

인공지능

07. Convolution Neural Network

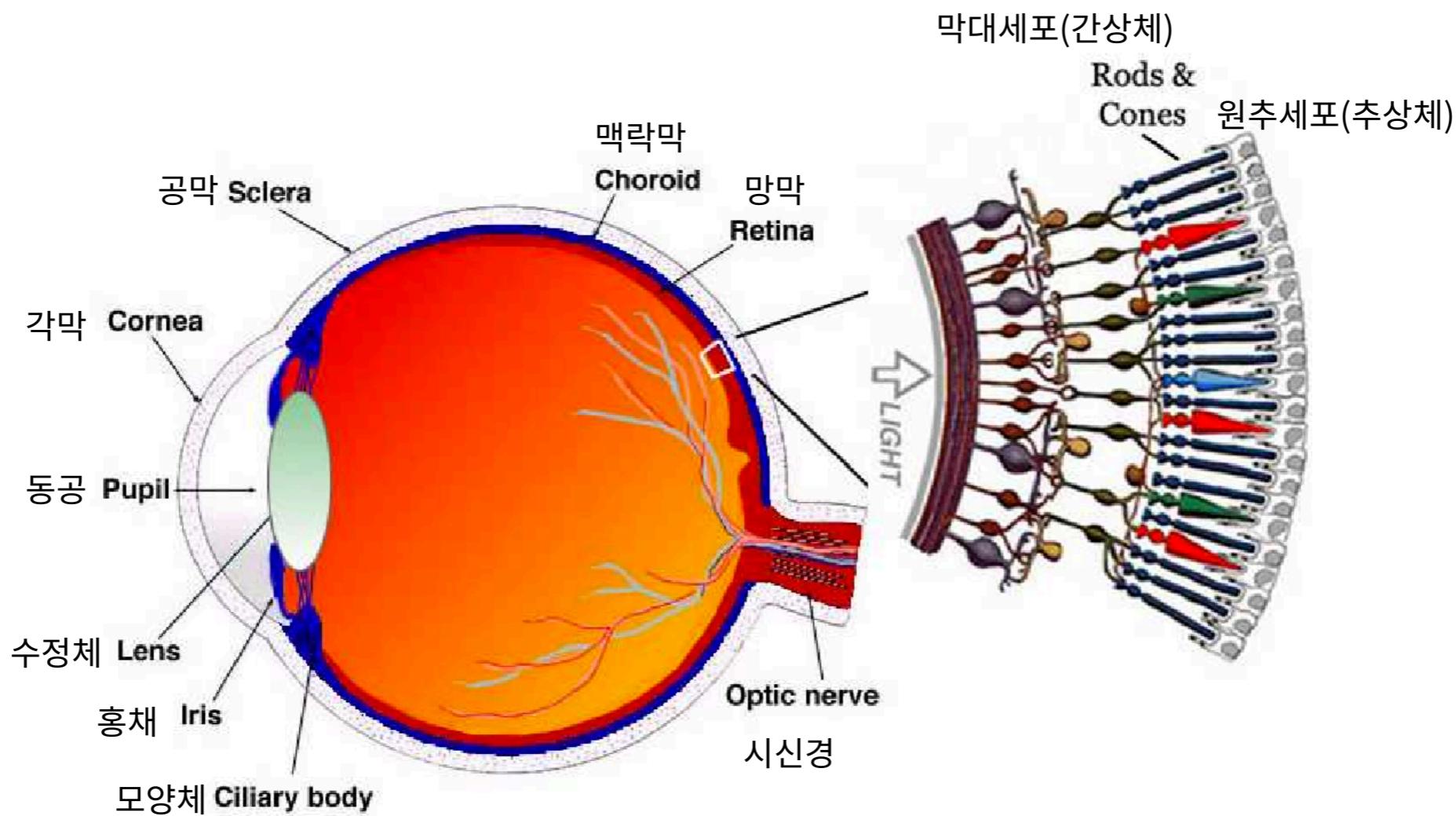
Dept. of Digital Contents



1. 컨볼루션 신경망

○ 망막과 수용장(receptive field)

■ 시각 기관의 구조



그림출처 : Detail to attention : exploiting limits of the human visual system for selective rendering(K.Cater, 2004)

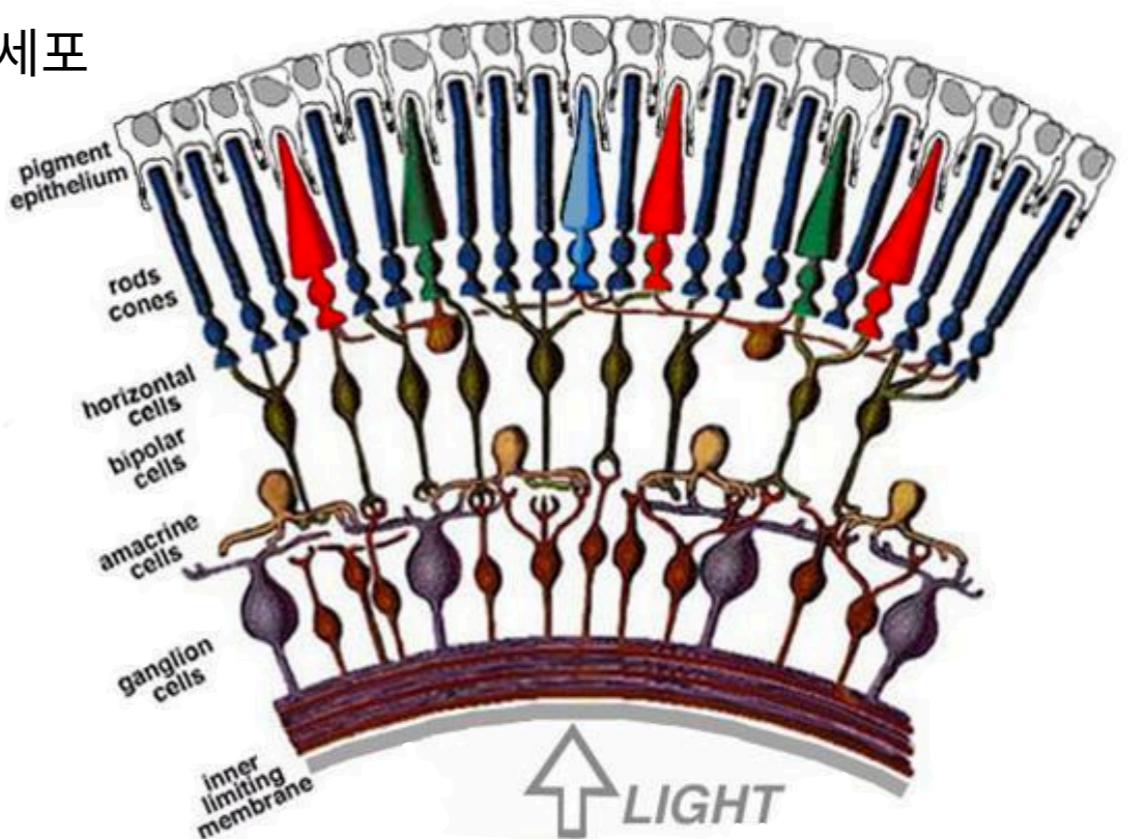
1. 컨볼루션 신경망

○ 망막과 수용장(receptive field)

■ 수용장

- 자극을 받을 때 특정한 시각피질세포에 영향을 미치는 망막의 부위
- 시각정보가 처리되는 최소 영역
- 신경절 세포(Retina ganglion cells, RGCs)

○ 정보를 눈으로부터 두뇌로 전달하는 책임을 지는 세포



1. 컨볼루션 신경망

○ 컨볼루션

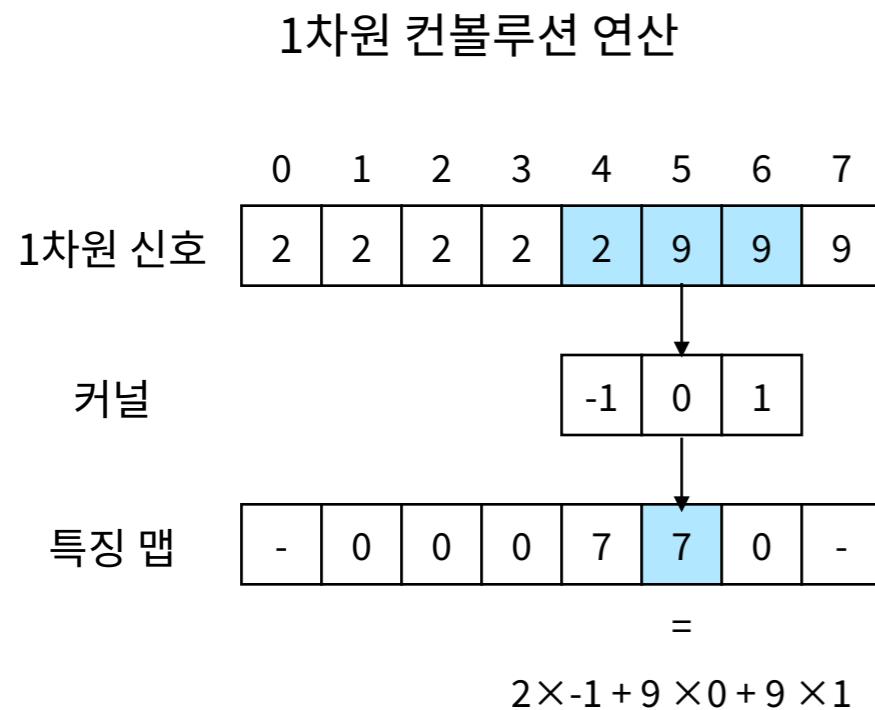
- 컨볼루션(Convolution) : 해부학의 대뇌 표면의 주름(뇌회)
- 수용장의 기능을 모사
 - 영역별 특징을 강화하는 필터 처리 연산
- 1980년 후쿠시마의 네오코그니트론
 - 수용장을 컨볼루션 연산으로 모방
 - 컨볼루션 신경망(CNN)의 원리를 최초로 제안
- 1998년 얀 르쿤(Yann Lecun) LeNet-5
 - 필기 숫자 인식 성능이 좋은 CNN 알고리즘
 - 미국의 수표 손글씨 자동 인식 시스템에 활용

2. CNN 구조

○ 특징 맵 추출

▣ 컨볼루션(Convolution) 연산

- 신호에서 특징을 추출하거나 신호를 변환하는데 사용하는 연산, 합성곱이라고도 함
 - 커널(Kernel) : 특징 추출을 위한 소규모 필터(Mask)
 - 특징맵(Feature map) : 커널을 사용하여 컨볼루션 연산을 통해 나온 결과

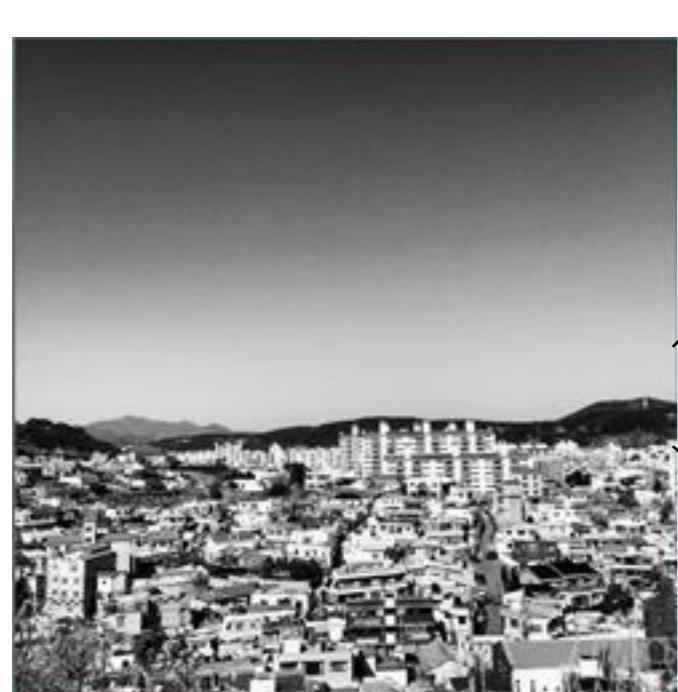


2. CNN 구조

○ 특징 맵 추출

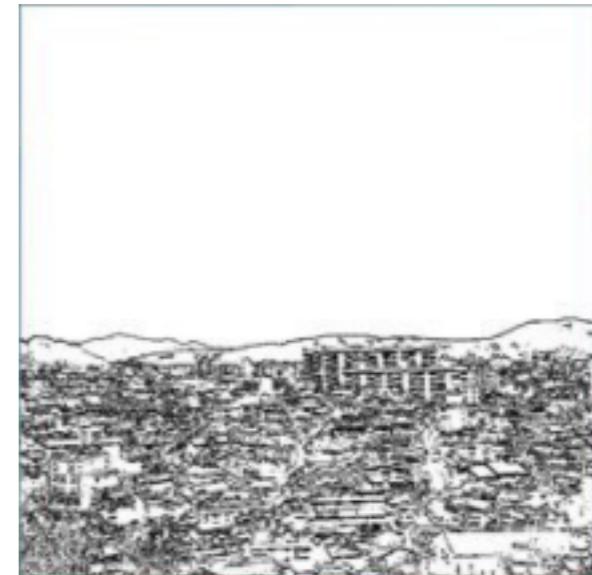
■ 컨볼루션(Convolution) 연산

- 수직 에지 커널과 수평 에지 커널



수평 에지 커널

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$



수직 에지 커널

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

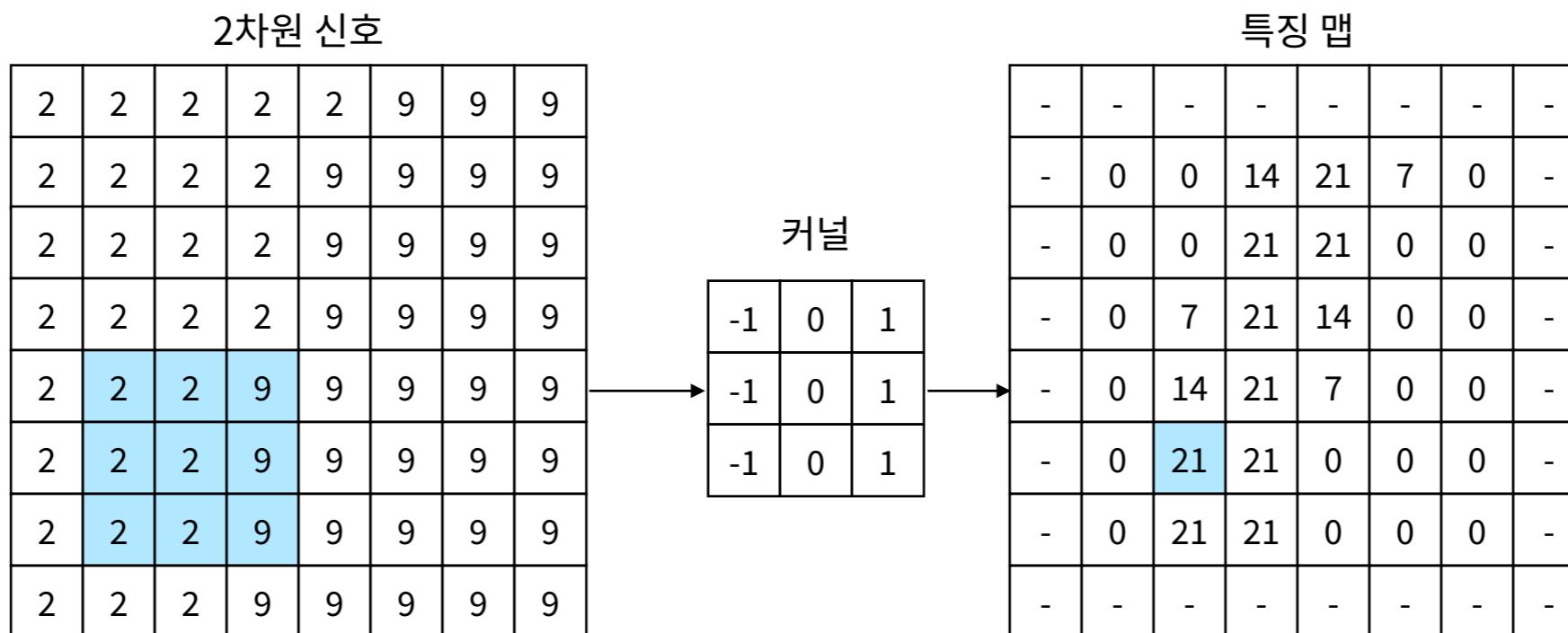


2. CNN 구조

○ 컨볼루션층과 풀링층

■ 컨볼루션층(Convolution Layer)

- 커널(Kernel) : 특징 추출을 위한 소규모 필터(Mask), 예제는 3x3 커널
- 보폭(Stride) : 커널의 이동 범위, 예제에서 stride = 1

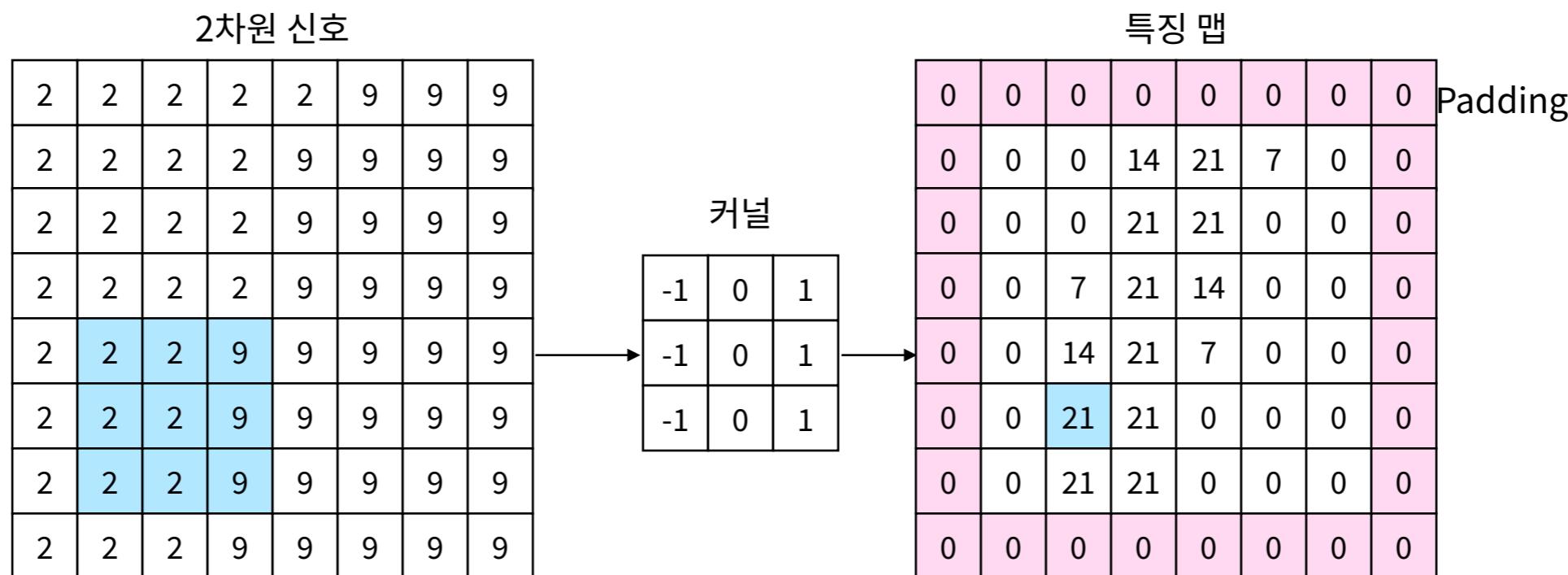


2. CNN 구조

○ 컨볼루션층과 풀링층

■ 컨볼루션층(Convolution Layer)

- 커널의 크기로 인해 계산을 할 수 없는 가장자리가 없어지는 문제 → 다층 컨볼루션의 경우 특징 맵이 입력 신호보다 너무 작아짐
- 덧대기(Padding) : 지정된 개수의 폭만큼 행과 열 값을 채우는 것(주로 0 또는 이웃 화소값을 채움)

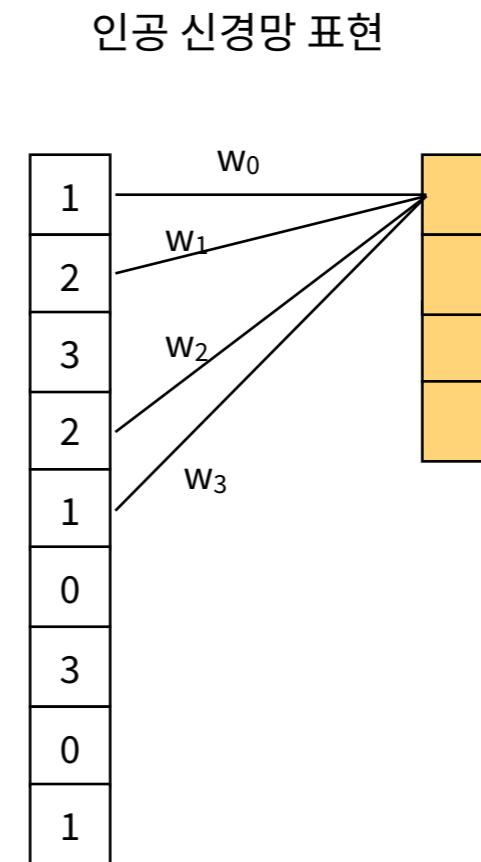
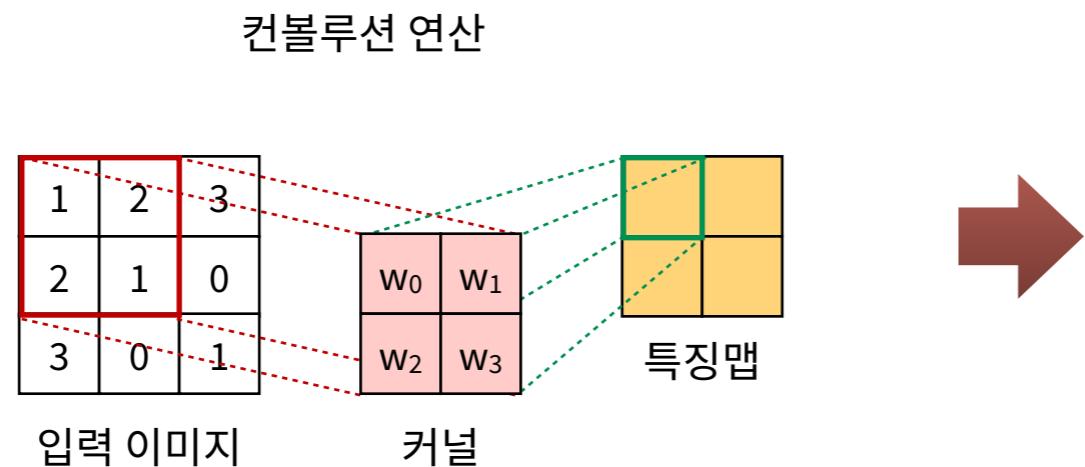


2. CNN 구조

○ 컨볼루션층과 풀링층

▣ 컨볼루션층(Convolution Layer)

- 컨볼루션 신경망은 다층 퍼셉트론보다 훨씬 적은 수의 가중치를 사용하여 공간적 구조 정보를 보존
 - 컨볼루션 층 : 컨볼루션 연산을 통해 특징 맵을 얻고, 활성화 함수를 지나는 연산을 하는 신경망의 은닉층
 - 활성함수 사용 가능

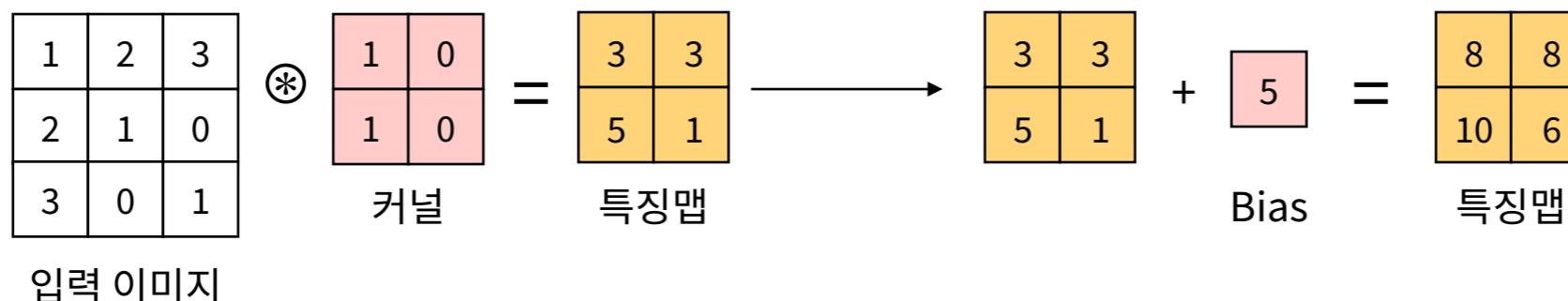


2. CNN 구조

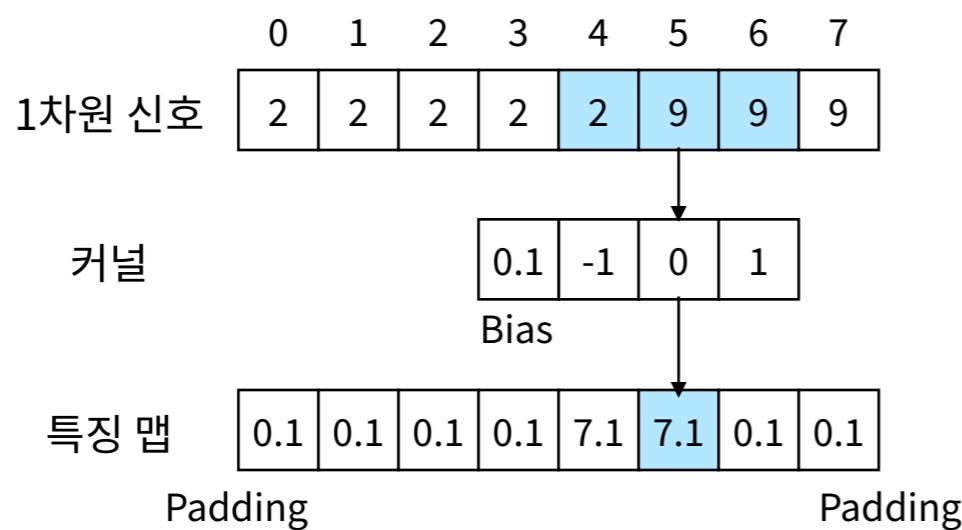
○ 컨볼루션층과 풀링층

■ 컨볼루션층(Convolution Layer)

- 편향(bias) 추가 가능 : 하나의 값만 존재하며 커널이 적용된 결과의 모든 원소에 더해짐



- 예) 바이어스를 가진 커널(1D)

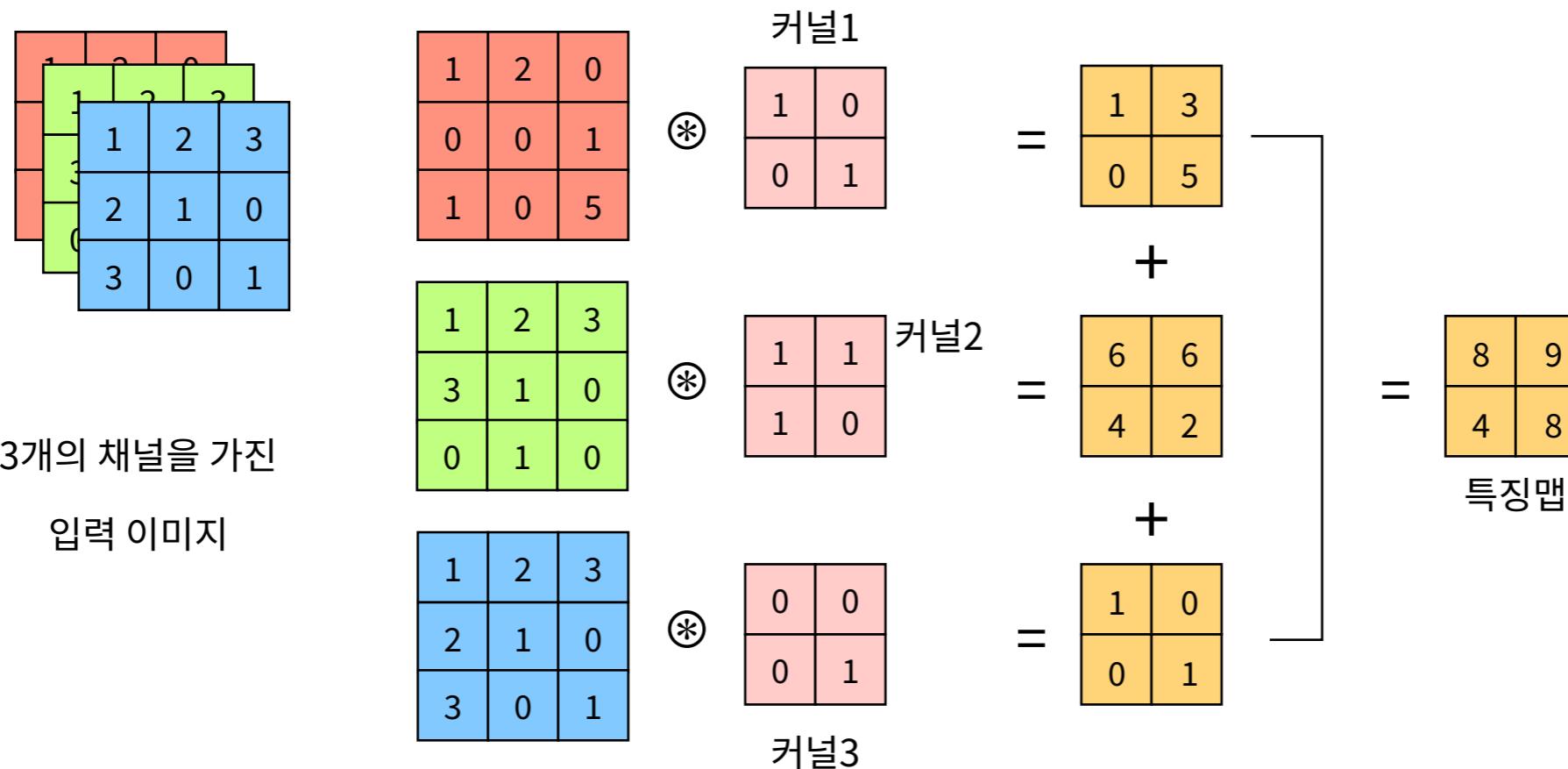


2. CNN 구조

○ 컨볼루션층과 풀링층

▣ 컨볼루션층(Convolution Layer)

- 다수의 채널을 가질 경우 컨볼루션 연산(예. RGB 3개의 채널을 갖는 영상)
 - 각 채널마다 커널이 필요

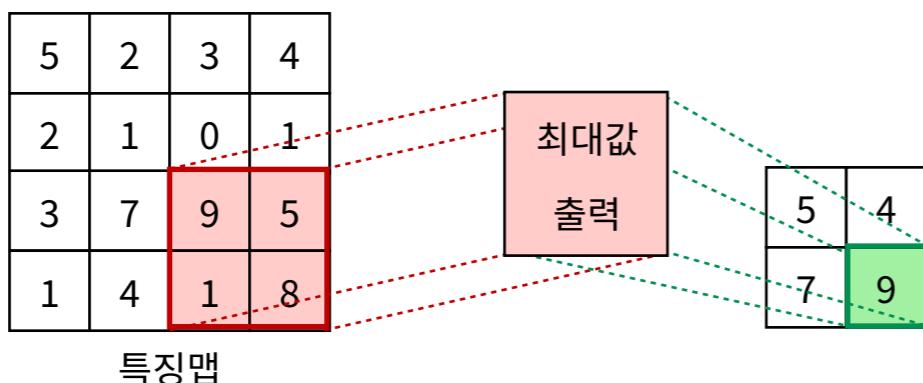


2. CNN 구조

○ 컨볼루션층과 풀링층

■ 풀링층(Pooling Layer)

- 컨볼루션층(합성곱 연산 + 활성화 함수) 다음에 풀링층 추가가 일반적
- 특징맵을 다운 샘플링하여 크기를 줄임 → 메모리 효율과 계산 속도를 끌어올리는 효과
- 풀링 연산
 - 최대 풀링(max pooling) : 커널 안의 화소중 최대값을 취함
 - 평균 풀링(average pooling) : 커널 안의 화소값들의 평균을 취함
- Stride = 2, 2x2 Kernel Max pooling 예



2. CNN 구조

○ 컨볼루션층과 풀링층

- 부분 연결성과 가중치 공유

- 부분 연결성

- 다층 퍼셉트론의 각 층은 완전 연결(Fully Connected)
 - 컨볼루션 신경망은 커널의 크기 만큼만 부분 연결

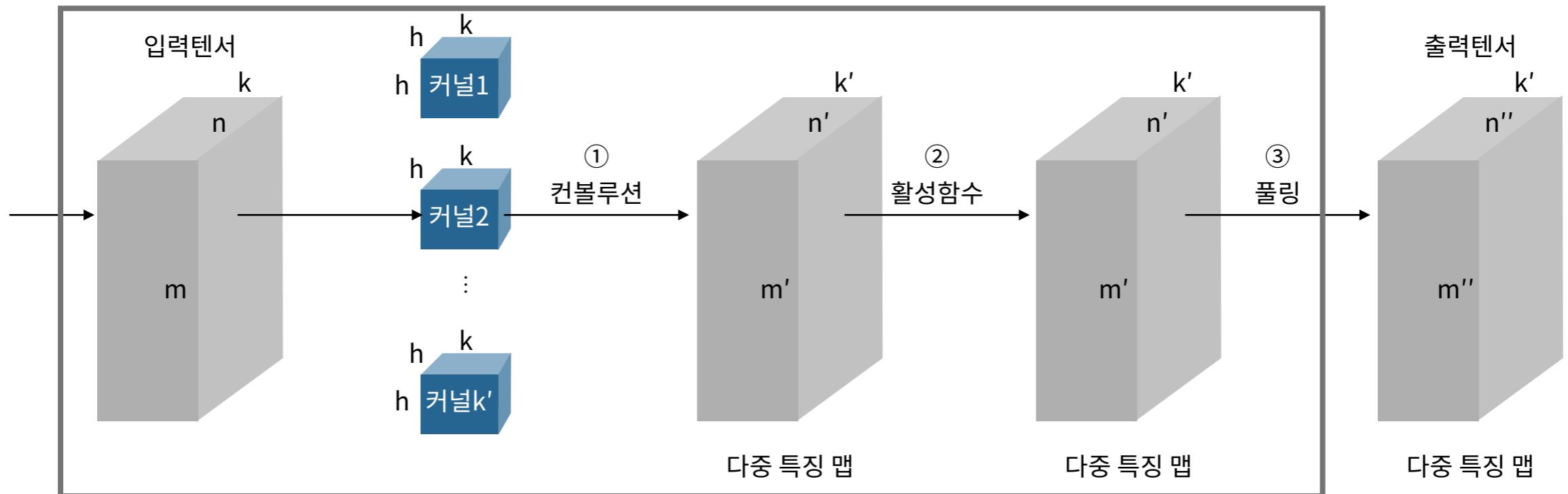
- 가중치 공유

- 다층 퍼셉트론의 인접한 두 층의 노드 쌍을 연결하는 에지에 가중치가 있음
 - 컨볼루션 신경망은 커널이 가중치 역할을 하며 모든 노드가 같은 가중치를 공유 → 학습이 최적화해야 하는 매개변수 개수를 획기적으로 줄여줌

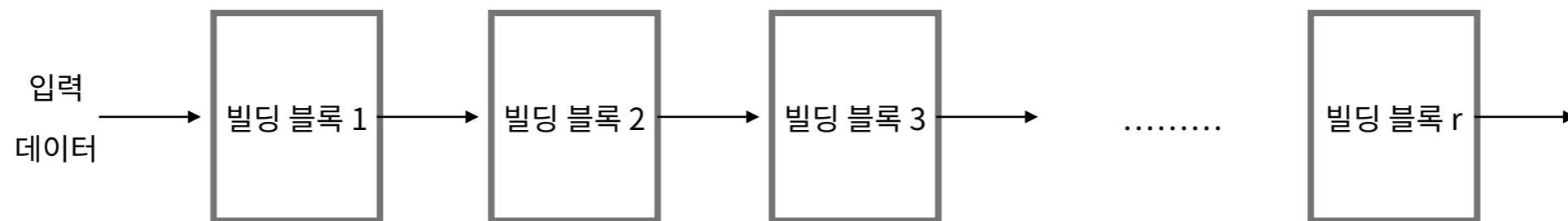
2. CNN 구조

○ 컨볼루션 신경망의 빌딩 블록

■ 컨볼루션 빌딩블록



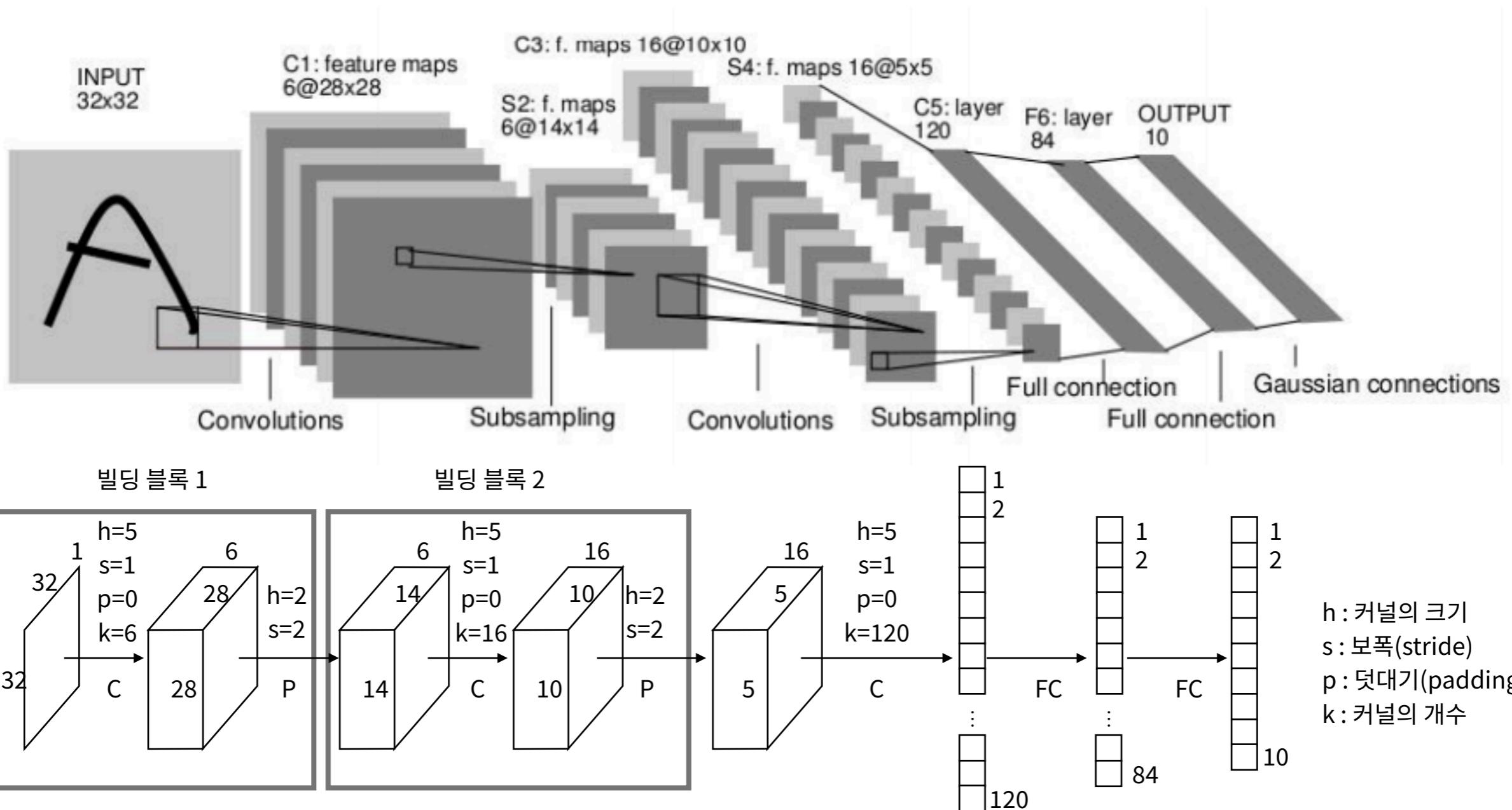
■ 컨볼루션 신경망



2. CNN 구조

○ 컨볼루션 신경망의 빌딩 블록

- LeNet-5 사례(LeCun 1998)



3. CNN 학습

○ 컨볼루션 신경망의 학습

- 학습을 통한 최적화 대상이 커널
 - 손실 함수와 옵티마이저
 - 손실함수 : 신경망이 범하는 오류를 측정, 다층 퍼셉트론과 같은 손실함수를 사용
 - 옵티마이저 : 손실함수의 최저점을 찾아감, 다층 퍼셉트론과 같은 옵티마이저를 사용
- 통째 학습
 - 고전적인 컴퓨터 비전 : 수작업 특징 → 사람의 직관에 따라 특징을 추출
 - 딥러닝
 - 특징 학습(feature learning) : 특징을 학습으로 알아냄
 - 통째 학습(end-to-end learning) : 특징과 최종 출력 전체 과정을 한꺼번에 학습

3. CNN 학습

○ 컨볼루션 신경망의 성능이 월등한 이유

- 통째 학습 : 특징 추출과 분류를 동시에 최적화
- 특징 학습 : 특징 맵은 컨볼루션 과정을 거치면서 저급 특징에서 고급 특징으로 발전
- 신경망의 깊이를 더욱 깊게하여 풍부한 특징을 추출
- 데이터의 원래 구조를 유지한 채 특징을 추출

4. CNN 프로그래밍

○ 필기 숫자 인식(LeNet-5 재현)

CO LeNet5_MNIST.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

```
[ ] import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.optimizers import Adam

[ ] # MNIST 데이터셋을 읽고 신경망에 입력할 형태로 변환
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(60000, 28, 28, 1)
x_test = x_test.reshape(10000, 28, 28, 1)
x_train = x_train.astype(np.float32)/255.0
x_test = x_test.astype(np.float32)/255.0
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

4. CNN 프로그래밍

○ 필기 숫자 인식(LeNet-5 재현)

```
[ ] # LeNet-5 신경망 모델 설계
cnn = Sequential()
cnn.add(Conv2D(6, (5,5), padding = 'same', activation ='relu',
              input_shape=(28, 28, 1)))
cnn.add(MaxPooling2D(pool_size=(2,2)))
cnn.add(Conv2D(16, (5,5), padding='valid', activation='relu'))
cnn.add(MaxPooling2D(pool_size=(2,2)))
cnn.add(Conv2D(120, (5,5), padding='valid', activation='relu'))
cnn.add(Flatten())
cnn.add(Dense(84, activation='relu'))
cnn.add(Dense(10, activation='softmax'))
```

▶ # 신경망 모델 학습

```
cnn.compile(loss='categorical_crossentropy', optimizer=Adam(),
            metrics=['accuracy'])
hist = cnn.fit(x_train, y_train, batch_size=128, epochs=30,
                validation_data=(x_test, y_test), verbose=2)
```

신경망 모델 정확률 평가

```
res = cnn.evaluate(x_test, y_test, verbose=0)
print('정확도는 ', res[1]*100, '%')
```

4. CNN 프로그래밍

○ 필기 숫자 인식(LeNet-5 재현)

- Conv2D (6, (5, 5), padding='same', activation='relu', input_shape=(28, 28, 1))
 - Conv2D : 영상 처리에 주로 사용되는 컨볼루션 레이어
 - 첫번째 인자 : filter, 컨볼루션 커널(필터)의 수
 - 두번째 인자 : kernel_size, 커널의 크기
 - strides : 보폭의 크기, default=(1,1)
 - padding : 덧대기, ‘valid’ = 유효한 영역만 출력(default), ‘same’=출력 이미지 사이즈가 입력 사이즈와 동일
 - activation: 활성화 함수 설정, ‘linear’, ‘relu’, ‘sigmoid’, ‘softmax’ 등
 - input_shape : 샘플 수를 제외한 입력 형태

4. CNN 프로그래밍

○ 필기 숫자 인식(LeNet-5 재현)

- ▣ MaxPooling2D (pool_size=(2,2))

- MaxPooling2D : 지역적인 사소한 변화가 영향을 미치지 않도록 함(MaxPooling)
 - pool_size : 수직, 수평 축소 비율
 - strides : default = pool_size

- ▣ Flatten ()

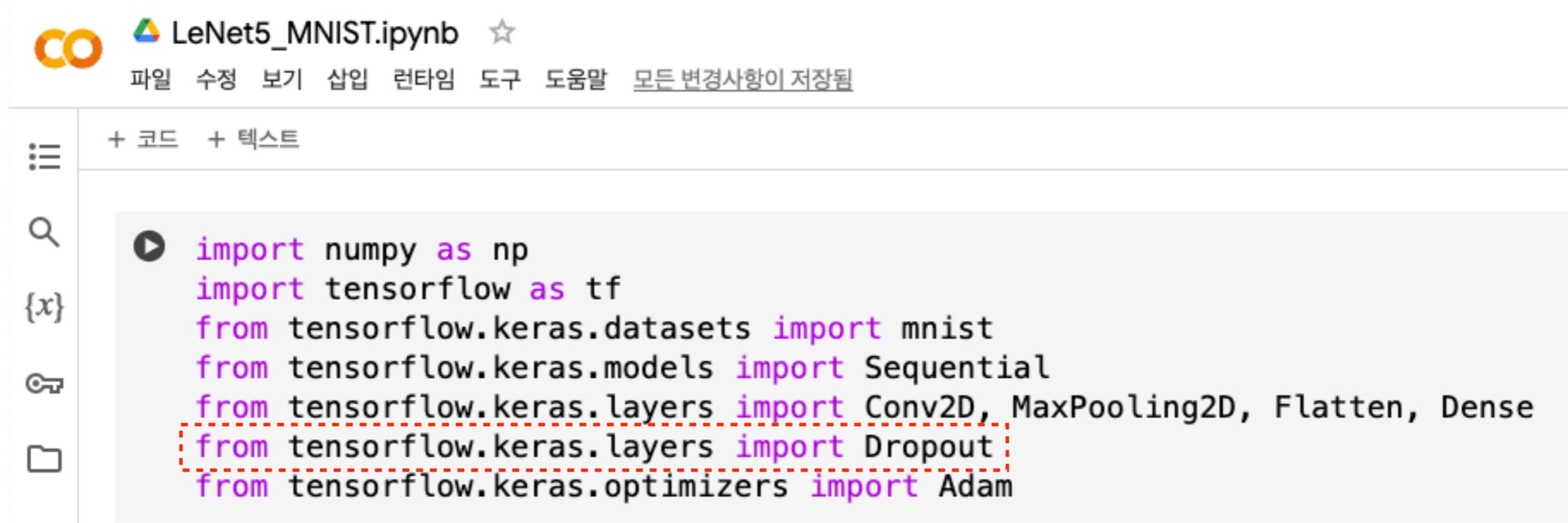
- 텐서를 일차원으로 변환해주는 기능

4. CNN 프로그래밍

○ 컨볼루션 신경망의 유연한 구조

■ 구조 변경

- C → C → P → dropout → FC → dropout → FC
- dropout : 과적합을 방지하기 위해 임의로 인풋 유닛을 0으로 설정하는 비율(rate) 설정



```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.optimizers import Adam
```

4. CNN 프로그래밍

○ 컨볼루션 신경망의 유연한 구조

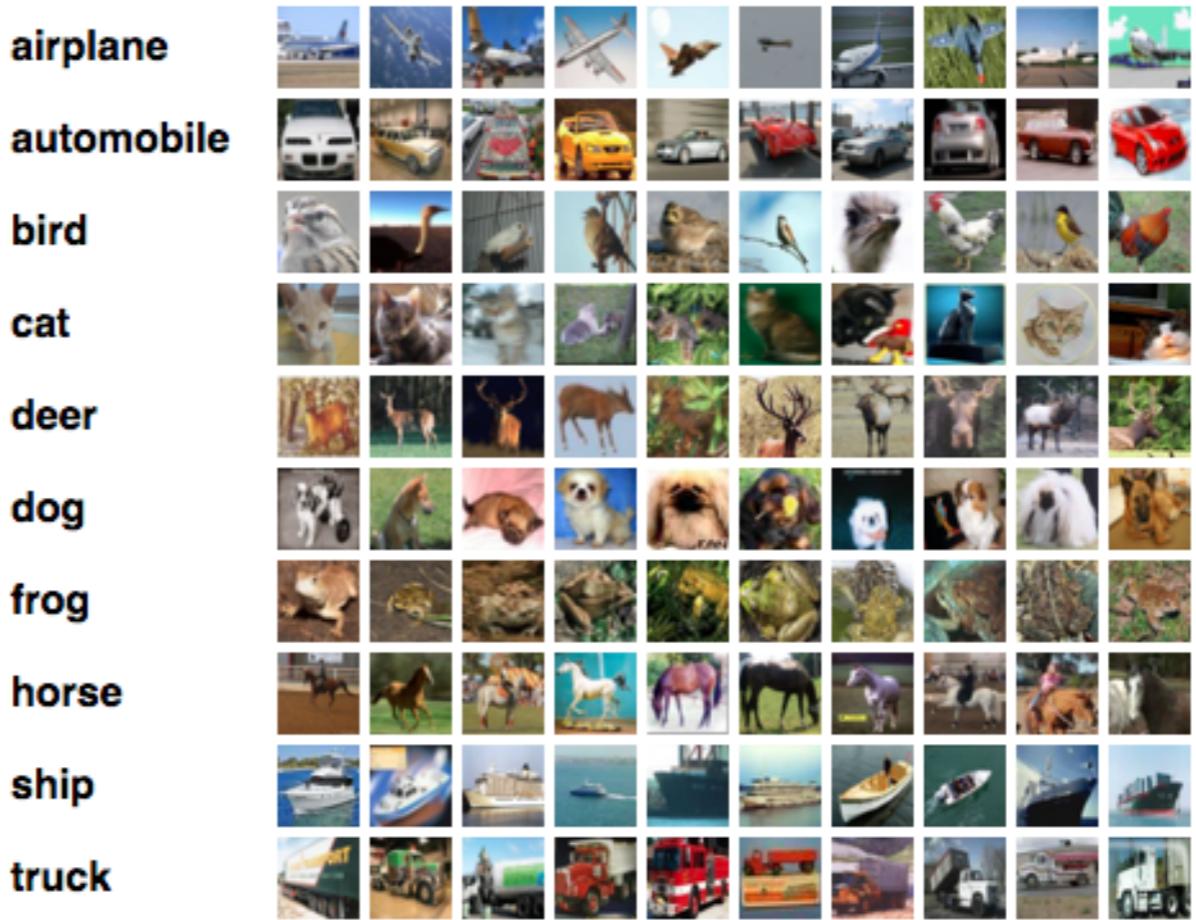
■ 구조 변경

```
▶ # LeNet-5 신경망 모델 설계
cnn = Sequential()
cnn.add(Conv2D(32, (3,3), activation ='relu', input_shape=(28, 28, 1)))
cnn.add(Conv2D(64, (3,3), activation='relu'))
cnn.add(MaxPooling2D(pool_size=(2,2)))
cnn.add(Dropout(0.25))
cnn.add(Flatten())
cnn.add(Dense(128, activation='relu'))
cnn.add(Dropout(0.25))
cnn.add(Dense(10, activation='softmax'))
```

4. CNN 프로그래밍

○ 자연영상 인식

- ImageNet, MSCoCo : 방대한 양의 자연 영상 데이터 셋
 - CIFAR-10
 - 10 클래스 : {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck}
 - 60000개 이미지(train : 50000)
 - 32×32 color map



4. CNN 프로그래밍

○ 자연영상 인식

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** CIFAR10.ipynb
- Toolbar:** 파일, 수정, 보기, 삽입, 런타임, 도구, 도움말, 저장 중...
- Sidebar:** + 코드, + 텍스트, search, copy, paste, refresh, and file/folder icons.
- Code Cells:**
 - [1]

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.optimizers import Adam
```
 - # CIFAR-10 데이터셋을 읽고 신경망에 입력할 형태로 변환

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train.astype(np.float32)/255.0
x_test = x_test.astype(np.float32)/255.0
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

4. CNN 프로그래밍

○ 자연영상 인식

```
[ ] # CIFAR10 신경망 모델
cnn = Sequential()
cnn.add(Conv2D(32, (3,3), activation='relu', input_shape=(32, 32, 3)))
cnn.add(Conv2D(32, (3,3), activation='relu'))
cnn.add(MaxPooling2D(pool_size=(2,2)))
cnn.add(Dropout(0.25))
cnn.add(Conv2D(64, (3,3), activation='relu'))
cnn.add(Conv2D(64, (3,3), activation='relu'))
cnn.add(MaxPooling2D(pool_size=(2,2)))
cnn.add(Dropout(0.25))
cnn.add(Flatten())
cnn.add(Dense(512, activation='relu'))
cnn.add(Dropout(0.5))
cnn.add(Dense(10, activation='softmax'))
```

```
[ ] # 신경망 모델 학습
cnn.compile(loss='categorical_crossentropy', optimizer=Adam(),
            metrics=['accuracy'])
hist = cnn.fit(x_train, y_train, batch_size=128, epochs=30,
                validation_data=(x_test, y_test), verbose=2)
```

4. CNN 프로그래밍

○ 자연영상 인식

```
# 신경망 모델 정확률 평가  
res = cnn.evaluate(x_test, y_test, verbose=0)  
print('정확도는 ', res[1]*100, '%')
```

```
▶ import matplotlib.pyplot as plt  
  
# 정확률 그래프  
plt.plot(hist.history['accuracy'])  
plt.plot(hist.history['val_accuracy'])  
plt.title('Model Accuracy')  
plt.xlabel('Epoch')  
plt.ylabel('Accuracy')  
plt.legend(['Train', 'Validation'], loc='best')  
plt.grid()  
plt.show()
```

4. CNN 프로그래밍

○ 자연영상 인식

```
# 손실함수 그래프
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='best')
plt.grid()
plt.show()
```

4. CNN 프로그래밍

○ 학습된 모델 저장과 재활용

■ save()

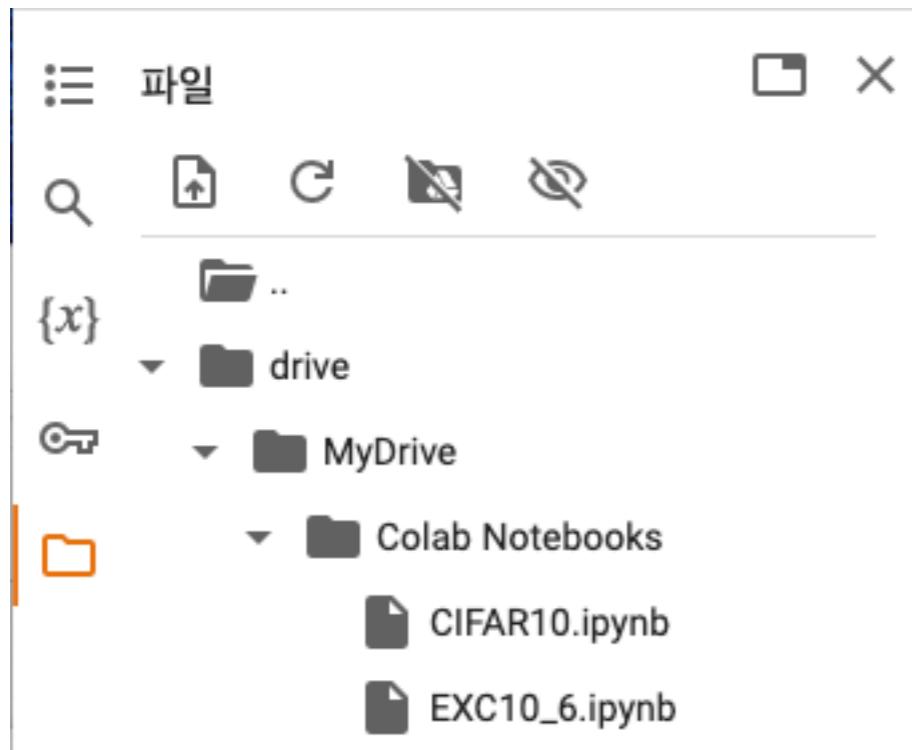
- 학습된 모델을 저장
- HDF5 파일 형식으로 저장 → 확장자는 h5로 지정

```
cnn.save('my_cnn.h5')
```

4. CNN 프로그래밍

○ 학습된 모델 저장과 재활용

- Colab 사용시 Google Drive에 파일 저장하기

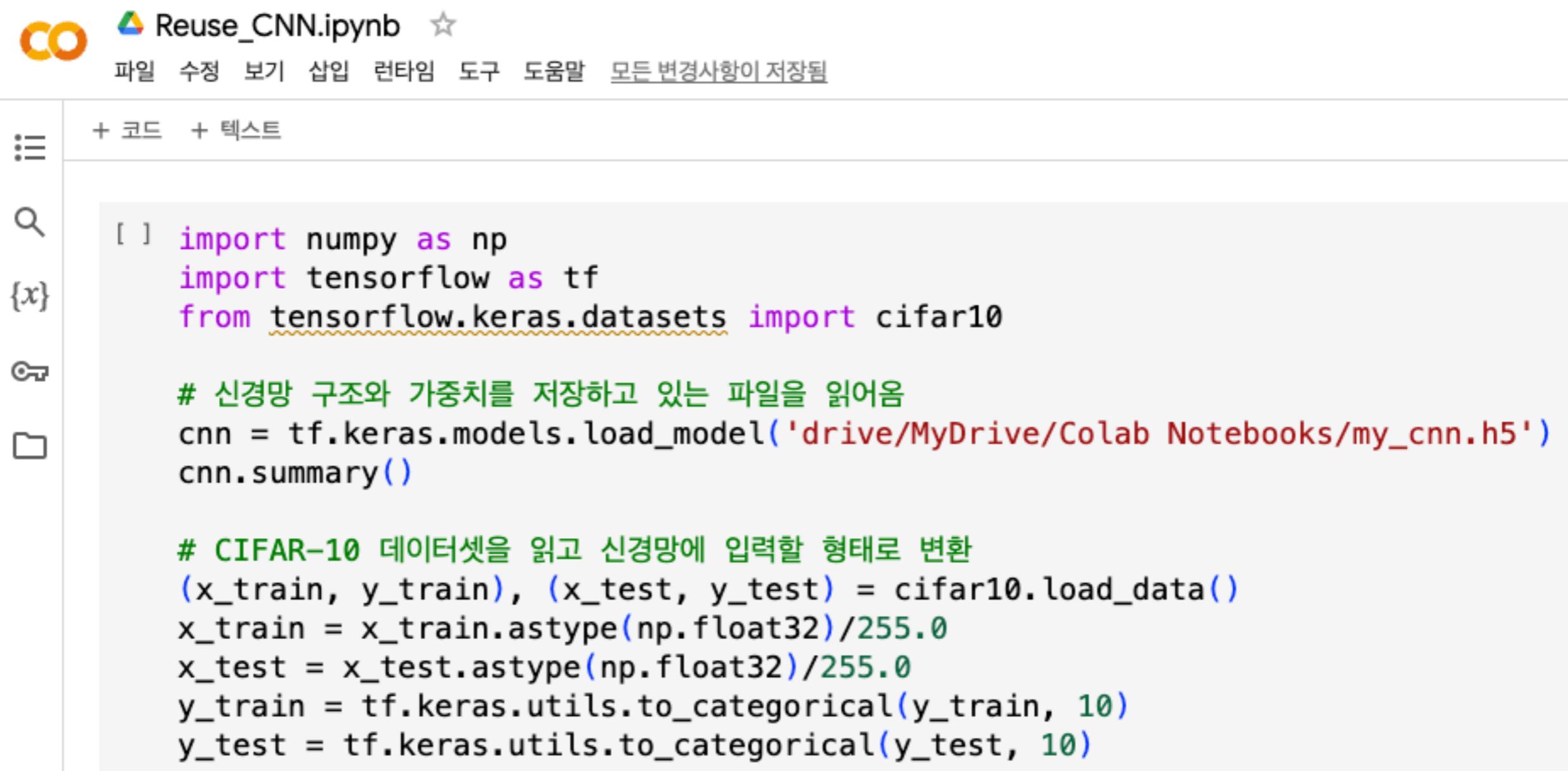


```
▶ cnn.save('drive/MyDrive/Colab Notebooks/my_cnn.h5')
```

4. CNN 프로그래밍

○ 학습된 모델 저장과 재활용

■ 저장된 학습 모델을 불러오기



The screenshot shows a Google Colab notebook titled "Reuse_CNN.ipynb". The notebook interface includes a toolbar with icons for file operations, a sidebar with search, refresh, and file/folder icons, and a main code editor area.

```
[ ] import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import cifar10

# 신경망 구조와 가중치를 저장하고 있는 파일을 읽어옴
cnn = tf.keras.models.load_model('drive/MyDrive/Colab Notebooks/my_cnn.h5')
cnn.summary()

# CIFAR-10 데이터셋을 읽고 신경망에 입력할 형태로 변환
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train.astype(np.float32)/255.0
x_test = x_test.astype(np.float32)/255.0
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

4. CNN 프로그래밍

○ 학습된 모델 저장과 재활용

■ 저장된 학습 모델을 불러오기

```
res = cnn.evaluate(x_test, y_test, verbose=0)
print('정확도는 ', res[1]*100, '%')
```

5. 딥러닝의 규제

○ 딥러닝의 규제

- 모델이 가진 학습 가능한 가중치 개수가 많아지면 복잡도가 증가
 - 과잉 적합(과대 적합) 가능성
 - 모델이 훈련 세트에서는 좋은 성능을 내지만 검증 세트에서 낮은 성능을 보임
 - 정확도/epochs 그래프의 측정 성능 간격이 큼
 - 훈련 세트의 다양성이 없어서 나오는 경우임
 - 규제(regulation) 기법을 이용해 과잉적합을 방지
 - 데이터 증대
 - 드롭아웃
 - 가중치 감쇠

5. 딥러닝의 규제

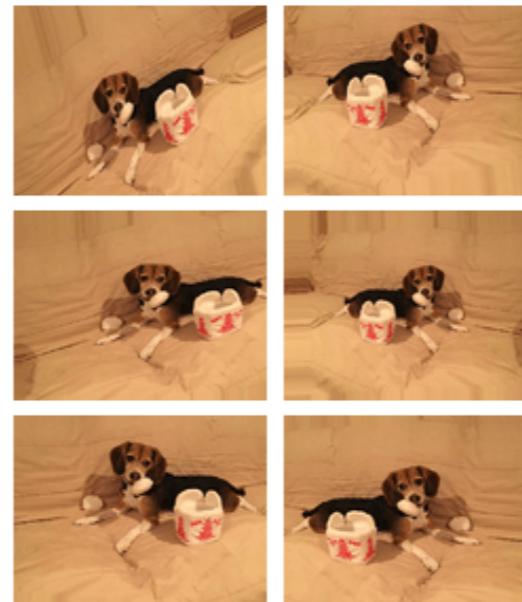
○ 데이터 증대

- 데이터 수집에 많은 시간과 비용이 따름
 - 데이터 증대(data augmentation)
 - 주어진 데이터를 적절하게 변형해 인위적으로 증대하는 전략
 - 영상의 경우 상하좌우 이동, 회전, 반전, 명암 조절 등의 변화

Input Image



Augmented Images



Keras

5. 딥러닝의 규제

○ 데이터 증대

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** DA.ipynb
- Toolbar:** 파일, 수정, 보기, 삽입, 런타임, 도구, 도움말
- Code Cells:** + 코드, + 텍스트
- Left Sidebar Icons:** CO, 🔍, {x}, 🔑, 폴더
- Code Content:**

```
✓  import numpy as np
    from tensorflow.keras.datasets import cifar10
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    import matplotlib.pyplot as plt

    # CIFAR-10의 클래스 이름
    class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog',
                   'frog', 'horse', 'ship', 'truck']

    # CIFAR-10 데이터 셋을 읽고 앞에서 12개에 대해서만 데이터 증대를 적용
    (x_train, y_train), (x_test, y_test) = cifar10.load_data()
    x_train = x_train.astype(np.float32)/255.0
    x_train = x_train[0:12,]
    y_train = y_train[0:12,]
```

5. 딥러닝의 규제

○ 데이터 증대

```
# 증대할 영상 그리기
plt.figure(figsize=(16,2))
plt.suptitle('First 12 images in the train set')
for i in range(12):
    plt.subplot(1, 12, i+1)
    plt.imshow(x_train[i])
    plt.xticks([]); plt.yticks([])
    plt.title(class_names[int(y_train[i])])
```

✓ 0초

▶ # 영상 증대기 생성

```
gen_size = 6    # 한번에 생성하는 양
generator = ImageDataGenerator(rotation_range=30.0, width_shift_range=0.2,
                               height_shift_range=0.2, horizontal_flip=True)
gen = generator.flow(x_train, y_train, batch_size=gen_size)
```

5. 딥러닝의 규제

○ 데이터 증대

```
# 첫번째 증대하고 그리기
img, label = gen.next()
plt.figure(figsize=(16,3))
plt.suptitle('Generator trial 1')
for i in range(gen_size):
    plt.subplot(1, gen_size, i+1)
    plt.imshow(img[i])
    plt.xticks([]); plt.yticks([])
    plt.title(class_names[int(label[i])])

# 두번째 증대하고 그리기
img, label = gen.next()
plt.figure(figsize=(16,3))
plt.suptitle('Generator trial 2')
for i in range(gen_size):
    plt.subplot(1, gen_size, i+1)
    plt.imshow(img[i])
    plt.xticks([]); plt.yticks([])
    plt.title(class_names[int(label[i])])
```

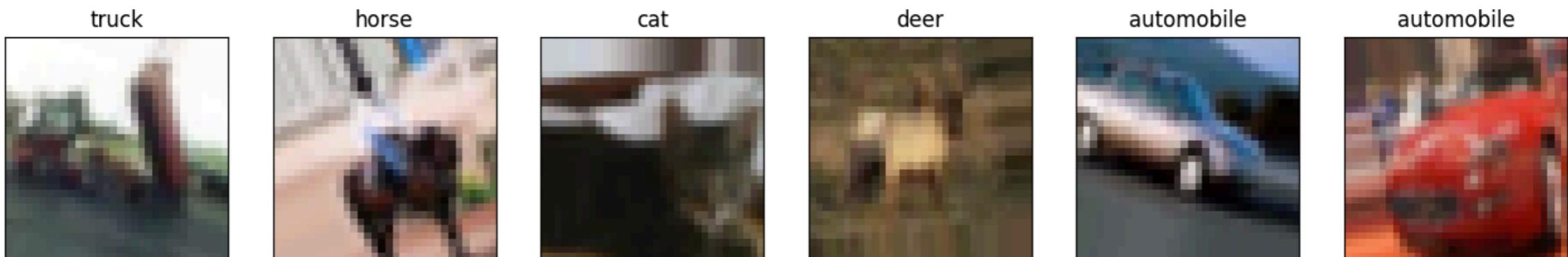
5. 딥러닝의 규제

○ 데이터 증대

First 10 images in the train set



Generator trial 1



Generator trial 2

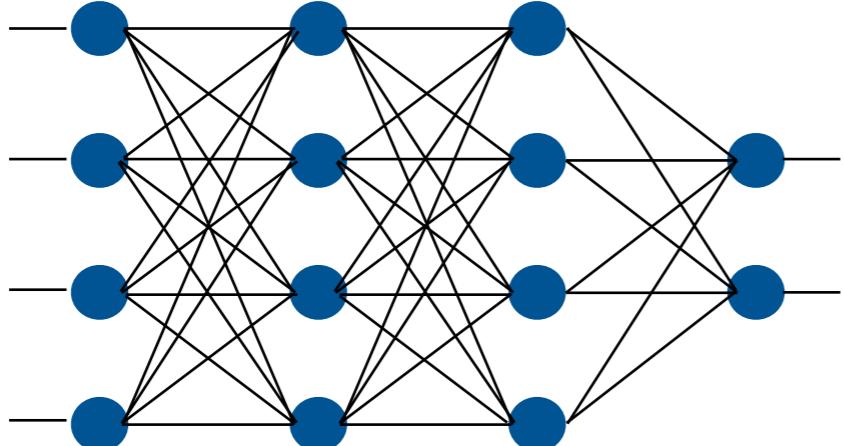


5. 딥러닝의 규제

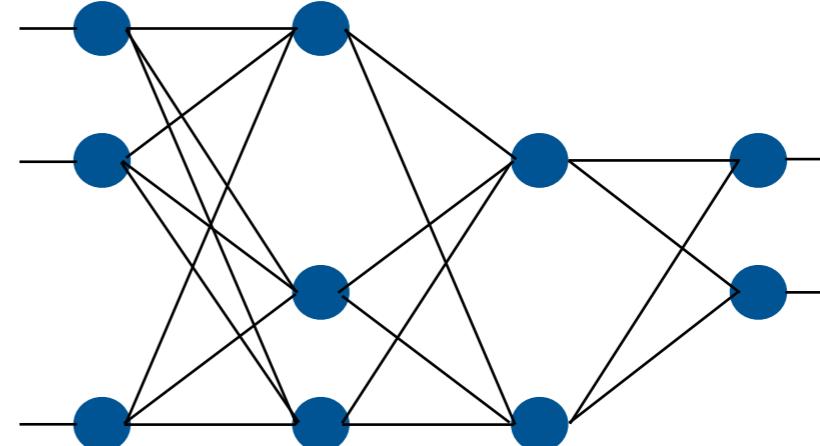
○ 드롭 아웃

- 일정 비율의 가중치를 임의로 선택해 학습에 참여시키지 않음
 - 지나친 편향을 추가하지 않고 노이즈를 추가하는 효과
 - 일정한 비율의 에지 입력을 0으로 설정
 - 불능이 될 에지는 샘플마다 독립적으로 정함

원래의 신경망(4-4-4-2 구조)



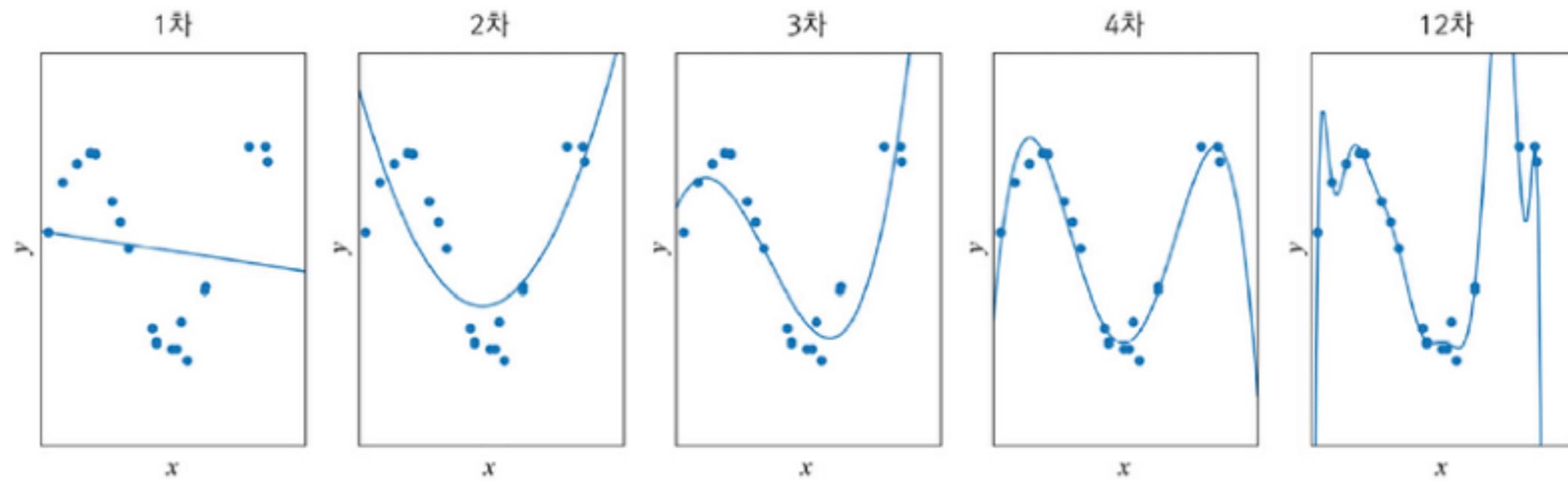
드롭 아웃 적용 예



5. 딥러닝의 규제

○ 가중치 감소

- 20개 샘플 데이터 : x 는 특징, y 는 레이블
- 12차 곡선 방정식 : $y = 1005.7x^{12} - 27774.4x^{11} + \dots - 22852612.5x^1 - 12.8$
- 노이즈까지 모두 학습한 모델은 가중치가 높음
- 가중치를 적게 만들 수록 과잉적합이 어느정도 해결됨



5. 딥러닝의 규제

○ 가중치 감쇠

- 가중치 감쇠(weight decay) : 가중치를 작게 만들어 부드럽게 만드는 기법

- 신경망 가중치 U 에 대한 손실함수

$$J(\mathbf{U}^1, \mathbf{U}^2, \dots, \mathbf{U}^L) = \frac{1}{|M|} \sum_{x \in M} \|y - o\|^2$$

- 오차를 최소화

- 가중치 감쇠 = 오차 최소화 + 가중치 값 최소화

- L1 규제 : 라소 정규화(Lasso regularization)

- 손실함수에 가중치의 절대값인 L1 노름(norm)을 추가

- L2 규제 : 리지 정규화(ridge regularization)

- 손실함수에 가중치의 제곱인 L2 노름을 추가

5. 딥러닝의 규제

○ 가중치 감소

■ 텐서플로의 가중치 감소

```
from tensorflow.keras import regularizers  
  
model.add(Dense(64, input_dim=64, kernel_regularizer=regularizers.l2(0.01),  
               activity_regularizer=regularizers.l1(0.01)))
```

- kernel_regularizer : 가중치에 적용
- bias_regularizer : 바이어스에 적용
- activity_regularizer : 활성 함수의 결과에 적용