8.2 컨볼루션 신경망의 구조

■ 컨볼루션 신경망(CNN; convolutional neural network)

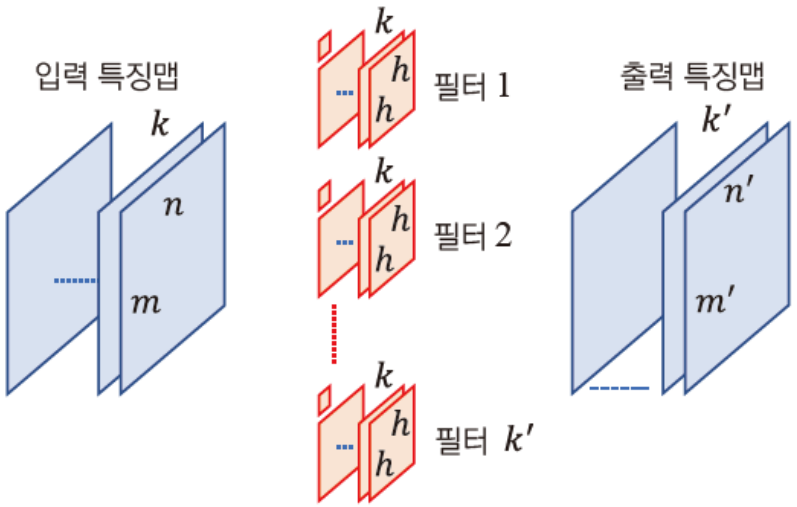
- CNN은 3.4절에서 공부한 컨볼루션 연산이 핵심
- 3.4절에서는 사람이 필터 설계(예, 가우시안 스무딩, 소벨 에지 등)
  - 이들 필터는 어디서 왔는가?
  - 인식에 최적인가?
  - 데이터셋에 맞는 최적 필터를 사용해야 하지 않을까?

■ CNN의 핵심 아이디어는 '최적의 필터를 학습'으로 알아냄

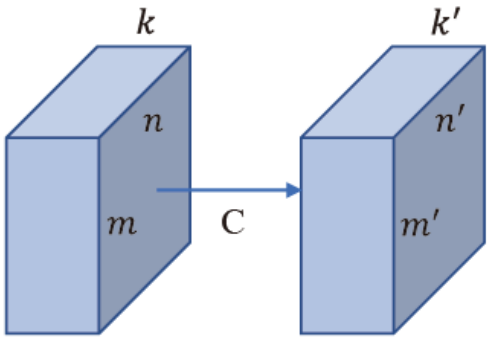
8.2.1 컨볼루션층과 풀링층

■ **컨볼루션층은 표준 컨볼루션에서 몇 가지 확장이 필요**

- 입력 특징 맵이 m*n*k 텐서라면, h*h*k 필터 사용
- 하나의 필터는 바이어스 하나를 가짐 (kh^2+1개의 가중치)
- 필터를 여러 개(k′개) 적용하여 풍부한 특징 맵 추출
- 출력 특징 맵은 m′*n′*k′ 텐서
- 덧대기$_{padding}$와 보폭$_{stride}$



(a) 세부 내용



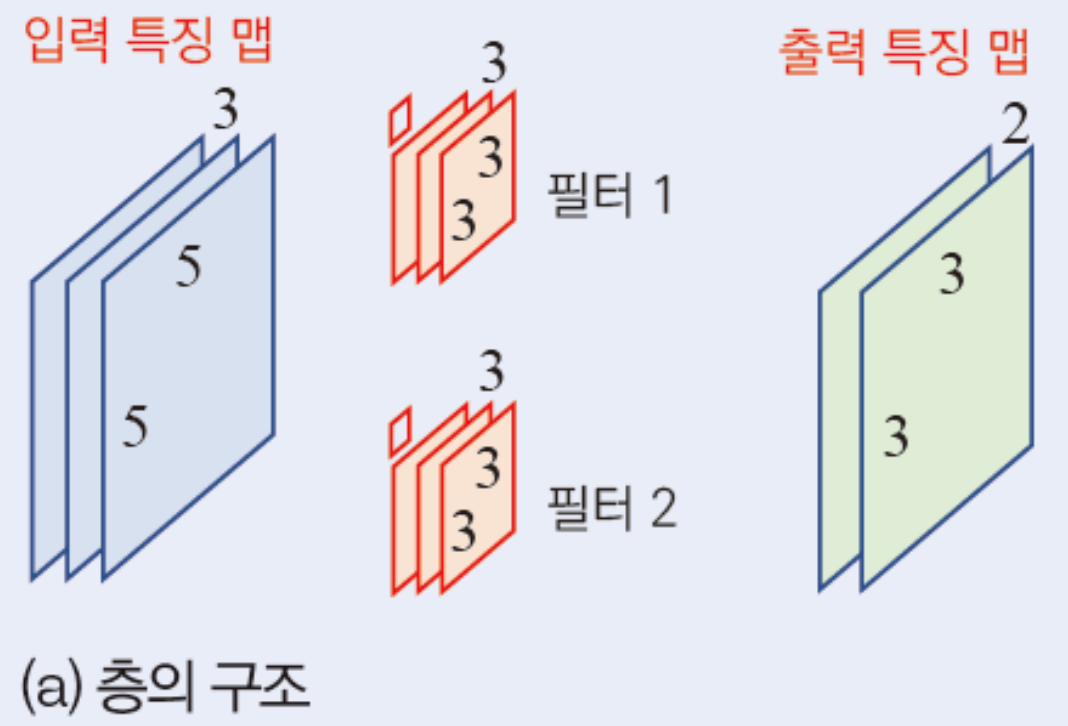
(b) 간결한 블록 표현

그림 8-6 컨볼루션층

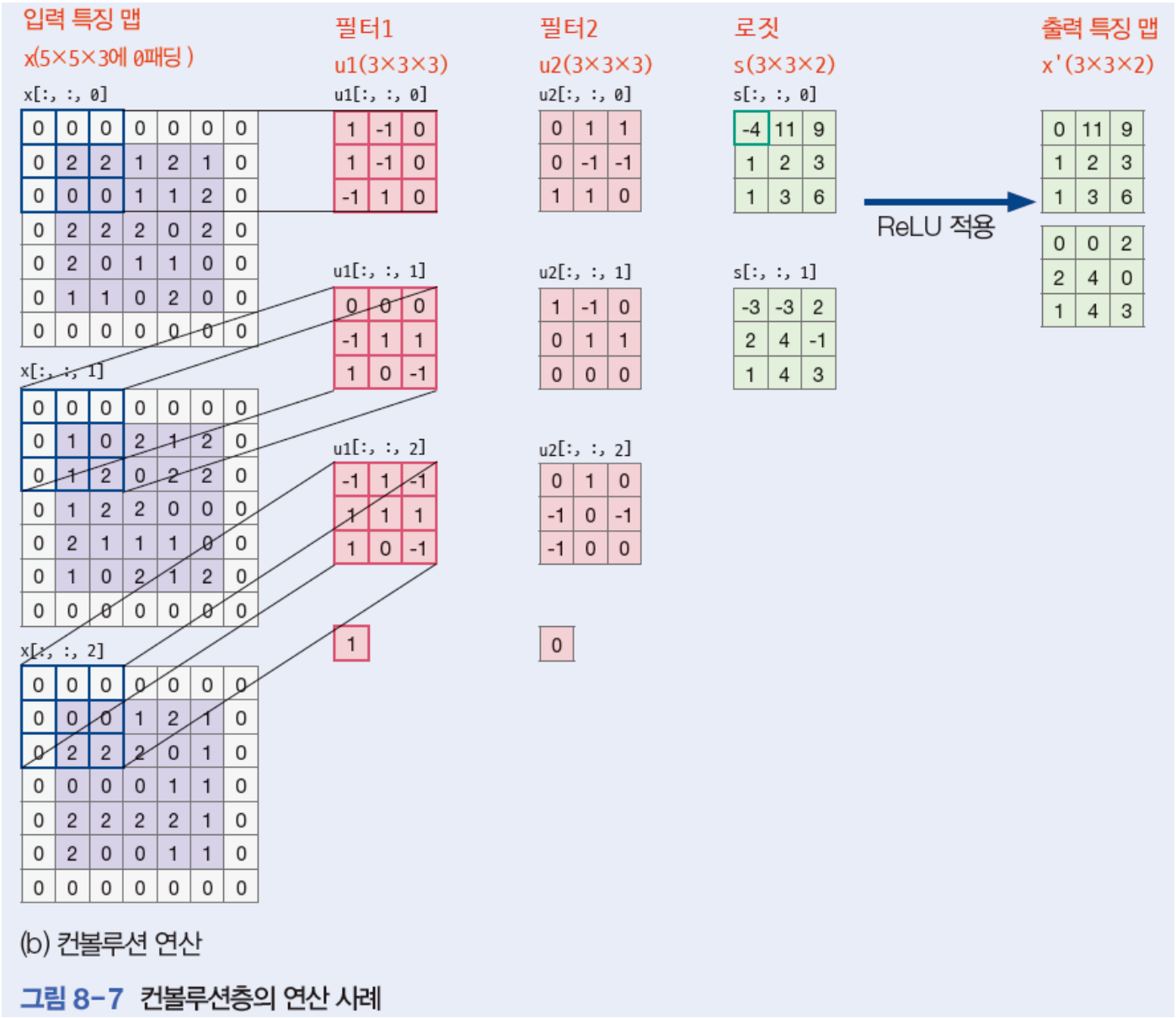
8.2.1 컨볼루션층과 풀링층

■ [예시 8-1] 컨볼루션층의 연산

- 5*5*3 특징 맵에 3*3*3 필터를 2개 적용. 0 패딩 적용, 보폭은 2
- 5*5*3 특징 맵이 3*3*2 특징 맵이 됨([그림 8-7])



8.2.1 컨볼루션층과 풀링층



(b) 컨볼루션 연산

그림 8-7 컨볼루션층의 연산 사례

8.2.1 컨볼루션층과 풀링층

- (0,0) 화소의 계산 사례

$$\underbrace{0\times1+0\times(-1)+0\times0+0\times1+2\times(-1)+2\times0+0\times(-1)+0\times1+0\times0+}_{\text{채널 0}}$$
$$\underbrace{0\times0+0\times0+0\times0+0\times(-1)+1\times1+0\times1+0\times1+1\times0+2\times(-1)+}_{\text{채널 1}}$$
$$\underbrace{0\times(-1)+0\times1+0\times(-1)+0\times1+0\times1+0\times1+0\times1+2\times0+2\times(-1)+}_{\text{채널 2}}\ \underbrace{1}_{\text{바이어스}}=-4$$

■ 가중치 공유weight sharing와 부분 연결성partial connection

- 입력 특징 맵의 모든 화소가 같은 필터 사용하니 가중치를 공유하는 셈
- 필터는 해당 화소 주위에 국한하여 연산 수행. 가중치 개수가 획기적으로 줄어듦
  - k′개의 h*h*k 필터를 쓰는 경우 가중치는 $k'(kh^2+1)$개

8.2.1 컨볼루션층과 풀링층

■ [예제 8-2] 컨볼루션층의 연산량

- 첫번째 층
  - 입력은 256*256*3 텐서. 0 덧대기하고 보폭 2이고 3*3 필터를 64개 사용하므로 출력은 128*128*64 텐서 (필터 모양은 3*3*3)
  - 128*128*28*64번의 곱셈 수행
- 두번째 층
  - 입력은 128*128*64 텐서. 0 덧대기하고 보폭 2이고 5*5 필터를 128개 사용하므로 출력은 64*64*128 텐서 (필터 모양은 5*5*64)
  - 64*64*1601*128번의 곱셈 수행

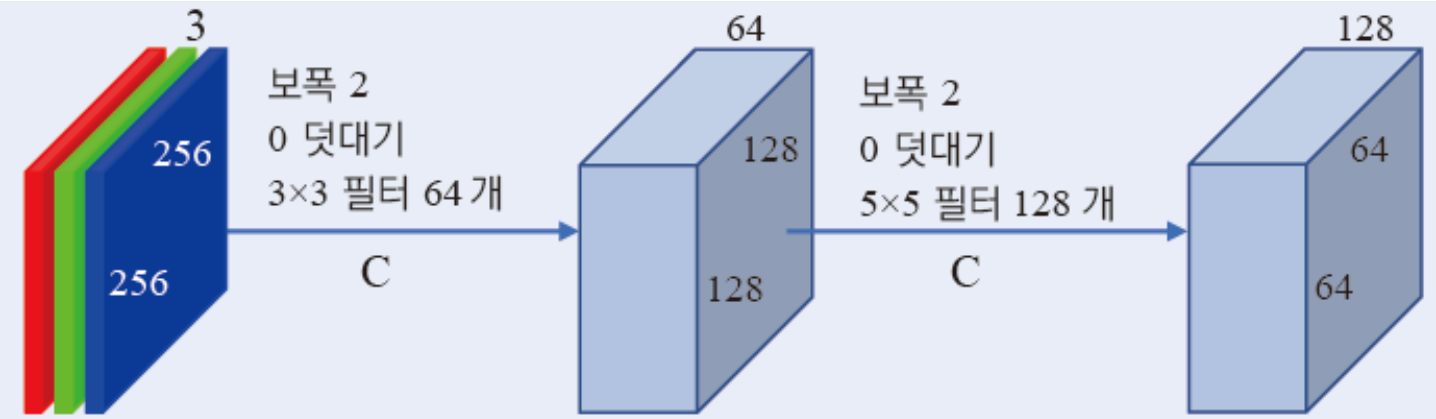


그림 8-8 컨볼루션층을 2개 쌓은 신경망

8.2.1 컨볼루션층과 풀링층

■ 풀링층

- 최대 풀링max pooling은 필터 안의 화소의 최대값 취함
- 평균 풀링average pooling은 필터 안의 화소의 평균을 취함
- 지나친 상세함을 줄이는 효과와 특징 맵의 크기를 줄이는 효과

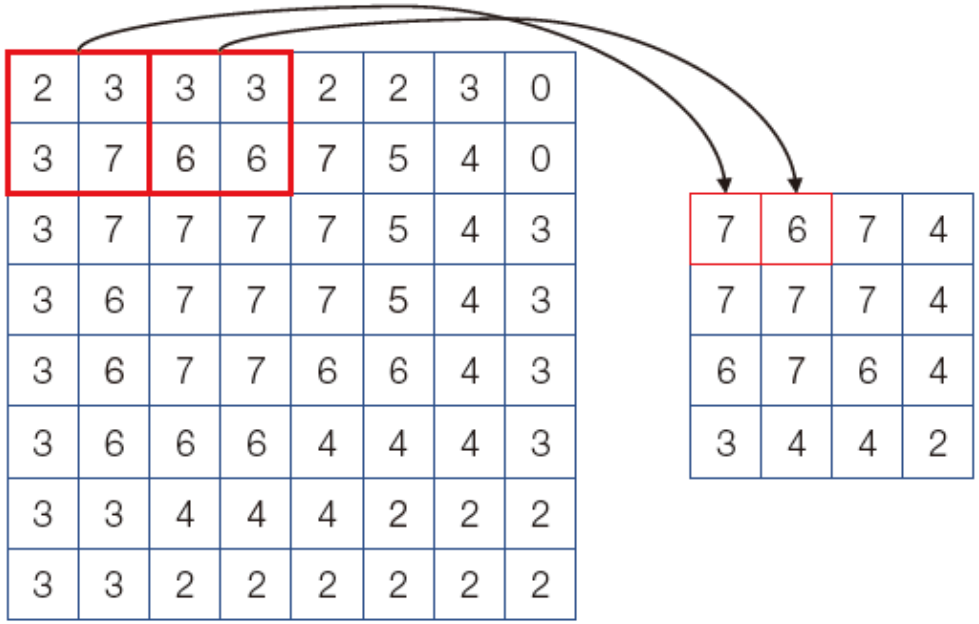
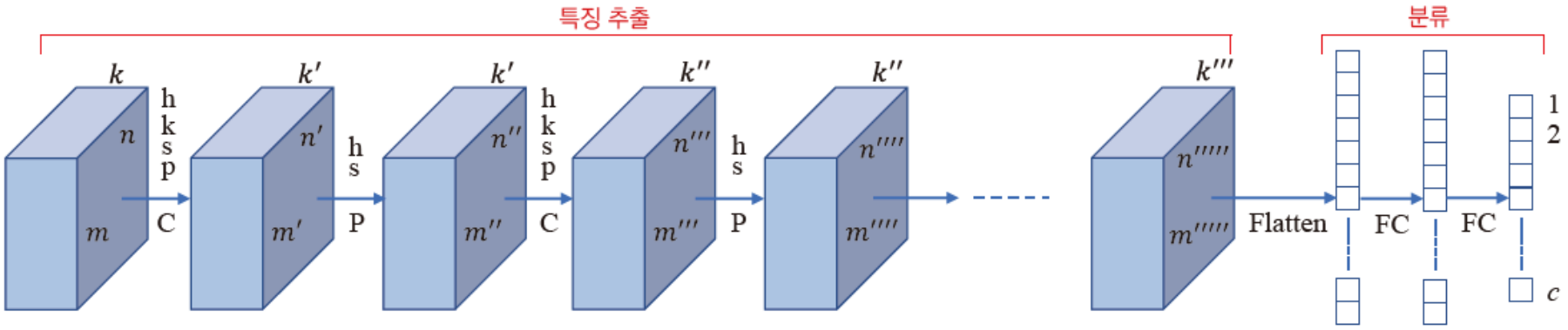


그림 8-9 풀링층(2×2 필터로 최대 풀링 적용, 보폭=2)

8.2.2 빌딩블록을 쌓아 만드는 컨볼루션 신경망

■ 빌딩블록 쌓기

- 보통 컨볼루션층과 풀링층을 번갈아 쌓음
- 풀링층에서는 텐서 깊이가 유지됨
- 신경망 앞 부분은 특징 추출, 뒷부분은 분류 담당



- C: 컨볼루션층
- P: 풀링층
- FC: 완전연결층
- Flatten: 2차원 구조를 1차원 구조로 변환
- h: 커널 크기
- k: 커널 개수
- s: 보폭
- p: 덧대기

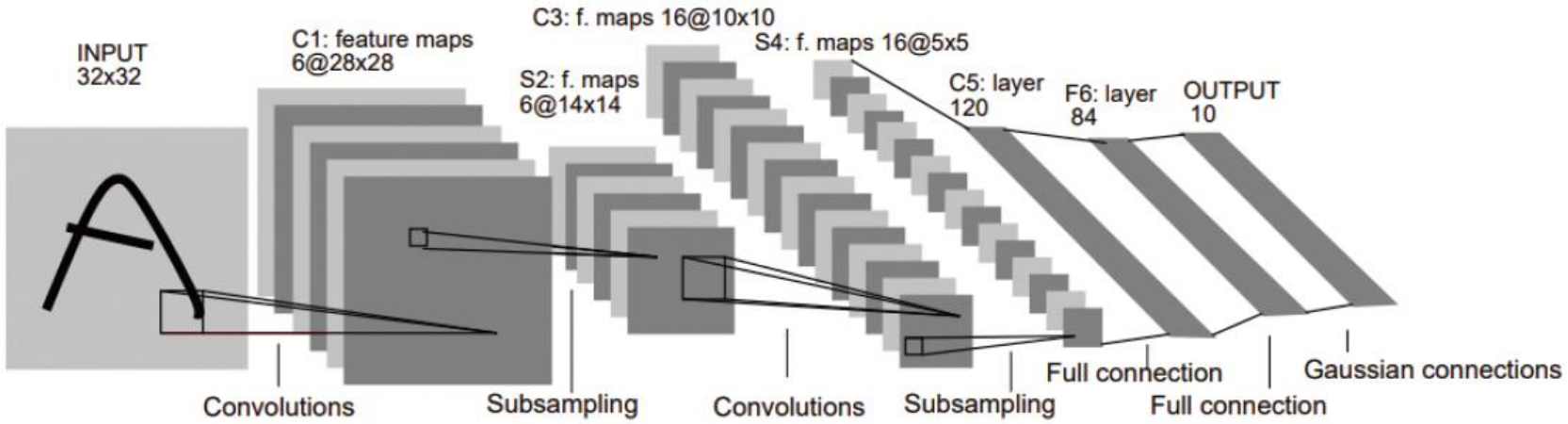
그림 8-10 컨볼루션층과 풀링층을 번갈아 쌓아 만드는 컨볼루션 신경망의 전형적인 구조

8.2.2 빌딩블록을 쌓아 만드는 컨볼루션 신경망

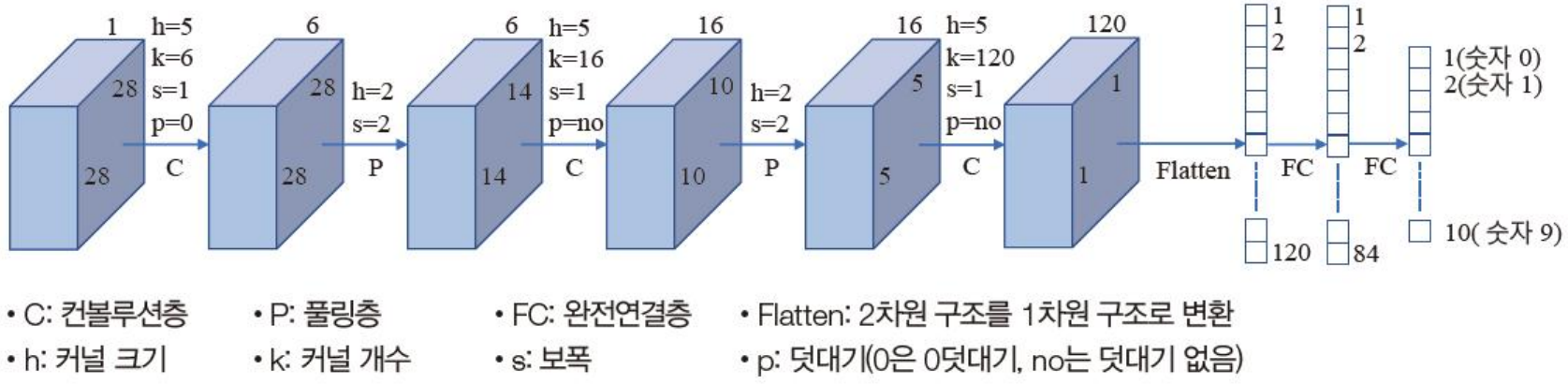
■ LeNet-5 사례[LeCun1998]

- C-P-C-P-C-FC-FC 구조
- 가중치 집합
  - 첫번째 컨볼루션층은 (5*5*1+1)*6개의 가중치
  - 두번째 컨볼루션층은 (5*5*6+1)*16
  - 세번째 컨볼루션층은 (5*5*16+1)*120
  - 첫번째 완전연결층은 (120+1)*84
  - 두번째 완전연결층은 (84+1)*10
  - 총 61,706개의 가중치

8.2.2 빌딩블록을 쌓아 만드는 컨볼루션 신경망



(a) [LeCun1998]의 그림



- C: 컨볼루션층
- P: 풀링층
- FC: 완전연결층
- Flatten: 2차원 구조를 1차원 구조로 변환
- h: 커널 크기
- k: 커널 개수
- s: 보폭
- p: 덧대기(0은 0덧대기, no는 덧대기 없음)

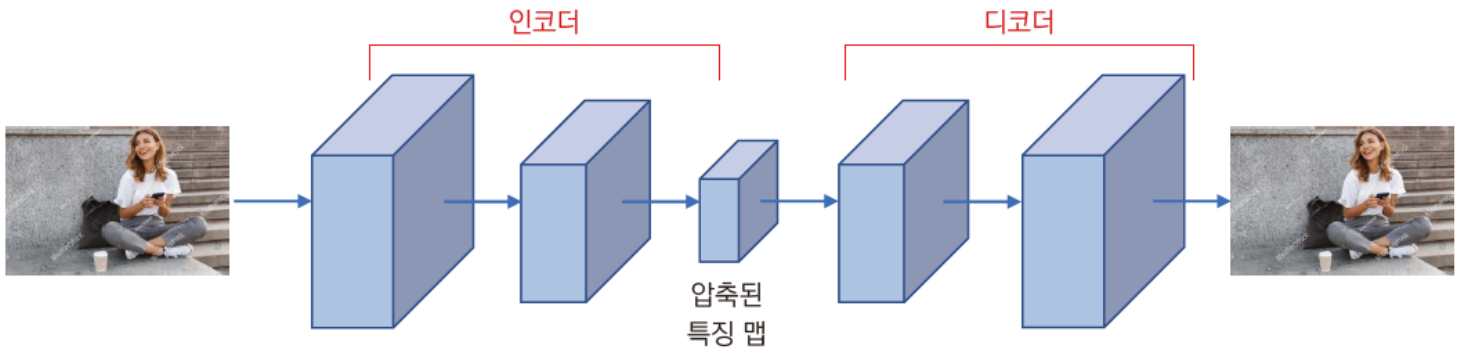
(b) [그림 8-10] 표기에 따른 그림

그림 8-11 LeNet-5의 구조

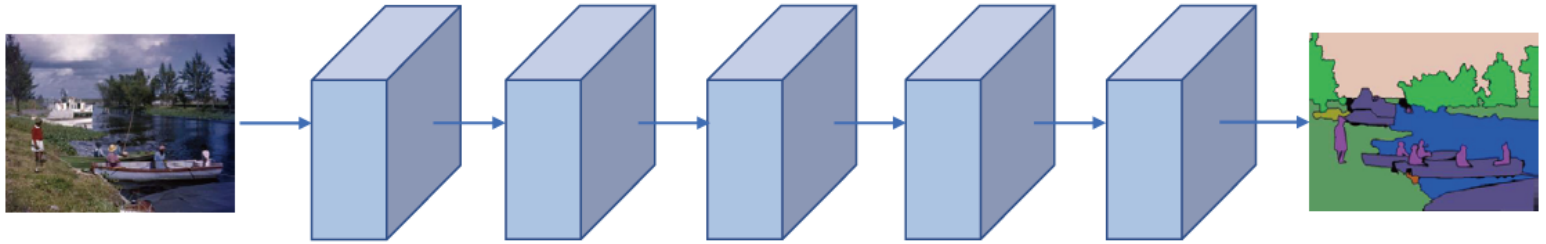
8.2.2 빌딩블록을 쌓아 만드는 컨볼루션 신경망

■ 유연한 구조

- 문제에 따라 다양한 모양으로 조립 가능
- 예) 오토인코더: 입력과 출력이 같은 신경망 (비지도 학습)
- 예) 분할을 위한 신경망: 출력은 분할 맵



(a) 오토인코더



(b) 영상 분할을 위한 컨볼루션 신경망

그림 8-12 컨볼루션 신경망의 유연한 구조

8.3 컨볼루션 신경망의 학습

■ 역전파 학습 알고리즘 사용(다층 퍼셉트론과 비슷)

- 컨볼루션층의 커널 화소와 온전연결층의 에지가 가중치에 대당
- 풀링층은 가중치 없음

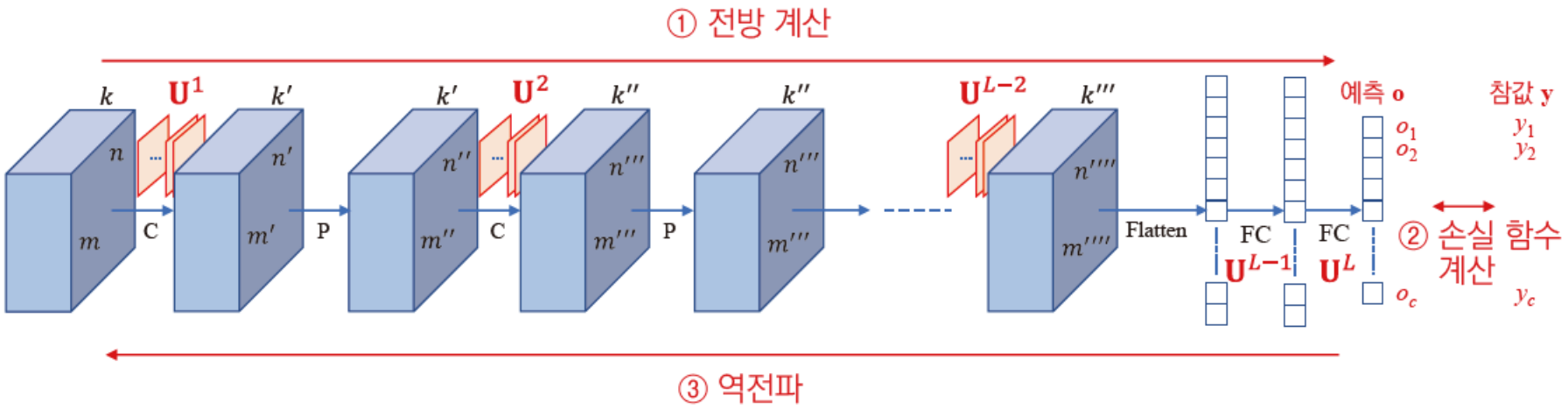


그림 8-13 컨볼루션 신경망을 학습하기 위한 역전파 알고리즘

8.3 컨볼루션 신경망의 학습

■ 특징 학습feature learning

- 학습 알고리즘은 주어진 데이터셋을 인식하는데 최적인 필터를 알아냄

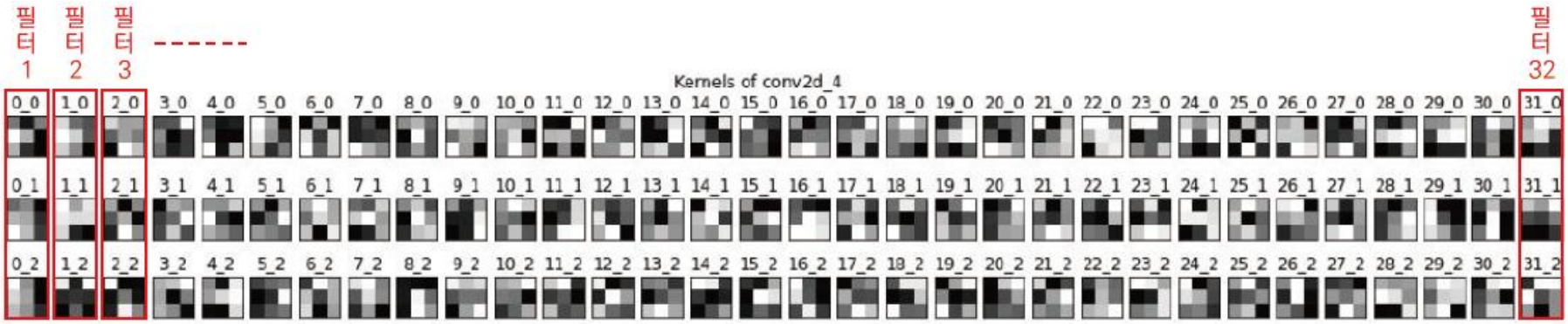


그림 8-14 CIFAR-10 데이터셋으로 학습한 컨볼루션 신경망의 최적 필터

8.3 컨볼루션 신경망의 학습

■ 통째 학습end-to-end learning

- 특징 학습과 분류기 학습을 한꺼번에 진행

(a) 수작업 특징을 사용하는 고전적 패러다임 (b) 통째 학습을 사용하는 딥러닝 패러다임

그림 8-15 딥러닝에 의한 컴퓨터 비전 방법론의 대전환

■ 컨볼루션 신경망이 우수한 이유

- 데이터의 원래 구조를 유지
- 특징 학습을 통해 최적의 특징을 추출
- 신경망의 깊이를 깊게 함

8.4 컨볼루션 신경망 구현

- 텐서플로를 이용한 컨볼루션 신경망 구현은 쉽다.
- LeNet-5 재현부터 시작

8.4.1 LeNet-5 재현

프로그램 8-1 LeNet-5로 MNIST 인식하기

```
01  import numpy as np
02  import tensorflow as tf
03  import tensorflow.keras.datasets as ds
04
05  from tensorflow.keras.models import Sequential
06  from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dropout,Dense
07  from tensorflow.keras.optimizers import Adam
08
09  (x_train,y_train),(x_test,y_test)=ds.mnist.load_data()
10  x_train=x_train.reshape(60000,28,28,1)
11  x_test=x_test.reshape(10000,28,28,1)
12  x_train=x_train.astype(np.float32)/255.0
13  x_test=x_test.astype(np.float32)/255.0
14  y_train=tf.keras.utils.to_categorical(y_train,10)
15  y_test=tf.keras.utils.to_categorical(y_test,10)
16
```

2차원 구조로 변환

데이터 준비

8.4.1 LeNet-5 재현

```
17  cnn=Sequential()
18  cnn.add(Conv2D(6,(5,5),padding='same',activation='relu',input_
    shape=(28,28,1)))
19  cnn.add(MaxPooling2D(pool_size=(2,2),strides=2))
20  cnn.add(Conv2D(16,(5,5),padding='valid',activation='relu'))
21  cnn.add(MaxPooling2D(pool_size=(2,2),strides=2))
22  cnn.add(Conv2D(120,(5,5),padding='valid',activation='relu'))
23  cnn.add(Flatten())
24  cnn.add(Dense(units=84,activation='relu'))
25  cnn.add(Dense(units=10,activation='softmax'))
26
27  cnn.compile(loss='categorical_crossentropy',optimizer=Adam
    (learning_rate=0.001),metrics=['accuracy'])
28  cnn.fit(x_train,y_train,batch_size=128,epochs=30,validation_
    data=(x_test,y_test),verbose=2) ①
29
30  res=cnn.evaluate(x_test,y_test,verbose=0)
31  print('정확률=',res[1]*100) ②
```

컨볼루션층(5*5 커널을 6개 사용)

1차원 구조로 변환

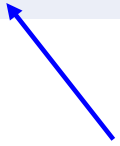
모델 선택 (신경망 구조 설계)

학습

예측(성능 측정)

8.4.1 LeNet-5 재현

```
Epoch 1/30 ①
469/469 - 3s - loss: 0.2555 - accuracy: 0.9212 - val_loss: 0.0794 - val_accuracy:
0.9742 - 3s/epoch - 6ms/step
Epoch 2/30
469/469 - 2s - loss: 0.0671 - accuracy: 0.9786 - val_loss: 0.0536 - val_accuracy:
0.9826 - 2s/epoch - 4ms/step
…
Epoch 29/30
469/469 - 2s - loss: 0.0037 - accuracy: 0.9988 - val_loss: 0.0506 - val_accuracy:
0.9902 - 2s/epoch - 5ms/step
Epoch 30/30
469/469 - 2s - loss: 0.0059 - accuracy: 0.9981 - val_loss: 0.0430 - val_accuracy:
0.9910 - 2s/epoch - 5ms/step
정확률= 99.09999966621399 ②
```



[프로그램 7-5]의 깊은 다층 퍼셉트론보다 정확률 0.68% 향상