



Group 3

CNN 모델 및 시각화

김서윤, 김아진, 서윤, 이철민

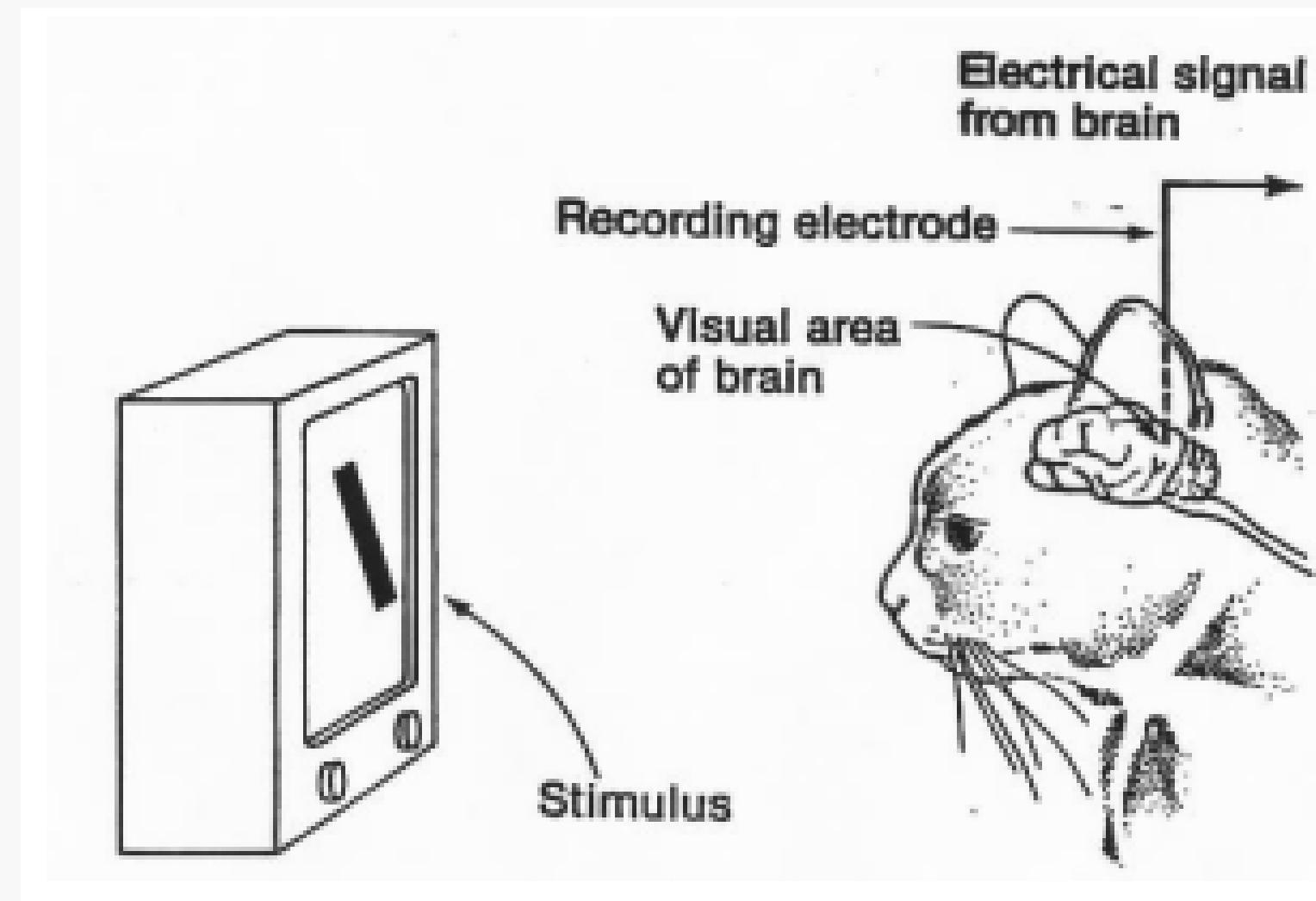
2024. 08. 10



CNN 개념

Convolutional Neural Network = 합성곱 신경망

이미지 데이터로부터 feature map을 자동으로 추출해 이미지의 특징을 학습하는 모델



[CNN 생성 배경]

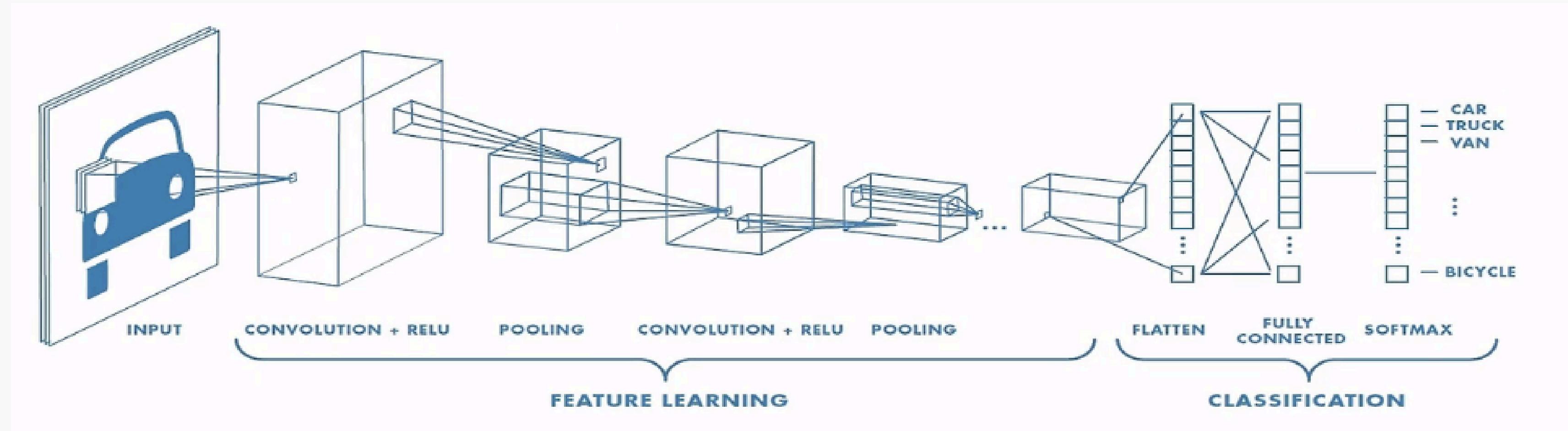
고양이가 물체를 어떻게 인식하는지를 알아보기 위해 고양이의 머리에 전극을 꽂은 채로 영상을 보여주면서 고양이 뇌의 시각 피질 영역이 어떤 반응을 하는지 지켜보는 실험을 함.

>>> 위 실험을 배경으로 CNN으로 발전

인간의 시신경 구조를 모방한 기술이기 때문에 사람이 여러 데이터를 보고 기억한 후 무엇인지 맞추는 것과 유사

CNN 구조

이미지의 특징을 추출하는 부분과 클래스를 분류하는 부분으로 나눌 수 있음



- 특징 추출

Convolution Layer : CNN에서 입력 데이터의 특징(feature)을 추출하는 레이어

Pooling Layer : Convolution Layer 다음에 위치하며 선택적인 레이어

- 이미지 분류

Flatten Layer : 이미지 형태의 데이터를 배열 형태로 만드는 레이어

Fully Connected Layer : 이미지 분류를 위한 레이어

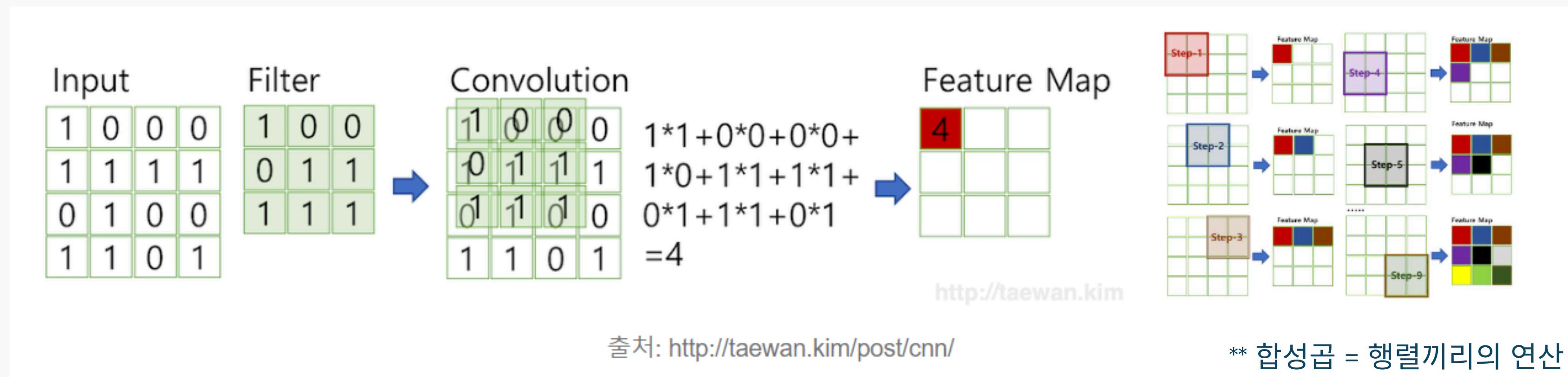
Filter (kernel, mask)

feature map을 생성하는 kernel (filter)

feature map = 이미지의 특징을 담은 맵

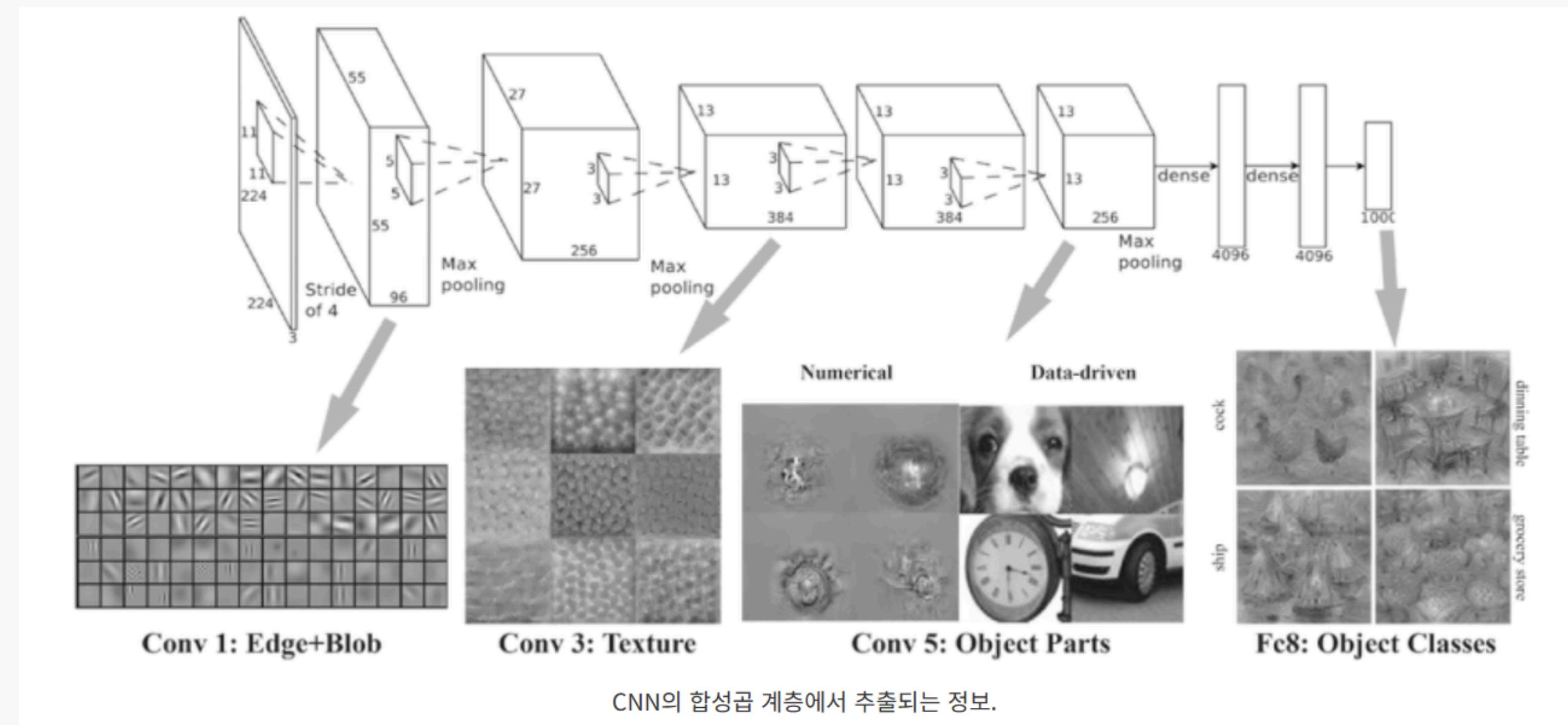
즉, 필터를 통해 만드는 최종 결과물이 Feature Map

filter의 목적 = 이미지의 특징을 찾아내는 것



- filter 크기에 따라 생성되는 Feature Map의 크기는 달라짐
- 생성된 feature map이 또 다른 레이어에 들어갈 때에는 Output이 아닌 Input이 됨
- 작은 Convolutional filter를 여러개 쌓는 것이 큰 Convolution filter 1개를 쌓는 것보다 더 좋은 성능을 낼 수 있음

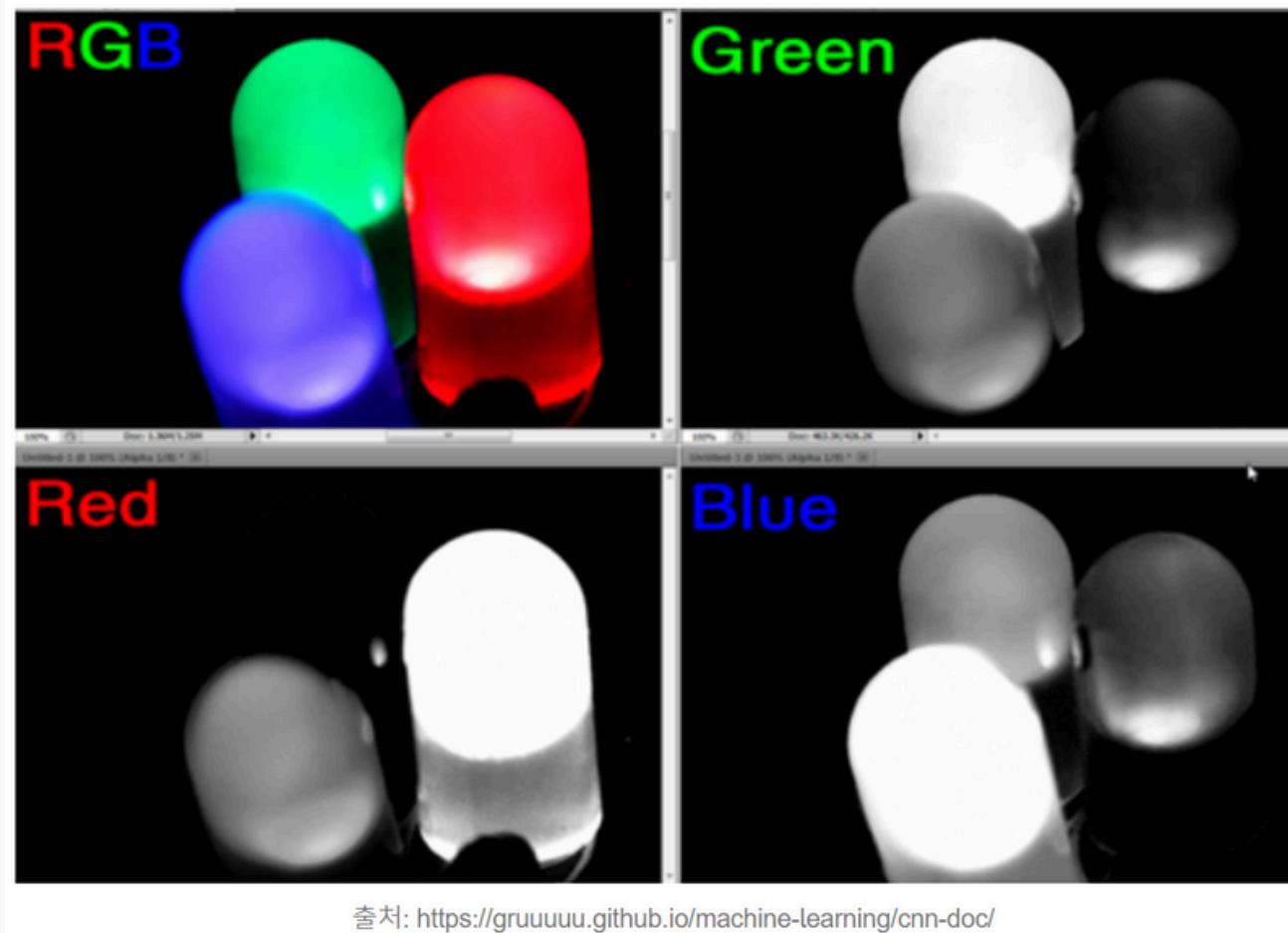
Filter (kernel, mask)



위 그림과 같이 합성곱 계층을 여러 겹 쌓으면, **층이 깊어지면서 더 복잡하고 추상화된 정보가 추출되기 때문** 1번째 층은 에지와 블롭, 3번째 층은 텍스쳐, 5번째 층은 사물의 일부, 마지막 완전연결 계층은 사물의 클래스에 뉴런이 반응

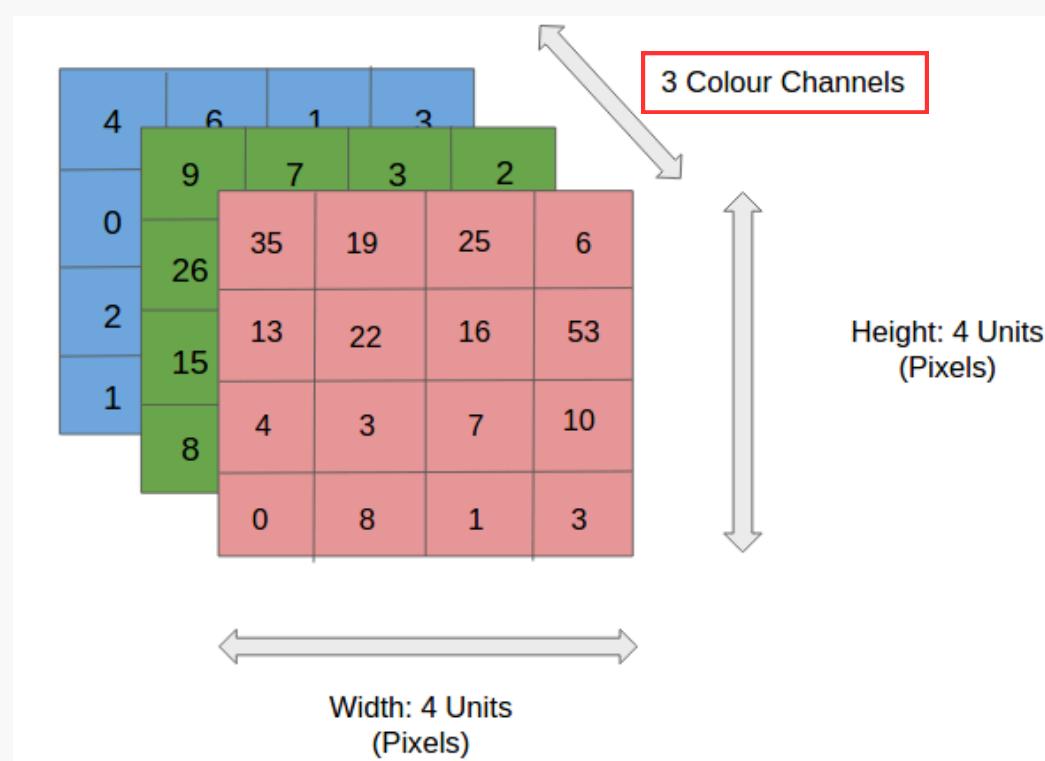
>>> 즉, **층이 깊어지면서 뉴런이 반응하는 대상이 단순한 모양에서 '고급' 정보로 변화**

Channel



우리가 알고 있는 Color 이미지
-> RGB 3개로 표현된 3차원 데이터

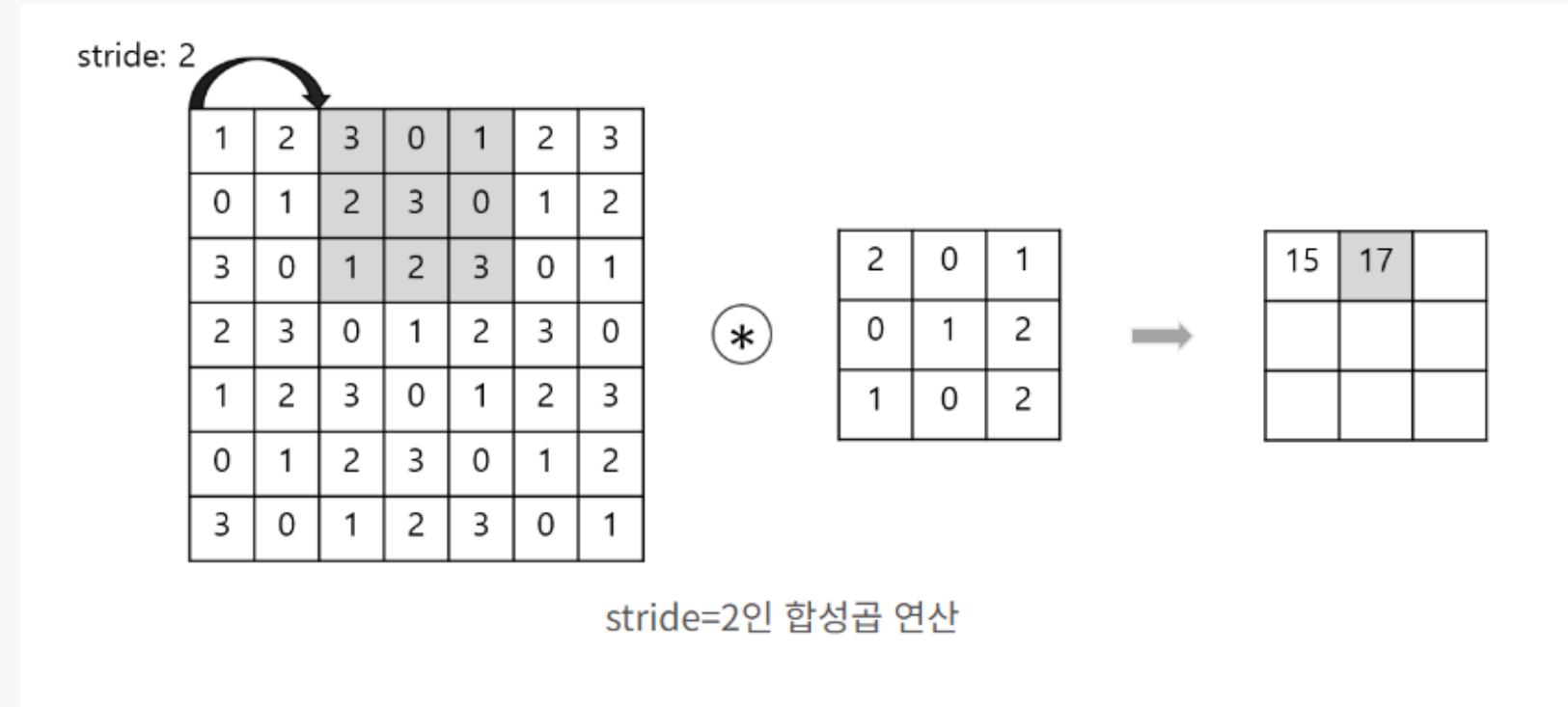
red channel, blue channel, green channel로 이루어져 있음
+ 흑백 이미지의 경우 하나의 Channel로 이루어져 있음



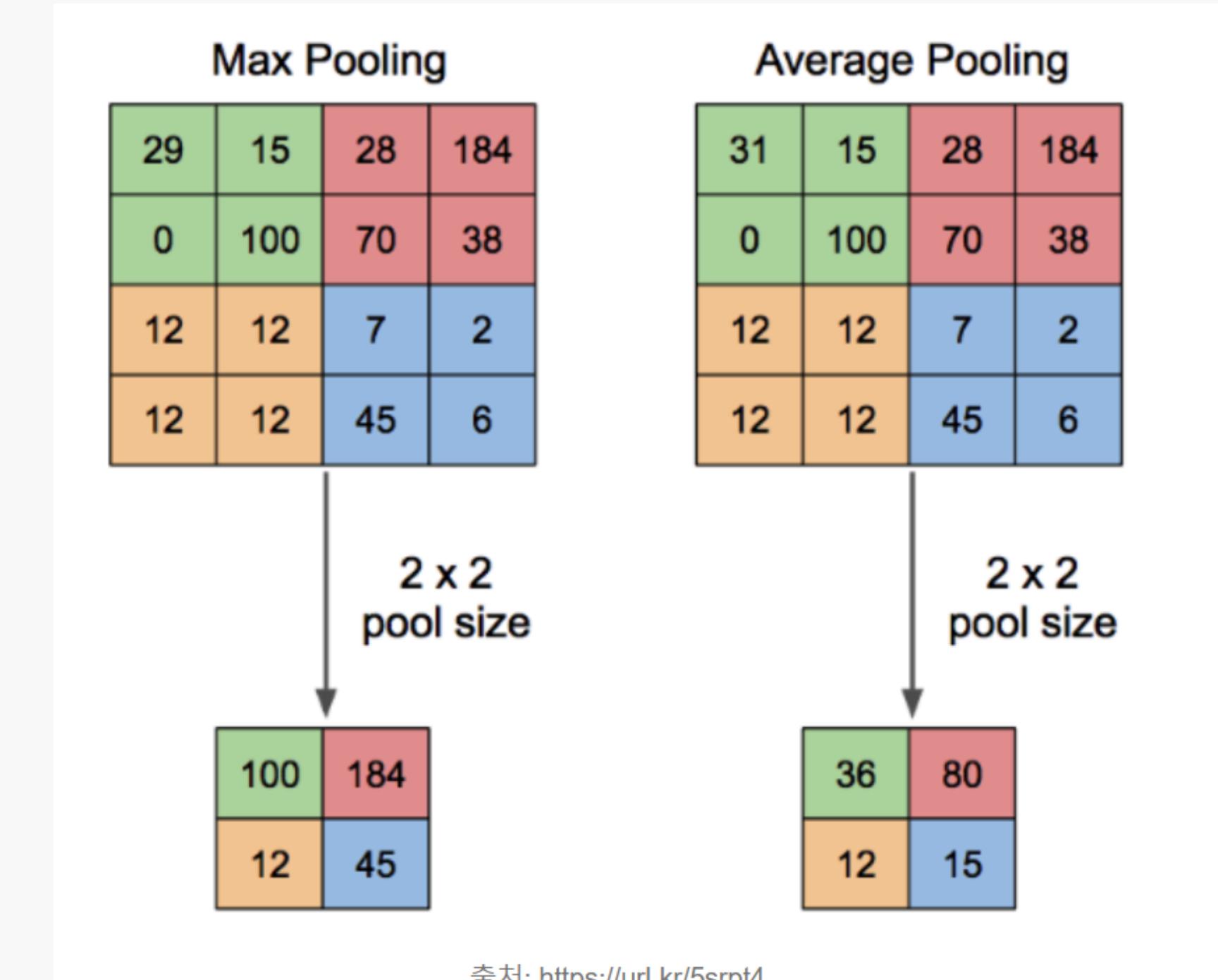
하나의 Input 이미지를 3개의 Channel로 분리시켜서 convolution을 진행
즉, 3개의 Channel은 red channel, blue channel, green channel인 것

Stride & Pooling Layer

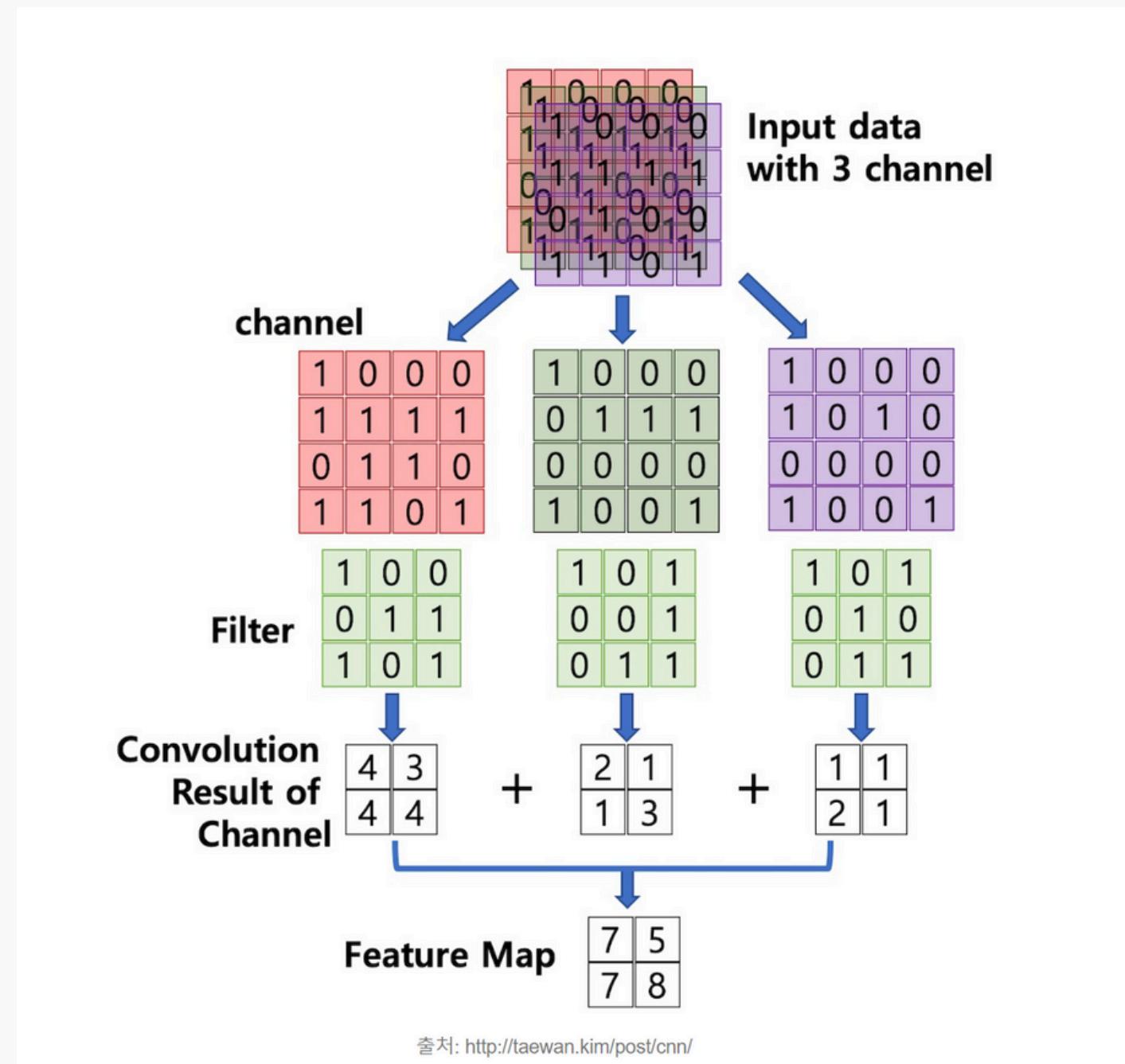
Stride : 필터의 이동량



- Pooling의 목적 : 이미지 크기를 줄임
파라미터 수를 줄일 수 있고 과적합 방지를 위해 쓰임
- Max Pooling
이전의 Matrics에서 가장 큰 값을 대표값으로 가져옴
- Average Pooling
이전의 Matrics에서 평균값을 대표값으로 가져옴



Convolution Layer



Convolution Layer의 최종 결과물 : 특징맵(feature map)

1. Input : Color Image

2. 3개의 Channel로 나눔

3. Filter에 따라 합성곱 연산

4. Output : Feature Map

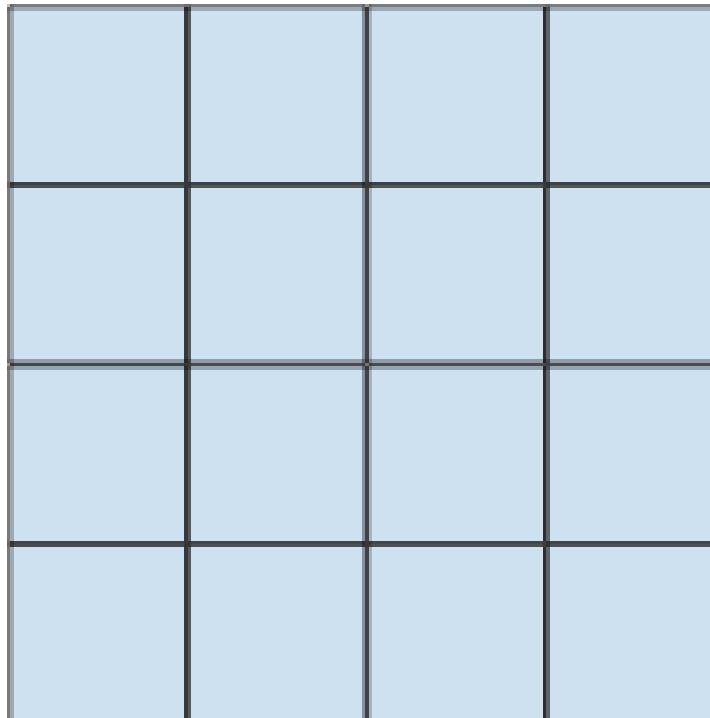
- 일반적으로 CNN에서 레이어가 깊어질수록 채널의 수는 많아지고 너비와 높이는 줄어듦
- Convolution Layer의 서로 다른 필터들은 각각 적절한 특징값을 추출하도록 학습됨
- 특징맵을 생성하는 필터까지도 학습이 가능해 비전문분야에 성능이 우수

Convolution Layer 실습 문제

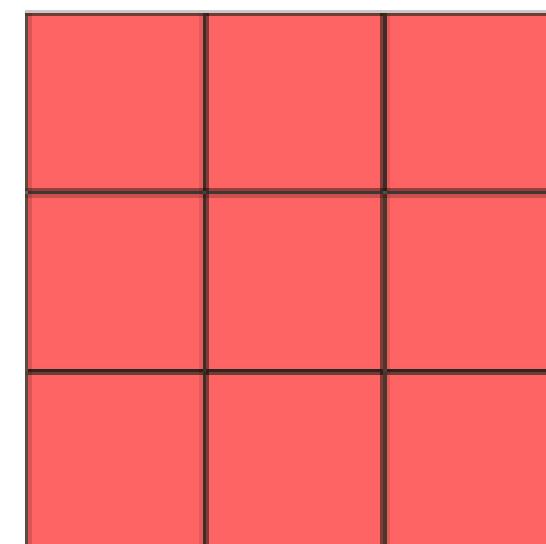
퀴즈 1

2차원인 3x3 컨볼루셔널 필터가 2차원 4x4 입력 특성 맵에 적용됩니다(패딩을 추가하지 않음).

Input Feature Map



Convolutional Filter



Output Feature Map



이미지 분류

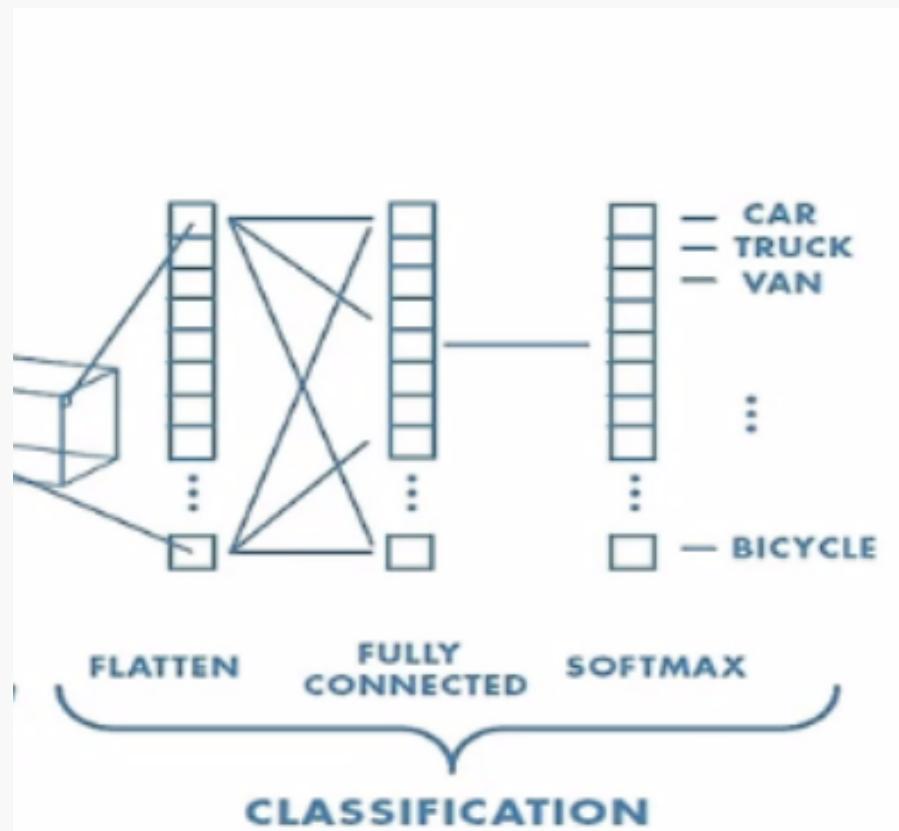
- **Flatten Layer**

Pooling output을 벡터화해주는 단계

추출된 주요 특징은 차원 데이터로 이루어져있지만, 분류를 위한 학습 레이어에서는 1차원 데이터로 바꿔서 학습해야함

- **Fully - Connected Layer (Dense Layer)**

Convolution / Pooling 프로세스의 결과를 취하여 이미지를 정의된 라벨로 분류하는데 사용

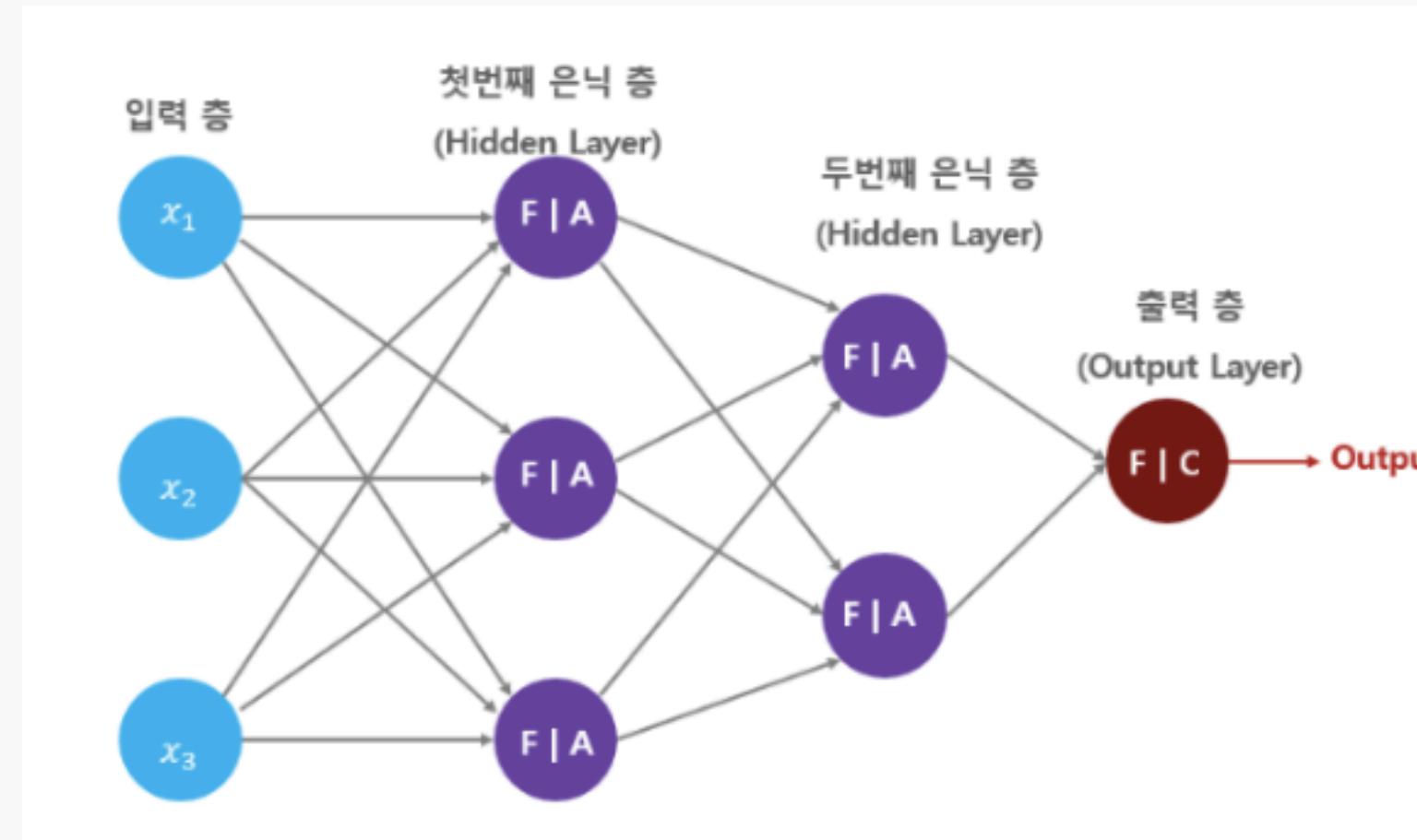


Fully Connected Layer = 완전히 연결되었다라는 뜻

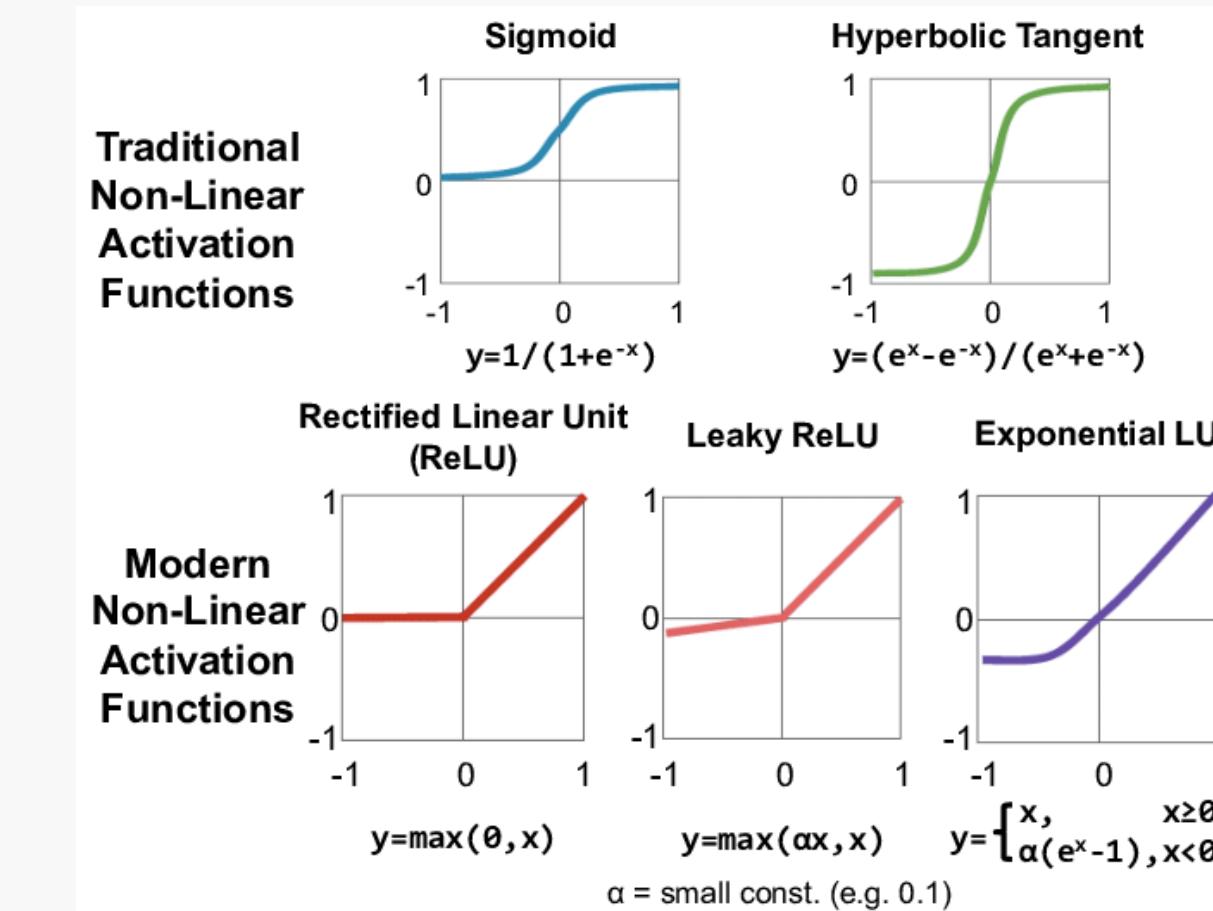
- 한층의 모든 뉴런이 다음층이 모든 뉴런과 연결된 상태

1. 2차원 배열 형태의 이미지를 1차원 배열로 평탄화
2. 활성화함수 (Relu, Leaky Relu, Tanh 등) 뉴런을 활성화
3. 분류기(softmax)함수로 분류

Activation function



활성화 함수 종류



Activation function : 비선형성의 데이터셋은 오버피팅되기 쉽기 때문에 사용해야함

- 대부분 은닉층의 활성화 함수는 ReLU를 적용
- Sigmoid는 은닉층에 거의 사용하지 않고, 이진분류 시 마지막 출력층에 사용
- 퀴즈 2 는 다중 분류시 마지막 출력층에 사용

CNN 구현하기

CNN - MNIST 데이터셋 사용 (PyTorch)

```
# 모델 정의
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.pool = nn.MaxPool2d(2)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = x.view(-1, 9216)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

self.conv1 = nn.Conv2d(1, 32, 3, 1):

- 입력 채널: 1 (입력 이미지는 그레이스케일로, 채널 수가 1입니다.)
- 출력 채널: 32 (32개의 필터를 사용하여 32개의 특성 맵을 생성합니다.)
- 커널 크기: 3x3
- 스트라이드: 1 (커널이 이동하는 간격)

self.pool = nn.MaxPool2d(2):

- 풀링 크기: 2x2

x = F.relu(self.conv1(x)):

- 합성곱: conv1 레이어를 통과하고, ReLU 활성화 함수를 적용

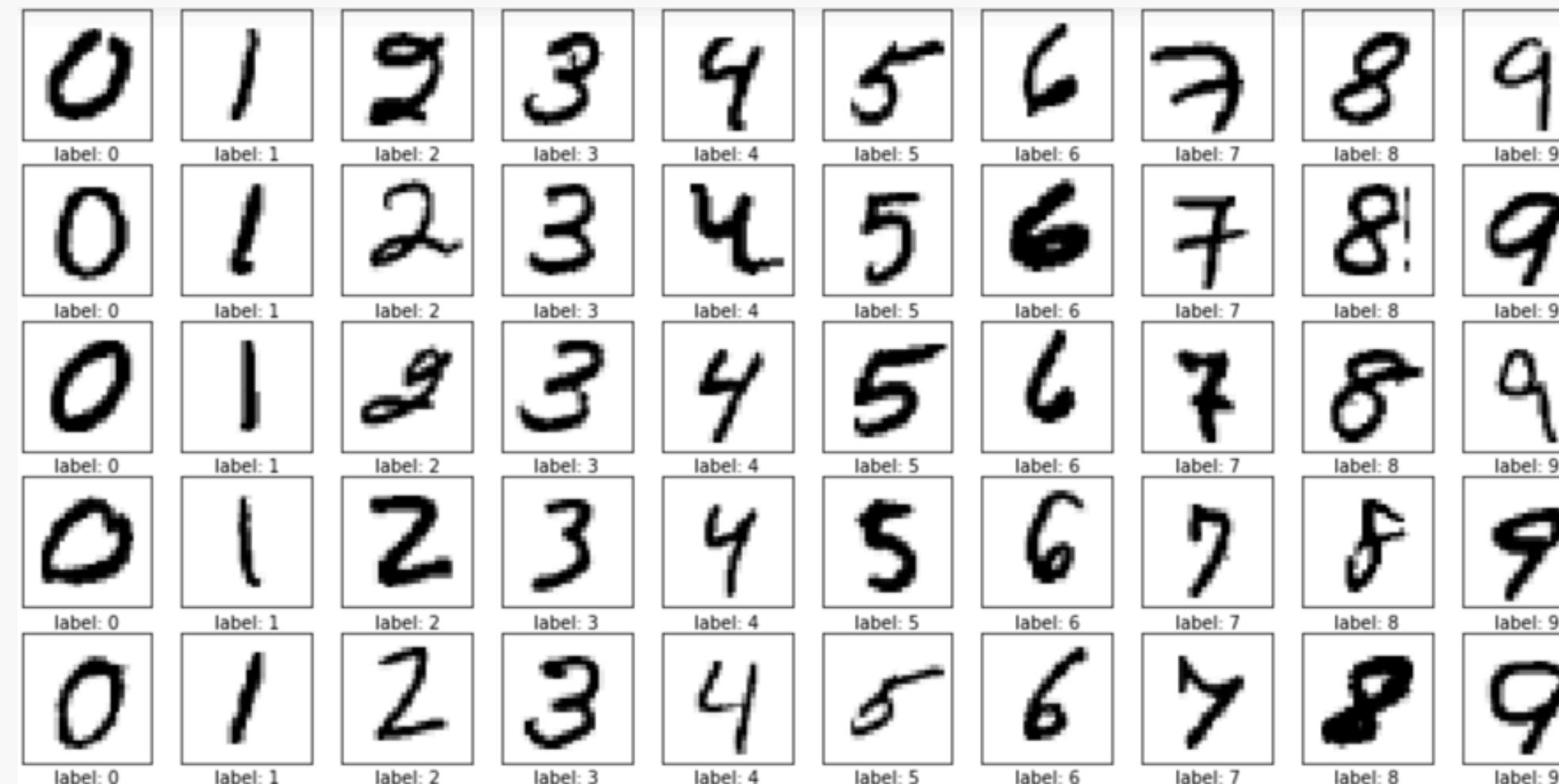
x = x.view(-1, 9216):

- Flatten: CNN의 출력 텐서를 1차원으로 변환

LeNet

LeNet 이란?

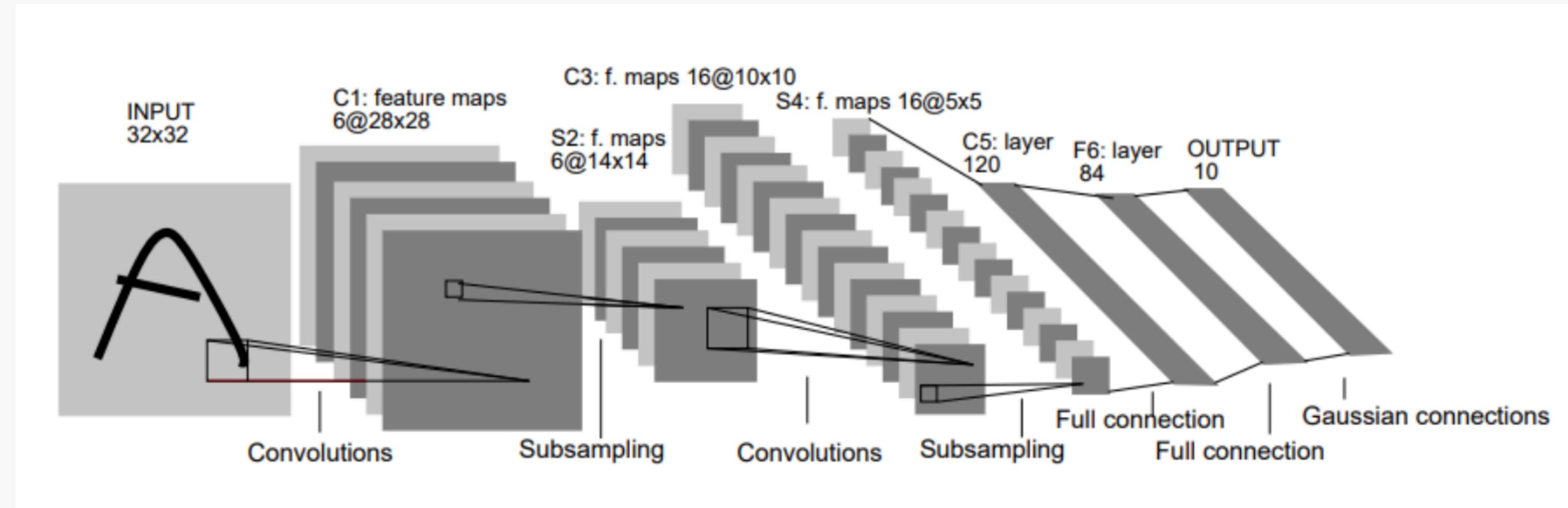
- LeNet은 손글씨 숫자를 인식하는 네트워크로, 1998년 Yann LeCun이 1998년 발표한 논문 ‘Gradient-Based Learning Applied to Document Recognition’에서 소개되었다.
- 손글씨 인식을 할때 기존에 활용되던 Multilayer Neural Network의 한계를 극복하기 위한 CNN 모델



LeNet 특징

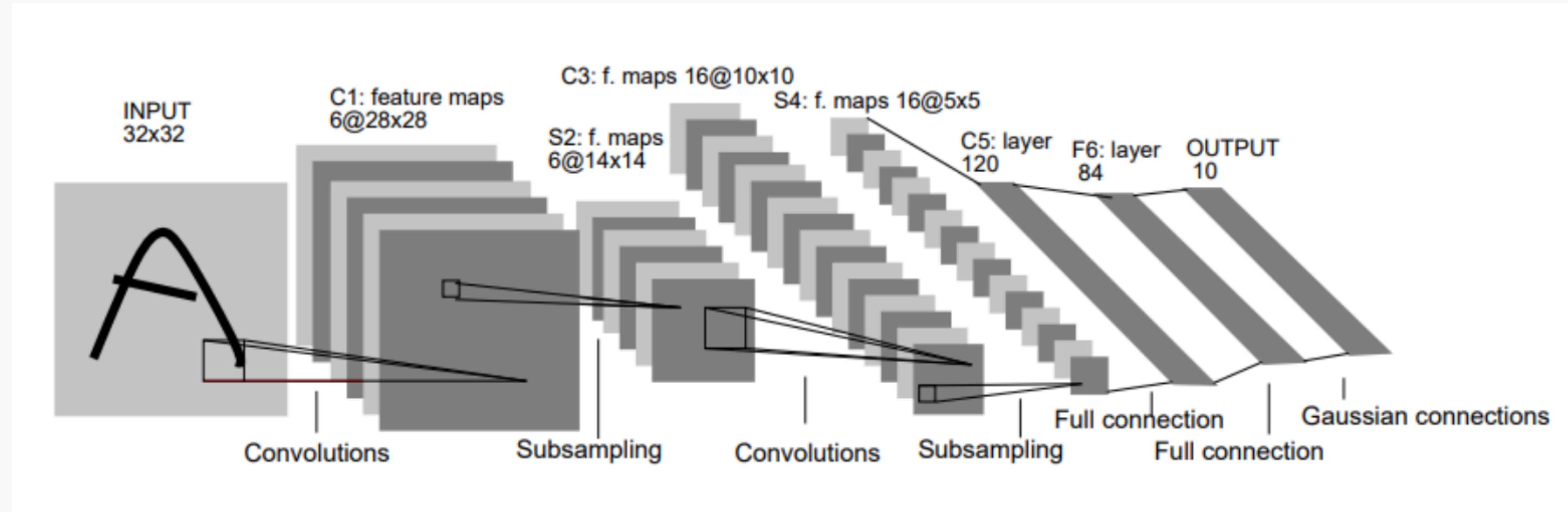
- LeNet 발표 전 활용되던 Fully-Connected Network는 2차원 이미지를 1차원으로 펼쳐진 데이터로 입력받음
- 이 때, 이미지가 갖는 공간적인 정보가 손실되는 문제가 있었는데, Yann LeCun은 Local receptive field, Shared weight, Sub-sampling의 개념을 결합해 이 문제를 해결함
- Local receptive field (국소 수용 영역): CNN에서 사용되는 개념으로, 신경만의 한 뉴런이 입력 데이터의 특정 부분을 관찰하는 영역을 말하며, 공간적으로 국소화된 특징을 캡쳐하는 데 도움을 준다.
- Shared weight (공유 가중치): CNN에서 동일한 가중치를 입력의 다른 영역에 걸쳐 사용하는 개념으로, 동일한 특징을 위치에 상관없이 감지할 수 있게 한다.
- Sub-sampling (서브 샘플링): 입력의 공간적 차원을 줄이는 과정으로, 계산량을 줄이고, 변환 불변성을 얻는 데 도움을 준다.

LeNet 구조



- 합성곱 계층과 원소를 줄이는 ‘서브샘플링’ 계층을 반복하고, 마지막으로 완전연결 계층을 거치면서 결과를 출력하는 형식
- sub sampling 방식은 최대 풀링(max pooling), 평균 풀링(average pooling), L2-norm pooling 등이 있는데, 현재는 주로 max pooling 방식을 사용하며 활성화 함수로 Sigmoid 를 활용한다는 특징이 있음.

LeNet 구조

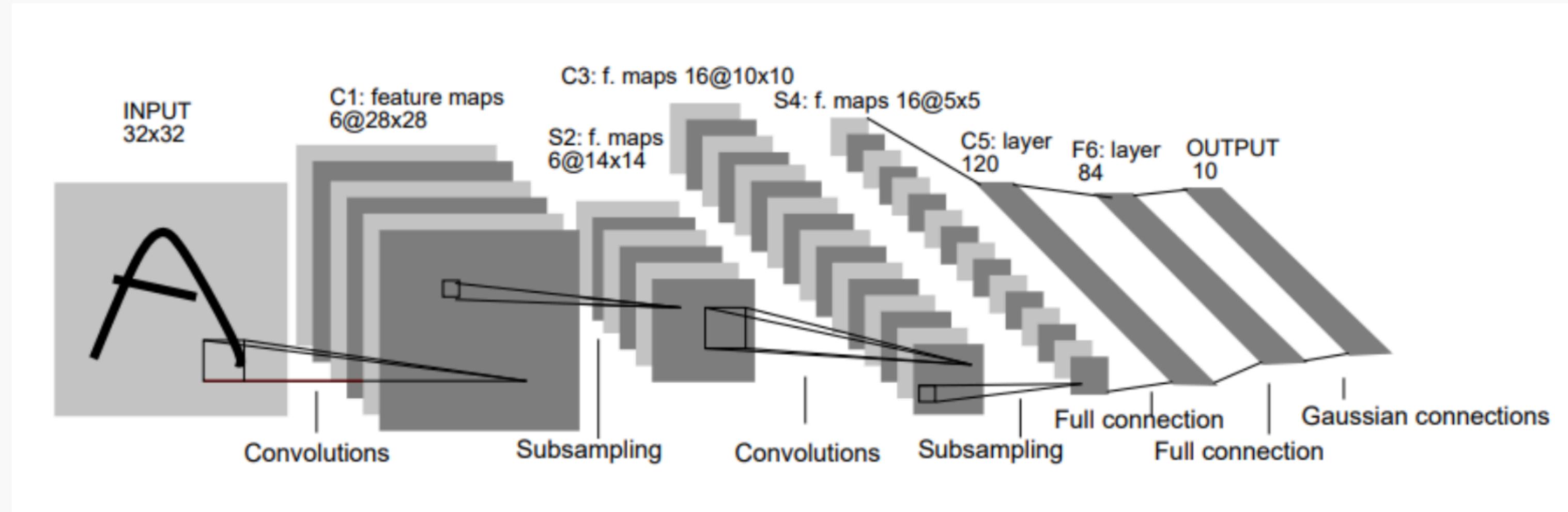


- 32 × 32 크기의 이미지를 학습하는 7층 합성곱 네트워크
- Cons(C1) -> Subsampling(S2) -> Cons(C3) -> **퀴즈 3번**-> Cons(C5) -> FC(F6) -> FC(F7)

Cx: convolution layer (합성곱 계층) / **Sx**: subsampling layer (풀링 계층)

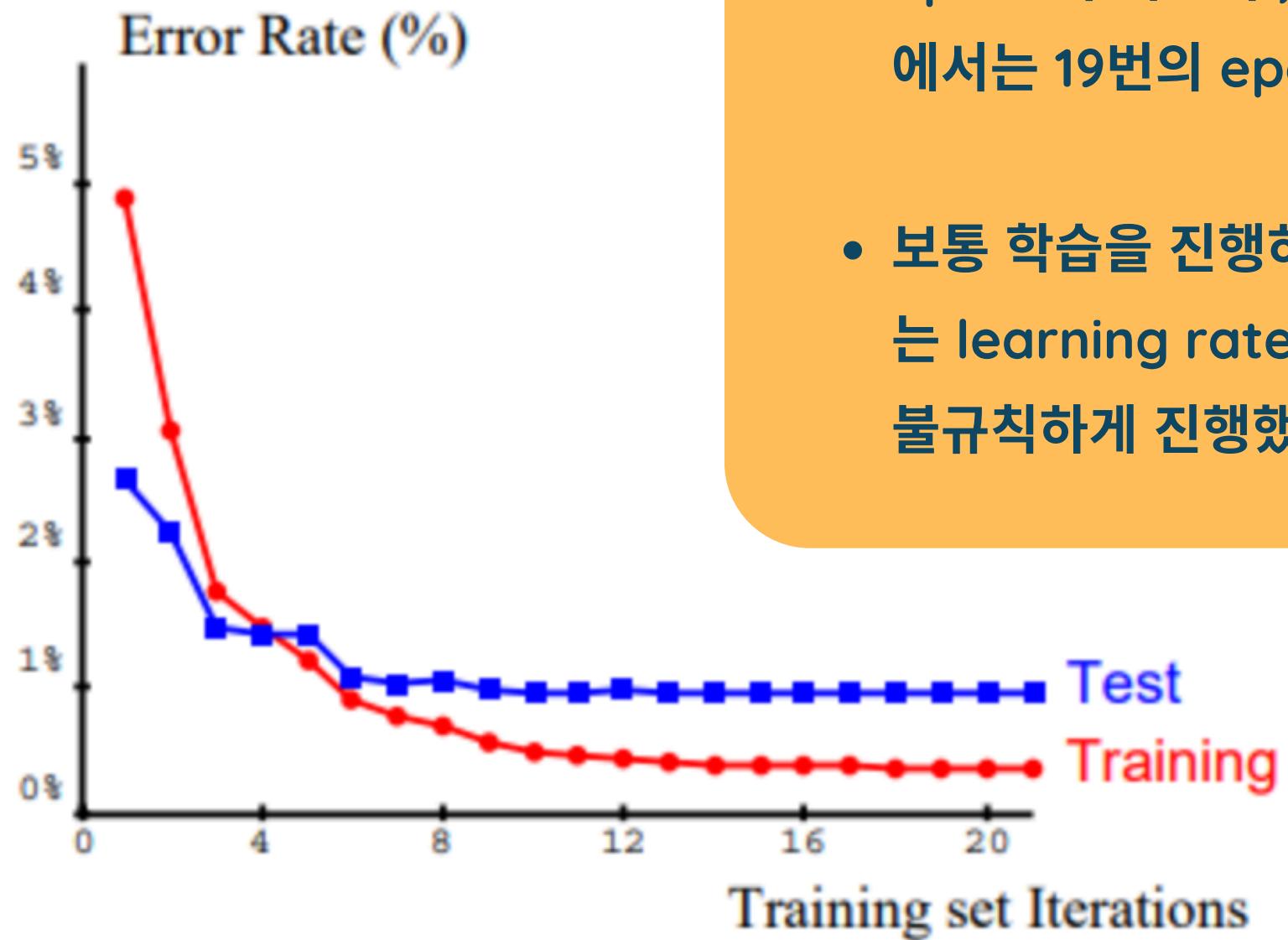
Fx: fully-connected layer (완전연결 계층) / **X**: index of the layer (각 층의 index)

LeNet 구조



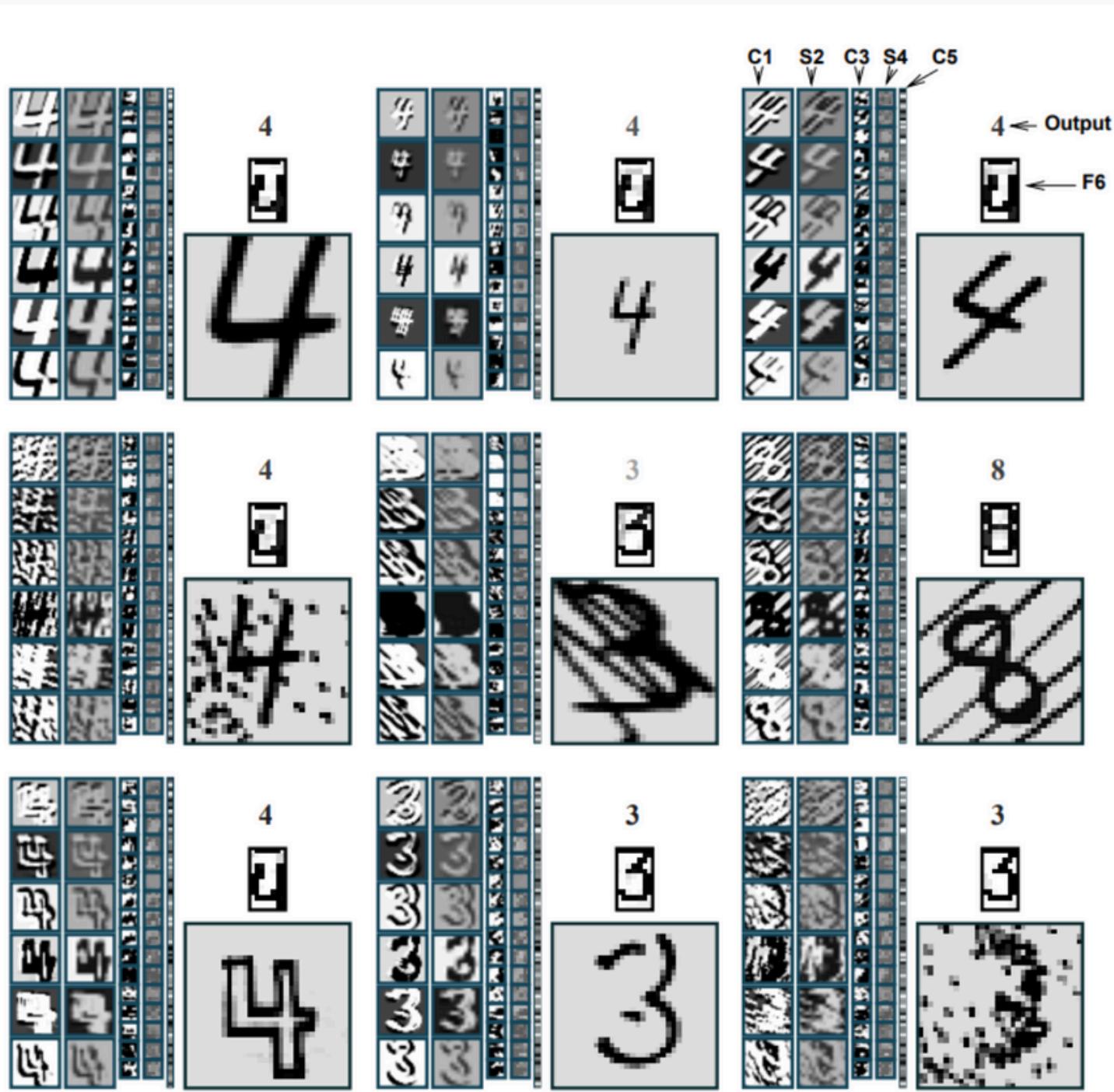
- 실제 문자 이미지는 28X28 크기인데, 입력 이미지가 32X32인 이유는 corner나 edge 같은 특징이 receptive field의 중앙 부분에 나타나도록 하기 위함
- 구조적인 변화로는 Convolution layer를 통과 후 생성되는 feature map의 개수가 4개, 12개에서 6개, 16개로 증가하였고, Fully-connected layer의 개수가 3개로 증가하였다.

LeNet 학습 결과



- MNIST 데이터셋에 LeNet 모델을 학습시켜서 Error rate를 확인한 결과 약 10번의 epoch이 지난 후, test 데이터셋에서 에러율이 0.95%에서 수렴하였고, train 데이터셋에서는 19번의 epoch이 지난 후, 0.35%에 수렴하였다.
- 보통 학습을 진행하면, training set에 대한 overfitting이 발생하는데, 해당 논문에서는 learning rate를 크게 유지하였기 때문에, Loss가 Local minimum에 빠지지 않고 불규칙하게 진행했기 때문에 overfitting 현상이 발생하지 않은 것으로 추정

LeNet 시각화

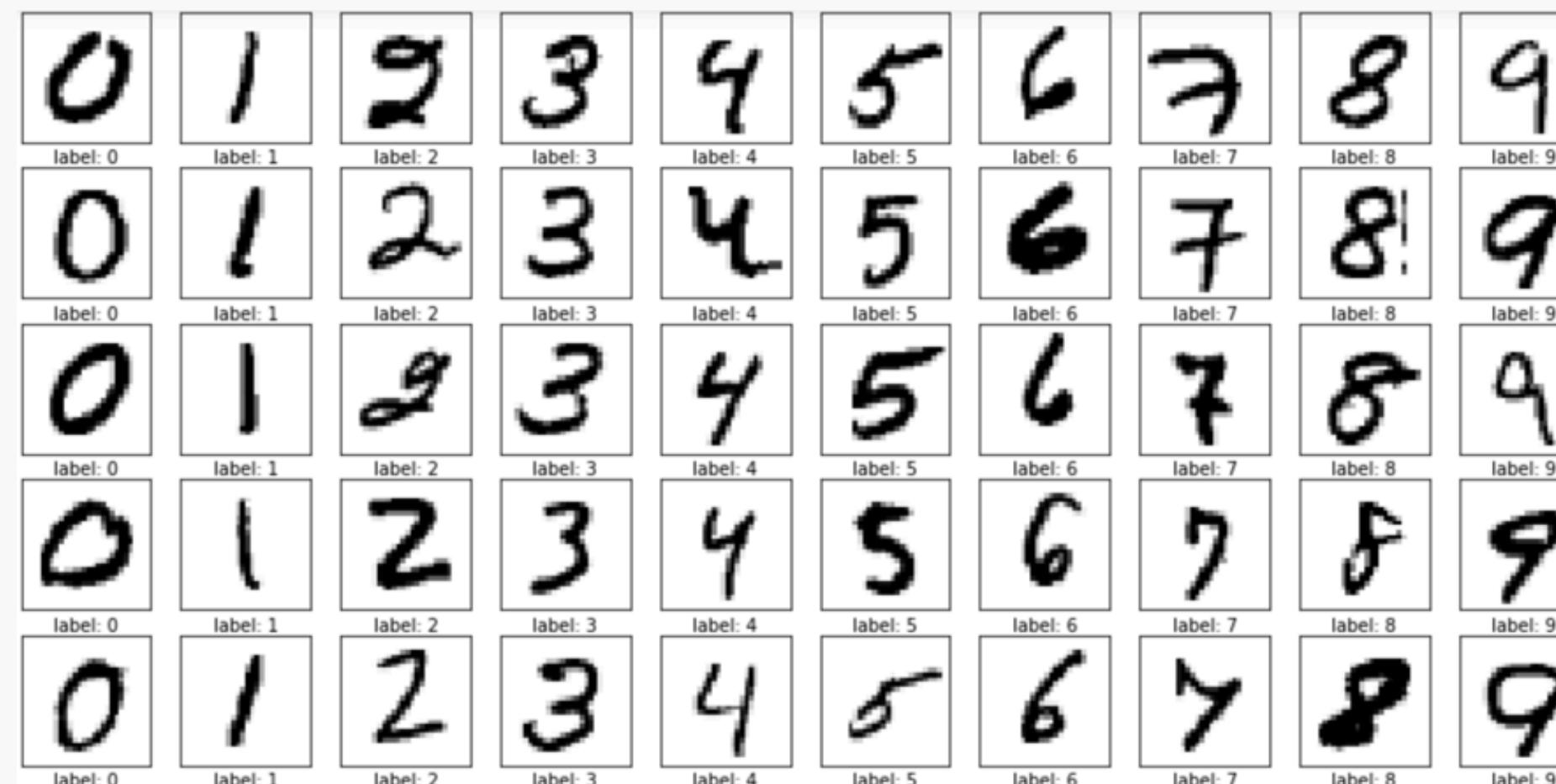


- 위 그림은 잡음과 왜곡이 있는 이미지에서 각각의 Layer가 어떤 결과를 도출했는지와 output 결과를 나타내는 그림
- 결과에서 볼 수 있듯이 잡음이 있거나 왜곡이 있는 이미지에서도 안정적인 결과를 보여준다.

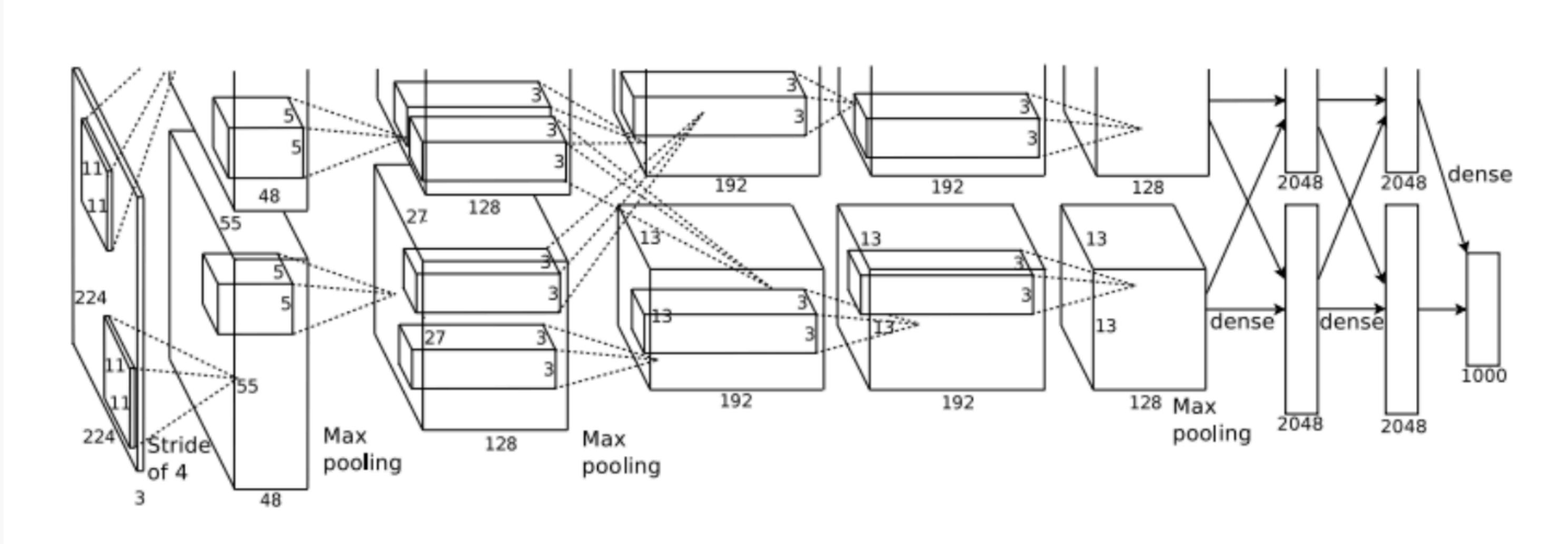
AlexNet

AlexNet 이란?

- 캐나다 토론토 대학에서 발표한 AlexNet은 2012년 개최된 ILSVRC (ImageNet Large Scale Visual Recognition Challenge)에서 압도적인 성능으로 우승한 CNN Network
- GPU를 사용한 병렬연산이 특징이며, LeNet과 구조적으로 큰 차이는 없지만, 성능 개선을 위한 세부적인 특징이 차이가 있음.



AlexNet 구조

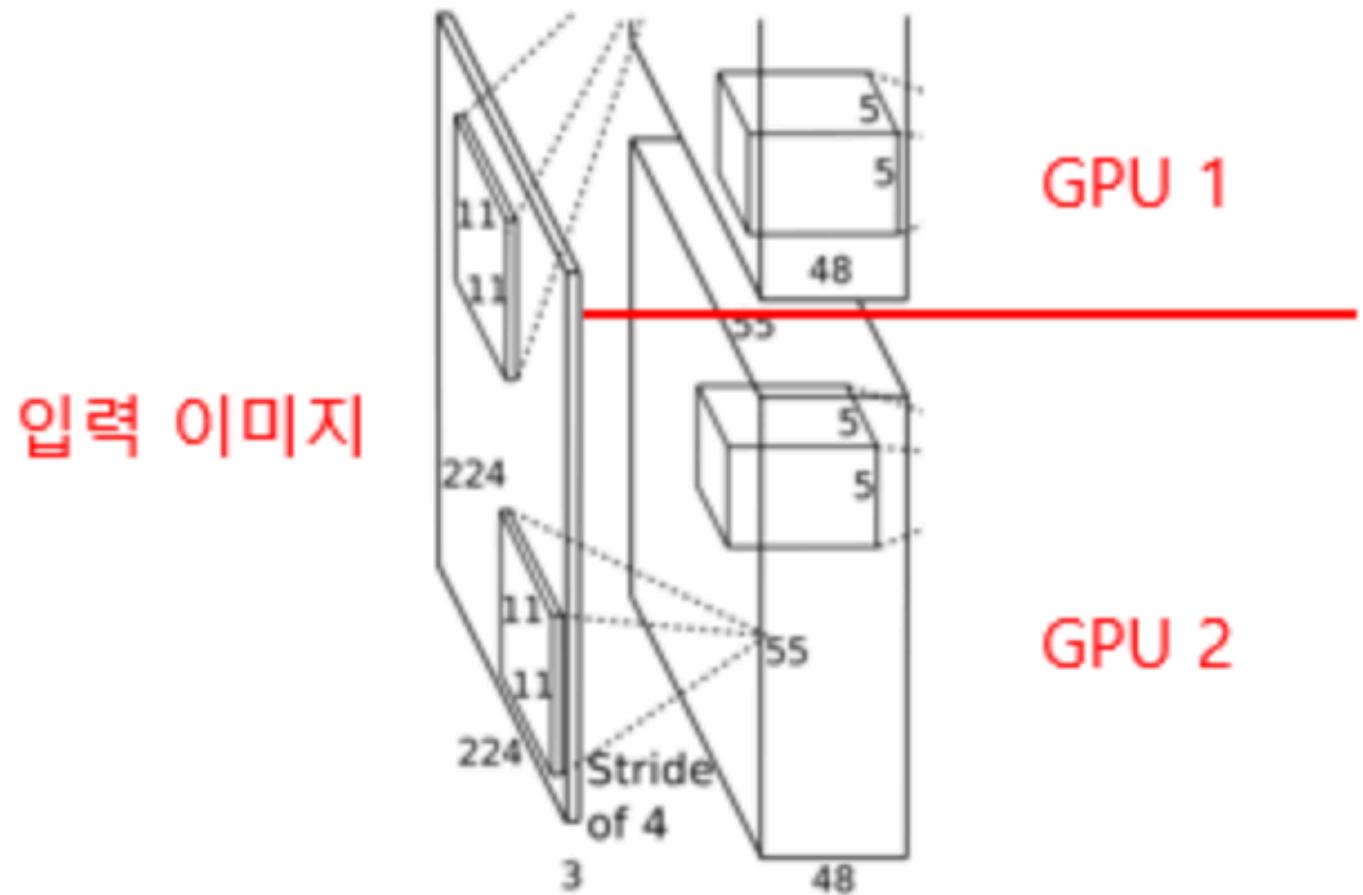


- 5개의 Convolution Layer(합성곱 계층)와 3개의 FC Layer(완전연결층)로 구성된 **8층 합성곱 네트워크**
- 마지막 output 결과는 softmax 연산을 거쳐 1000개의 class label에 대한 확률값으로 출력

AlexNet 특징

1) Multi GPU 사용

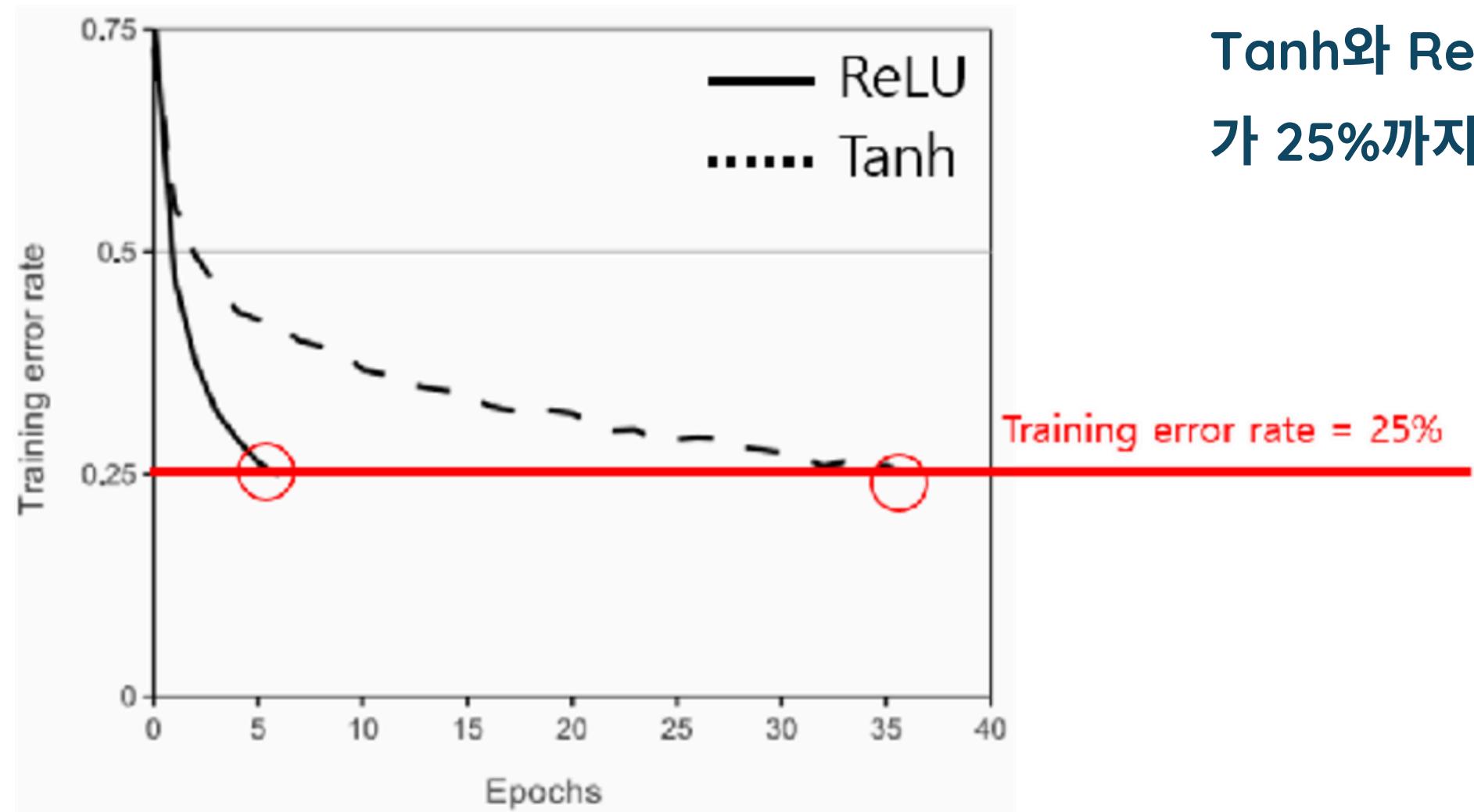
- AlexNet을 개발할 당시, 개발팀은 3GB 메모리의 한 개의 GPU만 가지고 있었기 때문에 1200만개의 샘플 데이터를 연산하기에 한계점을 가짐
- Network를 두 개의 GPU로 나눌 수 있는 구조로 설계
- 3번 Convolution layer와 Fully-Connected Layer에서 GPU끼리 파라미터를 공유하도록 해, 1,2번 GPU의 결과를 섞어 쓸 수 있도록 설계



AlexNet 특징

2) 활성화함수 ReLU 사용

- ReLU를 사용함으로써 학습 속도가 매우 빨라짐
- CIFAR-10 데이터셋과 4개의 Convolutional network를 사용해 Tanh와 ReLU의 학습 속도를 비교한 결과, training error rate 가 25%까지 내려가는 데, ReLU가 Tanh보다 약 7배 빠름



AlexNet 특징

3) LRN (Local Response Normalization)

- LRN은 kernel의 한 점에서 공간적으로 같은 위치에 있는 다른 channel의 값을 square-sum하여 하나의 filter에서 과도한 활성화 값이 나오는 것을 막는 역할

$$(k + \alpha \sum_{j=\max(0, i-n/2)}^{\min(N-1, i+n/2)} (\alpha_{x,y}^j)^\beta)$$

$a_{x,y}^i$: kernel의 i번째 channel에 있는 (x, y)점에서 값
 $b_{x,y}^i$: LRN을 적용한 결과 값
 k, n, α, β : normalization을 위한 hyper-parameter

Overfitting 방지

- AlexNet은 6천만개의 파라미터를 사용했으며, LeNet보다 Overfitting을 방지하기 위해 더 많은 신경을 씀

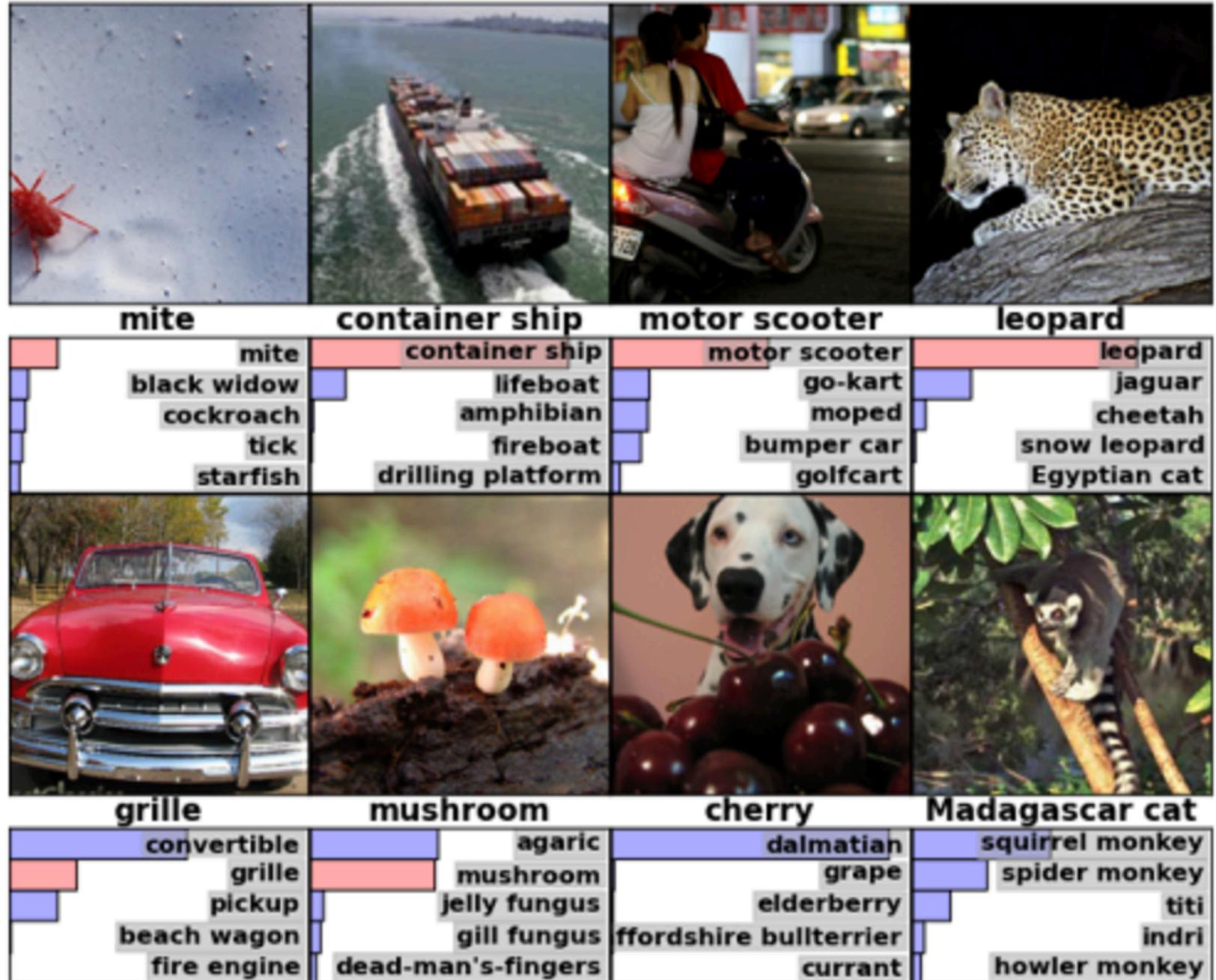
1) Data Augmentation

- AlexNet에서는 데이터를 늘리기 위해 두 가지 방법 적용.
- 첫 번째 방법은 ILSVRC의 256X256 크기의 원래 이미지로부터 224X224 크기의 이미지를 무작위로 추출.
1장의 이미지로부터 수평반전을 포함해 총 2,048장의 이미지 추출

2) 퀴즈 4번

- **퀴즈 4번** 은 학습할 때 일부 뉴런을 생략하는 방법
- 첫 번째와 두 번째 FC Layer에 대해 50% 확률로 hidden 뉴런을 생략하며 학습을 진행

AlexNet 결과



- 8개의 이미지로 확인했을 때, 이미지가 중앙에 위치해 있지 않더라도 정확하게 Class를 구분함
- 4번째 사진 leopard의 경우에도, 재규어, 치타 등 상위 5개의 label들이 거의 비슷한 형태를 보이는 동물들
- cherry 사진이나 grille 사진의 경우, 초점을 어디에 맞추느냐에 따라 충분히 나올 수 있는 답이기에 꽤 정확한 성능을 보인다고 판단

참고 레퍼런스

- <https://eunhye-zz.tistory.com/10>
- <https://bommbom.tistory.com/entry/CNN-모델-종류-LeNet-AlexNet-VGG-GoogleNet-비교>
- 밑바닥부터 시작하는 딥러닝
- <https://woochan-autobiography.tistory.com/860>
- <https://rubber-tree.tistory.com/116>
- <https://m.blog.naver.com/luvwithcat/222148953574>
- <https://all-young.tistory.com/43>
- <https://woochan-autobiography.tistory.com/860?pidx=2>



Thank you

