/* elice */

양재 AI School 인공지능 캠프

합성곱 (Convolution)과 합성곱 신경망 (CNN)

박상수 선생님

커리큘럼

1 **합성곱 (Convolution)**

합성곱 (Convolution)의 의미, 배경지식에 대해서 파악합니다.

2 **합성곱 신경망 (Convolutional Neural Network)**

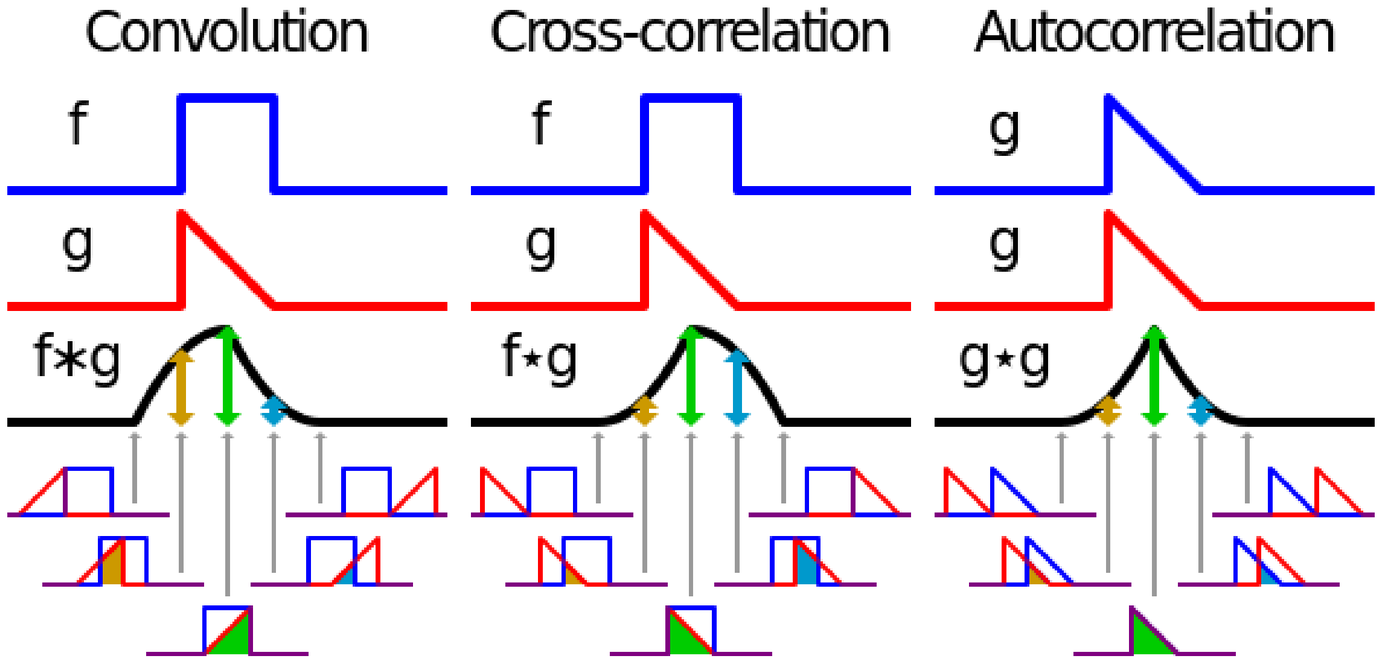
합성곱 신경망의 역사와 구조에 대해서 학습합니다.
합성곱 신경망의 대표적인 모델인 LeNet-5에 대해 파악합니다.

목차

합성곱 (Convolution)

합성곱, 컨볼루션, 회선 ???

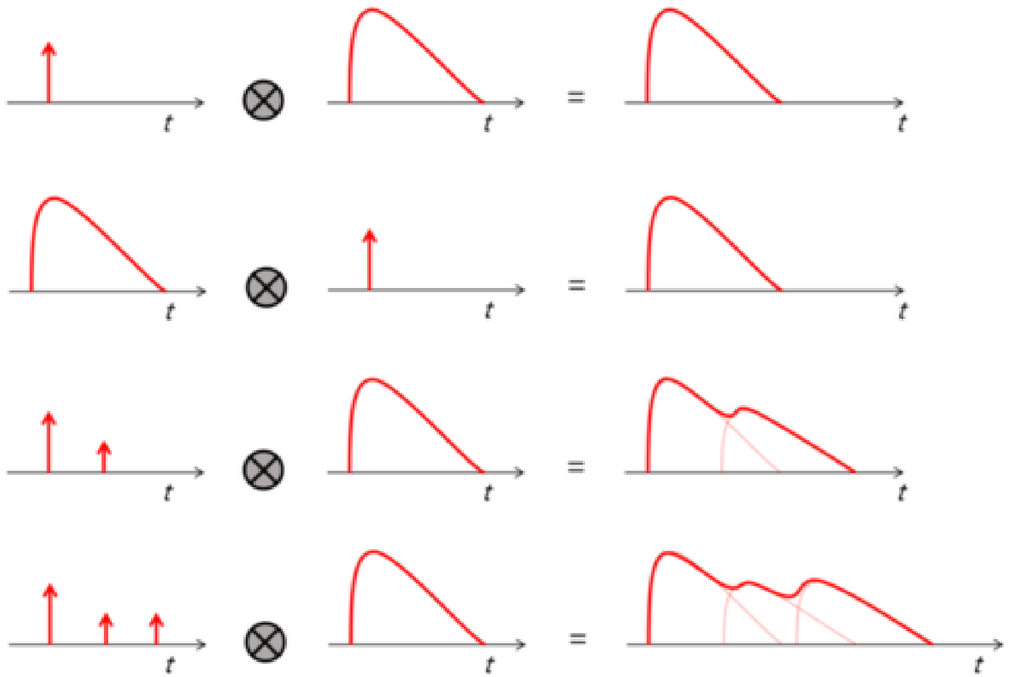
컨볼루션 (Convolution)



$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)\,(t-\tau)d\tau$$

하나의 함수와 또 다른 함수를 반전 이동한 값을 곱한 다음,
구간에 대해 적분하여 새로운 함수를 구하는 수학 연산

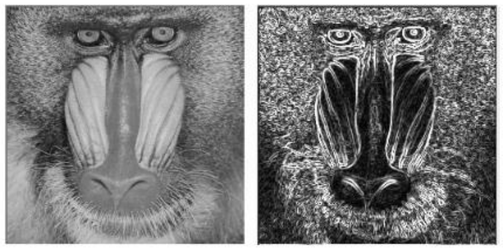
컨볼루션 (Convolution)



이미지와 컨볼루션 필터를 곱해서
새로운 이미지를 얻는 과정

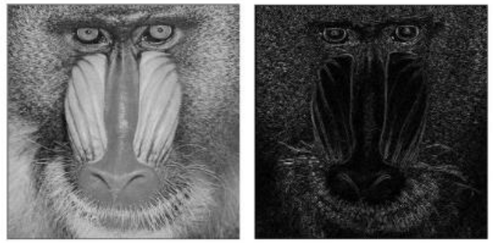
컨볼루션 (Convolution): 이미지

-1	-1	0
2	1	2
-1	-2	-1



소벨 마스크

2	-1	-1
-1	1	2
-2	0	2



로버트 마스크

2	1.2	-1
1	0	-1
1	-2	-1

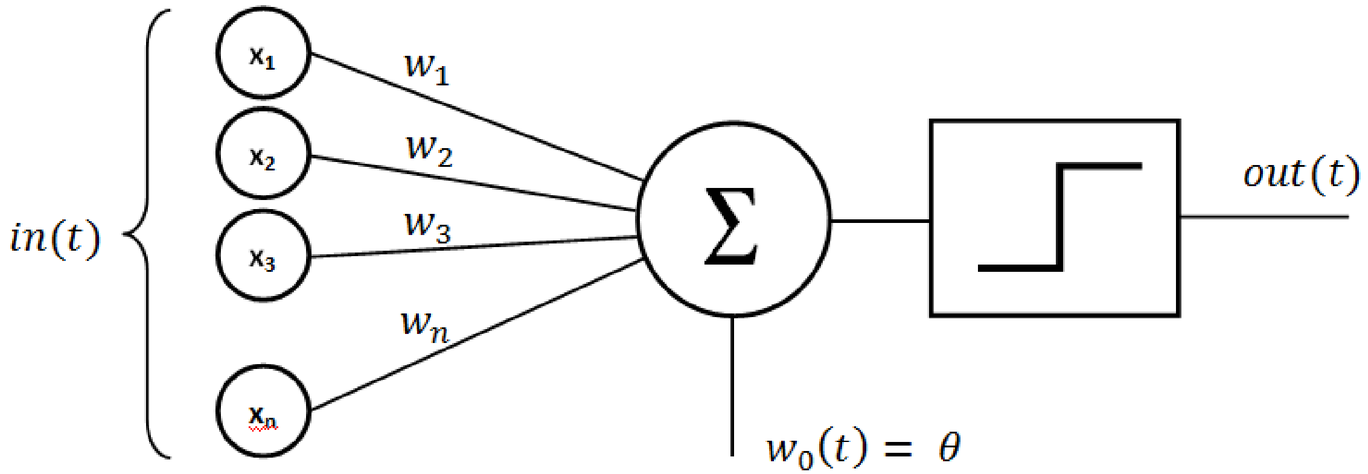
프라윗 마스크

필터의 종류에 따라 얻어지는 이미지가 상이함
(다른 특징을 추출)

합성곱 신경망 역사 (CNN)

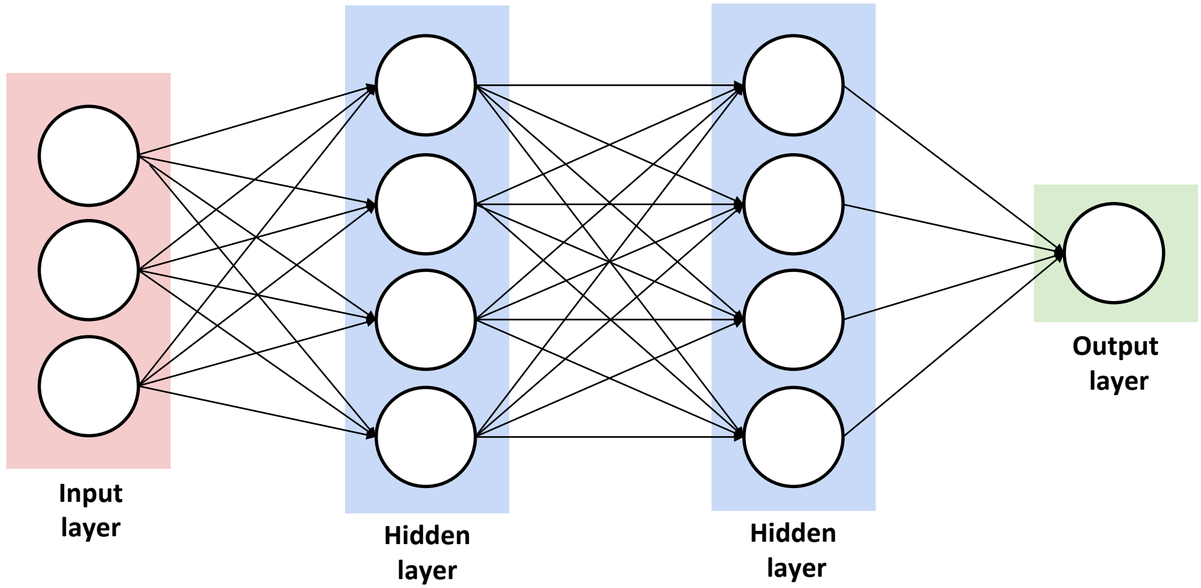
인간의 시신경 매커니즘을 모사한 구조

퍼셉트론



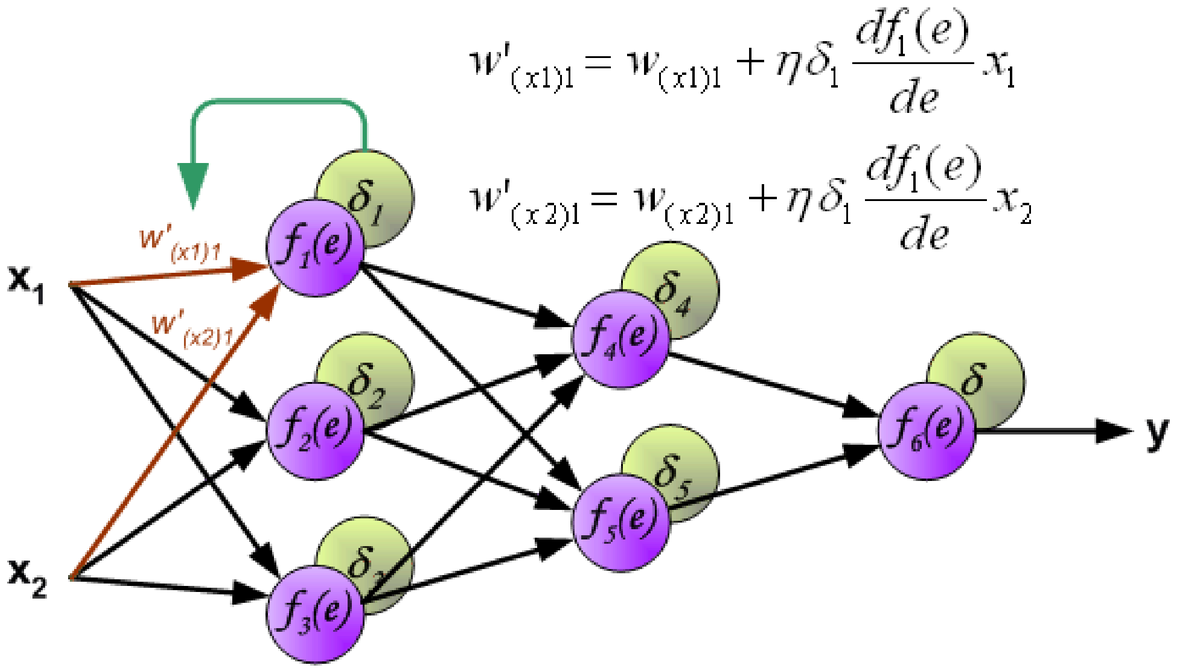
생물체 (인간)의 신경 세포(뉴런)을 모사한 구조

다층 퍼셉트론



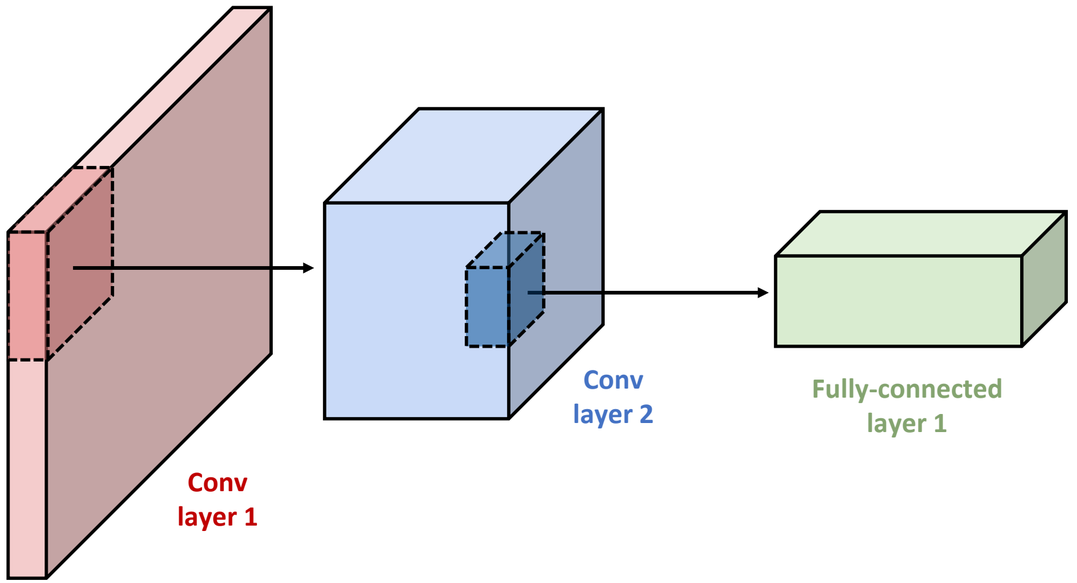
여러 레이어를 쌓아 더 많은 문제를 해결하는 모델

Back-propagation (학습 방법)



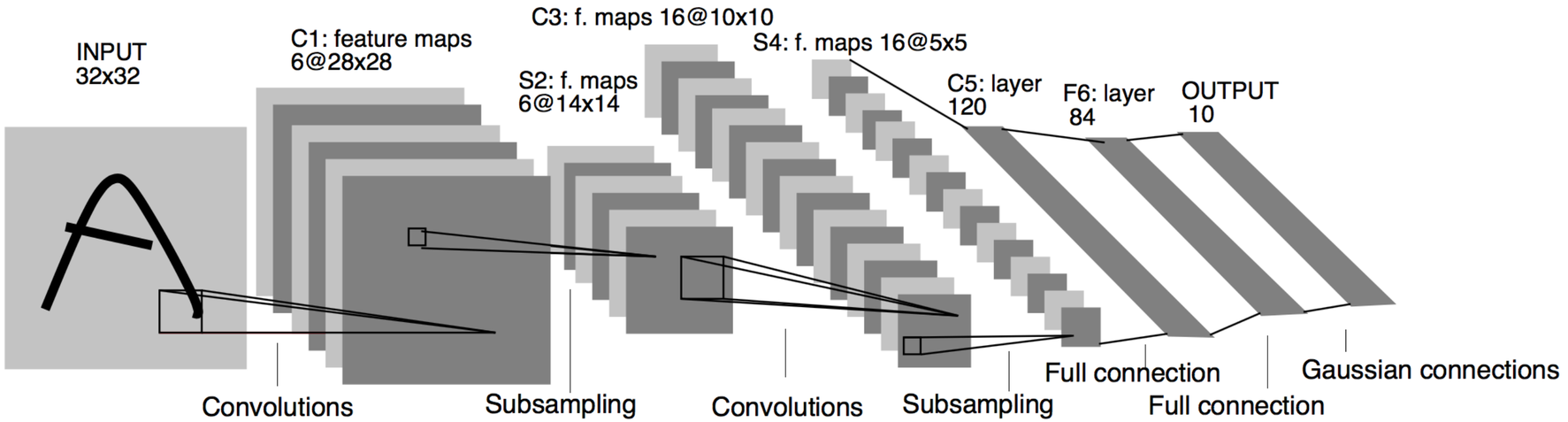
깊이가 깊은 신경망의 학습 방법

합성곱 신경망 (CNN)



시각적인 정보를 인식하는 신경망

최초의 합성곱 신경망 (LeNet-5)

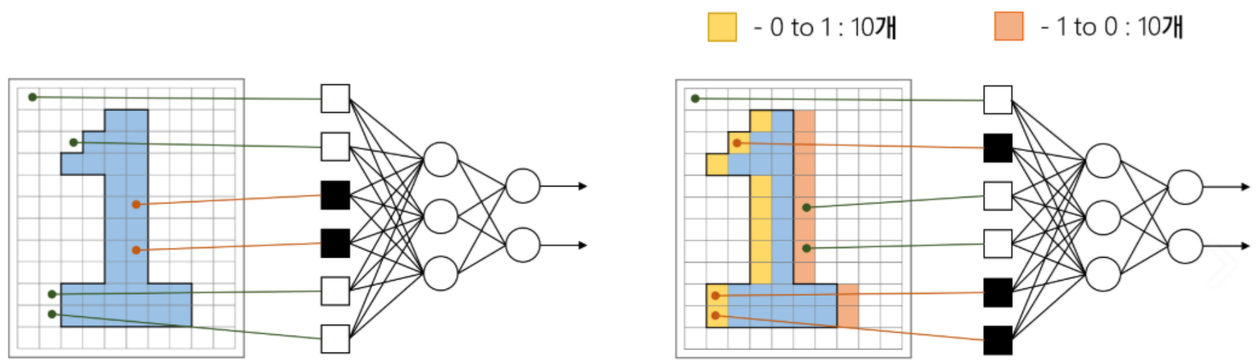


“Gradient-Based Learning Applied to Document Recognition”
현대 CNN의 시초가 되는 Neural Network 구조
우체국 수표인식 시스템에 적용

합성곱 신경망

인간의 시신경 매커니즘을 모사한 구조

기존 MLP의 문제점

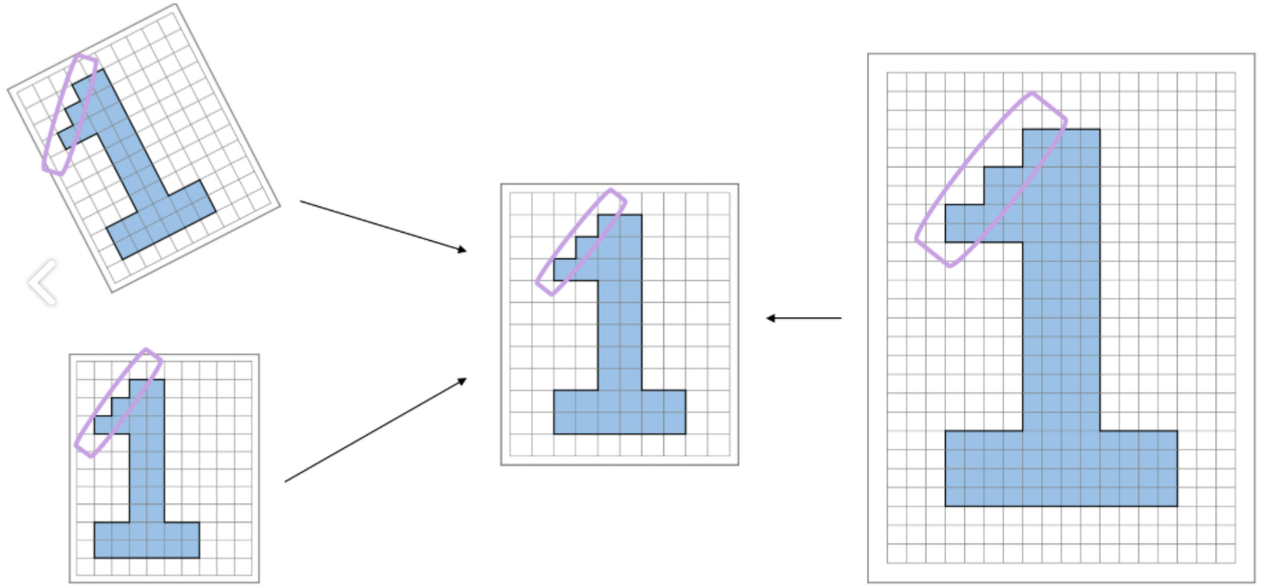


한 칸씩만 움직였는데

변화하는 인풋값이 20개

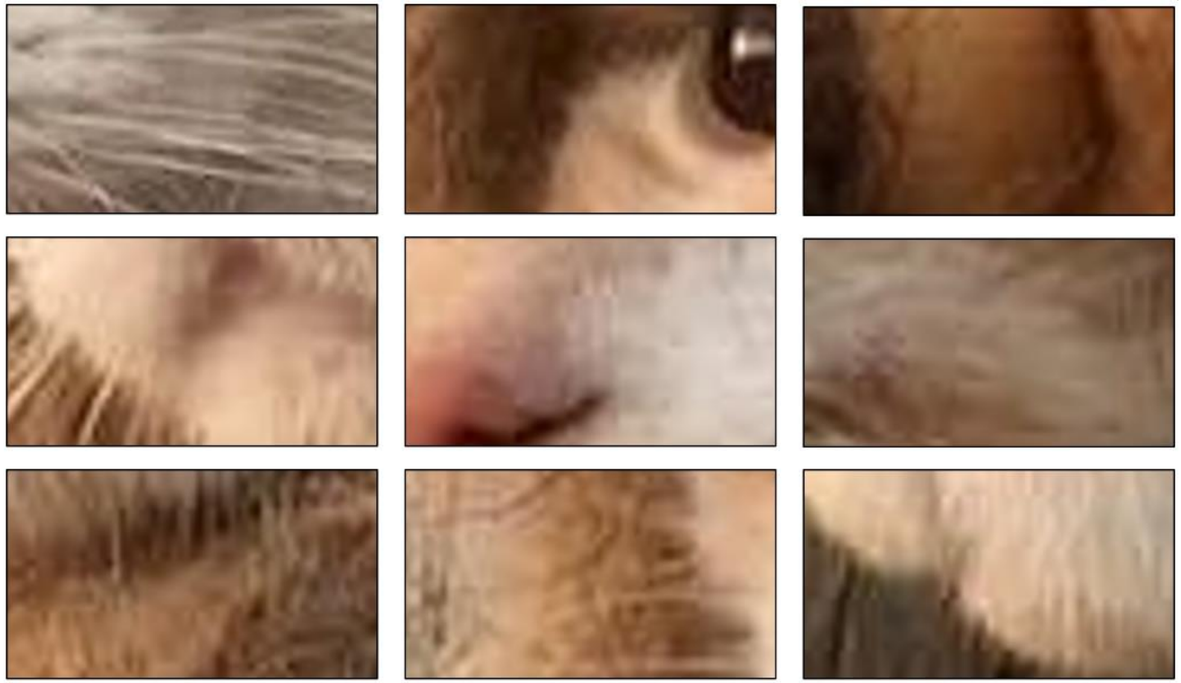
MLP 기반의 신경망: 입력 이미지의 크기와 동일한 크기
예) 32*32 크기의 이미지: 신경망 입력 크기는 32*32
만약에 이미지에 변화가 있다면 ?

기존 MLP의 문제점



이미지의 모든 변화에 신경망이 대응하도록 하는 것은 불가능함
만약에 이미지가 전체가 변한다면 다시 학습을 해야 함

특징을 사용한 합성곱 신경망



1단계: 라인 (가로, 세로, 대각선), 동그라미, 세모

특징을 사용한 합성곱 신경망

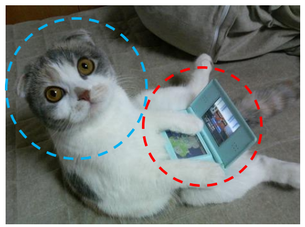
2단계: 눈, 코 귀, 발

특징을 사용한 합성곱 신경망

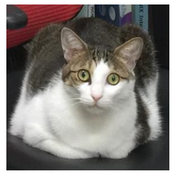
3단계: 사자같은 고양이

합성곱 신경망

입력 이미지



특징



분류

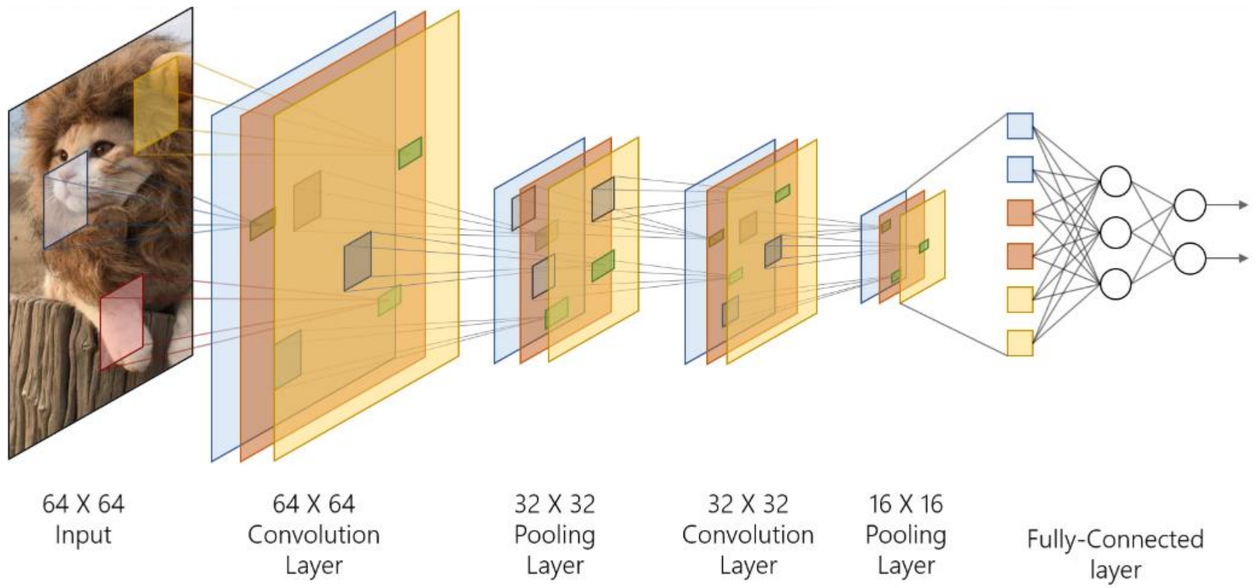
고양이

치타

3DS

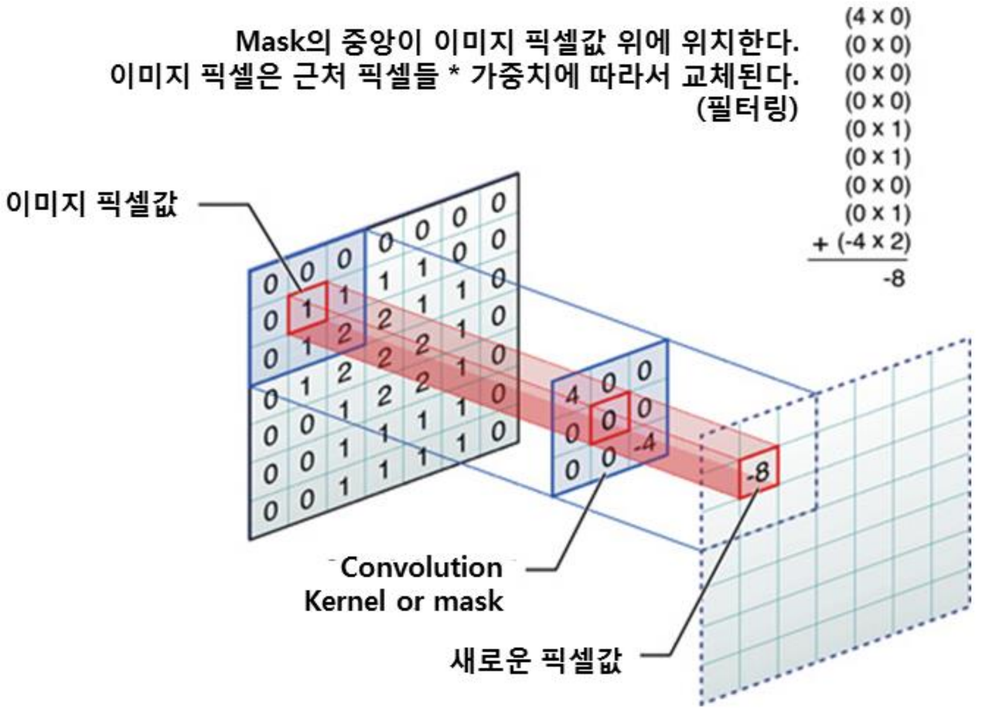
입력이미지의 특징을 추출, 분류하는 과정으로 동작

합성곱 신경망



특징 추출(Convolution, Pooling), 분류 (Fully-connected)

컨볼루션: 요소별 연산

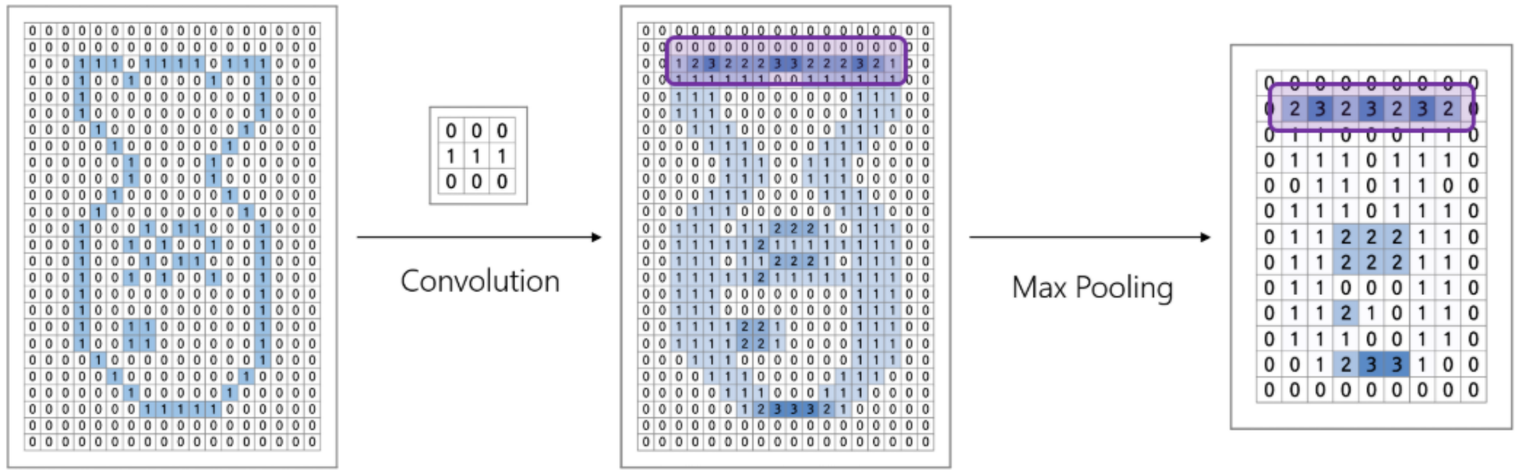


컨볼루션 커널과 입력 이미지의 요소별 연산 (Element-wise)

여러 개의 값을 곱하여 하나의 합으로 계산

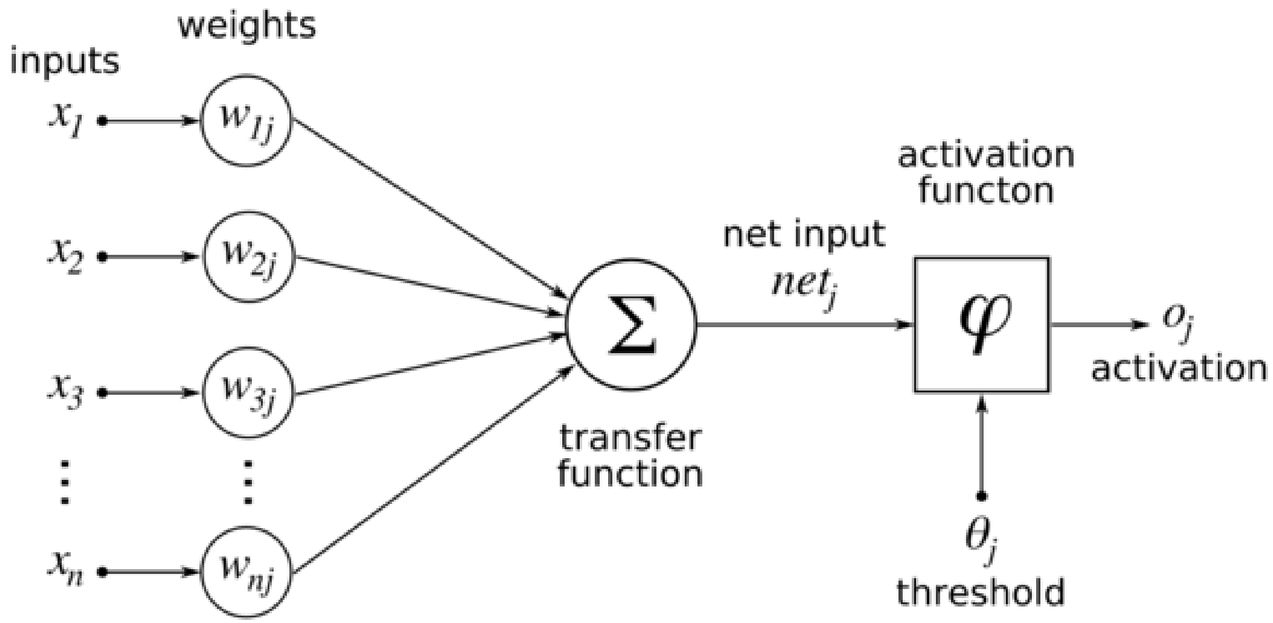
Result = $W_1*X_1 + W_2*X_2 + \cdots\cdots + W_9*X_9$

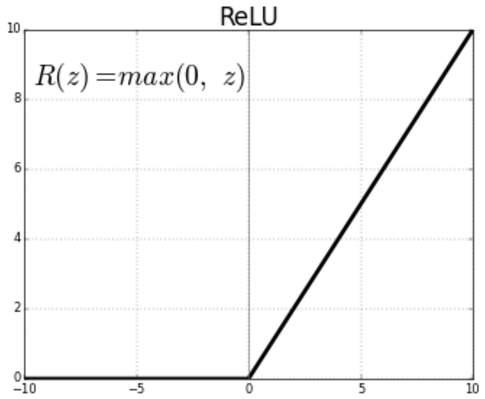
컨볼루션: 요소별 연산



컨볼루션: 이미지에서 어떠한 특징이 있는 지를 구하는 과정
필터가 이미지를 이동, 새로운 이미지 (피쳐맵)를 생성
컨볼루션 연산 이후에 지느러미 부분의 특징이 두드러짐

컨볼루션: 활성화 함수





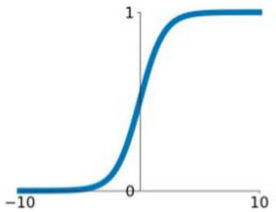
요소별 연산 이후, 다음 뉴런으로 값의 전달 여부 결정

Rectified Linear Unit (ReLU): 값이 0보다 크면 그대로, 아니면 0

컨볼루션: 활성화 함수

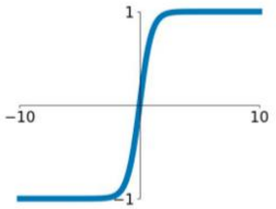
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



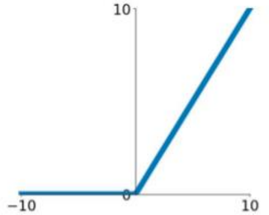
tanh

$$\tanh(x)$$



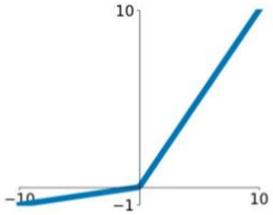
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

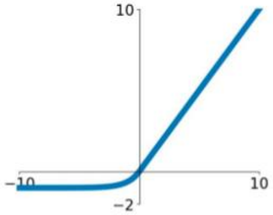


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

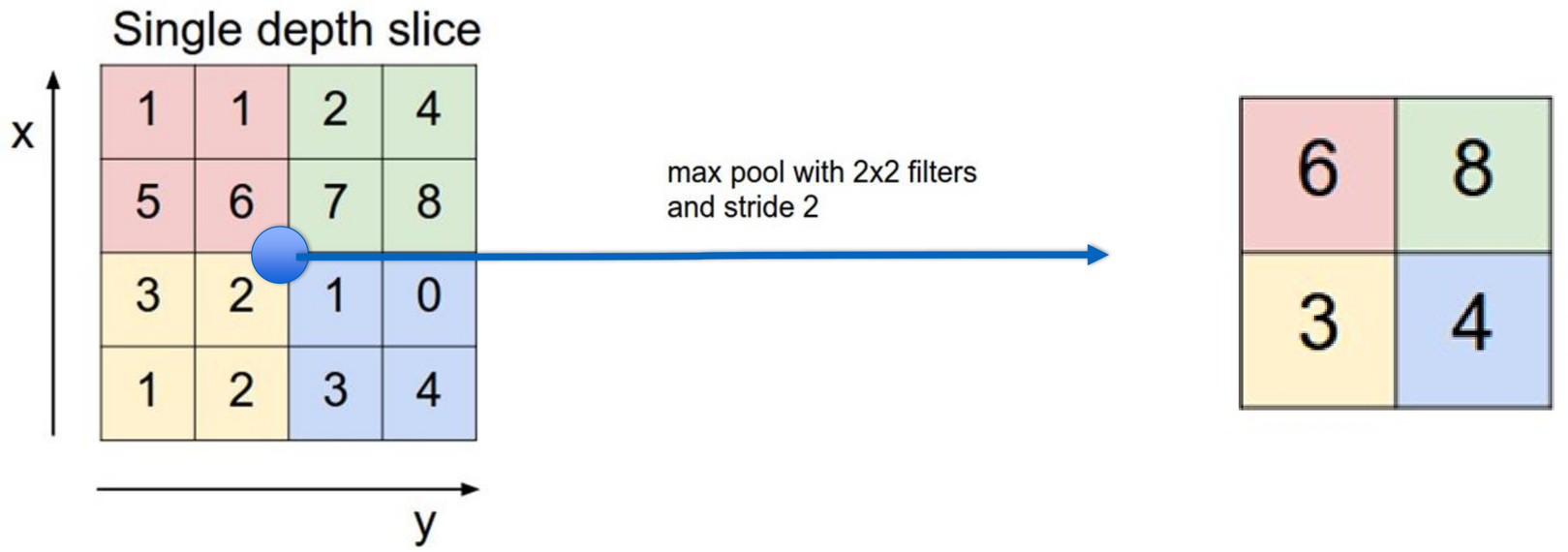
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



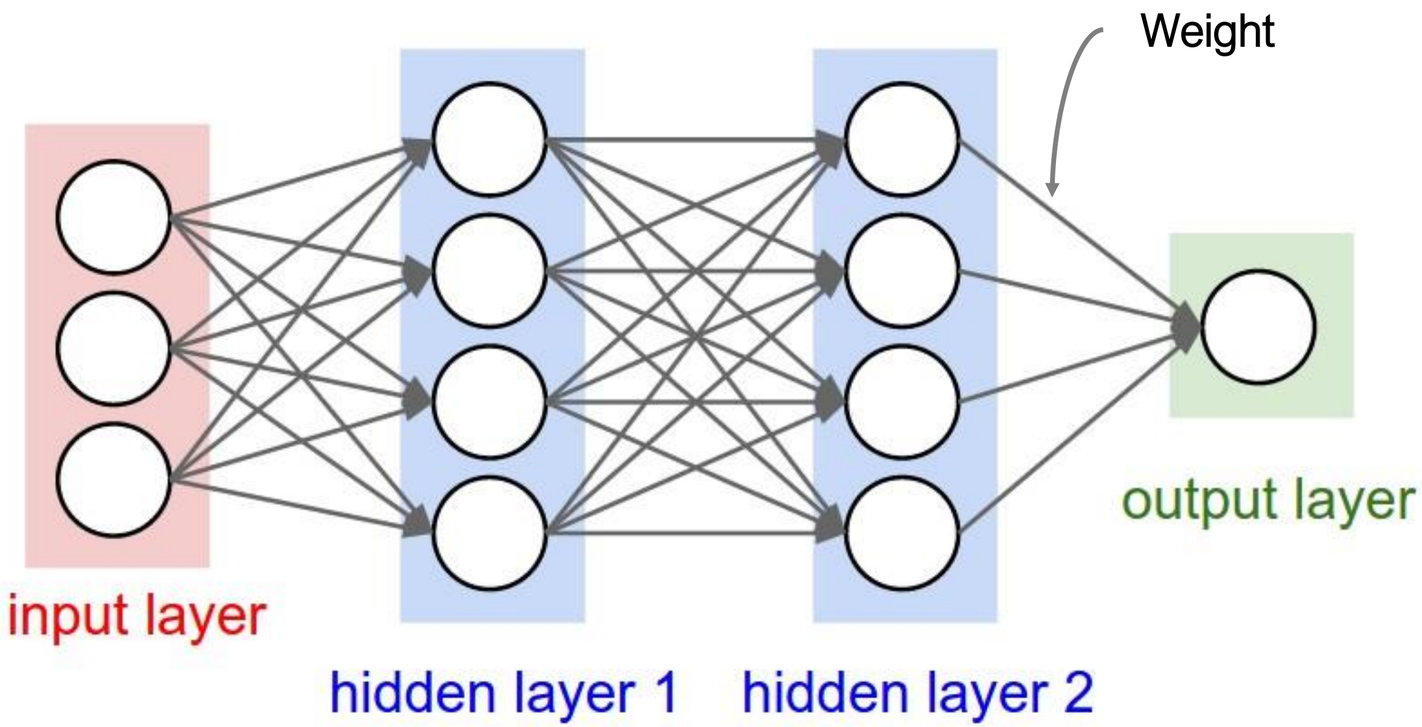
신경망 특성, 구조에 따라서 적합한 활성화 함수가 존재함

풀링 (서브샘플링)



풀링은 이미지의 왜곡의 영향 (노이즈)을 축소하는 과정
노이즈를 줄이기 위해 대표적인 값을 사용
Max (가장 큰 값만 사용), Average (뉴런 그룹의 평균값)

분류 (MLP)

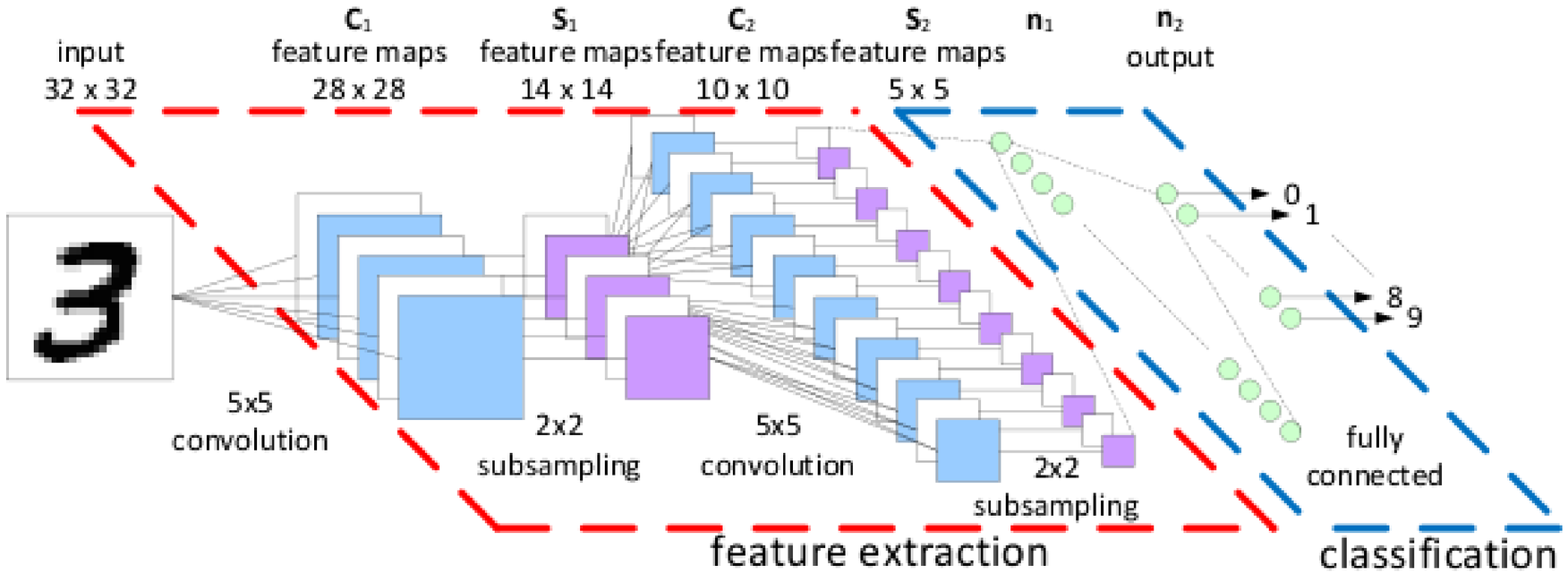


추출된 특징을 사용하여 이미지를 분류
분류 과정에서 얻은 값과 분류 클래스를 대조하여
가장 확신이 되는 분류를 분류 결과로 출력

LeNet-5

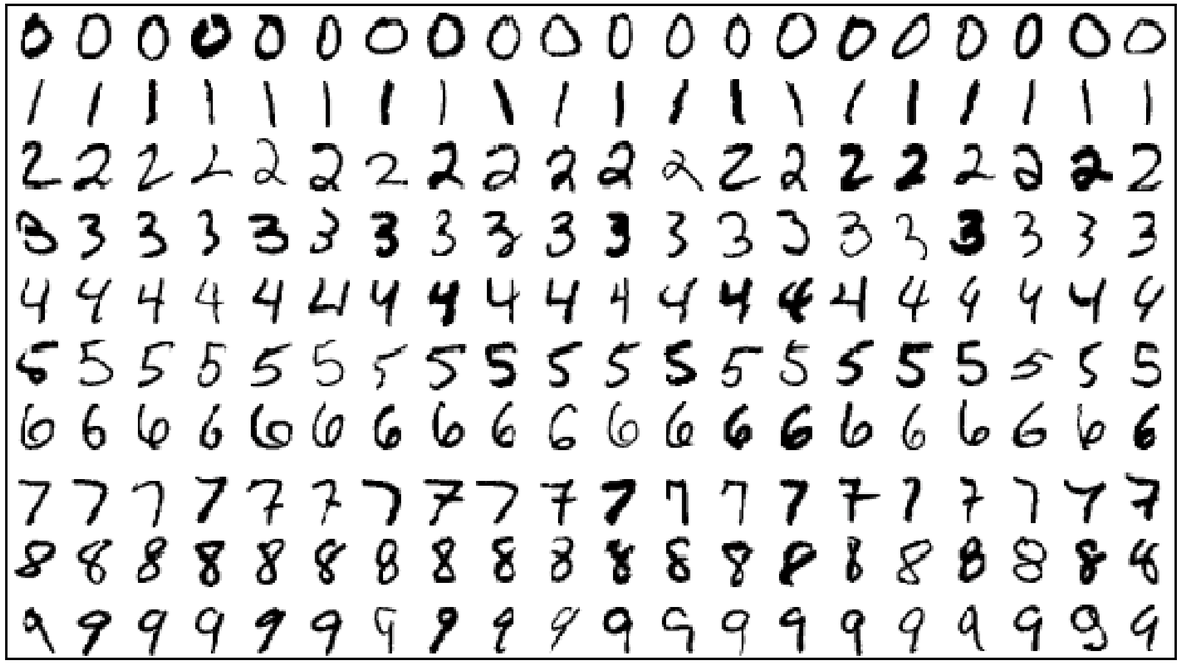
합성곱 신경망의 시작 !

LeNet-5



입력: 32*32 크기의 숫자 이미지 (MNIST)
우체국의 우편번호 인식을 위해 고안된 신경망
Feature extraction (컨볼루션, 풀링), Classification (분류기)

MNIST 벤치마크

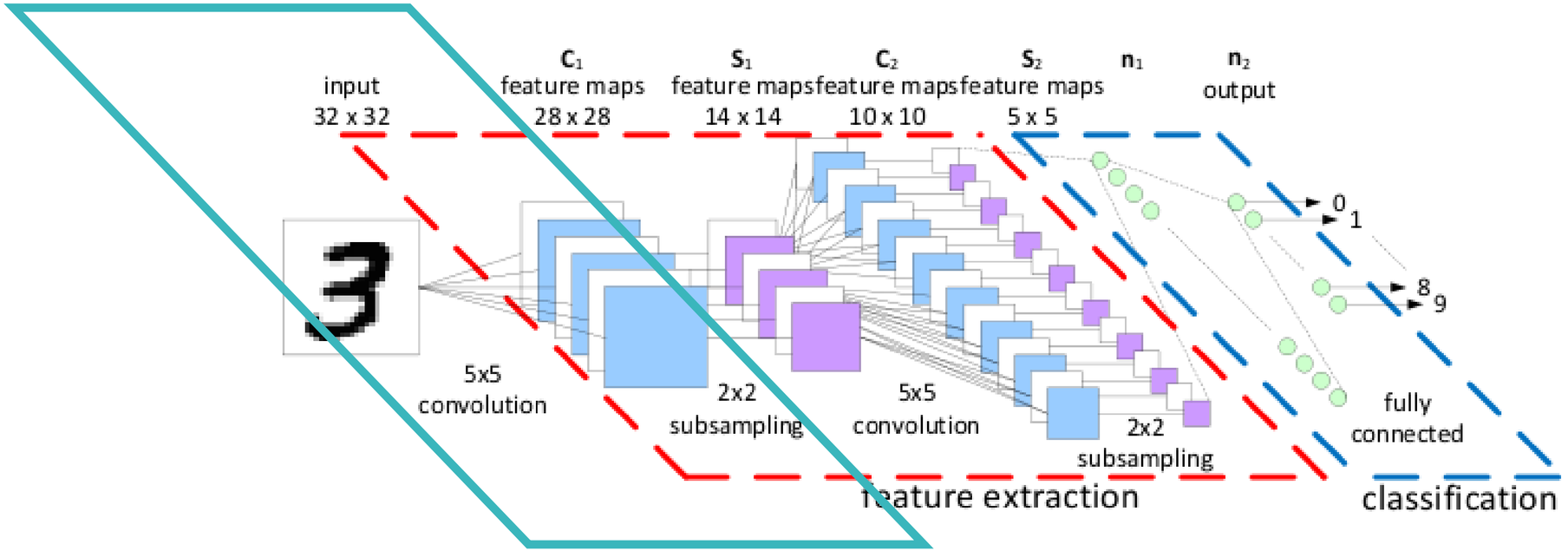


필기체 숫자 (0~9 사이의 값)

이미지 크기 (28*28)

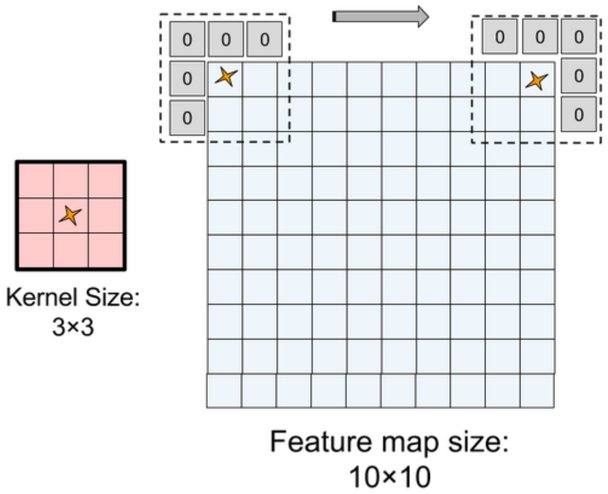
60,000장의 학습 셋, 10,000 장의 테스트 셋

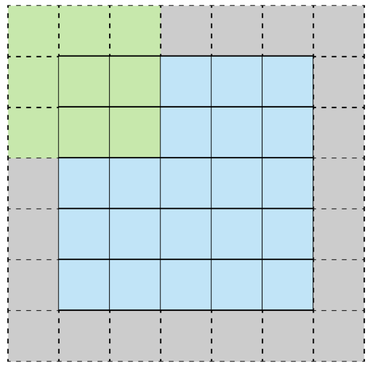
LeNet-5: 제로패딩

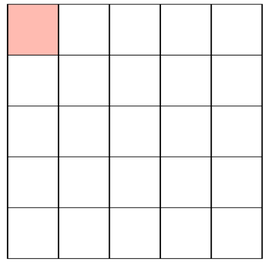


MNIST의 크기는 28*28, LeNet-5의 이미지는 32*32 ?

LeNet-5: 제로패딩



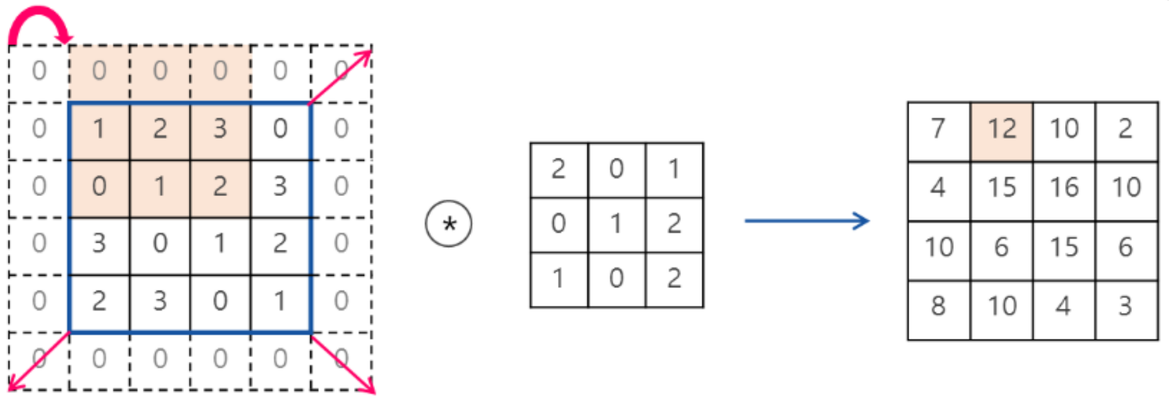

Stride 1 with Padding


Feature Map

입력 데이터 주변 값을 특정 값으로 채워 늘리는 것
(작아진 피쳐맵 크기, 가장 자리 정보 손실)
스트라이드: 커널이 입력 이미지에서 이동하는 간격

LeNet-5: 계산문제

패딩(P) : 1,
스트라이드(S) : 1



$$(OH, OW) = \left(\frac{H + 2P - FH}{S} + 1\ , \frac{W + 2P - FW}{S} + 1\right)$$

$$(OH, OW) = \left(\frac{4 + 2 * 1 - 3}{1} + 1\ , \frac{4 + 2 * 1 - 3}{1} + 1\right) = (\mathbf{4}, \mathbf{4})$$

- $(H,\ W)$: 입력크기
- $(FH,\ FW)$: 필터크기
- $(OH,\ OW)$: 출력크기
- P : 패딩
- S : 스트라이드

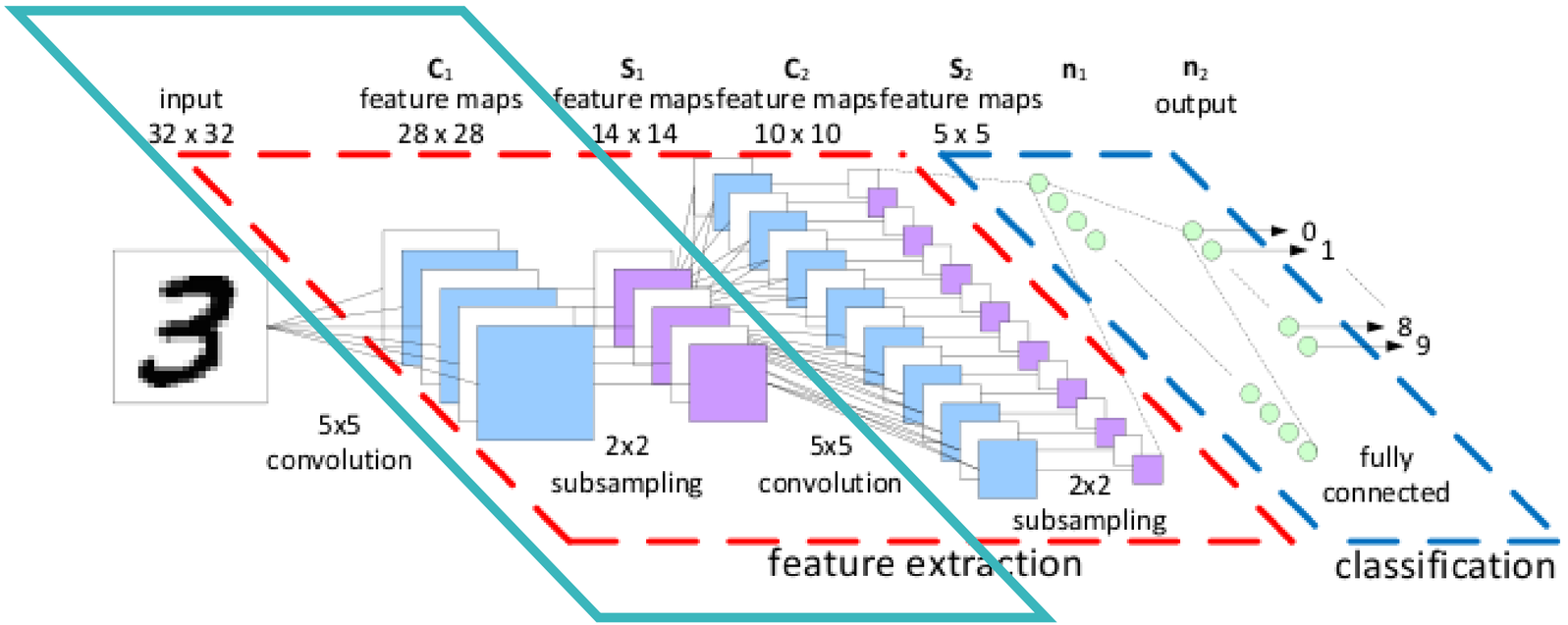
입력 (4*4), 패딩 (1), 스트라이드 (1)
컨볼루션 커널 (3*3), 출력 피쳐맵 (4*4)

LeNet-5: 계산문제

```
>>> model=tf.keras.Sequential()
// input (10*10) 10종류의 kernel (3*3), stride (1) => output image (10*10)
>>> model.add(Conv2D(input_shale=(10,10,1), filters=10, kernel_size=(3,3), stride=(1,1))
// input (10*10) 10종류의 kernel (3*3) => output image (10*10)
>>> model.add(Conv2D(input_shale=(10,10,1), filters=10, kernel_size=(3,3), padding='same'))
```

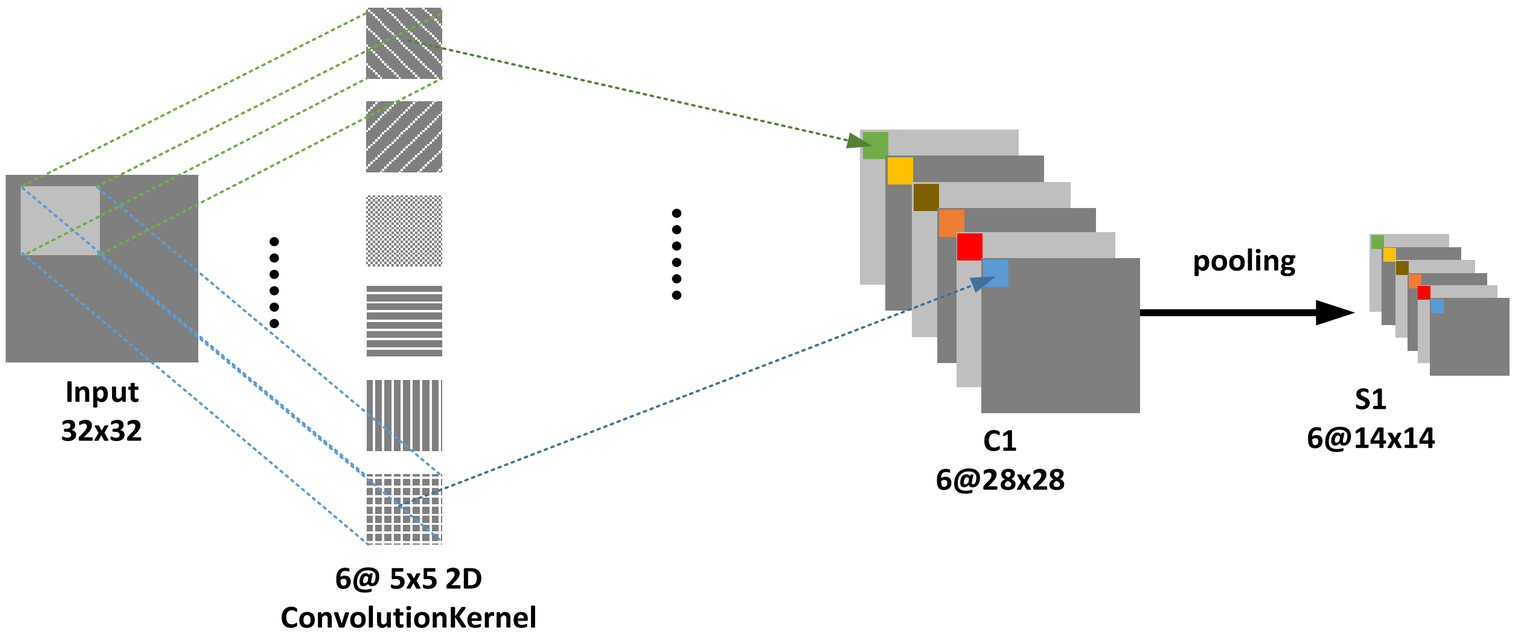
same (입력과 출력의 크기가 같도록), valid (패딩 0)

LeNet-5: 컨볼루션 & 풀링 #1



입력 이미지 (28*28), 6종류의 컨볼루션 커널 (5*5)
패딩 (4), 스트라이드 (1), 출력 이미지 (28*28)

LeNet-5: 컨볼루션 & 풀링 #1



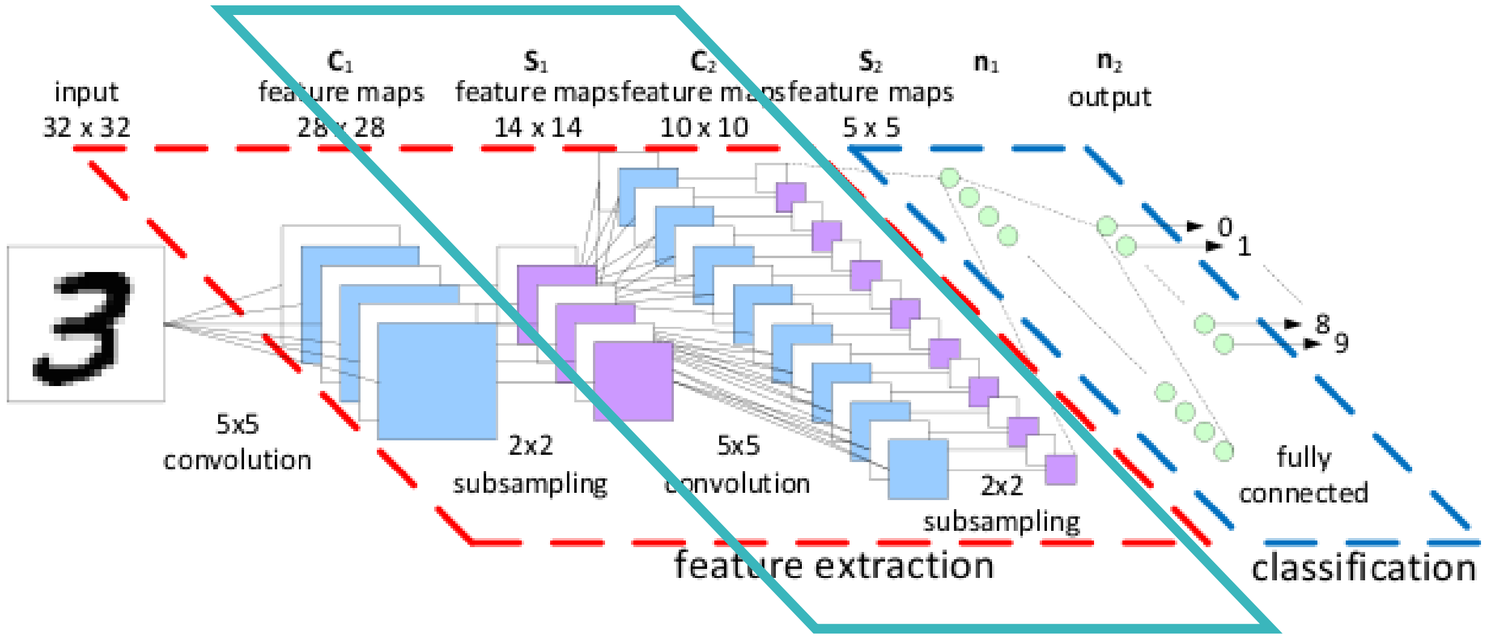
패딩된 입력 이미지 (32*32), 컨볼루션 (6@ 28*28)
풀링 (6@ 14*14)

LeNet-5: 풀링

```
>>> model=tf.keras.Sequential()
// 입력 이미지의 크기를 반으로 줄이는 방식 (stride 0)
>>> model.add(layers.MaxPooling2D((2, 2))), stride=None, data_format=None)
// 입력과 출력 이미지의 크기가 동일한 방식
>>> model.add(layers.MaxPooling2D((2, 2))), padding=Same, data_format=None)
```

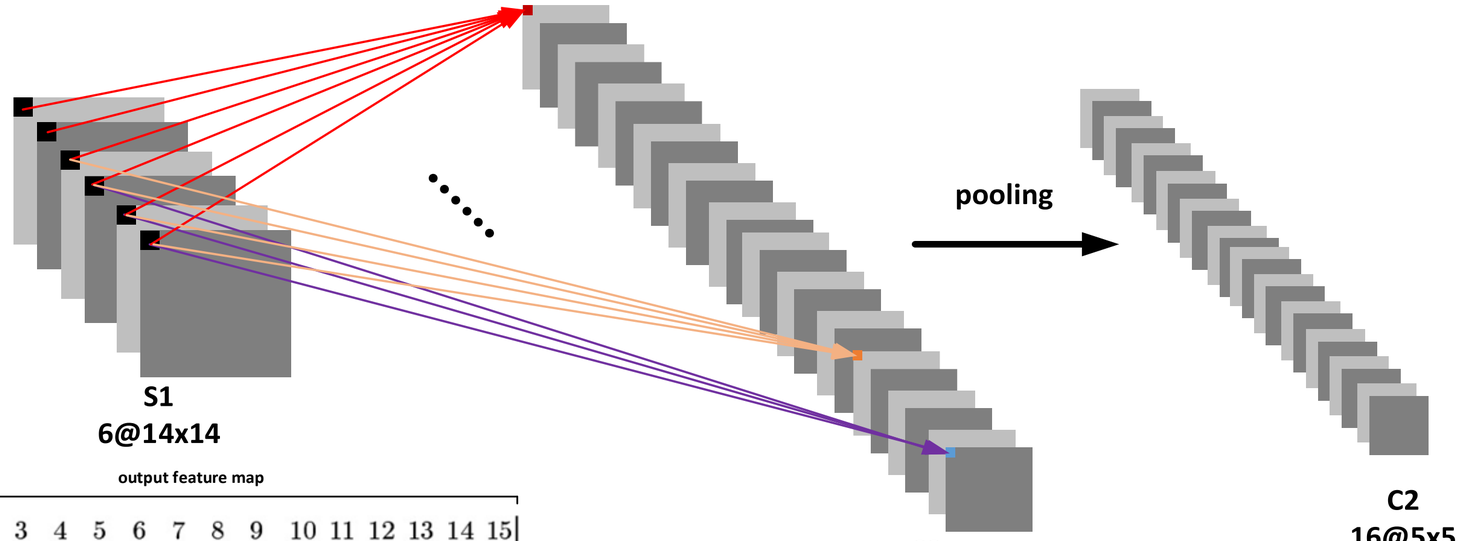
same (입력과 출력의 크기가 같도록), valid (패딩 0)
data_format (내부에서 데이터를 다루는 방식 선택)

LeNet-5: 컨볼루션 & 풀링 #2



6종류의 입력 이미지 (14*14), 16종류의 컨볼루션 커널 (5*5)
스트라이드 (1), 16종류의 출력 이미지 (10*10)

LeNet-5: 컨볼루션 & 풀링 #2

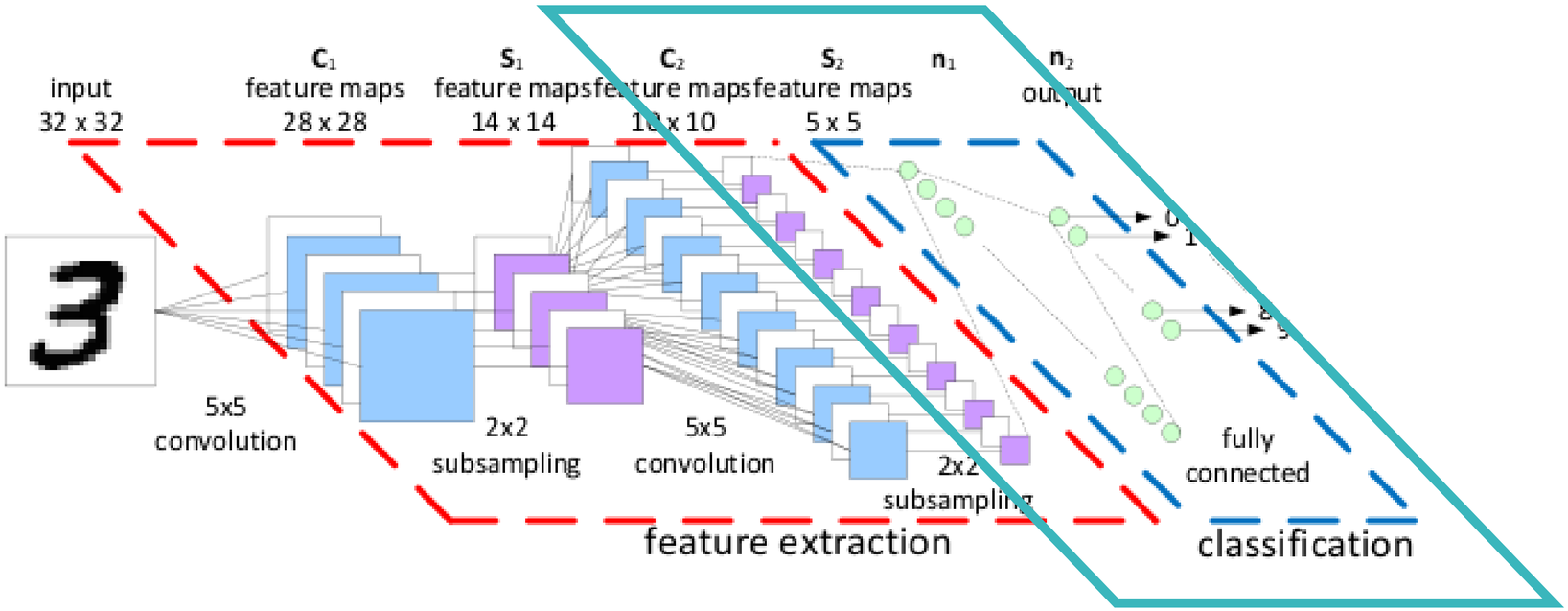


output feature map

Input feature map	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

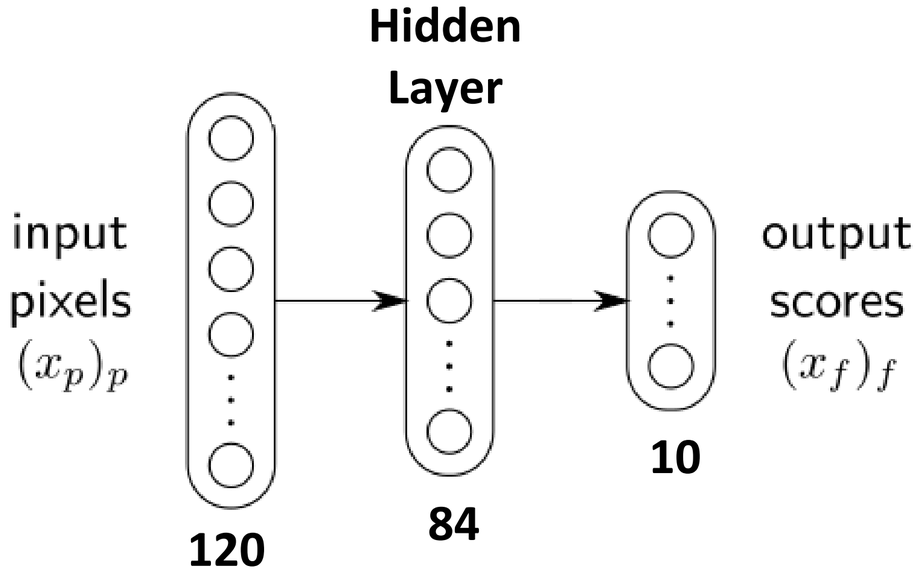
출력 이미지를 생성하는데 상이한 입력 이미지 개수, 종류
입력 이미지 (6@ 14*14), 컨볼루션 (16@ 5*5), 풀링 (16@ 5*5)

LeNet-5: 분류기



특징 추출기의 결과를 사용하여 이미지를 분류

LeNet-5: 분류기



특징 추출기의 결과를 사용하여 이미지를 분류
입력 (120*1), 히든 레이어 (84*120), 출력 (10*84)

LeNet-5: 분류기

```
>>> model=tf.keras.Sequential()
>>> model.add(layers.Flatten())
>>> model.add(layers.Dense(84, activation='relu'))
>>> model.add(layers.Dense(10, activation='softmax'))
>>> model.summary()
```

Flatten: 3차원 텐서를 1차원으로 변형
Dense 레이어: 1차원 형태의 벡터를 사용하는 레이어
최종 출력 결과는 ???

/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

medium.com/elice