

5장. 딥러닝 - II

5.2 컨볼루션 신경망

5.2.1 컨볼루션

5.2.2 풀링

5.2.3 컨볼루션 신경망의 구조

5.2.4 컨볼루션 신경망의 학습

5.2.5 대표적인 컨볼루션 신경망 모델

5.2.6 딥러닝 신경망의 전이 학습

컨볼루션 신경망

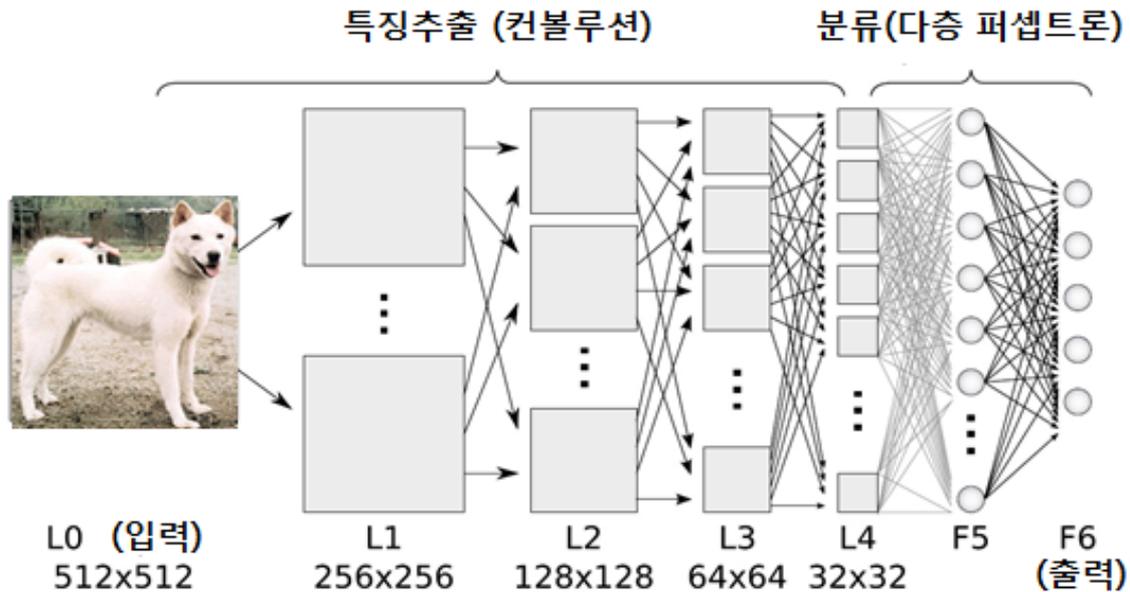
❖ 컨볼루션 신경망(convolutional neural network, CNN)

- 동물의 **시각피질**(visual cortex, 視覺皮質)의 구조에서 영감을 받아 만들어진 딥러닝 신경망 모델
 - **시각피질의 신경세포**
 - 시야 내의 특정 영역에 대한 자극만 수용
 - » 수용장(receptive field, 受容場)
 - 해당 영역의 특정 특징에 대해서만 반응
 - **시각 자극이 1차 시각피질을 통해서 처리된 다음, 2차 시각피질을 경유하여, 3차 시각피질 등 여러 영역을 통과하여 계층적인 정보처리**
 - 정보가 계층적으로 처리되어 가면서 점차 추상적인 특징이 추출되어 시각 인식
- 동물의 계층적 특징 추출과 시각인식 체계를 참조하여 만들어진 모델

컨볼루션 신경망

❖ 컨볼루션 신경망(Convolutional Neural Network, CNN)

- 전반부 : 컨볼루션 연산을 수행하여 **특징 추출**
- 후반부 : 특징을 이용하여 **분류**
- 영상분류, 문자 인식 등 인식문제에 높은 성능



5.2.1 컨볼루션

❖ 컨볼루션(covolution)

- 일정 영역의 값들에 대해 가중치를 적용하여 하나의 값을 만드는 연산

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

입력

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

컨볼루션 필터
커널
마스크

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

컨볼루션 결과

$$\begin{aligned} y_{11} = & w_{11}x_{11} + w_{12}x_{12} + w_{13}x_{13} \\ & + w_{21}x_{21} + w_{22}x_{22} + w_{23}x_{23} \\ & + w_{31}x_{31} + w_{32}x_{32} + w_{33}x_{33} \\ & + w_0 \end{aligned}$$

컨볼루션

❖ 컨볼루션

11	10	10	00	01
00	10	10	10	00
00	00	0	10	10
00	00	10	10	00
01	10	10	00	01

입력

1	0	1
0	1	0
1	0	1

컨볼루션 필터
커널
마스크

4	3	4
2	4	3
2	3	4

컨볼루션 결과

$$\begin{aligned}y_{11} = & w_{11}x_{11} + w_{12}x_{12} + w_{13}x_{13} \\ & + w_{21}x_{21} + w_{22}x_{22} + w_{23}x_{23} \\ & + w_{31}x_{31} + w_{32}x_{32} + w_{33}x_{33} \\ & + w_0\end{aligned}$$

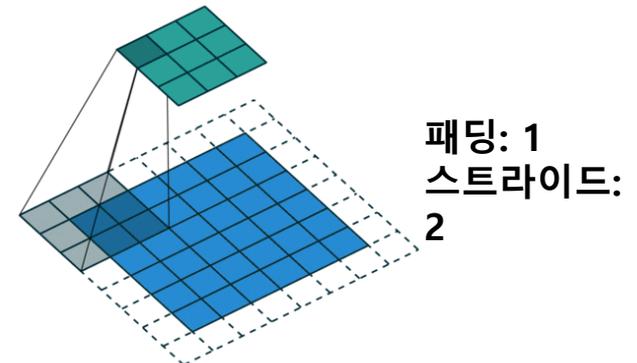
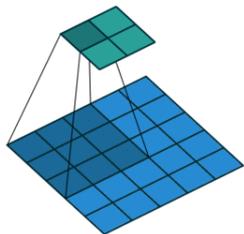
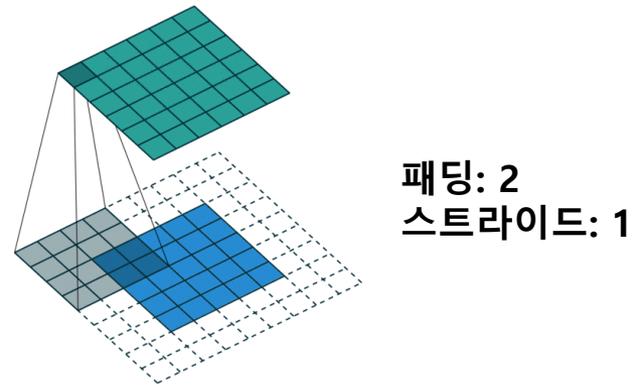
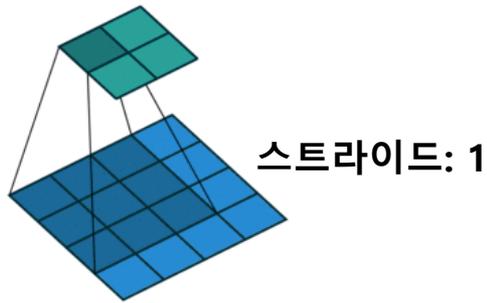
컨볼루션

❖ 스트라이드(stride, 보폭)

- 커널을 다음 컨볼루션 연산을 위해 이동시키는 칸 수

❖ 패딩(padding)

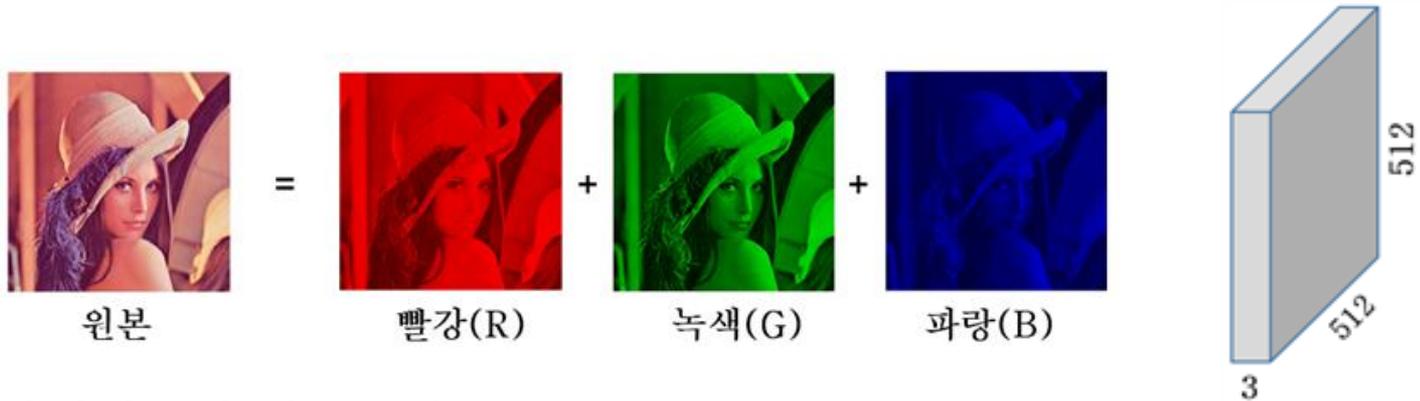
- 컨볼루션 결과의 크기를 조정하기 위해 입력 배열의 둘레를 확장하고 0으로 채우는 연산



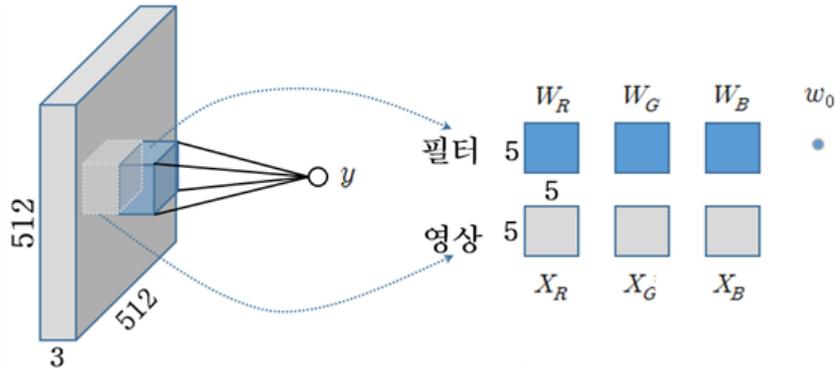
컨볼루션

❖ 컬러 영상의 컨볼루션

- 컬러 영상의 다차원 행렬 표현



- 컬러영상의 컨볼루션

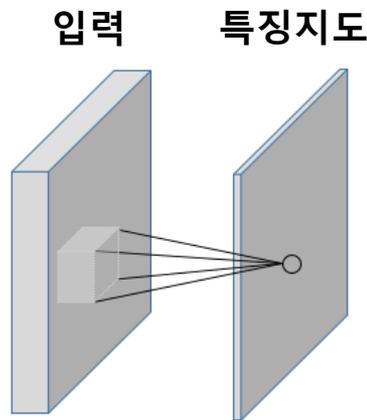


$$y = X_R * W_R + X_G * W_G + X_B * W_B + w_0$$

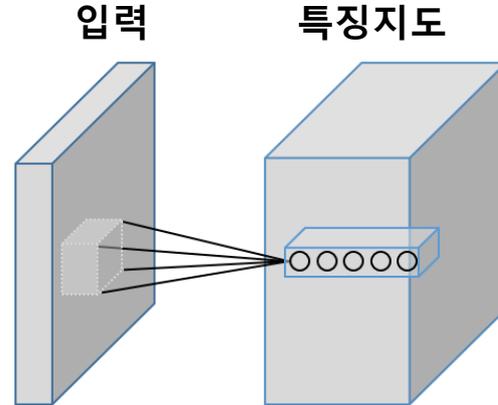
컨볼루션

❖ 특징지도(feature map)

- 컨볼루션 필터의 적용 결과로 만들어지는 2차원 행렬
- 특징지도의 원소값
 - 컨볼루션 필터에 표현된 특징을 대응되는 위치에 포함하고 있는 정도
- k개의 컨볼루션 필터를 적용하면 k의 2차원 특징지도 생성



1개 필터 적용

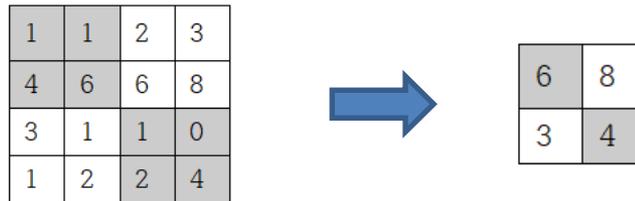


5개 필터 적용

5.2.2 풀링

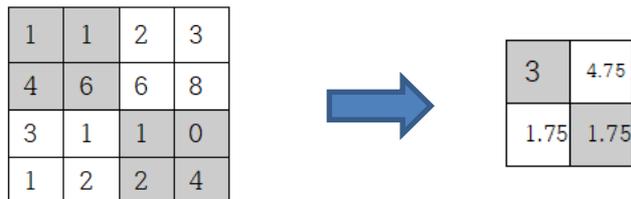
❖ 풀링(pooling)

- 일정 크기의 블록을 통합하여 하나의 대푯값으로 대체하는 연산
- **최대값 풀링(max pooling)**
 - 지정된 블록 내의 원소들 중에서 최대값을 대푯값으로 선택



- **평균값 풀링(average pooling)**

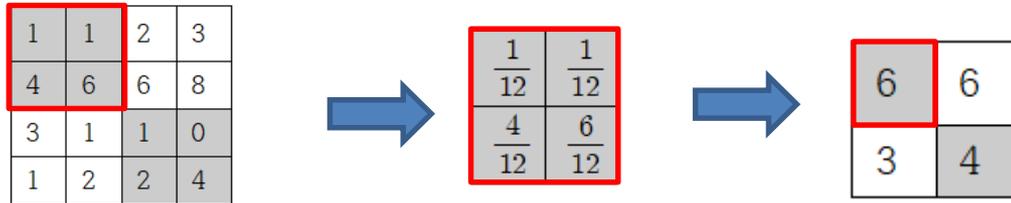
- 블록 내의 원소들의 평균값을 대푯값으로 사용



풀링

- 확률적 풀링(stochastic pooling)

- 블록 내의 각 원소가 원소값의 크기에 비례하는 선택 확률을 갖도록 하고, 이 확률에 따라 원소 하나를 선택



- 학습시: 확률적 풀링

$$p_i = \frac{a_i}{\sum_{k \in R_j} a_k}$$

p_i : 블록 R_j 에서 원소 a_i 가 선택될 확률

- 추론시 : 확률적 가중합 사용

$$s_j = \sum_{i \in R_j} p_i a_i$$

풀링

❖ 풀링 연산의 역할

- 중간 연산 과정에서 만들어지는 특징지도들의 크기 축소
 - 다음 단계에서 사용될 메모리 크기와 계산량 감소
- 일정 영역 내에 나타나는 특징들을 결합하거나, 위치 변화에 강건한 특징 선택

5.2.3 컨볼루션 신경망의 구조

❖ 컨볼루션 신경망의 구조

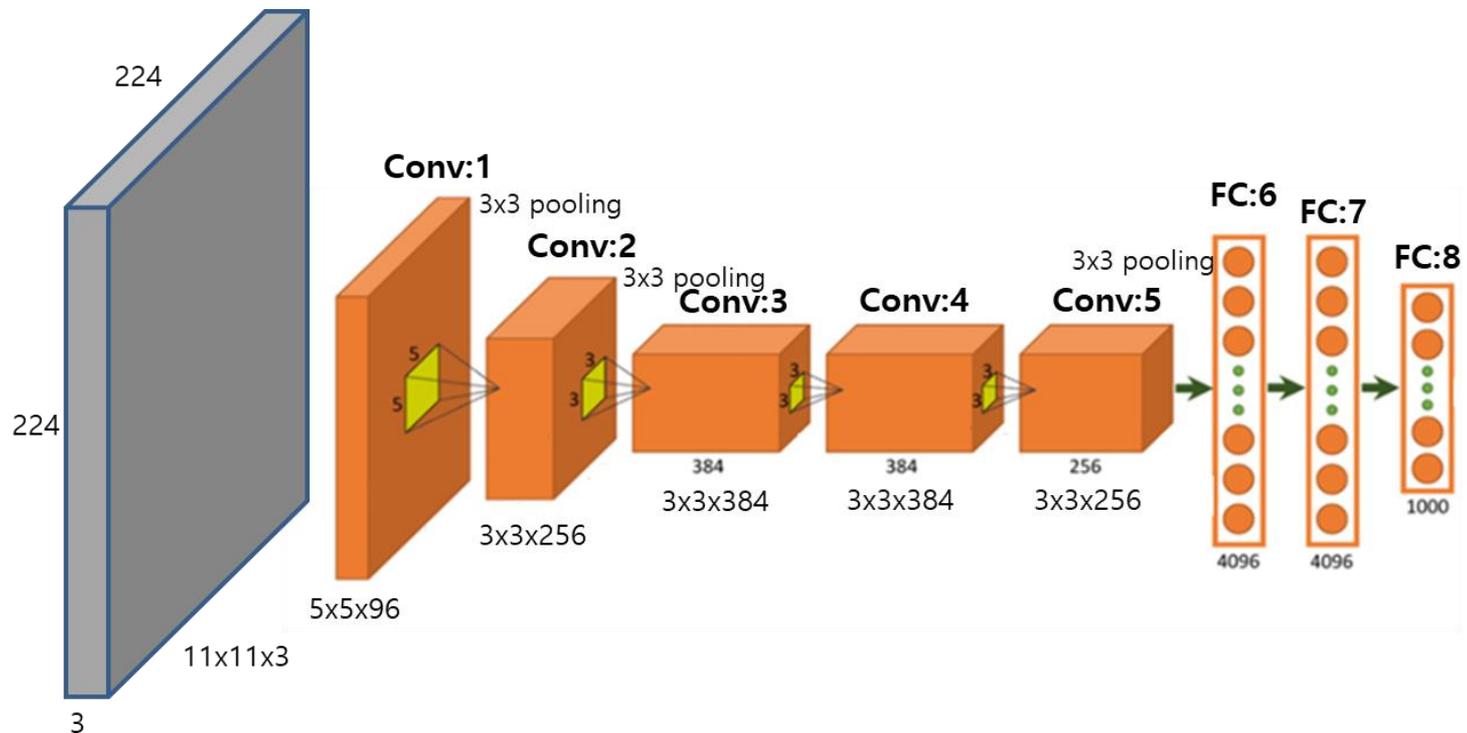
- 특징 추출을 위한 컨볼루션 부분
 - 컨볼루션 연산을 하는 **Conv**층
 - ReLU 연산을 하는 **ReLU**
 - 풀링 연산 **Pool**(선택)]
- 추출된 특징을 사용하여 분류 또는 회귀를 수행하는 다층 퍼셉트론 부분
 - 전방향으로 전체 연결된(fully connected) **FC**층 반복
 - 분류의 경우 마지막 층에 소프트맥스(softmax)을 하는 SM 연산 추가
 - 소프트맥스 연산 : 출력의 값이 0이상이면서 합은 1로 만들
- 컨볼루션 신경망 구조의 예
 - **Conv-ReLU-Pool-Conv-ReLU-Pool-Conv-ReLU-Pool-FC-SM**
 - **Conv-Pool-Conv-Pool-Conv-FC-FC-SM**
 - **Conv-Pool-Conv-Pool-Conv-Conv-Conv-Pool-FC-FC-SM**
 - **Conv-ReLU-Pool-Conv-ReLU-Pool-Conv-ReLU-Pool-FC-FC-SM**

} 반복

컨볼루션 신경망의 구조

❖ 컨볼루션 신경망의 구조 예

- Conv:1-Pool:1-Conv:2-Pool:2-Conv:3-Conv:4-Conv:5-Pool:4-FC:6-FC:7-FC:8



컨볼루션 신경망의 구조

❖ 컨볼루션 신경망의 학습대상 가중치 개수와 메모리 요구량

층	필터/블록 크기	필터 개수	스트라이드	패딩	노드개수 (출력 크기)	학습대상 가중치 개수
입력					224×224×3 (=150,528)	
Conv:1	11×11×3	96	4	3	55×55×96 (=290,400)	(11×11×3+1)×96 (=34,944)
Pool:1	3×3		2		27×27×96 (=69,984)	
Conv:2	5×5×96	256	1	2	27×27×256 (=186,624)	(5×5×96+1)×256 (=614,656)
Pool:2	3×3		2		13×13×256 (=43,264)	
Conv:3	3×3×256	384	1	1	13×13×384 (=64,896)	(3×3×256+1)×384 (=885,120)
Conv:4	3×3×384	384	1	1	13×13×384 (=64,896)	(3×3×384+1)×384 (=1,327,488)
Conv:5	3×3×384	256	1	1	13×13×256 (=43,264)	(3×3×384+1)×256 (=884,992)
Pool:5	3×3	256	2		6×6×256 (=9,216)	
FC:6					4096	6×6×256×4096 (=37,748,736)
FC:7					4096	4096×4096 (=16,777,216)
FC:8					1000	4096×1000 (=4,096,000)

- 가중치
개수 : **58,621,952**
메모리 요구량 :
4바이트 float 사용시
249,476,608 바이트
(≈ 237MB)
- 계산 결과저장
노드 개수: **781,736**
메모리 요구량: ≈ 3MB

5.2.4 컨볼루션 신경망의 학습

❖ 컨볼루션 신경망의 학습을 위한 목적함수

▪ 분류 문제

• 교차 엔트로피(cross entropy)

- 학습 데이터 출력 : t_{ik}
- 컨볼루션 신경망 출력 : $y_k(\mathbf{x}_i, \mathbf{w})$

$$E(\mathbf{w}) = -\log \sum_{i=1}^N \sum_{k=1}^K t_{ik} \log y_k(\mathbf{x}_i, \mathbf{w})$$

▪ 회귀 문제

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^K (t_{ik} - y_k(\mathbf{x}_i, \mathbf{w}))^2$$

❖ 적용 가능 학습 알고리즘

- 경사 하강법
- 경사 하강법의 변형

컨볼루션 신경망의 학습

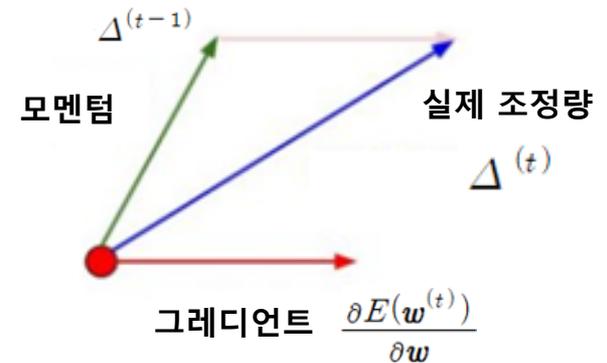
❖ 경사 하강법(Gradient descent method)

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial E(\mathbf{w}^{(t)})}{\partial \mathbf{w}}$$

❖ 모멘텀을 고려한 경사 하강법

$$\Delta^{(t)} = \alpha \Delta^{(t-1)} + \eta \frac{\partial E(\mathbf{w}^{(t)})}{\partial \mathbf{w}}$$

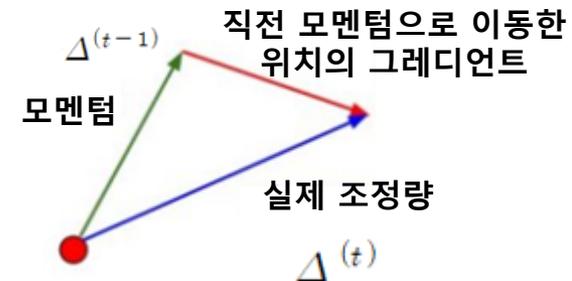
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \Delta^{(t)}$$



❖ NAG(Nesterov accelerated gradient) 방법

$$\Delta^{(t)} = \alpha \Delta^{(t-1)} + \eta \frac{\partial E(\mathbf{w}^{(t)} - \alpha \Delta^{(t-1)})}{\partial \mathbf{w}}$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \Delta^{(t)}$$



컨볼루션 신경망의 학습

❖ AdaGrad 방법

- 가중치별로 별도의 학습을 사용
- 이미 많이 움직였던 가중치에는 작은 학습을 사용

$$g_i^{(t)} = \frac{\partial E(w^{(t)})}{\partial w_i} \quad G_i^{(t)} = G_i^{(t-1)} + (g_i^{(t)})^2$$

$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta}{\sqrt{G_i^{(t)} + \epsilon}} g_i^{(t)}$$

❖ AdaDelta 방법

- Adagrad의 변형
- 과거 그래디언트의 영향을 점점 축소

$$E[g_i^2]_t = \gamma E[g_i^2]_{t-1} + (1 - \gamma)(g_i^{(t)})^2 \quad RMS[g_i]^{(t)} = \sqrt{E[g_i^2] + \epsilon}$$

$$E[w_i^2]_t = \gamma E[w_i^2]_{t-1} + (1 - \gamma) \left(\frac{\eta}{RMS[g_i]^{(t)}} g_i^{(t)} \right)^2 \quad RMS[w_i]^{(t)} = \sqrt{E[w_i^2] + \epsilon}$$

$$w_i^{(t+1)} = w_i^{(t)} - \frac{RMS[w_i]^{(t-1)}}{RMS[g_i]^{(t)}} g_i^{(t)}$$

컨볼루션 신경망의 학습

❖ RMSprop 방법

- 가중치별로 별도의 학습을 사용
- 학습율을 가중치별 누적합의 제곱근으로 나누어서 조정

$$E[g_i^2]_t = \gamma E[g_i^2]_{t-1} + (1 - \gamma)(g_i^{(t)})^2$$

$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta}{\sqrt{E[g_i^2]^{(t)} + \epsilon}} g_i^{(t)}$$

❖ ADAM 방법

- 가중치별로 별도의 학습을 사용
- 그래디언트의 1차 및 2차 모멘텀 사용

$$m^{(t)} = \beta_1 m^{(t-1)} + (1 - \beta_1) g_i^{(t)}$$

$$v^{(t)} = \beta_2 v^{(t-1)} + (1 - \beta_2) (g_i^{(t)})^2$$

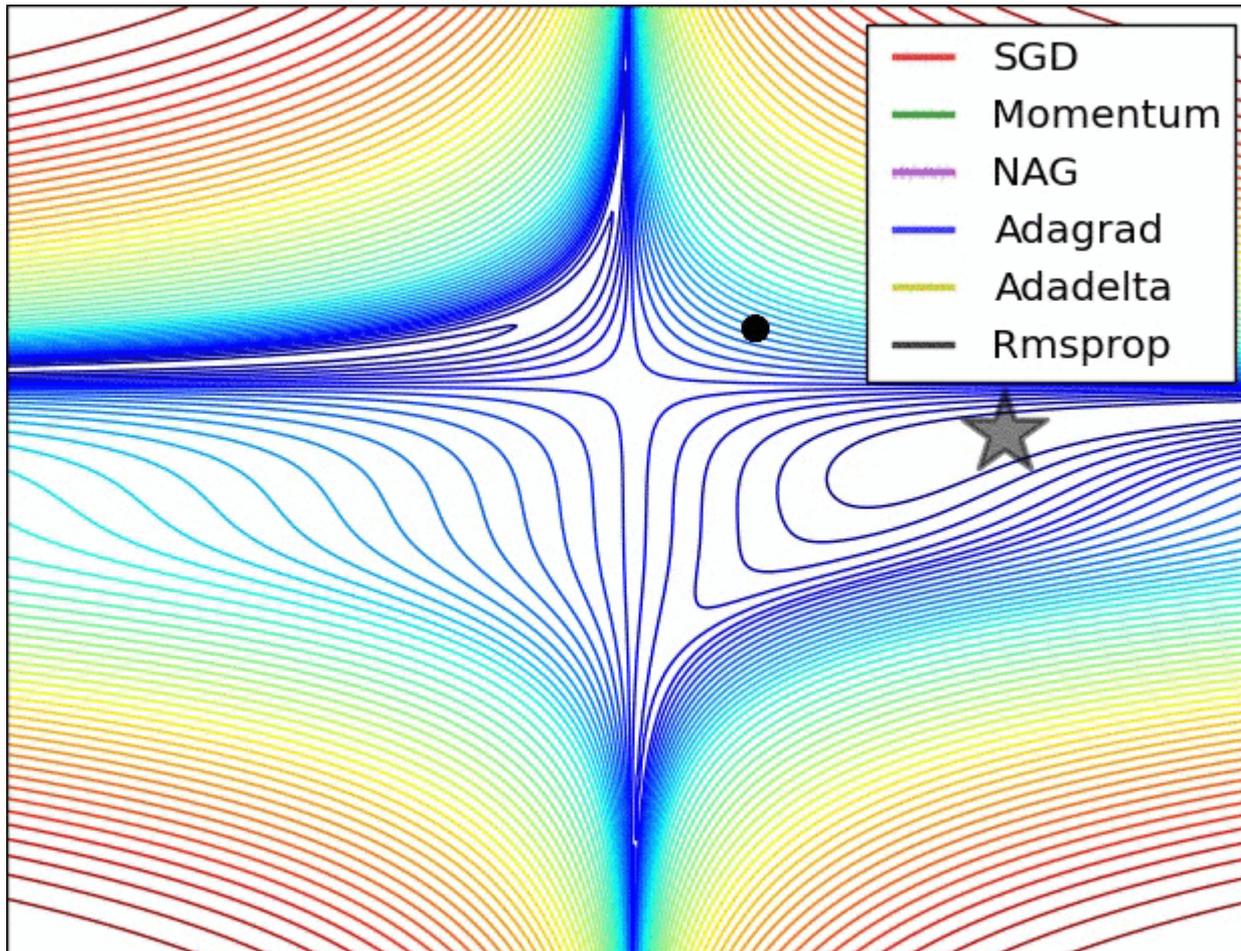
$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta}{\sqrt{\hat{v}^{(t)} + \epsilon}} \hat{m}^{(t)}$$

$$\hat{m}^{(t)} = \frac{m^{(t)}}{1 - \beta_1^{(t)}}$$

$$\hat{v}^{(t)} = \frac{v^{(t)}}{1 - \beta_2^{(t)}}$$

컨볼루션 신경망의 학습

❖ 경사 하강법 및 변형 방법에 따른 학습 형태의 예



5.2.5 대표적인 컨볼루션 신경망 모델

❖ 컨볼루션 신경망 모델

- LeNet
- AlexNet
- VGGNet
- GoogleNet
- ResNet
- ResNeXt
- DenseNet
- DPN (Dual Path Network)

대표적인 컨볼루션 신경망 모델

❖ ILSVRC 대회

▪ ImageNet 데이터베이스

- 영어 단어 ontology인 WordNet의 계층구조에 따라 정리된 영상 데이터베이스

ImageNet 2011 Fall Release (32326)
plant, flora, plant life (4486)
geological formation, formation (175)
natural object (1112)
sport, athletics (176)
artifact, artefact (10504)
fungus (308)
person, individual, someone, somebody, mortal, soul (6978)
animal, animate being, beast, brute, creature, fauna (3998)
Misc (20400)
julienne, julienne vegetable (0)
raw vegetable, rabbit food (0)
pulse (0)
goa bean (0)
kidney bean (0)
navy bean, pea bean, white bean (0)
pinto bean (0)
frijole (0)



▪ 분류 경쟁 부분

- 1,000개의 부류
- 1,200,000 개의 영상 데이터
- 상위-5 오류(top-5 error rate) 평가

대표적인 컨볼루션 신경망 모델

❖ ILSVRC 대회

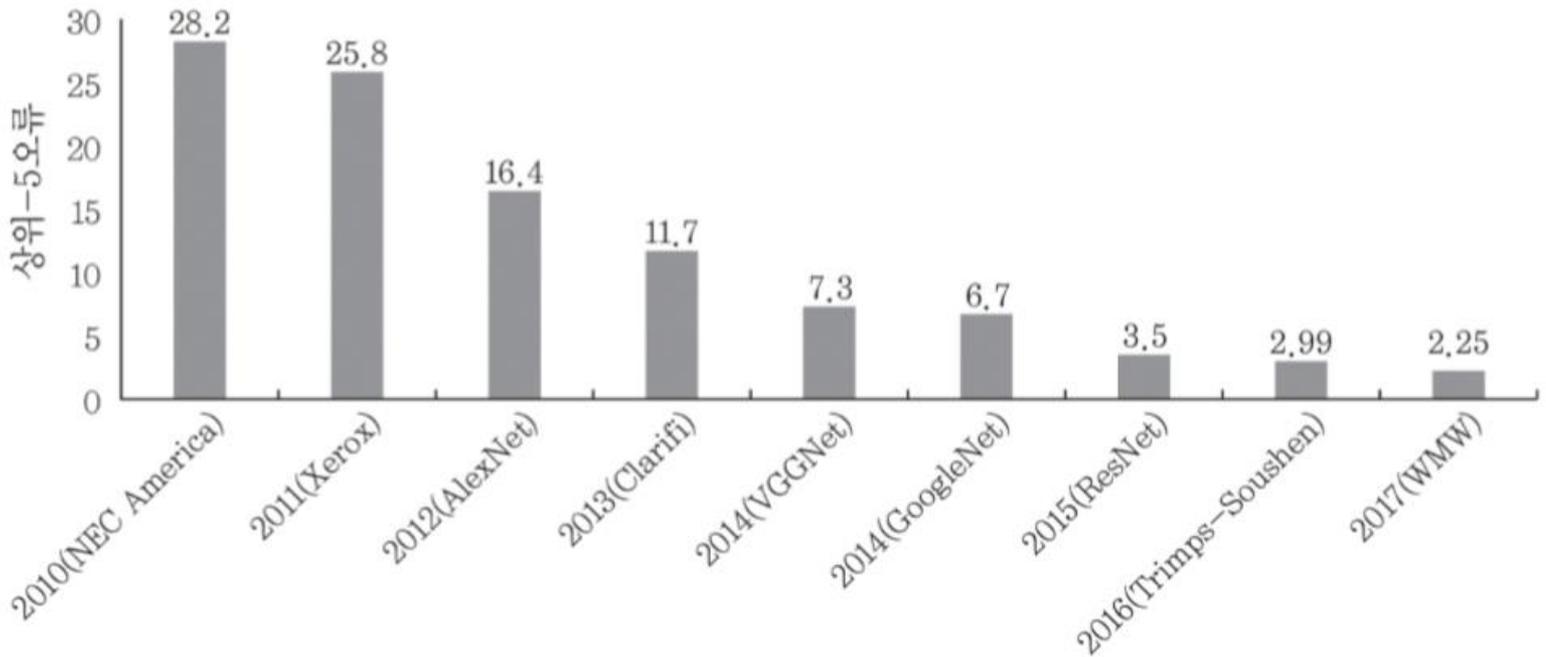


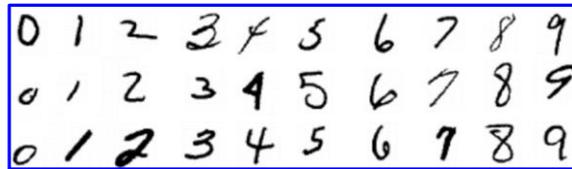
그림 5.19 ILSVRC 주요 우수팀의 성적

가로축은 연도, 괄호 안에는 팀 이름이나 모델 이름을 나타냄

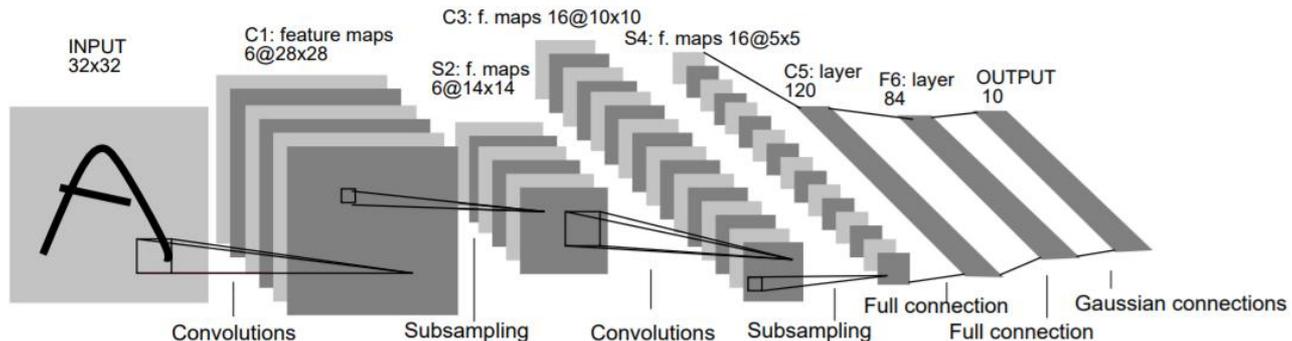
LeNet 모델

❖ LeNet 모델

- Yann LeCun 등의 제안(1998)
- LeNet5 모델
 - 5 계층 구조: Conv-Pool-Conv- Pool-Conv-FC-FC(SM)
- 입력 : 32x32 필기체 숫자 영상 (**MNIST** 데이터)



- 풀링 : 가중치x(2x2블록의 합) + 편차항
- 시그모이드 활성화 함수 사용
- 성능: 오차율 0.95%(정확도: 99.05%)



AlexNet 모델

❖ AlexNet

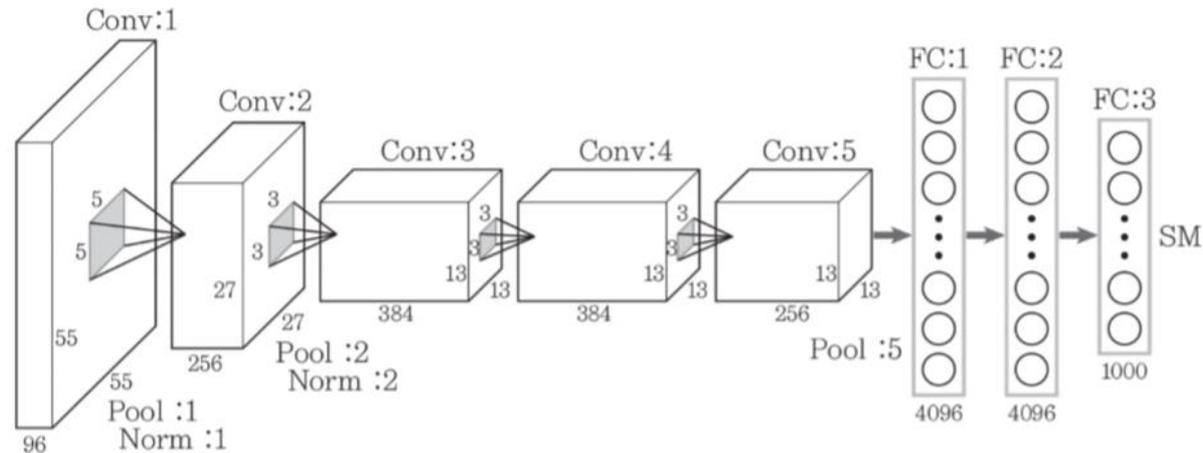
- 토론토 대학 Geoffrey E. Hinton 팀이 제안
- ILSVRC에서 2012년 우승
- 상위-5 오류율 : **16.43%**
 - 직전 년도 대비 9.4% 정확도 향상



AlexNet 모델

❖ AlexNet – cont.

- 8 계층의 구조
 - Conv-Pool-Norm-Conv-Pool-Norm-Conv- Conv-Conv-Pool-FC-FC-FC(SM)



- ReLU 함수를 사용한 첫 모델
- FC 층에 드롭아웃(dropout) 기법 사용
- 최대값 풀링(max pooling) 사용

AlexNet 모델

❖ AlexNet – cont.

- Norm: 국소 반응 정규화 연산 층
 - 인접한 여러 층의 출력값들을 이용하여 출력값 조정

$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta$$

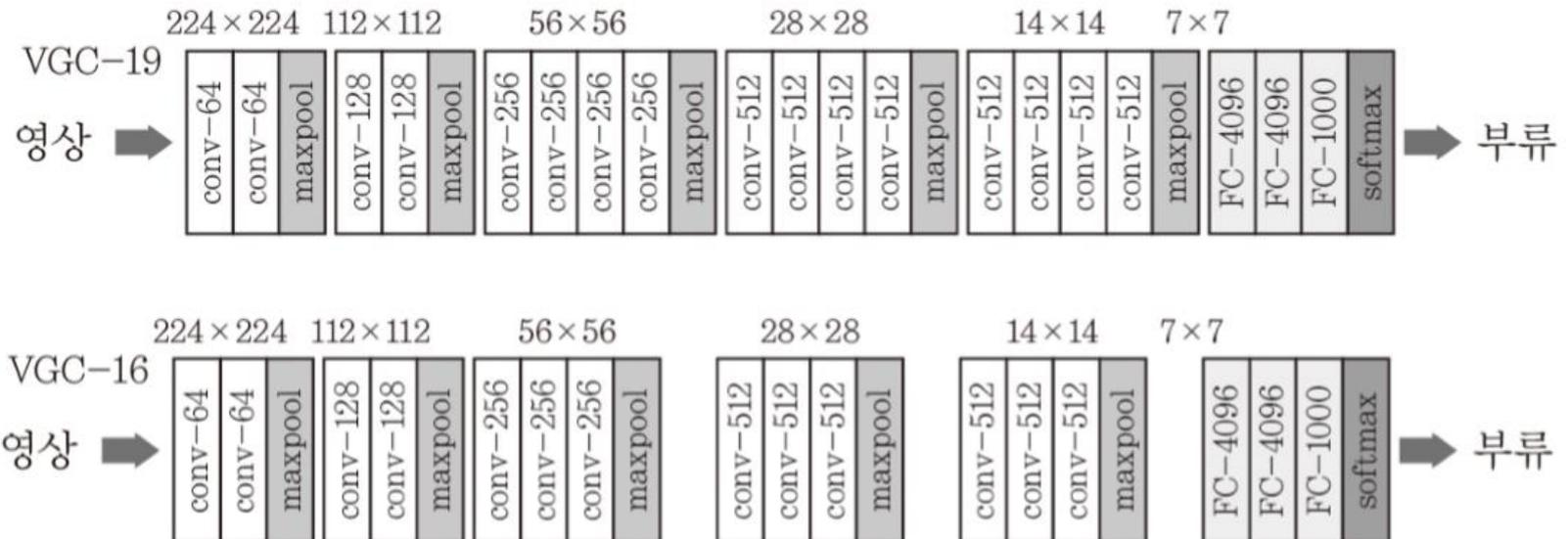
↑
위치 (x,y) 에 커널 i 를 적용하여 계산한 값

- 마지막층
 - 완전연결층(FC층)
 - 소프트맥스(SM) 사용
 - 1,000개의 부류를 나타내기 위해 1,000개의 노드

VGGNet 모델

❖ VGGNet

- 사이머니언와 지서만이 제안(2014년)
- VGG-16 모델(16개 층)
- VGG-19 모델(19개 층)
- 2014년 ILSVRC에서 2등 차지 (상위-5 오류율: 7.32%)
- 단순한 구조



VGGNet 모델

❖ VGGNet – cont.

- 모든 층에서 **3x3 필터** 사용
- **3x3** 필터 2회 적용 \Rightarrow **5x5** 필터 적용 효과
- **3x3** 필터 3회 적용 \Rightarrow **7x7** 필터 적용 효과

27 가중치

49 가중치

ReLU 3회 적용 \Rightarrow 복잡한 결정경계 표현 가능

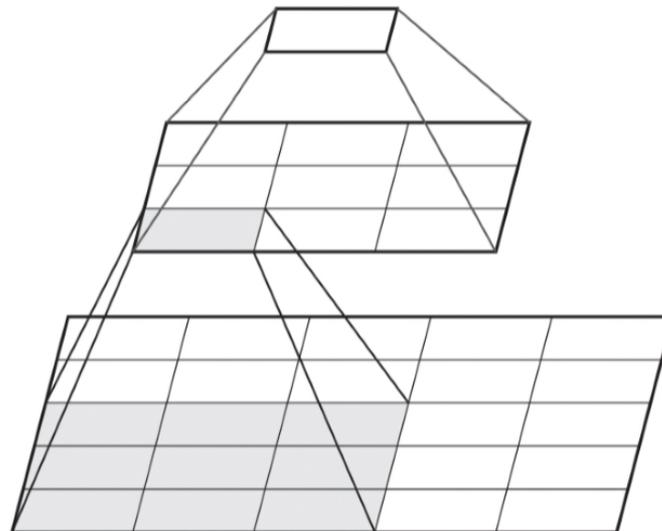
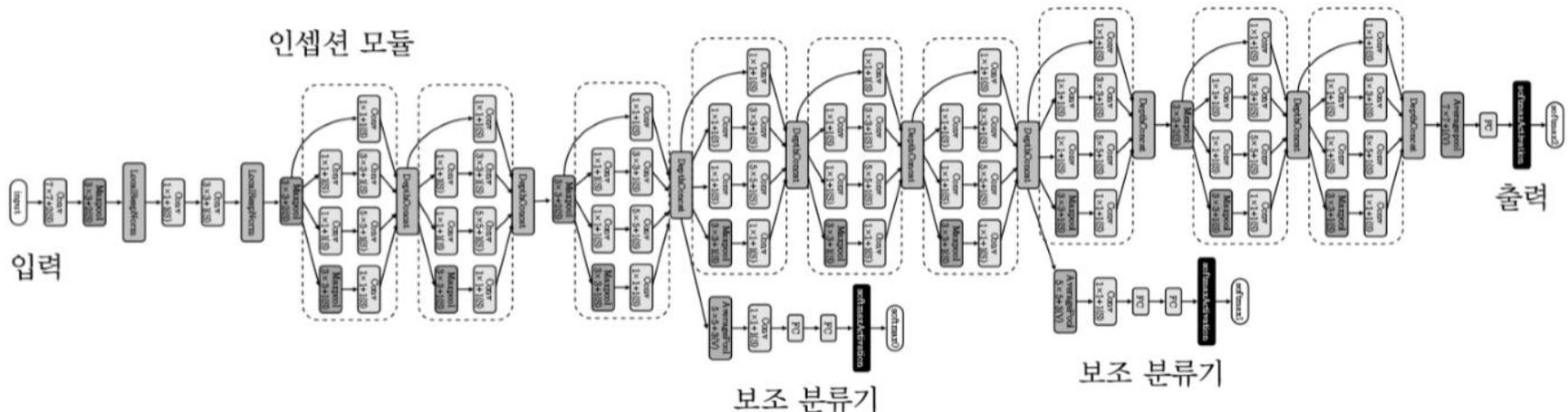


그림 5.23 2개 층의 3x3 컨볼루션에 의한 5x5 컨볼루션 구현

GoogleNet 모델

❖ GoogleNet

- 구글의 체게디 등이 개발
- 2014년 ILSVRC에서 우승(상위-5 오류율 : 6.67%)
- 22개 층의 구조
 - Conv-MPool-Conv-Incept-Incept-MPool-Incept-Incept-Incept-Incept-MPool-Incept-Incept-APool-FC-SM
 - MPool : 최대값 풀링
 - Apool: 평균값 풀링
 - Incept : 인셉션(Inception)모듈



GoogleNet 모델

❖ GoogleNet - cont.

▪ 인셉션(Inception) 모듈

- 직전 층의 처리결과에 1×1 컨볼루션, 3×3 컨볼루션, 5×5 컨볼루션을 적용
- 이들 크기의 수용장에 있는 특징들을 동시에 추출

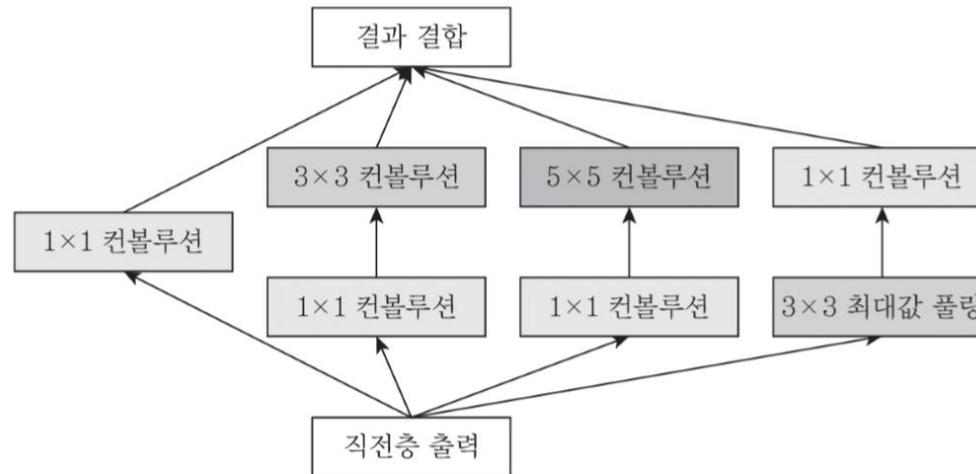


그림 5.25 GoogleNet에서 사용되는 인셉션(Inception) 모듈

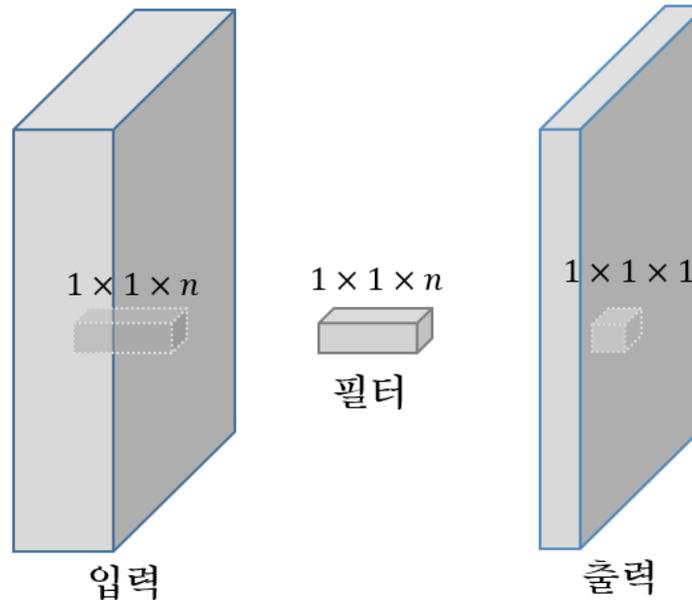
GoogleNet 모델

❖ GoogleNet - cont.

▪ 1x1 컨볼루션

- 동일한 위치의 특징지도의 값을 필터의 가중치와 선형결합
- 1x1 컨볼루션 필터의 개수를 조정하여 출력되는 특징지도의 개수를 조정

– $224 \times 224 \times 500 \Rightarrow (1 \times 1 \times 500) @ 120 \Rightarrow 224 \times 224 \times 120$



GoogleNet 모델

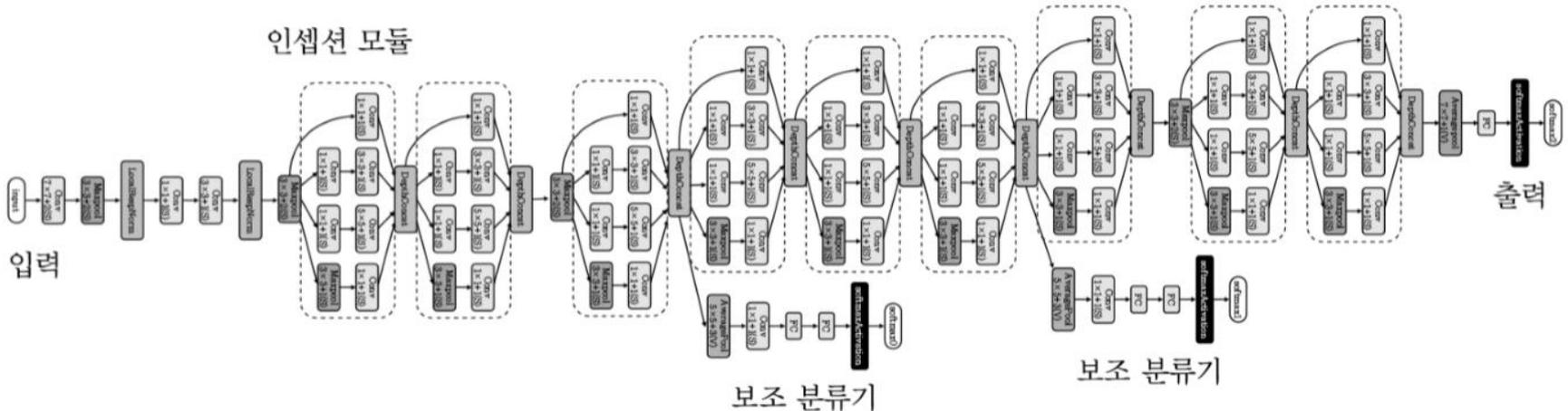
❖ GoogleNet - cont.

- 마지막 계층: 소프트맥스

- 22개 층 모델이지만, AlexNet 모델에 비해 가중치 개수는 10% 증가

- 기울기 소멸 문제 완화 장치

- 4번째, 7번째 계층에 보조 분류기 추가
- 보조 분류기를 통해 그래디언트 정보 제공



ResNet 모델

❖ ResNet (Residual Net)

- 카이밍 허 등이 개발
- 2015년 ILSVRC에서 우승(상위-5 오류율: 3.75%)
- 152개 층의 모델
 - Conv-Mpool
 - [Conv-ReLU-Conv-ReLU-Conv-ReLU]x3
 - [Conv-ReLU-Conv-ReLU-Conv-ReLU]x8
 - [Conv-ReLU-Conv-ReLU-Conv-ReLU]x36
 - [Conv-ReLU-Conv-ReLU-Conv-ReLU]x3
 - APool-FC-SM

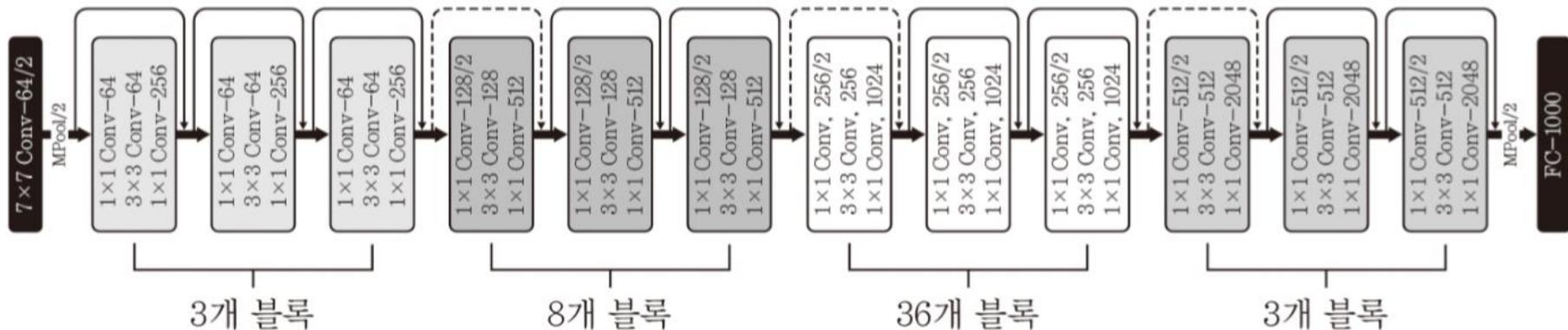


그림 5.27 ResNet의 구성

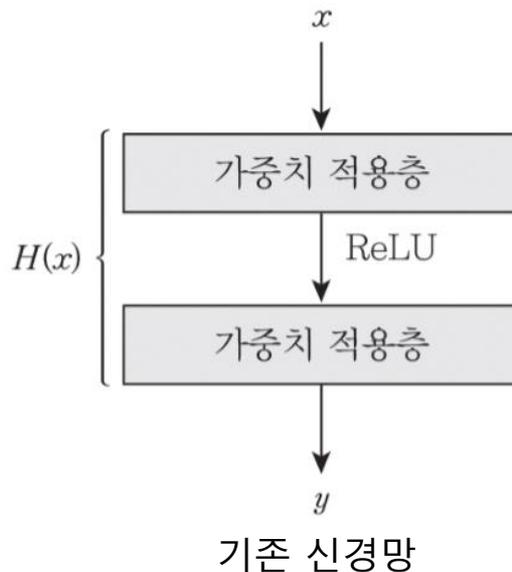
ResNet 모델

❖ ResNet – cont.

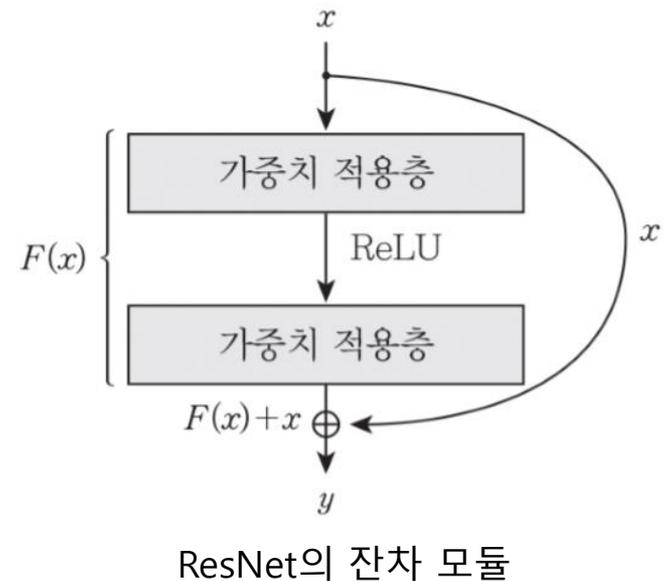
▪ 다수의 층 사용

- 상위 계층에서 의미있는 특징 추출 가능
- 다수 계층 사용시 기울기 소멸 문제 발생

▪ 잔차 모듈(residual module)



$$y = H(x)$$

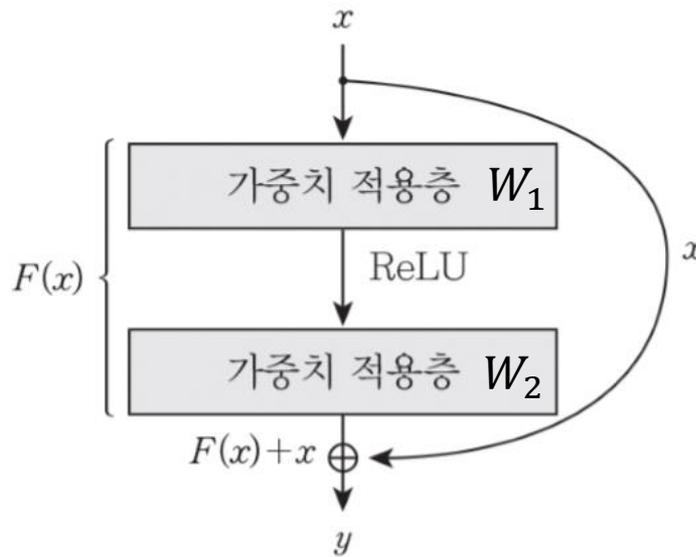


$$F(x) = y - x; \quad y = F(x) + x.$$

ResNet 모델

❖ ResNet – cont.

▪ 잔차 모듈



- 지름길 연결
- 항등 사상

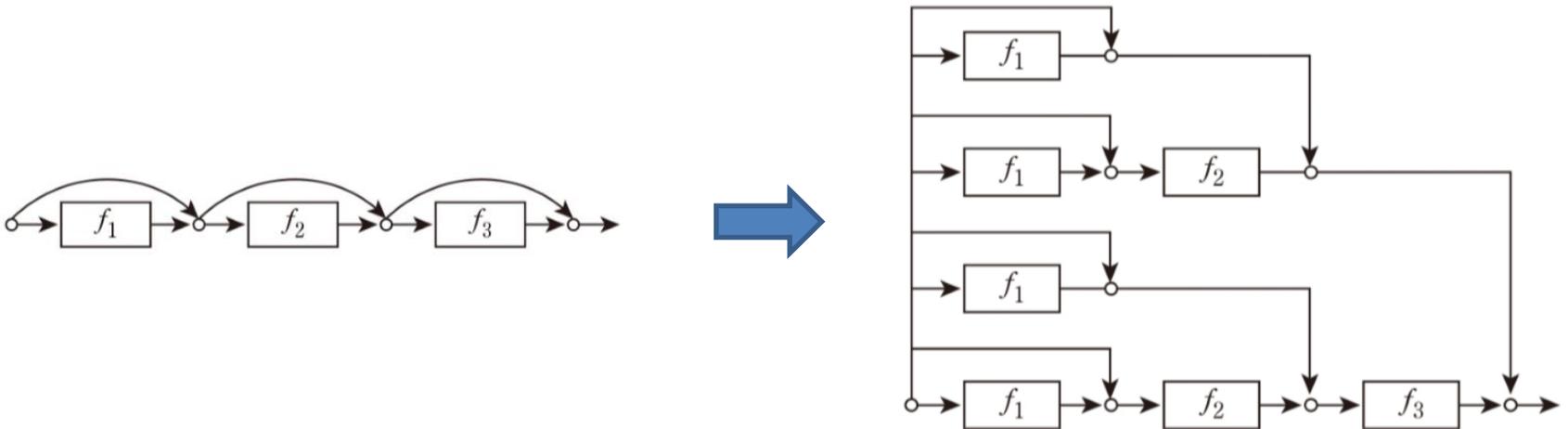
• 잔차 모듈 $F(x)$ 의 학습

$$y = F(x) + x = W_2 \rho(W_1 x) + x$$

ResNet 모델

❖ 잔차 모듈의 특징

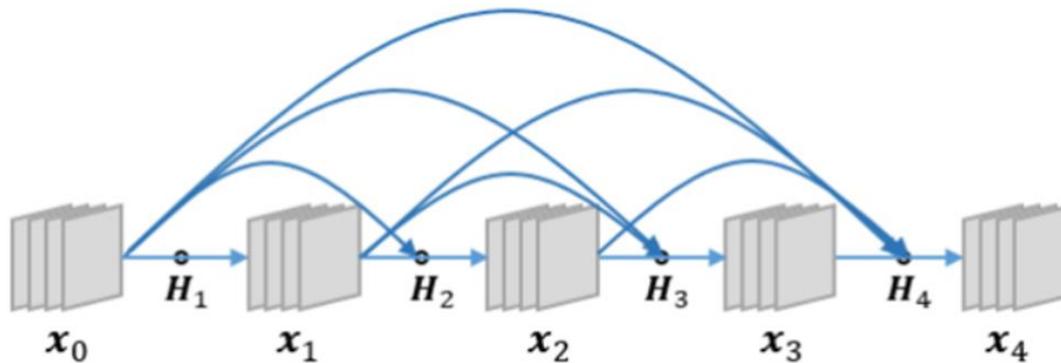
- 기대하는 출력과 유사한 입력이 들어오면 영벡터에 가까운 값을 학습
⇒ 입력의 작은 변화에 민감 ⇒ **잔차 학습**
- 다양한 경로를 통해 복합적인 특징 추출
 - 필요한 출력이 얻어지면 컨볼루션 층을 건너뛸 수 있음
 - 다양한 조합의 특징 추출 가능



DenseNet 모델

❖ DenseNet

- 가오 후양(Gao Huang) 등이 개발 (2016)
- 각 층은 모든 앞 단계에서 올 수 있는 지름길 연결 구성



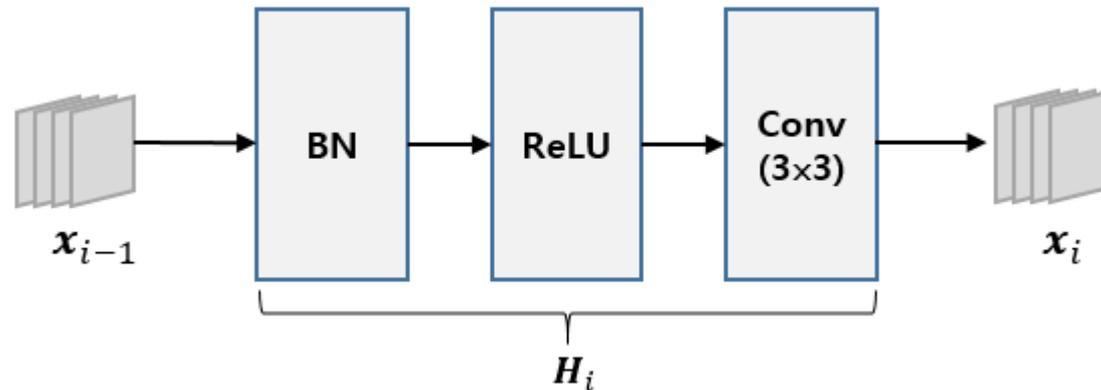
$$x_i = H_i([x_0, x_1, \dots, x_{i-1}])$$

↑
배치 정규화, ReLU, 컨볼루션 연산

DenseNet 모델

❖ DenseNet – cont.

- 노드의 연산: H_i
 - 배치 정규화(BN)-ReLU-(3x3 컨볼루션)
 - 각 층은 입력 특징지도와 같은 차원의 특징지도 생성



- 병목층
 - 1x1 컨볼루션
 - 출력되는 특징지도의 채널 수 축소
- 병목층이 있는 층
 - BN-ReLU-(1x1 컨볼루션)-BN-ReLU-(3x3 컨볼루션)

DenseNet 모델

❖ DenseNet – cont.

- 특징지도의 크기를 줄이기 위해 풀링 연산 적용 필요
- 밀집 블록(dense block)과 전이층(transition layer)으로 구성
 - 전이층 : 1x1 컨볼루션과 평균값 풀링(APool)으로 구성



그림 5.35 밀집 블록으로 구성된 DenseNet

DPN 모델

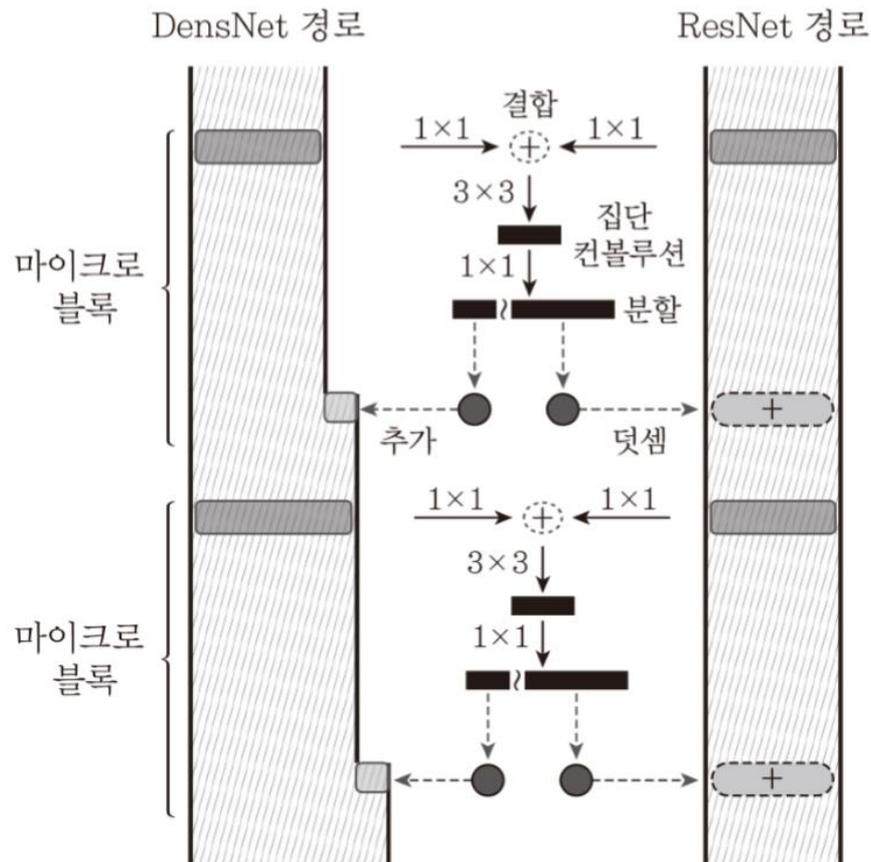
❖ DPN (Dual Path Network)

- ResNet과 DenseNet을 결합한 모델
- **ResNet**
 - 이전 단계의 동일한 특징 정보가 각 단계에 전달되어 이들 특징을 재사용하도록 하는 경향
 - 상대적으로 이전 단계의 특징들로부터 새로운 특징을 만드는 것에 소극적
- **DenseNet**
 - 새로운 특징이 추출될 가능성이 높음
 - 이전에 추출된 특징이 다시 추출될 가능성도 높음

DPN 모델

❖ DPN – cont.

- 마이크로 블록에서 DenseNet과 ResNet의 특징 결합



5.2.6 딥러닝 신경망의 전이 학습

❖ 전이 학습(transfer learning)

- 큰 규모의 딥러닝 신경망을 학습시킬 때는, 많은 학습 데이터와 상당한 학습 시간이 필요
- 대규모 영상 데이터베이스인 ImageNet 데이터를 학습한 여러 컨볼루션 신경망 모델 공개
- 공개된 모델을 가져다가 누구나 자신의 문제가 적용해 볼 수도 있고, 모델의 일부 활용 가능
- 학습된 컨볼루션 신경망의 컨볼루션 층들을 가져오고 뒤 단계에서 분류하는 다층 퍼셉트론 모델을 붙여서 학습