7장 CNN

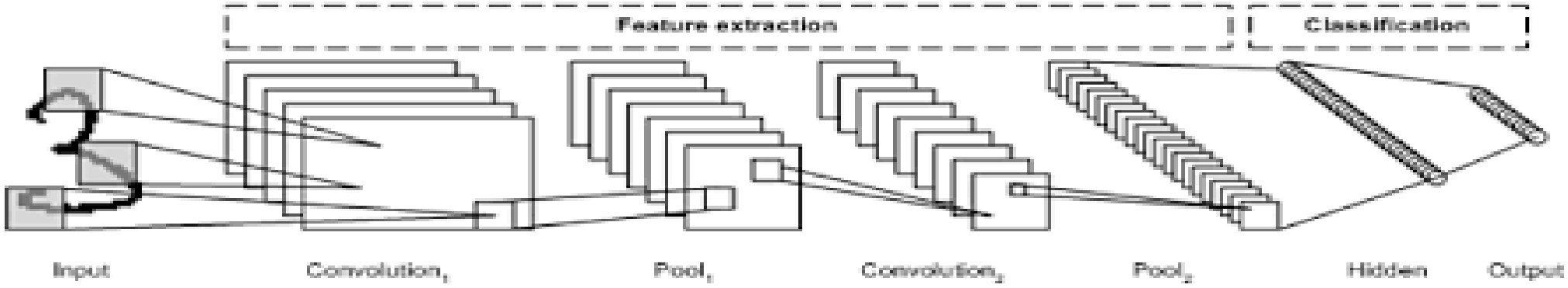
전북대학교
IT정보공학과
박나현

완전연결계층(Fully-connected) 문제점

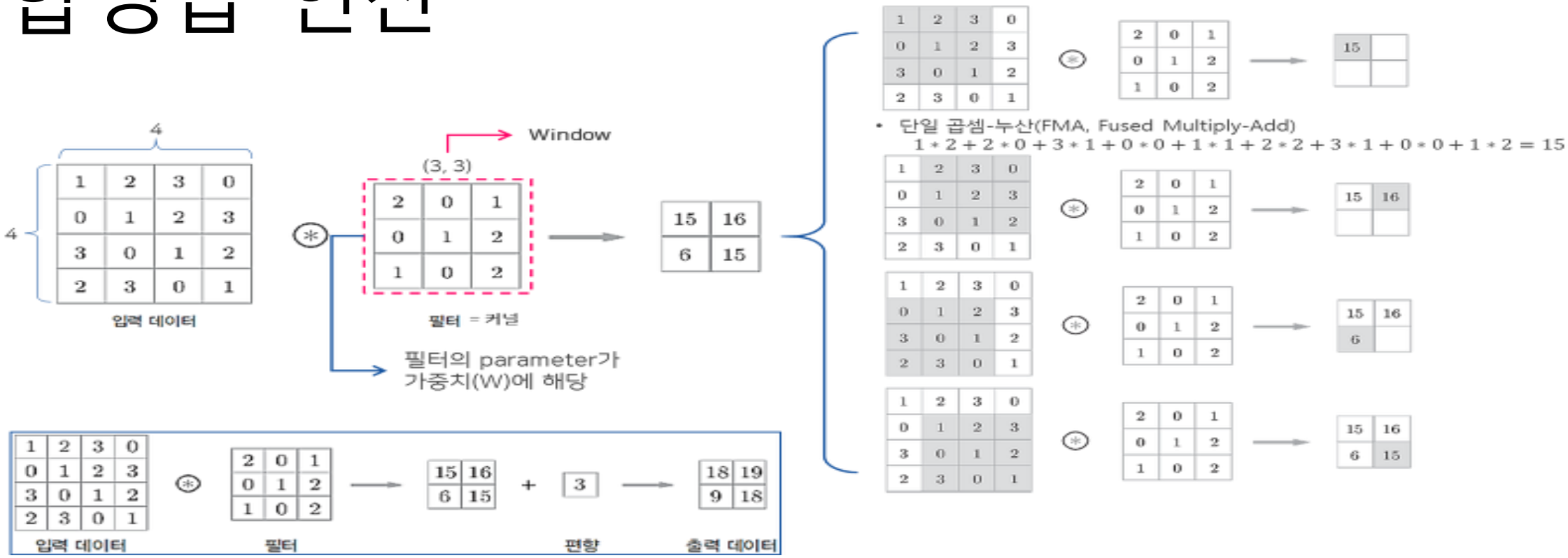
- FC에서 인접하는 계층의 뉴런이 모두 연결되고 출력의 수는 임의로 정할 수 있음
- **데이터 형상이 무시**되는 문제점 발생
- 3차원 데이터를 1차원 데이터로 평탄화 시켜주며 **공간적 구조 정보 유실**로 비효율적
  - 공간적 구조 정보
    - 거리가 가까운 어떤 픽셀들끼리 어떤 연관이 있고 어떤 픽셀들끼리 값이 비슷하거나 등을 포함

CNN(Convolutional Neural Network)

- **Convolution**
  - 일정 영역의 값들에 대해 가중치를 적용하여 하나의 값을 만드는 연산
- **다차원 배열 데이터**를 처리하도록 구성하여 형상 유지
  - 이미지 인식, 음성 인식에서 주로 사용
- 전반부: 연산을 수행하여 특징 추출 (Convolution, Pooling)
- 후반부: 특징을 이용하여 분류 (Multi-layer Perceptron)



합성곱 연산



- 커널(Kernel) : 필터(Filter), Conv Layer의 가중치에 해당
- 합성곱 연산
  - 입력데이터와 필터 간의 서로 대응하는 원소끼리 곱한 후 총합을 구함 (FMA)

패딩(Padding)

- 합성곱 연산을 수행하기 전, 입력데이터 주변을 특정 값으로 채워 늘리는 것
- 주로 출력데이터의 **공간적 크기를 조절**하기 위해 사용
  - Feature Map 크기가 입력의 크기와 동일하게 가능
- **가장 자리 정보 보존**

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

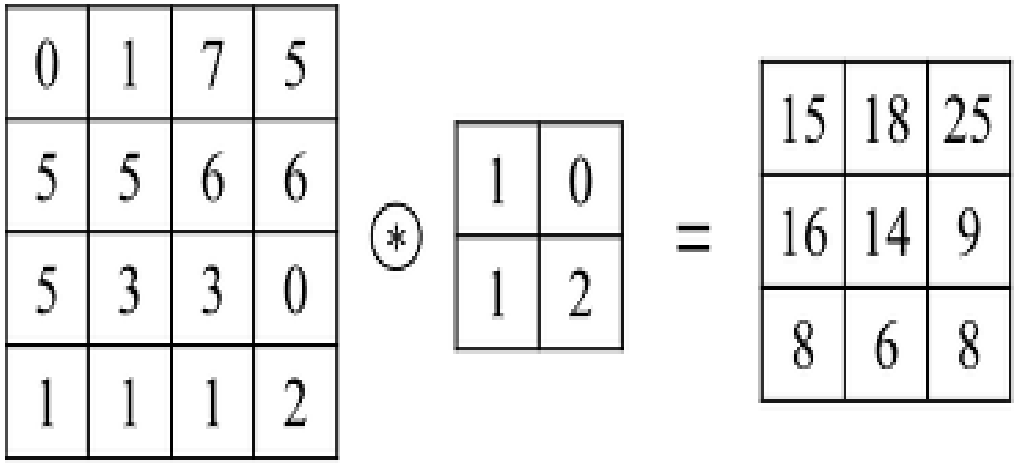
패딩 전

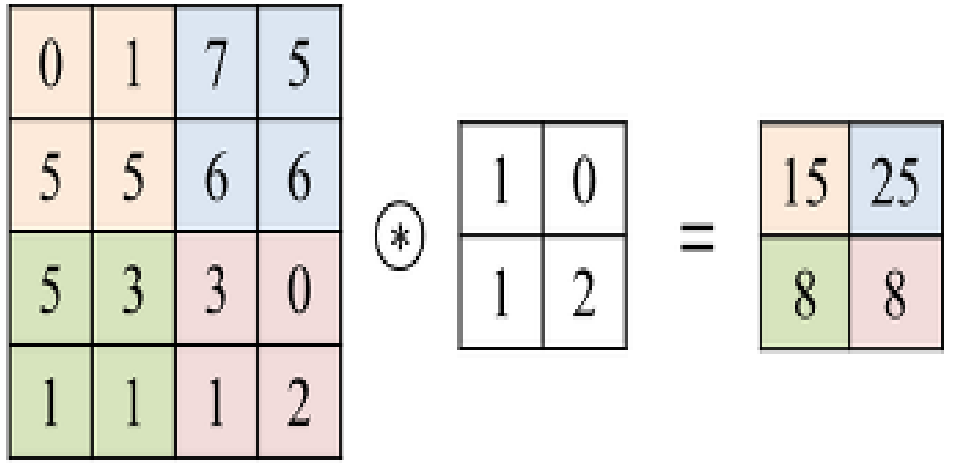
0	0	0	0	0	0	0
0	1	2	3	4	5	0
0	2	1	0	1	2	0
0	3	0	1	1	0	0
0	1	4	1	1	2	0
0	2	1	1	0	0	0
0	0	0	0	0	0	0

패딩 후

스트라이드(Stride)

- 입력데이터에 필터를 적용할 때 이동할 간격을 조절하는 것
- 출력의 데이터 크기를 조절하기 위해 사용
- 스트라이드 증가하면 출력 크기 감소





Stride 적용한 출력크기 계산

입력 크기 - (H, W)

필터 크기 - (FH, FW)

출력 크기 - (OH, OW)

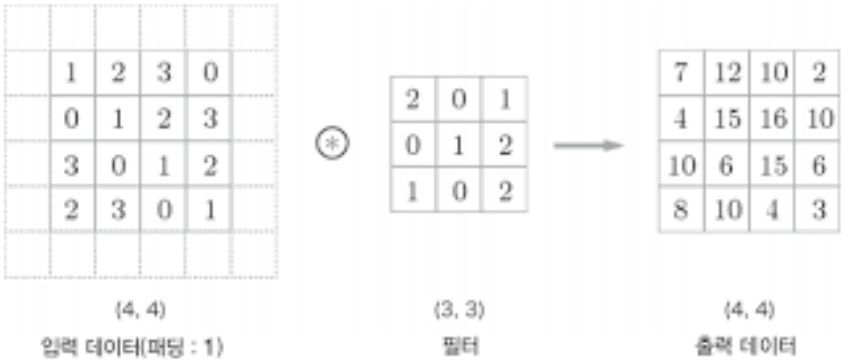
패딩 - P

스트라이드 - S

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

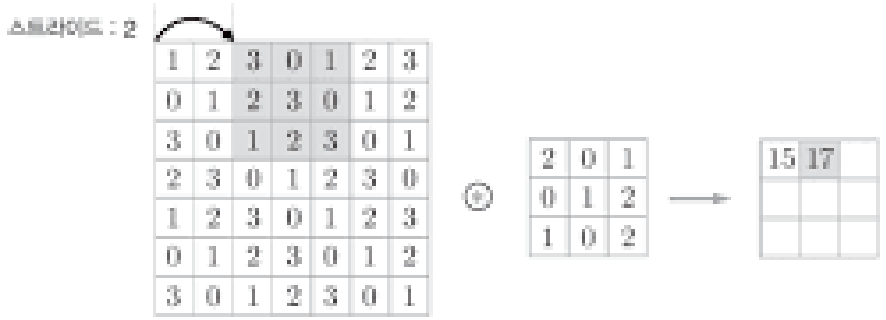
- 입력: (4,4), 패딩 : 1, 스트라이드 : 1, 필터 : (3,3)



$$OH = \frac{4+2\cdot1-3}{1} + 1 = 4$$

$$OW = \frac{4+2\cdot1-3}{1} + 1 = 4$$

- 입력: (7,7), 패딩 : 0, 스트라이드 : 2, 필터 : (3,3)

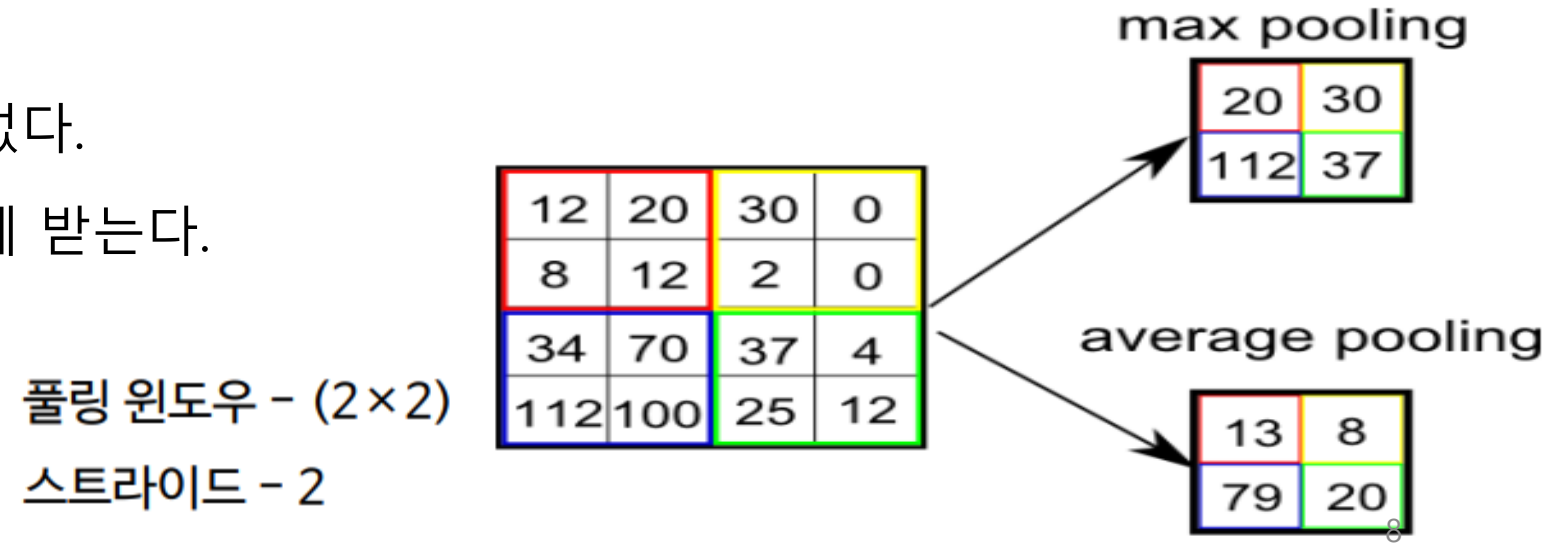


$$OH = \frac{7+2\cdot0-3}{2} + 1 = 3$$

$$OW = \frac{7+2\cdot0-3}{2} + 1 = 3$$

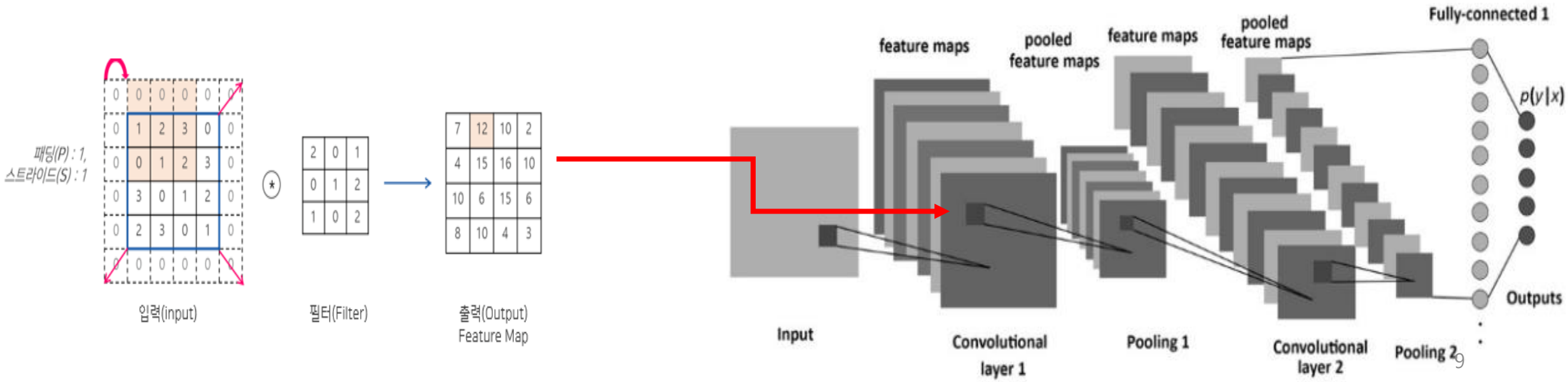
풀링(Pooling) 계층

- 영역을 원소 하나로 집약하여 공간 크기 조절
  - 최대 풀링 (Max Pooling) : 대상 영역에서 최대값을 구하는 연산
  - 평균 풀링 (Average Pooling) : 대상 영역에서 평균을 계산
- 풀링의 Window 크기와 Stride는 같은 값으로 설정하는 것이 보통
- 특징
  - 학습해야 할 매개변수가 없다.
  - 입력의 변화에 영향을 적게 받는다.
  - 채널 수가 변하지 않는다.



3차원 데이터의 합성곱 연산

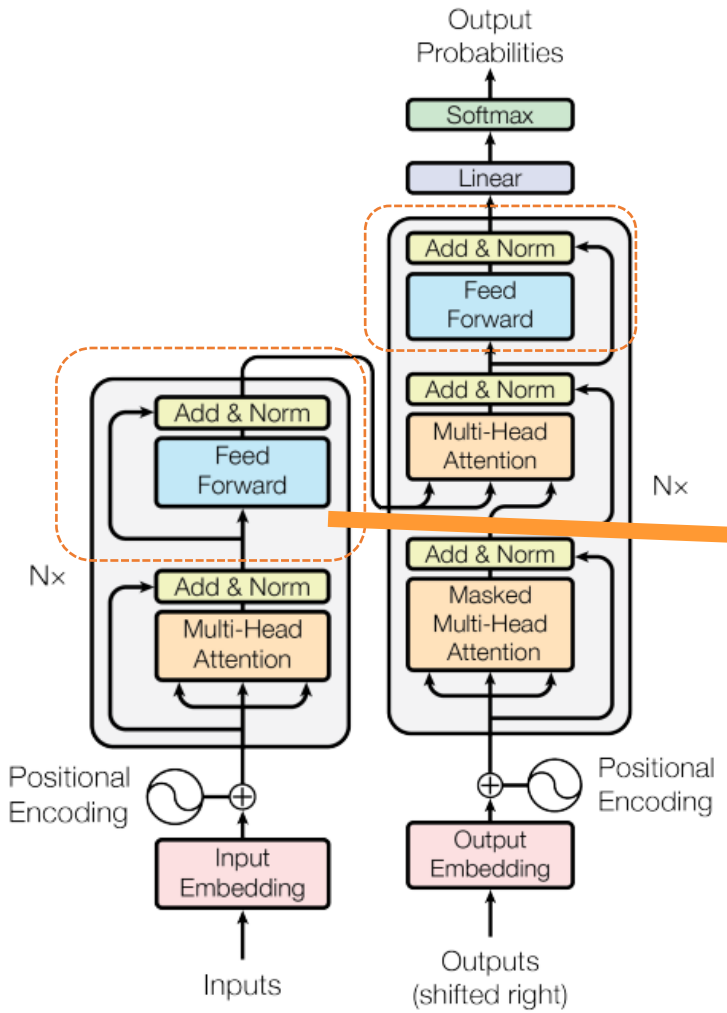
- 입력 데이터와 필터의 합성곱 연산을 채널마다 수행하고, 그 결과를 더해서 하나의 출력을 얻음
- 입력데이터의 채널 수와 필터의 채널 수가 같아야 함

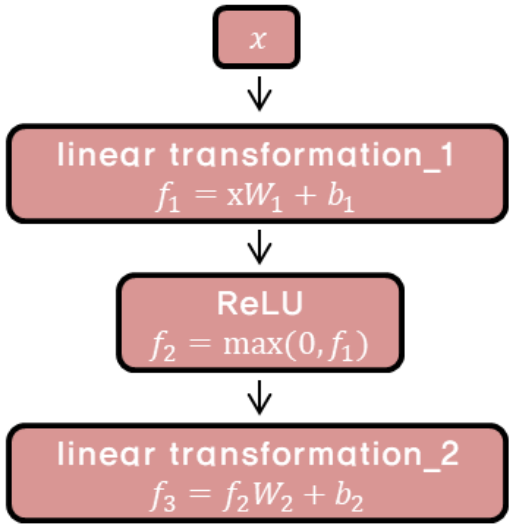


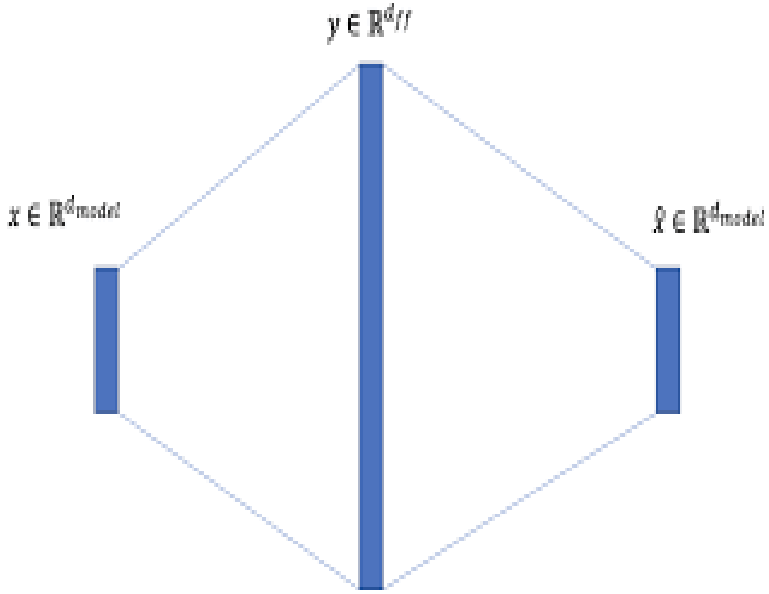
MNIST Classification

```
class CNN(torch.nn.Module):#nn.Module 모든 신경망 모듈의 기본이 되는 클래스
                        #각 층과 함수 등 신경망의 구성요소를 이 클래스 안에서 정의
    def __init__(self):
        super(CNN, self).__init__()
        self.keep_prob = 0.5 #drop out(drop out시키지 않고 유지하는)비율
        # L1 ImgIn shape=(?, 28, 28, 1)
        #    Conv     -> (?, 28, 28, 32)
        #    Pool     -> (?, 14, 14, 32)
        self.layer1 = torch.nn.Sequential( #nn.Sequential 순서대로 값을 전달하고 수행
            #Conv2d(input_channel_size, output_volume_size, kernel_size, stride, padding)
            torch.nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2))
        # L2 ImgIn shape=(?, 14, 14, 32)
        #    Conv      ->(?, 14, 14, 64)
        #    Pool      ->(?, 7, 7, 64)
        self.layer2 = torch.nn.Sequential(
            torch.nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2))
        # L3 ImgIn shape=(?, 7, 7, 64)
        #    Conv      ->(?, 7, 7, 128)
        #    Pool      ->(?, 4, 4, 128)
        self.layer3 = torch.nn.Sequential(
            torch.nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2, padding=1))
        # L4 FC 4x4x128 inputs -> 625 outputs
        self.fc1 = torch.nn.Linear(4 * 4 * 128, 625, bias=True)
        #bias를 False로 설정하면 layer에서 추가 bias를 학습하지 않는다
        #True는 기본 값으로 bias 학습
        torch.nn.init.xavier_uniform_(self.fc1.weight) #layer 특성에 맞춰 초기화하는 방법
        self.layer4 = torch.nn.Sequential(
            self.fc1,
            torch.nn.ReLU(),
            torch.nn.Dropout(p=1 - self.keep_prob))
        # L5 Final FC 625 inputs -> 10 outputs
        self.fc2 = torch.nn.Linear(625, 10, bias=True)
        torch.nn.init.xavier_uniform_(self.fc2.weight)
    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = self.layer3(out)
        out = out.view(out.size(0), -1)   # Flatten them for FC
        out = self.layer4(out)
        out = self.fc2(out)
        return out
```

Position-wise Feed-Forward Networks가 CNN으로 대체 가능한 이유







Position-wise Feed-Forward Networks가 CNN으로 대체 가능한 이유