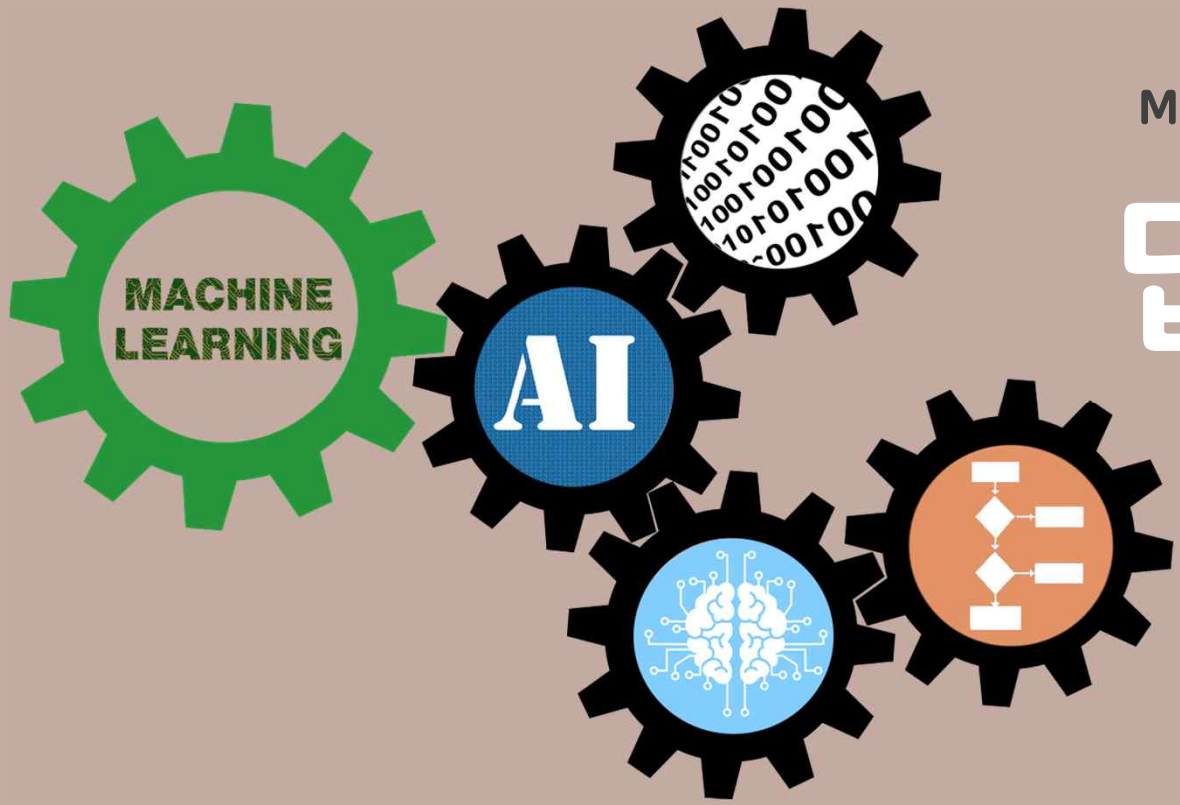


Machine Learning 11강

딥러닝 (1)

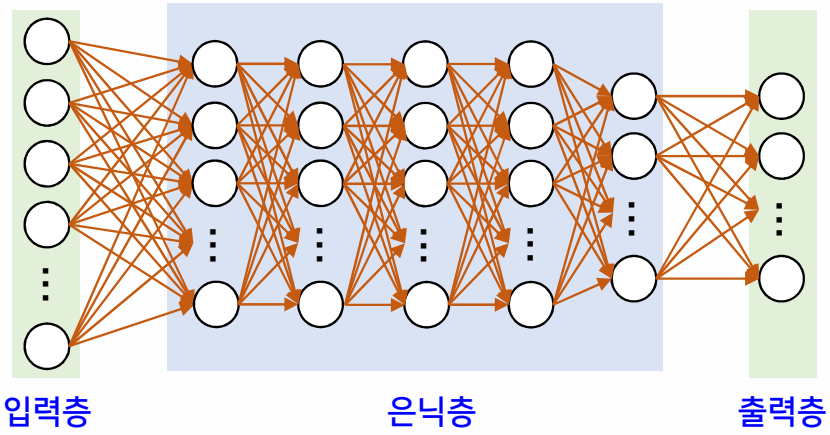
컴퓨터과학과 이관용 교수

학습목차

1 딥러닝의 등장

From MLP to 심층 신경망

- 딥러닝
  - 심층 신경망 기반의 머신러닝 분야
    - ✓ "심층 deep 신경망" ↔ "얕은 shallow 신경망"
- 기본적인 형태의 심층 신경망?
  - 많은 수의 은닉층을 가진 MLP
  - 장점
    - ✓ 더 효율적인 표현이 가능
  - 단점
    - ✓ 학습의 어려움 → 느린 수렴 속도, 낮은 일반화 성능



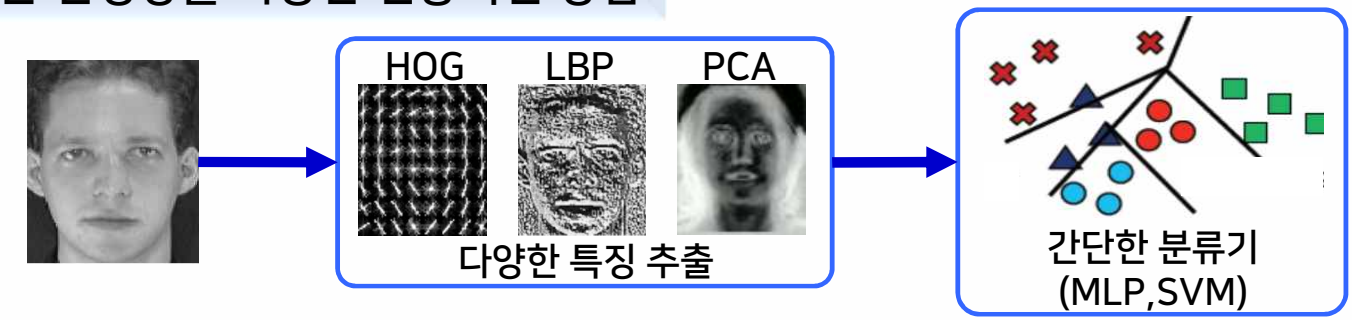
From MLP to 심층 신경망

- 학습의 어려움을 극복하게 만든 요인
  - 충분히 큰 데이터베이스
  - 높은 컴퓨팅 파워와 GPU를 활용하는 기술 등
  - 다양한 학습 기법의 개발
  - 더 정교한 모델의 등장

From MLP to 심층 신경망

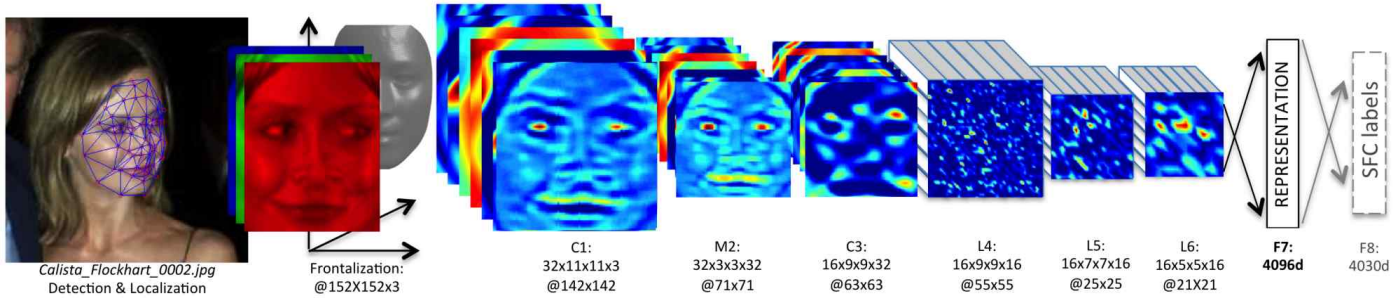
신경망을 통한 처리 과정에 대한 패러다임의 변화가 발생

얕은 신경망을 사용한 전통적인 방법



심층 신경망을 사용한 종단간 end-to-end 학습

"특징추출 과정과 특징에 의한 분류 과정을 한꺼번에 학습하는 방식"



출처: Y. Taigman 등, "DeepFace: Closing the gap to Human-Level Performance in Face Verification", CVPR2014

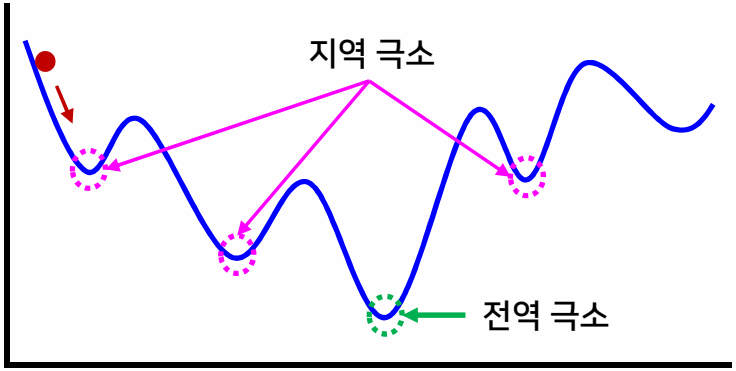
2

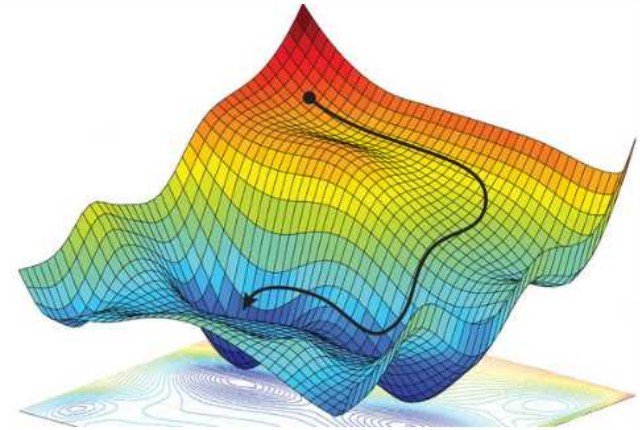
학습 성능 향상을 위한 기법

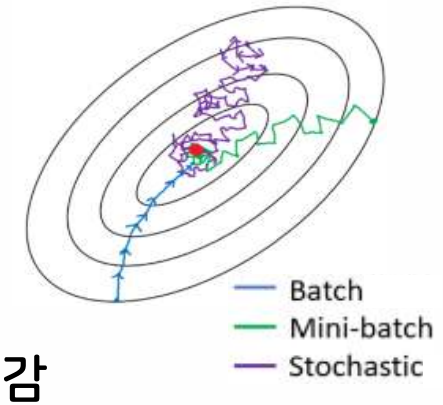
지역 극소 local minima

기울기 강하 학습법의 근본적인 문제

- 학습이 실패하는 경우 → 오차가 충분히 작지 않은 지역 극소에서 학습이 멈춤





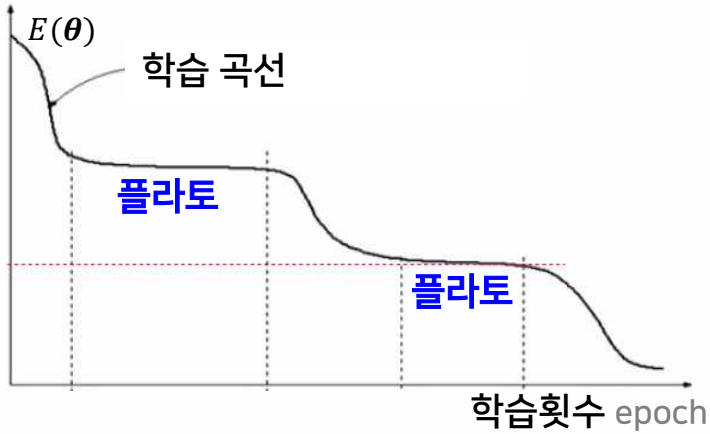


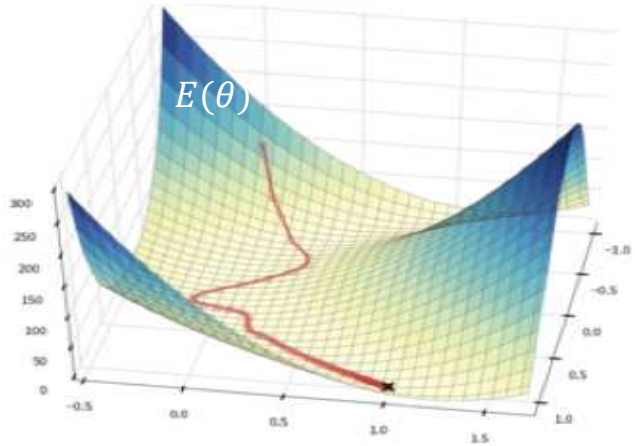
- 대안
  - ✓ simulated annealing → 학습률(η)을 처음에는 크게, 차차 줄여감
  - ✓ 확률적 기울기 강하 stochastic gradient descent → 한 번에 하나의 샘플만 사용

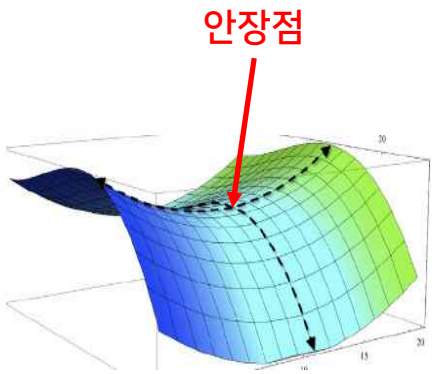
느린 학습 slow learning

플라토 plateau 문제

- 기울기 강하 학습의 오차함수의 학습곡선에서 평평한 구간plateau





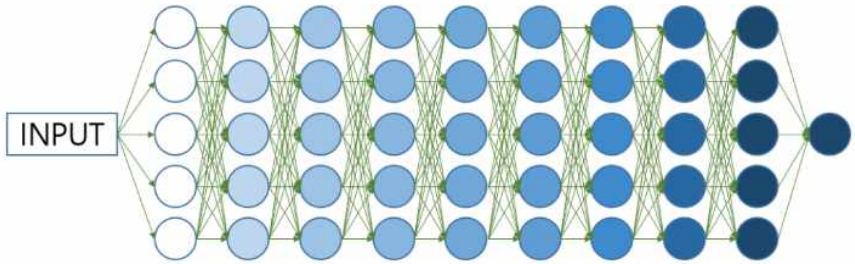


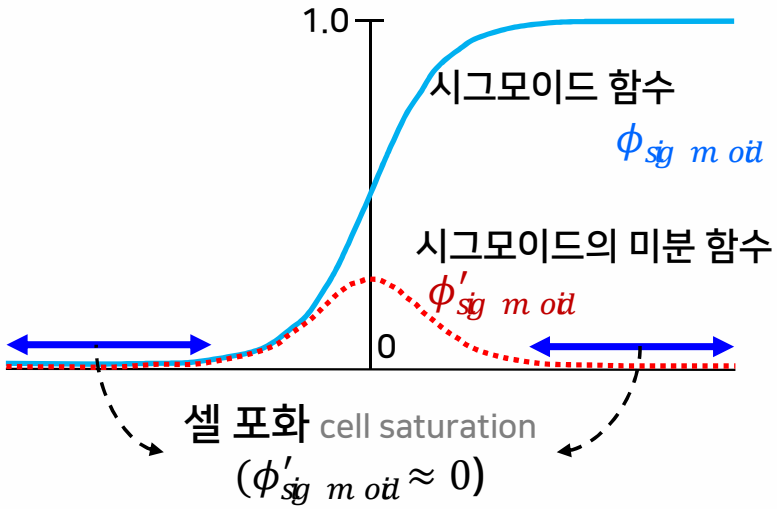
- 오차함수에는 플라토를 만드는 안장점이 무수히 많이 존재
  - ✓ 안장점saddle point → 극대/극소가 아닌 극점(미분값 0)

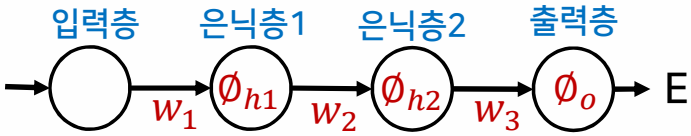
느린 학습 slow learning

기울기 소멸 문제 gradient vanishing problem

- 가중치 수정폭은 기울기의 크기에 의존 $\Delta\boldsymbol{\theta} \propto \frac{\partial E}{\partial \boldsymbol{\theta}}$
- 출력층으로부터의 오차 신호가 입력층으로 내려오면서 점점 약해져서 학습이 느려짐







$$\frac{\partial E}{\partial w1} = \frac{\partial E}{\partial Output} * \frac{\partial Output}{\partial Hidden2} * \frac{\partial Hidden2}{\partial Hidden1} * \frac{\partial Hidden1}{\partial w1}$$

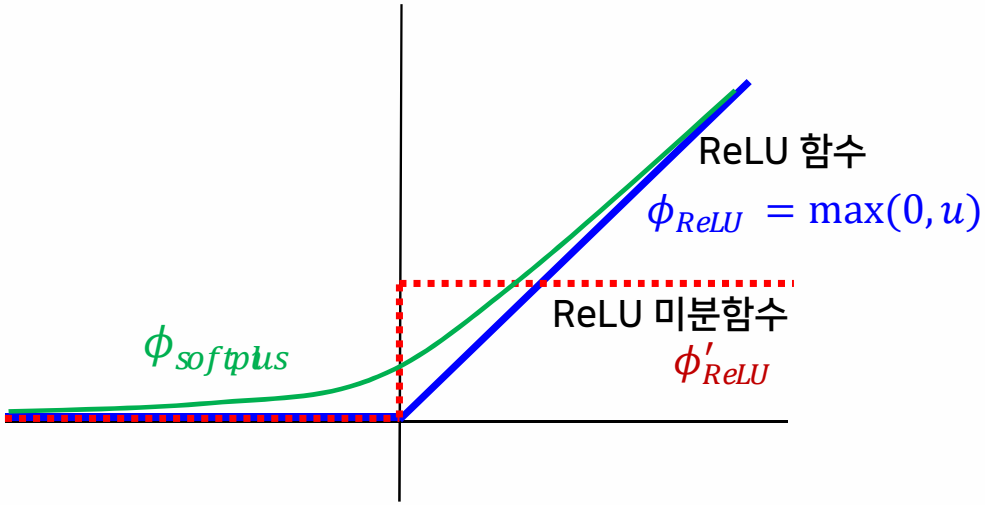
$\emptyset'_o$ $\emptyset'_{h2}$ $\emptyset'_{h1}$

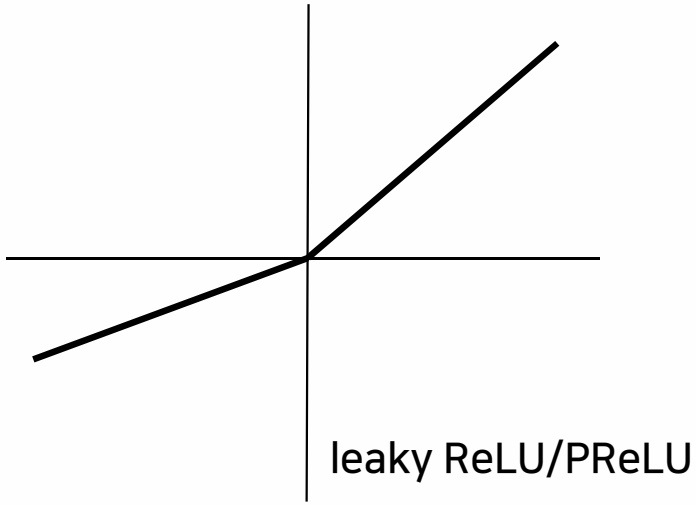
느린 학습의 해결책

활성화 함수의 변화

- 기울기 소멸 문제 → 활성화 함수의 기울기가 작아져서 발생
- sigmoid, tanh 함수 대신 기울기가 줄어들지 않는 함수 사용

  → ReLU, softplus, leaky ReLU, PReLU 등





느린 학습의 해결책

가중치 초기화

- 셀 포화가 일어나지 않도록 작은 값으로 설정
- 각 뉴런의 가중치가 서로 달라지도록 랜덤하게 설정

모멘텀 momentum

- 기울기 강하 학습법의 기울기 수정식

$$\boldsymbol{\theta}^{(\tau+1)} = \boldsymbol{\theta}^{(\tau)} + \Delta\boldsymbol{\theta}^{(\tau)} \longrightarrow \Delta\boldsymbol{\theta}^{(\tau)} = -\eta\nabla_{\theta}E\left(\boldsymbol{\theta}^{(\tau)}\right) + \gamma\Delta\boldsymbol{\theta}^{(\tau-1)}$$

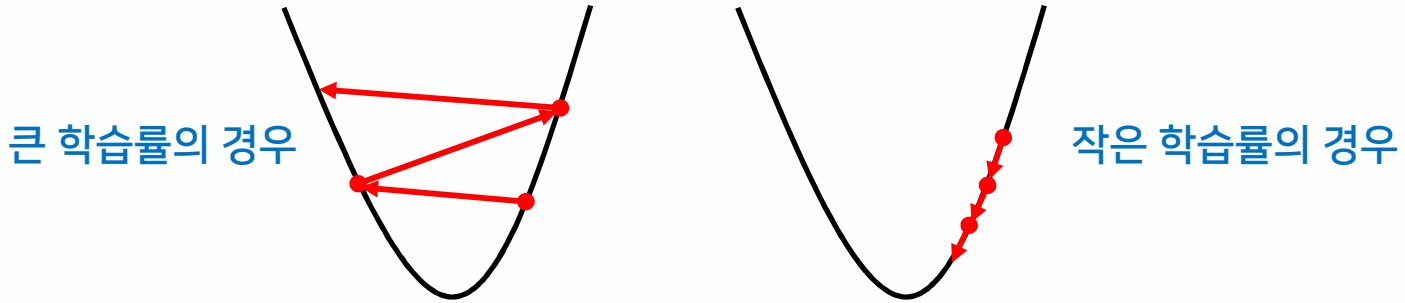
모멘텀 항

관성률

- 이전의 움직임(관성)을 반영
  - ✓ 학습 속도의 저하를 방지하거나 학습의 불안정성을 감소
- NAG Nesterov Accelerated Gradient $\Delta\boldsymbol{\theta}^{(\tau)} = -\eta\nabla_{\theta}E\left(\boldsymbol{\theta}^{(\tau)} + \gamma\Delta\boldsymbol{\theta}^{(\tau-1)}\right) + \gamma\Delta\boldsymbol{\theta}^{(\tau-1)}$

느린 학습의 해결책

적응적 학습률



- 가중치마다 서로 다른 학습률을 가짐
  - ✓ 가중치가 변화된 크기의 누적합을 활용하여 변화폭에 따라 학습률을 적응적으로 조정
- RMSProp, AdaDelta, Adam
  - ✓ Adam Adaptive Momentum → RMSProp과 모멘텀 방법의 결합

느린 학습의 해결책

배치 정규화 batch normalization

- 학습하는 동안 각 노드의 활성화 함수로 들어가는 입력이 셀 포화되지 않도록 정규화함
- 활성화 함수에 대한 입력분포를 항상 일정하게 유지

2차 미분 방법

- 오차함수의 2차 미분인 곡률curvature 정보를 활용

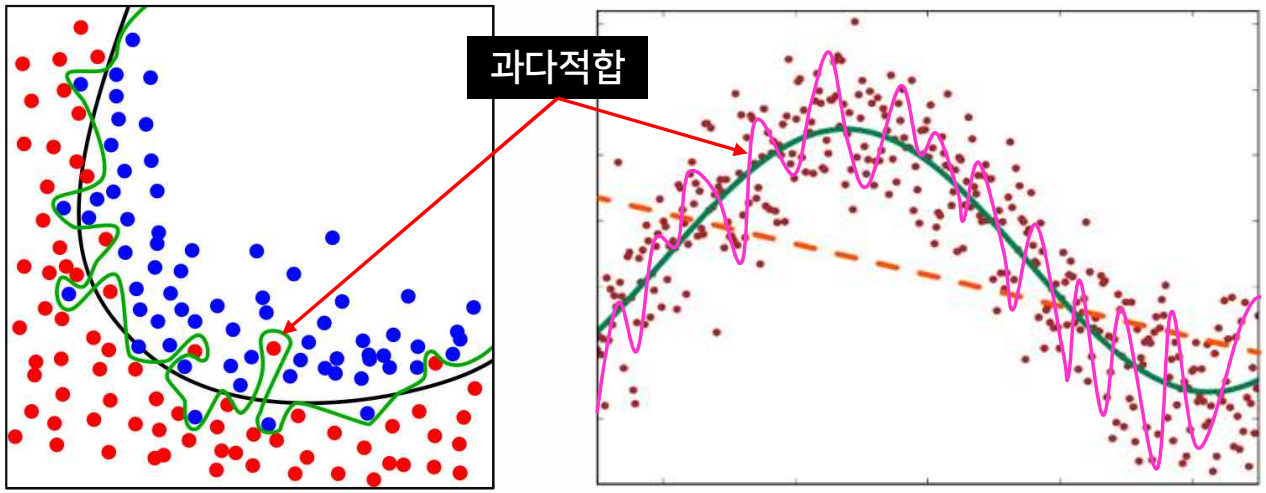
$$\Delta\boldsymbol{\theta}^{(\tau)} = -\eta\left(H\left(\boldsymbol{\theta}^{(\tau)}\right)\right)^{-1}\nabla_{\theta}E\left(\boldsymbol{\theta}^{(\tau)}\right)$$

- 이론적으로는 좋은 방법, 긴 계산 시간 → 실질적 사용에 한계

과다적합

과다적합 문제

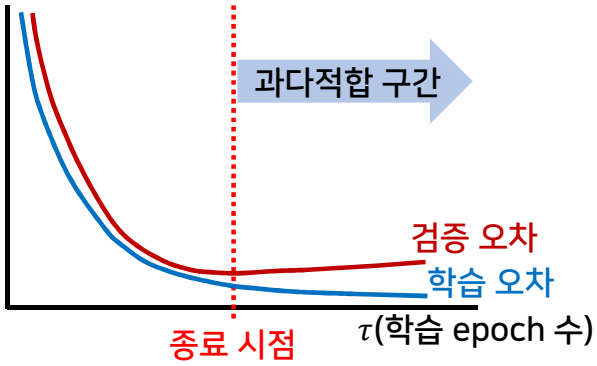
- 학습 데이터에 포함된 노이즈까지 학습하게 되어
  테스트 데이터에 대하여 정확도("일반화 성능")가 떨어지는 현상
- 신경망의 복잡도가 높을수록 발생할 가능성이 높아짐



과다적합의 해결책

조기 종료 early stopping

- 검증용 데이터 집합을 활용하여 과다적합 발생 전에 학습 종료 시점을 결정



정규항 regularization term

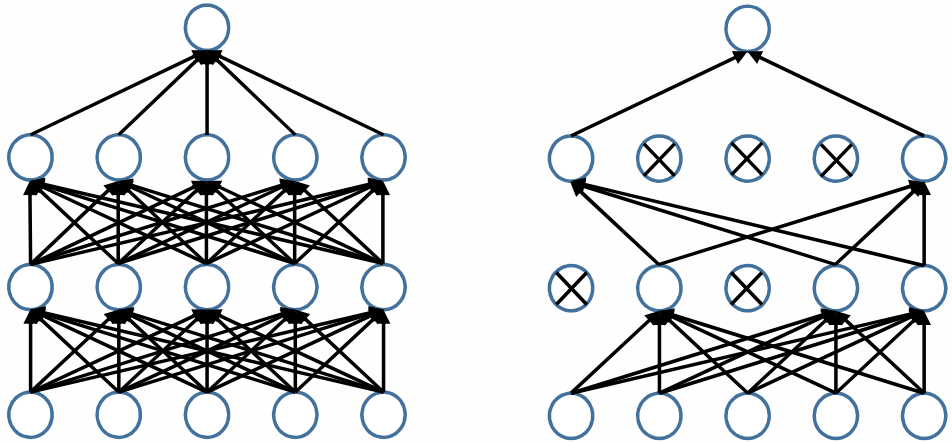
- 오차함수에 정규항 추가 → 가중치가 지나치게 커지는 것을 방지

$$E_{reg}(\boldsymbol{X};\boldsymbol{\theta}) = E_{sqr}(\boldsymbol{X};\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|^2$$ → 정규항

과다적합의 해결책

드롭아웃 dropout

- 학습에서 가중치 수정할 때 임의로 선택한 은닉 노드의 일부를 제외



기본 연결 구조 드롭아웃 적용

- ✓ 전체 모델이 가지는 복잡도보다는 낮은 모델로 학습하는 효과
- ✓ 작은 모델의 앙상블 평균과 유사한 효과 → 일반화 성능 향상

과다적합의 해결책

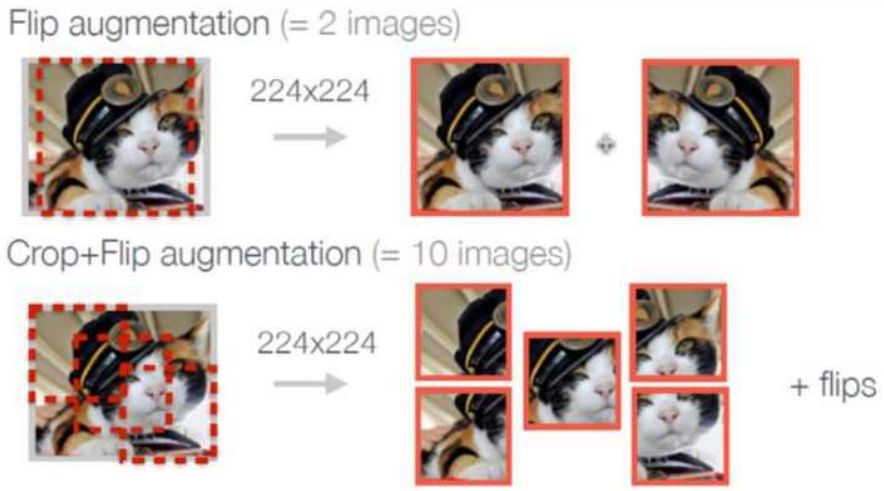
데이터 증대 data augmentation

- 원래 데이터에 대해 인위적인 변형을 가하여 추가적인 데이터 생성

  → 충분한 학습 데이터 확보

- 데이터 변형
  - ✓ 일반 데이터의 경우 → 노이즈 추가
  - ✓ 영상 데이터의 경우

    → 크기 조정, 회전, 위치 이동, 자르기 등



3 합성곱 신경망(CNN)

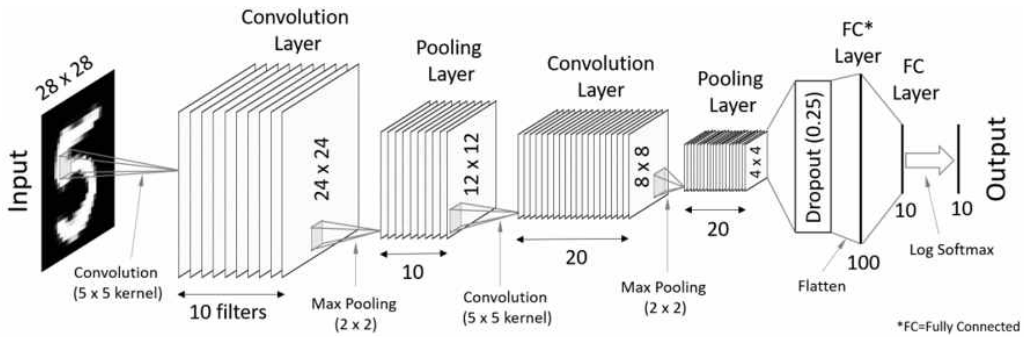
정교화된 심층 신경망 모델

- CNN, 합성곱 신경망
  - Convolutional Neural Networks
  - 인간의 시각 피질에서의 정보 처리 기제로부터 영감을 받은 모델
  - 영상 데이터 처리에 적합한 모델
- RNN, 순환 신경망
  - 기본 RNN → LSTM, GRU 등
  - 음성, 텍스트와 같은 시계열 데이터 처리에 적합한 모델

합성곱 신경망, CNN

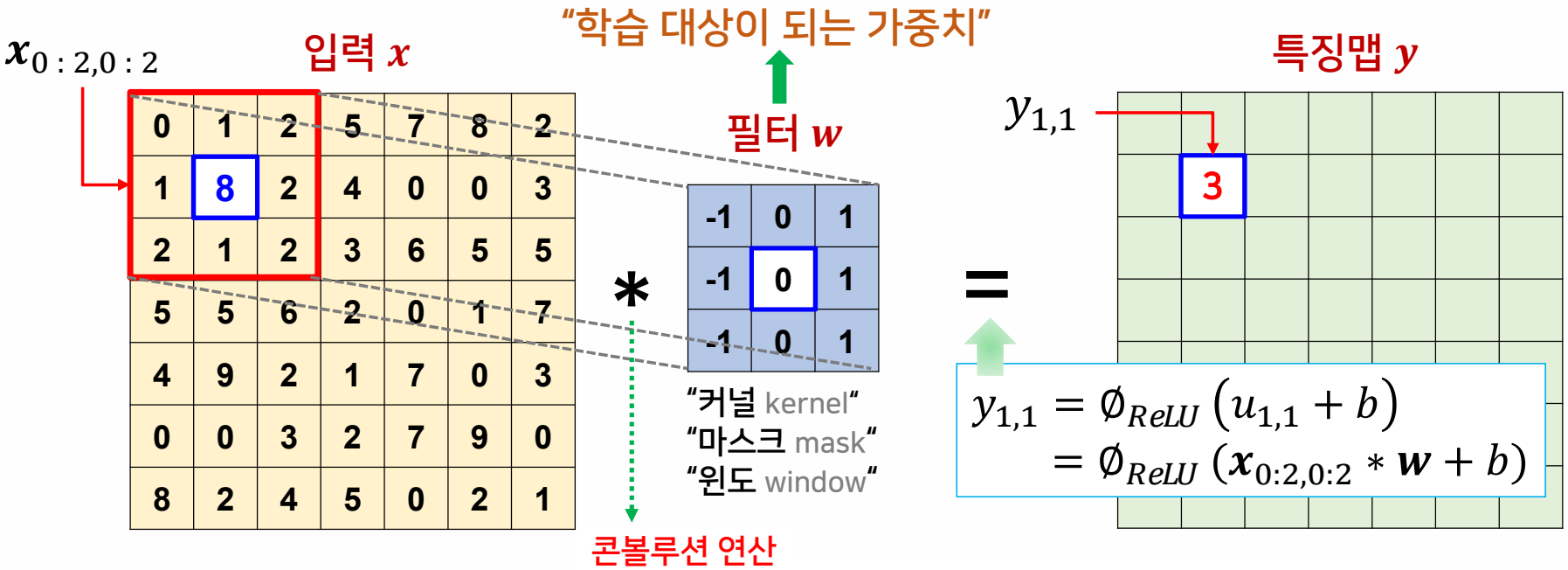
- 신경세포 → 3가지 유형(층)
  - 콘볼루션 convolution
  - 서브샘플링(풀링) subsampling(pooling)
  - 완전연결 fully connected
- 네트워크 구조 → 층상 구조
  - 입력층 → 2D 격자 구조 × 다중 채널
  - 콘볼루션층, 풀링층 → 2D 특징맵 × 다중 필터 filter, plane, kernel
    - ✓ 부분적인 연결 local connection , 가중치 공유 shared weight
  - 완전연결층 → MLP 구조
- 학습 알고리즘 → 오류 역전파 알고리즘

https://codetolight.files.wordpress.com/2017/11/network.png?w=1108



콘볼루션층

- 주어진 2D 입력에 콘볼루션 연산을 적용하여 특징맵을 생성
  - 해당 위치의 요소에 가중치를 곱해서 모두 더하는 선형 연산

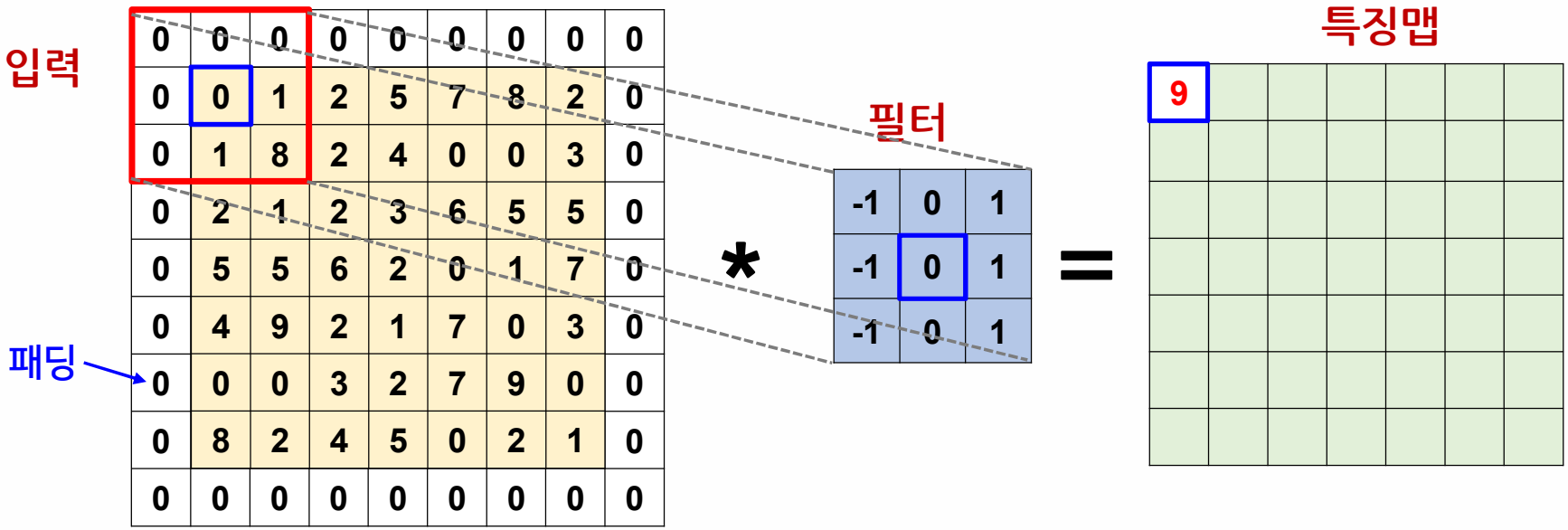


$$u_{1,1} = \boldsymbol{x}_{0:2,0:2} * \boldsymbol{w} = \sum_{i=0}^{2}\sum_{j=0}^{2} x_{i,j} w_{i,j}$$

콘볼루션층

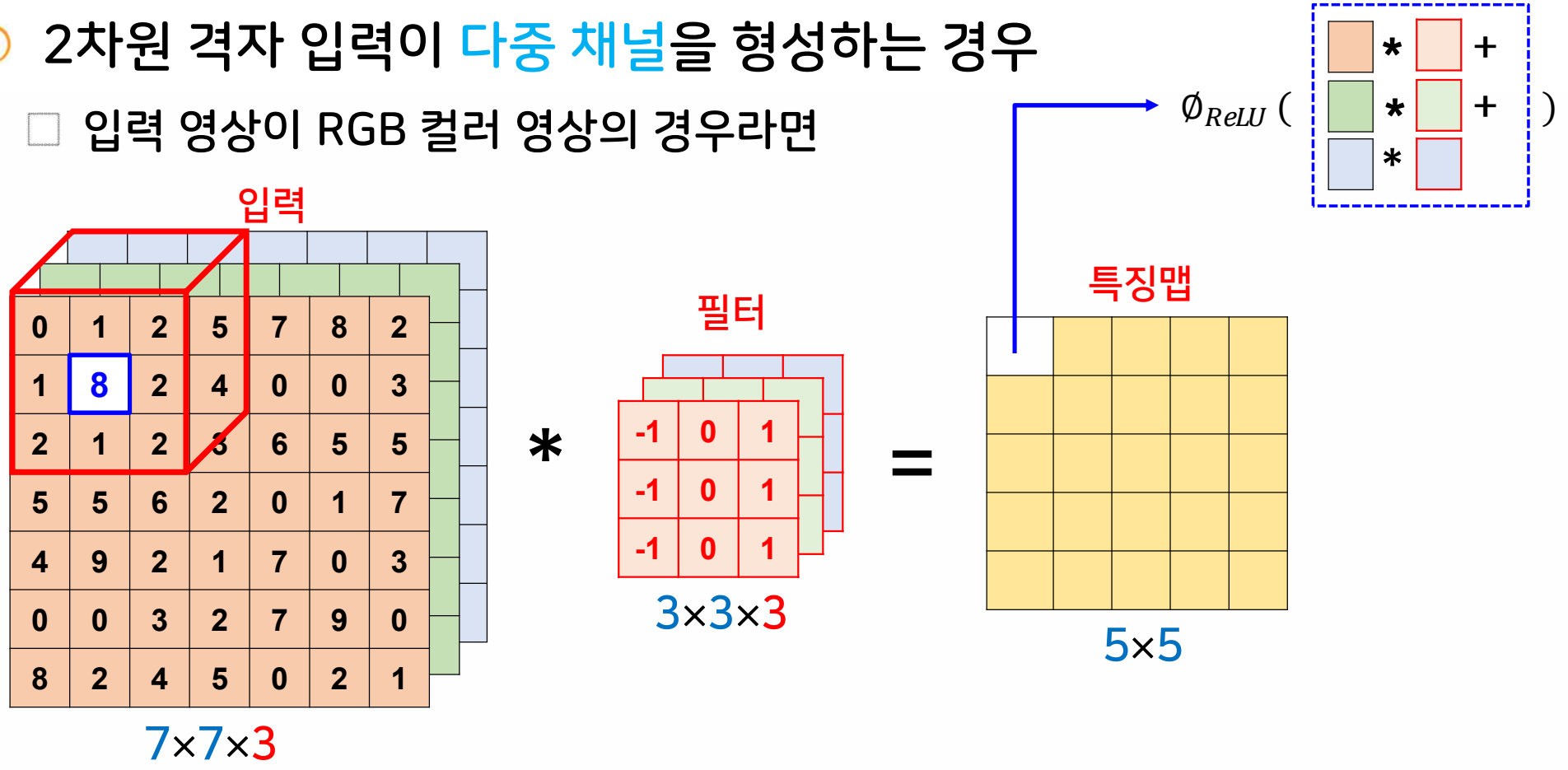
패딩 zero padding

- 입력 데이터의 가장자리를 0으로 채움
  - ✓ 필터 크기에 따라 패딩의 크기도 달라짐



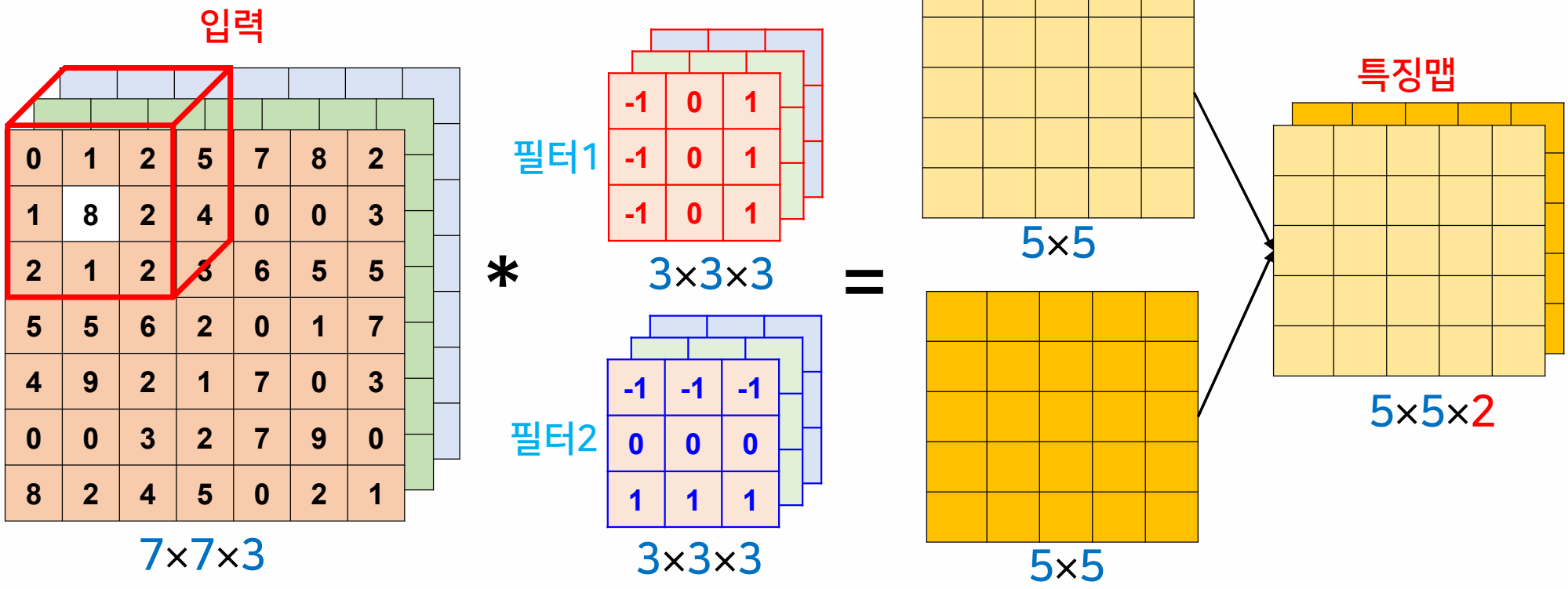
콘볼루션층

- 2차원 격자 입력이 다중 채널을 형성하는 경우
  - 입력 영상이 RGB 컬러 영상의 경우라면



콘볼루션층

- 다양한 형태의 특징을 추출하려면
  - 다수의 필터를 사용 → 다수의 특징맵 생성



콘볼루션층

보폭 stride

- 필터를 움직이는 간격을 조정하면 특징맵의 크기도 변함

1 입력

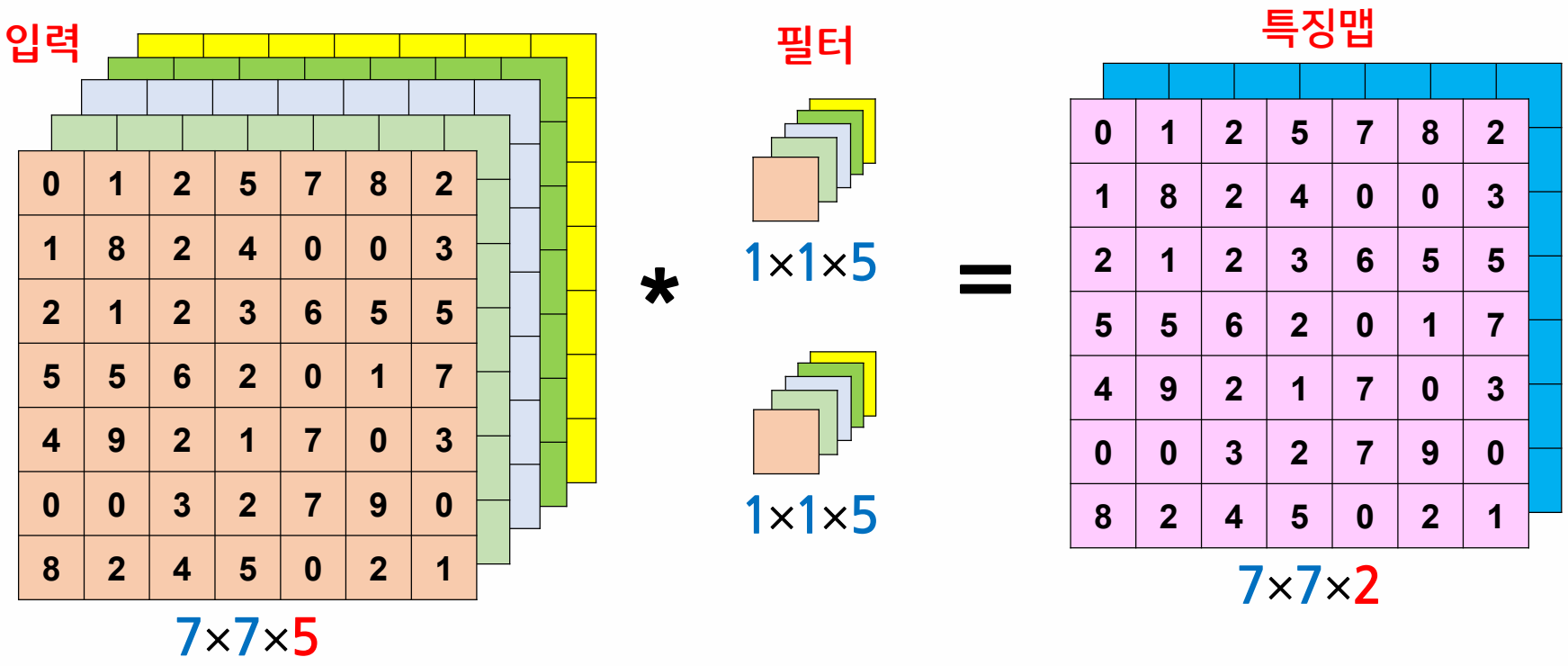
0	0	0	0	0	0	0	0	0
0	0	1	2	5	7	8	2	0
0	1	8	2	4	0	0	3	0
0	2	1	2	3	6	5	5	0
0	5	보폭 = 1인 경우					7	0
0	4	9	2	1	7	0	3	0
0	0	0	3	2	7	9	0	0
0	8	2	4	5	0	2	1	0
0	0	0	0	0	0	0	0	0

2 입력

0	0	0	0	0	0	0	0	0
0	0	1	2	5	7	8	2	0
0	1	8	2	4	0	0	3	0
0	2	1	2	3	6	5	5	0
0	5	보폭 = 2인 경우					7	0
0	4	9	2	1	7	0	3	0
0	0	0	3	2	7	9	0	0
0	8	2	4	5	0	2	1	0
0	0	0	0	0	0	0	0	0

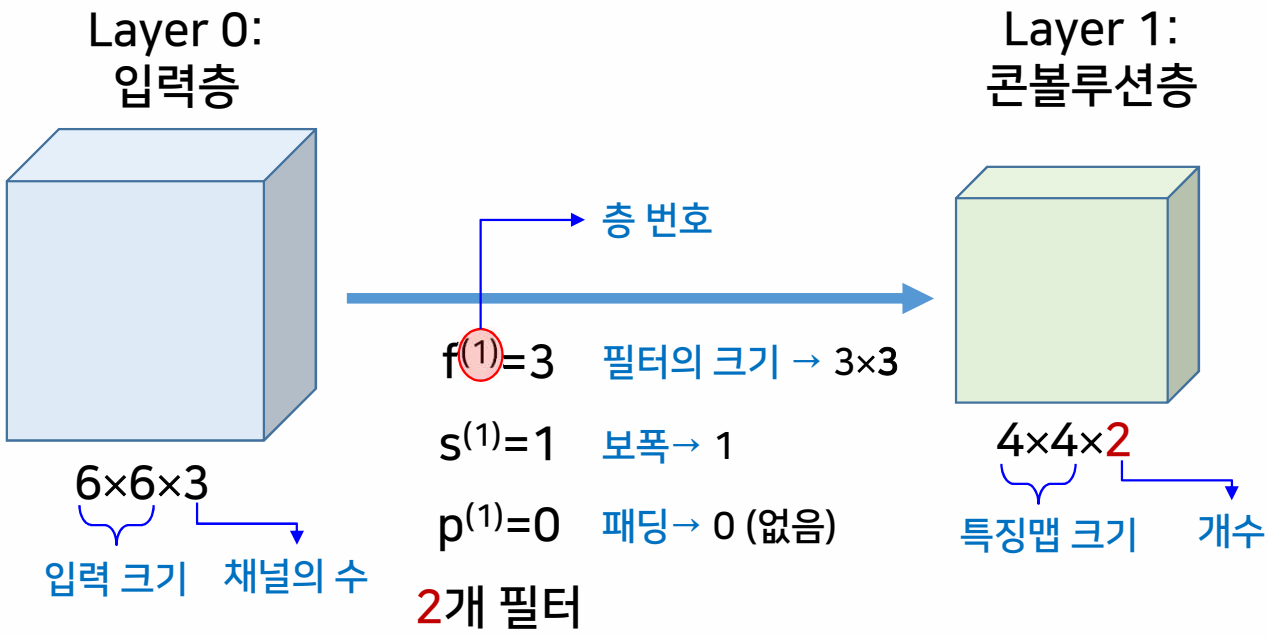
콘볼루션층

- 필터의 크기가 1×1인 경우 → 입력과 출력은 동일한 크기
  - 다중 채널의 입력에 대해 다중 필터를 적용하는 경우 → 데이터의 차원 축소 효과



콘볼루션층

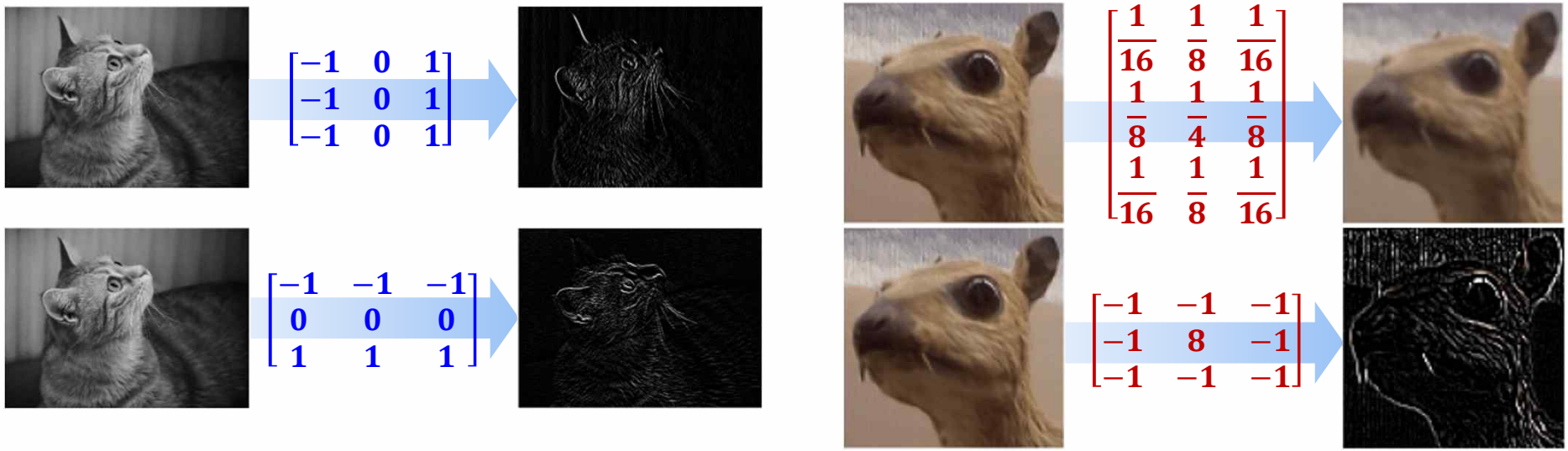
콘볼루션층의 간략한 표현 방법



콘볼루션층

왜 콘볼루션인가?

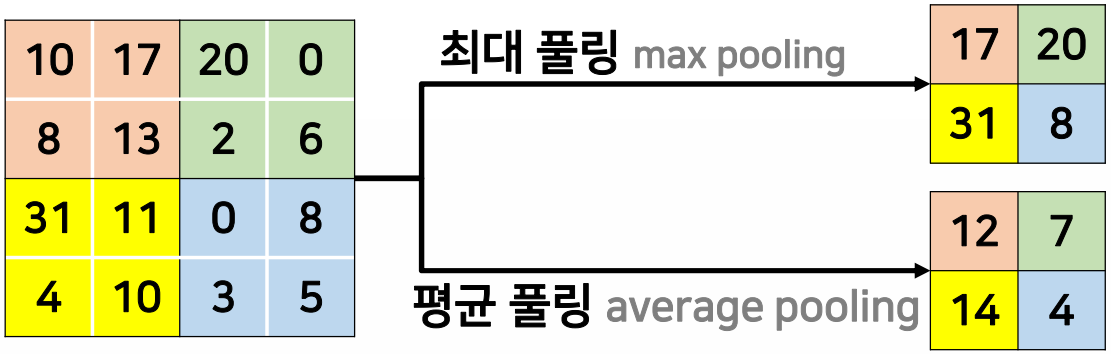
- 실제 두뇌의 시각 피질에서 영감을 받음



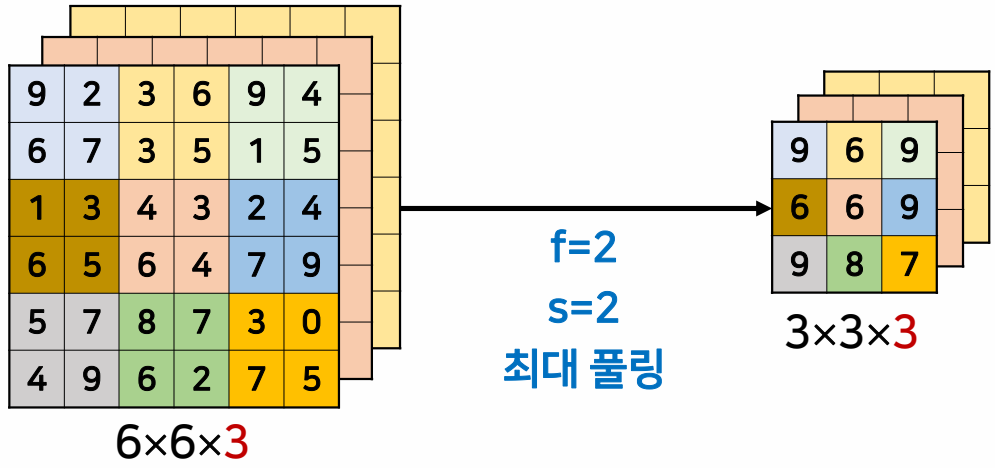
- 원 영상에서 의미 있는 특징 추출을 위해서
  - ✓ 기존의 수작업에 의한 설계가 아닌, CNN에서는 학습을 통해 추출

풀링층

- 풀링 연산



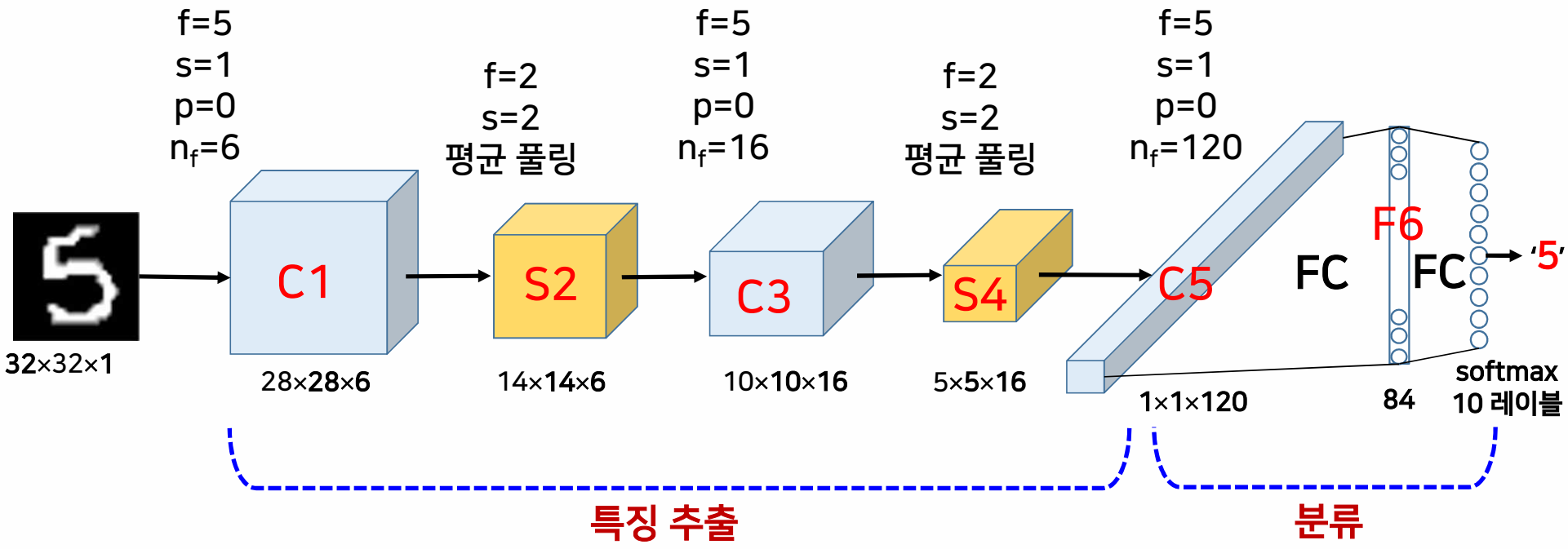
- 사용자 정의 파라미터



- 왜 풀링인가?
  - 특징맵의 크기를 작게 만듦
  - 데이터의 작은 위치 이동 변화를 수용할 수 있음

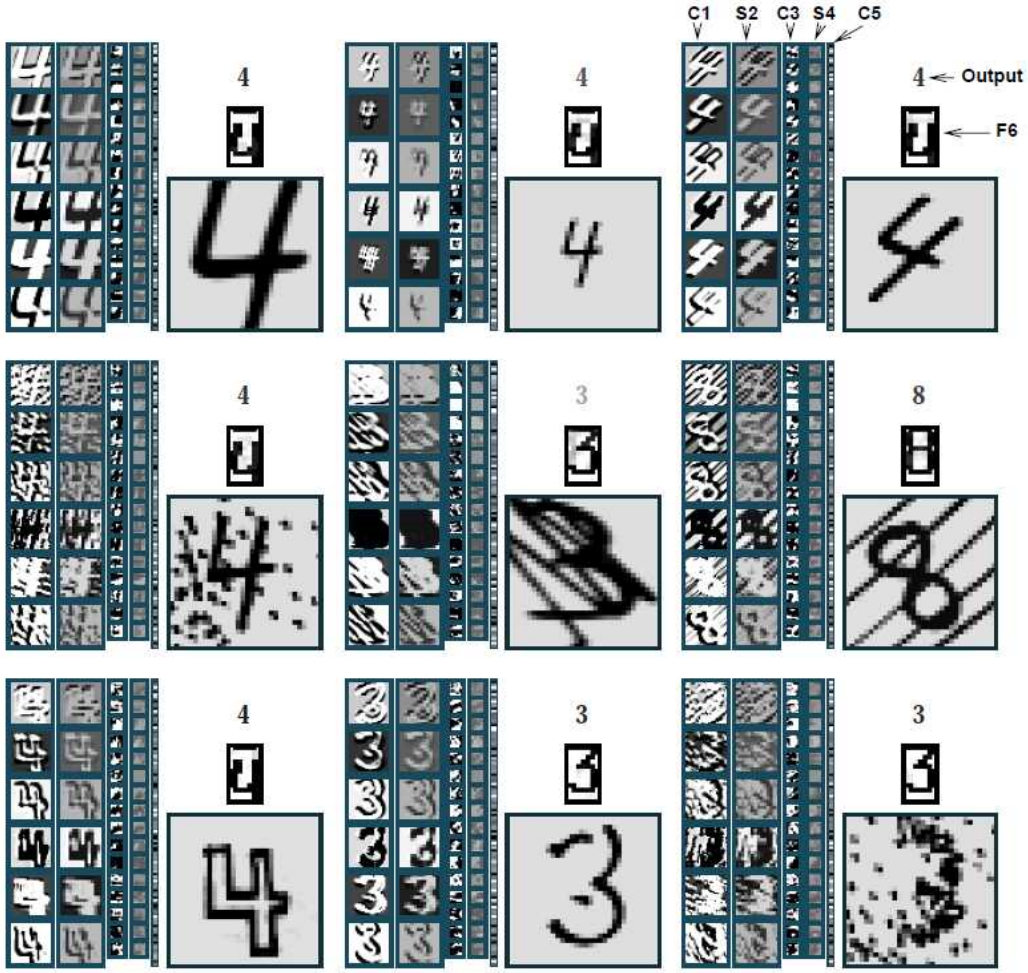
CNN의 예: LeNet-5

- 1998. Yann LeCun, 필기 숫자 인식 → "CNN의 첫 번째 성공 사례"



CNN의 예: LeNet-5

- LeNet-5가 추출한 특징

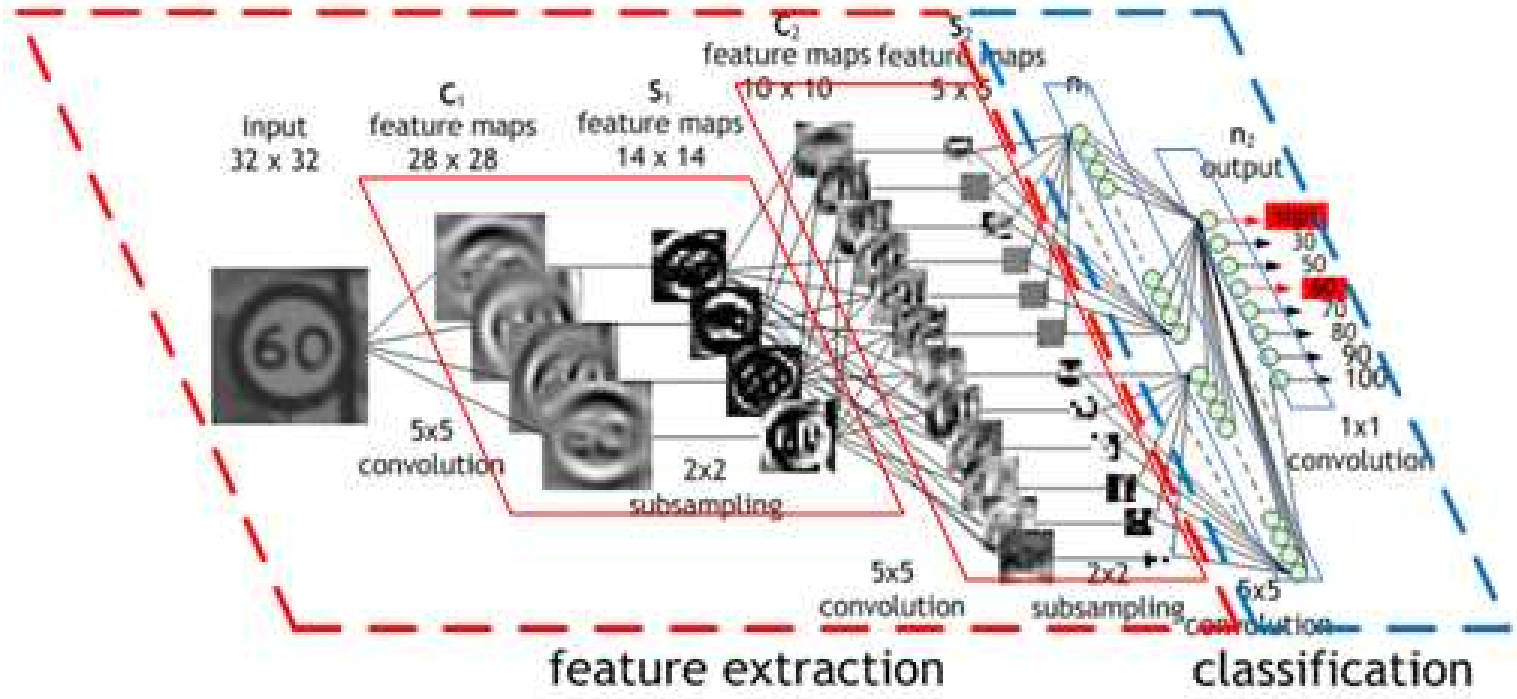


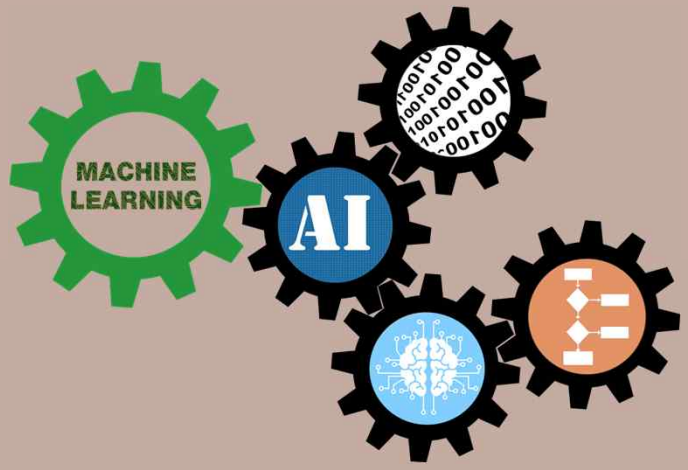
CNN의 예: LeNet-5

- 파라미터의 개수

	Para Num	Conn Num	Comments
C1	(5*5+1)*6=156	156*28*28=122304	Filter size(5,5)
S2	(1+1)*6=12	(2*2+1)*6*14*14=5880	Pool Size(2,2), 4 add, and a bias, total 5 conn
C3	(5*5+1)* (3*6+4*6+4*3+6*1)=1560 Or (5*5*3+1)*6+(5*5*4+1)*6+(5*5*4+1)*3+ (5*5*6+1)*1=1516	1560*10*10=156000 or 1516*10*10=151600	
S4	(1+1)*16=32	(2*2+1)*16*5*5=2000	Pool Size(2,2)
C5	5*5*16+1=401	401*120=48120	120@1*1,fc
F6	(120+1)*84=10164	10164	

교통 표지판 인식을 위한 LeNet





다음시간안내

제12강

딥러닝 (2)