



Day 4_convolutional neural network

Youngpyoryu@dongguk.edu

류 영 표



■ 근무 경력

- 동국대학교 수학과 졸
- 한국파스퇴르연구소 Image Mining 인턴(Deep learning)
- (주)셈웨어(수학컨텐츠, 데이터 분석 개발 및 연구인턴)
- 동국대학교 수학과대학원 응용수학 석사수료

■ 강의 경력

- 제 1회 대구대학교 수학과 빅데이터 여름학교 강사 및 조교
- 제 2회 대구대학교 수학과 빅데이터 여름학교 강사 및 조교
- 현대자동차 연구원 강의(인공지능/머신러닝/딥러닝/강화학습)
- 딥러닝 집중 교육과정 강사
- (재) 윌튼블록체인 6일 과정(파이썬기초, 크롤링, 머신러닝)

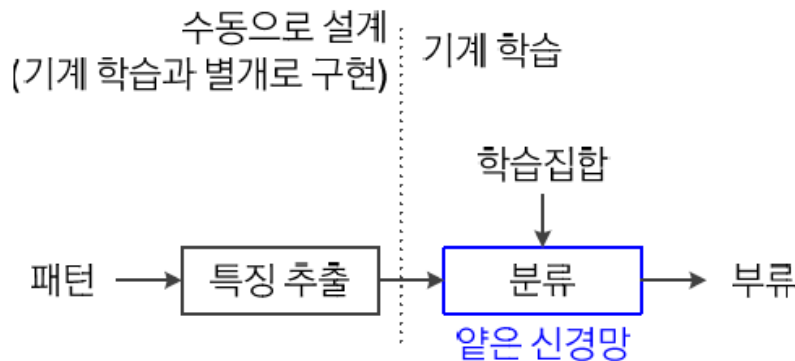
■ 주요 프로젝트 및 기타사항

- 제 4회 산업수학 스터디 그룹(산업문제 : 피부암 분류)
- 제 1회 산업 수학 스터디 그룹(산업문제 : 질병에 영향을 미치는 유전자 정보 분석)
- 인공지능(AI)기반 데이터사이언티스트 전문가 양성과정 1기 수료
- 제1회 인공지능(AI)기반 데이터사이언티스트 전문가 양성과정 최우수상 수상(Q&A 챗봇)
- 빅데이터 여름학교 참석(문제 : 혼잡도를 최소화하는 새로운 노선 건설 위치의 최적화 문제)

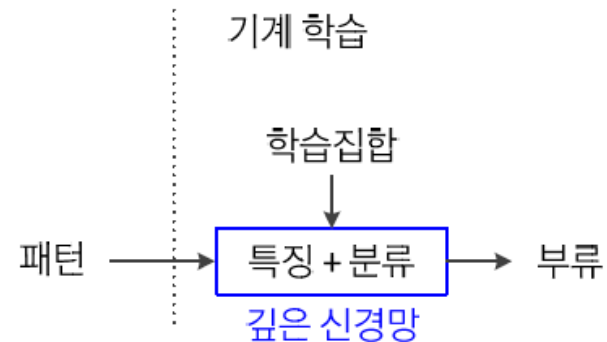
특징 학습의 부각

■ 기계 학습의 패러다임의 변화

- 고전적인 다층 퍼셉트론
 - 은닉층은 특징 추출기 (3.3.4절의 [그림 3-16])
 - 얇은 구조이므로 센서로 획득한 원래 패턴을 그대로 입력하면 낮은 성능
 - 따라서 사람이 수작업 특징을 구상하고 구현하여 신경망에 입력함
- 현대 기계 학습 (딥러닝)
 - 특징 추출을 학습으로 설계 ← **특징 학습**
 - 원래 패턴이 신경망의 입력 ← **통째 학습** end-to-end learning



(a) 고전적 패러다임에서의 수작업 특징



(b) 딥러닝에서의 특징 학습

그림 4-1 기계 학습의 패러다임 변화

특징 학습의 부각

■ 특징 학습^{feature learning} (또는 표현 학습^{representation learning})

- 앞단계 은닉층은 에지나 코너와 같은 저급 특징 추출
- 뒷단계 은닉층은 추상적인 형태의 고급 특징을 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 ([그림 8-24]의 자연 영상에 주석 달기 응용) 등이 가능해짐

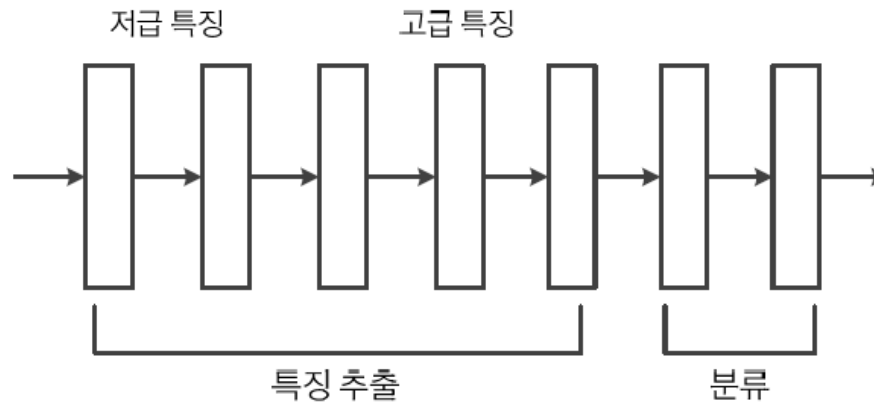
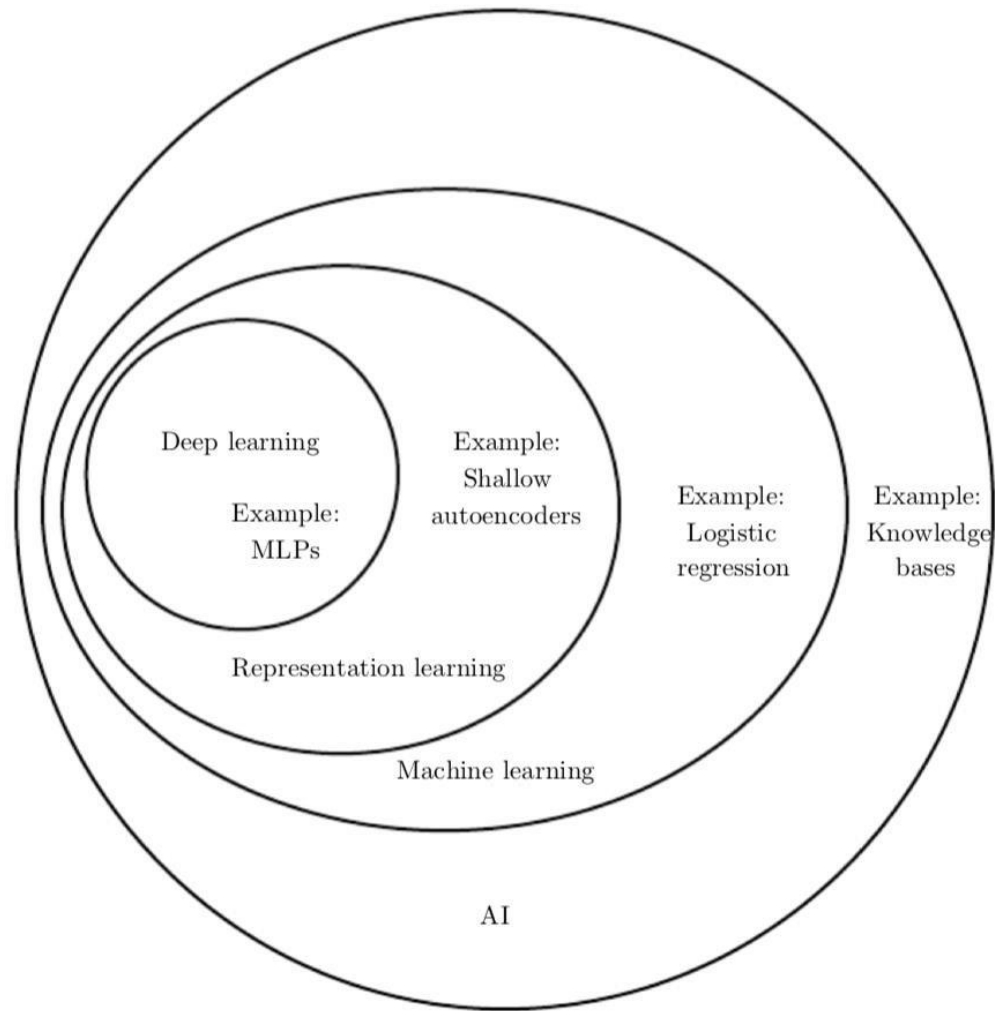


그림 4-2 깊은 신경망의 처리 절차

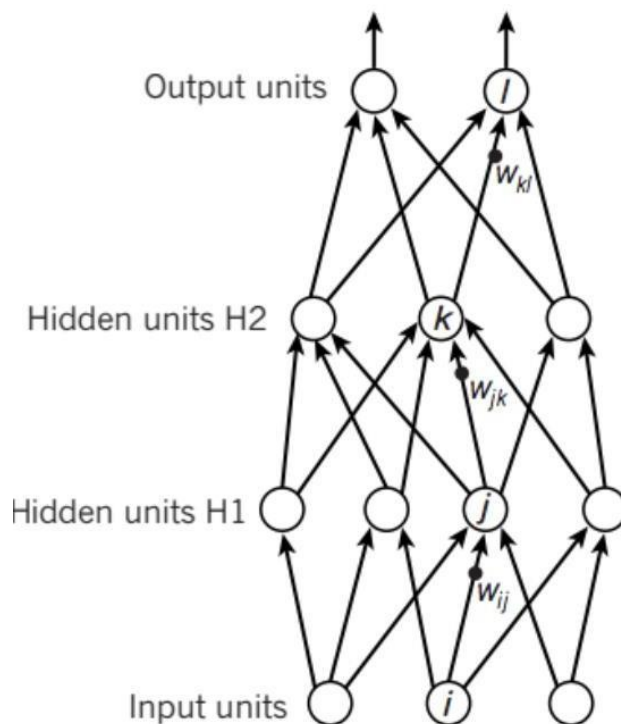
Methodology: Why Deep Learning?



Methodology: Why Deep Learning?

Deep Learning is ...

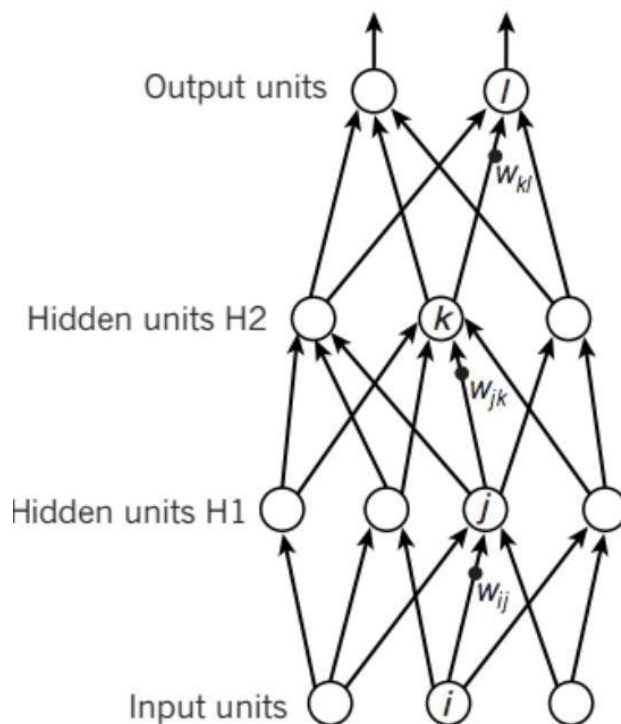
- A branch of machine learning based on a set of algorithms that attempt to model high level abstractions in data.



Methodology: Why Deep Learning?

Deep Learning is ...

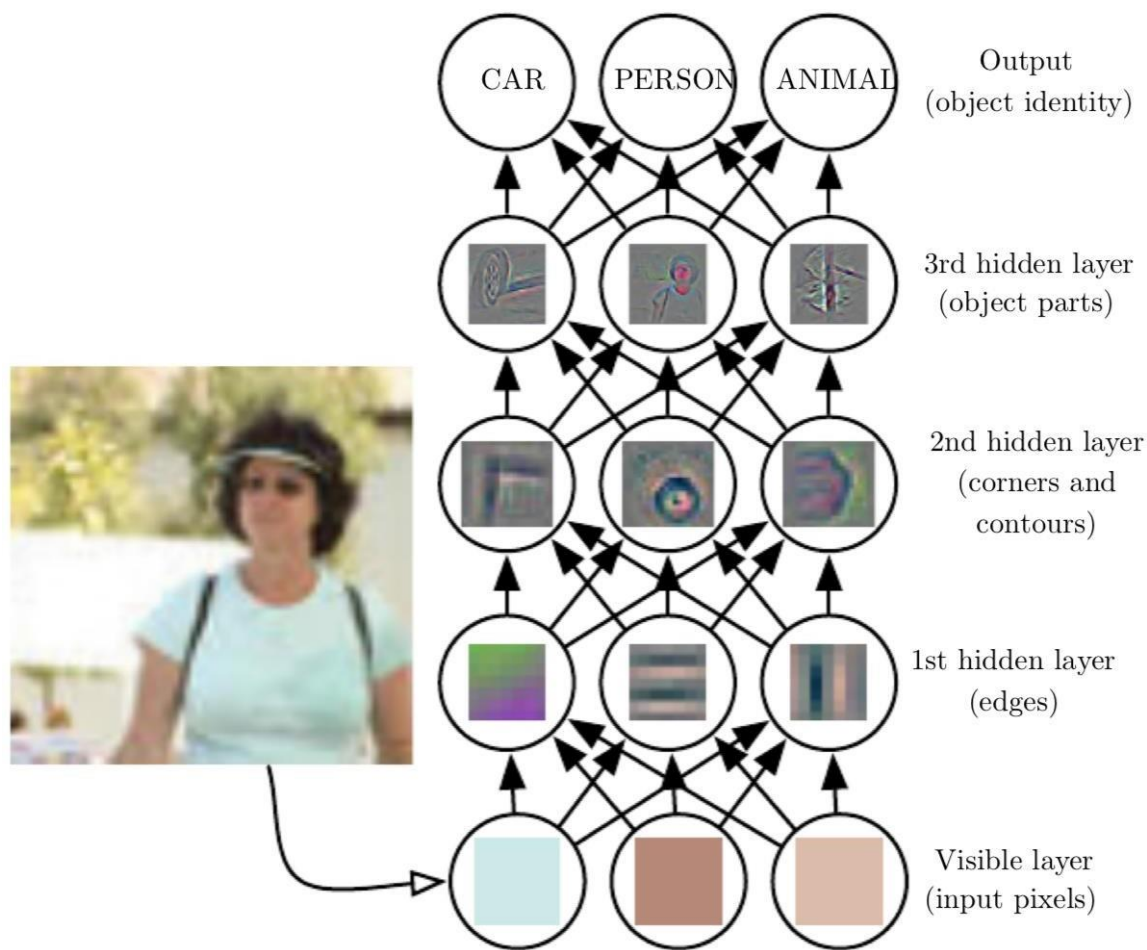
- A branch of machine learning based on a set of algorithms that attempt to model high level abstractions in data.



Methodology: Why Deep Learning?

Deep Learning is ...

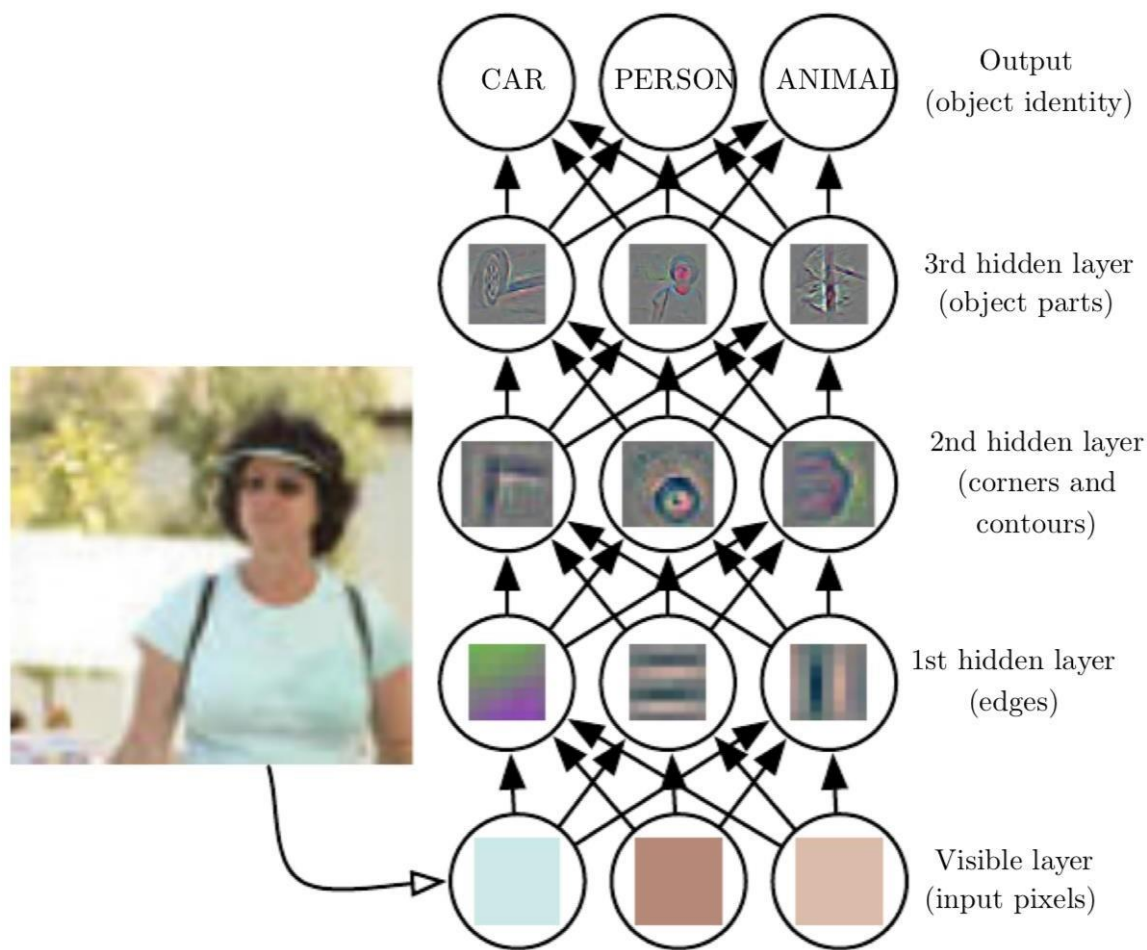
- A kind of representation learning.



Methodology: Why Deep Learning?

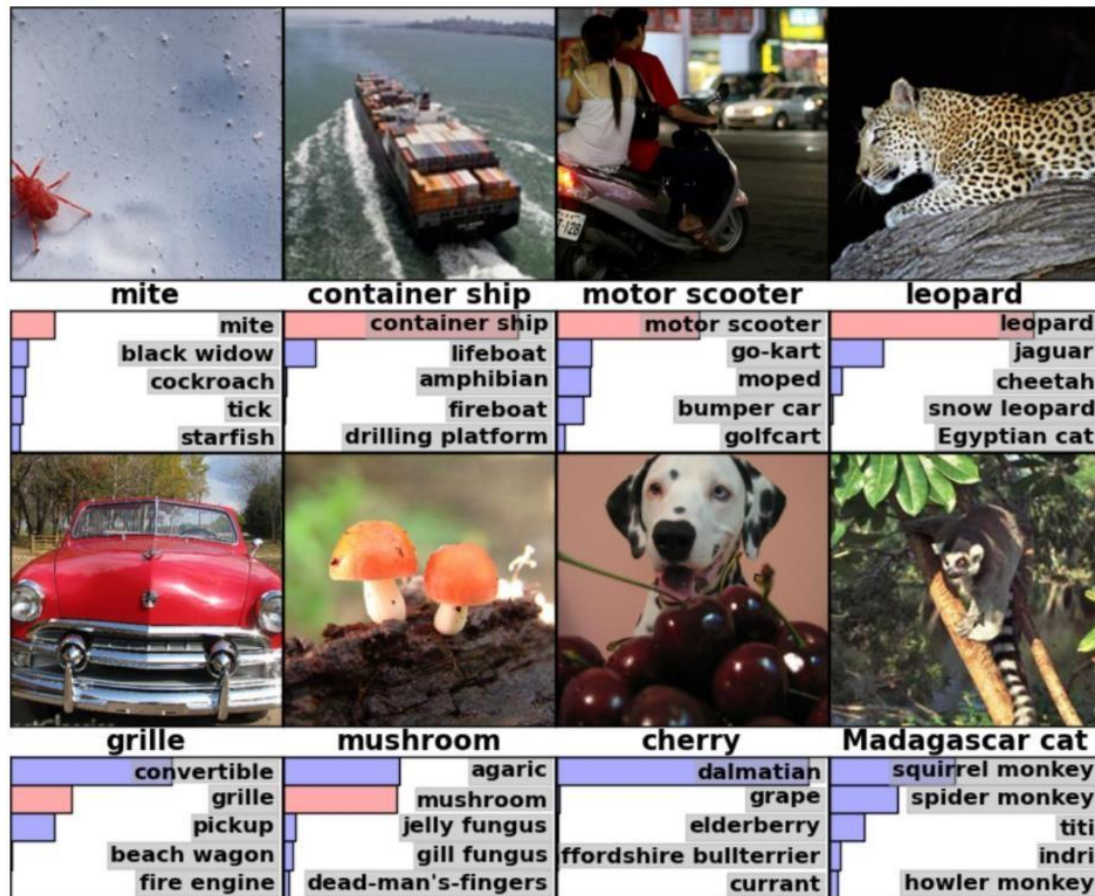
Deep Learning is ...

- A kind of representation learning.



Methodology: Why Deep Learning?

Performance improvements by deep learning



IMAGENET

- 1,000 object classes
- Images
 - Train: 1.2M
 - Test: 100k

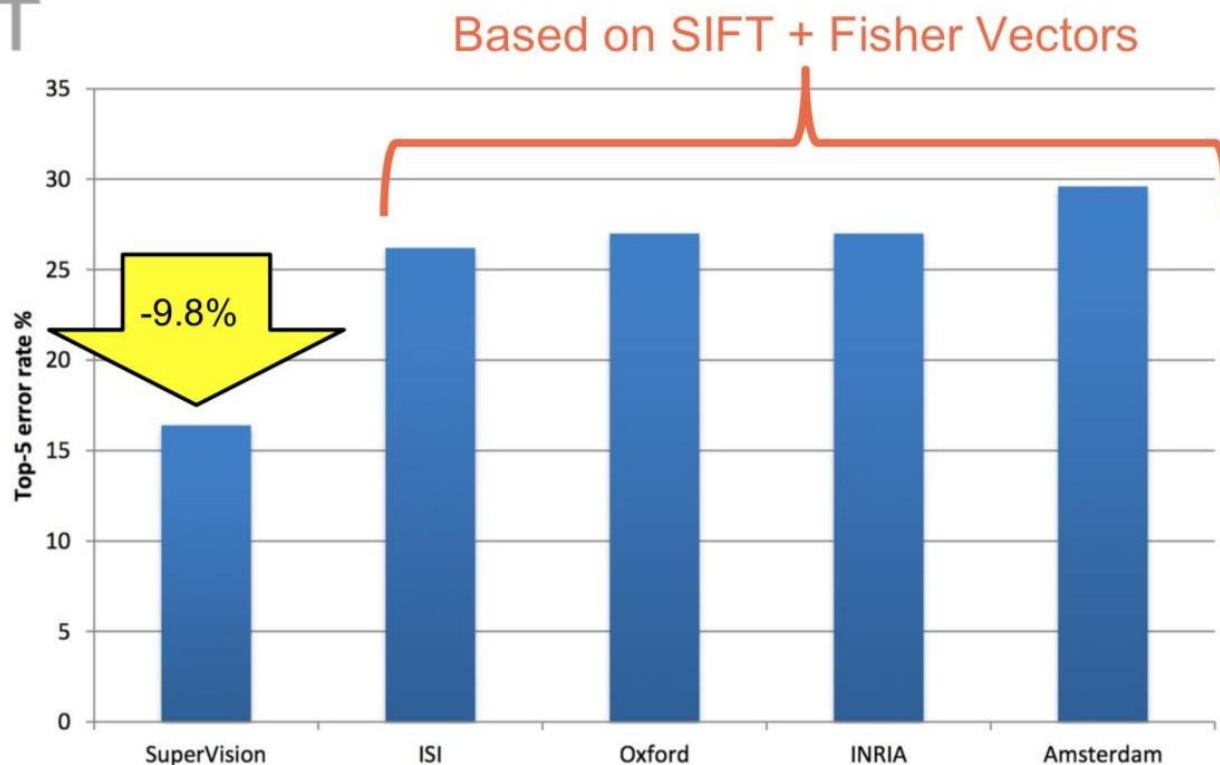
Methodology: Why Deep Learning?

Performance improvements by deep learning

- Better than previous machine learning methods

IMAGENET

Slide credit:
[Rob Fergus](#) (NYU)

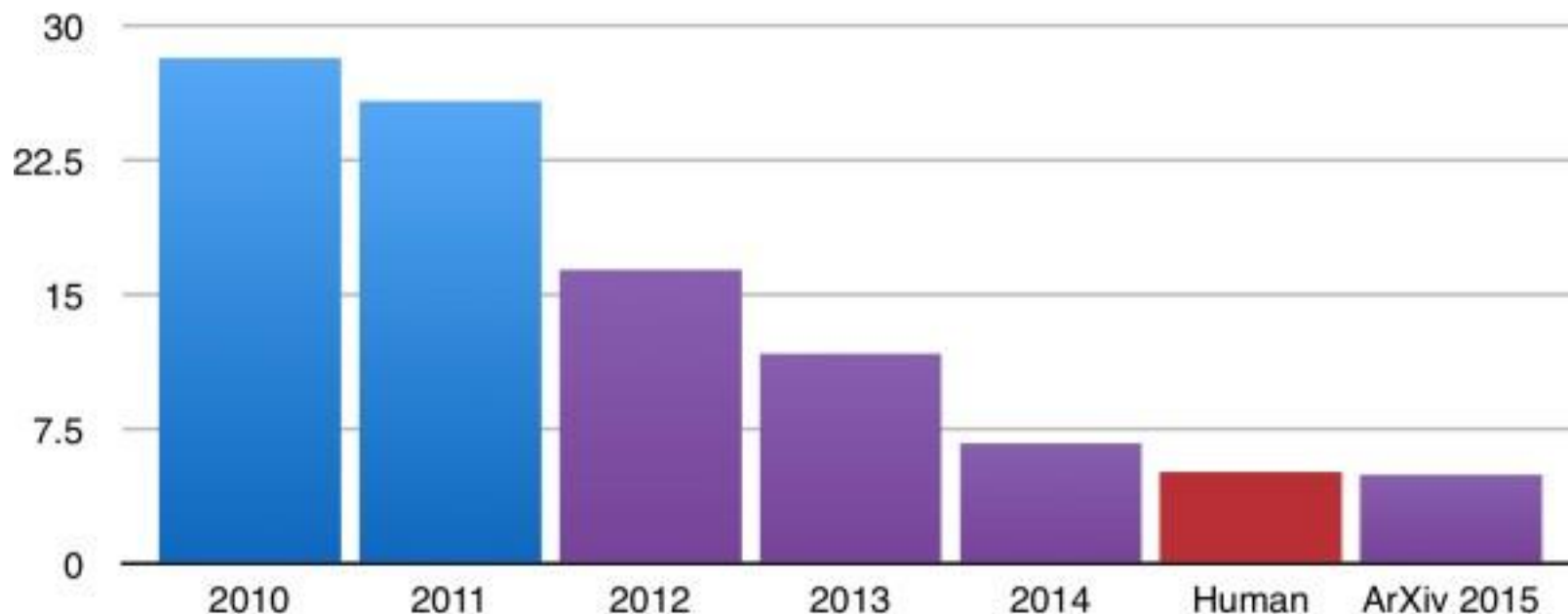


Methodology: Why Deep Learning?

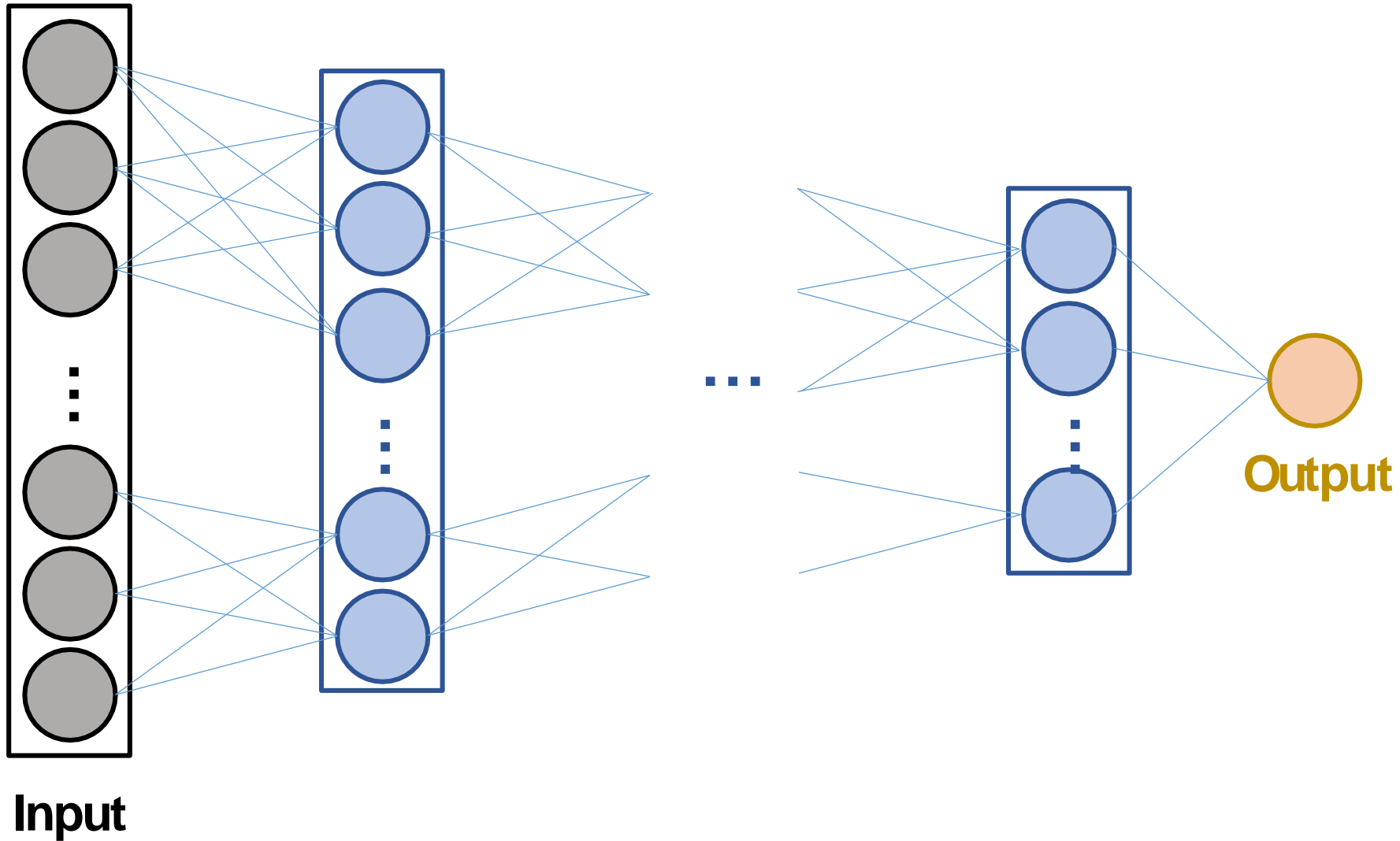
Performance improvements by deep learning

- Even better than human accuracy

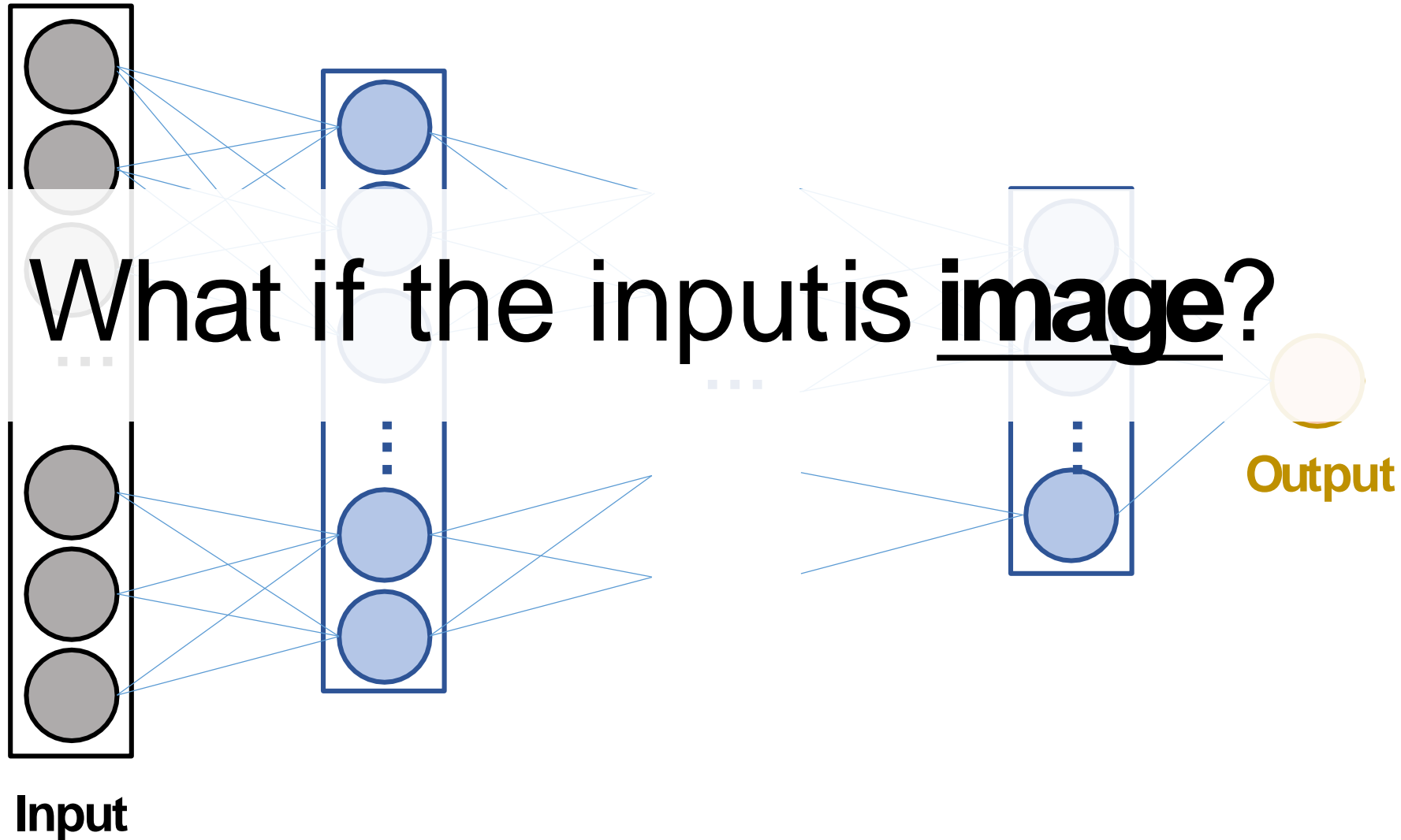
ILSVRC top-5 error on ImageNet



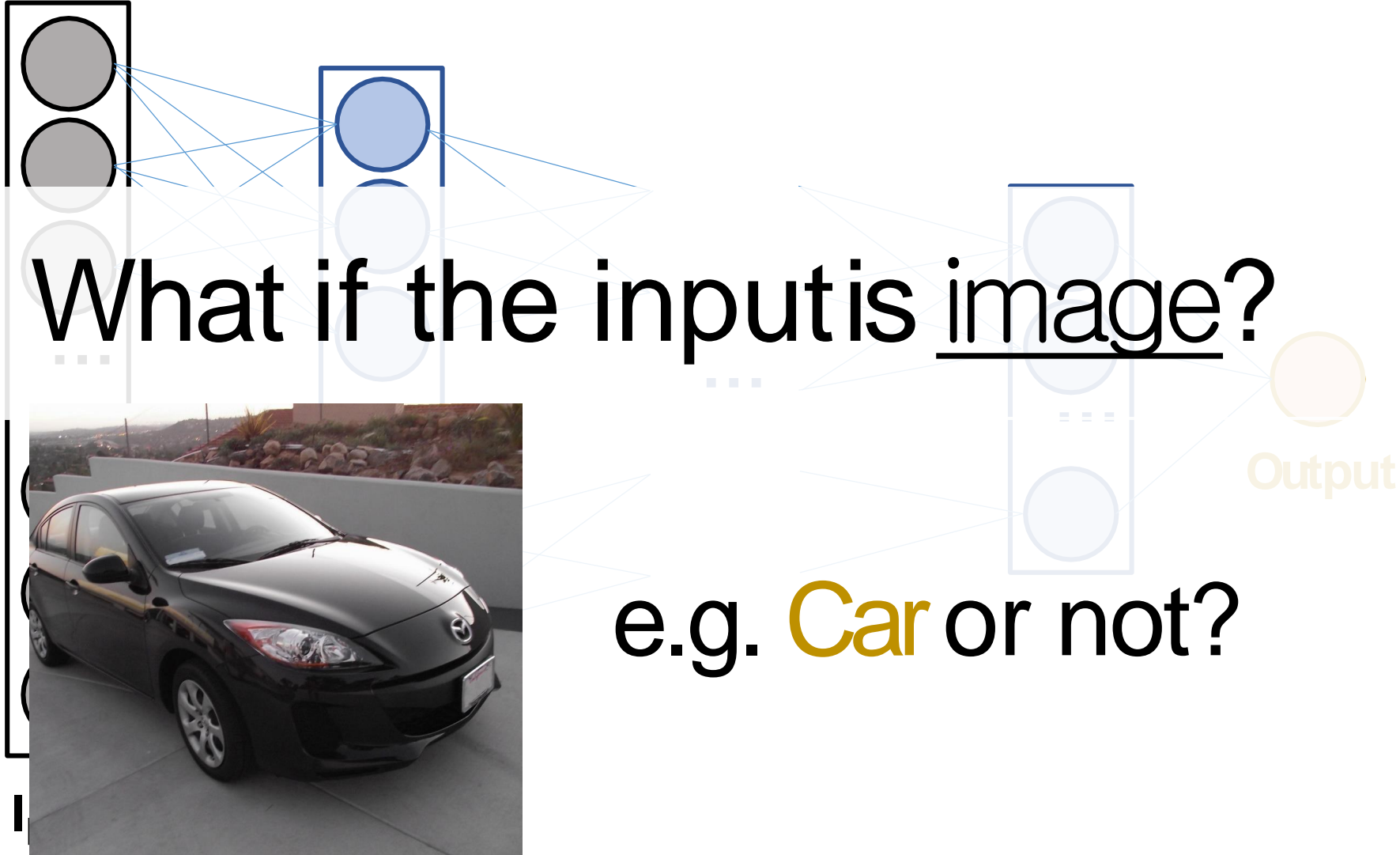
Revisited: Neural Networks



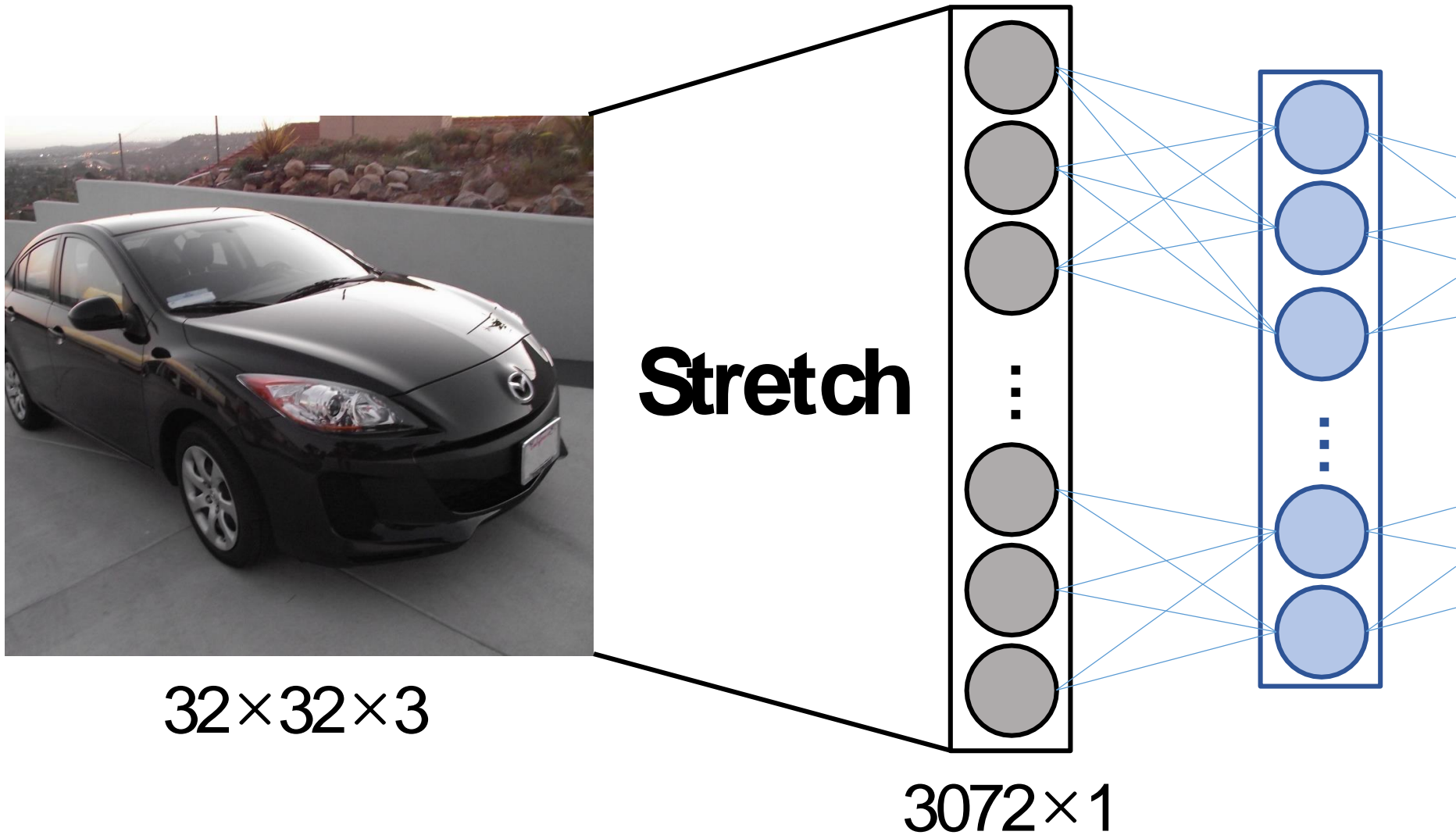
Revisited: Neural Networks



Revisited: Neural Networks

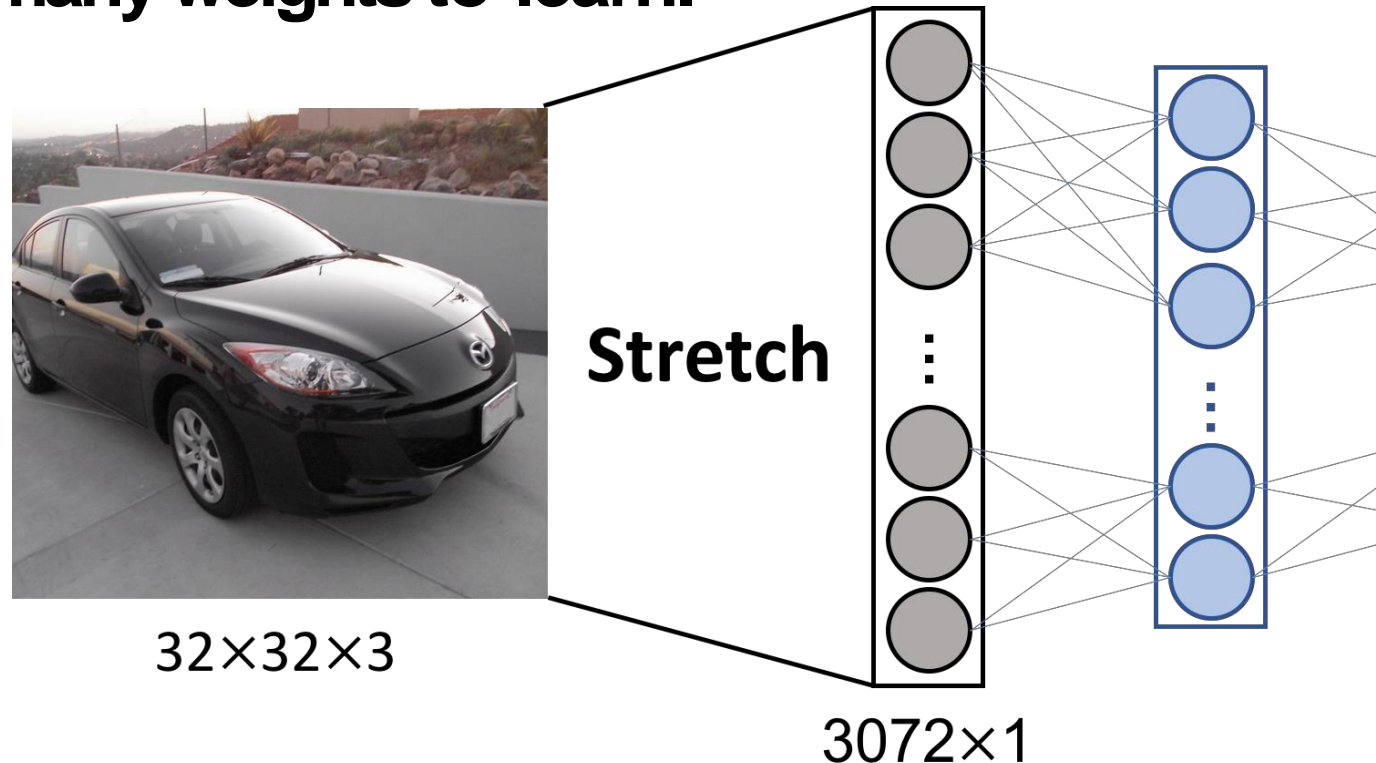


Revisited: Neural Networks



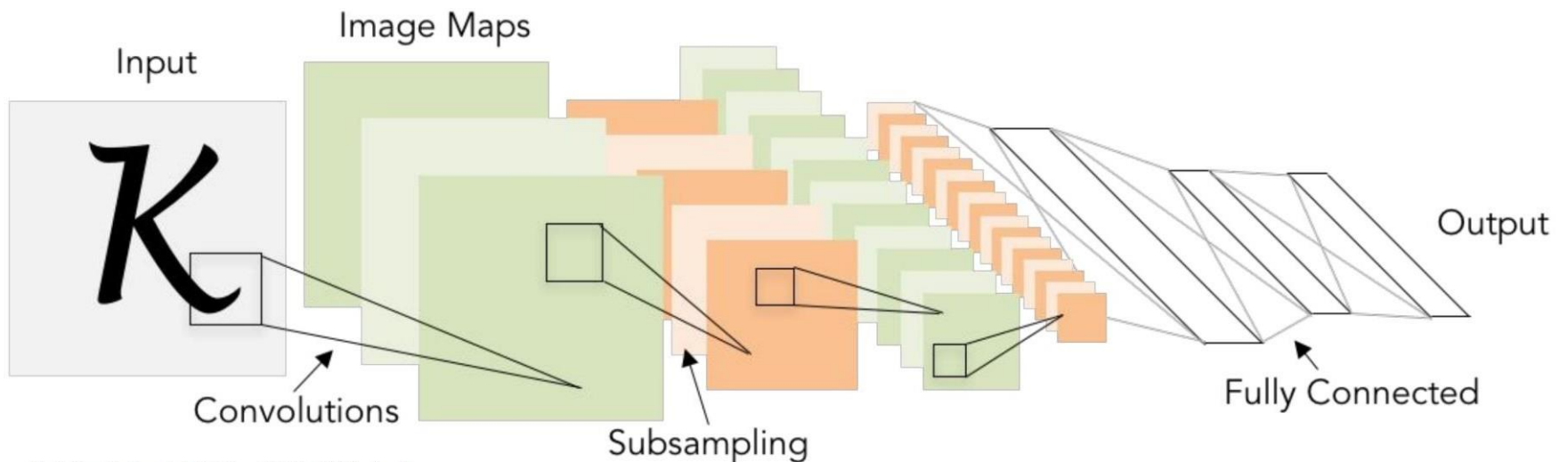
Revisited: Neural Networks

- However, this method (model) has some problems.
 1. **Ignore spatial (topological) information.**
 2. **Has too many weights to learn.**



Convolutional Neural Networks

- Key Idea
 1. **Sparse connectivity**
 2. **Parameter sharing**
 3. **Invariant to translation**



컨볼루션 신경망 CNN(convolutional neural network)

■ DMLP와 CNN의 비교

■ DMLP

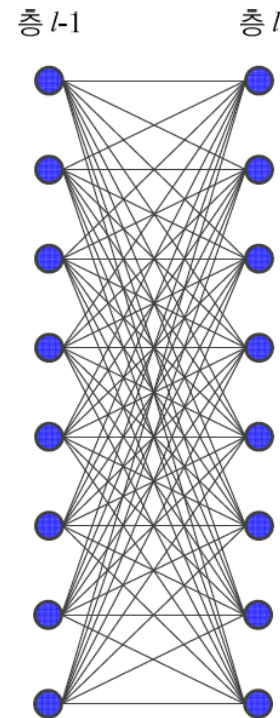
- 완전연결 구조로 높은 복잡도
- 학습이 매우 느리고 과잉적합 우려

■ CNN

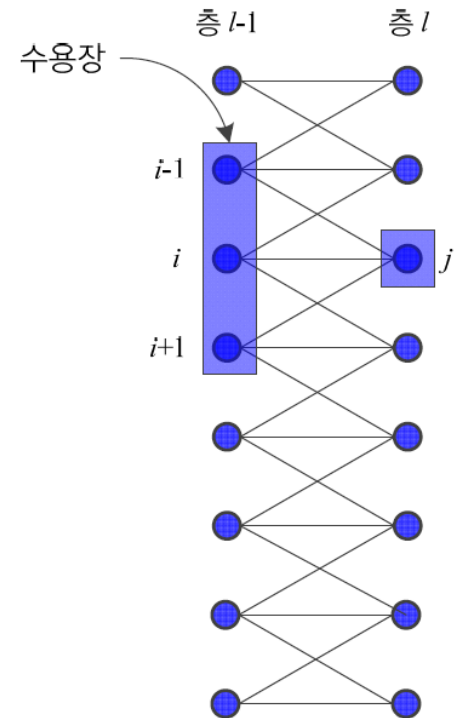
- 컨볼루션 연산을 이용한 부분연결(희소 연결) 구조로 복잡도 크게 낮춤
- 컨볼루션 연산은 좋은 특징 추출

■ CNN

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



(a) DMLP(완전연결)

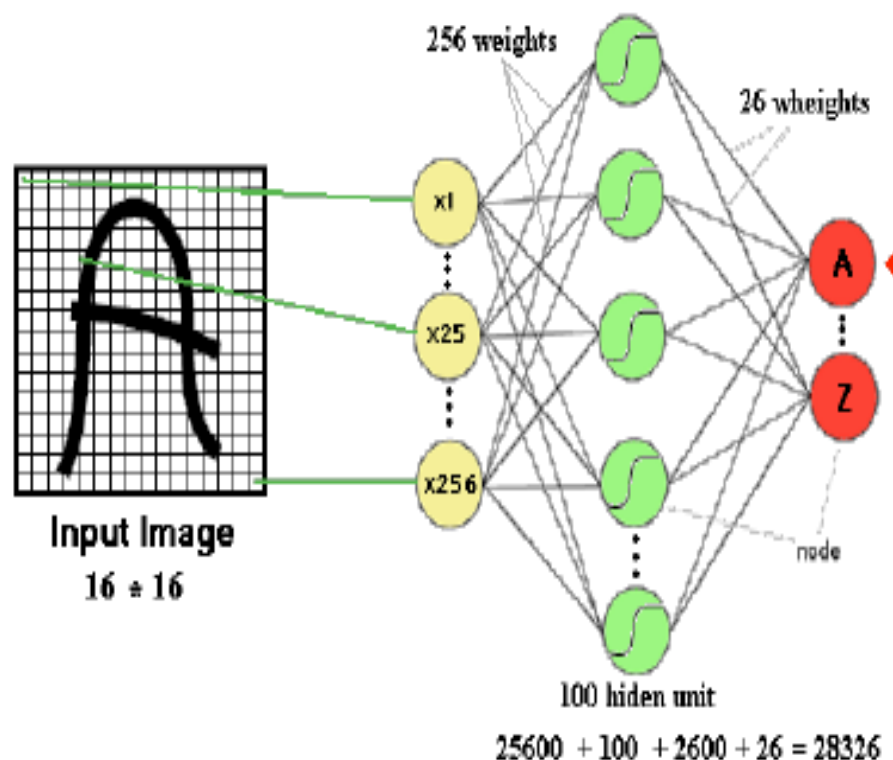


(b) CNN(부분연결)

그림 4-5 CNN의 부분연결과 수용장

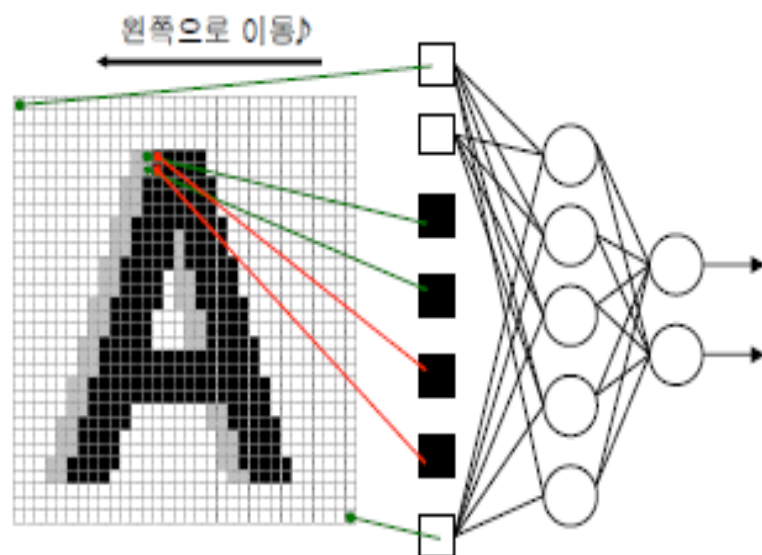
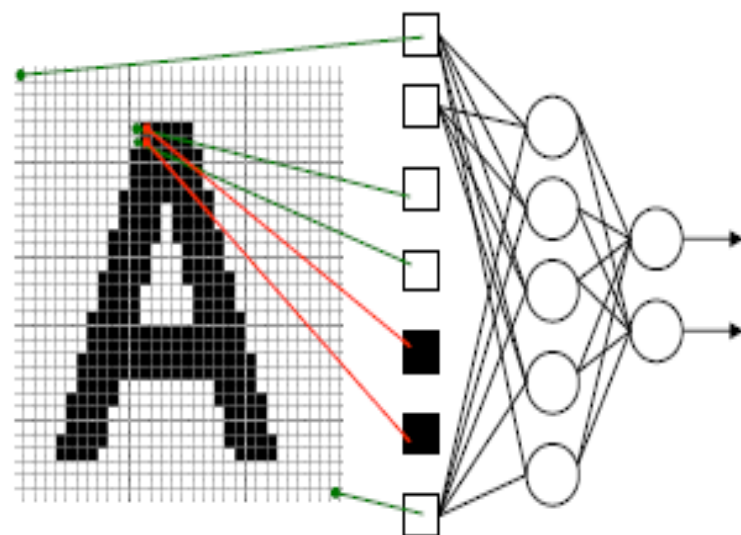
신경망을 사용한 이미지 처리

- 훈련해야할 파라미터의 개수가 매우 커짐



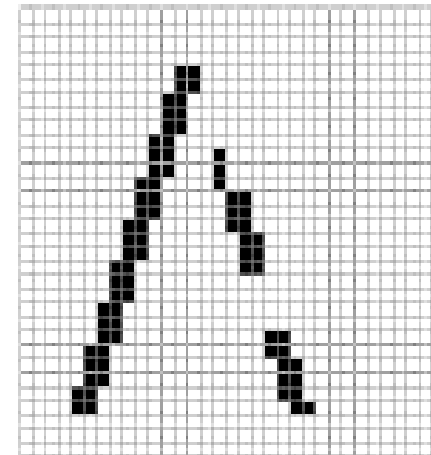
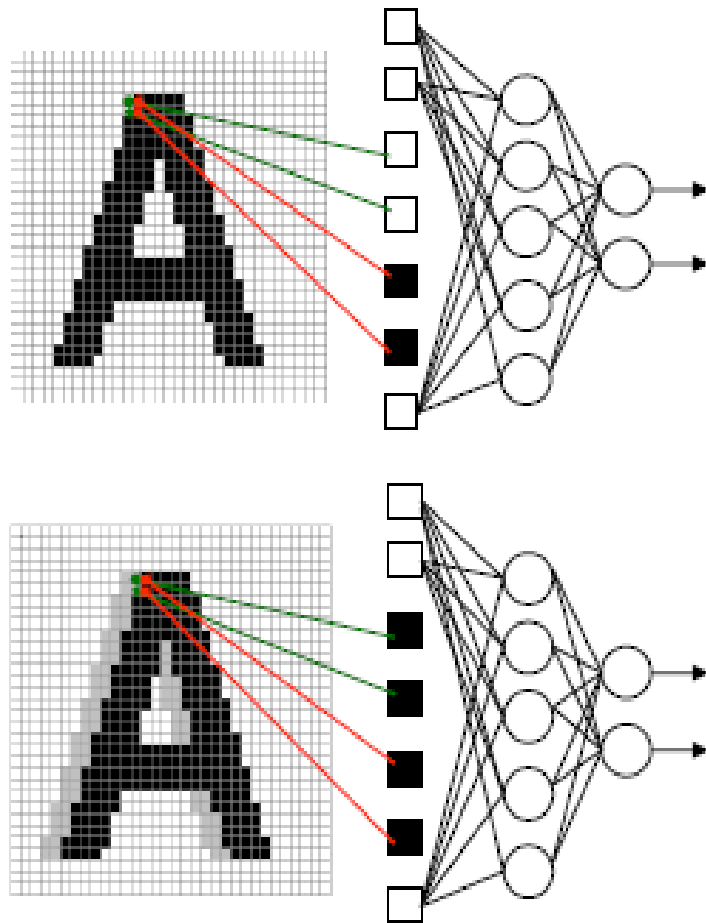
일반 신경망으로 이미지를 분류할 때 문제점

- 이미지의 위치, 크기, 각도 변화 등에 취약함
- 이미지의 위치가 변하는 경우 (1/2)



일반 신경망으로 이미지를 분류할 때 문제점

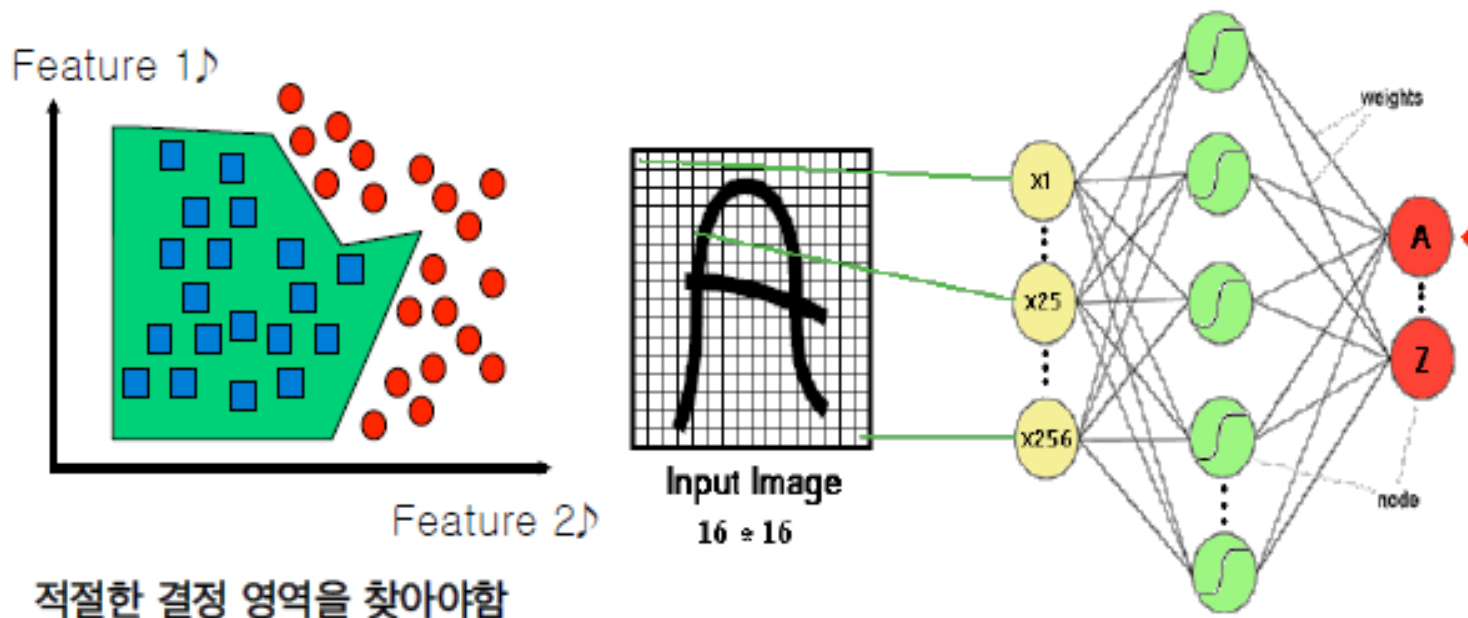
■ 이미지의 위치가 변하는 경우 (2/2)



왼쪽으로 2픽셀 이동할 경우
154개의 입력이 변화함
77개 : 검은색 -> 흰색)
77개 : 흰색 -> 검은색)

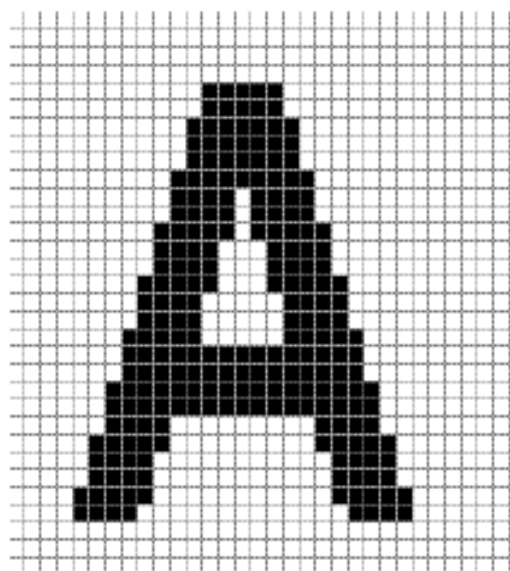
일반 신경망으로 이미지를 분류할 때 문제점

- 입력 데이터의 형태가 거의 무시됨
- 가공되지 않은 raw data로만 문제를 다룸



가능한 입력 조합의 개수

- 검은색과 흰색 패턴(흑백)의 개수: $2^{32 \times 32} = 2^{1024}$
- 256개 회색 스케일인 경우 패턴의 개수: $256^{32 \times 32} = 256^{1024}$



32 * 32 input image ♪

Convolution 이란 무엇일까?

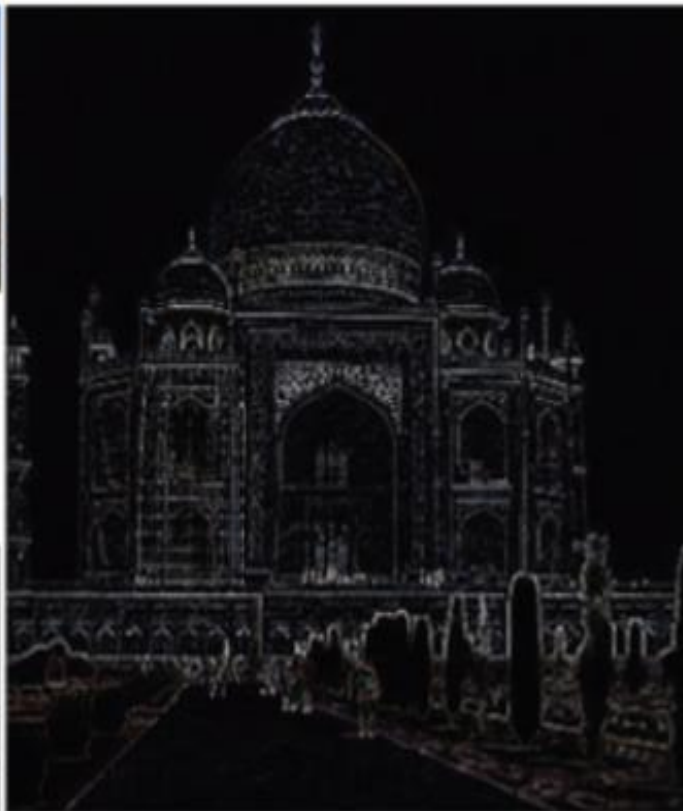
- 영상처리 분야에서 Convolution은 주로 필터(Filter) 연산에 사용되며, 영상으로부터 특정 Feature를 추출하기 위한 필터를 구할 때 사용
 - 그림에서 특징을 찾아내는 문제
- 기존에는 엔지니어의 Domain Knowledge를 통한 필터 개발
 - 필터 또한 학습을 통해 분류를 더 잘할 수 있을까?



원본이미지

	0	1	0	
	1	-4	1	
	0	1	0	

Kernel

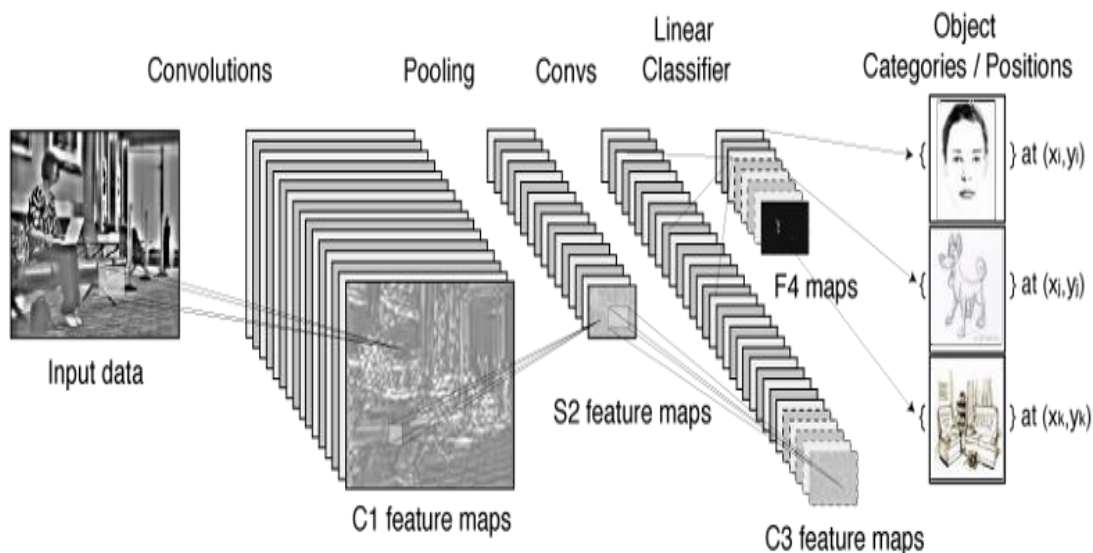


전체 구조

CNN

CNN(Convolution Neural Network, 합성곱 신경망)이란?

: CNN은 합성곱(Convolution) 연산을 사용하는 ANN(Artificial Neural Network)의 한 종류이다. Convolution을 사용하면 3차원 데이터의 공간적 정보를 유지한 채 다음 레이어로 보낼 수 있다. 대표적인 CNN으로는 LeNet(1998)과 AlexNet(2012)이 있으며, VGG, GoogleLeNet, ResNet 등은 층을 더 깊게 쌓은 CNN기반의 DNN(Deep Neural Network, 심층 신경망)이다.



Convolution Neural Network(CNN)

- Fully Connected Neural Network(DNN)의 경우, 입력 데이터가 독립이라는 가정 (1차원 데이터 한정)
 - 하지만 그림은 2차원이 아니다!
 - 사진 데이터를 1차원으로 바꾸는 과정에서 정보의 손실 발생
 - 정보 손실 없이 학습할 수는 없을까?

Feature extraction

Classification

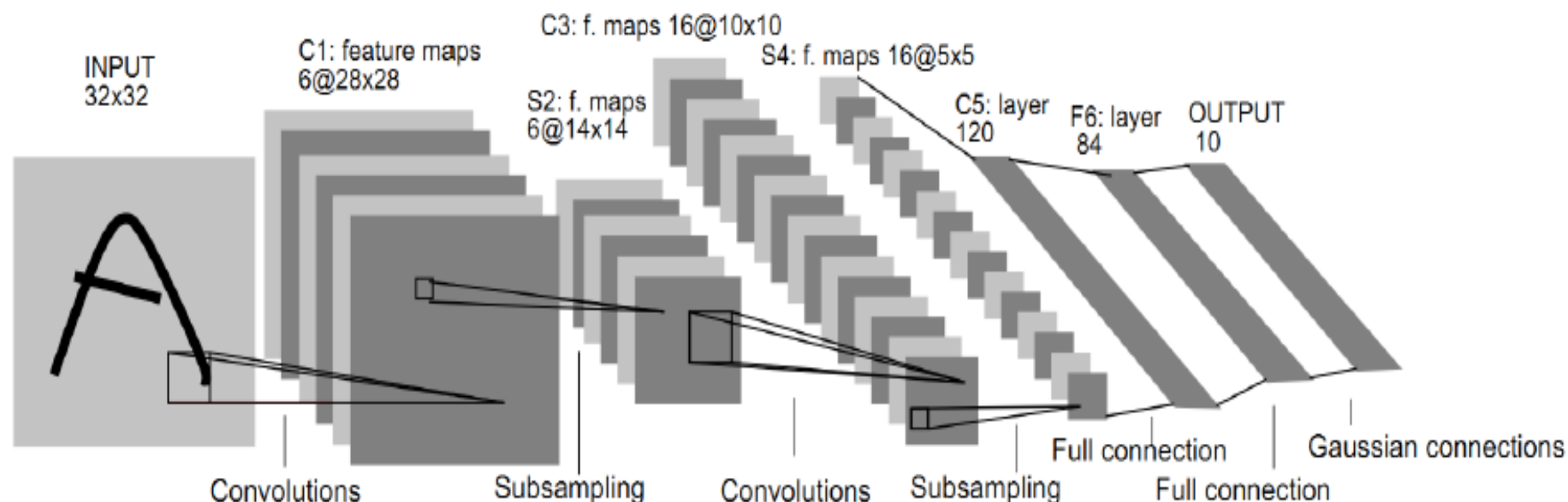


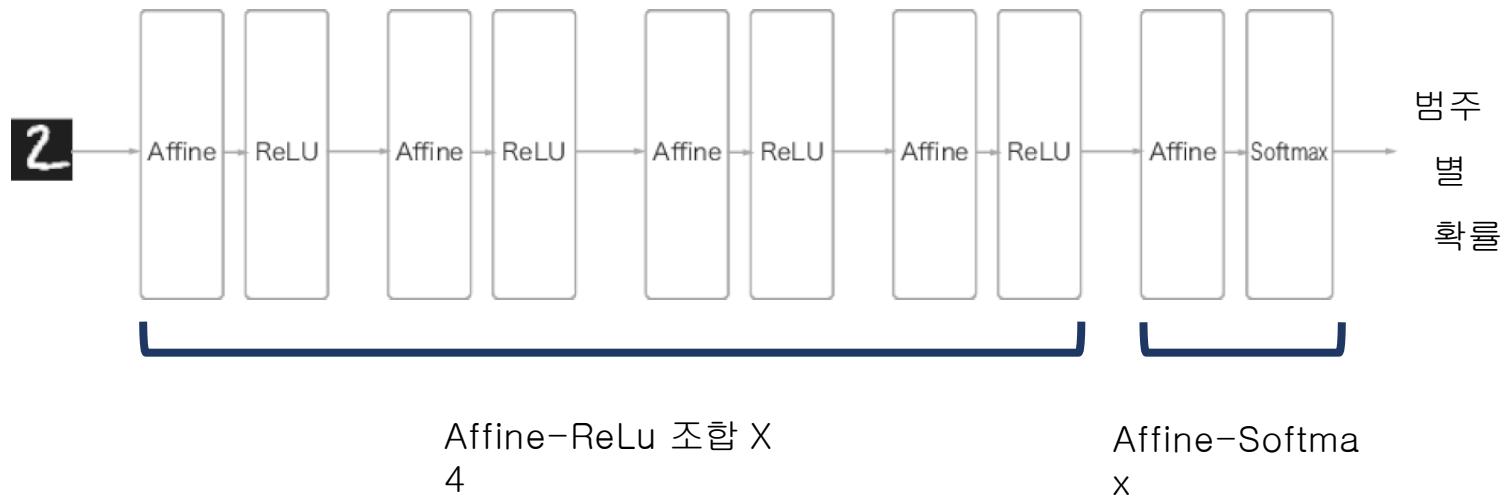
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

전체 구조

CNN

기존의 신경망(Artificial Neural Network) 구조

: 인접하는 계층의 모든 뉴런이 결합되어 있는 완전연결(fully-connected)로, Affine 계층으로 구성되어 있다.

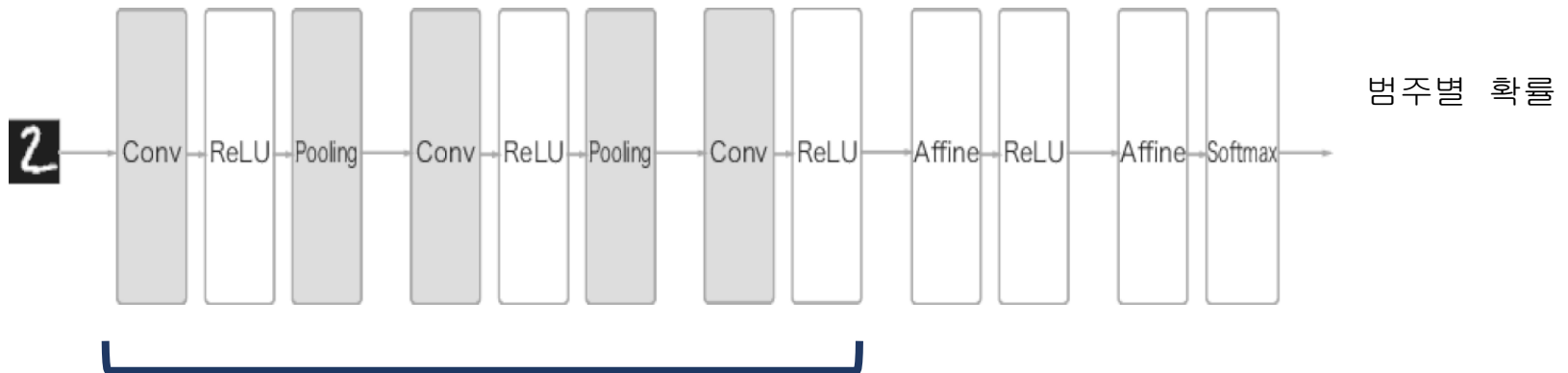


전체 구조

CNN

CNN 구조

: 신경망 구조에서 합성곱 계층(Conv), 풀링 계층(Pooling)이 추가된다.



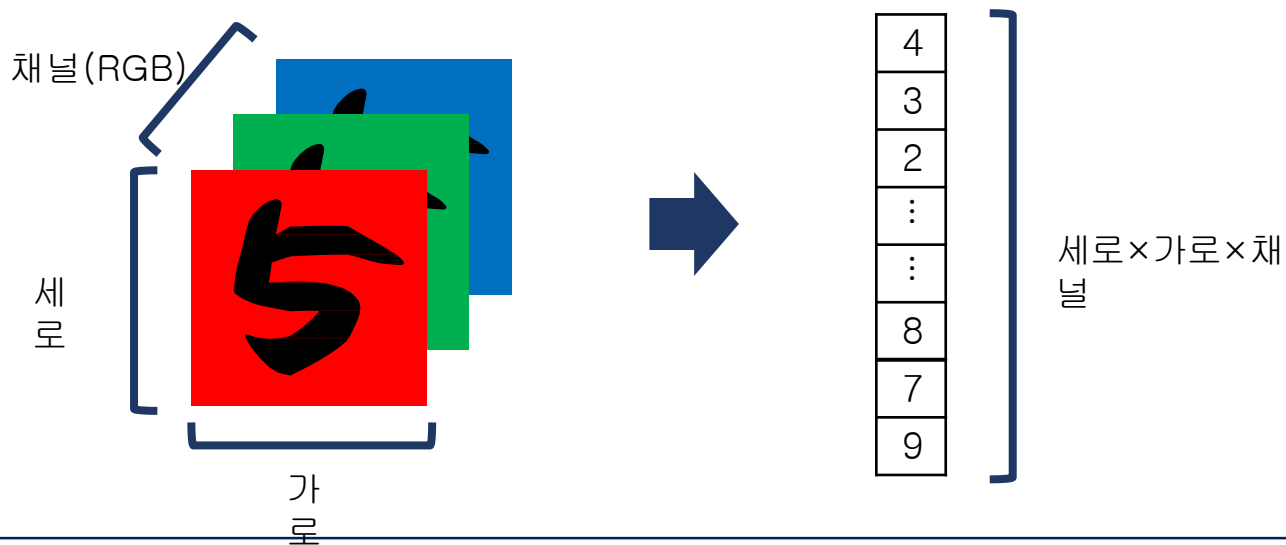
Conv - ReLu - (Pooling
)

합성곱 계층

완전연결 계층의 문제점

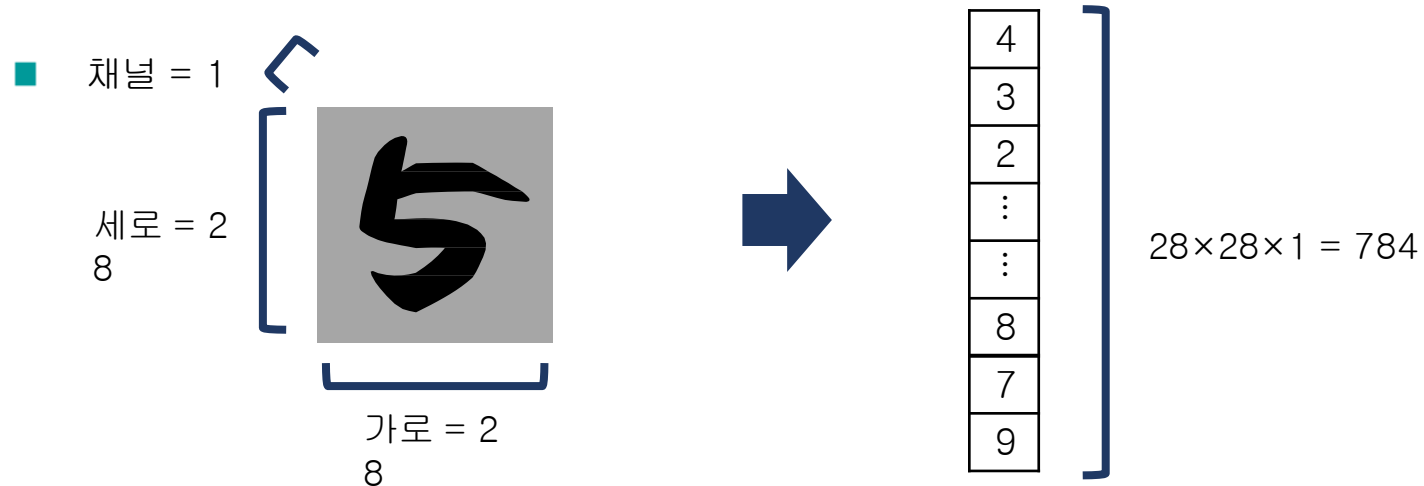
데이터 형상의 무시

: 세로·가로·채널(색상)로 구성된 3차원 데이터인 이미지가 완전연결 계층(fully-connected layer)에 입력될 때, 1차원 데이터로 변형되게 되면서 공간적 정보를 잃게 된다.



■ 데이터 형상의 무시

- : 예를 들어, 공간적으로 가까운 픽셀은 값이 비슷하거나, RGB의 각 채널은 서로 밀접하게 관련되어 있거나, 거리가 먼 픽셀끼리는 별 연관이 없는 등, 3차원 속 의미를 갖는 본질적인 패턴이 무시된다.

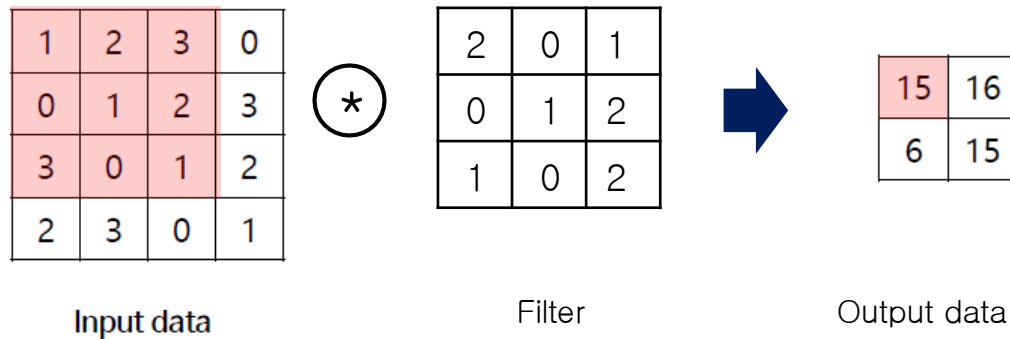









합성곱 계층

합성곱 연산

합성곱(Convolution)

: 특정 (높이, 너비)를 갖은 필터(Filter, Kernel)를 일정 간격(Stride)으로 이동해가며 입력 데이터에 적용 (원소 곱 후, 총합 : 단일 곱셈-누산)



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

4.3.1 컨볼루션층

■ 컨볼루션 연산

- 컨볼루션은 해당하는 요소끼리 곱하고 결과를 모두 더하는 선형 연산
- 식 (4.10)과 식 (4.11)에서 u 는 커널, z 는 입력, s 는 출력(특징 맵)

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

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

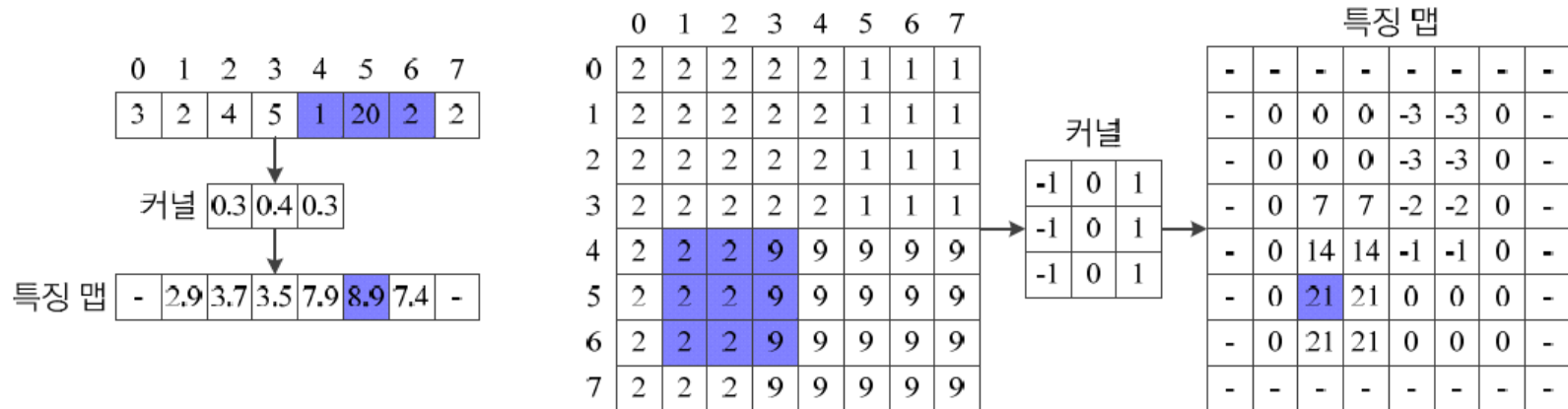
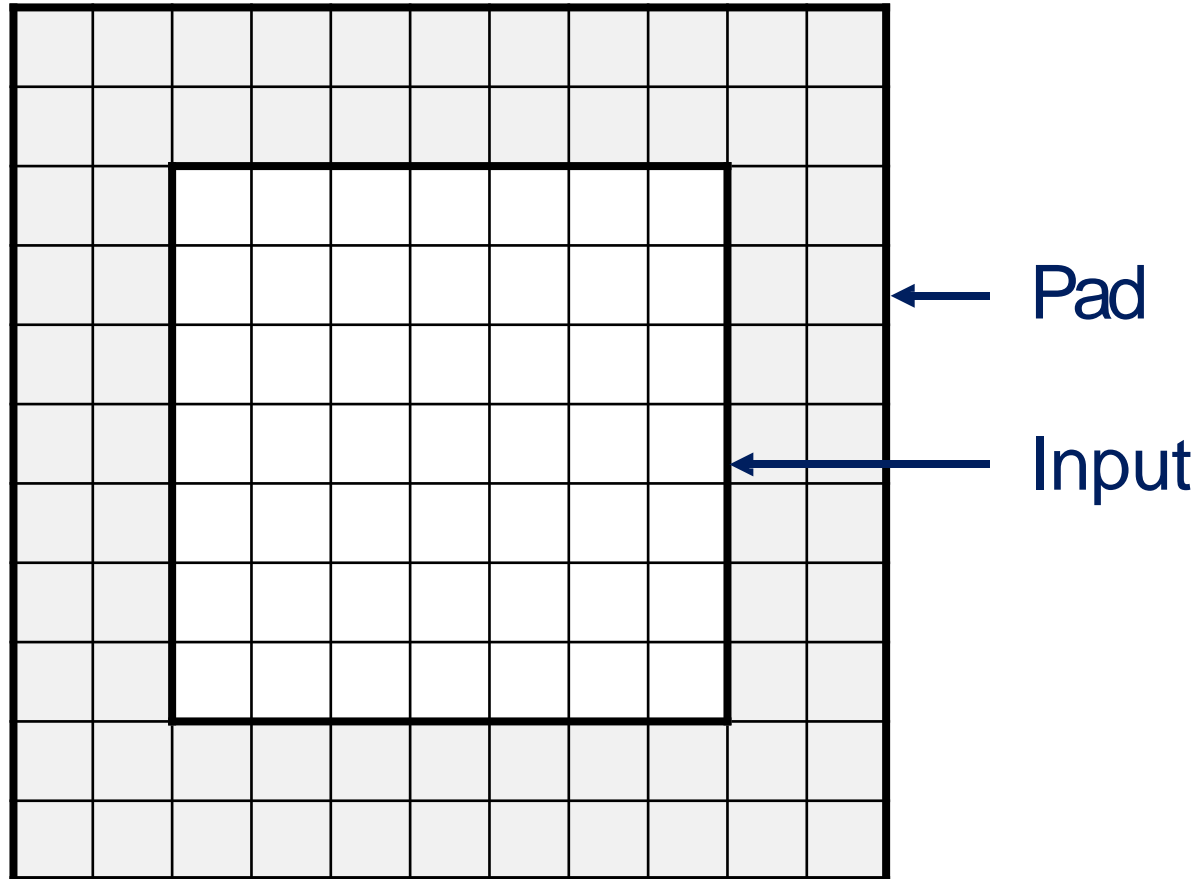


그림 4-6 컨볼루션 연산

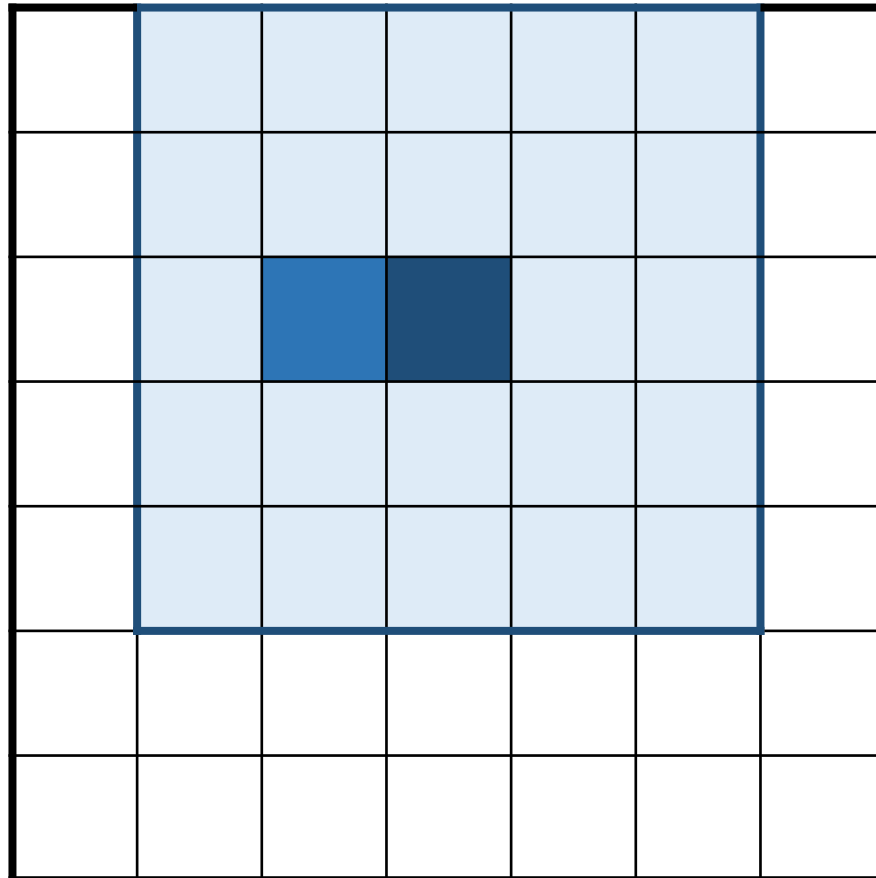
Convolution

- Padding



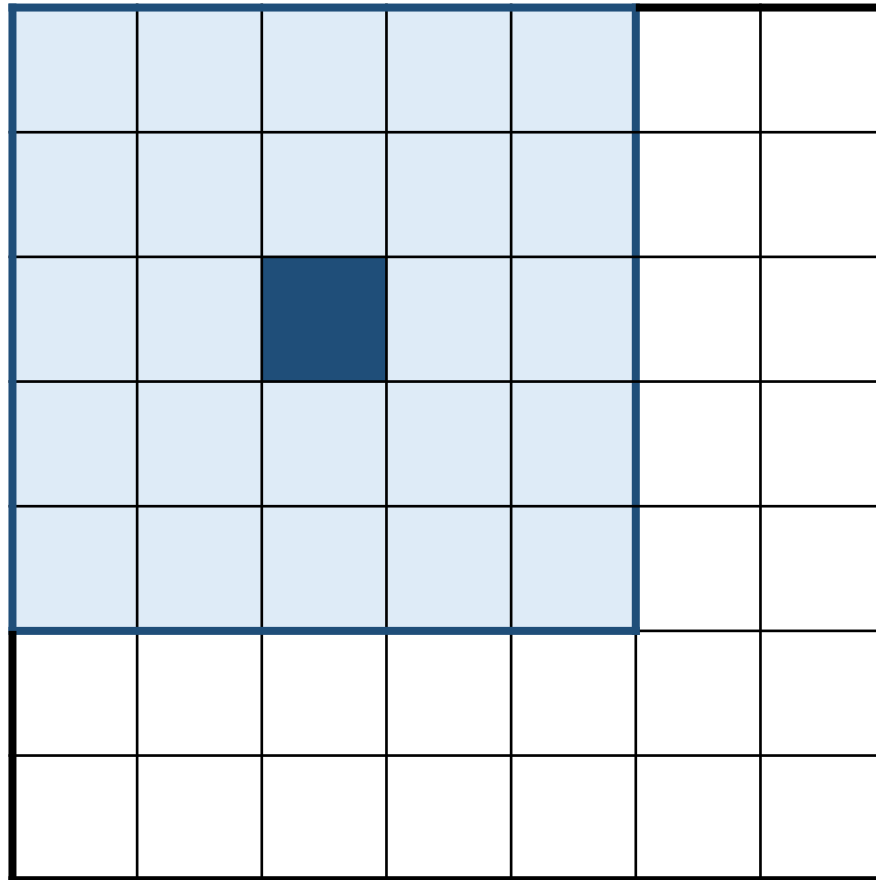
Convolution

- Padding



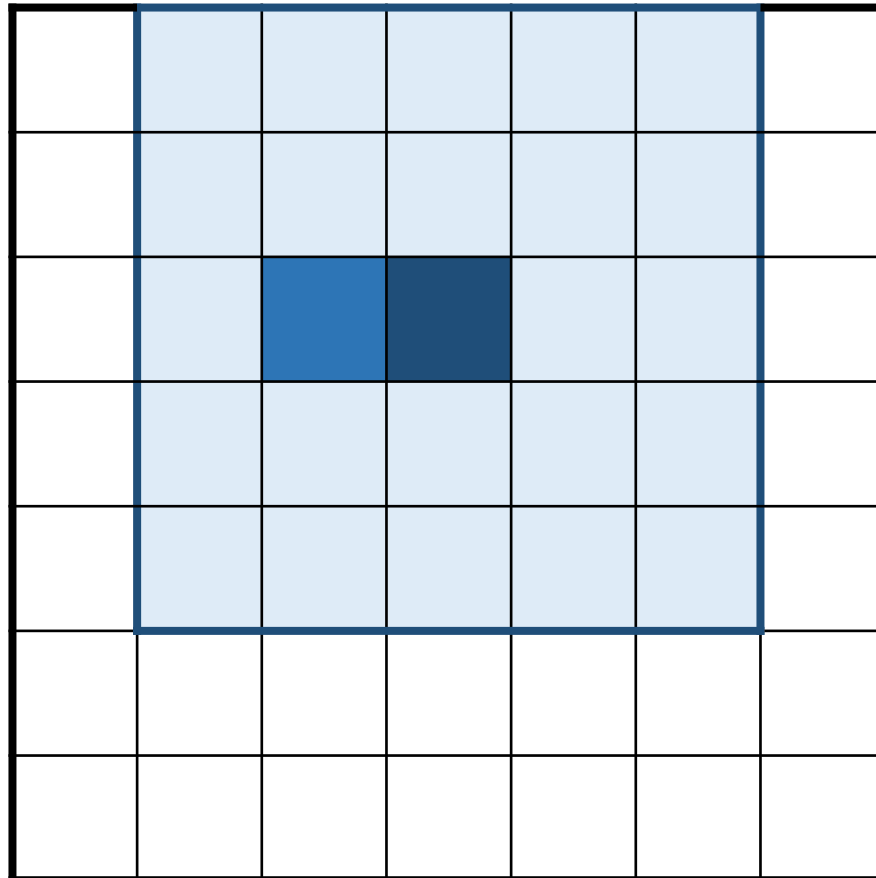
Convolution

- Padding



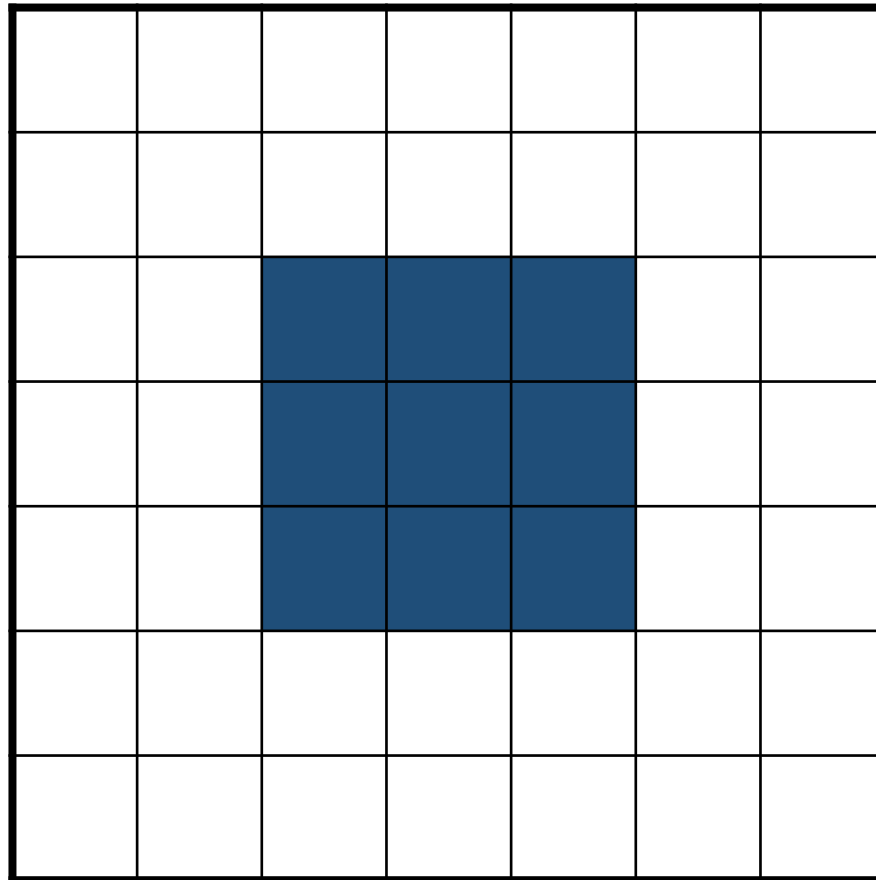
Convolution

- Padding



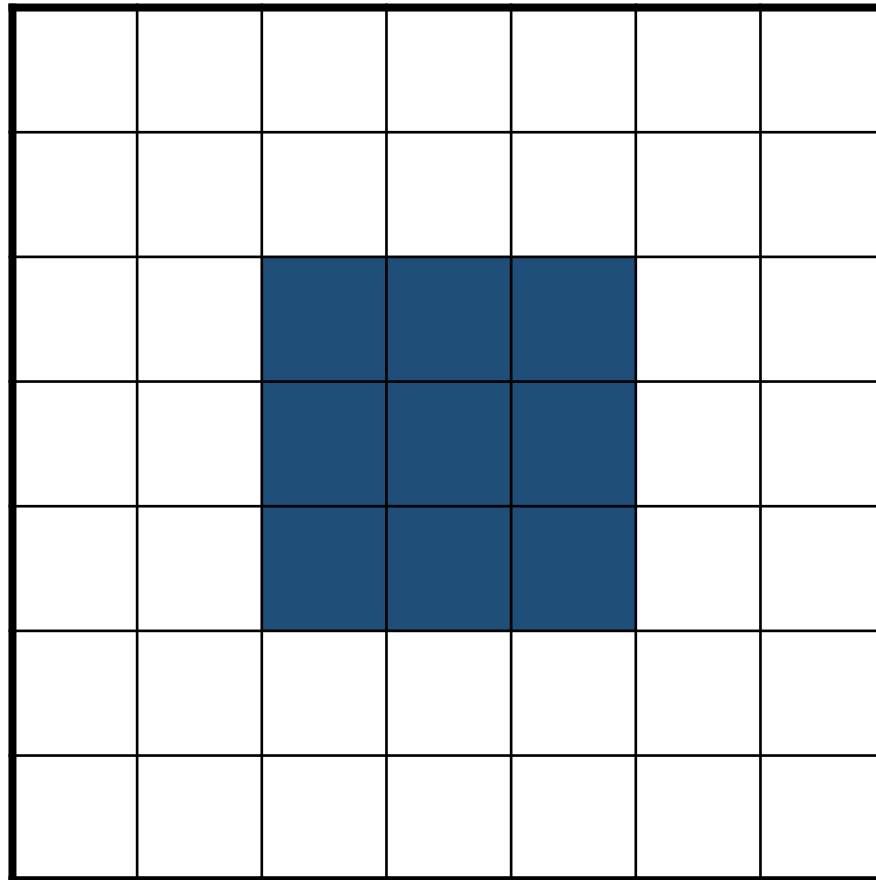
Convolution

- Padding



Convolution

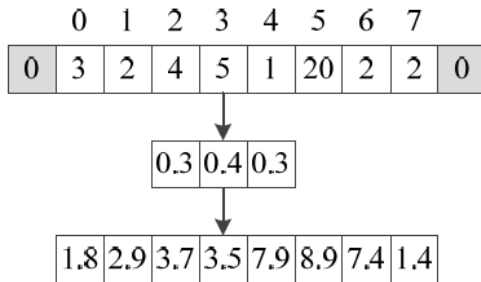
- Padding



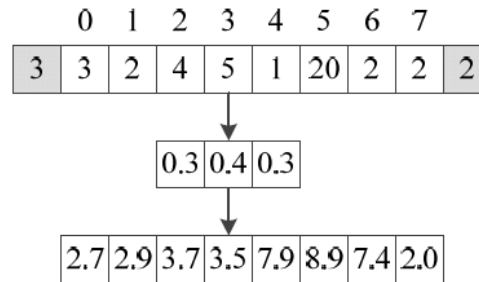
4.3.1 컨볼루션층

■ 덧대기

- 가장자리에서 영상의 크기가 줄어드는 효과 방지



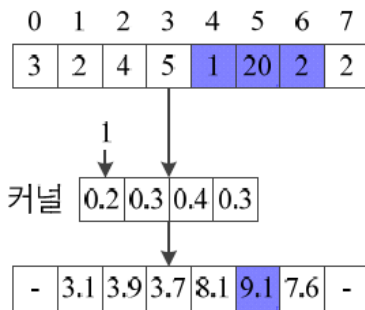
(a) 0 덧대기



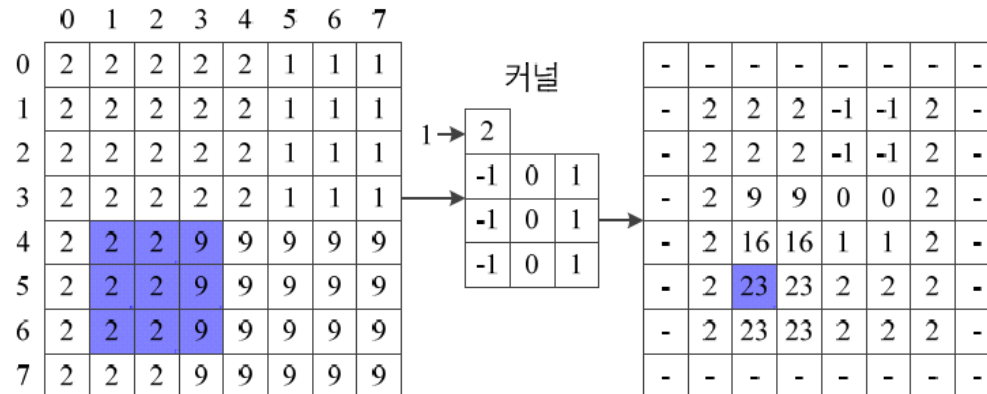
(b) 복사 덧대기

그림 4-7 덧대기(회색 노드가 덧댄 노드)

■ 바이어스 추가



(a) 1차원 컨볼루션



(b) 2차원 컨볼루션

그림 4-8 바이어스

4.3.1 컨볼루션층

■ 가중치 공유(묶인 가중치)

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

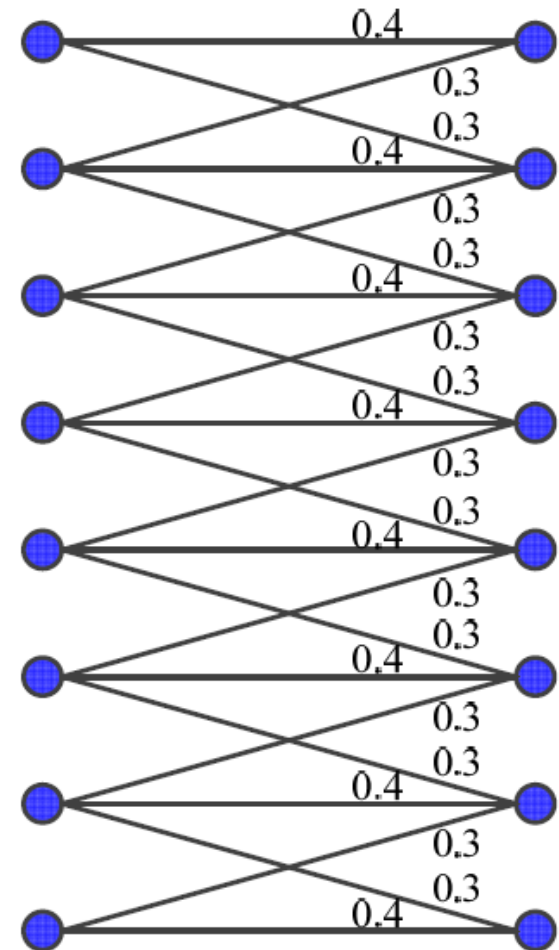


그림 4-9 CNN의 가중치 공유

4.3.1 컨볼루션층

■ 다중 특징 맵 추출

- 커널의 값에 따라 커널이 추출하는 특징이 달라짐
- 예) $\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$ 수직방향, $\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ 수평방향 에지 추출
- 따라서 하나의 커널만 사용하면 너무 빈약한 특징이 추출됨
- [그림 4-10]은 3개 커널을 사용하여 3개 특징 맵을 추출하는 상황
- 실제로는 수십~수백 개의 커널을 사용

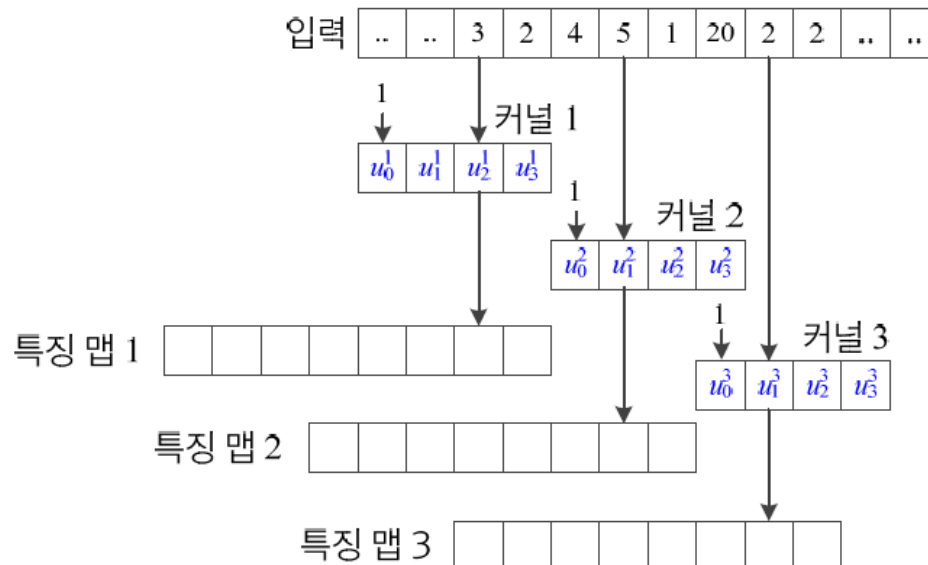


그림 4-10 다중 특징 맵 추출

4.3.1 컨볼루션층

■ 특징 학습

- 커널을 사람이 설계하지 않고, 학습으로 알아냄
 - u_i^k 는 k 번째 커널의 i 번째 매개변수
- 예) 2차원 영상이 7×7 커널을 64개 사용한다면, 학습은 $(7 \times 7 + 1) \times 64 = 3200$ 개의 매개변수를 찾아내야 함
- DMLP와 마찬가지로 오류 역전파로 커널을 학습함

4.3.1 컨볼루션층

■ 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

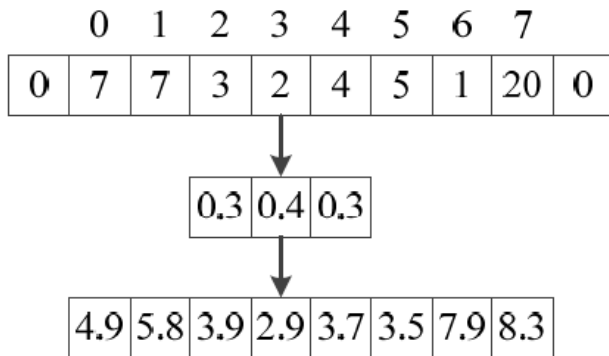


그림 4-11 CNN의 이동에 동변한 특성

■ 병렬분산 구조

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

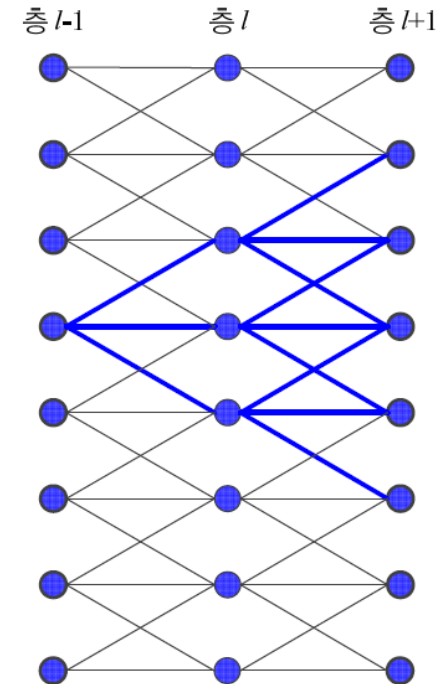
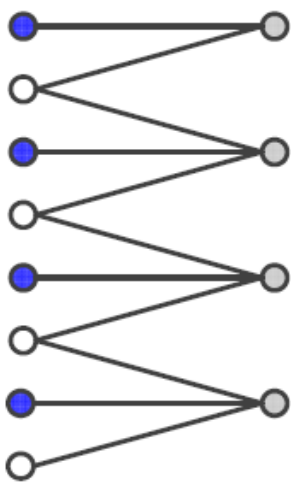


그림 4-12 CNN의 병렬 분산 구조

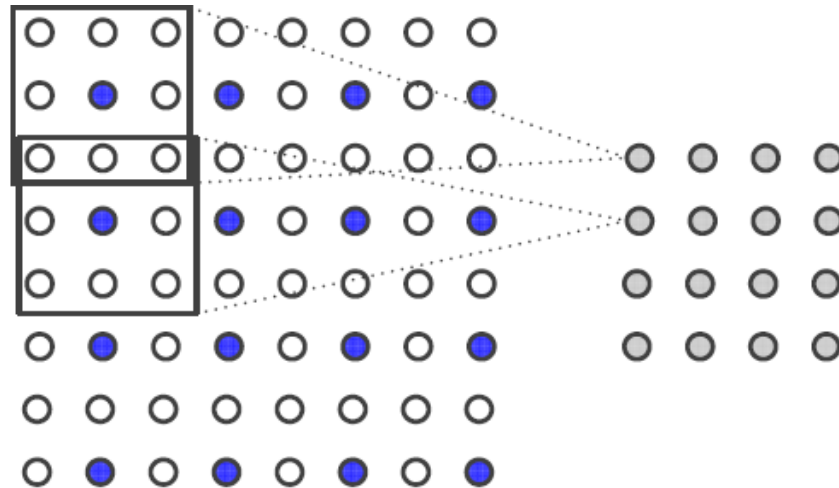
4.3.1 컨볼루션층

■ 큰 보폭에 의한 다운샘플링

- 지금까지는 모든 화소에 커널 적용 → 보폭을 1로 설정한 셈
- [그림 4-13]은 보폭이 2인 상황
- 일반적으로 보폭이 k 이면, k 개 마다 하나씩 샘플링하여 커널 적용 → 2차원 영상의 경우 특징 맵이 $1/k^2$ 로 작아짐



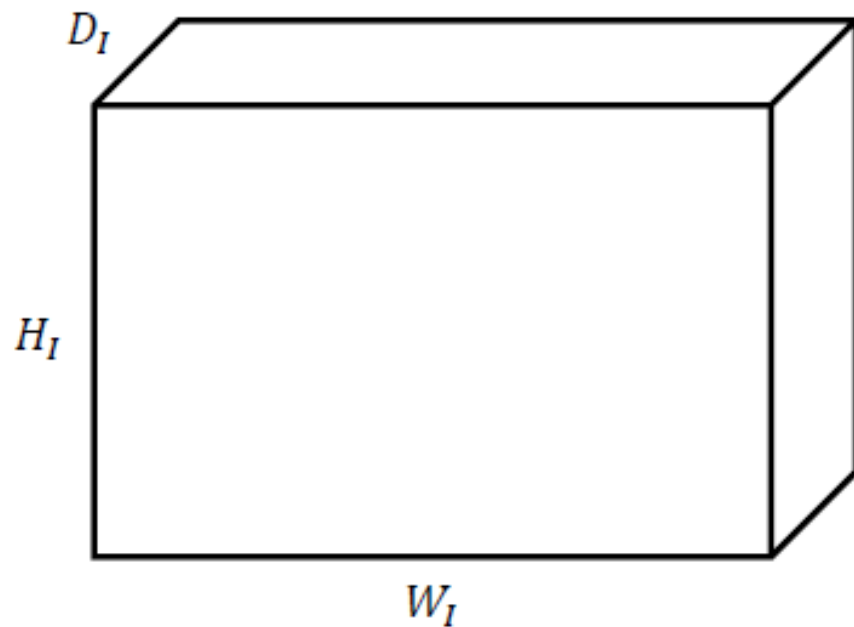
(a) 1차원 데이터(예: 음성)



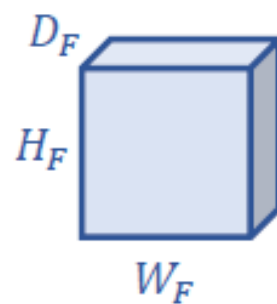
(b) 2차원 데이터(예: 영상)

그림 4-13 보폭이 2인 컨볼루션 연산

Pooling

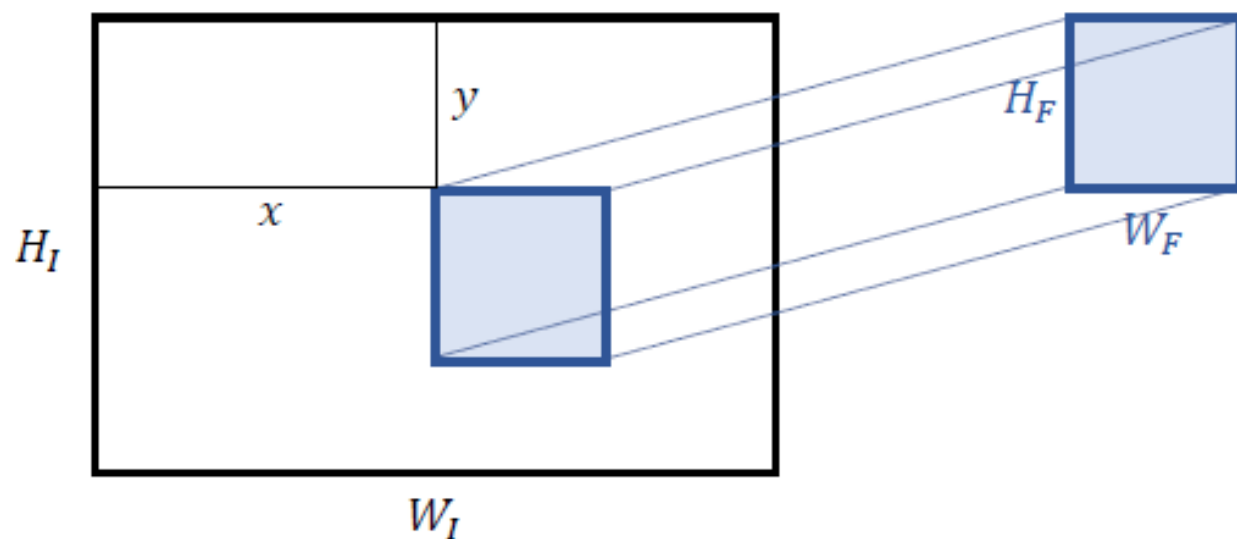


Input I : $[W_I, H_I, D_I] \in \mathbb{R}^3$



Filter F : $[W_F, H_F, D_F] \in \mathbb{R}^3$

Pooling

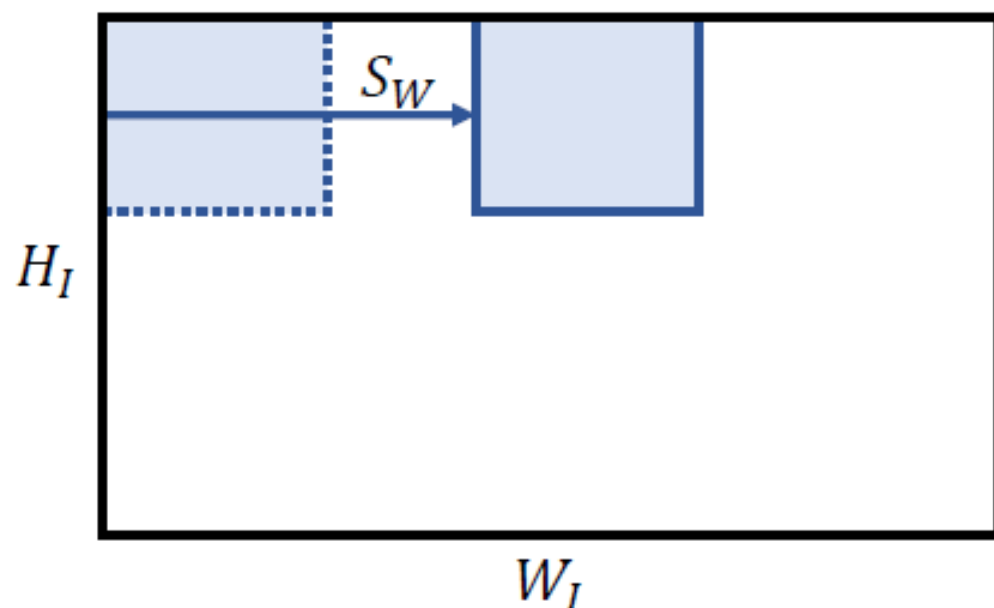


$$\text{Max: } O_{x,y} = \max_{i,j} I_{x+i,y+j}$$

$$\text{Average: } O_{x,y} = \frac{1}{W_F \times H_F} \sum_i^{W_F} \sum_j^{H_F} I_{x+i,y+j}$$

Convolution Arithmetic

- Assumption: use zero-padding of size P



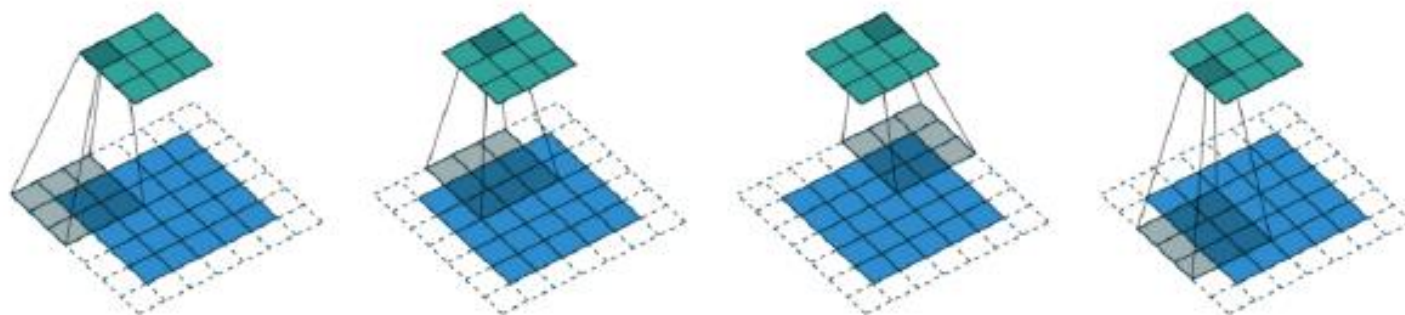
Output O : $[W_O, H_O, D_O] \in \mathbb{R}^3$

$$W_O = \left\lfloor \frac{W_I - W_F + 2P}{S_W} \right\rfloor + 1$$

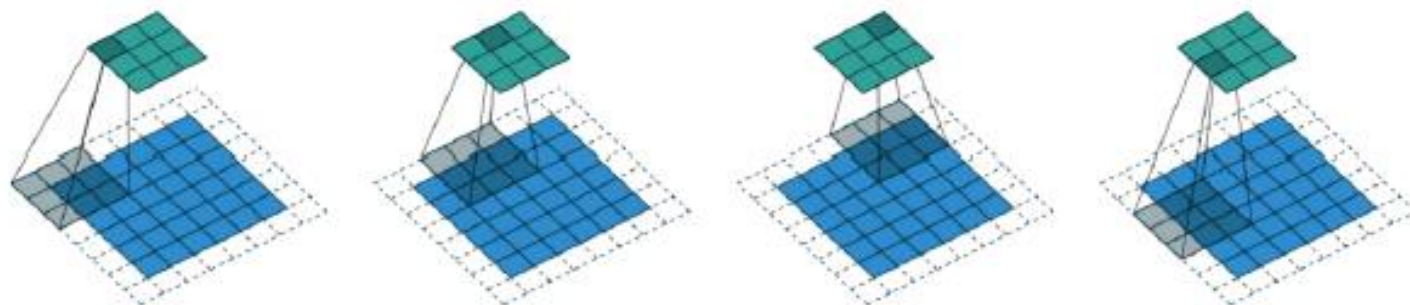
Convolution Arithmetic

- Valid convolution: $W_F = 3, P = 1, S_W = 2$

$W_I = 5$



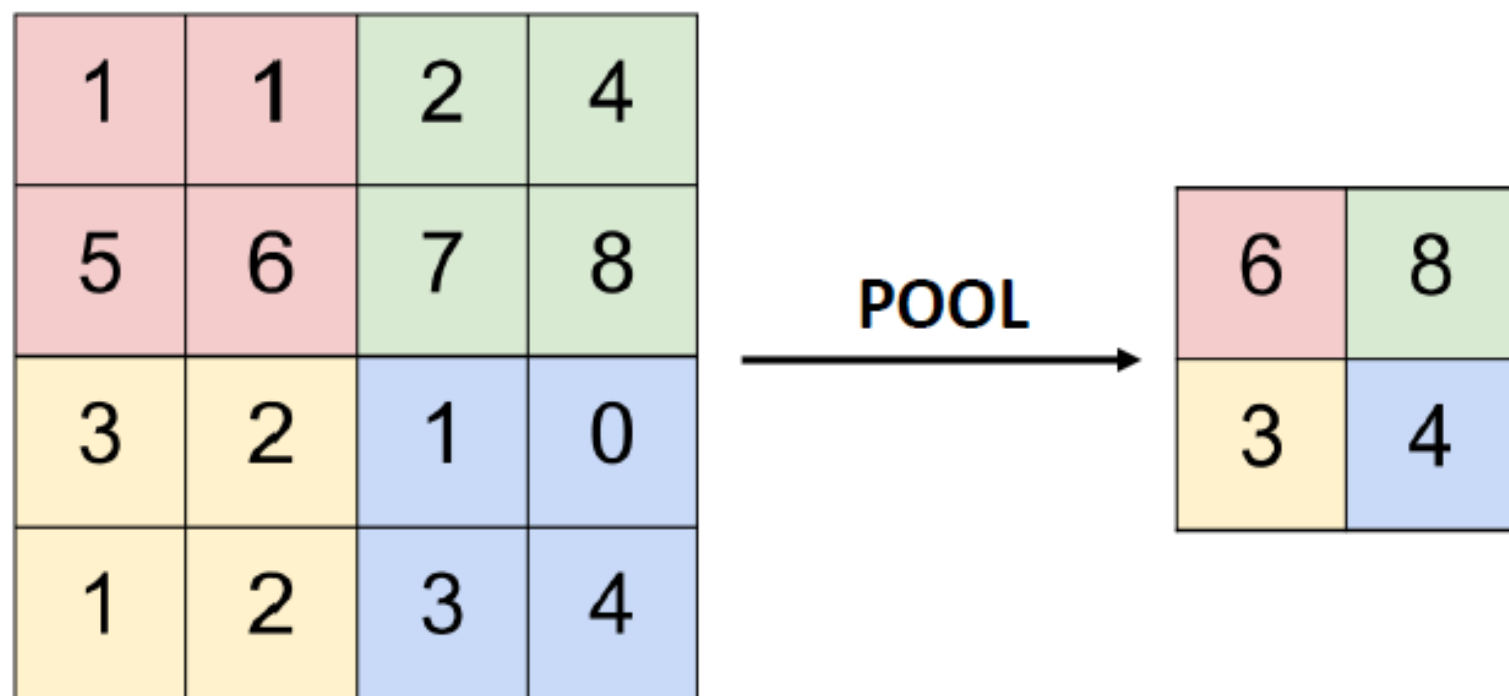
$W_I = 6$



- <http://aikorea.org/cs231n/convolutional-networks/>

Pooling

- e.g. **max pooling** with **2×2 filters** and stride **2**



4.3.1 컨볼루션층

■ 텐서에 적용

- 3차원 이상의 구조에도 적용 가능
 - 예) RGB 컬러 영상은 $3*m*n$ 의 3차원 텐서 ([그림 4-14])

3*3*3 입력 영상

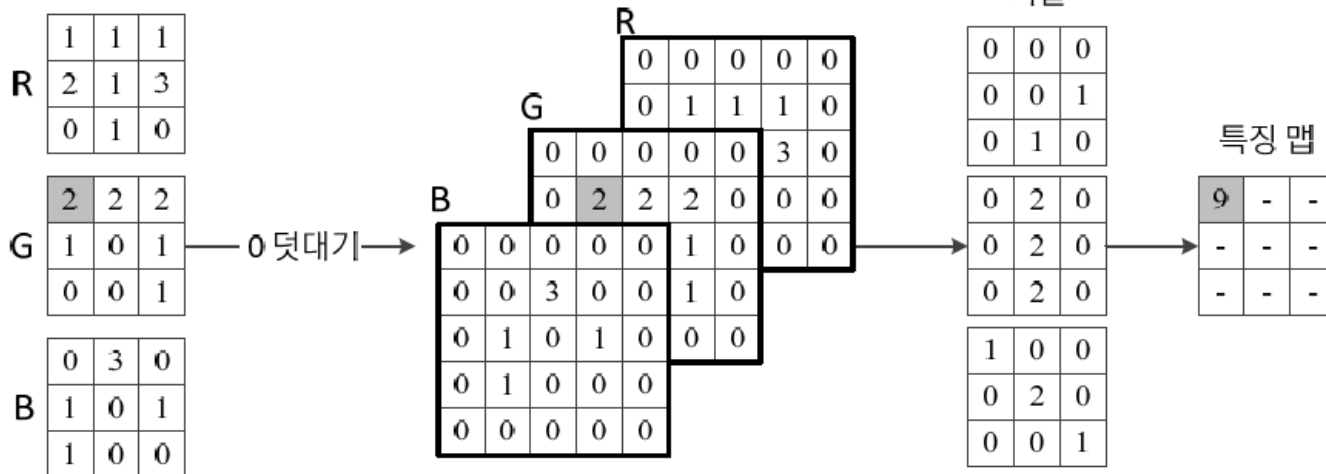


그림 4-14 텐서의 컨볼루션 연산(0 덧대기 적용)

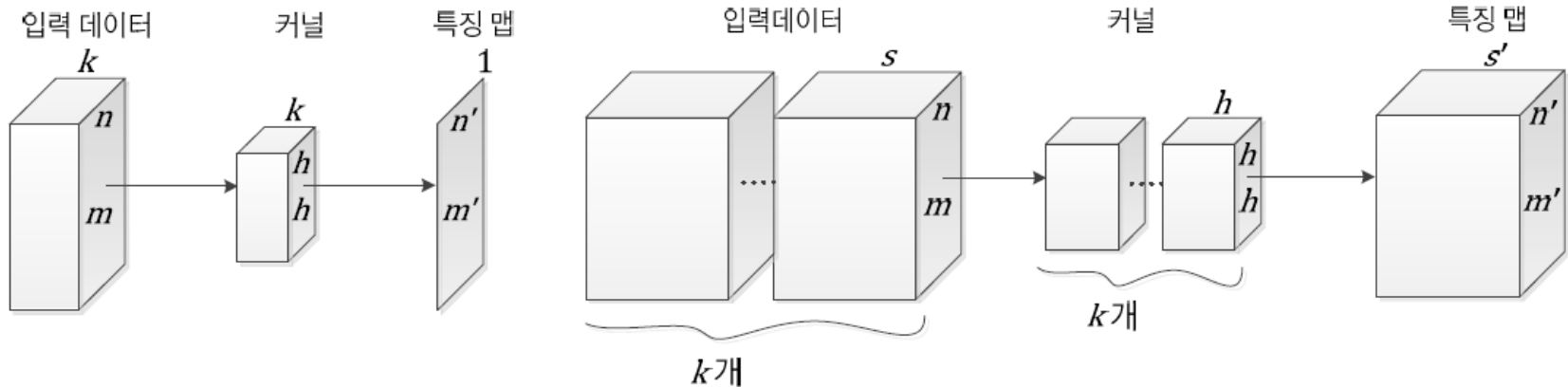
- 특징 맵의 회색 노드의 계산 예시

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 2 & 1 \end{pmatrix}}_R \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 1 & 0 \end{pmatrix}}_G \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 1 & 0 \end{pmatrix}}_B \odot \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}}_{c_1} \underbrace{\begin{pmatrix} 0 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 0 \end{pmatrix}}_{c_2} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{c_3} = 9$$

4.3.1 컨볼루션층

■ 3차원 구조의 데이터에 적용

- 채널이 k 개인 3차원 격자 구조([그림 4-15(a)])
 - [그림 4-14]를 블록 형태로 다시 그린 것 (예, RGB 컬러 영상)



(a) 다중채널 데이터(예: RGB 컬러 영상)

(b) 3차원 데이터(예: 동영상, MRI 뇌 영상)

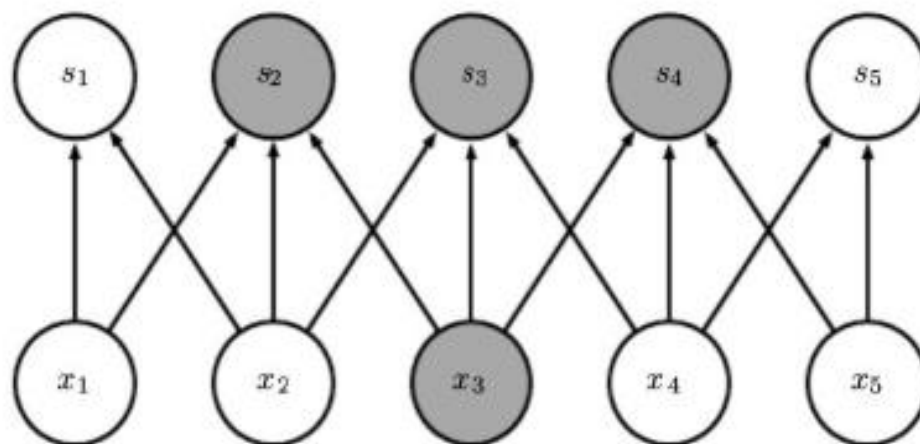
그림 4-15 텐서의 컨볼루션 연산(직육면체로 표현하기)

- 4차원 텐서로 표현하는 데이터 ([그림 4-15(b)])
 - 예) 컬러 동영상($3 \times s \times m \times n$), MRI 뇌영상 ($1 \times s \times m \times n$)
 - $k \times h \times h \times h$ 커널을 $s \times m \times n$ 공간을 이동하면서 적용

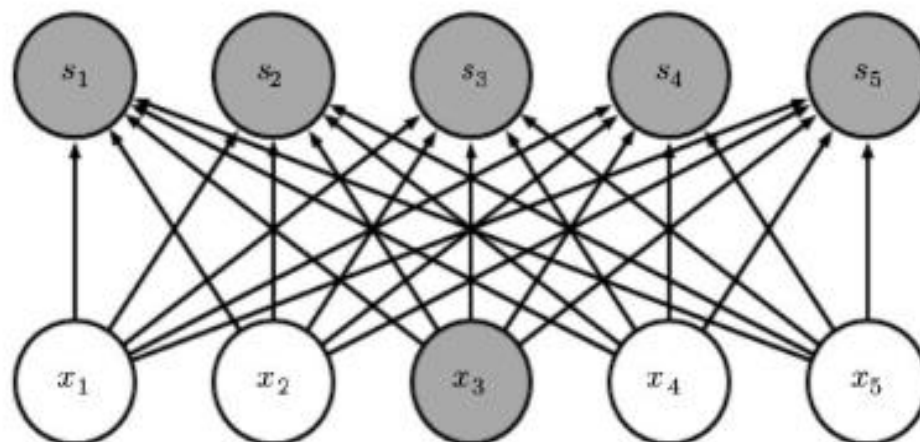
Key Idea: Sparse Connectivity

- Sparse connectivity, viewed from below

convolution



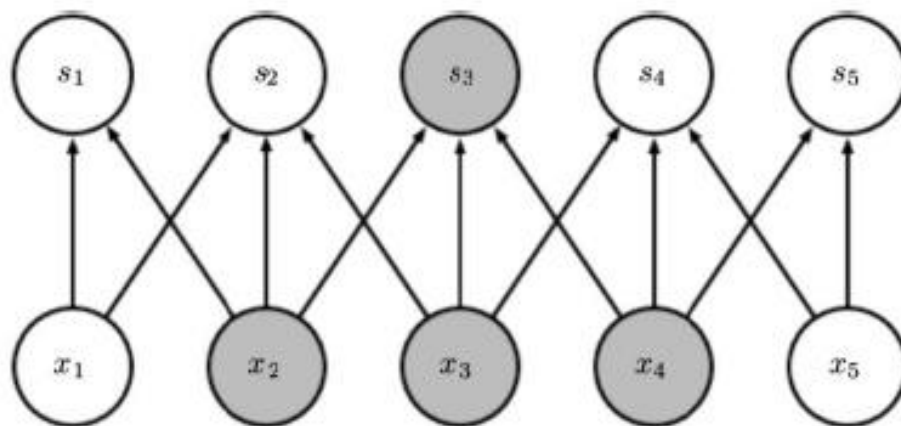
FC



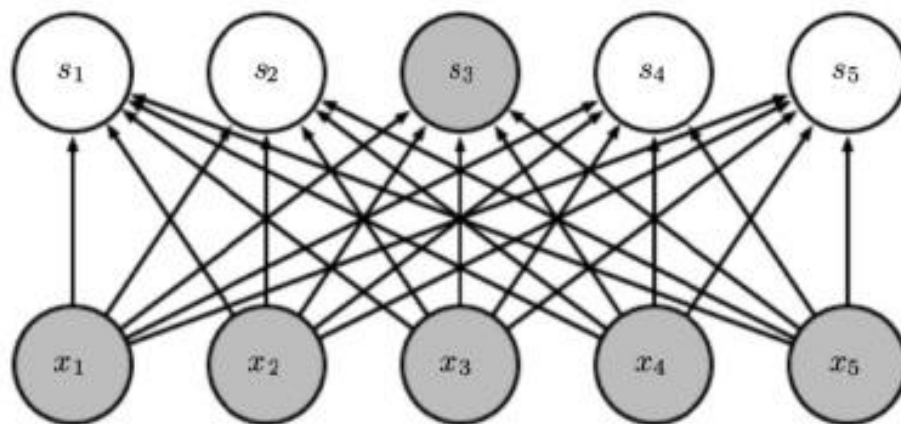
Key Idea: Sparse Connectivity

- Sparse connectivity, viewed from above

convolution



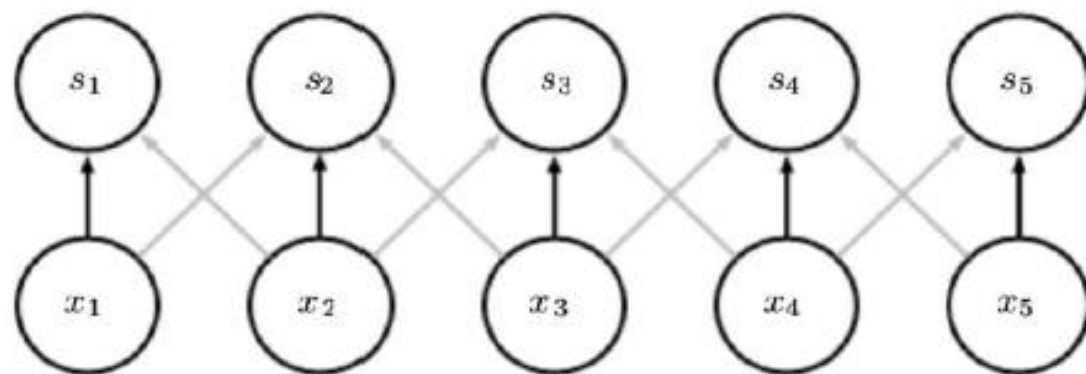
FC



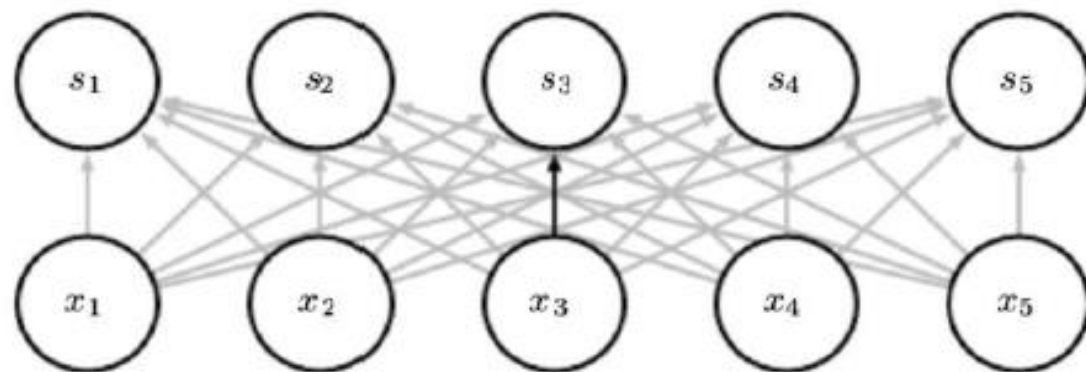
Key Idea: Parameter Sharing

- $O(k)$ parameters, but running time: $O(kn)$

convolution



FC



4.3.2 풀링층

■ 풀링 연산

- [그림 4-16]은 최대 풀링 예
- 최대 풀링, 평균 풀링, 가중치 평균 풀링 등
- 보폭을 크게 하면 다운샘플링 효과

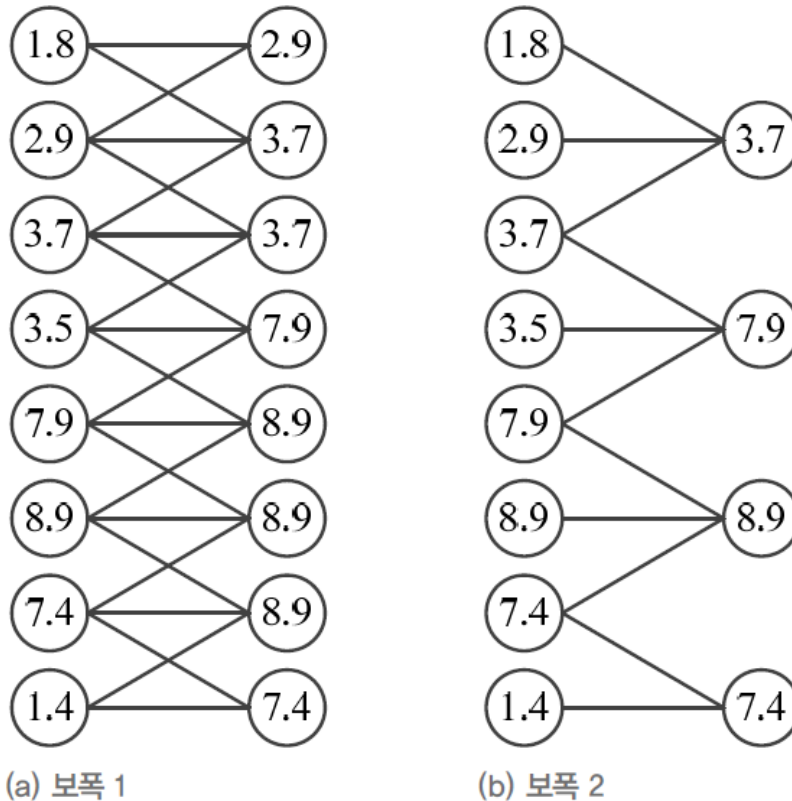
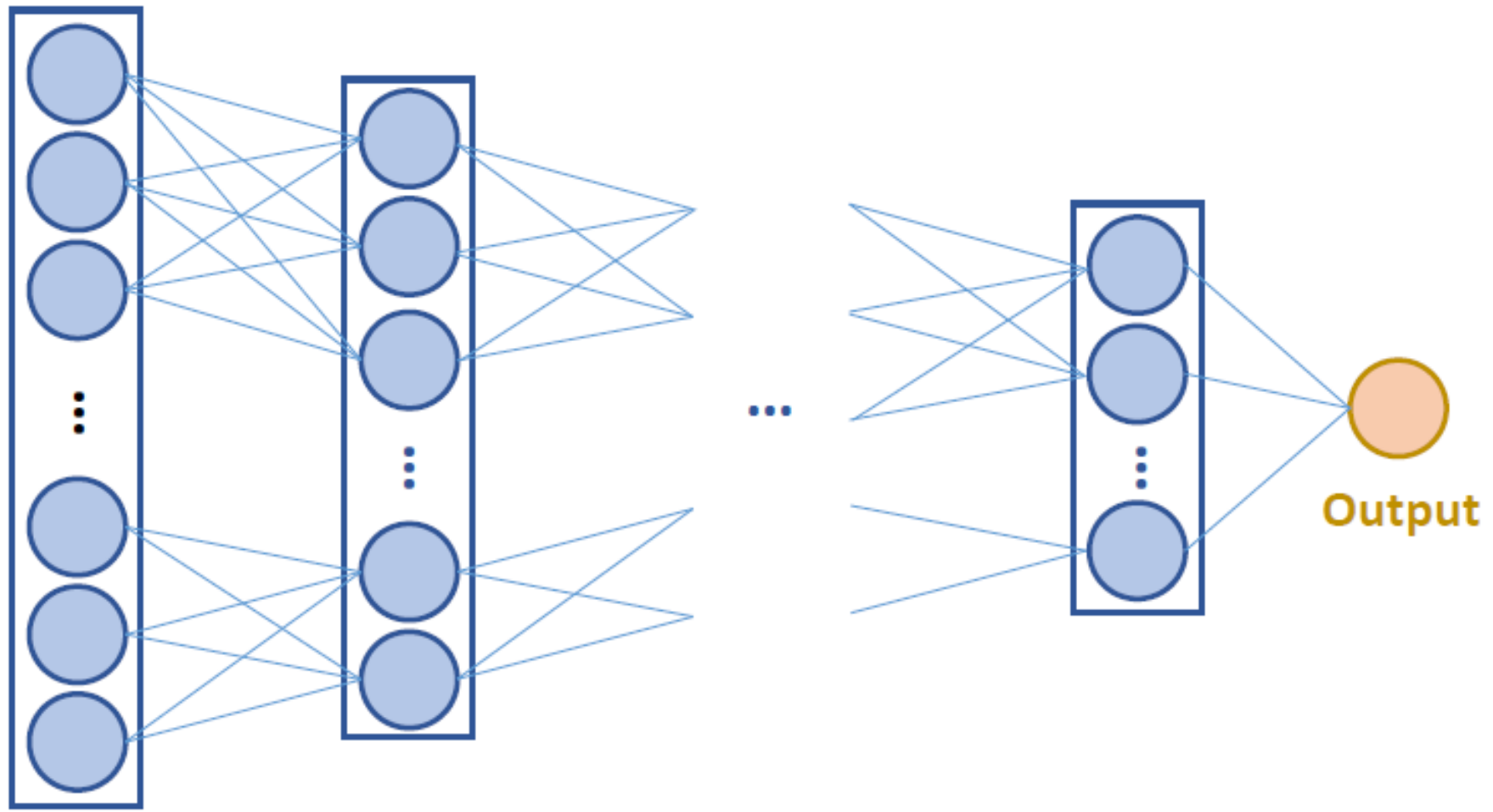


그림 4-16 최대 풀링

Fully Connected



4.3.2 풀링층

■ 풀링 연산의 특성

- 풀링은 상세 내용에서 요약통계를 추출함
- 매개변수가 없음
- 특징 맵의 수를 그대로 유지함
- 작은 이동에 둔감 → 물체 인식이나 영상 검색 등에 효과적임

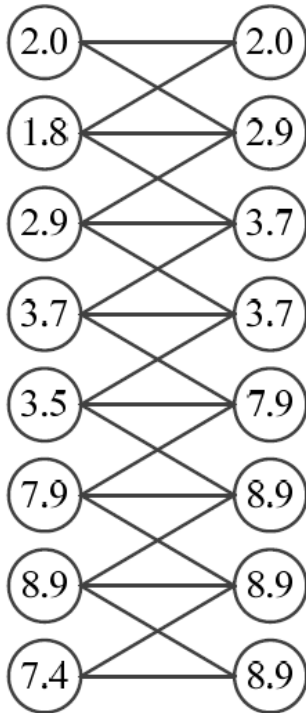


그림 4-17 작은 이동에 둔감한 최대 풀링

4.3.3 전체 구조

■ 빌딩블록

- CNN은 빌딩블록을 이어 붙여 깊은 구조를 만듦
- [그림 4-18]은 전형적인 빌딩블록: 컨볼루션층 → 활성화함수(주로 ReLU 사용) → 풀링층
- 다중 커널을 사용하여 다중 특징 맵을 추출함

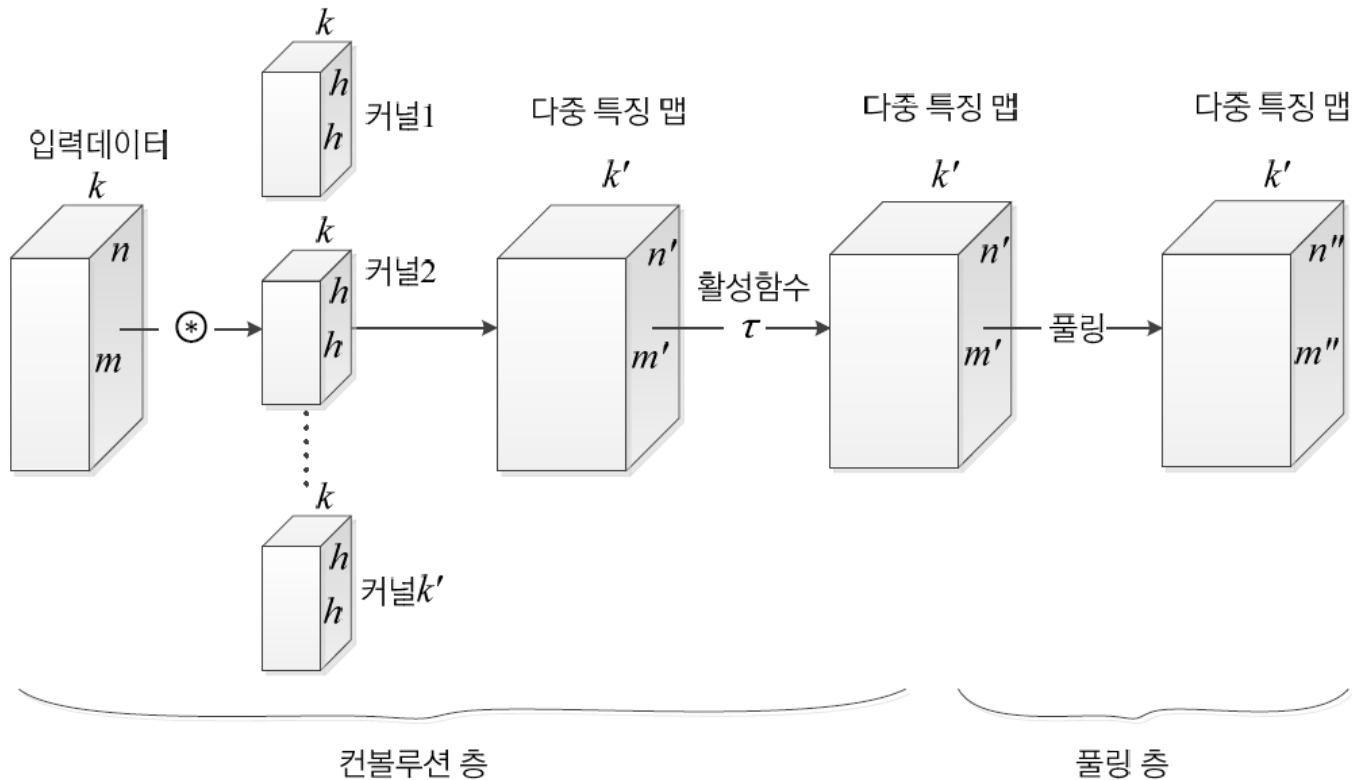


그림 4-18 CNN의 빌딩블록

4.3.3 전체 구조

■ 초창기 CNN 사례로서 LeNet-5

- 특징 추출: C-P-C-P-C의 다섯 층을 통해 28*28 명암 맵을 120차원의 특징 벡터로 변환
- 분류: 은닉층이 하나인 MLP
- CNN의 첫 번째 성공사례: 필기 숫자 인식기 만들어 수표 인식 자동화 시스템 구현

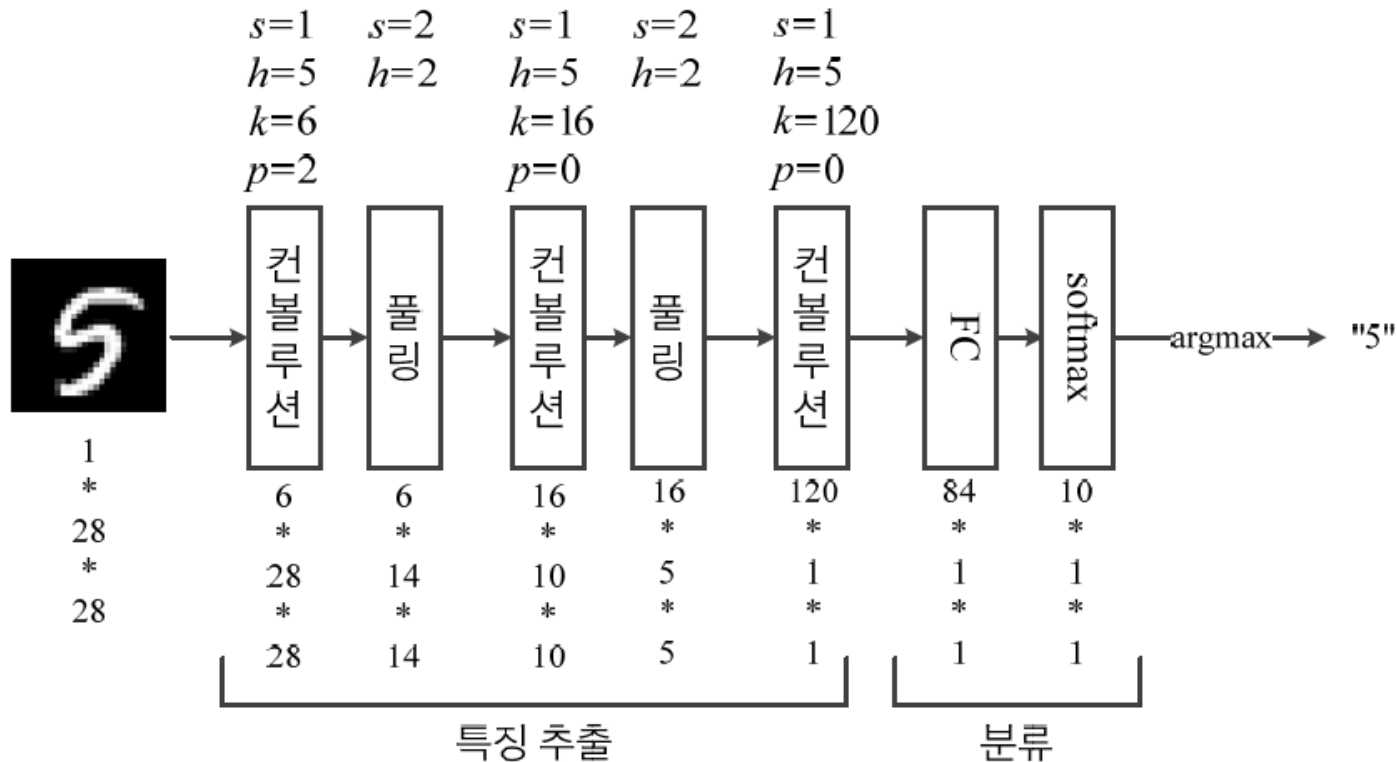


그림 4-19 LeNet-5 구조

4.3.3 전체 구조

■ 가변 크기의 데이터 다루기

- 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

4.4 컨볼루션 신경망 사례연구

- 4.4.1 AlexNet
- 4.4.2 VGGNet
- 4.4.3 GoogLeNet
- 4.4.4 ResNet

4.4 컨볼루션 신경망 사례연구

■ 자연영상 분류라는 도전적 문제

- ImageNet 데이터베이스
 - 2만 부류에 대해 부류별로 500~1000장의 영상을 인터넷에서 수집하여 구축하고 공개
- ILSVRC 대회 (CVPR 학술회에서 개최)
 - 1000부류에 대해 분류, 검출, 위치 지정 문제: 1순위와 5순위 오류율로 대결
 - 120만 장의 훈련집합, 5만 장의 검증집합, 15만 장의 테스트집합
 - 우승: AlexNet(2012) → Clarifit(2013) → GoogLeNet&VGGNet(2014) → ResNet(2015)
- 우승한 CNN은 프로그램과 가중치를 공개함으로써 널리 사용되는 표준 신경망이 됨



(a) 'swing' 부류



(b) 'Great white shark' 부류

4.4.1 AlexNet

■ 구조

- 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

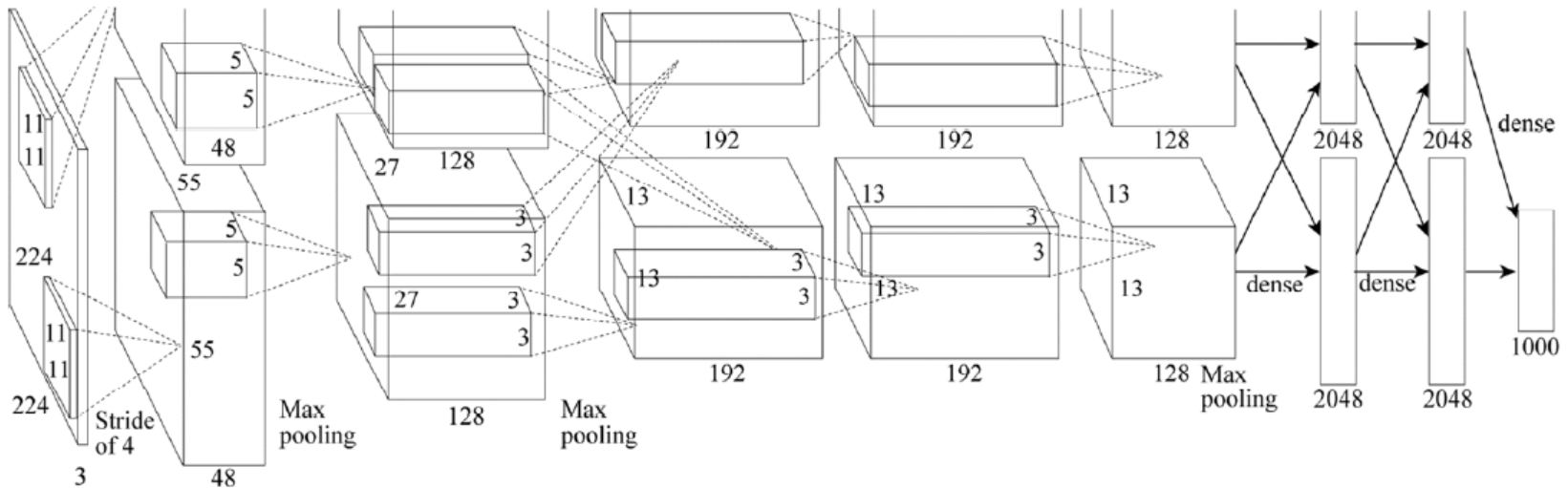


그림 4-21 AlexNet 구조[Krizhevsky2012]

4.4.1 AlexNet

■ AlexNet이 학습에 성공한 요인

- 외부 요인
 - ImageNet이라는 대용량 데이터베이스
 - GPU를 사용한 병렬처리
- 내부 요인
 - 활성화함수로 ReLU 사용
 - 지역 반응 정규화 기법 적용
 - 과잉적합 방지하는 여러 규제 기법 적용
 - 데이터 확대(잘라내기과 반전으로 2048배로 확대)
 - 드롭아웃 등

■ 테스트 단계에서 앙상블 적용

- [그림 5-26]과 [그림 12-5]
- 2~3%만큼 오류율 감소 효과

4.4.2 VGGNet

■ 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]

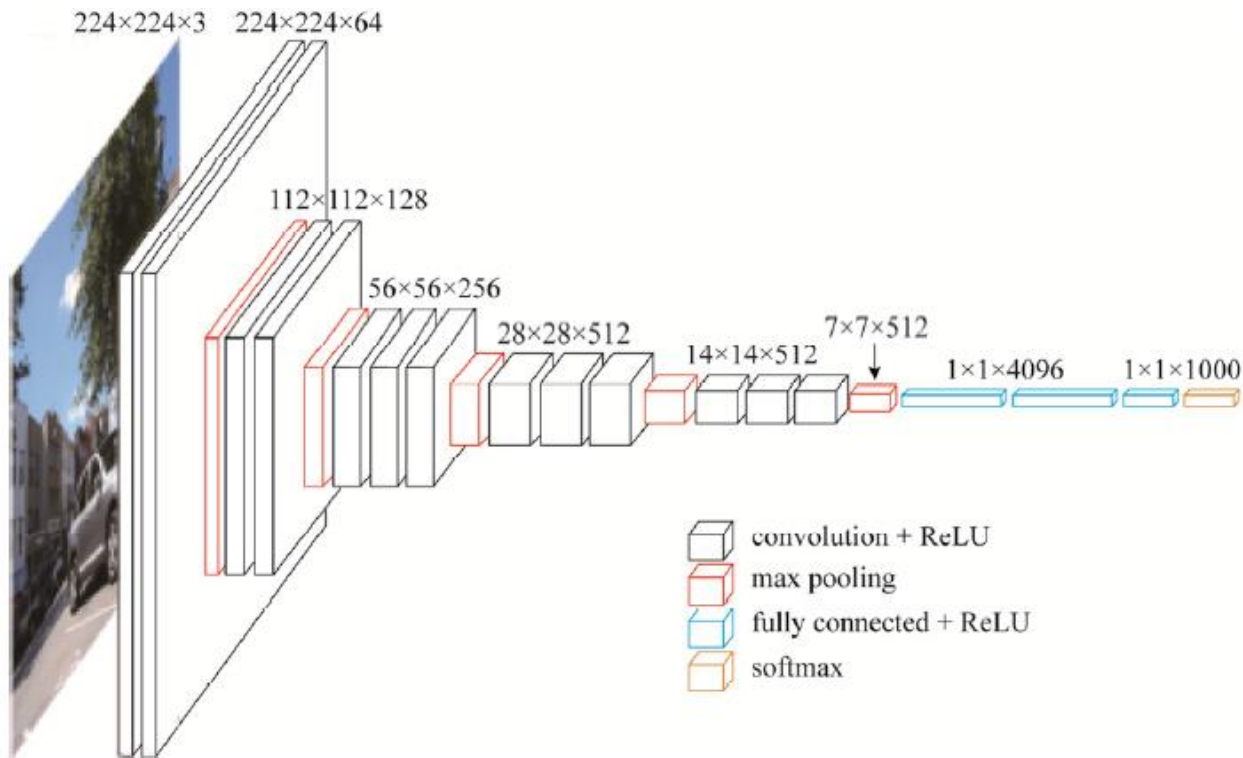


그림 4-22 VGGNet 구조[Simonyan2015]

4.4.2 VGGNet

■ 1*1 커널

- 차원 축소 효과
- [그림 4-23]의 예)
 - $m*n$ 의 특징 맵 8개에 1*1 커널을 4개 적용 → $m*n$ 의 특징 맵 4개가 됨
 - 다시 말하면, $8*m*n$ 텐서에 $8*1*1$ 커널을 4개 적용하여 $4*m*n$ 텐서를 출력하는 셈
- ReLU와 같은 비선형 활성화함수를 적용하면 특징 맵의 분별력 증가
- 『네트워크 속의 네트워크(NIN)』에서 유래 [Lin2014]
- VGGNet은 적용 실험을 하였지만 최종 선택하지는 않음 (GoogLeNet이 많이 사용함)

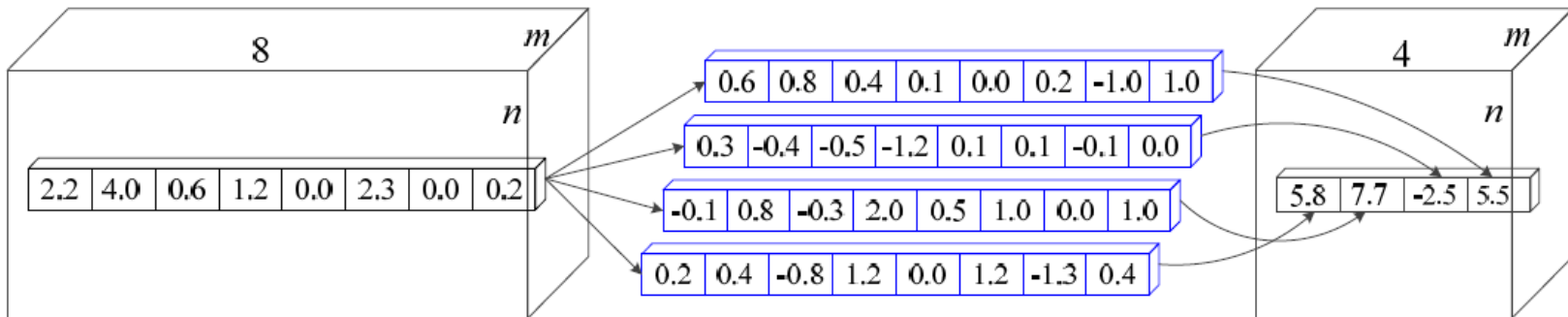


그림 4-23 1*1 컨볼루션 예제

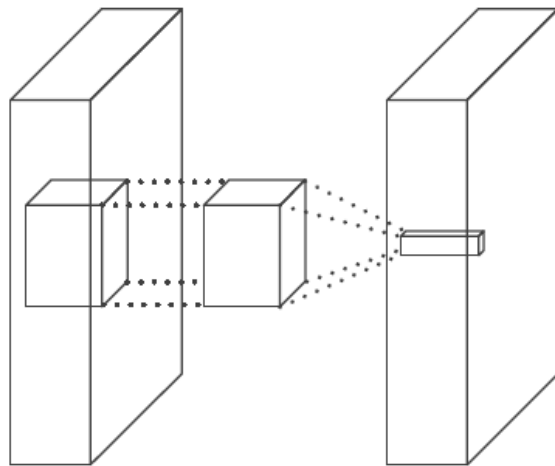
4.4.3 GoogLeNet

■ GoogLeNet의 핵심 아이디어인 인셉션 모듈

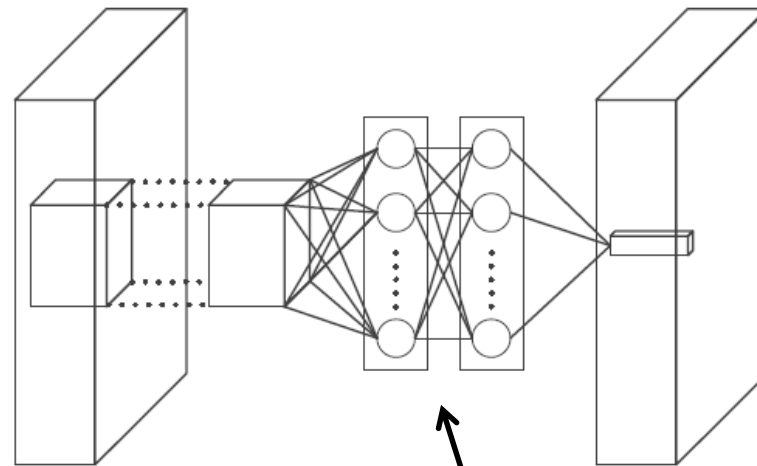
- NIN의 구조를 수정한 것

■ NIN 구조

- MLPconv층이 컨볼루션 연산을 대신함
- MLPconv는 커널을 옮겨가면서 MLP의 전방 계산을 수행함



(a) 기존 컨볼루션층



(b) NIN의 MLPconv층

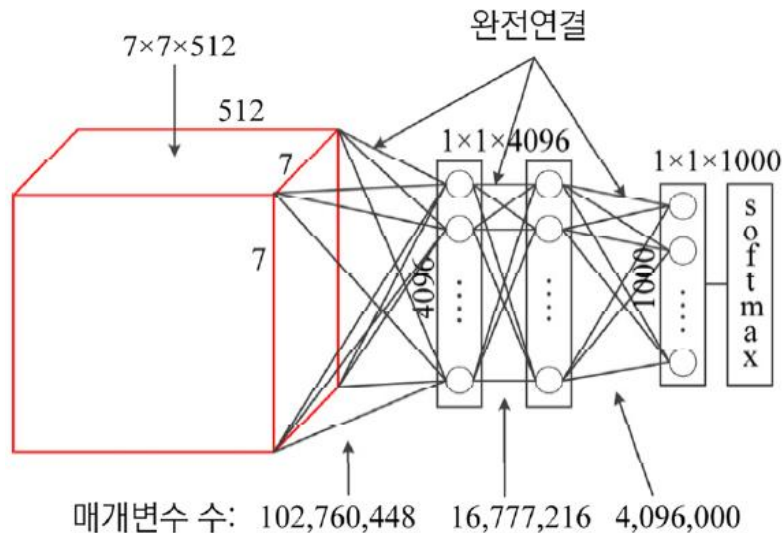
그림 4-24 기존 컨볼루션 신경망과 NIN의 비교

MLPconv층 (마이크로 네트워크)

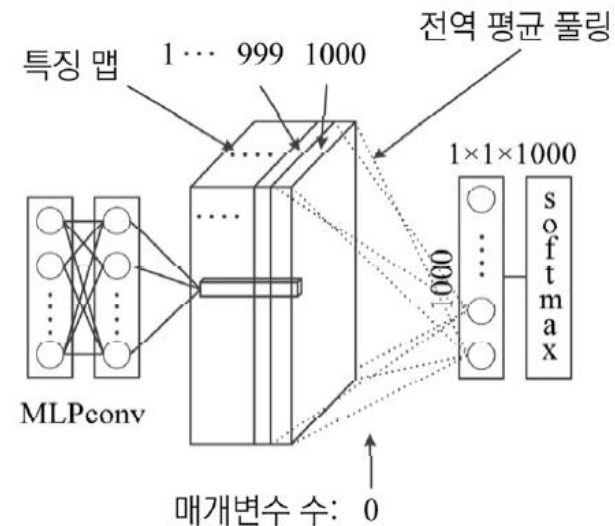
4.4.3 GoogLeNet

■ NIN이 사용하는 전역 평균 풀링

- [그림 4-25(a)]의 VGGNet의 완전연결층
 - 1억2천2백만 개의 매개변수를 가짐 (VGGNet의 전체 매개변수의 85%) → 과잉적합 원인
- [그림 4-25(b)]의 전역 평균 풀링
 - MLPconv가 부류 수만큼 특징 맵을 생성하면, 특징 맵 각각을 평균하여 출력 노드에 입력 → 이 방식으로 매개변수를 없앴



(a) VGGNet의 완전연결



(b) NIN의 전역 평균 풀링

그림 4-25 완전연결과 NIN의 전역 평균 풀링의 비교

4.4.3 GoogLeNet

■ GoogLeNet은 NIN 아이디어를 확장한 신경망

- 인셉션 모듈
 - 마이크로 네트워크로 MLPconv 대신 네 종류의 컨볼루션 연산 사용
 - 1×1 컨볼루션을 사용하여 차원 축소

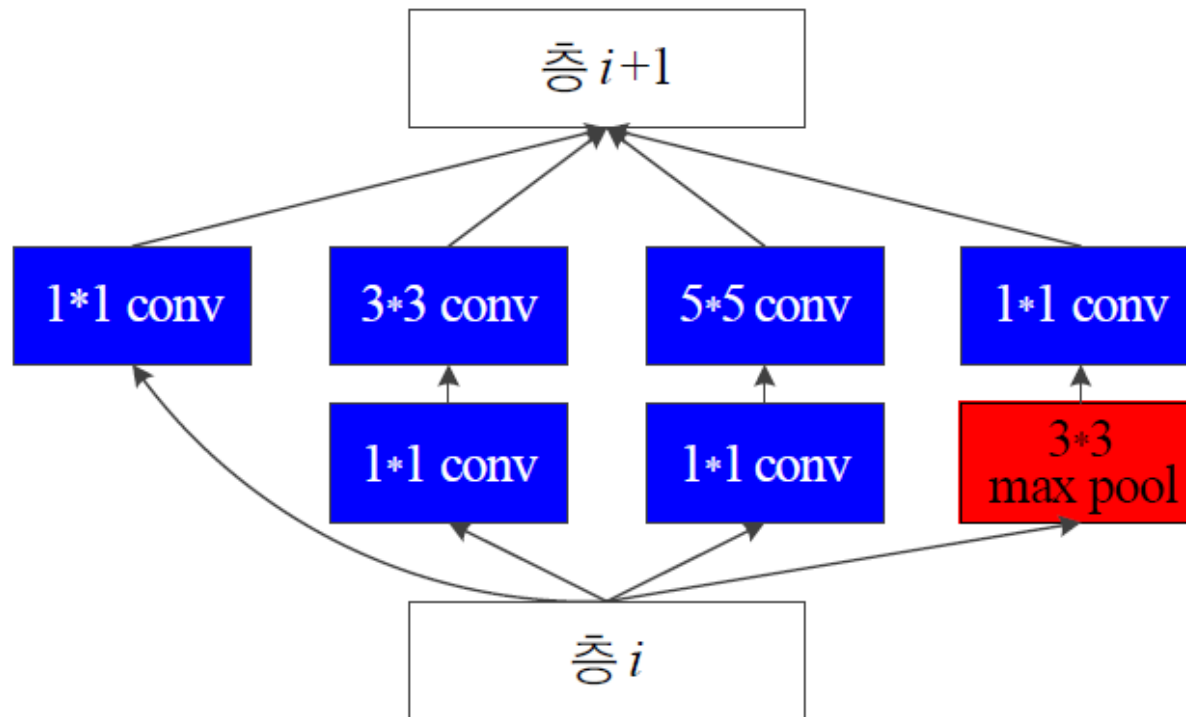


그림 4-26 GoogLeNet의 인셉션 모듈

4.4.3 GoogLeNet

- 인셉션 모듈을 9개 결합한 GoogLeNet ([그림 4-27])
 - 매개변수가 있는 층 22개, 없는 층 5개로 총 27개 층
 - 완전연결층은 1개에 불과
 - 1백만 개의 매개변수를 가지며, VGGNet의 완전연결에 비하면 1%에 불과

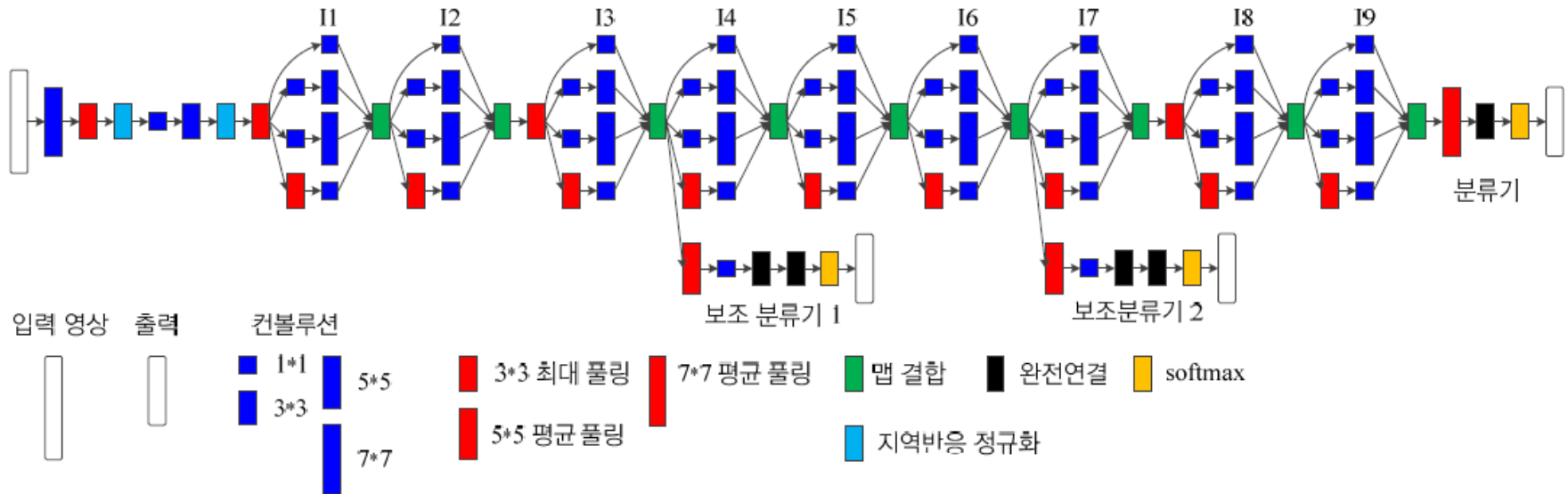


그림 4-27 GoogLeNet의 구조

4.4.4 ResNet

■ ResNet

- 잔류 학습이라는 아이디어를 이용하여 성능 저하를 피하면서 층 수를 대폭 늘림(최대 1202층까지)
- 원래 컨볼루션 신경망

$$\mathbf{F}(\mathbf{x}) = \tau(\mathbf{x} \circledast \mathbf{w}_1) \circledast \mathbf{w}_2$$

$$\mathbf{y} = \tau(\mathbf{F}(\mathbf{x}))$$

- 잔류 학습은 지름길 연결된 \mathbf{x} 를 더한 $\mathbf{F}(\mathbf{x}) + \mathbf{x}$ 에 τ 를 적용. $\mathbf{F}(\mathbf{x})$ 는 잔류

$$\mathbf{y} = \tau(\mathbf{F}(\mathbf{x}) + \mathbf{x})$$

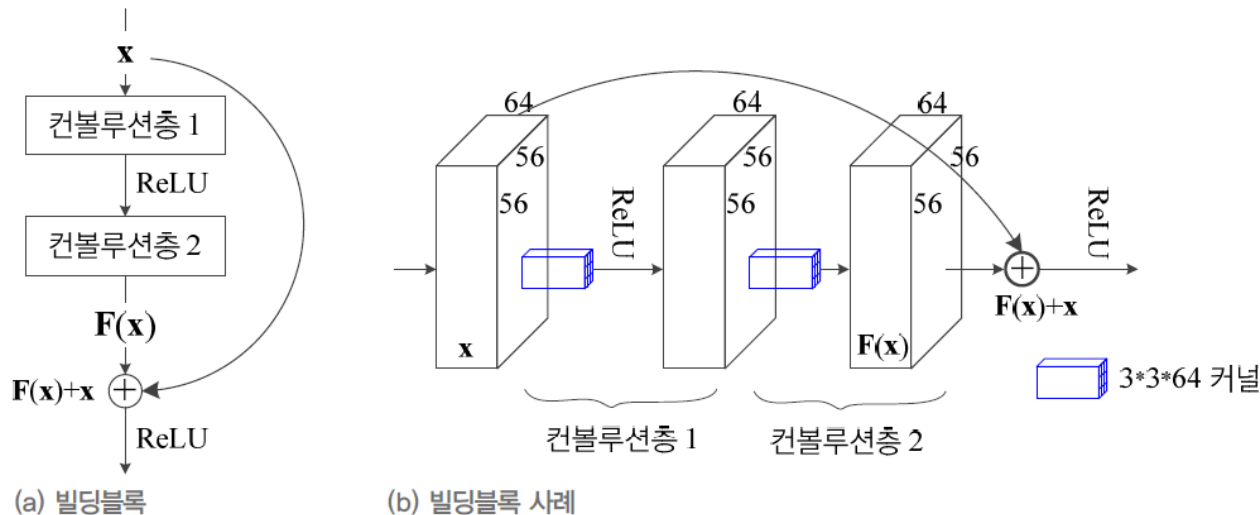


그림 4-28 잔류 학습의 구조와 동작

4.4.4 ResNet

■ 지름길 연결을 두는 이유는?

- 그레이디언트 소멸 문제 해결
- 식 (4.14)의 그레이디언트 식에서 $\frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} \mathbf{F}(\mathbf{x}_i)$ 이 -1이 될 가능성이 거의 없음

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L} \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L} \left(1 + \frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} \mathbf{F}(\mathbf{x}_i) \right) \quad (4.14)$$

4.4.4 ResNet

■ 34층짜리 ResNet 예시

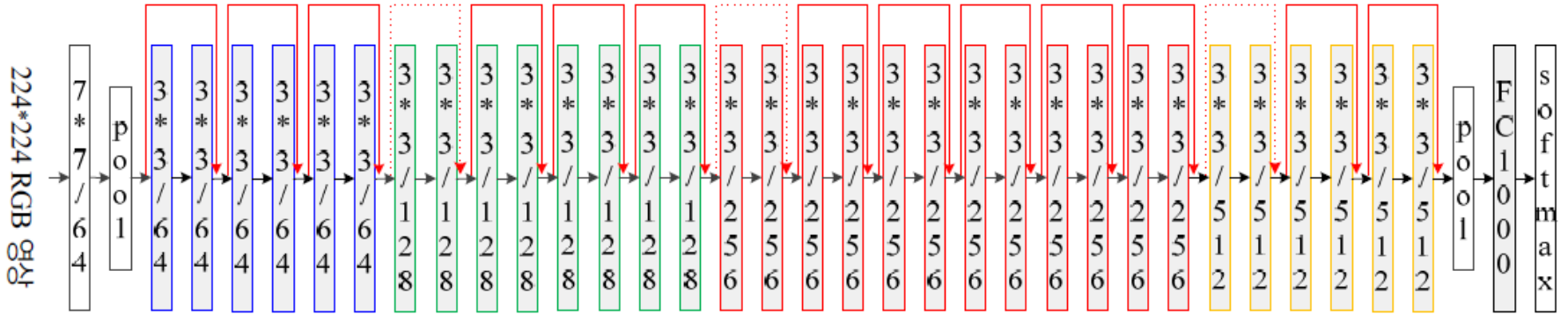


그림 4-29 ResNet 예제(34층)

■ VGGNet과 같은 점

- 3*3 커널 사용

■ VGGNet과 다른 점

- 잔류 학습 사용
- 전역 평균 풀링 사용(FC 층 제거)
- 배치 정규화 적용(드롭아웃 적용 불필요) → 배치 정규화는 5.2.6절

4.4.4 ResNet

■ ILSVRC 대회 성적

- 2012년 AlexNet의 15.3% 오류율은 당시로서 경이로운 성능
- 2015년에 ResNet은 3.5% 오류율 달성

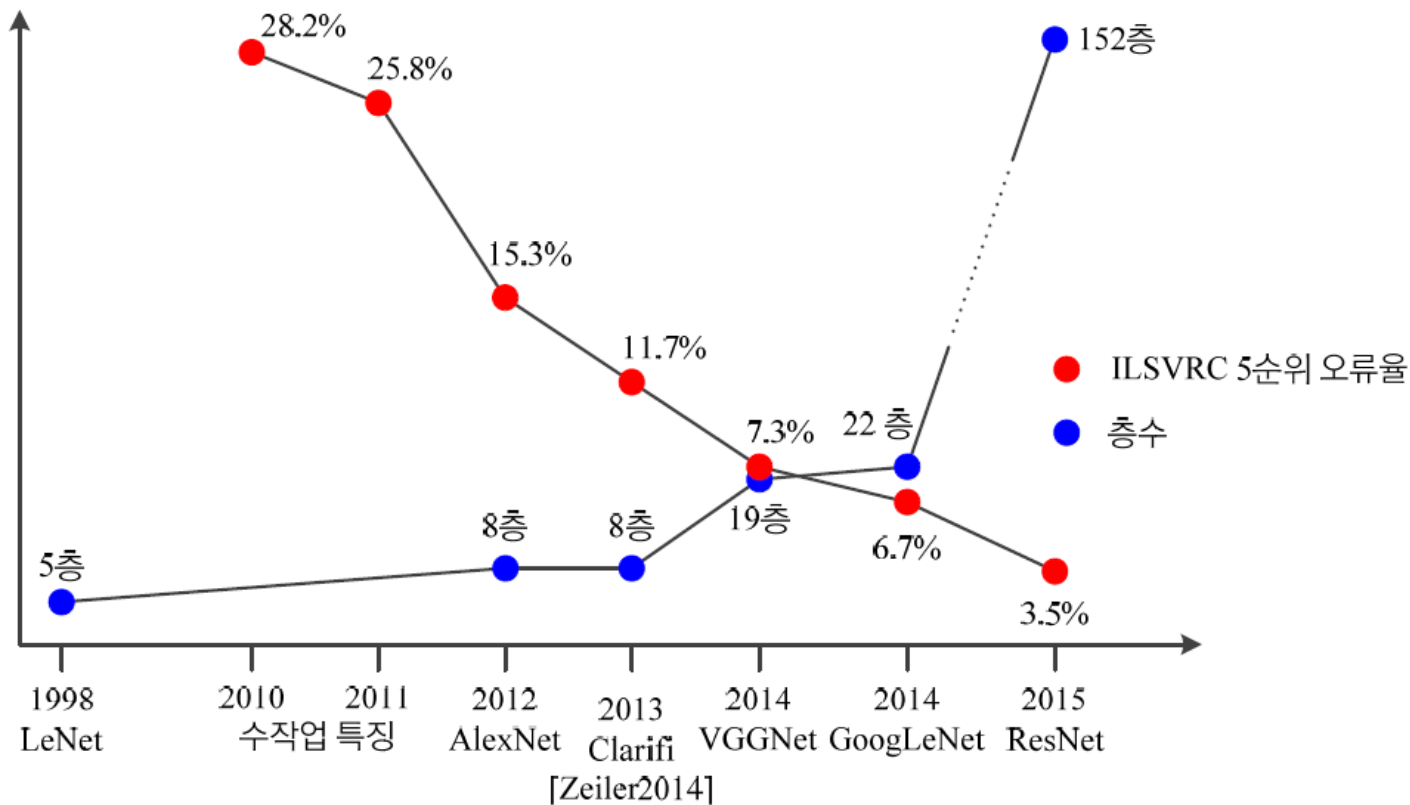


그림 4-30 CNN의 발전 추세

4.4.4 ResNet

■ ILSVRC 대회

- 분류 문제는 성능 포화 (사람 성능에 필적함)
- 물체 검출 문제에 집중함

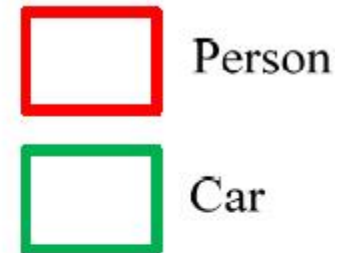


그림 4-31 ILSVRC 물체 검출 문제

4.5 생성 모델

■ 4.5.1 생성 모델이란?

■ 4.5.2 GAN

눈을 감고 황금 들녘을 상상해보라. 근사한 영상이 떠오른다면 머리 속에 있는 ‘생성 모델 generative model’이 작용한 탓이다. 사람의 생성 모델은 세상에 나타나는 현상을 오랫동안 지켜보면서 학습한 결과이다. 만일 기계 학습이 훈련집합을 사용하여 비슷한 생성 모델을 구축할 수 있다면 강한 인공지능^{strong AI}에 한발 다가설 수 있다. 왜냐하면 생성 모델은 분별 모델에 비해 데이터 생성 과정에 대한 보다 깊은 이해를 필요로 하기 때문이다[Karpathy2015].

4.5.1 생성 모델이란?

■ 분별 모델과 생성 모델의 비교

표 4-1 분별 모델과 생성 모델

모델	학습 단계가 할 일	예측 단계가 할 일	지도 여부
분별 모델	$P(y x)$ 추정	$f: x \mapsto y$	지도 학습
생성 모델	$P(x)$ 또는 $P(x y)$, $P(x, y)$ 추정	$f: \text{씨앗} \mapsto x$ 또는 $f: \text{씨앗 } y \mapsto x$, $f: \text{씨앗} \mapsto x, y$	비지도 학습

4.5.1 생성 모델이란?

예제 4-1

생성 모델과 분별 모델의 확률분포 추정과 예측

특징 벡터가 2차원이고 이진값을 가지며, 부류가 2개라 가정하자. 훈련집합은 다음과 같다.

$$\mathbb{X} = \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}, \mathbb{Y} = \{1, 1, 1, 0, 1, 0, 0, 0, 0, 1\}$$

생성 모델이 추정하는 확률 분포

$P(\mathbf{x}) =$

$\mathbf{x} = (0,0)^T$	0.2
$\mathbf{x} = (0,1)^T$	0.3
$\mathbf{x} = (1,0)^T$	0.1
$\mathbf{x} = (1,1)^T$	0.4

$P(\mathbf{x}|y) =$

	$y = 0$	$y = 1$
$\mathbf{x} = (0,0)^T$	0.0	0.4
$\mathbf{x} = (0,1)^T$	0.2	0.4
$\mathbf{x} = (1,0)^T$	0.2	0.0
$\mathbf{x} = (1,1)^T$	0.6	0.2

$P(\mathbf{x}, y) =$

	$y = 0$	$y = 1$
$\mathbf{x} = (0,0)^T$	0.0	0.2
$\mathbf{x} = (0,1)^T$	0.1	0.2
$\mathbf{x} = (1,0)^T$	0.1	0.0
$\mathbf{x} = (1,1)^T$	0.3	0.1

4.5.1 생성 모델이란?

분별 모델이 추정하는 확률 분포

$$P(y|\mathbf{x}) =$$

	$y = 0$	$y = 1$
$\mathbf{x} = (0,0)^T$	0.0	1.0
$\mathbf{x} = (0,1)^T$	0.33	0.67
$\mathbf{x} = (1,0)^T$	1.0	0.0
$\mathbf{x} = (1,1)^T$	0.75	0.25

학습을 마쳤으니, 이제 예측 단계를 수행해보자. 생성 모델이 $P(\mathbf{x})$ 를 사용하고, 네 가지 \mathbf{x} 값의 확률에 따라 $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ 에게 $[0.0, 0.2]$, $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ 에게 $(0.2, 0.5]$, $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 에게 $(0.5, 0.6]$, $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ 에게 $(0.6, 1.0]$ 구간을 부여하자. 난수로 0.34가 나오면 $(0.2, 0.5]$ 에 속하므로 $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ 을 생성하고, 0.83이 나오면 $(0.6, 1.0]$ 에 속하므로 $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ 을 생성한다.

분별 모델의 예측을 생각해보자. 만일 테스트 샘플 $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ 이 주어진다면, $P\left(y = 0 | \mathbf{x} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right) = 0.33$ 이고 $P\left(y = 1 | \mathbf{x} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right) = 0.67$ 이므로 $y = 1$ 이라고 분류하면 된다.

4.5.1 생성 모델이란?

■ 실제 상황에서 생성 모델

- 자연계에 내재한 데이터 발생 분포 $P_{data}(\mathbf{x}) \rightarrow$ 알아낼 수 없음

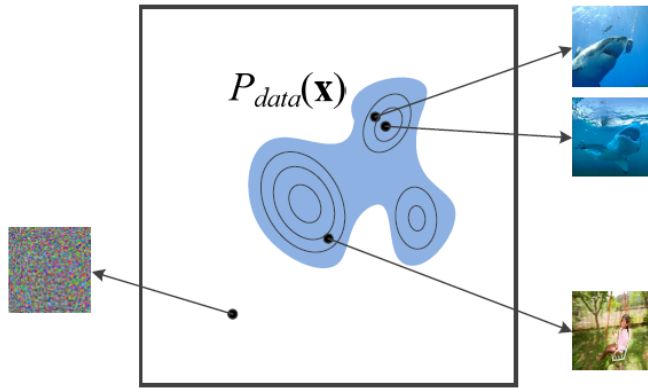


그림 4-32 확률분포 $P_{data}(\mathbf{x})$ 에 따른 자연 영상의 발생 과정

- $P_{data}(\mathbf{x})$ 를 모방하는 모델의 확률 분포 $P_{model}(\mathbf{x}; \theta)$
 - $P_{model}(\mathbf{x}; \theta)$ 를 명시적으로 추정하는 것도 불가능
 - 현대 기계 학습은 주로 딥러닝 모델을 사용하여 확률 분포를 암시적으로 표현
 - GAN(generative adversarial network), VAE(variational autoencoder), RNN(8장), RBM(10장)

4.5.2 GAN(generative adversarial network)

■ GAN의 우월한 성능

- 사람을 상대로 진짜와 가짜 구별하는 실험에서, MNIST 52.4%, CIFAR-10 78.7% (50%이면 완벽히 속임)

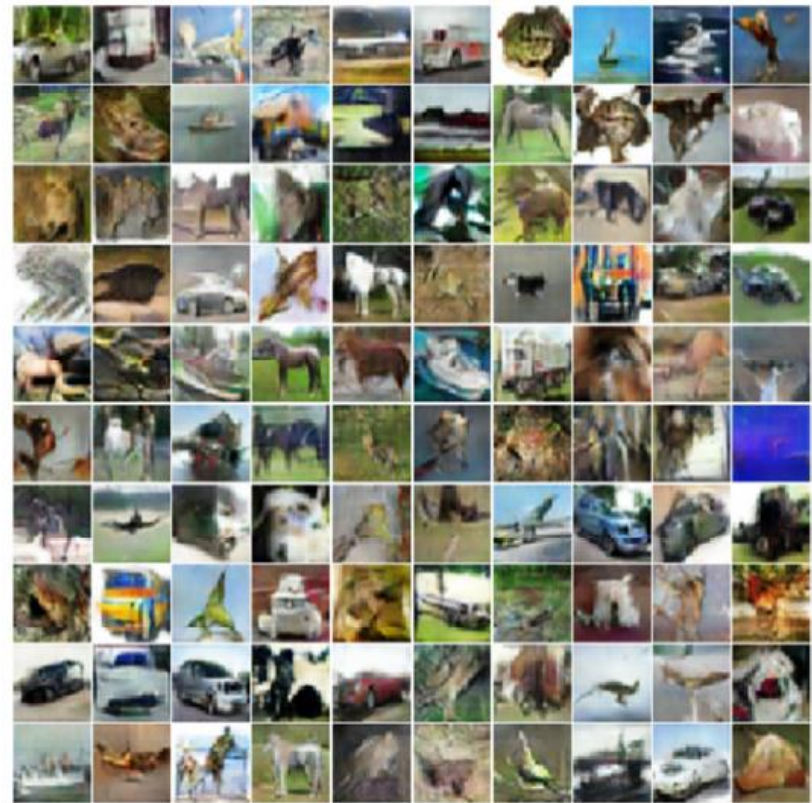


그림 4-33 GAN으로 생성한 영상 샘플

4.5.2 GAN

■ GAN의 아이디어

- 생성기 G와 분별기 D의 대립 구조
 - G는 가짜 샘플 생성(위조지폐범)
 - D는 가짜와 진짜를 구별(경찰)
- GAN의 목표는 위조지폐범의 승리(G가 만들어내는 샘플을 D가 구별하지 못하는 수준까지 학습)

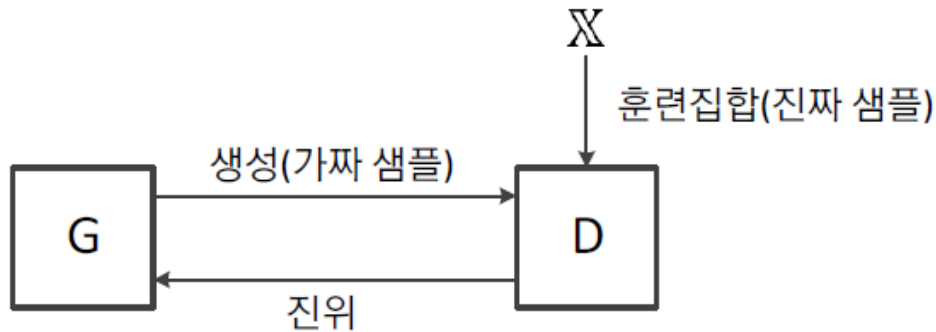


그림 4-34 GAN의 원리

4.5.2 GAN

■ 최초 GAN[Goodfellow2014]

- G와 D를 DMLP로 구현
 - G는 $f_G(z; \theta_G)$, D는 $f_D(x; \theta_D)$ 로 표기 (θ_G 와 θ_D 는 매개변수)
 - f_G 는 난수 발생기로 만든 벡터 z 를 입력으로 받아 가짜 영상을 출력
 - f_D 는 영상을 입력으로 받아 진짜(1) 또는 가짜(0)를 출력

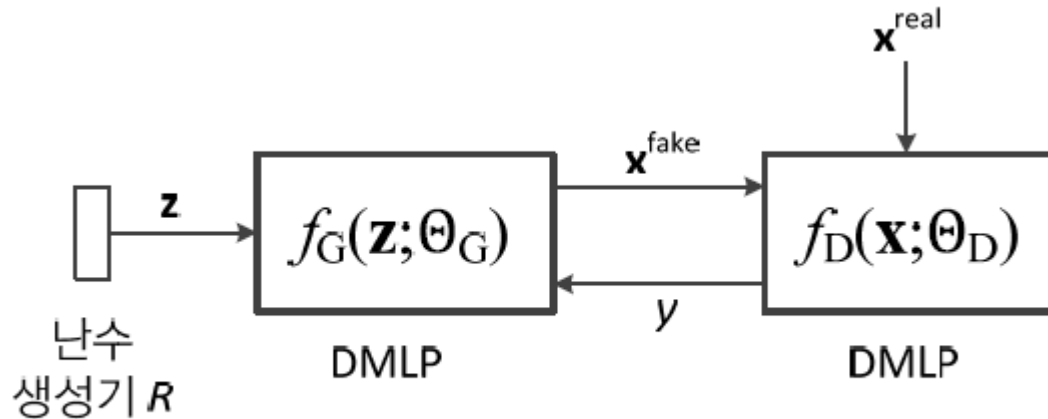


그림 4.35 GAN의 구조

4.5.2 GAN

■ GAN의 목적함수

- 목적함수 J_D 와 J_G 로 (MSE 대신) 로그우도(5.1절)를 사용
- 분별기 D의 목적함수 J_D

$$\hat{\theta}_D = \operatorname{argmax}_{\theta_D} J_D(\theta_D)$$

$$\begin{aligned}\circ \text{이때 } J_D(\theta_D) &= \log(f_D(\mathbf{x}^{\text{real}})) + \log(1 - f_D(\mathbf{x}^{\text{fake}})) \\ &= \log(f_D(\mathbf{x}^{\text{real}})) + \log(1 - f_D(f_G(\mathbf{z})))\end{aligned}\tag{4.15}$$

- 동작 예시1) \mathbf{x}^{real} 에 대해 0.999, \mathbf{x}^{fake} 에 대해 0.001 출력하면 $J_D = \log 0.999 + \log(1 -$

$$\hat{\theta}_G = \operatorname{argmin}_{\theta_G} J_G(\theta_G)$$

$$\circ \text{이때 } J_G(\theta_G) = \log(1 - f_D(f_G(\mathbf{z})))\tag{4.16}$$

4.5.2 GAN

■ GAN의 학습

알고리즘 4-2 GAN 학습

입력: 훈련집합 \mathbb{X} , 학습률 ρ , 분별기 반복 횟수 k , 미니배치 크기 m

출력: 분별기와 생성기의 최적 매개변수 $\hat{\theta}_D, \hat{\theta}_G$

```
1 repeat
2    $\theta_D$ 와  $\theta_G$ 를 초기화한다.
3   for(j = 1 to k)
4     난수 생성기 R로 m개의 벡터,  $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m\}$ 을 생성한다.
5      $\mathbb{X}$ 에서 m개의 샘플을 무작위 선택하여 미니배치  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ 을 구축한다.
6     그래디언트  $\frac{\partial J_D}{\partial \theta_D}$ 를 계산한다.
7      $\theta_D = \theta_D + \rho \frac{\partial J_D}{\partial \theta_D}$  // 최대화 문제이므로 그래디언트를 더함
8     난수 생성기 R로 m개의 벡터,  $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m\}$ 을 생성한다.
9     그래디언트  $\frac{\partial J_G}{\partial \theta_G}$ 를 계산한다.
10     $\theta_G = \theta_G - \rho \frac{\partial J_G}{\partial \theta_G}$  // 최소화 문제이므로 그래디언트를 뺌
11  until(멈춤 조건)
12   $\hat{\theta}_D = \theta_D, \hat{\theta}_G = \theta_G$ 
```

4.5.2 GAN

■ 개선된 GAN

- [Salimans2016]
 - 특징 매칭, 가상 배치 정규화, 미니배치 분별 등 기법 적용
 - [그림 4-33]은 생성된 샘플 예시
 - $P(\mathbf{x})$ 대신 $P(\mathbf{x}, y)$ 를 추정하여 레이블이 있는 샘플을 생성 → 준지도 학습에 활용 (7.3.2절)
- DCGAN^{deep convolutional GAN}
 - DMLP 대신 CNN을 사용
 - 4.4절의 CNN과 데이터 흐름이 반대. 즉 벡터를 입력 받아 3*64*64 영상을 출력

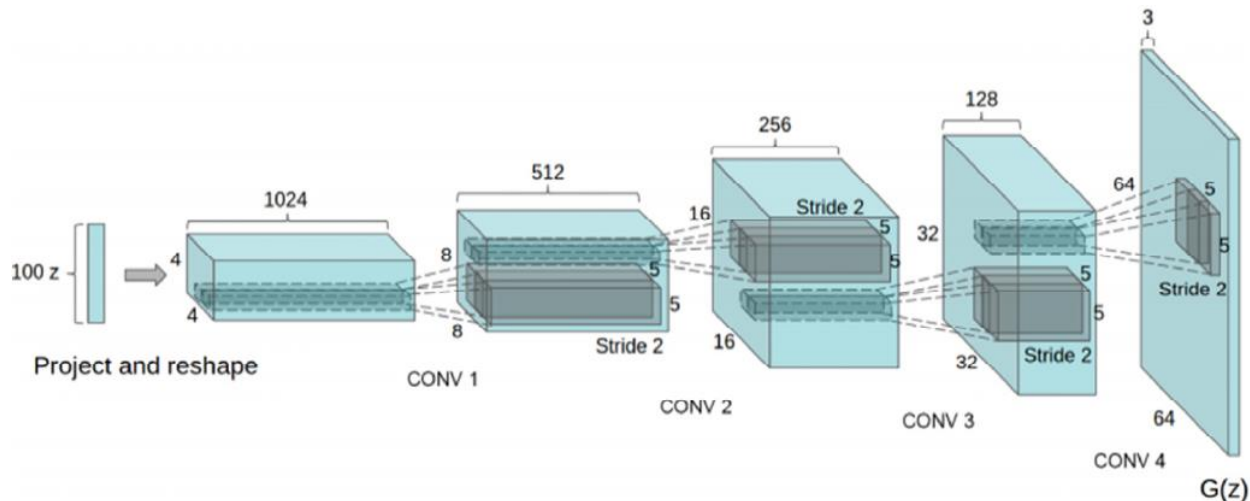


그림 4-36 DCGAN의 구조

4.6 딥러닝은 왜 강력한가?

■ 전체 과정을 동시에 최적화

- 고전적인 방법에서는
 - [그림 4-37]처럼 분할, 특징 추출, 분류를 따로 구현한 다음 이어 붙임
 - 사람의 직관에 따르므로 성능 한계. 인식 대상이 달라지면 새로 설계해야 함

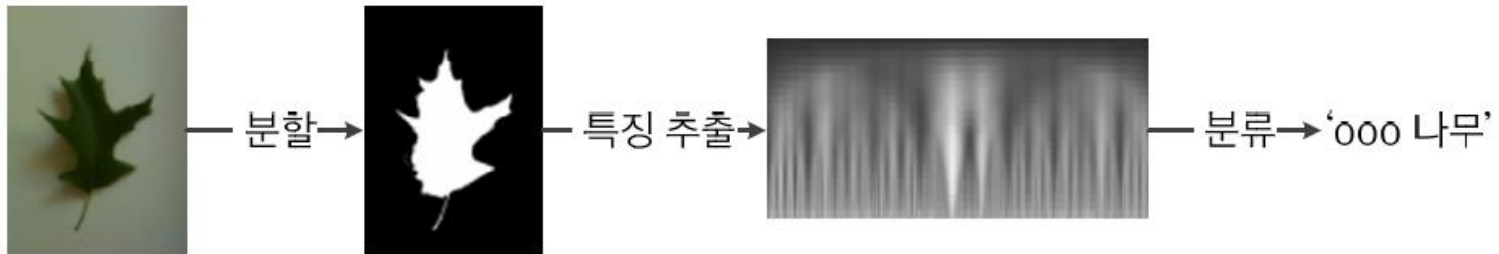


그림 4-37 여러 단계를 따로 설계 구현하는 고전적인 접근방식(나뭇잎 인식 사례)

- 딥러닝은 전체 과정을 동시에 최적화 (통째 학습이라 부름)

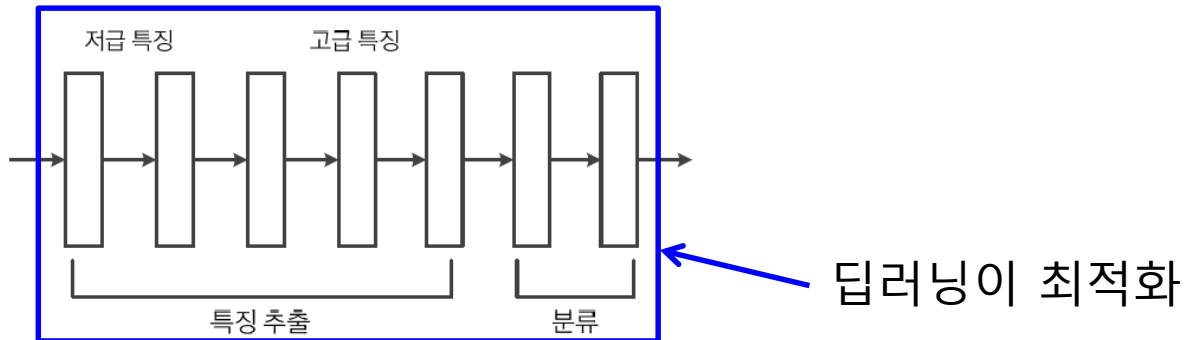


그림 4-2 깊은 신경망의 처리 절차

4.6 딥러닝은 왜 강력한가?

■ 깊이의 중요성

- 점선은 20개 노드를 가진 은닉층 하나 짜리 신경망
- 실선은 각각 10개 노드를 가진 은닉층 두 개 짜리 신경망 → 더 정교한 분할

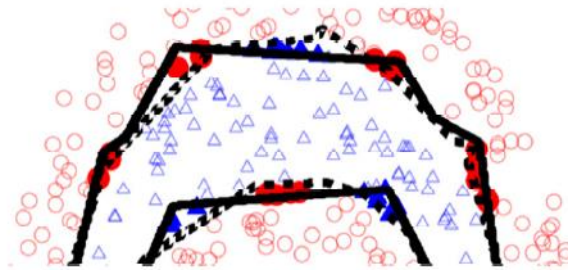
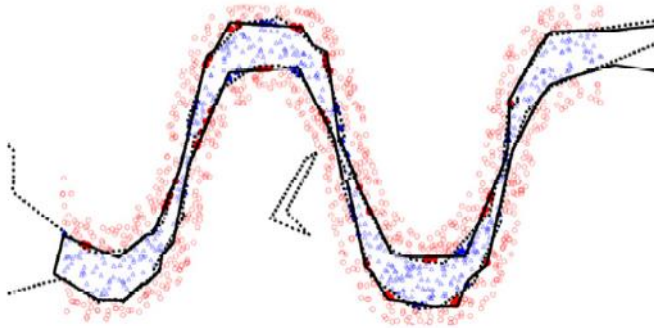


그림 4-38 은닉층의 개수가 늘어남에 따른 표현력 증가

4.6 딥러닝은 왜 강력한가?

■ 계층적 특징

- [그림 4-40]은 ImageNet으로 학습한 특징 맵 → 계층 구조
 - 깊은 신경망에서는 층의 역할이 잘 구분됨
- 반면 얇은 신경망은 하나 또는 두 개의 은닉층이 여러 형태의 특징을 모두 담당

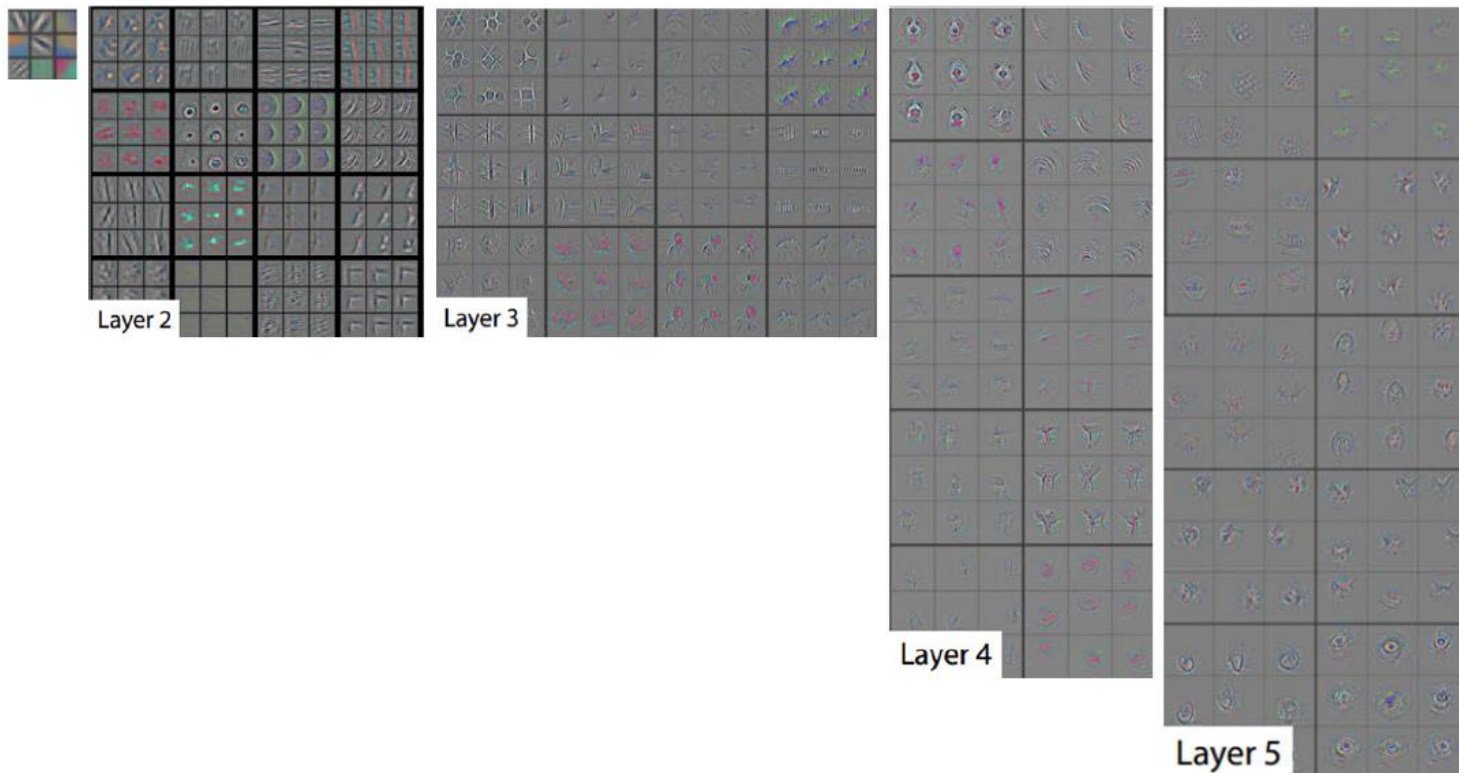


그림 4-40 CNN의 계층적 특징 추출

참고

- 기계학습, 오일석 지음
- 밑바닥부터 시작하는 딥러닝, 한빛미디어
- Seoul National Uni. Vision&learning Lab
 - Convolutional Neural networks.
- <http://aikorea.org/cs231n/convolutional-networks>