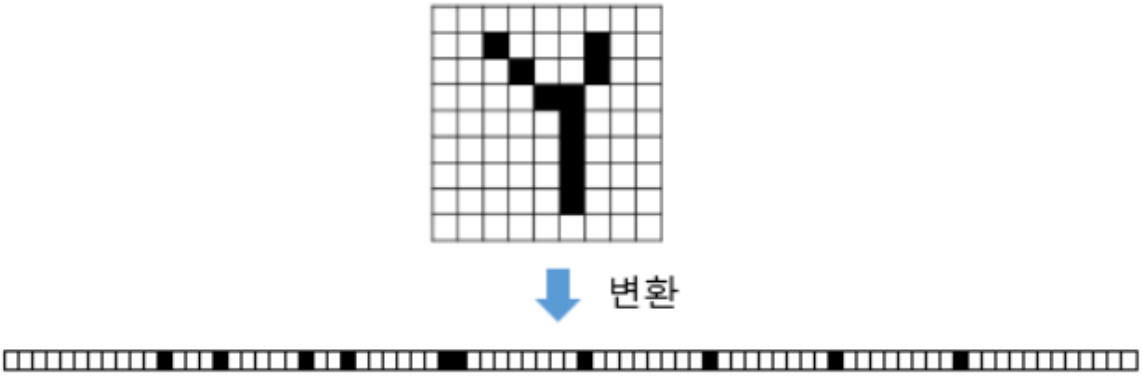
합성곱 신경망(Convolution Neural Network)

합성곱 신경망의 대두

- 합성곱 신경망 : **공간적인 구조 정보**를 보존하면서 학습할 수 있는 방법
  공간적인 구조 정보 : 거리가 가까운 픽셀들끼리 어떤 연관이 있는지를 파악
- 다층 퍼셉트론으로 이미지를 분류한다면, 이미지를 1차원 텐서인 벡터로 변환 후 다층 퍼셉트론의 입력층으로 사용해야 함 → 공간적인 구조 정보를 보존하지 못함



채널(Channel)

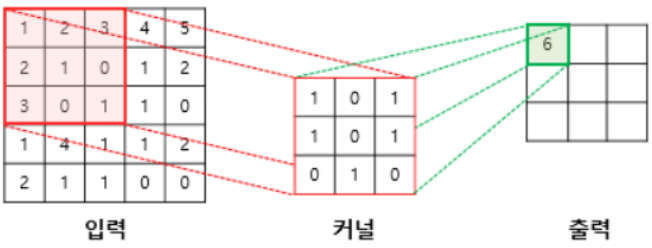
- 이미지 데이터 : 3차원(세로, 가로, 채널)의 형상 → 공간적 구조를 가짐

convolution layer에 유입되는 입력데이터에는 한 개 이상의 필터(/커널)가 적용된다. **1개 필터(/커널)**는 **Feature Map의 한 채널**가 됨. 즉, convolution layer에 **n**개의 필터(/커널)가 적용된다면 출력 데이터는 **n**개의 채널 가짐.

합성곱 연산(Convolution operation)

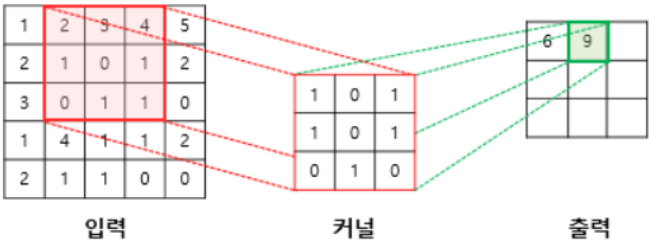
- 이미지의 특징 추출
- 합성곱 연산 수행 방법 : **커널(kernel)** 또는 **필터(filter)**라는 $n \times m$ **크기의 행렬**로 높이 $(height) \times$ 너비 $(width)$ 크기의 이미지 행렬을 처음부터 끝까지 겹치며 훑으면서, $n \times m$ 크기의 겹쳐지는 부분의 각 이미지와 커널의 원소의 값을 **곱**해서 모두 **더한** 값을 출력
- example : 3×3 **크기의 커널**로 5×5 크기의 이미지에 합성곱 연산 수행

1. 첫번째 스텝



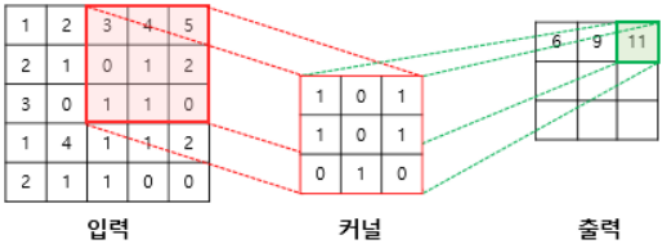
(1×1) + (2×0) + (3×1) + (2×1) + (1×0) + (0×1) + (3×0) + (0×1) + (1×0) = 6

2. 두번째 스텝



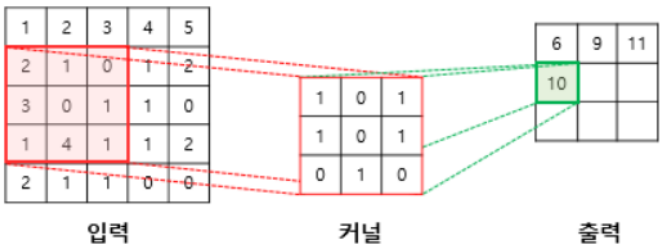
(2×1) + (3×0) + (4×1) + (1×1) + (0×0) + (1×1) + (0×0) + (1×1) + (1×0) = 9

3. 세번째 스텝



(3×1) + (4×0) + (5×1) + (0×1) + (1×0) + (2×1) + (1×0) + (1×1) + (0×0) = 11

4. 네번째 스텝



(2×1) + (1×0) + (0×1) + (3×1) + (0×0) + (1×1) + (1×0) + (4×1) + (1×0) = 10

....

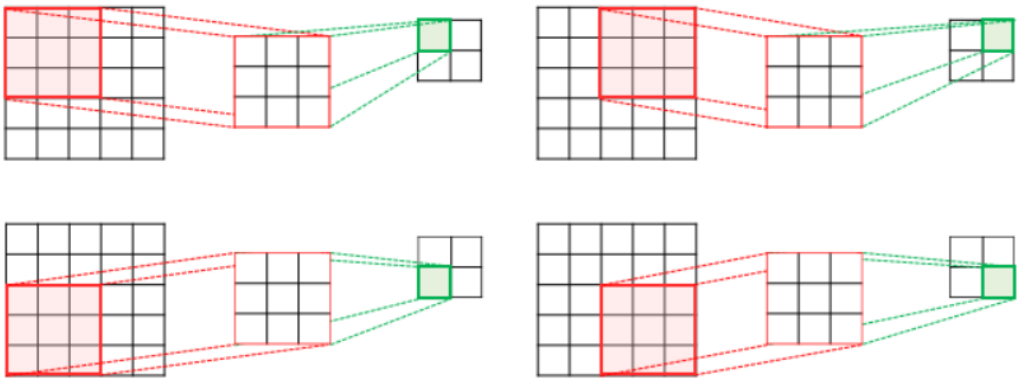
9번의 스탭을 거친 최종 결과

6	9	11
10	4	4
7	7	4

특성 맵(feature map)

- 스트라이드(stride) : 입력데이터에 필터를 적용할 때 **필터가 이동할 간격**

- 위 예제에서는 스트라이드가 1인 경우
- 아래 예제는 스트라이드가 2인 경우



패딩(Padding)

- 합성곱 연산의 결과로 얻은 feature map은 입력보다 크기가 작아짐

  → 합성곱 층을 여러개 쌓는다면 최종적으로 얻은 feature map은 초기 입력보다 크기가 매우 작아짐
- **패딩(padding)** : 합성곱 연산 수행 전, 입력데이터 주변을 특정값으로 채워 늘리는 것.
  합성곱 연산 이후에도 feature map 크기가 입력의 크기와 동일하게 유지되도록 하거나 feature map 크기를 조절하기 위해 사용됨

합성곱 층(convolution layer)

- 합성곱 연산 + 활성화 함수(ReLU 등)

feature map의 크기 계산 방법

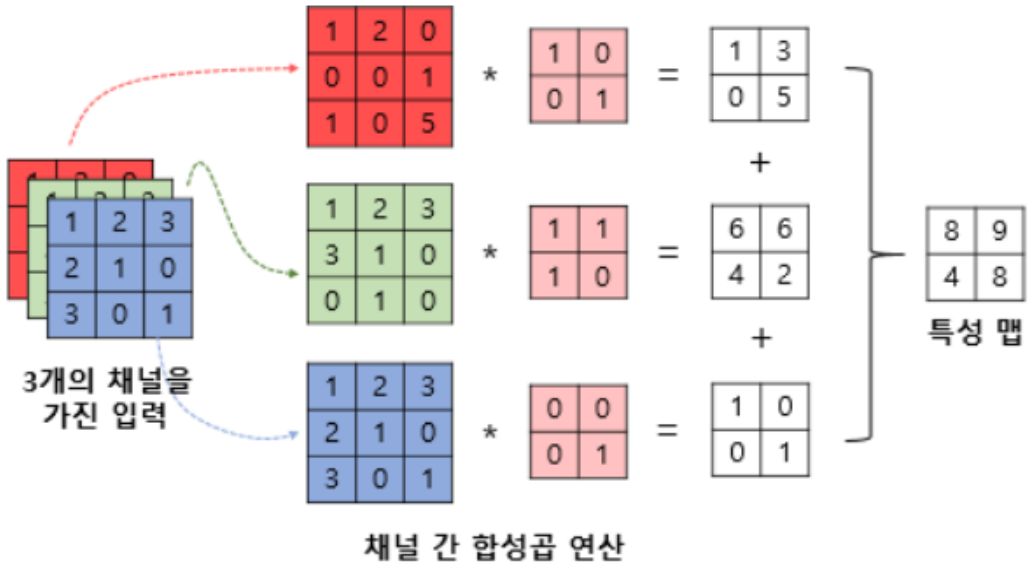
- 입력크기, 커널크기, 스트라이드 값 , 패딩 값→ feature map 크기 계산 가능

$$O_h = floor(\frac{I_h + 2P - k_h}{S} + 1)$$
$$O_w = floor(\frac{I_w + 2P - K_w}{S} + 1)$$

  - (I_h, I_w) : 입력의 크기
  - (K_h, K_w) : 필터/커널 크기
  - S : 스프라이드 (stride)
  - P : 패딩 (padding)
  - (O_h, O_w) : 출력 크기

3차원 텐서의 합성곱 연산

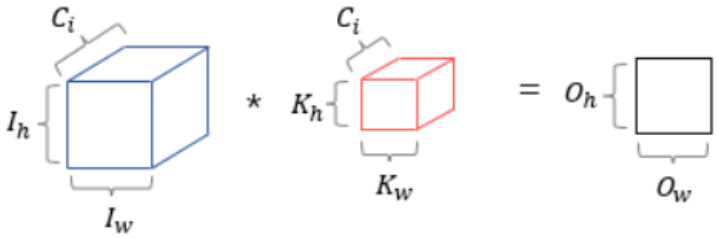
- 합성곱 연산의 입력 : **다수의 채널을 가진** 이미지 or 이전 연산 결과 (**feature map**)
  즉, 3차원 텐서
- 유의할 점 : **입력데이터**의 채널수와 **커널**의 채널 수는 같아야 함
- example : 3개의 채널을 가진 입력데이터와 3개의 채널을 가진 커널의 **합성곱 연산**



- 유의할 점 : 연산에서 사용되는 커널은 3개의 커널이 아니라 **3개의 채널을 가진 1개의 커널**

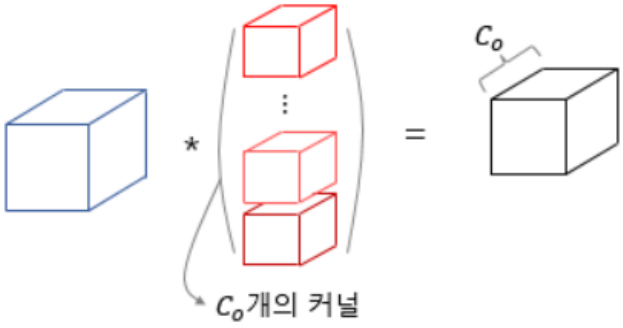
3차원 텐서의 합성곱 연산(일반화)

- 변수 의미
  - (I_h, I_w) : 입력의 크기
  - (K_h, K_w) : 필터/커널 크기
  - (O_h, O_w) : 출력 크기
  - C_i : 입력 데이터의 채널
  - C_o : 커널의 수
- 합성곱 연산에 사용하는 커널의 수가 **1**인 경우



→ (output :) 채널의 수가 1인 feature map

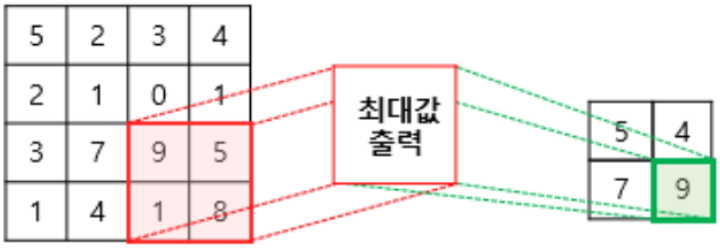
- 합성곱 연산에 사용하는 커널의 수가 **다수**(C_o)인 경우



→ (output :) 채널의 수가 C_o인 feature map

풀링(Pooling)

- 일반적으로 합성곱 층(합성곱 연산 + 활성화 함수) 다음에는 **풀링층**을 추가함
- 풀링 연산 : feature map을 다운 샘플링하여 feature map의 크기를 줄이는 연산 → feature map의 가중치 개수 줄임
- 종류
  - max pooling : 커널과 겹치는 영역 안에서 **최대값**을 추출하는 방식으로 다운 샘플링
  - average pooling : 커널과 겹치는 영역 안에서 **평균값**을 추출하는 방식으로 다운 샘플링
  - etc.
- example : 스트라이드가 2일 때, 2 x 2 크기 커널로 max pooling 연산



출처

- https://wikidocs.net/64066
- https://excelsior-cjh.tistory.com/180
- http://taewan.kim/post/cnn/