[4장] 딥러닝 기초

1. 딥러닝의 등장

- 1980년대에 이미 깊은 신경망에 대한 아이디어는 있었지만...
  a) 깊게 만들면 학습이 잘되지 않는 문제 ⇒ 그레이디언트 소멸 (gradient vanishing)
  b) 매개변수는 크게 늘어났지만, 훈련집합의 크기가 작아 ⇒ 과잉적합 (overfitting) 문제
  c) 과도한 계산 시간
- 딥러닝의 기술혁신 요인
  a) CNN이 딥러닝의 가능성을 열었다. 작은 크기의 마스크를 사용하기 때문에 MLP 보다 매개변수가 적다. 모든 노드가 같은 마스크를 공유하는 가중치 공유(weight sharing) 기법 사용
  b) 값싼 GPU 등장을 통한 학습 시간 10~100 배 단축
  c) 인터넷 덕분에 학습 데이터가 크게 늘어남. 또한 데이터에 가우시안 노이즈를 섞거나 영상을 조금 이동 및 회전하여 데이터를 인위적으로 수십~백 배 증가시키는 기법 (데이터 확대 기법)
  d) 계산은 단순하고, 성능 좋은 활성함수 개발 (ReLU), 이를 통해 그레이디언 소멸 문제 완화
  e) 학습에 효과적인 다양한 규제(regularization), 예를 들어, 가중치 축소기법, 일정 비율의 노드를 선택하여 불능으로 만들어 학습하는 드롭아웃(dropout)기법, 조기 멈춤, 데이터 확대, 앙상블 개발
  f) 층별 예비학습 (layerwise pre-training) 기법 개발, 그러나 지금은 잘 사용하지 않음.
- 특징 학습의 부각
  a) 과거 특징 추출은 수작업으로 : 기계학습과 별개로 구현한 후에 결합하여 사용하는 방식
  b) 현대 기계 학습은 통째 학습 (end-to-end learning) : 전체 과정을 한꺼번에 학습
  c) 은닉층은 특징 추출기: 원래 특징공간을 다른 특징공간으로 변환

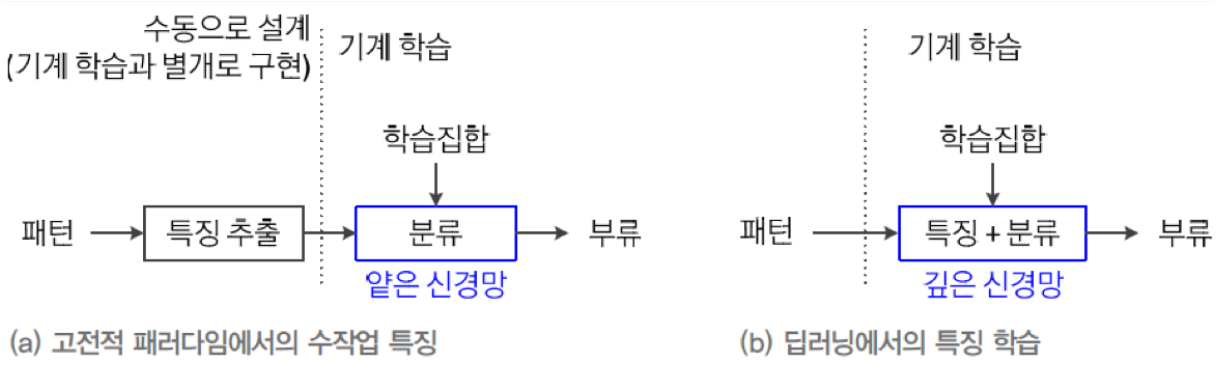


그림 4-1 기계 학습의 패러다임 변화

d) 앞 단계의 은닉층은 주로 에지나 코너와 같은 저급 특징을 (low-level feature) 추출하고, 뒤로 갈수록 더 추상적인 고급 특징을 (high-level feature) 추출

e) 특징을 학습하는 일이 중요해지면서 특징학습 (feature learning), 표현 학습(representation learning)이라는 용어를 많이 사용

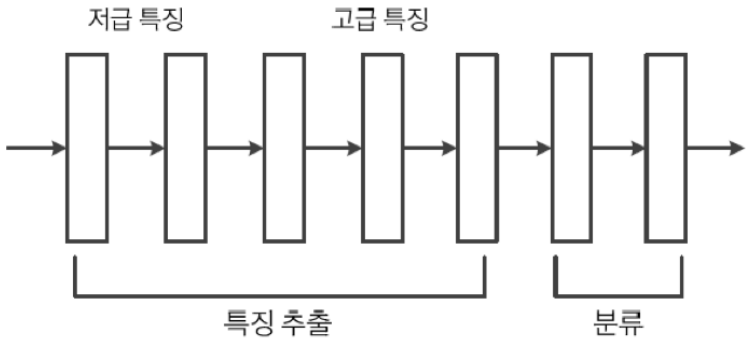


그림 4-2 깊은 신경망의 처리 절차

2. 깊은 다층 퍼셉트론 (Deep MLP)

- 구조와 동작

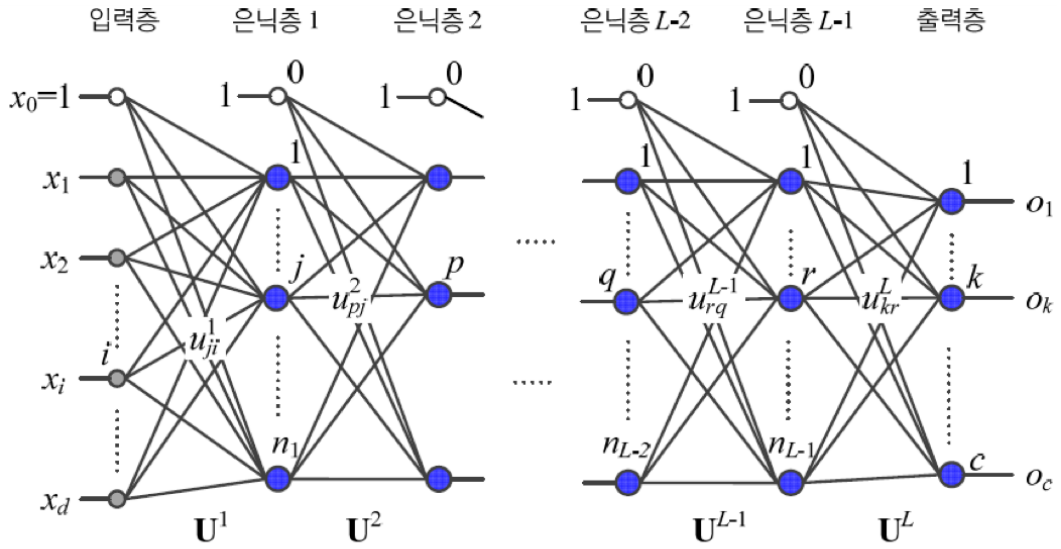


그림 4-3 깊은 MLP(DMLP)의 구조

a) L개의 층을 가진 DMLP의 경우, 0번째 입력층, $1 \sim L-1$ 은닉층, L번째 출력층으로 구성

b) l번째 층의 노드 수를 n_l로 표기. 0번째 층은 입력층의 노드 수는 $n_0 = d$로 표기, L번째 출력층의 노드 수는 $n_L = c$로 표기

c) $l-1$번째 층과 l번째 층을 연결하는 가중치는 $(n_{l-1}+1) \times n_l$ 개 존재하며, 이를 행렬로 표기하면

$$U^l = \begin{bmatrix} u_{10}^l & u_{11}^l & \cdots & u_{1n_{l-1}}^l \\ u_{20}^l & u_{21}^l & \cdots & u_{2n_{l-1}}^l \\ \vdots & & & \\ u_{n_l0}^l & u_{n_l1}^l & \cdots & u_{n_ln_{l-1}}^l \end{bmatrix} \quad \text{for } l = 1, 2, \cdots, L$$

d) DMLP의 동작을 함수로 간결하게 표기:

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

e) 표기의 간략화를 위해 입력 노드와 출력 노드를 다음과 같이 표기

$$\mathbf{z}^0 = \mathbf{x} \quad \rightarrow \quad \begin{bmatrix} z_0^0 \\ z_1^0 \\ z_2^0 \\ \vdots \\ z_{n_0}^0 \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \quad \text{and} \quad \mathbf{z}^L = \mathbf{o} \quad \rightarrow \quad \begin{bmatrix} z_1^L \\ z_2^L \\ \vdots \\ z_{n_L}^L \end{bmatrix} = \begin{bmatrix} o_1 \\ o_2 \\ \vdots \\ o_c \end{bmatrix}$$

f) l번째 층의 연산 :

$$\mathbf{z}^l = \tau_l(U^l \mathbf{z}_{l-1}) \qquad 1 \le l \le L$$

은닉층에 해당하는 $l = 1, 2, \cdots, L-1$번째 활성함수 $\tau_1 \sim \tau_{L-1}$는 주로 ReLU를, 출력층의 활성함수 τ_L은 로지스틱시그모이드, tanh, softmax 를 주로 사용

g) l번째 층의 j번째 노드의 연산 :

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

여기서

$$\mathbf{u}_j^l = \begin{bmatrix} u_{j0}^l & u_{j1}^l & u_{j2}^l & \cdots & z_{jn_{l-1}}^l \end{bmatrix} \qquad \mathbf{z}^{l-1} = \begin{bmatrix} 1 \\ z_1^{l-1} \\ z_2^{l-1} \\ \vdots \\ z_{n_{l-1}}^{l-1} \end{bmatrix}$$

- 학습

a) L번째 출력층의 k번째 노드와 연결된 그레이디언트 계산 : ΔU^L

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

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

b) $L-1$번째 층으로 부터 1번째 층까지의 그레이디언트 계산 : $\Delta U^l \quad l = L-1, L-2, \cdots, 1$

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

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

c) 학습 알고리즘의 주요 개선

- 은닉층에 시그모이드 함수 대신 ReLU를 도입
- 평균제곱오차라는 목적함수를 교차엔트로피 또는 로그우도로 대치

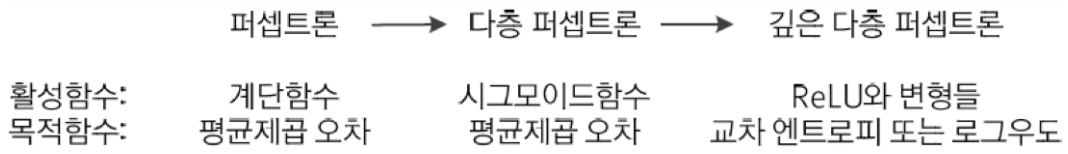


그림 4-4 다층 퍼셉트론의 역사적 발전 양상

d) 영상 인식의 경우, 적은 매개변수를 사용하는 CNN의 도입으로 학습이 보다 수월해졌다.

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)

3. 콘볼루션 신경망 (CNN: convolutional neural network)

- DMLP와 CNN 비교

a) DMLP

- 완전 연결 구조 (FC : fully connected)
- 가중치가 많아 학습이 더디고, 과잉적합 가능성

b) CNN

- 부분 연결 구조, 수용장(receptive field) 인간 시각과 유사
- 영상과 같은 2차원, 3차원 구조의 콘볼루션 연산에서 좋은 특징을 추출
- 격자 구조 내의 한 요소와 이웃 요소 사이에 관련성이 깊은 데이터에 적합
- 콘볼루션 연산을 적용하여 얻은 특징 맵을 다운샘플링(down-sampling)하기 위해 풀링(pooling) 연산을 이어서 수행

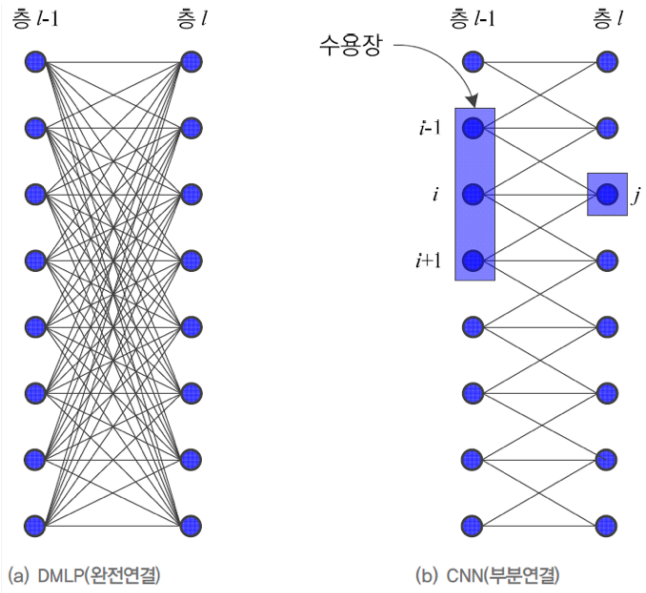
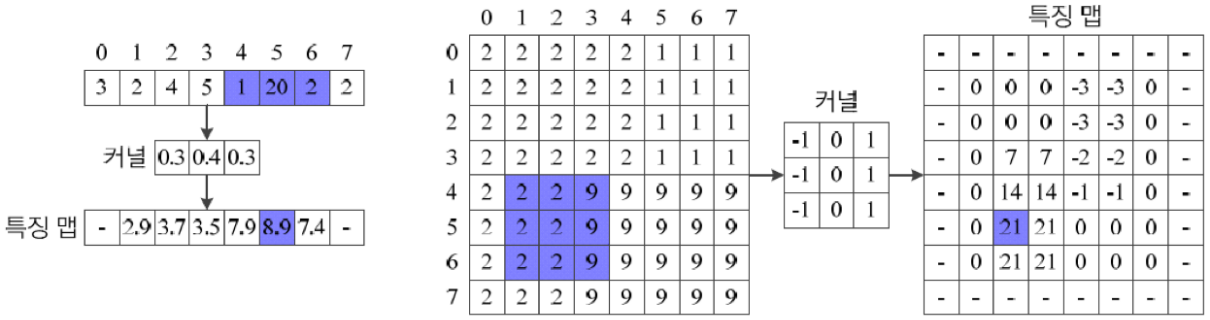


그림 4-5 CNN의 부분연결과 수용장

- 콘볼루션층

  a) 콘볼루션 연산



(a) 1차원 컨볼루션 (b) 2차원 컨볼루션

그림 4-6 컨볼루션 연산

- 노드들을 네모 격자로 표시
- 콘볼루션 연산을 수행할 마스크(mask)를 커널(kernel), 필터(filter), 윈도우(window)라고 하며, 해당하는 요소끼리 곱하고 결과를 모두 더하는 선형 연산이다. 예를 들어, 1차원의 경우, 수용장 1, 20, 2에 대해서 커널 0.3, 0.4, 0.3을 적용하여 특징맵의 해당 위치에

$$1 \times 0.3 + 20 \times 0.4 + 2 \times 0.3 = 8.9$$

2차원의 경우,

$$2 \times -1 + 2 \times 0 + 9 \times 1 + 2 \times -1 + 2 \times 0 + 9 \times 1 + 2 \times -1 + 2 \times 0 + 9 \times 1 = 21$$

- 커널의 크기 h는 보통 홀수로 설정, u는 커널, z는 입력, s는 출력 특징맵(feature map)

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

- 3차원 영상(RGB)에는 3차원 콘볼루션을 적용
- 콘볼루션 연산은 양쪽 가장자리에서 총 h-1개의 노드가 줄어든다. 이를 방지하기 위하여 덧대기(padding)을 수행
  i. 가장자리에 필요한 개수만큼 미리 0을 덧대기
  ii. 가장자리에 인접한 노드 값을 복사하는 덧대기

그림 4-7 덧대기(회색 노드가 덧댄 노드)

- CNN도 DMLP처럼 바이어스(bias)가 있다. 연산 결과에 바이어스 값만큼 더하는 효과

그림 4-8 바이어스

$$(0.2) + 1 \times 0.3 + 20 \times 0.4 + 2 \times 0.3 = 9.1$$

$$(2) + 2 \times -1 + 2 \times 0 + 9 \times 1 + 2 \times -1 + 2 \times 0 + 9 \times 1 + 2 \times -1 + 2 \times 0 + 9 \times 1 = 23$$

b) 가중치 공유와 다중 특징 맵 추출

- 입력 값에 같은 커널을 적용하므로 가중치를 공유 (weight sharing) 한다고도 하고, 같은 가중치를 여러곳에 묶어둔 것이라는 의미에서 묶인 가중치 (tied weight)라고도 함.
- 8 입력: 8 출력 구조의 경우, DMLP를 적용하면 여기에 총 64개의 가중치 매개변수가 필요하지만, CNN을 3개짜리 커널을 적용하면, 3개의 가중치 매개변수를 학습하면 됨. (모델의 복잡도 크게 낮아짐)
- 커널의 특징

  i. (0.3, 0.4, 0.3)의 커널은 노드 3개의 평균을 구하는 셈

  ii. (-0.5, 0.0, 0.5)의 커널은 변화가 있는 곳에서 반응이 큼. 변화가 없으면 0

$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$: 수직 에지 $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$: 수평 에지 $\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$ $\begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$: 대각선 에지

- 다양한 특징을 추출하기 위하여 실제로는 수십~수백 개의 커널을 사용

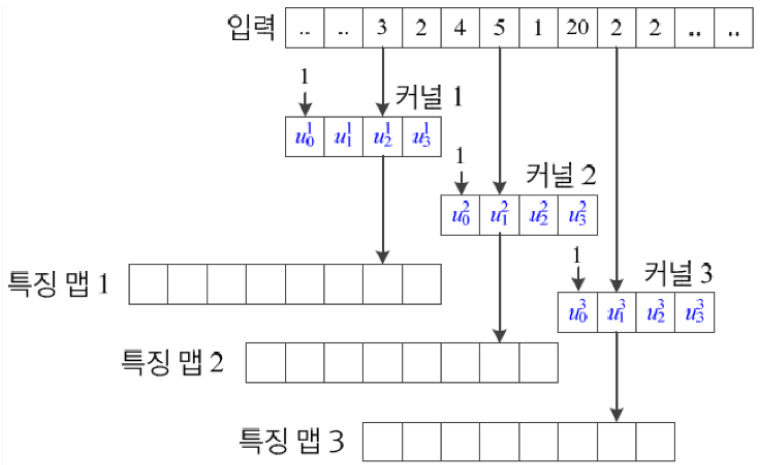


그림 4-10 다중 특징 맵 추출

- k번째 커널의 i번째 가중치 요소는 u_i^k로 표기
- DMLP와 마찬가지로 오류 역전파 알고리즘을 적용하여 커널을 학습으로 찾아냄.
- 커널은 어떤 특징 맵을 추출할지 규정하므로, 이를 이용한 학습과정을 특징 학습 (feature learning) 혹은 표현 학습(representation learning)이라고 부름

c) 콘볼루션 연산에 따른 CNN의 특징

- 이동에 동변(translation equivariant, 신호가 이동하면 이동 정보가 그대로 특징 맵에 반영) → 영상 인식에서 물체 이동이나 음성 인식에서 발음 지연에 효과적으로 대처
- 각 노드는 독립적으로 계산 가능하므로 병렬 구조
- 노드는 깊은 층을 거치면서 전체에 영향을 미치므로 분산 구조

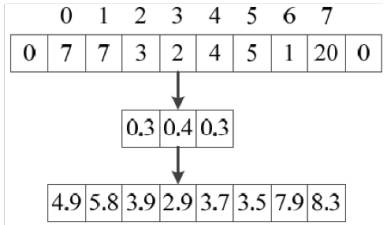


그림 4-11 CNN의 이동에 동변한 특성

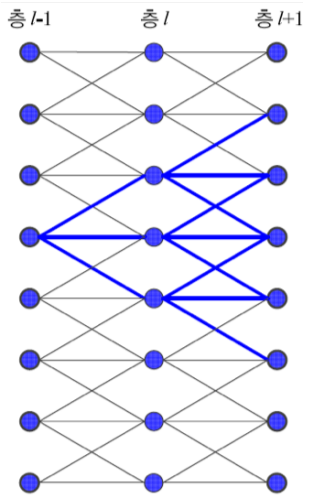
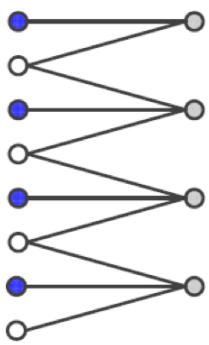


그림 4-12 CNN의 병렬 분산 구조


(a) 1차원 데이터(예: 음성)

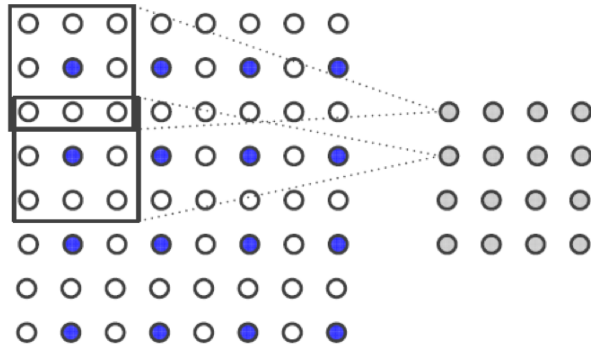

(b) 2차원 데이터(예: 영상)

그림 4-13 보폭이 2인 컨볼루션 연산

d) 큰 보폭에 의한 다운샘플링

- 데이터의 가장자리에 적절히 덧대기를 한다면 입력 데이터와 출력 데이터의 크기는 같다. 이러한 방식을 보폭(stride)이 1이라고 한다.
- 보폭을 2로 설정하면, 노드 2개 마다 하나씩 샘플링하여 콘볼루션 적용

- 보폭을 k로 설정하면, 노드 k개 마다 하나씩 샘플링하여 커널을 적용. 결과적으로 입력 데이터 보다 $1/k$배로 축소된 출력 데이터 얻음.
- 2차원 영상 $m \times n$에 대해 보폭 k를 적용하면, $\frac{m}{k} \times \frac{n}{k}$로 축소된 영상 획득. ($1/k^2$배로 축소된 출력 데이터)

e) 텐서에 적용

- RGB 3차원 이미지는 $3 \times m \times n$를 가지므로, 이에 대한 커널 또한 $3 \times h \times h$로 설정되어 3차원 이미지에 적절한 덧대기 후 적용 가능

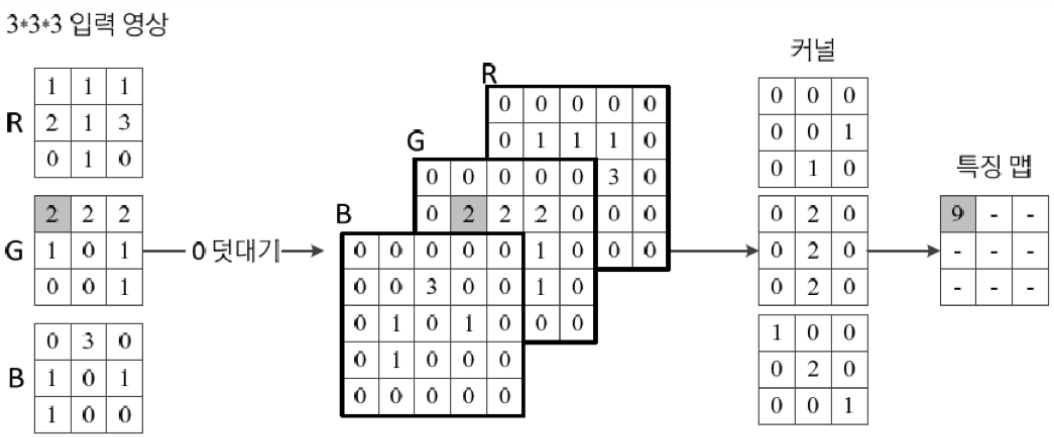
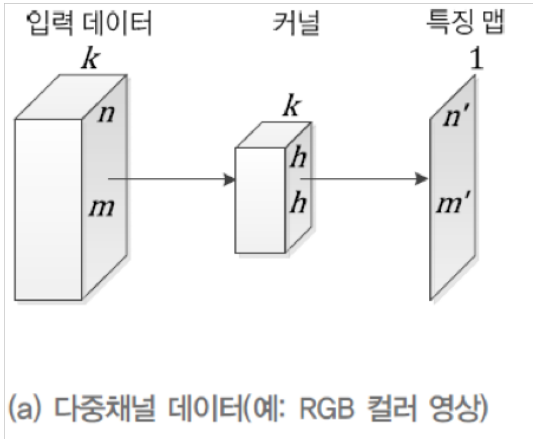


그림 4-14 텐서의 컨볼루션 연산(0 덧대기 적용)



(a) 다중채널 데이터(예: RGB 컬러 영상)

그림 4-15 텐서의 컨볼루션 연산(직육면체로

- 일반화하여 3차원 구조의 데이터를 $k \times m \times n$의 직육면체로 표현하고, 이에 상응하는 커널은 $k \times h \times h$의 크기를 가지면, 특징 맵은 $1 \times m' \times n'$의 크기를 가진다. 보폭이 1이면, $m' = m$이고, 2이면 $m' = \frac{m}{2}$이다.
- 여러개의 커널을 사용하면 다중 특징 맵을 추출할 수 있다. [그림 4-18 참조]

- 풀링층 (pooling layer)

a) CNN에서는 컨볼루션 연산 결과에 보통 ReLU 활성함수를 적용한 다음, 그 결과에 풀링연산을 적용

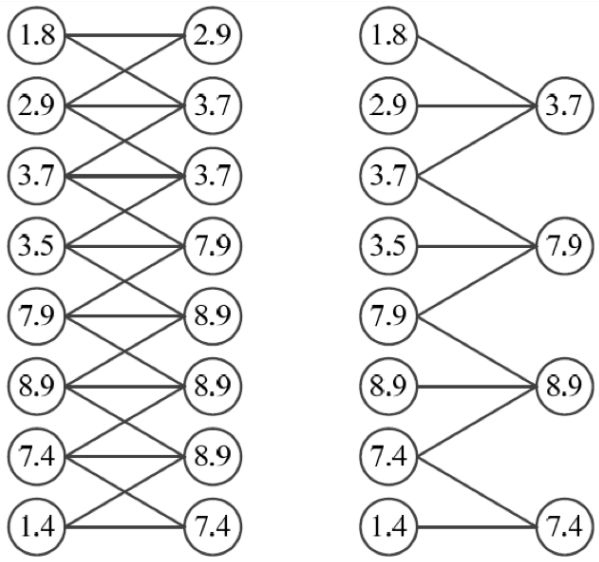


그림 4-16 최대 풀링

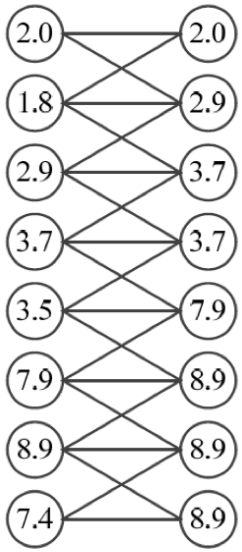


그림 4-17 작은 이동에 둔감한 최대 풀링

b) 최대 풀링, 평균 풀링, 가중치 평균 풀링 등 존재

c) 보폭을 크게 하면 다운샘플링 효과, 보폭을 2로 하면 특징 맵을 1/2로 줄일 수 있다.

d) 지나치게 상세한 정보를 포함한 특징 맵에서 요약 통계를 추출함으로써 성능 향상에 기여할 수 있음

e) 학습으로 부터 알아내야할 매개변수가 없음.

f) 풀링은 작은 변화에 둔감

- 전체 구조

a) 빌딩 블록

- CNN은 빌딩블록을 이어 붙여 깊은 구조를 만듦
- 전형적인 빌딩블록: 컨볼루션층 ⇒ 활성함수(주로 ReLU 사용) ⇒ 풀링층
- 다중 커널을 사용하여 다중 특징 맵을 추출 : 커널 k'개를 사용하면 $k' \times m' \times n'$ 크기의 특징 맵을 출력, m'은 보폭에 따라 결정

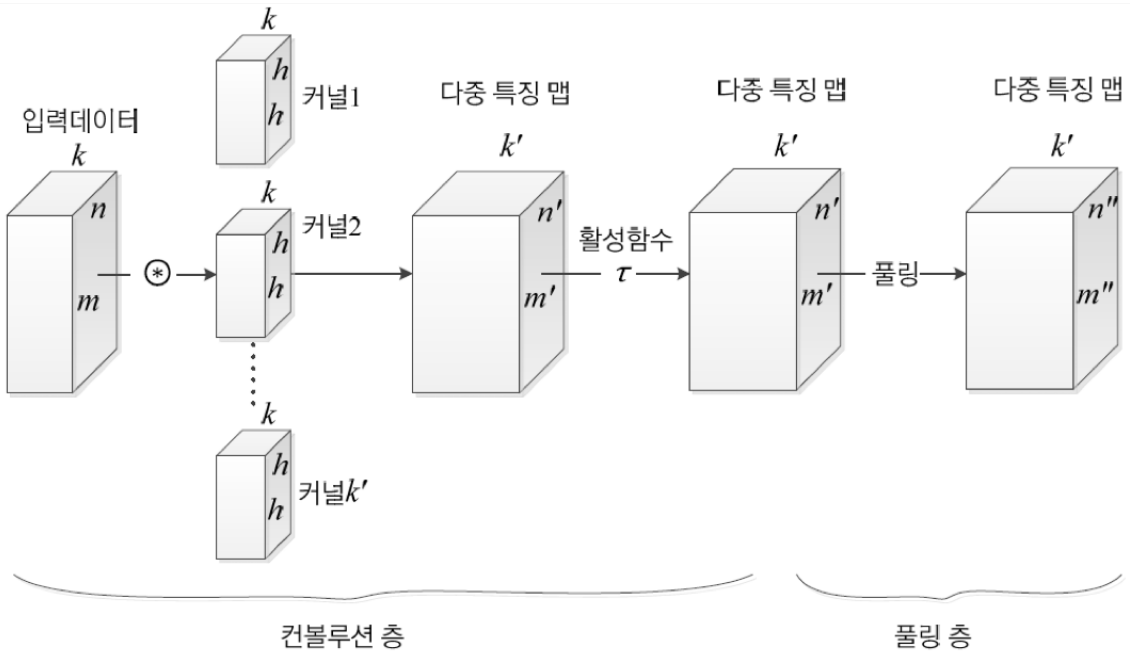


그림 4-18 CNN의 빌딩블록

b) LeNet-5 (CNN의 첫 번째 성공사례)

- 1998년 LeCun이 제안한 CNN 구조: 필기 숫자 인식을 위한 수표 인식 자동화 시스템
- 특징 추출: C-P-C-P-C의 다섯 층을 통해 28×28 명암 맵을 120차원의 특징 벡터로 변환
- 분류: 은닉층이 하나인 MLP

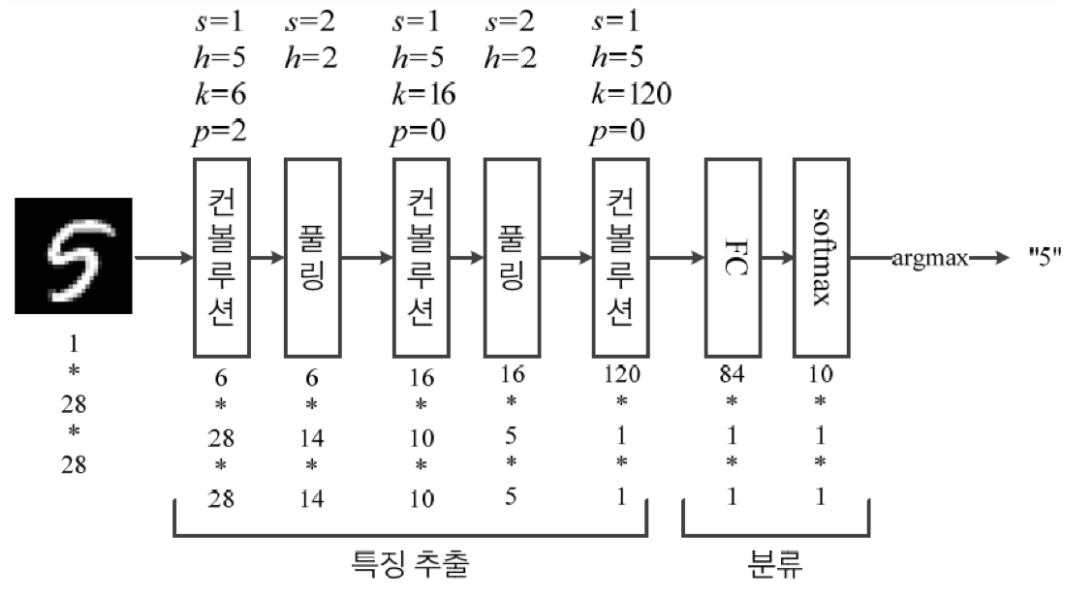


그림 4-19 LeNet-5 구조

- 첫번째 콘볼루션 층은 보폭 s=1, 커널크기 h=5, 커널개수 k=6, 덧대기 p=2로 설정, 출력 특징 맵은 $6 \times 28 \times 28$
- 이어지는 풀링층에서는 보폭 s=2를 사용하여 출력은 $6 \times 14 \times 14$
- 두번째 콘볼루션 층은 보폭 s=1, 커널크기 h=5, 커널개수 k=16, 덧대기 p=0로 설정, 출력 특징 맵은 $16 \times 10 \times 10$
- 이어지는 풀링층에서는 보폭 s=2를 사용하여 출력은 $16 \times 5 \times 5$
- 세번째 콘볼루션 층은 보폭 s=1, 커널크기 h=5, 커널개수 k=120, 덧대기 p=0로 설정, 출력 특징 맵은 $120 \times 1 \times 1$
- 마지막에 분류 기능을 추가하기 위하여 FC MLP 입력층(120개)-은닉층(84개)-출력층(10개) 를 인가하고, 출력 활성함수로서 softmax를 적용하고, argmax를 적용하여 분류

c) 가변 크기의 데이터 다루기

- DMLP는 특징 벡터의 크기가 달라지면 연산 불가능

- CNN은 가변 크기를 다룰 수 있는 강점. 입력의 크기가 변경되어도 콘볼루션 연산에는 영향을 주지 않음
- 컨볼루션층에서 보폭을 조정한다거나, 풀링층에서 커널이나 보폭을 조정하여 특징 맵 크기를 조절 가능

4. 콘볼루션 신경망 (CNN) 사례연구

a) ImageNet 데이터베이스 2만 부류에 대해 부류별로 500~1000장의 영상을 인터넷에서 수집하여 구축하고 공개

b) ILSVRC 대회 (CVPR 학술대회에서 개최)

- 1000부류에 대해 분류(classification), 검출(detection), 위치 지정 문제(localization): 1순위와 5순위 오류율로 대결
- 120만 장의 훈련집합, 5만 장의 검증집합, 15만 장의 테스트집합
- 우승: AlexNet(2012), Clarifi팀(2013), GoogLeNet&VGGNet(2014), ResNet(2015)

c) 우승한 CNN은 프로그램과 가중치를 공개함으로써 널리 사용되는 표준 신경망이 됨

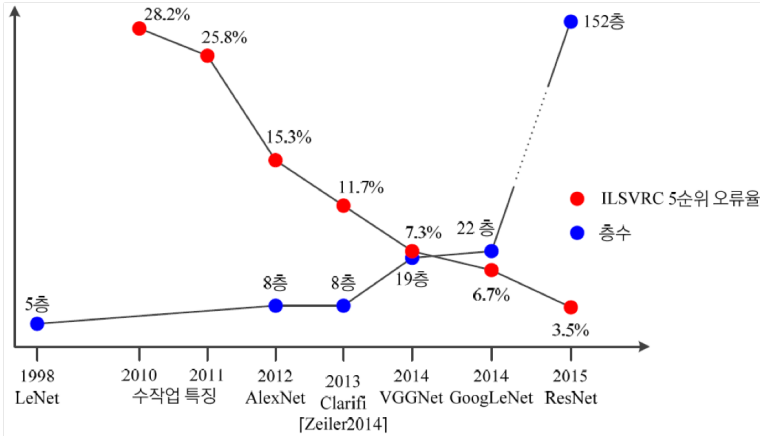


그림 4-30 CNN의 발전 추세

그림 4-31 ILSVRC 물체 검출 문제

- AlexNet (2012, 토론토대학, Alex Krizhevsky)

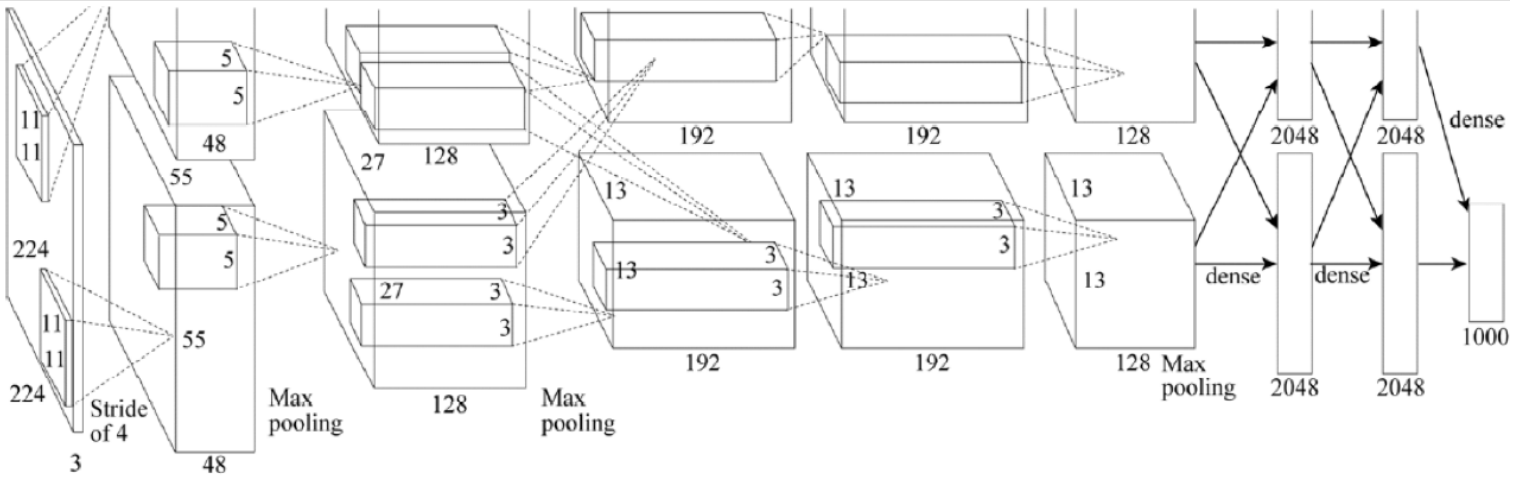


그림 4-21 AlexNet 구조[Krizhevsky2012]

a) 컨볼루션층 5개와 완전연결(FC) MLP 층 3개

b) 입력 $3 \times 224 \times 224$에 첫번째 콘볼루션층은 96개의 $3 \times 11 \times 11$ 커널을 보폭 4로 적용, 따라서 $96 \times 55 \times 55$ 특징 맵 얻음.

c) 컨볼루션층은 200만개, FC층은 6500만개 가량의 매개변수

d) FC층에 30배 많은 매개변수 → 향후 CNN은 FC층의 매개변수를 줄이는 방향으로 발전

e) AlexNet 성공요인
   - ImageNet이라는 대용량 데이터베이스
   - GPU를 사용한 병렬처리
   - 활성함수로 ReLU 사용 (계산시간 축소)
   - 커널의 컨볼루션 결과를 이웃 커널의 값을 고려하여 조정하는 지역 반응 정규화 기법 적용
   - 데이터 확대(잘라내기와 반전으로 2048배로 확대) (과잉적합 회피)
   - 드롭아웃 (과잉적합 회피)

- VGGNet (2014, Oxford) [VGG-16(컨볼루션 13층+FC 3층)]

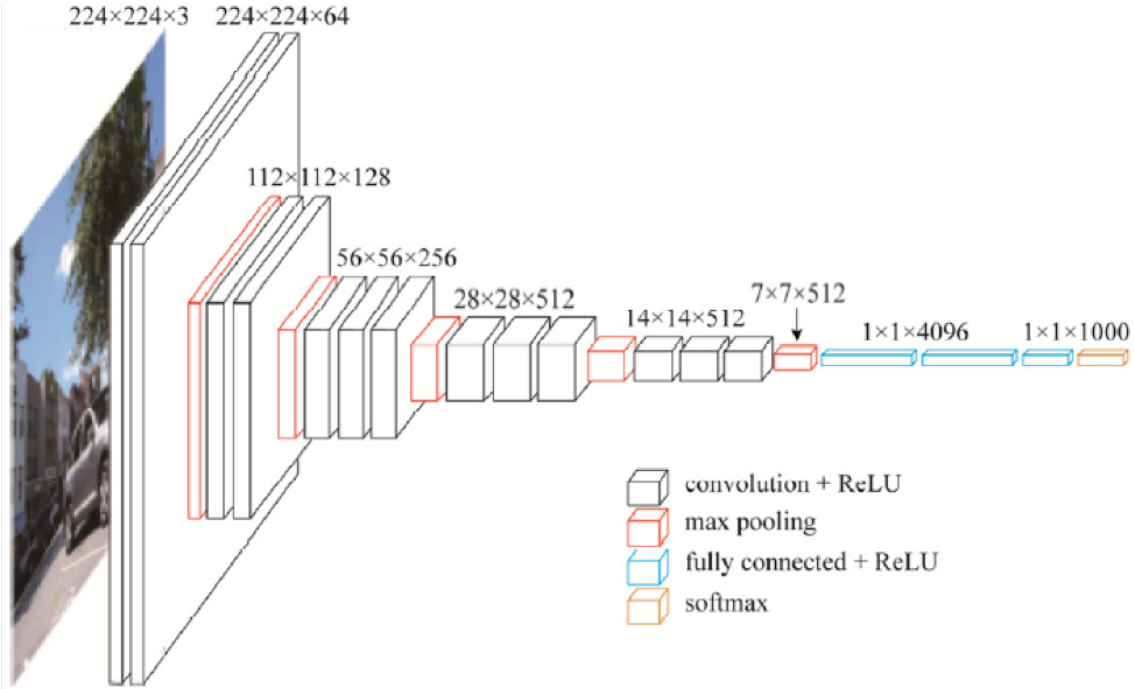


그림 4-22 VGGNet 구조[Simonyan2015]

a) 작은 커널을 사용하여 신경망을 더욱 깊게 만듦

b) 입력 $3 \times 224 \times 224$에 2개의 콘볼루션층을 적용한다. 각각 보폭이 1인 3×3 커널을 64개 적용, 따라서 $64 \times 224 \times 224$ 특징 맵 얻음.

c) 이어서 보폭 2로 2×2 커널로 최대 풀링 수행

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

e) 1×1 커널

- 차원 축소 효과
- $m \times n$의 특징 맵 8개에 1×1 커널을 4개 적용 $\rightarrow$ $m \times n$의 특징 맵 4개가 됨
- 다시 말하면, $8 \times m \times n$ 텐서에 $8 \times 1 \times 1$ 커널을 4개 적용하여 $4 \times m \times n$ 텐서를 출력하는 셈
- ReLU와 같은 비선형 활성함수를 적용하면 특징 맵의 분별력 증가
- 네트워크 속의 네트워크(NIN)에서 유래
- GoogLeNet에서 적극적으로 사용

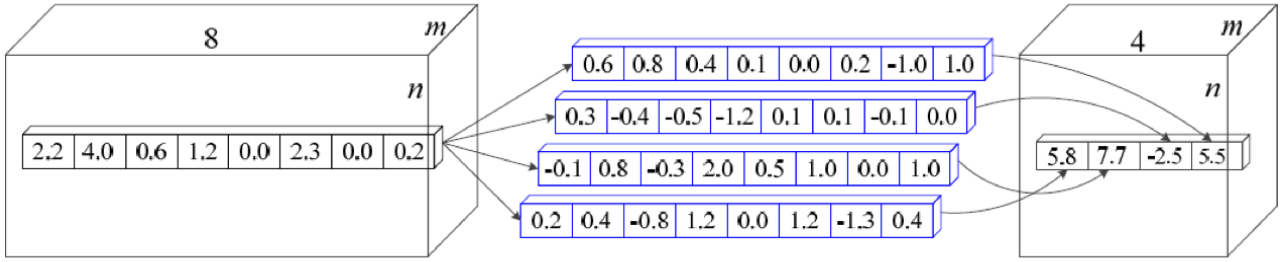


그림 4-23 1*1 컨볼루션 예제

- GoogLeNet (2014, Google)

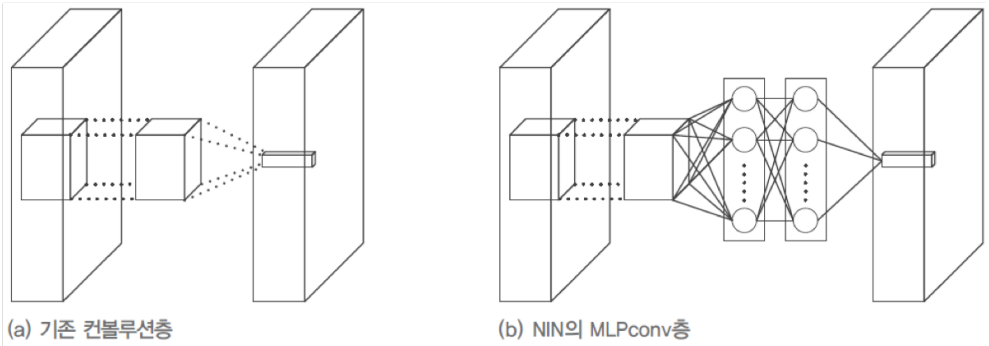


그림 4-24 기존 컨볼루션 신경망과 NIN의 비교

a) NIN 구조
   - MLPconv층이 컨볼루션 연산을 대신함
   - MLPconv는 커널을 옮겨가면서 MLP의 전방 계산을 수행함

b) 전역 평균 풀링: MLPconv가 부류 수만큼 특징 맵을 생성하면, 특징 맵 각각을 평균하여 출력 노드에 입력 (매개변수 축소)

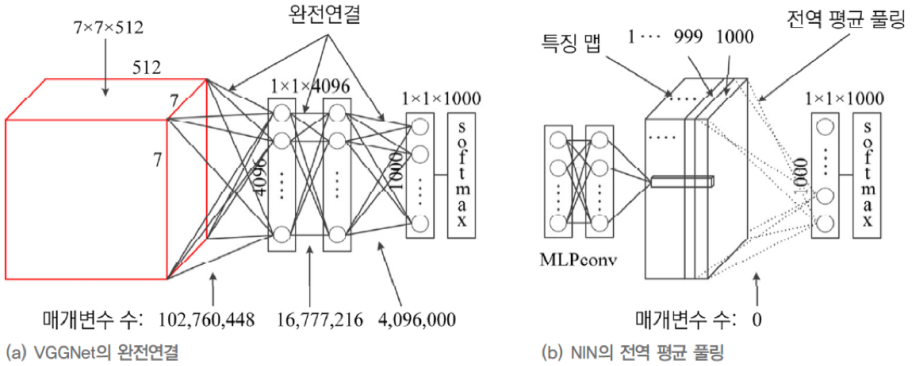


그림 4-25 완전연결과 NIN의 전역 평균 풀링의 비교

c) GoogLeNet의 핵심 아이디어는 NIN을 수정한 인셉션 (inception) 모듈
   - 마이크로 네트워크로 MLPconv 대신 네 종류의 컨볼루션 연산 사용

– 1 × 1 컨볼루션을 사용하여 차원 축소

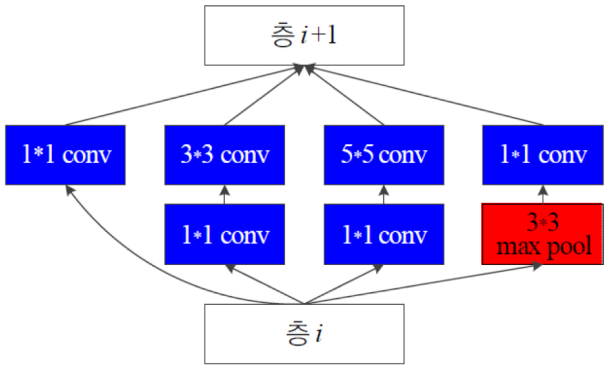


그림 4-26 GoogLeNet의 인셉션 모듈

d) 인셉션 모듈을 9개 결합한 GoogLeNet

– 매개변수가 있는 층 22개, 없는 층 5개로 총 27개 층

– 완전열결층은 1개에 불과 : 1백만 개의 매개변수를 가지며, VGGNet의 완전연결에 비하면 1%

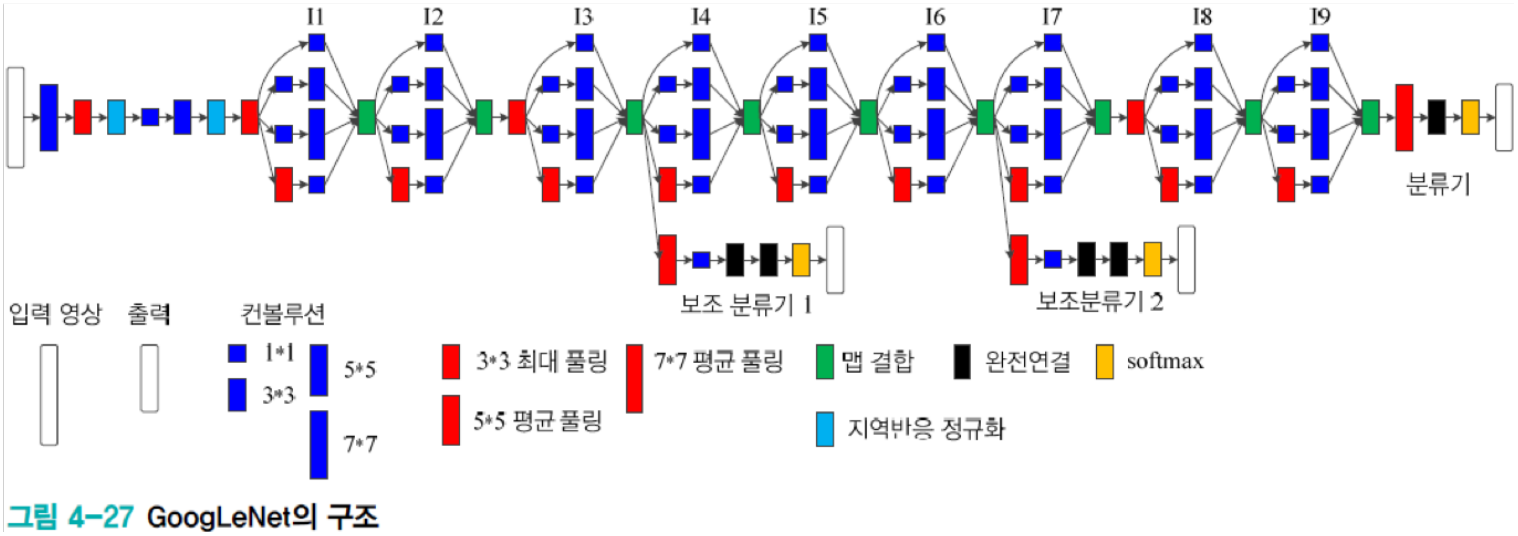


그림 4-27 GoogLeNet의 구조

- ResNet (2015, Microsoft)
  a) 층수가 늘어나면 처음에는 성능하다가 포화상태에 이르고, 어느 지점을 지나면 급격히 저하됨.
  b) ResNet은 잔류 학습(residual learning)이라는 아이디어를 이용하여 성능 저하를 피하면서 층 수를 대폭 늘림(최대 1202층까지)

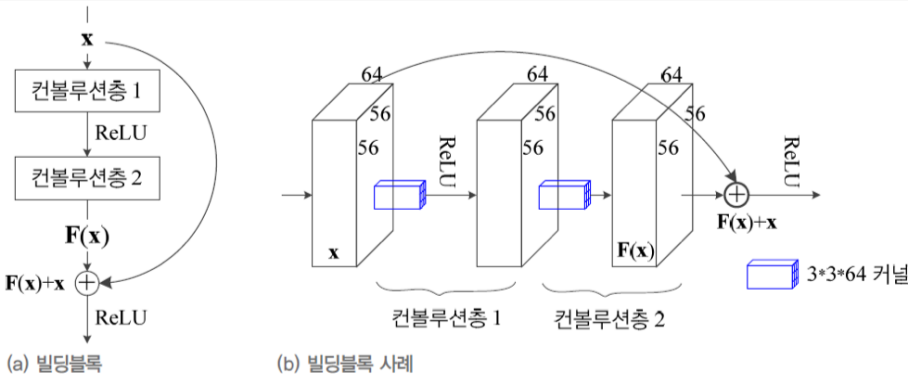


그림 4-28 잔류 학습의 구조와 동작

  c) 원래의 콘볼루션 신경망과 잔류학습 신경망 : 잔류 학습은 지름길 연결된 x를 더한 $F(x)+x$에 τ를 적용. $F(x)$는 잔류

$$F(x) = \tau(x * w_1) * w_2 \qquad F(x) = \tau(x * w_1) * w_2$$
$$y = \tau(F(x)) \qquad y = \tau(F(x) + x)$$

  d) 콘벌루션 층 2개마다 지름길 연결을 두는 이유 : 그레이디언트 소멸 문제 해결
  e) VGGNet과 같은 점 : 3×3 커널 사용
  f) VGGNet과 다른 점
    - 잔류 학습 사용
    - 전역 평균 풀링 사용(FC 층 제거)

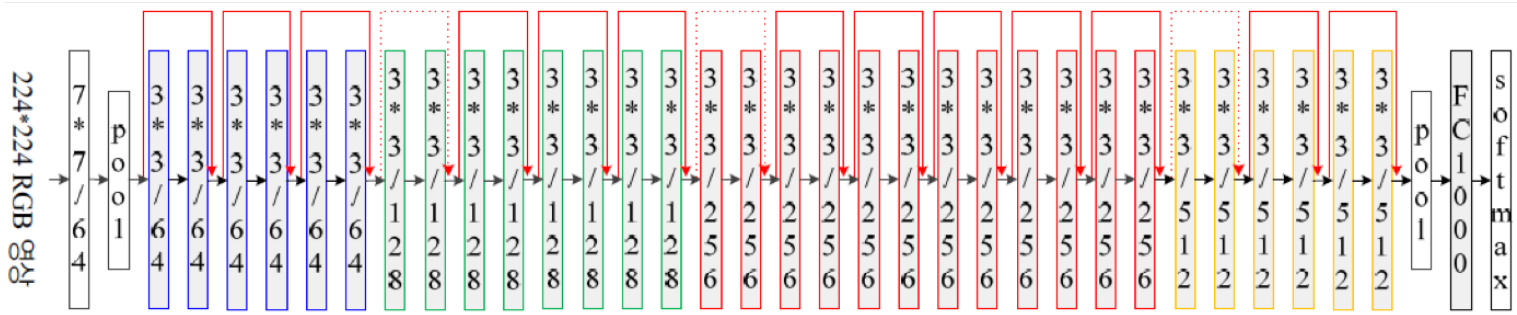


그림 4-29 ResNet 예제(34층)

– 배치 정규화 적용(드롭아웃 적용 불필요) (5.2.6절 참고)

(학습참여 과제 3)

(1) Home Test 풀어서 제출 (upload!) (마감 3월 30일 수요일 24시)

(주의) 학습참여 과제 1 & 2도 마감 또한 3월 30일임.