DAY 6

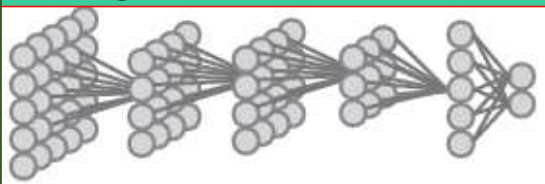
RNN/LSTM/GRU 이론 실습
심화과정

0. 복습

Deep Learning
신경망의 70년

Day4 학습 목표

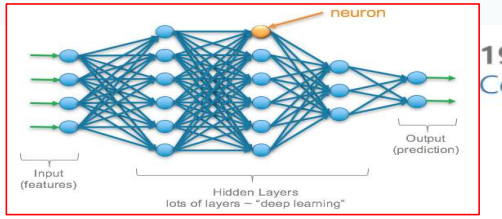
Image classification



CNN Model

Day2 학습 목표

regression, classification



1998년
Convolution Neural Network (CNN)

얀 르쿤

2016년
DeepMind AlphaGo

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

2006년
Deep Learning

제프리 힌튼

Day6 학습 목표 : 심화

2012년
AlexNet
IMAGENET

Day3 학습 목표

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

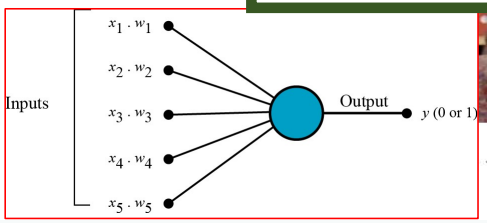
19C-20C 초반
생체 신경망

이반 파블로프

Day1 학습 목표

1956년
Perceptron

로젠블랫



regression, classification

1982년
Recurrent Neural Network (RNN)

홉필드

Day5 학습 목표

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

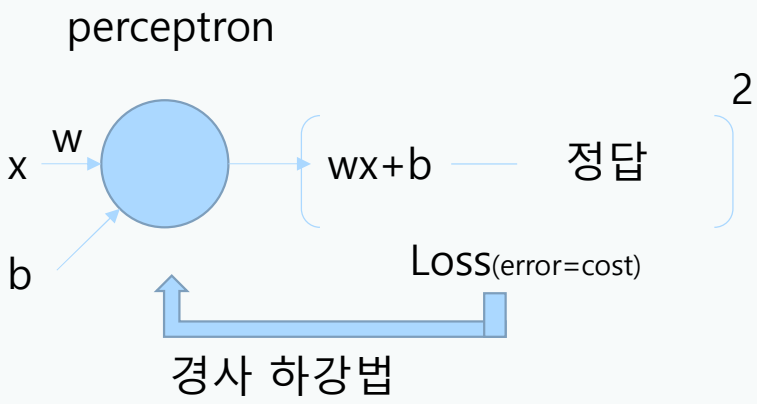
2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

진도상황에 따라 변경될 수 있음

0. 복습

1. Perceptron

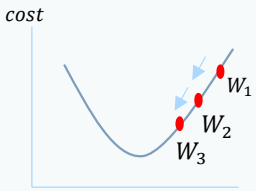


$$(WX - Y)^2$$
$$= W^2X^2 - 2WXY + Y^2$$
$$= AW^2 - BW + C$$
$$= \text{cost(error)}$$

기울기

$$W_{update} = W - \alpha \frac{\partial}{\partial W} cost(W)$$

러닝 레이트



Regression

출력 함수 = None

loss = mse

Binary classification

출력함수 = sigmoid

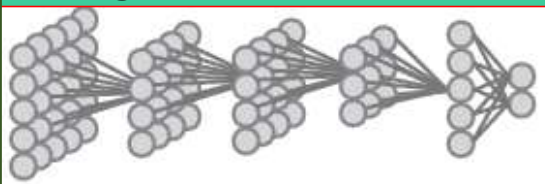
loss = binary crossentropy

0. 복습

Deep Learning
신경망의 70년

Day4 학습 목표

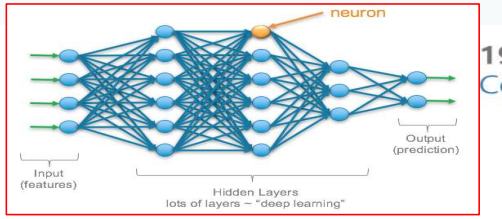
Image classification



CNN Model

Day2 학습 목표

regression, classification



1998년
Convolution Neural Network (CNN)

얀 르쿤

2016년
DeepMind AlphaGo

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

2006년
Deep Learning

제프리 힌튼

Day6 학습 목표 : 심화

2012년
AlexNet
IMAGENET

Day3 학습 목표

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

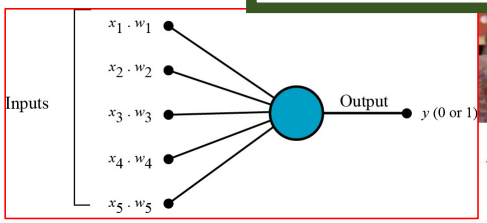
19C-20C 초반
생체 신경망

이반 파블로프

Day1 학습 목표

1956년
Perceptron

로젠블랫



1982년
Recurrent Neural Network (RNN)

홉필드

Day5 학습 목표

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

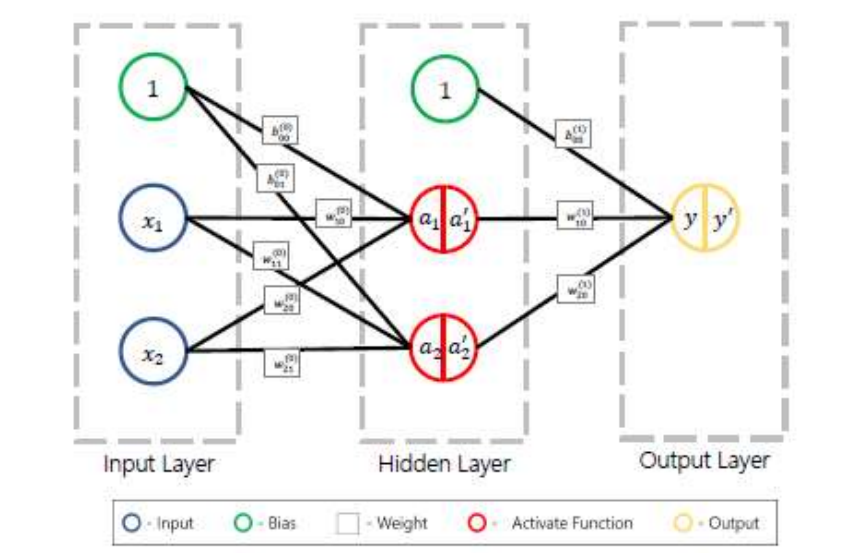
이안 굿펠로우

regression, classification

진도상황에 따라 변경될 수 있음

0. 복습

2. Multi Layer Perceptron



다층 구조가 가능 하기 위해

activation function 사용 = sigmoid

hidden layer의 역할

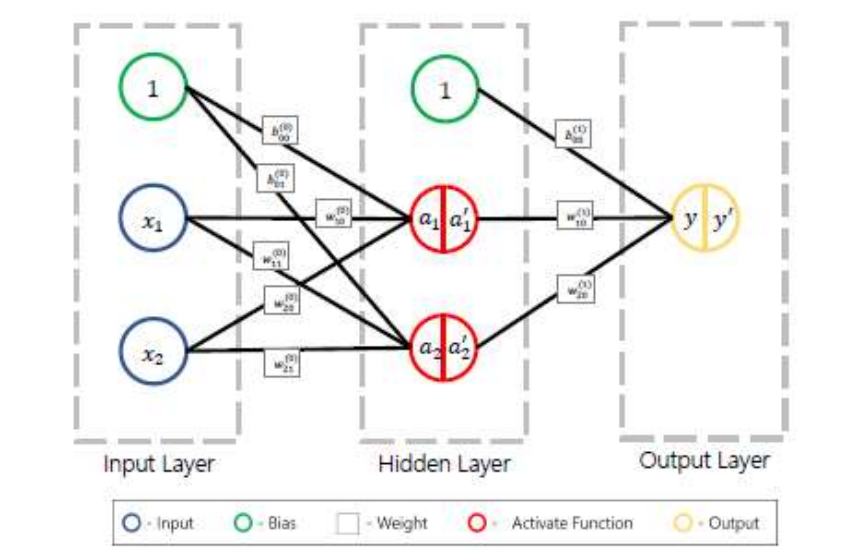
output layer가 잘 동작하도록 입력 데이터를 representation 해주는 역할

입력 차원 : (batch size, 데이터 변수)

모델 학습 알고리즘 : backpropagation

0. 복습

2. Multi Layer Perceptron



Regression

출력 함수 = None

loss = mse

Binary classification

출력함수 = sigmoid

loss = binary crossentropy

Categorical classification

출력함수 = softmax

loss =categorical crossentropy

(one hot encording 사용)

0. 복습

2. Multi Layer Perceptron

다층 구조가 가능 하기 위해

여기서 hidden layer의 activation function과 output layer의 activation function은 다르다!

Hidden layer의 activation function

1. 여러 layer를 쌓는 효과를 보기위해 비선형적이어야함.
2. Layer들의 w,b가 잘 업데이트 되기 위해 오차 역전파에서 오차의 미분값이 잘 전달되도록해야함
3. 미분 가능해야함

Output layer의 activation function은

Task에 따라 선정.

Regression 의 activation function은 None (regression은 출력값이 숫자값으로 입력*w +b값이 예측값)

Classification의 activation function은 sigmoid or softmax (classification은 0 or 1의 제한된 출력)

0. 복습

2. Multi Layer Perceptron 다층 구조가 가능 하기 위해

Multi Layer Perceptron(=Neural Network) 구조 할때

Input layer : 입데이터 전달 받는 역할

Hidden Layer : 데이터를 representation

Output Layer : representation 을 전달받아 regression/classification

의 역할을 하도록 설계.

Day1 학습 목표

즉, 위와 같이 설계하여 loss를 0으로 수렴시키도록 역전파 알고리즘으로 학습하면

Hidden Layer의 모델 W와 B는 Out put layer가 지정된 task를 잘 할수 있게 representation되도록 학습되어 짐

Output Layer의 w와 b는 지정된 task를 잘하도록 학습 되어 짐.

즉, 결론적으로 학습이 잘되면 각 layer들은 각각의 역할을 잘 수행한다고 해석함.

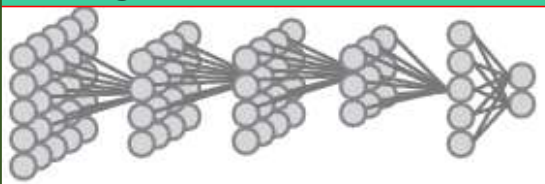
학습이 잘 안되면 위의 역할을 잘 수행 못한다고 해석함.

0. 복습

Deep Learning
신경망의 70년

Day4 학습 목표

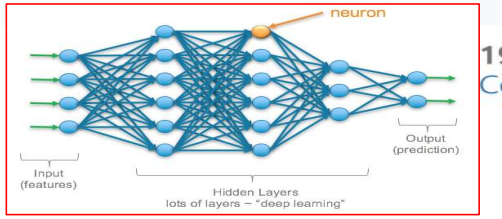
Image classification



CNN Model

Day2 학습 목표

regression, classification



1998년
Convolution Neural Network (CNN)

얀 르쿤

2016년
DeepMind AlphaGo

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

2006년
Deep Learning

제프리 힌튼

Day6 학습 목표 : 심화

2012년
AlexNet
IMAGENET

Day3 학습 목표

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

19C-20C 초반
생체 신경망

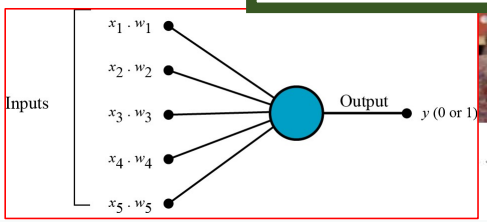
이반 파블로프

1956년
Perceptron

로젠블랫

Day1 학습 목표

1982년
Recurrent Neural Network (RNN)



홉필드

Day5 학습 목표

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

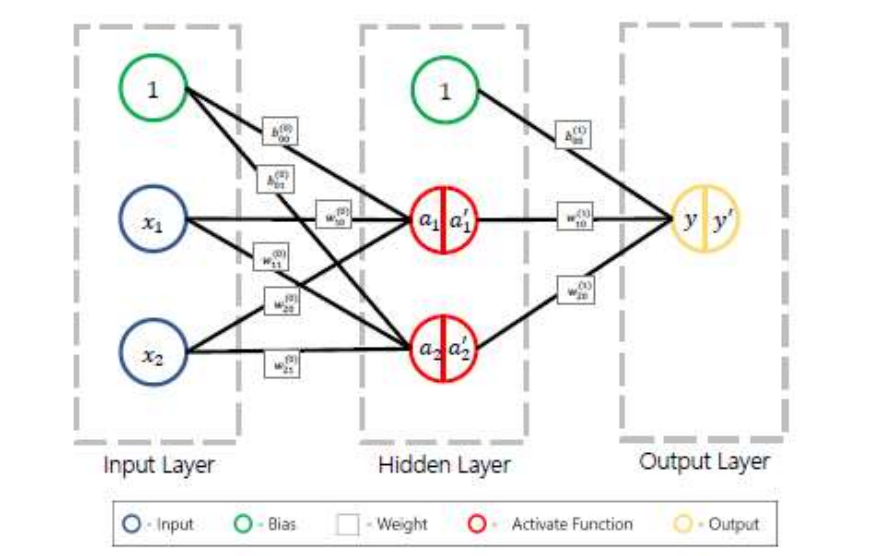
이안 굿펠로우

regression, classification

진도상황에 따라 변경될 수 있음

0. 복습

2. Multi Layer Perceptron

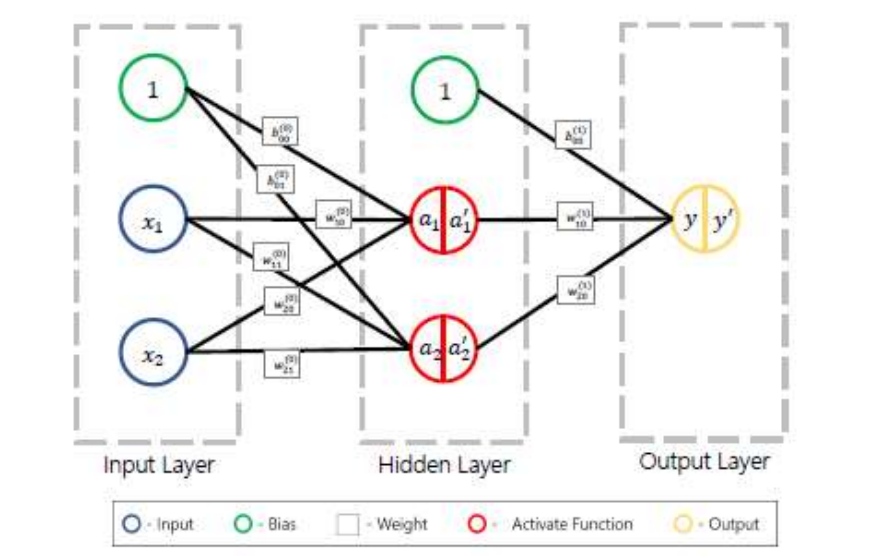


Multi Layer Percpetron 문제

1. 오비피팅
2. 느린 학습 속도
3. local minima
4. vanishing gradient

0. 복습

2. Multi Layer Perceptron



Multi Layer Percpetron 대표적 해결법

1. batchnormalization
2. dropout
3. regulization
4. small model
5. data augmentation
6. 새로운 activation 함수
7. 좋은 optimizer

0. 복습

3. 학습에서 중요한것

데이터를 모을 때 현장과 비슷하고 발생할 모든 케이스의 데이터를 많이 모아야함.

학습/검증/시험 데이터 나눠서 학습시 성능 확인을 하고 오버피팅 확인을 해야함

데이터를 나눌때 분포가 고르게 되어야함

학습 모델 웨이트가 고루 잘 퍼져잇는게 일반적으로 잘 학습되었다고 함.

러닝 레이트, 모델 파라미터 초기값, 모델 설계, batch size, epoch 등이 중요

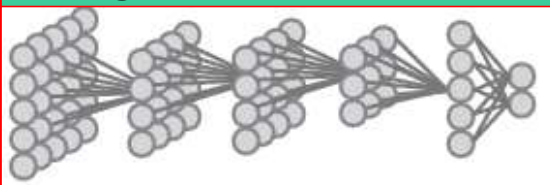
검증데이터는 학습시 모델 업데이트에 적용되지 않음

0. 복습

Deep Learning
신경망의 70년

Day4 학습 목표

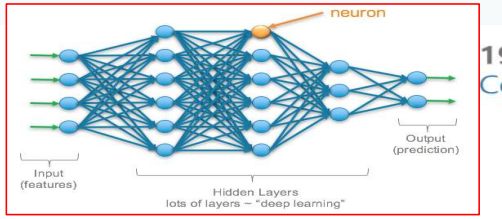
Image classification



CNN Model

Day2 학습 목표

regression, classification



1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

1998년
Convolution Neural Network (CNN)

얀 르쿤

2006년
Deep Learning

제프리 힌튼

2016년
DeepMind AlphaGo

Day6 학습 목표 : 심화

2012년
AlexNet
IMAGENET

Day3 학습 목표

1차 암흑기 (1974-1980)

2차 암흑기 (1987-1993)

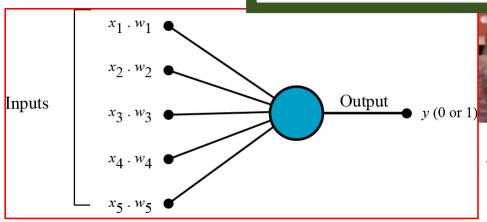
19C-20C 초반
생체 신경망

이반 파블로프

Day1 학습 목표

1956년
Perceptron

로젠블랫



regression, classification

1982년
Recurrent Neural Network (RNN)

홉필드

Day5 학습 목표

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

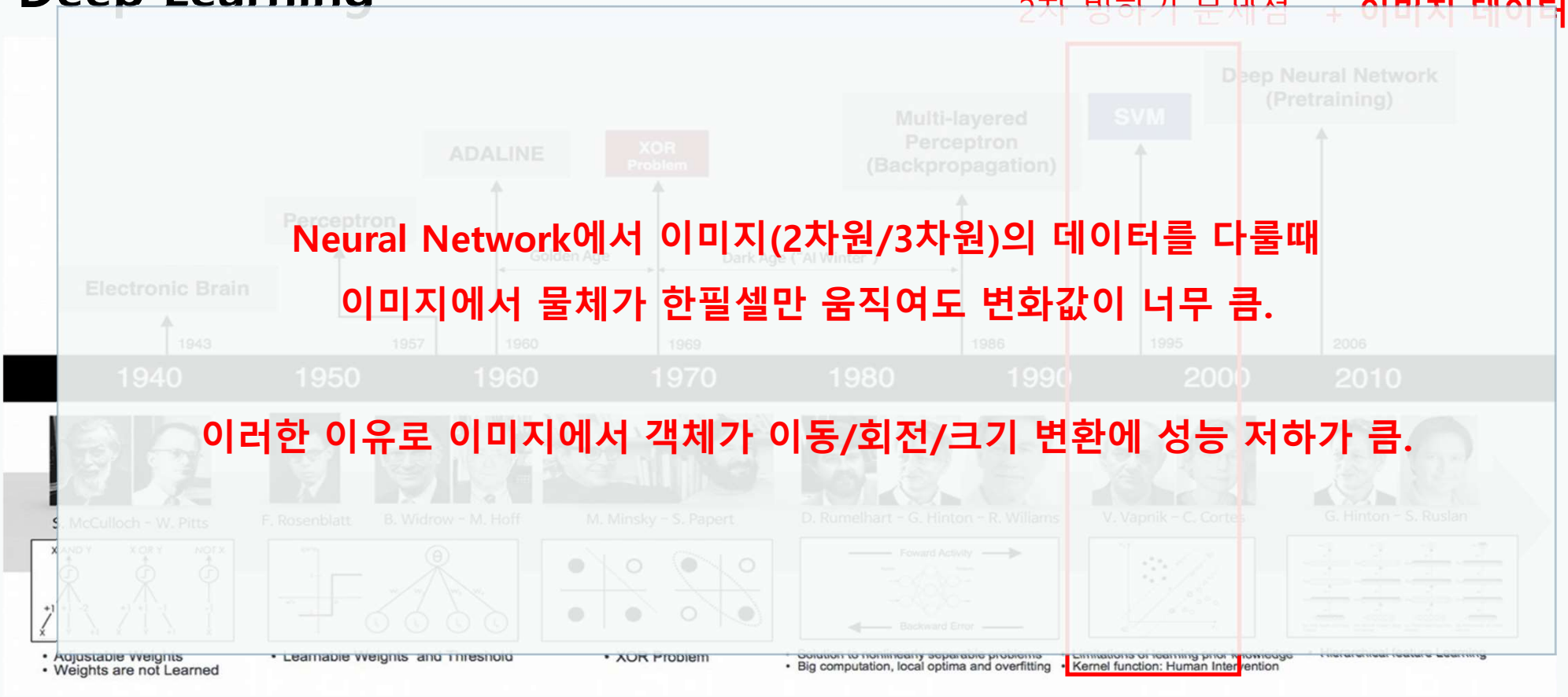
이안 굿펠로우

진도상황에 따라 변경될 수 있음

0. 복습

Deep Learning

Neural Network 의 2차 방학기 문제점 + 이미지 다루기 + 이미지 데이터 문제



Neural Network에서 이미지(2차원/3차원)의 데이터를 다룰때

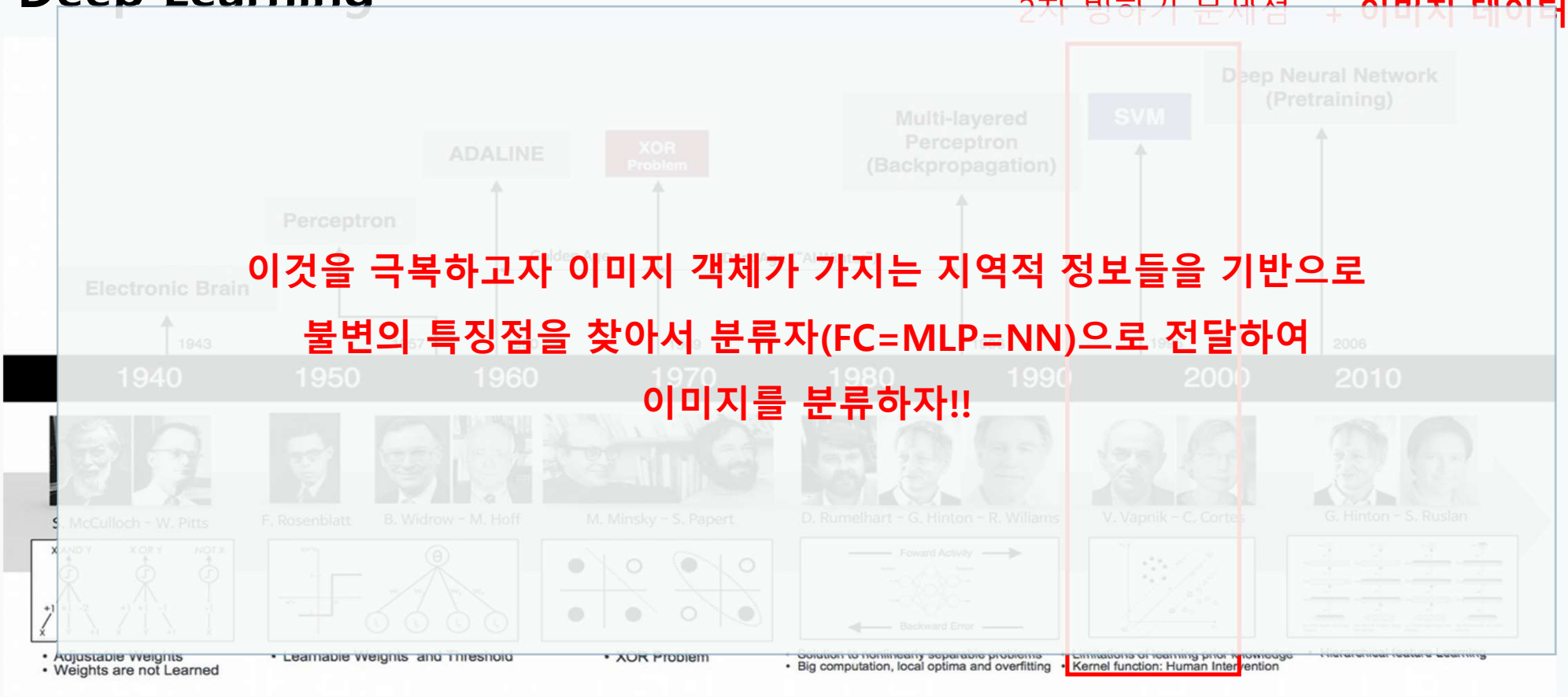
이미지에서 물체가 한필셀만 움직여도 변화값이 너무 큼.

이러한 이유로 이미지에서 객체가 이동/회전/크기 변환에 성능 저하가 큼.

0. 복습

Deep Learning

Neural Network 의 2차 빙하기 문제점 + 이미지 다루기 + 이미지 데이터 문제



이것을 극복하고자 이미지 객체가 가지는 지역적 정보들을 기반으로

불변의 특징점을 찾아서 분류자(FC=MLP=NN)으로 전달하여

이미지를 분류하자!!

0. 복습

Deep Learning

Neural Network 의 2차 빙하기 문제점 + 이미지 다루기 + 이미지 데이터 문제

기존에는 HAND CRAFT라는 사람이 설계한

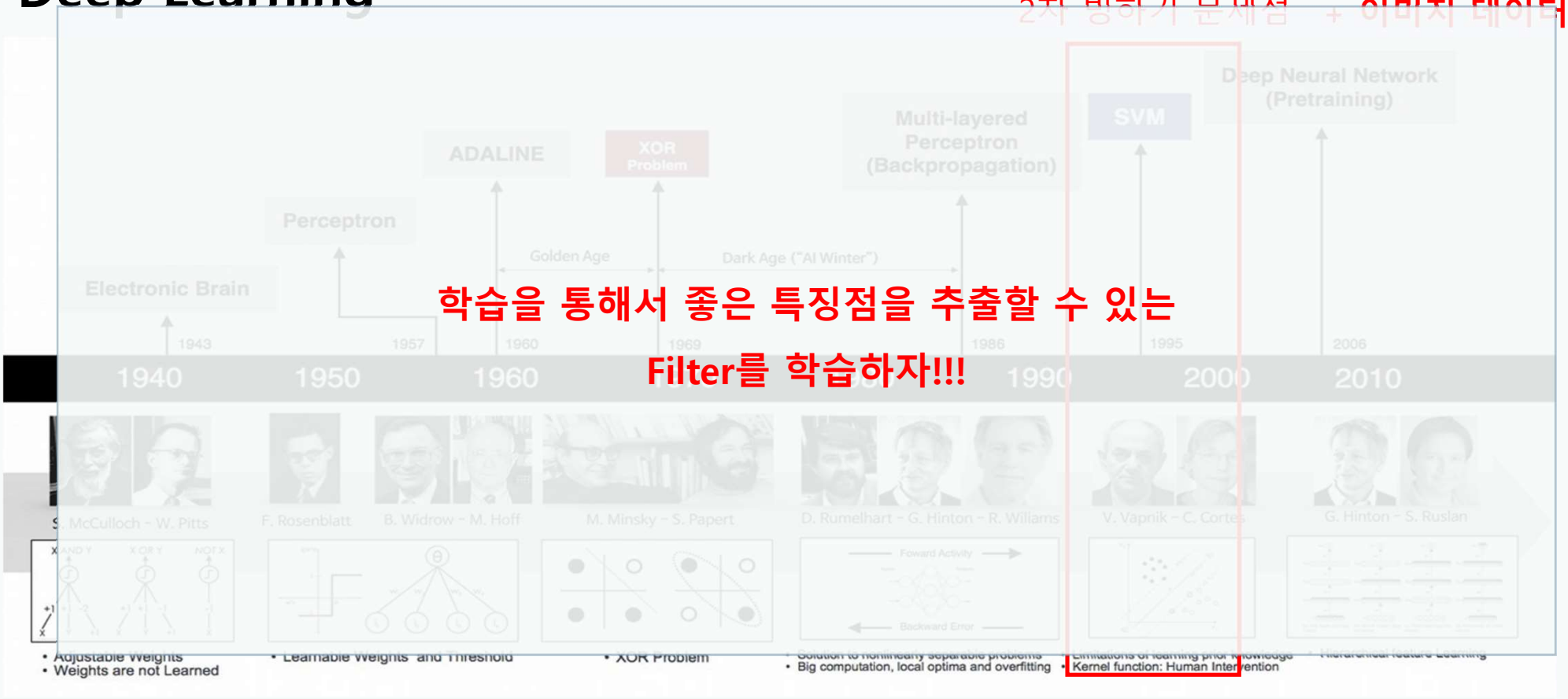
Filter로 특징점을 추출!!!

하지만 성능이 만족스럽지 못함.

0. 복습

Deep Learning

Neural Network 의 2차 빙하기 문제점 + 이미지 다루기 + 이미지 데이터 문제



학습을 통해서 좋은 특징점을 추출할 수 있는

Filter를 학습하자!!!

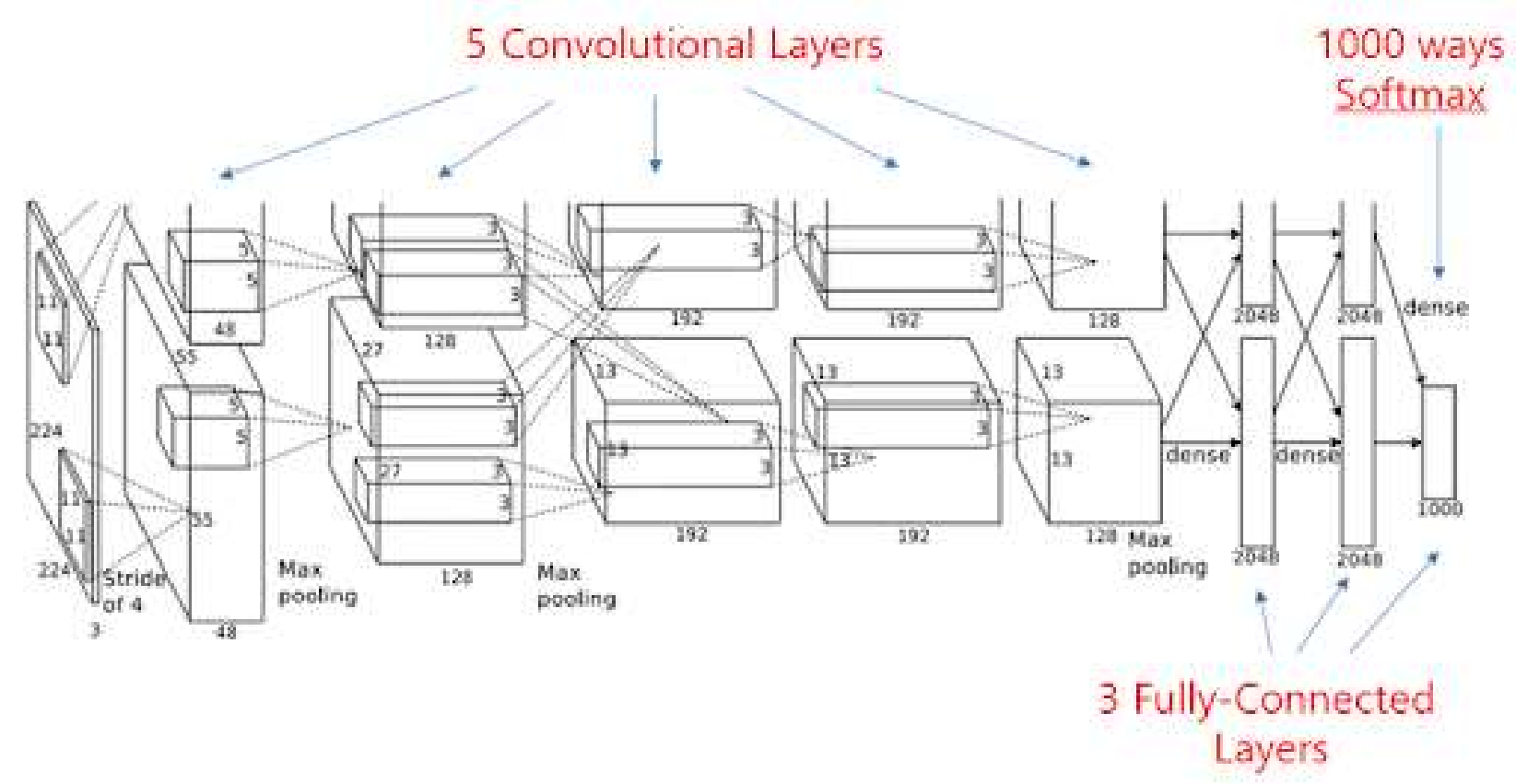
0. 복습

Convolutional Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



ALEX NET

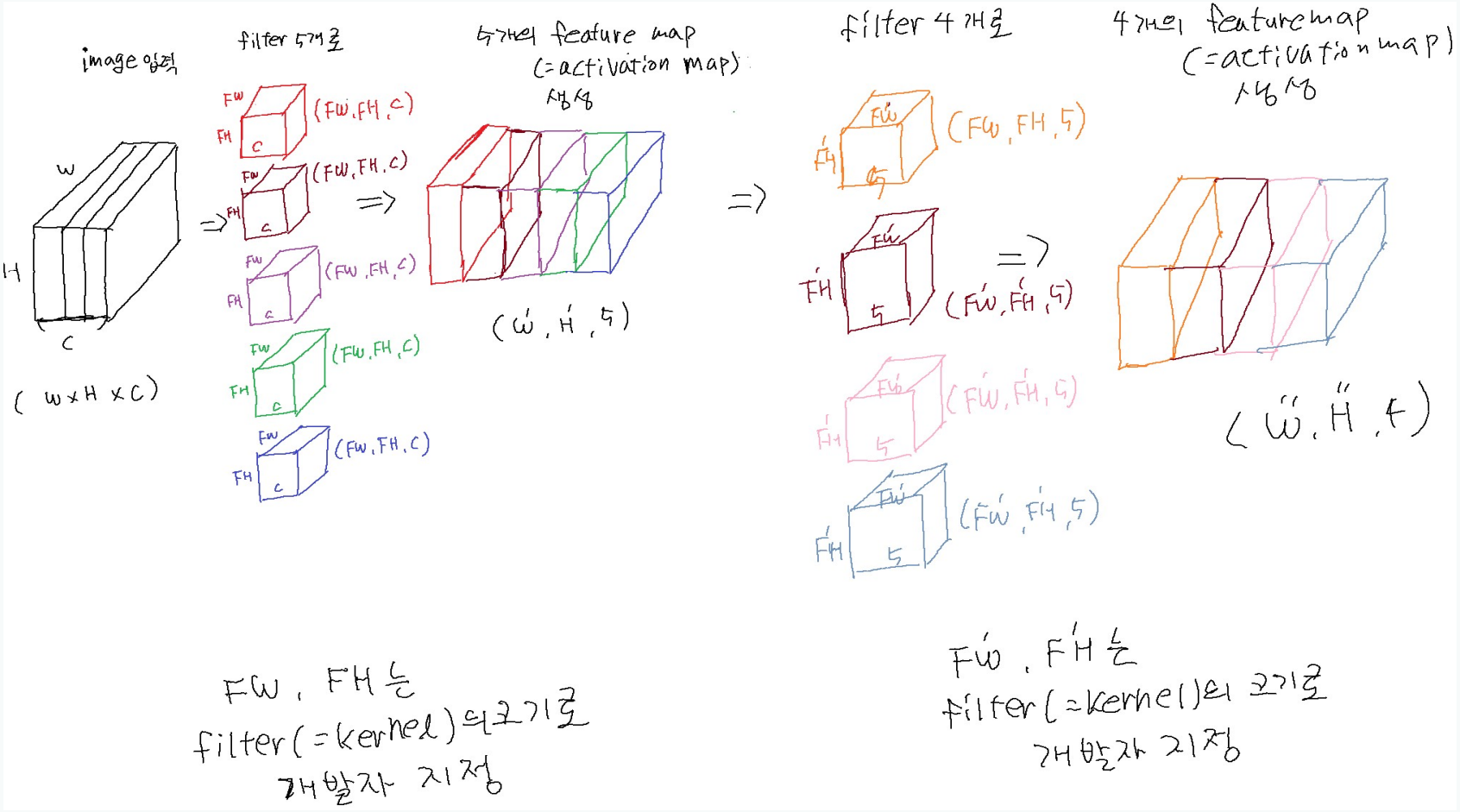
0. 복습

Convolutional Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



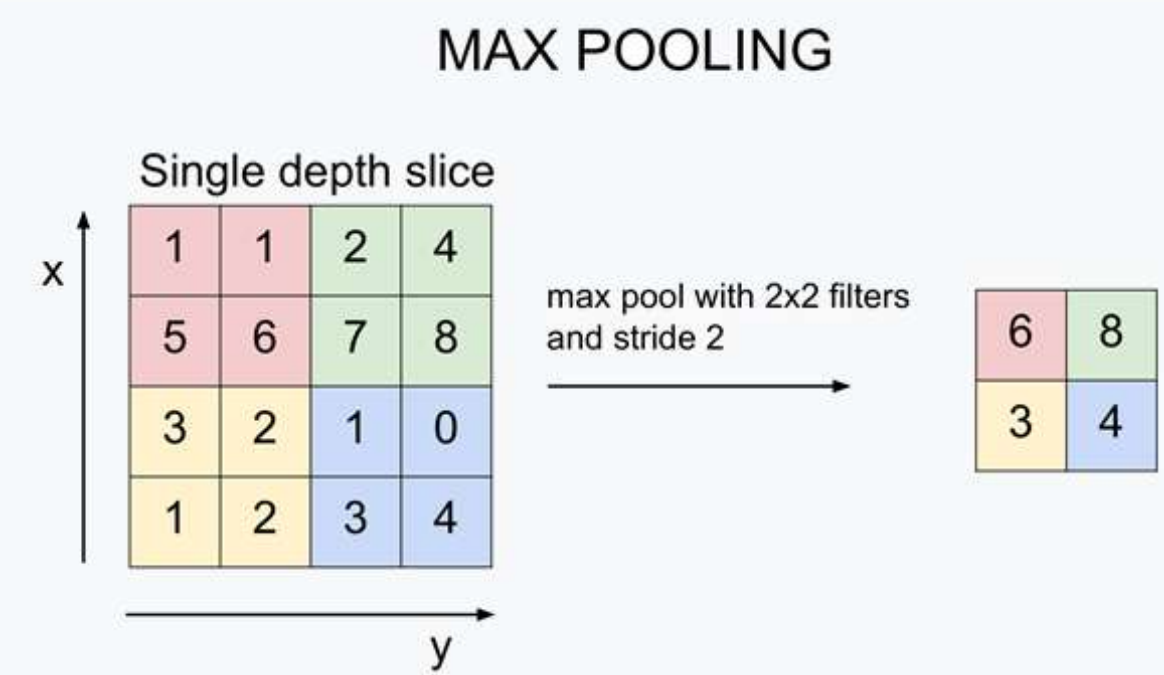
0. 복습

Convolutional Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



즉 filter(=kernel) 을 통해 feature을 만들었다면
만들어진 feature map에서 큰 특징을 가지는 값만 모아서
새로운 feature map을 만드는 과정

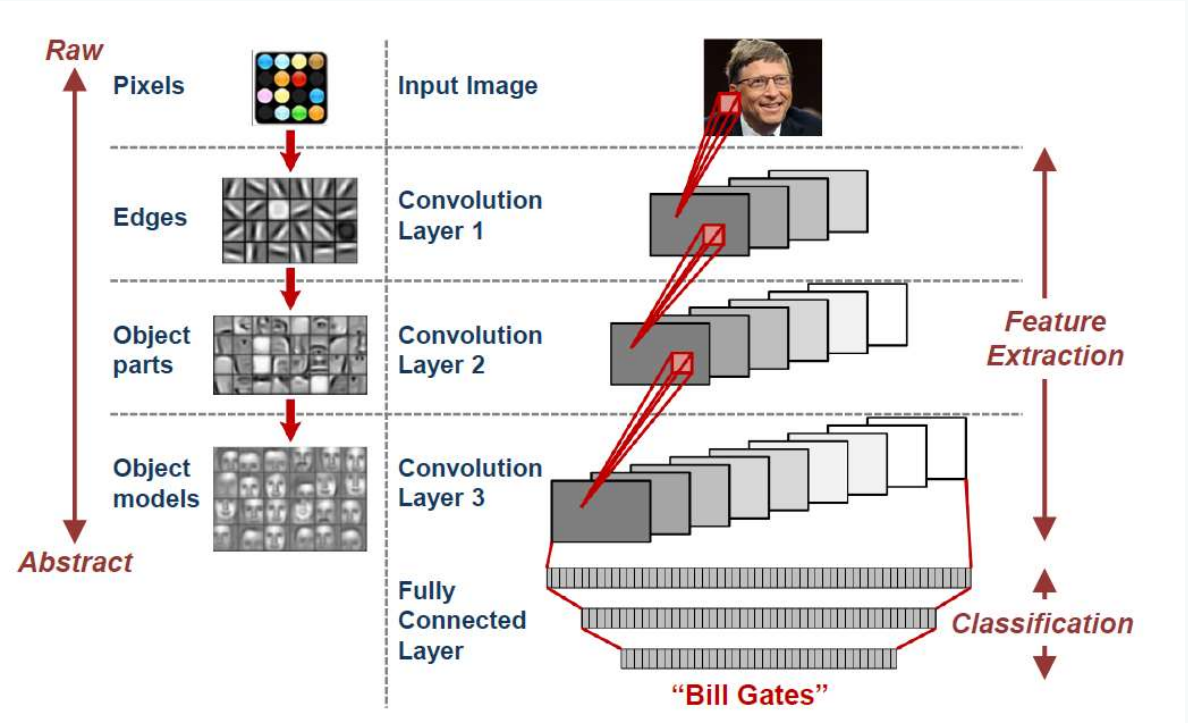
0. 복습

Convolutional Neural Network

Convolution Layer(conv)

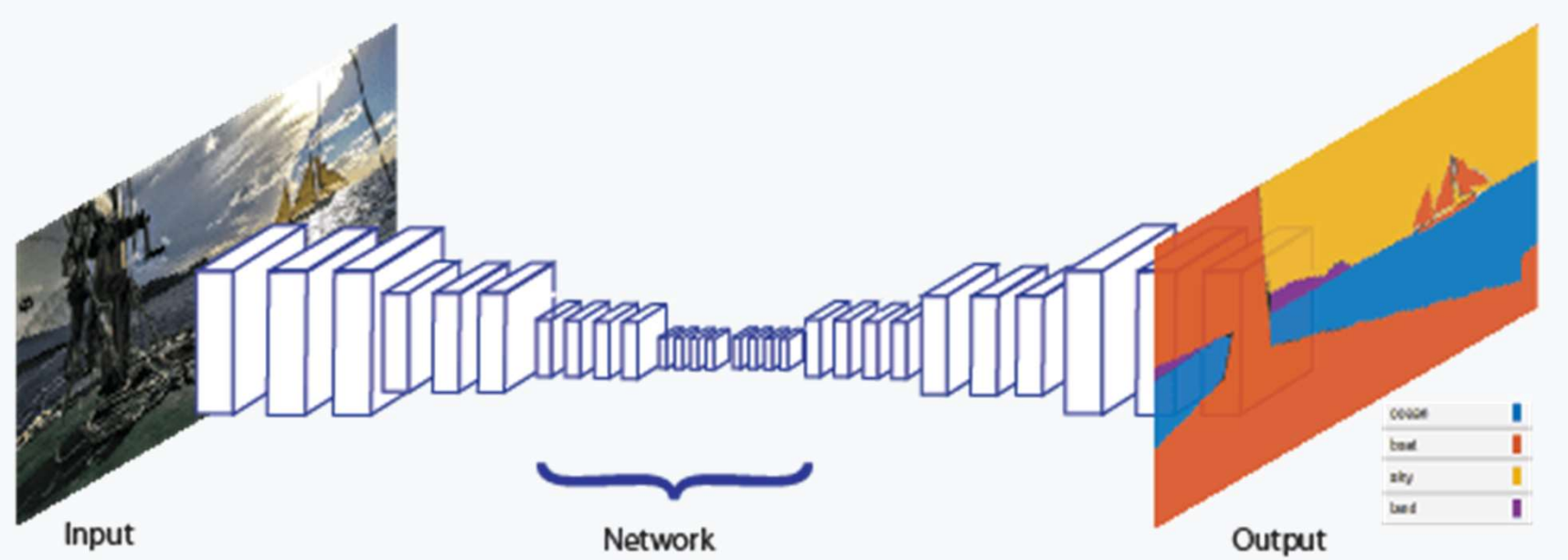
Pooling Layer(pool)

Fully Connected Layer(FC)



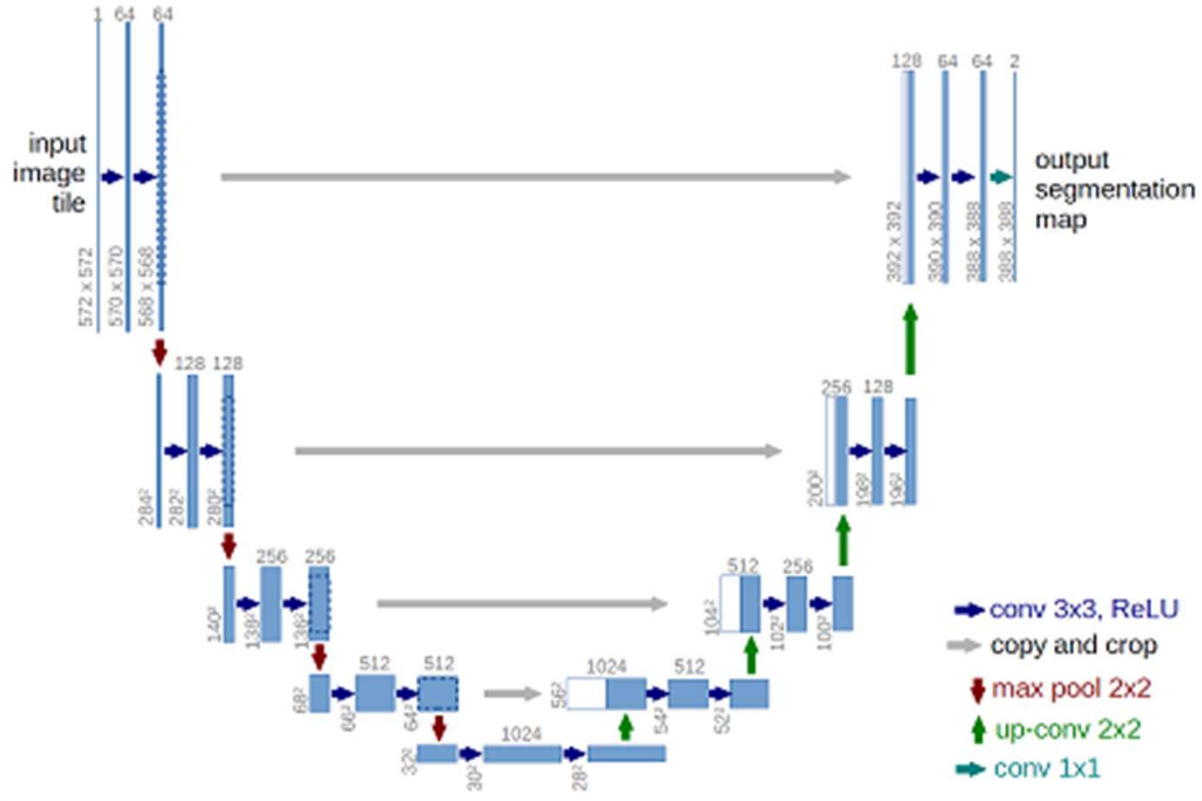
0. 복습

Transposed Convolution



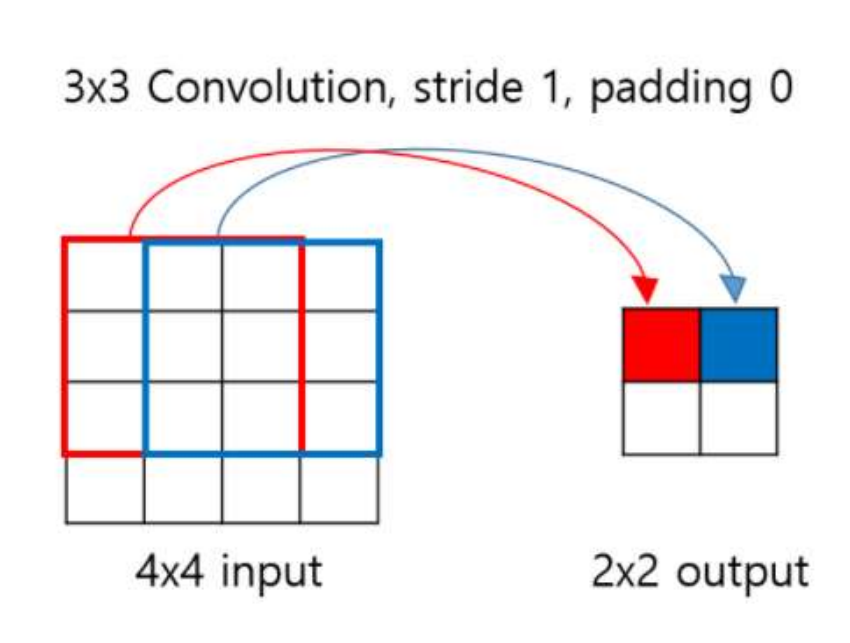
0. 복습

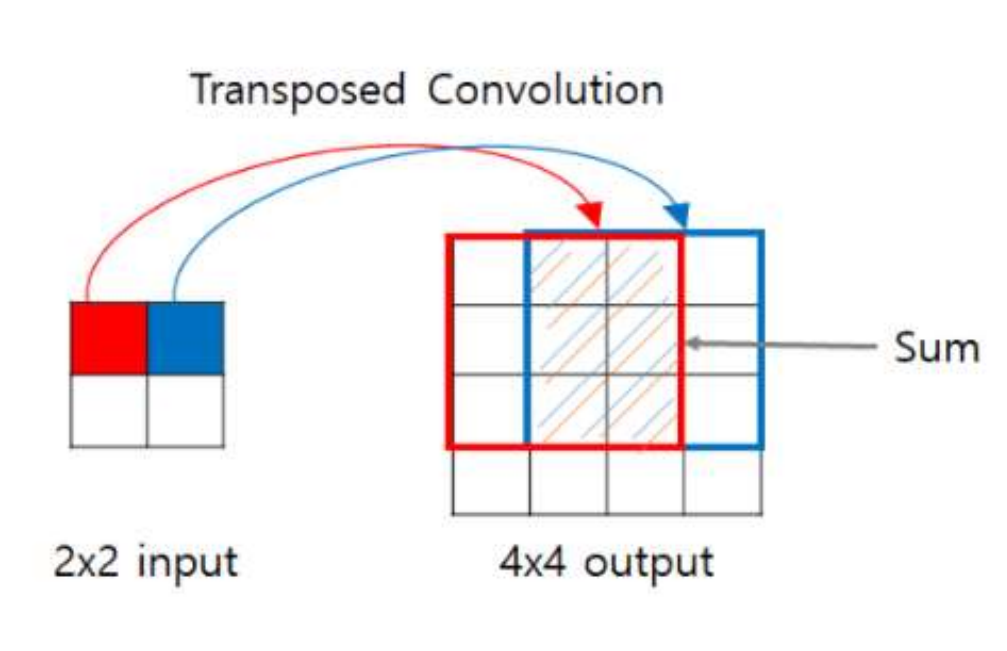
unet


input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

0. 복습

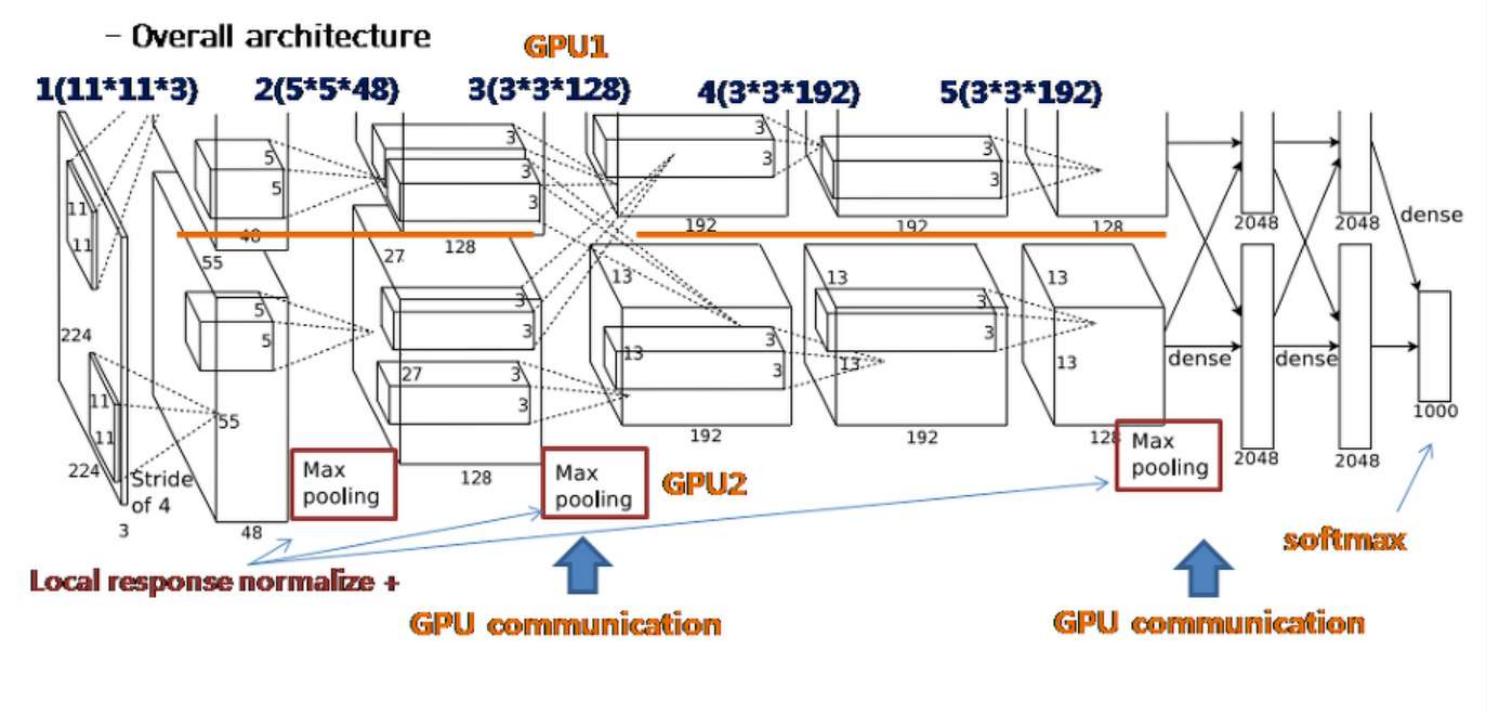
Transposed Convolution





0. 복습

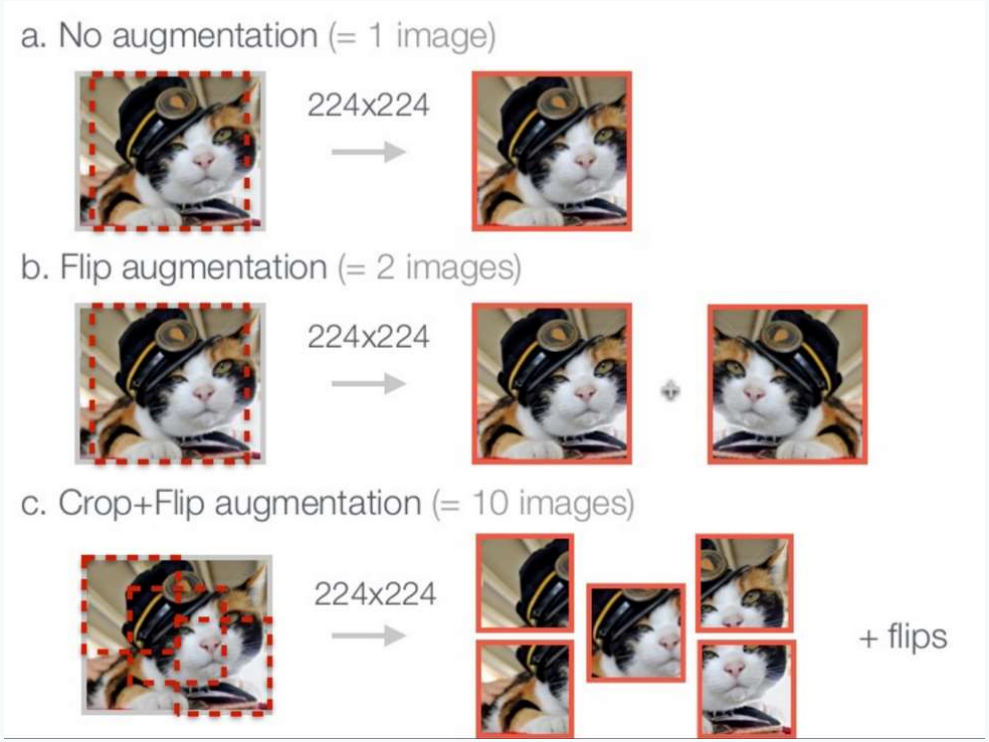
Alex NET : 구조



Pooling size =3, stride =2
최초의 max pooling 시도 -> overfitting 막는데 도움을 줌

0. 복습

Alex NET – Data augmentation



총 2048배
학습데이터 증가

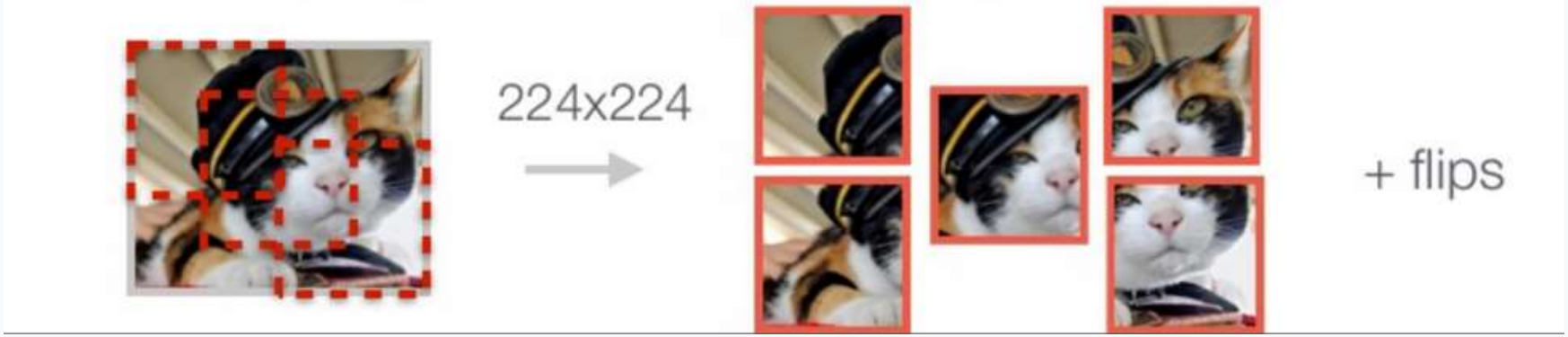
0. CNN 모델 – PAPER REVIEW

Alex NET – inference (앙상블 적용)

테스트할 이미지 한장에서 이미지 4귀퉁이 + 가운데에서

224x224 size로 crop한 5장을 flip하여 10장의 이미지를 만듦

10장을 테스트(inference)하여 결과를 앙상블 하여 결과를 나타냄.



0. 복습

VGG NET – 학습이미지 : 224x224 고정

Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

싱글 스케일 이미지 사이즈만으로 학습

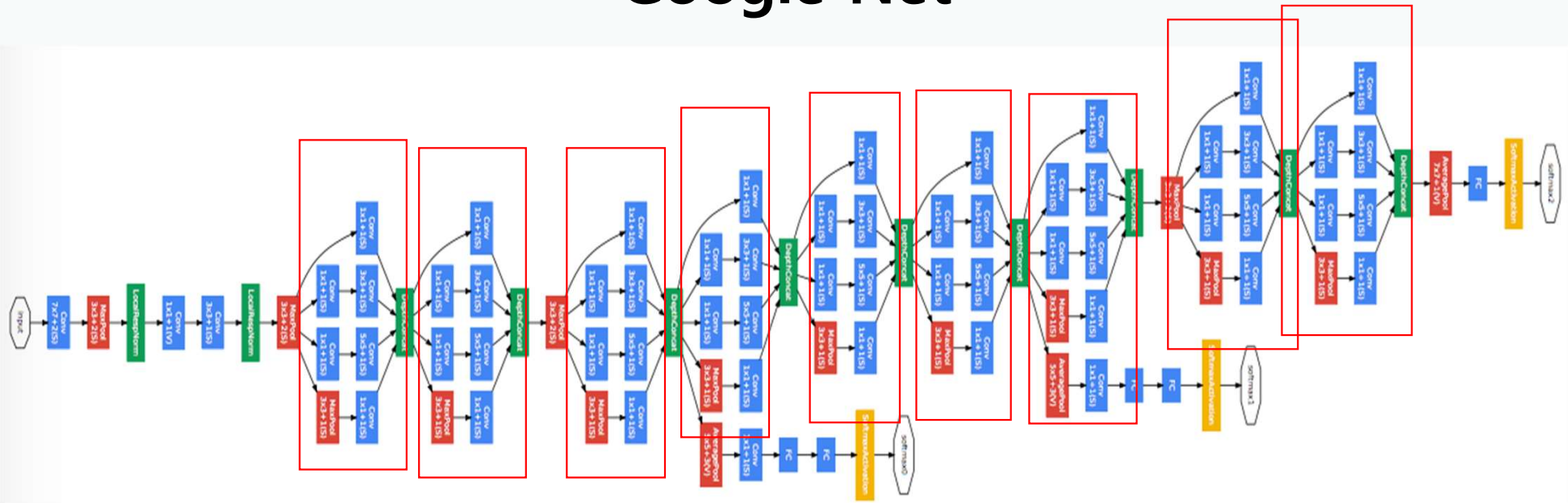
Table 4: **ConvNet performance at multiple test scales.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	7.5
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	**24.8**	7.5

멀티 스케일 이미지 사이즈만으로 학습

0. 복습

Google Net

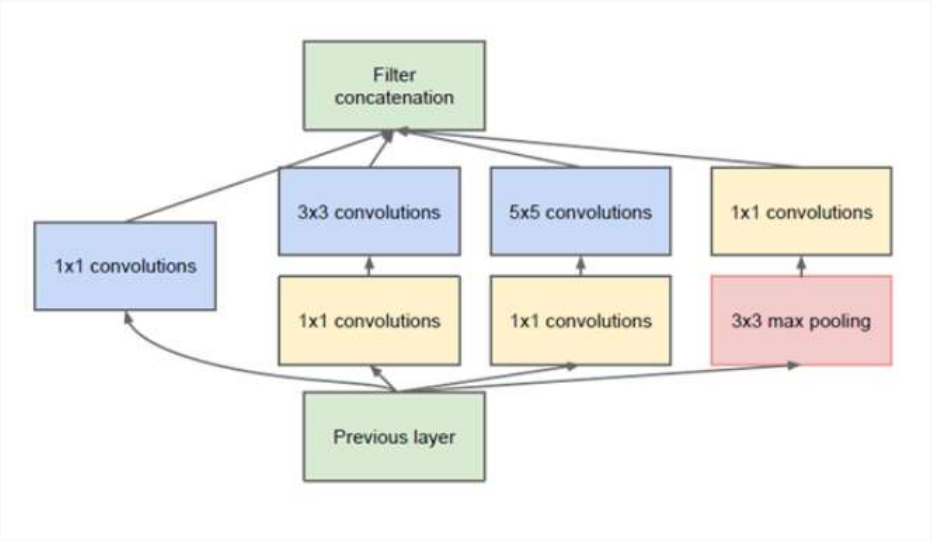


- Inception 모듈 사용
- 22개의 layers, 5M parameters(AlexNet의 1/12 수준)

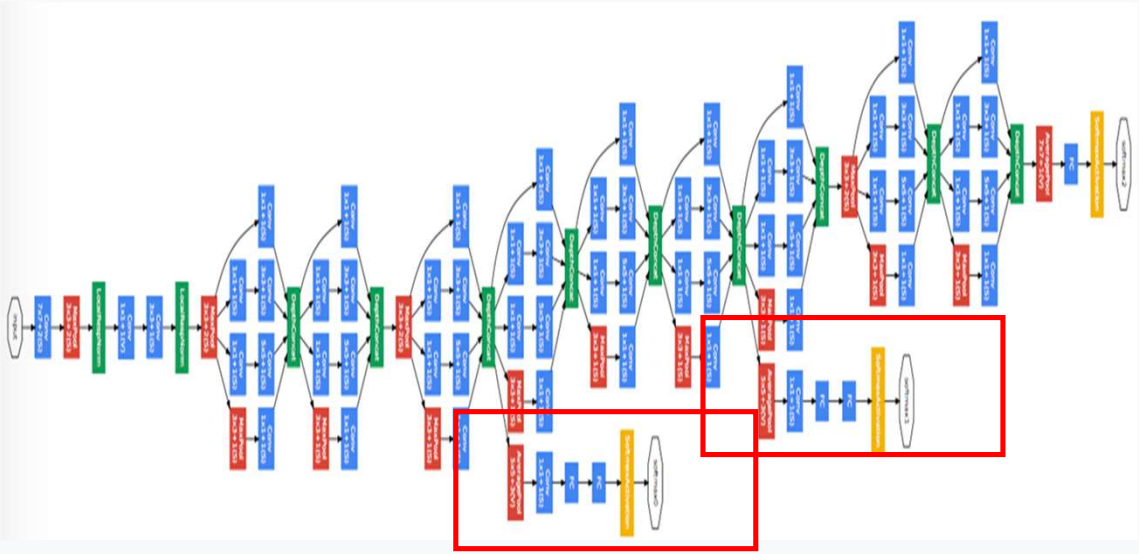
0. 복습

Google Net – 모델 솔루션

Inception module

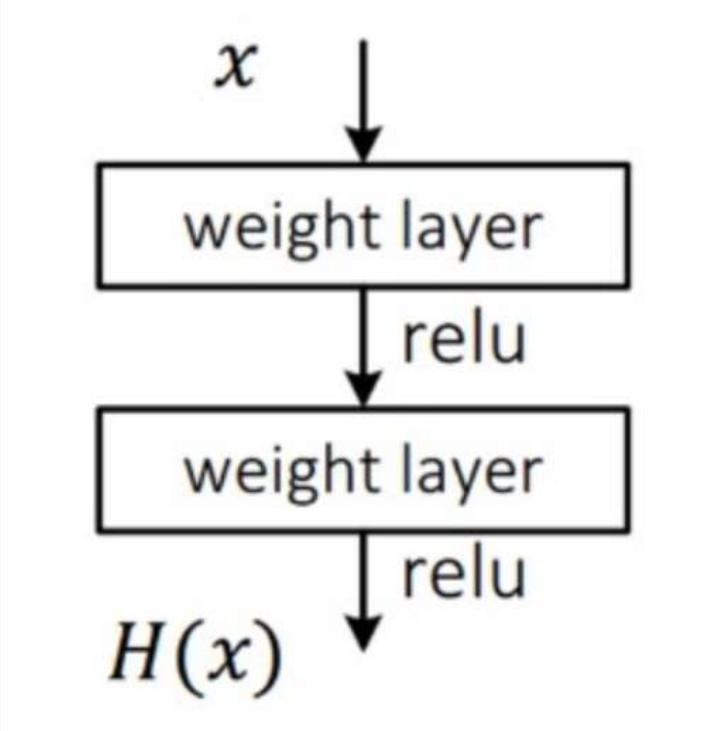


auxiliary classifier

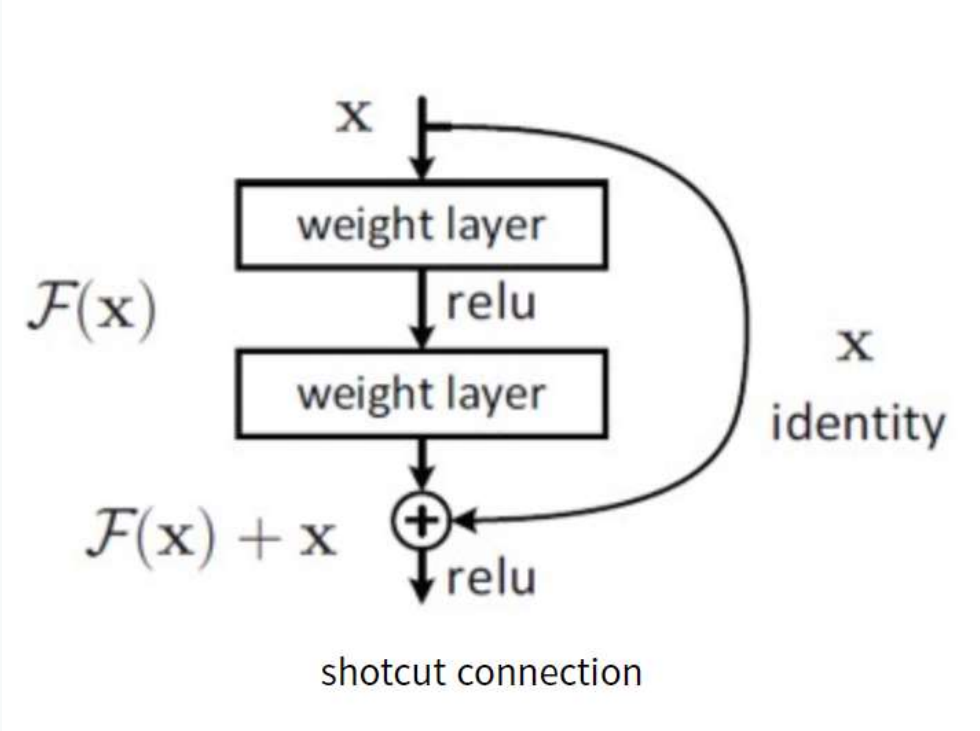


0. 복습

Res Net - Idea



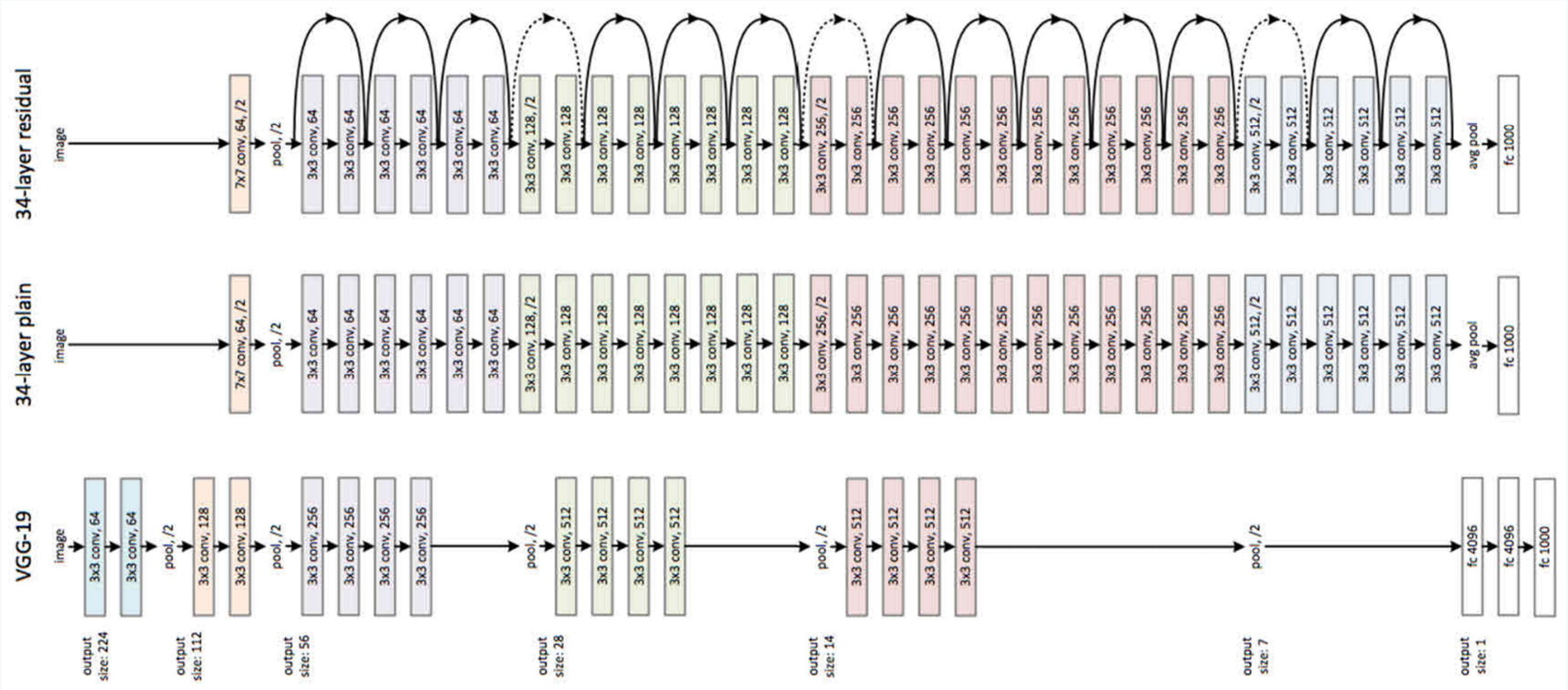
기존 CNN 구조



Resnet 구조

0. 복습

Res Net - 구조



0. 복습

DenseNet 구조

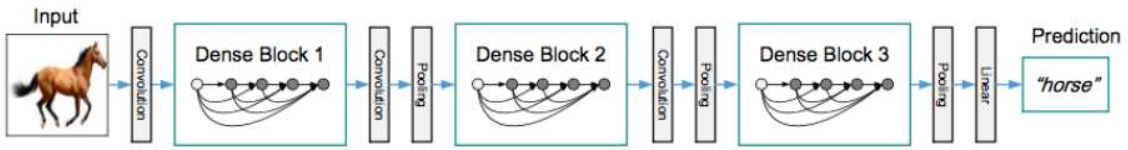


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

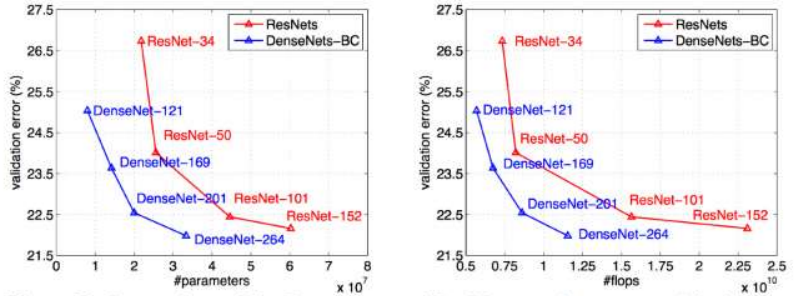


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

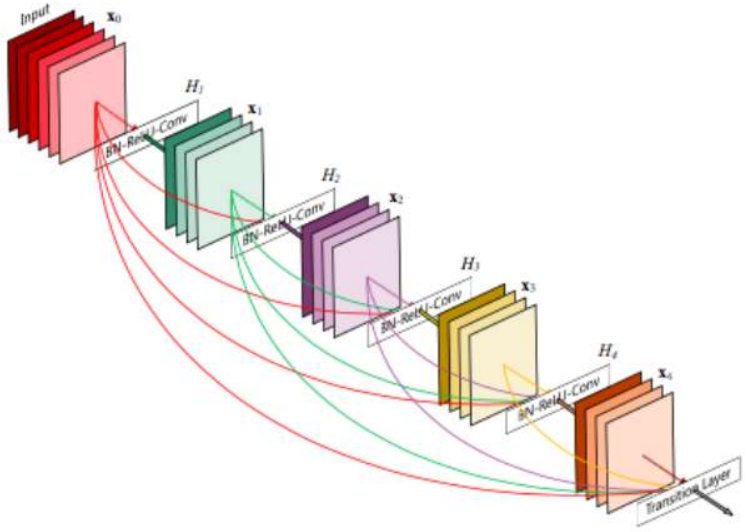


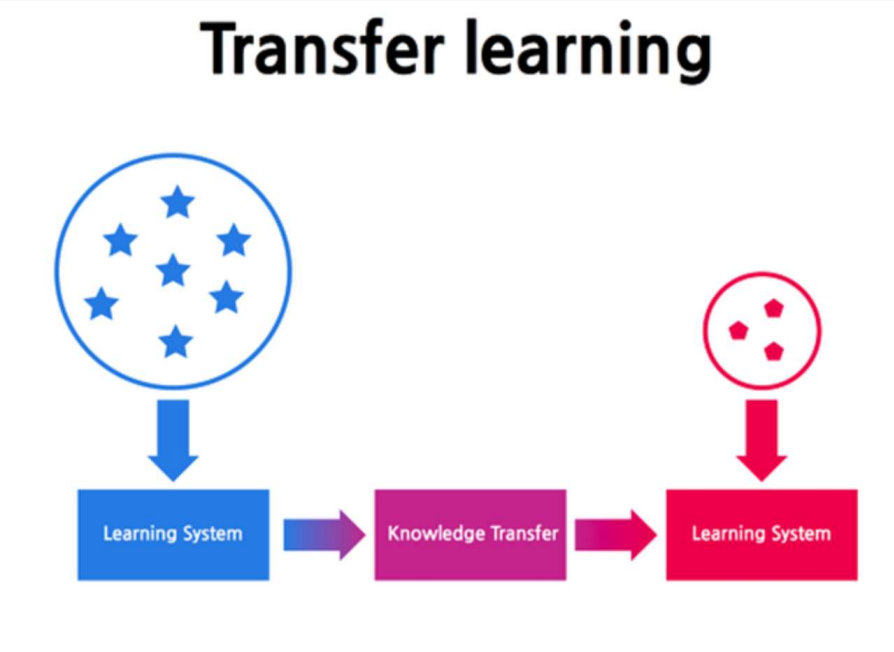
Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

0. 복습

Transfer Learning with CNNs

Transfer Learning 이란?

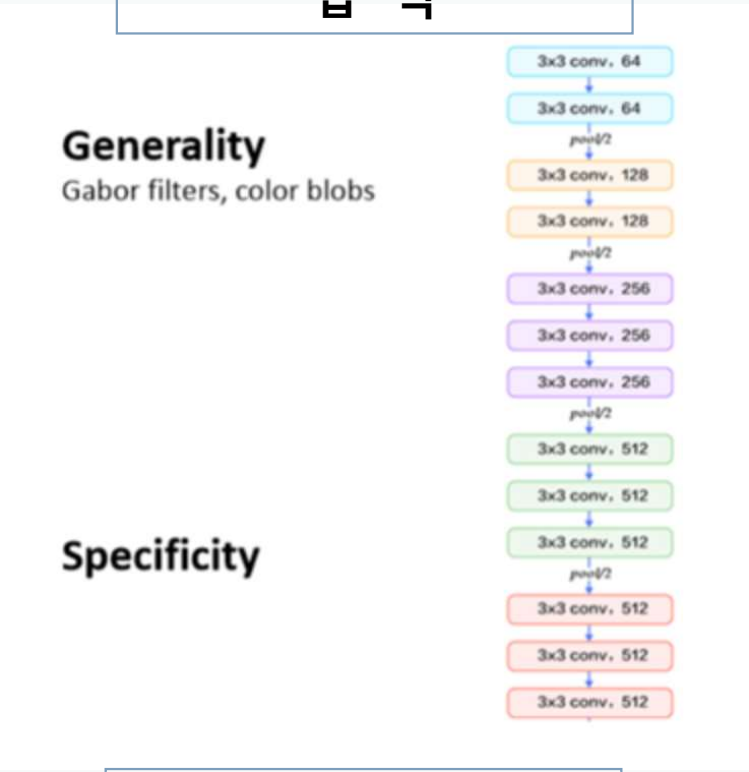
기존의 만들어진 모델을 사용하여 새로운 모델을 만들시 학습을 빠르게 하며, 예측을 더 높이는 방법.



0. 복습

Transfer Learning with CNNs

입 력

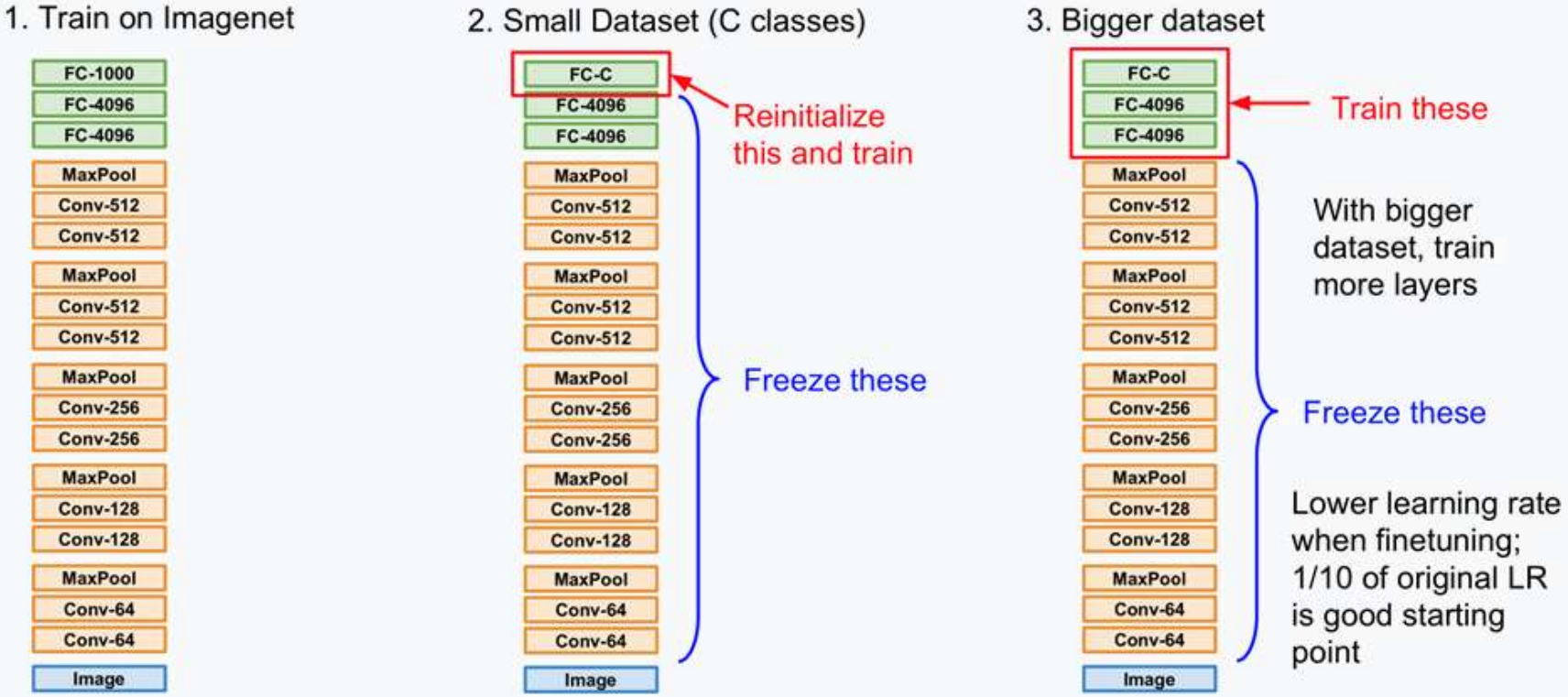


출 력

0. 복습

Transfer Learning with CNNs

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

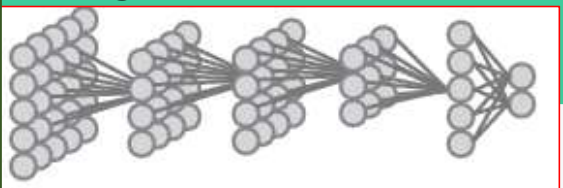


0. 오늘 하루 동안 무엇을

Deep Learning
신경망의 70년

Day4 학습 목표

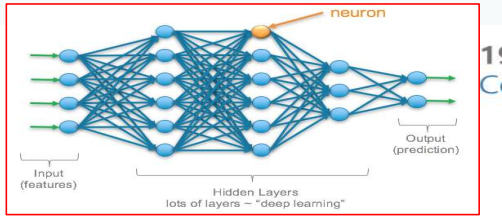
Image classification



CNN Model

Day2 학습 목표

regression, classification



1998년
Convolution Neural Network (CNN)

얀 르쿤

2016년
DeepMind AlphaGo

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

2006년
Deep Learning

제프리 힌튼

Day6 학습 목표 : 심화

2012년
AlexNet
IMAGENET

Day3 학습 목표

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

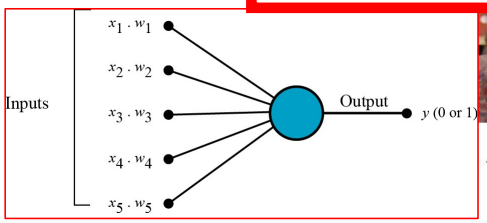
19C-20C 초반
생체 신경망

이반 파블로프

Day1 학습 목표

1958년
Perceptron

로젠블랫



regression, classification

1982년
Recurrent Neural Network (RNN)

홉필드

Day6 학습 목표

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

985

진도상황에 따라 변경될 수 있음

0. 오늘 하루 동안 무엇을 하나요?

1. RNN

- RNN
- LSTM
- GRU

2. RNN 실습

- RNN 주식 예측하기
- LSTM 주식 예측 하기
- GRU 주식 예측하기

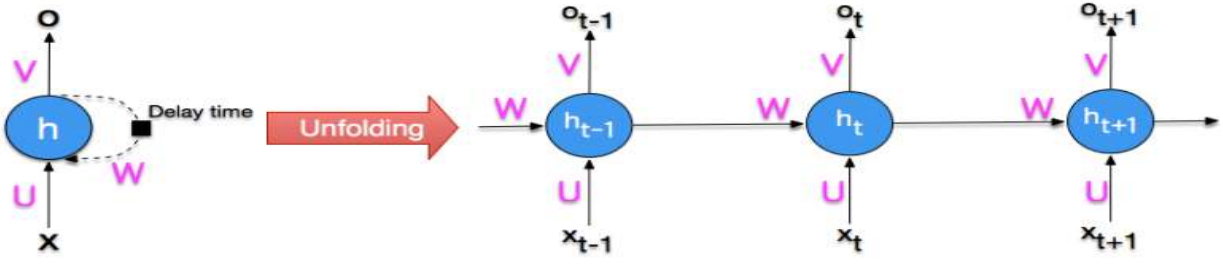
3. 심화과정

- 모델 경량화
- Grad CAM
- ~~Seq2Seq~~
- ~~Object Detection 실습 with annotation~~

1. RNN

RNN

- RNN은 Deep Learning 알고리즘 중 순차적인 데이터를 학습하여 classification 또는 predictio을 수행.
- 기존의 DNN의 경우 각 layer 마다 파라미터들이 독립적이었으나 RNN은 이를 공유
- 따라서, 현재의 출력 결과는 이전 time step의 결과에 영향을 받으며 hidden layer는 일종의 메모리 역할.



<RNN Architecture>

RNN - 어플리케이션

텍스트 생성

이전 단어들을 보고 다음 단어가 나올 확률을 예측하는 언어 모델을 기반으로 텍스트를 생성하는 generative model을 만들 수 있음

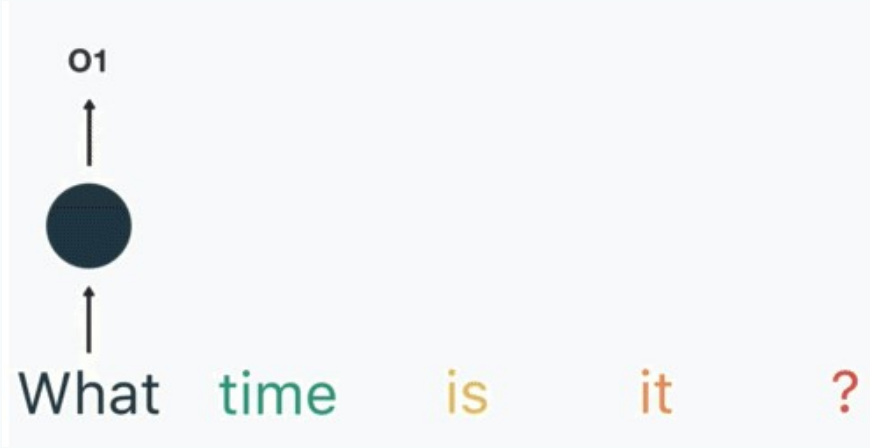
언어 번역

Word sequence를 입력으로 사용하여 번역할 언어의 word sequence로 출력

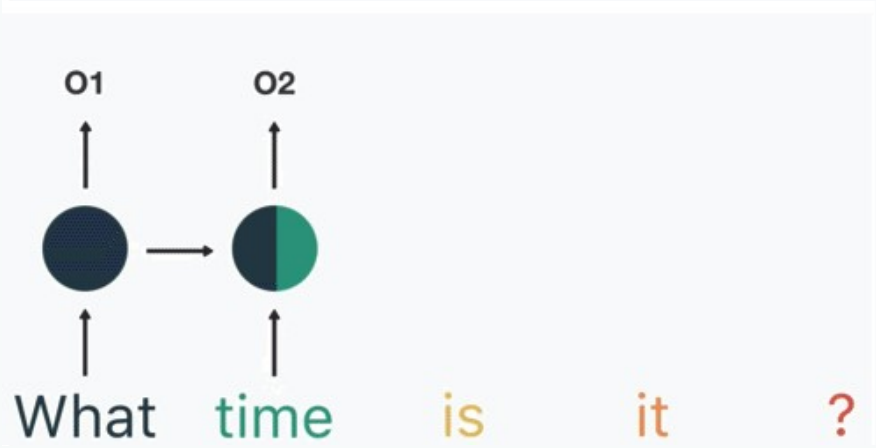
음성 인식

Acoustic signal을 입력으로 받아 phonetic segment의 sequence 또는 probabillity distribution을 추측

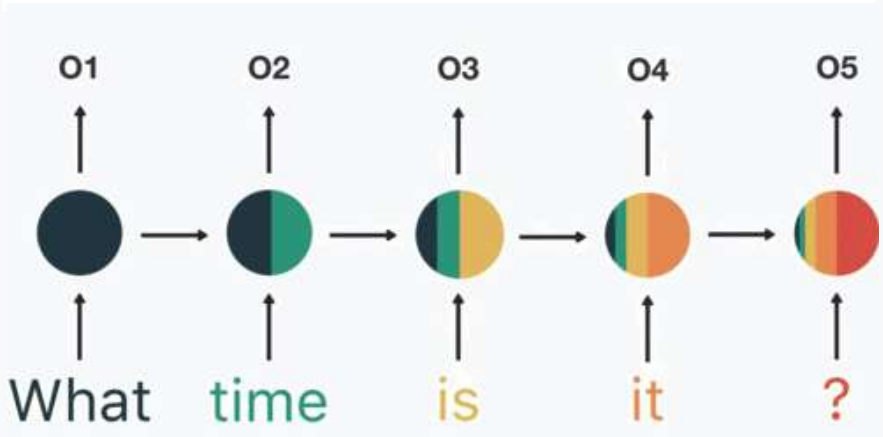
RNN - 예제



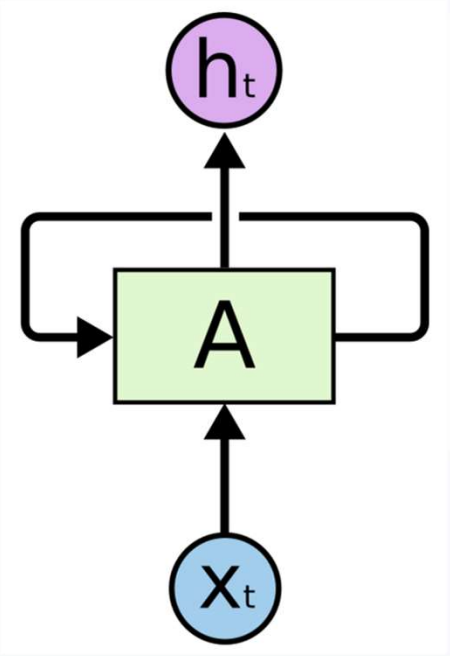
RNN - 예제



RNN - 예제



RNN - 예제



$$h_t = f(h_{t-1}, x_t)$$

new state　　old state　　input data at time t

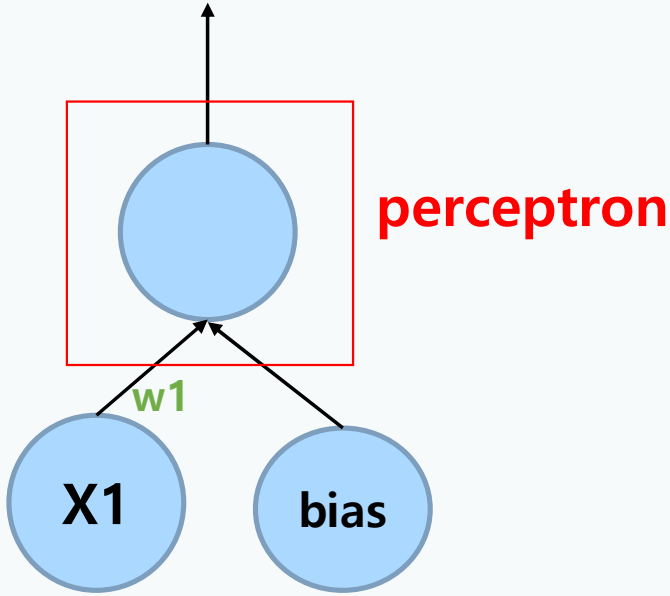
some function

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

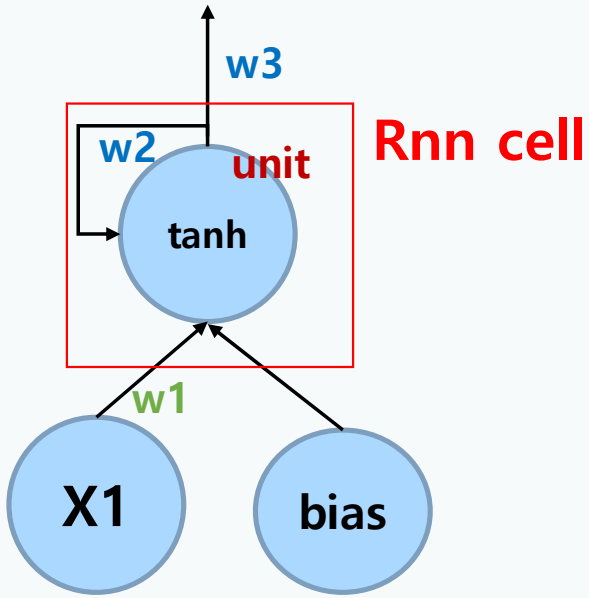
1. RNN

RNN 모델

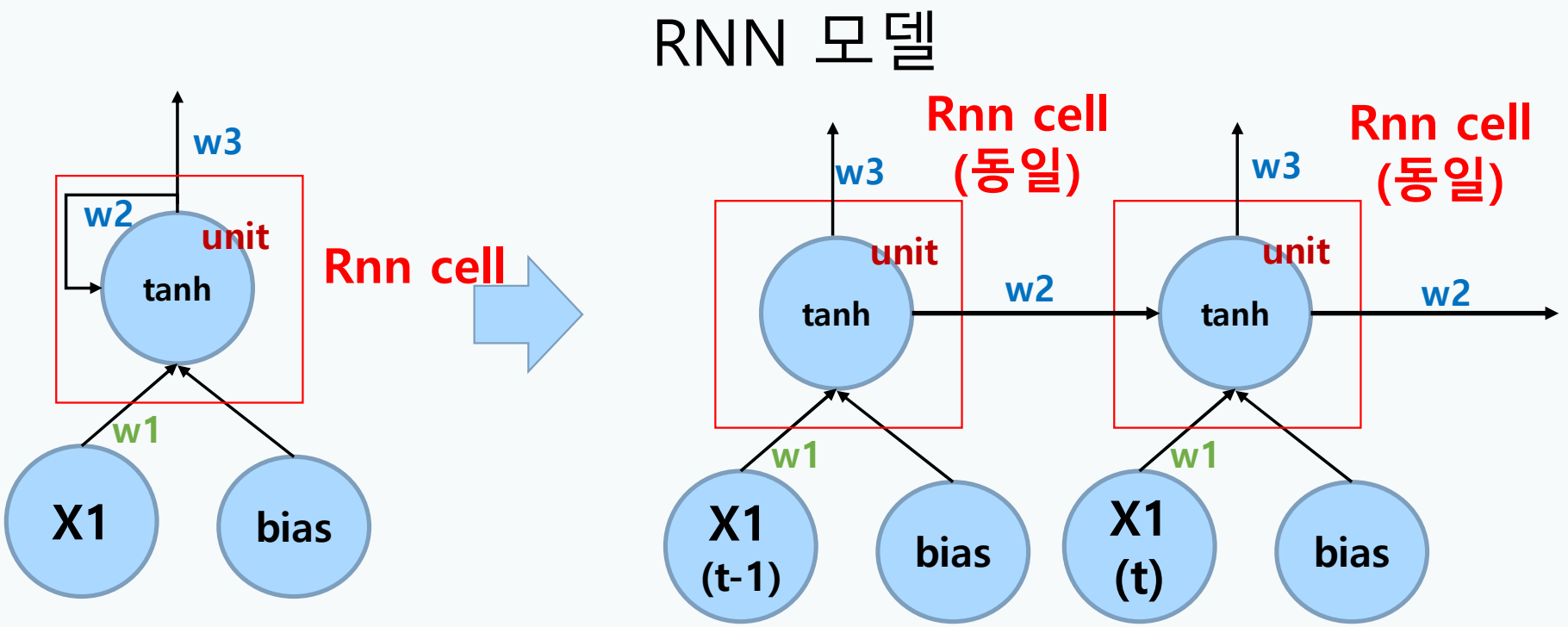


1. RNN

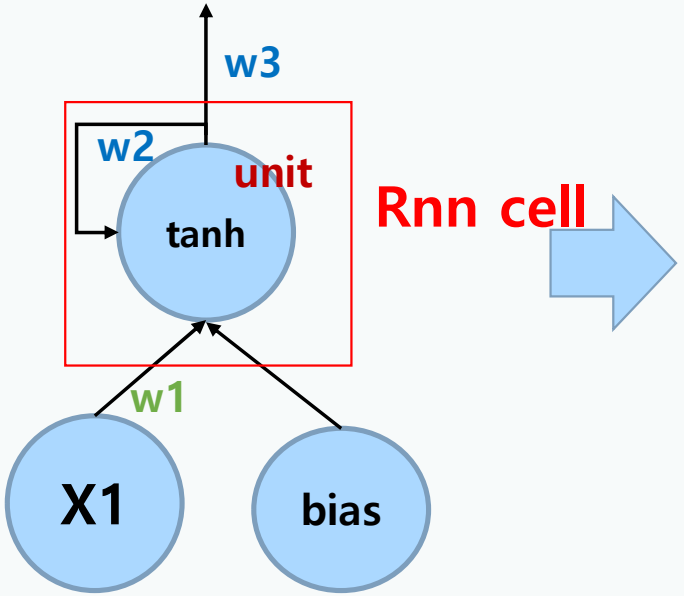
RNN 모델



1. RNN

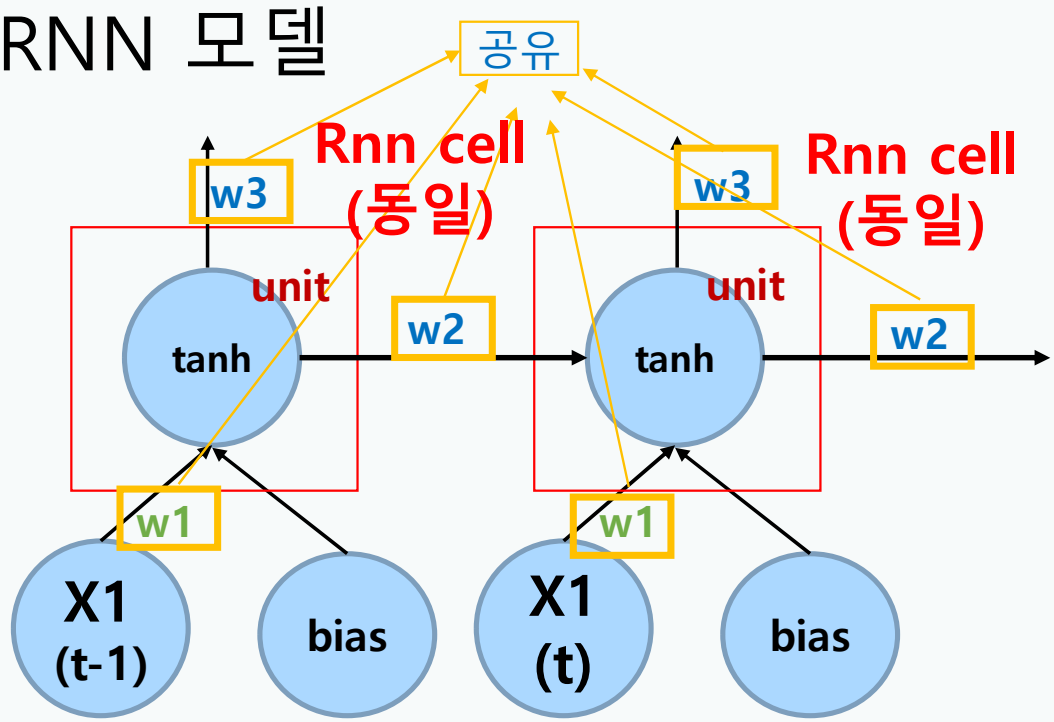

RNN 모델
w3
w2
unit
tanh
Rnn cell
w1
X1
bias
Rnn cell
(동일)
Rnn cell
(동일)
w3
w3
unit
unit
w2
w2
tanh
tanh
w1
w1
X1
(t-1)
bias
X1
(t)
bias

1. RNN 실습

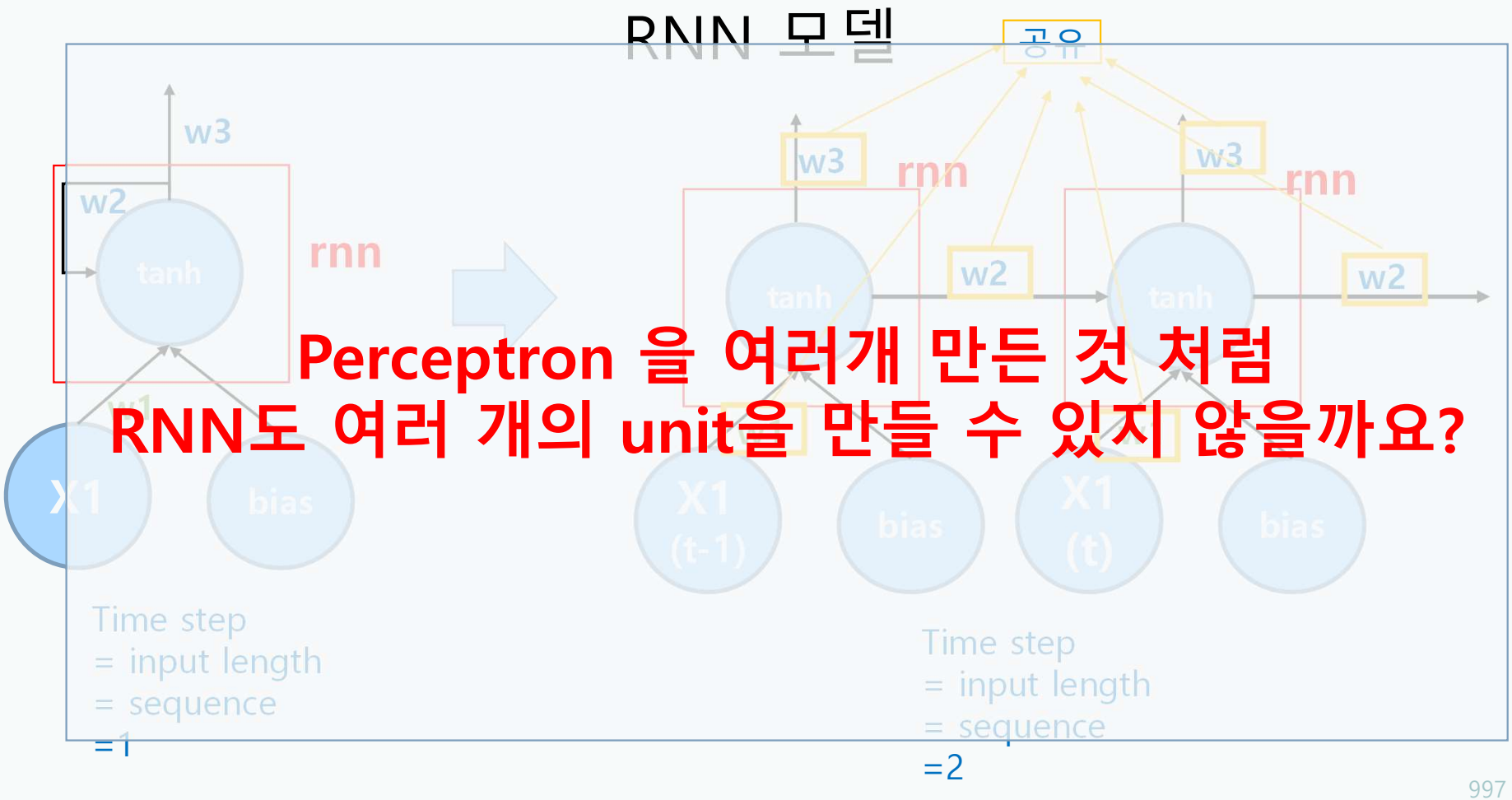


Rnn cell

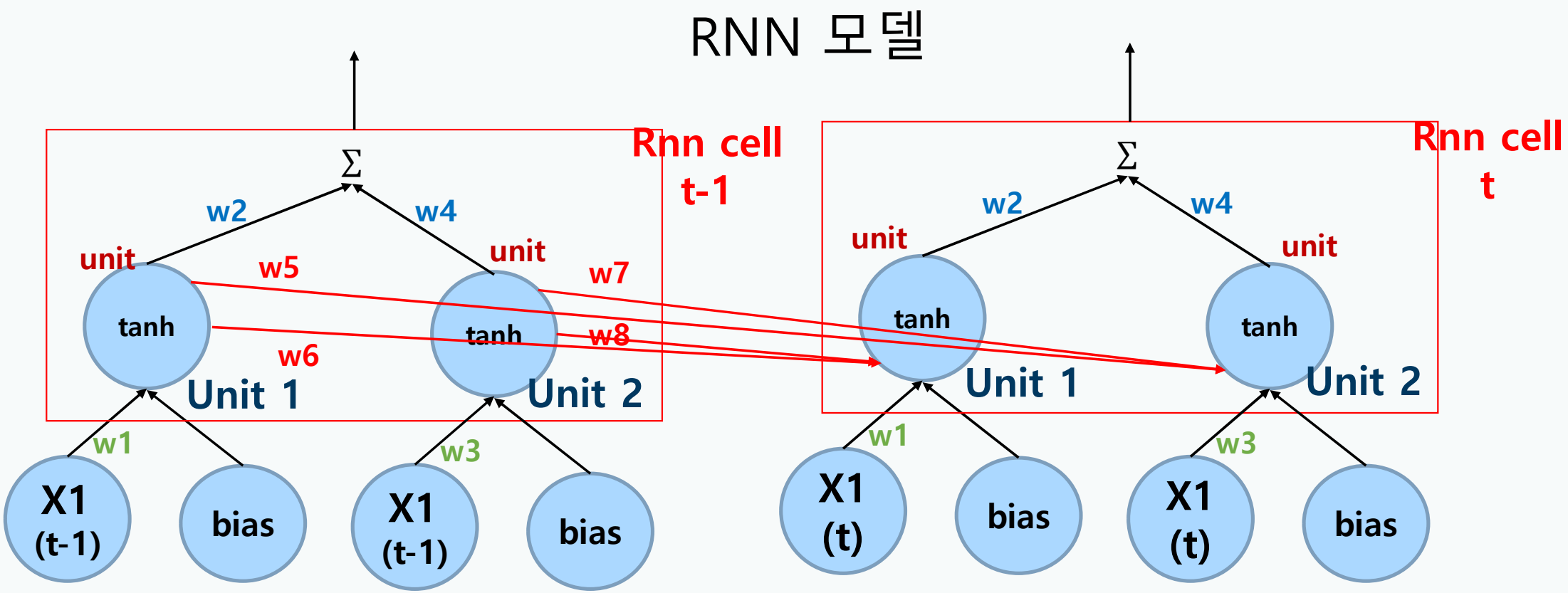
RNN 모델



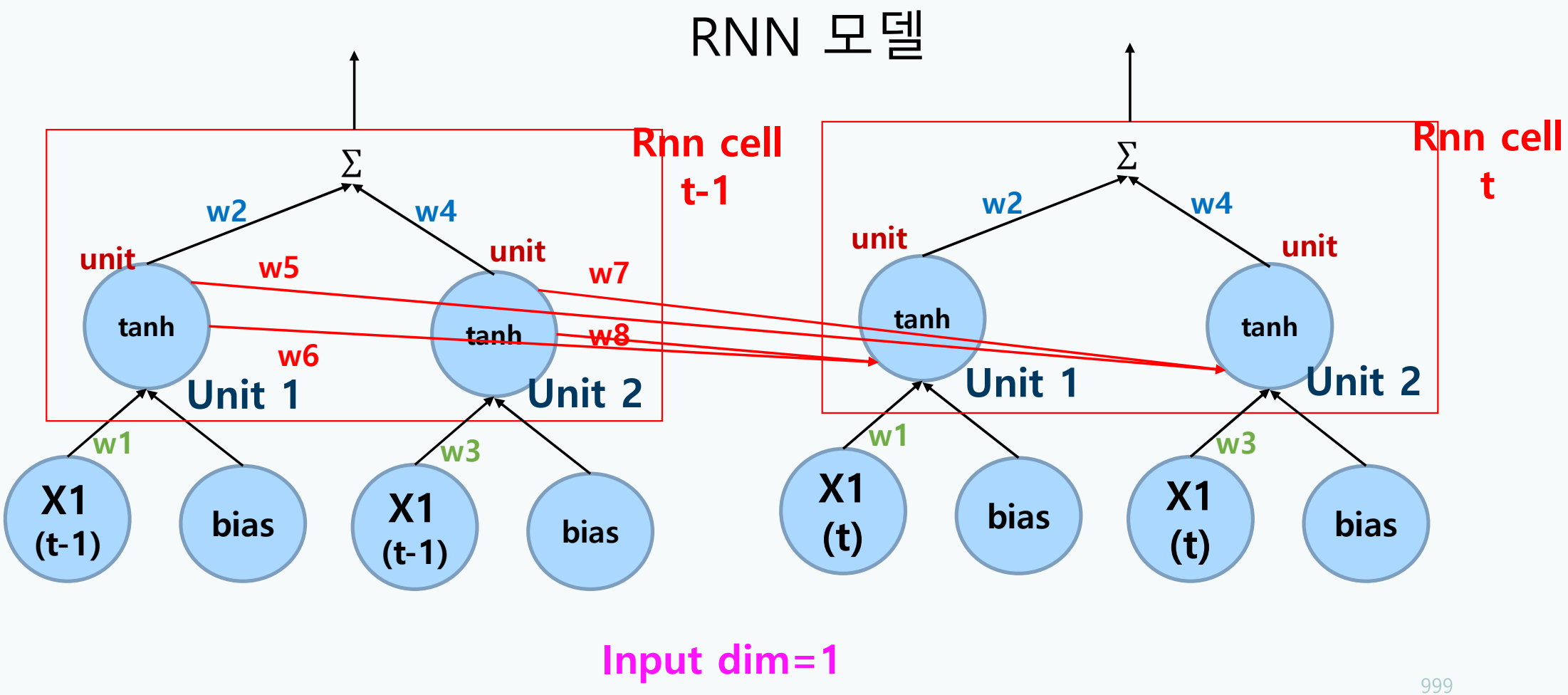
1. RNN


RNN 모델
공유
w3
w2
tanh
rnn
X1
bias
X1 (t-1)
X1 (t)
Time step
= input length
= sequence
=1
=2
Perceptron 을 여러개 만든 것 처럼
RNN도 여러 개의 unit을 만들 수 있지 않을까요?

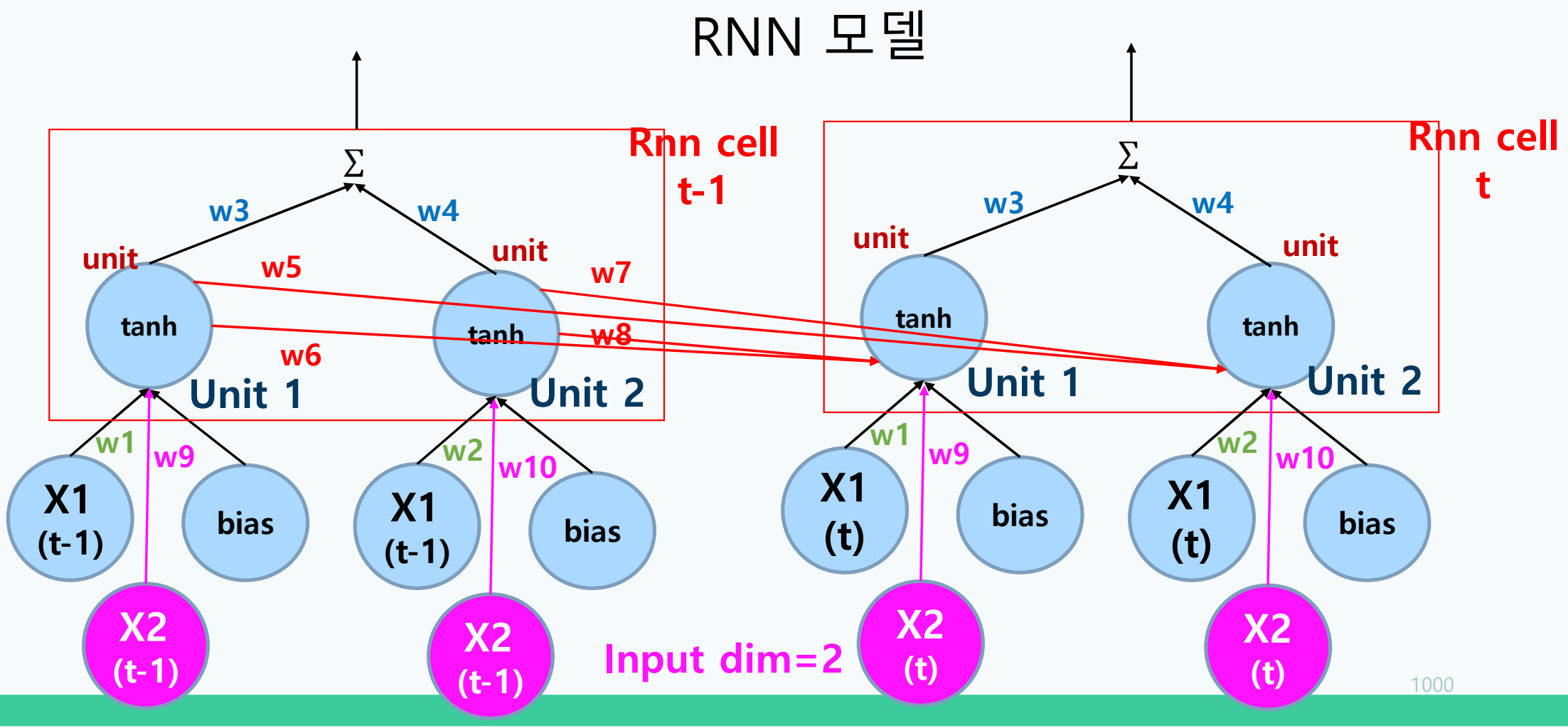
1. RNN


RNN 모델
Rnn cell
t-1
Rnn cell
t
Σ
Σ
w2
w4
w2
w4
unit
unit
unit
unit
w5
w7
w6
w8
tanh
tanh
tanh
tanh
Unit 1
Unit 2
Unit 1
Unit 2
w1
w3
w1
w3
X1
(t-1)
bias
X1
(t-1)
bias
X1
(t)
bias
X1
(t)
bias

1. RNN

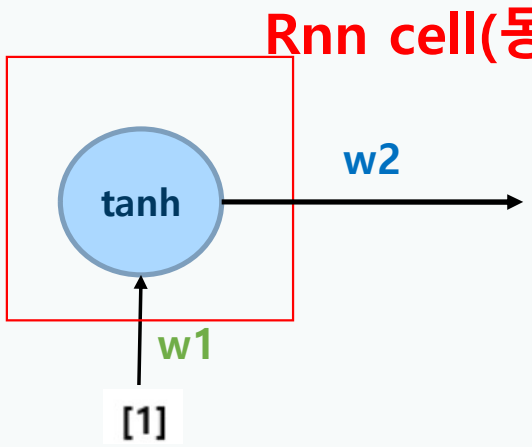

RNN 모델
Rnn cell t-1
Rnn cell t
Σ
w2
w4
unit
unit
w5
w7
tanh
tanh
w6
w8
Unit 1
Unit 2
w1
w3
X1 (t-1)
bias
X1 (t-1)
bias
X1 (t)
bias
X1 (t)
bias
Input dim=1

1. RNN


RNN 모델
Rnn cell t-1
Rnn cell t
Σ
w3
w4
unit
unit
w5
w7
w6
w8
tanh
tanh
Unit 1
Unit 2
w1
w9
w2
w10
X1 (t-1)
bias
X2 (t-1)
X1 (t)
X2 (t)
Input dim=2

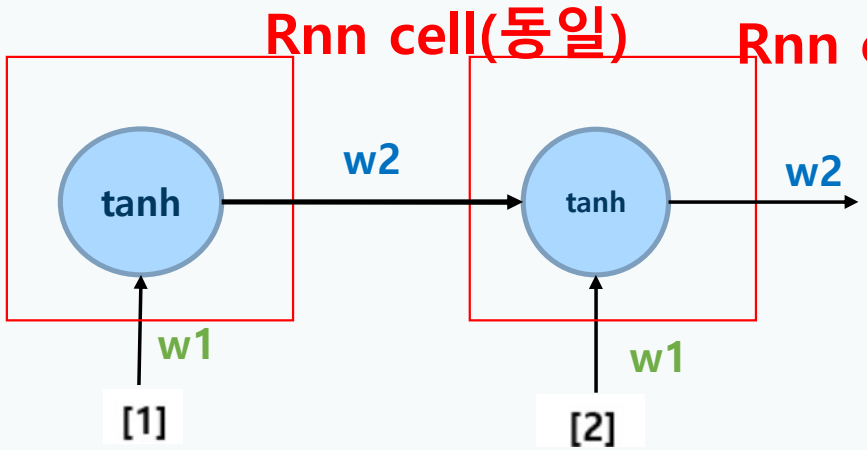
1. RNN

Many – to –One $[1, 2, 3] \rightarrow [4]$



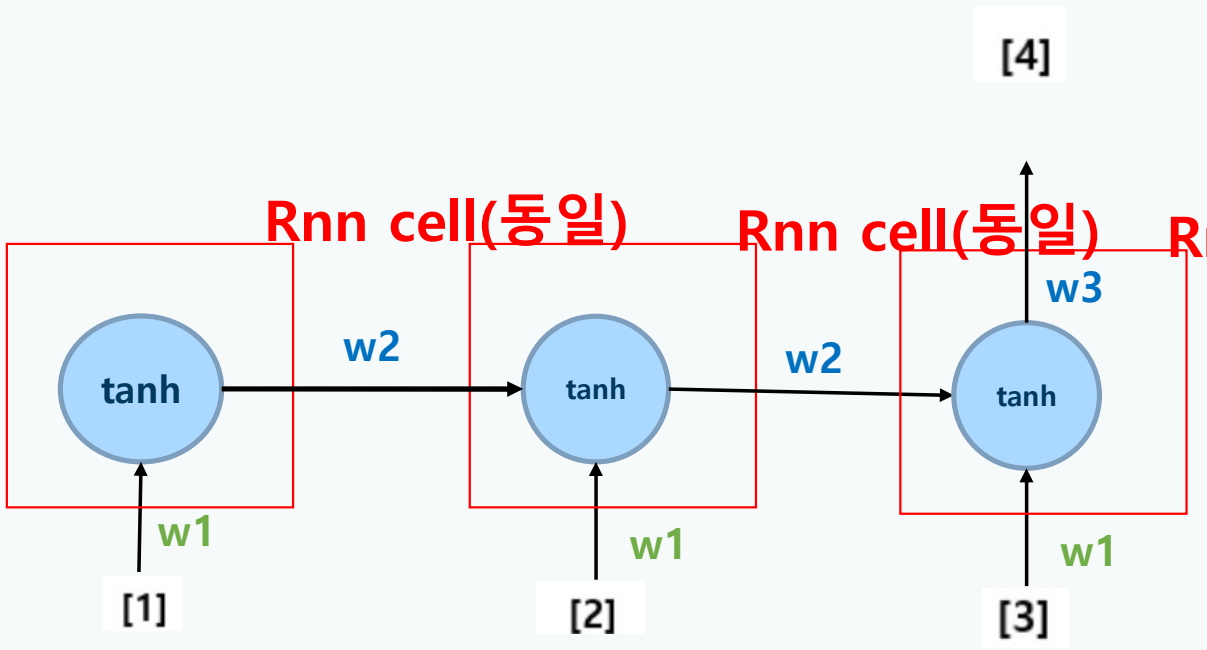
1. RNN

Many – to –One $[1, 2, 3] \rightarrow [4]$



1. RNN

Many – to –One $[1, 2, 3] \rightarrow [4]$



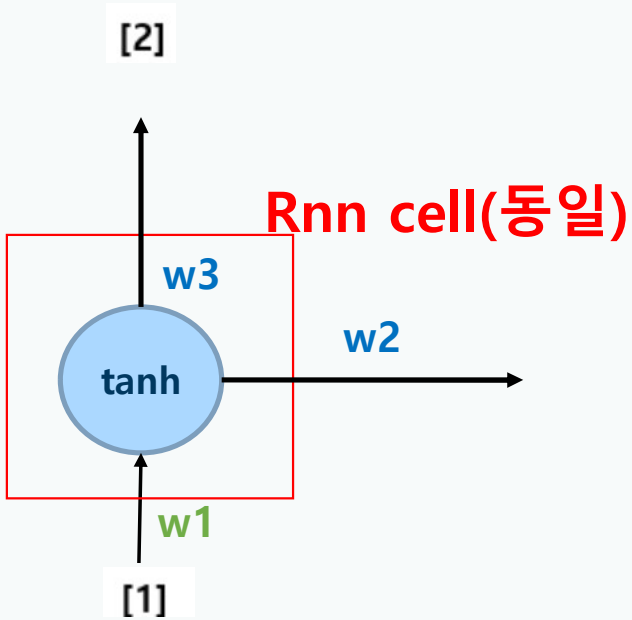
Input
(batch size, time, input 갯수)
1, 3, 1

Output
(batch size, time, out)
1, 1, 1

return_sequences=False

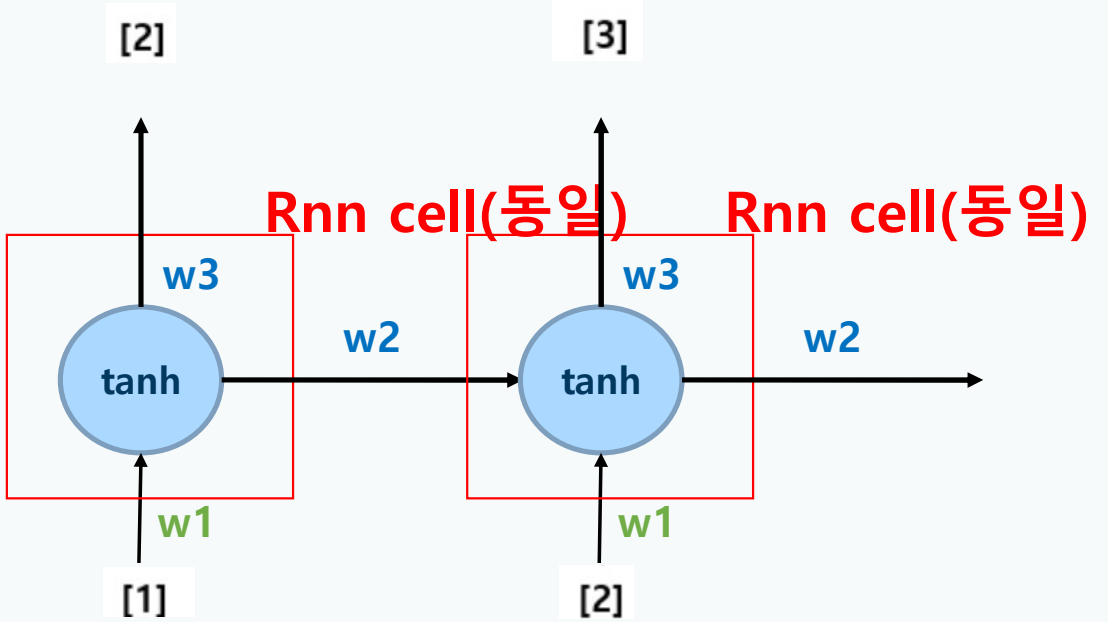
1. RNN

Many – to –Many $[1, 2, 3] \rightarrow [2, 3, 4]$



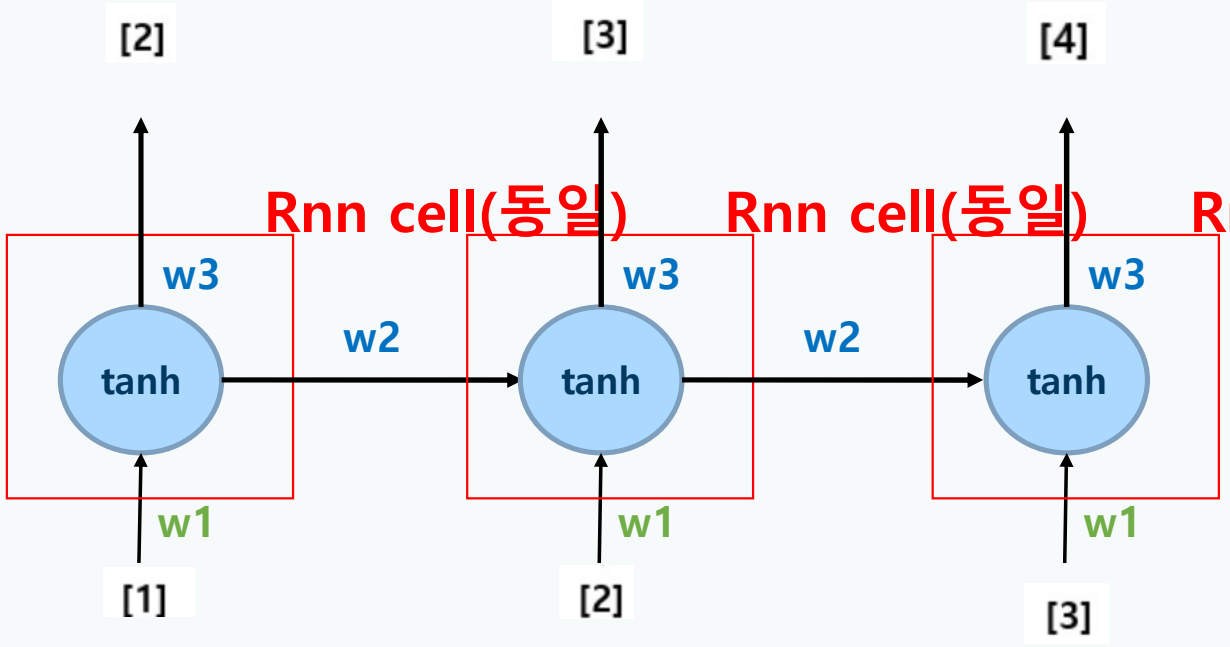
1. RNN

Many – to –Many $[1, 2, 3] \rightarrow [2, 3, 4]$



1. RNN

Many – to –Many $[1, 2, 3] \rightarrow [2, 3, 4]$

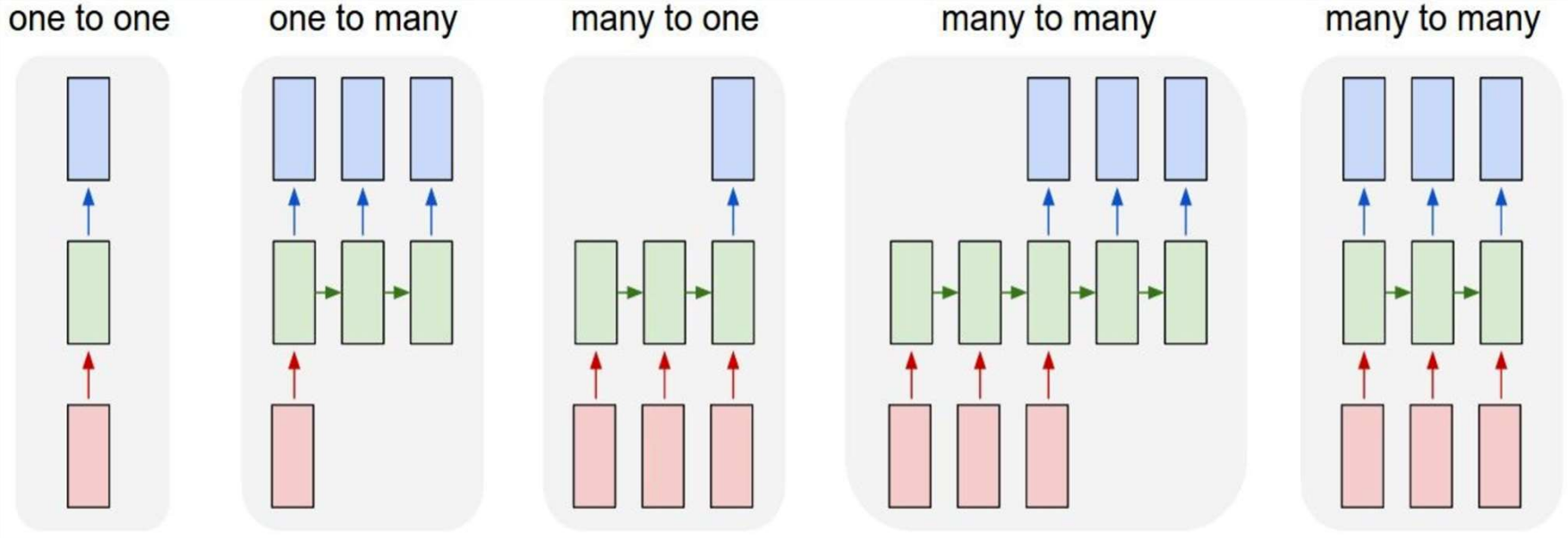


Input
(batch size, time, input 갯수)
3, 3, 1

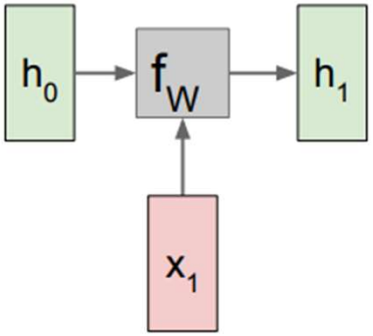
Output
(batch size, time, out)
1, 3, 1

return_sequences=True

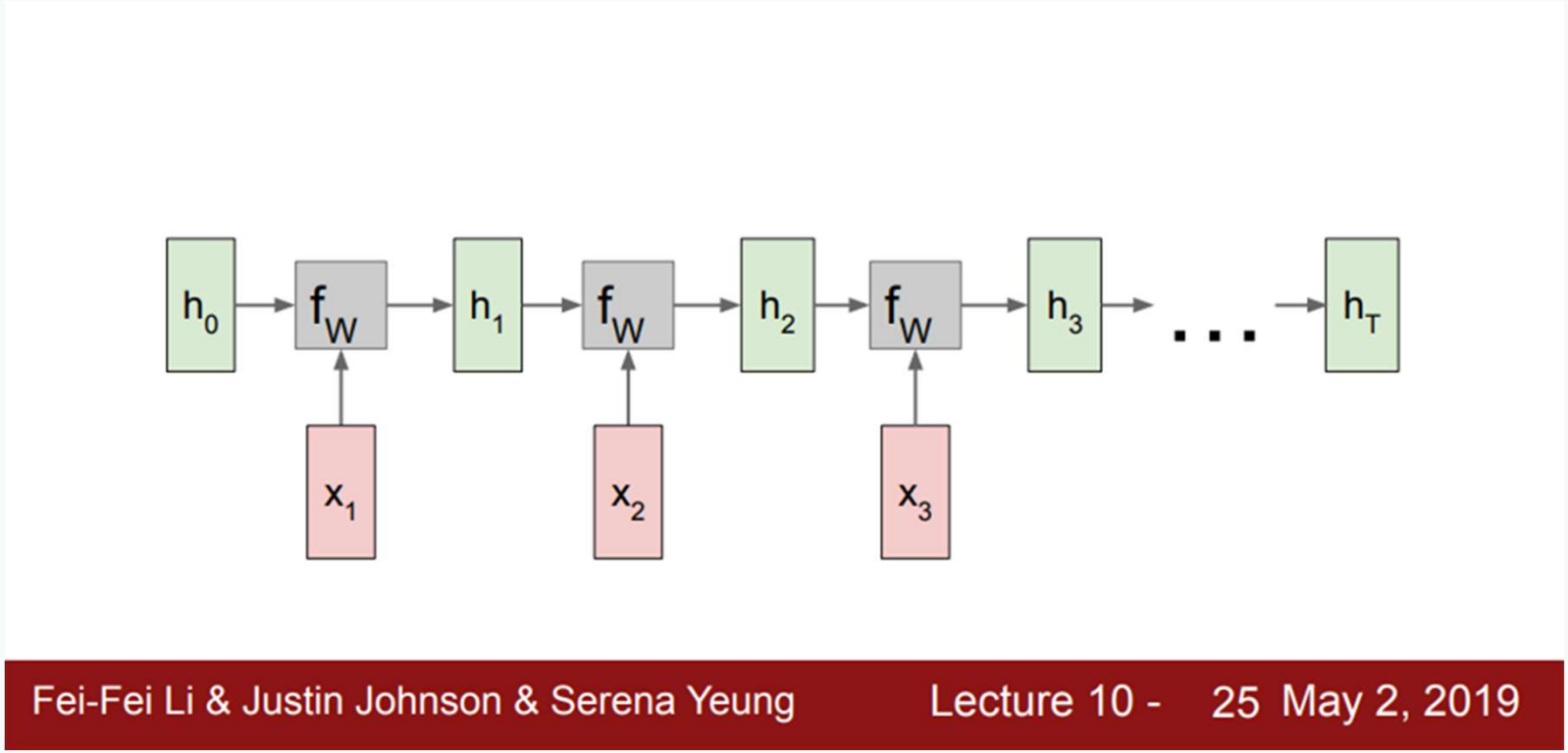
RNN - 예제



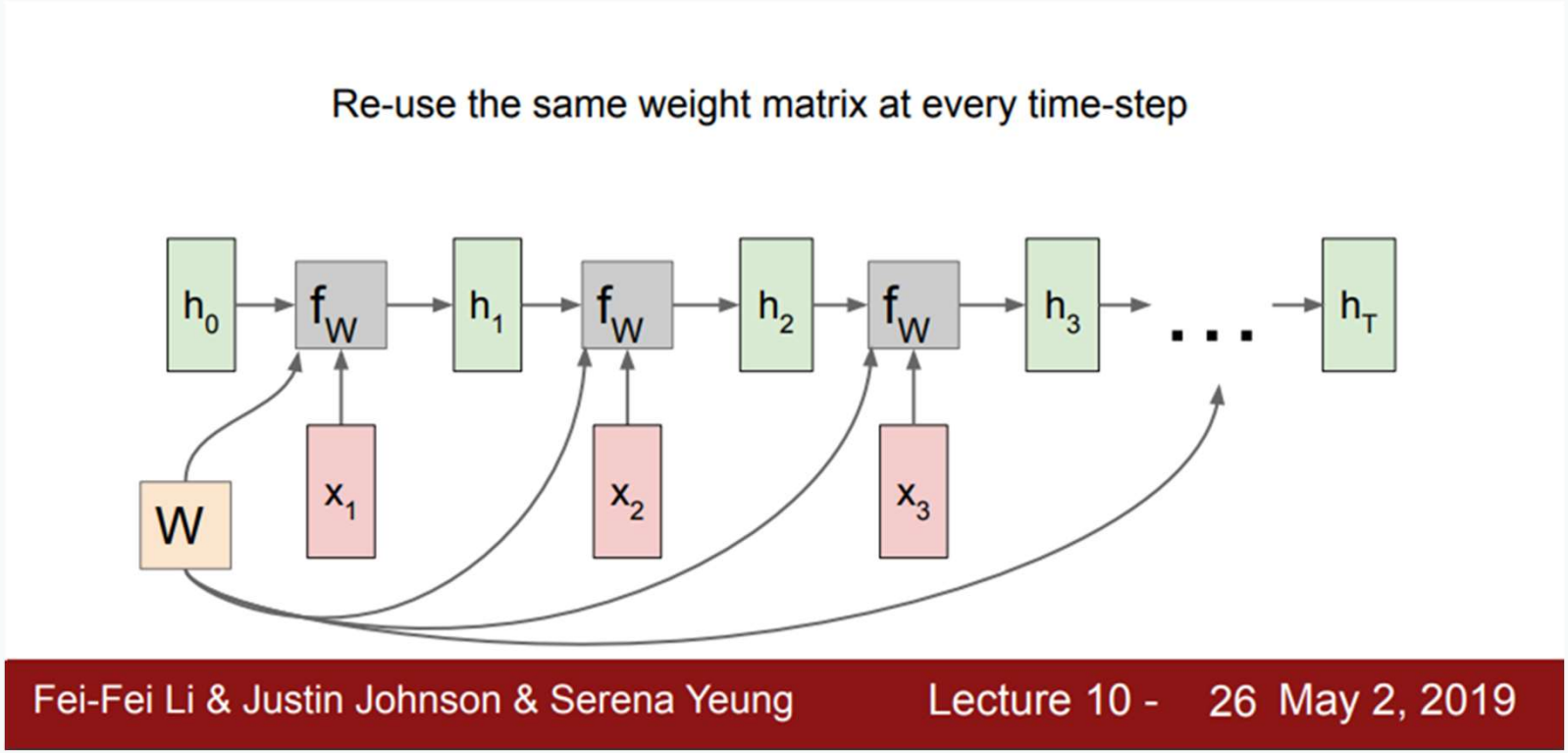
RNN – Computational Graph



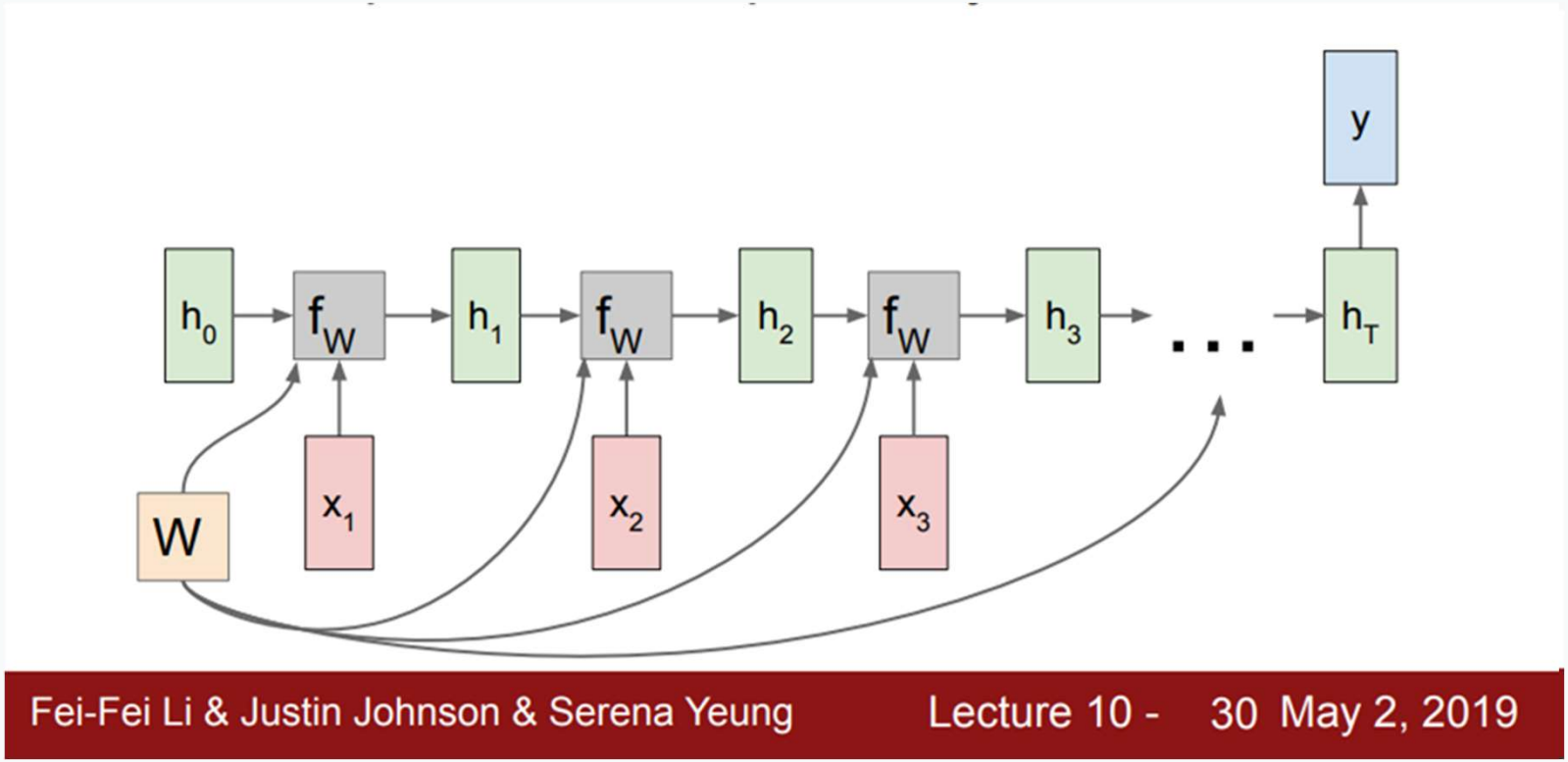
RNN – Computational Graph



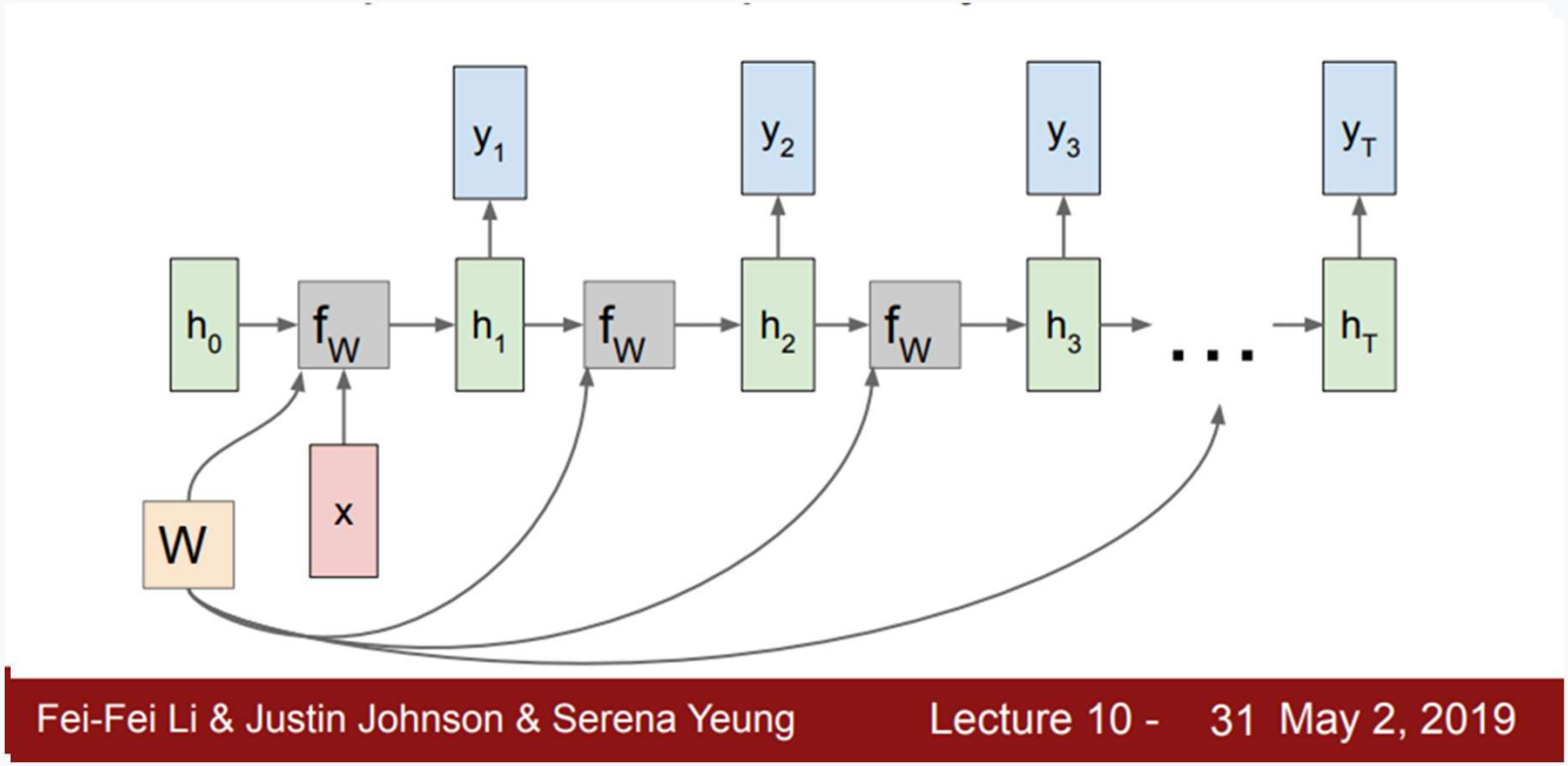
RNN – Computational Graph



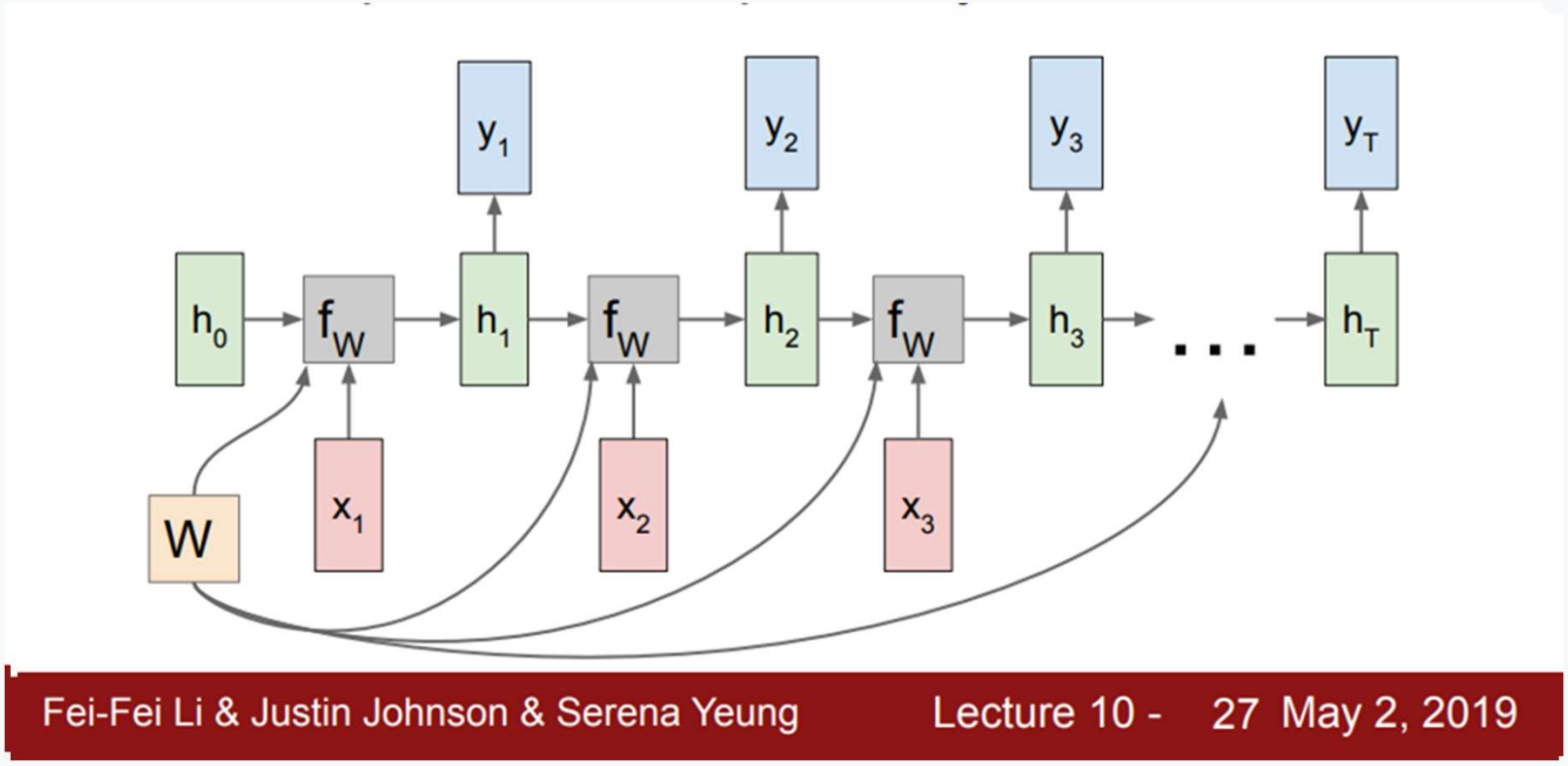
RNN – Many to One



RNN – Many to One



RNN – Many to Many



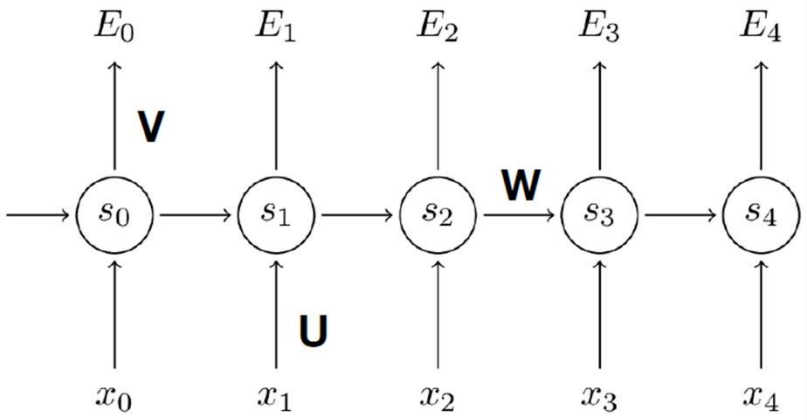
RNN – time backpropagation

Cross entropy loss at time t and total cross entropy

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t)$$

$$= -\sum_t y_t \log \hat{y}_t$$



RNN formula

$$h_t = \tanh(Ux_t + Wh_{t-1})$$

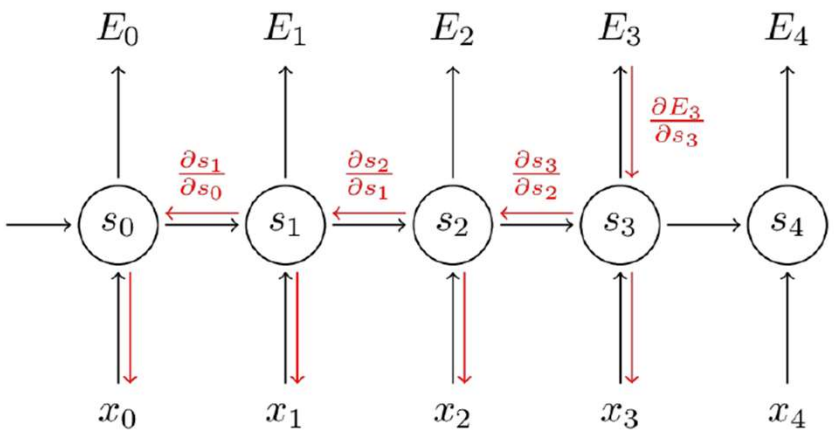
$$\hat{y}_t = \mathrm{softmax}(Vh_t)$$

RNN – time backpropagation

V에 대한 미분 at t=3

where $z_3 = Vh_3$

$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial V}$$
$$= \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V}$$
$$= (\hat{y}_3 - y_3) \otimes h_3$$



RNN formula

$$h_t = \tanh(Ux_t + Wh_{t-1})$$
$$\hat{y}_t = \mathrm{softmax}(Vh_t)$$

RNN – time backpropagation

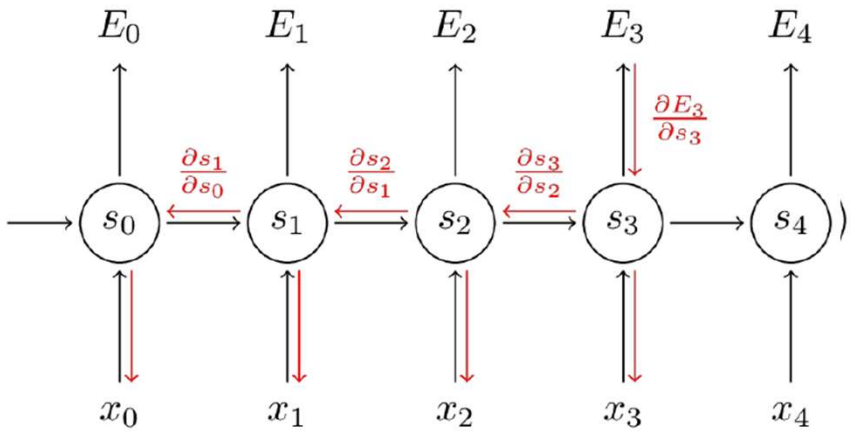
W에 대한 미분 at t=3

where $z_3 = Vh_3$

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial W}$$

where $h_3 = \tanh(Ux_3 + Wh_2)$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3}\frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial h_k}\frac{\partial h_k}{\partial W}$$



RNN formula

$$h_t = \tanh(Ux_t + Wh_{t-1})$$
$$\hat{y}_t = \mathrm{softmax}(Vh_t)$$

RNN – time backpropagation

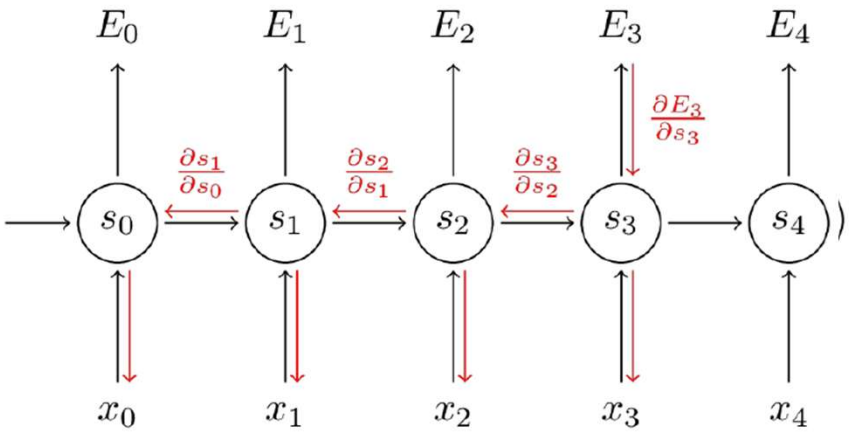
W에 대한 미분 at t=3

where $z_3 = Vh_3$

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial W}$$

where $h_3 = \tanh(Ux_3 + Wh_2)$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial h_k}\frac{\partial h_k}{\partial W}$$

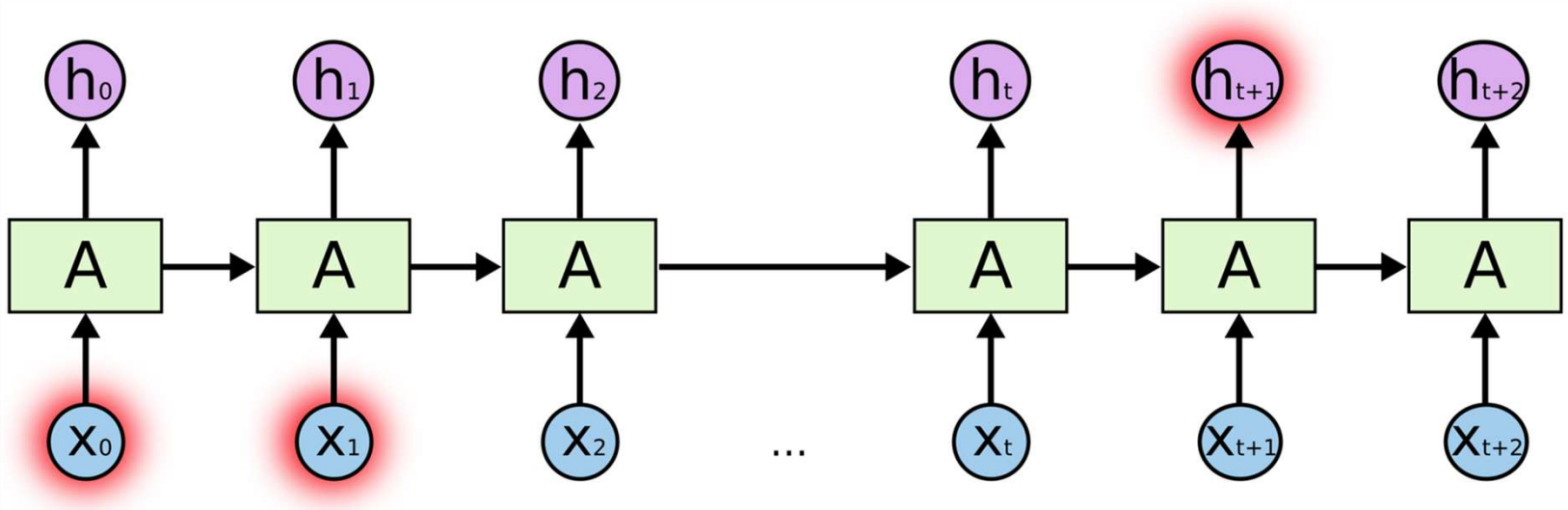


RNN formula

$$h_t = \tanh(Ux_t + Wh_{t-1})$$
$$\hat{y}_t = \mathrm{softmax}(Vh_t)$$

1. RNN

RNN – time backpropagation



RNN – 문제

- Vanishing Gradient
- 시간적으로 멀리 있는 정보는

  Vanishing Gradient 현상이 일어나 학습이 힘들다.

RNN – 해결방법

LSTM
(Long Short Term Memory)

2. LSTM

LSTM(Long Short Term Memory)

- Gate의 도입으로 입력, 출력을 조절할 수 있게 됐다
- 메모리를 이용하여 Long-range correlation을 학습할 수 있게 됐다
- Error를 입력에 따라 다른 강도로 전달할 수 있다
- Vanishing Gradient 문제 해결

2. LSTM

LSTM(Long Short Term Memory)

- Gate의 도입으로 입력, 출력을 조절할 수 있게 됐다
- 메모리를 이용하여 Long-range dependency를 학습할 수 있게 됐다
- Error를 입력에 따라 다른 강도로 전달할 수 있다
- Vanishing Gradient 문제 해결

이렇게 동작하는 핵심 아이디어는

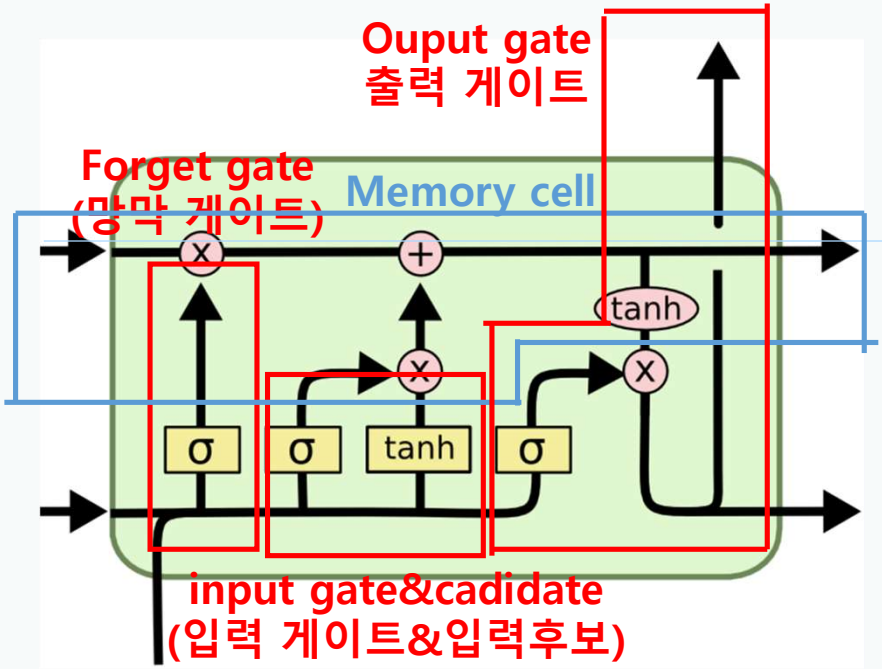
이전 단계의 정보를 memory cell에 저장하여 흘려 보냄.

이때, 현재 시점의 정보를 바탕으로 과거의 내용을 얼마나 잊을지 곱해주고

그결과에 현재의 정보를 더해서 다음 시점으로 정보를 전달.

2. LSTM

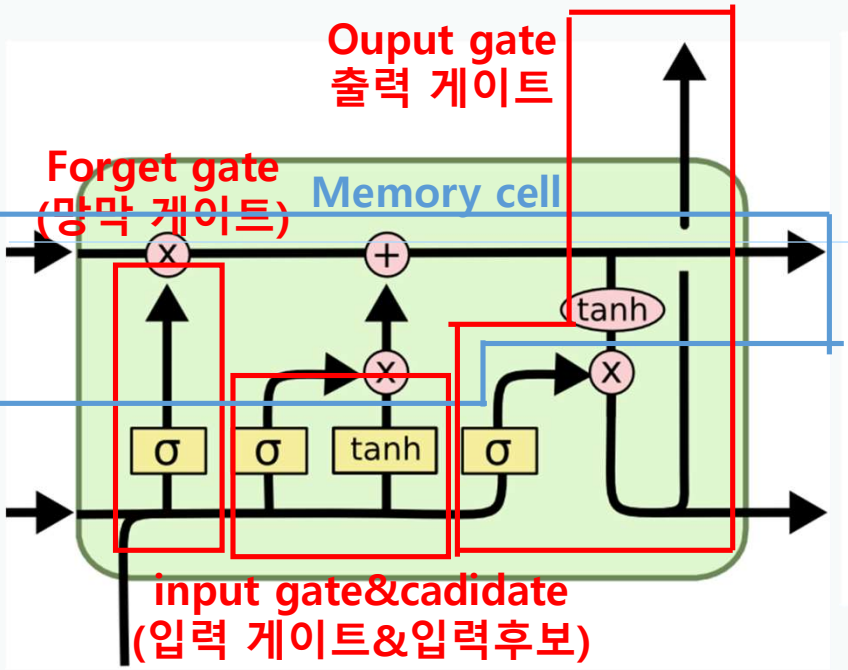
LSTM(Long Short Term Memory)



$$f_t = \sigma(W_{hf}h_{t-1} + W_{xh}x_t)$$
$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t)$$
$$o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_t)$$
$$\tilde{C}_t = \tanh(W_{hC}h_{t-1} + W_{xC}x_t)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * \tanh(C_t)$$

2. LSTM

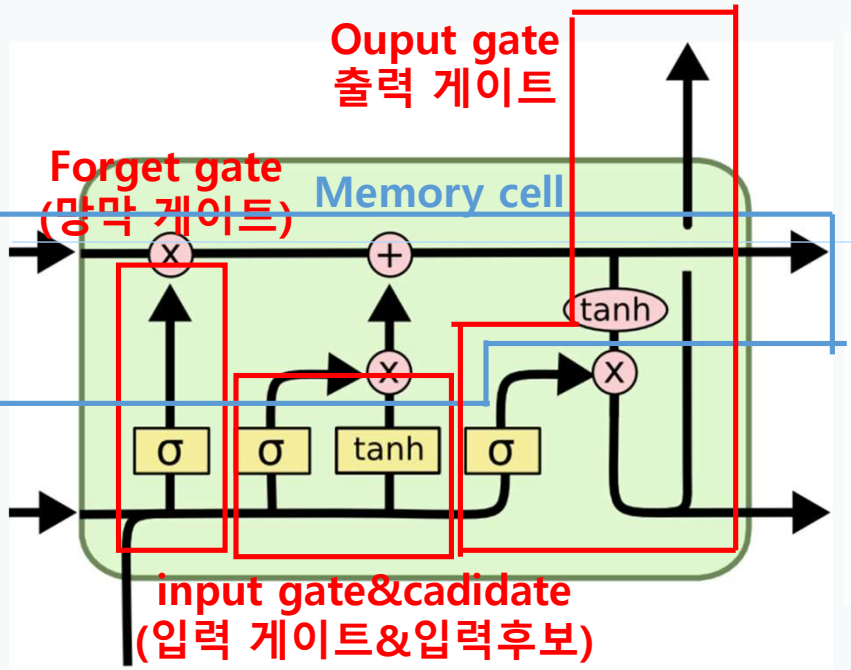
LSTM(Long Short Term Memory)



forget gate	$f_t = \sigma(W_{hf}h_{t-1} + W_{xh}x_t)$
input gate	$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t)$
output gate	$o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_t)$
candidate memory	$\tilde{C}_t = \tanh(W_{hC}h_{t-1} + W_{xC}x_t)$
memory cell	$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
hidden state	$h_t = o_t * \tanh(C_t)$

2. LSTM

LSTM(Long Short Term Memory)



forget gate	$f_t = \sigma(W_{hf}h_{t-1} + W_{xh}x_t)$
input gate	$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t)$
output gate	$o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_t)$
candidate memory	$\tilde{C}_t = \tanh(W_{hC}h_{t-1} + W_{xC}x_t)$
memory cell	$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
hidden state	$h_t = o_t * \tanh(C_t)$

2. LSTM

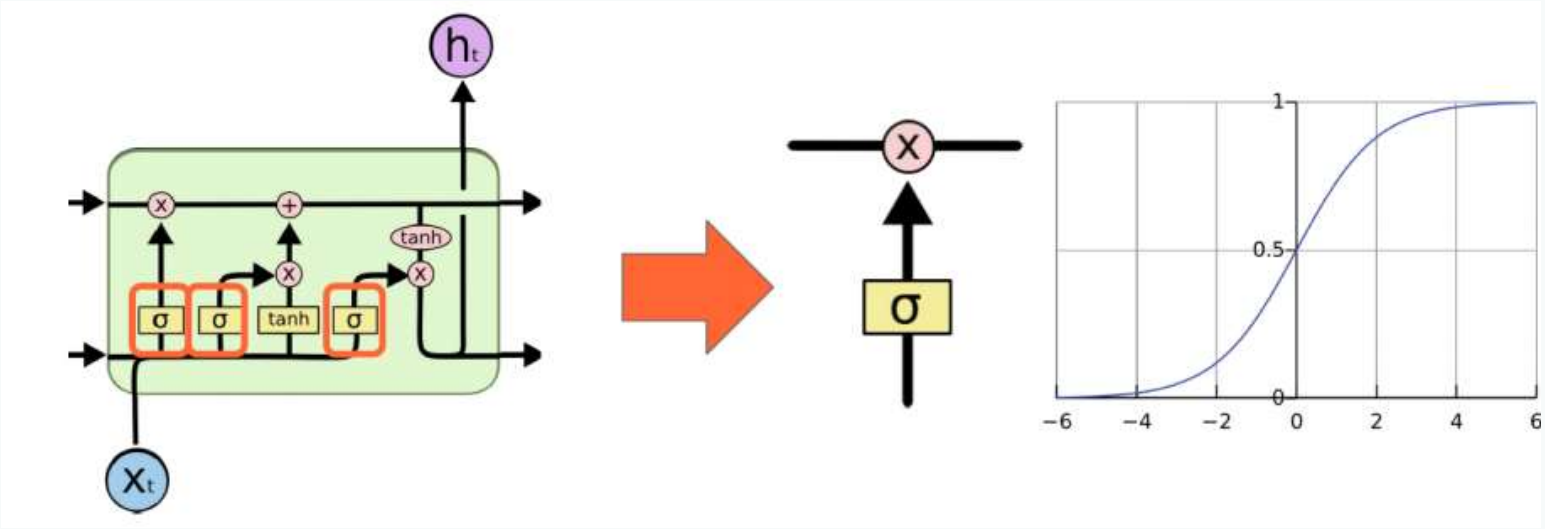
LSTM(Long Short Term Memory)

forget gate	$f_t = \sigma(W_{hf}h_{t-1} + W_{xh}x_t)$
input gate	$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t)$
output gate	$o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_t)$
candidate memory	$\tilde{C}_t = \tanh(W_{hC}h_{t-1} + W_{xC}x_t)$
memory cell	$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
hidden state	$h_t = o_t * \tanh(C_t)$

2. LSTM

LSTM(Long Short Term Memory)

3개의 gate는 sigmoid 유닛으로 제어 됨



2. LSTM

LSTM(Long Short Term Memory)-Forget Gate

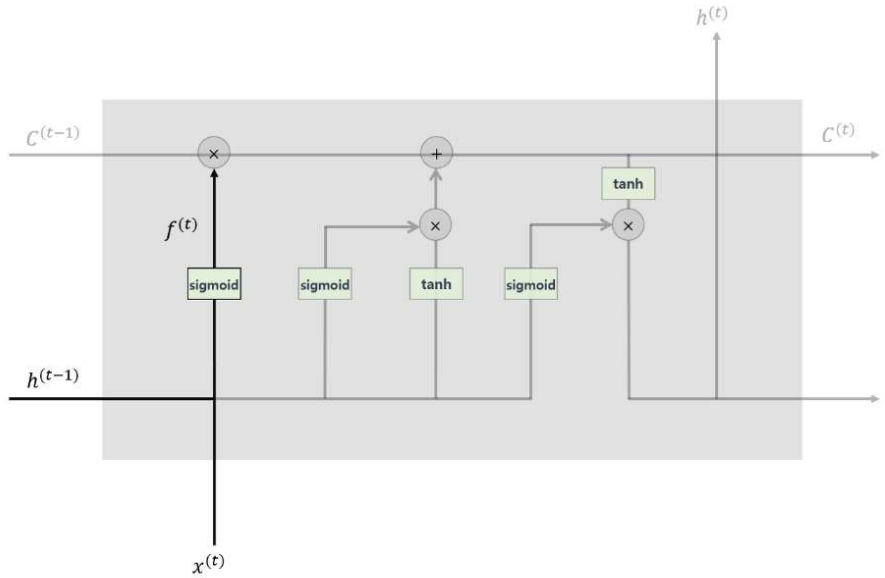


Figure 5: LSTM, Forget Gate

이전 은닉층 값(h_{t-1})과 현재 데이터에 각각 웨이트가 곱해져서 시그모이드로 전달

시그모이드가 결국 0~1사이 출력이므로

1에 가까울수록 과거 정보(ct)를 많이 기억,

0에 가까울 수록 과거 정보 (ct) 를 많이 망각

$$f_t = \sigma(W_{hf}h_{t-1} + W_{xh}x_t)$$

2. LSTM

LSTM(Long Short Term Memory)-Input gate&candidate

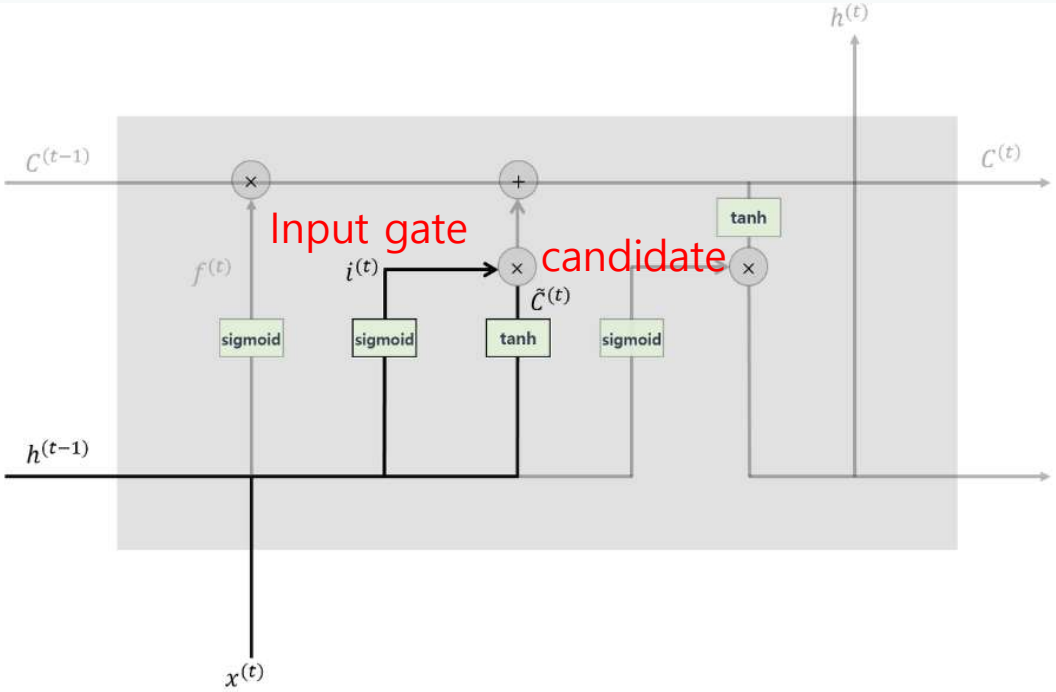


Figure 6: LSTM, Input Gate & Candidate

현 시점의 정보를 셀에 입력할 크기를 정해주는 입력게이트 (input gate)

현 시점의 정보를 계산하는 입력 후보 (cadidate)

현시점이 실제로 갖고 있는 정보(입력후보)가 얼마나 중요한지(입력게이트)를 반영하여 셀에 기록

$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t)$$

$$\tilde{C}_t = \tanh(W_{hC}h_{t-1} + W_{xC}x_t)$$

2. LSTM

LSTM(Long Short Term Memory)-memory cell

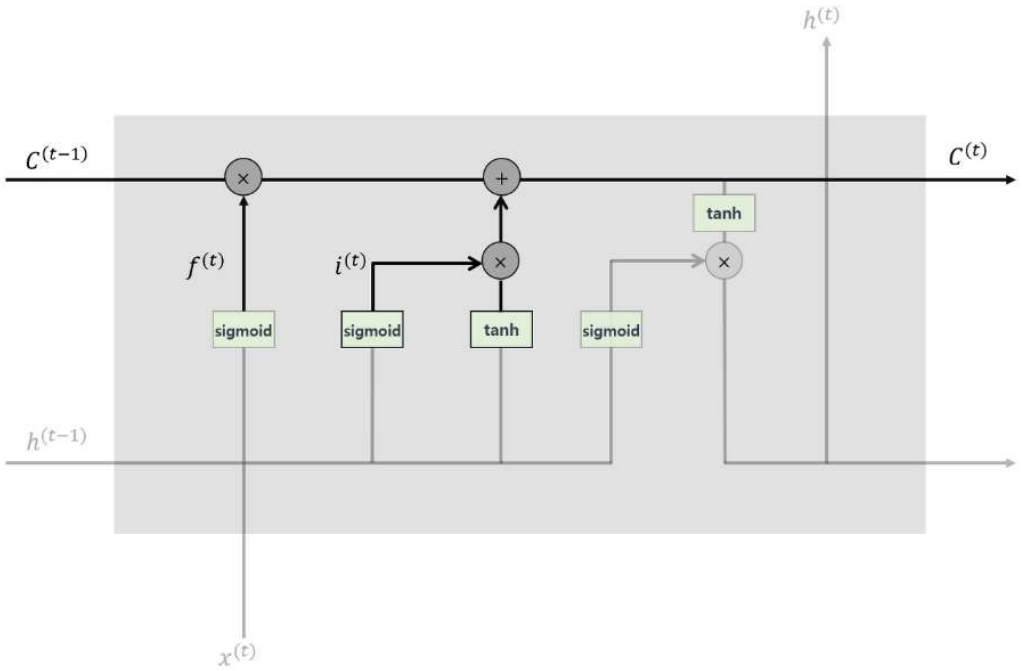


Figure 7: LSTM, Update Memory Cell

Forget gate , input gate, candidate를 이용하여 memory cell에 저장.

과거의 정보를 forget gate에서 계산된 만큼 망각 현시점의 정보 후보에 input gate의 중요도 곱셈 두개를 더하여 현시점 기준 memory cell 계산

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

2. LSTM

LSTM(Long Short Term Memory)-output gate

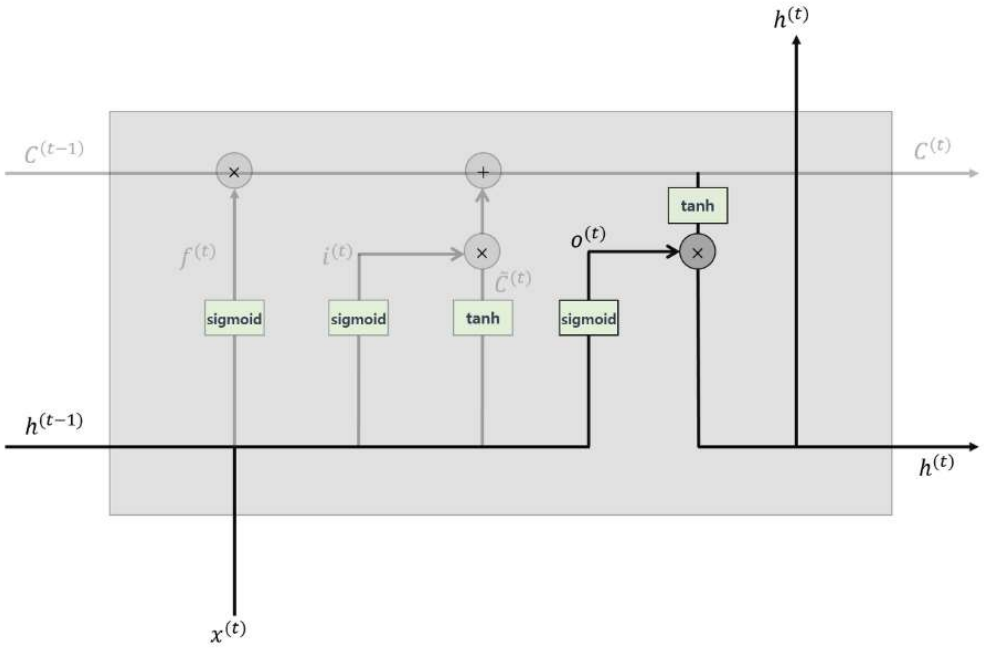


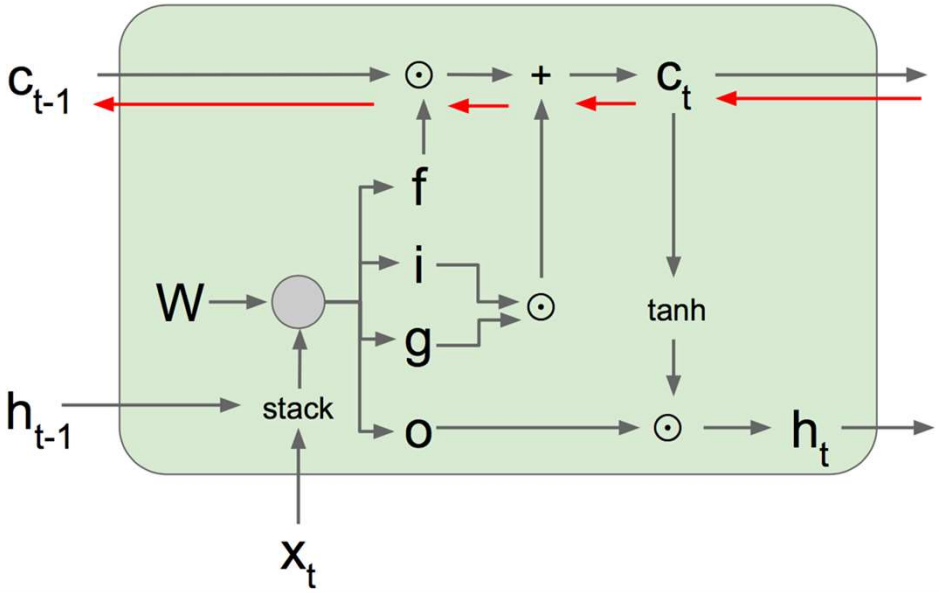
Figure 8: LSTM, Output Gate

현시점의 memory cell을 현시점의 은닉층 값으로 출력할 양을 결정하는 Ouput gate

$$o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_t)$$
$$h_t = o_t * \tanh(C_t)$$

2. LSTM

LSTM(Long Short Term Memory)



Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

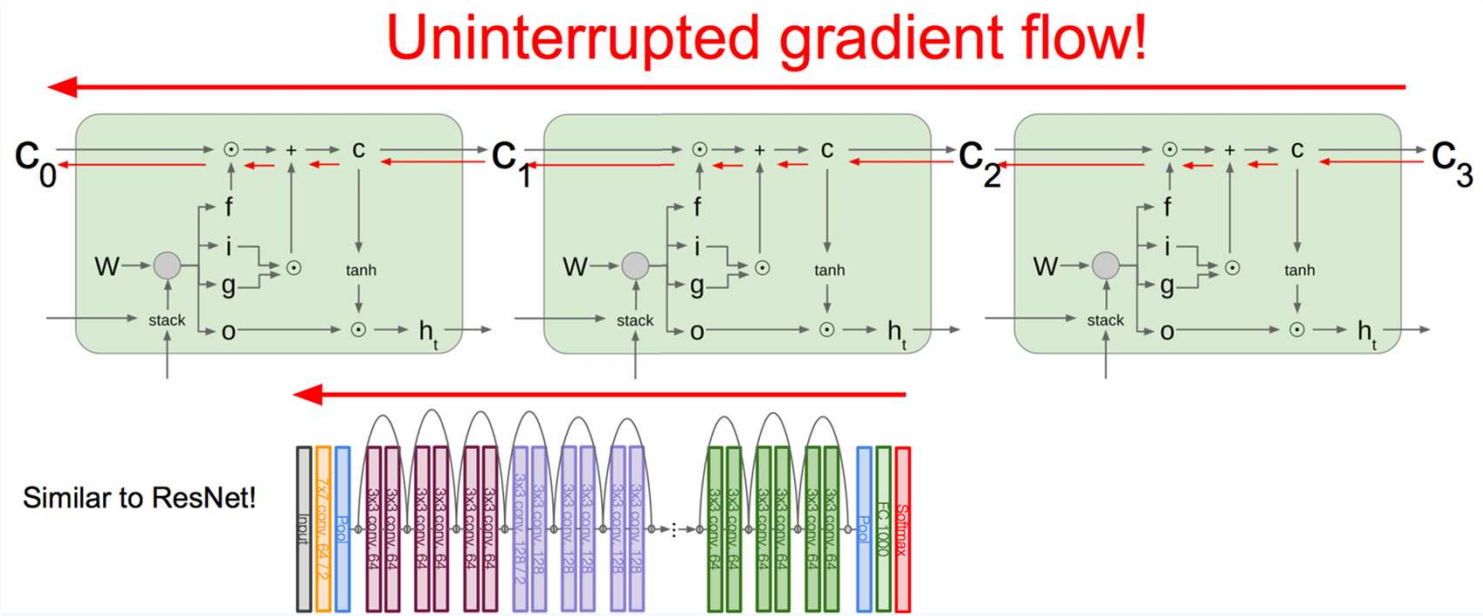
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

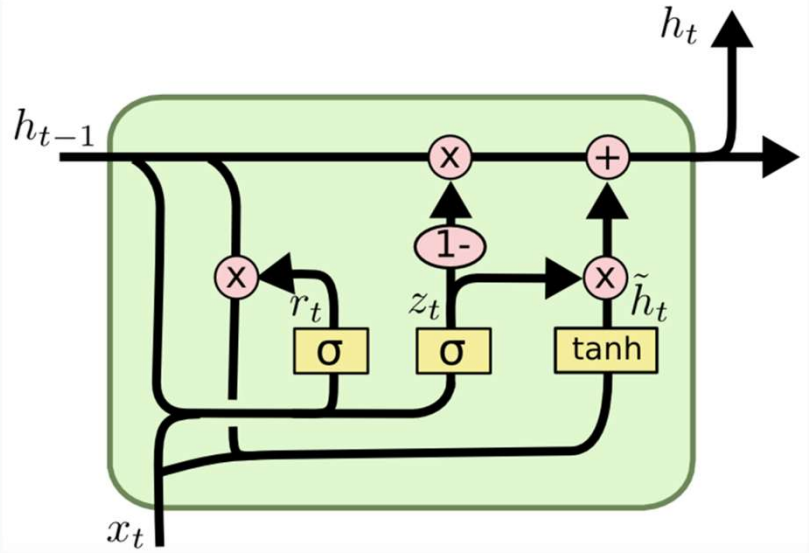
2. LSTM

- **LSTM(Long Short Term Memory)**



2. LSTM

GRU(Gated Recurrent Unit)



$$z_t = \sigma(W_{hz}h_{t-1} + W_{xz}x_t)$$
$$r_t = \sigma(W_{hr}h_{t-1} + W_{xr}x_t)$$
$$\tilde{h}_t = \tanh(W_{hh}(r_t * h_{t-1}) + W_{xh}x_t)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

3. GRU

GRU(Gated Recurrent Unit)

update gate	$z_t = \sigma(W_{hz}h_{t-1} + W_{xz}x_t)$
reset gate	$r_t = \sigma(W_{hr}h_{t-1} + W_{xr}x_t)$
new memory	$\tilde{h}_t = \tanh(W_{hh}(r_t * h_{t-1}) + W_{xh}x_t)$
final memory state	$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

3. GRU

GRU(Gated Recurrent Unit) – reset gate

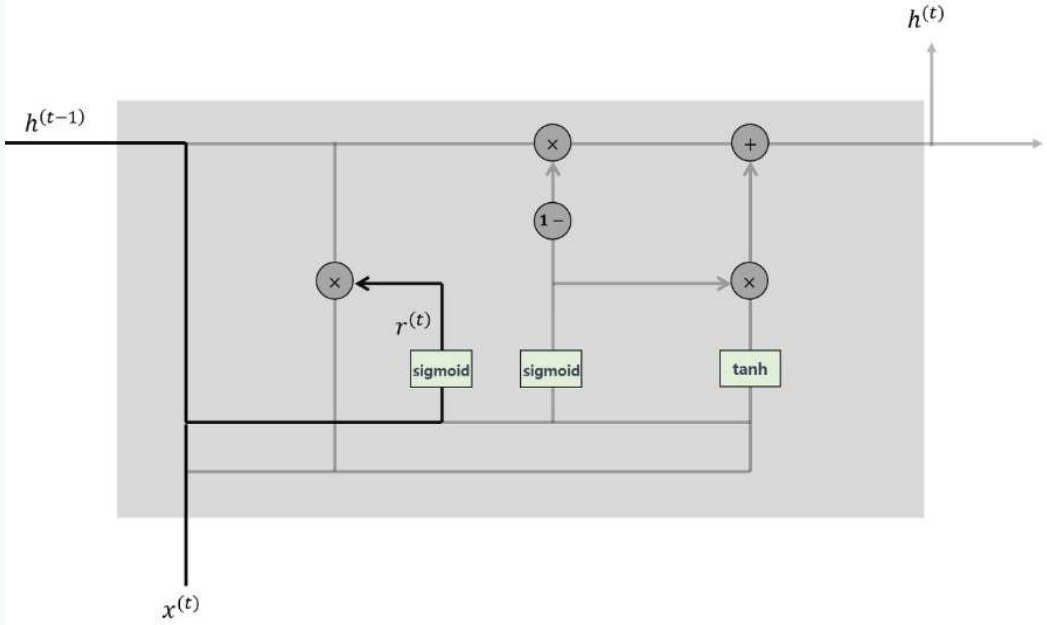


Figure 2: GRU, Reset Gate

과거 정보를 적당히 리셋 하자!

Sigmoid 함수를 출력으로 이용하여 0~1 값을 이전 은닉층에 곱함

$$r_t = \sigma(W_{hr}h_{t-1} + W_{xr}x_t)$$

3. GRU

GRU(Gated Recurrent Unit) – update gate

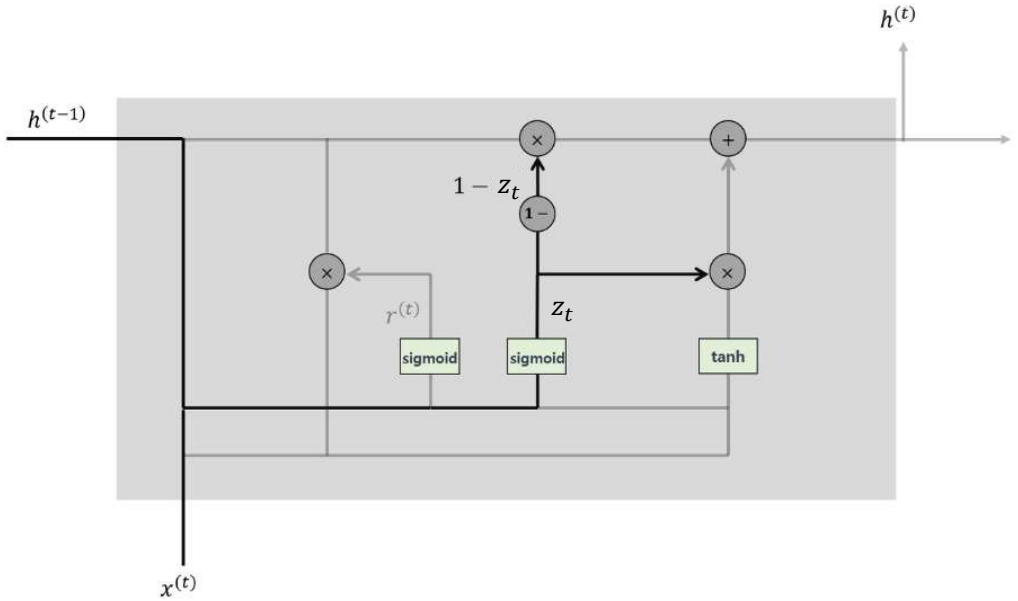


Figure 3: GRU, Update Gate

과거와 현재의 정보를 최신화 비율 결정

Sigmoid로 출력된 결과(u)는 현시점의 정보의 양을 결정.

1에서 뺀값은 직전 시점의 은닉층의 정보에 곱

각각이 input gate와 forger gate와 유사

$$z_t = \sigma(W_{hz}h_{t-1} + W_{xz}x_t)$$

3. GRU

GRU(Gated Recurrent Unit) - candidate

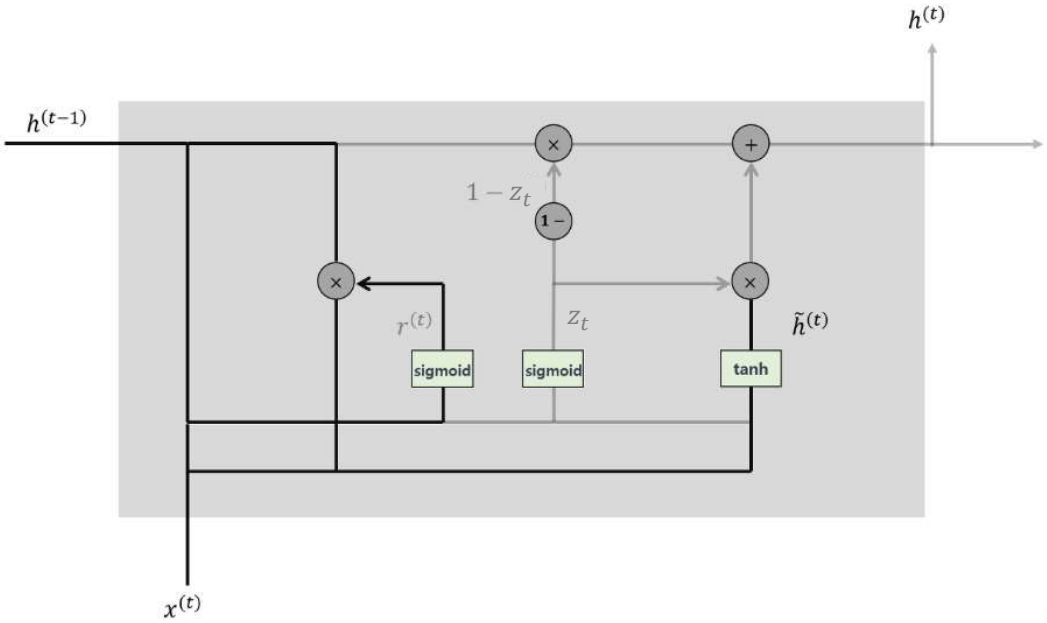


Figure 4: GRU, Candidate

현 시점의 정보 후보군을 계산.

과거 은닉층의 정보를 그대로 이용하지 않고

Reset gate의 결과를 곱하여 이용

$$\tilde{h}_t = \tanh(W_{hh}(r_t * h_{t-1}) + W_{xh}x_t)$$

3. GRU

GRU(Gated Recurrent Unit) – hidden layer

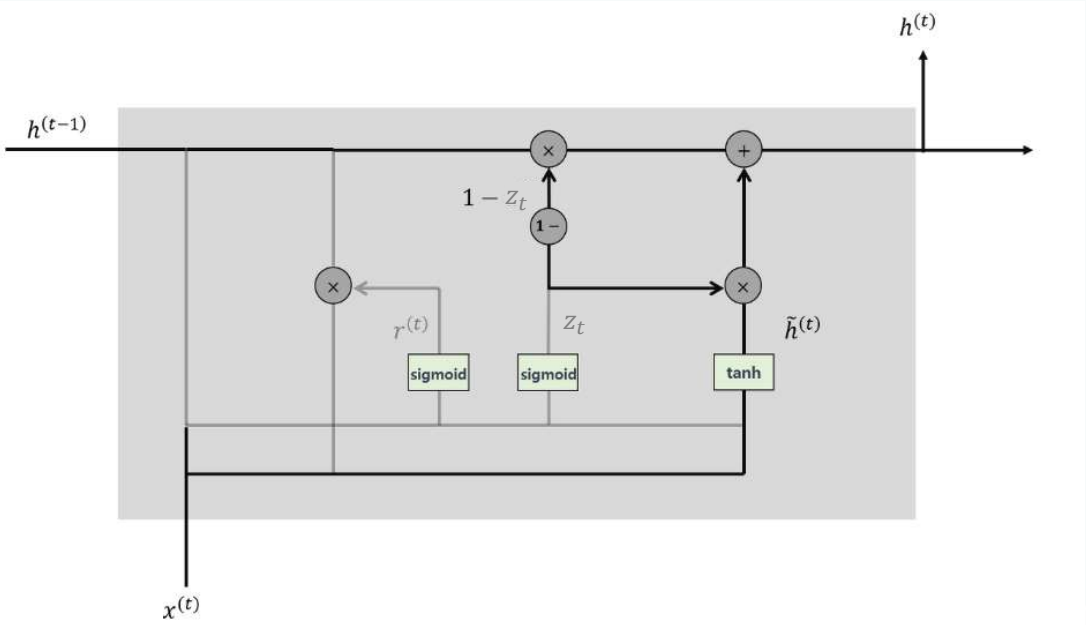


Figure 5: GRU, hidden layer

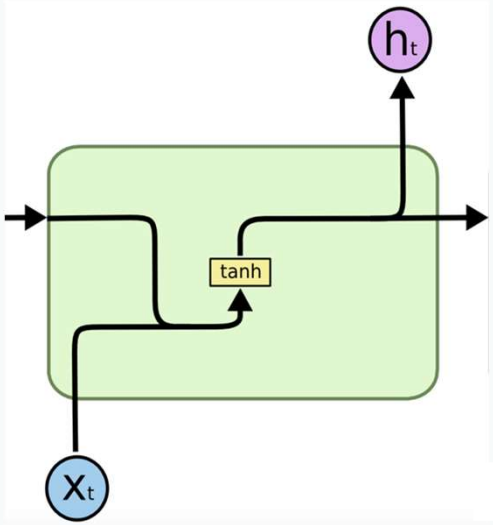
Update gate 결과와 cadidate결과를
결합하여 현시점의 은닉층을 계산.

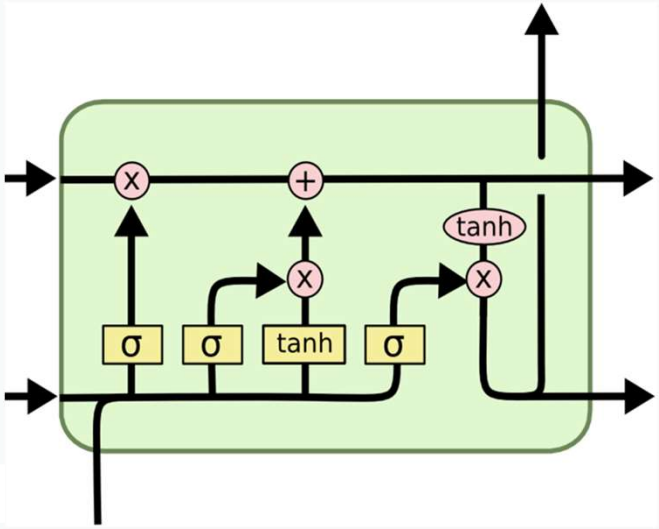
Sigmoid 함수 결과는 현시점 결과의 정보양 결정
1-sigmoid 함수 결과는 과거 시점의 정보양 결정

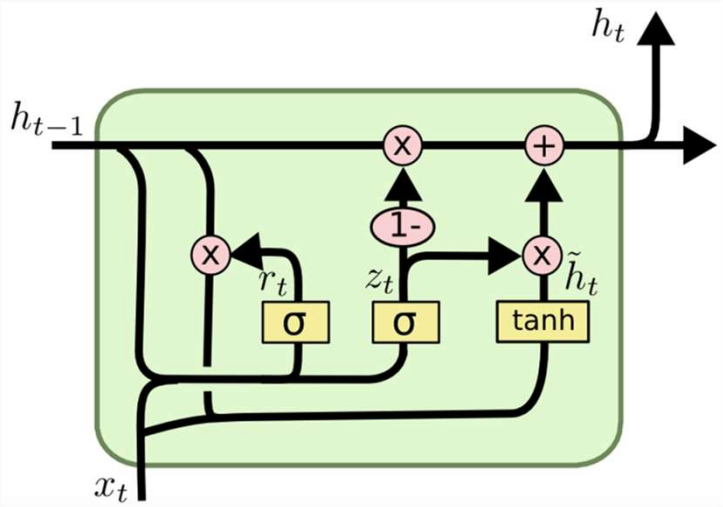
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

3. GRU

- **시계열 데이터 처리 Deep Learning 모델**







RNN **LSTM** **GRU**

4. RNN 실습

DATA 확인

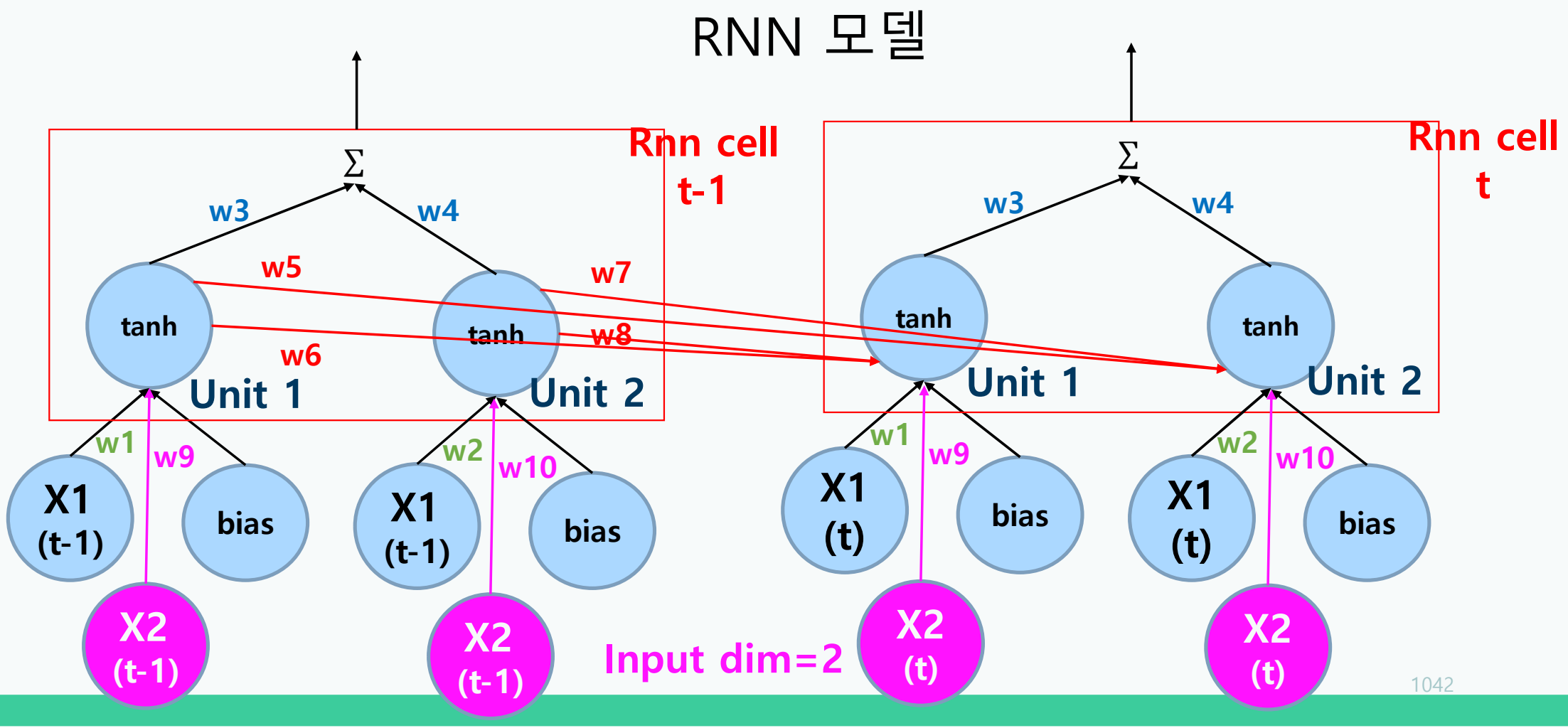
X_dim

Time step
= input length
= sequence

속성

	x1	x2	x3	x4	Y1
T1	2	45	27	9	2
T2	24	68	41	508	3
T3	3	21	2	22	5
T4	55	62	23	215	88
T5	9	2	22	576	2
…	…	…	….	…	…
Tn	123	227	87	124	3

4. RNN 실습


RNN 모델
Rnn cell t-1
Rnn cell t
Σ
w3
w4
w5
w6
w7
w8
tanh
tanh
Unit 1
Unit 2
w1
w9
w2
w10
X1 (t-1)
bias
X2 (t-1)
X1 (t)
X2 (t)
Input dim=2

4. RNN 실습

모델 만들기

모델을 만들어 보겠습니다.

4. RNN 실습

모델 만들기

모델을 만들어 보겠습니다.

```
model.add(SimpleRNN(32,input_shape=(batch_size,time_step,input_dim),
                    activation = 'tanh',
                    return_sequences=True,
                    stateful=True ))
```

4. RNN 실습

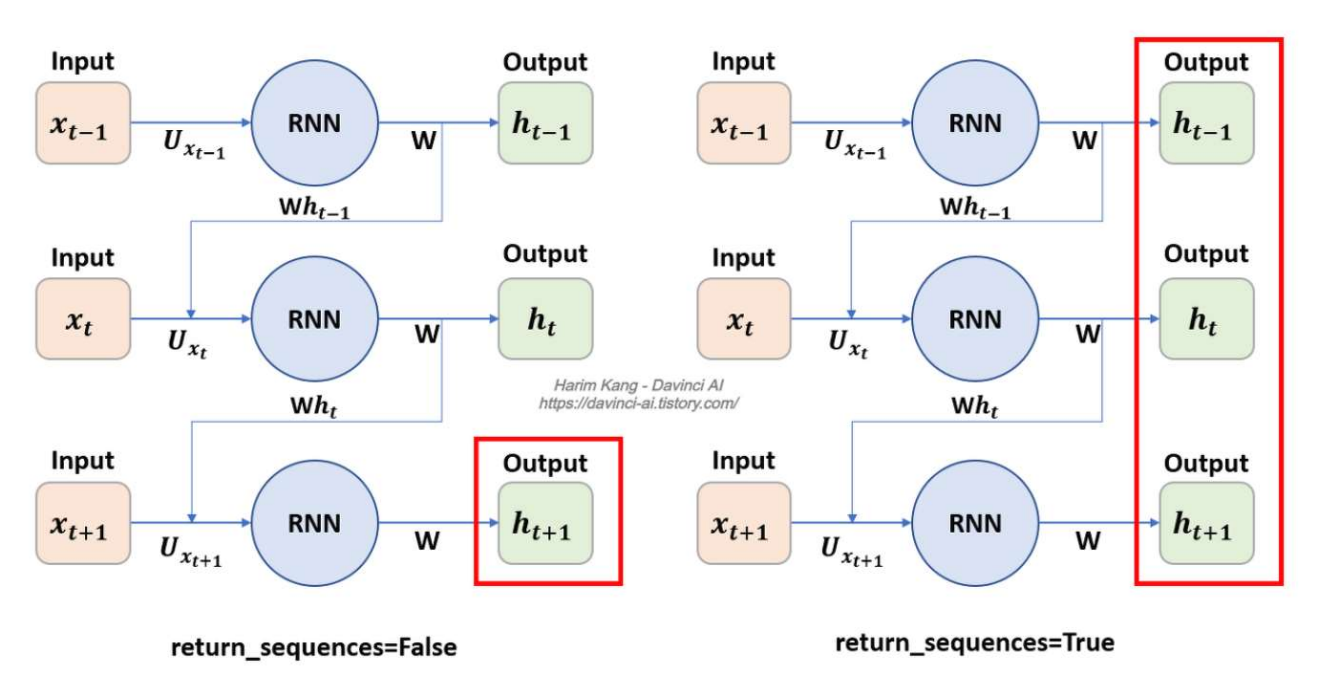
모델 만들기

모델을 만들어 보겠습니다.

```
model.add(SimpleRNN(32,input_shape=(batch_size,time_step,input_dim),
                    activation = 'tanh',
                    return_sequences=True,
                    stateful=True ))
```

4. RNN 실습

모델 만들기

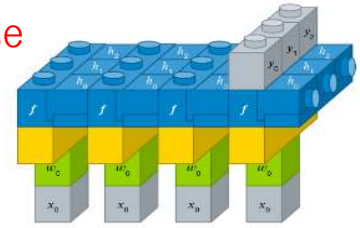


주로 One-to-many , many-to-many 출력을 위해 사용

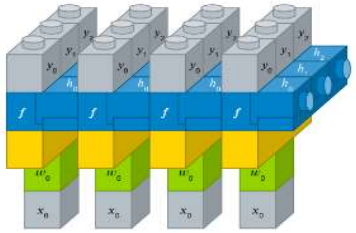
4. RNN 실습

모델 만들기

False



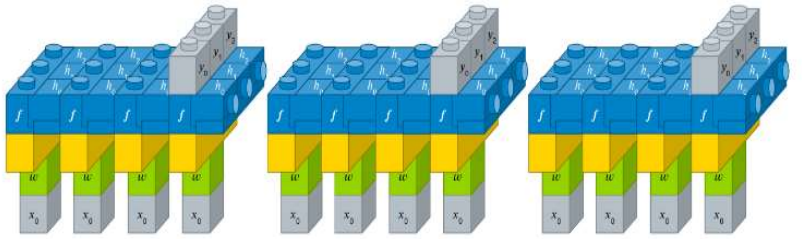
True



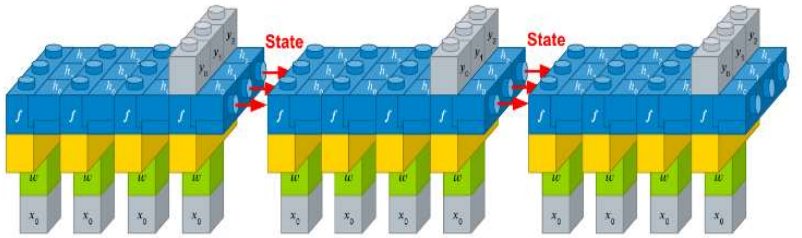
return_sequences

각시퀀스에서 출력 할것인지
마지막 시퀀스에서 출력할 것인지

False



True



stateful

학습 샘플의 가장 마지막 상태가 다음
샘플 학습시에 입력으로 전달의 여부

4. RNN 실습

RNN Regression(prediction)

주식 데이터를 통해 가격을 예측해 보겠습니다.

4. RNN 실습

- **RNN Regression(prediction) – 주식 예측**

Input dim

Time step = input length = sequence

	Open	High	Low	Volume	Close
1	828.659973	833.450012	828.349976	1247700	831.659973
2	823.02002	828.070007	821.655029	1597800	828.070007
3	819.929993	824.400024	818.97998	1281700	824.159973
4	819.359985	823	818.469971	1304000	818.97998
5	819	823	816	1053600	820.450012
…	…	…	….	…	…
733	568.00257	568.00257	552.922516	13100	558.462551

속성 속성 속성

4. RNN 실습

- **RNN Regression(prediction) – 주식 예측**

그런데 시계열 데이터는 어떻게 읽어오나요?

4. RNN 실습

- **RNN Regression(prediction) – 주식 예측**

tf.keras.preprocessing.sequence.TimeseriesGenerator 사용

4. RNN 실습

- **RNN Regression(prediction) – 주식 예측**

`tf.keras.preprocessing.sequence.TimeseriesGenerator`

시계열 전처리 함수

4. RNN 실습

• RNN Regression(prediction) – 주식 예측

```
data_dir = '../dataset'
fname = os.path.join(data_dir, 'data-02-stock_daily.csv')

## 데이터 읽어오기.
df = pd.read_csv(fname)
dataset = df.values
ori_X = dataset[:,0:4]
ori_Y = dataset[:,4]
```

Data 읽어와서
입력/정답 나누기

```
seq_length=4
X_train, X_test, Y_train, Y_test = train_test_split(ori_X,ori_Y, test_size=0.2, shuffle=False)
```

Train / test 분리

```
data_gen=tf.keras.preprocessing.sequence.TimeseriesGenerator(X_train, Y_train,
                                length=seq_length, sampling_rate=1,
                                batch_size=1,
                                start_index=0)
```

데이터 불러오기

```
for i in range(2):
    x, y = data_gen[i]
    print(x, y)
```

데이터 프린트

Practice : P_05_00_time_data.ipynb

4. RNN 실습

• RNN Regression(prediction) – 주식 예측

```
seq_length=4
X_train, X_test, Y_train, Y_test = train_test_split(ori_X,ori_Y, test_size=0.2, shuffle=False)
data_gen=tf.keras.preprocessing.sequence.TimeseriesGenerator(X_train, Y_train,
                                length=seq_length, sampling_rate=1,
                                batch_size=1,
                                start_index=0)
for i in range(2):
```

Start index

0번째에서 4개의 시간 길이에서 4개의 데이터 셋을 가져옴

입력

```
Open,High,Low,Volume,Close
828.659973,833.450012,828.349976,1247700,831.659973
823.02002,828.070007,821.655029,1597800,828.070007
819.929993,824.400024,818.97998,1281700,824.159973
819.359985,823,818.469971,1304000,818.97998
819,823,816,1053600,820.450012
816,820.958984,815.48999,1198100,819.23999
811.700012,815.25,809.780029,1129100,813.669983
809.51001,810.659973,804.539978,989700,809.559998
807,811.840027,803.190002,1155300,808.380005
```

정답

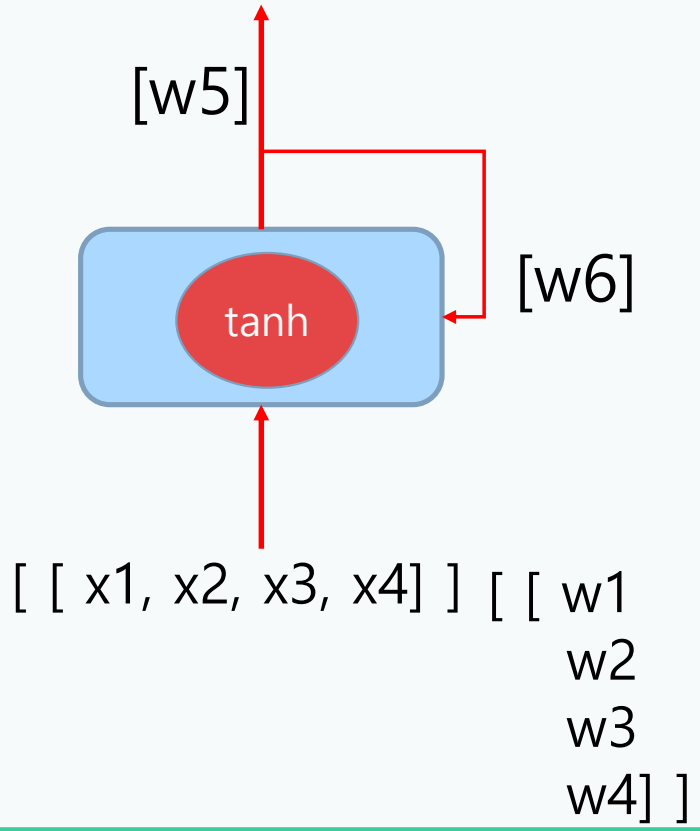
length : 읽어올 데이터 시간길이

sampling_rate : length 길이 안에서 sampling rate

Practice : P_05_00_time_data. ipynb

4. RNN 실습

• RNN Regression(prediction) – 주식 예측



seq_length = 1, x_data_dim = 4

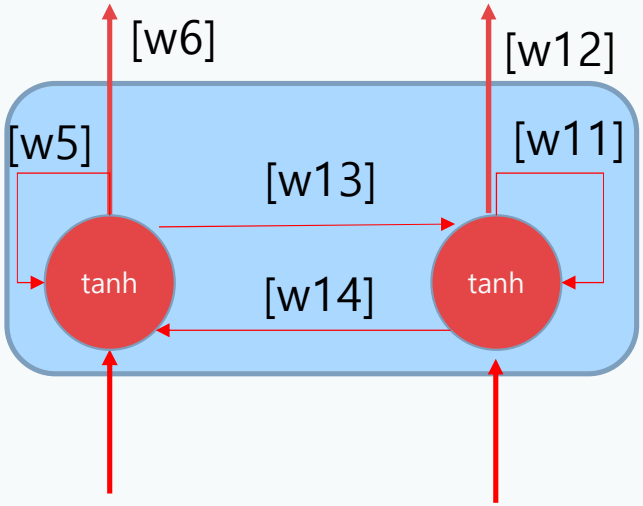
```
## 모델
input_Layer = tf.keras.layers.Input(shape=(seq_length, x_data_dim))
Out_Layer= tf.keras.layers.SimpleRNN(1, activation='tanh')(input_Layer)
model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1, 4)]            0
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 1)                 6
=================================================================
Total params: 6
Trainable params: 6
Non-trainable params: 0
```


Practice : P_05_01_RNN_stock_prediction. ipynb

4. RNN 실습

• RNN Regression(prediction) – 주식 예측



[[x1, x2, x3, x4]] [[x1, x2, x3, x4]]

[[w1 [[w7

w2 w8

w3 w9

w4]] w10]]

seq_length = 1 , x_data_dim = 4

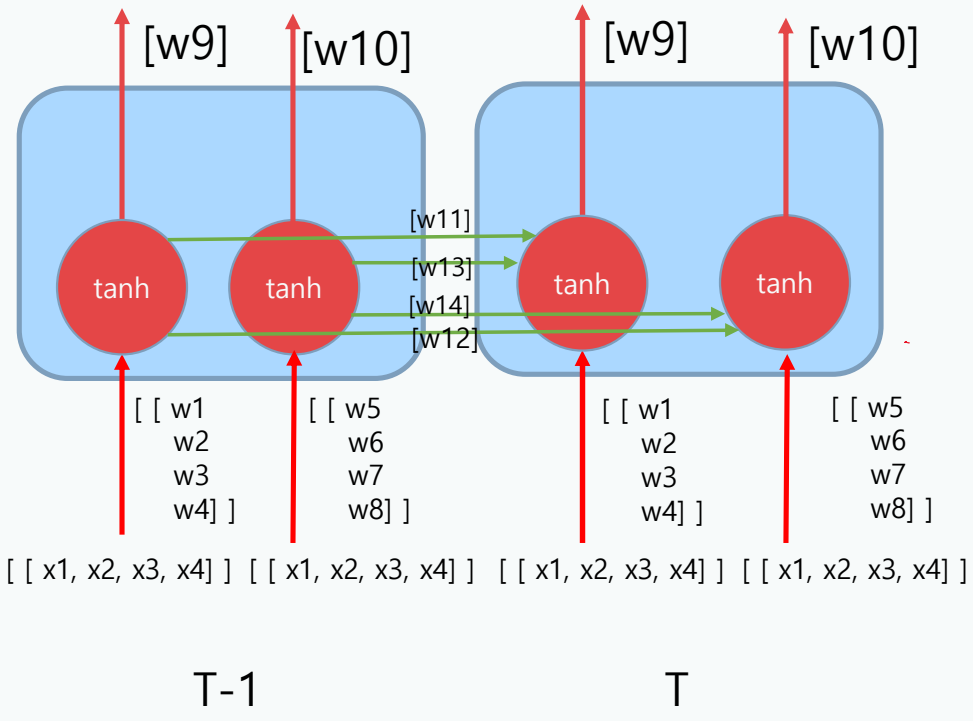
```
66  ## 모델
67  input_Layer = tf.keras.layers.Input(shape=(seq_length, x_data_dim))
68  Out_Layer= tf.keras.layers.SimpleRNN(2, activation='tanh')(input_Layer)
69  model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
70  model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1, 4)]            0
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 2)                 14
=================================================================
Total params: 14
Trainable params: 14
Non-trainable params: 0
```


Practice : P_05_01_RNN_stock_prediction. ipynb

4. RNN 실습

• RNN Regression(prediction) – 주식 예측



seq_length = 2 , x_data_dim = 4

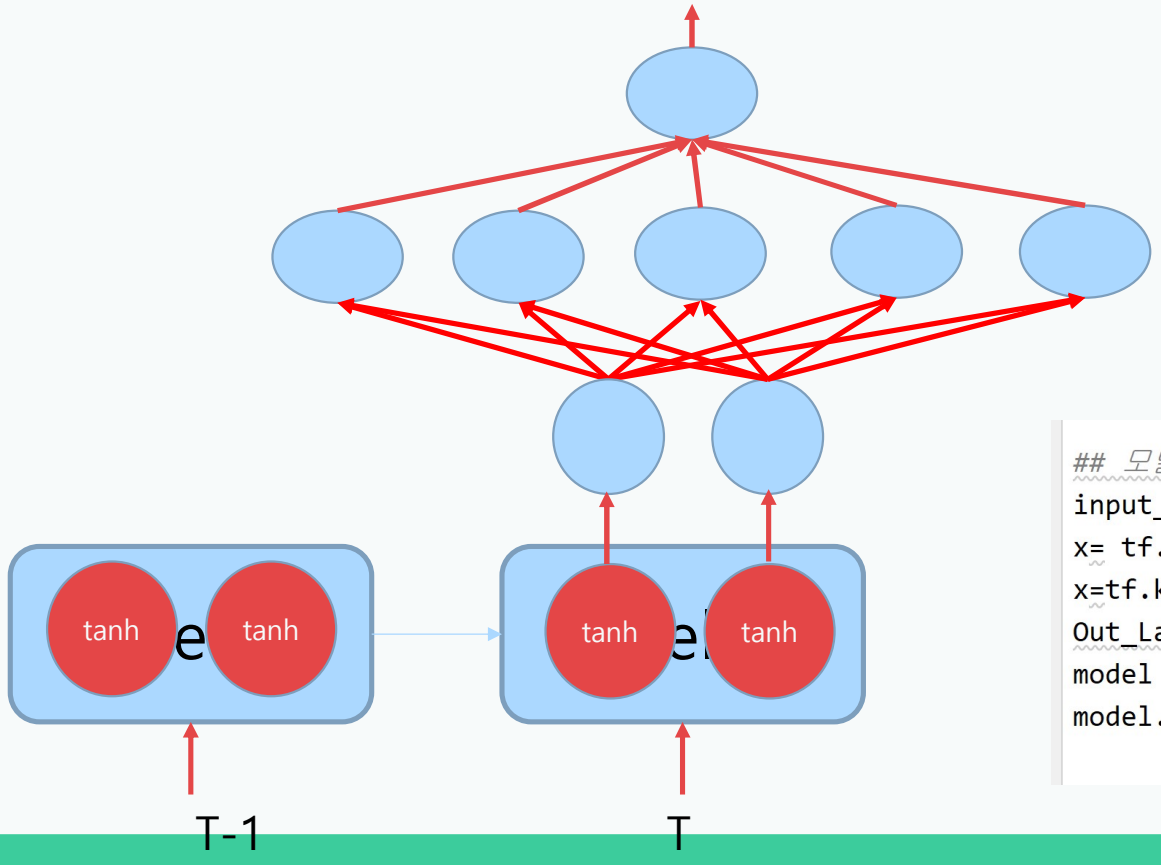
```
66  ## 모델
67  input_Layer = tf.keras.layers.Input(shape=(seq_length, x_data_dim))
68  Out_Layer= tf.keras.layers.SimpleRNN(2, activation='tanh')(input_Layer)
69  model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
70  model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2, 4)]            0
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 2)                 14
=================================================================
Total params: 14
Trainable params: 14
Non-trainable params: 0
```


Practice : P_05_01_RNN_stock_prediction. ipynb

4. RNN 실습

- **RNN Regression(prediction) – 주식 예측**



seq_length = 2 , x_data_dim = 4

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 2, 4)]	0
simple_rnn (SimpleRNN)	(None, 2)	14
dense (Dense)	(None, 5)	15
dense_1 (Dense)	(None, 1)	6

```
## 모델
input_Layer = tf.keras.layers.Input(shape=(seq_length,x_data_dim))
x= tf.keras.layers.SimpleRNN(2)(input_Layer)
x=tf.keras.layers.Dense(5)(x)
Out_Layer=tf.keras.layers.Dense(1)(x)
model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()
```


Practice : P_05_01_RNN_stock_prediction.ipynb

4. RNN 실습

• RNN Regression(prediction) – 주식 예측

데이터 읽어와서 train / val / test 나누기

```
seq_length=5
x_data_dim=4
batch_size=5
min_max_normalization_flag=True

data_dir = '../dataset'
fname = os.path.join(data_dir, 'data-02-stock_daily.csv')
df = pd.read_csv(fname)
dataset=df.copy()
ori_Y=dataset.pop("Close")
ori_X=dataset.copy()

X_train, X_test, Y_train, Y_test = train_test_split(ori_X,ori_Y, test_size=0.2, shuffle=False)
X_train, X_val, Y_train, Y_val= train_test_split(X_train,Y_train, test_size=0.2, shuffle=False)

## 데이터의 min , max, mean, std 값 구하기.
```


Practice : P_05_01_RNN_stock_prediction.ipynb

4. RNN 실습

• RNN Regression(prediction) – 주식 예측

```
## 모델
input_Layer = tf.keras.layers.Input(shape=(seq_length, x_data_dim))
x= tf.keras.layers.SimpleRNN(50,activation='tanh')(input_Layer) ## RNN
x=tf.keras.layers.Dense(100,activation='relu')(x)
x=tf.keras.layers.Dense(50,activation='relu')(x)
Out_Layer=tf.keras.layers.Dense(1,activation=None)(x)
model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()

model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()

loss_function=tf.keras.losses.mean_squared_error
optimize=tf.keras.optimizers.Adam(learning_rate=0.001)
metric=tf.keras.metrics.mean_absolute_error
model.compile(loss=loss_function,
              optimizer=optimize,
              metrics=[metric])

history = model.fit(sequence_train_x, sequence_train_y, epochs=500, batch_size=100, validation_data=(sequence_val_x,sequence_val_y))
print(model.evaluate(sequence_test_x, sequence_test_y))
```

모델 설계

모델 loss, optimizer, metric 선언 compile

모델 학습

모델 학습 완료 후 평가 및 테스트

Practice : P_05_01_RNN_stock_prediction.ipynb

4. RNN 실습

• LSTM Regression(prediction) – 주식 예측

```
65  ## 모델
66  input_Layer = tf.keras.layers.Input(shape=(seq_length, x_data_dim))
67  x=tf.keras.layers.LSTM(50,activation='tanh')(input_Layer) ##LSTM
68  x=tf.keras.layers.Dense(100,activation='relu')(x)
69  x=tf.keras.layers.Dense(50,activation='relu')(x)
70  Out_Layer=tf.keras.layers.Dense(1,activation=None)(x)
71  model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
72  model.summary()
73
74  model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
75  model.summary()
76
77  loss_function=tf.keras.losses.mean_squared_error
78  optimize=tf.keras.optimizers.Adam(learning_rate=0.001)
79  metric=tf.keras.metrics.mean_absolute_error
80  model.compile(loss=loss_function,
81                optimizer=optimize,
82                metrics=[metric])
83
84  history = model.fit(sequence_train_x, sequence_train_y, epochs=500, batch_size=100, validation_data=(sequence_val_x,sequence_val_y))
85  print(model.evaluate(sequence_test_x, sequence_test_y))
```

모델 설계

모델 loss, optimizer, metric 선언 compile

모델 학습

모델 학습 완료 후 평가 및 테스트

Practice : P_05_01_LSTM_stock_prediction.ipynb

4. RNN 실습

• GRU Regression(prediction) – 주식 예측

```
66  input_Layer = tf.keras.layers.Input(shape=(seq_length, x_data_dim))
67  x=tf.keras.layers.GRU(50,activation='tanh')(input_Layer) ##GRU
68  x=tf.keras.layers.Dense(100,activation='relu')(x)
69  x=tf.keras.layers.Dense(50,activation='relu')(x)
70  Out_Layer=tf.keras.layers.Dense(1,activation=None)(x)
71  model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
72  model.summary()
73
74  model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
75  model.summary()
76
77  loss_function=tf.keras.losses.mean_squared_error
78  optimize=tf.keras.optimizers.Adam(learning_rate=0.001)
79  metric=tf.keras.metrics.mean_absolute_error
80  model.compile(loss=loss_function,
81                optimizer=optimize,
82                metrics=[metric])
83
84  history = model.fit(sequence_train_x, sequence_train_y, epochs=500, batch_size=100, validation_data=(sequence_val_x,sequence_val_y))
85  print(model.evaluate(sequence_test_x, sequence_test_y))
86
```

모델 설계

모델 loss, optimizer, metric 선언 compile

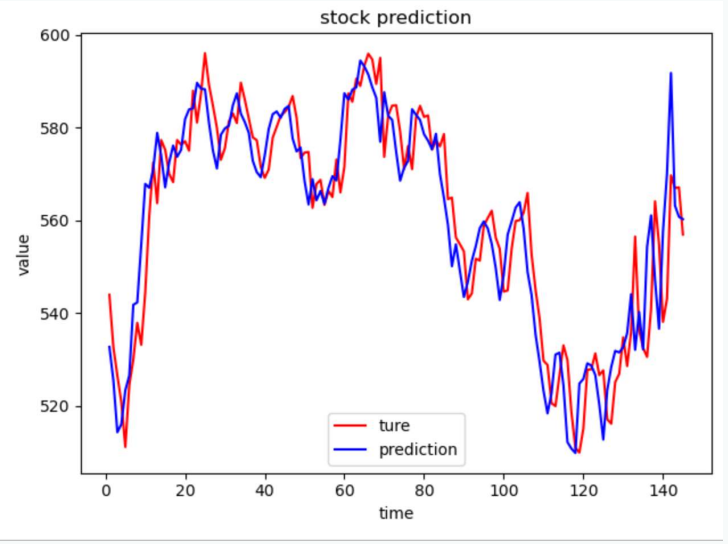
모델 학습

모델 학습 완료 후 평가 및 테스트

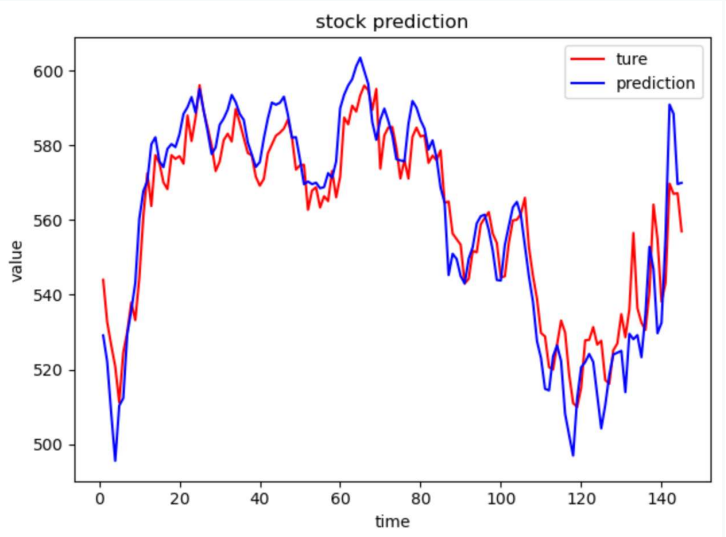
Practice : P_05_01_GRU_stock_prediction.ipynb

4. RNN 실습

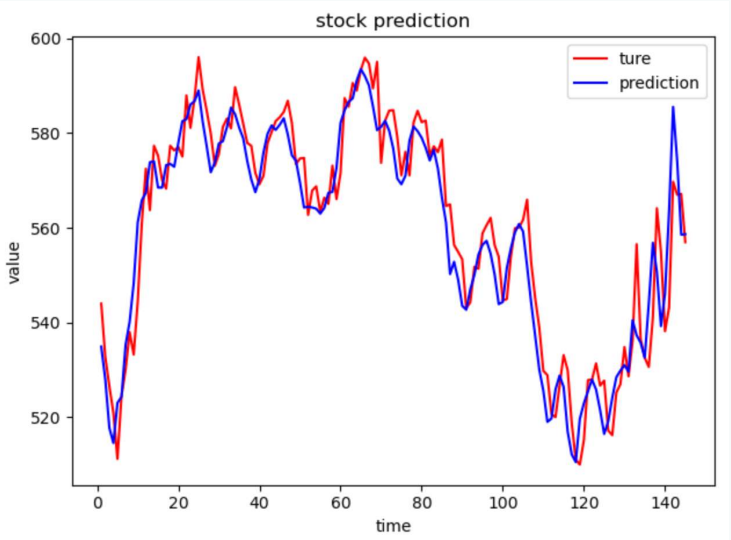
- **주식 예측 비교** **Seq_length=2**



rnn



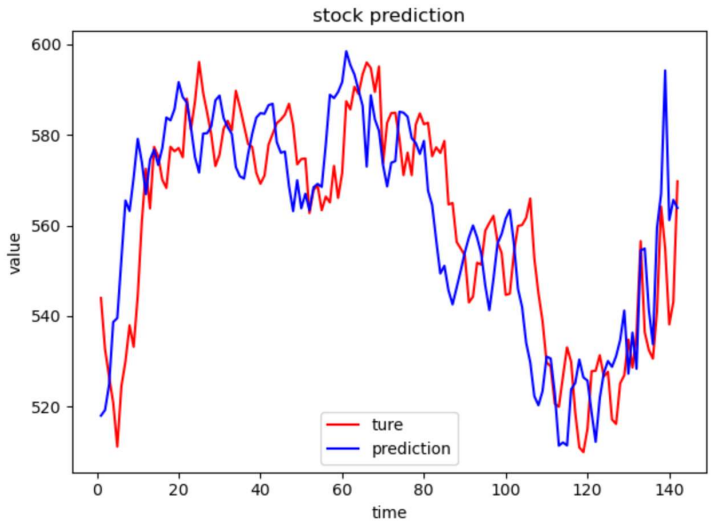
lstm



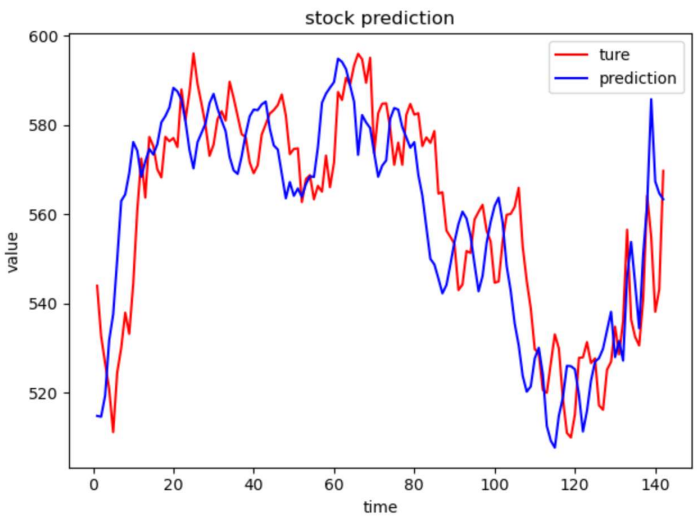
gru

4. RNN 실습

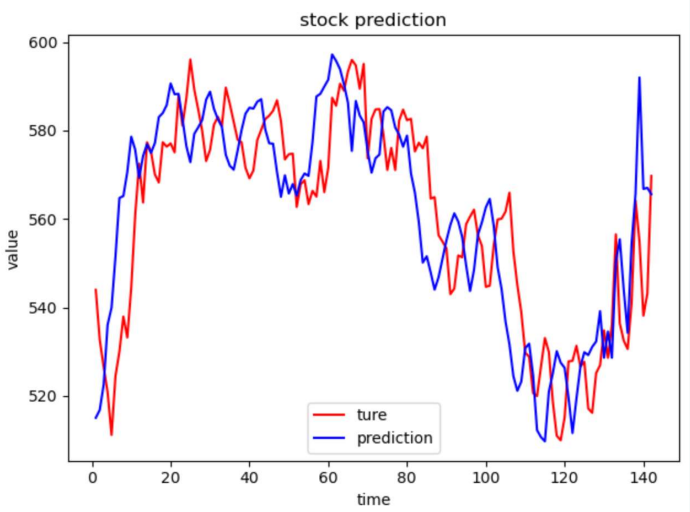
- **주식 예측 비교** **Seq_length=5**



rnn



lstm



gru

4. RNN 실습

RNN Regression(prediction) – A company 주식 예측

A_company_stock.csv 데이터

	Open	High	Low	Volume	Close
1	29700	30000	29680	30000	10588400
2	29960	30040	29300	29300	12647050
3	29800	30000	29360	30000	11357700
4	29840	30040	29780	29980	10887800
5	30040	30040	29440	29700	8009300
...	...	...	....	...	...
1226	40850	41950	40550	41400	14460521

Exersice : 05_01_a_company_stock_prediction.ipynb

4. RNN 실습

- **RNN Regression(prediction)**

성능을 더 높이거나 오버피팅을 막는 기법은 없을까요?

4. RNN 실습

RNN Regression(prediction)

1. **Stacking 순환 층**

   네트워크이 표현 능력을 증가.단, 계산 비용이 커짐

2. **순환 dropout**

   순환 층에서 overfitting을 방지하기 위해 케라스에 내장되어 있는 dropout

3. **양방향 순환 층**

   순환 네트워크에 같은 정보를 다른 방향으로 주입하여 정확도를 높이고 기억을 좀더 오래 보존하는 것.

4. RNN 실습

RNN Regression(prediction)

1. **Stacking 순환 층**

   네트워크이 표현 능력을 증가.단, 계산 비용이 커짐

2. **순환 dropout**

   순환 층에서 overfitting을 방지하기 위해 케라스에 내장되어 있는 dropout

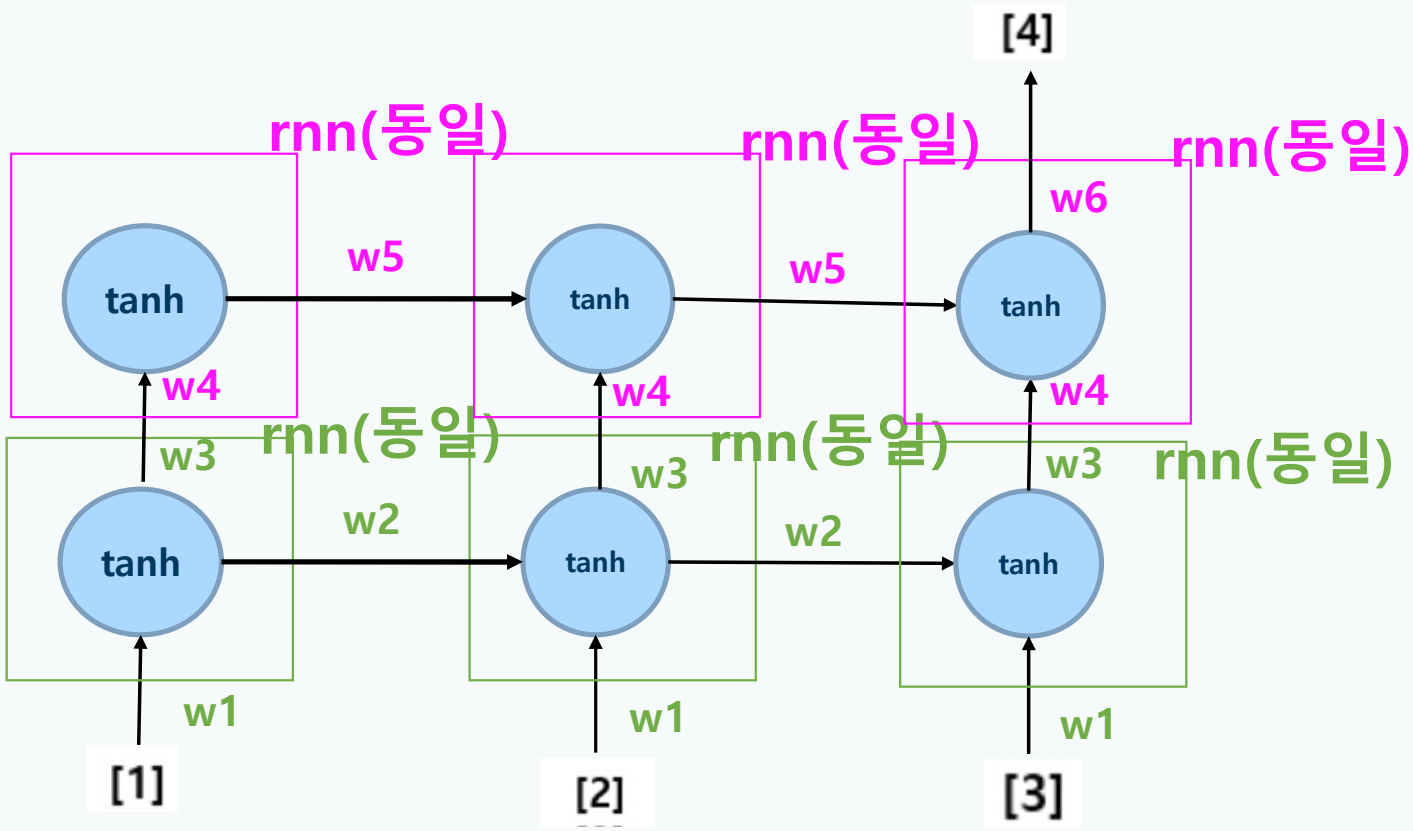
3. **양방향 순환 층**

   순환 네트워크에 같은 정보를 다른 방향으로 주입하여 정확도를 높이고 기억을 좀더 오래 보존하는 것.

4. RNN 실습

Many – to –One

$$[1, 2, 3] \rightarrow [4]$$




```
return_sequences=True
```


Practice : P_05_02_rnn_stock_prediction_stacking.py

4. RNN 실습

• RNN Regression(prediction) – 주식 예측

```
## 모델
input_Layer = tf.keras.layers.Input(shape=(seq_length,x_data_dim))
x= tf.keras.layers.SimpleRNN(20,return_sequences=True)(input_Layer)
x=tf.keras.layers.SimpleRNN(10)(x)
x=tf.keras.layers.Dense(30)(x)
Out_Layer=tf.keras.layers.Dense(1)(x)

model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()


loss_function=tf.keras.losses.mean_squared_error
optimize=tf.keras.optimizers.Adam(lr=0.001)
metric=tf.keras.metrics.mean_absolute_error
model.compile(loss=loss_function,
              optimizer=optimize,
              metrics=[metric])

history = model.fit_generator(
      data_gen_train,
      validation_data=data_gen_val,
      steps_per_epoch=50,
      epochs=500,
      validation_steps=1
)
print(model.evaluate_generator(data_gen_test))
prediction_Y=model.predict_generator(data_gen_test).flatten()
```

모델 설계

모델 loss, optimizer, metric 선언 compile

모델 학습

모델 학습 완료 후 평가 및 테스트

Practice : P_05_02_rnn_stock_prediction_stacking.ipynb

4. RNN 실습

• RNN Regression(prediction) – 주식 예측

```
## 모델
input_Layer = tf.keras.layers.Input(shape=(seq_length,x_data_dim))
x= tf.keras.layers.SimpleRNN(20,return_sequences=True)(input_Layer)
x=tf.keras.layers.SimpleRNN(10)(x)
x=tf.keras.layers.Dense(30)(x)
Out_Layer=tf.keras.layers.Dense(1)(x)

model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()


loss_function=tf.keras.losses.mean_squared_error
optimize=tf.keras.optimizers.Adam(lr=0.001)
metric=tf.keras.metrics.mean_absolute_error
model.compile(loss=loss_function,
              optimizer=optimize,
              metrics=[metric])

history = model.fit_generator(
      data_gen_train,
      validation_data=data_gen_val,
      steps_per_epoch=50,
      epochs=500,
      validation_steps=1
)
print(model.evaluate_generator(data_gen_test))
prediction_Y=model.predict_generator(data_gen_test).flatten()
```

모델 설계

모델 loss, optimizer, metric 선언 compile

모델 학습

모델 학습 완료 후 평가 및 테스트

Practice : P_05_02_rnn_stock_prediction_stacking.ipynb

4. RNN 실습

RNN Regression(prediction)

1. Stacking 순환 층

   네트워크이 표현 능력을 증가.단, 계산 비용이 커짐

2. 순환 dropout

   순환 층에서 overfitting을 방지하기 위해 케라스에 내장되어 있는 dropout

05_01_a_company_stock_prediction 에
Stacking을 적용해주세요

3. 양방향 순환 층

   순환 네트워크에 같은 정보를 다른 방향으로 주입하여 정확도를 높이고 기억을 좀더 오래 보존하는 것.

4. RNN 실습

RNN Regression(prediction)

```
## 모델
input_Layer = tf.keras.layers.Input(shape=(seq_length,x_data_dim))
x= tf.keras.layers.SimpleRNN(20,dropout = 0.1, recurrent_dropout = 0.5, return_sequences=True)(input_La
x= tf.keras.layers.SimpleRNN(10, dropout = 0.1, recurrent_dropout = 0.5)(x)
x= tf.keras.layers.Dense(30)(x)
Out_Layer=tf.keras.layers.Dense(1)(x)

model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()


loss_function=tf.keras.losses.mean_squared_error
optimize=tf.keras.optimizers.Adam(lr=0.001)
metric=tf.keras.metrics.mean_absolute_error
model.compile(loss=loss_function,
              optimizer=optimize,
              metrics=[metric])

history = model.fit_generator(
      data_gen_train,
      validation_data=data_gen_val,
      steps_per_epoch=50,
      epochs=500,
      validation_steps=1
)
print(model.evaluate_generator(data_gen_test))
prediction_Y=model.predict_generator(data_gen_test).flatten()
```

모델 설계

모델 loss, optimizer, metric 선언 compile

모델 학습

모델 학습 완료 후 평가 및 테스트

Practice : P_05_03_rnn_stock_prediction_stacking_drop_out.ipynb

4. RNN 실습

RNN Regression(prediction)

1. 순환 dropout

   순환 층에서 overfitting을 방지하기 위해 케라스에 내장되어 있는 dropout

2. Stacking 순환 층

   네트워크이 표현 능력을 증가.단, 계산 비용이 커짐

3. 양방향 순환 층

   순환 네트워크에 같은 정보를 다른 방향으로 주입하여 정확도를 높이고 기억을 좀더 오래 보존하는 것.

05_01_a_company_stock_prediction 에
Stacking + drop out 을 적용해주세요

Practice : P_05_03_rnn_stock_prediction_stacking_drop_out.ipynb

4. RNN 실습

RNN Regression(prediction)

1. **Stacking 순환 층**

   네트워크이 표현 능력을 증가.단, 계산 비용이 커짐

2. **순환 dropout**

   순환 층에서 overfitting을 방지하기 위해 케라스에 내장되어 있는 dropout

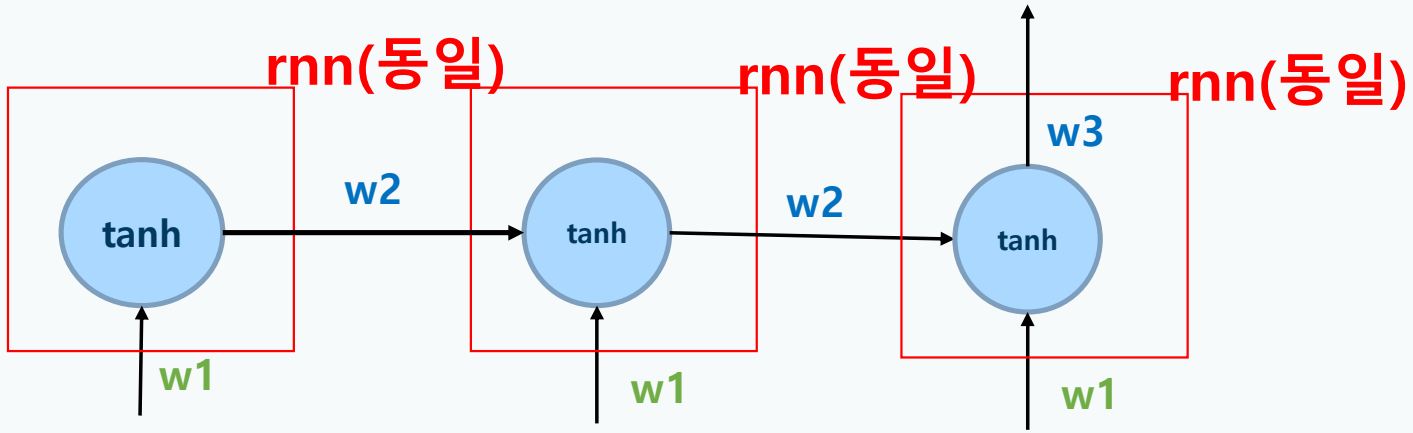
3. **양방향 순환 층**

   순환 네트워크에 같은 정보를 다른 방향으로 주입하여 정확도를 높이고 기억을 좀더 오래 보존하는 것.

Practice : P_05_03_stock_prediction_stacking_drop_out.ipynb

4. RNN 실습

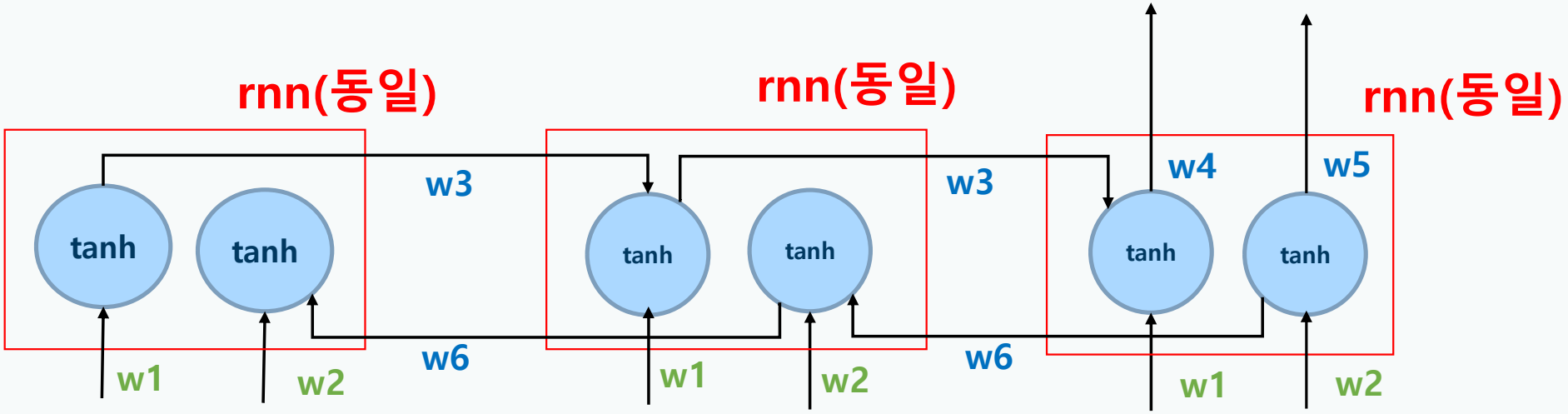
RNN Regression(prediction) - 양방향



Practice : P_05_04_rnn_stock_prediction_Bidirectional.py

4. RNN 실습

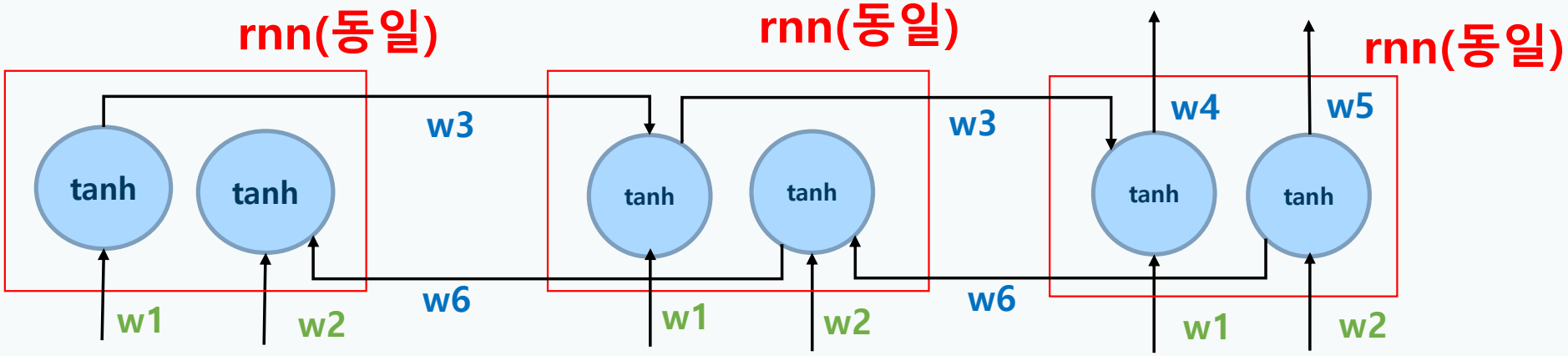
RNN Regression(prediction) - 양방향



Practice : P_05_04_rnn_stock_prediction_Bidirectional.py

4. RNN 실습

RNN Regression(prediction) - 양방향



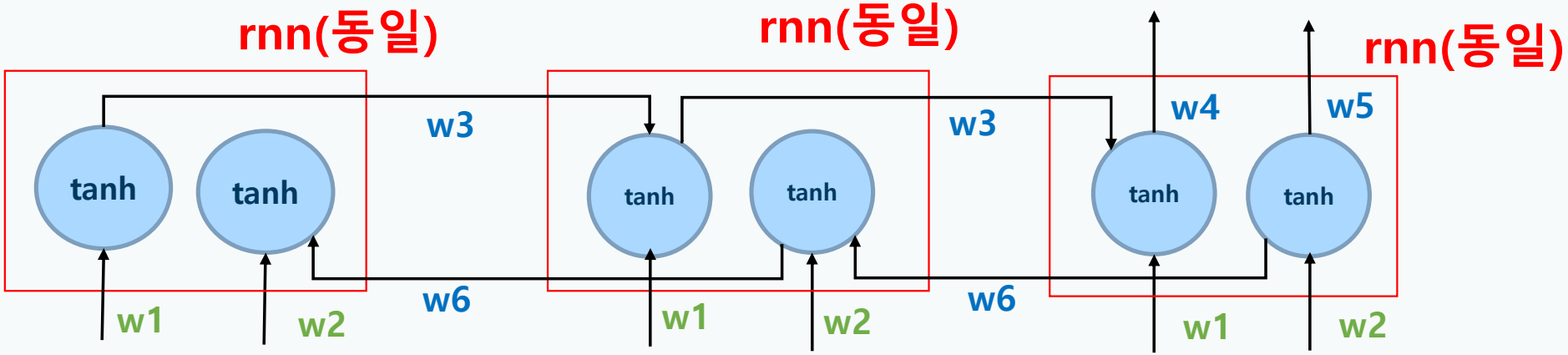

```
## 모델
input_Layer = tf.keras.layers.Input(shape=(1,1))
Out_Layer=tf.keras.layers.Bidirectional(tf.keras.layers.SimpleRNN(1))(input_Layer
model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1, 1)]            0
_________________________________________________________________
bidirectional (Bidirectional (None, 2)                 6
=================================================================
Total params: 6
```


Practice : P_05_04_rnn_stock_prediction_Bidirectional.py

4. RNN 실습

RNN Regression(prediction) - 양방향



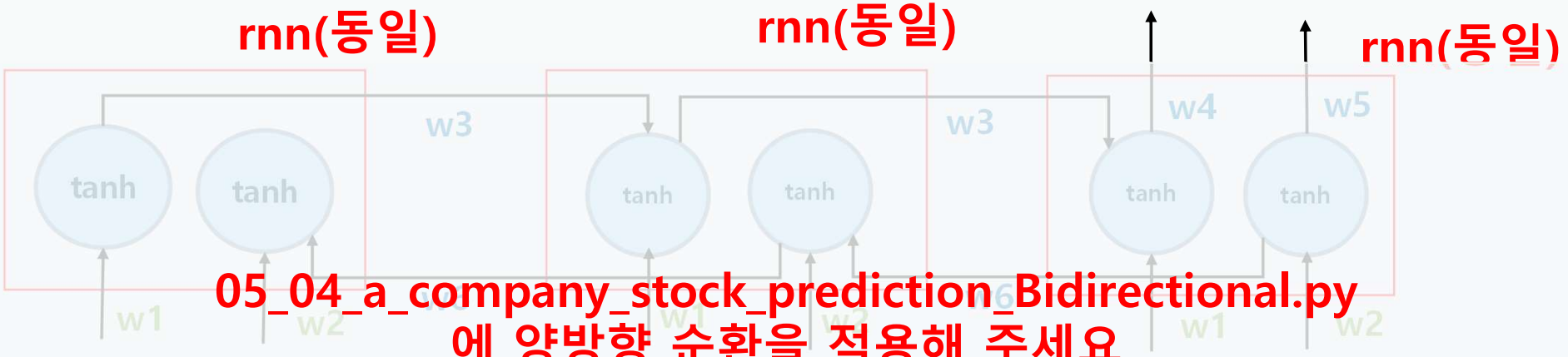

```
## 모델
input_Layer = tf.keras.layers.Input(shape=(1,1))
Out_Layer=tf.keras.layers.Bidirectional(tf.keras.layers.SimpleRNN(1))(input_Layer
model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1, 1)]            0
_________________________________________________________________
bidirectional (Bidirectional (None, 2)                 6
=================================================================
Total params: 6
```


Practice : P_05_04_rnn_stock_prediction_Bidirectional.py

4. RNN 실습

RNN Regression(prediction) - 양방향



05_04_a_company_stock_prediction_Bidirectional.py
에 양방향 순환을 적용해 주세요

```
## 모델
input_Layer = tf.keras.layers.Input(shape=(1,1))
Out_Layer=tf.keras.layers.Bidirectional(tf.keras.layers.SimpleRNN(1))(input_Layer)
model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1, 1)]            0
_________________________________________________________________
bidirectional (Bidirectional (None, 2)                 6
=================================================================
Total params: 6
```


Practice : P_05_04_rnn_stock_prediction_Bidirectional.py

오늘 하루 고생 하셨습니다.

Q & A

참 고 자 료

- cs231n 강의
- 블록과 함께하는 파이썬 딥러닝 케라스 이야기
- 밑바닥부터 시작하는 딥러닝 시즌1,2
- 기타 구글 딥러닝 이미지 및 블로그 자료