

Deep Learning ?

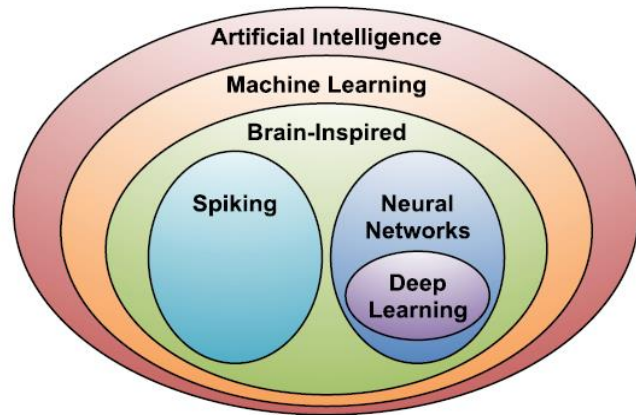


Fig. 1. Deep learning in the context of artificial intelligence.

DNNs : Deep Neural Networks

- DNNs는 AI의 넓은 분야에서 Deep Learning이라 부른다.
- Machine Learning은 컴퓨터가 프로그램 된 것 없이 컴퓨터가 학습할 수 있는 능력을 준다.
- 인간의 뇌의 주된 계산하는 부분은 "뉴런(neuron)"이다.

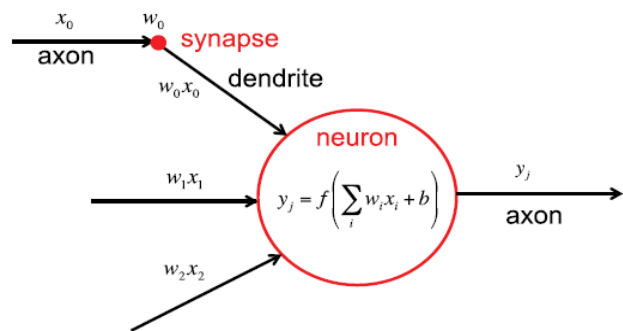


Fig. 2. Connections to a neuron in the brain. x_i , w_i , $f(\cdot)$, and b are the activations, weights, nonlinear function, and bias, respectively. (Figure adopted from [7].)

- 사람의 뇌에는 평균 860억 개의 뉴런들이 있고, 이는 축삭돌기(axon)와 시냅스(synapse)가 곁해진 수상돌기(dendrite)가 뉴런의 입력으로 들어가고 이는 또 다른 뉴런의 입력 axon이 된다.

"Neural Networks 는 input의 weighted sum인 nonlinear function이다"

Deep Learning ?

DNNs는 단순하지 않고 복잡하다.

Ex) Image의 pixel이 입력으로 들어가면

첫 번째 layer : lines과 edges를 해석

두 번째 layer : lines과 edge를 shape로 결합해서 high level features의 존재 가능성 측정

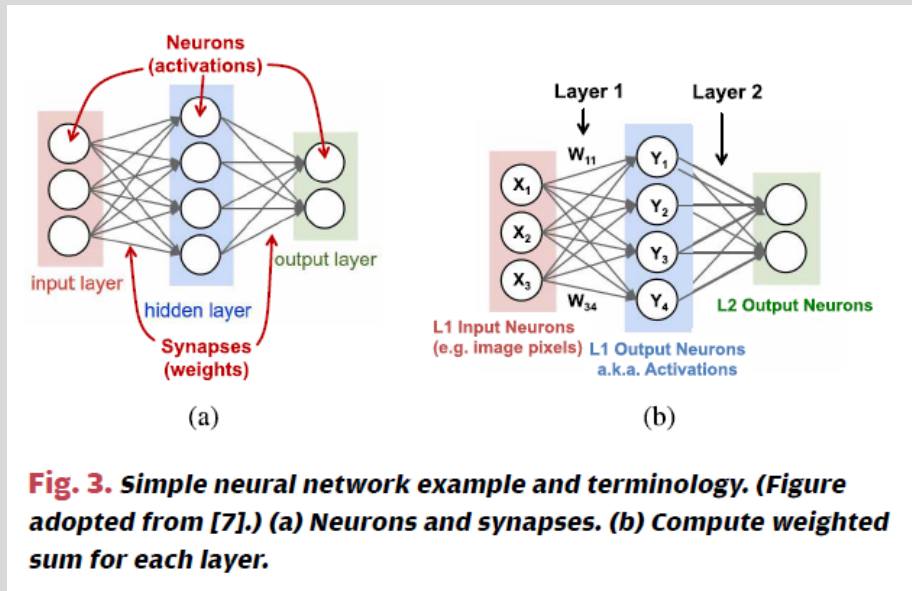
·
·
·

최종 layer : 모든 information을 결합해 high-level features들이 특정 물체나 장면으로 이루어졌다는 가능성을 준다.

“DNNs are capable of learning high-level features with more complexity and abstraction than shallower neural networks”

(High-level features란? : Object의 일부분이나 전체
↔ Low-level features : edge, corner, color)

이 Deep feature hierarch는 DNNs가 many tasks에서 더 나은 performance를 얻도록 해준다.



<“Simple” Neural Network>

Fig.3 (a) → Synapses는 weights로 언급

Deep Learning ?

학습(learning) 이란?

DNNs의 경우 학습이란 **가중치를 결정하는 것**을 포함한다.

학습은 **Training**이라고도 한다.

추론(Inference)이란?

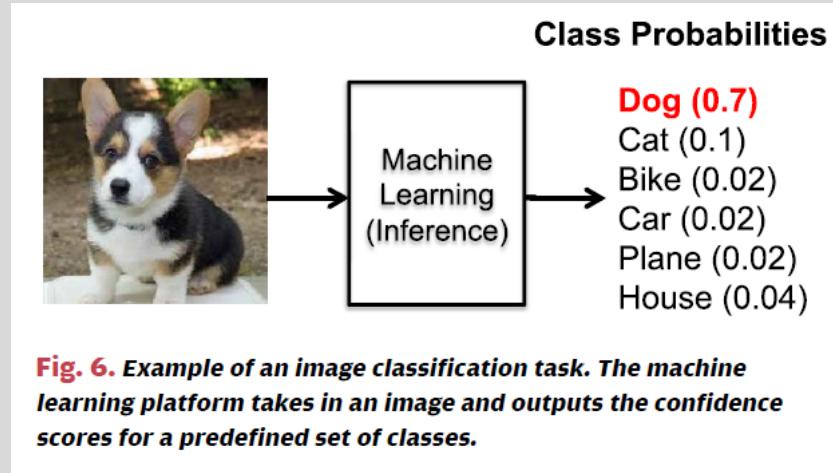
새로운 입력이 들어오면 데이터를 갖고 학습을 거쳐서 결정된 Weight로 Output을 계산한다.

이렇게 계산된 Output을 "**추론 값, 예측 값**"이라 부른다.

DNN을 training하는 대단히 중요한 목적은 올바른 값에 대해서는 높은 점수를, 틀린 값에 대해선 낮은 점수를 출력하는 Weight를 결정하는 것이다.

즉, large training set에 대해서 평균 loss를 줄이기 위함이다.

어떻게 평균 loss가 최소일 때 가중치를 찾는 방법은 뒤에서 소개와 다음 발표에서 자세히 이야기한다.



Deep Learning ?

<Weight를 training시키는 방법들>

1. **Supervised Learning** : Training Samples are labeled (정답을 갖고 학습)
2. **Unsupervised Learning** : Training Samples are not labeled(정답 없이 학습, 스스로 규칙을 찾아야 함)
3. **Semi-Supervised Learning** : a small subset of the training data is labeled
4. **Reinforcement learning** : 현재 상태가 주어졌을 때, DNN은 선택 가능한 행동 중 보상이 최대가 되는 행동, 방법을 선택한다.
→ 현실적으로 많은 데이터를 모으지 못할 경우 사용할 수 있다.
5. **Fine-tuning**
Weight를 고정시키고 임의의 입력 x 를 넣었을 때 출력 값을 구한다.(Forward propagation)
Forward propagation으로 구한 출력 값을 back propagation으로 입력 값 x' 을 구한다.
 $|x - x'|$ 이 최소가 되는 **Weight**를 고른다.
→ 이것은 Restricted Boltzmann Machine(RBM)이라 한다.

RBM으로 설정된 Weight에 Labeled된 x 를 넣어서 출력 값(정답)이 나오도록 Weight를 미세 조정(fine-tuning)한다.

Deep Learning ?

<2010년대 Deep Learning이 성공적일 수 있었던 3가지 요소>

1. The amount of available information to train the networks
2. The amount of compute capacity available
3. The evolution of Algorithmic techniques

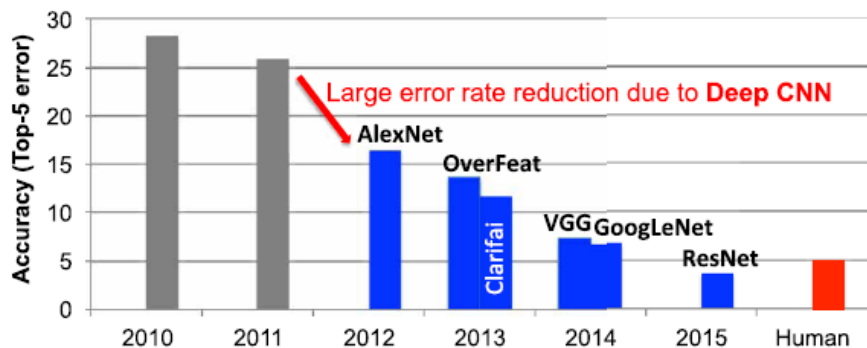


Fig. 7. Results from the ImageNet Challenge [14].

2012년에 AlexNet이 ImageNet Challenge에서 error rate를 급격히 감소시킬 수 있던 이유
→ GPUs 기반으로 높은 계산 능력을 가진 DNN

2015년에 ResNet은 인간보다 뛰어난 정확도를 보였다.



*Deep Learning*의 활용

- Image and Video
 - Speech and Language
 - Medical DNNs
 - Game play
 - Robot DNNs
 - etc
- 

“상황에 따라” 클라우드 또는 임베디드에서 DNN 수행, 각각의 장단점은?

- 임베디드(Embedded)

장점 : 의미 있는 정보를 빠르게 추출 가능, 사생활 보호

Latency나 security risk를 고려했을 때 임베디드 방식이 사용 ex) 자율주행, Apple Siri...etc

단점 : severe energy consumption, compute and memory cost limitations

- 클라우드(Cloud)

장점 : 대부분의 경우에서 많은 양의 데이터를 DNN model로 학습시키는 것은 많은 시간이 필요하다. → 클라우드에서 주로 수행 (하지만 Interference는 클라우드 또는 device에서 모두 가능할 수 있다.)

또, 임베디드의 단점 커버

단점 : Latency, Privacy, Security ... etc

<선형 회귀(Linear Regression) : 가장 좋은 예측선 찾기>

1. 선형 회귀(Linear Regression)이란?
2. 최소 제곱법(Method of Least Squares)
3. 평균 제곱 오차(Mean Square Error, MSE)
4. 오차 수정하는 방법 : 경사 하강법(Gradient Descent)
5. 다중 선형 회귀(Multiple Linear Regression)

다음 발표에서

딥러닝을 위한 수학 간단하게

1. 선형 회귀(Linear Regression)이란?

용훈이의 기분은 ()에 따라 다르다.

() : 정보, information

()에 따라 용훈이의 기분이 변한다.

→ () = x , 용훈이의 기분 = y

→ x 값이 변함에 따라 y 값이 변한다. → x : 독립변수 , y : 종속변수

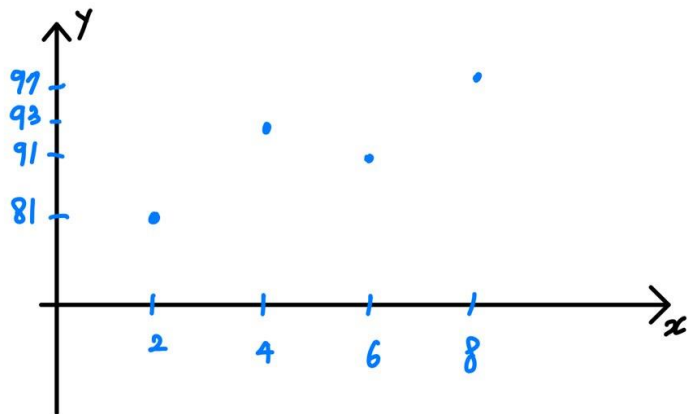
→ 선형 회귀란 독립 변수 x 를 사용해 종속 변수 y 의 움직임을 예측하고 설명하는 일

→ x 값이 여러 개이면(x_1, x_2, x_3, \dots) 이면 다중 선형 회귀(multiple linear regression)이라 한다.

딥러닝을 위한 수학 간단하게

1. 선형 회귀(Linear Regression)이란? : 정확한 직선을 그려내는 과정

x	2	4	6	8
y	81	93	91	97



정확한 직선 $y = ax + b$ 를 그려 내야 한다.

→ 최적의 a 와 b 를 찾는 작업

→ 정확한 a 와 b 의 값을 따라 움직이는 직선에 x 값을 대입하면 예측 y 를 구할 수 있다.

딥러닝을 위한 수학 간단하게

2. 최소 제곱법(Method of Least Squares)

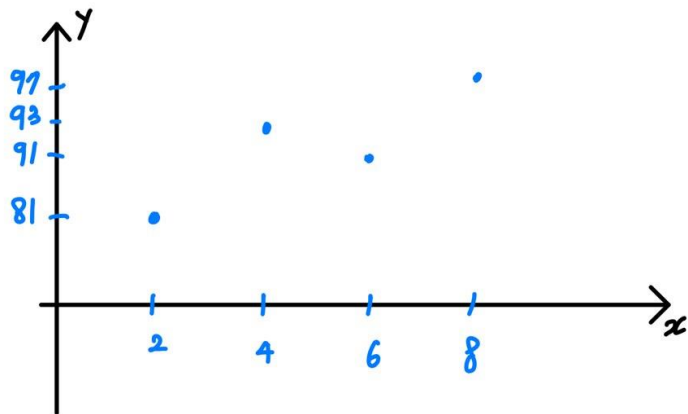
x	2	4	6	8
y	81	93	91	97

정확한 직선 $y = ax + b$ 를 그려 내야 한다. → 최적의 a 와 b 를 찾는 작업

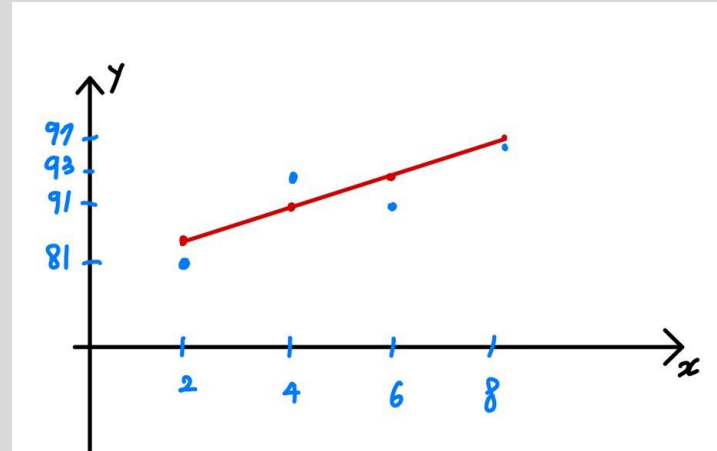
$$a = \frac{\sum(x-x.\text{mean})*(y-y.\text{mean})}{\sum(x-x.\text{mean})^2}, \quad b = y.\text{mean} - x.\text{mean} * a$$

→ $x.\text{mean} = x$ 의 평균 = 5, $y.\text{mean} = y$ 의 평균 = 90.5

→ $a=2.3$, $b=79$ → $y = 2.3x + 79$




x	2	4	6	8
y	81	93	91	97
$y=2.3x+79$	83.6	88.2	92.8	97.4



딥러닝을 위한 수학 간단하게

2. 최소 제곱법(Method of Least Squares)

```
jupyter Untitled1 Last Checkpoint: 11시간 전 (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [6]: import numpy as np

x=[2, 4, 6,8];
y=[81, 93, 91, 97];

mx=np.mean(x)
my=np.mean(y)
print("x의 평균값 : ",mx)
print("y의 평균값 : ",my)

divisor=sum([(i-mx)**2 for i in x])

def top(x,mx,y,my):
    d=0
    for i in range(len(x)):
        d+=(x[i]-mx)*(y[i]-my) # (x-x.mean)*(y-y.mean)의 합
    return d
dividend=top(x,mx,y,my)

print("분모 : ",divisor)
print("분자 : ", dividend)
a=dividend/divisor
b=my-(mx*a)

print("기울기 a=",a)
print("y절편 b =",b)

x의 평균값 : 5.0
y의 평균값 : 90.5
분모 : 20.0
분자 : 46.0
기울기 a= 2.3
y절편 b = 79.0
```

딥러닝을 위한 수학 간단하게

최소 제곱법으로 가장 최고의 직선을 구할 수 있었다.

최소 제곱법의 한계점 : 여러 개의 입력을 처리하지 못한다.

실제로 딥러닝은 대부분 입력 값이 여러 개인 상황에서 이를 해결하기 위해 실행된다.

→ 평균 제곱 오차(Mean Square Error, MSE)

딥러닝을 위한 수학 간단하게

3. 평균 제곱 오차(Mean Square Error, MSE) → ex) $y' = 3x + 79$

x	2	4	6	8
y	81	93	91	97
$y' = 3x + 79$	82	88	94	100
$y' - y$	1	-5	3	3

오차 = 예측 값(y') - 실제 값(y)

→ 양수와 음수가 섞여서 오차의 합=0이 될 수 있다.

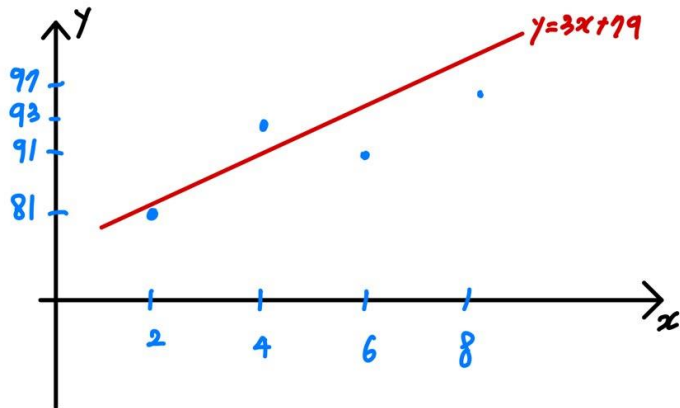
→ 따라서 오차 = $(y' - y)^2$

→ 오차의 합 = $\sum (y' - y)^2$

→ 평균 제곱 오차(MSE) = $\frac{1}{n} * \sum (y' - y)^2$

→ $MSE = \frac{1+25+9+9}{4} = \frac{44}{4} = 11$

→ 선형 회귀는 임의의 직선을 그어 MSE를 구하고 MSE가 최소가 되는 a 와 b 를 찾는 작업



딥러닝을 위한 수학 간단하게

3. 평균 제곱 오차(Mean Square Error, MSE) → ex) $y' = 3x + 79$

```
jupyter Untitled1 Last Checkpoint: 지난주 토요일 오후 3:24 (unsaved changes) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [19]: -*- coding: utf-8 -*-
import numpy as np

fake_a_b=[3,76]

data = [[2, 81], [4,93], [6,91], [8,97]]
x = [i[0] for i in data]
y = [i[1] for i in data]

def predict(x):
    return fake_a_b[0]*x+fake_a_b[1]
def mse(y_hat,y):
    return ((y_hat-y)**2).mean()

def mse_val(predict_result,y):
    return mse(np.array(predict_result),np.array(y))

predict_result=[]

for i in range(len(x)):
    predict_result.append(predict(x[i]))
    print("공부한 시간 = %.f , 실제 점수 = %.f, 예측 점수 = %.f" %(x[i],y[i],predict(x[i])))

print("mse 최종값: "+str(mse_val(predict_result,y)))

공부한 시간 = 2 , 실제 점수 = 81, 예측 점수 = 82
공부한 시간 = 4 , 실제 점수 = 93, 예측 점수 = 88
공부한 시간 = 6 , 실제 점수 = 91, 예측 점수 = 94
공부한 시간 = 8 , 실제 점수 = 97, 예측 점수 = 100
mse 최종값: 11.0
```

Deep Learning 키워드

End – to – end, Feature, hand-crafted, ... etc

Showing 1-25 of 171,486 for **End to end** ✕

☐ Conferences (130,123) ☐ Journals (34,355) ☐ Magazines (3,763) ☐ Early Access Articles (1,581)
☐ Books (927) ☐ Standards (494) ☐ Courses (243)

Show

☒ All Results
☐ My Subscribed Content
☐ Open Access

Year

Single Year Range

From 1905 To 2021

Author

☐ Select All on Page Sort By: Relevance Standards Dictionary Terms

- ☐ **An End-to-End Performance Analysis for Service Chaining in a Virtualized Network**
Emmanouil Fountoulakis ; Qi Liao ; Nikolaos Pappas
IEEE Open Journal of the Communications Society
Year: 2020 | Volume: 1 | Journal Article | Publisher: IEEE
▶ Abstract [\(html\)](#) [PDF](#) (2731 Kb) [CC](#)
- ☐ **End-to-end energy efficient communication**
Lars Dittmann
IET International Conference on Communication Technology and Application (ICCTA 2011)
Year: 2011 | Conference Paper | Publisher: IET
▶ Abstract [PDF](#) (227 Kb) [CC](#)
- ☐ **A Study on Comparative Analysis of End-to-End Routing and Opportunistic Routing**
Chuta Minamiguchi ; Natsuko Kawabata ; Ryo Nakamura ; Hiroyuki Ohsaki
2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)
Year: 2018 | Volume: 01 | Conference Paper | Publisher: IEEE
Cited by: Papers (1)
▶ Abstract [\(html\)](#) [PDF](#) (130 Kb) [CC](#)

link
packet
channel
MAC
port
ambient temperature
agent
bridge
path
symbol
system
frame
repeater

Showing 1-25 of 438,728 for **Feature** ✕

☐ Conferences (342,798) ☐ Journals (82,263) ☐ Magazines (7,434) ☐ Early Access Articles (1,581)
☐ Books (1,356) ☐ Standards (587) ☐ Courses (185)

Show

☒ All Results
☐ My Subscribed Content
☐ Open Access

Year

Single Year Range

From 1890 To 2021

Author

☐ Select All on Page Sort By: Relevance

- ☐ **Extracting Audio-Visual Features for Emotion Recognition Through Active Feature Selection**
Fasih Haider ; Senja Pollak ; Pierre Albert ; Saturnino Luz
2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)
Year: 2019 | Conference Paper | Publisher: IEEE
▶ Abstract [PDF](#) (838 Kb) [CC](#)
- ☐ **Mass Classification in Mammograms Using Selected Geometry and Texture Features, and a New SVM-Based Feature Selection Method**
Xiaoming Liu ; Jinshan Tang
IEEE Systems Journal
Year: 2014 | Volume: 8, Issue: 3 | Journal Article | Publisher: IEEE
Cited by: Papers (79)
▶ Abstract [\(html\)](#) [PDF](#) (436 Kb) [CC](#)
- ☐ **A combined feature representation of deep feature and hand-crafted features for person re-identification**
Youjiao Li ; Li Zhuo ; Xiaochen Hu ; Jing Zhang
2016 International Conference on Business Information and Communication Systems (ICBIS)

Showing 1-25 of 1,632 for **hand-crafted** ✕

☐ Conferences (1,218) ☐ Journals (352) ☐ Early Access Articles (54) ☐ Books (4)
☐ Magazines (4)

Show

☒ All Results
☐ My Subscribed Content
☐ Open Access

Year

Single Year Range

From 1985 To 2021

Author

☐ Select All on Page Sort By: Relevance

- ☐ **Agent's activity recognition: a focus on comparison of automatically-learned and hand-crafted features**
Yan Wang ; Yuguo Chen ; Hongnian Yu
2019 International Conference on Advanced Mechatronic Systems (ICAMechS)
Year: 2019 | Conference Paper | Publisher: IEEE
▶ Abstract [PDF](#) (150 Kb) [CC](#)
- ☐ **Deep Learned vs. Hand-Crafted Features for Action Classification**
Pablo Andres Millan Arias ; Julian Armando Quiroga Sepulveda
2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)
Year: 2018 | Conference Paper | Publisher: IEEE
▶ Abstract [\(html\)](#) [PDF](#) (100 Kb) [CC](#)
- ☐ **Combining CNN with Hand-Crafted Features for Image Classification**
Zhou Tianyu ; Miao Zhenjiang ; Zhang Jianhu
2018 14th IEEE International Conference on Signal Processing (ICSP)

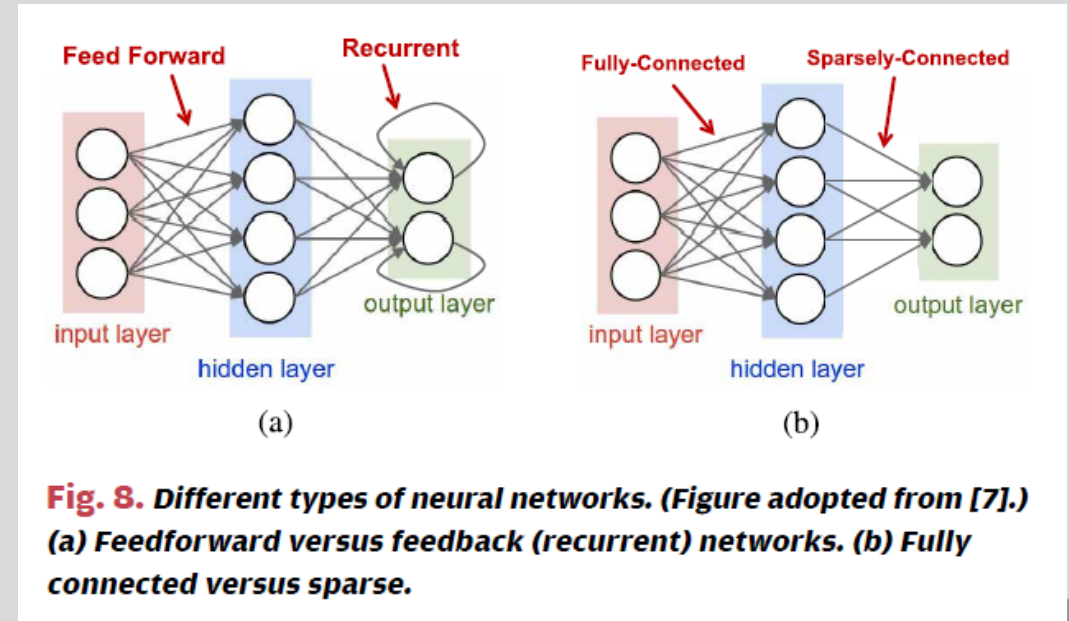
Deep Learning

DNNs는 상황에 따라 다양한 모양과 사이즈로 만들 수 있다.

크게 Feedforward and Recurrent(feedback) 방식과
Fully connected(FC) + Sparsely connected 2가지 방식이 있다.

Feedforward : 앞으로의 징후를 계산에 의해 예측하고
그 정보에 기준하여 제어를 행하는 방식 → 자동제어
Feedback : 목적에서 이탈될 때 소기의 목적에 맞도록
제어하는 방식


신기하게도 많은 사례에서 Sparsely connected가 정확
도를 더 높인다는 사례가 있다. (Weight를 0으로)





Deep Learning 모델

일반적인 Deep Learning 모델 3개

1. Convolutional Neural Networks(CNNs)
 2. Recurrent Neural Networks(RNNs)
 3. Generative Adversarial Networks(GANs)
- 

Deep Learning 모델

1. Convolutional Neural Networks(CNNs)

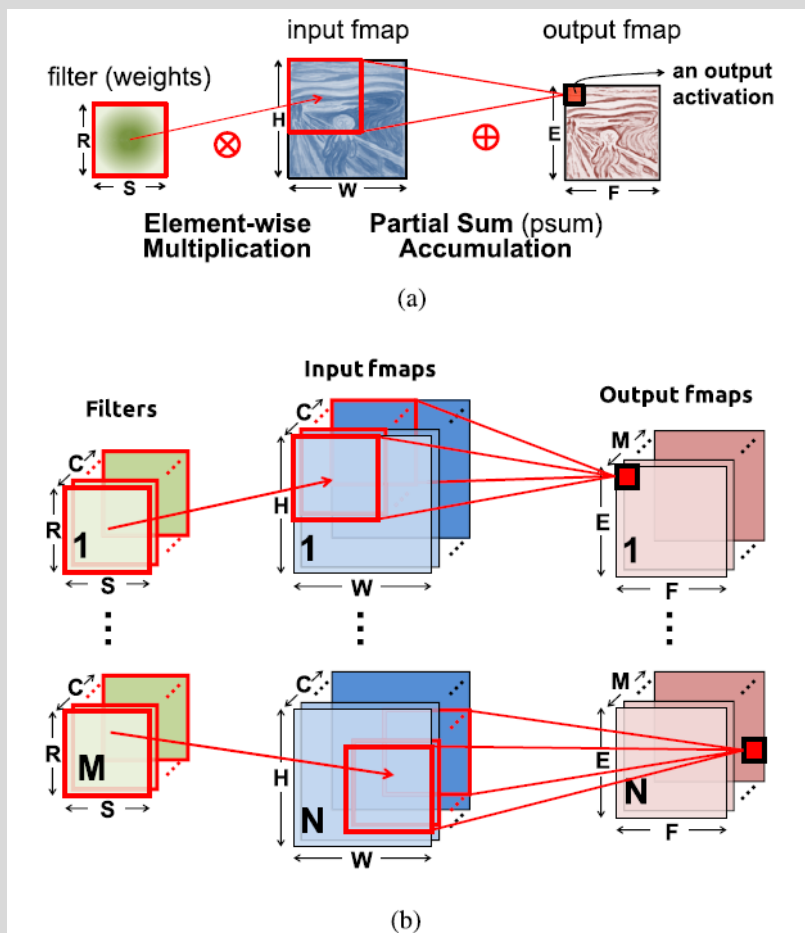


Fig. 9. Dimensionality of convolutions. (a) 2-D convolution in traditional image processing. (b) High dimensional convolutions in CNNs.

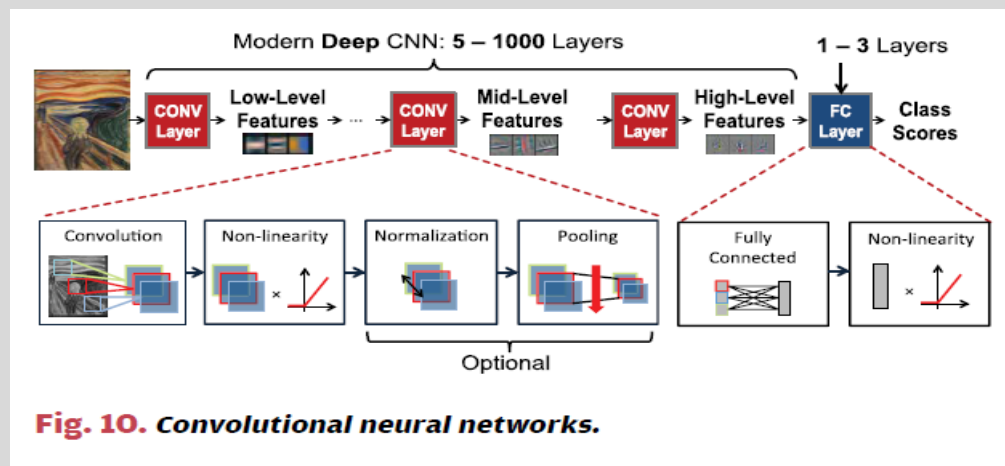


Fig. 10. Convolutional neural networks.

2x2 pooling, stride 2

9	3	5	3
10	32	2	2
1	3	21	9
2	6	11	7

Max pooling

32	5
6	21

Average pooling

18	3
3	12

Fig. 12. Various forms of pooling. (Figure adopted from Caffe Tutorial [46].)

2. Recurrent Neural Networks(RNNs)

가변적인 크기의 Sequence 데이터를 modeling 하기 위해 등장

기존의 NNs과의 차이는 "memory"

→ 새로운 입력이 들어올 때마다 자신의 기억을 조금씩 수정

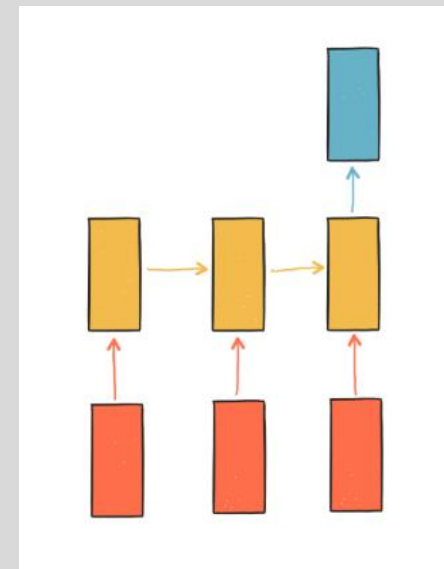
Ex) 자동번역, 이미지 캡션

Q. Sequence 데이터는 무조건 RNN을 써야하나?

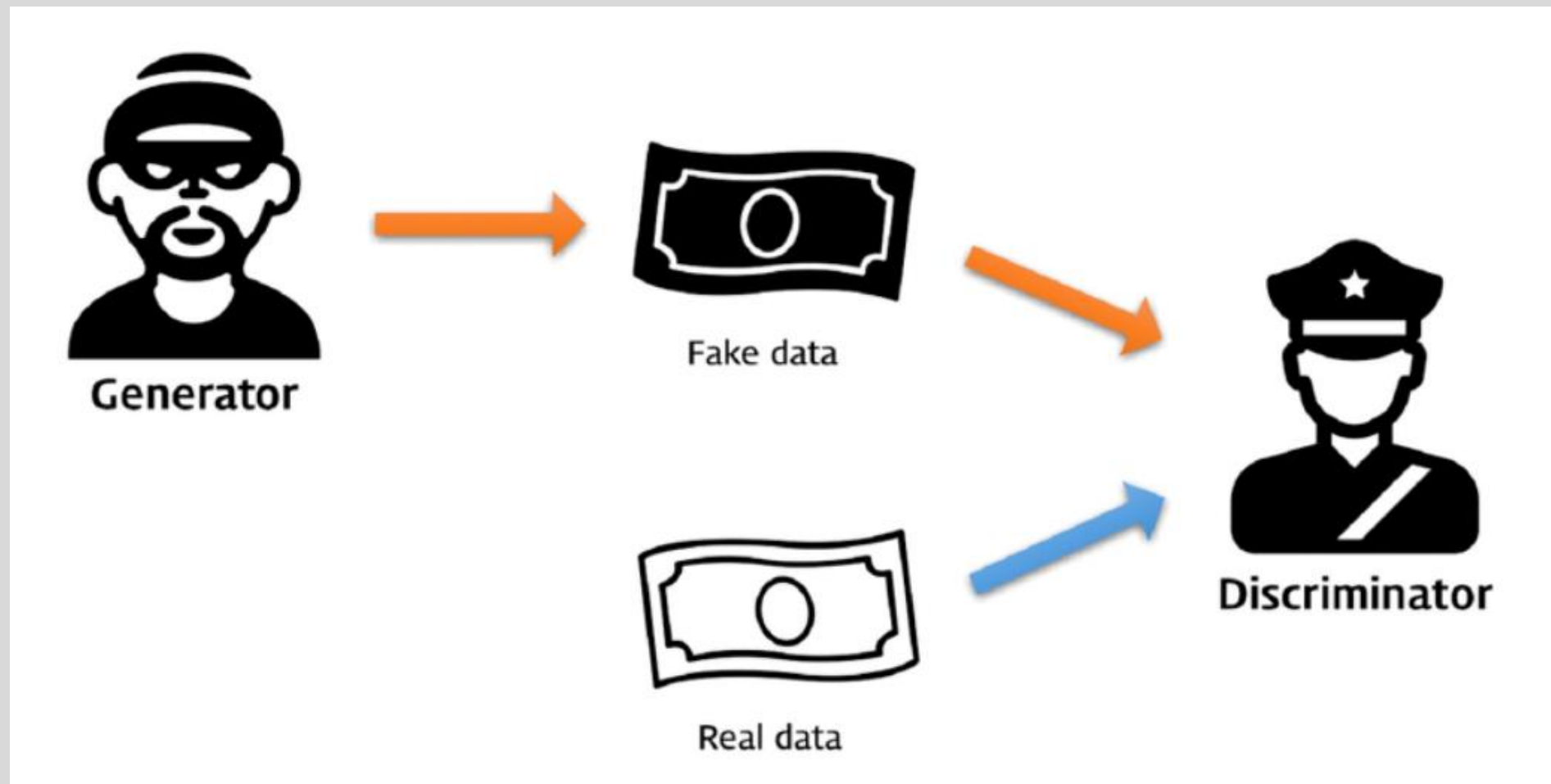
A. 아니다. Sequence 데이터의 Classification/Regression을 풀 때 CNN을 사용해도 된다.
심지어 CNN의 성능이 월등히 좋을 때도 많이 있다.

그럼 왜 RNN?

→ RNN은 Ultimate Deep learning Model이라 불린다. CNN은 지난 데이터를 추가적으로 고려할 필요가 없는데, RNN은 함께 고려된다.



3. Generative Adversarial Networks(GANs)



잘못된 학습

"One of the major drawbacks of DNNs is their need for large data sets to prevent overfitting during training."

Overfitting(과적합)이란?

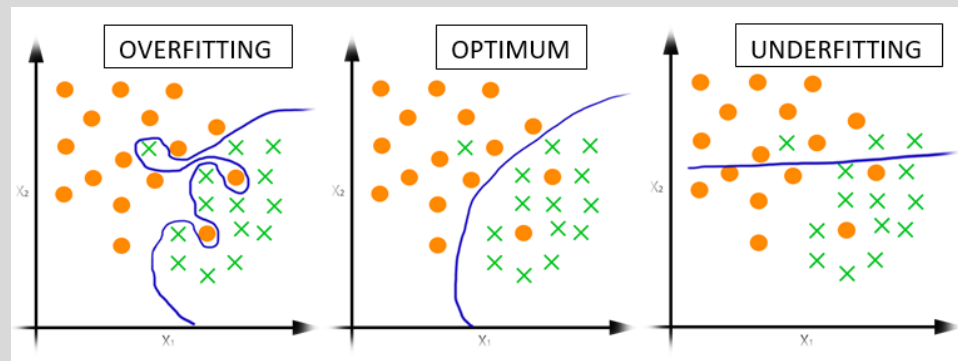
학습된 데이터에 대한 오차는 줄어도 실제 사례에 적용할 경우 오차가 많이 증가한다.

→ Ex) 사람 : 주입식 교육 → 응용력 부족

→ 원인 : Layer가 너무 많거나 변수가 복잡하거나 Test Set과 학습 Set이 중복될 경우 등등..

→ 특히 딥러닝은 학습 단계에서 Input, Hidden, Output Layers의 노드들에 상당히 많은 변수들이 투입됨

: Overfitting 항상 주의





참고 문헌

1. Efficient Processing of Deep Neural Networks : A Tutorial and Survey, by VIVIENNE SZE, Senior Member IEEE...
 2. RhythmNet : End – to – end Heart Rate Estimation From Face via Spatial-Temporal Representation by Xuesong Niu, Shiguang Shan
 3. 그 외 구글로부터
- 