

Week 8. CNN (Convolutional Neural Network)

Kisoo Kim

Kwangwoon University, Information Convergence

Kisooofficial@naver.com


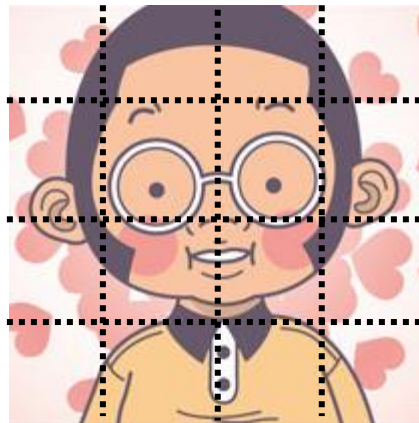
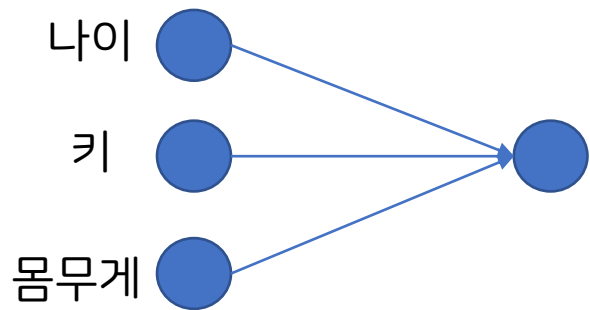
목차

- Introduction
 - 기존 정형 데이터의 구조와 이미지의 구조적 차이
- 합성곱 신경망(Convolution Neural Network)
 - 구조
 - 여러 가지 층 - 입력층(input layer), 합성곱층(convolution layer), 풀링층(pooling layer), 완전연결층, 출력층
 - Convolution, Padding, Stride
- 전이 학습(Transfer Learning)

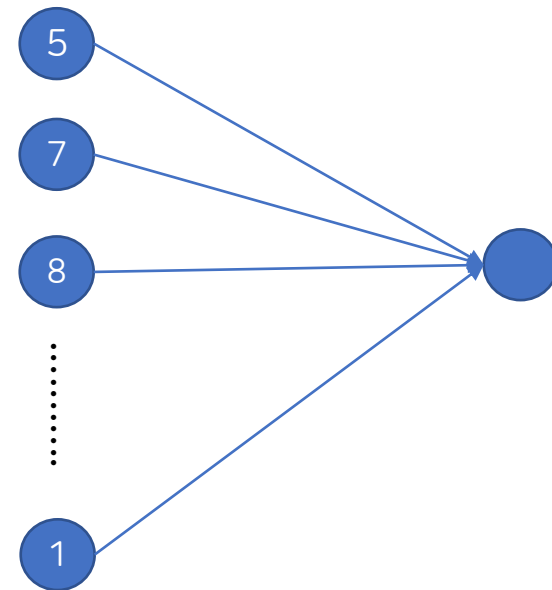
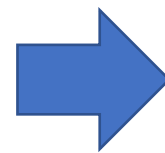
Introduction

- 기존 정형 데이터의 구조와 이미지의 구조적 차이

Index	나이	키	몸무게
1	25	172	64
2	24	180	92
3	25	177	75
4	22	169	71
5	28	188	80

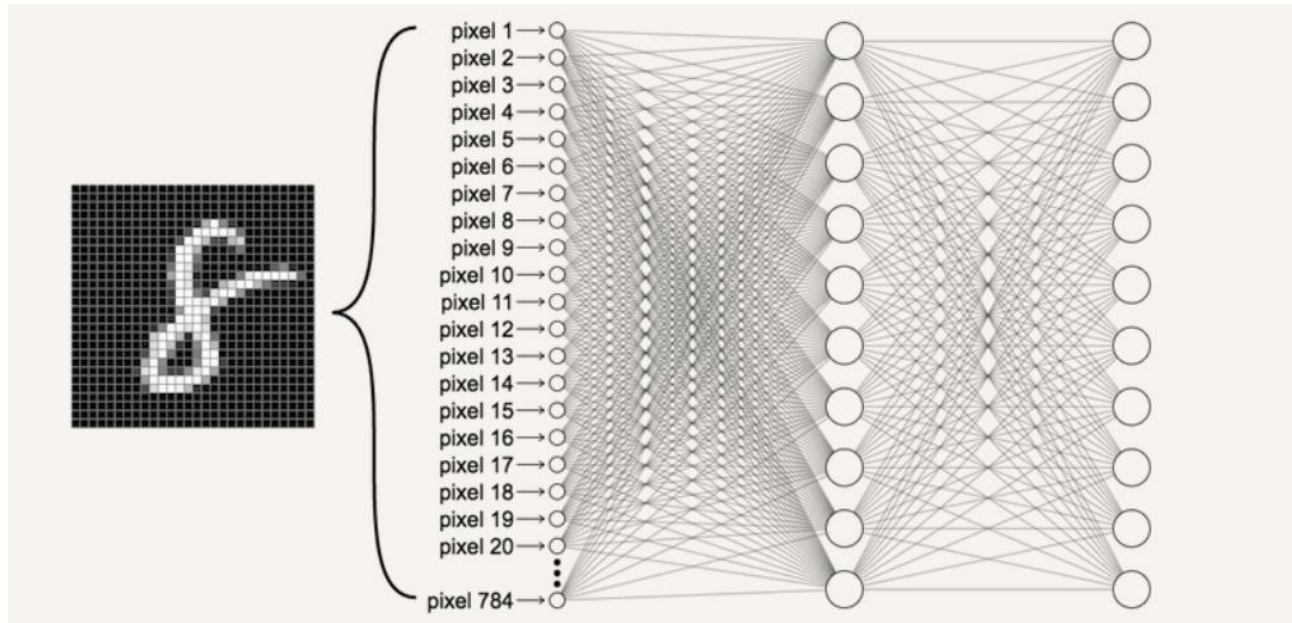


5	7	8	25
8	1	9	3
19	10	4	6
2	200	22	1



Introduction

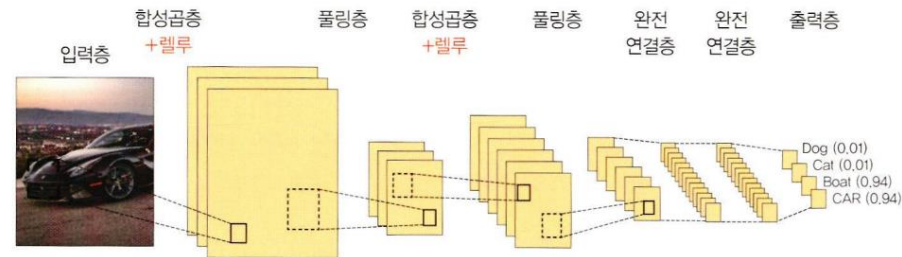
- MNIST 데이터를 ANN 구조로 학습했을 때 생기는 문제점
 - 이미지의 중요한 공간적인 정보를 학습할 수 없음
 - 앞의 4 x 4 예제에서 4번째 픽셀과 5번째 픽셀은 인접하지 않지만, ANN에서는 인접하게 보임
 - 너무 많은 파라미터를 학습해야 함
 - Ex. 첫 번째 은닉층의 output dimension이 500이면, 처음에 학습해야 할 파라미터의 수가 784×500



Convolution Neural Network

- 합성곱 신경망(Convolution Neural Network)의 구조
 - 입력층(Input layer)
 - 합성곱층(Convolution layer)
 - 이미지의 특성을 추출하는 과정
 - 풀링층(Pooling layer)
 - 이미지의 정보를 압축하는 과정
 - 완전연결층(fully-connected layer)
 - 출력층(output layer)

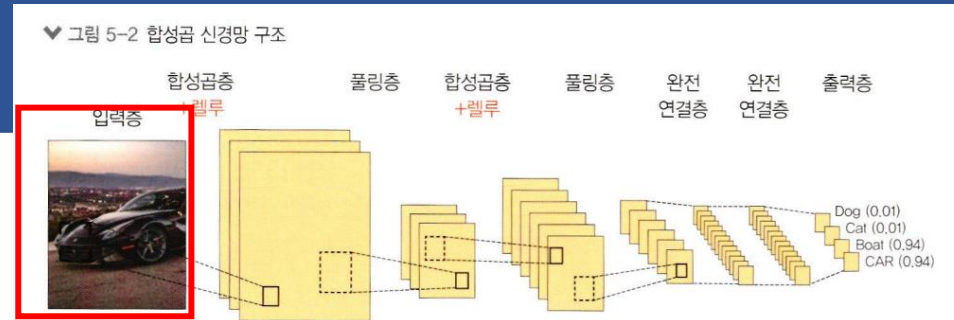
▼ 그림 5-2 합성곱 신경망 구조



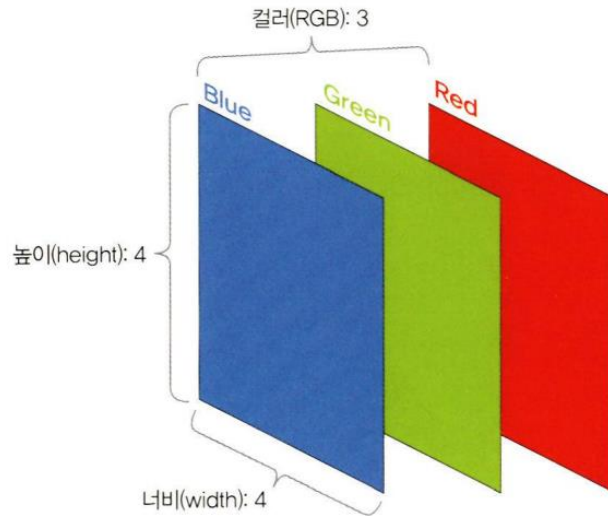
Convolution Neural Network 구조

- 입력층(Input layer)

- 입력 이미지 데이터가 최초로 거치게 되는 계층
- 기본적으로는 단순 1차원이 아닌 높이(height), 너비(width), 채널(channel)의 값을 3차원 형태의 구조
 - 이미지가 흑백 데이터의 경우에는 채널(channel)이 1이며, RGB 형태의 컬러 데이터인 경우에는 채널이 3
 - Ex. 앞선 예시에서는 컬러 데이터이므로 입력 데이터의 shape가 (4, 4, 3)



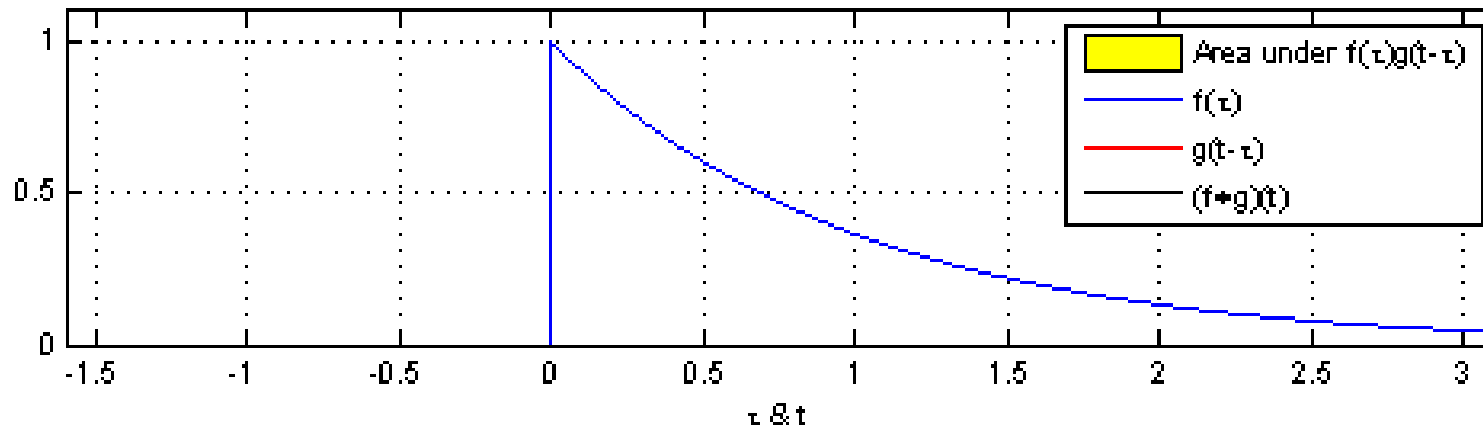
▼ 그림 5-3 채널



Convolution

- 합성곱(Convolution)

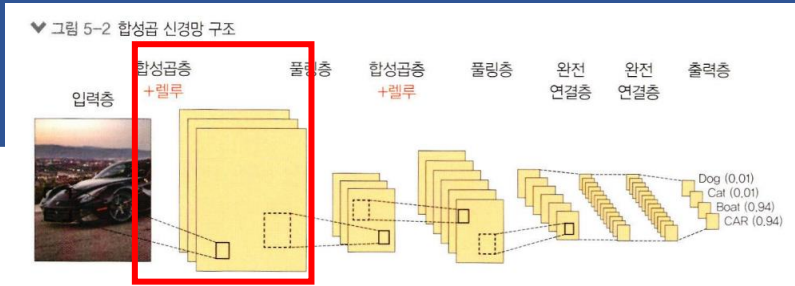
- 하나의 함수와 또 다른 함수를 반전 이동한 값을 곱한 다음, 구간에 대해 적분하여 새로운 함수를 구하는 과정
- $$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$
 - 아래 예시에서 파란색 부분이 f, 빨간색 부분이 g이며, g에서는 반전을 시킨 상태
 - 이미지에서 빨간색에 해당하는 부분을 커널(Kernel)이라고 함



Convolution Neural Network 구조

- 1D Convolution

- Element-wise하게 각 요소의 값들을 곱해서 더하기



Input (x)

1	3	2	3	0	4	1	2
---	---	---	---	---	---	---	---

Kernel (w)

1	0	-1
---	---	----

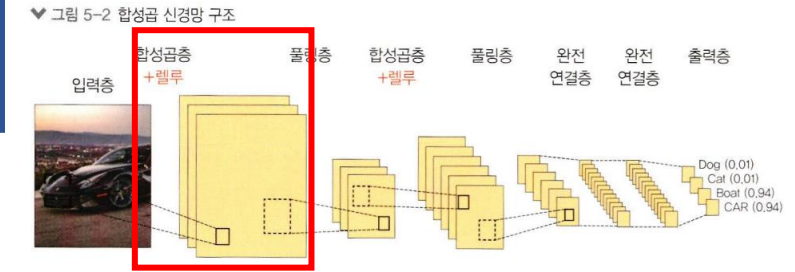
Output (x-w+1)

--	--	--	--	--	--

Convolution Neural Network 구조

- 1D Convolution

- Element-wise하게 각 요소의 값들을 곱해서 더하기



Input (x)	1	3	2	3	0	4	1	2
-----------	---	---	---	---	---	---	---	---

Kernel (w)	1	0	-1
------------	---	---	----

Output (x-w+1)	-1	0	2	-1	-1	2
----------------	----	---	---	----	----	---

Convolution Neural Network 구조

- 2D Convolution

- 1D Convolution에서와 비슷한 방법으로 진행

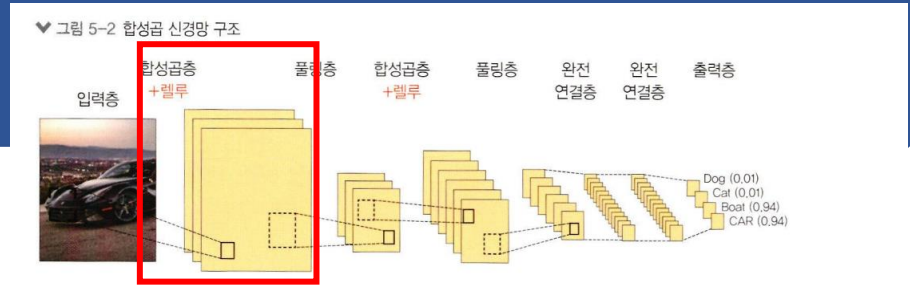
- 1D Convolution과의 차이는 2D Convolution에서는 커널이 상하좌우 모두 이동할 수 있음
 - Input의 형태는 $n \times n$, kernel의 형태는 $k \times k$ 의 형태

Input :

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Kernel :

1	0	1
0	1	0
1	0	1



Convolution Neural Network 구조

- 2D Convolution
 - Input : $x \times x$, kernel : $w \times w$
 - Output : $(x - w + 1) \times (x - w + 1)$
- 2D Convolution의 동작 과정
 - 초록색은 원본 이미지의 input, 노란색 부분이 kernel

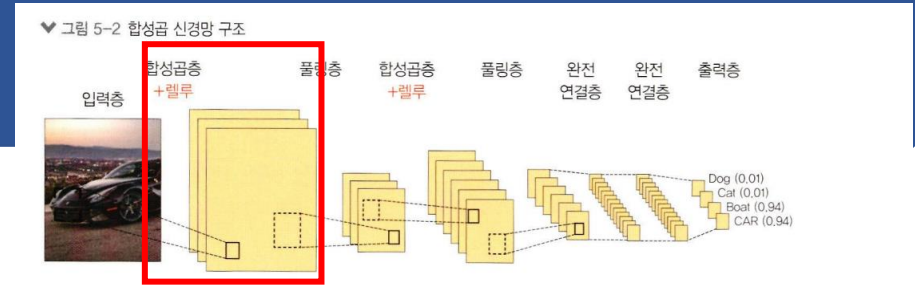
1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

[code implementation]
`nn.Conv2d(kernel_size = (3, 3))`



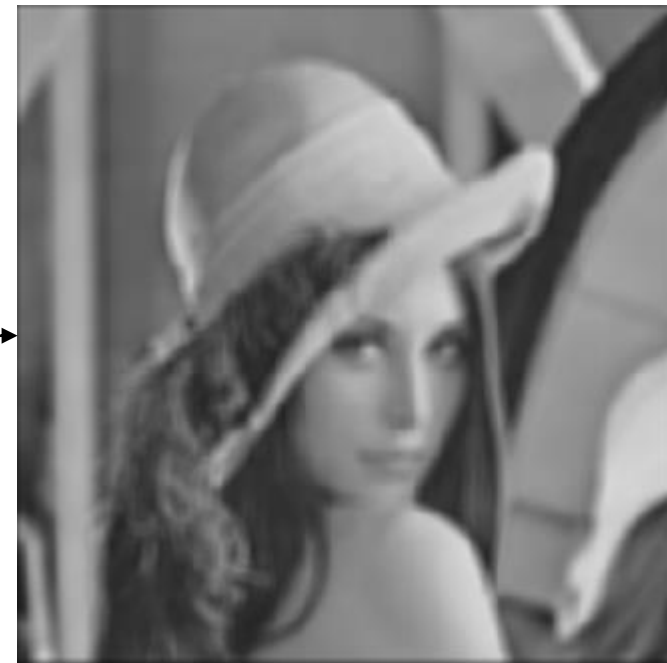
Convolution Neural Network 구조

- 2D Convolution
 - Image Smoothing (Blurring)에 사용되는 커널
 - Average filtering



Original

0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04



Blurring



Convolution Neural Network 구조

- 2D Convolution
 - Edge Detection



Original Image

1	0	-1
1	0	-1
1	0	-1



Vertical Edges

1	1	1
0	0	0
-1	-1	-1

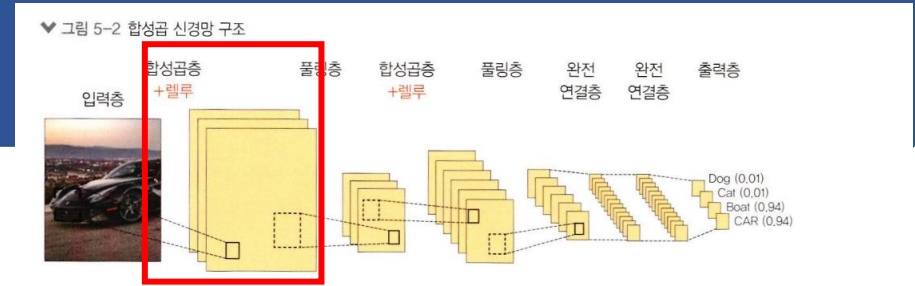


Horizontal Edges



Convolution Neural Network 구조

- 커널(Kernel)의 역할
 - 이미지의 특성을 추출하는 데 중요한 역할을 함
 - 이미지마다 적합한 커널은 모두 다름.
 - 커널에 들어가는 매개변수들은 딥러닝을 기반으로 학습되어짐.
 - 단, ANN을 이용해서 하는 것은 적합하지 않음.
 - 처음에 초기화를 시작한 후에 학습을 통해 최적의 커널 값을 찾아냄
 - 이미지의 여러 가지 특성을 추출하기 위해 커널의 개수를 하나가 아닌 여러 개를 쓸 수 있음.



Convolution Neural Network 구조

- 합성곱의 여러 가지 시나리오

- Input은 흑백 이미지이고, 3 x 3 커널이 2개인 경우
 - Input shape : 5 x 5 x 1, kernel shape : 3 x 3 x 2

1	1	1	1	1
2	2	2	2	2
0	0	0	0	1
1	0	1	1	0
2	1	2	1	2

Input = 5 x 5 x 1

*

1	1	1
0	0	0
1	0	1

Kernel1 = 3 x 3 x 1

1	0	1
1	0	1
1	0	1

Kernel2 = 3 x 3 x 1

6	3	3	4
6	8	7	7
6	4	2	5
6	5	0	

Output = 3 x 3 x 2

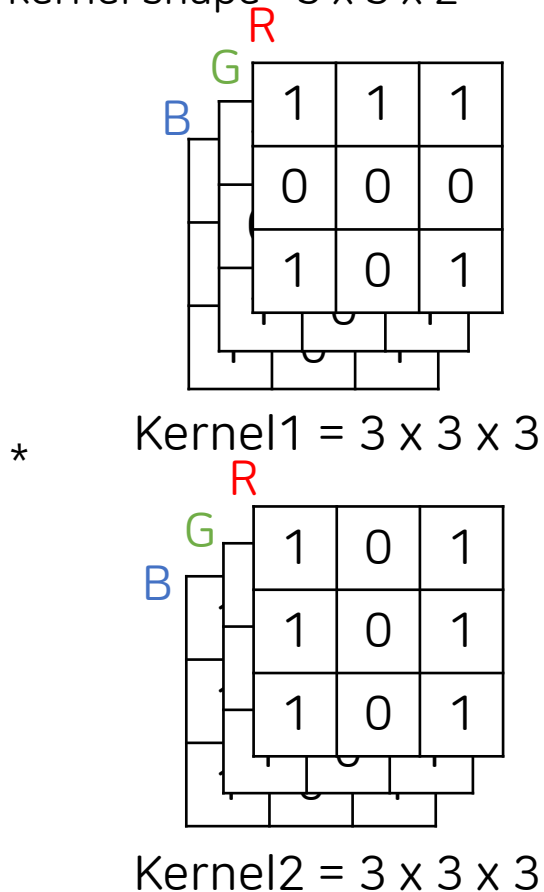
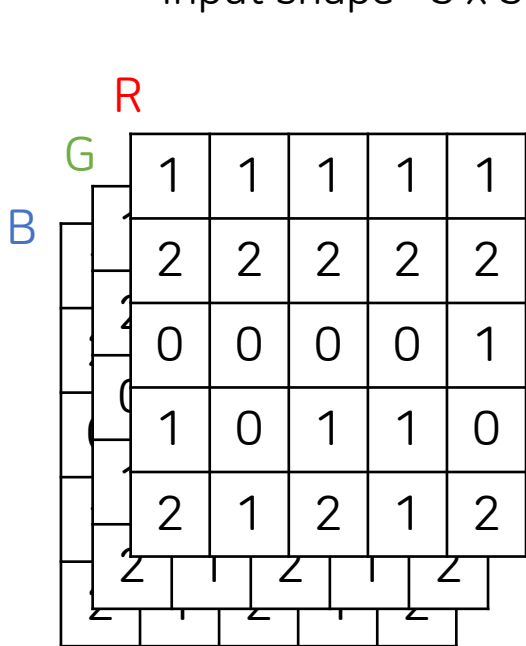


[code implementation]
`nn.Conv2d(in_channels = 1,
 out_channels = 2, kernel_size = (3, 3))`

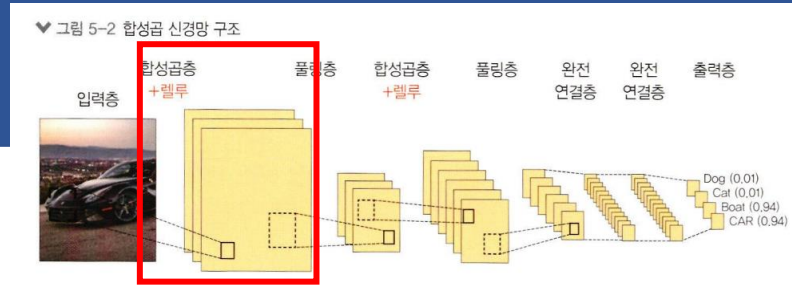
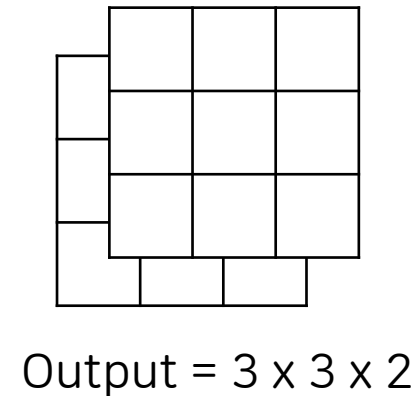
Convolution Neural Network 구조

- 합성곱의 여러 가지 시나리오

- Input은 컬러 이미지이고, 3 x 3 커널이 2개인 경우
 - Input shape : 5 x 5 x 3, kernel shape : 3 x 3 x 2



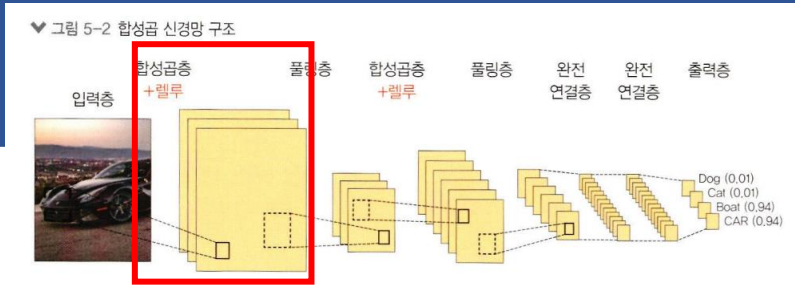
[code implementation]
`nn.Conv2d(in_channels = 3,
out_channels = 2, kernel_size = (3, 3))`



Convolution Neural Network 구조

- 합성곱의 여러 가지 시나리오

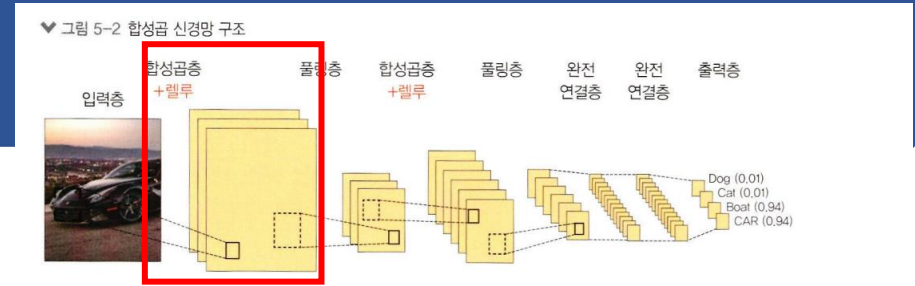
- Input은 컬러 이미지이고, 3 x 3 커널이 2개인 경우
 - Input shape : 5 x 5 x 1, kernel shape : 3 x 3 x 2



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1																									
2																									
3																									
4																									
5																									
6																									
7																									
8																									
9																									
10																									
11																									
12																									
13																									
14																									
15																									
16																									
17																									
18																									
19																									
20																									
21																									

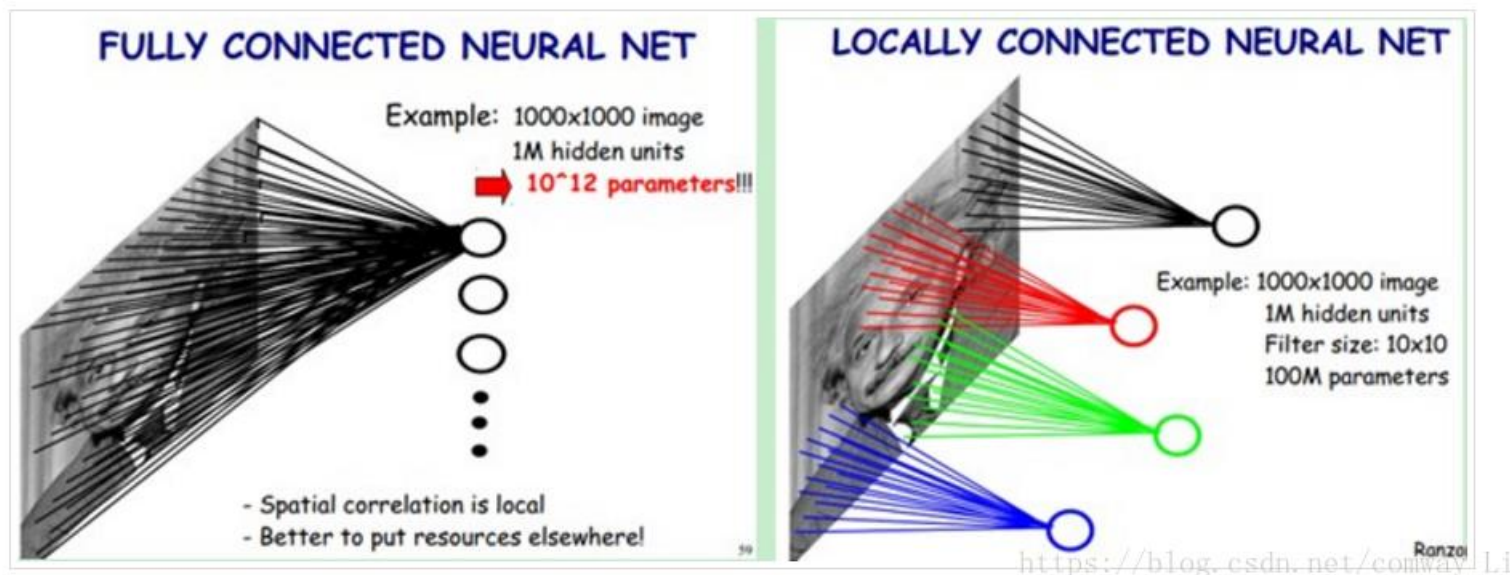
Convolution Neural Network 구조

- 합성곱 적용 시 이미지의 크기 변화(no padding, stride)
 - Input shape : (W, H, C)
 - W : width of the input, H : height of the input, C : input channels
 - Kernel : (w, h, C, D)
 - w x h : kernel size, C : input channels, D : number of the kernels (output의 채널 수와 동일)
 - Output : (W - w + 1, H - h + 1, D)
 - W - w + 1 : Width of the output, H - h + 1 : Height of the output, D : output channels
 - Number of the parameters : $(w \times h \times C + 1) \times D$



Convolution Neural Network의 특징

- Locality and Weight sharing
 - 이미지의 중요한 공간적인 정보를 커널(kernel)을 통해 추출할 수 있음 --- Locality
 - ANN에서는 fully-connected layer이라고 하며, CNN에서는 Locality-connected layer라고도 표현
 - 이미지의 공간적인 정보를 추출할 때, 같은 가중치를 이용함. --- weight sharing



Convolution Neural Network의 특징

- 패딩(Padding)

- 커널을 이용하여 합성곱(Convolution) 연산 시에 이미지의 크기가 줄어듦
 - 이미지의 크기가 줄어드는 것을 방지하기 위해 인위적으로 크기를 늘리는 작업을 함
- 이미지의 크기를 보존할 수 있다는 장점이 존재
- Zero padding : 주변을 0으로 채워넣음.
- 합성곱(Convolution) 연산을 하기 전, 패딩을 거친 다음에 합성곱(Convolution) 연산을 진행함.

1	0	1
0	1	0
1	0	1

1	1	1	1	1
2	2	2	2	2
0	0	0	0	1
1	0	1	1	0
2	1	2	1	2

Input = 5 x 5 x 1

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	2	2	2	2	2	0
0	0	0	0	0	1	0
0	1	0	1	1	0	0
0	2	1	2	1	2	0
0	0	0	0	0	0	0

Input = 7 x 7 x 1

3	5	5	5	3
3	4	4	4	3
2	6	5	5	4
2	4	3	6	1
2	3	3	2	3

output = 5 x 5 x 1

Convolution Neural Network의 특징

- 스트라이드(Stride)

- 합성곱 연산 시에 커널을 얼마나 움직일 것인지를 결정
 - 아무 말 하지 않으면 1
 - Feature map의 사이즈를 줄이는 데 영향을 줌

[code implementation]
`nn.Conv2d(in_channels = 1,
out_channels = 1, kernel_size = (3, 3),
padding = 1, stride = 2)`

- 합성곱 연산량이 많아지게 되면 시간이 오래걸리므로 stride를 조절해서 연산 진행

1	1	1	1	1
2	2	2	2	2
0	0	0	0	1
1	0	1	1	0
2	1	2	1	2

Input = 5 x 5 x 1

1	0	1
0	1	0
1	0	1

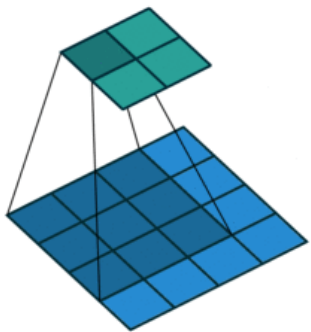
Kernel = 3 x 3 x 1

4	5
4	6

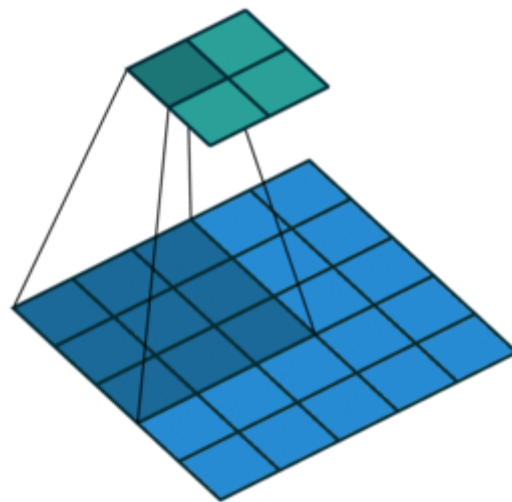
Output = 2 x 2 x 1

Convolution 연산

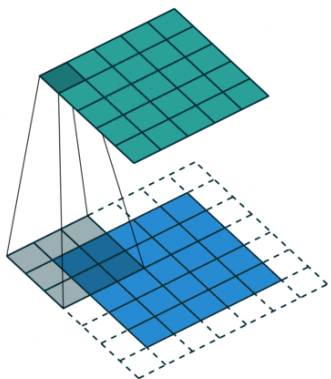
- Padding과 stride를 적용한 결과



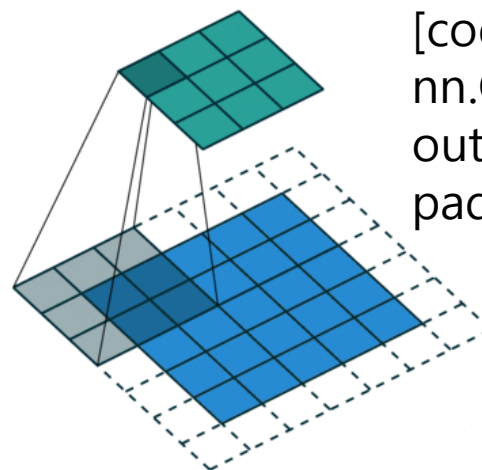
No padding, stride = 1



No padding, stride = 2



1 padding, stride = 1



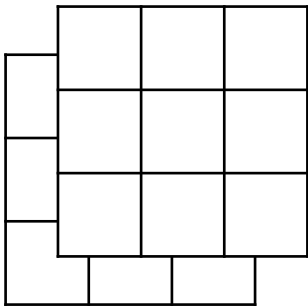
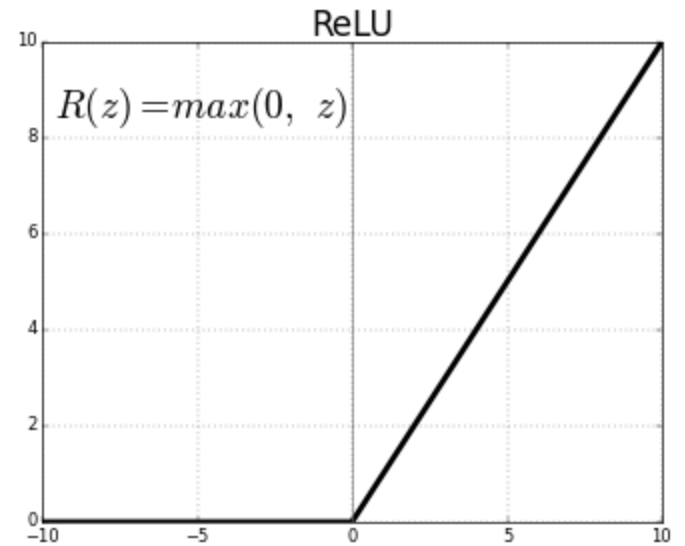
1 padding, stride = 2

[code implementation]
`nn.Conv2d(in_channels = 1,
out_channels = 1, kernel_size = (3, 3),
padding = 1, stride = 2)`

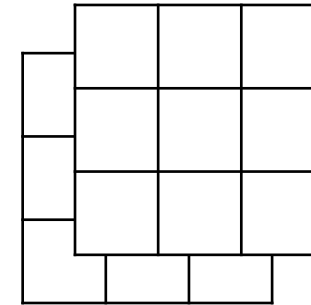
Convolution Neural Network의 특징

- Nonlinear Activation

- ANN에서와 마찬가지로 합성곱 연산이 끝난 후에는 활성화 함수를 적용함
 - 보통은 ReLU 함수를 많이 이용함
 - ReLU 함수 : 결과가 0보다 크면 그대로, 0보다 작으면 0으로 처리



Output = 3 x 3 x 2

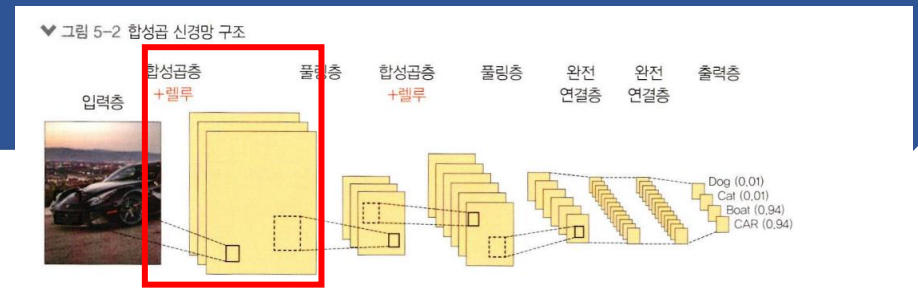


Output = 3 x 3 x 2

[code implementation]
`nn.Conv2d(in_channels = 3,
out_channels = 2, kernel_size = (3, 3),
padding = 1, stride = 2)
nn.ReLU()`

Convolution Neural Network 구조

- 합성곱 적용 시 이미지의 크기 변화(no padding, stride)
 - Input shape : (W, H, C)
 - W : width of the input, H : height of the input, C : input channels
 - Kernel : (w, h, C, D)
 - w x h : kernel size, C : input channels, D : number of the kernels (output의 채널 수와 동일)
 - Padding : P, Stride : S
 - Output : $((W + 2P - w) / S + 1, (H + 2P - h) / S + 1, D)$
 - $(W + 2P - w) / S + 1$: Width of the output, $(H + 2P - h) / S + 1$: Height of the output, D : output channels
 - Number of the parameters : $(w \times h \times C + 1) \times D$



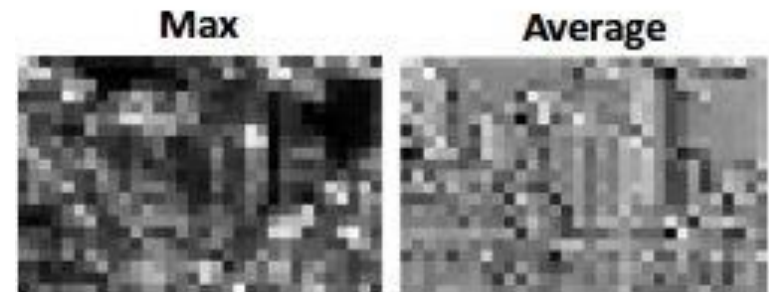
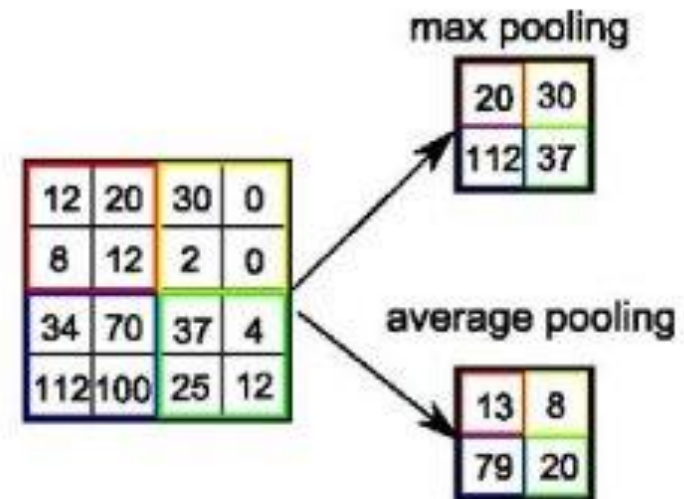
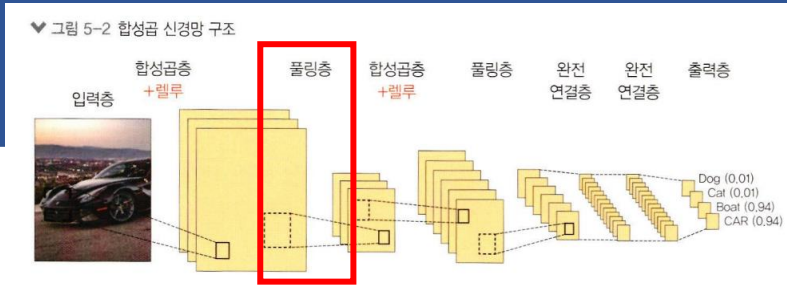
Convolution Neural Network 구조

- 풀링층(Pooling Layer)

- 합성곱층과 유사하게 특성 맵의 차원을 다운 샘플링하여 연산량을 감소
 - 주요한 특성 벡터(Feature vector)를 추출하여 학습을 효과적으로 할 수 있게 함
 - 주변의 특성들을 요약하는 역할에 비유 가능
- 최대 풀링(max pooling)과 평균 풀링(average pooling)
 - 최대 풀링(max pooling) : 대상 영역에서 최댓값을 추출
 - 평균 풀링(average pooling) : 대상 영역에서 평균을 반환

[code implementation]

```
nn.Conv2d(in_channels = 3, out_channels  
= 2, kernel_size = (3, 3), padding = 1,  
stride = 2)  
nn.ReLU()  
nn.MaxPool2d(kernel_size = 2, stride = 2)
```



Convolution Neural Network 구조

- 풀링층(Pooling Layer)을 적용한 후의 크기 변화

- Input shape : (W, H, C)

- W : width of the input, H : height of the input, C : input channels

- 하이퍼 파라미터

- 필터의 크기 (F)

- 스트라이드 (S)

- Output : $((W - F) / S + 1, (H - F) / S + 1, D)$

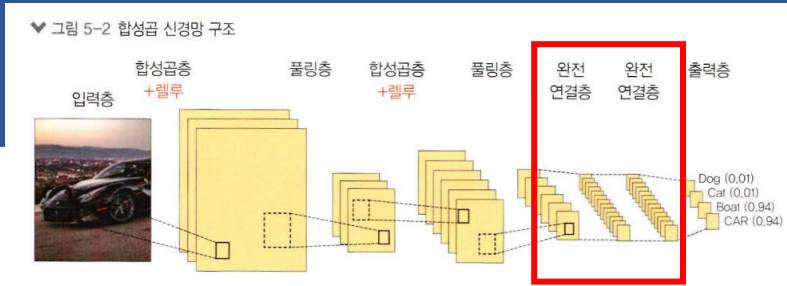
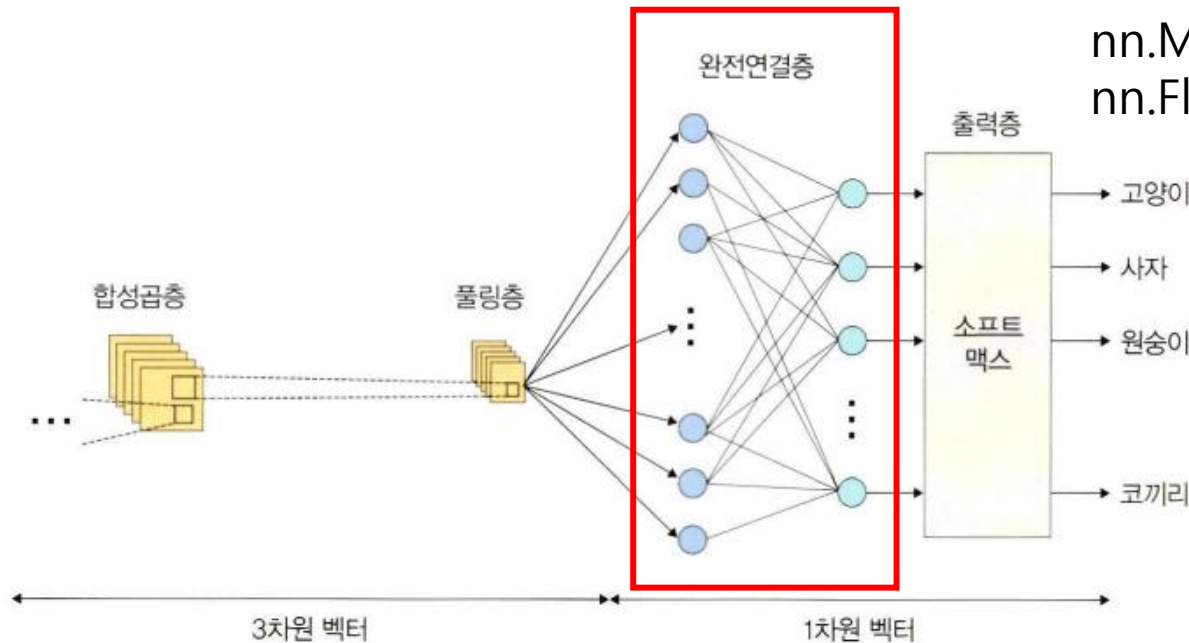
- $(W-F)/S + 1$: width of the output, $(H-F)/S + 1$: height of the output, D : output channels



Convolution Neural Network 구조

- 완전연결층(fully connected layer)

- 합성곱층(Convolution Layer)와 풀링층(Pooling layer)를 거치면서 차원이 축소
- 차원 축소가 된 후, 완전연결층으로 전달
 - 이 과정을 Flatten이라고 함
 - Ex. 분류 문제를 푸려면 ANN의 형태로 바꿔준 다음에 전달해야 함.



[code implementation]

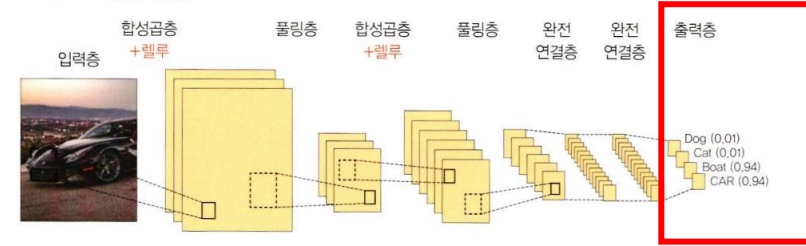
```
nn.Conv2d(in_channels = 3, out_channels  
= 2, kernel_size = (3, 3), padding = 1,  
stride = 2)  
nn.ReLU()  
nn.MaxPool2d(kernel_size = 2, stride = 2)  
nn.Flatten()
```

Convolution Neural Network 구조

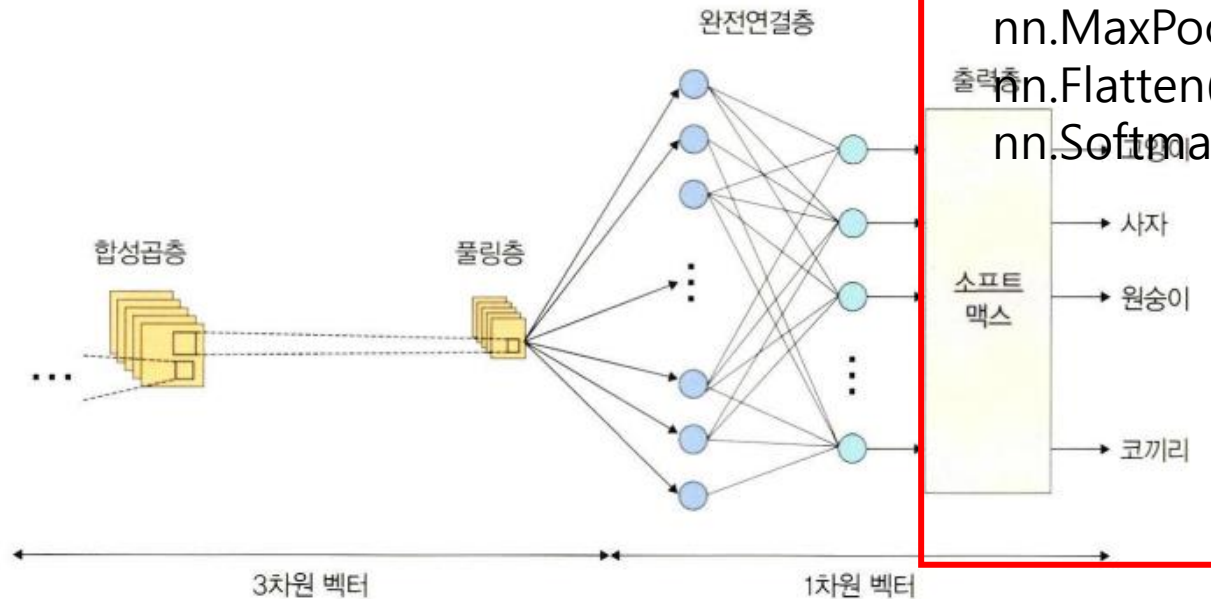
- 출력층(output layer)

- 소프트맥스 활성화 함수를 이용하여 입력 받은 값을 0 ~ 1 사이의 값으로 출력
- 소프트맥스 활성화 함수를 적용한 결과는 각 레이블에 속할 확률 값이 출력
 - 가장 높은 확률 값을 갖는 레이블이 최종 값으로 선정

그림 5-2 합성곱 신경망 구조



```
nn.Conv2d(in_channels = 3, out_channels = 2, kernel_size = (3, 3), padding = 1, stride = 2)
nn.ReLU()
nn.MaxPool2d(kernel_size = 2, stride = 2)
nn.Flatten()
nn.Softmax()
```



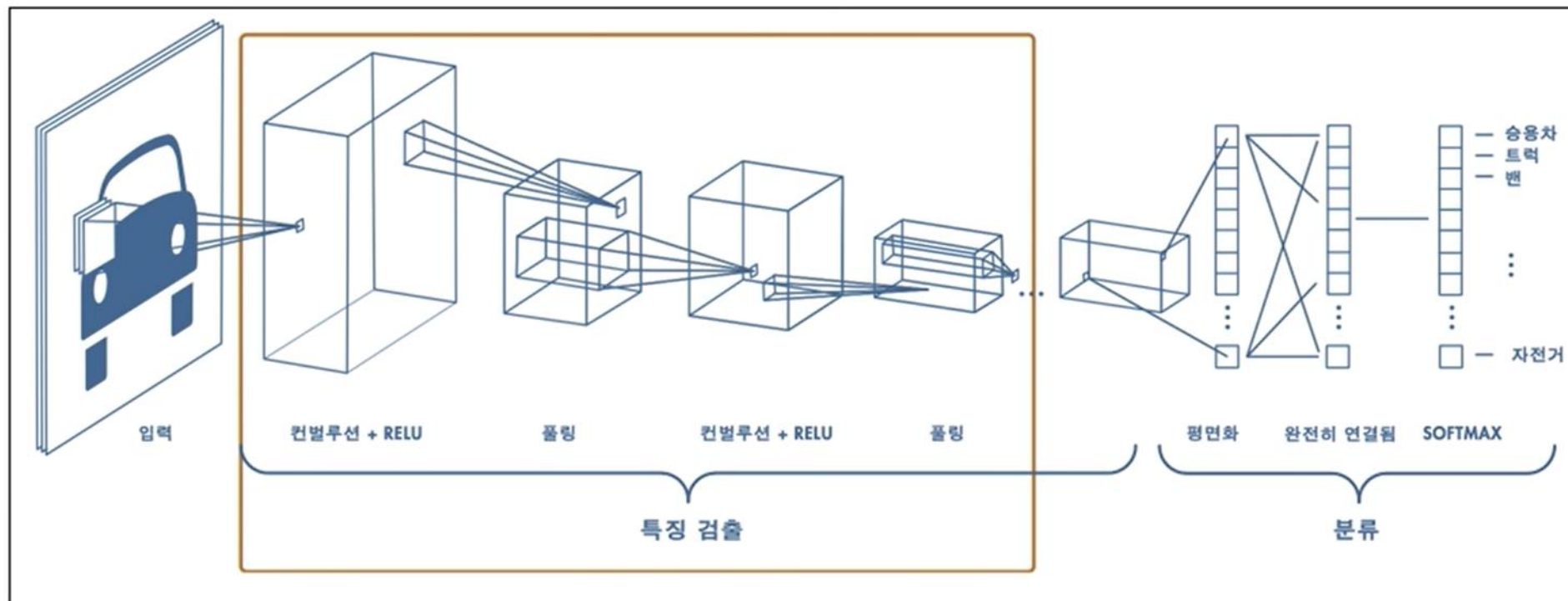
Convolution Neural Network 구조

- Example
 - Input size : (128, 128, 3)
 - Kernel size : 3 x 3 x 3 with 8 filters
 - Padding size : 1
 - Stride : 2
 - 컨볼루션 연산 진행 시 결과는? 파라미터의 개수는?

Convolution Neural Network 구조

- 전체적인 구조

이미지 영상인식의 혁명과도 같은 CNN



Transfer Learning

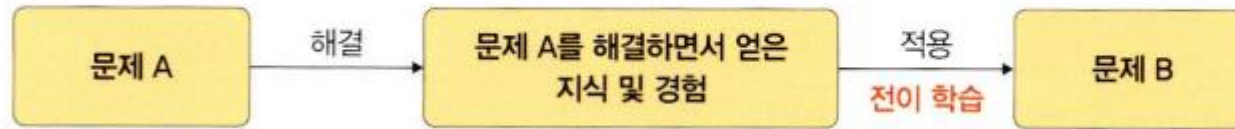
- 전이 학습(Transfer Learning)의 대두
 - 이미지넷(ImageNet)의 분류 문제
 - 1000개의 클래스, 1억 2500만개의 이미지(1억 2000만개의 training과 500만개의 validation)
 - Training : 138GB, Validation : 6GB
 - 의문점
 - 대용량 이미지를 가지고 충분히 학습할 수 있는 환경이 주어지는가?
 - 만약 1000개의 클래스에 포함되지 않은 다른 분류 문제가 주어진다면 해결할 수 있는가?
 - 해결책
 - 이전에 학습시킨 모델 자체를 그대로 사용해서 다른 문제에 풀어보는 것은 어떨까?

Transfer Learning

- 전이 학습(Transfer Learning)

- 아주 큰 데이터셋을 써서 훈련된 모델의 가중치를 그대로 가져와 우리가 해결하려는 과제에 맞게 보정
 - 여기서 언급한 아주 큰 데이터셋을 사용하여 훈련된 모델을 사전 훈련된 모델(pre-trained model)이라고 함

▼ 그림 5-29 전이 학습



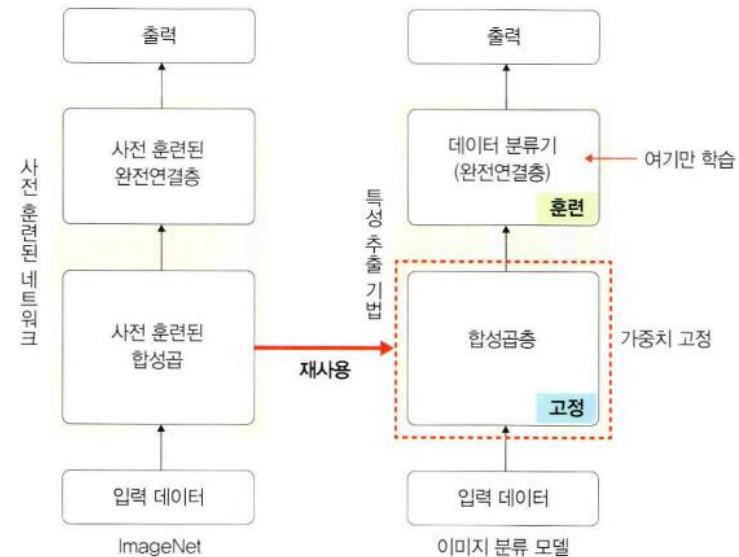
- 예시 : 의자에 대한 이미지 분류 문제

- 이미지넷 데이터셋에는 의자와 관련된 정보가 없음
- 사전 학습 모델이 일반적인 이미지의 특징을 충분히 파악했다면, 기존의 이미지로 모서리나 모양과 같은 큰 틀 확인 가능

Transfer Learning

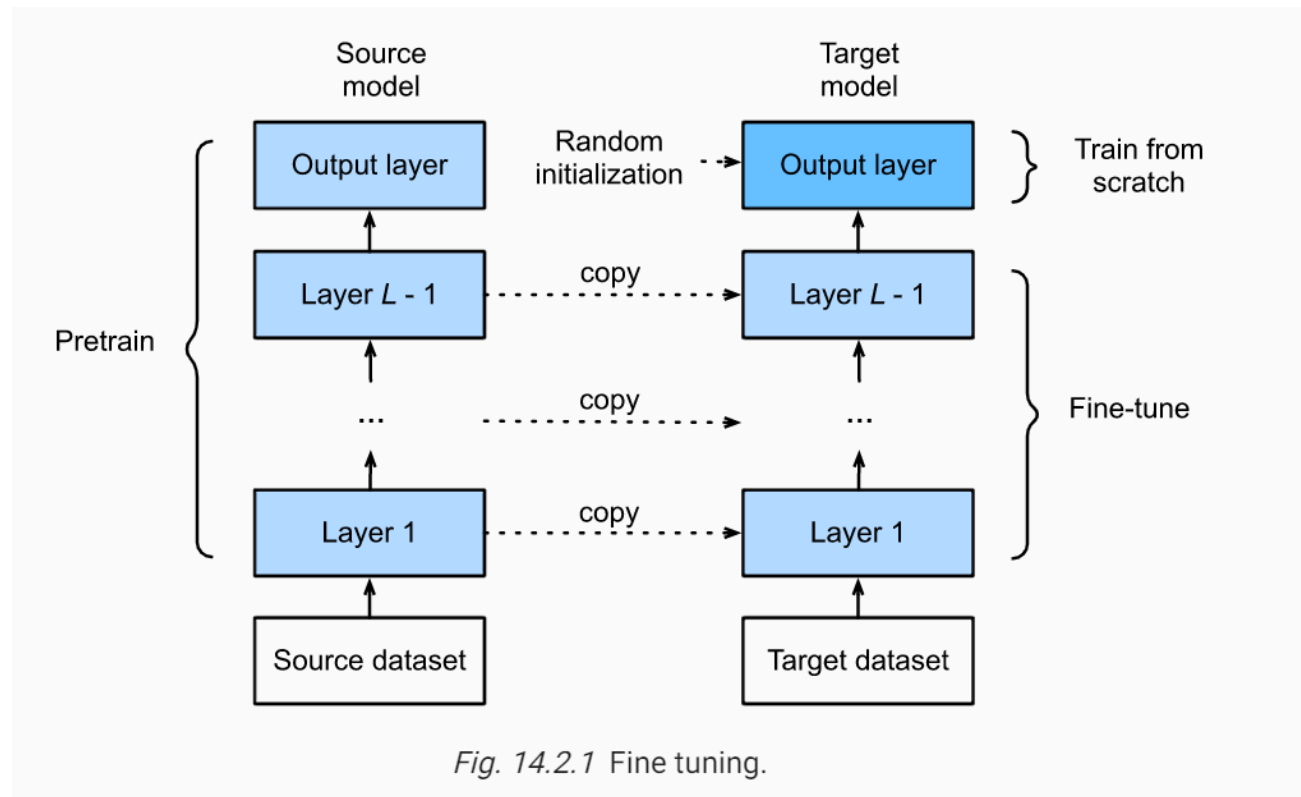
- 특성 추출(feature extractor)
 - 사전 훈련된 모델을 가져온 후 마지막에 완전연결층 부분만 새로 만듦
 - 즉, 학습할는 경우에는 마지막 완전연결층(이미지의 카테고리를 결정하는 부분)만 학습하고 나머지는 학습 X
 - Xception, Inception V3, ResNet50, VGG16, VGG19, MobileNet이 있음

▼ 그림 5-30 특성 추출 기법



Transfer Learning

- 미세 조정(fine-tuning) 기법
 - 특성 추출(feature extractor) 기법에서 더 나아가 사전 훈련된 모델과 합성곱층, 데이터 분류기의 가중치를 모두 update하여 훈련시키는 방식



Transfer Learning

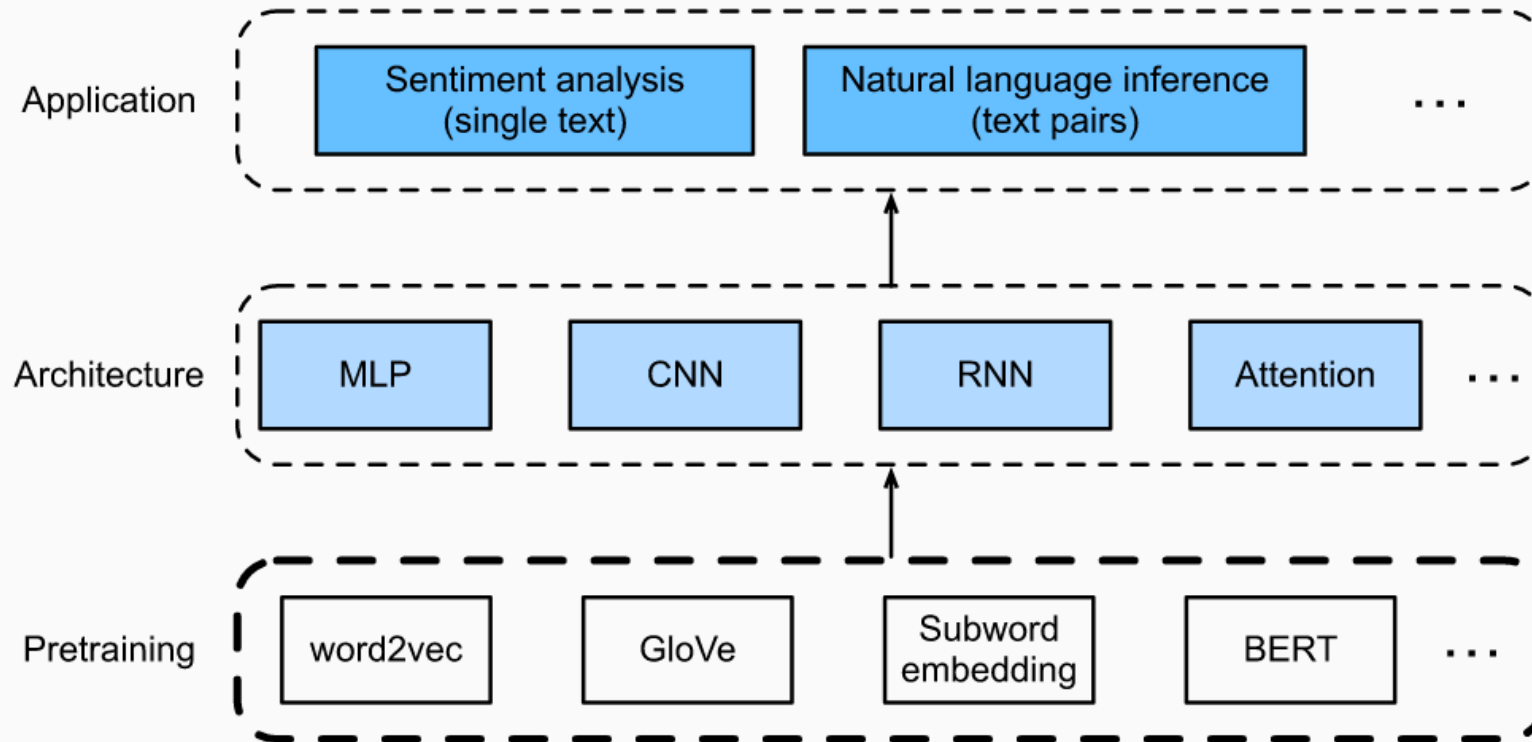


Fig. 15.1 Pretrained text representations can be fed to various deep learning architectures for different downstream natural language processing applications. This chapter focuses on the upstream text representation pretraining.