CNN을 활용한 이미지 딥러닝 실습

- Classification -

Presented by: Park JunYeong

wndwls1024@naver.com

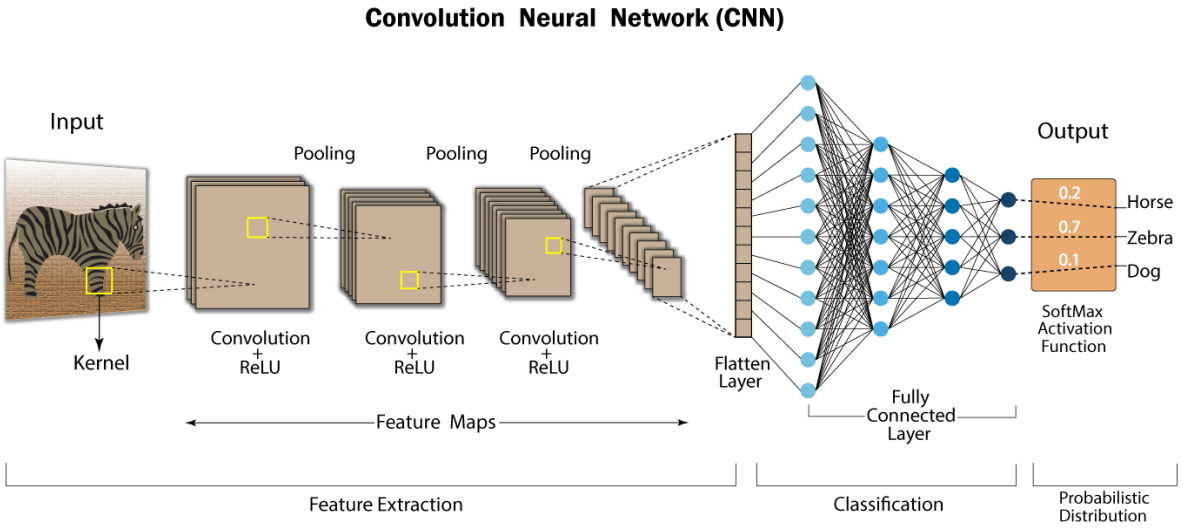
Contents

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

Overview

Convolutional Layer과 Fully-Connected Layer

- CNN, Convolutional Neural Network는 보통 2개의 부분으로 이루어진다.
- Convolutional Layer가 모여있는 feature extraction부분,
- 그 뒤에 붙어서 classification을 수행하는 classification 부분으로 이루어진다

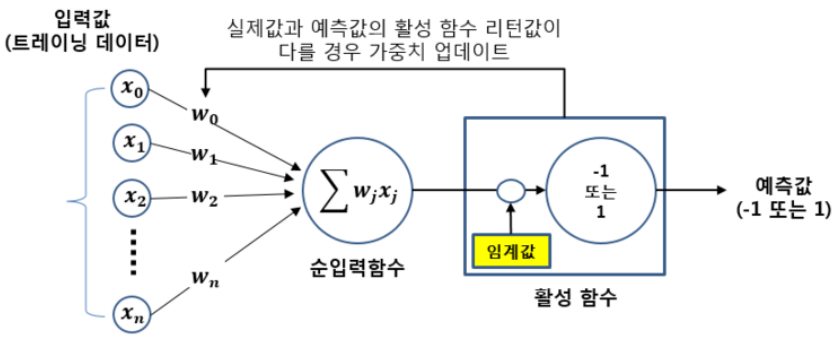


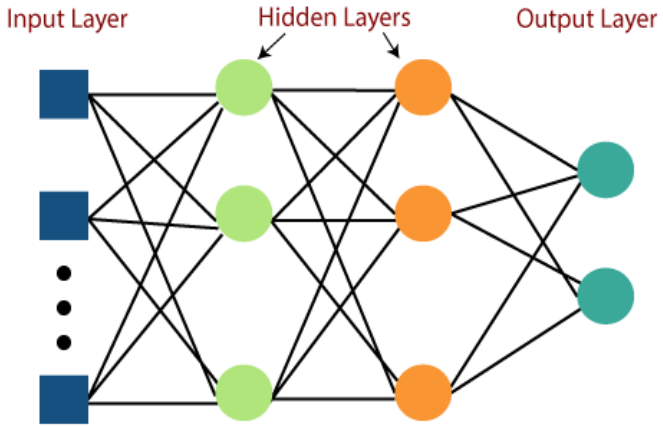
Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

Overview

Multi-layer perceptron

- 퍼셉트론(Perceptron): 인공 신경망(Aritificial Neural Network, ANN)의 구성 요소(unit)로서 다수의 값을 입력받아 하나의 값으로 출력하는 알고리즘
- Input Layer, Hidden Layer, Output Layer로 구성.
- 주로 입력벡터의 feature들 사이 상관관계를 파악할 때 사용
- Output Layer에서 Softmax 혹은 Sigmoid 함수를 사용해 결과를 확률화 하거나, 최종 예측 결과를 출력함
- 여러 Hidden Layer 계층이 쌓여있음





Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

Overview

Activation Function

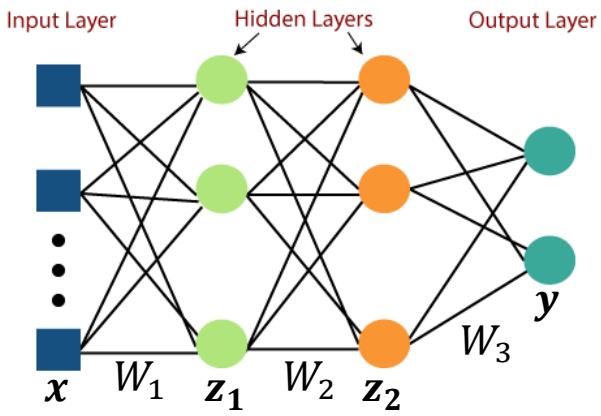
- 선형 연산에 비선형성을 더해주기 위해 사용되는 함수

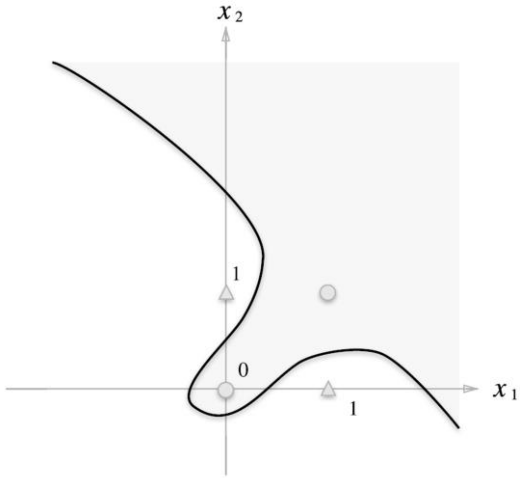
$$\boldsymbol{z_1} = \boldsymbol{x} \cdot W_1$$

$$\boldsymbol{z_2} = \boldsymbol{z_1} \cdot W_2$$

$$\boldsymbol{y} = \boldsymbol{z_2} \cdot W_3$$

- 만약 비선형성이 없다면? ➔ 여러 hidden layer를 쌓는 효과가 사라진다 + 최적의 경계선을 찾는데 한계
- $\boldsymbol{y} = \boldsymbol{x} \cdot W_1 \cdot W_2 \cdot W_3 = \boldsymbol{x} \cdot W_{out}$ $\quad (W_{out} = W_1 \cdot W_2 \cdot W_3)$
- 따라서 $\boldsymbol{z_1} = \boldsymbol{ReLU}(\boldsymbol{x} \cdot W_1)$ 와 같이 비선형성을 추가해준다.





- Sigmoid, Softmax, ReLU, Leaky ReLU … 등 여러가지 종류가 있고, 각각의 특징과 사용 목적이 있음.

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

Overview: Image

Image Processing

Classification

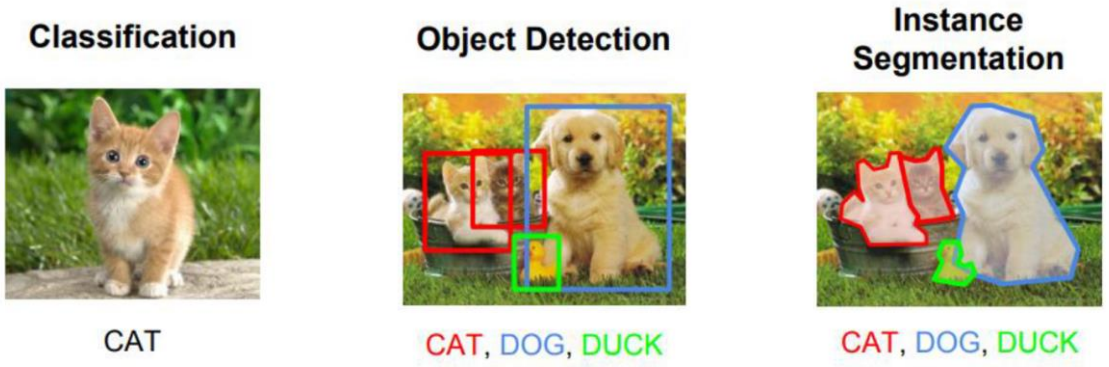
- 이미지를 각 클래스에 맞도록 분류하는 것. 클래스의 개수에 따라 Binary Classification과 Multi-class classification으로 나눈다.

Object Detection

- 이미지에서 사물을 탐지해 바운딩 박스로 표시하고 클래스를 분류한다.

Segmentation

- 픽셀 당 연산을 통해 이미지에서 사물의 테두리를 탐지한다. 클래스의 개수에 따라 Instance Segmentation과 Semantic Segmentation으로 나눈다.



Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

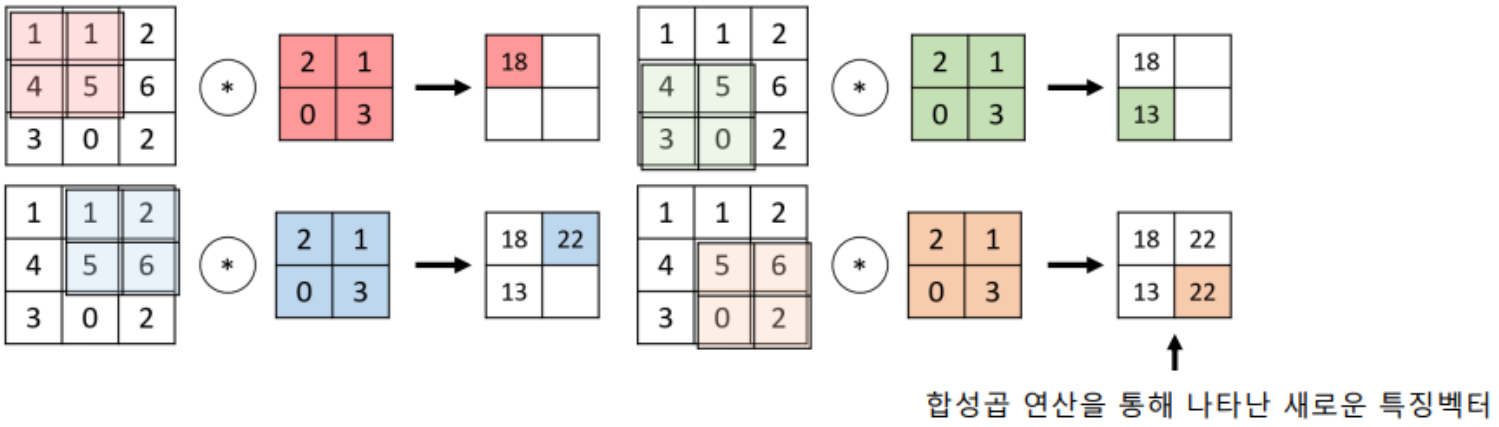
Overview: CNN

이미지 데이터를 처리

이미지는 3차원 형상이므로 3차원 형상을 입력으로 받아서 다음 계층에도 3차원으로 전달하는 방법을 사용 -〉 합성곱 신경망 (CNN) 사용

합성곱 신경망(CNN)

합성곱 연산은 입력 데이터에 필터를 적용한다. 필터의 윈도우를 일정 간격으로 이동하며 입력 데이터에 적용하고, 이를 통해서 새로운 특징벡터를 추출한다



Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

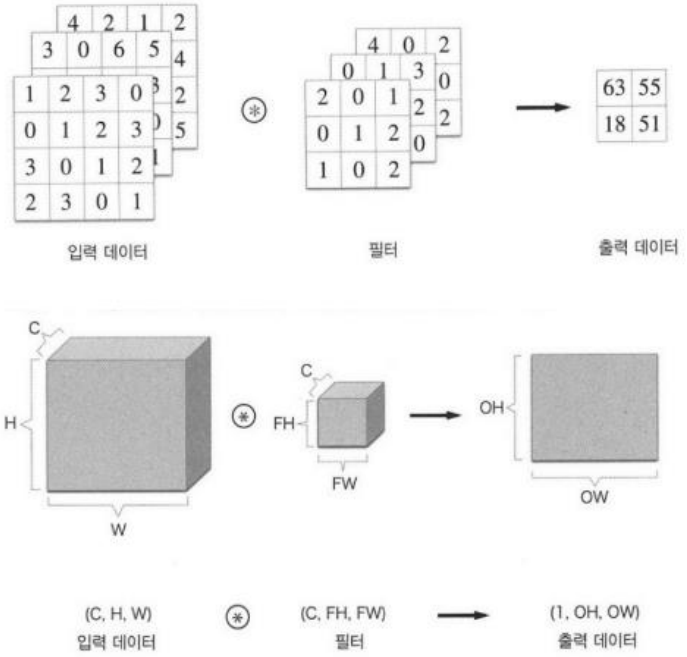
Overview: CNN

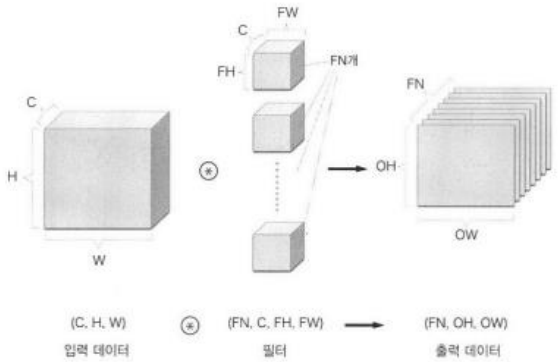
이미지에서 실제로 이뤄지는 합성곱 연산

입력 데이터: 3차원

입력 데이터: 하나의 필터는 입력 데이터의 갯수만큼의 채널로 이루어졌다.

필터는 보통 한개만 사용되는 것이 아닌 여러 개가 동시에 사용된다.





Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

Overview: CNN

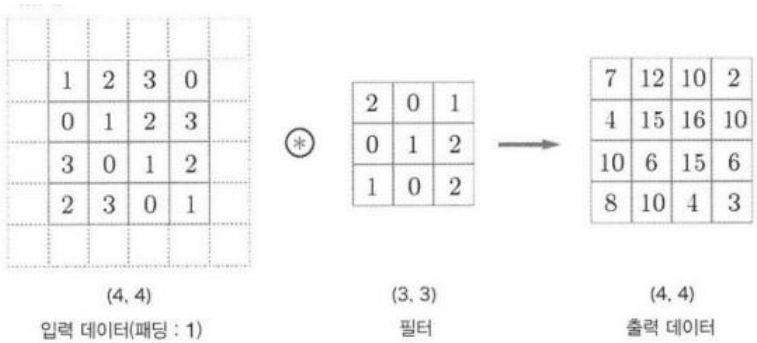
CNN연산에서 고려해야할 요소들

패딩(padding)

합성곱 연산을 수행하기 전에 데이터 주변을 특정 값으로 채우는 것 합성곱 연산을 수행한 이후 출력 데이터의 크기가 입력 데이터와 같도록 하기 위해서 주로 사용된다.

입력 데이터가 nxn이고, 필터의 사이즈가 mxm일 경우, 출력 데이터는 (n-m)x(n-m)이 된다. 출력 데이터를 nxn으로 맞춰주기 위해서 padding을 쓰는 것.

Padding 종류: zero padding, constant padding(지정된 상수로 채움)

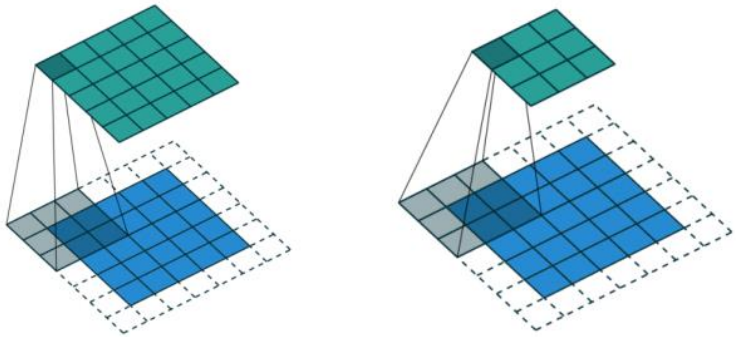


Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

Overview: CNN

스트라이드(Stride)

필터를 적용하는 위치의 간격 스트라이드 값이 클수록 출력데이터의 크기가 작아진다.



Stride 1 Stride 2

스트라이드와 패딩을 사용했을 때 출력 데이터의 크기

$$OH = \frac{H + 2P - F}{S} + 1$$

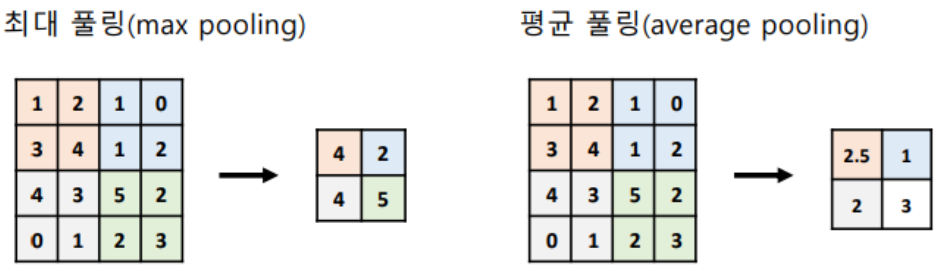
$$OW = \frac{W + 2P - F}{S} + 1$$

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

Overview: CNN

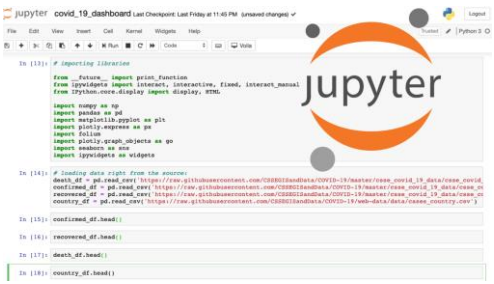
풀링 Pooling

- 필터를 적용하는 위치의 간격 스트라이드 값이 클수록 출력데이터의 크기가 작아진다.
- 가로 세로를 줄이는 연산이다.
- CNN에서는 여러 convolutional layer를 거친 후 pooling layer를 통해 공간 크기를 줄인다.
- 보통 max pooling을 많이 사용한다. ➔ 특징) 학습해야 할 매개변수가 없다.
- 채널수가 변하지 않는다.
- 입력의 변화에 영향을 적게 받는다.(입력 데이터가 조금 변해도 풀링 변화는 잘 변하지 않는다)



Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

실습 환경 종류



Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

실습 환경

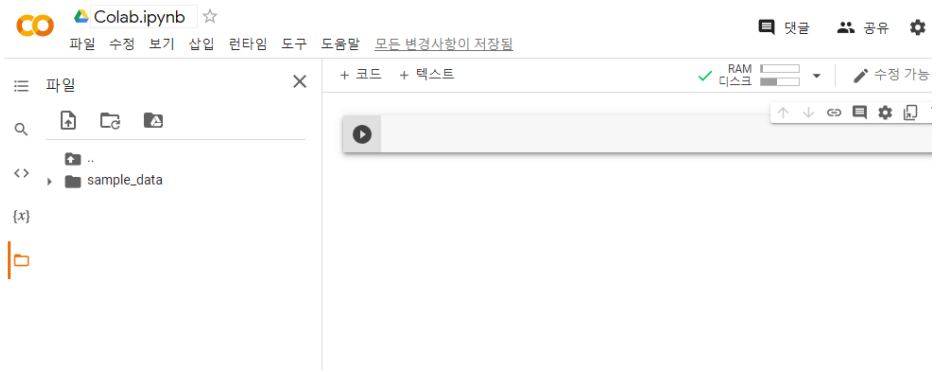
colab

구글 코랩은 구글 클라우드 기반의 무료 Jupyter Notebook 이다.

접근성이 쉬우며 개개인의 PC보다 좋은 성능의 GPU를 무료로 액세스 할 수 있어 많은 사람들이 딥러닝을 학습하는 데 많이 사용하고 있다.

- Jupyter Notebook은 대화형 파이썬 인터프리터(Interpreter)로서 웹 브라우저 환경에서 파이썬 코드를 작성 및 실행할 수 있는 툴이다.

https://colab.research.google.com/



Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

실습 환경

colab

https://colab.research.google.com/

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

CHAPTER 1

이미지 분류

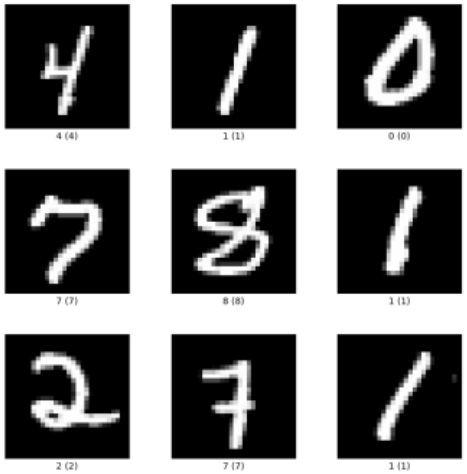
(Image Classification)

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

Image Classification

MNIST dataset

- MNIST(Modified National Institute of Sandards and Technology database) 데이터셋은 손으로 쓴 숫자들로 이루어진 대형 데이터베이스이다.
- 60,000개의 트레이닝 이미지와 10,000개의 테스트 이미지를 포함한다



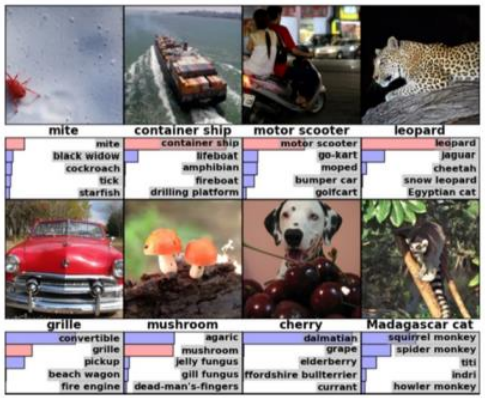
하나의 이미지는 28x28의 모양으로 구성되어 있으며,
Pytorch의 torchvision에서 제공하는 dataset으로 불러올 경우
Train dataset은 60000개의 데이터로 구성되어있고, 각 데이터는
리스트로 구성되어있는데, 각 리스트는
1x28x28의 이미지데이터와 정수 label 1개가 들어가있다.

- **Train**
  - 0
    - 1x28x28
    - 1x28x28
    - ...
    - 총 60000개
  - 1
    - 1번 Label
    - ...
    - 총 60000개
- **Test**
  - Trian과 동일한 구조

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

대표 모델: AlexNet

- AlexNet은 2012년에 발표되었다.
- 당시 ILSVRC(ImageNet Large-Scale Visual Recognition Challenge)의 2012년 대회에서 top-5 에러가 15.4%로 2위의 26.2%에 비해 높은 성능으로 1위를 차지하고 주목을 받았다.
- 또한 Top-5 에러도 비교적 납득할 수 있는 에러들이 많았다.

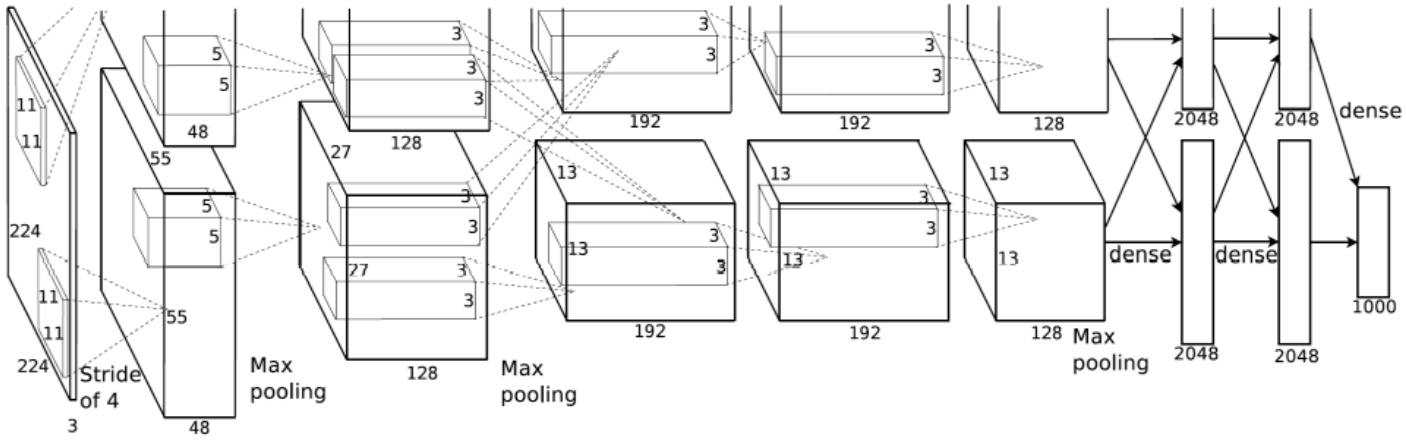


AlexNet의 특징들

1. Max-pooling layer사용
2. Local response normalization사용 – (최근에는 잘 사용x)
3. GPU를 학습에 사용
4. Dropout 적용
5. 활성화함수로 ReLU 사용

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

대표 모델: AlexNet



256x256x3의 이미지를 받아서 총 1000개의 클래스로 구분을 해야 하는 모델이므로 LeNet에 비해서 훨씬 복잡한 구성을 갖는다.

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

대표 모델: AlexNet

특징들

더 큰 네트워크를 위해 2개의 GPU를 이용해 병렬 연산을 수행

3번째 conv layer와 Fully connected layer를 제외하면 독립적으로 훈련을 진행

이를 통해 training error를 1~2% 줄일 수 있었고, 학습 시간도 더 빨라짐

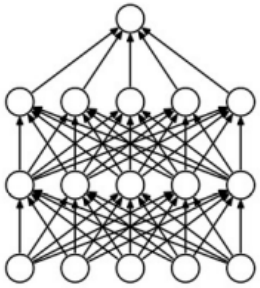
-〉 딥러닝 모델에서 GPU를 이용해 학습하는 것이 중요하게 됨

GPU에서 학습

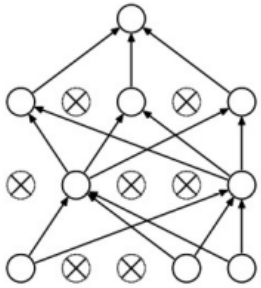
Dropout은 히든 레이어의 특정 뉴런의 출력을 일정 확률로 0으로 만드는 것이다.

이를 통해 하나의 뉴런의 값에 크게 의존하지 않도록 모델이 학습한다.

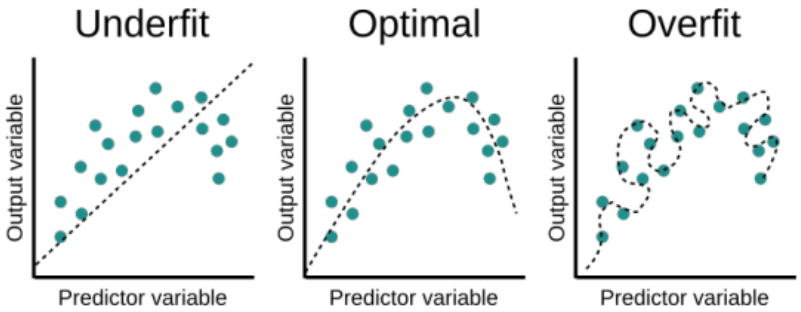
-〉 overfitting(과적합)을 막는 중요한 방법 중 하나



(a) Standard Neural Net



(b) After applying dropout.



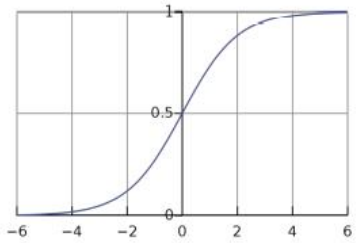
Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

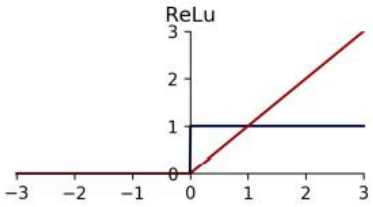
대표 모델: AlexNet

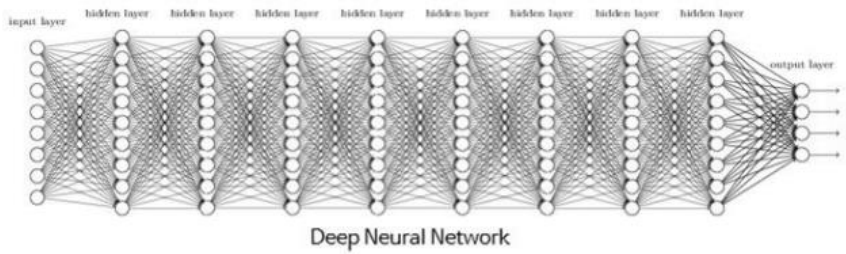
특징들

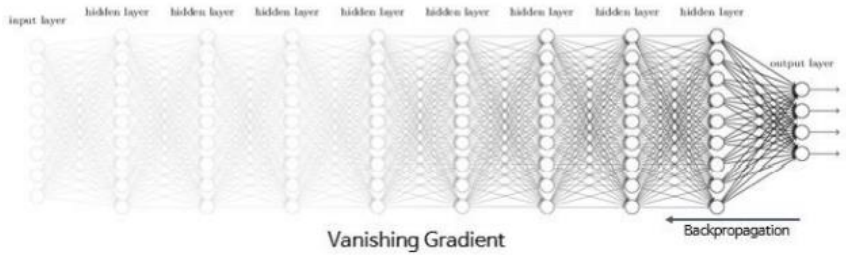
ReLU

기존 시그모이드에서는 기울기(미분값)이 0.25이므로 Chain rule에 의해서 계속 0.25씩 곱하다보면 네트워크의 초반쪽에서는 값이 0으로 수렴해가고, 이러면 학습이 안 된다.









Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

대표 모델: AlexNet

Weight decay

L2 regualrization을 사용하였다.

이 방법 또한 과적합을 막아주는 효과를 갖는다.

가중치 초기화

평균 0, 표준편차 0.01의 정규분포로 가중치를 초기화하였다.

2, ,4, 5번째 convolution layer와 fully-connected layer에 한해 1로 초기화하였다.

Learning rate(학습률)은 0.01로 시작해 오차가 더 줄지 않으면 10으로 나누는 식으로 0.01부터 총 3번 나누었다

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

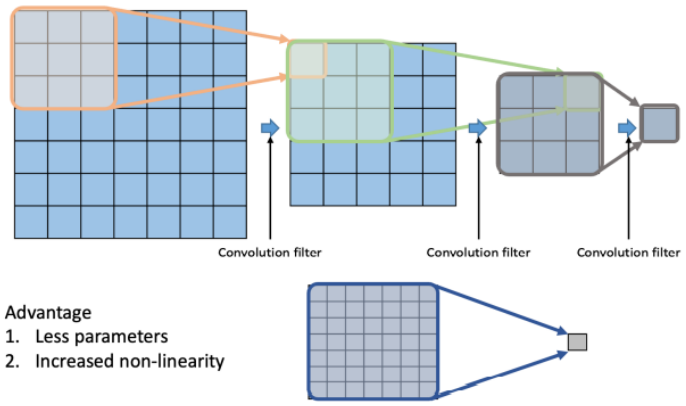
대표 모델: VGG

Very Deep Convolutional Networks for Large-Scale Image Recognition

2012년 AlexNet의 주목 이후 사람들은 더 높은 정확도를 얻기 위해 더 깊은 신경망이 필요하다는 것을 알게 됨.
이에 따라 더 많은 layer를 갖는 CNN이 등장하게 된다.

층을 깊게 할 경우 성능향상을 얻을 수 있다고 생각하여 층을 깊게 쌓는것이 중심을 둔 논문이 VGG이다.
단, 층을 깊게 쌓을때, CNN의 커널의 크기가 5x5나 7x7라면 parameter의 숫자가 엄청 많아진다.
그래서 3x3 크기의 커널만을 이용하여 깊이를 늘리고, 5x5, 7x7크기의 커널을 대체하고자 하였다.

7x7 vs 3x3 2개



7x7 하나 사용시 하나의 활성화 함수를 사용한다.
3x3 3개 사용시 3개의 활성화 함수를 사용한다.
활성화 함수는 non-linearity를 증가시키는 요인

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

대표 모델: VGG

3x3 커널을 쓸 때 장점

1. 적은 수의 parameter

3x3 convfilter 3개 vs 7x7 convfilter 1개

$3(3^2C^2) = 27C^2$ vs $7^2C^2 = 49C^2$

C = number of channels

2. Non-linearity(비선형성) 증가

이러한 관점에서 1x1 convolution filter도 non-linearity를 증가시킨다고 주장

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

대표 모델: VGG

대부분의 훈련 방법은 AlexNet을 따른다.

Batch size=256, momentum=0.9, dropout은 첫 fully-connected layer에 적용

Learning rate는 0.01로 시작해서 정확도가 더 늘어나지 않으면 10으로 나눠서 총 3번 적용

AlexNet에 비해서 더 깊고 많은 parameter를 가져서 모델의 복잡도는 더 높음

다만 특징을 추출하기 위해 더 적은 epochs가 필요하다

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers

Fully connected layer를 1x1 convolutional filter를 사용해서 convolution layer로 바꾸는 방법을 사용
여러개의 GPU를 이용해 훈련을 진행하였고,
각 batch들이 gradient(기울기)를 구하고 이들의 평균을 이용해 gradient값으로 사용한다.

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

대표 모델: ResNet

VGG에서 많은 레이어를 이용해 모델의 성능을 높이는 작업들을 진행하였다.

그렇기에 사람들은 더 많은 layer를 갖는 CNN을 만들어 높은 성능을 얻을 수 있을 것이라 기대했다.

그러나 실제로는 20층이 넘어가게 되면 성능이 낮아지는 현상이 발생하였다.

ResNet은 Residual Learning이라는 개념을 통해 모델의 층이 깊어져도 학습이 잘 되도록 하였다.

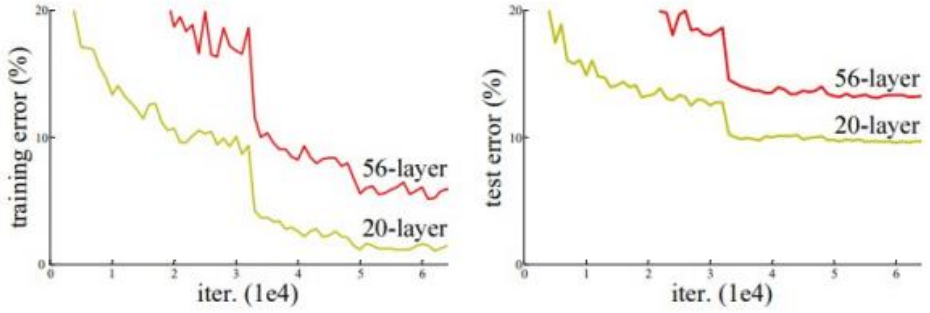
ResNet은 152층까지 네트워크를 쌓으며 ILSVRC 15와 COCO 15라는 대회에서 우승했다.

성능 좋고, 구현도 편하고, 단순하다.

깊은 네트워크의 문제점

과적합이 일어난 것이 아닌, 레이어가 깊어질수록 정확도가 낮아진다.
Degradation of accuracy.
Gradient vanishing / explosion도 발생 가능.

이런 현상의 원인을 layer가 깊어졌을때 optimization이 제대로 되지 않아서라고 추측



Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

대표 모델: ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

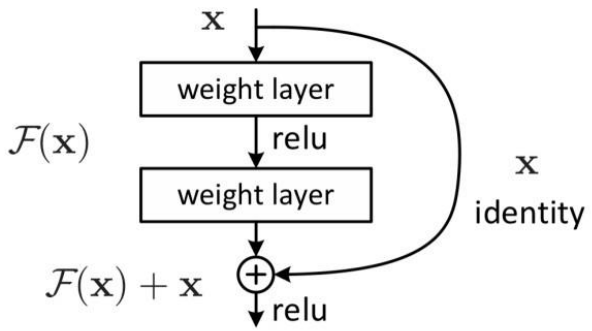


Figure 2. Residual learning: a building block.

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

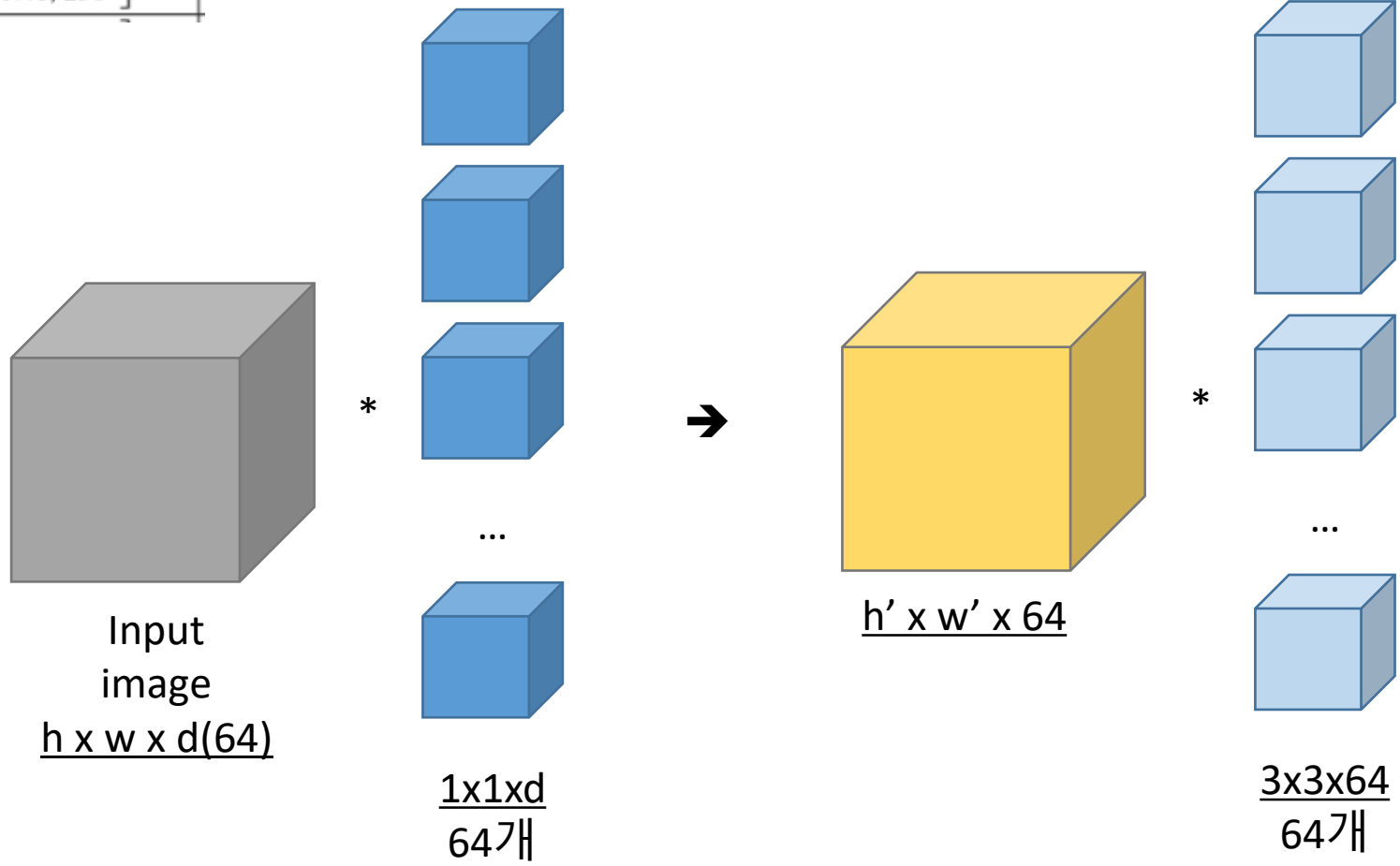
CHAPTER 2

pytorch를 활용한 딥러닝 모델 구현 실습

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

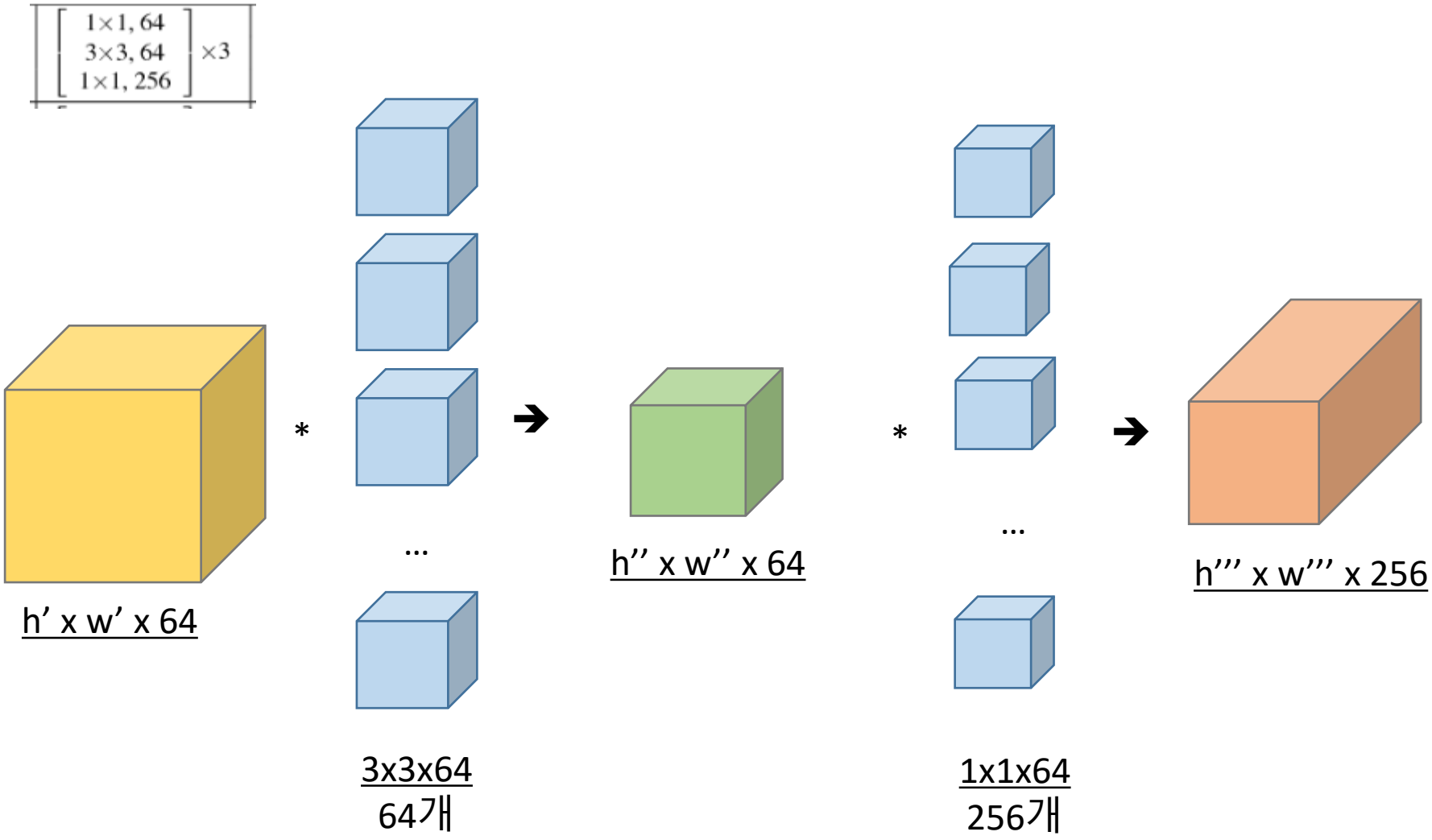
모델 구현: ResNet

$$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$$



Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

모델 구현: ResNet



Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.

수고하셨습니다.

Copyright 2016. @ Machine Learning & Pattern Analysis lab., Dankook Uni., all rights reserved.