딥러닝의 통계적이해

6강. 합성곱 신경망의 기초(1)

1. 컴퓨터 비전과 디지털 이미지
2. 합성곱 신경망의 역사
3. 합성곱 연산

한국방송통신대 이긍희 교수

오늘의 학습목표

1. 컴퓨터 비전을 이해한다.
2. 합성곱 신경망의 역사를 이해한다.
3. 합성곱 연산을 이해한다.

1. 컴퓨터 비전과 디지털 이미지

디지털 이미지

- 아날로그 데이터는 연속형 신호 → 일정 간격으로 샘플링(sampling)하여 측정
  - 이미지 샘플링을 많이 하면 화소가 높아지고, 샘플링을 적게 하면 화소가 낮아짐
  - 아날로그 데이터는 무한한 범위, 디지털 데이터는 2진수 즉 비트(bit)로 표현
    - 8비트는 $2^8 = 256$개 값으로 데이터 표현

디지털 이미지

- 이미지 : $f(x, y)$형태의 2차원 함수
  - 흑백 이미지 : 1개의 행렬
    - 8x8의 흑백 이미지 : 4비트($2^4 = 16$) 색상 행렬 표현
    - 0 : 검정색, 16: 흰색
  - 컬러 이미지 : 적색(Red), 녹색(Green), 청색(Blue) 3개 행렬 조합, 색상으로 표현

디지털 이미지(흑백)

```
[[ 0.  0.  1.  9. 15. 11.  0.  0.]
 [ 0.  0. 11. 16.  8. 14.  6.  0.]
 [ 0.  2. 16. 10.  0.  9.  9.  0.]
 [ 0.  1. 16.  4.  0.  8.  8.  0.]
 [ 0.  4. 16.  4.  0.  8.  8.  0.]
 [ 0.  1. 16.  5.  1. 11.  3.  0.]
 [ 0.  0. 12. 12. 10. 10.  0.  0.]
 [ 0.  0.  1. 10. 13.  3.  0.  0.]]
```

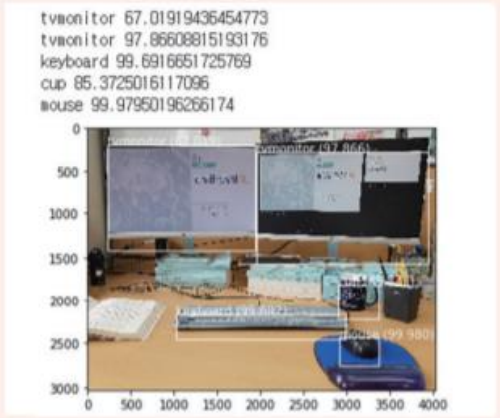
디지털 이미지(컬러)

```
[[[ 25  55  36]
  [ 23  53  34]
  [ 24  52  33]
  ...
  [ 62 143 110]
  [ 60 141 108]
  [ 58 139 106]]

 [[ 17  47  28]
  [ 19  49  30]
  [ 25  53  34]
  ...
  [ 61 142 109]
  [ 60 141 108]
  [ 57 138 105]]

 [[ 17  47  28]
  [ 19  49  30]
  [ 24  52  33]
  ...
  [ 65 146 113]
  [ 63 144 111]
  [ 61 142 109]]

 ...
```

컴퓨터 비전

- 컴퓨터 저장 이미지 또는 동영상으로부터 유용한 정보 추출·분석·이해
  - 이미지 분류, 객체 탐색·검출, 객체 분할, 얼굴 인식, 이미지 합성 등

(a) 객체탐색



(b) 객체분할

이미지 인식 데이터베이스

- 이미지 인식 경진대회 → 컴퓨터 비전 능력의 획기적 발전

(a) 이미지넷

(b) COCO

출처 : (a) http://imagenet.org (b) http://cocodataset.org

2. 합성곱 신경망의 역사

이미지 인식과 신경망

- 완전연결 신경망 : 이미지 작업 어렵고, 과대적합 문제 발생

  - 이미지 데이터는 입력층에 수천만 개의 변수가 입력됨

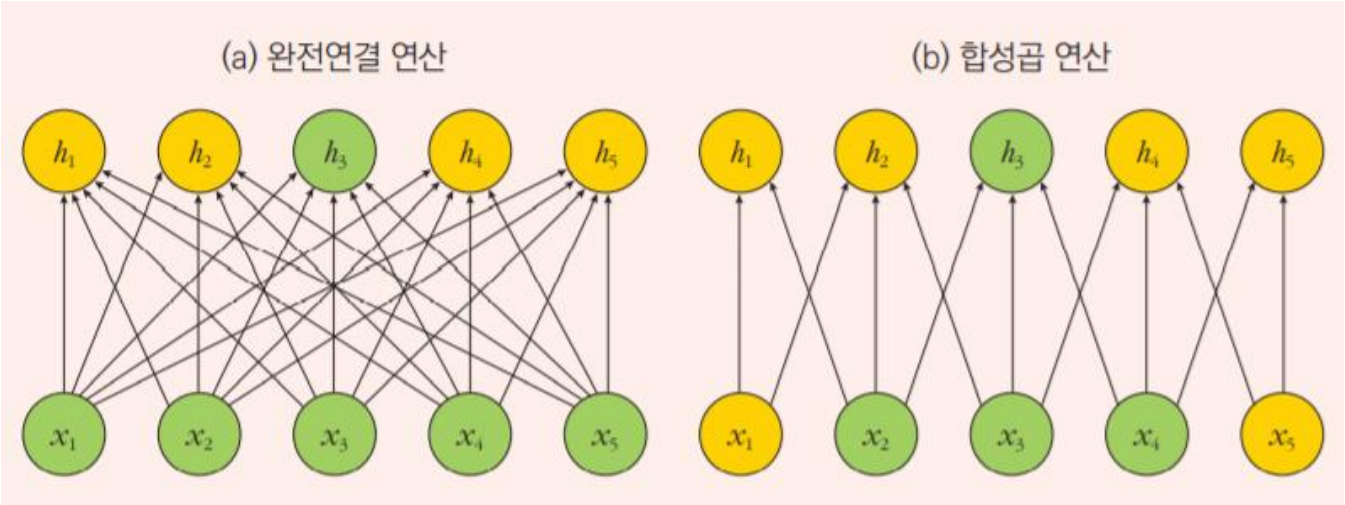
뇌 시각의 연구

- 1959년 허블(Hubel)과 비셀(Wiesel)의 고양이 실험
  - 시각 뉴런들이 뉴런마다 하는 일이 다르고, 그들이 결합하여 작용함
  - 영상 인식 시, 특정 영역의 뉴런이 활성화
  - 상위 뉴런이 더 추상적인 것 파악 : 계층적 특성

  → 컴퓨터 비전 연구자들은 합성곱(convolution) 신경망을 생각

합성곱 연산

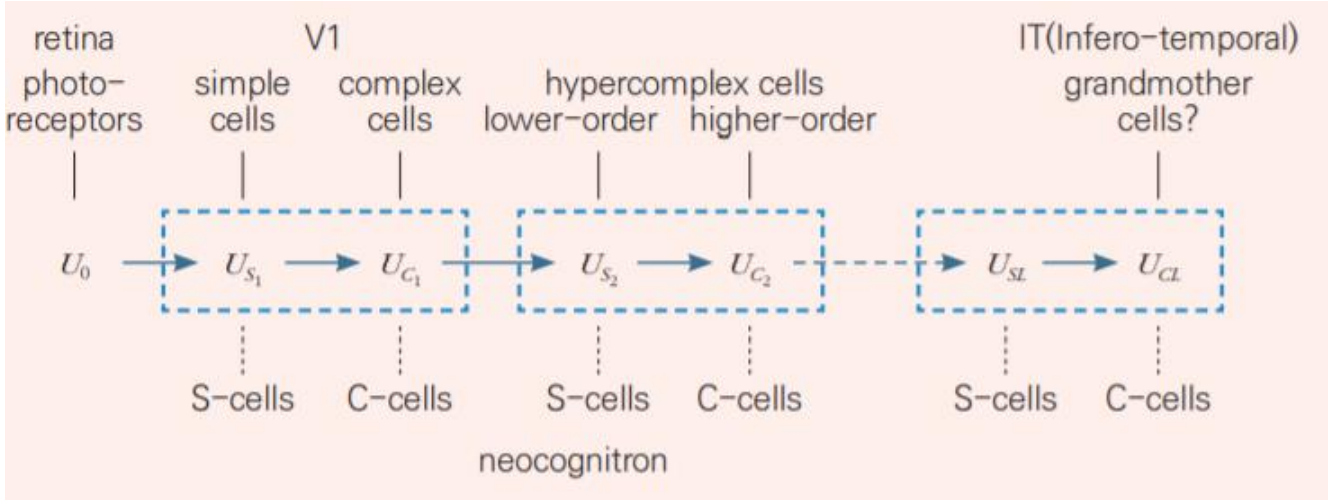
- 가중치를 공유, 근처에 있는 뉴런들만 이동 연산
  → 완전연결 신경망보다 희소한 연결



출처 : Goodfellow et al.(2016)

네오코그니트론(Neocognitron)

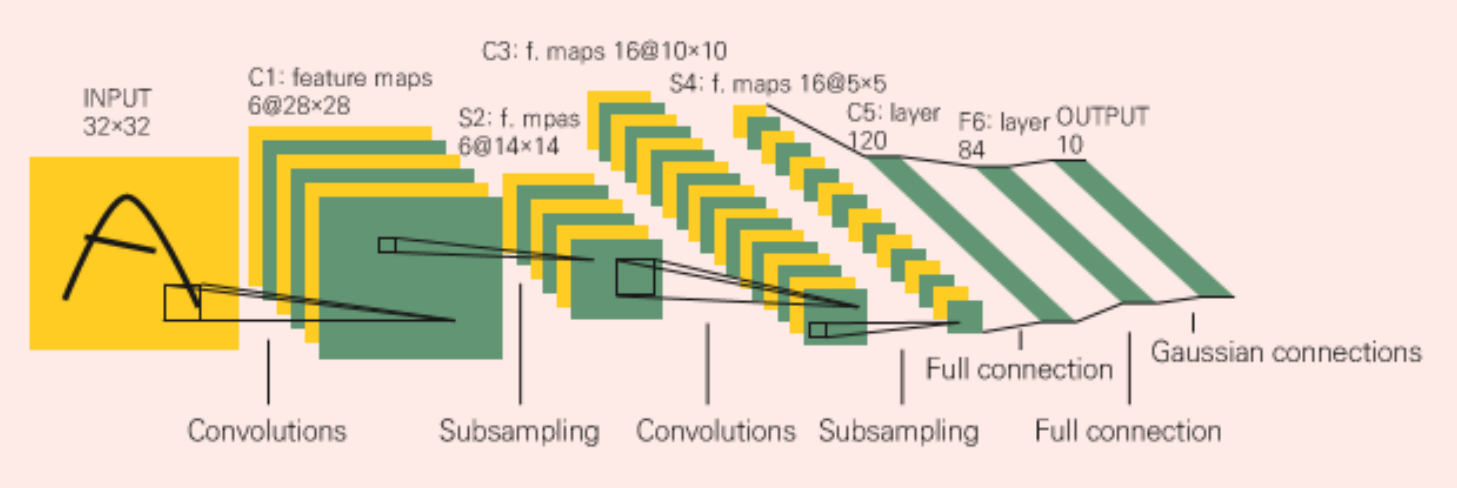
- 1979년 후쿠시마의 합성곱 신경망과 유사한 신경망 제안
  - S-cell : 국부적 특징 파악, C-cell : 이를 결합
  - S-cell과 C-cell을 계층적 적용 → 이미지 식별



출처 : : Fukushima(1980)

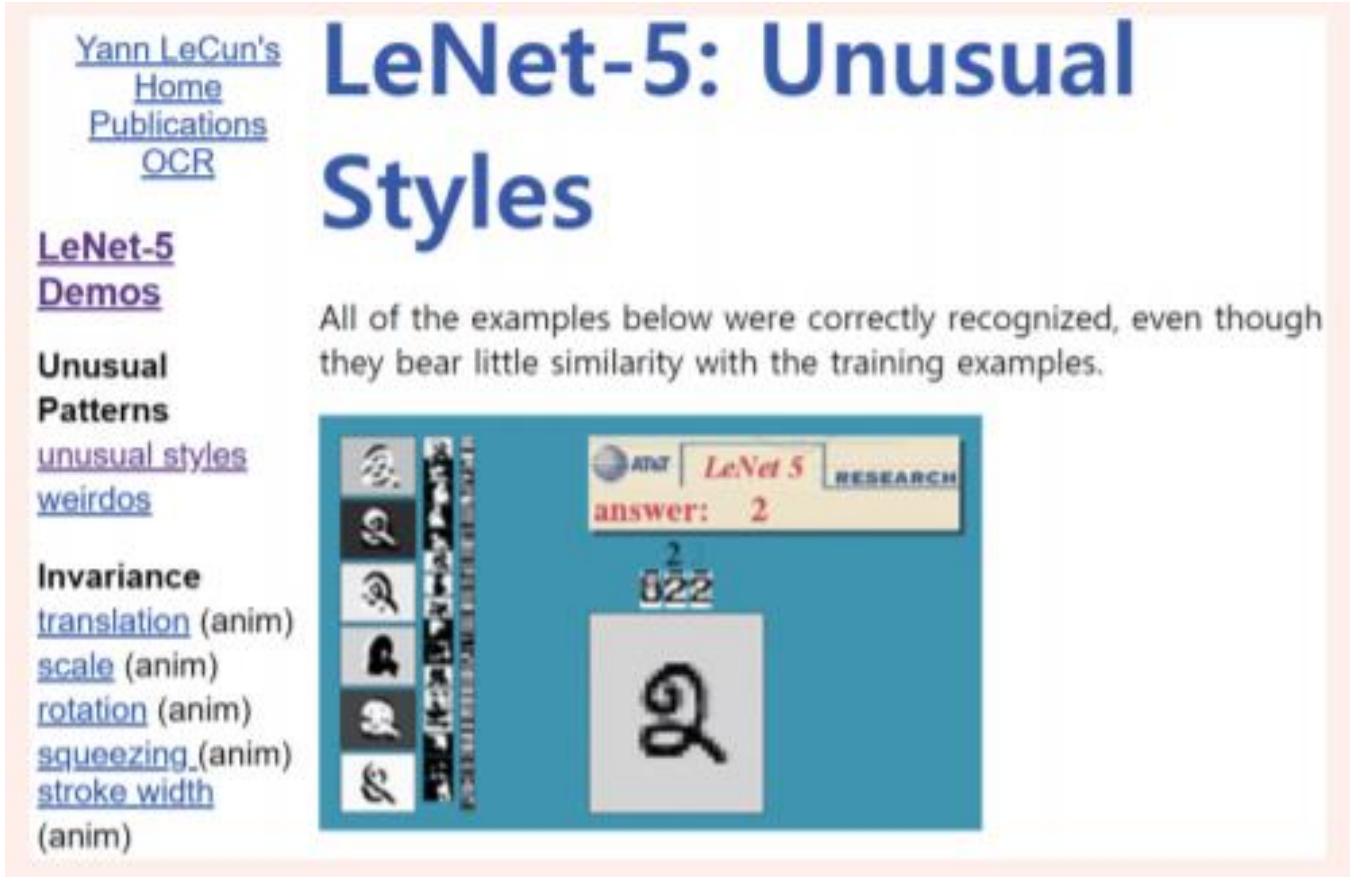
LeNet-5

- 1998년 얀 르쿤(Y. LeCun) 연구팀의 합성곱 신경망
  - 3개 합성곱(Convolution) 층에 완전연결 신경망을 추가



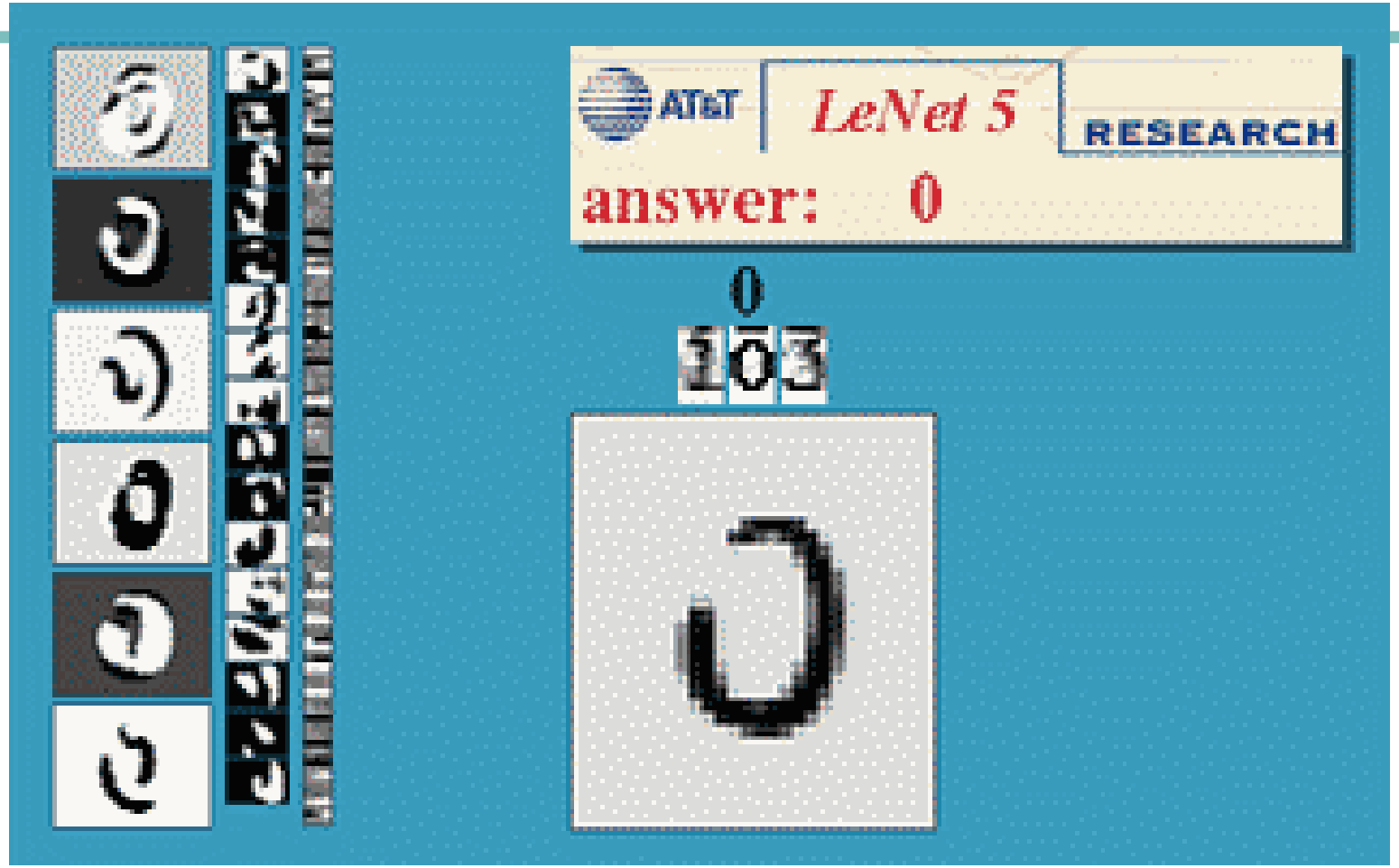
출처 : LeCun et al.(1998)

LeNet-5



출처 :http://yann.lecun.com/exdb/lenet/index.html

LeNet-5



출처 :http://yann.lecun.com/exdb/lenet/index.html

이미지 인식 경진대회(ILSVRC)

- 2012년 이미지넷(Imagenet)의 경진대회(ILSVRC)에서 층이 깊은 합성곱 신경망 AlexNet이 획기적인 성과를 보였음
  - 그 후 GoogLeNet, VGGNet, ResNet 등 층이 깊은 합성곱 신경망이 꾸준히 만들어짐

3. 합성곱 연산

합성곱 연산

- 필터(filter, kernel)를 통해 입력 데이터에 원소 단위로 연산

입력 데이터

a	b	c
d	e	f
g	h	k

a	b	c
d	e	f
g	h	k

필터

w_{11}	w_{12}
w_{21}	w_{22}

합성곱 연산 결과

$aw_{11}+bw_{12}$ $+dw_{21}+ew_{22}$	$bw_{12}+cw_{12}$ $+ew_{21}+fw_{22}$
$dw_{11}+ew_{12}$ $+gw_{21}+hw_{22}$	$ew_{11}+fw_{12}$ $+hw_{21}+kw_{22}$

합성곱 연산의 예

입력 데이터

0	1	2
3	4	5
6	7	0

필터

0	1
2	3

합성곱 연산

- 이미지 데이터 크기 $n \times n$, 필터의 크기 $f \times f \rightarrow$ 필터에 의해 생성된 결과 데이터의 크기 $(n-f+1) \times (n-f+1)$
  - 입력 이미지의 크기 3×3, 필터의 크기 2×2

    $\rightarrow$ 출력 이미지의 크기

    $(3-2+1) \times (3-2+1) = 2 \times 2$

합성곱 필터와 이미지

0	0	0
0	1	0
0	0	0

```
[[[ 25  55  36]
  [ 23  53  34]
  [ 24  52  33]
  ...
  [ 62 143 110]
  [ 60 141 108]
  [ 58 139 106]]

 [[ 17  47  28]
  [ 19  49  30]
  [ 25  53  34]
  ...
  [ 61 142 109]
  [ 60 141 108]
  [ 57 138 105]]

 [[ 17  47  28]
  [ 19  49  30]
  [ 24  52  33]
  ...
  [ 65 146 113]
  [ 63 144 111]
  [ 61 142 109]]

 ...
```

합성곱 필터와 이미지

-1	-1	-1
-1	8	-1
-1	-1	-1

```
[[[ 25  55  36]
  [ 23  53  34]
  [ 24  52  33]
  ...
  [ 62 143 110]
  [ 60 141 108]
  [ 58 139 106]]

 [[ 17  47  28]
  [ 19  49  30]
  [ 25  53  34]
  ...
  [ 61 142 109]
  [ 60 141 108]
  [ 57 138 105]]

 [[ 17  47  28]
  [ 19  49  30]
  [ 24  52  33]
  ...
  [ 65 146 113]
  [ 63 144 111]
  [ 61 142 109]]

 ...
```

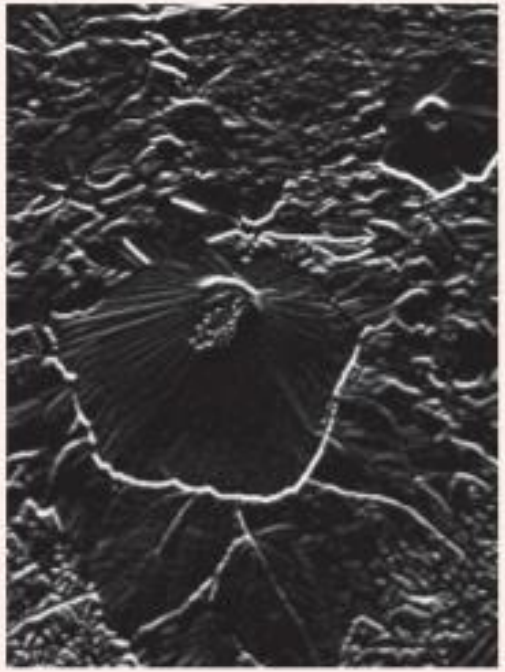
합성곱 필터와 이미지

1	0	-1
2	0	-2
1	0	-1

```
[[[ 25  55  36]
  [ 23  53  34]
  [ 24  52  33]
  ...
  [ 62 143 110]
  [ 60 141 108]
  [ 58 139 106]]

 [[ 17  47  28]
  [ 19  49  30]
  [ 25  53  34]
  ...
  [ 61 142 109]
  [ 60 141 108]
  [ 57 138 105]]

 [[ 17  47  28]
  [ 19  49  30]
  [ 24  52  33]
  ...
  [ 65 146 113]
  [ 63 144 111]
  [ 61 142 109]]

 ...
```

합성곱 필터와 이미지

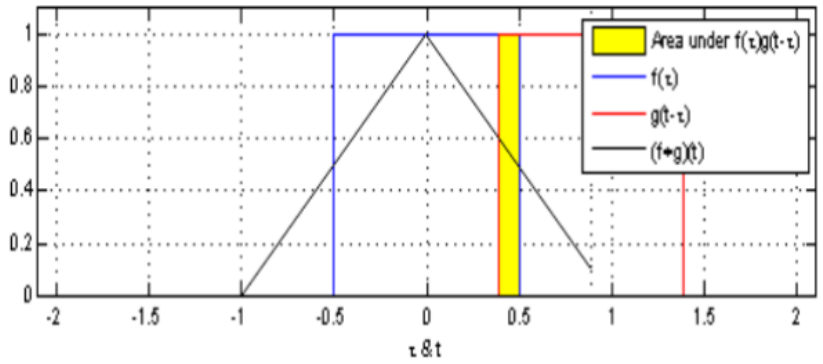
1	2	1
0	0	0
−1	−2	−1

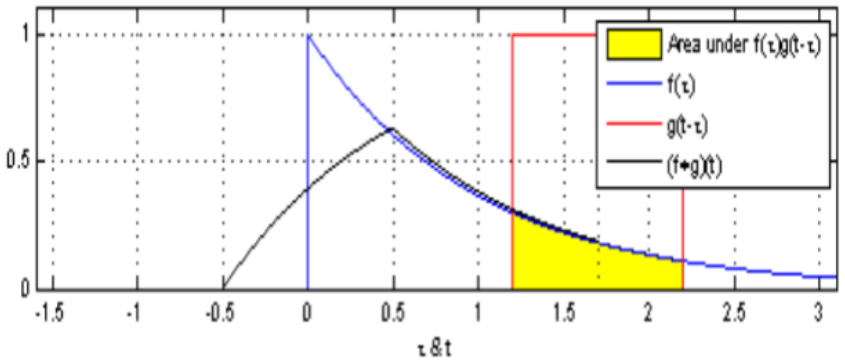


```
[[[ 25  55  36]
  [ 23  53  34]
  [ 24  52  33]
  ...
  [ 62 143 110]
  [ 60 141 108]
  [ 58 139 106]]

 [[ 17  47  28]
  [ 19  49  30]
  [ 25  53  34]
  ...
  [ 61 142 109]
  [ 60 141 108]
  [ 57 138 105]]

 [[ 17  47  28]
  [ 19  49  30]
  [ 24  52  33]
  ...
  [ 65 146 113]
  [ 63 144 111]
  [ 61 142 109]]

 ...
```

합성곱 연산

- f는 입력 데이터, g는 필터, 합성곱 결과는 $f \circ g$
  - 연속형 : $f \circ g(t) = \int_{-\infty}^{\infty} f(x)g(t-x)dx$
  - 이산형 : $f \circ g(i) = \sum_{j=-\infty}^{\infty} f(j)g(i-j)$





출처 : https://en.wikipedia.org/wiki/Convolution

패딩

- 합성곱 필터를 연속 적용 → 입력 데이터의 크기 점점 축소

- 입력 데이터 크기 = 출력 데이터 크기
  → 입력 데이터값 주변에 데이터를 추가 : 패딩(padding)

패딩

- 입력 데이터 값의 주변에 0을 추가 → 입력 데이터의 크기를 크게 함
  → 가장자리 데이터의 정보 손실을 막을 수 있음
- 4×4 입력 데이터 + 패딩 → 6×6로 바꾼 후 3×3 필터 적용
  → 입력 데이터와 같은 크기인 4×4 출력 데이터

합성곱 필터와 패딩

입력 데이터

0	0	0	0	0	0	0
0	1	1	2	2	0	0
0	2	2	1	1	0	0
0	0	0	1	1	2	0
0	1	1	0	0	2	0
0	2	2	0	0	1	0
0	0	0	0	0	0	0

필터

1	1	1
0	0	0
1	1	1

출력 데이터

4	5	4	2	1
2	5	7	8	5
6	7	5	4	3
4	5	4	5	4
2	2	1	2	2

패딩

- $n \times n$ 입력 데이터 + p개 패딩 : $(n+2p) \times (n+2p)$
  → $f \times f$ 필터 적용
  → 출력 데이터 : $(n+2p-f+1) \times (n+2p-f+1)$
  - 입·출력층이 같은 크기 : $(n+2p-f+1) = n$

  → 패딩 크기 : $p = \frac{(f-1)}{2}$

  (예) 3×3 필터 적용시 패딩 크기 $(3-1)/2 = 1$

스트라이드

- 스트라이드(stride) : 필터 이동 칸 수
  - 스트라이드 2로 지정 : 두 칸씩 옮겨가며 합성곱 연산
  - 스트라이드 값 s → 출력 데이터 크기는 $\frac{(n+2p-f)}{s}+1$
    - 스트라이드 값이 크면 출력 데이터의 크기는 축소

- 스트라이드 2 → $(5+2-3)/2+1=3$,
  - 출력 데이터의 크기 : 3×3

스트라이드

입력 데이터

0	0	0	0	0	0	0
0	1	1	2	2	0	0
0	2	2	1	1	0	0
0	0	0	1	1	2	0
0	1	1	0	0	2	0
0	2	2	0	0	1	0
0	0	0	0	0	0	0

필터

1	1	1
0	0	0
1	1	1

출력 데이터

2	4	1
6	5	3
2	1	2

컬러 이미지에 대한 합성곱 연산

- 컬러 이미지 데이터는 R, G, B 3개의 행렬
  - 필터도 R, G, B별로 다르게 적용 : 필터의 수가 3개

컬러 이미지에 대한 합성곱 연산

입력 데이터

0	0	0	0	0	0	0
0	1	1	2	3	0	0
0	3	2	1	1	0	0
0	0	0	1	1	2	0
0	1	1	0	0	3	0
0	2	3	0	0	1	0
0	0	0	0	0	0	0

필터

1	1	1
0	0	0
1	1	1

출력 데이터

5	6	4	2	1
2	5	8	9	6
7	8	5	5	4
5	6	5	5	4
2	2	1	3	3

컬러 이미지에 대한 합성곱 연산

입력 데이터

0	0	0	0	0	0	0
0	1	2	3	4	0	0
0	0	4	3	2	1	0
0	1	2	3	4	0	0
0	2	3	3	4	0	0
0	0	4	3	2	1	0
0	0	0	0	0	0	0

필터

1	0	1
1	0	1
1	0	1

출력 데이터

6	7	12	7	6
8	11	18	10	10
9	12	19	10	10
9	12	19	10	10
7	8	13	7	6

입력 데이터

0	0	0	0	0	0	0
0	2	3	4	3	2	0
0	4	3	2	0	1	0
0	1	3	5	7	9	0
0	2	3	5	6	7	0
0	1	0	2	1	0	0
0	0	0	0	0	0	0

필터

1	0	0
0	1	0
0	0	1

출력 데이터

5	12	9	9	3
9	18	19	23	10
9	19	22	29	13
6	16	20	28	14
3	10	10	14	7

최종 출력 데이터

16	25	25	18	10
19	34	45	42	26
25	39	46	44	27
20	34	44	43	28
12	20	24	24	16

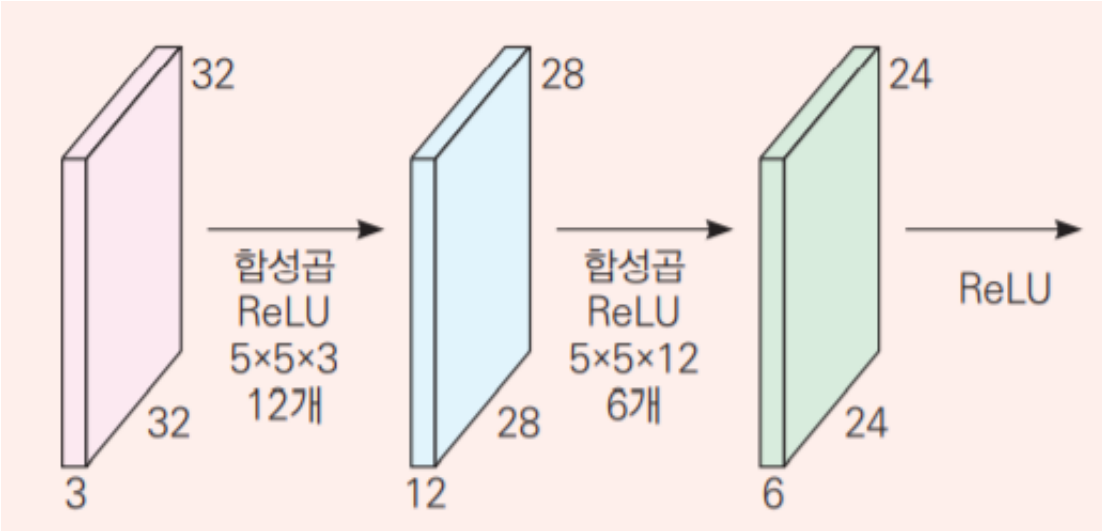
컬러 이미지에 대한 합성곱 연산

- 컬러 이미지 데이터의 합성곱 연산을 통해 결과물의 크기를 계산
  - $32 \times 32 \times 3$의 컬러 이미지 → 5×5 필터 3개 적용, 스트라이드 1, 패딩 없음

  → $(32 - 5 + 1) \times (32 - 5 + 1) \times 3 = 28 \times 28 \times 3$

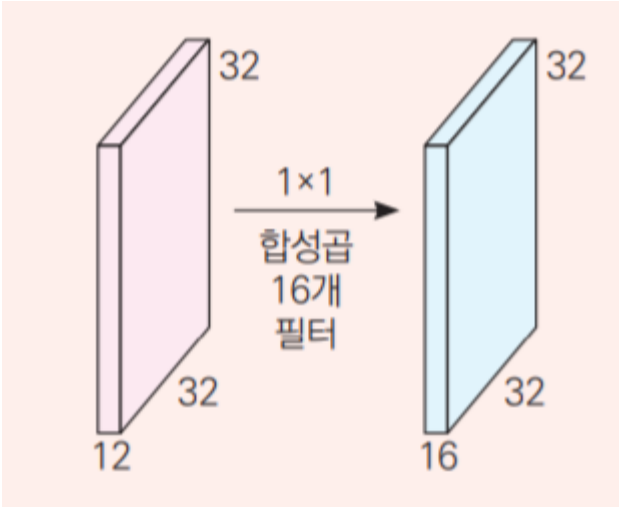
컬러 이미지에 대한 합성곱 연산

- $32 \times 32 \times 3$ 입력 이미지 → $5 \times 5 \times 3$ 12개 합성곱 필터
  → $28 \times 28 \times 12$ → ReLU 함수
  → $5 \times 5 \times 12$의 6개 합성곱 필터 → $24 \times 24 \times 6$의 출력
  → ReLU 함수 적용



1X1 합성곱 연산

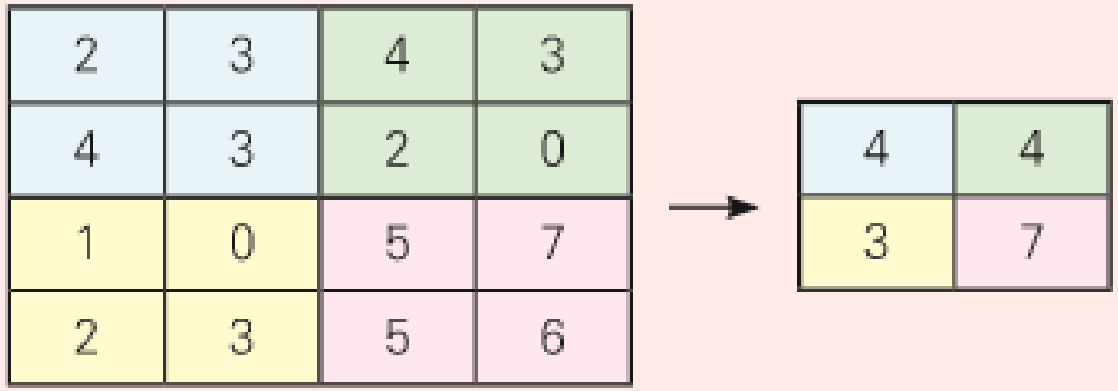
- 1×1 합성곱 연산 : 채널을 묶어주는 효과
  - $32 \times 32 \times 12$ 입력 이미지
    $\rightarrow 1 \times 1 \times 12$ 필터 16개로 합성곱 연산
    $\rightarrow 32 \times 32 \times 16$ 이미지
  - 합성곱 연산의 계산량을 줄이는데 유용



풀링

- 풀링(pooling) : 이미지 크기 줄이는 연산
  → 합성곱 연산 뒤 이루어짐
- 4×4 입력 이미지를 2×2 크기로 4개로 나누고 2×2 풀링 적용
  - 최대 풀링(max pooling) : 4개 구역에서 뽑은 최댓값으로 2×2 이미지 → 합성곱 신경망에서 주로 이용
  - 평균 풀링(average pooling) : 구역별 평균

최대 풀링



풀링

- 풀링의 장점
  - 이미지 이동, 왜곡에도 그 값이 크게 변하지 않음
  - 학습 없이 이루어져 계산량이 적음
  - 이미지의 크기를 작게 하여 과대적합을 막을 수 있음

합성곱 연산의 특성

- 희소성 : 뉴런을 전부 연결하지 않고, 이웃 뉴런들끼리 연결하고 가중치를 공유하는 희소한 연결 → 모형의 복잡도를 낮추어 과대적합 문제 해소
- 이동 등변성 : 입력 데이터의 이동만큼 이동
- 국지 연결성 : 이미지 데이터의 한 영역에서 보면 각 필터를 적용한 값들은 각각 달라져도 그 위치는 같음

학습정리

- ✓ 컴퓨터 비전은 디지털 이미지에서 유용한 정보를 자동 추출, 분석, 이해하는 것이다.
- ✓ 디지털 이미지는 고차원 입력벡터로, 완전연결 신경망 모형을 이용할 때 과적합 문제가 발생한다.

학습정리

- ✓ 합성곱 연산은 가중치 공유, 국소 연결이 가능한 연산이다.
- ✓ 대표적 초기 합성곱 신경망으로는 LeNet-5가 있으며, 이후 AlexNet 등은 딥러닝 시대를 열었다.

딥러닝의 통계적이해

다음시간안내

7강.
합성곱신경망의 기초(2)