



충북대학교 전자정보대학  
소프트웨어학부

# CNN 모델

# Convolutional Neural Network Models

이건명

충북대학교 소프트웨어학부

인공지능 : 튜링 테스트에서 딥러닝까지

# 학습 내용

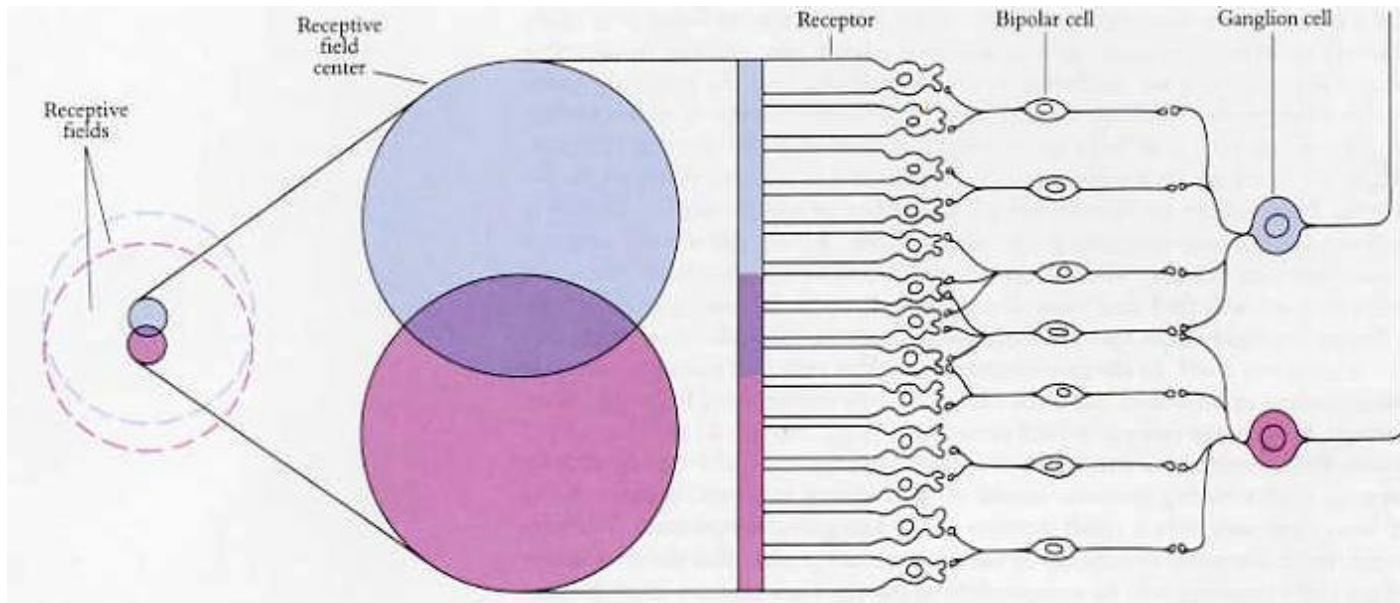
---

- **컨볼루션 신경망의 구조에 대해서 알아본다.**
- **컨볼루션 신경망에서 사용되는 컨볼루션, 풀링 등의 연산에 대해 알아본다.**

# 1. 컨볼루션 신경망

## ❖ 컨볼루션 신경망 (convolutional neural network, CNN)

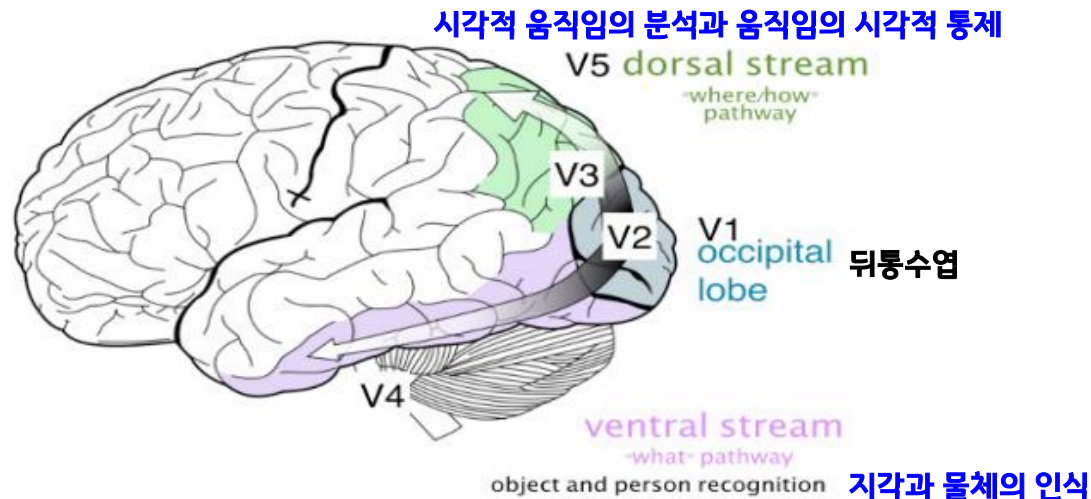
- 동물의 시각피질(visual cortex, 視覺皮質)의 구조에서 영감을 받아 만들어진 딥러닝 신경망 모델
  - 시각피질의 신경세포
    - 시야 내의 특정 영역에 대한 자극만 수용
      - » 수용장(receptive field, 受容場)
    - 해당 영역의 특정 특징에 대해서만 반응



# 컨볼루션 신경망

## ❖ 컨볼루션 신경망(convolutional neural network, CNN) – cont.

- 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

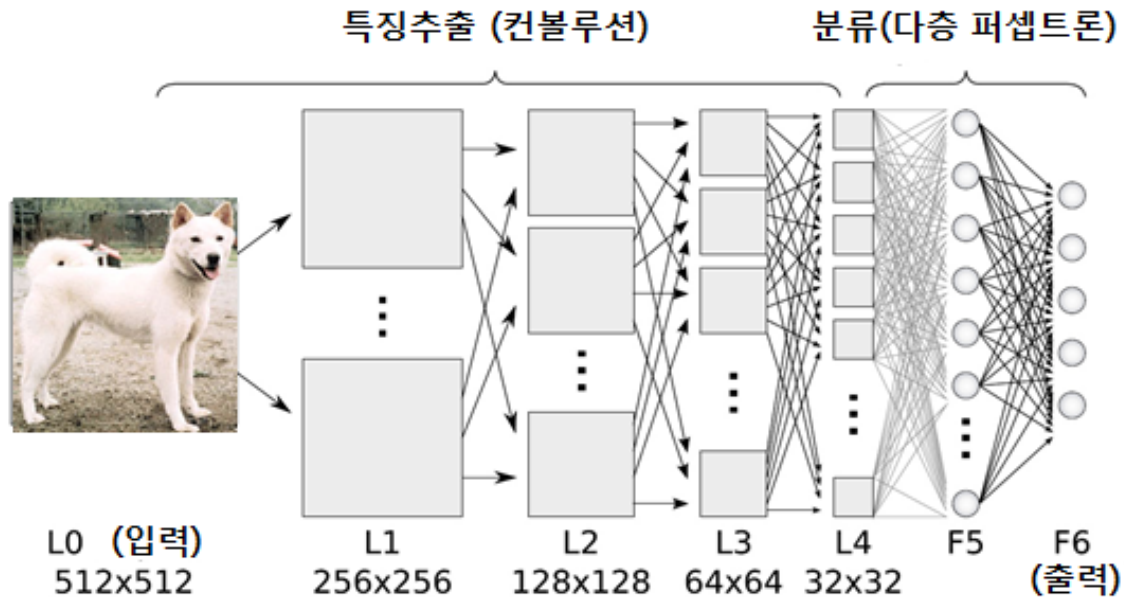


- 동물의 계층적 특징 추출과 시각인식 체계를 참조하여 만들어진 모델

# 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



## 2. 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	10	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

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

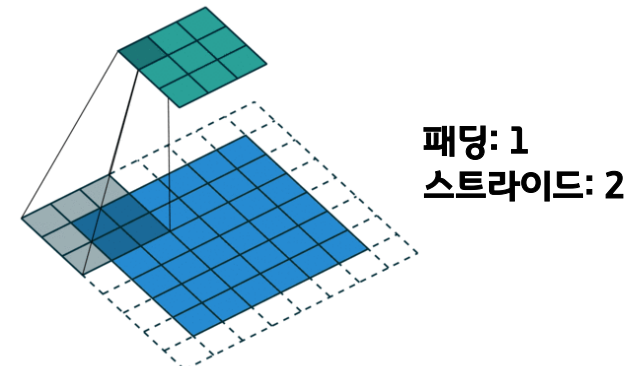
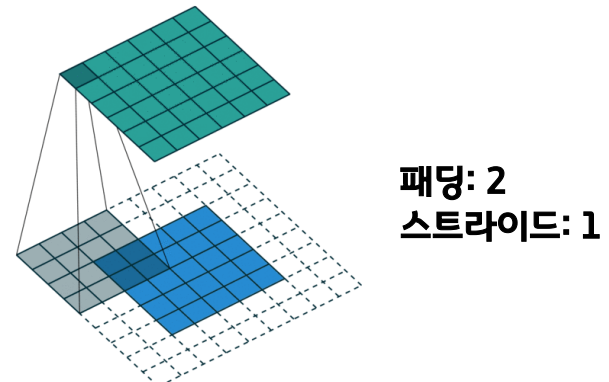
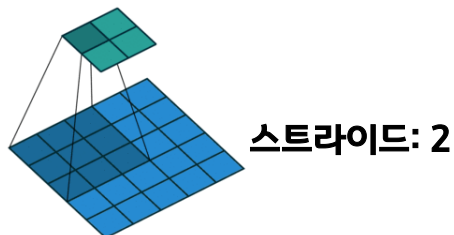
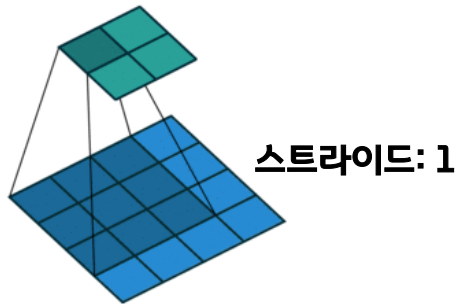
# 컨볼루션

## ❖ 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





# [실습] 컨볼루션

```
import numpy as np
```

```
def Conv2D(X, W, w0, p=(0,0), s=(1,1)):
    n1 = X.shape[0] + 2*p[0] # 패딩 반영
    n2 = X.shape[1] + 2*p[1]
    X_p = np.zeros(shape=(n1,n2))
    X_p[p[0]:p[0]+X.shape[0], p[1]:p[1]+X.shape[1]] = X # 입력 X 복사
    res = []
    for i in range(0, int((X_p.shape[0] - W.shape[0])/s[0])+1, s[0]):
        res.append([ ])
        for j in range(0, int((X_p.shape[1] - W.shape[1])/s[1])+1, s[1]):
            X_s = X_p[i:i+W.shape[0], j:j+W.shape[1]] # 컨볼루션 영역
            res[-1].append(np.sum(X_s * W) + w0) # 컨볼루션
    return np.array(res))
```

```
X = np.array([[1,1,1,0,0], [0,1,1,1,0], [0,0,1,1,1], [0,0,1,1,0], [0,1,1,0,0]])
W = np.array([[1,0,1], [0,1,0], [1,0,1]])
w0 = 1
```

```
conv = Conv2D(X, W, w0, p=(0,0), s=(1,1))
print('X = ', X)
print('\nW = ', W)
print('\n컨볼루션 결과 p=(0,0), s=(1,1) \n', conv)
conv = Conv2D(X, W, w0, p=(1,1), s=(1,1))
print('\n컨볼루션 결과 p=(1,1), s=(1,1) \n', conv)
conv = Conv2D(X, W, w0, p=(1,1), s=(2,2))
print('\n컨볼루션 결과 p=(1,1), s=(2,2) \n', conv)
```

```
X =
[[1 1 1 0 0]
 [0 1 1 1 0]
 [0 0 1 1 1]
 [0 0 1 1 0]
 [0 1 1 0 0]]
```

```
W =
[[1 0 1]
 [0 1 0]
 [1 0 1]]
```

컨볼루션 결과  $p=(0,0), s=(1,1)$

```
[[5. 4. 5.]
 [3. 5. 4.]
 [3. 4. 5.]]
```

컨볼루션 결과  $p=(1,1), s=(1,1)$

```
[[3. 3. 4. 2. 2.]
 [2. 5. 4. 5. 2.]
 [2. 3. 5. 4. 4.]
 [2. 3. 4. 5. 2.]
 [1. 3. 3. 2. 2.]]
```

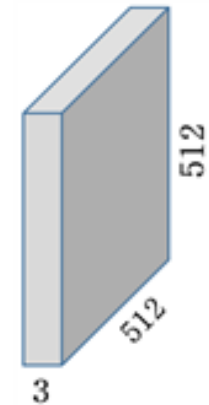
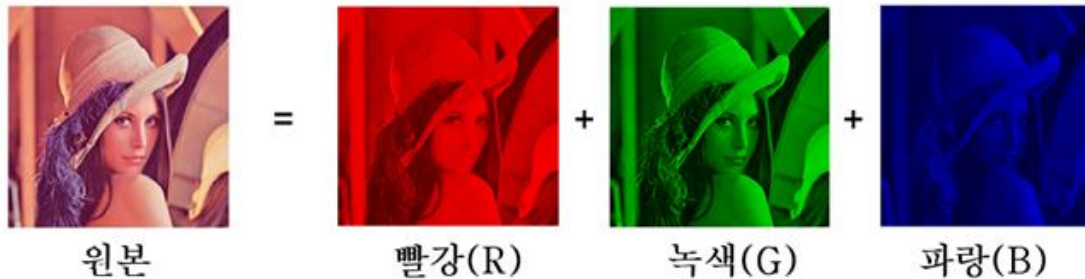
컨볼루션 결과  $p=(1,1), s=(2,2)$

```
[[3. 4.]
 [2. 5.]]
```

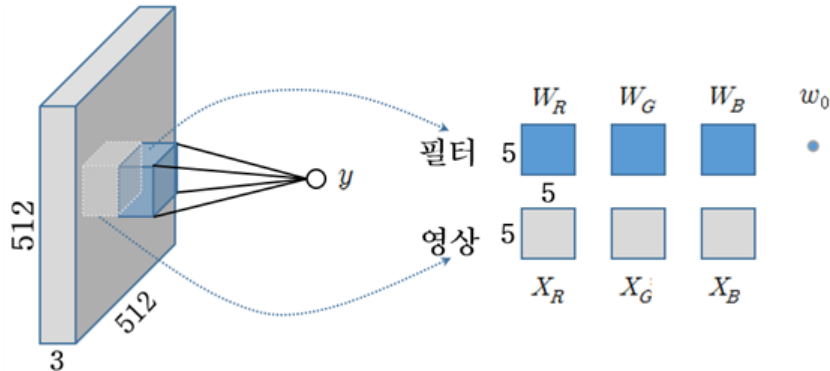
# 컨볼루션

## ❖ 컬러 영상의 컨볼루션

### ▪ 컬러 영상의 다차원 행렬 표현



### ▪ 컬러영상의 컨볼루션

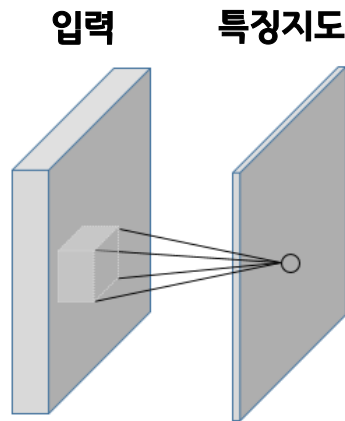


$$y = X_R^* W_R + X_G^* W_G + X_B^* W_B + w_0$$

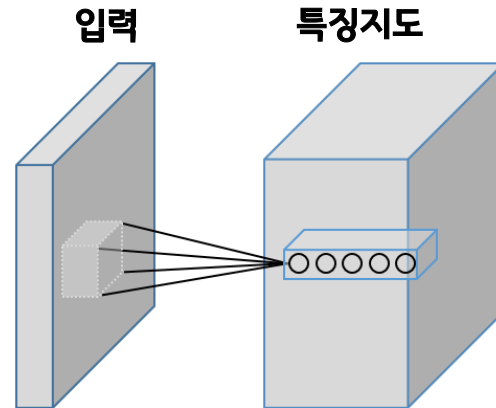
# 컨볼루션

## ❖ 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 k개의 2차원 특징지도 생성



1개 필터 적용




5개 필터 적용

# 3. 풀링

## ❖ 풀링(pooling)

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


1	1	2	3
4	6	6	8
3	1	1	0
1	2	2	4



6	8
3	4

- **평균값 풀링(average pooling)**
  - 블록 내의 원소들의 평균값을 대푯값으로 사용

1	1	2	3
4	6	6	8
3	1	1	0
1	2	2	4

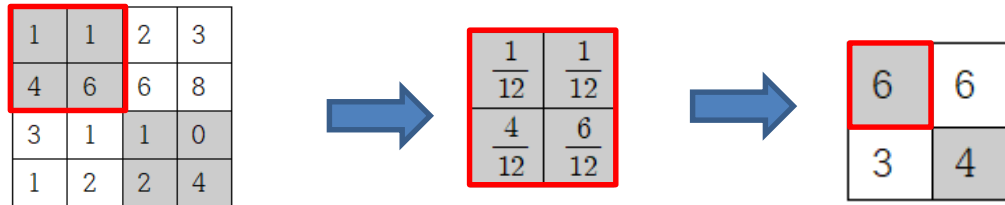


3	4.75
1.75	1.75

# 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

$$p_i = \frac{a_i}{\sum_{k \in R_j} a_k} \quad p_i : \text{블록 } R_j \text{에서 원소 } a_i \text{가 선택될 확률}$$

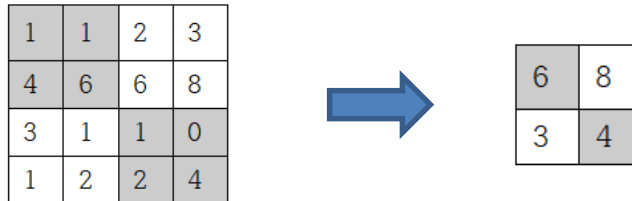
- 추론시 : 확률적 가중합 사용

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

# 폴링

## ❖ 폴링 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



# [실습] 풀링

```
import numpy as np
```

```
def maxPooling(mat, K, L):
```

```
    M, N = mat.shape
```

```
    MK = M // K
```

```
    NL = N // L
```

```
    pmat = mat[:MK*K, :NL*L].reshape(MK, K, NL, L).max(axis=(1, 3))
```

```
    return pmat
```

```
mat = np.array([[ 20, 200, -5, 23],  
                [-13, 134, 119, 100],  
                [120, 32, 49, 25],  
                [-120, 12, 9, 23]])
```

```
print(maxPooling(mat, 2,2))
```

```
[[200 119]  
 [120 49]]
```

# 4. 컨볼루션 신경망의 구조

## ❖ 컨볼루션 신경망의 구조

### ▪ 특징 추출을 위한 **컨볼루션 부분**

- 컨볼루션 연산을 하는 Conv층
- ReLU 연산을 하는 ReLU
- 풀링 연산 Pool(선택)]

} 반복

### ▪ 추출된 특징을 사용하여 **분류** 또는 **회귀**를 수행하는 **다층 퍼셉트론 부분**

- 전방향으로 전체 연결된(fully connected) FC층 반복
- 분류의 경우 마지막 층에 소프트맥스(softmax)을 하는 SM 연산 추가
  - 소프트맥스 연산 : 출력의 값이 0이상이면서 합은 1로 만듦

### ▪ 컨볼루션 신경망 구조의 예

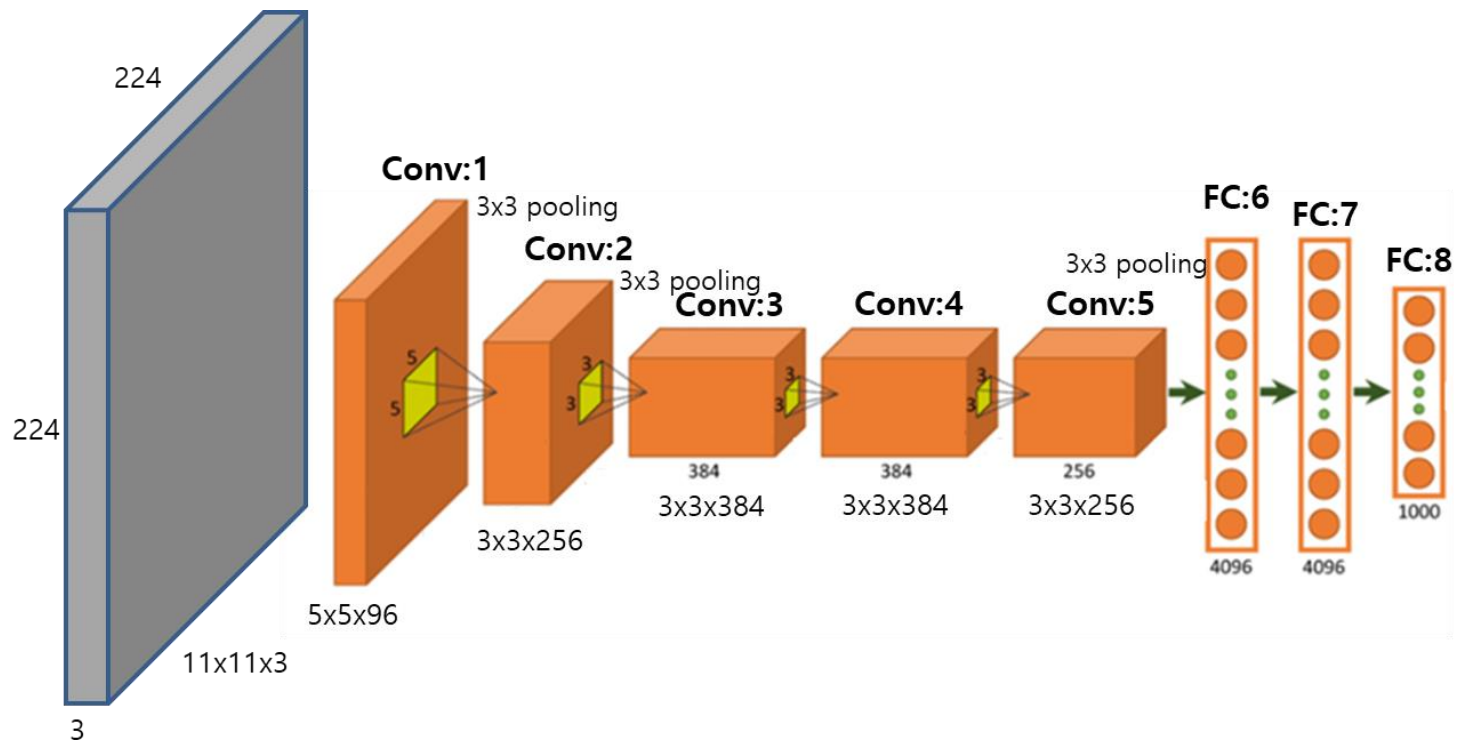
- Conv-ReLU-Pool-Conv-ReLU-Pool-Conv-ReLU-Pool-FC-SM
- Conv-Pool-Conv-Pool-Conv-FC-FC-SM
- Conv-Pool-Conv-Pool-Conv-Conv-Conv-Pool-FC-FC-SM
- Conv-ReLU-Pool-Conv-ReLU-Pool-Conv-ReLU-Pool-FC-FC-SM



# 컨볼루션 신경망의 구조

## ❖ 컨볼루션 신경망의 구조 예

- Conv:1-Pool:1-Conv:2-Pool:2-Conv:3-Conv:4-Conv:5-Pool:5-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					$224 \times 224 \times 3$ (=150,528)	
Conv:1	$11 \times 11 \times 3$	96	4	3	$55 \times 55 \times 96$ (=290,400)	$(11 \times 11 \times 3 + 1) \times 96$ (=34,944)
Pool:1	$3 \times 3$		2		$27 \times 27 \times 96$ (=69,984)	
Conv:2	$5 \times 5 \times 96$	256	1	2	$27 \times 27 \times 256$ (=186,624)	$(5 \times 5 \times 96 + 1) \times 256$ (=614,656)
Pool:2	$3 \times 3$		2		$13 \times 13 \times 256$ (=43,264)	
Conv:3	$3 \times 3 \times 256$	384	1	1	$13 \times 13 \times 384$ (=64,896)	$(3 \times 3 \times 256 + 1) \times 384$ (=885,120)
Conv:4	$3 \times 3 \times 384$	384	1	1	$13 \times 13 \times 384$ (=64,896)	$(3 \times 3 \times 384 + 1) \times 384$ (=1,327,488)
Conv:5	$3 \times 3 \times 384$	256	1	1	$13 \times 13 \times 256$ (=43,264)	$(3 \times 3 \times 384 + 1) \times 256$ (=884,992)
Pool:5	$3 \times 3$	256	2		$6 \times 6 \times 256$ (=9,216)	
FC:6					4096	$6 \times 6 \times 256 \times 4096$ (=37,748,736)
FC:7					4096	$4096 \times 4096$ (=16,777,216)
FC:8					1000	$4096 \times 1000$ (=4,096,000)

### ● 가중치

개수 : **58,621,952**

메모리 요구량 :

4바이트 float 사용시  
249,476,608 바이트  
( $\approx$  237MB)

### ● 계산 결과저장

노드 개수: **781,736**

메모리 요구량:  $\approx$  3MB

# 가중치의 학습

## ❖ 미분 구현의 방법

### ▪ 수치적 미분(numerical differentiation)

- 작은  $h$ 을 사용하여  $f(x + h)$ 와  $f(x)$ 의 값을 계산하여 미분

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}.$$

### ▪ 기호적 미분(symbolic differentiation)

- 미분의 수식 이용

$$\frac{d}{dx} \left( \frac{x^2 \cos(x - 7)}{\sin(x)} \right) = x^2 \sin(7 - x) \csc(x) + x^2 (-\cos(7 - x)) \cot(x) \csc(x) + 2 x \cos(7 - x) \csc(x)$$

# 가중치의 학습

## ❖ 미분 구현의 방법 – cont.

### ▪ 자동 미분(automatic differentiation)

- 함수를 **기본 연산**(primitive)의 제어 흐름(control flow)로 나타내서, 각 기본 연산의 **도함수**를 사용하여 연쇄법칙에 의해 미분 값 계산

$$f(x_1, x_2) = x_1 x_2 + \sin x_1$$

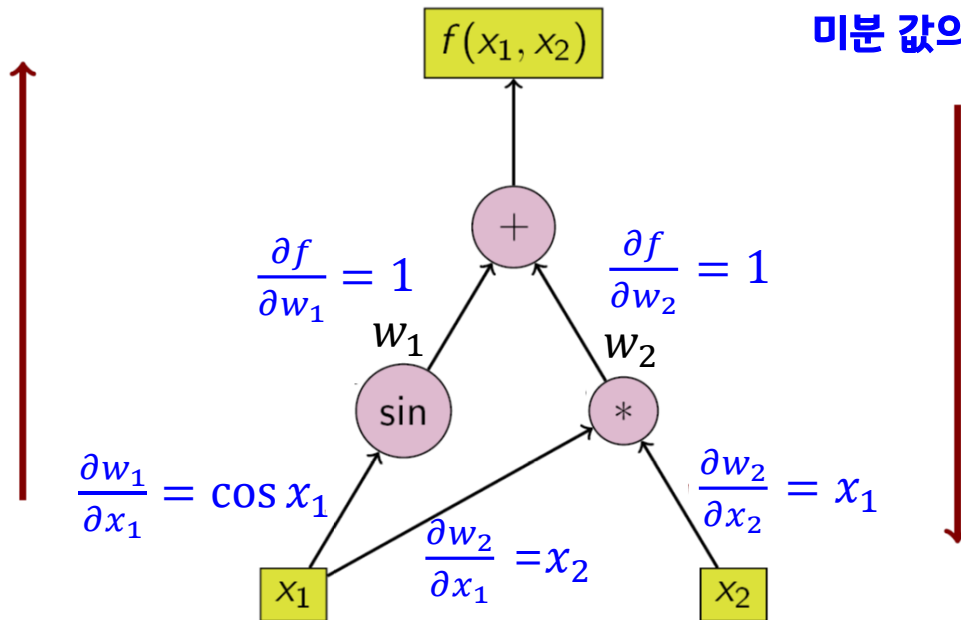
함수 값의 계산

$$f(x_1, x_2) = w_1 + w_2$$

$$w_1 = \sin x_1$$

$$w_2 = x_1 x_2$$

미분 값의 계산



$$\begin{aligned} \frac{\partial f}{\partial x_1} &= \frac{\partial f}{\partial w_1} \frac{\partial w_1}{\partial x_1} + \frac{\partial f}{\partial w_2} \frac{\partial w_2}{\partial x_1} \\ &= \cos x_1 + x_2 \end{aligned}$$

$$\begin{aligned} \frac{\partial f}{\partial x_2} &= \frac{\partial f}{\partial w_2} \frac{\partial w_2}{\partial x_2} \\ &= x_1 \end{aligned}$$

# Quiz

❖ 컨볼루션 연산에서 사용되는 작은 윈도우를 무엇이라고 하는가?

- ① 필터
- ② 풀링
- ③ 스트라이드
- ④ 패딩

❖ 컨볼루션 연산 후 출력 크기를 입력 크기와 동일하게 유지하기 위한 방법은 무엇인가?

- ① 스트라이드 증가
- ② 필터 크기 감소
- ③ 패딩 추가
- ④ 풀링 적용

❖ 컨볼루션 신경망에서 사용되는, 주로 최대값을 선택하는 다운샘플링 방법은 무엇인가?

- ① 필터링
- ② 스트라이딩
- ③ 패딩
- ④ 풀링

# Quiz

❖ 컨볼루션 연산에서 스트라이드는 무엇을 의미하는가?

- ① 필터의 크기
- ② 필터가 이동하는 간격
- ③ 출력 크기를 조절하기 위해 사용되는 기술
- ④ 필터의 개수

❖ RGB 이미지에 대해 3x3 컨볼루션 필터를 적용할 때, 필터의 깊이(채널 수)는?

- ① 1
- ② 2
- ③ 3
- ④ 4

❖ 5x5 입력에 3x3 필터를 스트라이드 1로 적용할 때, 출력 크기는?

- ① 5x5
- ② 4x4
- ③ 3x3
- ④ 2x2

# Quiz

## ❖ 컨볼루션 연산의 주된 목적은?

- ① 이미지의 차원을 늘리는 것
- ② 이미지의 주요 특징을 감지하는 것
- ③ 이미지의 밝기를 조절하는 것
- ④ 이미지의 해상도를 향상시키는 것

## ❖ 7x7 입력에 3x3 필터를 스트라이드 2로 적용하면 출력 크기는?

- ① 2x2
- ② 3x3
- ③ 5x5
- ④ 6x6

## ❖ 컨볼루션 신경망의 상위 계층에서 발견되는 특징은?

- ① 기본적인 엣지와 질감
- ② 복잡한 객체와 패턴
- ③ 픽셀의 밝기와 색상
- ④ 이미지의 해상도와 크기

# Quiz

## ❖ 컨볼루션 연산에서 '1x1 컨볼루션'이 사용되는 목적은?

- ① 이미지의 해상도를 향상시키기 위해
- ② 연산량을 줄이면서 채널 간의 상호 작용을 가능하게 하기 위해
- ③ 이미지의 밝기를 조절하기 위해
- ④ 필터의 크기를 조절하기 위해

## ❖ 특징지도의 깊이(채널수)는 무엇을 결정하는가?

- ① 입력 이미지의 크기
- ② 필터의 개수
- ③ 풀링 레이어의 종류
- ④ 스트라이드의 크기

## ❖ 특징지도에서 각 활성화는 무엇을 의미하나요?

- ① 필터의 크기
- ② 특정 특징에 대한 응답 정도
- ③ 네트워크의 학습률
- ④ 이미지의 차원 수



# Quiz

❖ CNN의 하위 계층의 특징지도는 주로 어떤 정보를 포착하는가?

- ① 객체의 추상적 표현
- ② 복잡한 패턴과 구조
- ③ 엣지와 질감
- ④ 전체적인 이미지 구조

❖ CNN의 상위 계층의 특징지도는 무엇을 표현하는 경향이 있는가?

- ① 값의 분포
- ② 전체 이미지의 색상
- ③ 이미지의 기본 질감
- ④ 이미지의 복잡한 패턴 및 객체 표현

❖ CNN에서 풀링 후에 특징지도의 크기는 어떻게 변하는가?

- ① 증가한다.
- ② 감소한다.
- ③ 동일하게 유지된다.
- ④ 두 배로 늘어난다.

# Quiz

## ❖ 풀링 레이어의 주요 역할은 무엇인가요?

- ① 특징지도의 깊이를 줄이는 것
- ② 특징지도의 공간적 차원을 감소시키는 것
- ③ 필터의 개수를 증가시키는 것
- ④ 네트워크의 학습률을 조절하는 것

## ❖ 최대값 풀링(max pooling) 연산이 수행하는 작업은 무엇인가요?

- ① 주어진 영역에서 최소값을 선택하는 것
- ② 주어진 영역에서 최대값을 선택하는 것
- ③ 주어진 영역의 평균값을 계산하는 것
- ④ 주어진 영역의 합계를 계산하는 것

## ❖ 평균값 풀링(average pooling)에서 주어진 영역의 출력 값은 어떻게 계산되나요?

- ① 영역의 합계
- ② 영역의 최대값
- ③ 영역의 최소값
- ④ 영역의 평균값

# Quiz

## ❖ 풀링 연산이 가중치를 포함하는가?

- ① 포함한다.
- ② 포함하지 않는다.
- ③ 네트워크의 깊이에 따라 다르다.
- ④ 필터의 크기에 따라 다르다.

## ❖ 풀링 레이어는 어떤 변화에 대해 불변성(invariance)을 제공하는가?

- ① 회전에 대한 불변성
- ② 스케일(확대/축소)에 대한 불변성
- ③ 이동에 대한 불변성
- ④ 색상에 대한 불변성

## ❖ 풀링 연산 후 특징지도의 크기는 어떻게 변하는가?

- ① 증가한다.
- ② 감소한다.
- ③ 동일하게 유지된다.
- ④ 두 배로 늘어난다.

# Quiz

## ❖ 자동미분의 핵심 목적은?

- ① 함수의 근사값을 찾는 것
- ② 함수의 최대값을 계산하는 것
- ③ 함수의 도함수를 계산하는 것
- ④ 함수의 적분을 계산하는 것

## ❖ 자동미분을 사용하는 주요 이유는?

- ① 근사값을 계산하기 위해
- ② 미분 계산의 정확성과 효율성을 높이기 위해
- ③ 함수의 적분을 자동으로 계산하기 위해
- ④ 모든 함수에 대한 해석적 해를 찾기 위해

## ❖ 자동미분이 계산 그래프를 사용하는 이유는?

- ① 그래프의 각 노드에서 함수의 최대값을 계산하기 위해
- ② 그래프의 각 노드에서 적분을 수행하기 위해
- ③ 그래프의 각 노드에서 연산과 그 도함수를 표현하기 위해
- ④ 그래프의 구조를 시각화하기 위해

# Quiz

❖ 자동미분에서 중간 변수의 도함수를 계산하기 위해 사용되는 연쇄규칙의 적용방법은?

- ① 중간 변수의 도함수를 모두 더한다.
- ② 중간 변수의 도함수를 모두 곱한다.
- ③ 중간 변수의 도함수의 최대값을 선택한다.
- ④ 중간 변수의 도함수의 평균을 계산한다.

❖ 자동미분이 신경망 학습에서 중요한 이유는?

- ① 가중치 초기화를 자동으로 수행하기 위해
- ② 신경망의 구조를 최적화하기 위해
- ③ 신경망의 오차 역전파를 효율적으로 계산하기 위해
- ④ 신경망의 활성화 함수를 최적화하기 위해

❖ 자동미분의 순방향 모드(forward mode)는 무엇을 계산하는가?

- ① 각 입력에 대한 모든 출력의 미분
- ② 각 출력에 대한 모든 입력의 미분
- ③ 함수의 값
- ④ 함수의 최대값

# Quiz

❖ 채널 크기가 4인 입력에 대해서 3x3 커널이 5개 적용될 때 학습될 파라미터의 개수는?

- ①  $3 \times 3 \times 4 \times 5$
- ②  $3 \times 3 \times 4 \times 5 + 5$
- ③  $4 \times 4 \times 3 \times 5$
- ④  $3 \times 3 \times 5 + 4$

❖ 입력 노드가 10개인 노드 20개의 완전연결층이 있을 때 파라미터의 개수는?

- ①  $10 + 20$
- ②  $10 \times 20$
- ③  $10 \times 20 + 10$
- ④  $10 \times 20 + 20$

❖ 4x4 크기의 윈도우를 스트라이드 2로 적용하는 풀링 연산에 있는 파라미터 개수는?

- ① 0
- ② 2
- ③  $4 \times 4$
- ④  $4 \times 4 \times 2$

# Quiz

❖ CNN은 주로 어떤 종류의 데이터에 사용되는가?

- ① 텍스트 데이터
- ② 시계열 데이터
- ③ 이미지 데이터
- ④ 표 데이터

❖ CNN에서 패딩(padding)의 주요 목적은?

- ① 과적합을 방지하기 위함
- ② 출력 특징지도의 크기를 조절하기 위함
- ③ 필터의 수를 증가시키기 위함
- ④ 학습률을 조절하기 위함

❖ CNN의 컨볼루션 층 다음에 주로 사용되는 활성화 함수는?

- ① Sigmoid
- ② Softmax
- ③ ReLU
- ④ Tanh

# Quiz

❖ CNN은 주로 어떤 종류의 문제에 사용되지 않는가?

- ① 이미지 분류
- ② 시각적 객체 탐지
- ③ 자연어 처리
- ④ 이미지 세그멘테이션

❖ CNN 구조에서 완전연결층(fully connected layer)은 주로 어떤 목적으로 사용되는가?

- ① 이미지의 특징을 추출하기 위함
- ② 모델의 복잡성을 줄이기 위함
- ③ 네트워크의 깊이를 높이기 위함
- ④ 최종 예측을 수행하기 위함

❖ CNN에서 사용되는 드롭아웃(dropout) 기법의 주요 목적은?

- ① 필터의 크기 조절하기
- ② 과적합을 방지하기
- ③ 학습 속도 향상시키기
- ④ 필터의 갯수를 줄이기



# Quiz

- ❖ CNN에서 입력 이미지의 채널 수와 필터의 채널 수는 어떻게 다른가?
  - ① 항상 다르다
  - ② 항상 같다
  - ③ 필터의 채널 수가 더 많다
  - ④ 입력 이미지의 채널 수가 더 많다
  
- ❖ CNN에서 최대값 풀링(max pooling)의 기능은?
  - ① 가장 큰 값을 선택하기
  - ② 평균 값을 선택하기
  - ③ 최소 값을 선택하기
  - ④ 모든 값을 합산하기
  
- ❖ CNN에서 사용되는 스트라이드(stride)는 무엇을 의미하는가?
  - ① 필터의 크기
  - ② 풀링 층의 종류
  - ③ 컨볼루션 필터가 움직이는 간격
  - ④ 네트워크의 깊이

# Quiz

❖ CNN에서 배치 크기는 무엇을 의미하는가?

- ① 학습률의 크기
- ② 한 번의 업데이트에 사용되는 샘플 수
- ③ 이미지의 해상도
- ④ 필터의 개수

❖ 어떤 연산이 CNN의 연산 부하 대부분을 차지하는가?

- ① 활성화 함수
- ② 풀링 연산
- ③ 컨볼루션 연산
- ④ 배치 정규화