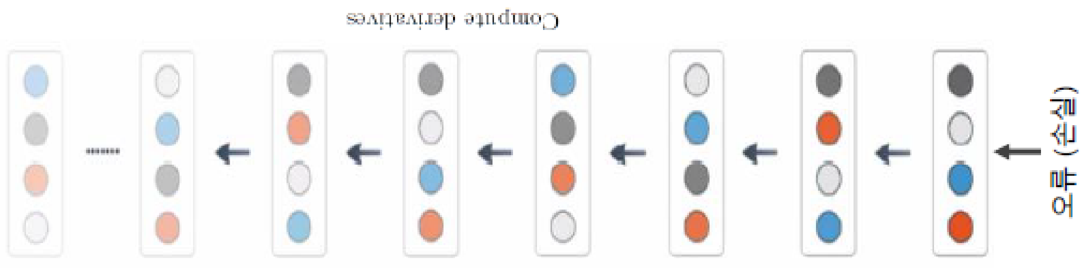
딥러닝

- 다층 퍼셉트론에 은닉층을 여러 개 추가하면 깊은 신경망이 됨
- 딥러닝은 깊은 신경망을 학습시킴
- 딥러닝은 새로운 응용을 창출하고 인공지능 제품의 성능을 획기적으로 향상
    - 현대 기계 학습을 주도

딥러닝의 등장

- 배경
    - 1980년대에 이미 깊은 신경망 아이디어 등장
    - 하지만 실현 불가능(당시, 깊은 신경망은 학습이 안 됨)
        - 그레디언트 소멸 문제
        - 작은 훈련 집합
        - 과다한 연산과 시간 소요 (값비싼 슈퍼컴퓨터)
    - 일부 연구자들은 실망스러운 상황에서도 지속적인 연구
        - 학습률에 따른 성능 변화 양상
        - 모멘텀의 영향
        - 은닉 노드 수에 따른 성능 변화
        - 데이터 전처리의 영향
        - 활성함수의 영향
        - 규제 기법의 영향
- 그레디언트 소멸 문제



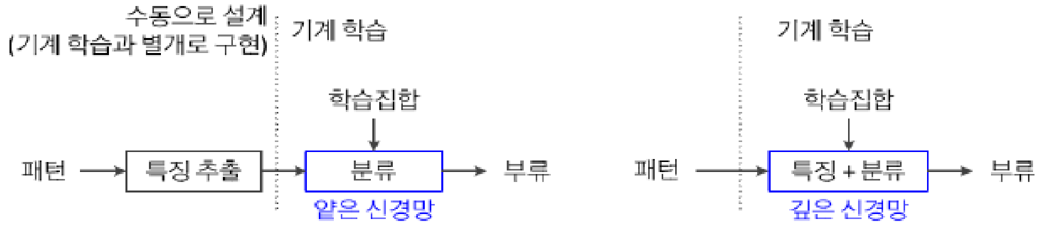
딥러닝의 기술 혁신 요인

- 요인
    - **컨볼루션 신경망**(CNN)이 딥러닝의 가능성을 엶
        - 매개변수의 공유를 통해서 효율적인 학습 접근 제공
    - 계산은 단순한데 성능은 더 **좋은 활성함수**
    - 과잉적합을 방지하는데 효과적인 **다양한 규제기법**
    - 층별 **예비학습** 기법 개발
    - 값싼 **GPGPU**의 등장
    - 인터넷 덕분으로 **학습 데이터 양과 질의 향상**

특징 학습의 부각

- 기계학습의 패러다임의 변화

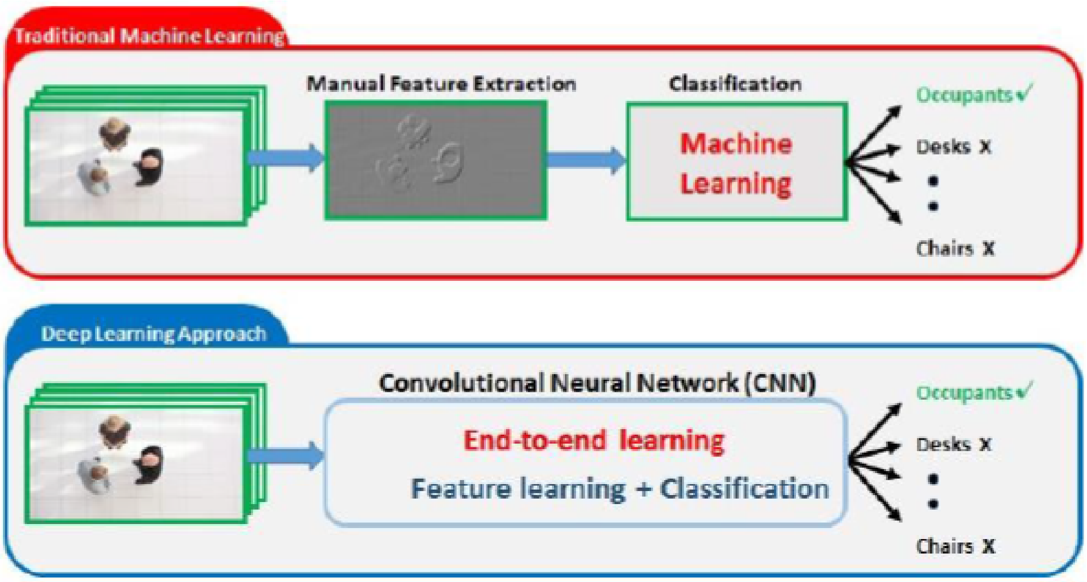
  - 고전적인 다층 퍼셉트론
    - 은닉층은 **특징 추출기**
    - 얕은 구조(제한적 특징 추출)이므로 가공하지 않은 획득한 원래 패턴을 그대로 입력하면 **낮은 성능**
    - 따라서 사람이 **수작업** 특징을 선택하거나 추출하여 신경망에 입력함
  - 현대 기계학습 (딥러닝)
    - 데이터로부터 **특징 추출**하도록 학습 <- **특징 학습**
    - 전체 특징을 신경망의 입력 <- 종단간 학습



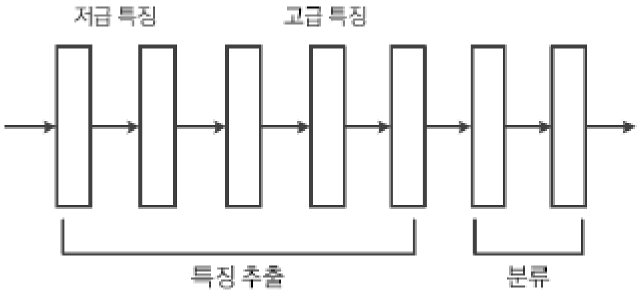
(a) 고전적 패러다임에서의 수작업 특징 (b) 딥러닝에서의 특징 학습

그림 4-1 기계 학습의 패러다임 변화

- 고전적인 기계학습과 딥러닝 비교 예



- 특징 학습
  - **낮은 단계 은닉층**은 선이나 모서리와 같은 **간단한(저급) 특징** 추출
  - **높은 단계 은닉층**은 추상적인 형태의 **복잡한(고급) 특징**을 추출
  - 특징 학습이 강력해짐에 따라
    - 기존 응용에서 획기적인 성능 향상
      - 영상 인식, 음성 인식, 언어 번역 등
    - 새로운 응용 창출
      - 분류나 회귀뿐 아니라 생성 모델이나 화소 수준의 영상 분할
      - CNN과 LSTM의 협력 모델 등이 가능해짐



깊은 다층 퍼셉트론

구조와 동작

- 깊은 다층 퍼셉트론의 구조
  - **입력층**(d+1개의 노드)과 **출력층**(c개의 노드)
  - L-1개의 **은닉층** (입력층은 0번째 은닉층, 출력층은 L번째 은닉층으로 간주)
    - l번째 은닉층의 노드 수를 n_l로 표기

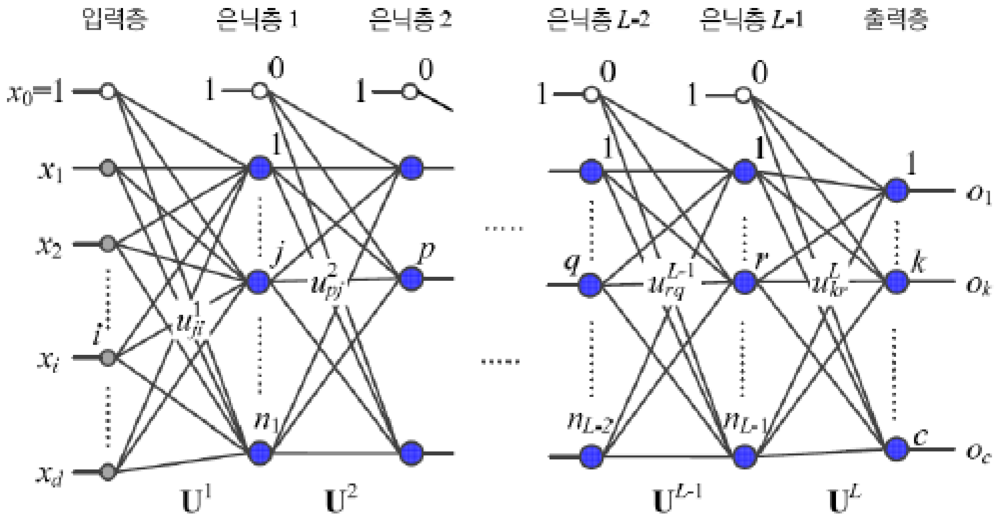


그림 4-3 깊은 MLP(DMLP)의 구조

- DMLP의 **가중치 행렬**
  - u_{ji}^l은 l-1번째 층의 i번째 노드와 l번째 층의 j번째 노드를 연결하는 가중치
  - l-1번째 층과 l번째 층을 연결하는 가중치는 총 $(n_{l-1}+1)n_l$개

$$\text{가중치 행렬:}\quad \mathbf{U}^l = \begin{pmatrix} u_{10}^l & u_{11}^l & \cdots & u_{1n_{l-1}}^l \\ u_{20}^l & u_{21}^l & \cdots & u_{2n_{l-1}}^l \\ \vdots & \vdots & \ddots & \vdots \\ u_{n_l0}^l & u_{n_l1}^l & \cdots & u_{n_ln_{l-1}}^l \end{pmatrix}, l = 1, 2, \cdots, L$$

- DMLP의 **동작**
  - **MLP의 동작**을 나타내는 식 (3.12)를 보다 많은 단계로 **확장**한 것

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

- **동작**을 구체적으로 쓰면
  - 입력층의 특징 벡터를 내부 표현으로 바꾸어 쓰면

$$\mathbf{z}^0 = (z_0, z_1, z_2, \cdots, z_{n_0})^{\mathrm{T}} = (1, x_1, x_2, \cdots, x_d)^{\mathrm{T}}$$

- l번째 층의 j번째 노드가 수행하는 연산

$$z_j^l = \tau_l(s_j^l)$$

이때 $s_j^l = \mathbf{u}_j^l \mathbf{z}^{l-1}$이고,

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

- 행렬 표기를 이용하여 **l번째 층의 연산** 전체를 쓰면

$$l\text{번째 층의 연산: } \mathbf{z}^l = \boldsymbol{\tau}_l(\mathbf{U}^l \mathbf{z}^{l-1}), \ 1 \le l \le L$$

학습

- **DMLP 학습**은 3장의 **MLP 학습**과 **유사**
  - DMLP는 그레디언트 계산과 가중치 갱신을 **더 많은 단계**에 걸쳐 수행
- **오류 역전파** 알고리즘
  - L번째 층(출력층)의 그레디언트 계산

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

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

  - l+1번째 층의 정보를 이용하여 l번째 층의 그레디언트 계산

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

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

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 $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)^T$, $\mathbf{y} = (y_1, y_2, \cdots, y_c)^T$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)

- 역사적 고찰
  - 학습 알고리즘의 주요 개선

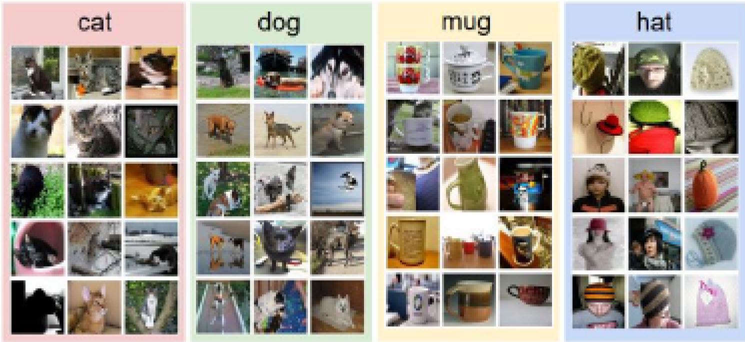
	퍼셉트론 →	다층 퍼셉트론 →	깊은 다층 퍼셉트론
활성함수:	계단함수	시그모이드함수	ReLU와 변형들
목적함수:	평균제곱 오차	평균제곱 오차	교차 엔트로피 또는 로그우도

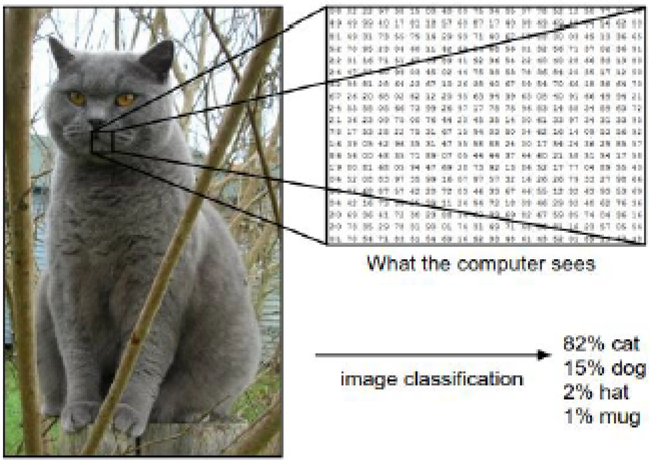
그림 4-4 다층 퍼셉트론의 역사적 발전 양상

  - CNN의 부상
    - MNIST 인식 경쟁
    - ILSVRC 영상 인식 경쟁 : **CNN**이 DMLP보다 확연히 **우월**

컨볼루션 신경망

- **영상 인식**의 예





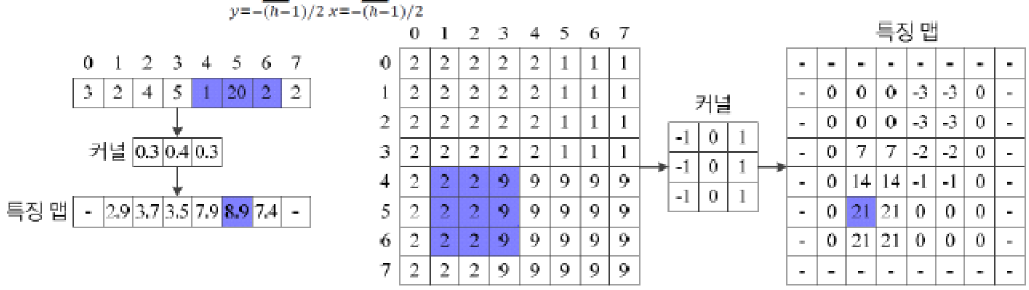
- 오늘날 영상 분야에서 다양하게 활용 됨
    - 분류, 검색, 검출, 분할
- DMLP와 CNN의 비교
    - DMLP
        - 완전 연결 구조로 높은 복잡도
        - 학습이 매우 느리고 과잉적합 우려
    - CNN
        - **컨볼루션 연산**을 이용한 **부분연결(희소 연결)** 구조로 **복잡도 크게 낮춤**
        - **컨볼루션 연산**은 좋은 **특징 추출**
- CNN
    - **격자 구조**(영상, 음성 등)를 갖는 **데이터**에 적합
    - **수용장**은 인간시각과 유사
    - 가변 크기의 입력 처리 가능
- CNN의 완전 연결 신경망과 차별
    - 각 층의 입출력의 특징형상 유지
    - 영상의 **공간정보**를 유지하면서 공간적으로 **인접한 정보의 특징을 효과적으로 인식**
    - 학습에 의해 결정된 **복수의 커널들(혹은 필터들)**에 **대응되는 특징들을 추출하는 층**을 가짐
    - 추출된 영상의 **특징을 요약하고 강화**하는 층을 가짐
    - 각 커널은 **파라미터를 공유함**으로써 완전 연결 신경망 대비 **학습 파라미터가 매우 적음**

컨볼루션층(CONV)

- **컨볼루션** 연산
    - 컨볼루션은 해당하는 요소끼리 곱하고 결과를 모두 더하는 **선형 연산 (합성곱)**
    - 식 (4.10)과 식 (4.11)에서 u는 **커널**(혹은 **필터**), z는 **입력**, s는 **출력 (특징 맵)**
        - 영상에서 **특징을 추출하기 위한 용도**로 사용됨(**공간 필터**)

$$s(i) = z \circledast u = \sum_{x=-(h-1)/2}^{(h-1)/2} z(i+x)u(x) \qquad (4.10) \longleftarrow \text{1차원 입력}$$

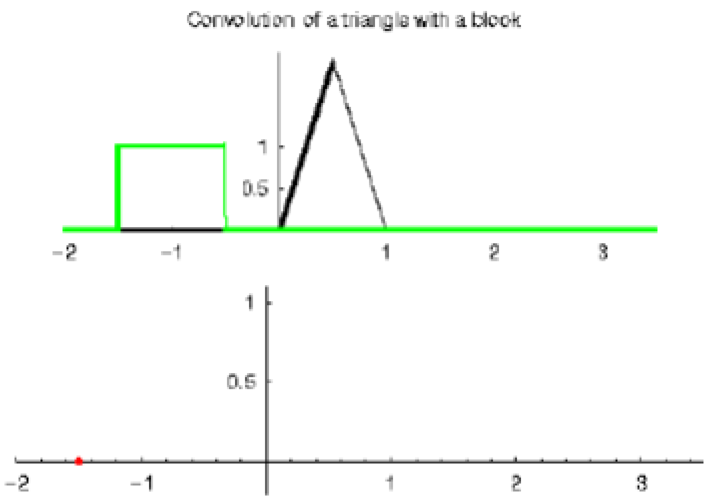
$$s(j,i) = z \circledast u = \sum_{y=-(h-1)/2}^{(h-1)/2} \sum_{x=-(h-1)/2}^{(h-1)/2} z(j+y, i+x)u(y,x) \qquad (4.11) \longleftarrow \text{2차원 입력}$$



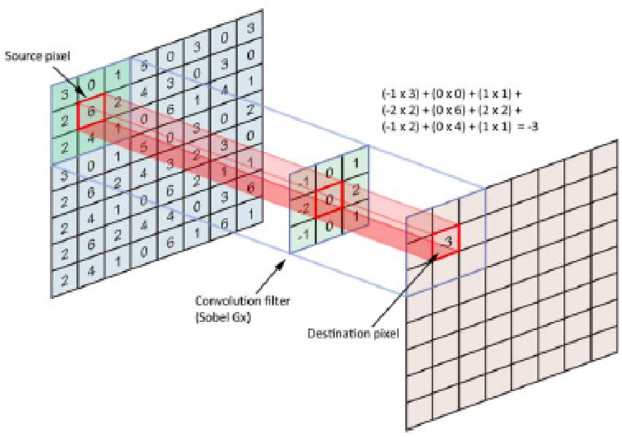
(a) 1차원 컨볼루션 (b) 2차원 컨볼루션

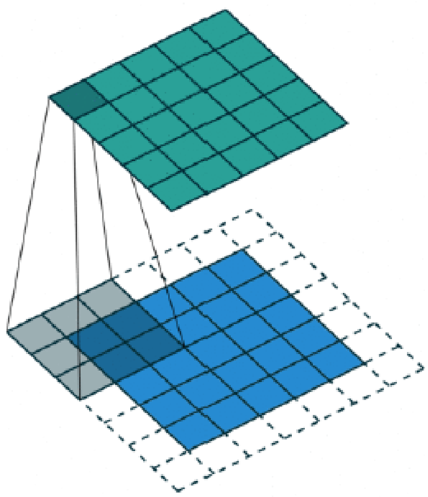
그림 4-6 **컨볼루션 연산**

- **1차원** 컨볼루션 연산의 예

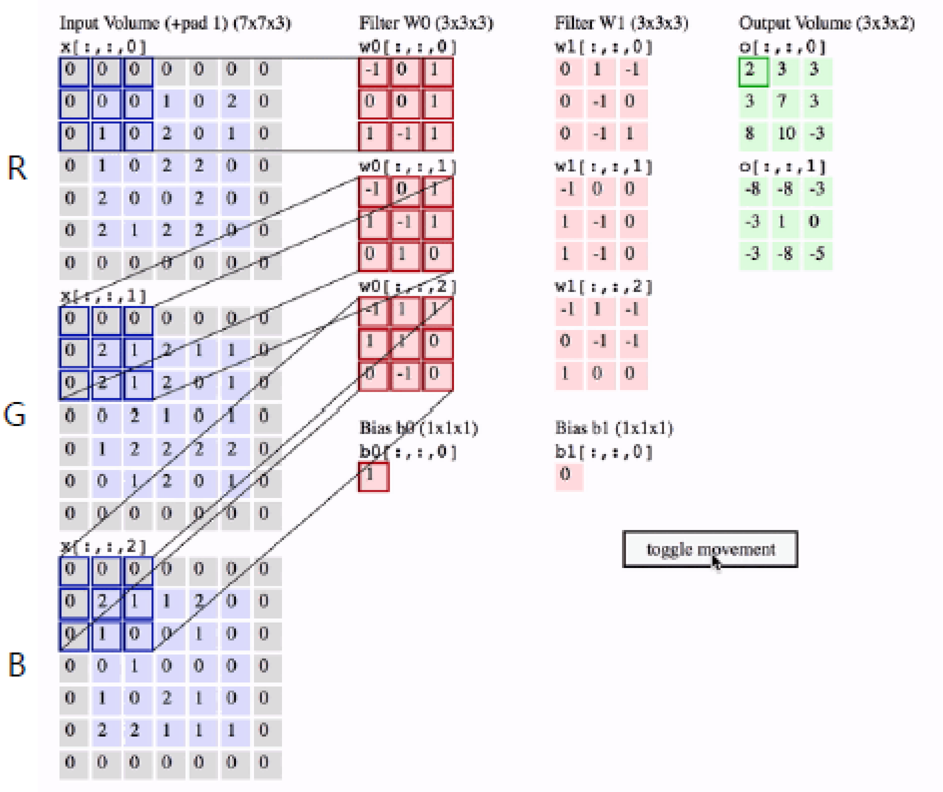


- **2차원** 컨볼루션 연산의 예





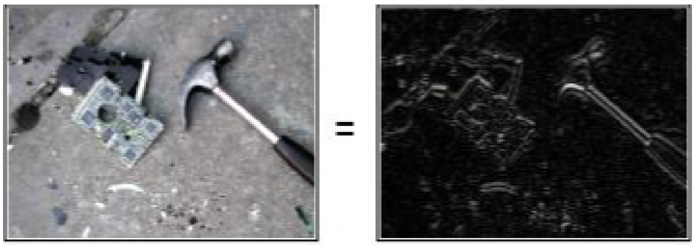
- **3차원**(혹은 **채널**) 컨볼루션 연산의 예



- **영상**에서의 **컨볼루션** 연산 예

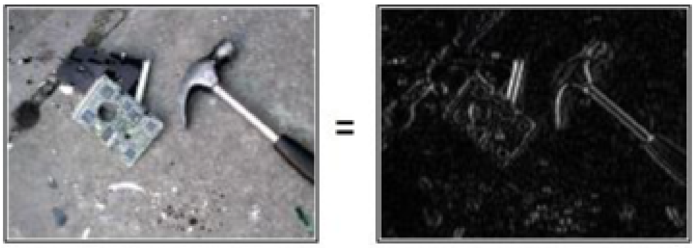
$$\begin{Bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{Bmatrix} \times$$

Horizontal



$$\begin{Bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{Bmatrix} \times$$

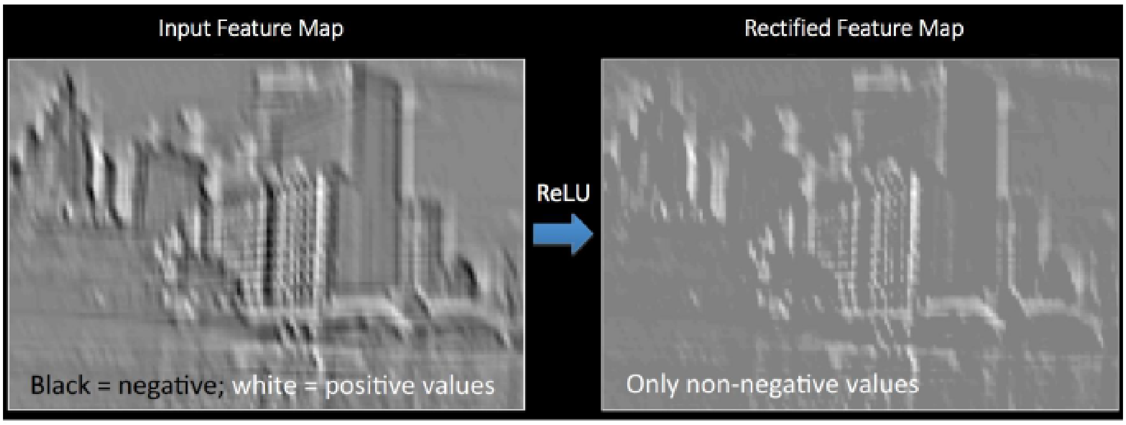
Vertical



- **영상**에서의 **다수의 컨볼루션** 연산 예

- 커널의 값에 따라 커널이 추출하는 특징이 달라짐

- **영상**에서의 **ReLU(활성함수)** 연산의 예



활성화된 특징 맵

- **덧대기**(Padding)
  - 가장자리에서 영상의 크기가 줄어드는 효과 방지(각 층의 입출력의 특징 형상 유지)

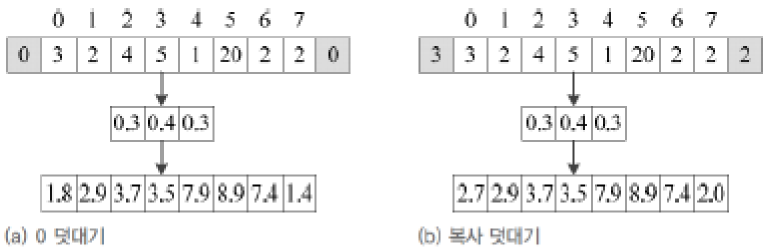


그림 4-7 덧대기(회색 노드가 덧댄 노드)

- 바이어스 추가

그림 4-8 바이어스

- **가중치 공유** 혹은 **묶인 가중치**
  - 모든 **노드가 동일한 커널**을 사용
    - 가중치를 공유하므로 매개변수는 3개에 불과
  - 모델의 복잡도가 크게 낮아짐

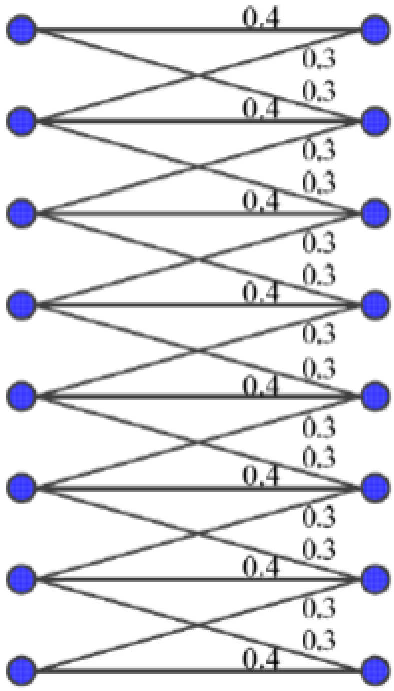


그림 4-9 CNN의 가중치 공유

- **다중 특징 맵 추출**
  - **커널의 값에 따라 커널이 추출하는 특징이 달라짐**

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}: \text{수직방향}, \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}: \text{수평방향 선 혹은 모서리 추출}$$

  - 따라서 하나의 커널만 사용하면 너무 빈약한 특징이 추출됨
  - 3개 커널을 사용하여 3개 특징 맵을 추출하는 상황

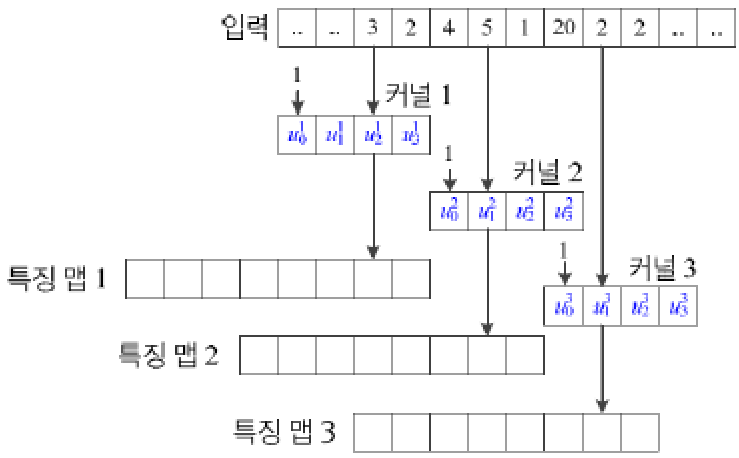


그림 4-10 다중 특징 맵 추출

    - 실제로는 수십 ~ 수백 개의 커널을 사용
- **특징 학습**
  - 커널을 사람이 설계하지 않고, **학습으로 찾음**
    - u_i^k는 k번째 커널의 i번째 매개변수

        - 2차원 영상이 7*7 커널을 64개 사용한다면 학습은 (7*7 +1)*64 = 3200개의 매개변수 필요
    - DMLP와 마찬가지로 **오류 역전파로 커널을 학습**
- 컨볼루션 연산에 따른 CNN의 특성
    - 이동에 동변 (신호가 이동하면 이동 정보가 그대로 특징 맵에 반영)
        - 영상 인식에서 물체 이동이나 음성 인식에서 발음 지연에 효과적으로 대처

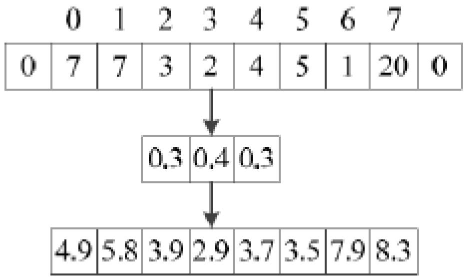


그림 4-11 CNN의 이동에 동변한 특성

    - 병렬분산 구조
        - 각 노드는 독립적으로 계산 가능하므로 병렬 구조
        - 노드는 깊은 층을 거치면서 전체에 영향을 미치므로 분산 구조

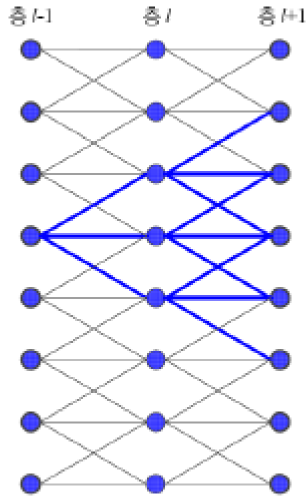
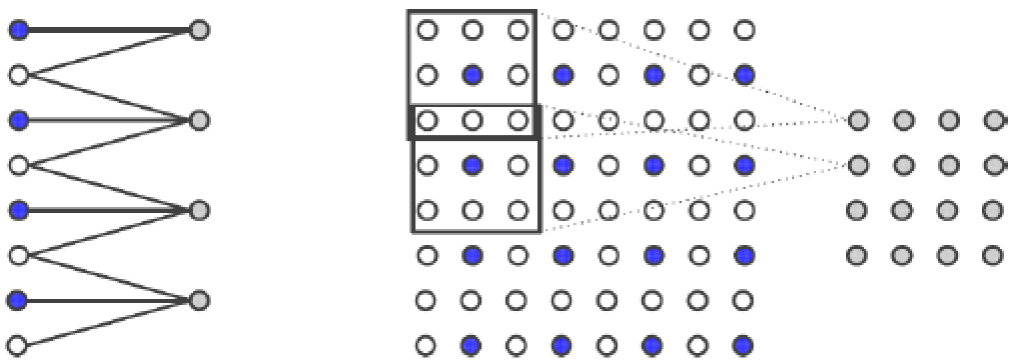


그림 4-12 CNN의 병렬 분산 구조

- 큰 **보폭**에 의한 **다운샘플링**
    - 지금까지는 모든 화소에 커널 적용 -> 보폭을 1로 설정한 셈
    - 일반적으로 보폭이 k이면, k개 마다 하나씩 샘플링하여 커널 적용
        - 2차원 영상의 경우 특징 맵이 $1/k^2$로 작아짐



(a) 1차원 데이터(예: 음성) (b) 2차원 데이터(예: 영상)

그림 4-13 보폭이 2인 컨볼루션 연산

- **텐서**에 적용
  - 3차원 이상의 구조에도 적용 가능
    - ex) RGB 컬러 영상은 3mn의 3차원 텐서

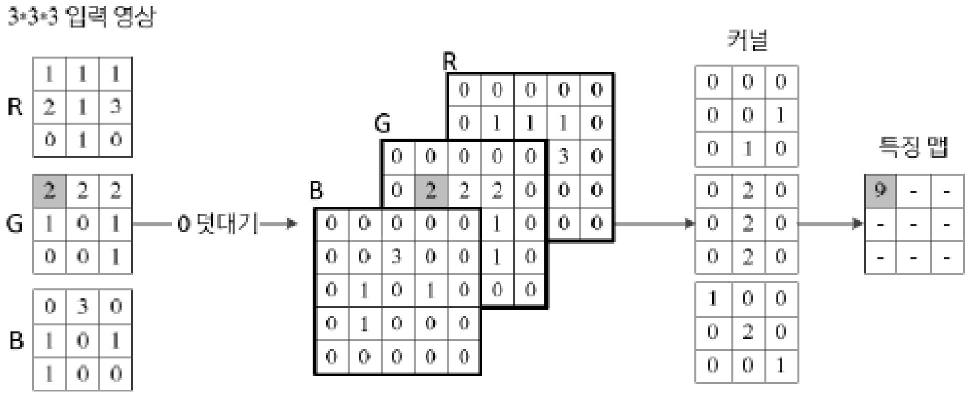
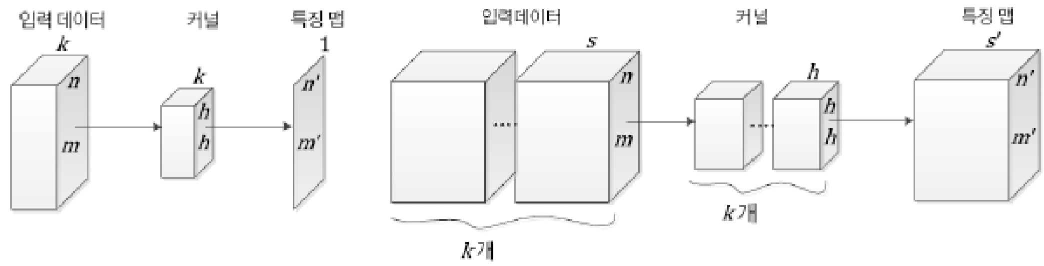


그림 4-14 텐서의 컨볼루션 연산(0 덧대기 적용)

  - 특징 맵의 회색 노드의 계산 예시

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

- **3차원** 구조의 데이터에 적용
  - 채널이 k개인 3차원 격자 구조



(a) 다중채널 데이터(예: RGB 컬러 영상) (b) 3차원 데이터(예: 동영상, MRI 뇌 영상)

그림 4-15 텐서의 컨볼루션 연산(직육면체로 표현하기)

    - [그림 4-14]를 블록 형태로 다시 그린 것
  - 4차원 텐서로 표현하는 데이터
    - 컬러 동영상($3 * s * m * n$), MRI 뇌영상($1 * s * m * n$)
    - $k * h * h * h$커널을 $s * m * n$ 공간을 이동하면서 적용

풀링층(POOL)

- **풀링** 연산
  - **최대 풀링**, **평균 풀링**, 가중치 평균 풀링 등

○ 보폭을 크게 하면 다운샘플링 효과

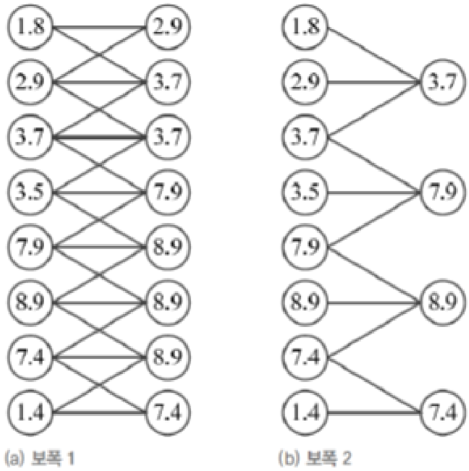
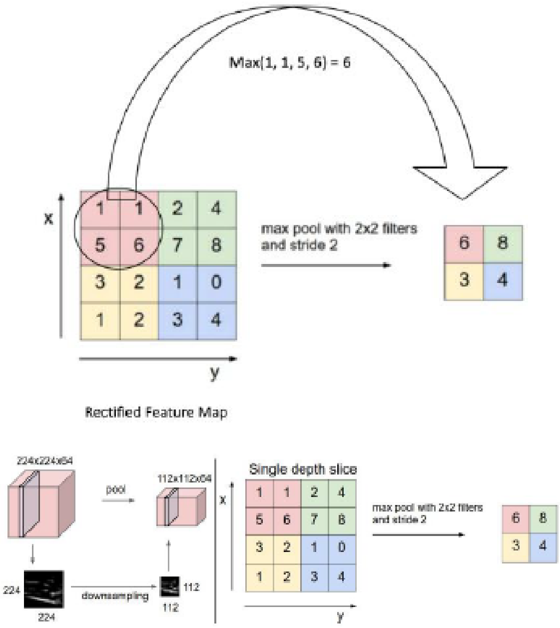


그림 4-16 최대 풀링



- 풀링 연산의 특성
  - 풀링은 상세 내용에서 요약 혹은 평균 등의 통계적 대표성을 추출
  - 매개변수가 없음
  - 특징 맵의 수를 그대로 유지함(크기 X)
  - 작은 변화에 둔감 -> 물체 인식이나 영상 검색 등에 효과적임

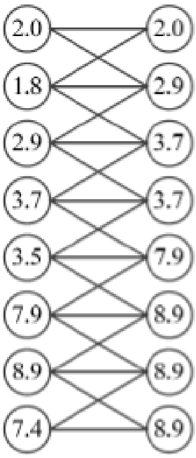
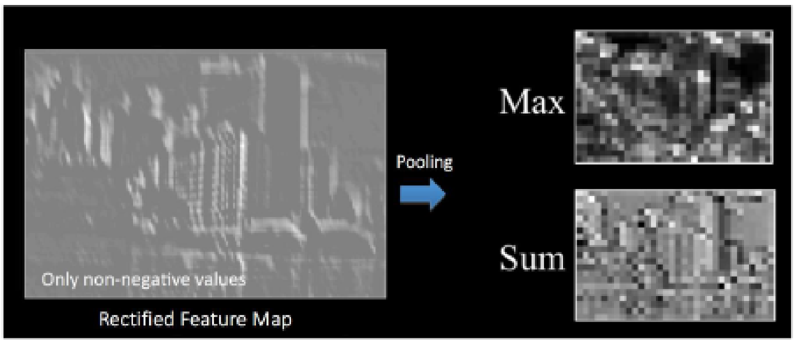


그림 4-17 작은 이동에 둔감한 최대 풀링



전체 구조

- 빌딩 블록
  - CNN은 빌딩 블록을 이어 붙여 **깊은 구조로 확장**
  - [그림 4-18]은 전형적인 빌딩블록 : 컨볼루션층 -> 활성함수 (주로 ReLU 사용) -> 풀링층
  - 다중 커널을 사용하여 다중 특징 맵을 추출

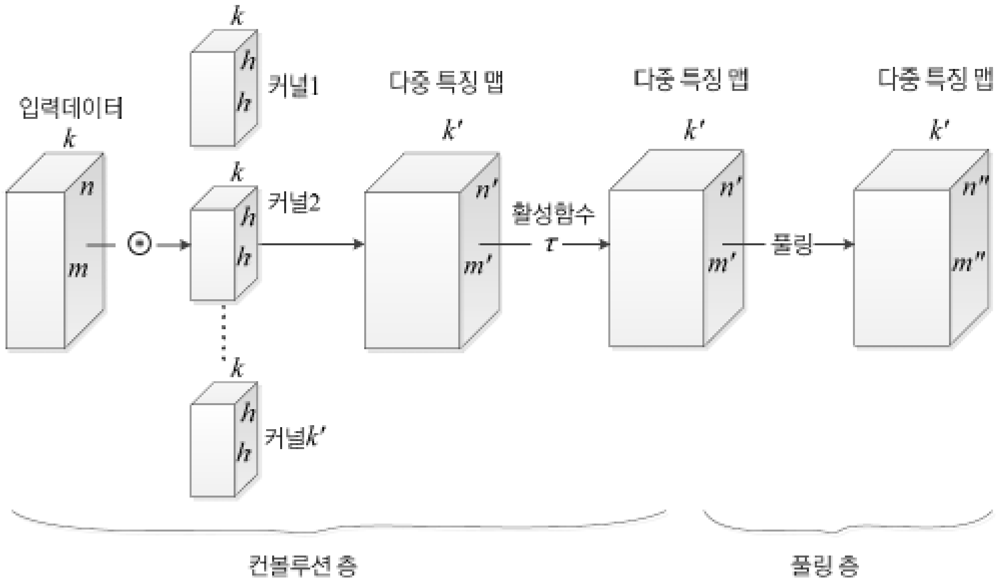


그림 4-18 CNN의 빌딩블록

- 출력의 **크기**와 **매개변수의 수**
  - 입력 : W1 * H1 * D1
  - K개 F * F 커널, 보폭 S, 덧대기 P
  - 출력의 크기

    $W2 * H2 * D2$

    $W2 = (W1 - F + 2P)/S + 1$

    $H2 = (H1 - F + 2P)/S + 1$

    $D2 = K$
  - 매개변수의 수
    - 커널마다 (F*F*D1)개의 가중치와 1개의 바이어스를 가짐
    - 따라서, 전체 매개변수의 수는 $(F * F * D1)K + K$
  - 일반적으로 F = 2, S = 2 혹은 F = 3, S = 2를 사용
- 초창기 CNN 사례 **Lenet-5**
  - **특징 추출 : CONV - POOL - CONV - POOL - CONV**의 다섯 층을 통해 **28*28 명암 영상**을 **120차원의 특징벡터로 변환**
  - **분류** : 은닉층이 하나인 MLP
  - CNN의 첫 번째 성공 사례 : 필기 숫자 인식기 만들어 수표 인식 자동화 시스템 구현

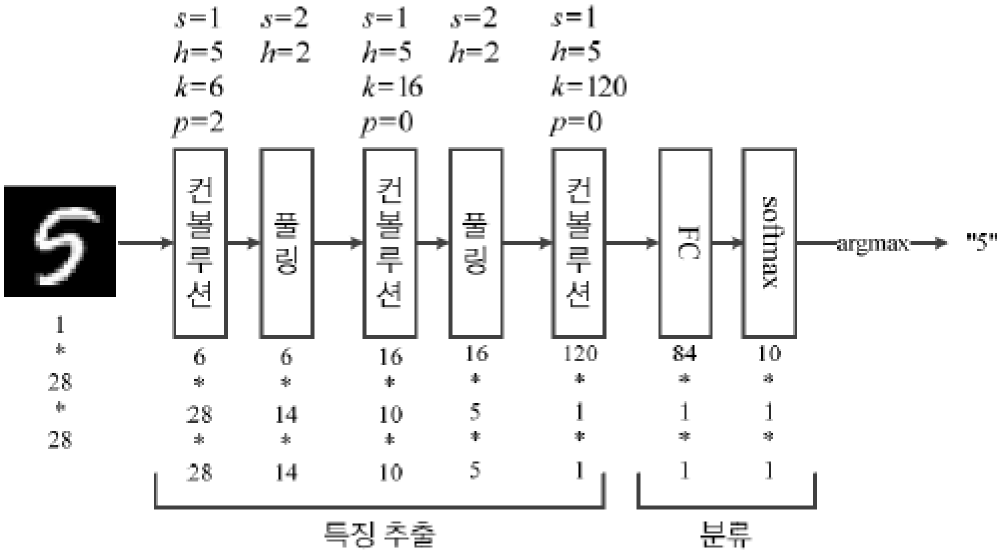


그림 4-19 LeNet-5 구조

- 가변 크기의 데이터 다루기
  - DML는 특징 벡터의 크기가 달라지면 연산 불가능
  - CNN은 가변 크기를 다룰 수 있는 강점
    - 컨볼루션층에서 보폭을 조정한다거나, 풀링층에서 커널이나 보폭을 조정하여 특징 맵 크기를 조절