KAIT 한국정보통신진흥협회
Korea Association for ICT Promotion

KAIT
ICT 기업의 성장 플랫폼

BM 과정

2019. 07

에스투비즈
대표컨설턴트
이 이 백

KAIT 한국정보통신진흥협회
Korea Association for ICT Promotion

목 차

I. Big Data Architecture와 AI 시스템의 Position

II. AI 구축 방법론

III. Industry 4.0

IV. AI 시스템 구축을 위한 기본 지식과 기술

V. Dee Learning Revisit

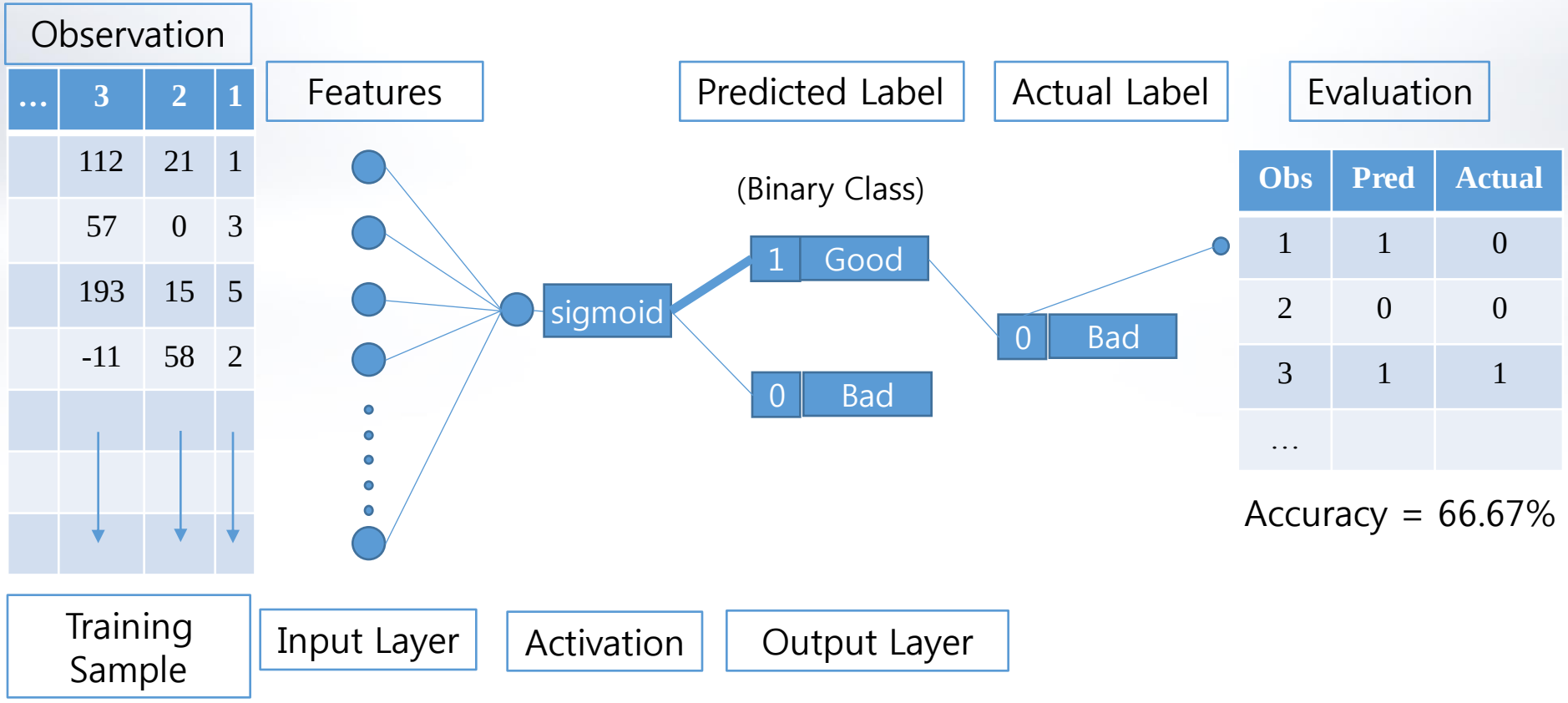
VI. Business Model과 AI Topic 발굴

V Deep Learning Revisit

1. Perceptron
2. MLP
3. CNN
4. RNN/LSTM
5. Autoencoder

1. Perceptron

- Perceptron > Classification
  - **Case : Logistic Regression for Binary Class Label**



Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

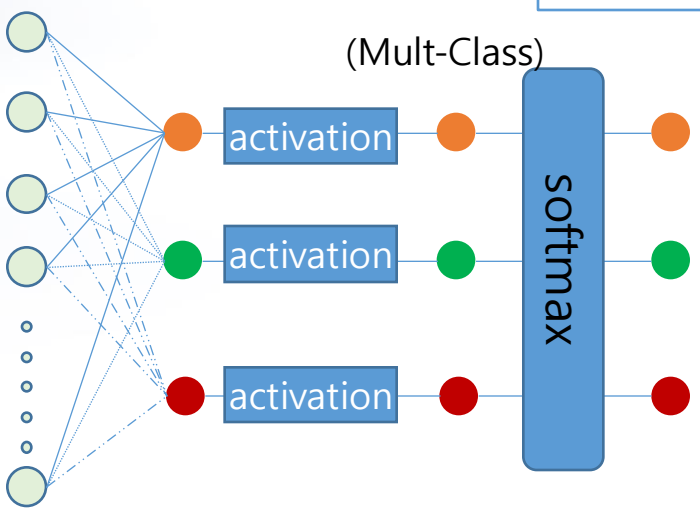
1. Perceptron

- Perceptron > Classification
  - **Case : Softmax Regression for Multi-Class Label**

Observation

…	3	2	1
	112	21	1
	57	0	3
	193	15	5
	-11	58	2

Features



Predicted Label

Evaluation

Obs	Softmax	Pred	Actual
1	[0.1,0.1,0.8]	[0,0,1]	[0,0,1]
2	[0.6,0.2,0.2]	[1,0,0]	[0,0,1]
3	[0.01,0.97,0.02]	[0,1,0]	[0,1,0]
…			

Accuracy = 66.67%

Cross-entropy = ???

Training Sample

Input Layer

Activaion

Output Layer

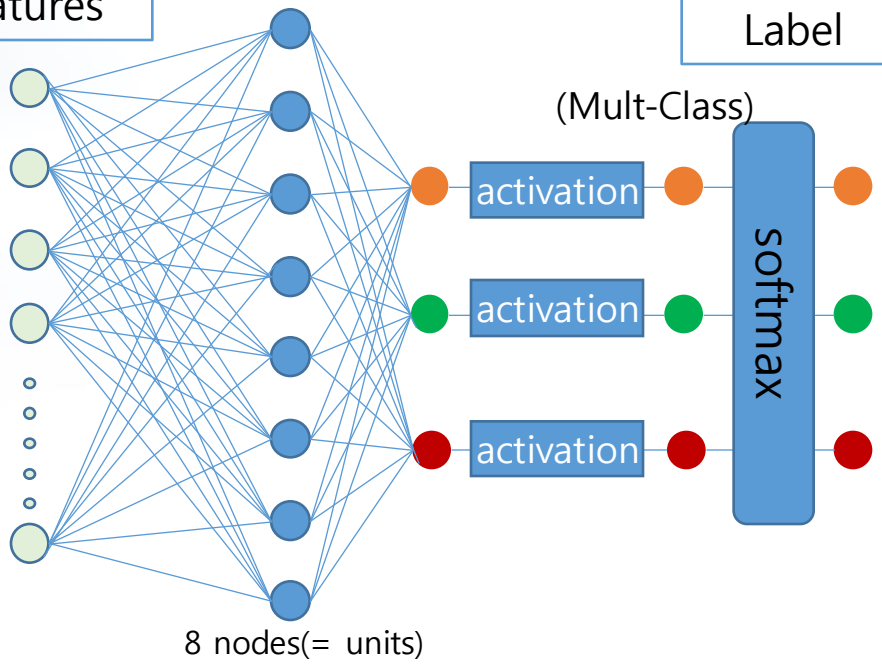
Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

1. Perceptron

- Single Layer Perceptron > Classification
  - **Case : Single Hidden Layer for Softmax Classification**

Observation

…	3	2	1
	112	21	1
	57	0	3
	193	15	5
	-11	58	2
	↓	↓	↓



Evaluation

Obs	Softmax	Pred	Actual
1	[0.1,0.1,0.8]	[0,0,1]	[0,0,1]
2	[0.6,0.2,0.2]	[1,0,0]	[0,0,1]
3	[0.01,0.97,0.02]	[0,1,0]	[0,1,0]
…			

Accuracy = 66.67%

Cross-entropy = ???

Training Sample

Input Layer

Single Hidden Layer

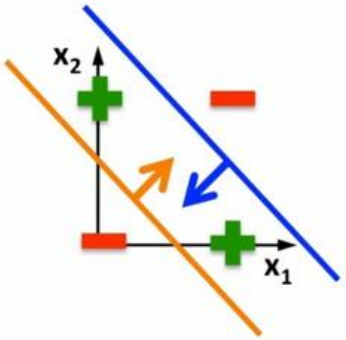
Output Layer

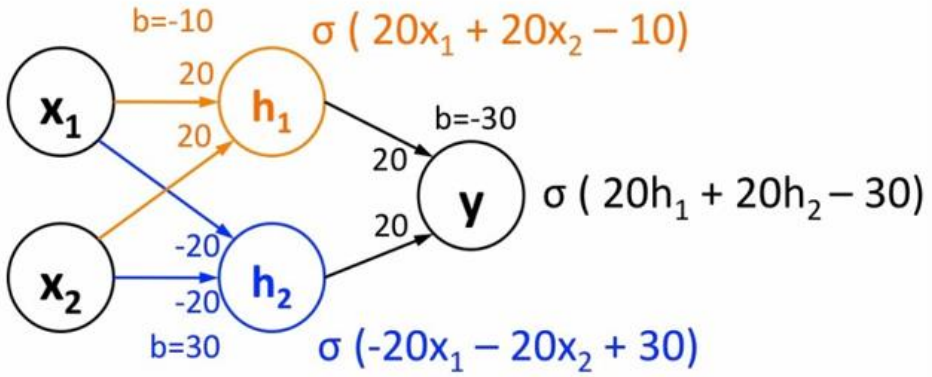
Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

1. Perceptron

- Single Layer Perceptron > Classification
  - **Linear Classification의 한계**

Linear classifiers cannot solve this





σ(20*0 + 20*0 – 10) ≈ 0
σ(20*1 + 20*1 – 10) ≈ 1
σ(20*0 + 20*1 – 10) ≈ 1
σ(20*1 + 20*0 – 10) ≈ 1

σ (-20*0 – 20*0 + 30) ≈ 1
σ (-20*1 – 20*1 + 30) ≈ 0
σ (-20*0 – 20*1 + 30) ≈ 1
σ (-20*1 – 20*0 + 30) ≈ 1

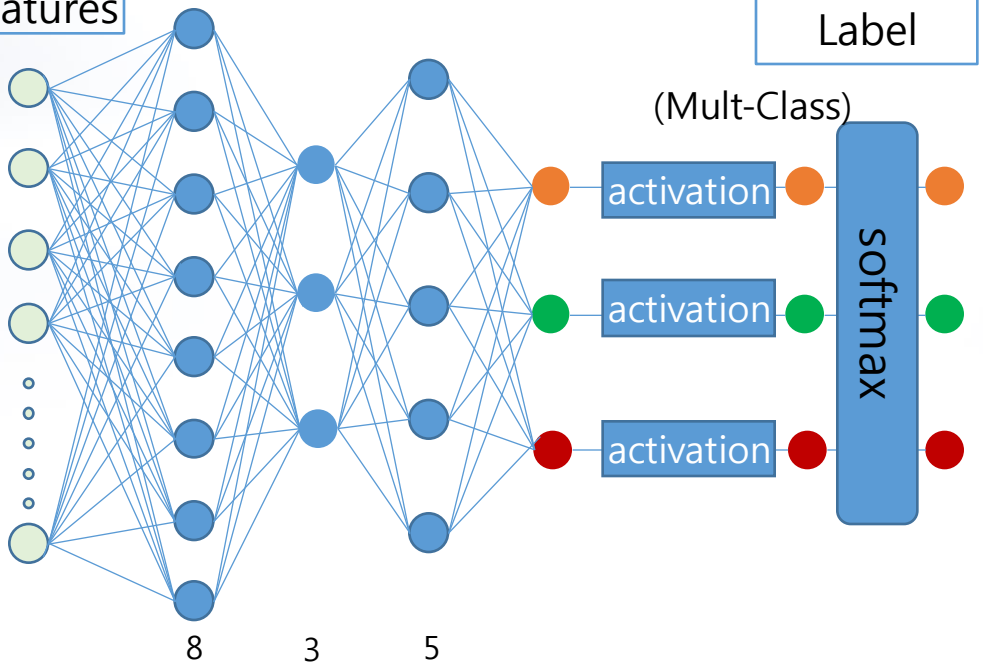
σ (20*0 + 20*1 – 30) ≈ 0
σ (20*1 + 20*0 – 30) ≈ 0
σ (20*1 + 20*1 – 30) ≈ 1
σ (20*1 + 20*1 – 30) ≈ 1

1. Perceptron

- Multi Layer Perceptron > Classification
  - **Case : Multi Hidden Layer for Softmax Classification**

Observation

…	3	2	1
	112	21	1
	57	0	3
	193	15	5
	-11	58	2



Evaluation

Obs	Softmax	Pred	Actual
1	[0.1,0.1,0.8]	[0,0,1]	[0,0,1]
2	[0.6,0.2,0.2]	[1,0,0]	[0,0,1]
3	[0.01,0.97,0.02]	[0,1,0]	[0,1,0]
…			

Accuracy = 66.67%

Cross-entropy = ???

Training Sample

Input Layer

Multi Hidden Layer

Output Layer

Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

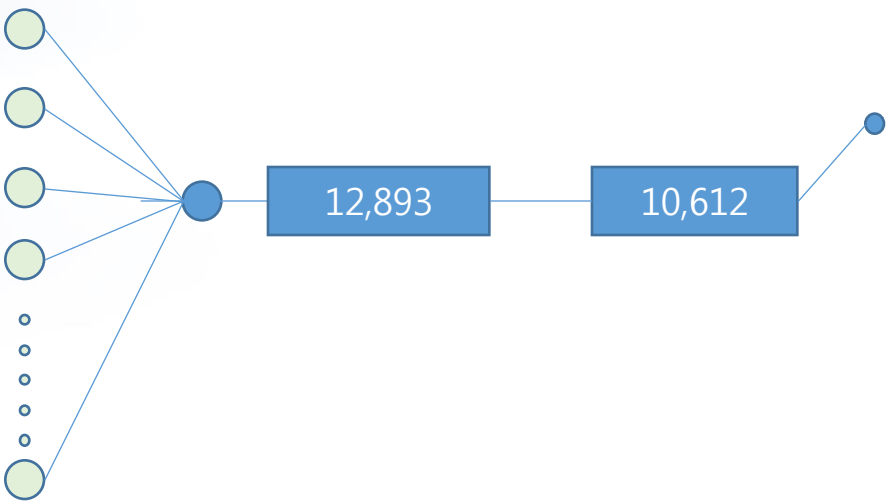
1. Perceptron

- Perceptron > Regression
  - **Case : Regression for Continuous Valued Label**

Observation

…	3	2	1
	112	21	1
	57	0	3
	193	15	5
	-11	58	2

Training Sample



Evaluation

Obs	Pred	Actual	Diff
1	12,893	10,612	2,281
2	10,002	11,322	-1,320
3	18,022	17,001	1,019
…			

MSE = ?
MAE = ?
RMSE = ?
MAPE = ?

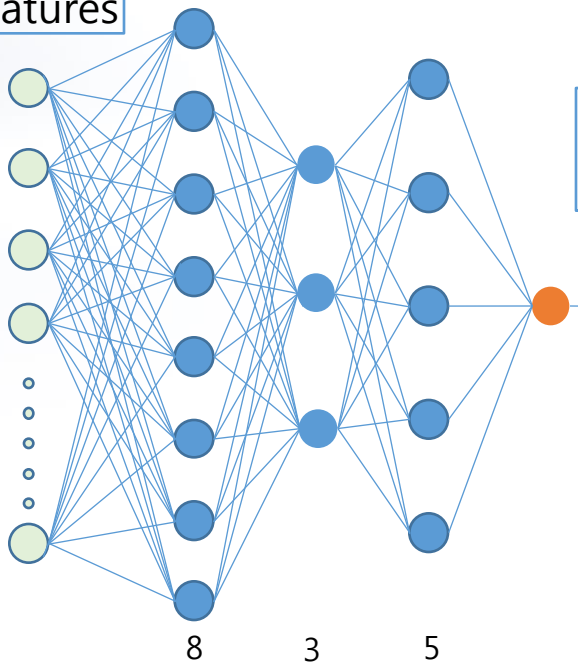
Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

1. Perceptron

- Multi Layer Perceptron > Regression
  - **Case : Multi-Layer Perceptron for Continuous Valued Label**

Observation

…	3	2	1
	112	21	1
	57	0	3
	193	15	5
	-11	58	2



Predicted Label: 12,893

Actual Label: 10,612

Evaluation

Obs	Pred	Actual	Diff
1	12,257	12,800	-743
2	11,328	11,300	28
3	17,230	17,100	130
…			

MSE = ?
MAE = ?
RMSE = ?
MAPE = ?

Training Sample

Input Layer

Multi Hidden Layer

Output Layer

Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

2. MLP

- Parameter Tuning
  - 목적 : 일반화를 보증하는 오차 최소화

일반화

- 훈련 Sample에 의존한 예측의 문제점 → Over-Fitting
- Small Sample & Large Number of Features → Under-Fitting
- 일반화를 위한 Data 준비
  - 2 or 3 Set : Training Data, (Validation Data,) Test Data
  - Training Data → Model 수식 산출
  - Validation Data → Model Evaluation
  - Test Data → Real World Application 적용 가능성 최종 결정 및 적용할 Model 확정

오차 최소화의 영향 요소

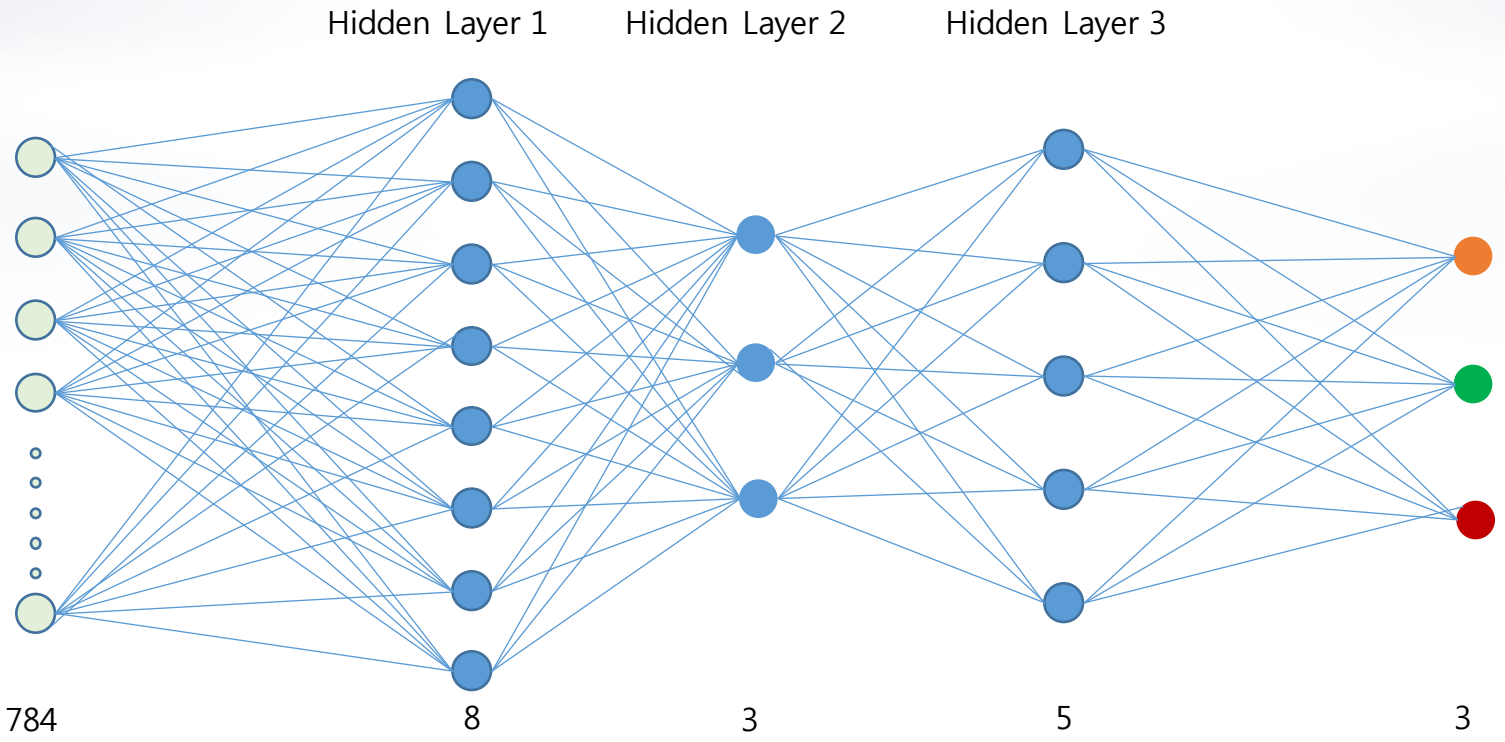
- Input Layer
  - 가중치(weight)의 변화
  - 입력변수 → 파생변수 → 특징 추출
- Hidden Layer
  - Multi-Input Variable간의 연산 → 파생변수화
  - 새로운 가충치의 연산
  - 변수들간의 관계성 또는 독립성에 대한 특징 학습
  - Activation Function 선택
- Output Layer
  - 모델의 예측값 산출
  - 현실적 기대치에 못미칠 경우 재학습 수행 – Backpropagation
  - Model Evaluation

Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

2. MLP

- Parameters : Hidden Layers & Neurons

Parameter	내용	Case
Number of Hidden Layer	▪ 히든레이어의 수	[L1, L2, L3]
Number of Neurons per Hidden Layer	▪ 각 히든레애어의 뉴런수	[8, 3, 5]



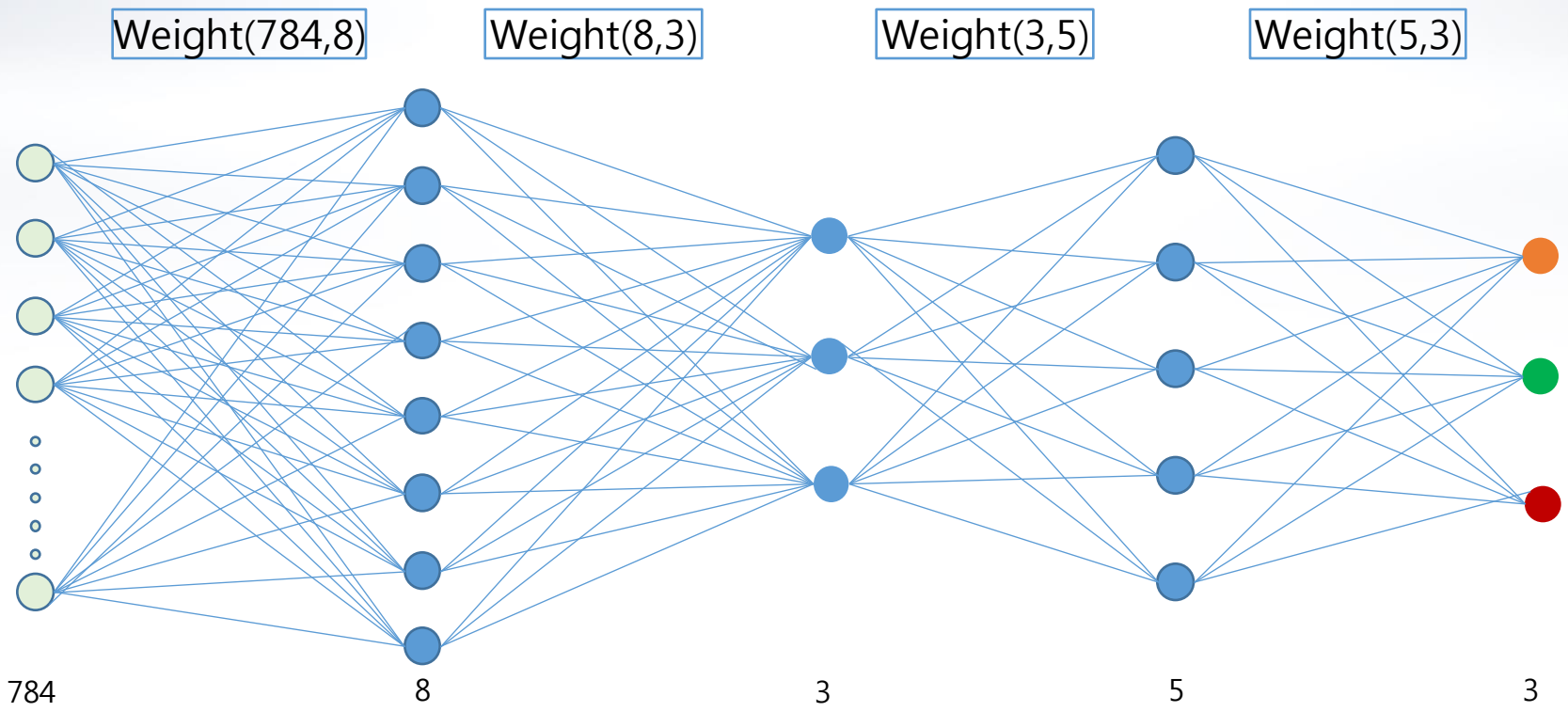
2. MLP

- Parameters : Weight

Parameter	내용
Weight Initialization	▪ 가중치 초기값 산출

가중치 초기화 방법

- 정규분포의 random value
- 정규분포 +/- 2시그마내 random value
- 상수값 : 예) 0, 1



Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

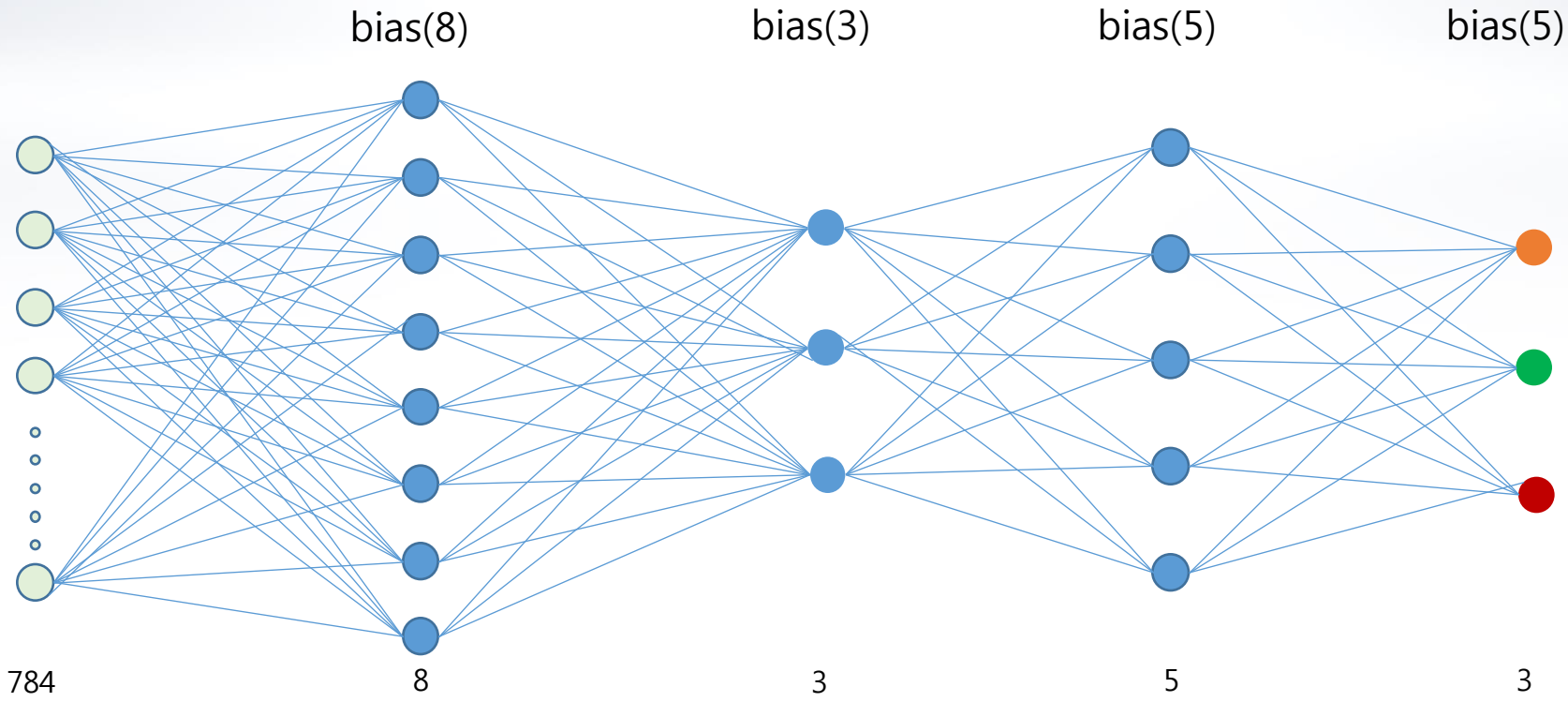
2. MLP

- Parameters : bias

Parameter	내용
bias Initialization	▪ 가중치 초기값 산출

bias 초기화 방법

- 정규분포의 random value
- 정규분포 +/- 2시그마내 random value
- 상수값 : 예) 0, 1



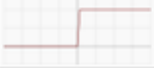
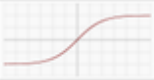
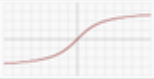
Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

2. MLP

- Parameters : Activations for Forward Propagation

Parameter	내용
Activation function	▪ 활성화함수 선택

activation method

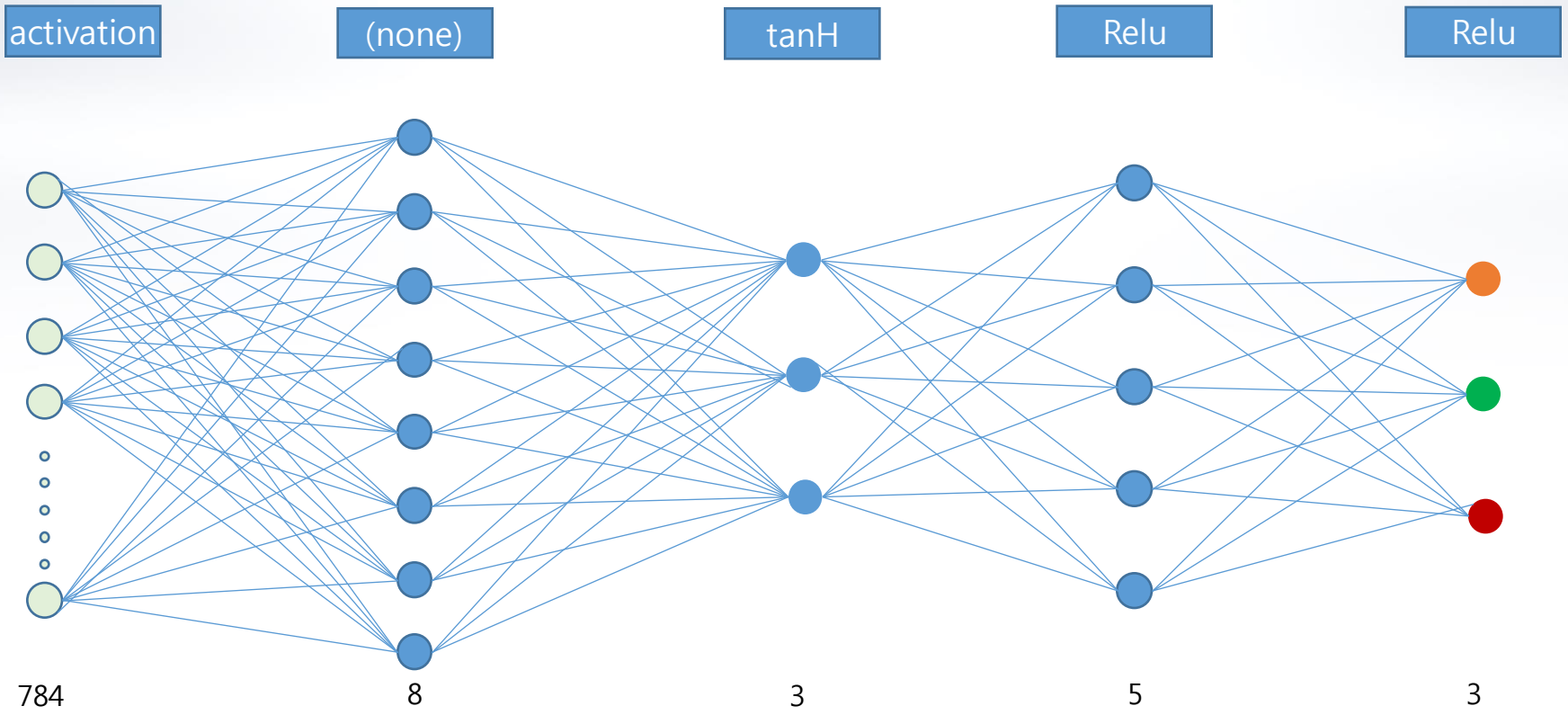
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

2. MLP

- Parameter for Forward Propagation

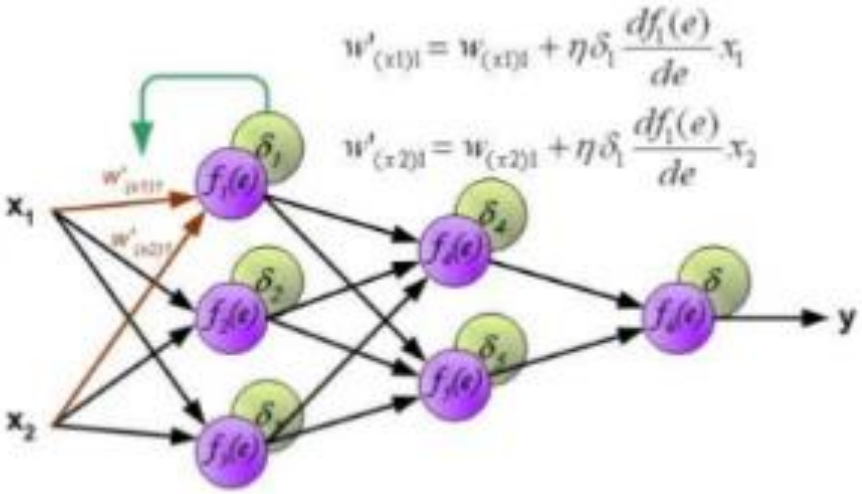
Parameter	내용
Activation function	▪ 활성화함수 선택

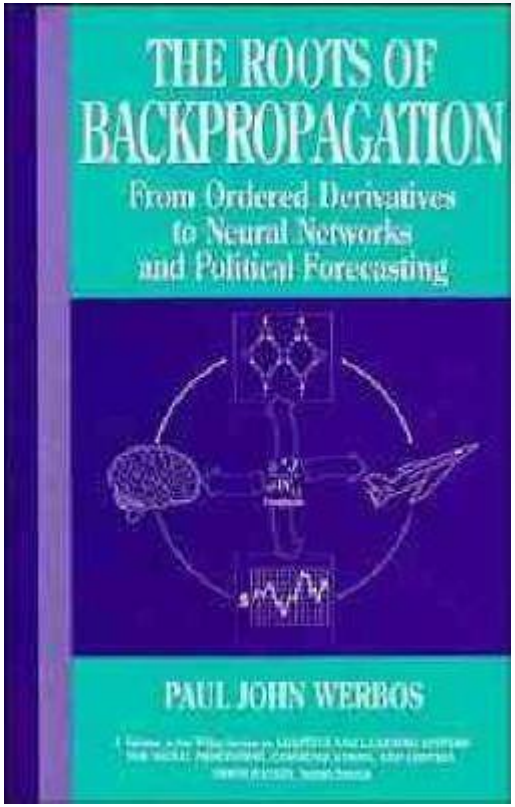


Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

2. MLP

- Parameters : Backward Propagation Method
  - Paul Werbos(1974)의| Ph.D. thesis
  - Book : The Roots of Backpropagation





2. MLP

- Parameters : Backward Propagation Method
  - Hinton(1986)
    - Paper : Learning Distributed Representations of Concepts

Geoffrey Hinton

Appointment
Advisor
Learning in Machines & Brains

Institution
University of Toronto
Google
Department of Computer Science

Country
Canada

3

Learning distributed representations of concepts

GEOFFREY E. HINTON

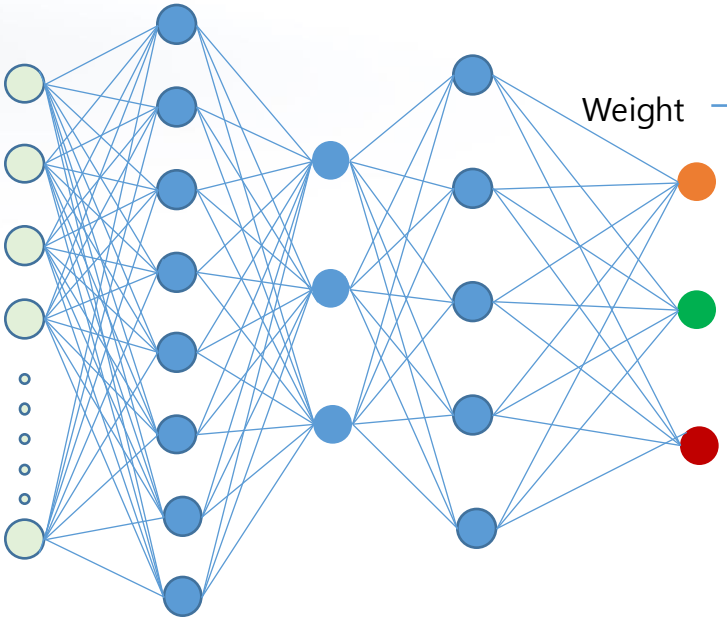
KAIT 한국정보통신진흥협회 Korea Association for ICT Promotion

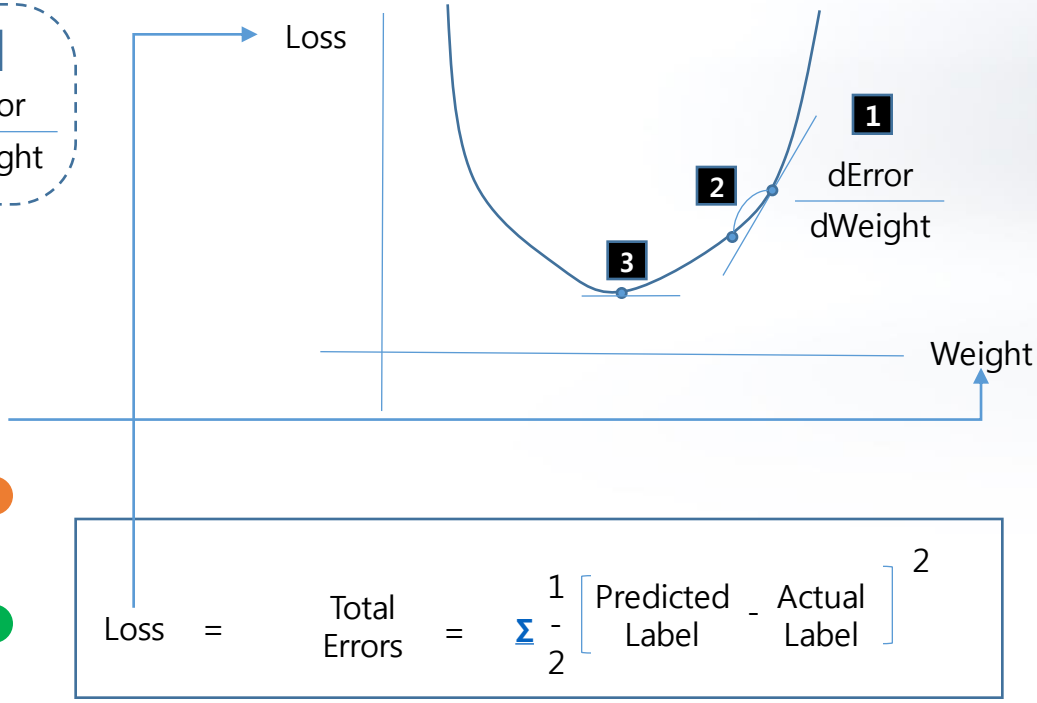
2. MLP

- Parameters : Backward Propagation Method
  - 전방위 학습과 반대 방향으로 거슬러 올라가기
  - Weight Update - learning rate를 조정 계수로 하여 오차가 최소화할 때 가지 가중치 갱신

$$\text{New Weight} = \text{Weight} - \text{LearningRate} * \frac{\text{dError}}{\text{dWeight}}$$

1 2 3 에 도달할 때까지 iteration





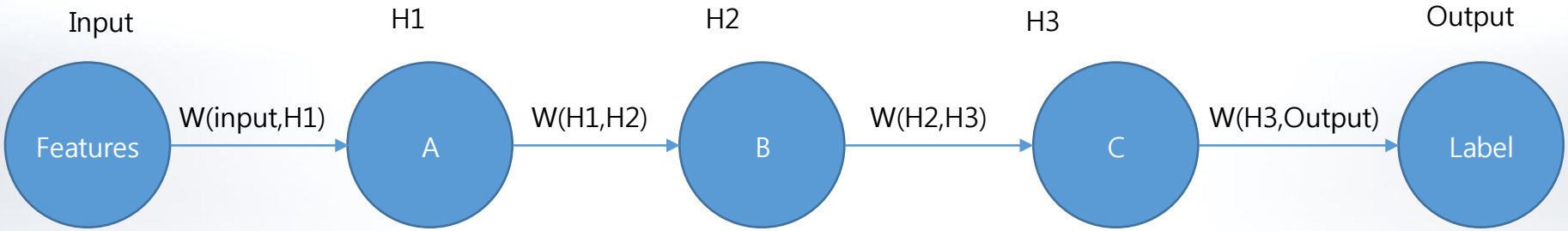
$$\text{Loss} = \text{Total Errors} = \sum \frac{1}{2}\left[\text{Predicted Label} - \text{Actual Label}\right]^2$$

$$\frac{\text{dErrors}}{\text{dPredicted\_Labels}} = -2 * \frac{1}{2}\left[\text{Predicted Label} - \text{Actual Label}\right]$$

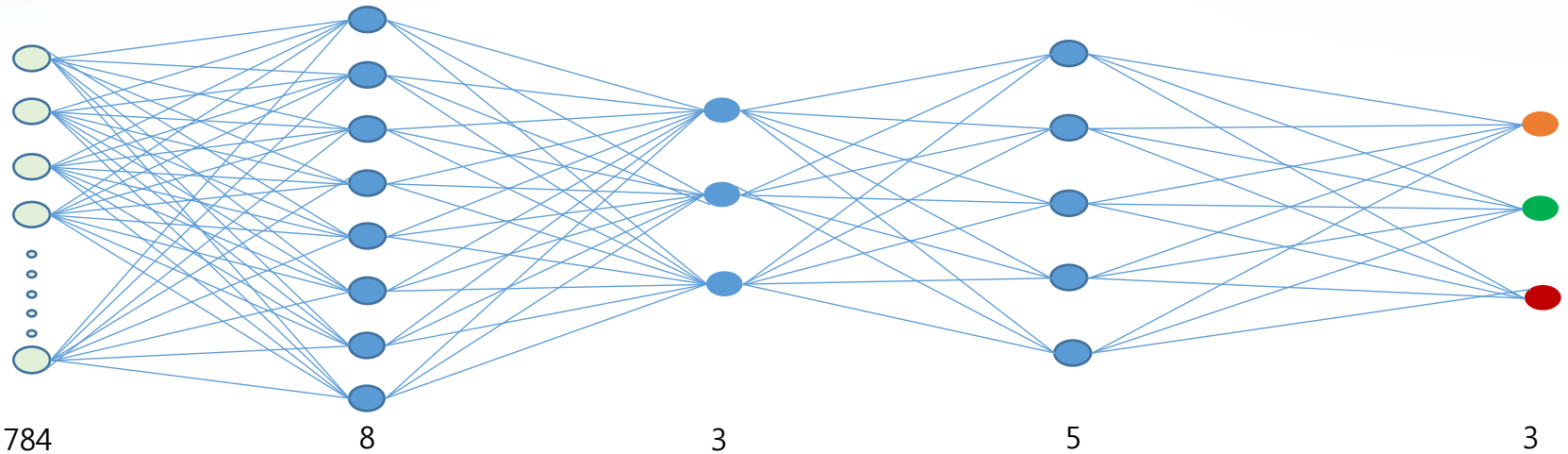
Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

2. MLP

- Parameters : Backward Propagation Method
  - 전방위 학습과 반대 방향으로 거슬러 올라가기



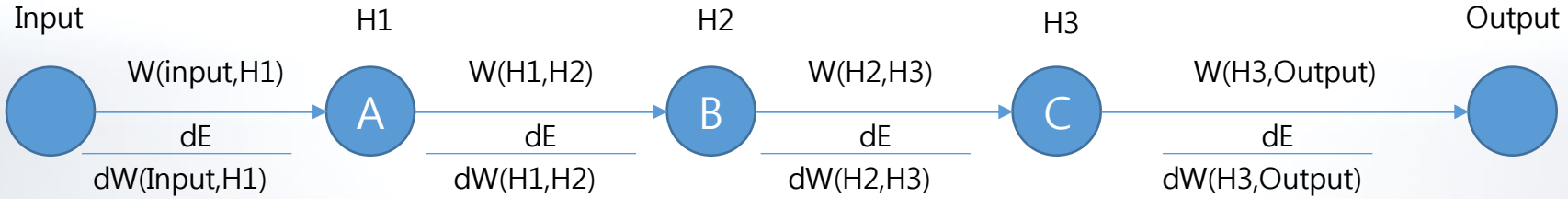
$$\frac{dPredLabel}{dFeatures} * \frac{dPredLabel}{dC\_} * \frac{dC}{dB} * \frac{dB}{dA} * \frac{dA}{dFeatures}$$



Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

2. MLP

- Parameters : Backward Propagation Method
  - 전방위 학습과 반대 방향으로 거슬러 올라가기

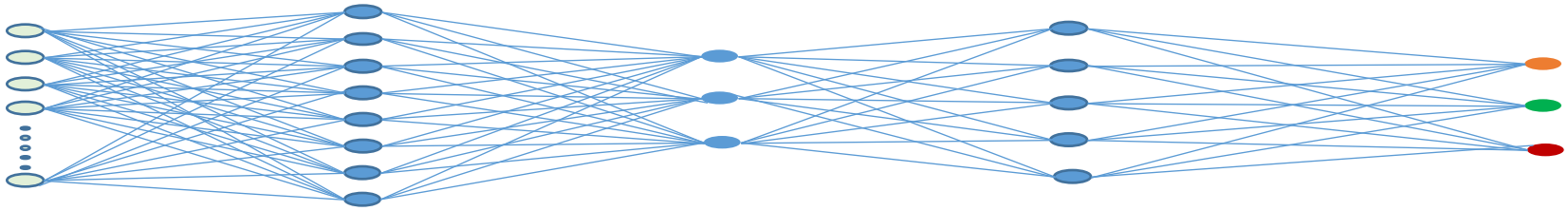


$$\frac{dE}{dPredLabel} * \frac{dPredLabel}{dW(H3,Output)}$$

$$\frac{dE}{dPredLabel} * \frac{dPredLabel}{dC} * \frac{dC}{dW(H2,H3)}$$

$$\frac{dE}{dPredLabel} * \frac{dPredLabel}{dC} * \frac{dC}{dB} * \frac{dB}{dW(H1,H2)}$$

$$\frac{dE}{dPredLabel} * \frac{dPredLabel}{dC} * \frac{dC}{dB} * \frac{dB}{dA} * \frac{dA}{dW(H1,H2)}$$



Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

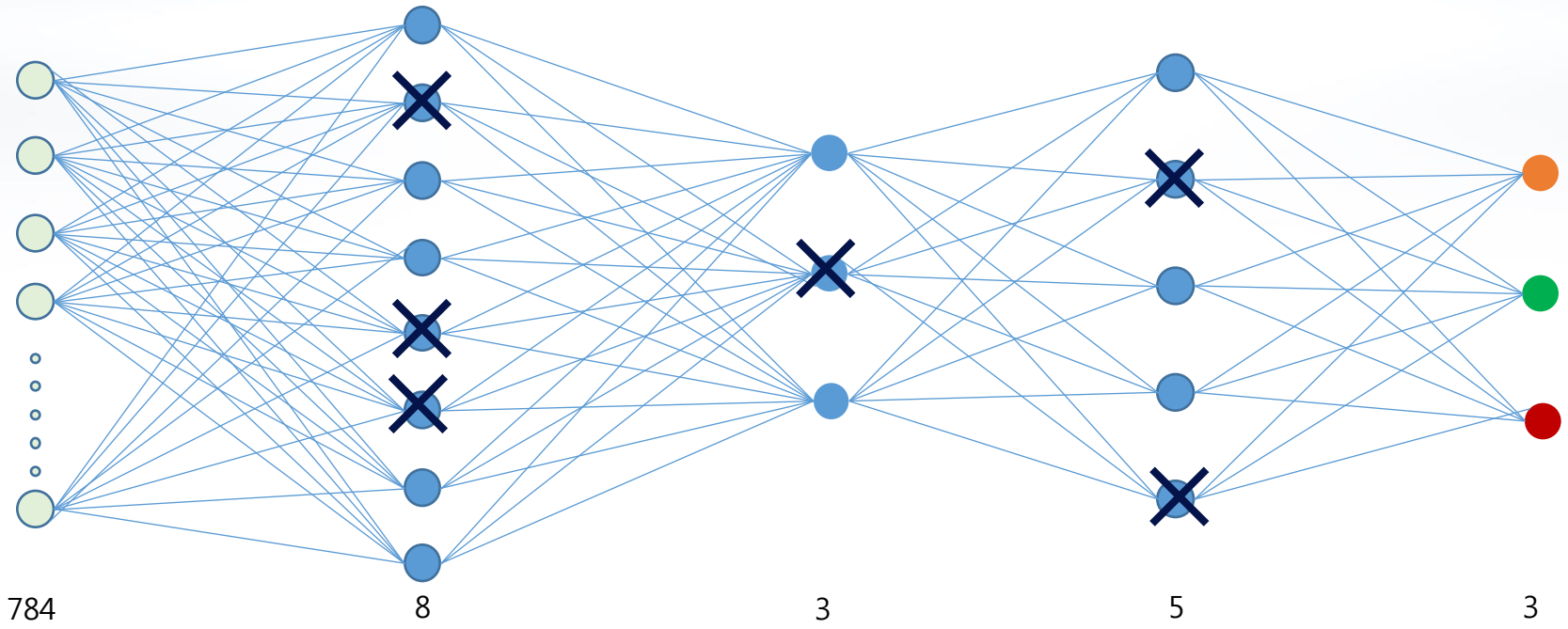
2. MLP

- Parameters : Regularization > Dropout
  - 학습을 덜 하겠다는 의미 → 과적합 방지

Parameter	내용
Dropout Rate	▪ 가중치 탈락 비율

Regularization Method

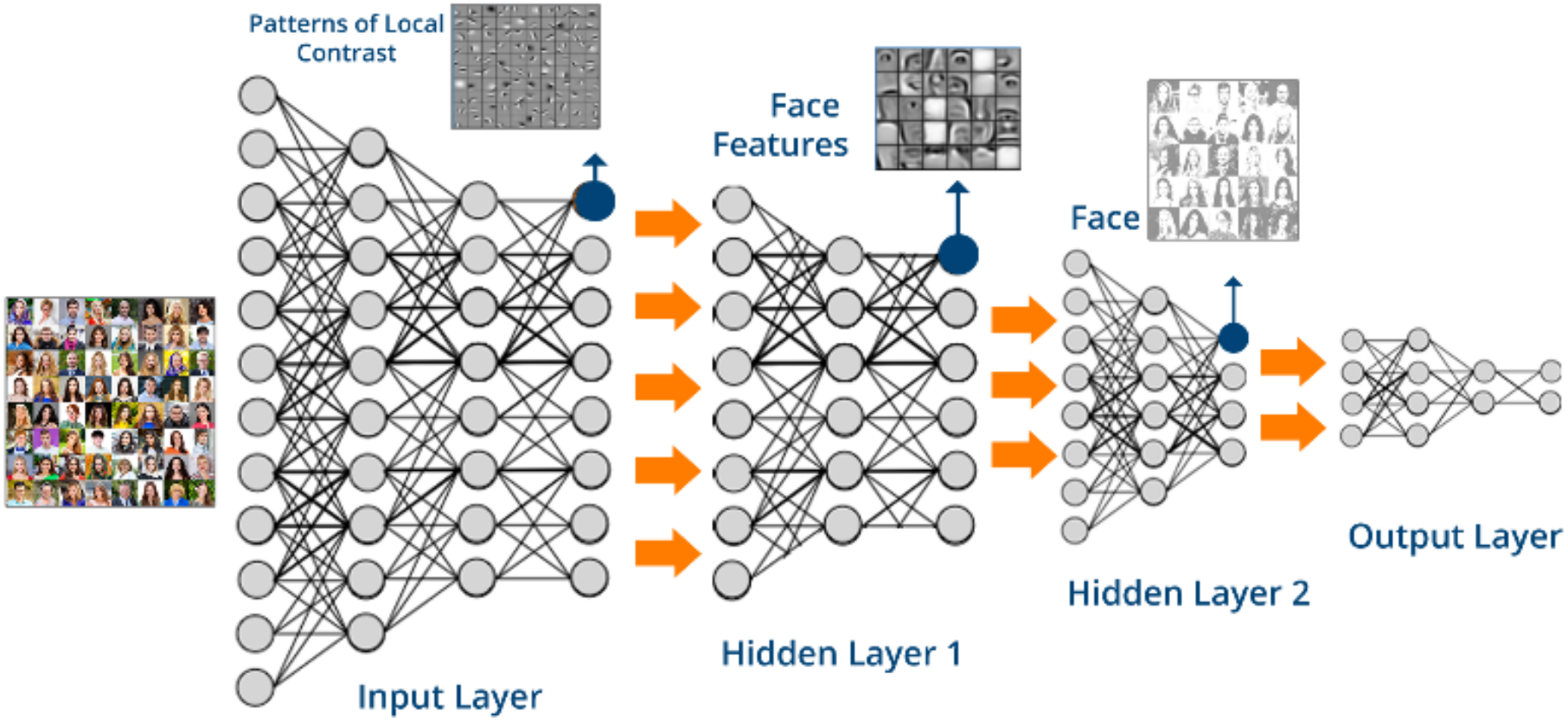
- L1 Regularization → Lasso
- L2 Regularization → Ridge
- Drop-Out → Deep Learning



Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

2. MLP

- Hidden Layer를 거치면서 Label에 적합하는 특징 패턴을 추출하는 과정



Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

2. MLP

- Activation

```python
#!/usr/bin/env /c/Apps/Anaconda3/python

# Import
import tensorflow as tf
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

# Import data
data = pd.read_csv('./data_stocks.csv')

# Drop date variable
data = data.drop(['DATE'], 1)

# Dimensions of dataset
n = data.shape[0]
p = data.shape[1]

# Make data a np.array
data = data.values

# Training and test data
train_start = 0
train_end = int(np.floor(0.8*n))
test_start = train_end + 1
test_end = n
data_train = data[np.arange(train_start, train_end), :]
data_test = data[np.arange(test_start, test_end), :]

# Scale data
scaler = MinMaxScaler(feature_range=(-1, 1))
scaler.fit(data_train)
data_train = scaler.transform(data_train)
data_test = scaler.transform(data_test)

# Build X and y
X_train = data_train[:, 1:]
y_train = data_train[:, 0]
X_test = data_test[:, 1:]
y_test = data_test[:, 0]

# Number of stocks in training data
n_stocks = X_train.shape[1]

# Neurons
n_neurons_1 = 1024
n_neurons_2 = 512
n_neurons_3 = 256
n_neurons_4 = 128

# Session
net = tf.InteractiveSession()

# Placeholder
X = tf.placeholder(dtype=tf.float32, shape=[None, n_stocks])
Y = tf.placeholder(dtype=tf.float32, shape=[None])


zer = tf.variance_scaling_initializer(mode="fan_avg", distribu
r = tf.zeros_initializer()

# Hidden weights
W_hidden_1 = tf.Variable(weight_initializer([n_stocks, n_neurons_1]))
bias_hidden_1 = tf.Variable(bias_initializer([n_neurons_1]))
W_hidden_2 = tf.Variable(weight_initializer([n_neurons_1, n_neurons_2]))
bias_hidden_2 = tf.Variable(bias_initializer([n_neurons_2]))
W_hidden_3 = tf.Variable(weight_initializer([n_neurons_2, n_neurons_3]))
bias_hidden_3 = tf.Variable(bias_initializer([n_neurons_3]))
W_hidden_4 = tf.Variable(weight_initializer([n_neurons_3, n_neurons_4]))
bias_hidden_4 = tf.Variable(bias_initializer([n_neurons_4]))

# Output weights
W_out = tf.Variable(weight_initializer([n_neurons_4, 1]))
bias_out = tf.Variable(bias_initializer([1]))

# Hidden layer
hidden_1 = tf.nn.relu(tf.add(tf.matmul(X, W_hidden_1), bias_hidden_1))
hidden_2 = tf.nn.relu(tf.add(tf.matmul(hidden_1, W_hidden_2), bias_hidden_2))
```

실습 중 배포

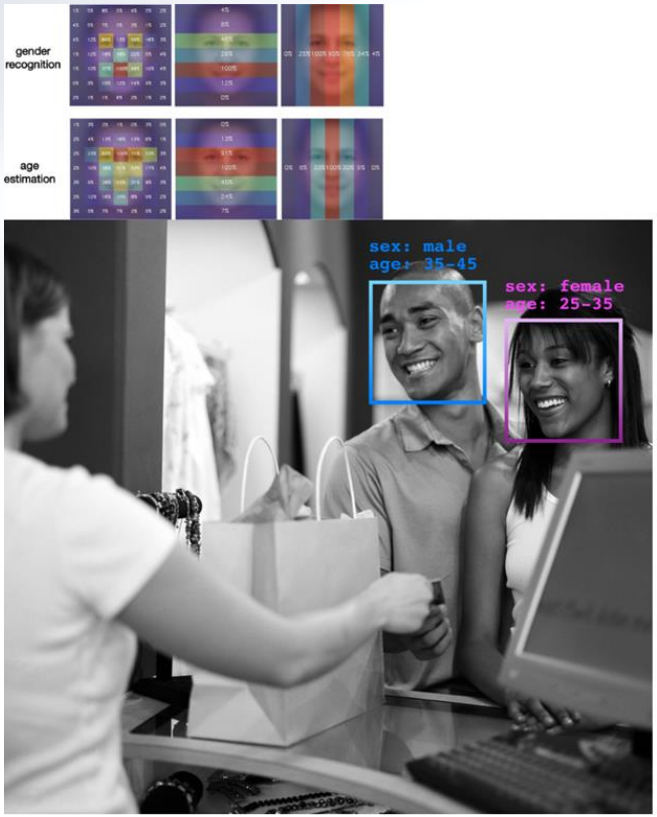
3. CNN

- Convolutional Neural Network(Yann LeCun, 1998)
- Convolution의 의미
  - 함수의 합성곱
- Input Layer → Convolution Layer
  - Reshape flattened input vector to 3d tensor
    - 3d tensor = [height, width, depth]
    - depth = [Red, Green, Blue] = 3 channel
      - ✓ [255, 0, 0] → Red = 1 Channel
- Convolution → ReLU → Max Pooling
  - Convolution
    - Stack of filtered images(여과지를 이동하면서 투영된 부분이미지의 적재)
      - ✓ 부분이미지 적재 → Feature Map을 만들어가는 과정
    - Kernel Filtering은 Forwarding과 동일한 효과
    - Stride → Padding → Activation(ReLU)
    - ReLU의 역할 : Normalizing
  - Pooling
    - Filtering 과정에서 적재한 특징 배열의 개별값들중 중 최대값만 취함
      - ✓ 최대값 만 선택적으로 취함 = down sampling
    - Stack을 축소(shrinking)시켜 큰 특징만 취하는 효과
      - ✓ Window, Stride, Max
- Fully Connection
  - Why? 신경망 학습 → MLP 와 동일

3. CNN

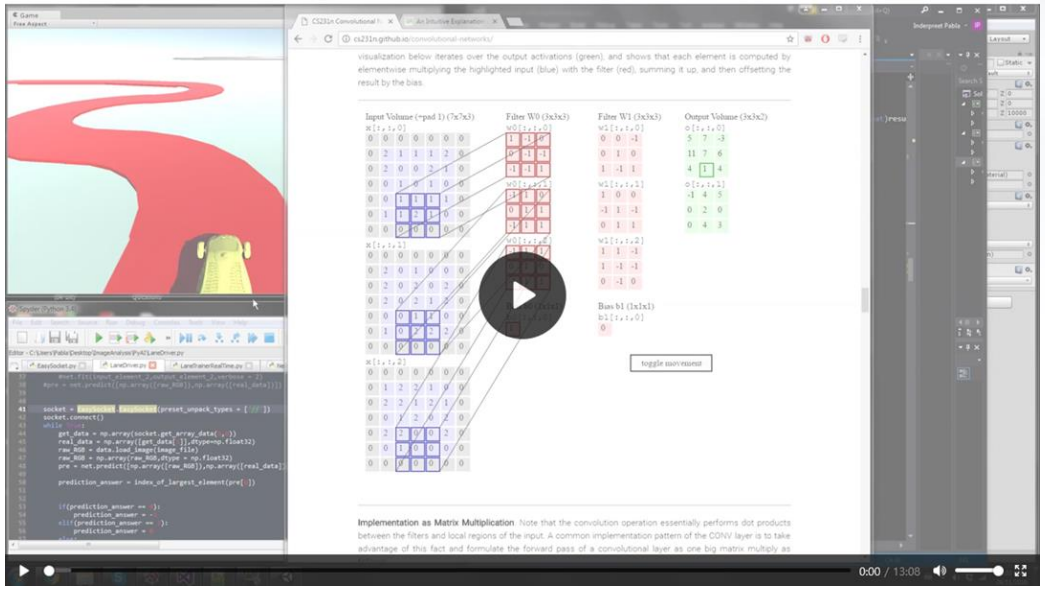
- CNN의 Output Image

성별 및 연령 예측



Auto-Driving Simulation

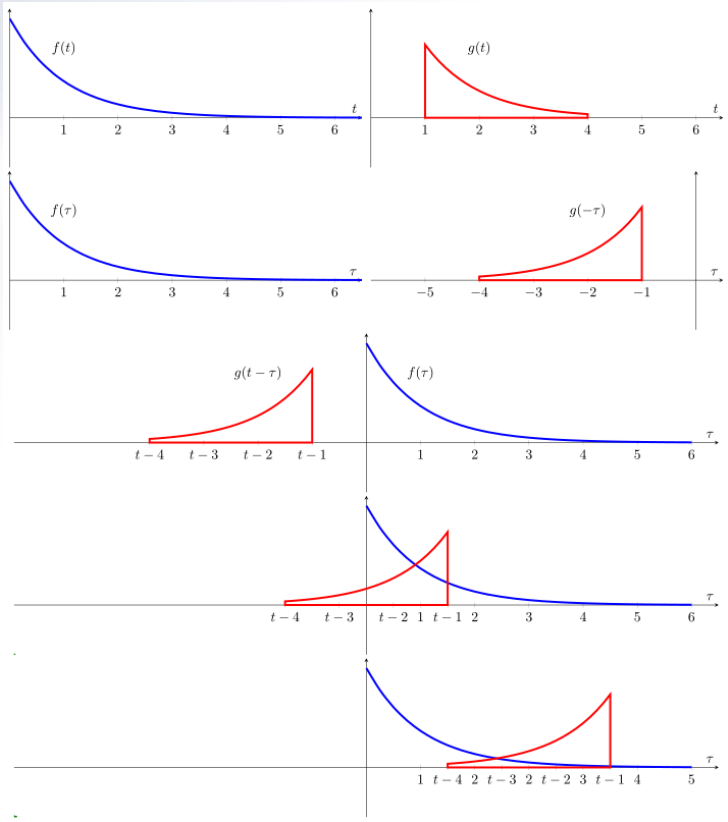
Self driving car : Convolutional Neural Network driving a car in an open source simulator

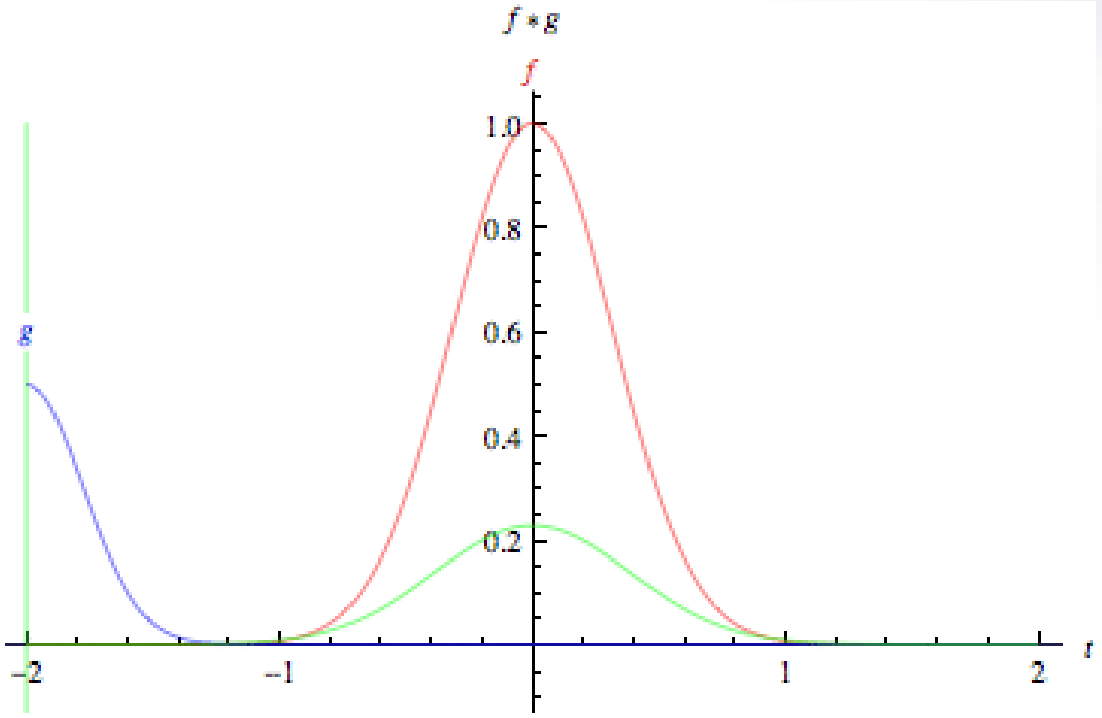


Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

3. CNN

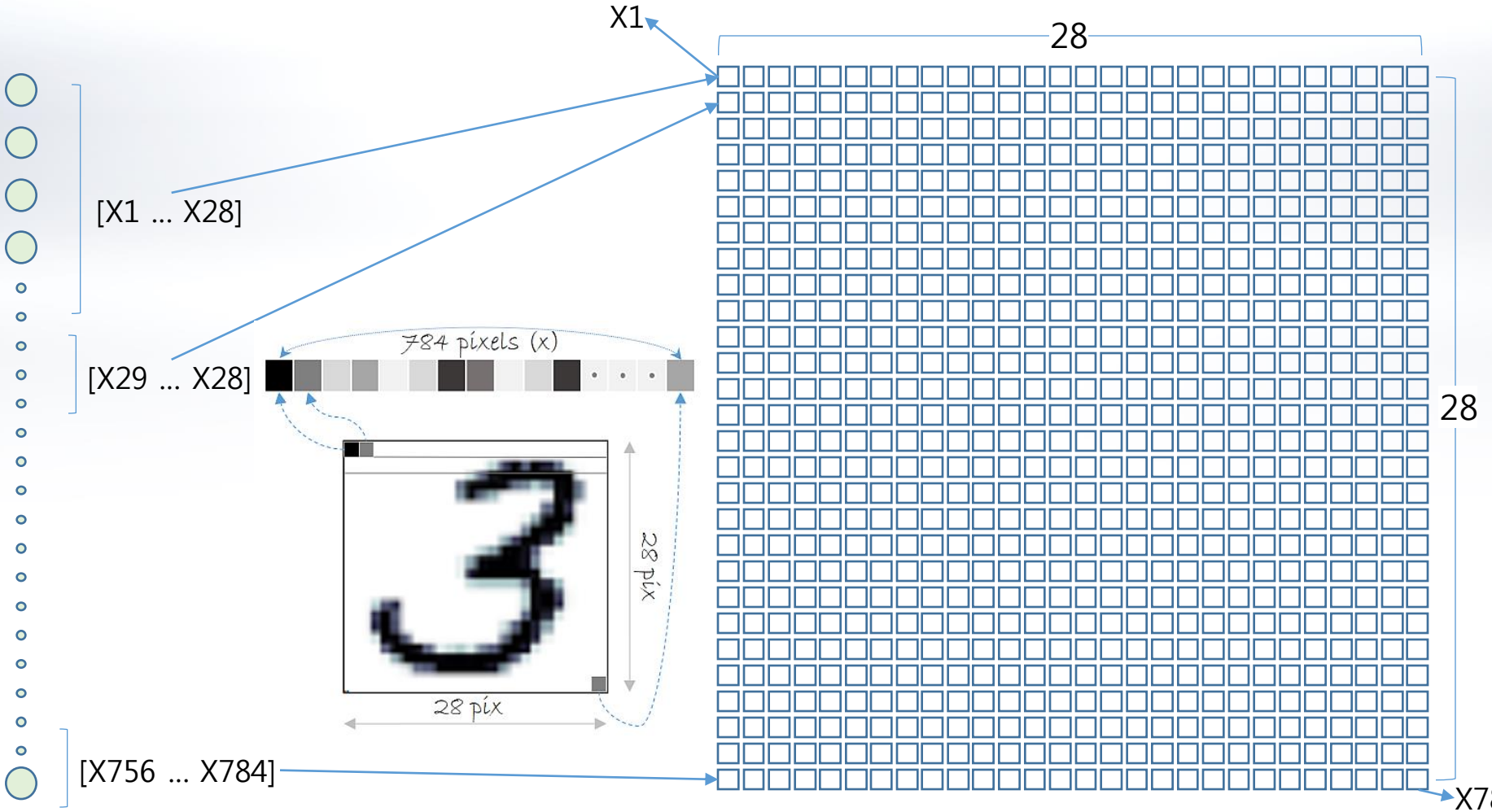
- Convolution
  - Roll Together(라틴어에서 파생)
  - Mathematics
    - Convolution은 2개의 함수를 겹쳐서 적분값을 측정(합성곱)





3. CNN

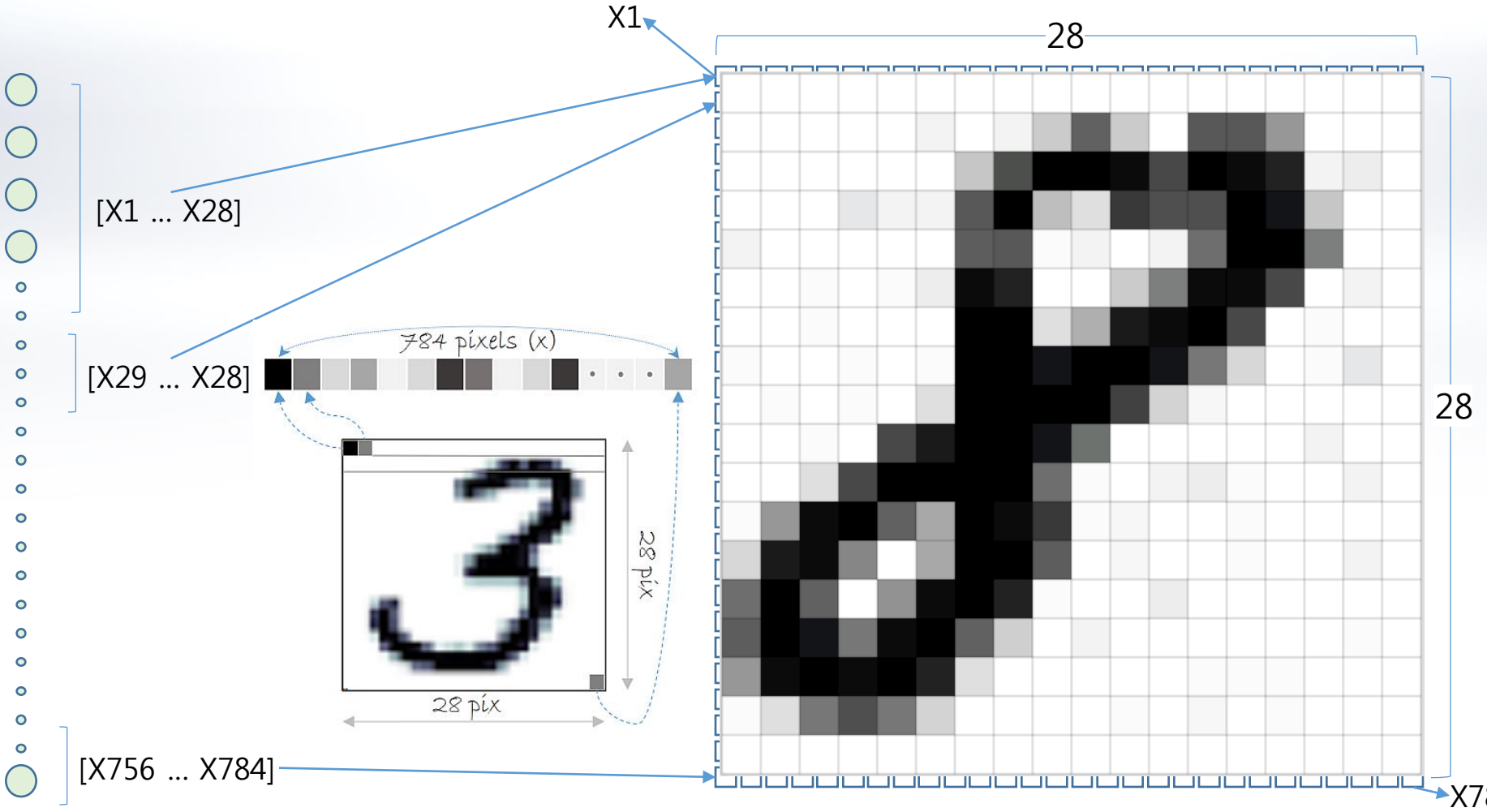
- Convolution
  - Reshape : Vector → Matrix



Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

3. CNN

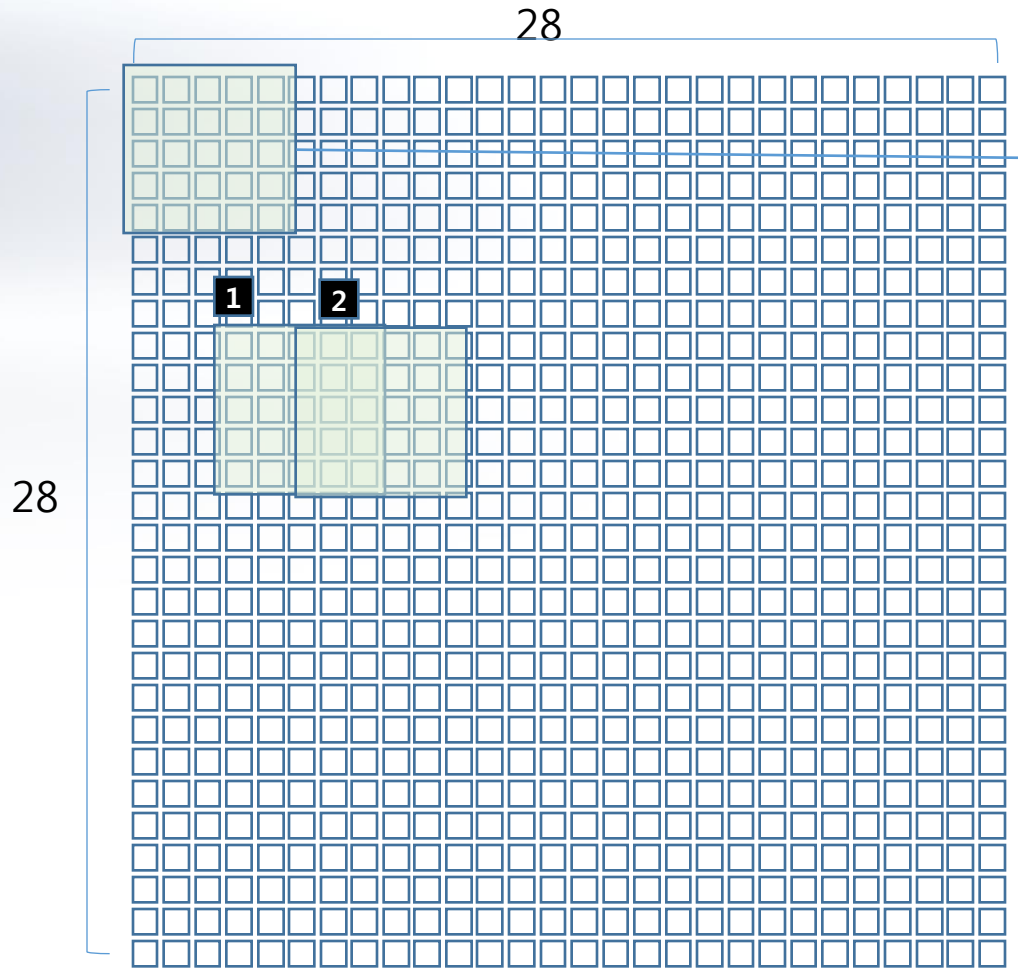
- Convolution
  - Reshape : Vector → Matrix

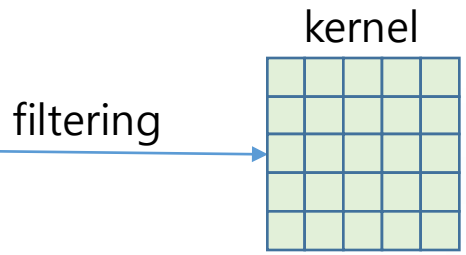


Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

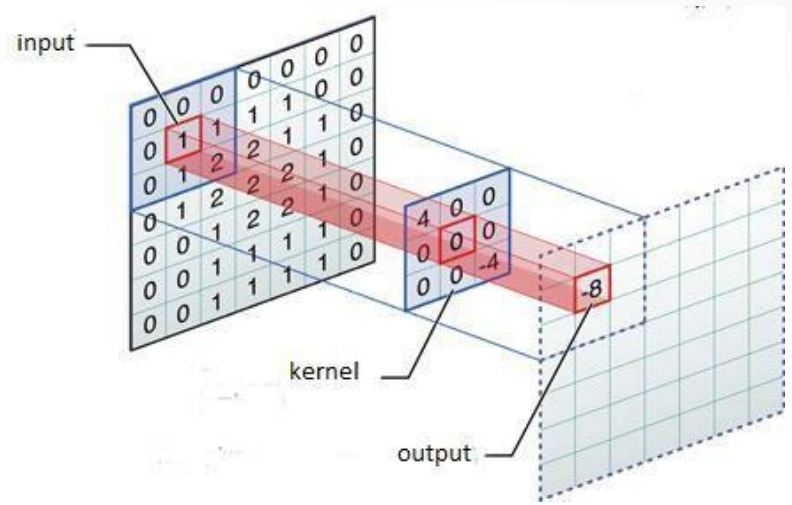
3. CNN

- Convolution
  - Matrix의 부분 집합 선택하여 Filtering





Stride(2,2)



3. CNN

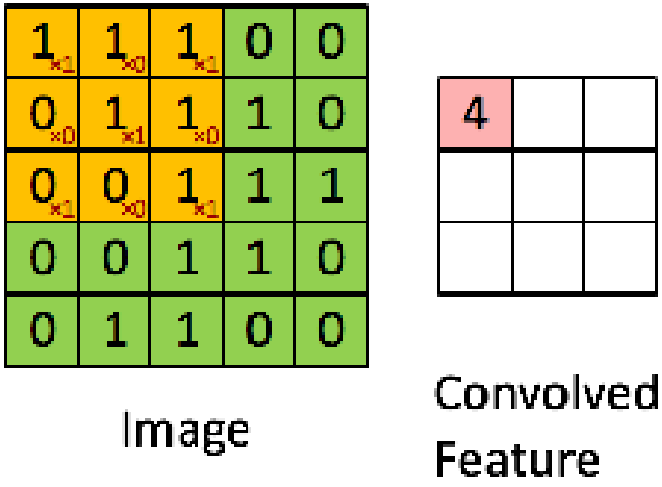
- Filter Size의 결정
  - Filer Size가 클 경우 shrunk
  - Recommended
    - (F – 1/) / 2
    - Zero padding
    - One or Two stride

Zero Padding

(A3 – 1) / 2 = 1

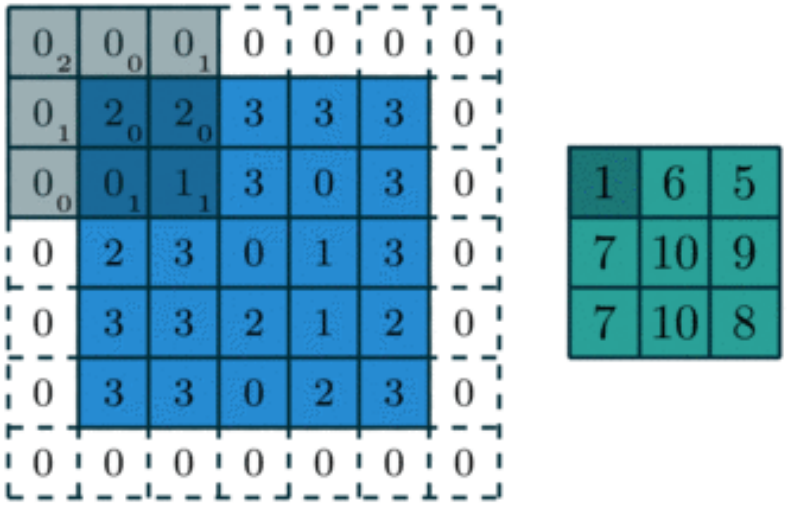
(A3 – 1) / 2 = 1

Without Padding
Stride [1,1,1,1]



[source] https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

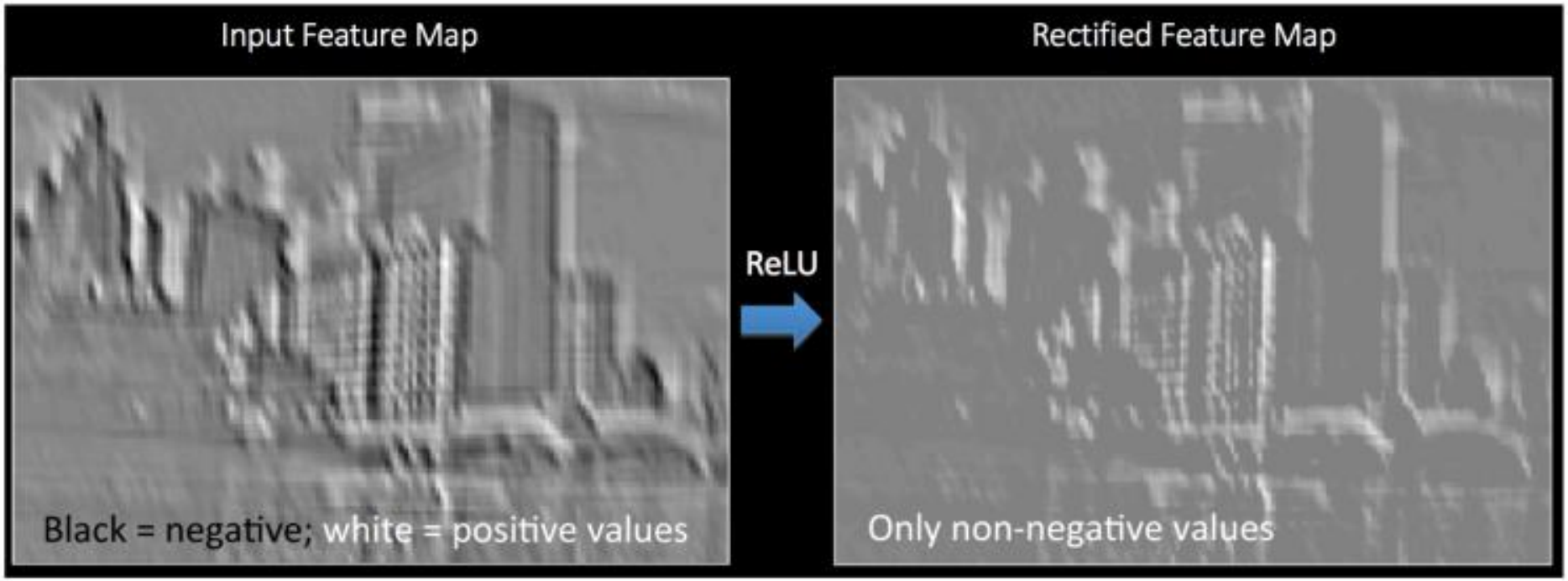
Zero Padding
Stride[1,2,2,1]



[source]

3. CNN

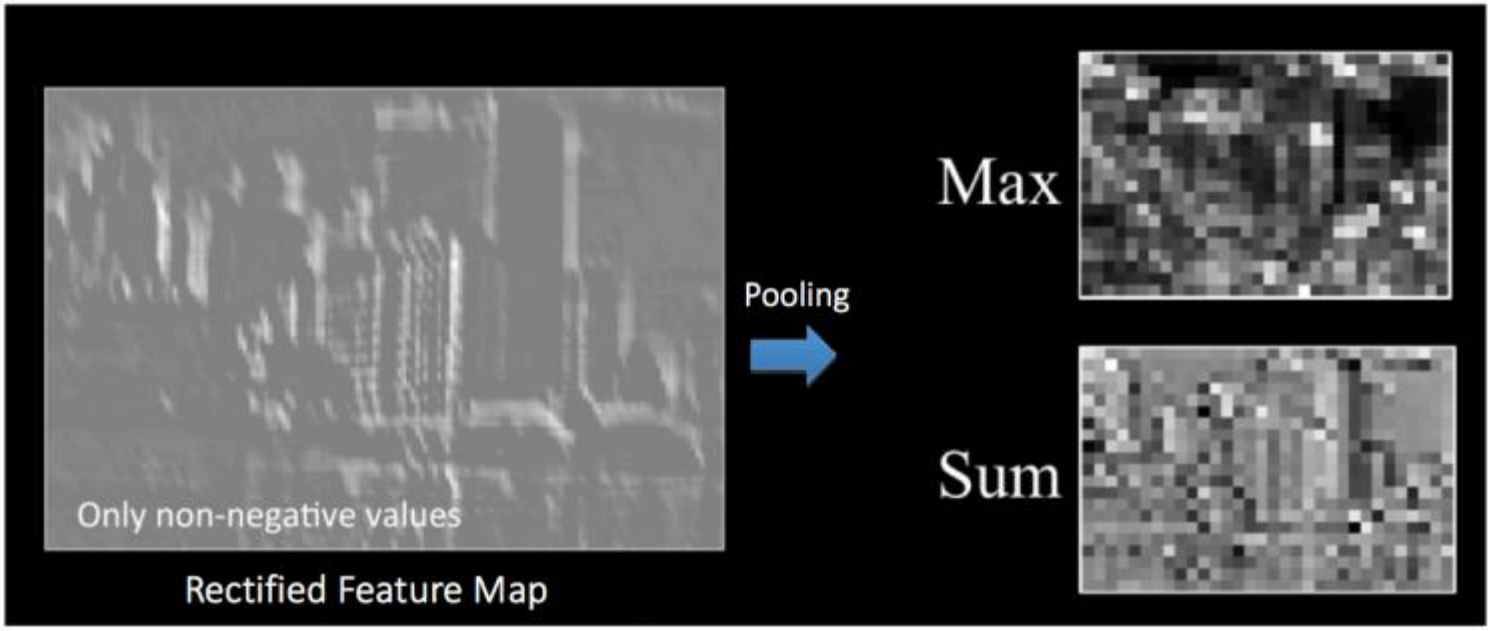
- Feature Map > ReLU effect



[source] https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

3. CNN

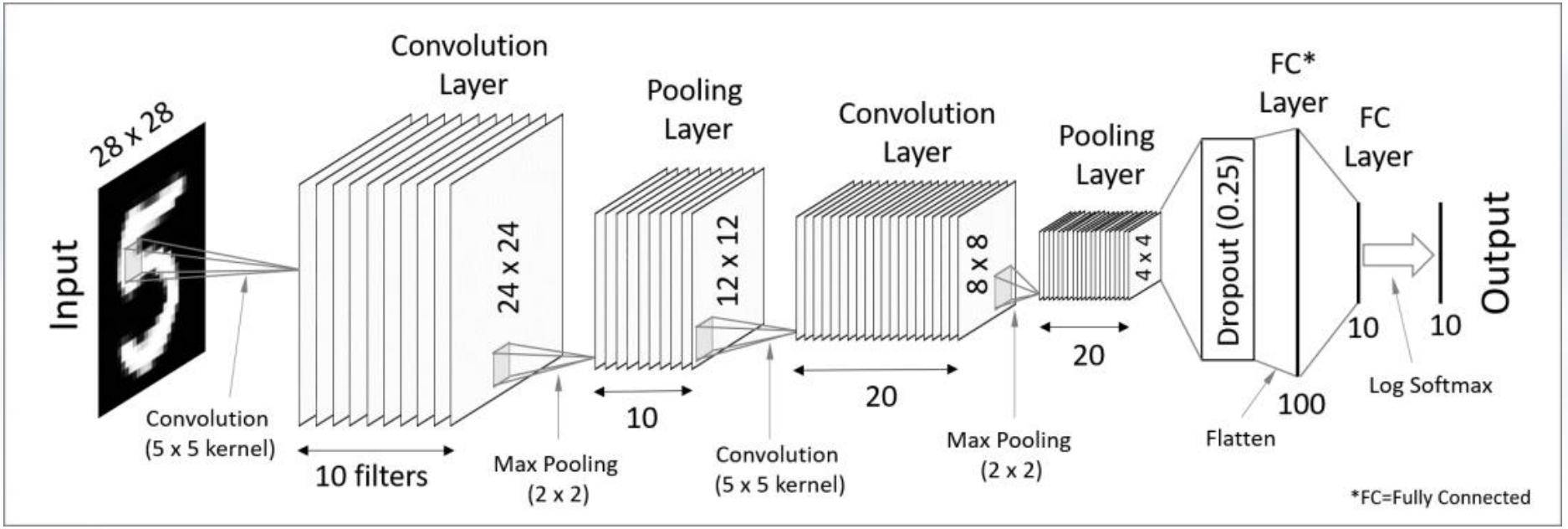
- Feature Map > ReLU effect > Max Pooling



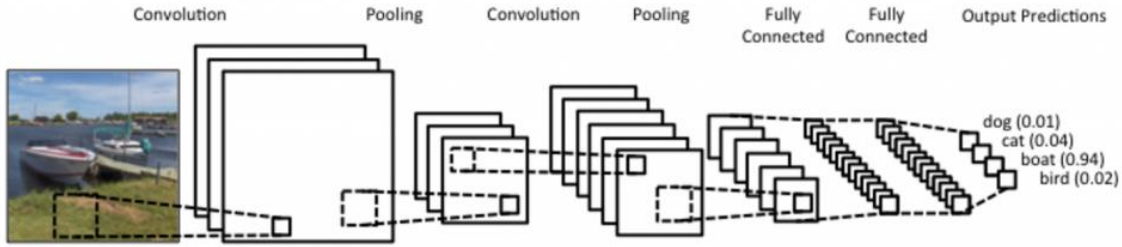
[source] https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

3. CNN

- Convolution



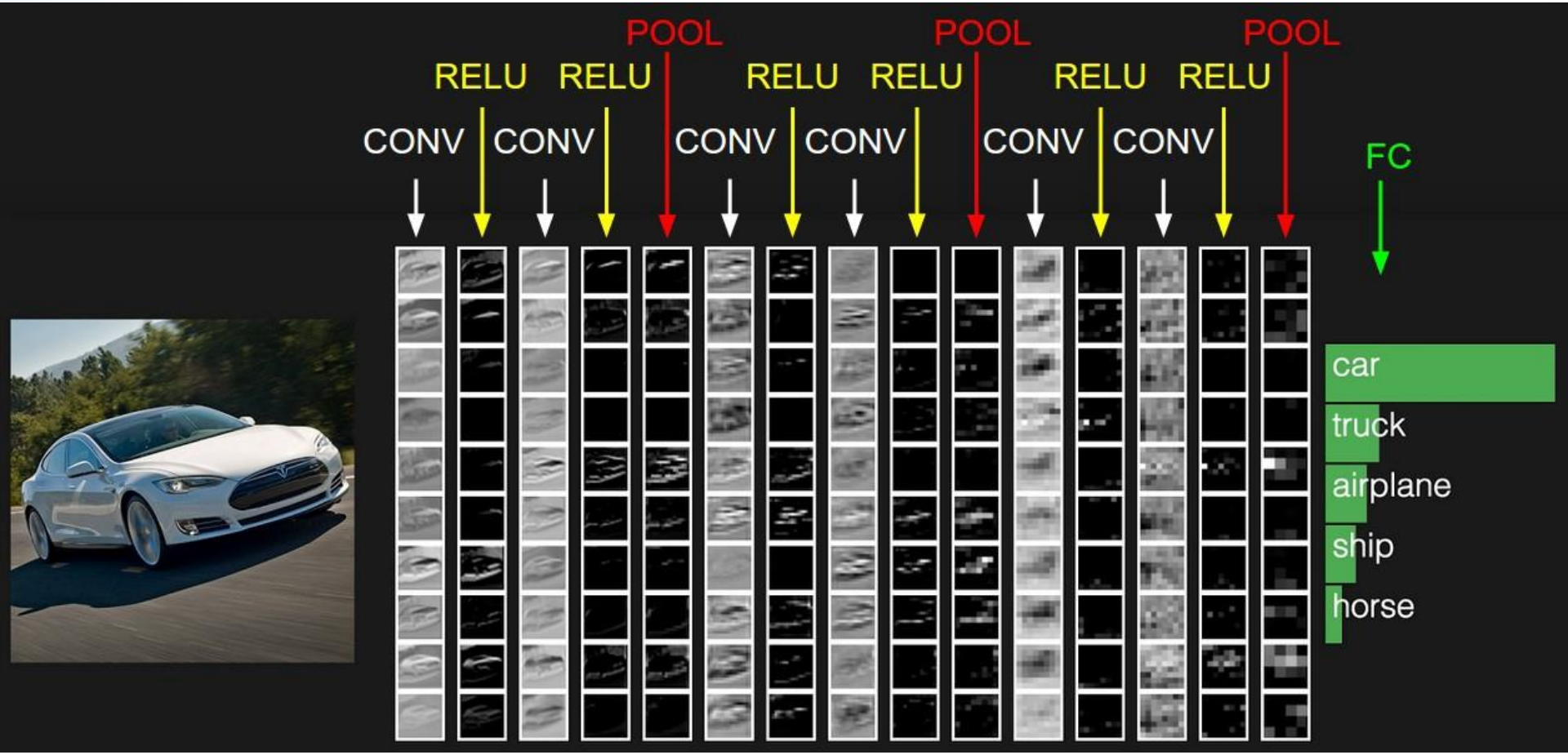
[source] https://codetolight.wordpress.com/2017/11/29/getting-started-with-pytorch-for-deep-learning-part-3-neural-network-basics/



source : http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

3. CNN

- 특징 추출의 과정
- [Convolution → ReLU → Max Pooling] 의 반복 수행



[source] https://medium.com/@udemeudofia01/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17

3. CNN

- Applying Convolutional Neural Network

[source] https://medium.com/@ismailou.sa/convolutional-neural-networks-and-their-application-in-self-driving-cars-33fa0a1625c8

3. CNN

- Program Code

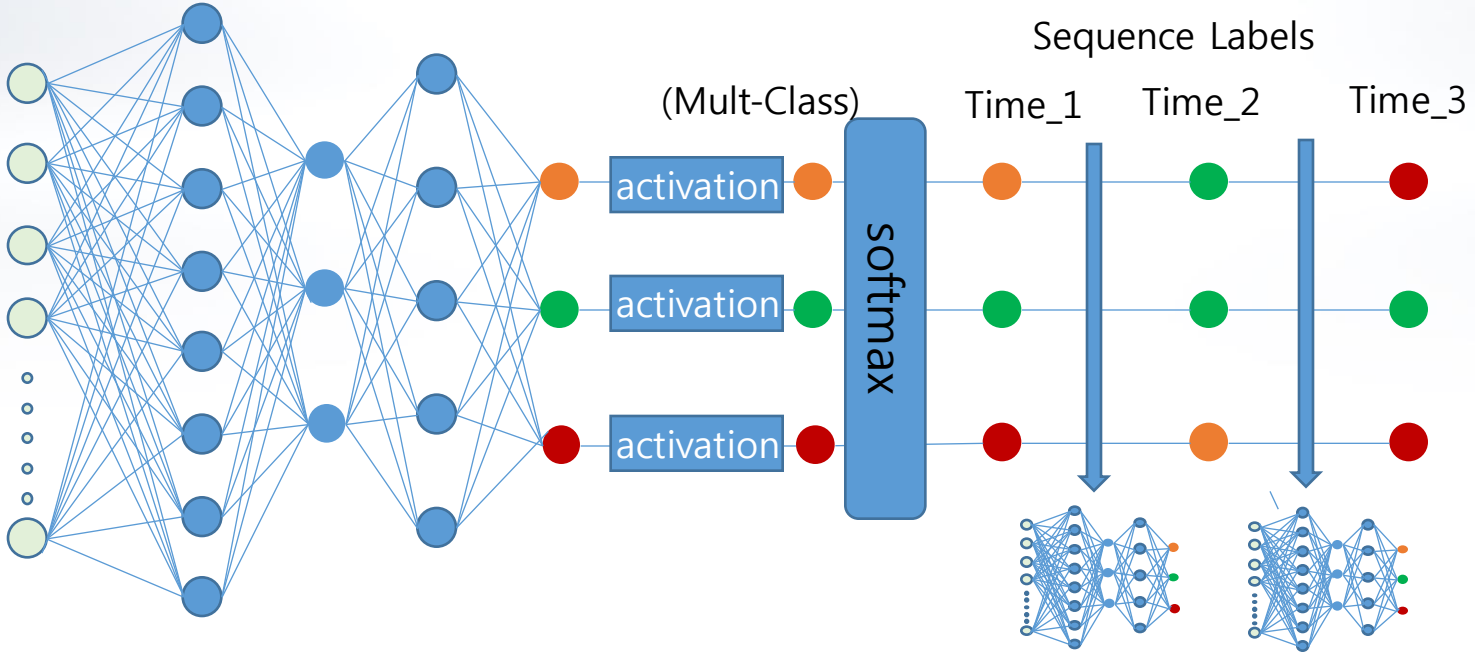
```python
#!/usr/bin/env /c/Apps/Anaconda3/python

# Import
import tensorflow as tf
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

# Import data
data = pd.read_csv('./data_stocks.csv')

# Drop date variable
data = data.drop(['DATE'], 1)

# Dimensions of dataset
n = data.shape[0]
p = data.shape[1]

# Make data a np.array
data = data.values

# Training and test data
train_start = 0
train_end = int(np.floor(0.8*n))
test_start = train_end + 1
test_end = n
data_train = data[np.arange(train_start, train_end), :]
data_test = data[np.arange(test_start, test_end), :]

# Scale data
scaler = MinMaxScaler(feature_range=(-1, 1))
scaler.fit(data_train)
data_train = scaler.transform(data_train)
data_test = scaler.transform(data_test)

# Build X and y
X_train = data_train[:, 1:]
y_train = data_train[:, 0]
X_test = data_test[:, 1:]
y_test = data_test[:, 0]

# Number of stocks in training data
n_stocks = X_train.shape[1]

# Neurons
n_neurons_1 = 1024
n_neurons_2 = 512
n_neurons_3 = 256
n_neurons_4 = 128

# Session
net = tf.InteractiveSession()

# Placeholder
X = tf.placeholder(dtype=tf.float32, shape=[None, n_stocks])
Y = tf.placeholder(dtype=tf.float32, shape=[None])

# Initializer
weight_initializer = tf.variance_scaling_initializer(mode="fan_avg", distribu
bias_initializer = tf.zeros_initializer()

# Hidden weights
W_hidden_1 = tf.Variable(weight_initializer([n_stocks, n_neurons_1]))
bias_hidden_1 = tf.Variable(bias_initializer([n_neurons_1]))
W_hidden_2 = tf.Variable(weight_initializer([n_neurons_1, n_neurons_2]))
bias_hidden_2 = tf.Variable(bias_initializer([n_neurons_2]))
W_hidden_3 = tf.Variable(weight_initializer([n_neurons_2, n_neurons_3]))
bias_hidden_3 = tf.Variable(bias_initializer([n_neurons_3]))
W_hidden_4 = tf.Variable(weight_initializer([n_neurons_3, n_neurons_4]))
bias_hidden_4 = tf.Variable(bias_initializer([n_neurons_4]))

# Output weights
W_out = tf.Variable(weight_initializer([n_neurons_4, 1]))
bias_out = tf.Variable(bias_initializer([1]))

# Hidden layer
hidden_1 = tf.nn.relu(tf.add(tf.matmul(X, W_hidden_1), bias_hidden_1))
hidden_2 = tf.nn.relu(tf.add(tf.matmul(hidden_1, W_hidden_2), bias_hidden_2))
```

실습 중 배포

Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

4. RNN/LSTM

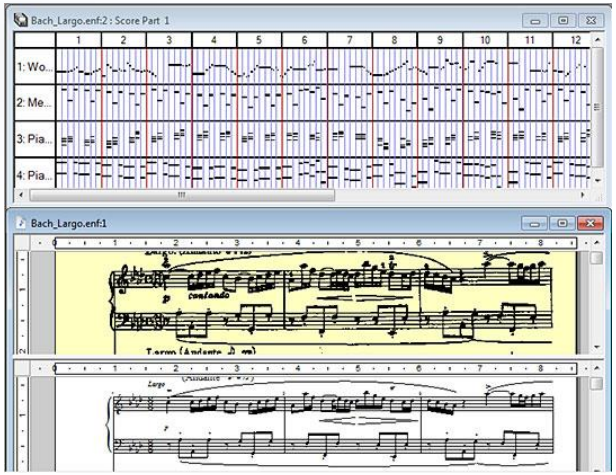
- 문제의 제기
  - Requirement : 실세계에서는 단일예측값을 확장한 연속된 예측값의 필요
  - 실세계에서의 연속 예측값 예시
    - 1주일간의 날씨 예측
    - 시간별 주가 예측

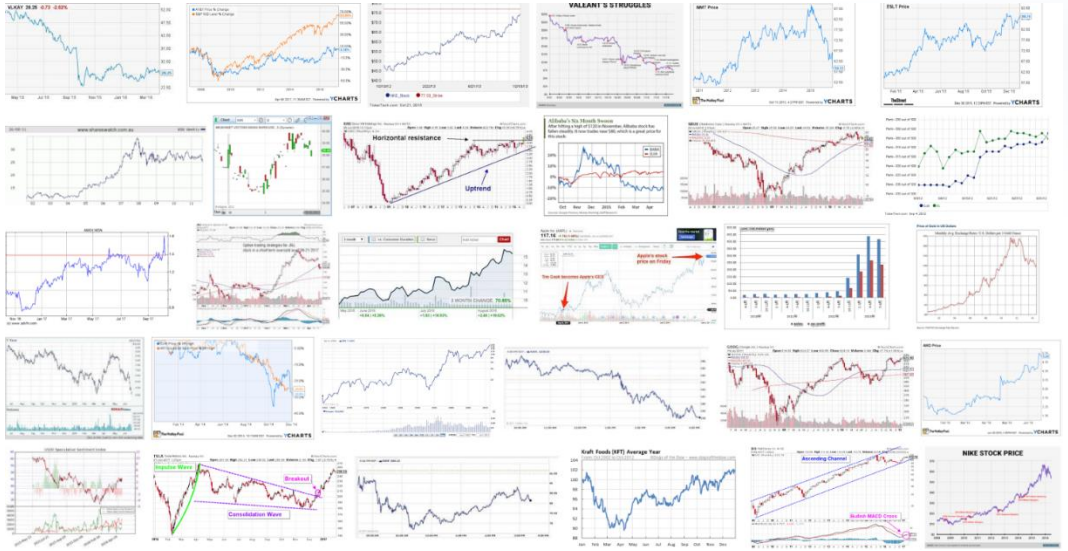


결과값을 입력에 반영하는
재귀적 신경망 학습 필요

4. RNN/LSTM

- Recurrent Neural Network(David Rumelhart, 1986)
  - Learning Sequence Data
  - Sequence Data
    - 수평 또는 수직으로 연속적으로 tensor 의 확장
  - 유형
    - Time Series : Price, Temperature, RPM
    - Streaming : Music, Voice
    - Order : 문장





Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

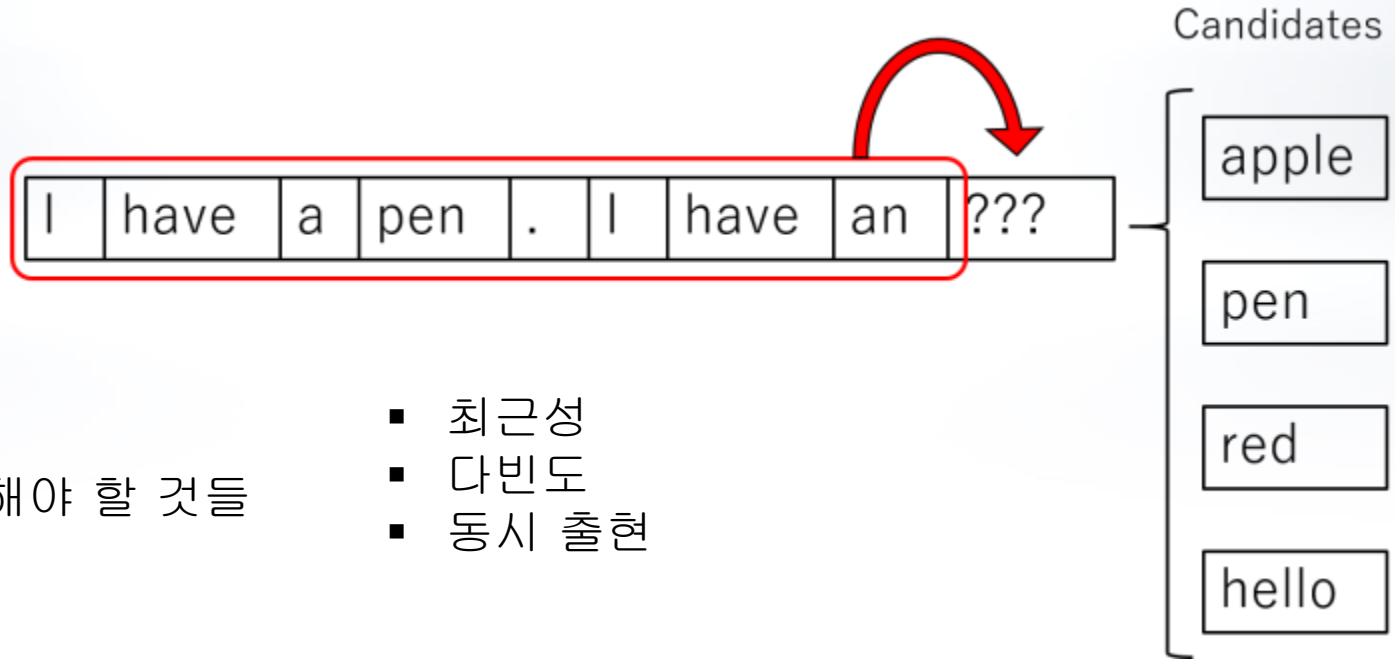
4. RNN/LSTM

- Sequence Data에 있어서의 예측 = Next Sequence

Sequence Type	Features	Label	Label 수
Time Series	1 2 3 4 5 4 5 6 7 8 7 8 9 10 9 10 11	12	Single
Time Series	1 2 3 4 5 4 5 6 7 8 7 8 9 10 9 10 11	12 13	Multi
문장	곰 세마리가 한 집에 있어 아빠곰 엄마곰	애기곰	Single
문장	주가 하락...(중략)...경제 부양 필요	불황	Single
음악	미 레 도 레 미 미	미	Single
음악	미레 도레 미 미 미	레레레	Multi

4. RNN/LSTM

- RNN – Natural Language



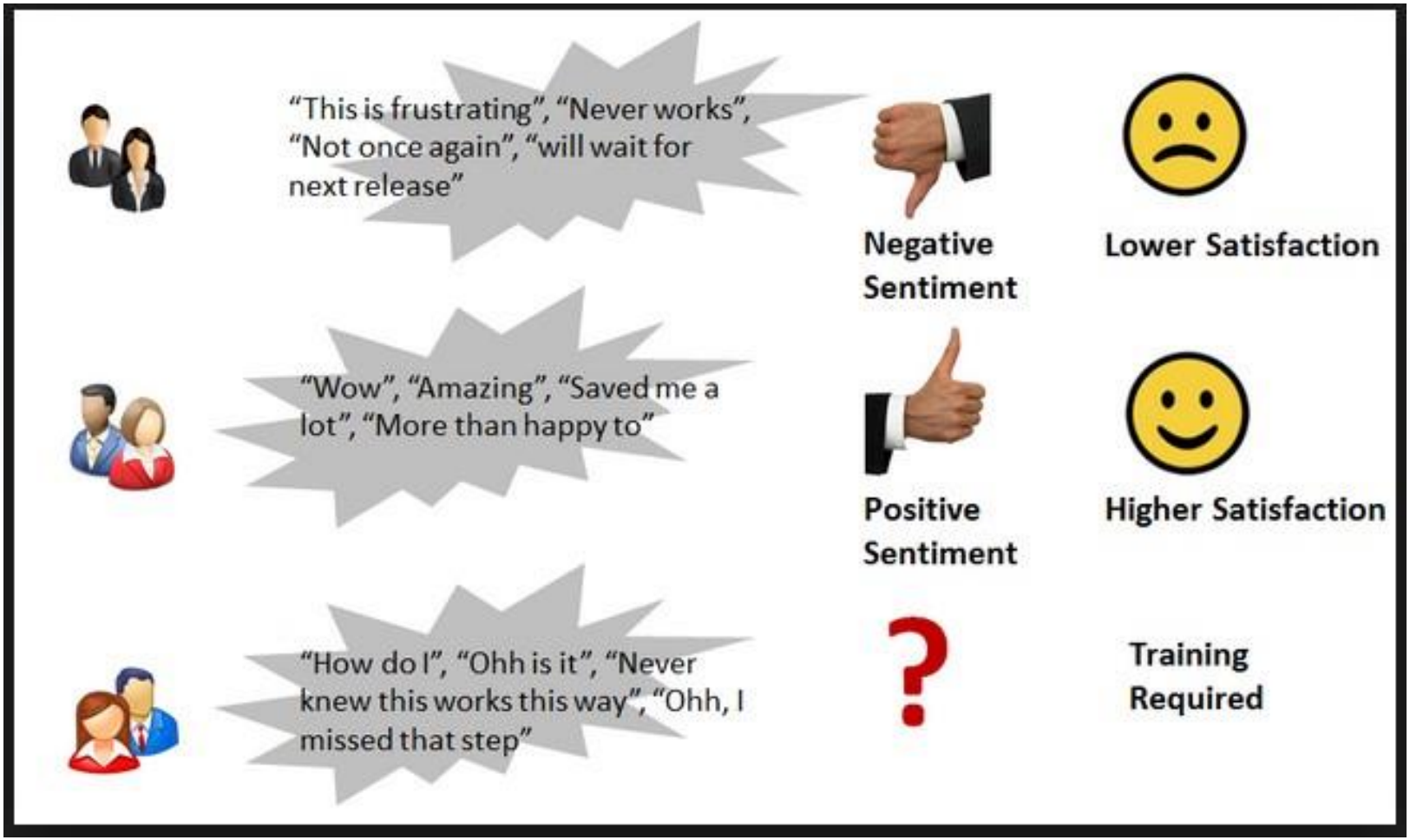
가중치를 부여해야 할 것들

- 최근성
- 다빈도
- 동시 출현

source : http://corochann.com/recurrent-neural-network-rnn-introduction-1286.html

4. RNN/LSTM

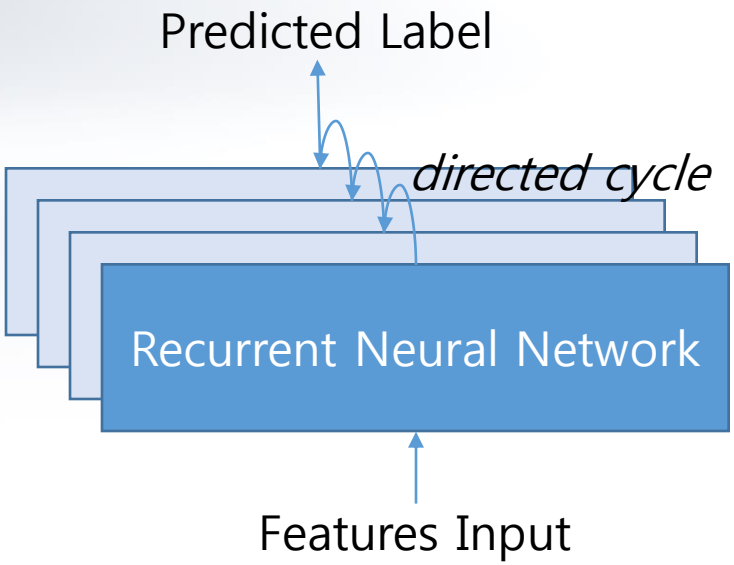
- RNN > 감성 식별



[source] https://datasciencecmu.wordpress.com/2014/04/18/future-of-sentiment-analysis-and-problems-faced/

4. RNN/LSTM

- Recurrent
  - Sequence Data에 있어서의 예측 = Next Sequence



h_t : t시점의 *hidden state*

h_{t-1} : $t-1$ 시점의 *hidden state*

$Features_{t-1}$: t시점의 *hidden state*

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

$$h_t = tanH(W_{hh}h_{t-1} + W_{Featuresh}Features_t)$$

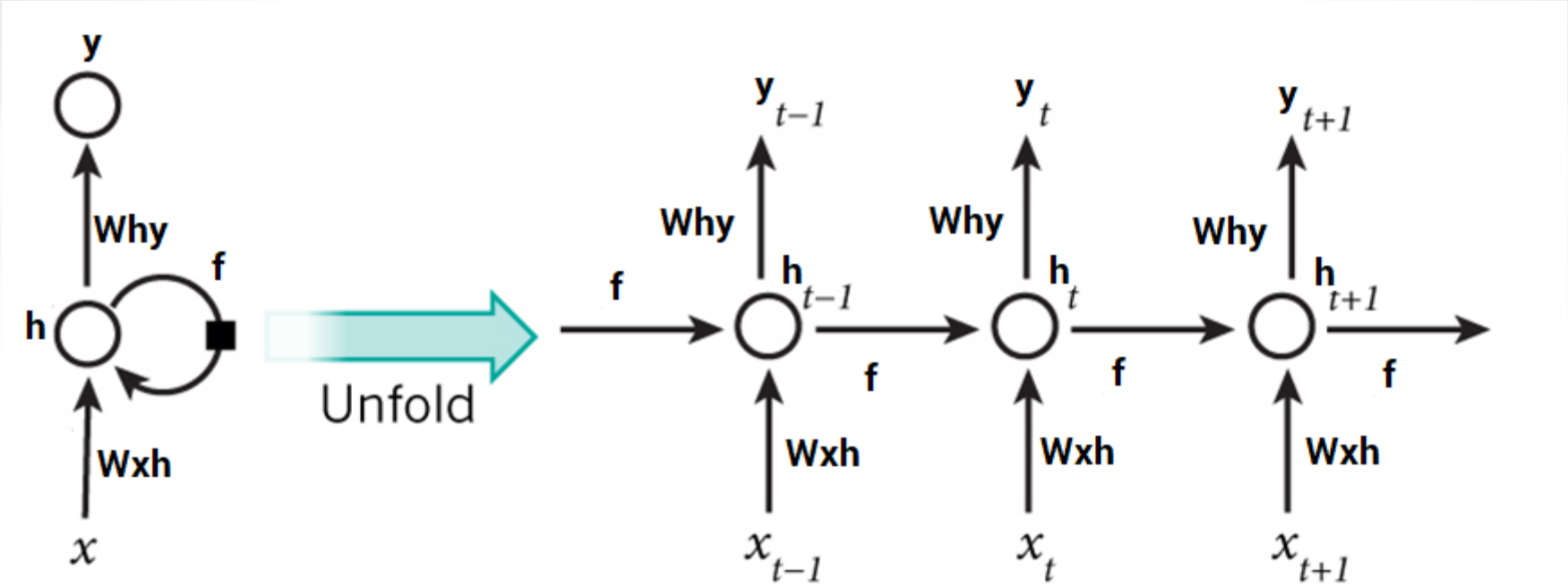
W_{hh} : *recurrent neurons Weight*

$W_{Features*h}$: *Features*에 대입되는 *Weight*

$$Label_t = W_{h*Label} * h_t$$

4. RNN/LSTM

- Recurrent
  - Forward Propagation in a Recurrent Neuron
    - soruce : https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/



[soruce] https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

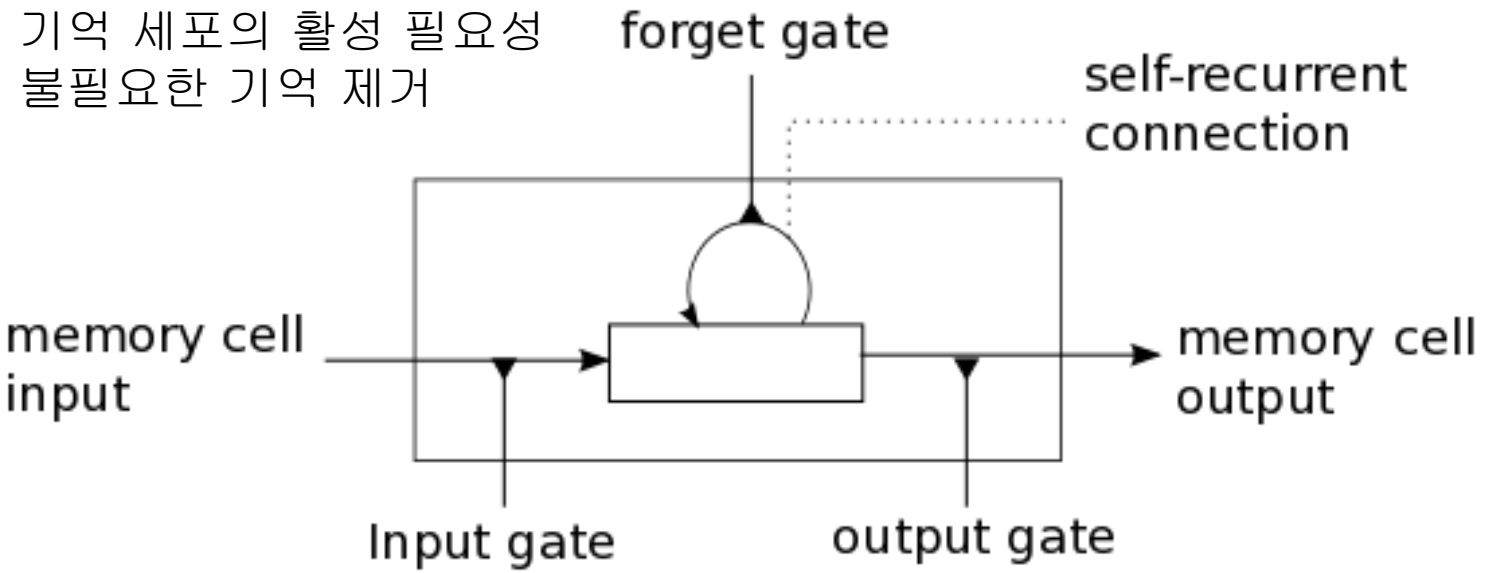
4. RNN/LSTM

- RNN의 구조

Point : 현재의 상태에 영향을 줄법한 과거 찾기

기억 세포의 양날

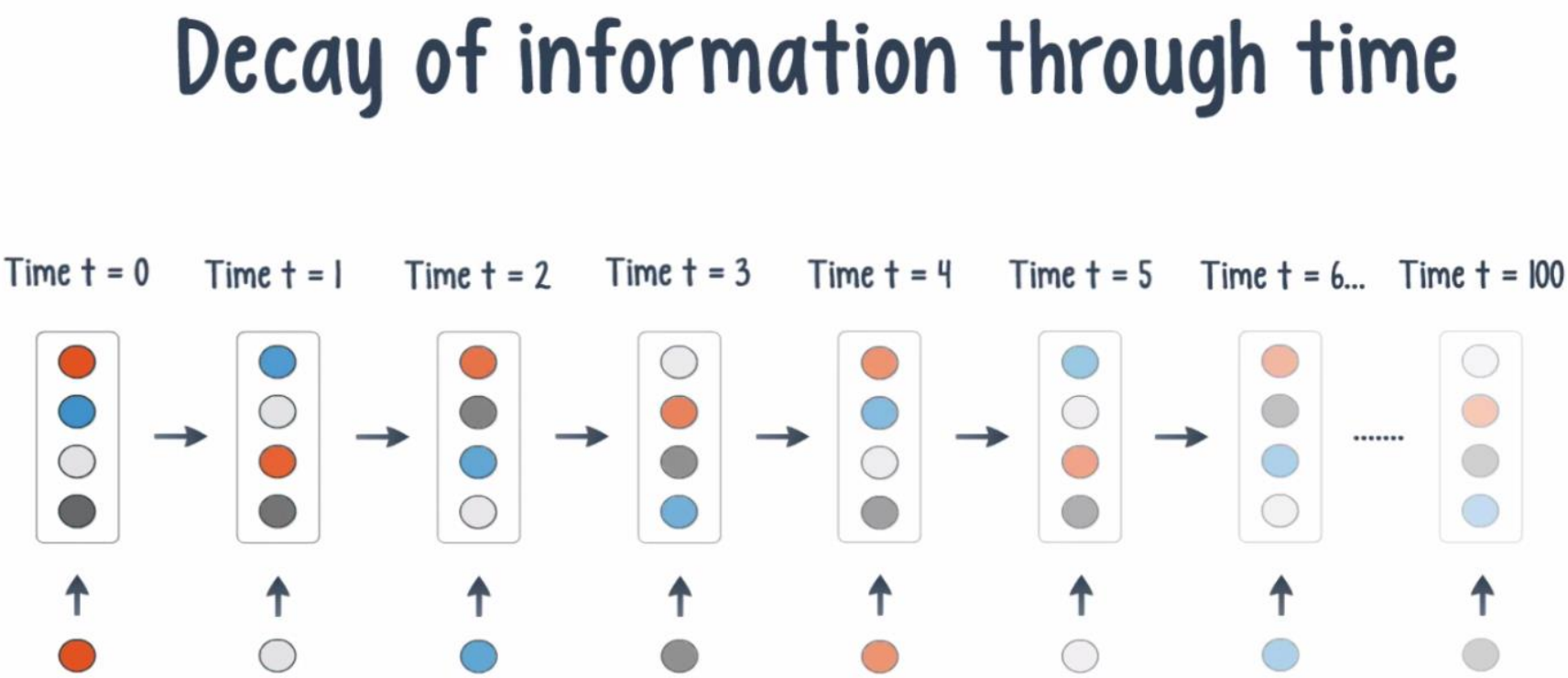
- 기억 세포의 활성 필요성
- 불필요한 기억 제거



기억해야 할만한 것들...

4. RNN/LSTM

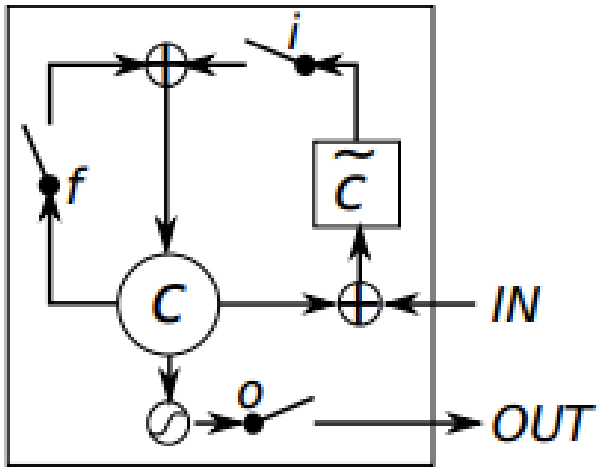
- RNN의 문제점



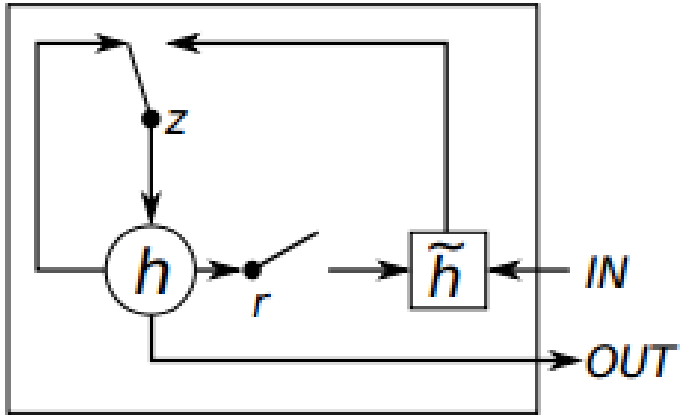
[source] : https://towardsdatascience.com/using-rnns-for-machine-translation-11ddded78ddf

4. RNN/LSTM

- Solution of RNN with Vanishing and Exploding Gradient Problem
  - LSTM : Long Short-Term Memory
  - GRU : Gated Recurrent Unit



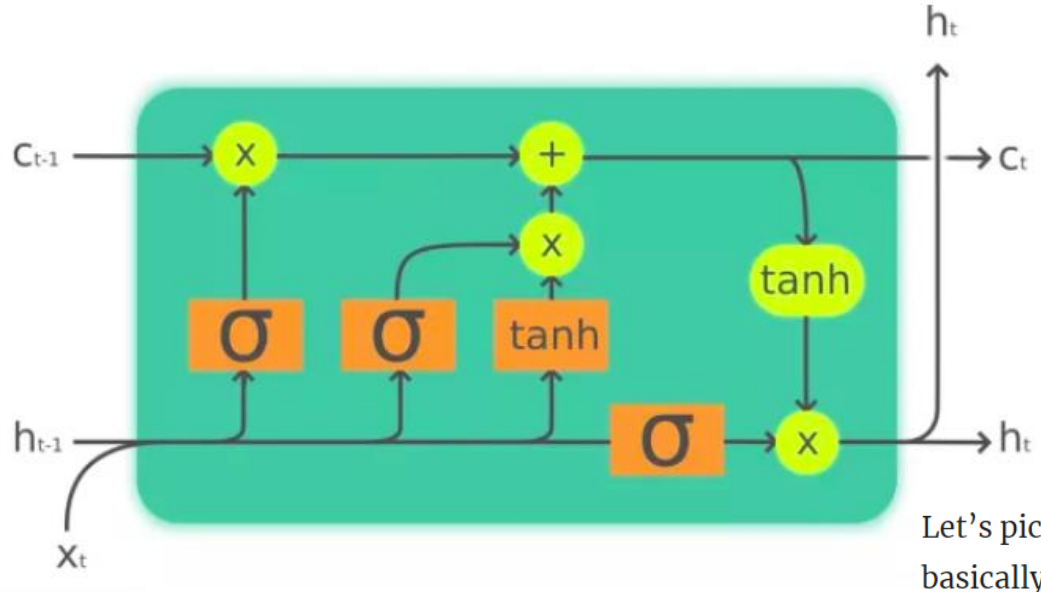
(a) Long Short-Term Memory

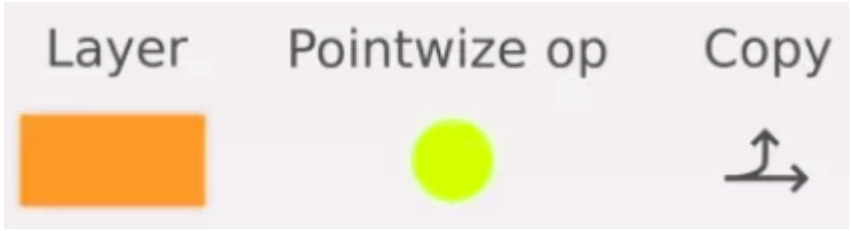


(b) Gated Recurrent Unit

4. RNN/LSTM

- Solution of RNN with Vanishing and Exploding Gradient Problem
  - LSTM : Long Short-Term Memory





$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

Let's pick this equation apart: c_t is the new cell state, which is basically the memory of the LSTM.

f_t is called the "forget gate": it dictates how much of the previous cell state to **retain** (but is slightly confusingly named the forget gate).

i_t is the "input gate" and dictates how much to update the cell state with new information.

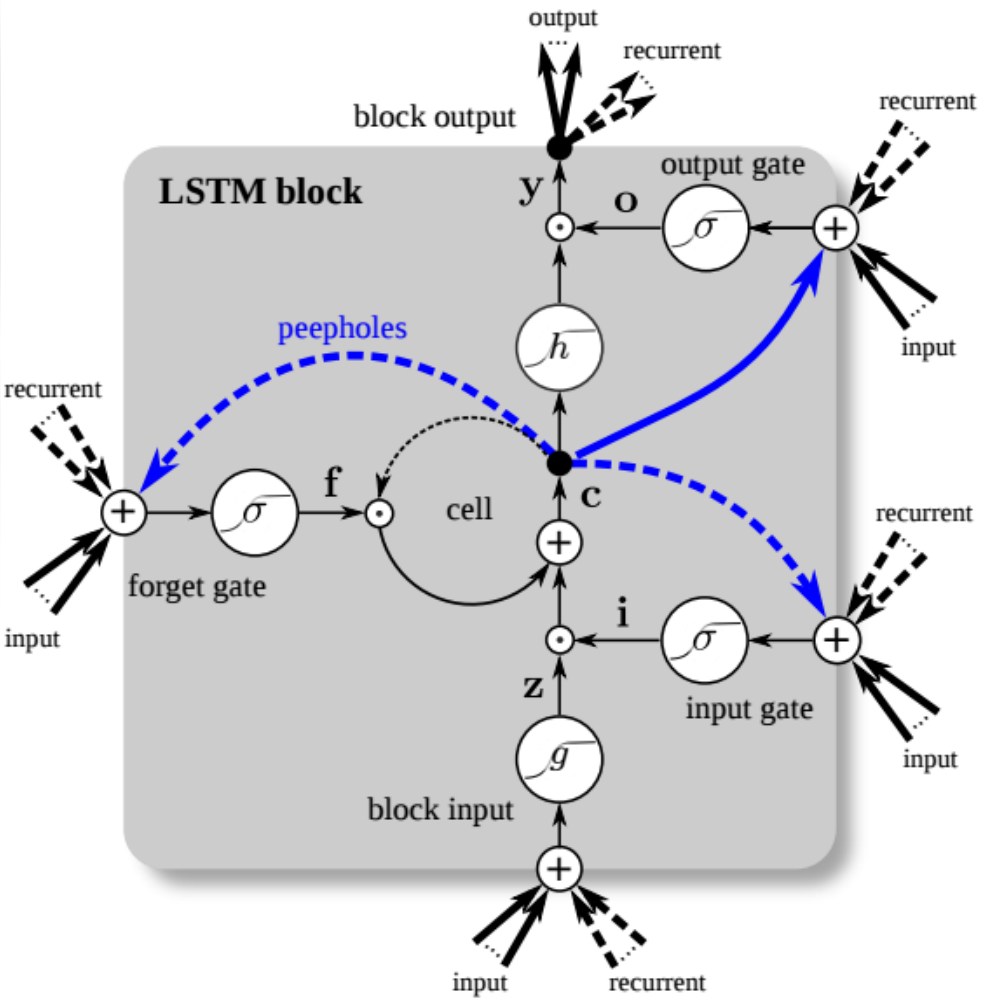
Finally, g_t is the information we use to update the cell state.

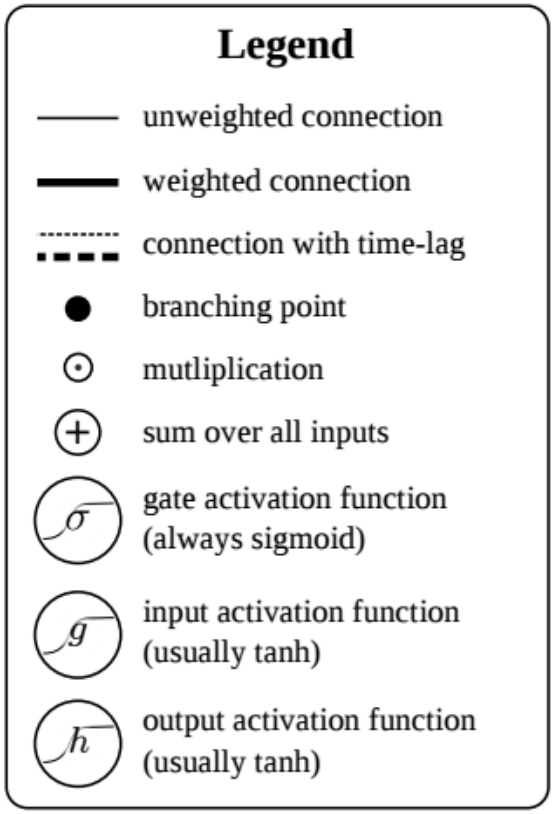
[source] http://mlexplained.com/2019/02/15/building-an-lstm-from-scratch-in-pytorch-lstms-in-depth-part-1/

4. RNN/LSTM

- LSTM의 구조

[source] https://developer.nvidia.com/discover/lstm

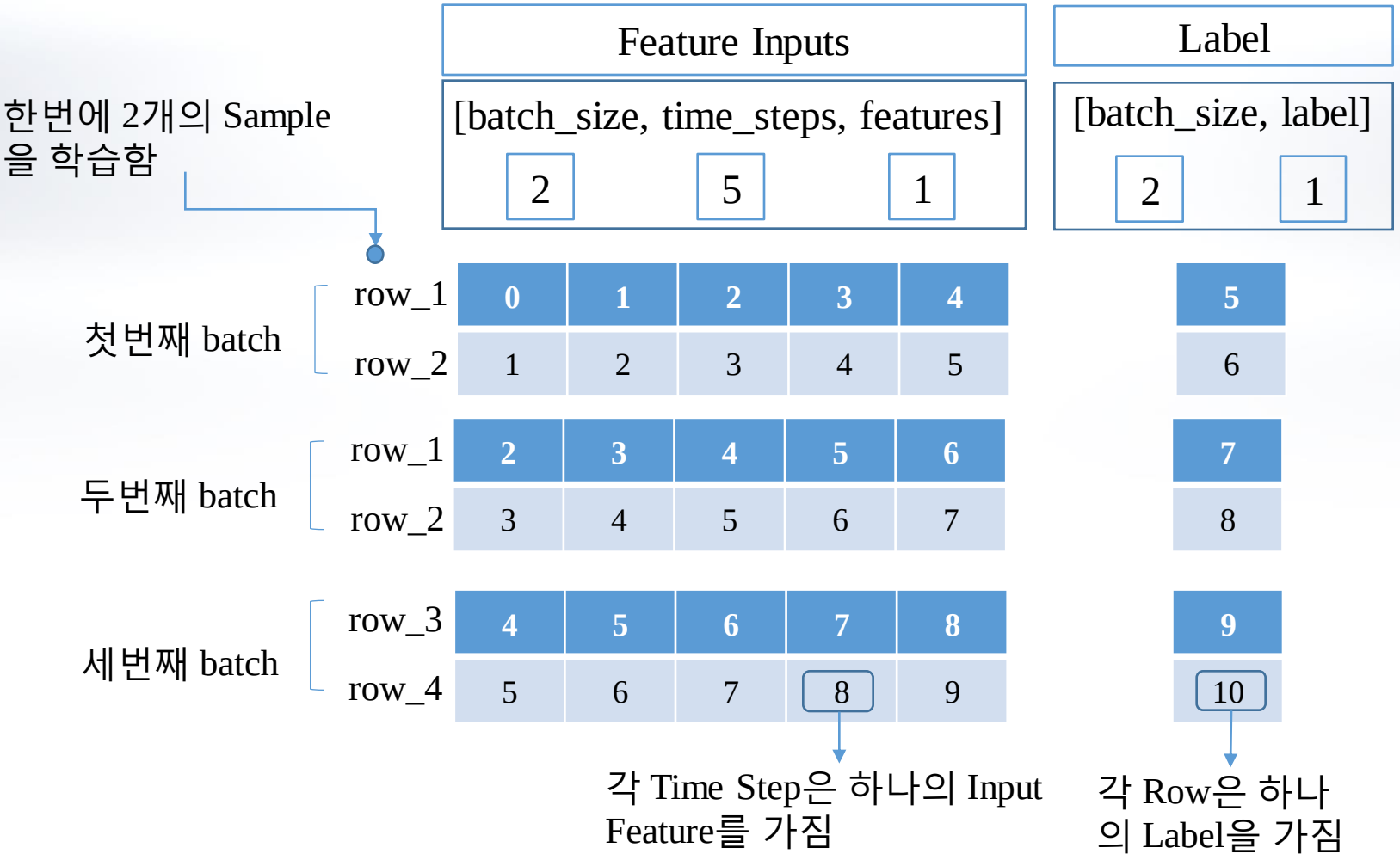




Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

4. RNN/LSTM

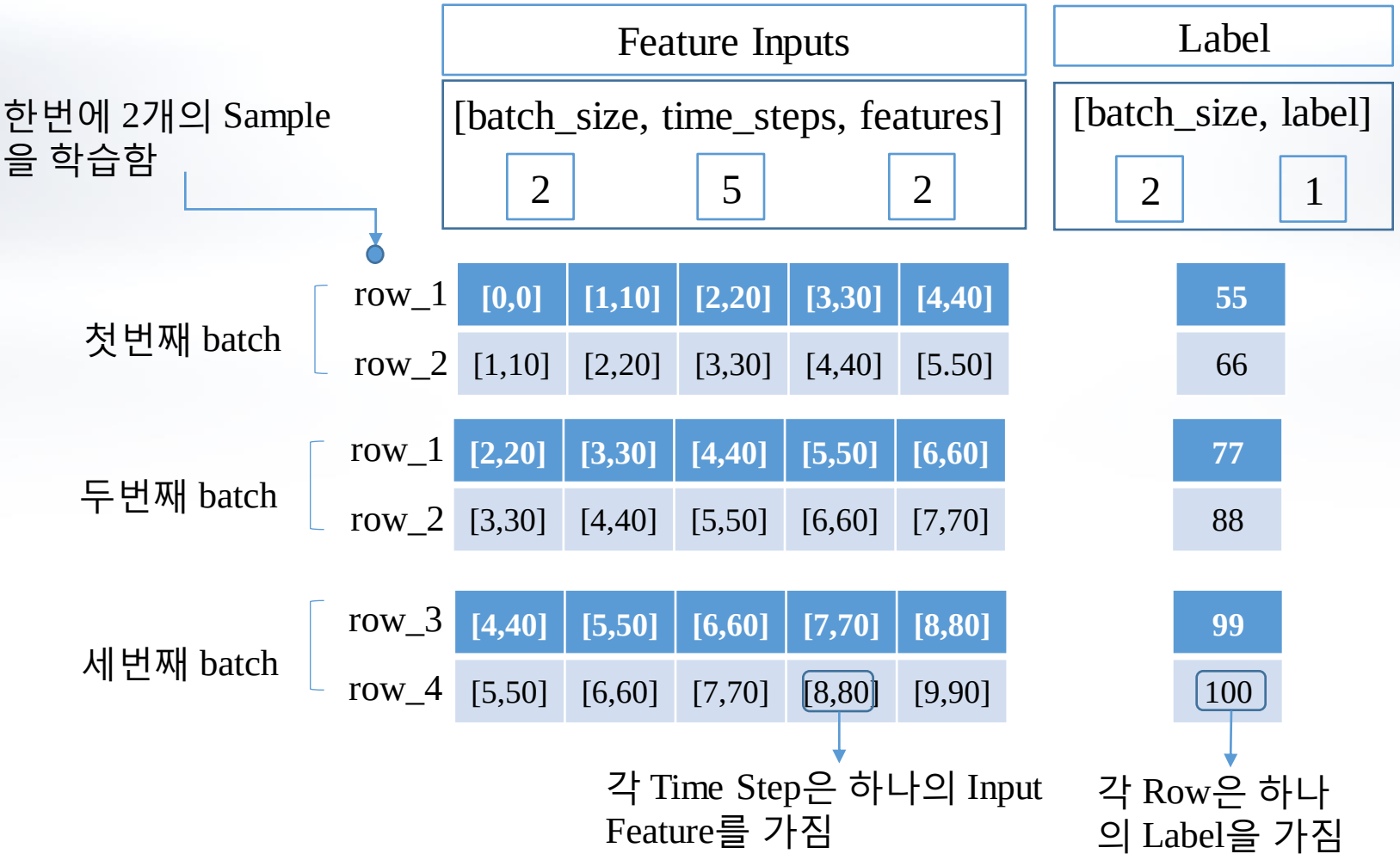
- LSTM : Long Short-Term Memory



Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

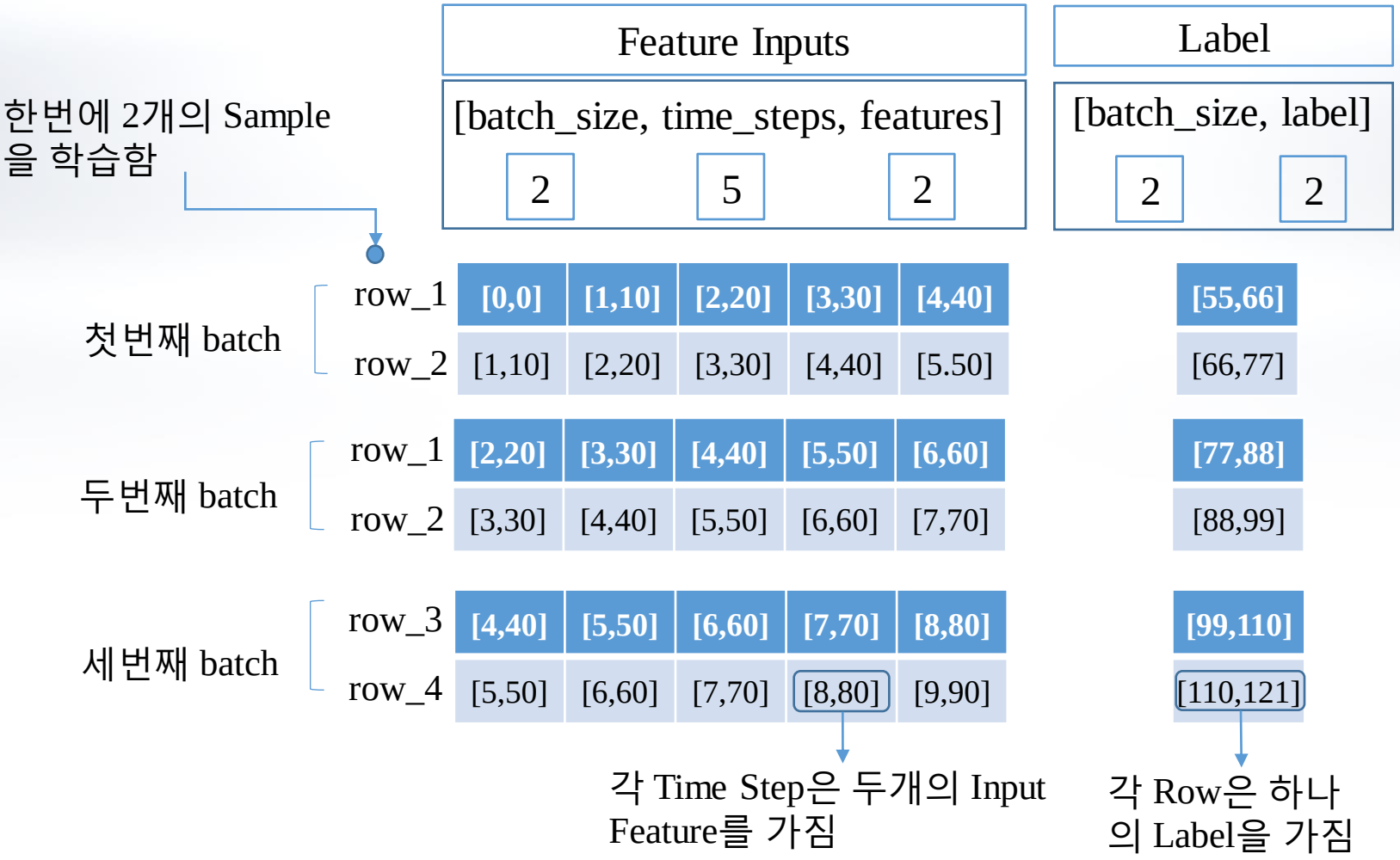
4. RNN/LSTM

- LSTM : Long Short-Term Memory



4. RNN/LSTM

- LSTM : Long Short-Term Memory



4. RNN/LSTM

- LSTM : Long Short-Term Memory
  - Text Detection

Feature Inputs	Label
[batch_size, time_steps, features]	[batch_size, label]
1 6 5	1 5

		H	I	H	E	L	L	I	H	E	L	L	O
첫번째 batch	row_1	[1,0,0,0,0]	[0,1,0,0,0]	[1,0,0,0]	[0,0,1,0]	[0.0,0,1,0]	[0,0,0,1,0]	[0,1,0,0,0]	[1,0,0,0]	[0,0,1,0]	[0.0,0,1,0]	[0,0,0,1,0]	[0,0,0,0,1]

One-Hot Encoding

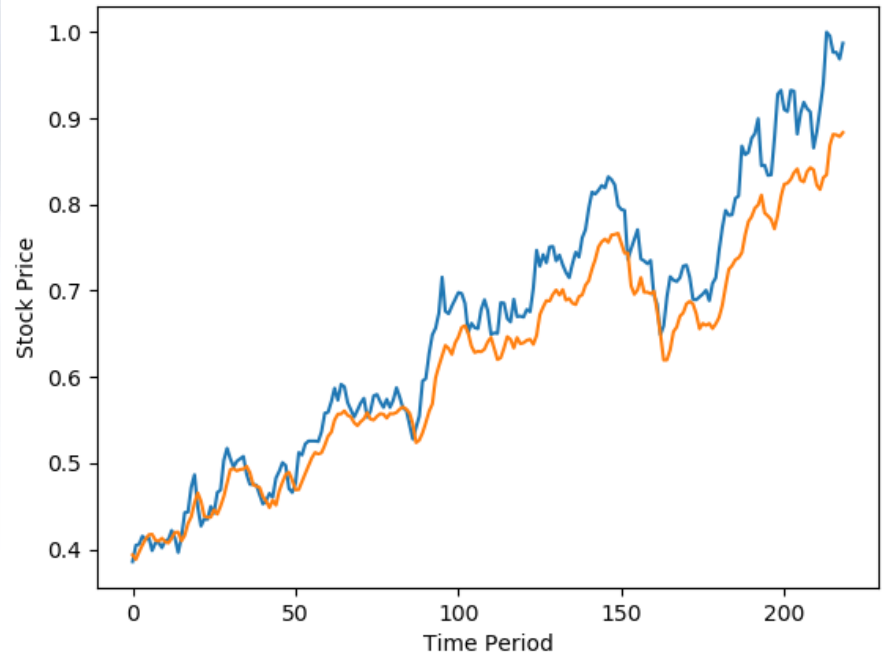
Encode	Decode
[1,0,0,0,0]	H
[0,1,0,0,0]	I
[0,0,1,0,0]	E
[0,0,0,1,0]	L
[0,0,0,0,1]	O

```
placeholderFeatures = tf.placehoder(
    [None, 6, 5]
)

placeholderLabel = tf.placehoder(
    [None, 6]
)

predLabel, states_ = tf.contrib.rnn.BasicLSTMCell(num_units =      )

fcFeatures = tf.reshape(predLabel, [-1, hidden_size])
predLabel = tf.reshape(inputs=fcFeatures, num_outputs=5)
```

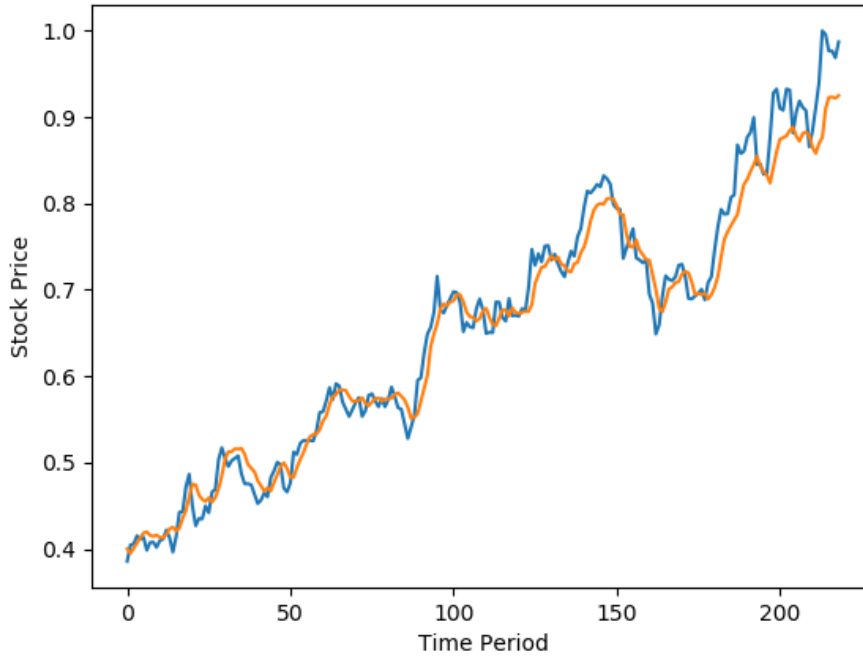
4. RNN/LSTM

- LSTM > 주가 예측

주가 = f(시가, 고가, 저가, 종가, 거래량)

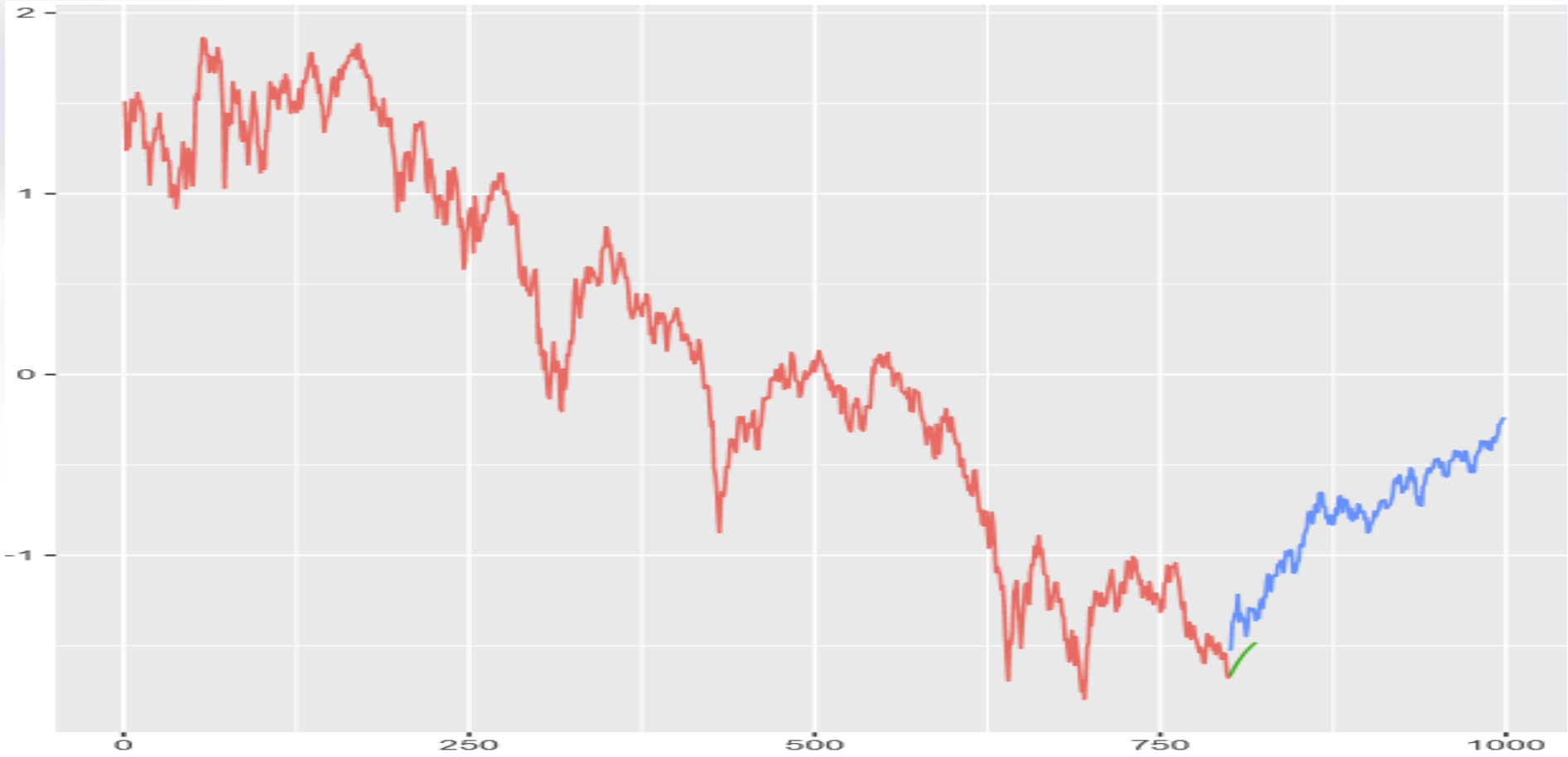


주가 = f(시가, 고가, 저가, 종가, 거래량, **환율**)



4. RNN/LSTM

- LSTM > 주가 예측



Author Yi-Beck Lee(Yibeck.Lee@gmail.com)

4. RNN/LSTM

- Program Code

```python
#!/usr/bin/env /c/Apps/Anaconda3/python

# Import
import tensorflow as tf
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

# Import data
data = pd.read_csv('./data_stocks.csv')

# Drop date variable
data = data.drop(['DATE'], 1)

# Dimensions of dataset
n = data.shape[0]
p = data.shape[1]

# Make data a np.array
data = data.values

# Training and test data
train_start = 0
train_end = int(np.floor(0.8*n))
test_start = train_end + 1
test_end = n
data_train = data[np.arange(train_start, train_end), :]
data_test = data[np.arange(test_start, test_end), :]

# Scale data
scaler = MinMaxScaler(feature_range=(-1, 1))
scaler.fit(data_train)
data_train = scaler.transform(data_train)
data_test = scaler.transform(data_test)

# Build X and y
X_train = data_train[:, 1:]
y_train = data_train[:, 0]
X_test = data_test[:, 1:]
y_test = data_test[:, 0]

# Number of stocks in training data
n_stocks = X_train.shape[1]

# Neurons
n_neurons_1 = 1024
n_neurons_2 = 512
n_neurons_3 = 256
n_neurons_4 = 128

# Session
net = tf.InteractiveSession()

# Placeholder
X = tf.placeholder(dtype=tf.float32, shape=[None, n_stocks])
Y = tf.placeholder(dtype=tf.float32, shape=[None])

izer = tf.variance_scaling_initializer(mode="fan_avg", distribu
bias_initializer = tf.zeros_initializer()

# Hidden weights
W_hidden_1 = tf.Variable(weight_initializer([n_stocks, n_neurons_1]))
bias_hidden_1 = tf.Variable(bias_initializer([n_neurons_1]))
W_hidden_2 = tf.Variable(weight_initializer([n_neurons_1, n_neurons_2]))
bias_hidden_2 = tf.Variable(bias_initializer([n_neurons_2]))
W_hidden_3 = tf.Variable(weight_initializer([n_neurons_2, n_neurons_3]))
bias_hidden_3 = tf.Variable(bias_initializer([n_neurons_3]))
W_hidden_4 = tf.Variable(weight_initializer([n_neurons_3, n_neurons_4]))
bias_hidden_4 = tf.Variable(bias_initializer([n_neurons_4]))

# Output weights
W_out = tf.Variable(weight_initializer([n_neurons_4, 1]))
bias_out = tf.Variable(bias_initializer([1]))

# Hidden layer
hidden_1 = tf.nn.relu(tf.add(tf.matmul(X, W_hidden_1), bias_hidden_1))
hidden_2 = tf.nn.relu(tf.add(tf.matmul(hidden_1, W_hidden_2), bias_hidden_2))
```

실습 중 배포

5. Autoencoder

- 자기 지도 학습
  - 입력 차원들의 재구성을 통한 잠재적(latent) 공통 특징 추출
  - 자기 복원(self reconstruction)
  - 비지도학습과 유사
- 자료의 재표현
  - 데이터의 재표현을 통한 유사도 측정
  - 재표현 과정은 PCA의 변수 축소과정과 동일한 맥락임
  - 재표현 기법
    - Encoding : Latent Representation
    - Decoding : Reconstruction Input Data
  - 재표현 과정에서 정보 손실 발생
    - 정보 손실량를 검토하여 이상치 식별
    - Reconstruction Error가 클수록 이상치

Formula

$$\phi : \mathcal{X} \rightarrow \mathcal{F}$$

$$\psi : \mathcal{F} \rightarrow \mathcal{X}$$

$$\phi, \psi = \arg\min_{\phi,\psi} \|X - (\psi \circ \phi)X\|^2$$

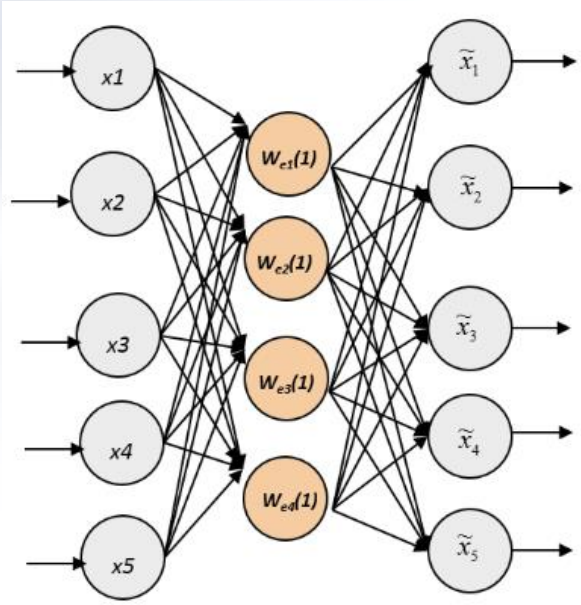
$$\mathbf{z} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

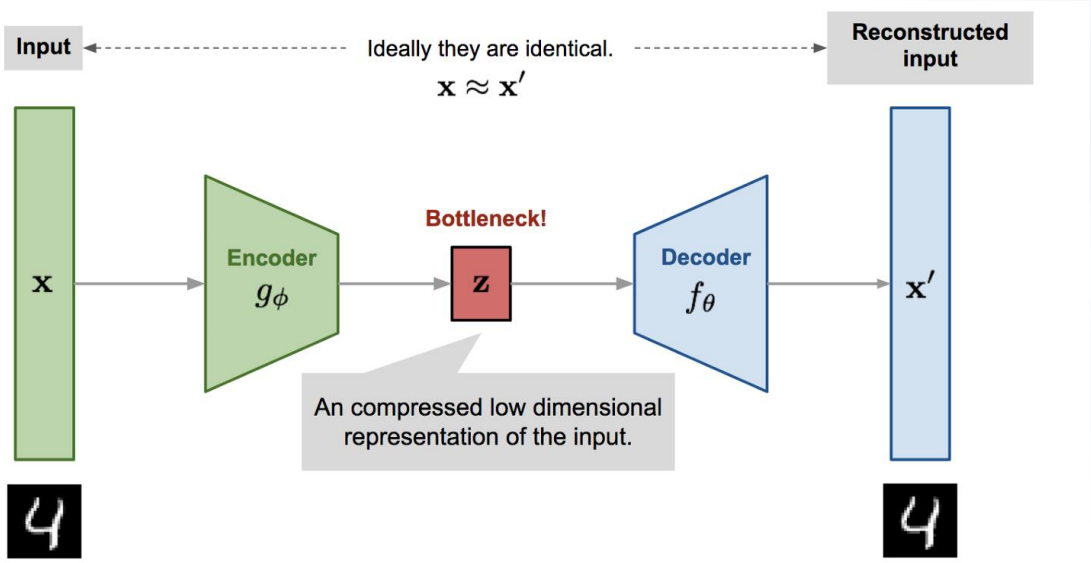
$$\mathbf{x}' = \sigma'(\mathbf{W}'\mathbf{z} + \mathbf{b}')$$

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')\|^2$$

5. Autoencoder

- 차원의 축소
  - Input Dimension(5) 보다 Hidden Dimension(4)의 차원이 적음





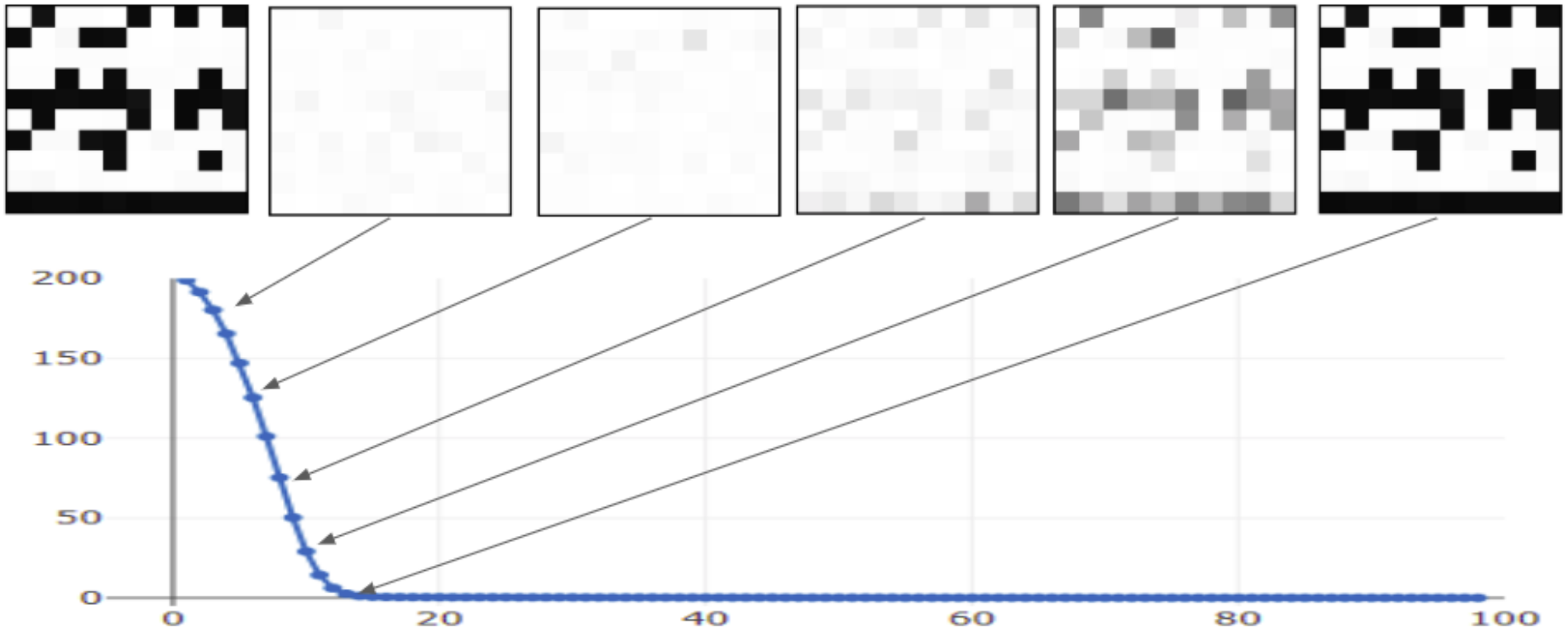
$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')\|^2$$

5. Autoencoder

- 학습
  - 네트워크의 좌우 대칭 구성
  - 입력차원의 수 = 출력차원의 수
  - output = reconstructed input
  - encoder의 hidden dimension의 축소
  - encoder의 hidden dimension의수 = decoder의 hidden dimension의 수
  - 출력차원들은 모든 Raw Data의 변환된 값을 산출
- Output
  - 차원 축소모든 Raw Data의 변환된 값을 산출(입력 차원의 수를 줄이는 것은 아님)
  - 정보 손실을 최소화
  - Reconstruction Error= Input Vector - Output Vector
  - PCA보다 높은 성능
  - Sparse한 자료에도 상대적으로 높은 성능

5. Autoencoder

- 학습 과정 데모 시연



[source] https://github.com/max-ng/Autoencoder-in-javascript

5. Autoencoder

- Program Code

```python
#!/usr/bin/env /c/Apps/Anaconda3/python

# Import
import tensorflow as tf
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

# Import data
data = pd.read_csv('./data_stocks.csv')

# Drop date variable
data = data.drop(['DATE'], 1)

# Dimensions of dataset
n = data.shape[0]
p = data.shape[1]

# Make data a np.array
data = data.values

# Training and test data
train_start = 0
train_end = int(np.floor(0.8*n))
test_start = train_end + 1
test_end = n
data_train = data[np.arange(train_start, train_end), :]
data_test = data[np.arange(test_start, test_end), :]

# Scale data
scaler = MinMaxScaler(feature_range=(-1, 1))
scaler.fit(data_train)
data_train = scaler.transform(data_train)
data_test = scaler.transform(data_test)

# Build X and y
X_train = data_train[:, 1:]
y_train = data_train[:, 0]
X_test = data_test[:, 1:]
y_test = data_test[:, 0]

# Number of stocks in training data
n_stocks = X_train.shape[1]

# Neurons
n_neurons_1 = 1024
n_neurons_2 = 512
n_neurons_3 = 256
n_neurons_4 = 128

# Session
net = tf.InteractiveSession()

# Placeholder
X = tf.placeholder(dtype=tf.float32, shape=[None, n_stocks])
Y = tf.placeholder(dtype=tf.float32, shape=[None])

실습 중 배포

izer = tf.variance_scaling_initializer(mode="fan_avg", distribu
bias_initializer = tf.zeros_initializer()

# Hidden weights
W_hidden_1 = tf.Variable(weight_initializer([n_stocks, n_neurons_1]))
bias_hidden_1 = tf.Variable(bias_initializer([n_neurons_1]))
W_hidden_2 = tf.Variable(weight_initializer([n_neurons_1, n_neurons_2]))
bias_hidden_2 = tf.Variable(bias_initializer([n_neurons_2]))
W_hidden_3 = tf.Variable(weight_initializer([n_neurons_2, n_neurons_3]))
bias_hidden_3 = tf.Variable(bias_initializer([n_neurons_3]))
W_hidden_4 = tf.Variable(weight_initializer([n_neurons_3, n_neurons_4]))
bias_hidden_4 = tf.Variable(bias_initializer([n_neurons_4]))

# Output weights
W_out = tf.Variable(weight_initializer([n_neurons_4, 1]))
bias_out = tf.Variable(bias_initializer([1]))

# Hidden layer
hidden_1 = tf.nn.relu(tf.add(tf.matmul(X, W_hidden_1), bias_hidden_1))
hidden_2 = tf.nn.relu(tf.add(tf.matmul(hidden_1, W_hidden_2), bias_hidden_2))
```

KAIT 한국정보통신진흥협회
Korea Association for ICT Promotion