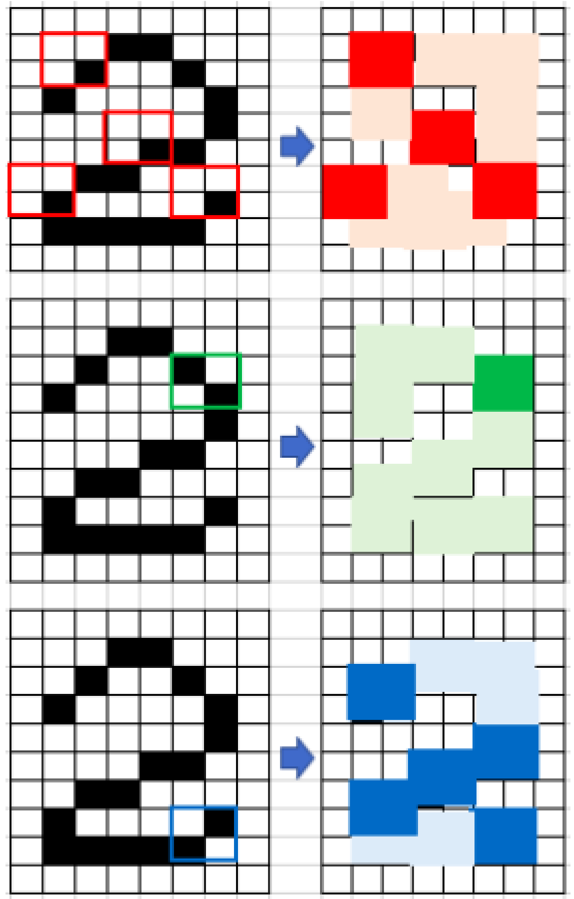


Figure (G)

Encoder: Filtering + MaxPooling

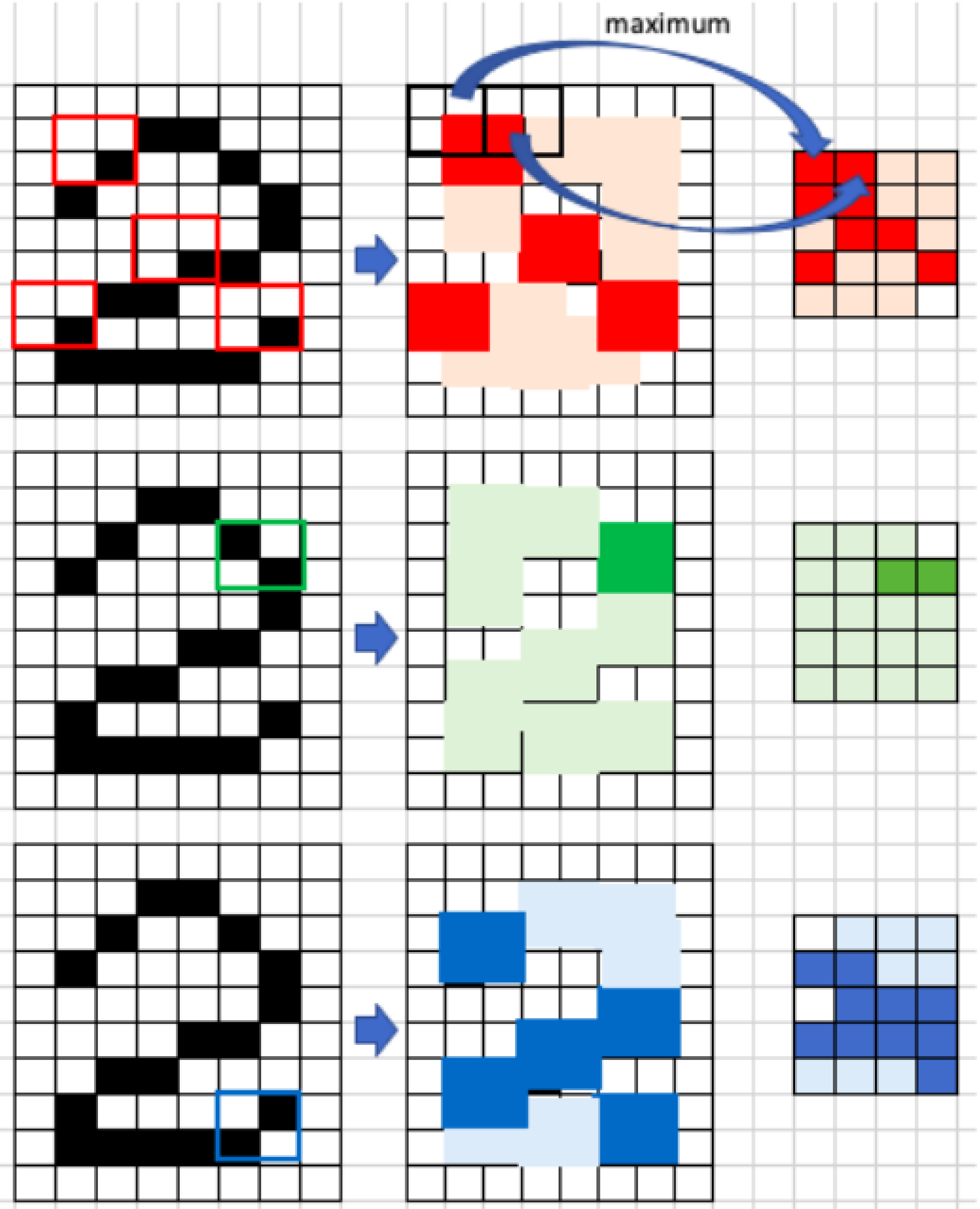


Figure (H): Max Pooling

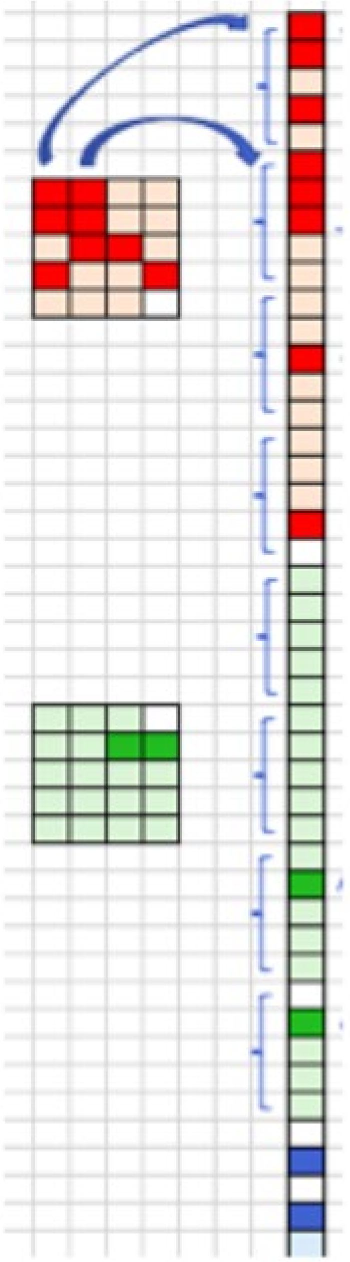
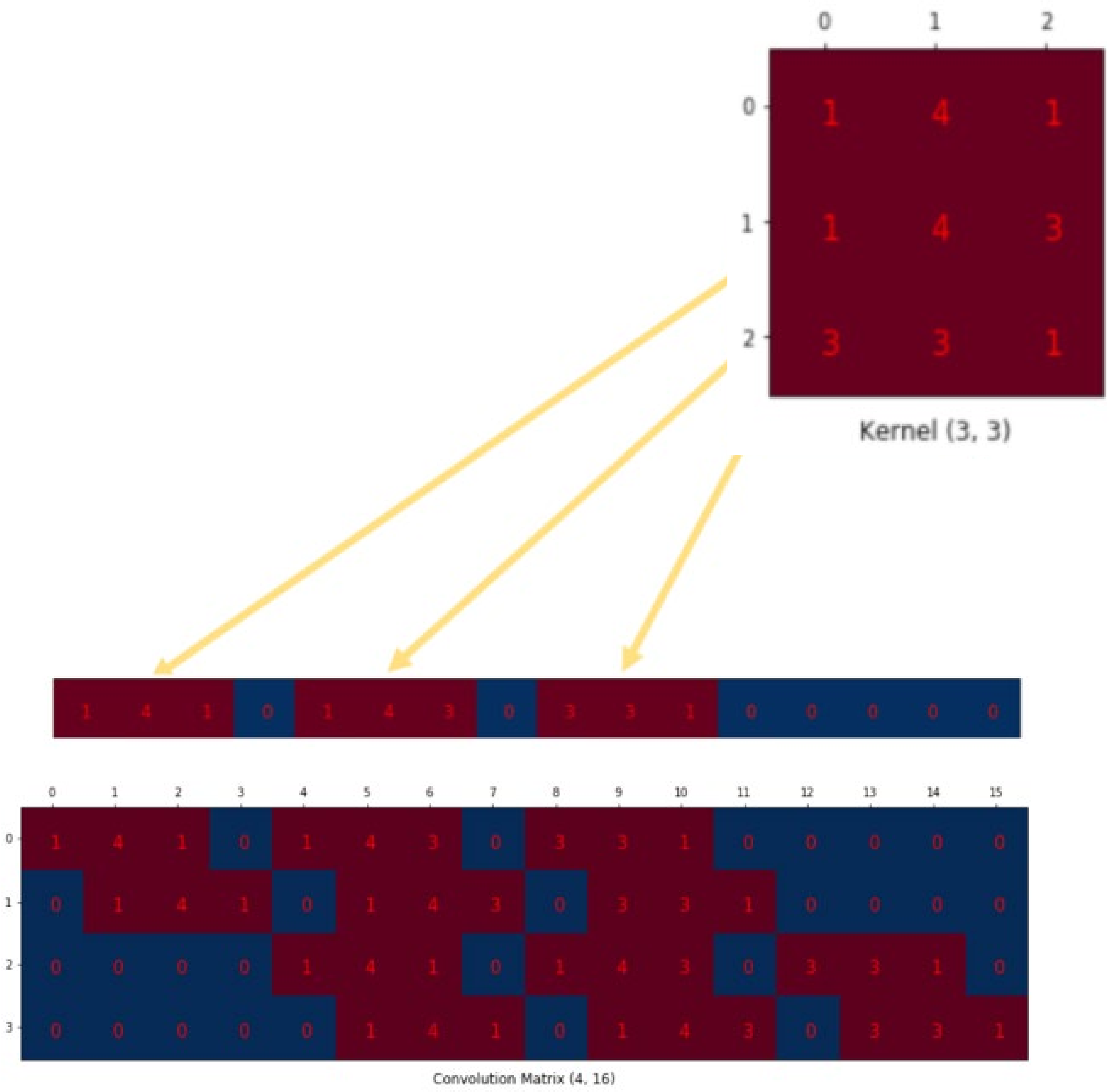


Figure (J)

Max pooling

Decoder: Filtering + UpSampling

Kernel을 다음과 같이 16 * 4 Matrix로 펼친다.



	0	1	2
0	1	4	1
1	1	4	3
2	3	3	1

Kernel (3, 3)

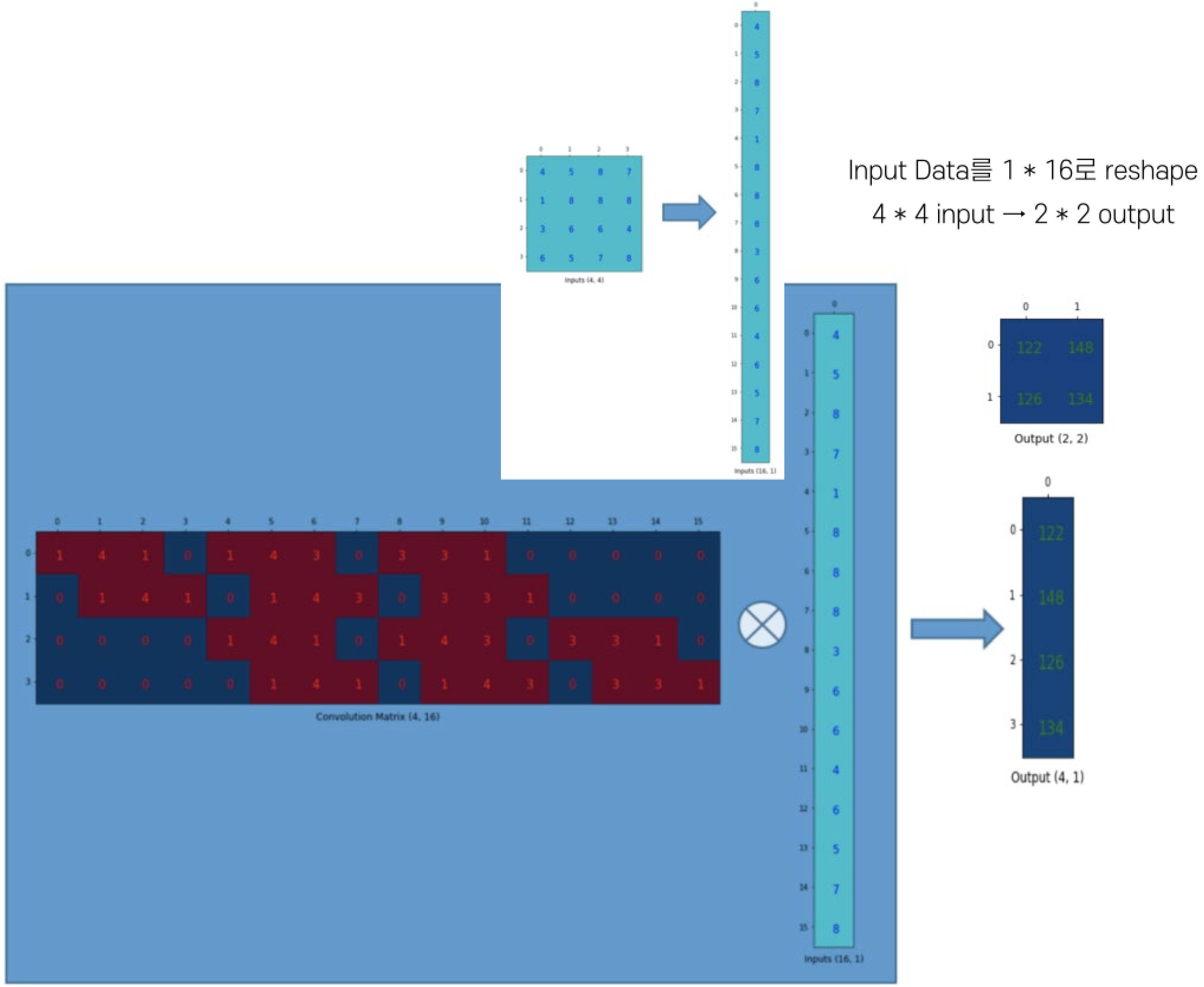
1	4	1	0	1	4	3	0	3	3	1	0	0	0	0	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	4	1	0	1	4	3	0	3	3	1	0	0	0	0	0
1	0	1	4	1	0	1	4	3	0	3	3	1	0	0	0	0
2	0	0	0	0	1	4	1	0	1	4	3	0	3	3	1	0
3	0	0	0	0	0	1	4	1	0	1	4	3	0	3	3	1

Convolution Matrix (4, 16)

Input Data를 1 * 16로 reshape

4 * 4 input → 2 * 2 output



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

Inputs (4, 4)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	4	1	0	1	4	3	0	3	3	1	0	0	0	0	0
1	0	1	4	1	0	1	4	3	0	3	3	1	0	0	0	0
2	0	0	0	0	1	4	1	0	1	4	3	0	3	3	1	0
3	0	0	0	0	0	1	4	1	0	1	4	3	0	3	3	1

Convolution Matrix (4, 16)

	0
0	4
1	5
2	8
3	7
4	1
5	8
6	8
7	8
8	3
9	6
10	6
11	4
12	6
13	5
14	7
15	8

Inputs (16, 1)

	0	1
0	122	148
1	126	134

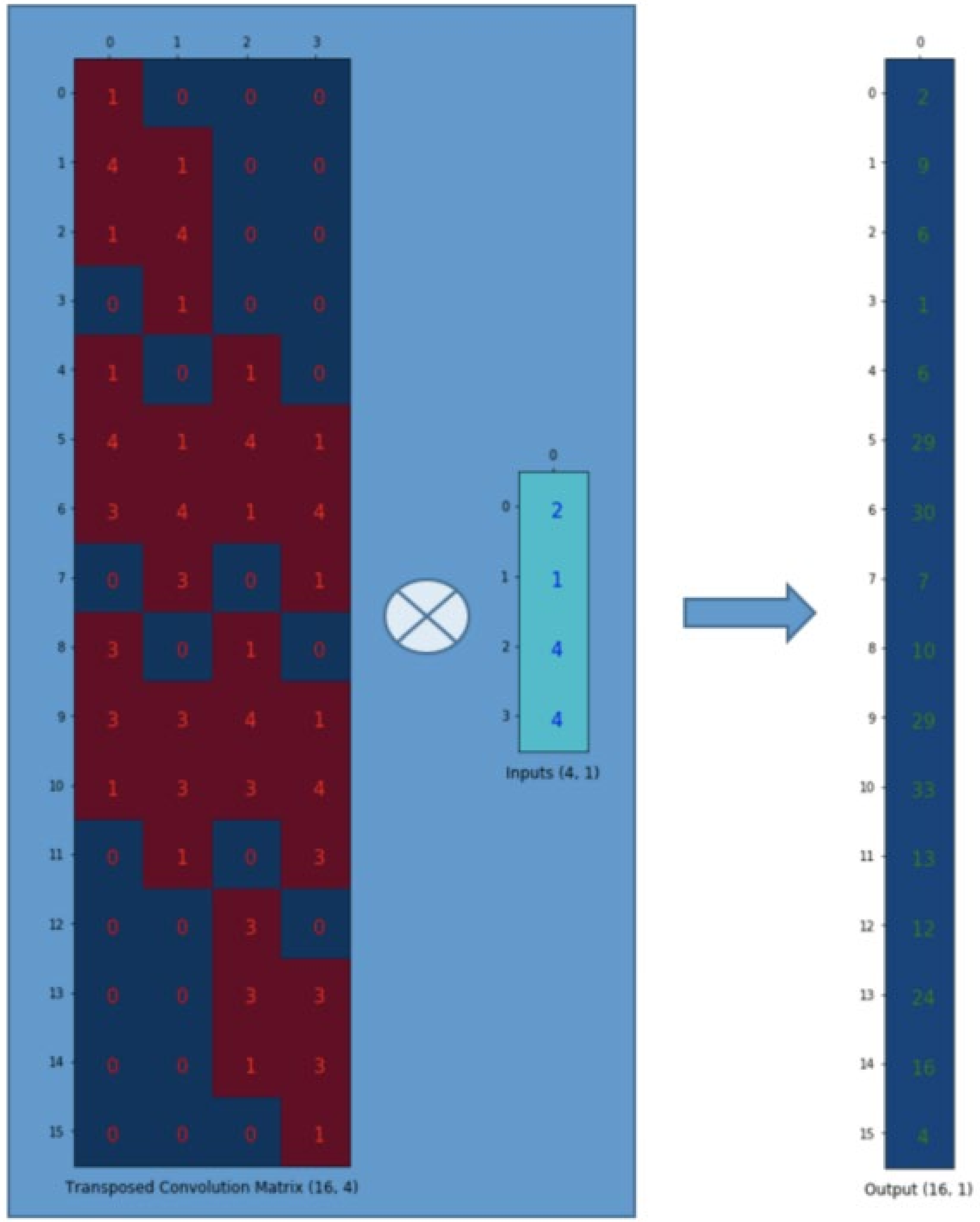
Output (2, 2)

	0
0	122
1	148
2	126
3	134

Output (4, 1)

Decoder: Filtering + UpSampling

Transpose Convolutional matrix



Transposed Convolution Matrix (16, 4)

	0	1	2	3
0	1	0	0	0
1	4	1	0	0
2	1	4	0	0
3	0	1	0	0
4	1	0	1	0
5	4	1	4	1
6	3	4	1	4
7	0	3	0	1
8	3	0	1	0
9	3	3	4	1
10	1	3	3	4
11	0	1	0	3
12	0	0	3	0
13	0	0	3	3
14	0	0	1	3
15	0	0	0	1

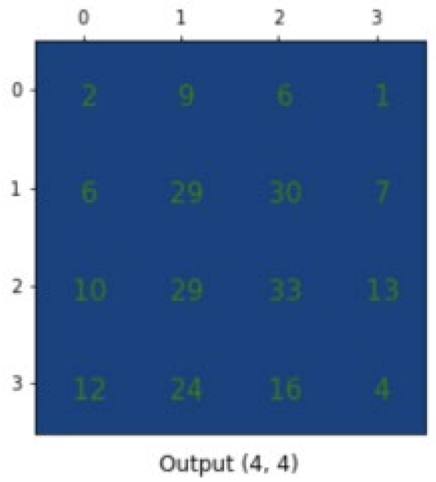
Inputs (4, 1)

	0
0	2
1	1
2	4
3	4

Output (16, 1)

	0
0	2
1	9
2	6
3	1
4	6
5	29
6	30
7	7
8	10
9	29
10	33
11	13
12	12
13	24
14	16
15	4

Convolution By Matrix Multiplication



Output (4, 4)

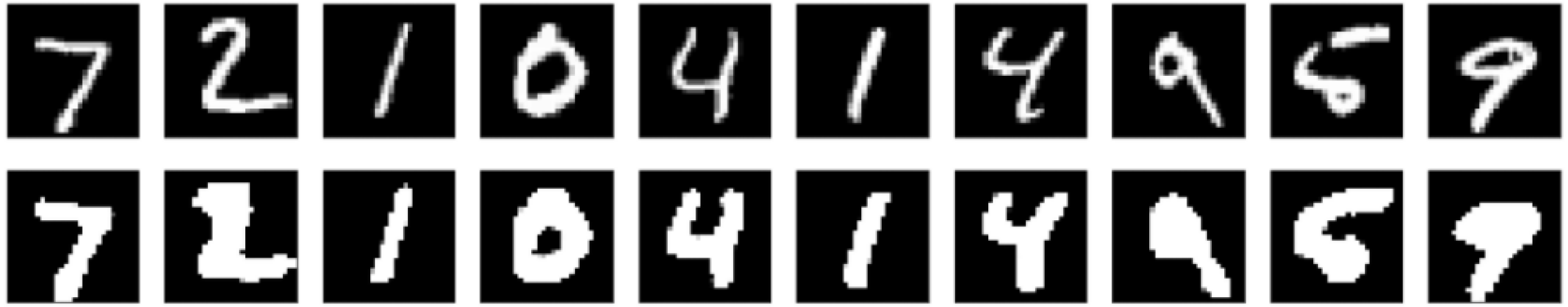
	0	1	2	3
0	2	9	6	1
1	6	29	30	7
2	10	29	33	13
3	12	24	16	4

Input Data(2*2)를 1*4로 reshape

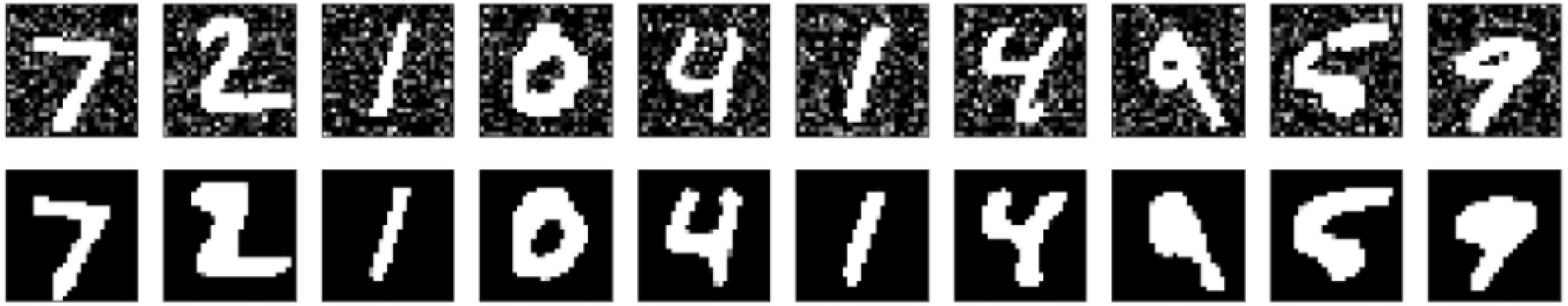
2*2 input → 4*4 output

궁금해서 한 번 해본 Autoencoder

모델 학습 결과물:



결과물:



결론: 그냥 CNN이다

Encoder & Decoder가 있는

Learning motion manifold with CAE

다량의, 노이즈가 낀 모션 데이터를 CAE로 학습해보자

Biologically impossible
Unrealistic “Fast” motion

Valid motion

Time series of human pose: Visible unit

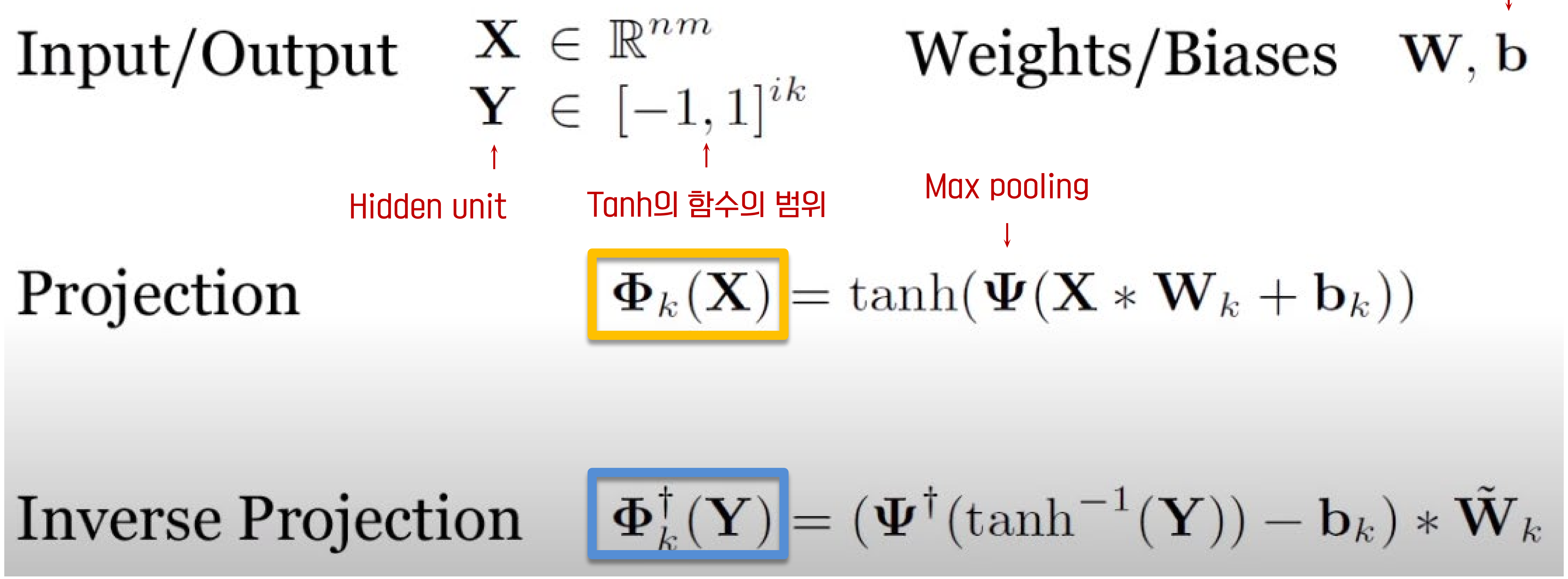
n: num of frames

m: num of degree of freedom,

↓

Initial value: 0

↓



Input/Output $\mathbf{X} \in \mathbb{R}^{nm}$

$\mathbf{Y} \in [-1, 1]^{ik}$

↑ ↑

Hidden unit Tanh의 함수의 범위

Weights/Biases $\mathbf{W}, \mathbf{b}$

Max pooling

↓

Projection $\boxed{\Phi_k(\mathbf{X})} = \tanh(\Psi(\mathbf{X} * \mathbf{W}_k + \mathbf{b}_k))$

Inverse Projection $\boxed{\Phi_k^\dagger(\mathbf{Y})} = (\Psi^\dagger(\tanh^{-1}(\mathbf{Y})) - \mathbf{b}_k) * \tilde{\mathbf{W}}_k$

15 frames = 30/2 frames per sec/2

↓

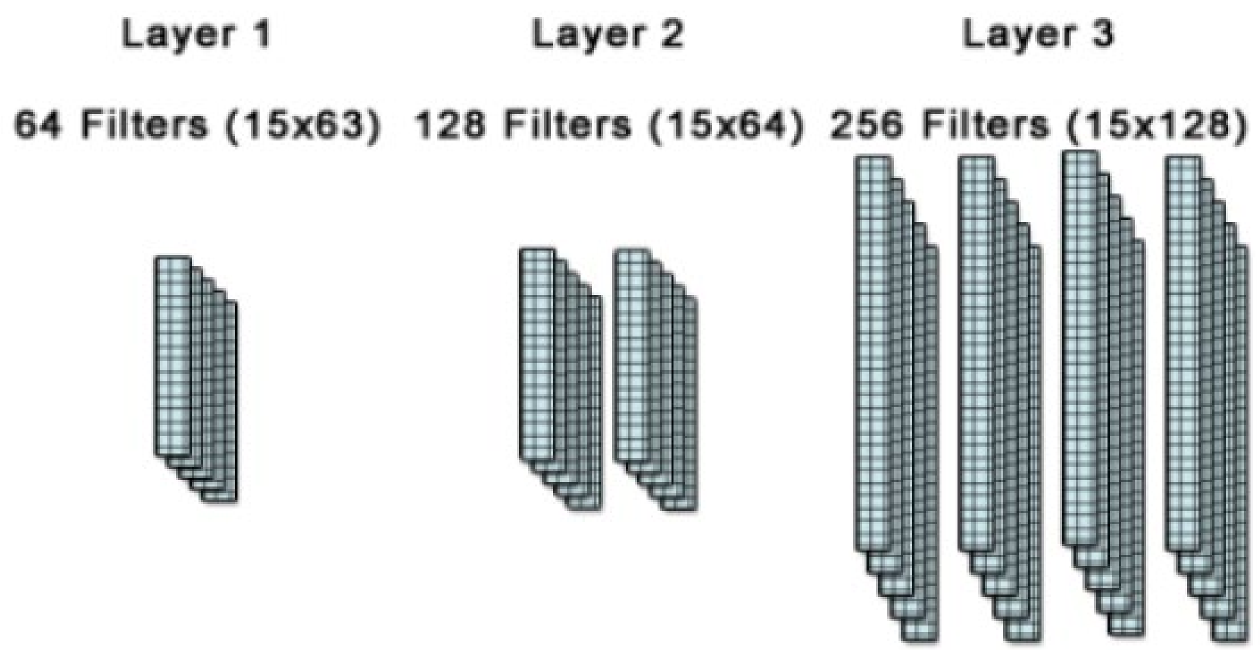


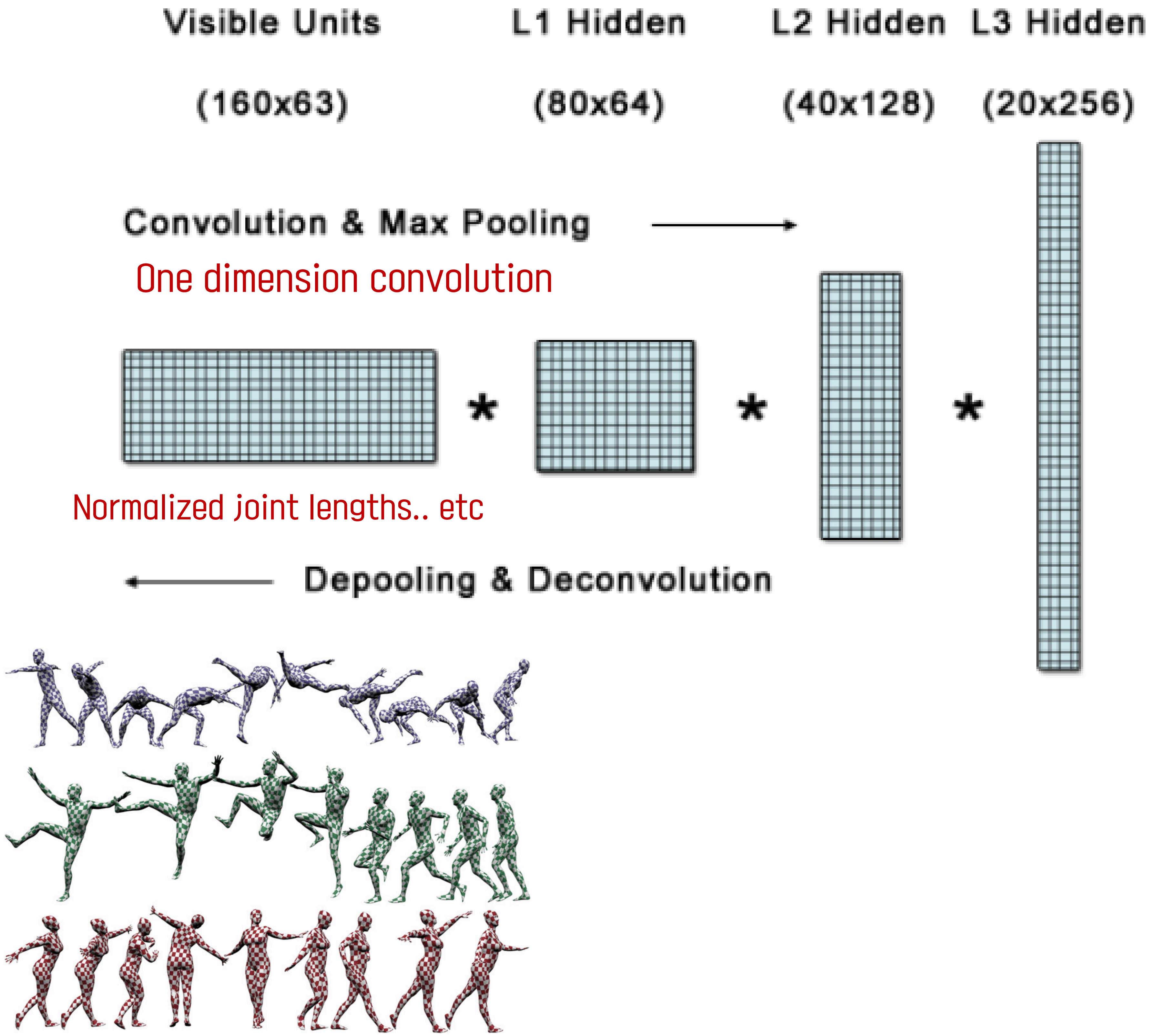
Figure 2: *Structure of the Convolutional Autoencoder. Layer 1 contains 64 filters of size 15x63. Layer 2 contains 128 filters of size 15x64. Layer 3 contains 256 filters of size 15x128. The first dimension of the filter corresponds to a temporal window, while the second dimension corresponds to the number of features/filters on the layer below.*

Sub-sampled input data = X

160 frames * 63 degree of freedom of 20 joints

↓

※ 30 frame per sec,
160 frames roughly covers 5 sec
= covers distinct motion



One dimension convolution

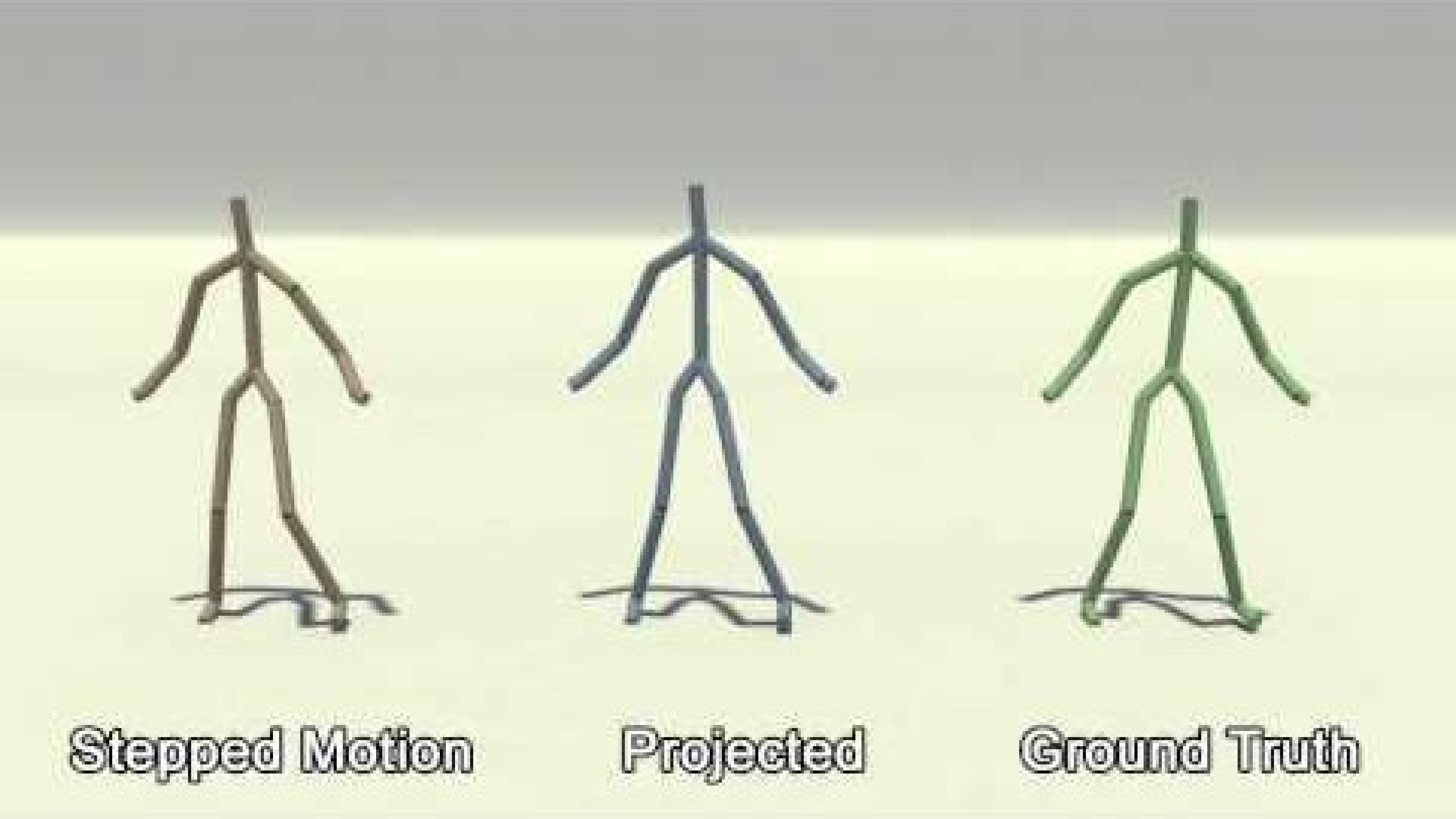
Normalized joint lengths.. etc

Encoder: $\boxed{\mathbf{\Phi}_k(\mathbf{X})} = \tanh(\mathbf{\Psi}(\mathbf{X} * \mathbf{W}_k + \mathbf{b}_k))$

Decoder: $\boxed{\mathbf{\Phi}_k^{\dagger}(\mathbf{Y})} = (\mathbf{\Psi}^{\dagger}(\tanh^{-1}(\mathbf{Y})) - \mathbf{b}_k) * \tilde{\mathbf{W}}_k$

$$Loss(\mathbf{X}) = \|\overset{\substack{\text{Ground truth data}\\ \downarrow}}{\mathbf{X}} - \boxed{\mathbf{\Phi}^{\dagger}(\underset{\substack{\uparrow\\ \text{노이즈가 첨가된 data}}}{\boxed{\mathbf{\Phi}(\mathbf{X}_c)}})}\|_2^2 + \overset{\substack{0.01\\ \downarrow}}{\alpha}\,\overset{\substack{\text{Additional sparsity constraint}\\ \text{: ensures min number of hidden units be active at a time}\\ \downarrow}}{\|\mathbf{\Phi}(\mathbf{X}_c)\|_1}$$

Mean Squared Error Loss


Stepped Motion
Projected
Ground Truth

CHAPTER 03

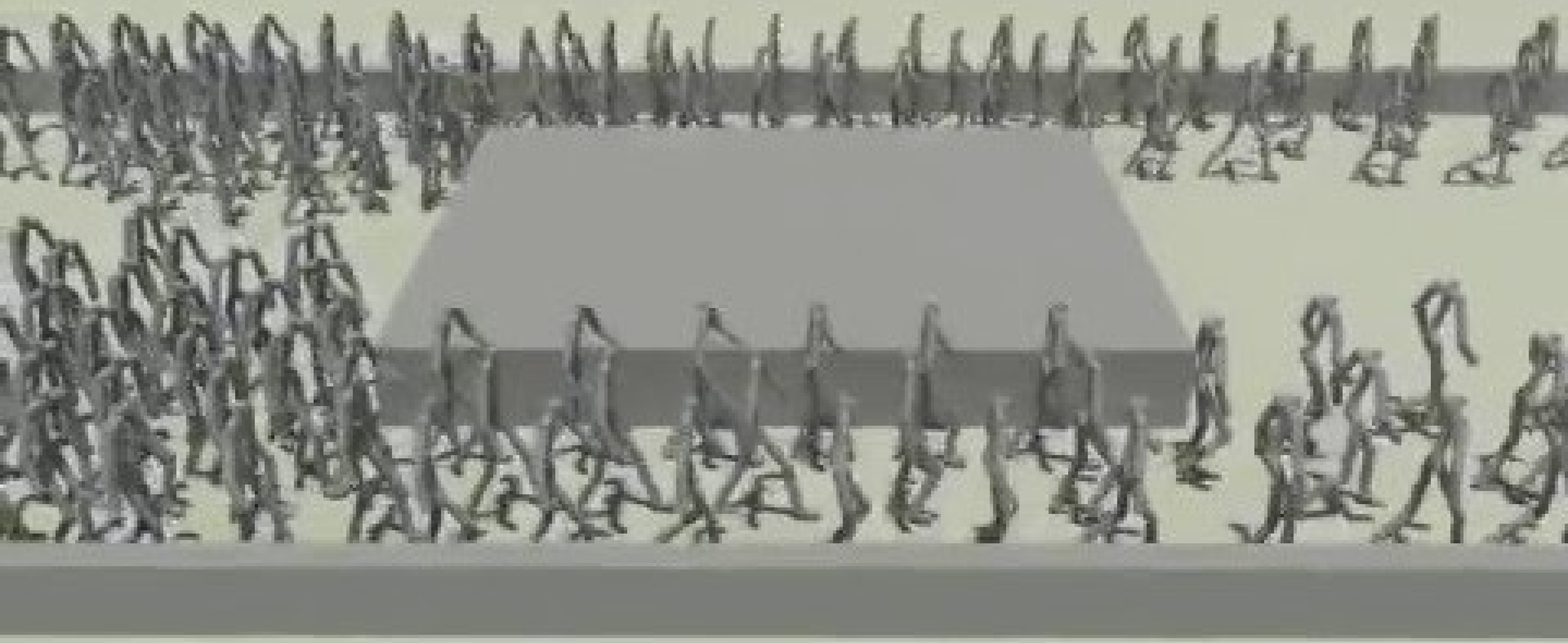
DL framework for Character Motion synthesis

03

CHAPTER

요즘 너무 바빠서 이 부분을 제가 잘 준비를 해오지 못했습니다.
다음학기에 하게 된다면 이어서 준비해보도록 하겠습니다 ㅠㅠ

DL framework for Character Motion synthesis



THANK YOU EVERYONE

Copyright 2020 © GitHub: @ameliacode