인 공 지 능

[심층학습 기초 II]

jaekoo@kookmin.ac.kr

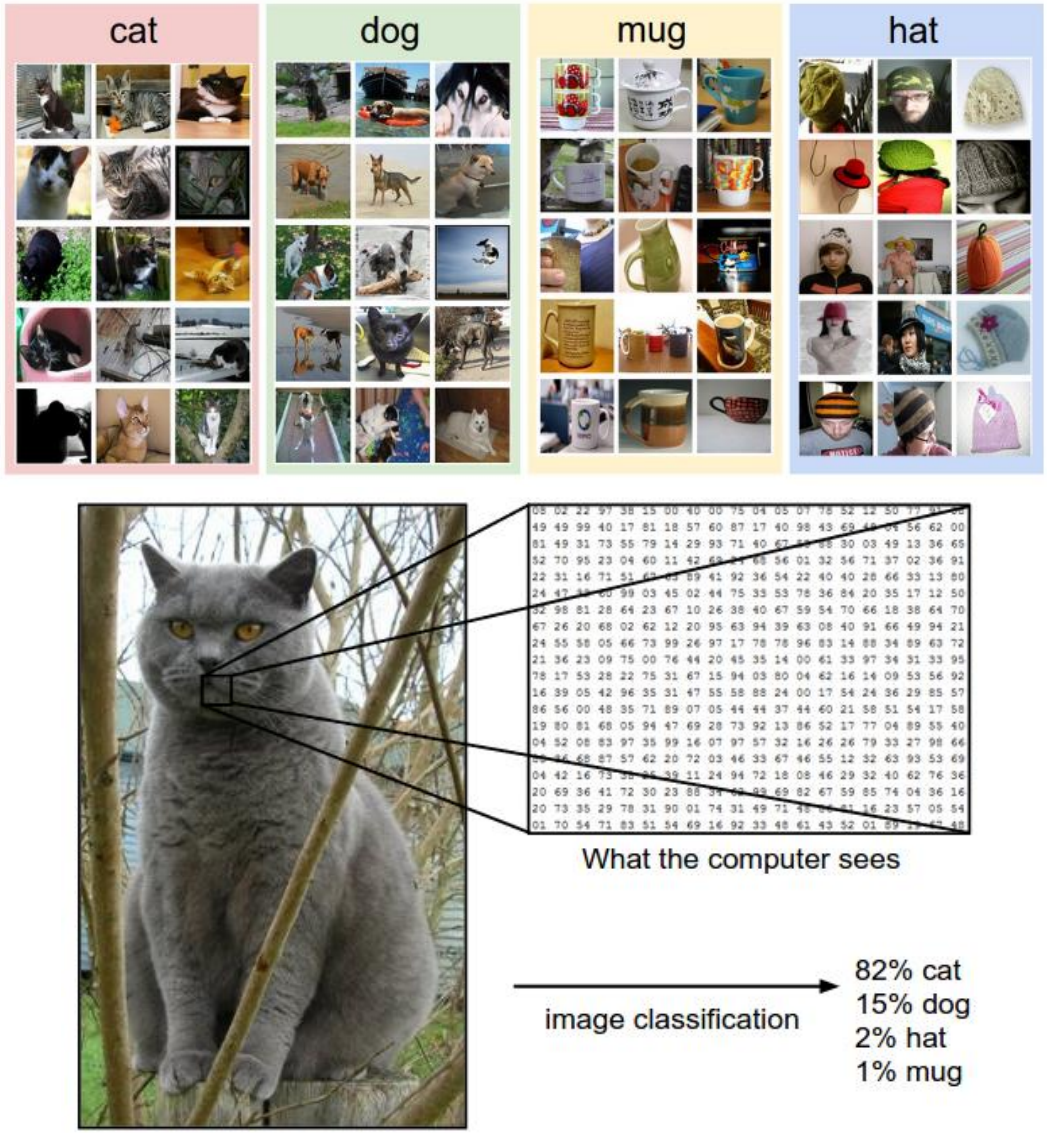
KMU 국민대학교
KOOKMIN UNIVERSITY

4.3 컨볼루션 (합성곱) 신경망CNN(convolutional neural network)

jaekoo@kookmin.ac.kr

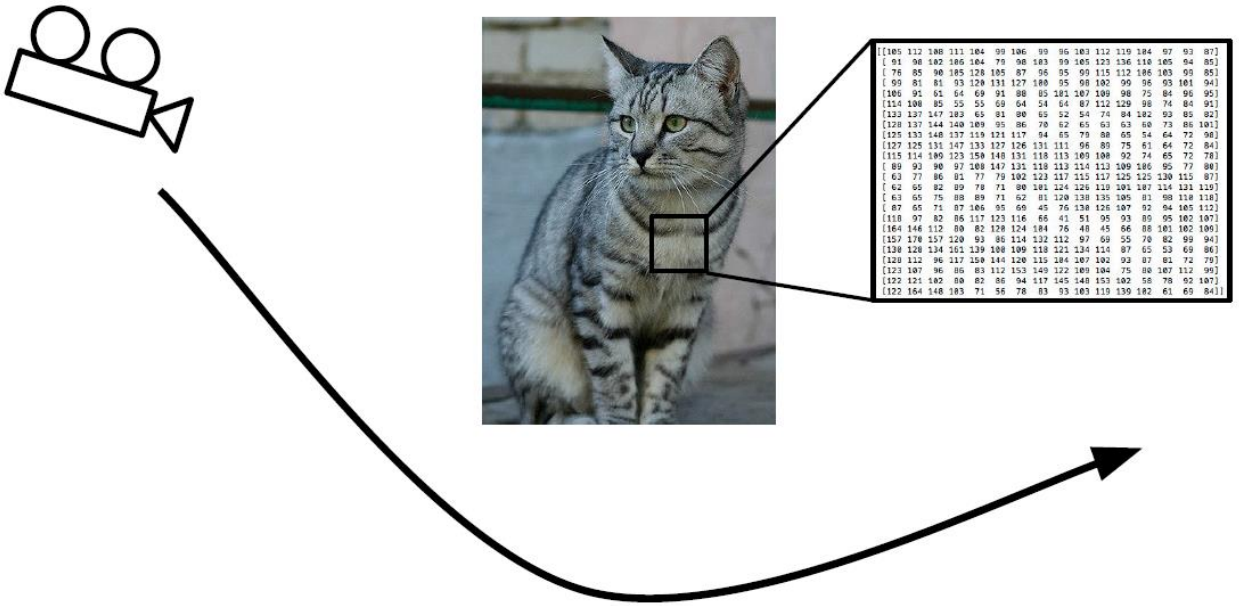
4.3 컨볼루션 (합성곱) 신경망

- 영상 인식의 예



4.3 컨볼루션 신경망

- 컴퓨터 비전의 어려운 점
  - 관점의 변화: 동일한 객체라도 영상을 찍는 카메라의 이동에 따라 모든 픽셀값이 변화됨



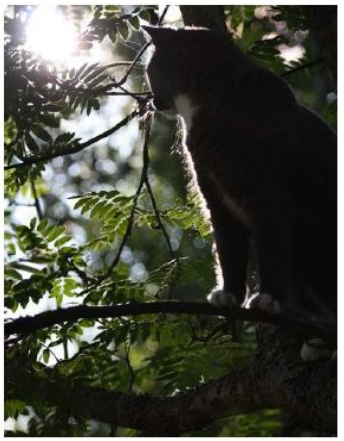
4.3 컨볼루션 신경망

- 컴퓨터 비전의 어려운 점
  - 경계색 (보호색)으로 배경과 구분이 어려운 경우

4.3 컨볼루션 신경망

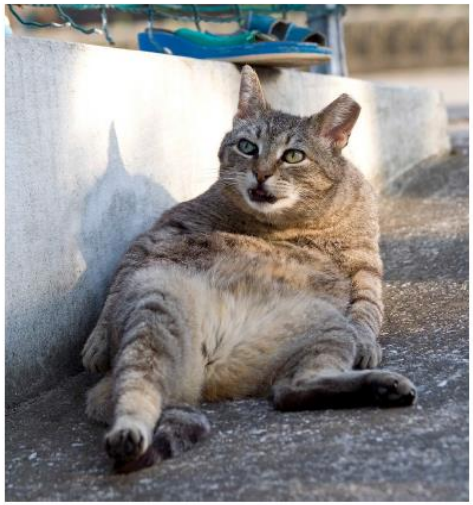
- 컴퓨터 비전의 어려운 점
  - 조명에 따른 변화

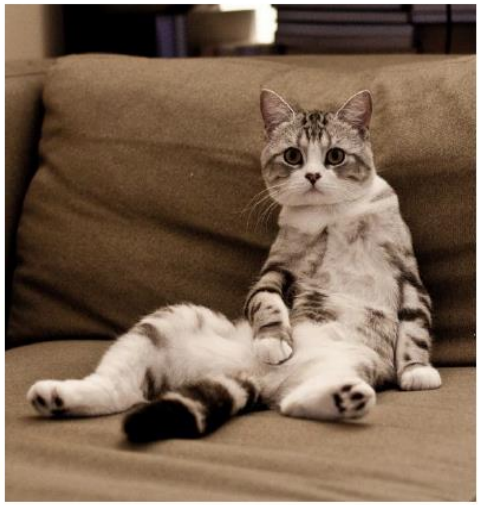


4.3 컨볼루션 신경망

- 컴퓨터 비전의 어려운 점
  - 기형적인 형태의 영상 존재



4.3 컨볼루션 신경망

- 컴퓨터 비전의 어려운 점
  - 일부가 가려진 영상 존재

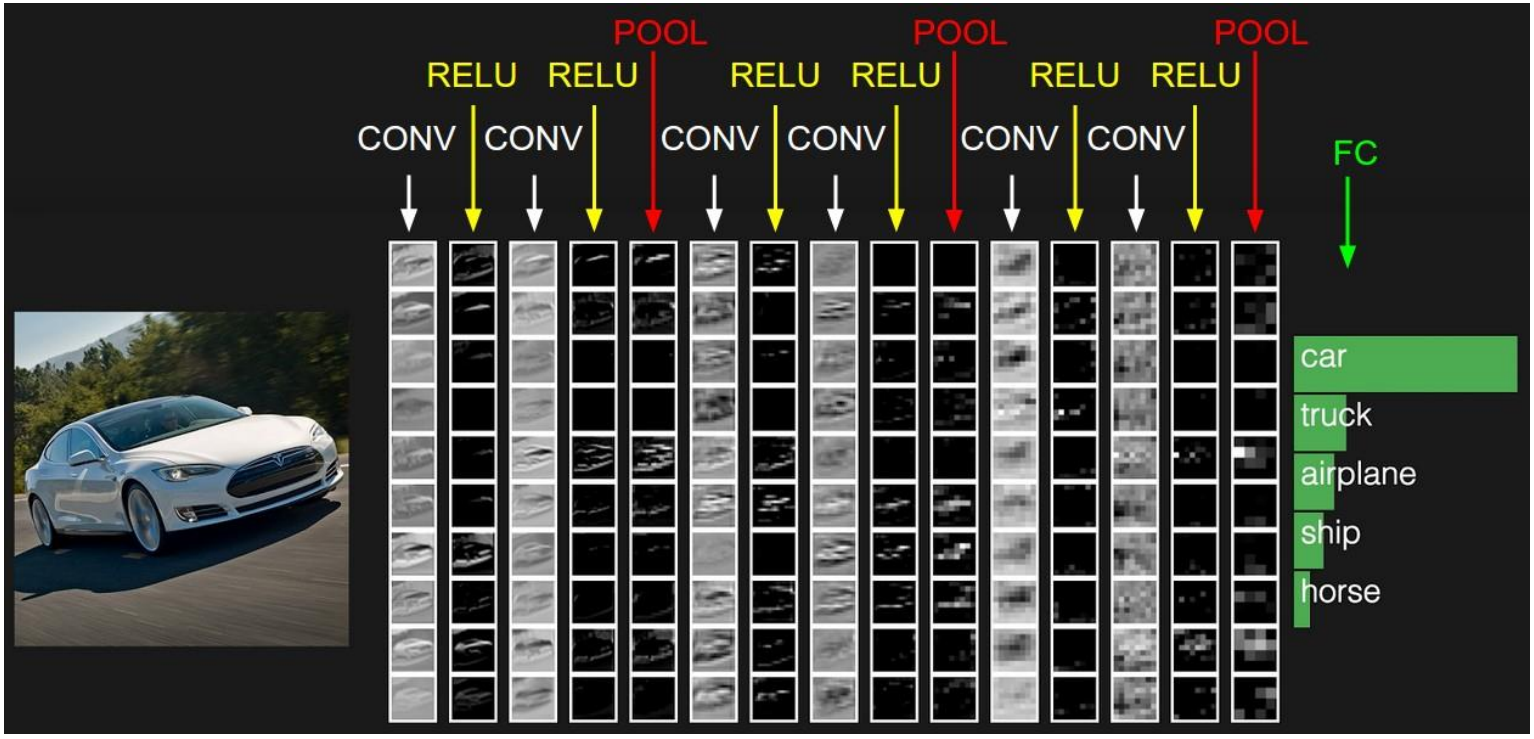


4.3 컨볼루션 신경망

- 컴퓨터 비전의 어려운 점
  - 같은 종류 간의 변화가 큼

4.3 컨볼루션 신경망

- 4.3.1 컨볼루션층 (CONV)
  - 선형함수인 컨볼루션과 비선형 함수인 활성함수의 조합
- 4.3.2 풀링층 (POOL)
  - 컨볼루션의 얻어진 특징을 통계적으로 압축
- 4.3.3 전체 구조
  - CONV-POOL-…-FC



4.3 컨볼루션 신경망

- 오늘날, 영상 분야에서 다양하게 활용됨

분류 classification

검색 retrieval

Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

검출 detection

Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

분할 segmentation

Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

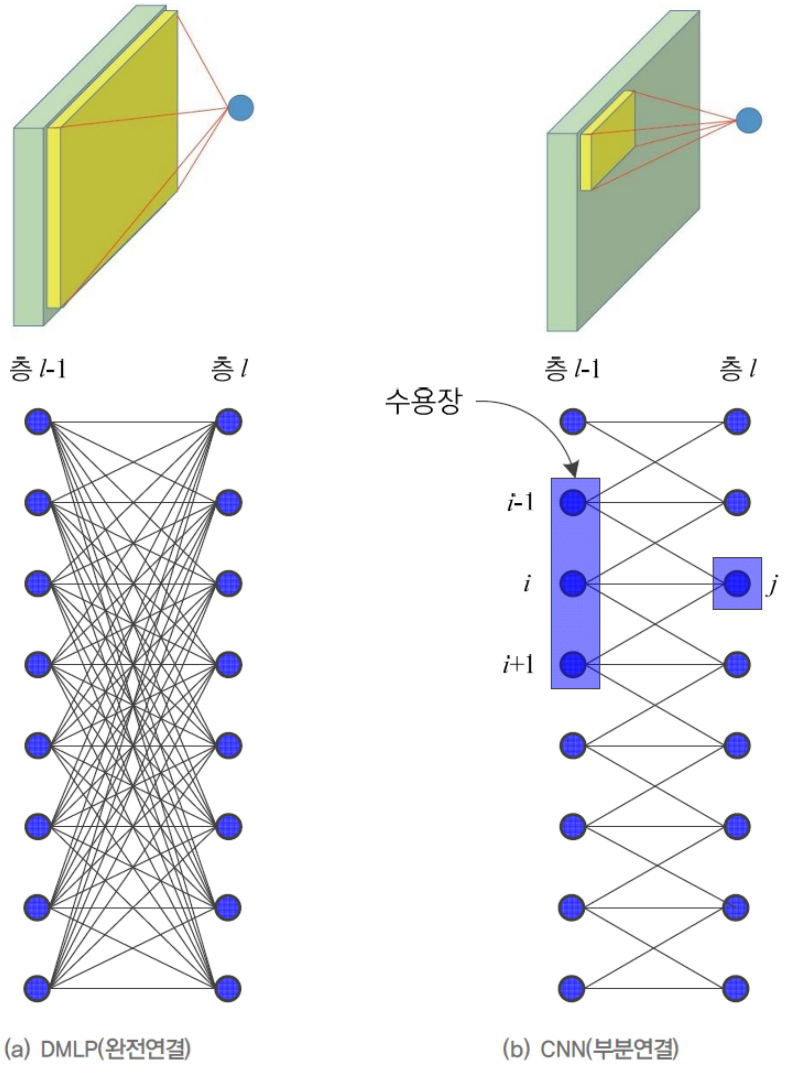
4.3 컨볼루션 신경망

DMLP와 CNN의 비교

- DMLP
  - 완전 연결fully connection 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 특징

- 격자grid 구조 (영상, 음성 등)를 갖는 데이터에 적합
- 수용장receptive field은 인간시각과 유사
- 가변 크기의 입력 처리 가능



(a) DMLP(완전연결) (b) CNN(부분연결)

그림 4-5 CNN의 부분연결과 수용장

4.3 컨볼루션 신경망

CNN의 완전 연결 신경망fully connected neural networks과 차별

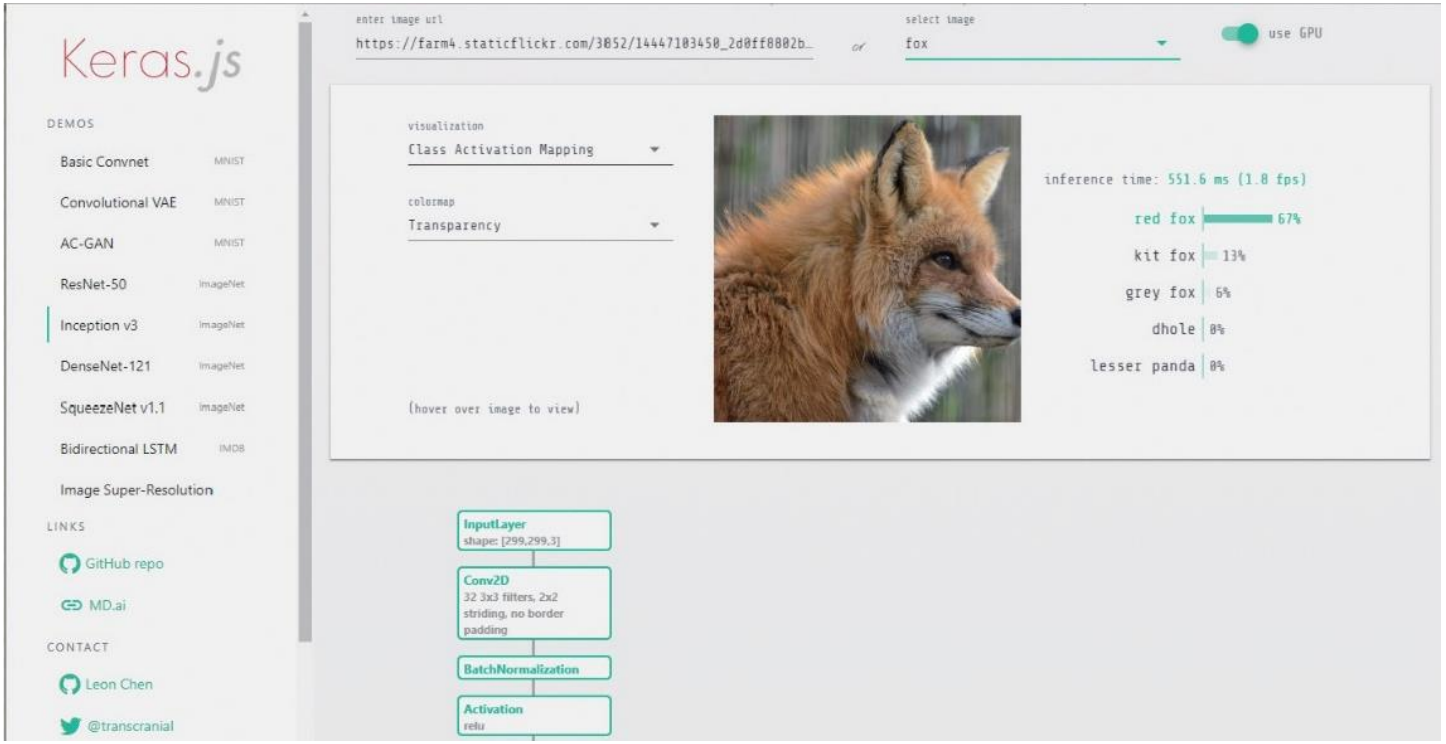
- [CONV] 학습에 의해 결정된 복수의 커널들 (혹은 필터들)에 대응되는 특징들을 추출하는 층
  - 각 층의 입출력의 특징형상 유지 (특징맵)
  - 영상의 공간 정보를 유지하면서 공간적으로 인접한 정보의 특징을 효과적으로 인식
  - 각 커널 (필터)은 매개변수를 공유함으로써 완전 연결 신경망 대비 학습 매개변수가 매우 적음

- [POOL] 추출된 영상의 특징을 요약하고 강화하는 층

- 가변 크기의 데이터 다루기
  - 완전연결신경망은 특징 벡터의 크기가 달라지면 연산 불가능
  - CNN은 가변 크기를 다룰 수 있는 강점
    - 컨볼루션층에서 보폭을 조정한다거나, 풀링층에서 커널이나 보폭을 조정하여 특징 맵 크기를 조절

4.3 컨볼루션 신경망

- 컨볼루션 신경망 예제
  - https://transcranial.github.io/keras-js/#/



4.3.1 컨볼루션층

컨볼루션 (합성곱)convolution 연산

- 컨볼루션은 해당하는 요소끼리 곱하고 결과를 모두 더하는 선형 연산
- 식 (4.10)과 식 (4.11)에서 u는 커널kernel (혹은 필터filter), z는 입력, s는 출력 (특징 맵feature map)
  - 영상에서 특징을 추출하기 위한 용도로 사용됨 (=공간 필터spatial filtering)

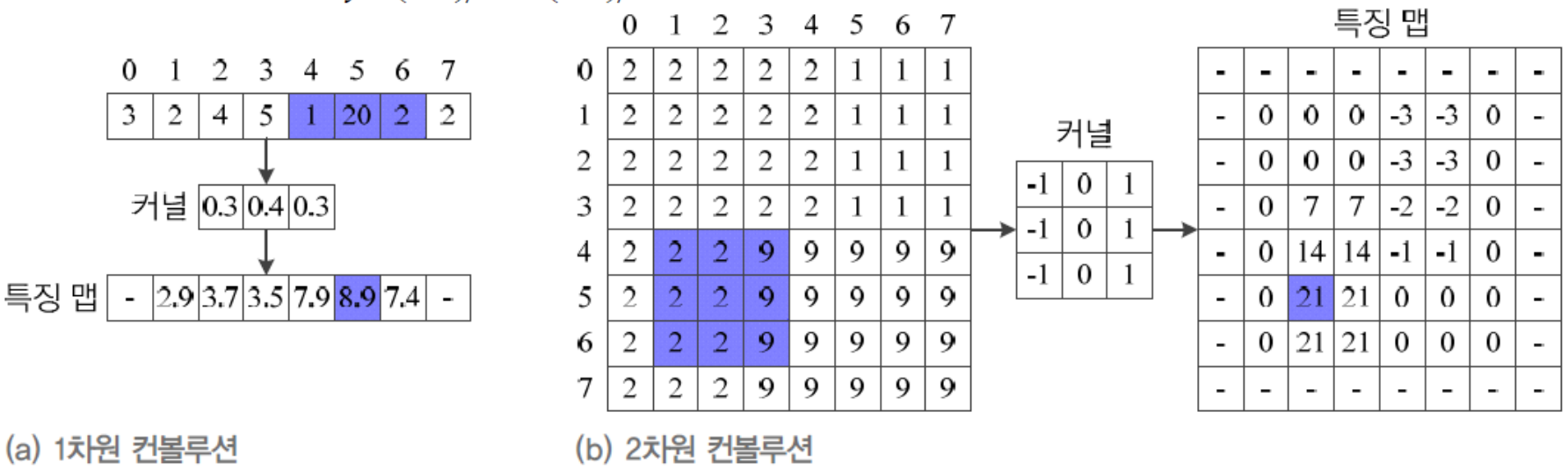
※필터: 무언가를 걸러내는 도구

$$s(i) = z \circledast u = \sum_{x=-(h-1)/2}^{(h-1)/2} z(i+x)u(x) \qquad (4.10)$$

← 1차원 입력

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

← 2차원 입력

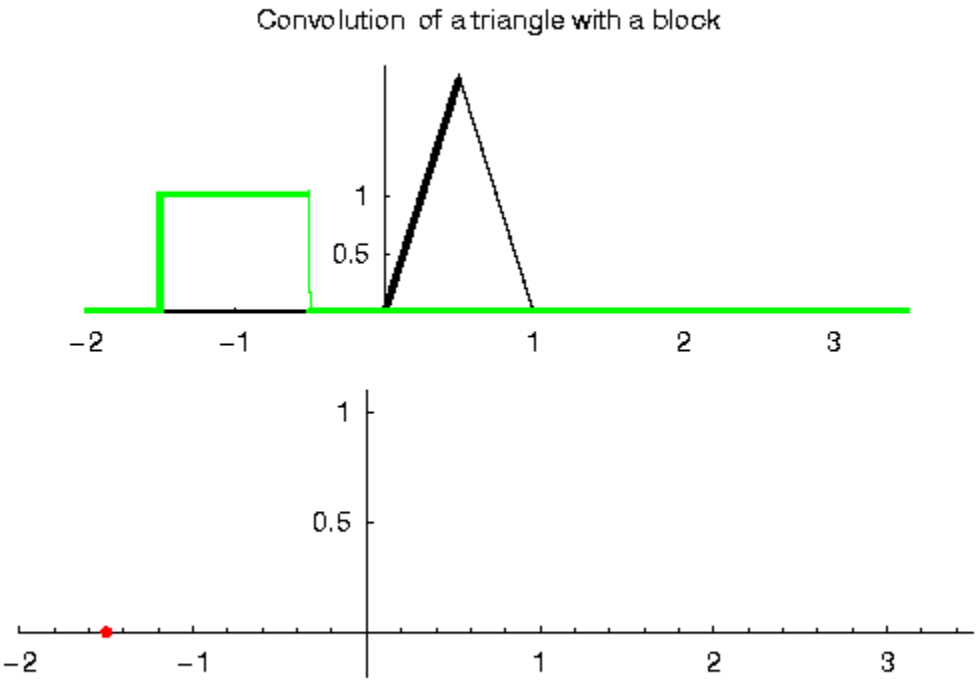


(a) 1차원 컨볼루션 (b) 2차원 컨볼루션

그림 4-6 컨볼루션 연산

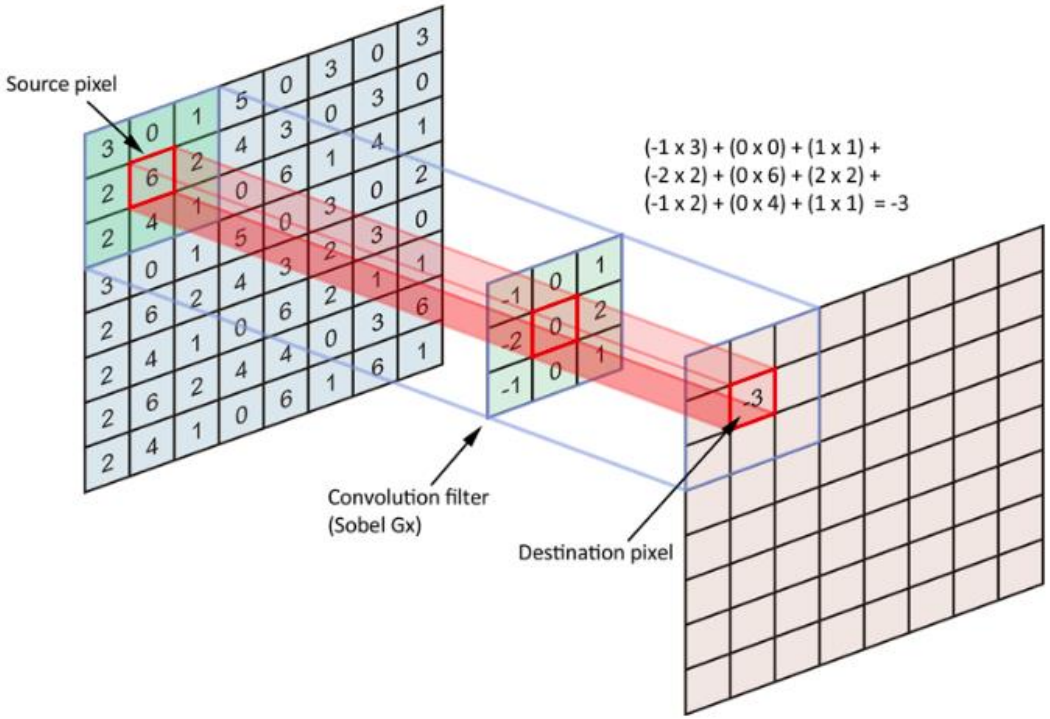
4.3.1 컨볼루션층

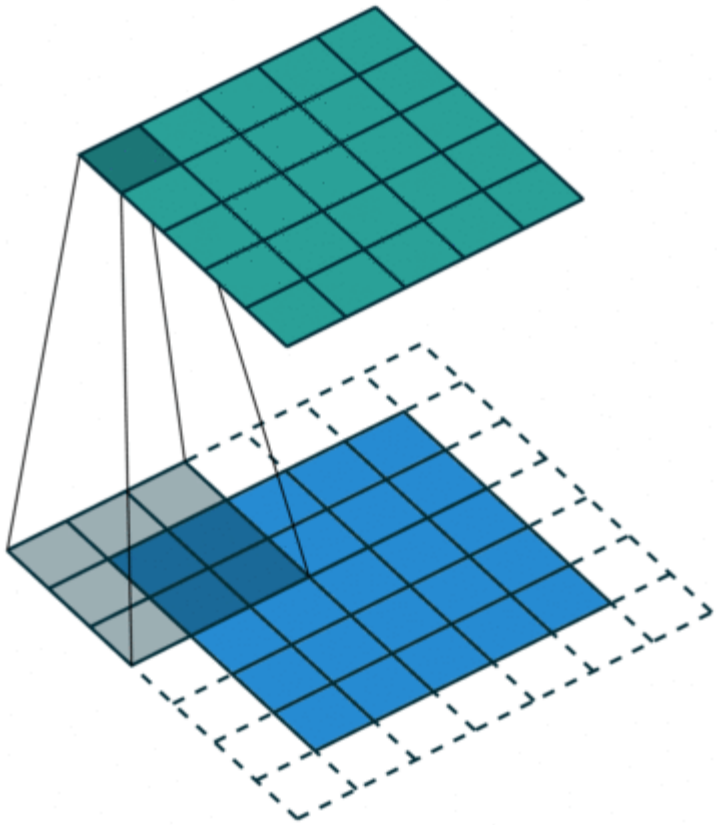
- 1차원 컨볼루션 연산의 예



4.3.1 컨볼루션층

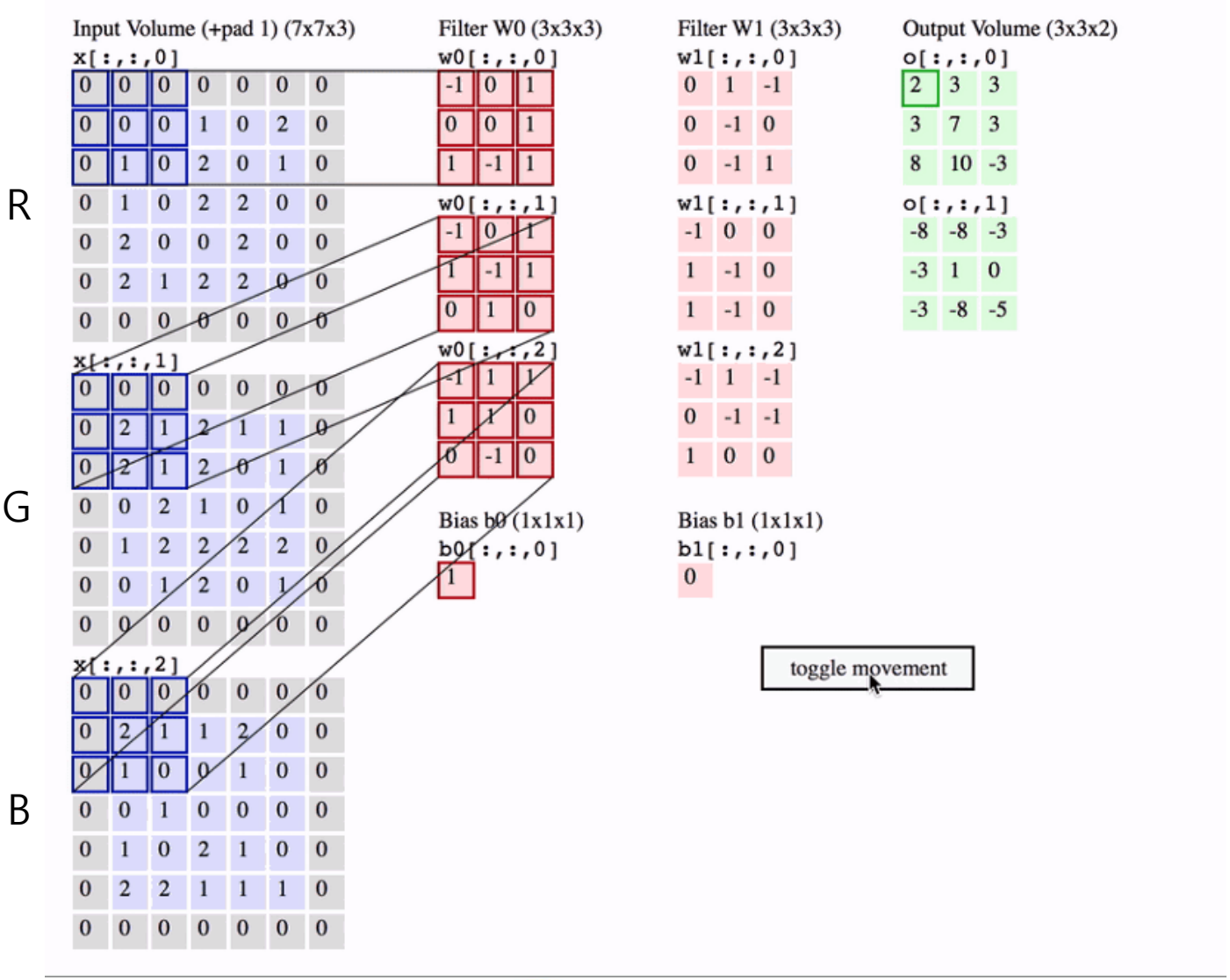
- 2차원 컨볼루션 연산의 예





4.3.1 컨볼루션층

- 3차원 (혹은 채널channel) 컨볼루션 연산의 예



4.3.1 컨볼루션층

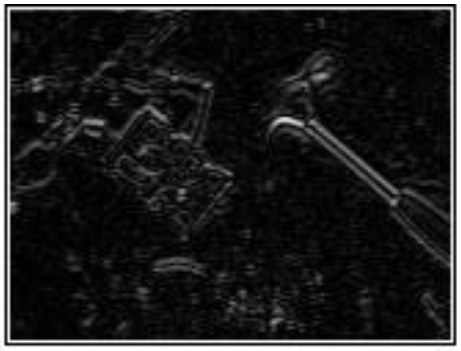
- 영상에서의 컨볼루션convolution 연산 예

$$\begin{Bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{Bmatrix}$$

Horizontal

x

=



$$\begin{Bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{Bmatrix}$$

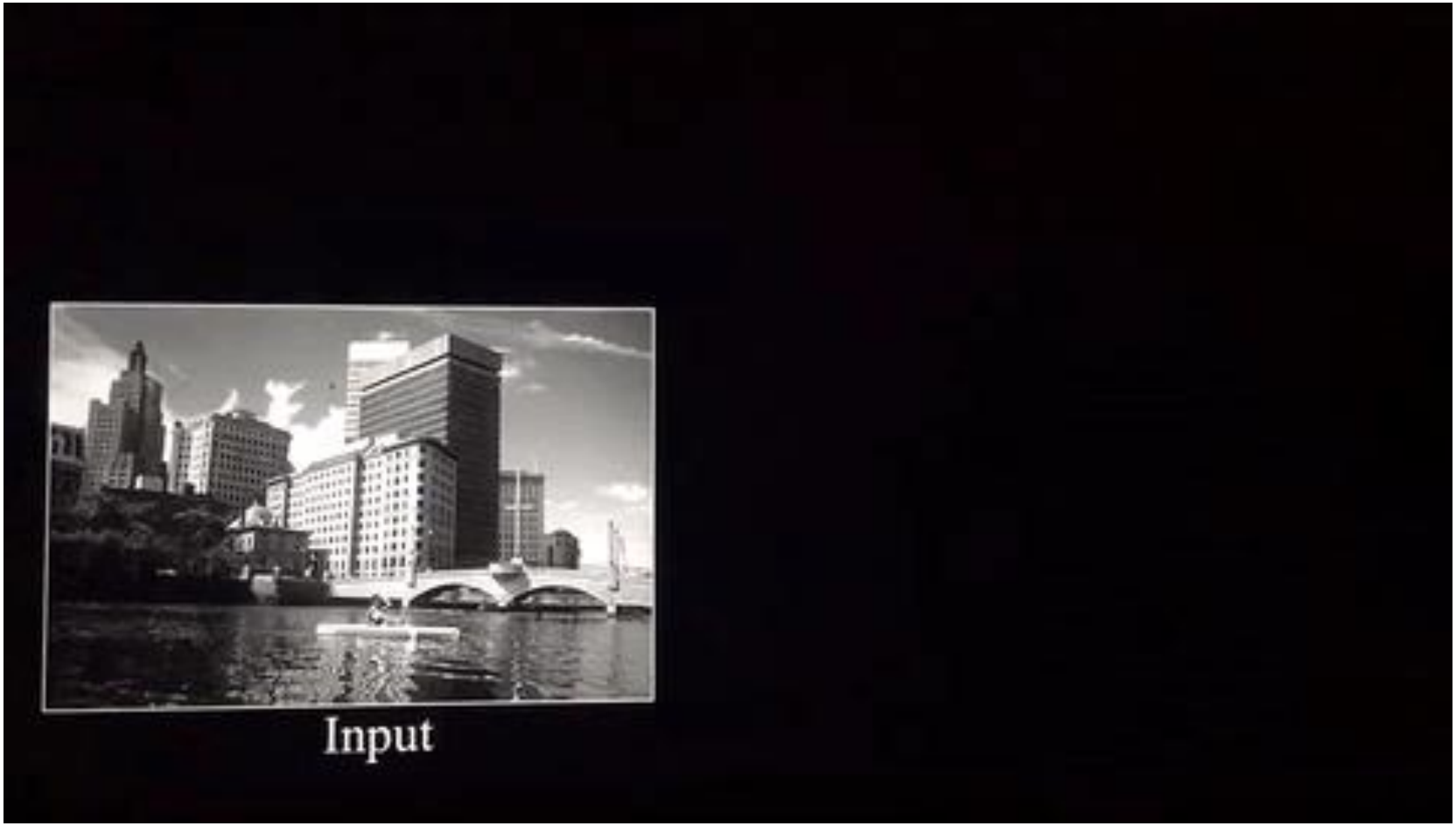
Vertical

x

=

4.3.1 컨볼루션층

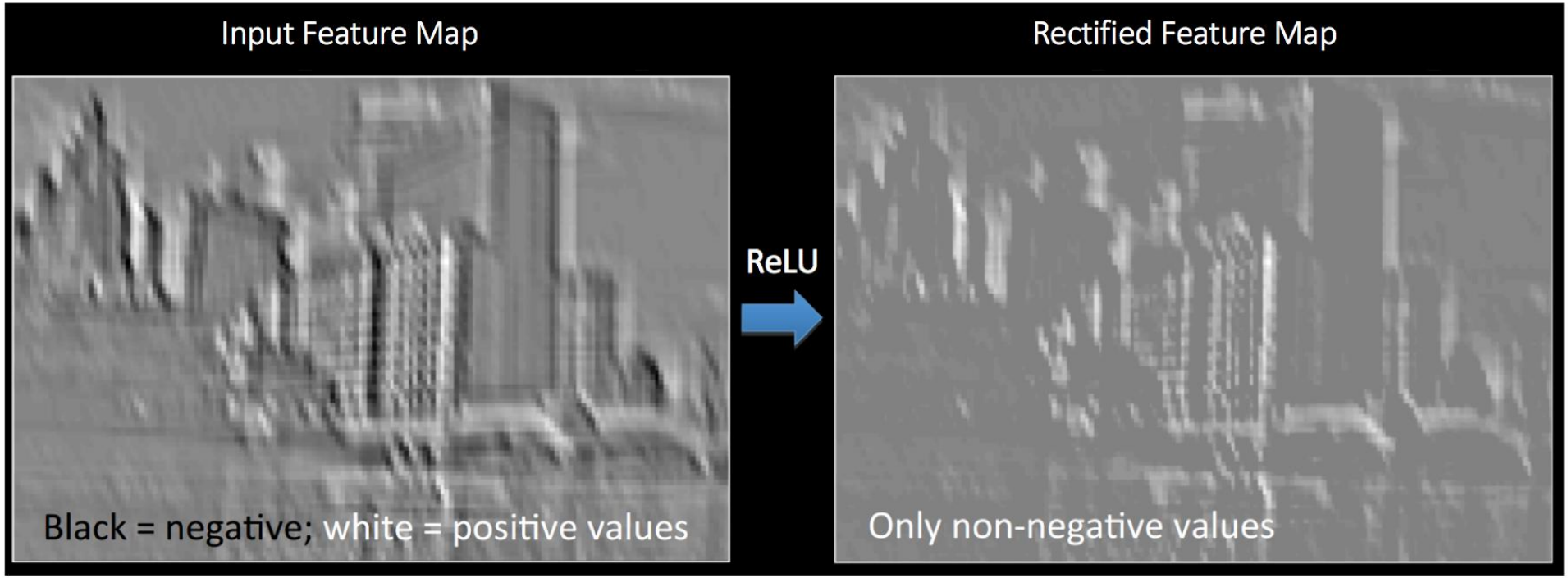
- 영상에서의 다수의 컨볼루션convolution 연산 예



커널의 값에 따라 커널이 추출하는 특징이 달라짐

4.3.1 컨볼루션층

- 영상에서의 ReLU (활성함수) 연산의 예

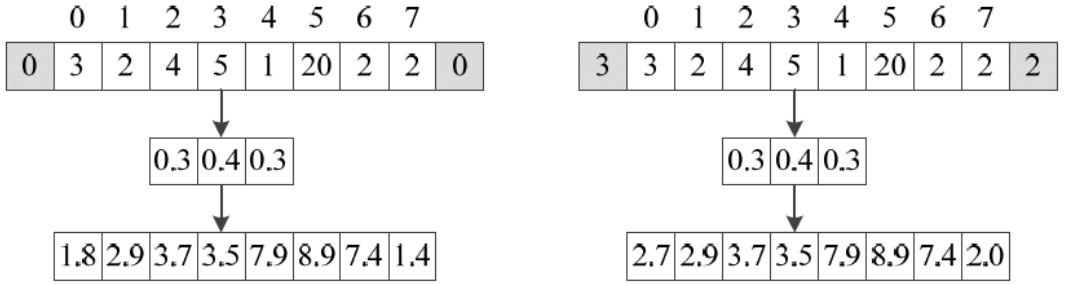


활성화된 특징 맵

4.3.1 컨볼루션층

덧대기padding

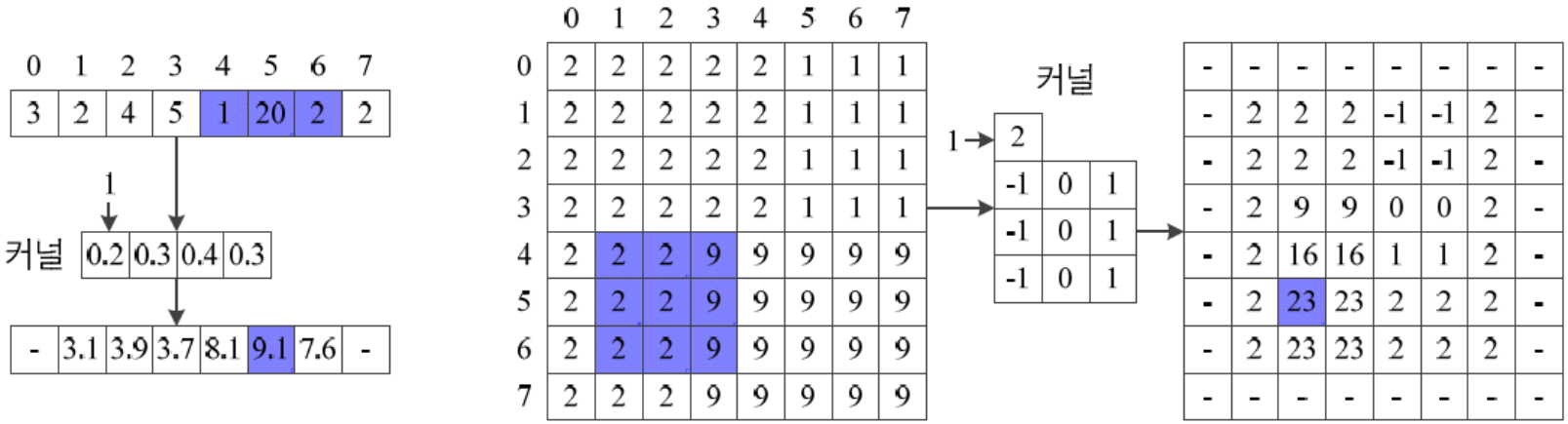
- 가장자리에서 영상의 크기가 줄어드는 효과 방지 (각 층의 입출력의 특징형상 유지)



(a) 0 덧대기 (b) 복사 덧대기

그림 4-7 덧대기(회색 노드가 덧댄 노드)

편향bias 추가



(a) 1차원 컨볼루션 (b) 2차원 컨볼루션

그림 4-8 바이어스

4.3.1 컨볼루션층

- 가중치 공유weight sharing 혹은 parameter sharing (묶인 가중치)
  - 모든 노드가 동일한 커널kernel을 사용

    (즉 가중치를 공유)하므로 매개변수는 3개에 불과
  - 모델의 복잡도가 크게 낮아짐

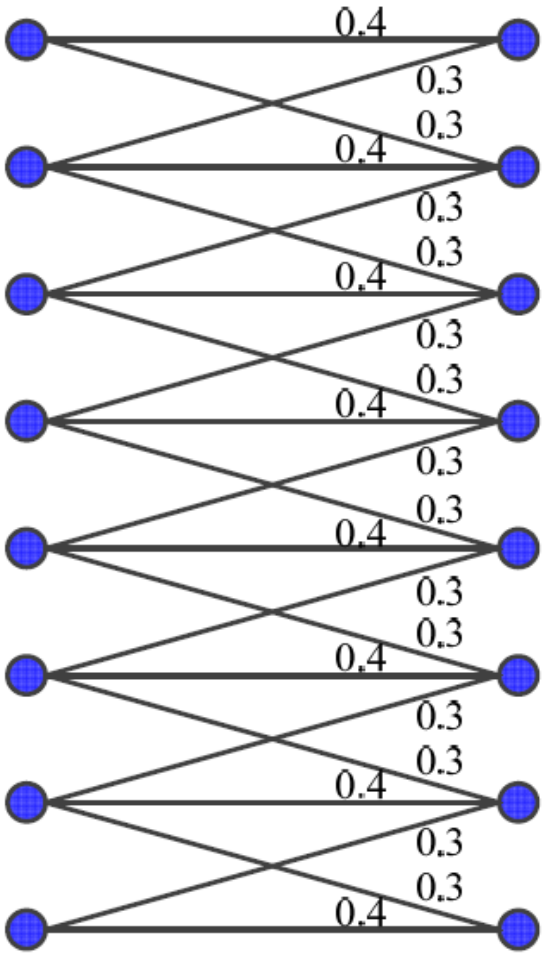


그림 4-9 CNN의 가중치 공유

4.3.1 컨볼루션층

다중 특징 맵 추출

- 커널의 매개변수 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 선 혹은 모서리 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

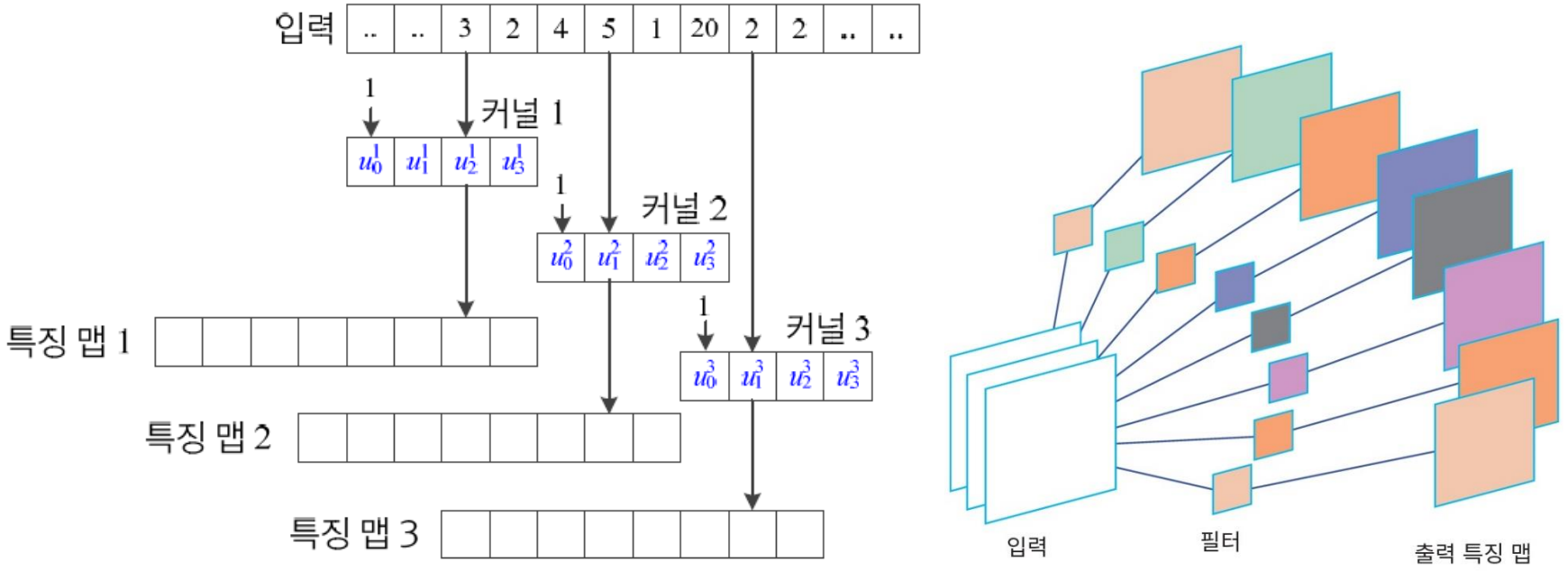


그림 4-10 다중 특징 맵 추출

4.3.1 컨볼루션층

특징 학습

- 커널을 사람이 설계하지 않고, 학습으로 찾음
  - u_i^k는 k번째 커널의 i번째 매개변수
- 예) 2차원 영상이 7*7 커널을 64개 사용한다면,

  학습은 (7*7+1)*64=3200개의 매개변수를 찾아내야 함
- DMLP와 마찬가지로 오류 역전파로 커널을 학습

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 (형상유지)

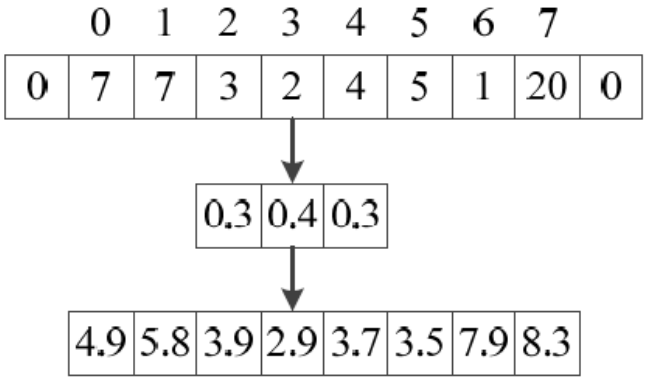


그림 4-11 CNN의 이동에 동변한 특성

- 병렬분산 구조
  - 각 노드는 독립적으로 계산 가능하므로 병렬 구조 (GPU)
  - 노드는 깊은 층을 거치면서 전체에 영향을 미치므로 분산 구조

    (특징 점진적 통합 → 추상화)

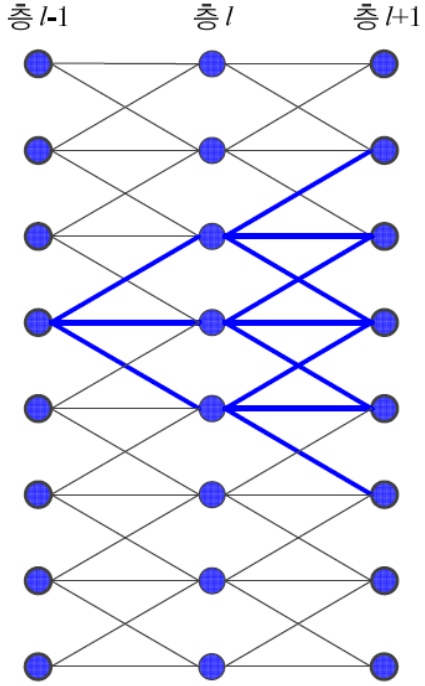
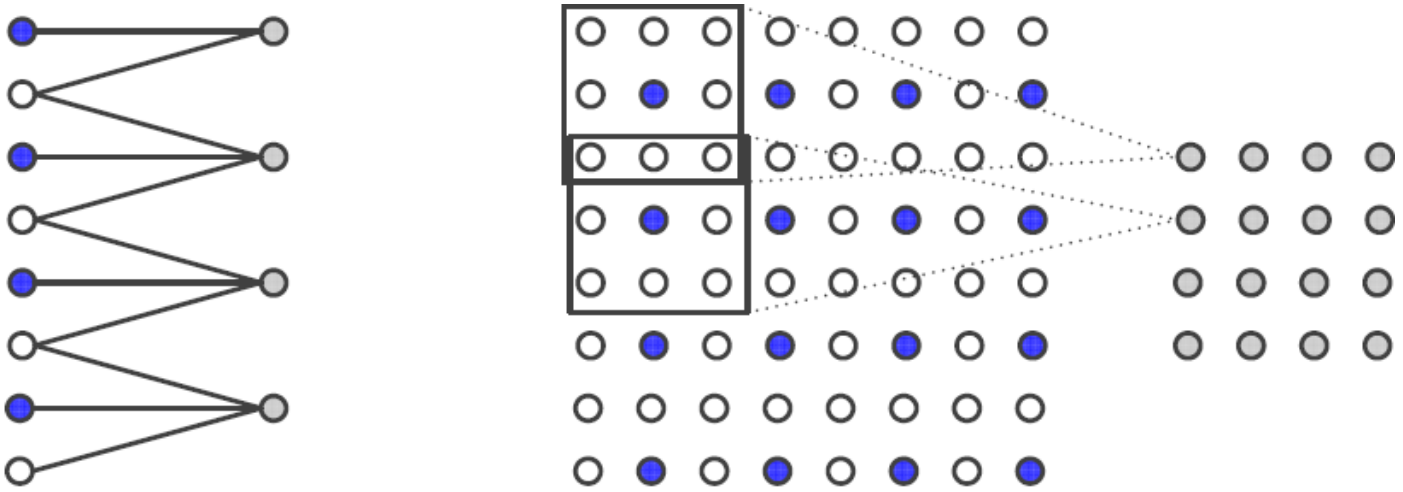


그림 4-12 CNN의 병렬 분산 구조

4.3.1 컨볼루션층

- 큰 보폭stride에 의한 다운샘플링down-sampling
  - 지금까지는 모든 화소에 커널 적용 → 보폭을 1로 설정한 셈
  - [그림 4-13]은 보폭이 2인 상황
  - 일반적으로 보폭이 k이면, k개 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

4.3.1 컨볼루션층

텐서 적용

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

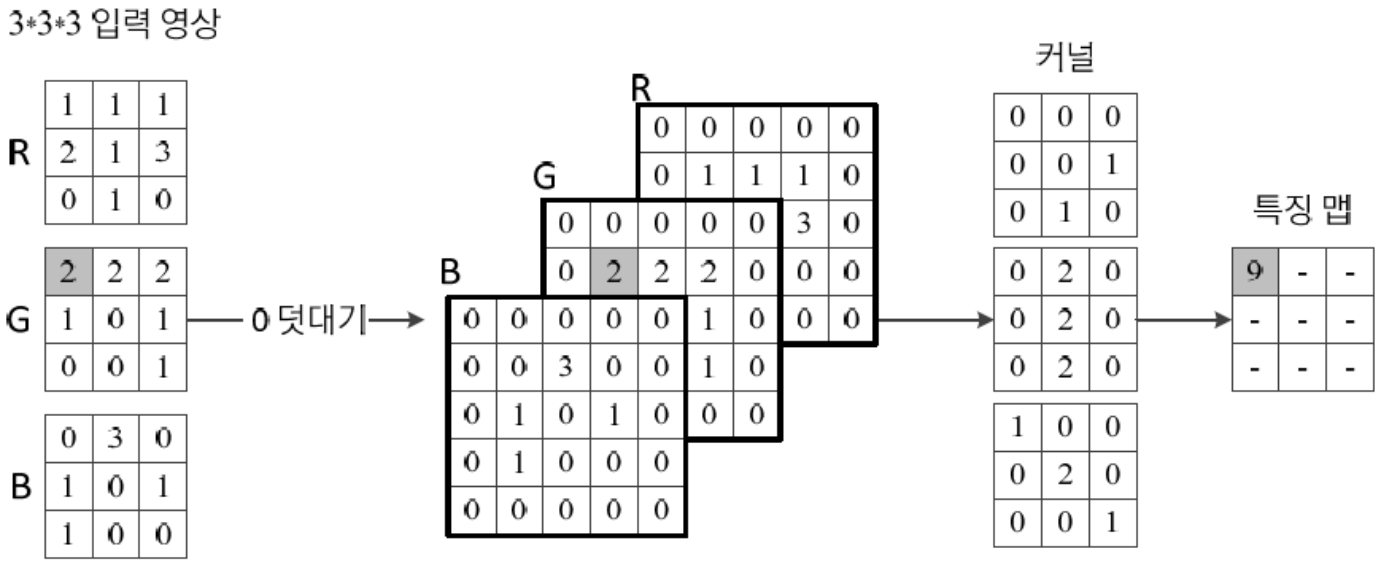


그림 4-14 텐서의 컨볼루션 연산(0 덧대기 적용)

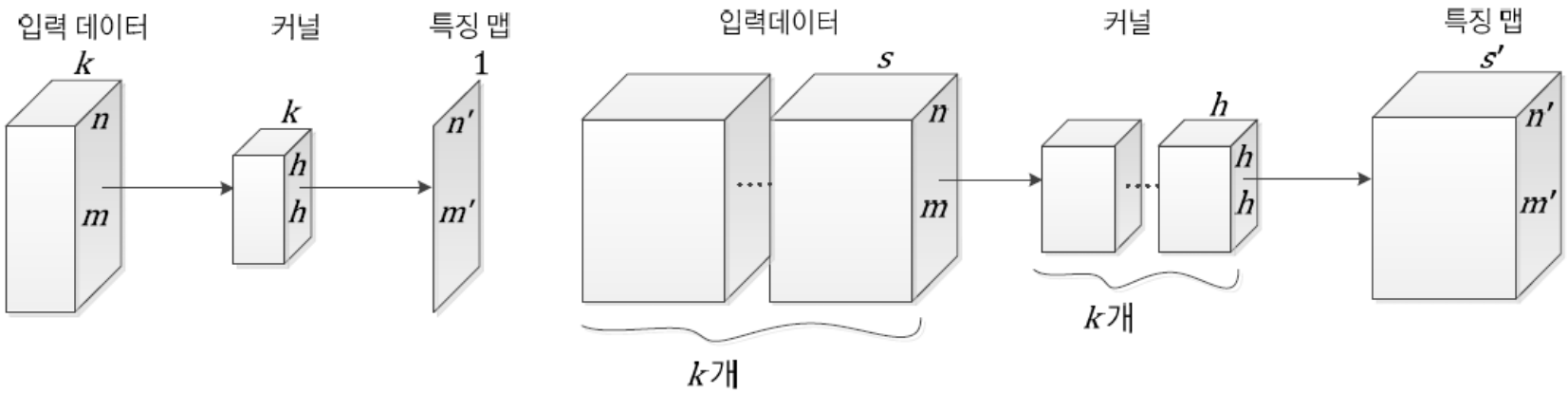
  - 특징 맵의 회색 노드의 계산 예시

$$\underbrace{\begin{pmatrix}0&0&0\\0&1&1\\0&2&1\end{pmatrix}}_{R}\underbrace{\begin{pmatrix}0&0&0\\0&2&2\\0&1&0\end{pmatrix}}_{G}\underbrace{\begin{pmatrix}0&0&0\\0&0&3\\0&1&0\end{pmatrix}}_{B}\circledast\underbrace{\begin{pmatrix}0&0&0\\0&0&1\\0&1&0\end{pmatrix}}_{c_1}\underbrace{\begin{pmatrix}0&2&0\\0&2&0\\0&2&0\end{pmatrix}}_{c_2}\underbrace{\begin{pmatrix}1&0&0\\0&2&0\\0&0&1\end{pmatrix}}_{c_3}=9$$

4.3.1 컨볼루션층

3차원 구조의 데이터 적용

- 채널이 k개인 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 ($3*s*m*n$), MRI 뇌영상 ($1*s*m*n$)
  - $k*h*h*h$ 커널을 $s*m*n$ 공간을 이동하면서 적용

4.3.2 풀링층

- 풀링pooling 연산
  - [그림 4-16]은 최대 풀링max pooling 예
  - 최대 풀링, 평균 풀링, 가중치 평균 풀링 등
  - 보폭을 크게 하면 다운샘플링downsampling 효과

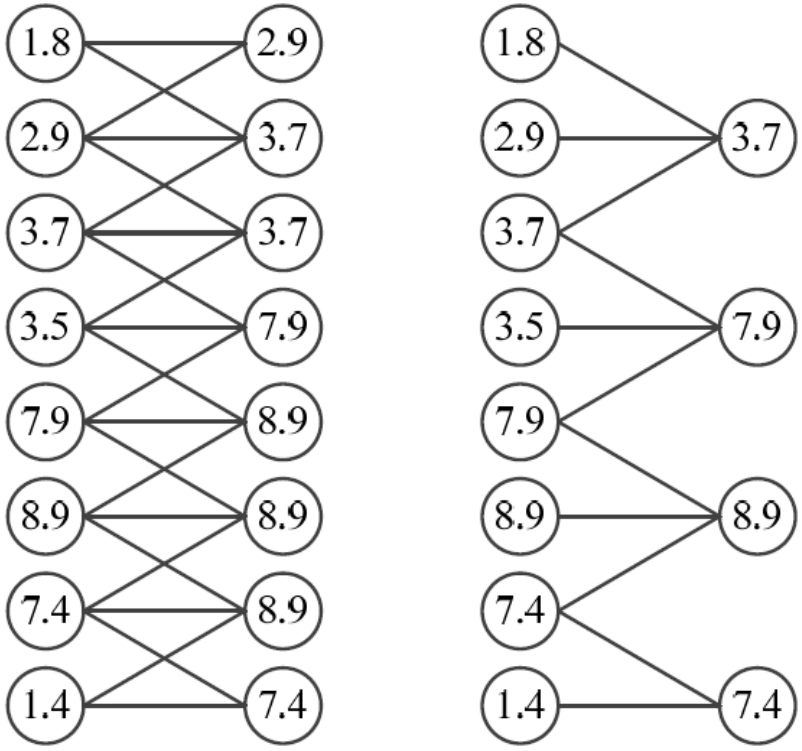
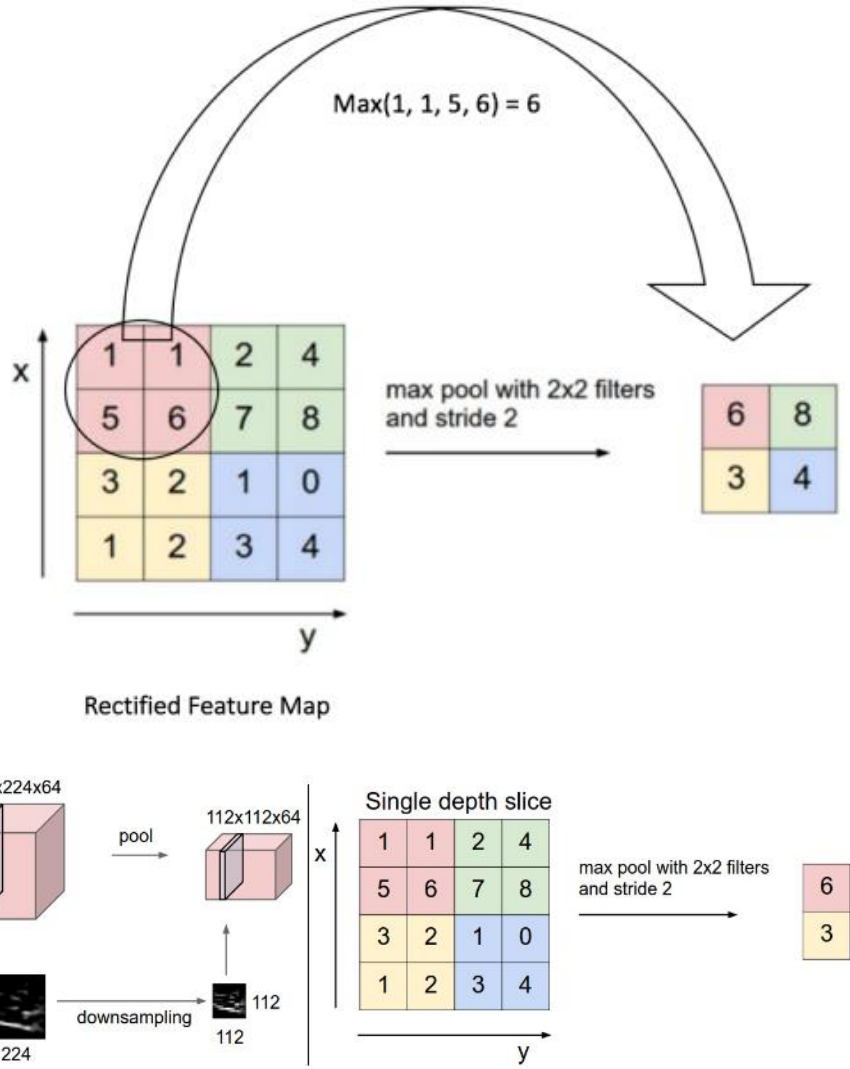


그림 4-16 최대 풀링



4.3.2 풀링층

풀링 연산의 특성

- 풀링은 상세 내용에서 요약 혹은 평균 등의 통계적 대표성을 추출함
- 매개변수가 없음
- 특징 맵의 수를 그대로 유지함 (크기X)
- 연산 효율화 (연산 횟수, 연결 가중치 개수 줄임)
- 작은 변화에 둔감 → 물체 인식이나 영상 검색 등에 효과적임

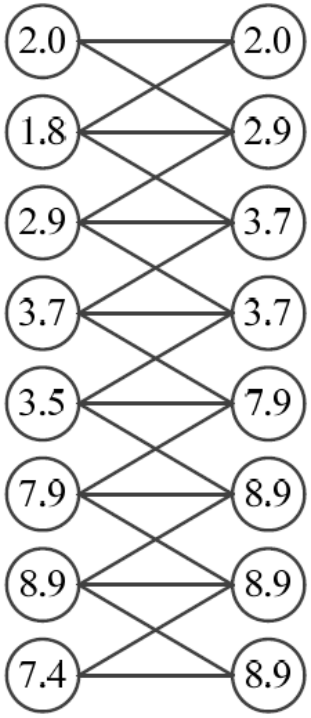
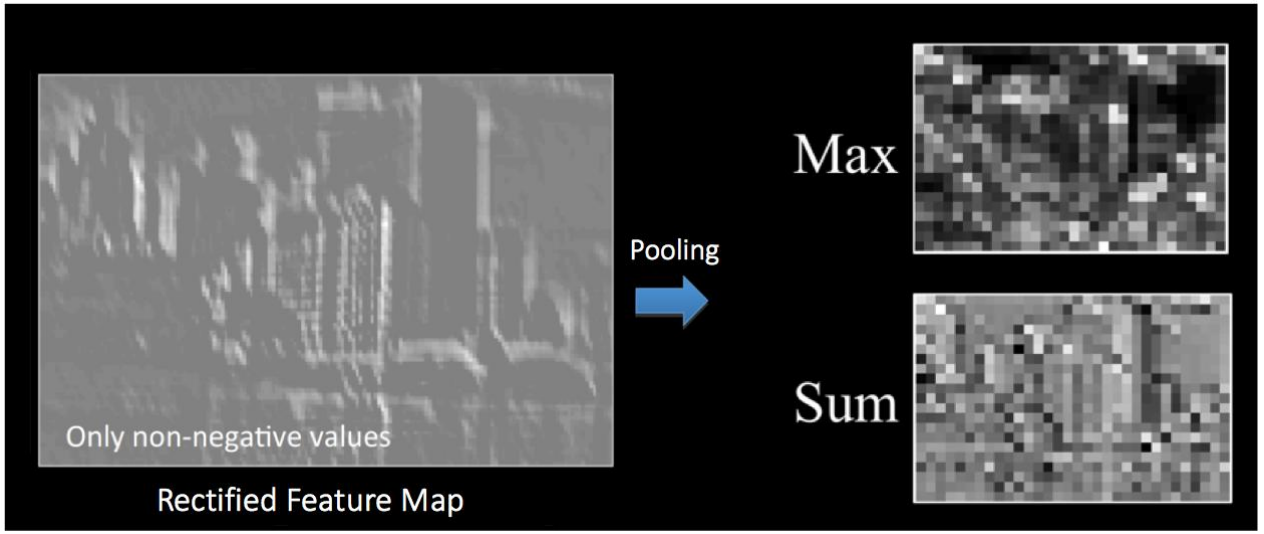


그림 4-17 작은 이동에 둔감한 최대 풀링



4.3.3 전체 구조

- 빌딩 블록building block
  - CNN은 빌딩 블록을 이어 붙여 깊은 구조로 확장
  - [그림 4-18]은 전형적인 빌딩블록: 컨볼루션층 → 활성함수 (주로 ReLU 사용) → 풀링층
  - 다중 커널을 사용하여 다중 특징 맵을 추출함

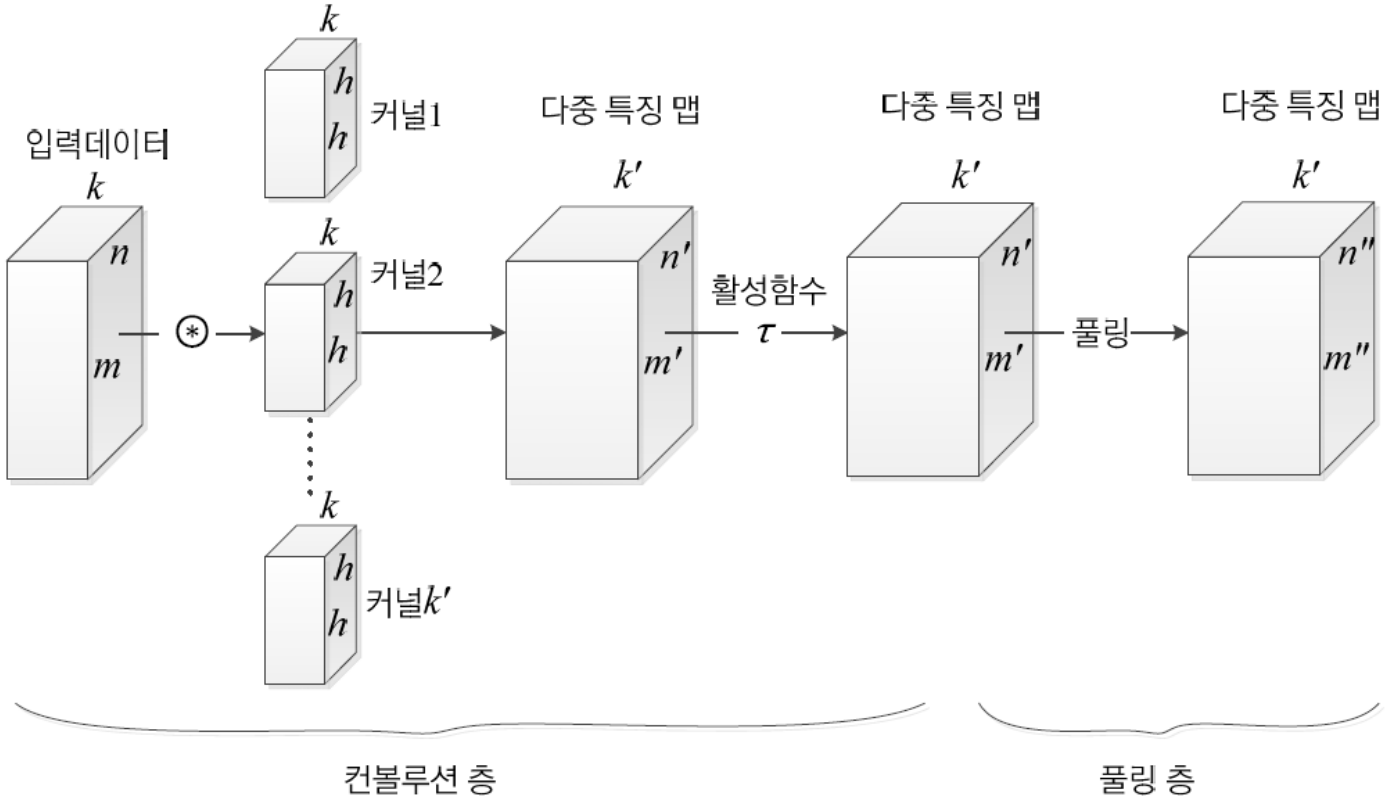


그림 4-18 CNN의 빌딩블록

4.3.3 전체 구조

컨볼루션 층의 출력 크기와 매개변수 수

- 입력: W1*H1*D1
- K개 F*F 커널, 보폭 S, 덧대기 P

→ 출력의 크기: W2*H2*D2

$$W2 = (W1-F+2P)/S+1$$

$$H2 = (H1-F+2P)/S+1$$

$$D2 = K$$

→ 매개변수의 수:

커널마다 (F*F*D1)개의 가중치와 1개의 바이어스를 가짐

따라서, 전체 매개변수의 수는 (F*F*D1)K + K

- 일반적으로 F=2, S=2 혹은 F=3, S=1를 사용함

4.3.3 전체 구조

예제

- 입력 크기: 32*32*3
- 10개의 5*5 커널 보폭 1, 덧대기 2
- 출력 크기: (32+2*2-5)/1+1=32 → 32*32*10
- 층의 매개변수 개수: 5*5*3+1=76 (+1 편향 추가) → 76*10=760

4.3.3 전체 구조

- 초창기 CNN 사례로서 LeNet-5
  - 특징 추출: CONV-POOL-CONV-POOL-CONV의 다섯 층을 통해 28*28 명암 영상을 120차원의 특징 벡터로 변환
    - 평균 풀링 사용
  - 분류: 은닉층이 하나인 MLP
  - CNN의 첫 번째 성공사례: 필기 숫자 인식기 만들어 수표 인식 자동화 시스템 구현

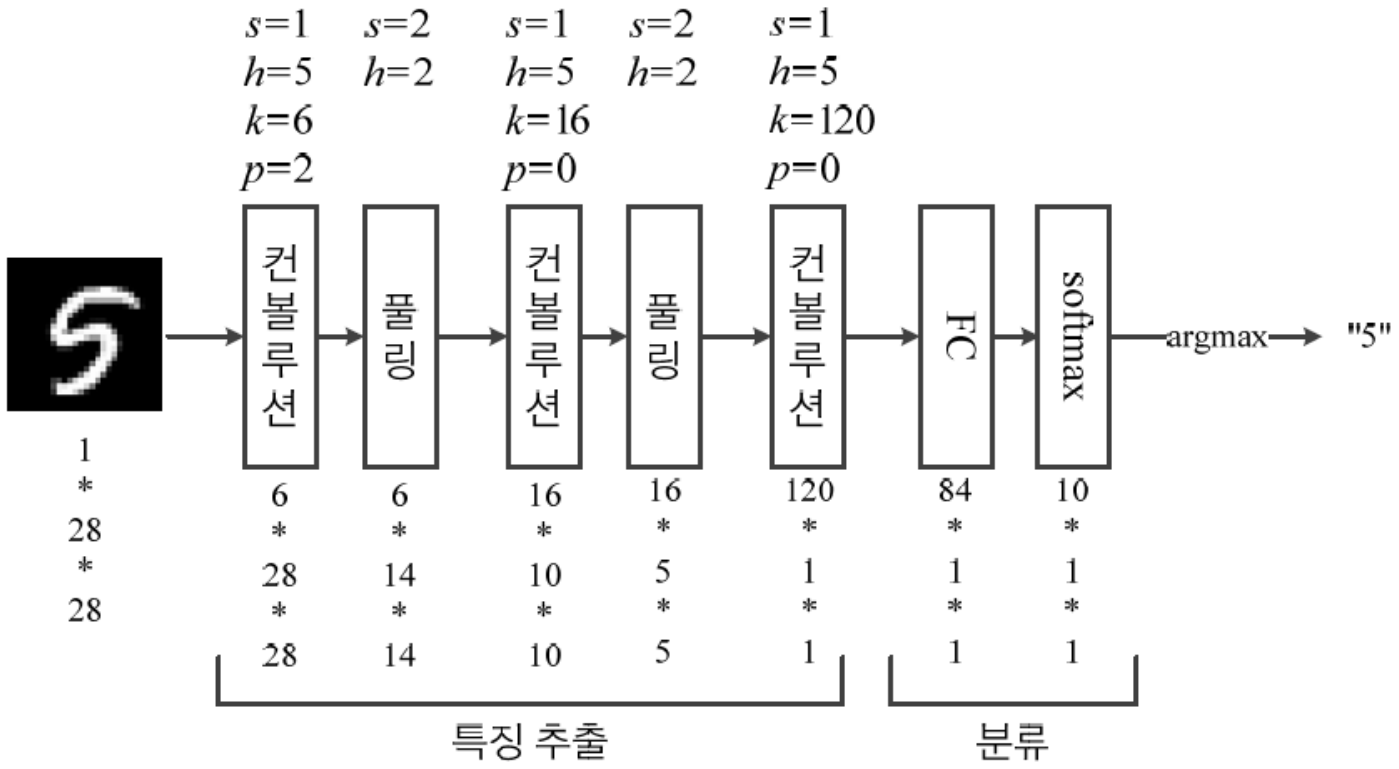


그림 4-19 LeNet-5 구조

4.3.3 전체 구조

- CNN CIFAR10 훈련 예제
  - http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

ConvNetJS CIFAR-10 demo

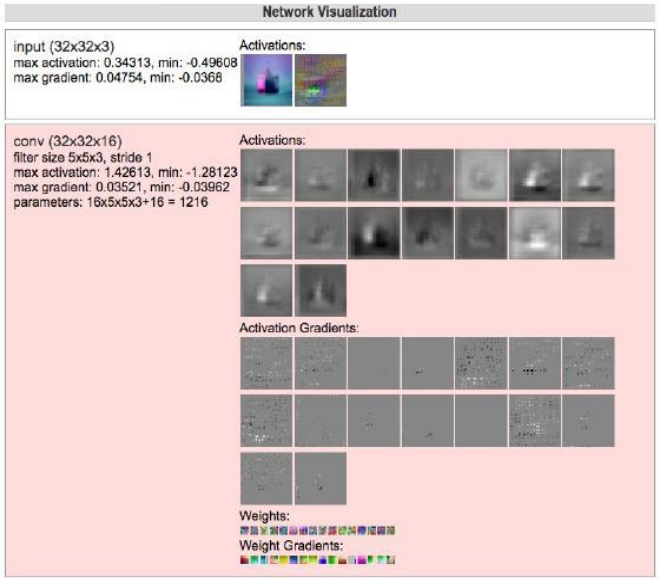
Description

This demo trains a Convolutional Neural Network on the CIFAR-10 dataset in your browser, with nothing but Javascript. The state of the art on this dataset is about 90% accuracy and human performance is at about 94% (not perfect as the dataset can be a bit ambiguous). I used this python script to parse the original files (python version) into batches of images that can be easily loaded into page DOM with img tags.

This dataset is more difficult and it takes longer to train a network. Data augmentation includes random flipping and random image shifts by up to 2px horizontally and verically.

By default, in this demo we're using Adadelta which is one of per-parameter adaptive step size methods, so we don't have to worry about changing learning rates or momentum over time. However, I still included the text fields for changing these if you'd like to play around with SGD+Momentum trainer.

Report questions/bugs/suggestions to @karpathy.



4.3.3 전체 구조

- CNN 시각화 예제
  - https://poloclub.github.io/cnn-explainer/