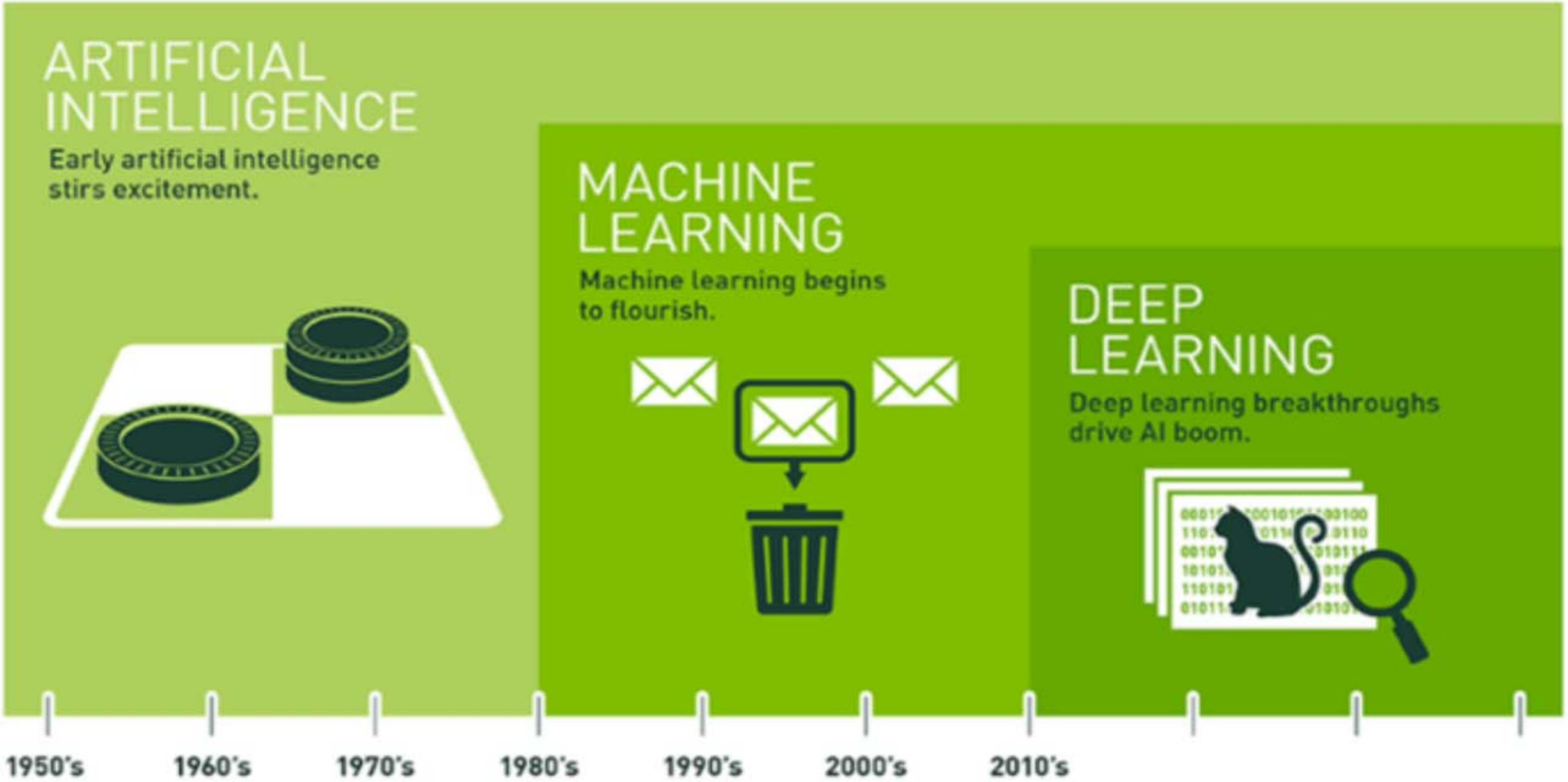
딥러닝(Deep Learning) 이론

(주)딥비전

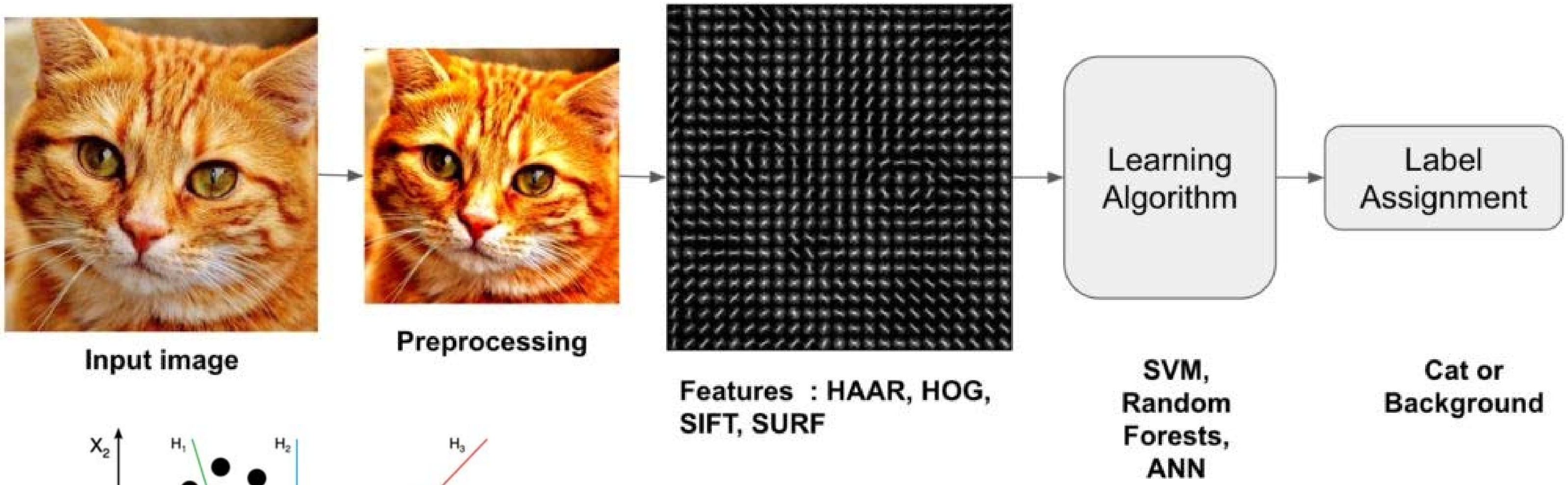
인공지능 > 머신러닝 > 딥러닝

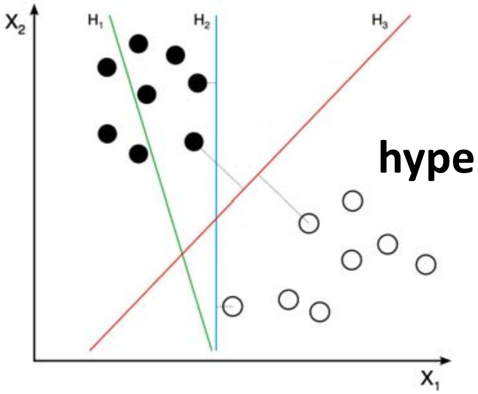


https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Machine Learning

Feature Extraction -> Train -> Test

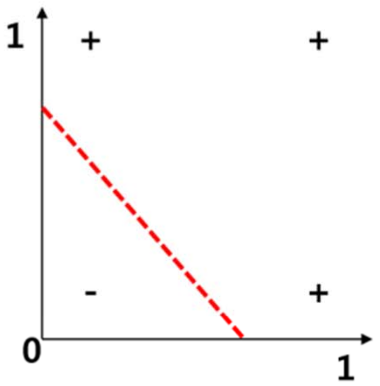




hyperplane

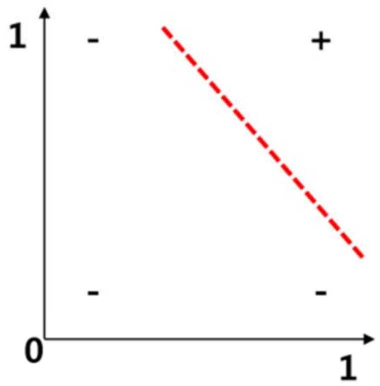
Neural Network - XOR

OR



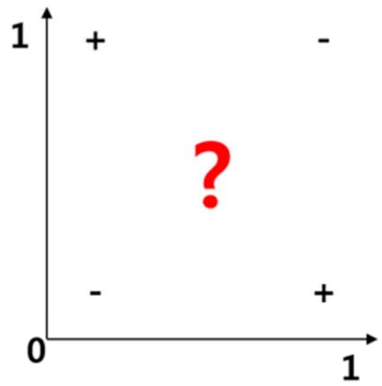
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

AND



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

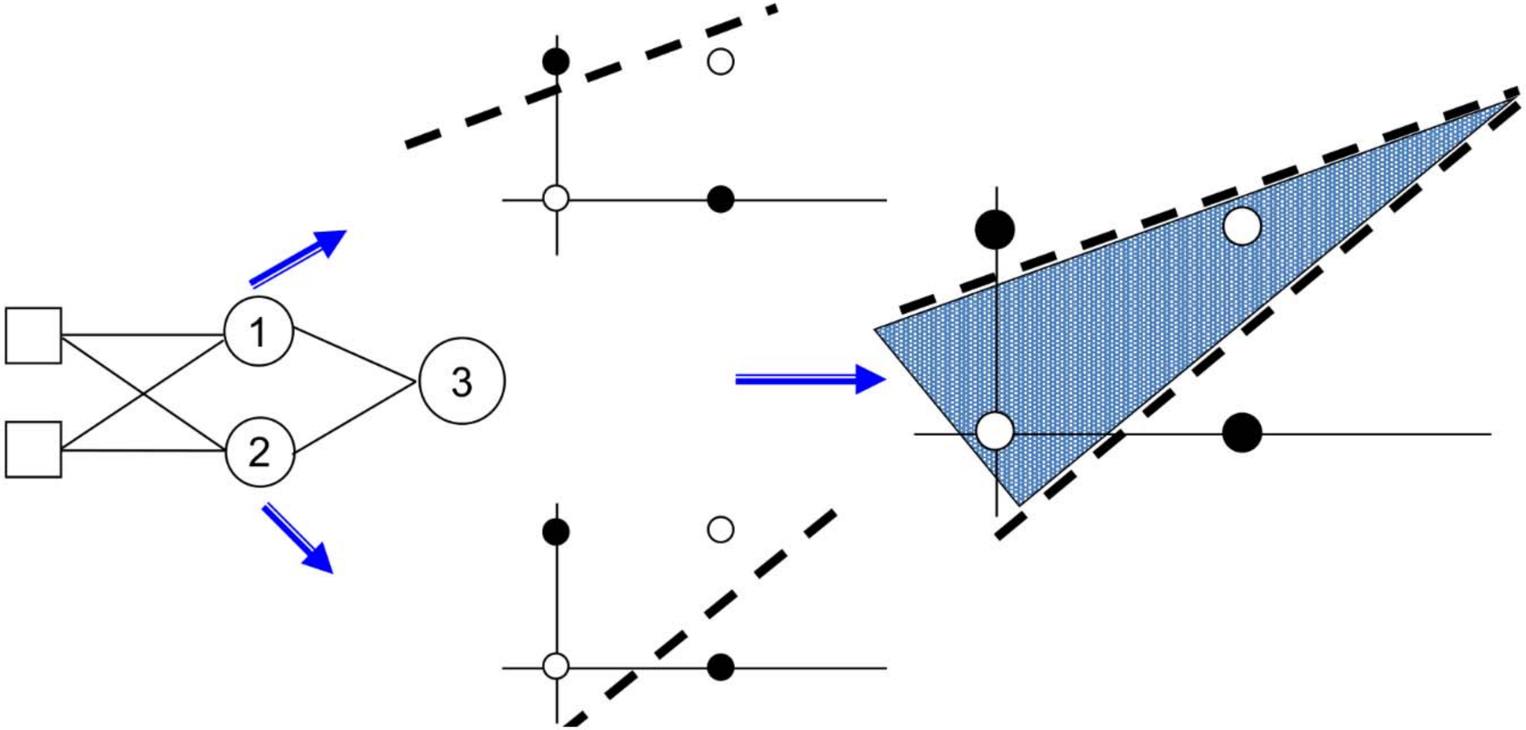
XOR



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

http://www.birc.co.kr/2018/01/22/xor-%EB%AC%B8%EC%A0%9C%EC%99%80-neural-network/

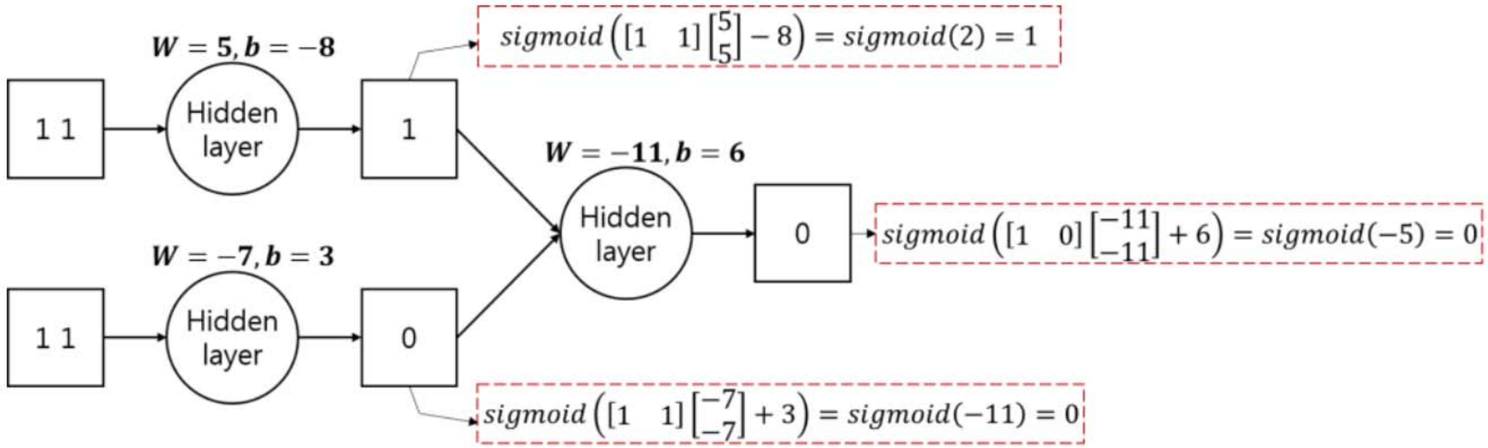
Neural Network - XOR



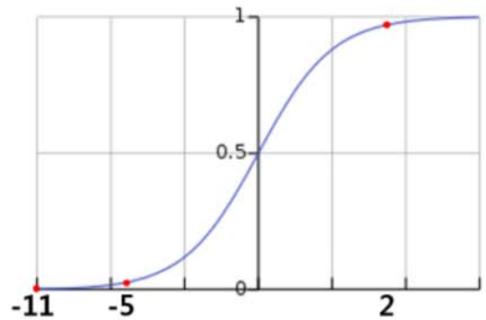
https://slidewiki.org/deck/1099-1/slide/9024-2/9024-2:7/view

2. 머신러닝(Machine Learning)

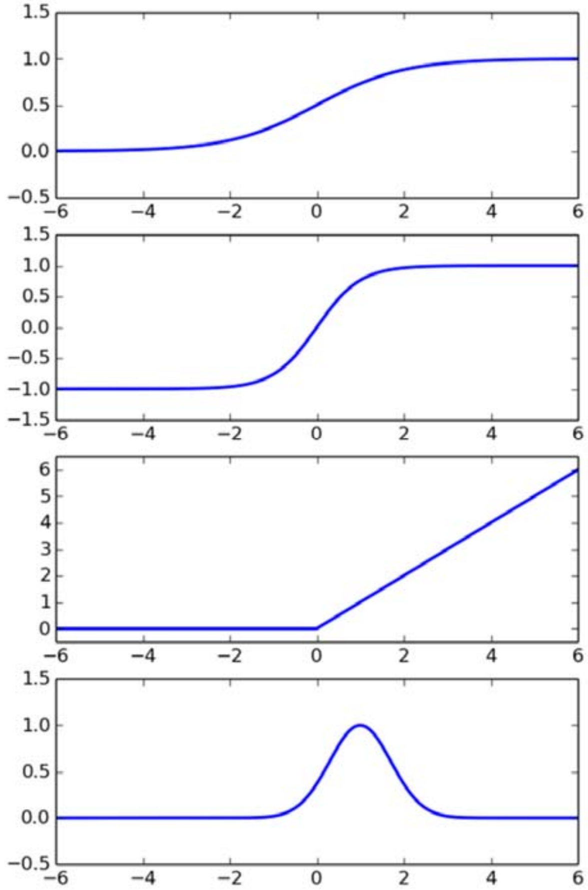
Neural Network - XOR



x_1	x_2	y_1	y_2	$\hat{y}$	XOR
0	0	0	1	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	0	0	0



Neural Network - activation function



Sigmoid

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

Hyperbolic Tangent

$$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Rectified Linear

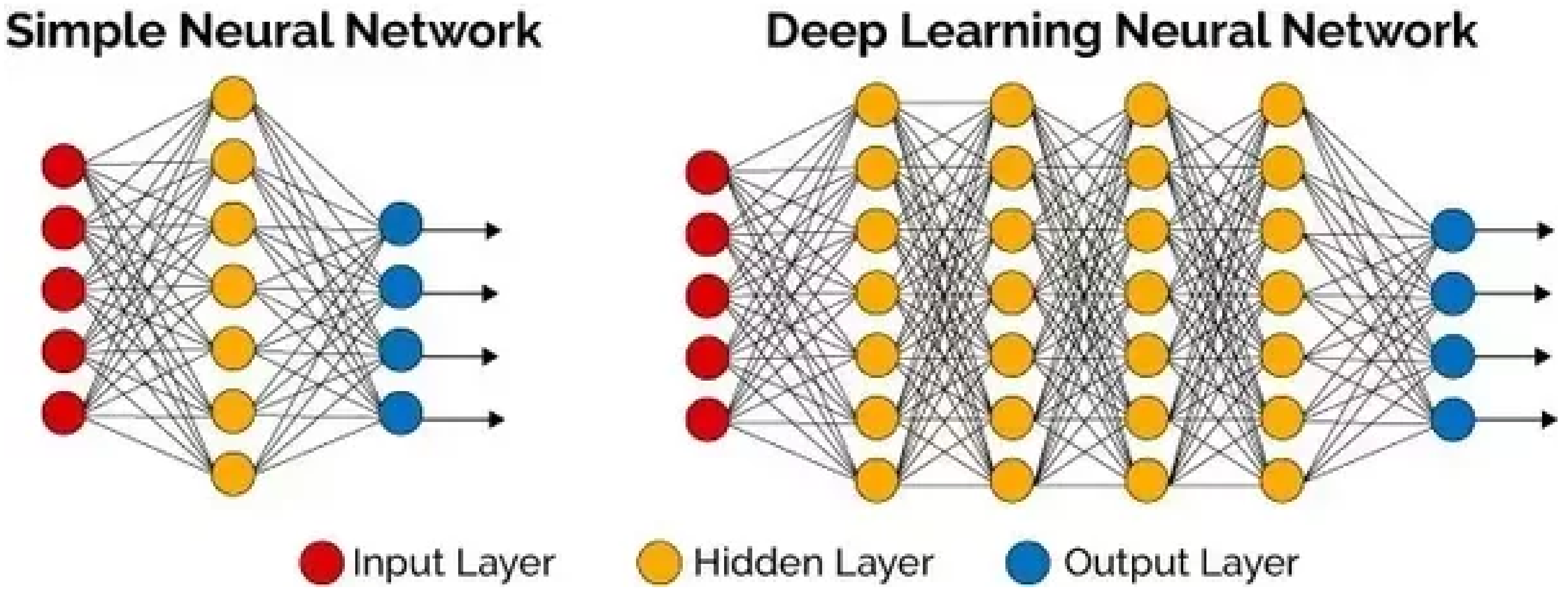
$$\phi(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$

Radial Basis Function

$$\phi(z, c) = e^{-(\epsilon \| z - c \|)^2}$$

https://www.researchgate.net/figure/Common-neural-network-activation-functions_fig7_305881131

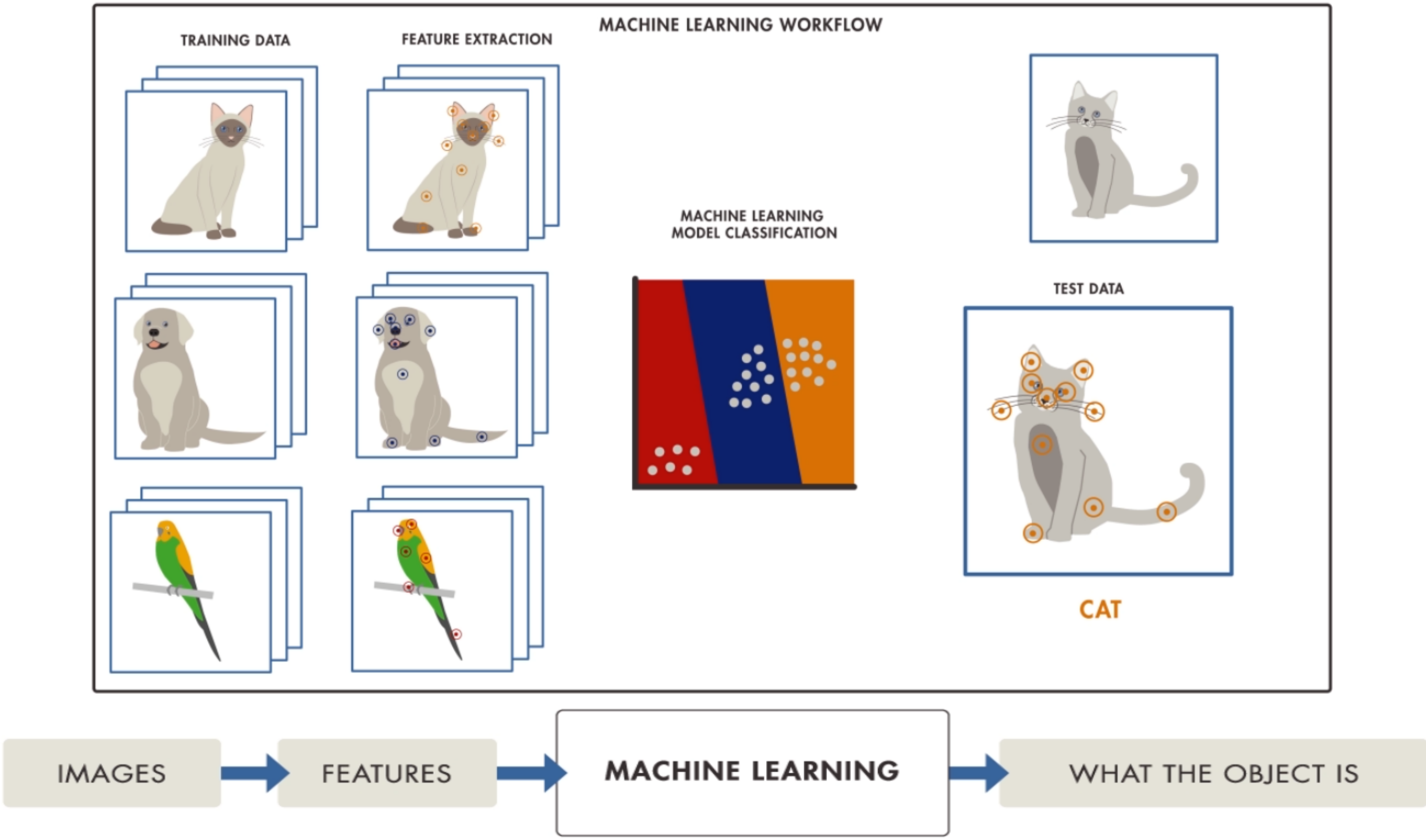
Neural Network vs Deep Neural Network



https://www.quora.com/What-is-the-difference-between-Neural-Networks-and-Deep-Learning

3. 딥러닝(Deep Learning)

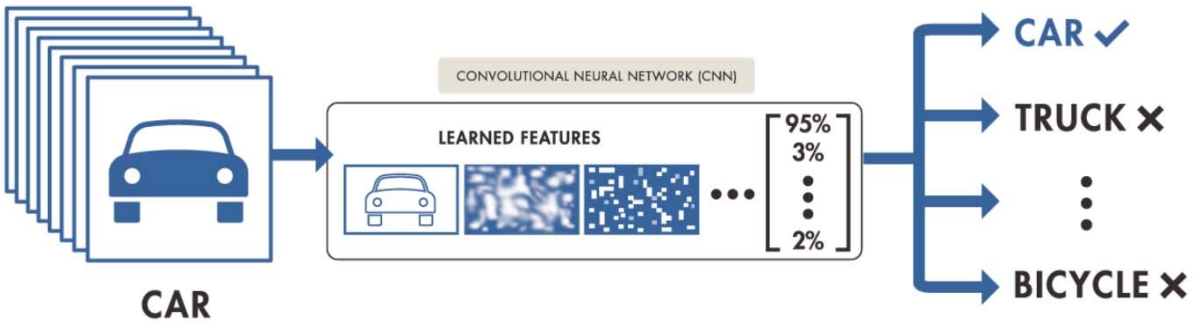
머신러닝과 딥러닝 비교

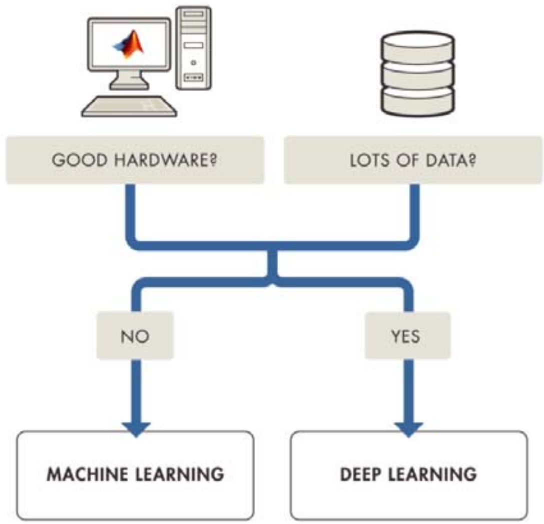


https://kr.mathworks.com/videos/introduction-to-deep-learning-machine-learning-vs-deep-learning-1489503513018.html?s_tid=srchtitle

3. 딥러닝(Deep Learning)

머신러닝과 딥러닝 비교

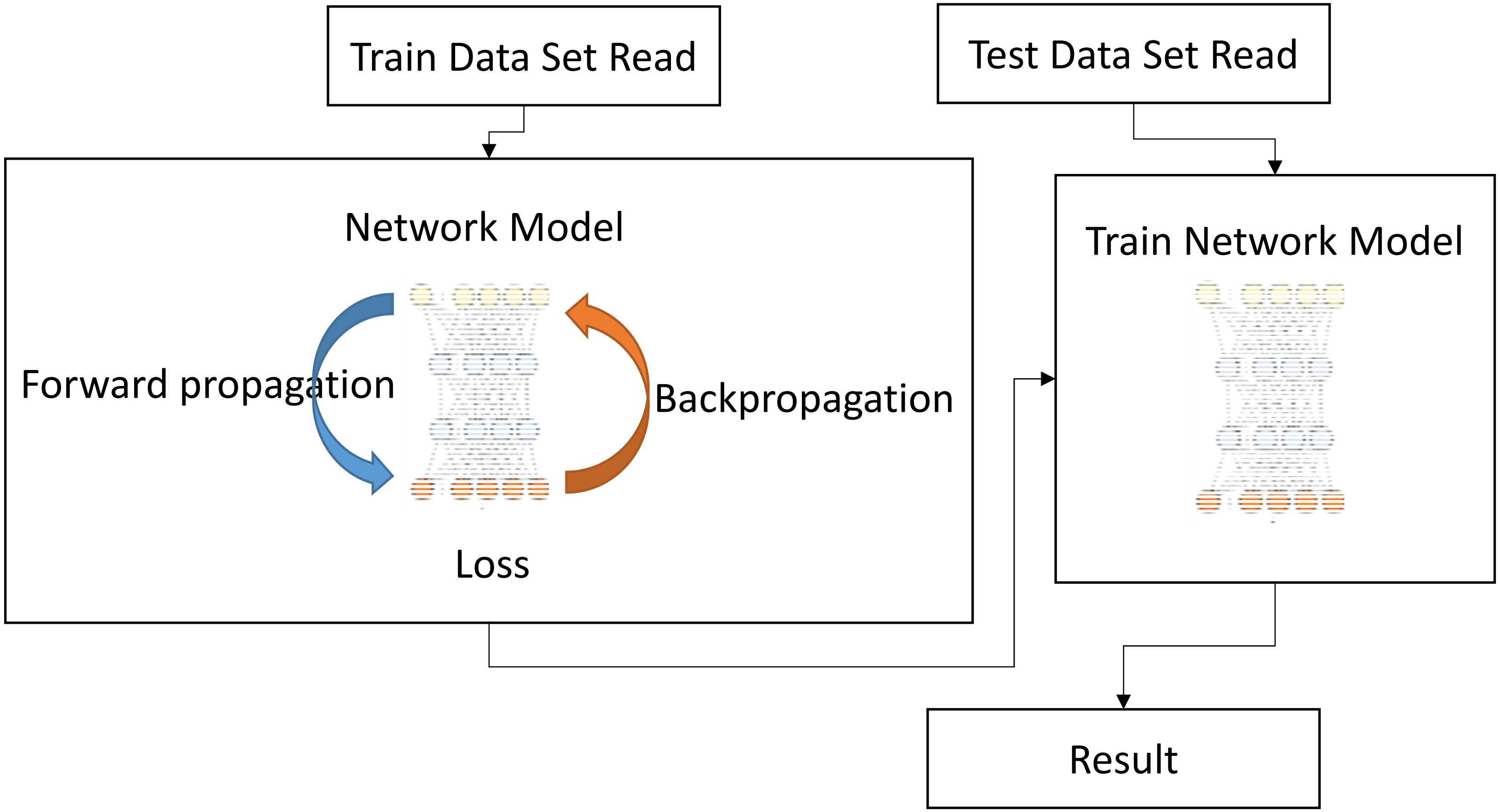




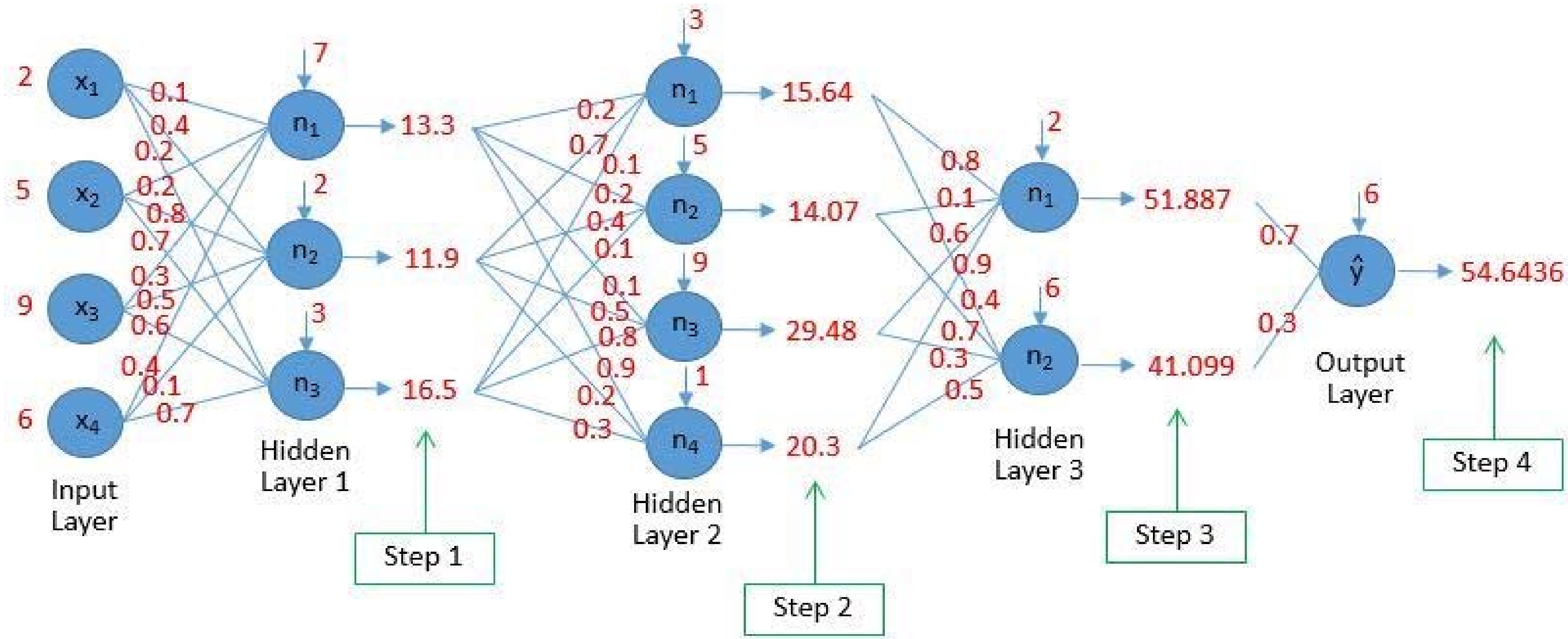
	Machine Learning	Deep Learning
Training dataset	Small	Large
Choose your own features	Yes	No
# of classifiers available	Many	Few
Training time	Short	Long

https://kr.mathworks.com/videos/introduction-to-deep-learning-machine-learning-vs-deep-learning-1489503513018.html?s_tid=srchtitle

딥러닝 학습 절차


Train Data Set Read
Test Data Set Read
Network Model
Forward propagation
Backpropagation
Loss
Train Network Model
Result

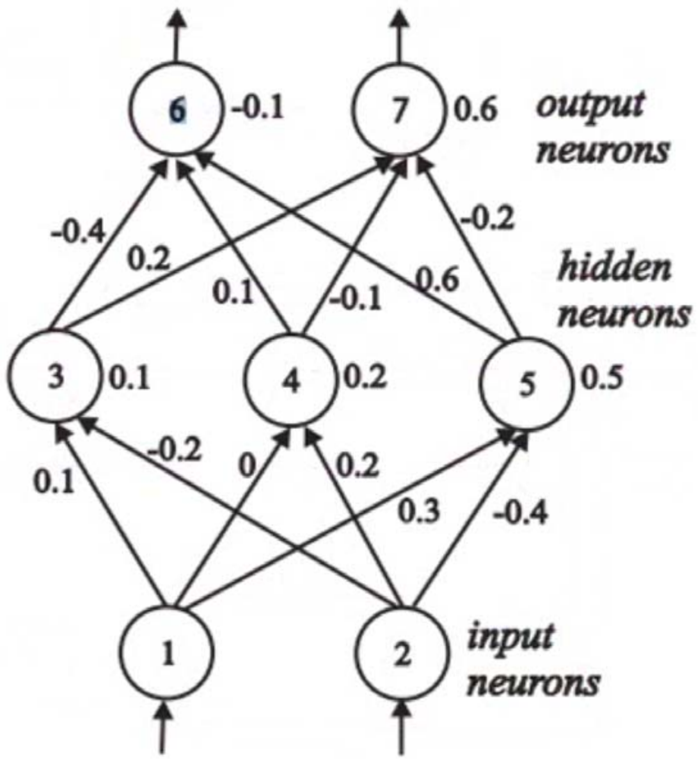
Forward Propagation



https://dwbi1.wordpress.com/2018/02/07/building-a-neural-network/

Backpropagation

- 2 classes, 2 dim. input data
  - training set:
    ex.1: 0.6 0.1 | class 1 (banana)
    ex.2: 0.2 0.3 | class 2 (orange)
    …
- Network architecture
  - How many inputs?
  - How many hidden neurons?
    - Heuristic:
      n=(inputs+output_neurons)/2
  - How many output neurons?
  - What encoding of the outputs?
    - 10 for class 1, 01 for class 0
- Initial weights and learning rate
  - Let's η=0.1 and the weights are set as in the picture



https://www.youtube.com/watch?v=uygTXSx-yEE

Backpropagation

1. Forward pass for ex. 1 - calculate the outputs o_6 and o_7

o_1=0.6, o_2=0.1, target output 1 0, i.e. class 1

- Activations of the hidden units:

$net_3= o_1 *w_{13}+ o_2*w_{23}+b_3=0.6*0.1+0.1*(-0.2)+0.1=0.14$

$o_3=1/(1+e^{-net3}) =0.53$

$net_4= o_1 *w_{14}+ o_2*w_{24}+b_4=0.6*0+0.1*0.2+0.2=0.22$

$o_4=1/(1+e^{-net4}) =0.55$

$net_5= o_1 *w_{15}+ o_2*w_{25}+b_5=0.6*0.3+0.1*(-0.4)+0.5=0.64$

$o_5=1/(1+e^{-net5}) =0.65$

- Activations of the output units:

$net_6= o_3 *w_{36}+ o_4*w_{46}+ o_5*w_{56} +b_6=0.53*(-0.4)+0.55*0.1+0.65*0.6-0.1=0.13$

$o_6=1/(1+e^{-net6}) =0.53$

$net_7= o_3 *w_{37}+ o_4*w_{47}+ o_5*w_{57} +b_7=0.53*0.2+0.55*(-0.1)+0.65*(-0.2)+0.6=0.52$

$o_7=1/(1+e^{-net7}) =0.63$

https://www.youtube.com/watch?v=uygTXSx-yEE

Backpropagation

2. Backward pass for ex. 1

- Calculate the output errors δ_6 and δ_7 (note that d_6=1, d_7=0 for class 1)

$\delta_6 = (d_6 - o_6) * o_6 * (1-o_6)=(1-0.53)*0.53*(1-0.53)=0.12$

$\delta_7 = (d_7 - o_7) * o_7 * (1-o_7)=(0-0.63)*0.63*(1-0.63)=-0.15$

- Calculate the new weights between the hidden and output units (η=0.1)

$\Delta w_{36} = \eta * \delta_6 * o_3 = 0.1*0.12*0.53=0.006$

$w_{36}^{new} = w_{36}^{old} + \Delta w_{36} = -0.4+0.006=-0.394$

$\Delta w_{37} = \eta * \delta_7 * o_3 = 0.1*-0.15*0.53=-0.008$

$w_{37}^{new} = w_{37}^{old} + \Delta w_{37} = 0.2-0.008=-0.19$

Similarly for w_{46}^{new}, w_{47}^{new}, w_{56}^{new} and w_{57}^{new}

For the biases b_6 and b_7 (remember: biases are weights with input 1):

$\Delta b_6 = \eta * \delta_6 * 1 = 0.1*0.12=0.012$

$b_6^{new} = b_6^{old} + \Delta b_6 = -0.1+0.012=-0.012$

https://www.youtube.com/watch?v=uygTXSx-yEE

Backpropagation

- Calculate the errors of the hidden units δ_3, δ_4 and δ_5

$\delta_3 = o_3 * (1-o_3) * (w_{36} * \delta_6 + w37 * \delta_7) =$

$= 0.53*(1-0.53)(-0.4*0.12+0.2*(-0.15))=-0.019$

Similarly for δ_4 and δ_5

- Calculate the new weights between the input and hidden units ($\eta=0.1$)

$\Delta w_{13} = \eta * \delta_3 * o_1 = 0.1*(-0.019)*0.6=-0.0011$

$w_{13}^{new} = w_{13}^{old} + \Delta w_{13} = 0.1-0.0011=0.0989$

Similarly for w_{23}^{new} , w_{14}^{new} , w_{24}^{new} , w_{15}^{new} and w_{25}^{new}; b_3, b_4 and b_6

https://www.youtube.com/watch?v=uygTXSx-yEE

3. 딥러닝(Deep Learning) - 프로그래밍 언어, 프레임워크, 네트워크

프로그래밍 언어

C/C++
- 범용 컴파일 프로그래밍 언어
- 다양한 라이브러리 포함

Python
- 범용 인터프리터 프로그래밍 언어
- Numpy, Scipy 등 과학계산 및 머신러닝을 위한 패키지가 발전됨

Matlab
- 과학 계산용 프로그래밍 언어(Mathworks사에서 개발)

R
- 통계 및 그래프용 프로그래밍 언어(뉴질랜드 오클랜드 대학교 개발)

기타
- Java, Lua, Go, Scala

프레임워크

TensorFlow
Caffe
Keras
Torch
Theano

Deeplearning4j
MxNet
Microsoft Cognitive Toolkit (CNTK)
Lasagne
BigDL

프레임워크

- TensorFlow
가장 인기있는 딥러닝 라이브러리 중 하나
Google Brain 팀에서 개발했으며 2015년 오픈소스로 공개
Python 기반 라이브러리, CPU 및 GPU와 모든 플랫폼, 데스크톱 및 모바일에서 사용 가능
C++ 및 R과 같은 다른 언어 지원
딥러닝 모델을 직접 작성, Keras 라이브러리를 사용하여 직접 작성 가능

- Caffe
최초의 딥러닝 라이브러리 중 하나, 표현, 속도 및 모듈성을 염두에 두고 개발
Python 인터페이스를 가지고 있는 C++ 라이브러리
CNN(Convolutional Neural Networks)을 모델링 할 때 기본 애플리케이션 사용
Caffe Model Zoo에서 미리 훈련된 여러 네트워크를 바로 사용 가능
CNN 모델링이나 이미지 처리 문제 해결
Caffe를 고성능 개방형 학습 모델을 구축 할 수 있는 Caffe2 출시

- Keras
직접 모델을 만드는 경우 Theano와 Tensorflow 보다 적용이 쉬움
K효율적인 신경망 구축을 위한 단순화 된 인터페이스로 개발
Theano 또는 Tensorflow에서 작동하도록 구성
Python으로 작성, 매우 가볍고 배우기 적합
적은 코드 작성으로 Keras를 사용하여 신경망을 만들 수 있음

네트워크

CNN (Convolutional Neural Network)
RNN (Recurrent Neural Network)
RBM (Restricted Boltzmann Machine)

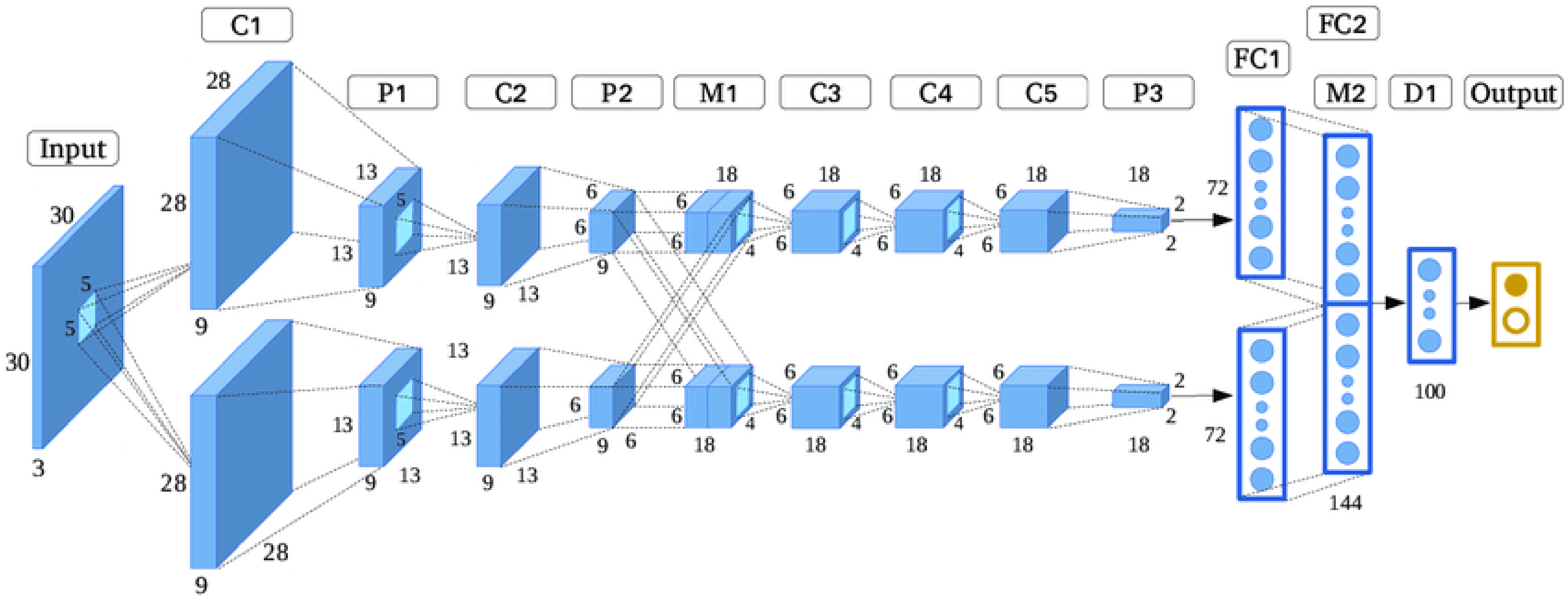
CNN 모델

AlexNet
GoogleNet
ResNet
DenseNet
RCNN(Region Based CNNs)
CNNs for NLP

FAST 객체 탐색 기법

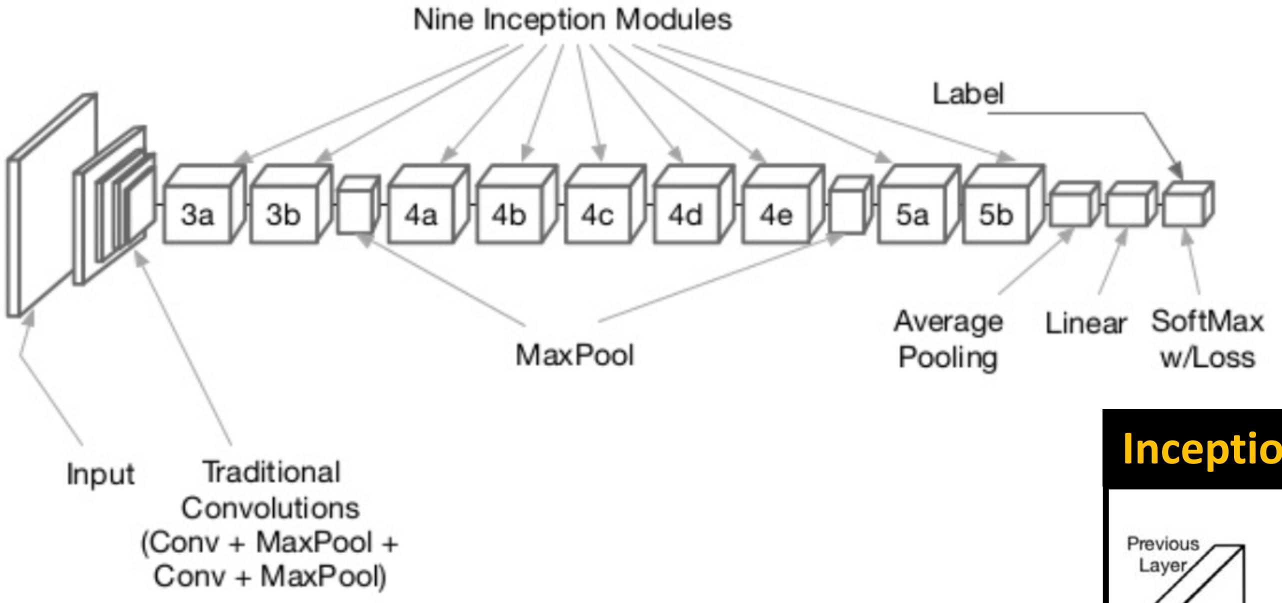
YOLO(You only Look Once)
SSD(Single Shot Detector)
Fast R-CNN, Faster R-CNN, Mask R-CNN

CNN Model -AlexNet

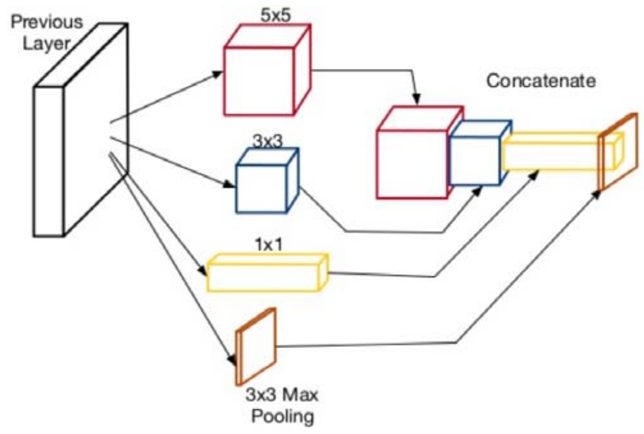


https://www.researchgate.net/figure/AlexNet-like-architecture_fig4_320723863

CNN Model - GoogleNet

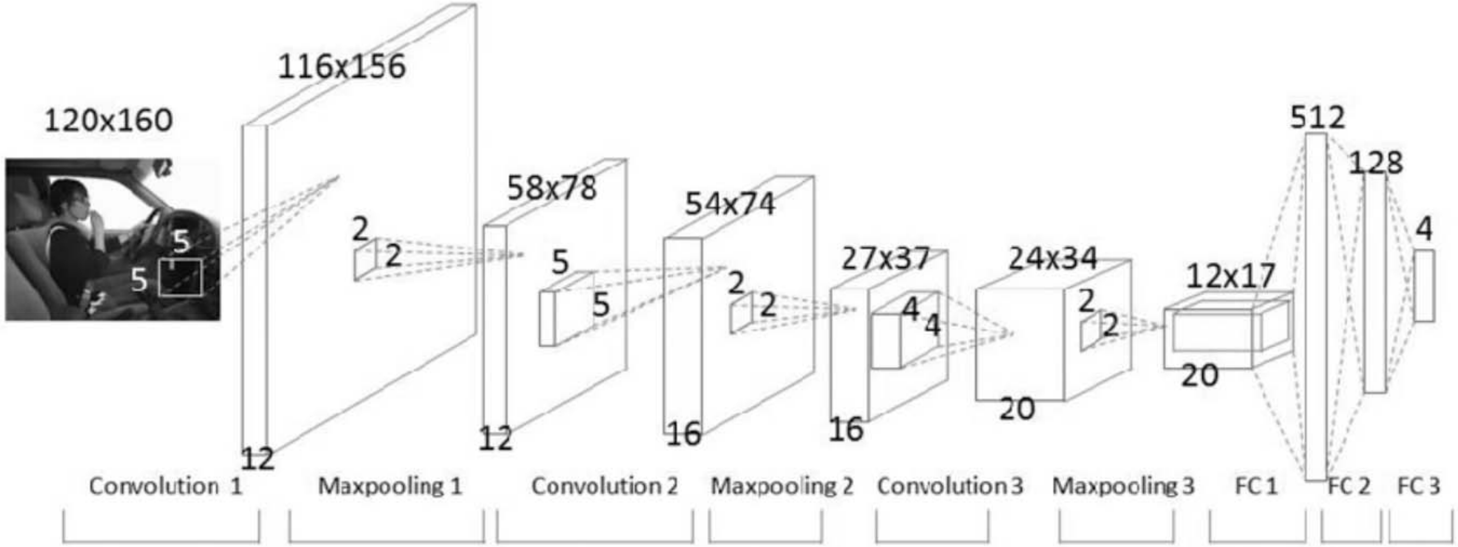


Inception Module



https://www.slideshare.net/aurot/googlenet-insights

CNN 기초

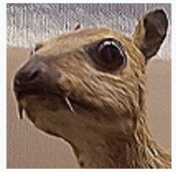
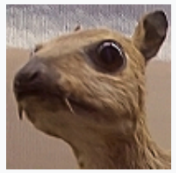


- Filter
- Kernel
- Convolution
- Stride
- Feature Map
- Activation Function
- Channel
- Padding
- Activation Map
- Pooling Layer
- Fully Connected Layer
- Dropout
- Soft Max

https://www.researchgate.net/figure/Architecture-of-our-unsupervised-CNN-Network-contains-three-stages-each-of-which_283433254

3. 딥러닝(Deep Learning)

Filter, Kernel

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Unsharp masking 5 × 5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

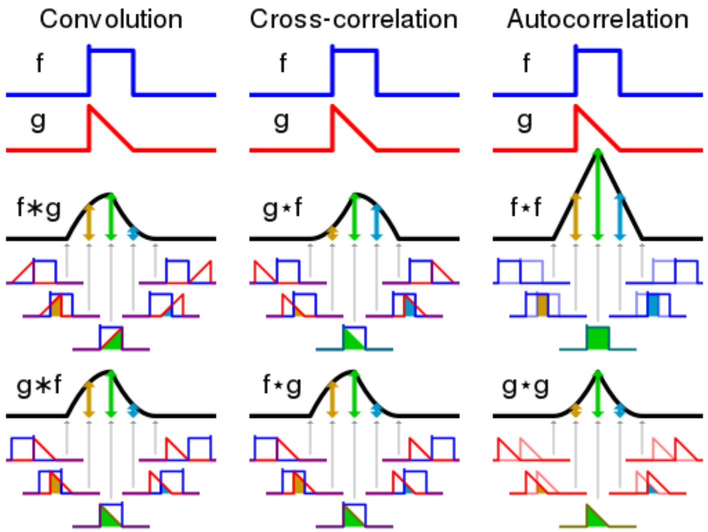
https://en.wikipedia.org/wiki/Kernel_(image_processing)

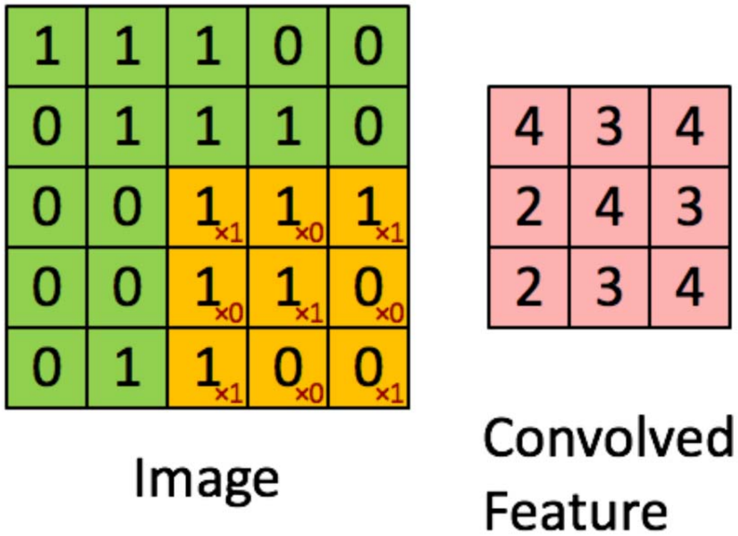
Convolution

- 컨볼루션의 정의
  - 두 함수를 합성하여 만든 새로운 함수로 아래와 같이 정의

$$h(x) = (f * g)(x) = \int f(a)g(x-a)da$$

  - 개념적으로는 두 함수가 서로 볍치는 면적이 컨볼루션 함수의 값

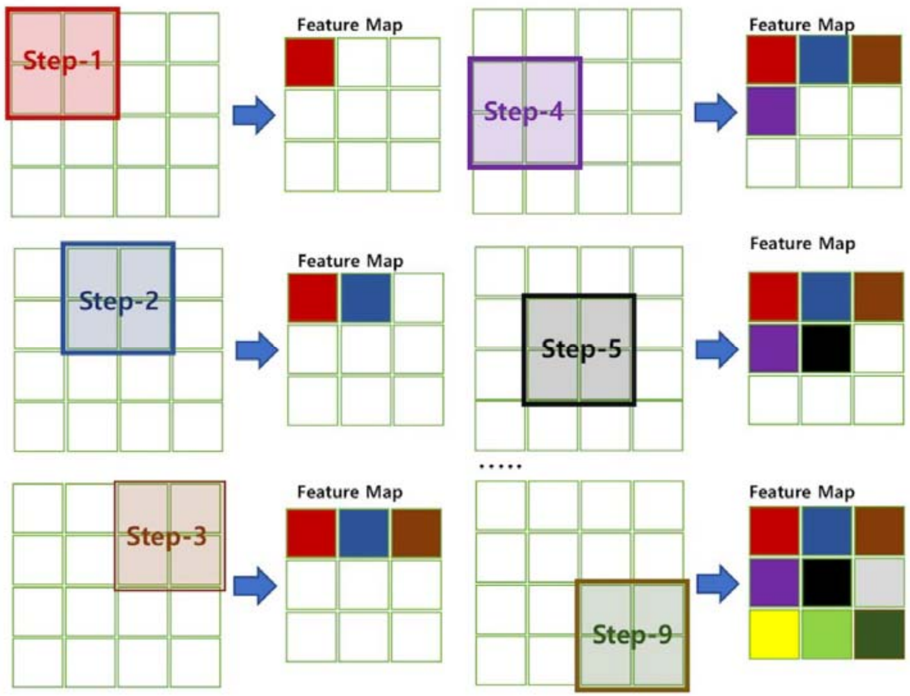




https://en.wikipedia.org/wiki/Convolution
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Stride, Step, Feature Map, Activation Function

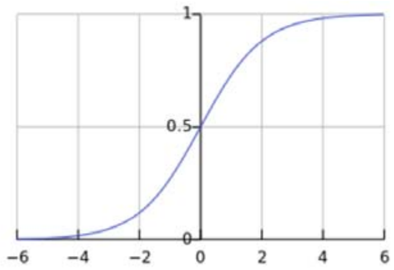
stride가 1로 필터를 입력 데이터에 순회하는 예시



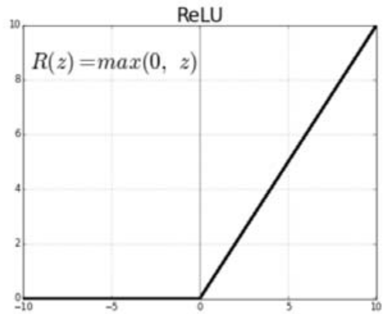
stride가 2로 설정되면 필터는 2칸씩 이동하면서 합성곱 계산

Activation Map은 Feature Map 행렬에 활성 함수를 적용한 결과

Activation Function



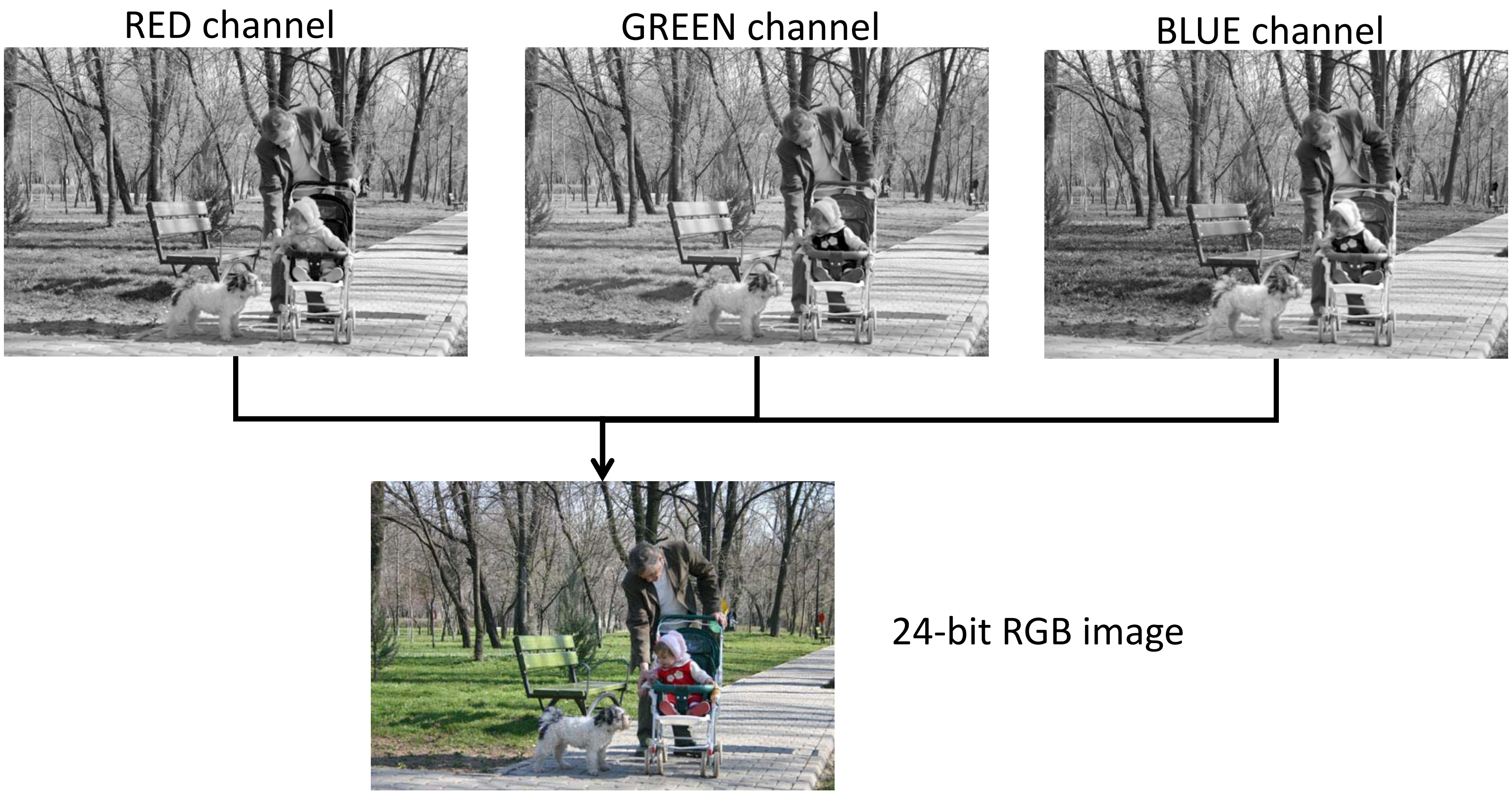
Sigmoid



ReLu

3. 딥러닝(Deep Learning)

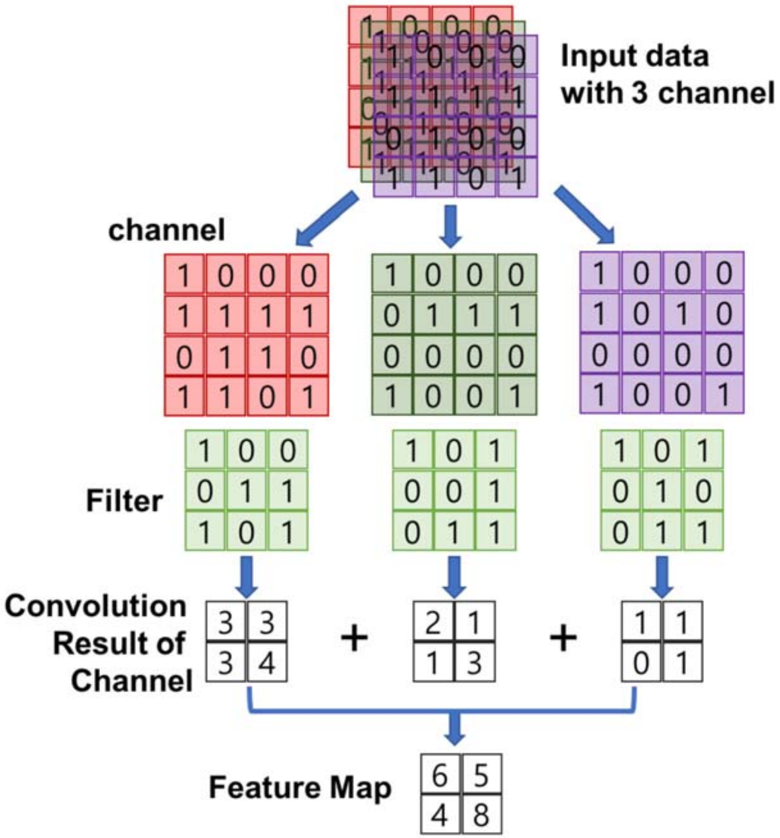
Channel

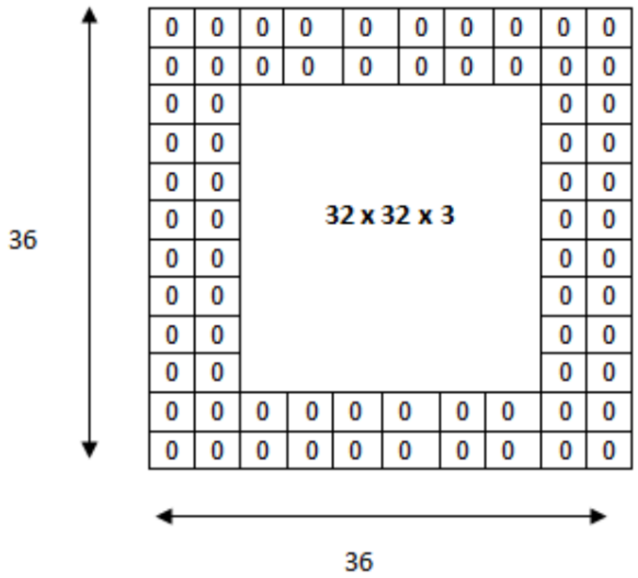


https://en.wikipedia.org/wiki/Channel_(digital_image)

Multi- Channel, Feature Map, Activation Map, Padding

Activation Map은 Feature Map 행렬에 활성 함수를 적용한 결과

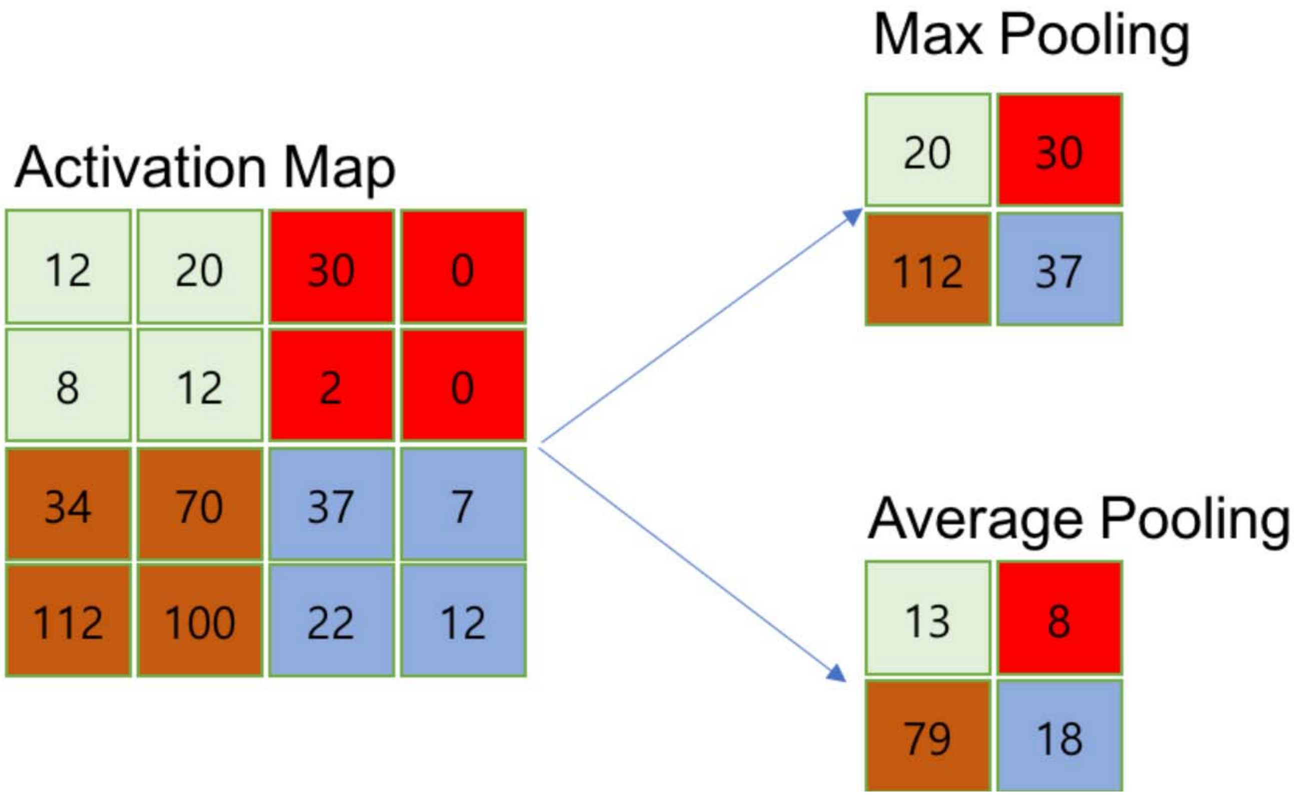




https://stats.stackexchange.com/questions/274601/convolutional-neural-networks-what-is-done-first-padding-or-convolving

Pooling Layer

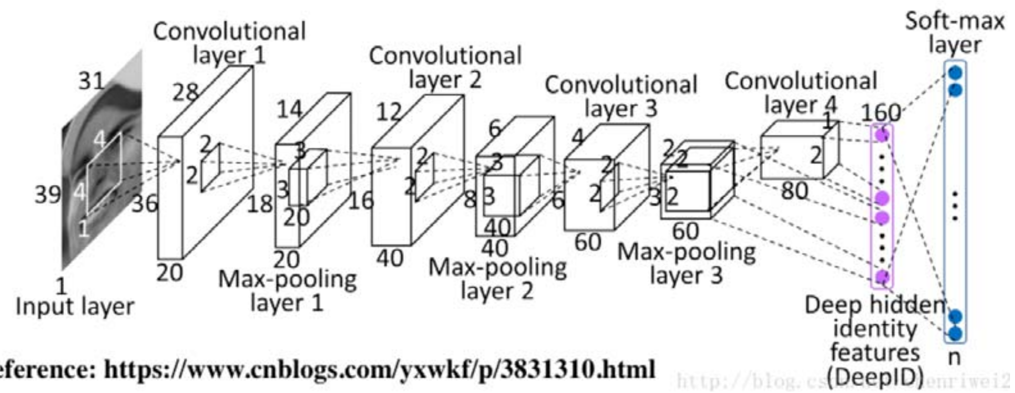
Pooling Layer 종류 : Max Pooling, Average Pooling, Min Pooling



http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

전체 파라미터 수와 레이어 Input/Output 요약

layer	Filter	Stride	Pooling	활성함수	Input Shape	Output Shape	파라미터 수
Convolution Layer 1	(4, 4, 20)	1	X	relu	(39, 31, 1)	(36, 28, 20)	320
Max Pooling Lyaer 1	X	2	(2, 2)	X	(36, 28, 20)	(18, 14, 20)	0
Convolution Layer 2	(3, 3, 40)	1	X	relu	(18, 14, 20)	(16, 12, 40)	360
Max Pooling Lyaer 2	X	2	(2, 2)	X	(16, 12, 40)	(8, 6, 40)	0
Convolution Layer 3	(3, 3, 60)	1	1	relu	(8, 6, 40)	(6, 4, 60)	540
Max Pooling Lyaer 3	X	2	(2, 2)	X	(6, 4, 60)	(3, 2, 60)	0
Convolution Layer 4	(2, 2, 80)	1	1	relu	(3, 2, 60)	(2, 1, 80)	320
Flatten	X	X	X	X	(2, 1, 80)	(160, 1)	0
fully connected Layer	X	X	X	softmax	(160, 1)	(100, 1)	160,000



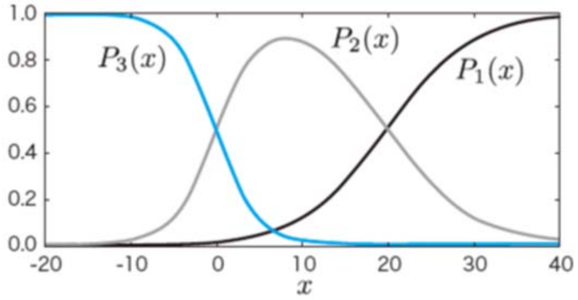
Reference: https://www.cnblogs.com/yxwkf/p/3831310.html

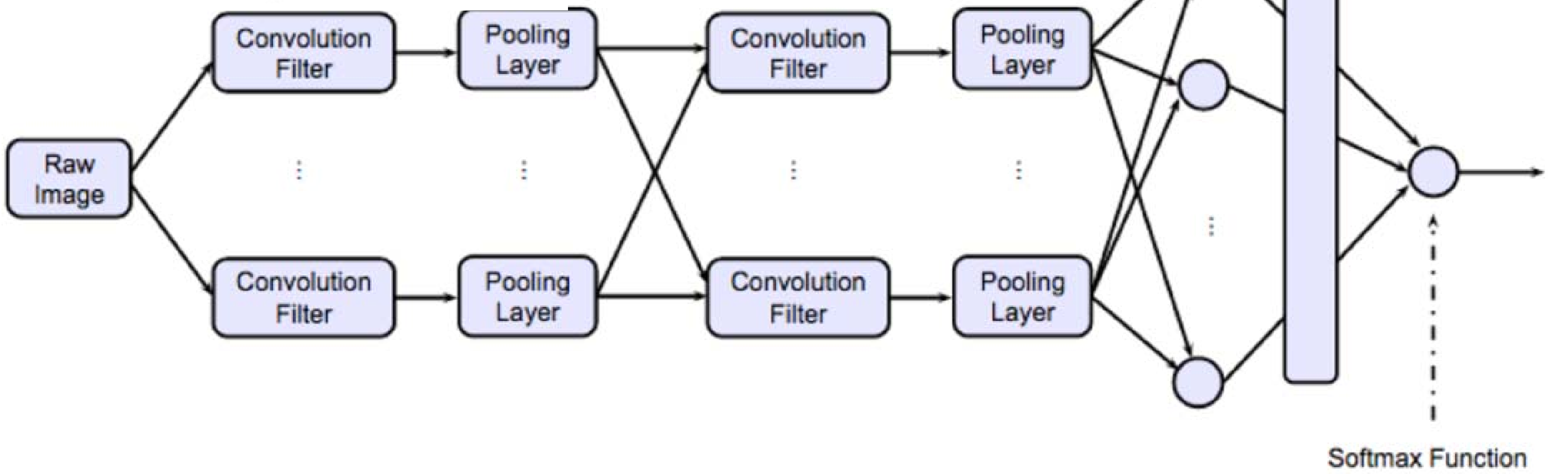
(주)딥비전

Fully Connected Layer, Softmax Layer

- Softmax

- 앞에서 언급한 sigmoid나 ReLu와 같은 Activation Function의 일종
- 여러 종류의 분류를 가질 수 있는 함수



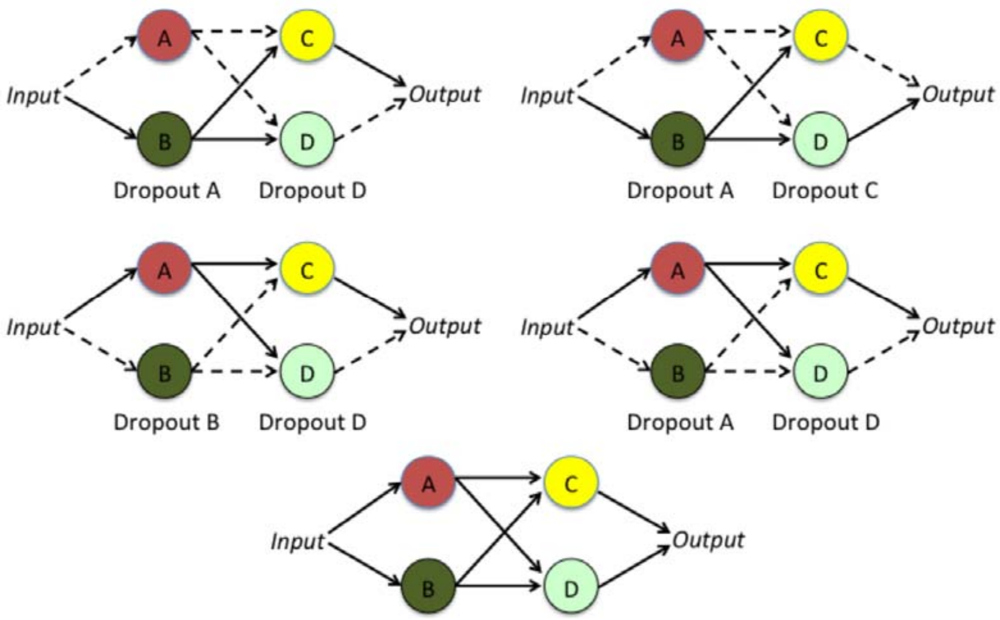


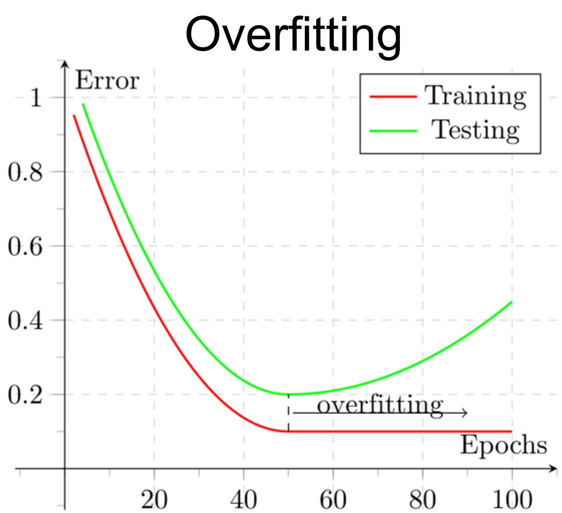
http://bcho.tistory.com/1149

Dropout Layer, Over-fitting

- **Dropout** : Overfitting을 줄이기 위한 정규화 기법

CNN에서는 Dropout Layer를 Fully connected network 뒤에 놓지만, 상황에 따라 max pooling 계층 뒤에 놓기도 함

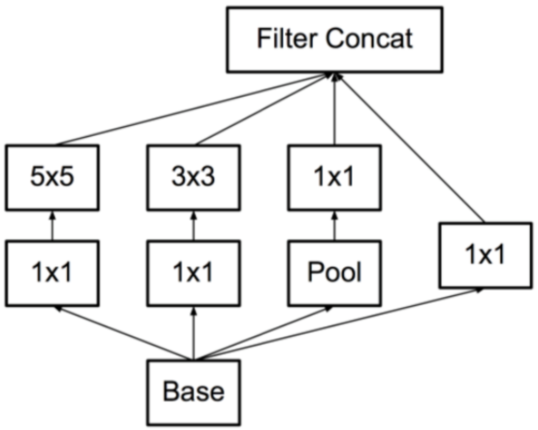




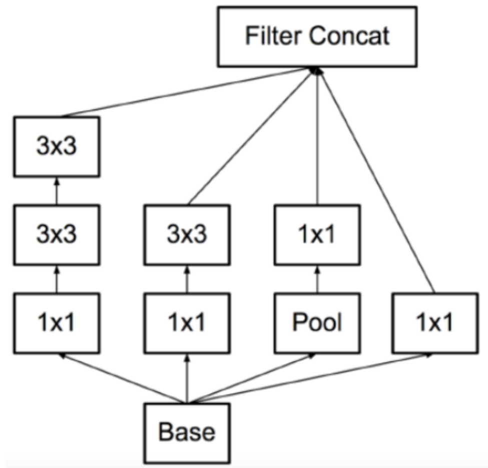
https://commons.wikimedia.org/wiki/File:2d-epochs-overfitting.svg
https://medium.com/@vivek.yadav/why-dropouts-prevent-overfitting-in-deep-neural-networks-937e2543a701

(주)딥비전

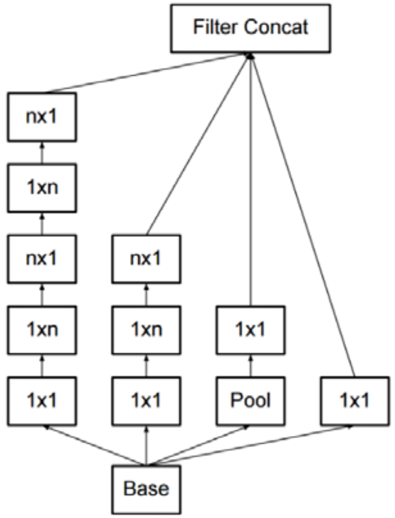
Inception, Factorization, Asymmetric Factorization



Inception



Factorization



Asymmetric Factorization

https://norman3.github.io/papers/docs/google_inception.html

감사합니다.

Thank you.