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

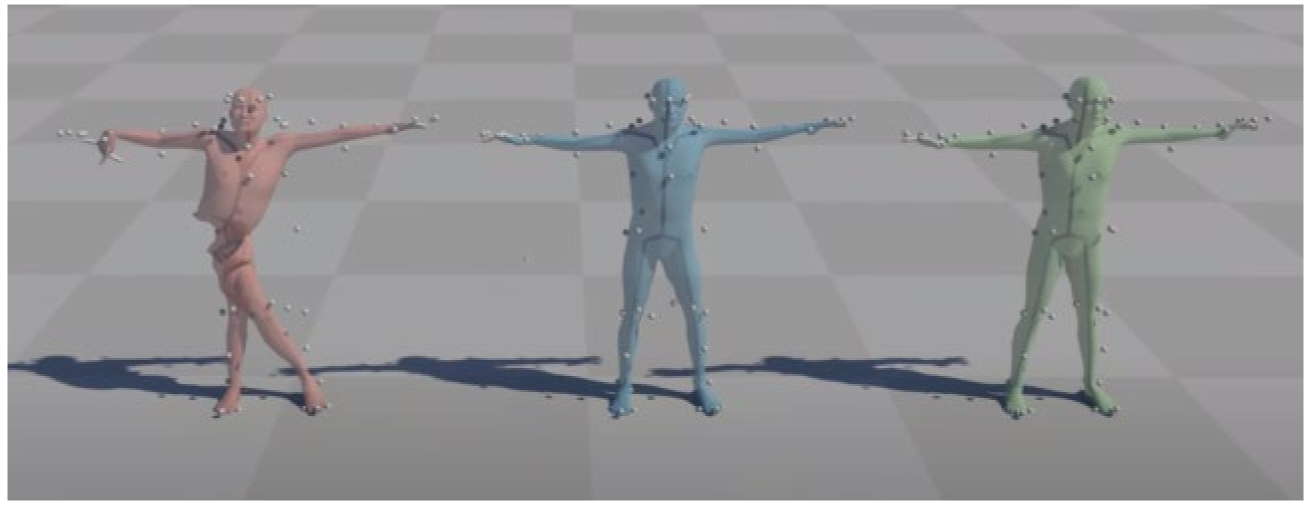
CHAPTER 01

Deep learning in CGI

Denoise

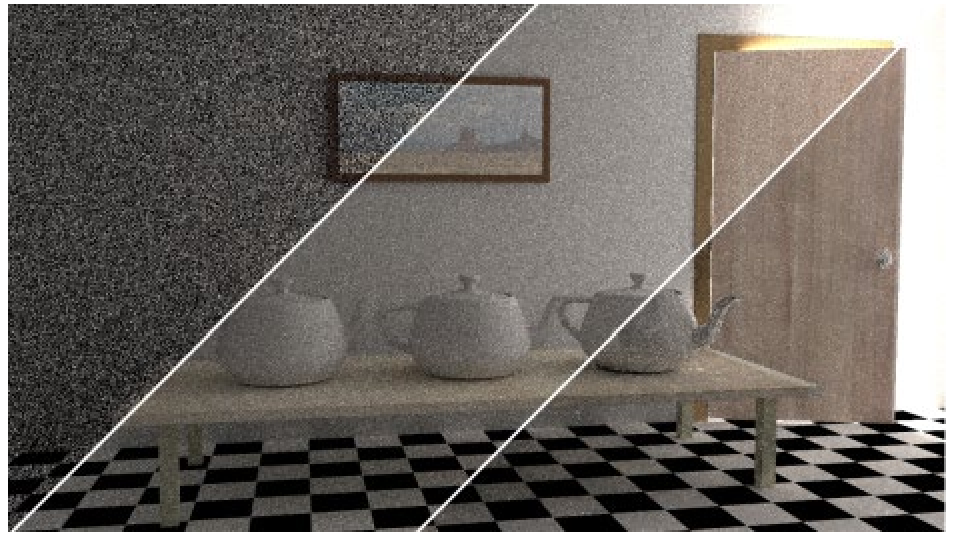
Improving corrupted data

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



출처: Ubisoft

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



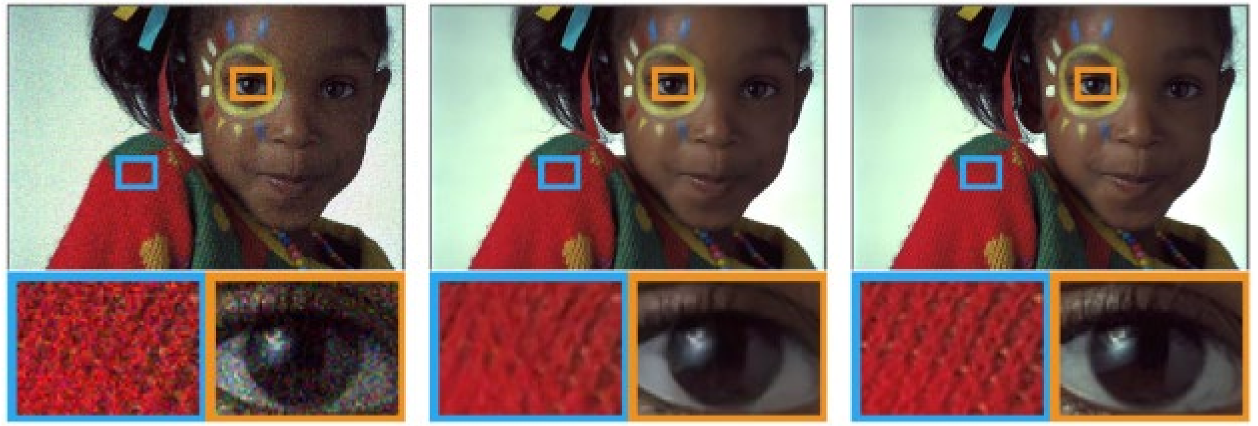


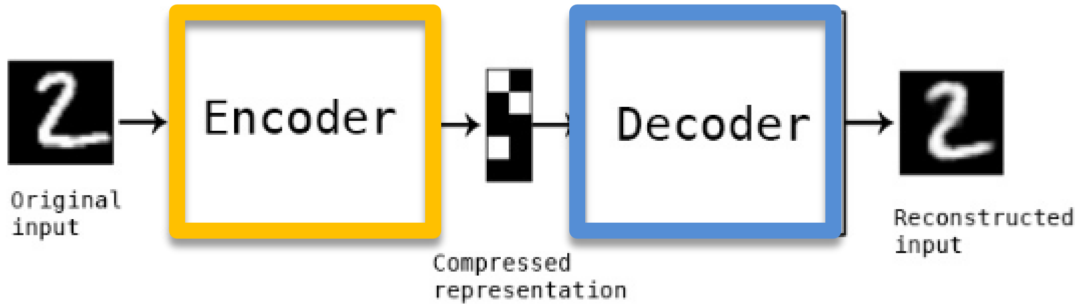
Figure 1. Example results for Poisson noise ($\lambda = 30$). Our result was computed by using noisy targets.

AN OVERVIEW

CHAPTER 02

Learning motion manifold with Convolutional AE

What is Convolutional AutoEncoder?



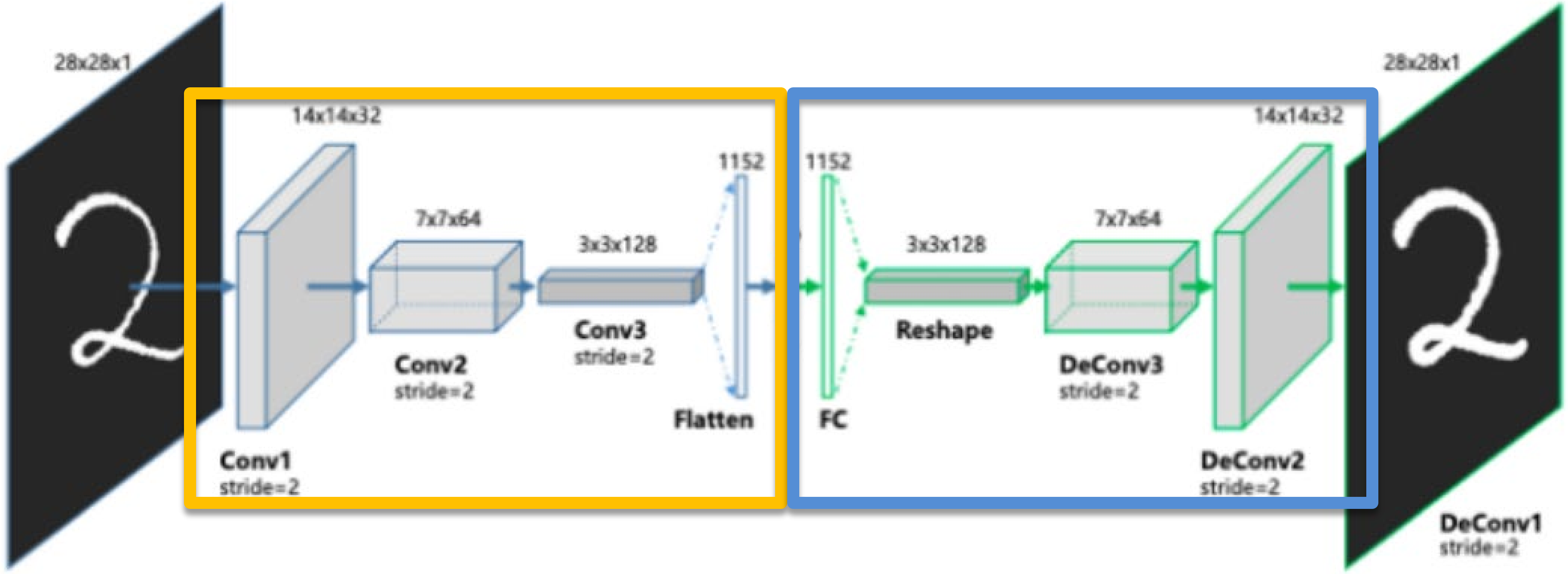
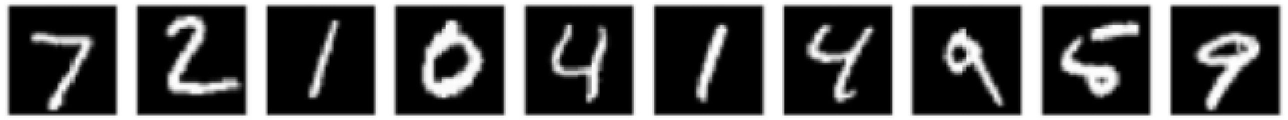


Figure (D)

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,
```

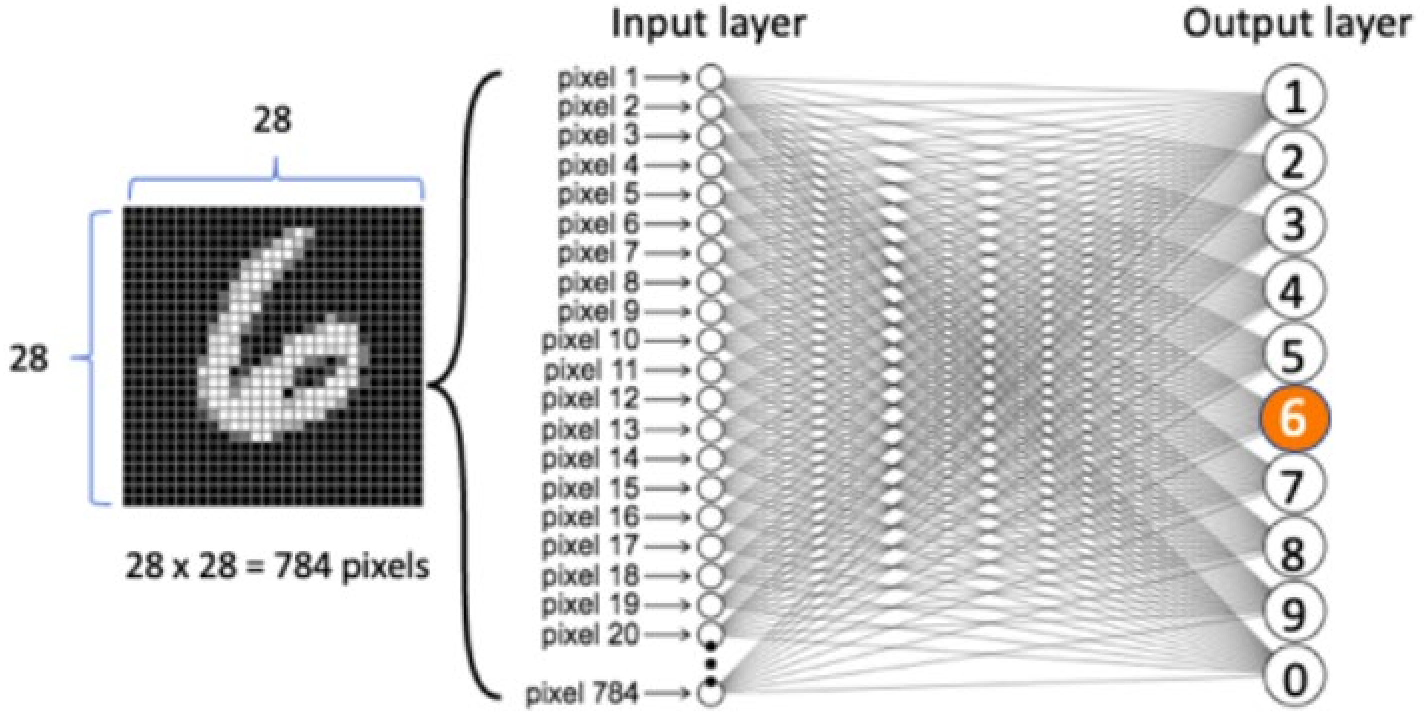


Figure (B)

Encoder: Filtering + MaxPooling

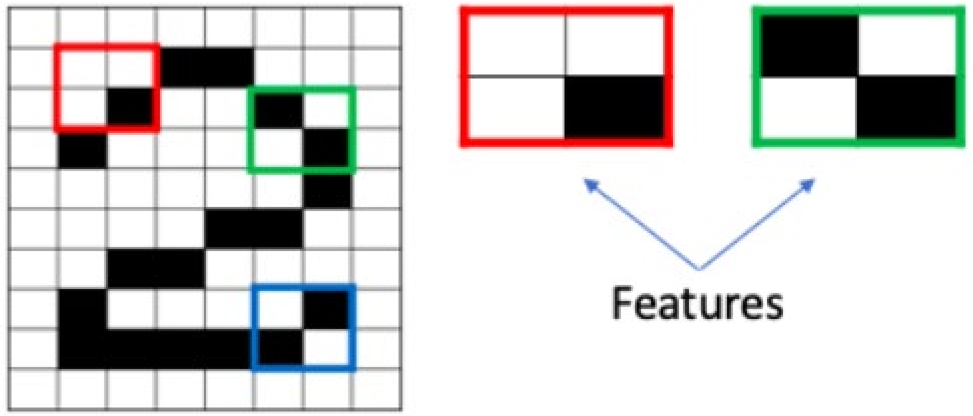
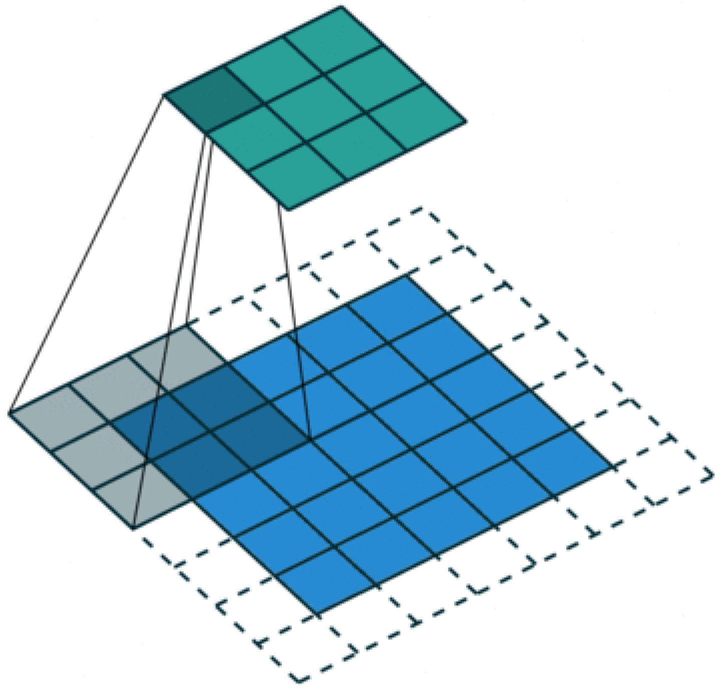


Figure (E): The Feature Maps



filtering

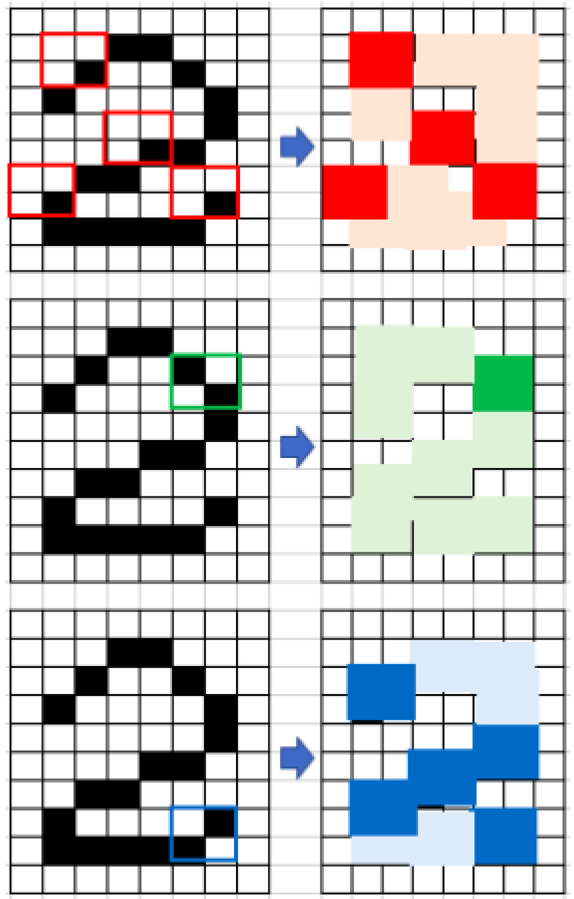


Figure (G)

Encoder: Filtering + MaxPooling

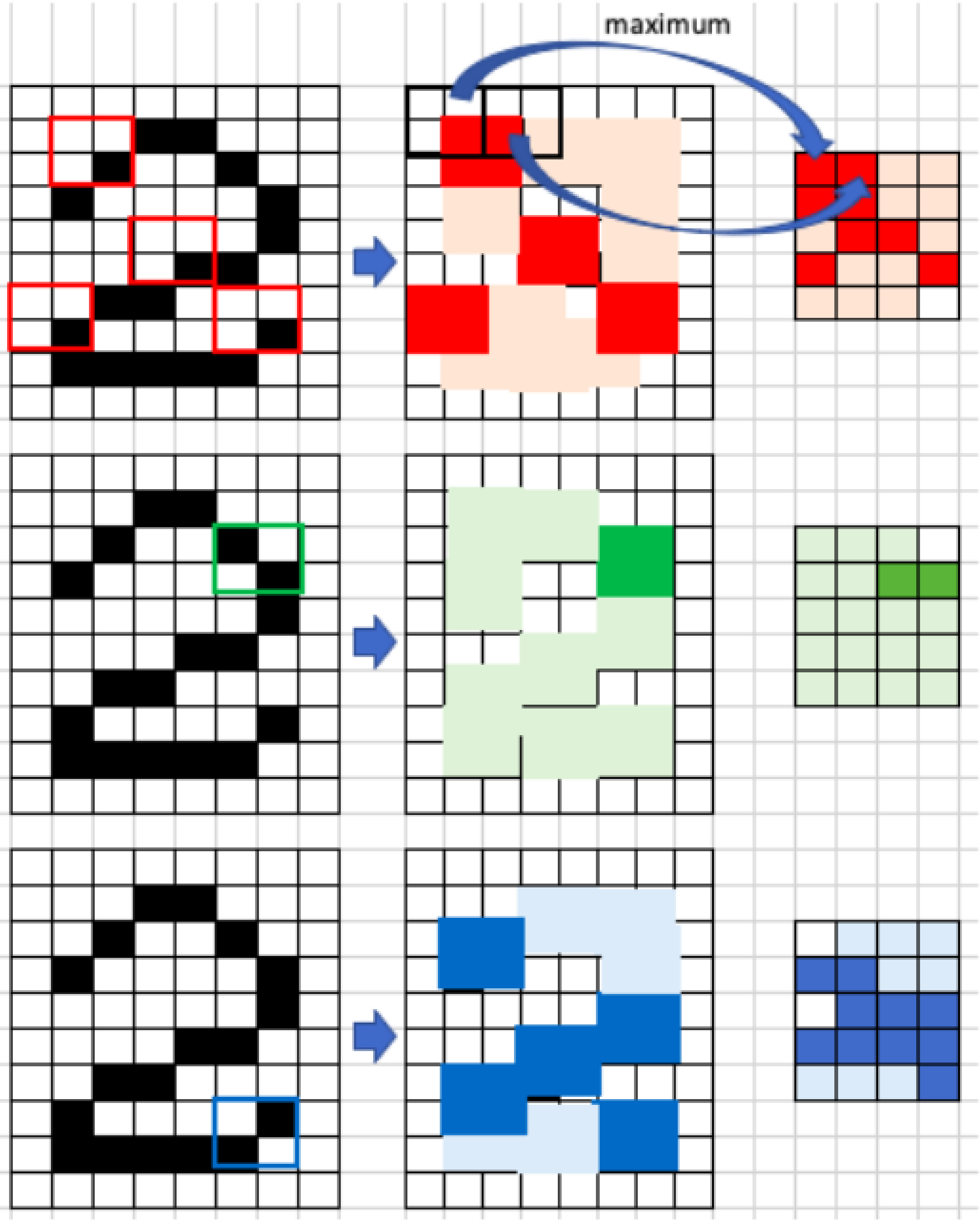
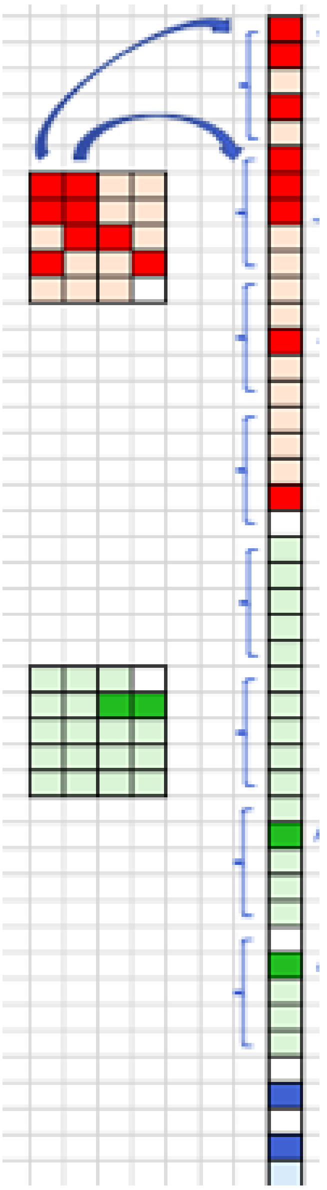


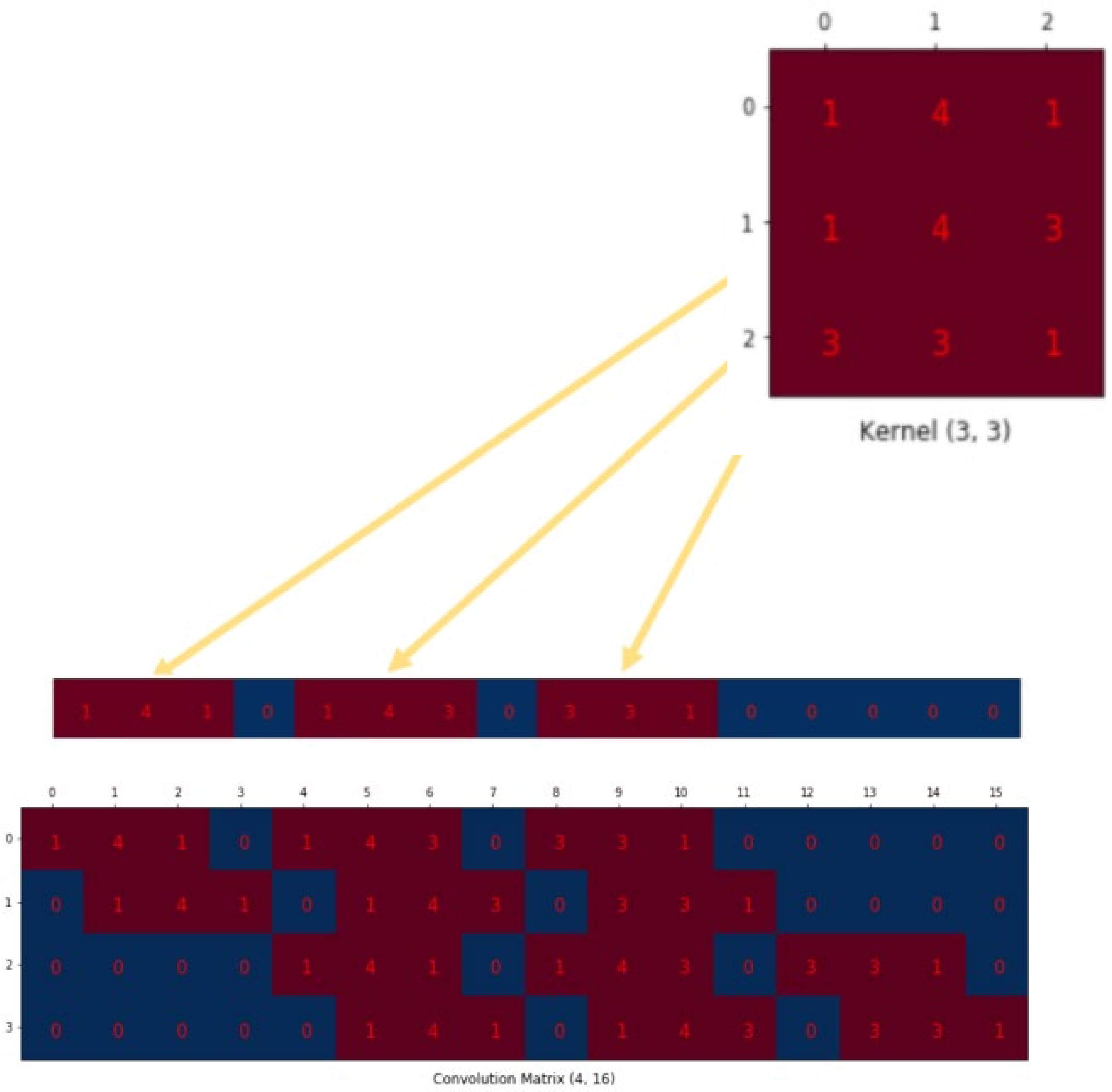
Figure (H): Max Pooling



Max pooling

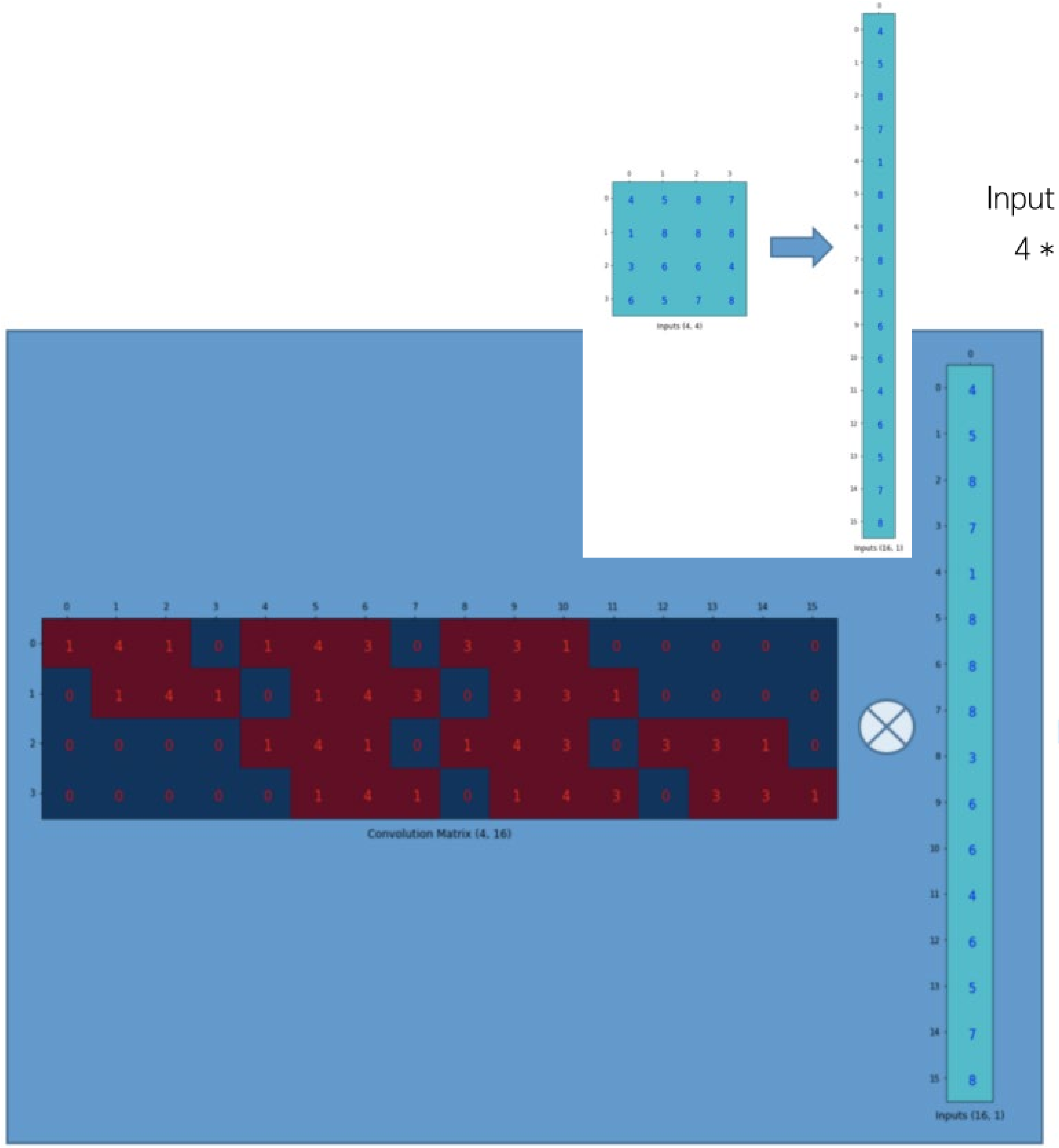
Decoder: Filtering + UpSampling

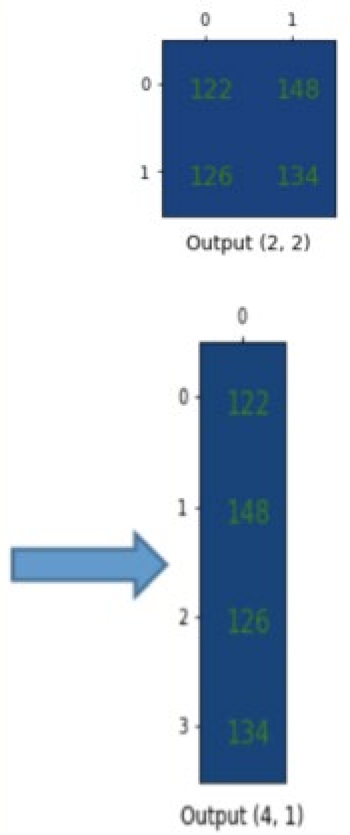
Kernel을 다음과 같이 16 * 4 Matrix로 펼친다.



Input Data를 1 * 16로 reshape

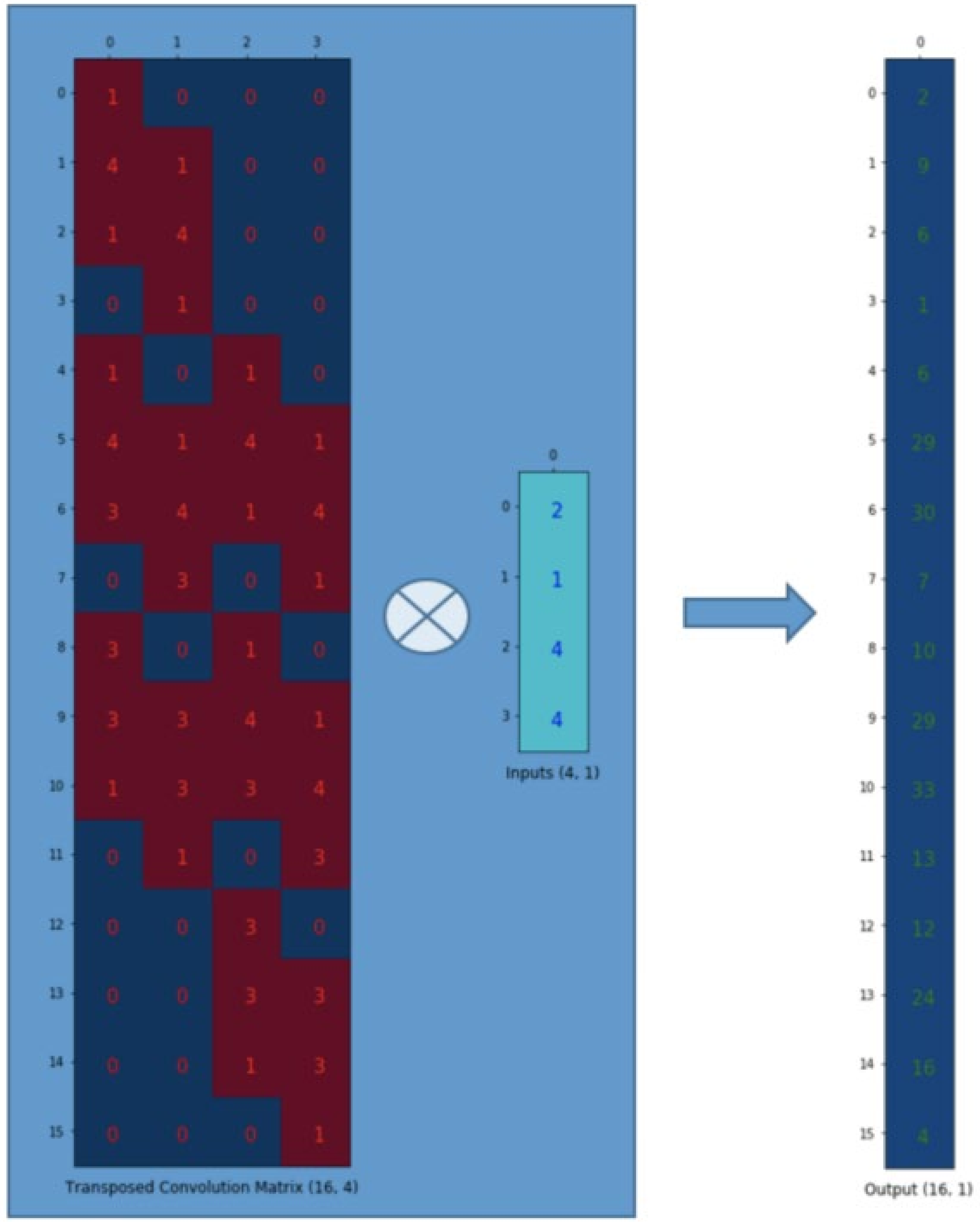
4 * 4 input → 2 * 2 output



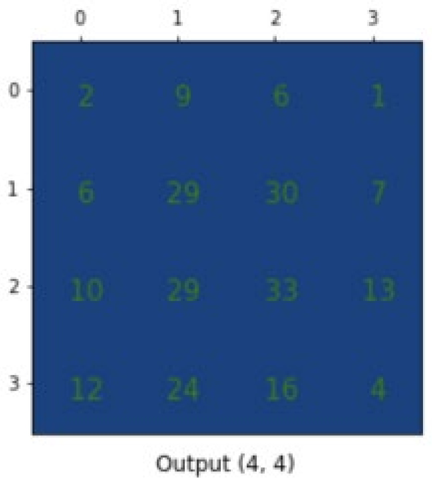


Decoder: Filtering + UpSampling

Transpose Convolutional matrix



Convolution By Matrix Multiplication



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 $\mathbf{X} \in \mathbb{R}^{nm}$

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

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

Hidden unit

Tanh의 함수의 범위

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

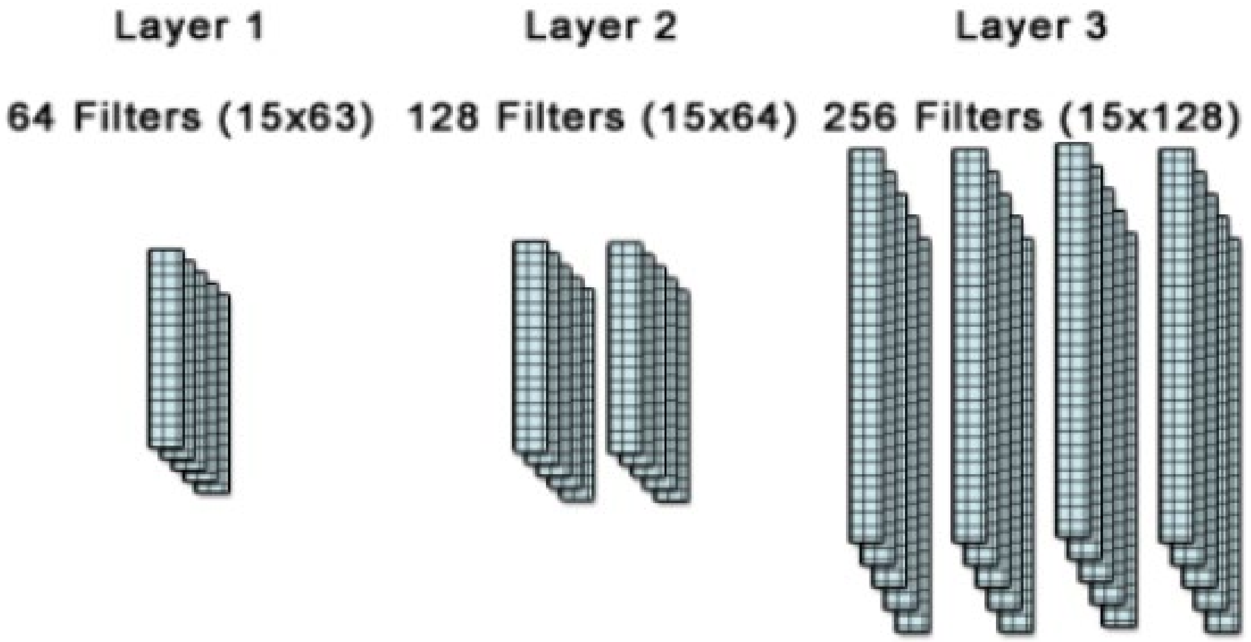


Figure 2: *Structure of the Convolutional Autoencoder. Layer 1 contains 64 filters of size 15x63. Layer 2 contains 128 filters of size 15x64. Layer 3 contains 256 filters of size 15x128. The first dimension of the filter corresponds to a temporal window, while the second dimension corresponds to the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

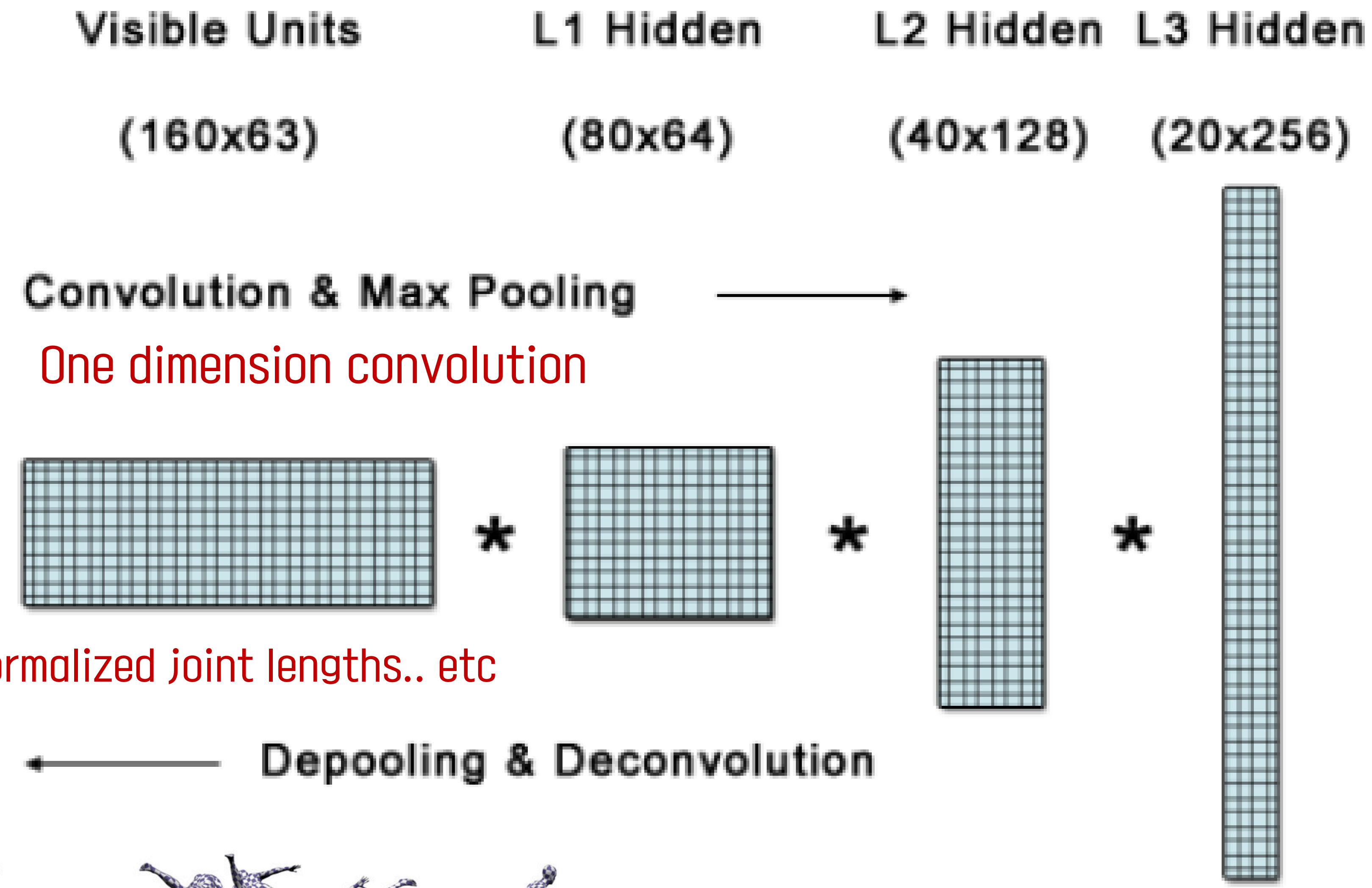
Sub-sampled input data = X

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

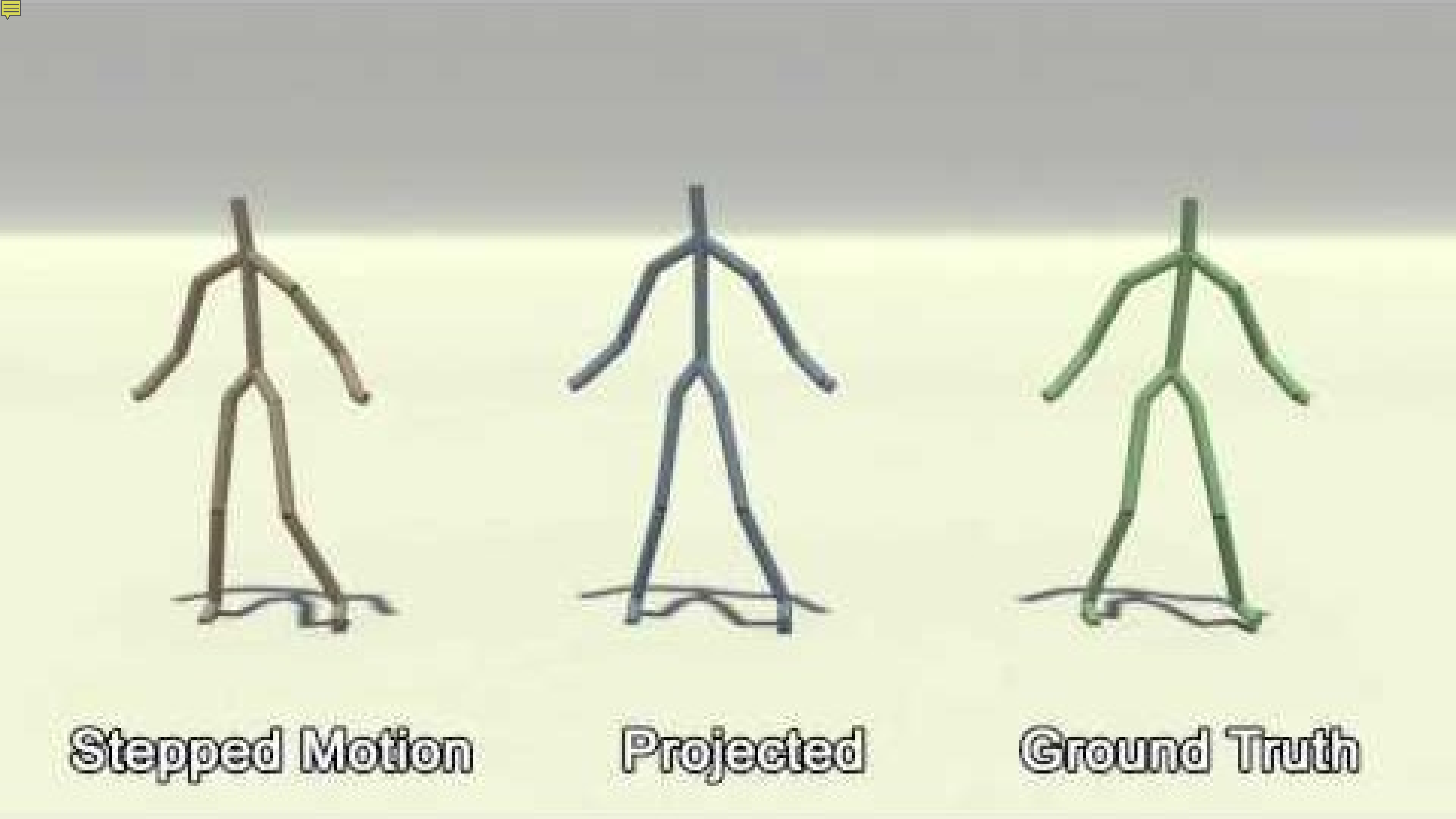
Ground truth data

0.01

$$Loss(\mathbf{X}) = \|\mathbf{X} - \boxed{\mathbf{\Phi}^\dagger(\boxed{\mathbf{\Phi}(\mathbf{X}_c)})}\|_2^2 + \alpha \|\mathbf{\Phi}(\mathbf{X}_c)\|_1$$

Mean Squared Error Loss

노이즈가 첨가된 data


Stepped Motion
Projected
Ground Truth

CHAPTER 03

DL framework for Character Motion synthesis

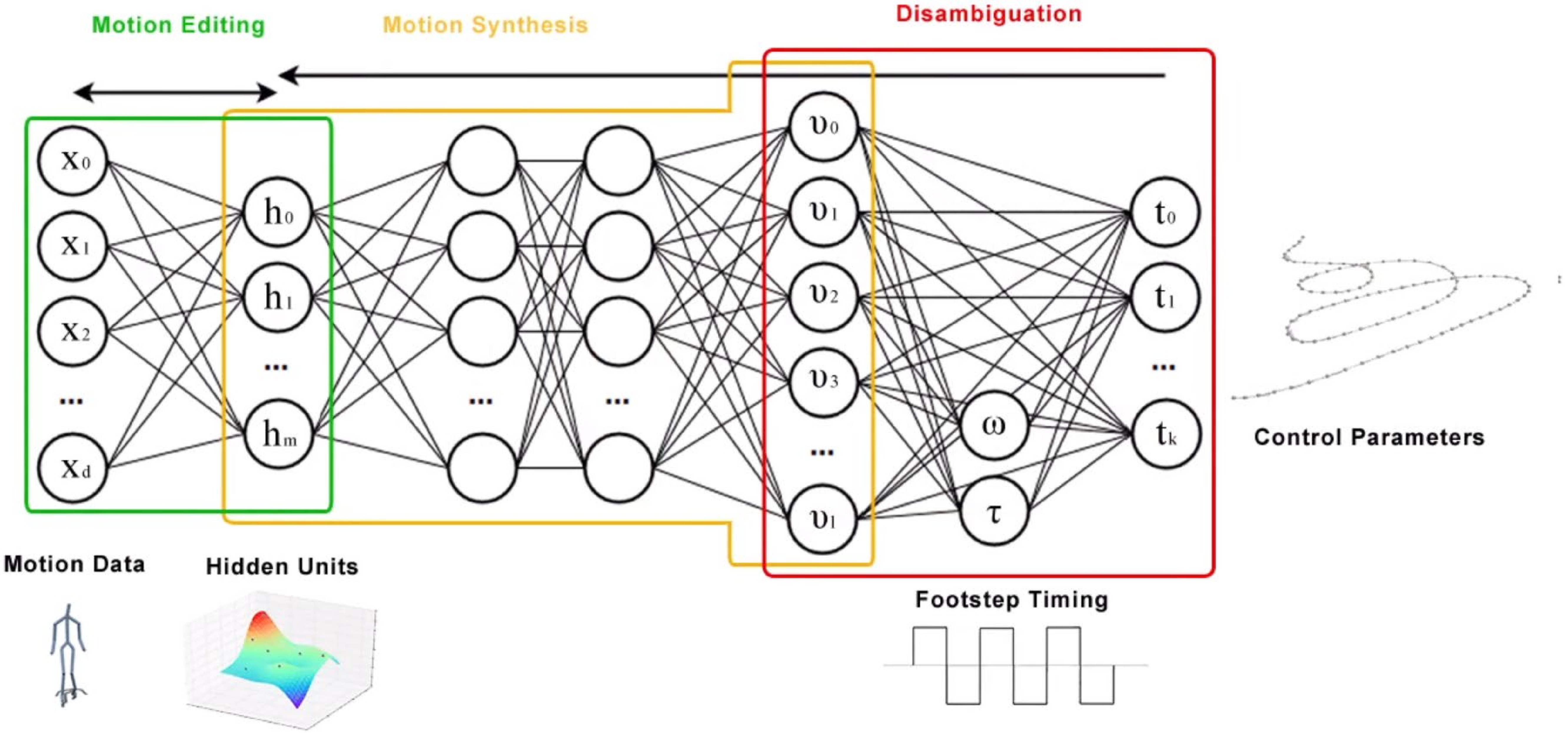
Solved Issues

Hovering Character → Some-how similar to human“walking”

Motion Editing

Overview

Building Motion Manifold
(6hour)
↓



= Visible layer

= High level parameters (trajectory)
= Motion manifold

Building Motion manifold (위 내용과 거의 동일)

Half of second ∵ best result

Max pooling

n(Hidden units) = 256

Temporal filter width = 25

Forward operation: $\boxed{\mathbf{\Phi}(\mathbf{X})} = ReLU(\mathbf{\Psi}(\mathbf{X} * \mathbf{W}_0 + \mathbf{b}_0))$ $\mathbf{W}_0 \in \mathbb{R}^{m \times d \times w_0}$

Degree of freedom = 70

Hidden Unit

Backward operation: $\boxed{\mathbf{\Phi}^{\dagger}(\mathbf{H})} = (\mathbf{\Psi}^{\dagger}(\mathbf{H}) - \mathbf{b}_0) * \tilde{\mathbf{W}}_0.$ $\mathbf{H} \in \mathbb{R}^{\frac{n}{2} \times m}$ n = 240, m = 256

0.1

Cost function: $Cost(\mathbf{X}, \theta) = \|\mathbf{X} - \mathbf{\Phi}^{\dagger}(\mathbf{\Phi}(\mathbf{X}))\|_2^2 + \alpha \, \|\theta\|_1$

Additional sparsity term:
ensure min num of network parameter

- ★ GD algorithm: Adam
- ★ Dropout to avoid overfitting

Structure of the Feedforward Network

$h_1, h_2 \quad w_1, w_2, w_3 \quad l$

64, 128, 45, 25, 15 and 7,

$\mathbf{T} \in \mathbb{R}^{n \times k}$

Contact: 1, Else: -1

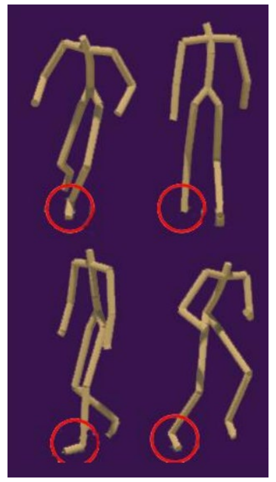
$\mathbf{F} \in \{-1, 1\}^{n \times 4}$

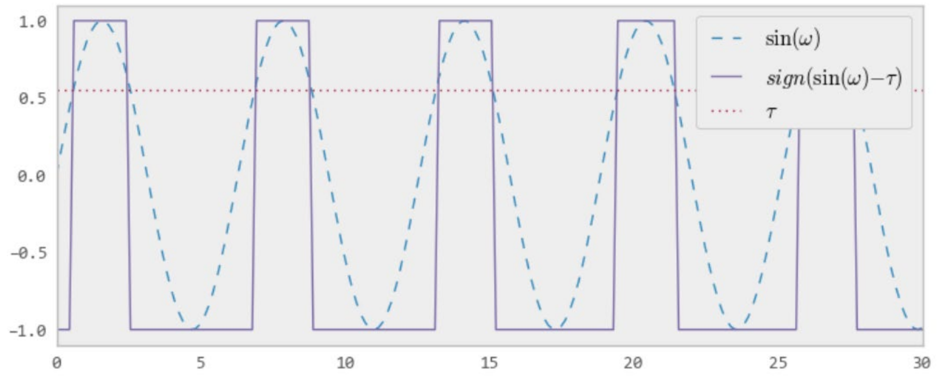
$$\boxed{\boldsymbol{\Pi}(\mathbf{T})} = ReLU(\boldsymbol{\Psi}(ReLU(ReLU(\boxed{\boldsymbol{\Upsilon}(\mathbf{T})} * \mathbf{W}_1 + \mathbf{b}_1) * \mathbf{W}_2 + \mathbf{b}_2) * \mathbf{W}_3 + \mathbf{b}_3)), \quad (4)$$

$$\boldsymbol{\Upsilon}(\mathbf{T}) = [\mathbf{T} \;\; \mathbf{F}]$$

where $\mathbf{W}_1 \in \mathbb{R}^{h_1 \times l \times w_1}$, $\mathbf{b}_1 \in \mathbb{R}^{h_1}$, $\mathbf{W}_2 \in \mathbb{R}^{h_2 \times h_1 \times w_2}$, $\mathbf{b}_2 \in \mathbb{R}^{h_2}$, $\mathbf{W}_3 \in \mathbb{R}^{m \times h_2 \times w_3}$, $\mathbf{b}_3 \in \mathbb{R}^{m}$, h_1, h_2 are the

$$Cost(\mathbf{T}, \mathbf{X}, \phi) = \|\mathbf{X} - \boldsymbol{\Phi}^{\dagger}\boxed{\boldsymbol{\Pi}(\mathbf{T})}\|_2^2 + \alpha\, \|\phi\|_1$$

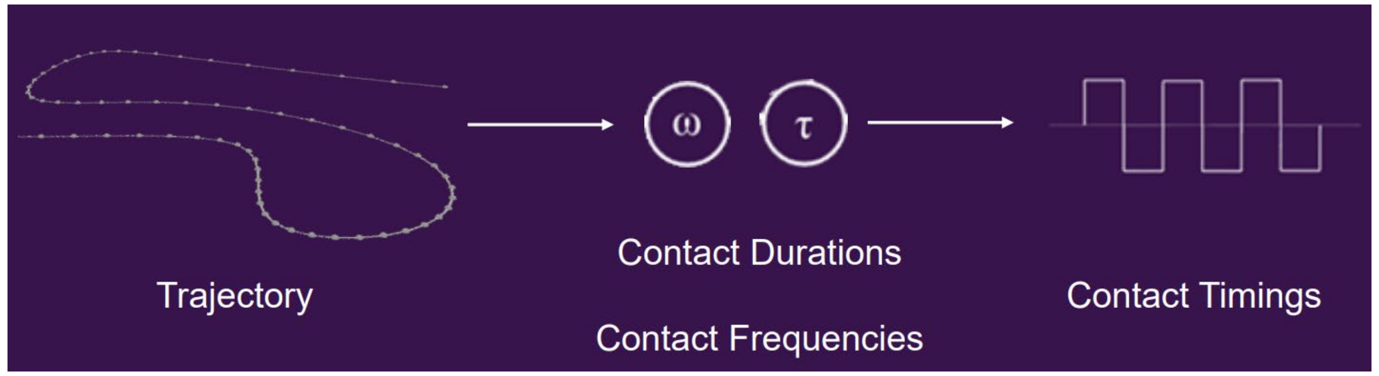




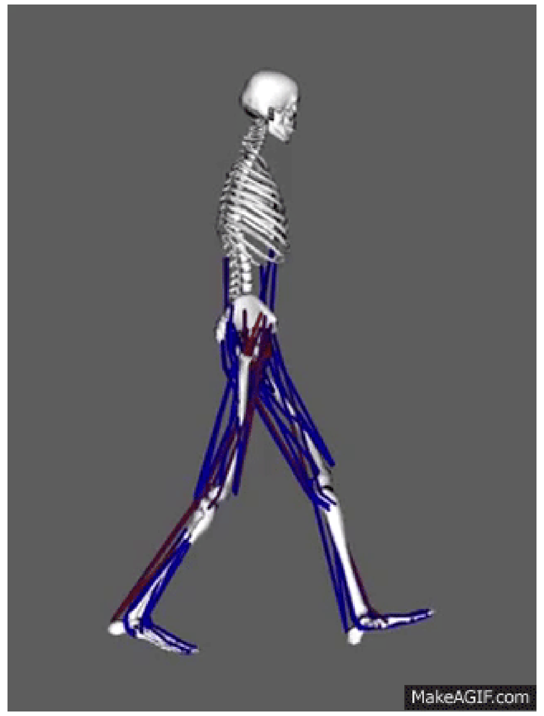
where $\mathbf{F} \in \{-1, 1\}^{n \times 4}$ is a matrix that represents the contact states of left heel, left toe, right heel, and right toe at each frame, and

$$\mathbf{F}(\omega, \tau) = \begin{bmatrix} sign(\sin(c\,\omega + a^h) - b^h - \tau^{lh}) \\ sign(\sin(c\,\omega + a^t) - b^t - \tau^{lt}) \\ sign(\sin(c\,\omega + a^h + \pi) - b^h - \tau^{rh}) \\ sign(\sin(c\,\omega + a^t + \pi) - b^t - \tau^{rt}) \end{bmatrix}^{\mathsf{T}}$$

where ω and τ control the *frequency* and *step duration* at each frame



Structure of the Feedforward Network



Gait cycle (보행 주기)

보행 주기의 나쁜 예(?):

Contact frequency: $\omega_i = \Delta\omega_i + \Delta\omega_{i-1} + \ldots + \Delta\omega_0 \quad \Delta\omega_i = \frac{\pi}{L_i}$ wavelength of the steps

Contact duration: $\tau_i = \cos\dfrac{\pi d_i}{u_i + d_i}.$ of the number of frames with the foot up u_i over the number of frames with the foot down d_i.

$$\text{matrix } \mathbf{\Gamma} = \{\tau^{lh}, \tau^{lt}, \tau^{rh}, \tau^{rt}, \mathbf{\Delta}\omega\}$$

Locomotion Path

↓

$$\mathbf{\Gamma}(\mathbf{T}) = ReLU(\mathbf{T} * \mathbf{W}_4 + \mathbf{b}_4) * \mathbf{W}_5 + \mathbf{b}_5$$

$w_4, w_5 \ h_4 \ k, l$

3 and 5

$$\mathbf{W}_4 \in \mathbb{R}^{h_4 \times k \times w_4},\ \mathbf{b}_4 \in \mathbb{R}^{h_4},\ \mathbf{W}_5 \in \mathbb{R}^{l \times h_4 \times w_5},\ \mathbf{b}_5 \in \mathbb{R}^{l}$$

Foot contact information

↓

Network trained → $\mathbf{F}(\omega, \tau) = \begin{bmatrix} sign(\sin(c\,\omega + a^h) - b^h - \tau^{lh}) \\ sign(\sin(c\,\omega + a^t) - b^t - \tau^{lt}) \\ sign(\sin(c\,\omega + a^h + \pi) - b^h - \tau^{rh}) \\ sign(\sin(c\,\omega + a^t + \pi) - b^t - \tau^{rt}) \end{bmatrix}^{\mathsf{T}}$

Motion Editing

Apply constraints in the hidden space

Positional Constraints:

↑

Fixing foot sliding

$$Pos(\mathbf{H}) = \sum_j \|\mathbf{v}_r^{\mathbf{H}} + \omega^{\mathbf{H}} \times \mathbf{p}_j^{\mathbf{H}} + \mathbf{v}_j^{\mathbf{H}} - \mathbf{v}_j'\|_2^2.$$

Bone Length Constraints:

↑

Preserve rigidity

$$Bone(\mathbf{H}) = \sum_i \sum_b |\|\mathbf{p}_{b_{j1}}^{\mathbf{H}i} - \mathbf{p}_{b_{j2}}^{\mathbf{H}i}\| - l_b|^2$$

Trajectory Constraints:

↑

Constrain motion into precise trajectory

$$Traj(\mathbf{H}) = \|\omega^{\mathbf{H}} - \omega'\|_2^2 + \|\mathbf{v}_r^{\mathbf{H}} - \mathbf{v}_r'\|_2^2$$

$$\mathbf{H}' = arg \min_{\mathbf{H}} \; Pos(\mathbf{H}) + Bone(\mathbf{H}) + Traj(\mathbf{H}).$$

Motion Editing

Apply style in the hidden unit values which produce Gram matrix

Gram Matrix에 대한 짧은 설명 by 홍교수님

http://blog.naver.com/PostView.nhn?blogId=atelierjpro&logNo=221180412283

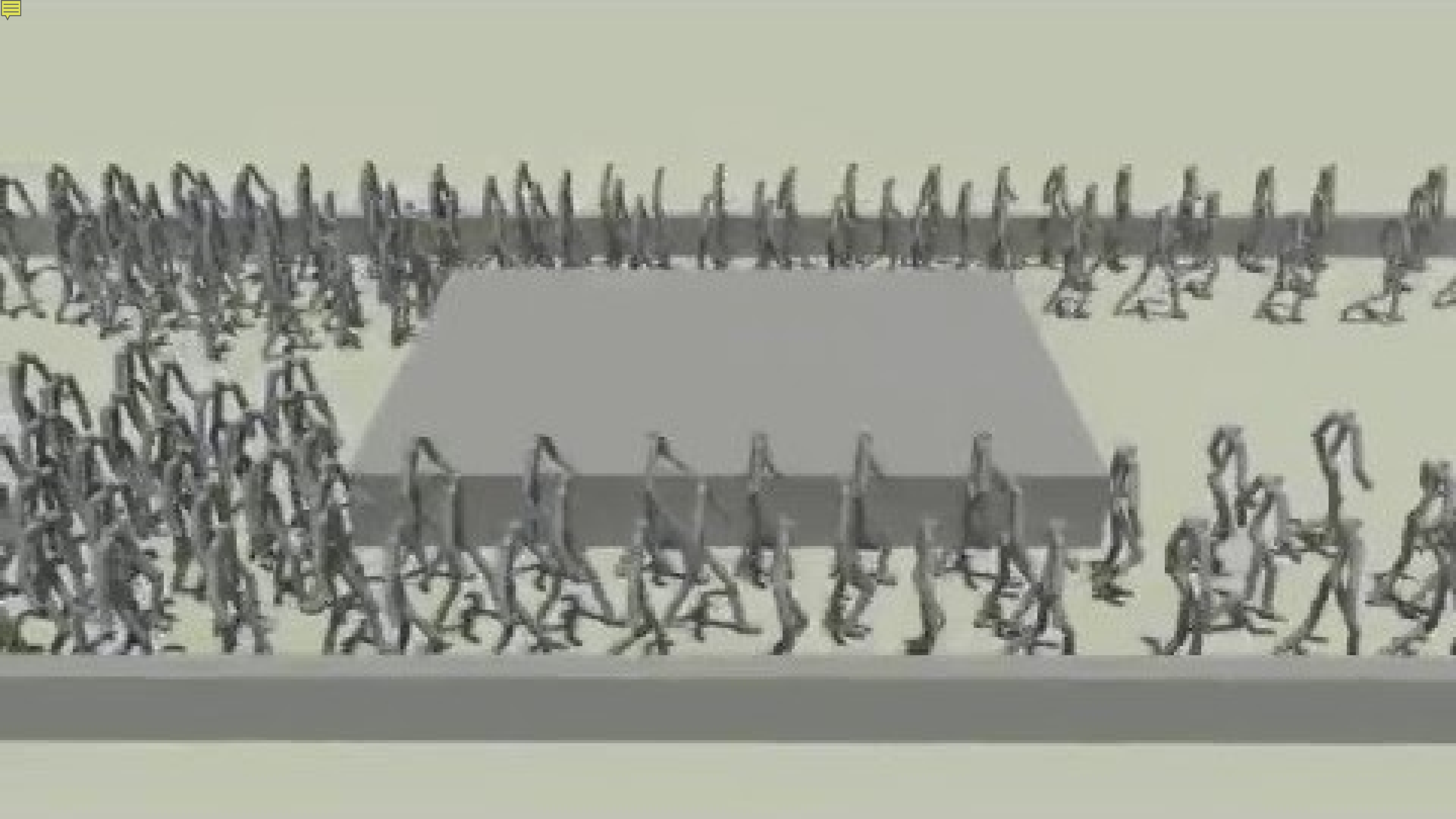
Style 상관계수 = 1.0

Content 상관계수 = 0.01

$$Style(\mathbf{H}) = s\|G(\mathbf{\Phi}(\mathbf{S})) - G(\mathbf{H})\|_2^2 + c\|\mathbf{\Phi}(\mathbf{C}) - \mathbf{H}\|_2^2$$

Compute Gram matrix

$$G(\mathbf{H}) = \frac{\sum_i^n \mathbf{H}_i \mathbf{H}_i^\intercal}{n}.$$



THANK YOU EVERYONE