딥러닝 몸풀기

누구나 이해할 수 있는 딥러닝

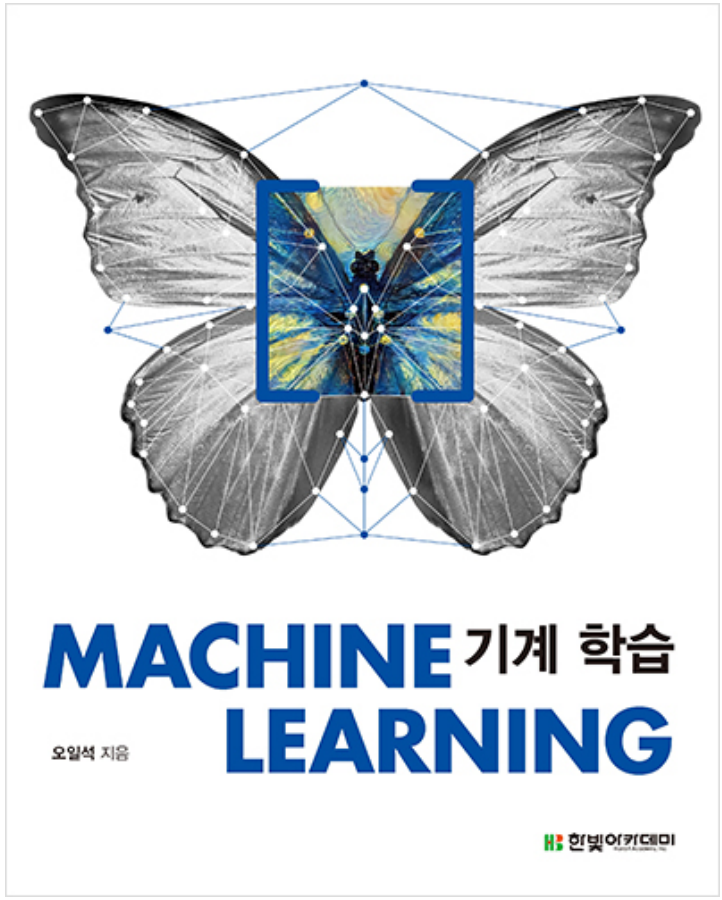
Local Laboratory

딥러닝 몸풀기

Local Laboratory

1. 인공지능과 기계학습 그리고 딥러닝
2. 다층 퍼셉트론
3. 딥러닝의 기초

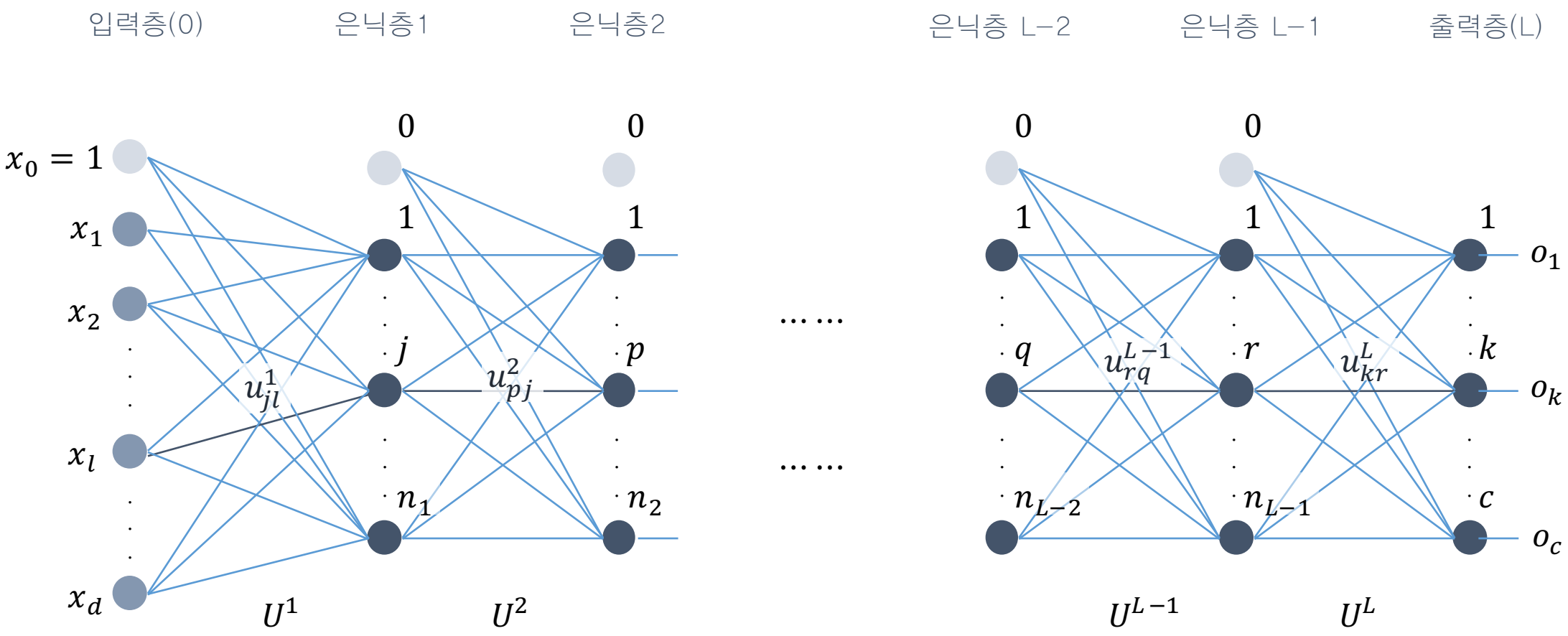
Reference



기계학습, 오일석

Self-Driving Car Nano degree, Udacity

딥러닝 Deep MLP / Deep Neural Network



입력층 d+1개, 출력층 c개, L−1개의 은닉층
l번째 은닉층의 노드 수를 n_l로 표기

$$U^l = \left(\begin{bmatrix} u^l_{10} & u^l_{11} & \cdots & u^l_{1n_{l-1}} \\ u^l_{20} & u^l_{21} & & u^l_{2n_{l-1}} \\ \vdots & & \ddots & \vdots \\ u^l_{n_l 0} & u^l_{n_l 1} & \cdots & u^1_{n_l n_{l-1}} \end{bmatrix} \right), \qquad l = 1,2,3,\cdots,L$$

DMLP의 가중치 행렬
l-1번째 층과 l번째 층을 연결하는 가중치는 총 $(n_{1-1}+1)n_l$

$$\mathbf{o} = \mathbf{f}(\mathbf{x}) = \mathbf{f}_L\left(\cdots \mathbf{f}_2\left(\mathbf{f}_1(\mathbf{x})\right)\right)$$

입력층 특징 $\mathbf{x}$를 내부 표현으로 바꿔쓰면,

$$z^0 = (z_0, z_1, z_2, \cdots, z_{n0})^{\mathrm{T}} = (1, x_1, x_2, \cdots, x_d)^{\mathrm{T}}$$

l번째 층의 j번째 노드가 수행하는 연산

$$z_j^l = \tau_l(s_j^l) \qquad s_j^l = \mathbf{u}_j^l \, \mathbf{z}^{l-1}$$

$$\mathbf{z}^{l-1} = \left(1, z_1^{l-1}, z_2^{l-1}, \cdots, z_{n_{l-1}}^{l-1}\right)^{\mathrm{T}}, \qquad \mathbf{u}_j^l = \left(u_{j0}^l, u_{j1}^l, \cdots, u_{j\,n_{l-1}}^l\right)$$

행렬표기를 이용하여 l번째 층의 연산 전체를 쓰면,

$$\mathbf{z}^l = \boldsymbol{\tau}_l\left(\mathbf{U}^l \mathbf{z}^{l-1}\right), \qquad 1 \le l \le L$$

L번째 층(출력층)의 그레이디언트 계산

$$\delta_k^L = \tau_L'\left(s_k^L\right)(y_k - o_k), \qquad 1 \le k \le c$$

$$\frac{\partial J}{\partial u_{kr}^L} = -\delta_k^L z_r^{L-1}, \qquad 0 \le r \le n_{L-1}, 1 \le k \le c$$

$l+1$번째 층의 정보를 이용하여 l 번째 층의 그레이디언트 계산($l = L-1, L-2, \dots, 1$)

$$\delta_j^l = \tau_L'\left(s_j^l\right) \sum_{p=1}^{n_{l+1}} \delta_p^{l+1} u_{pj}^{l+1}, \qquad 1 \le j \le n_l$$

$$\frac{\partial J}{\partial u_{ji}^l} = -\delta_j^l z_i^{l-1}, \qquad 0 \le i \le n_{l-1}, 1 \le j \le n_l$$

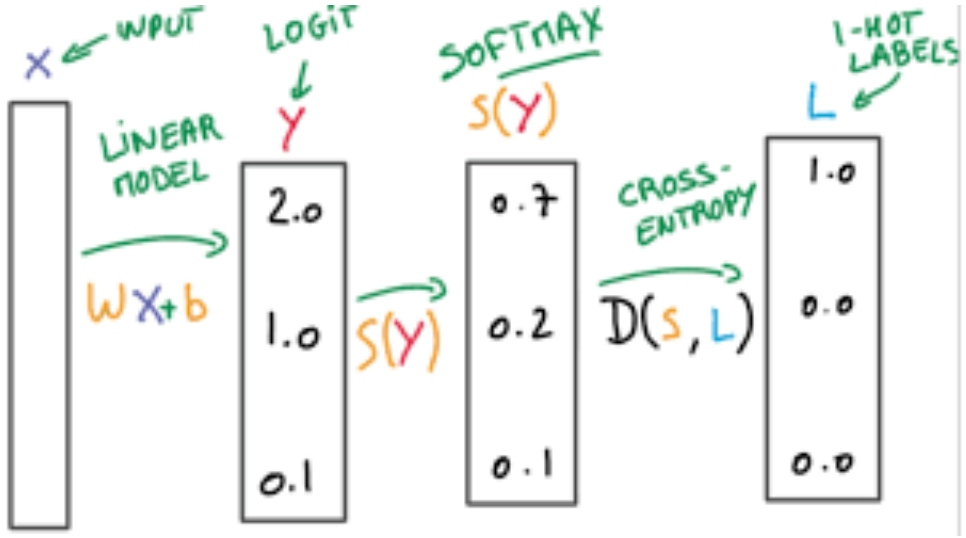
미니배치 스토캐스틱 경사 하강법

알고리즘 4-1 DMLP를 위한 미니배치 스토캐스틱 경사 하강법

입력: 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ, 미니배치 크기 t
출력: 가중치 행렬 $\mathbf{U}^l, l = 1,2,\cdots,L$

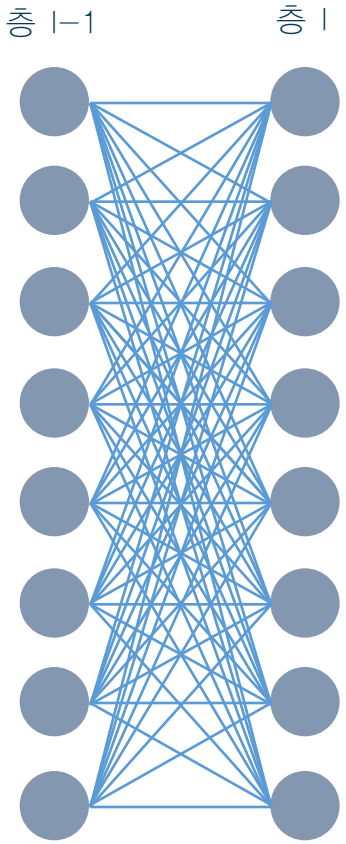
$\mathbf{U}^l, l = 1,2,\cdots,L$을 초기화한다.
repeat
  $\mathbb{X}$와 $\mathbb{Y}$에서 t개의 샘플을 무작위로 뽑아 미니배치 $\mathbb{X}'$와 $\mathbb{Y}'$를 만든다.
  for (l=1 to L) $\Delta\mathbf{U}^l = \mathbf{0}$
  for ($\mathbb{X}'$의 샘플 각각에 대해)
    현재 처리하는 샘플을 $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)^{\mathrm{T}}$, $\mathbf{y} = (y_1, y_2, \cdots, y_c)^{\mathrm{T}}$라 표기한다.
    $x_0, z_0^1, z_0^2, \cdots, z_0^L$을 1로 설정한다.
      // 전방 계산
    $\mathbf{x}$를 $\mathbf{z}^0$에 대입한다. // 식 (4.3)
    for (l=1 to L) // 왼쪽 층에서 오른쪽 층으로 진행하면서 전방 계산
      for (j=1 to n_l) // 각 노드에 대해
        $s_j^l = \mathbf{u}_j^l \mathbf{z}^{l-1}$ // 식 (4.4)
        $z_j^l = \tau_l(s_j^l)$ // 식 (4.4)
      // 오류 역전파의 단계 1: 그레이디언트 계산
    for (k=1 to c) $\delta_k^L = \tau_L'(s_k^L)(y_k - o_k)$ // 식 (4.6)
    for (k=1 to c) for (r=0 to n_{L-1}) $\Delta u_{kr}^L = \Delta u_{kr}^L + (-\delta_k^L z_r^{L-1})$
    for (l=L-1 to 1) // 오른쪽 층에서 왼쪽 층으로 진행하면서 오류 역전파
      for (j=1 to n_l) $\delta_j^l = \tau_l'(s_j^l) \sum_{p=1}^{n_{l+1}} \delta_p^{l+1} u_{pj}^{l+1}$ // 식 (4.8)
      for (j=1 to n_l) for (i=0 to n_{l-1}) $\Delta u_{ji}^l = \Delta u_{ji}^l + (-\delta_j^l z_i^{l-1})$
    // 오류 역전파의 단계 2: 가중치 갱신
  for (l=L to 1)
    for (j=1 to n_l) for (i=0 to n_{l-1}) $u_{ji}^l = u_{ji}^l - \rho\left(\frac{1}{t}\right)\Delta u_{ji}^l$
until (멈춤 조건)

	퍼셉트론 Perception	→	**다층 퍼셉트론** MLP	→	**깊은 다층 퍼셉트론** DMLP or DNN
활성함수:	계단함수		시그모이드함수		ReLU와 변형들
비용함수:	평균제곱 오차		평균제곱 오차		교차 엔트로피 또는 로그우도

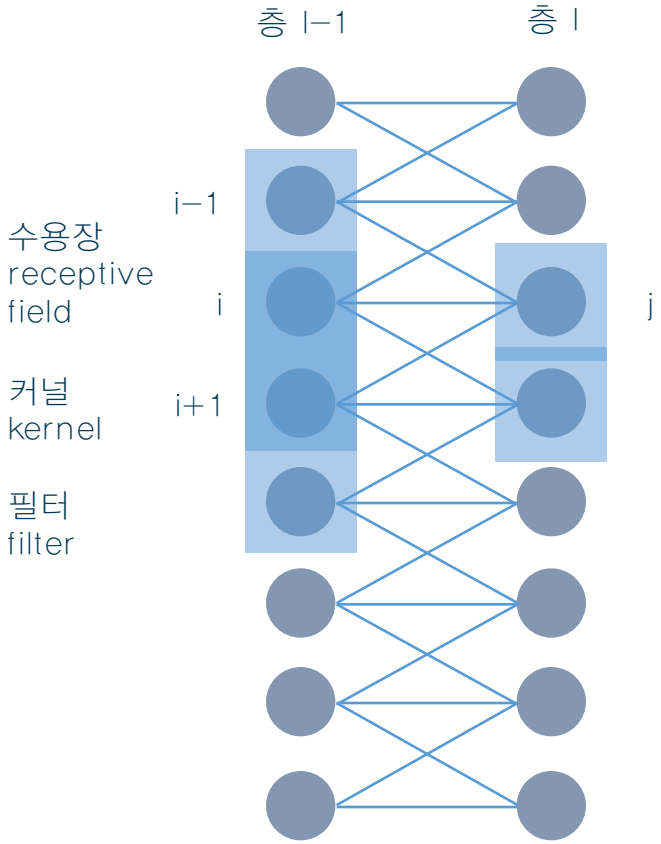


logit
softmax
one-hot encoding

DMLP(DNN)와 CNN 비교



DMLP



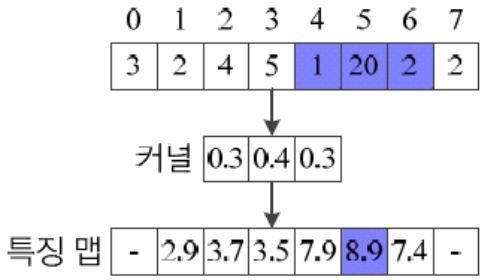
CNN

DMLP와 CNN의 비교

- DMLP
  - 완전연결 구조로 높은 복잡도
  - 학습이 매우 느리고 과잉적합 우려
- CNN
  - 컨볼루션 연산을 이용한 부분연결 (희소 연결) 구조로 복잡도 크게 낮춤
  - 컨볼루션 연산은 좋은 특징 추출

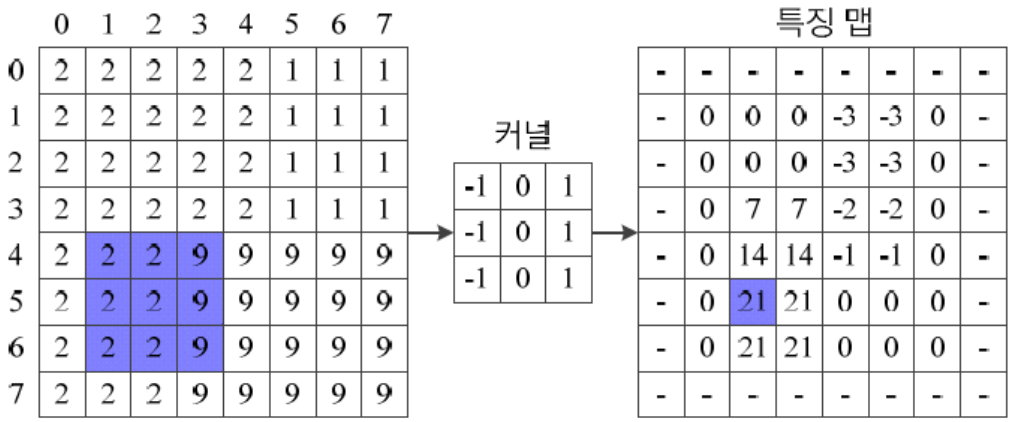
CNN

- 격자 구조(영상, 음성 등)를 갖는 데이터에 적합
- 수용장은receptive field 인간시각과 유사
- 가변 크기의 입력 처리 가능

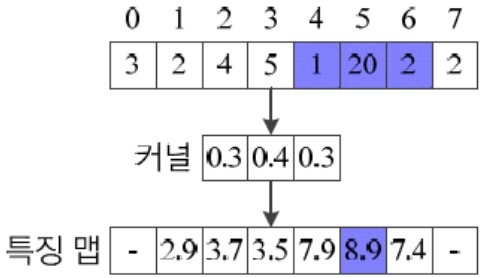


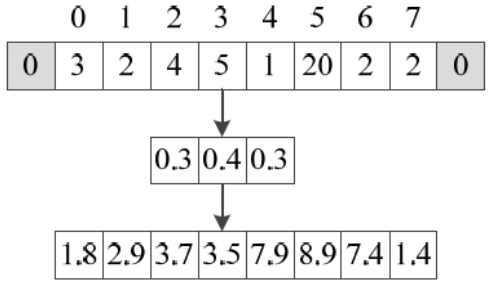
특징맵 feature map

$$s(i) = z \circledast u = \sum_{x=(h-1)/2}^{(h-1)/2} z(i+x)u(x)$$

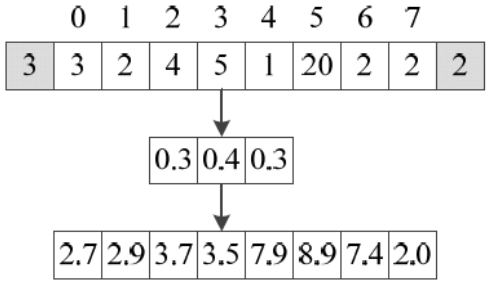


$$s(j,i) = z \circledast u = \sum_{y=(h-1)/2}^{(h-1)/2} \sum_{x=(h-1)/2}^{(h-1)/2} z(j+y, i+x)u(y,x)$$



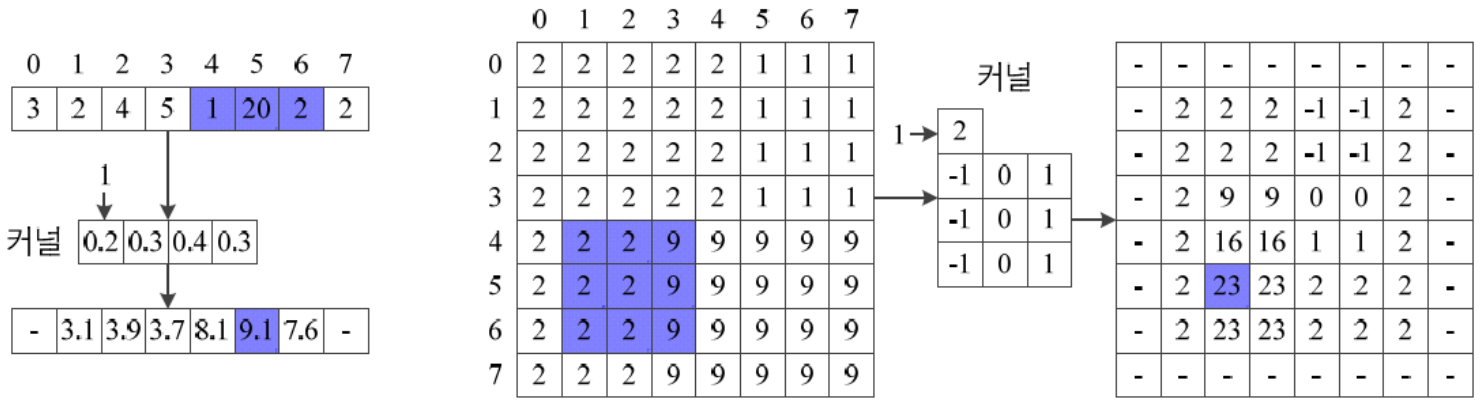


(a) 0 덧대기



(b) 복사 덧대기

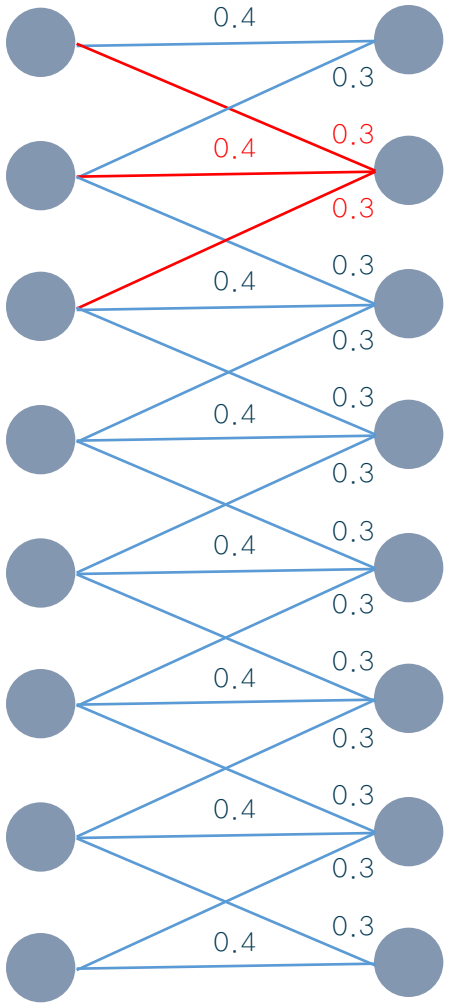
그림 4-7 덧대기(회색 노드가 덧댄 노드)



(a) 1차원 컨볼루션 (b) 2차원 컨볼루션

그림 4-8 바이어스

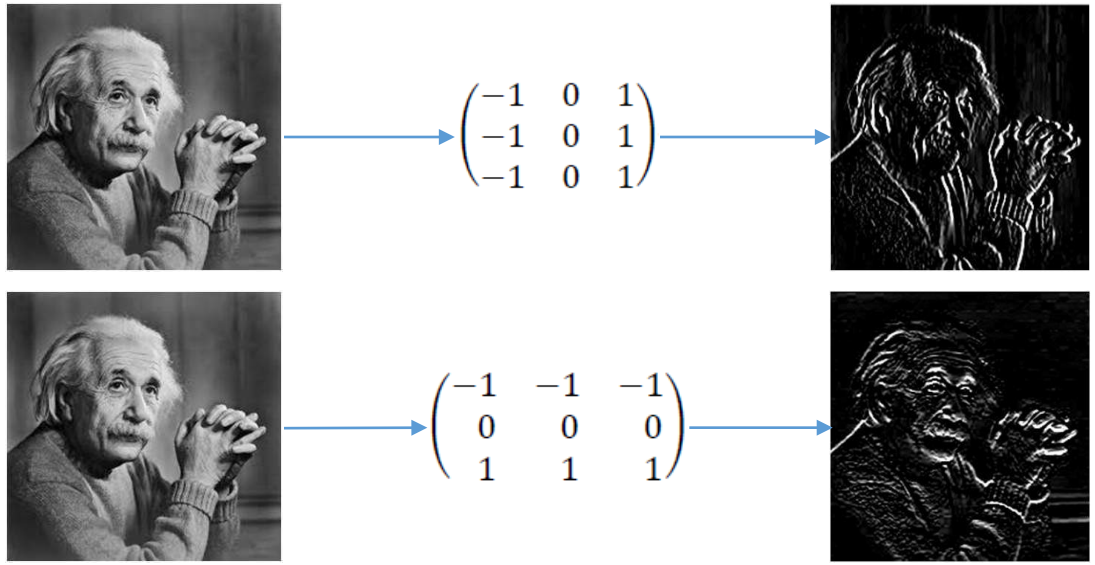
가중치 공유

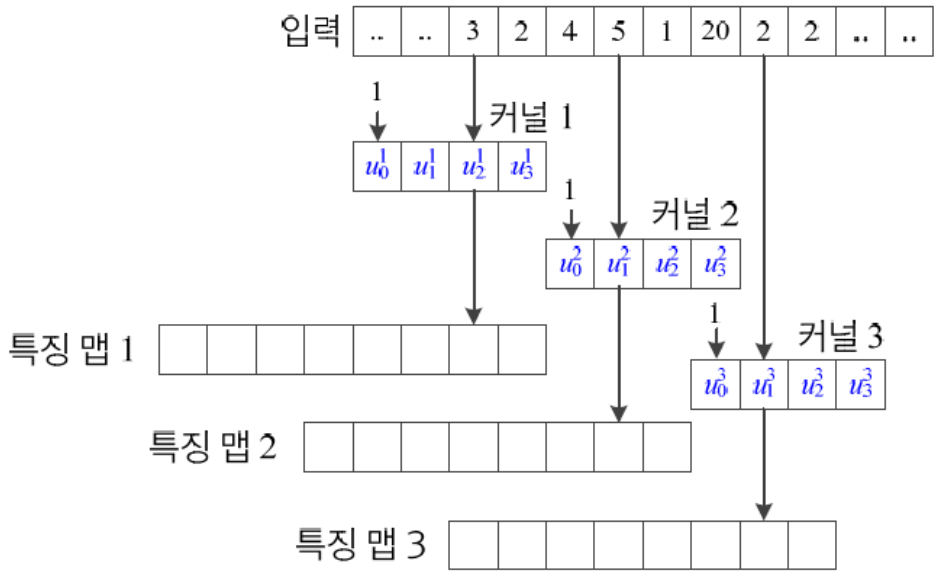


- 모든 노드가 동일한 커널을 사용
  (즉 가중치를 공유)하므로 매개변수는 3개에 불과
- 모델의 복잡도가 크게 낮아짐

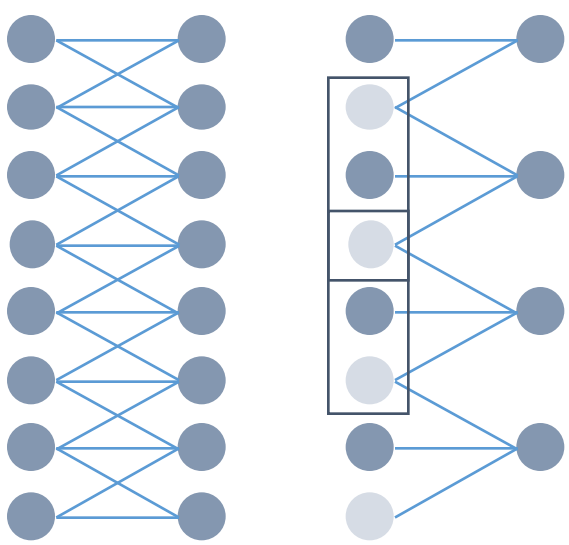
Convolution Neural Network 필터의 역할

커널의 값에 따라 커널이 추출하는 특징이 달라짐
하나의 커널만 사용하는 것보다 여러개의 커널을 이용하는 것이 특징 추출에 도움을 줌

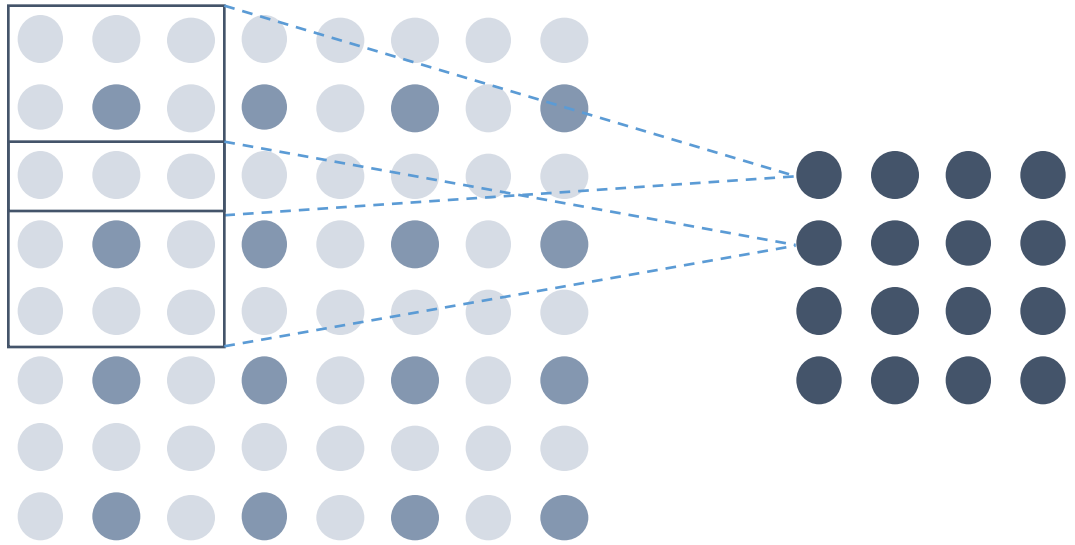



입력
.. .. 3 2 4 5 1 20 2 2 .. ..
1
커널 1
u_0^1 u_1^1 u_2^1 u_3^1
1
커널 2
u_0^2 u_1^2 u_2^2 u_3^2
1
커널 3
u_0^3 u_1^3 u_2^3 u_3^3
특징 맵 1
특징 맵 2
특징 맵 3

- 커널을 사람이 설계하지 않고, 학습으로 알아냄
- 예) 2차원 영상이 7*7 커널을 64개 사용한다면,
  학습은 (7*7+1)*64=3200개의 매개변수를 찾아내야 함
- DMLP와 마찬가지로 오류 역전파로 커널을 학습



1차원 데이터



2차원 데이터(예: 영상, 이미지)

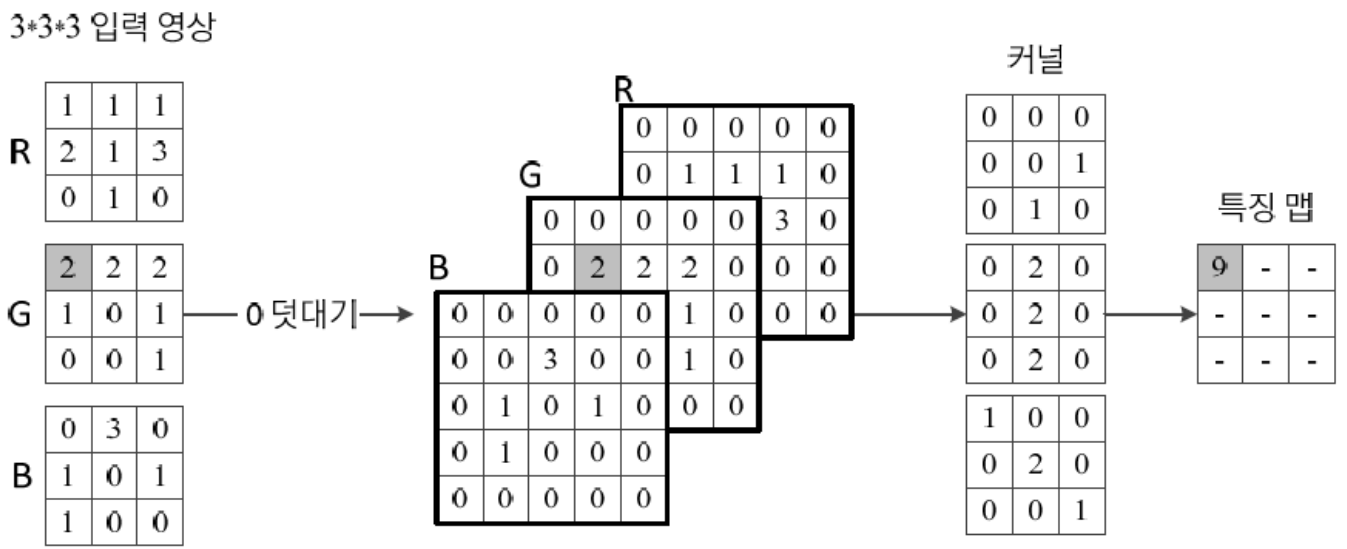
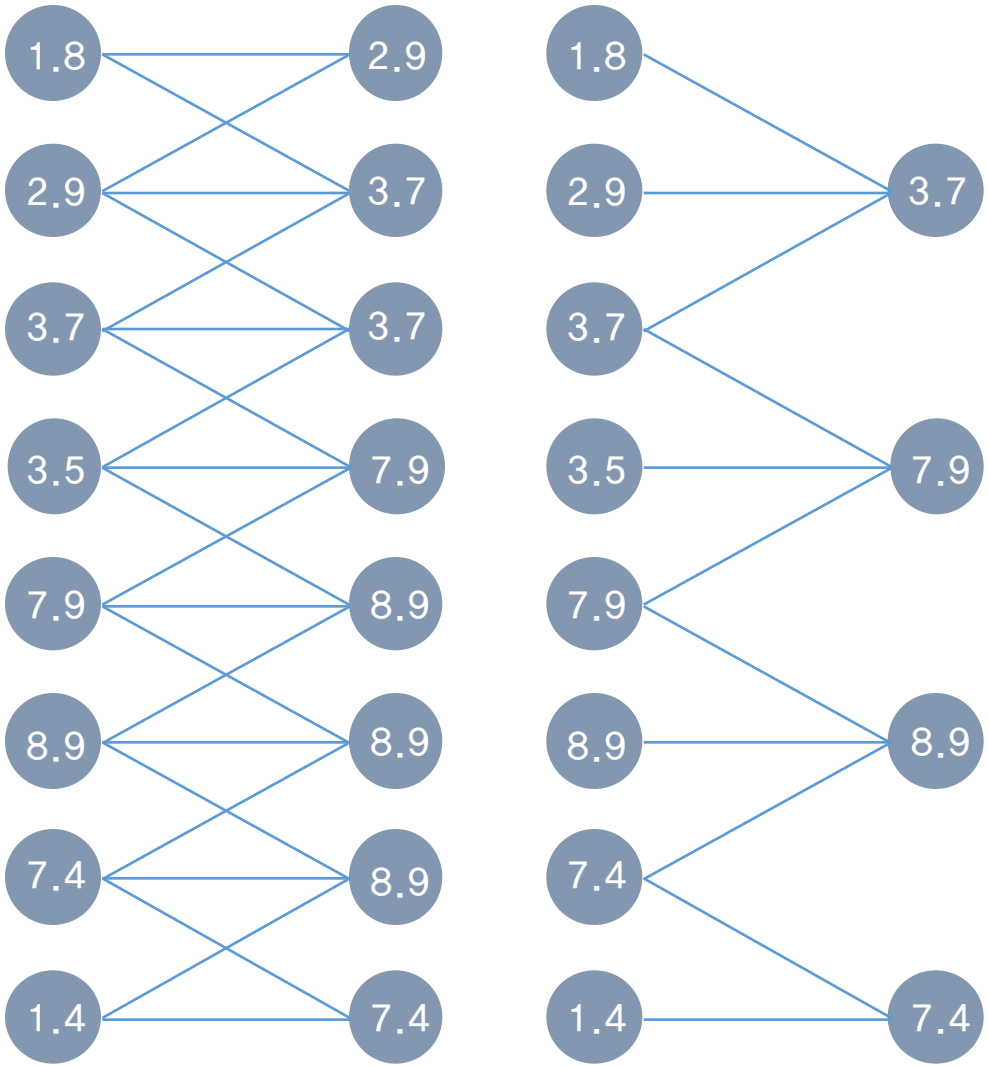


그림 4-14 텐서의 컨볼루션 연산(0 덧대기 적용)

$$\underbrace{\begin{pmatrix}0&0&0\\0&1&1\\0&2&1\end{pmatrix}}_{R}\underbrace{\begin{pmatrix}0&0&0\\0&2&2\\0&1&0\end{pmatrix}}_{G}\underbrace{\begin{pmatrix}0&0&0\\0&0&3\\0&1&0\end{pmatrix}}_{B}\circledast\underbrace{\begin{pmatrix}0&0&0\\0&0&1\\0&1&0\end{pmatrix}}_{c_1}\underbrace{\begin{pmatrix}0&2&0\\0&2&0\\0&2&0\end{pmatrix}}_{c_2}\underbrace{\begin{pmatrix}1&0&0\\0&2&0\\0&0&1\end{pmatrix}}_{c_3}=9$$



최대풀링, 평균 풀링, 가중치 평균 풀링

보폭(stride)을 크게 하면 **다운샘플링** 효과

풀링 연산의 특성

- 풀링은 상세 내용에서 요약통계를 추출함
- 매개변수가 없음
- 특징 맵의 수를 그대로 유지함
- 작은 이동에 둔감 –〉 물체인식이나 영상 검색 등에 효과적임

Convolution Neural Network 빌딩 블록

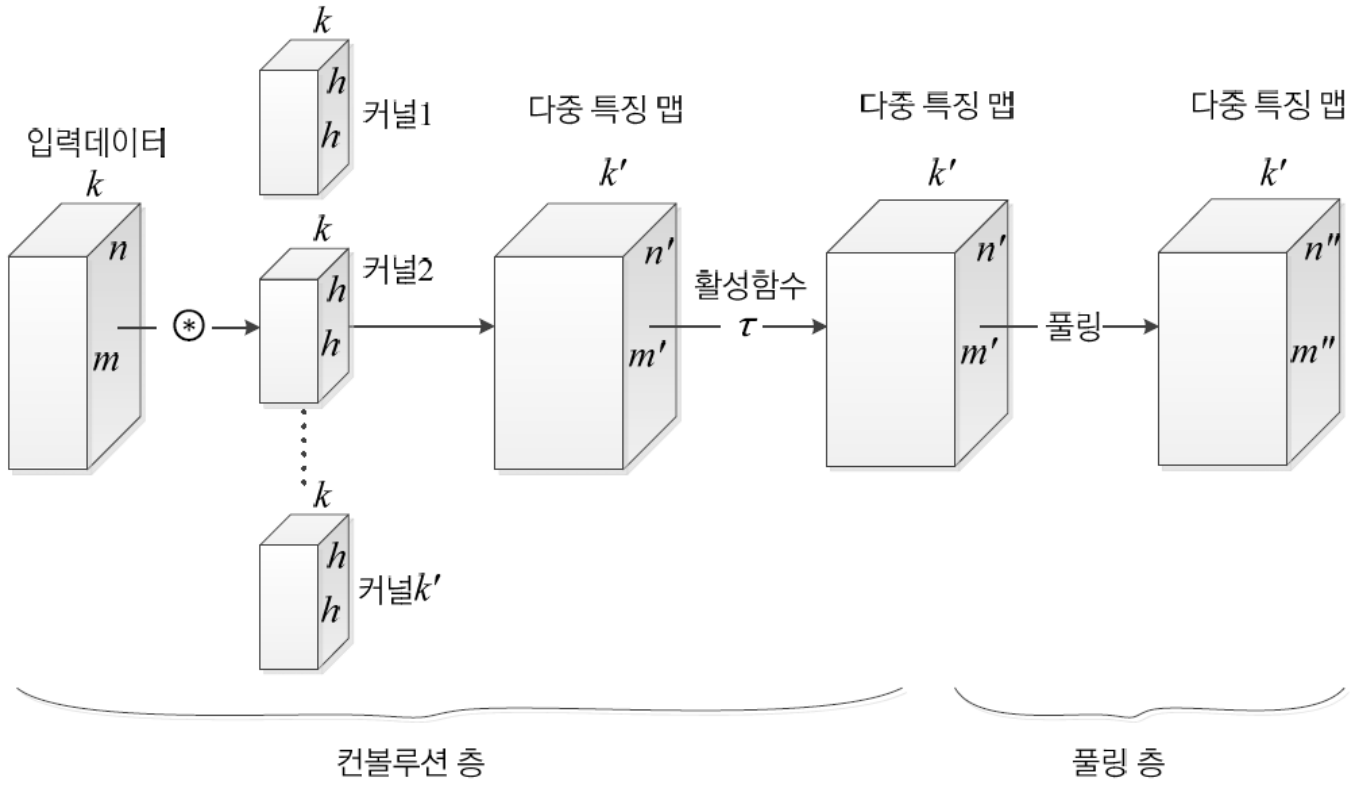
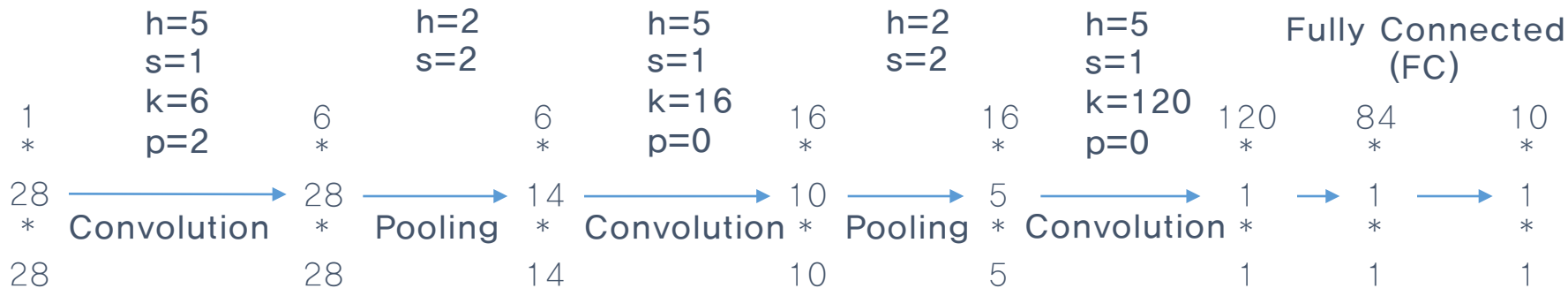
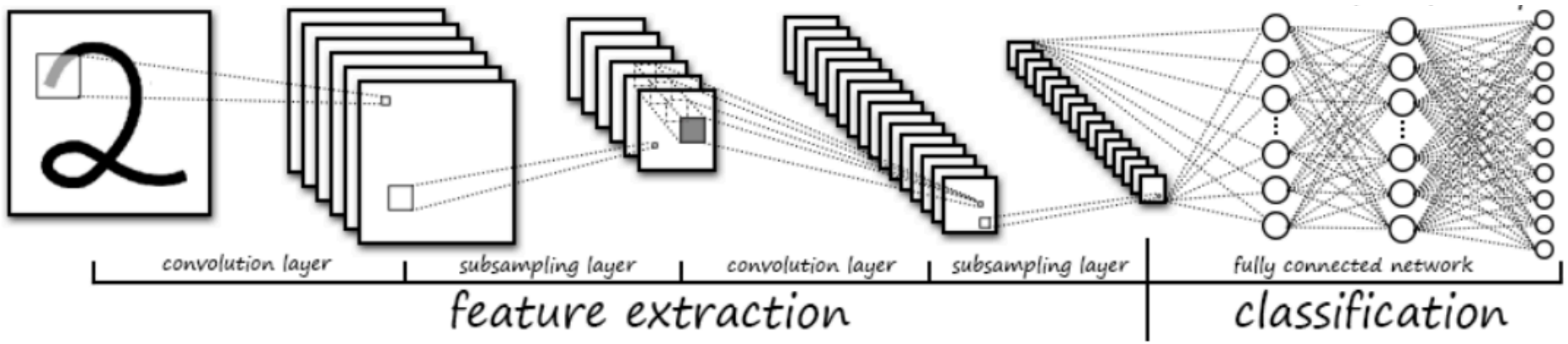


그림 4-18 CNN의 빌딩블록

- CNN은 빌딩블록을 이어 붙여 깊은 구조를 만듦
- 전형적인 빌딩블록 : 컨볼루션층 → 활성함수(주로 ReLU) → 풀링층
- 다중커널을 사용하여 다중 특징 맵을 추출함

Convolution Neural Network 응용사례 : LeNet-5





DMLP는 특징 벡터의 크기가 달라지면 연산 불가능

CNN은 가변 크기를 다룰 수 있는 강점

컨볼루션층에서 보폭을 조정한다거나, 풀링층에서 커널이나 보폭을 조정하여 특징 맵 크기를 조절

컨볼루션층 5개와 완전연결(FC)층 3개
8개 층에 290400-186624-64896-43264-4096-4096-1000개의 노드 배치
컨볼루션층은 200만개, FC층은 6500만개 가량의 매개변수
FC층에 30배 많은 매개변수 ☾ 향후 CNN은 FC층의 매개변수를 줄이는 방향으로 발전

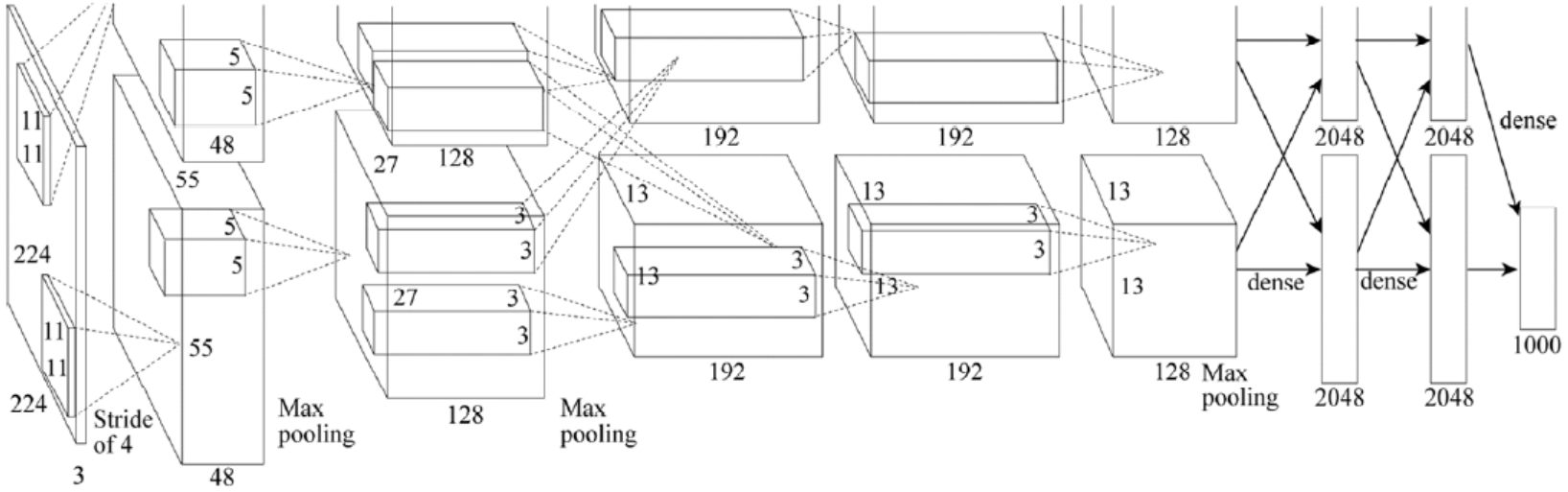
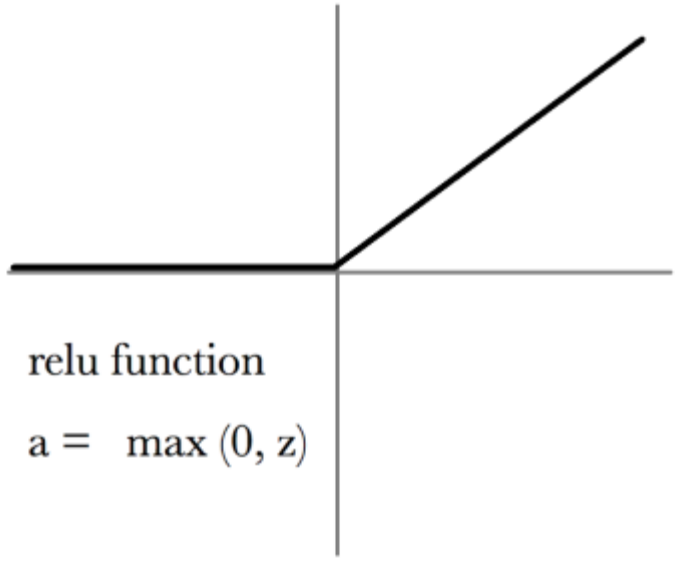


그림 4-21 AlexNet 구조[Krizhevsky2012]

AlexNet이 학습에 성공한 요인

외부 요인
- ImageNet이라는 대용량 데이터베이스
- GPU를 사용한 병렬처리

내부 요인
- 활성함수로 ReLU 사용
- 지역 반응 정규화 기법 적용
- 과잉적합 방지하는 여러 규제 기법 적용
- 데이터 확대(잘라내기와 반전으로 2048배로 확대)
- 드롭아웃 등



VGGNet의 핵심 아이디어

3*3의 작은 커널을 사용하여 신경망을 더욱 깊게 만듦

컨볼루션층 8~16개를 두어 AlexNet의 5개에 비해 2~3배 깊어짐

16층짜리 VGG-16(컨볼루션 13층+FC 3층)

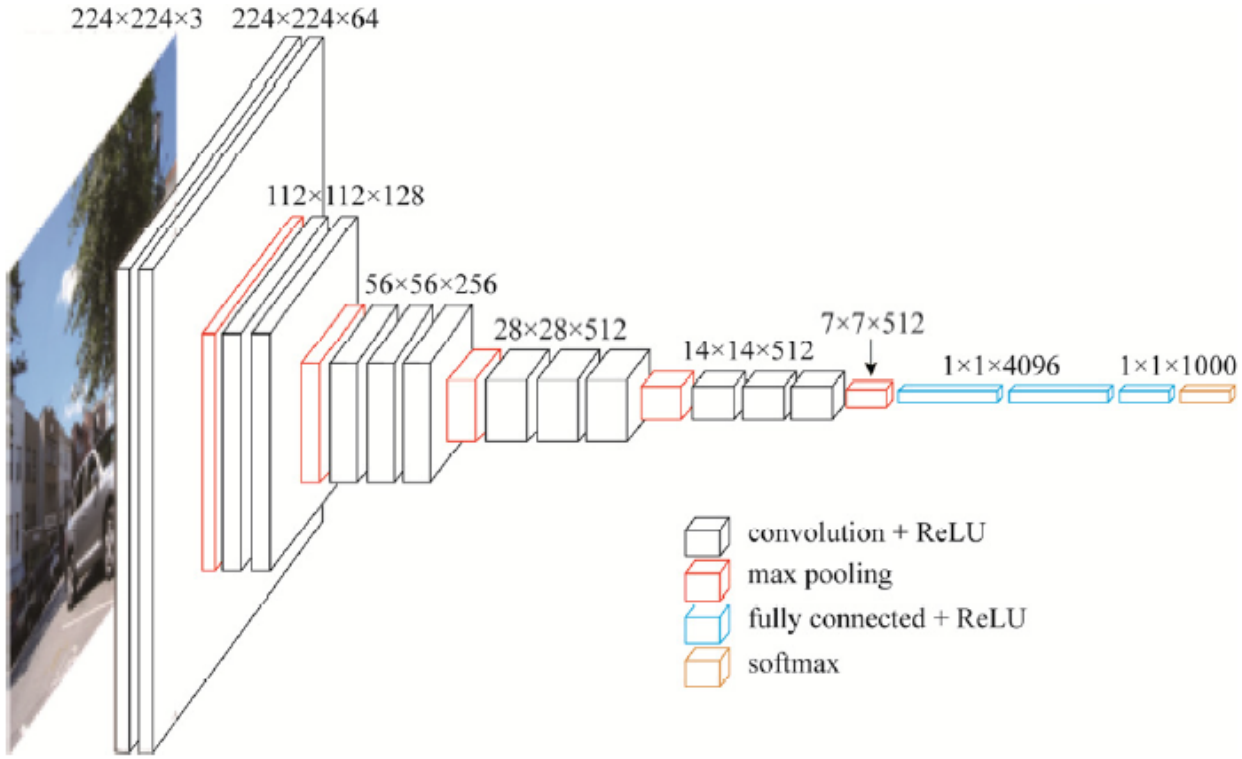


그림 4-22 VGGNet 구조[Simonyan2015]

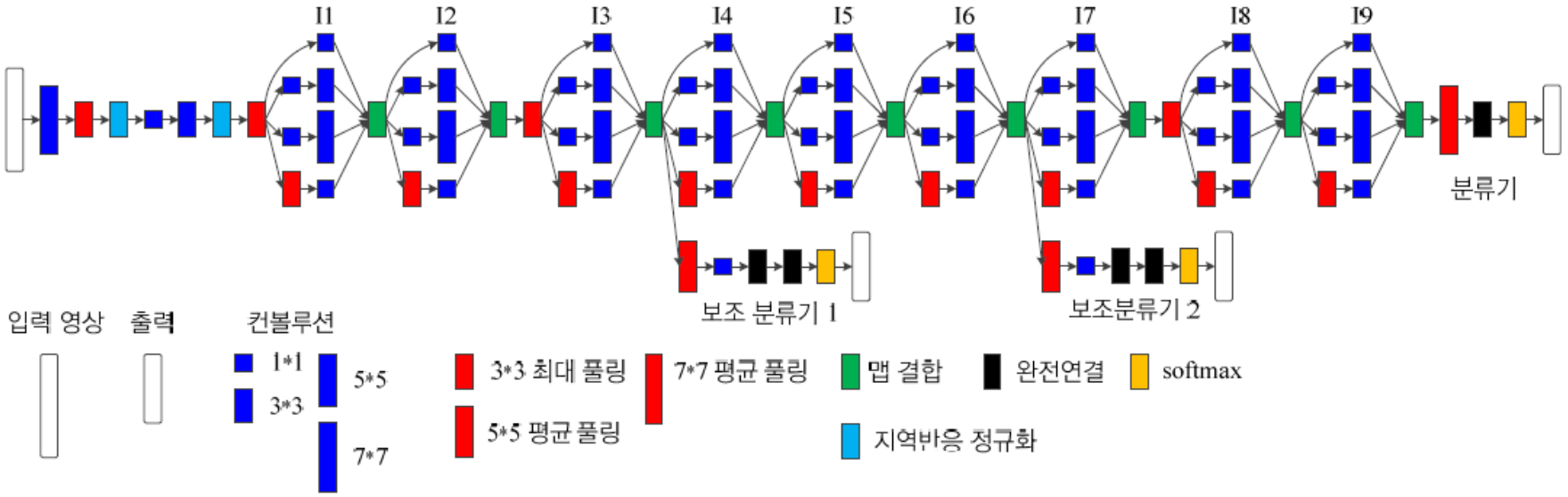


그림 4-27 GoogLeNet의 구조

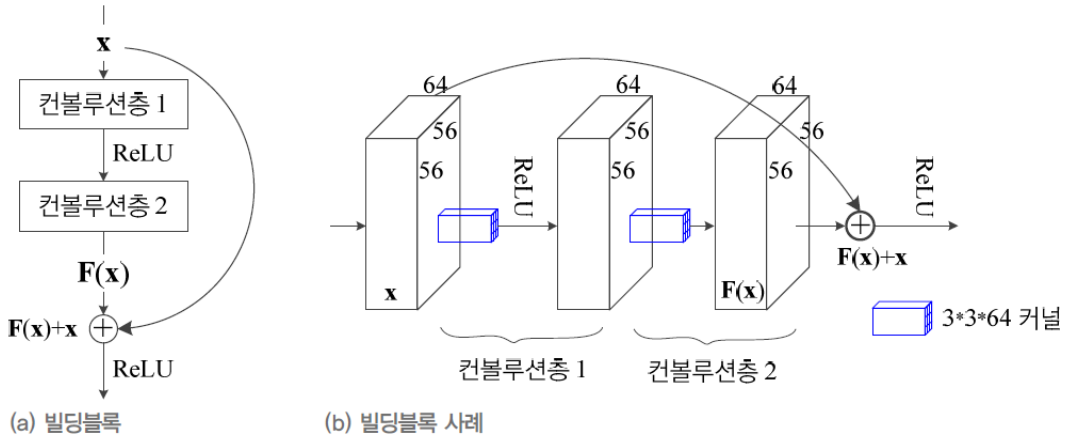


그림 4-28 잔류 학습의 구조와 동작

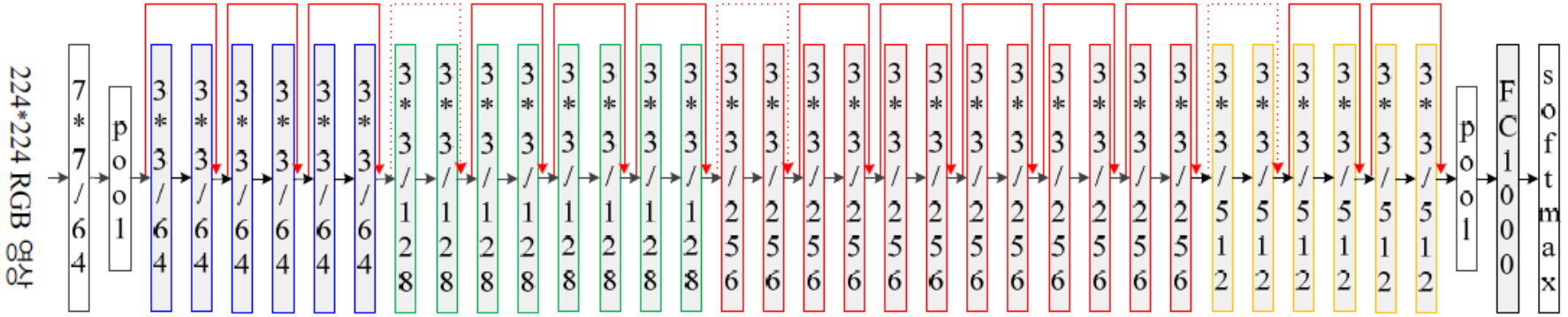


그림 4-29 ResNet 예제(34층)

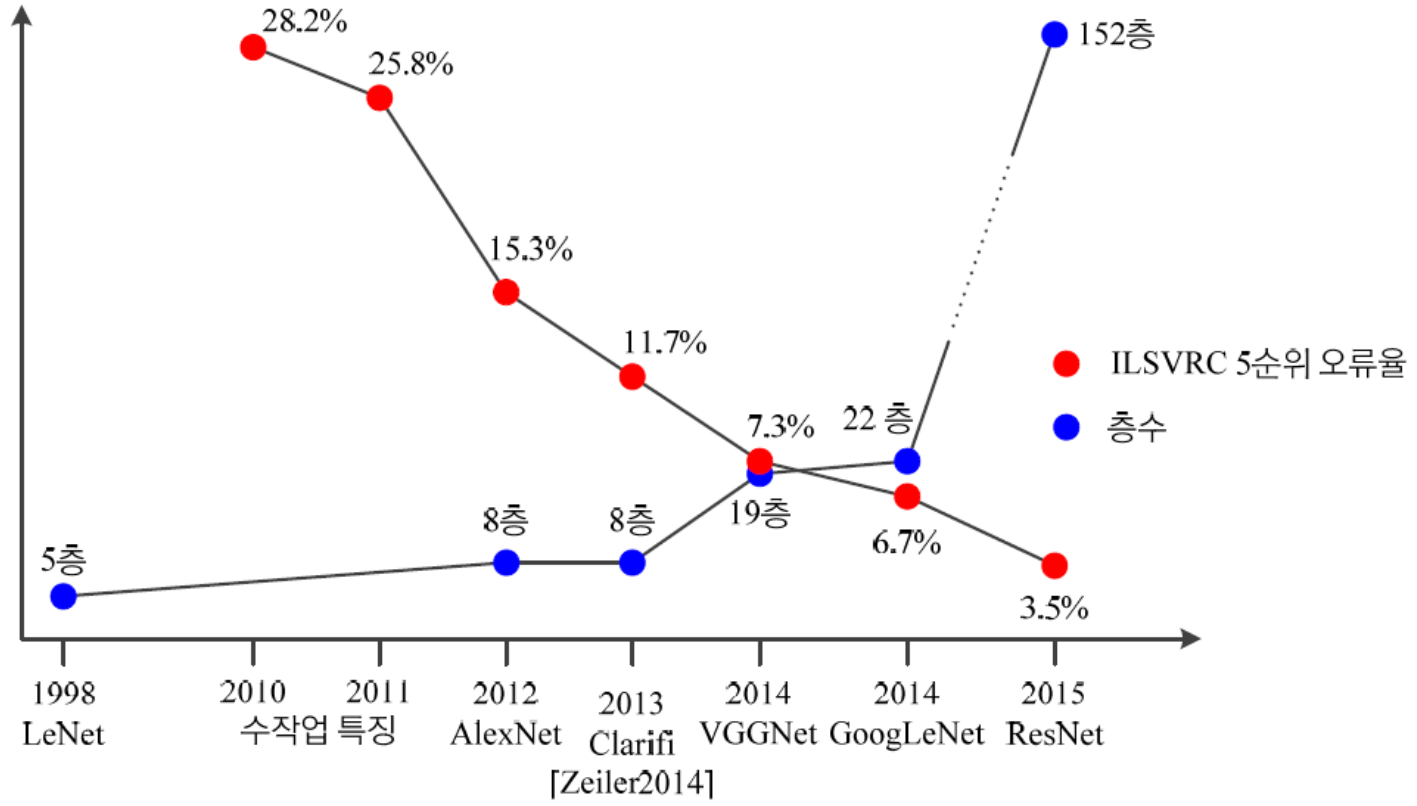
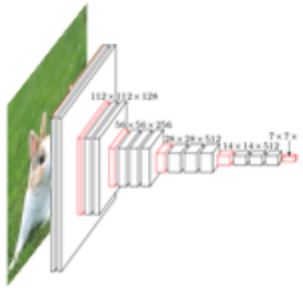
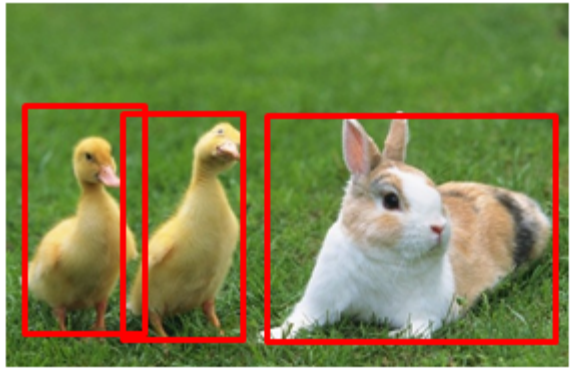


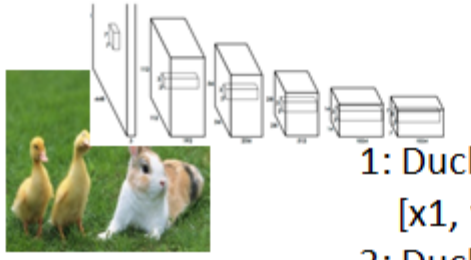
그림 4-30 CNN의 발전 추세



bunny

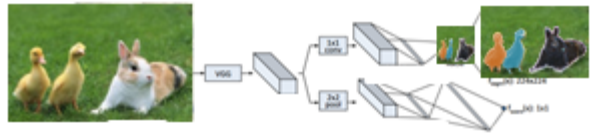
Classification





1: Duck ,
[x1, y1, x2, y2]
2: Duck,
[x1, y1, x2, y2]
3: Bunny,
[x1, y1, x2, y2]

Object Detection



Segmentation