week7_Chapter9 CNN을 활용한 이미지 인식

출처

https://deep-flame.tistory.com/18

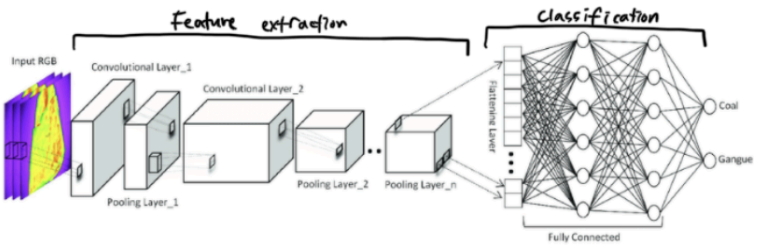
https://rubber-tree.tistory.com/116

1. CNN이란?

attachment:a0485e26-c3d3-46bd-a9c4-8b423facbf23:20250305-1429-58.6603855.mp4

- 개요
  - Convolution neural network의 약자이며 이미지 분석에 자주 사용되는 주변 정보에 영향을 받는 데이터에 적합한 딥러닝 기법.
  - 인간의 시신경 구조를 모방한 기술.
  - 특징맵을 생성하는 필터까지도 학습이 가능해 CV 분야에서 성능이 우수함.
  - 자율주행자동차, 얼굴인식과 같은 객체 인식 등 필요한 분야에 많이 사용되고 있음
  - 이미지의 공간 정보를 유지한 채 학습을 하게 하는 모델로써 1D로 변환하는 것이 아닌 2D 그대로 작업함.
- 구조
  - Feature extraction/learning(이미지 특징 추출)
    - Convolution layer: 입력 데이터에 필터를 적용 후 활성화 함수를 반영하는 필수 요소.
    - Pooling layer
  - Classification(클래스 분류)

- 이미지 분류를 위해 Fully Connected Layer가 추가되며 이미지의 특징을 추출하는 부분과 이미지를 분류하는 부분 사이에 이미지 형태의 데이터를 배열 형태로 만드는 Flatten layer가 위치함.



CNN은 Convolution과 Pooling을 반복적으로 사용하면서 불변하는 특징을 찾고, 그 특징을 입력 데이터로 Fully-connected 신경망에 보내 Classification을 수행.

- CNN의 장점
  - 공간적 관계 유지

    입력 이미지의 작은 영역에 대해서만 필터 적용. 즉, 인접 픽셀 간의 관계 유지하고 학습할 수 있으며, 이미지의 공간 패턴을 인식하는 데 유리함.
  - 파라미터 수 감소

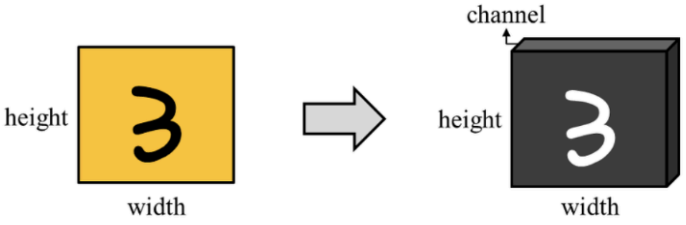
    동일한 필터를 이미지 전체에 걸쳐 적용하기 때문에 파라미터 수가 크게 줄어듦. 이로 인해 모델이 더 효율적이고 메모리가 절약됨.
  - 변환 불변성

    필터가 이미지의 모든 위치에 동등하게 적용되기 때문에, 객체의 위치 변화에 대해 비교적 강인함. 이미지 내 객체의 위치가 약간 변해도 그 특징을 잘 인식할 수 있게 함.

1) Convolution(합성곱 연산 처리)

- 개요

이미지 데이터는 높이x너비x채널의 3차원 텐서로 표현될 수 있으며 이미지 색상이 RGB 코드로 표현되었다면, 채널의 크기는 3이 되고 각각의 채널은 R,G,B값이 저장됨.



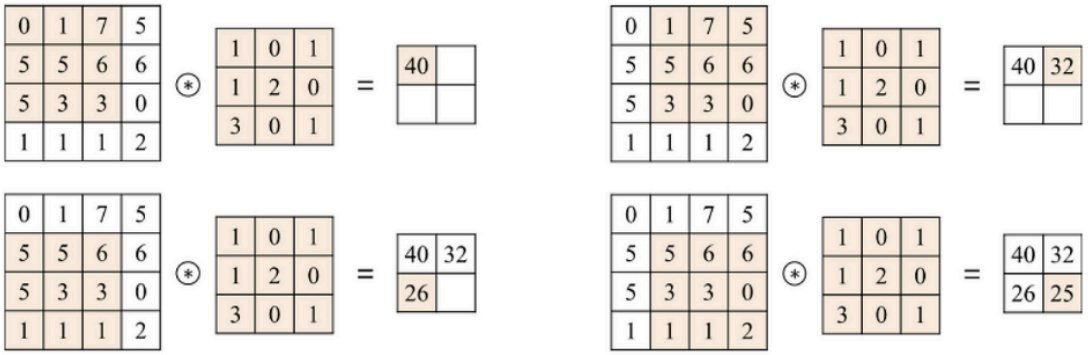
<컬러 이미지 데이터에 대한 tensor 표현>

- **filter 적용**

하나의 합성곱 계층에는 입력되는 이미지 채널 개수만큼 필터가 존재하며, 각 채널에 할당된 필터를 적용함으로써 합성곱 계층의 출력 이미지가 생성됨.

보통 3x3 혹은 5x5와 같은 정방향 배열이 filter가 되며 이를 합성곱 처리에서는 '커널'이라고 부름.

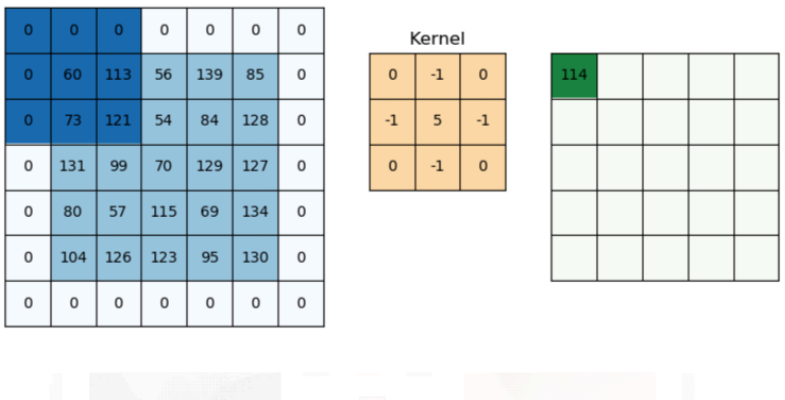
example 1) 4×4×1 텐서 형태의 입력 이미지에 대해 3x3 필터(각 filter 안에 weight)를 적용하여 2x2×1 텐서 형태 이미지 생성.



하나의 채널에 대한 Convolution(합성곱) 계층의 동작

example 2) padding이 적용된 합성곱 수행 예시

11148cnn_convolve_with_padding.gif (800×400)



0 * 0 + -1 * 0 + 0 * 0 +

0 * -1 + 60 * 5 + 113 * -1

0 * 0 + 73 * -1 + 121 * 0 = 114

attachment:a50c5370-00d9-4aea-9d60-c09e826b0d08:20250305-1242-44.2783626.mp4

- **stride**

이미지에 대해 필터를 적용할 때 필터의 이동량을 의미.

stride 값이 1이면 filter가 한번에 1 fixel씩 이동하며 stride 값이 2이면 2 fixel씩 이동하며 feature map을 만들 때 stride 값이 커질수록 feature map이 작아짐.

보통 CNN 구현할 때 합성곱 계층의 stride는 주로 1로 설정됨.

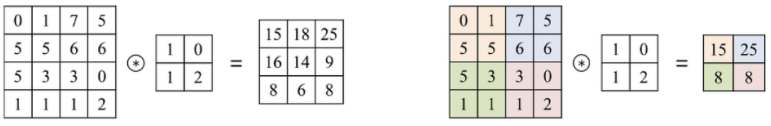
<stride 값이 1인 경우>

attachment:80121b59-ac01-43cd-8572-9a3a56ba914f:20250305-1240-49.4938115.mp4

<stride 값이 2인 경우>

attachment:19fade21-084d-4943-8fe2-f1f63a7bdfcd:20250305-1243-52.1108116.mp4

(왼쪽) stride = 1인 경우, (오른쪽)stride = 2인 경우

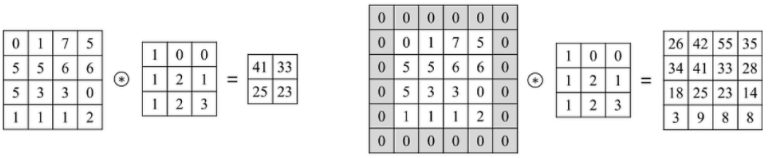


- **padding**

입력 이미지에 대해 합성곱을 수행하면, 출력 이미지의 크기는 입력 이미지의 크기보다 작아지게 되는데 합성곱 계층을 거치면서 이미지의 크기는 점점 작아지게 되고, 이미지의 가장자리에 위치한 픽셀들의 정보는 사라지게 되는 문제점 발생.

이를 해결하고자 사용되는 것이 padding이며, 입력 이미지의 가장 자리에 특정 값으로 설정된 픽셀들을 추가하여 입력 이미지와 출력 이미지의 크기를 같거나 비슷하게 만드는 역할을 수행.

CNN에서는 주로 이미지 가장자리에 0의 값을 갖는 zero-padding 이용됨.

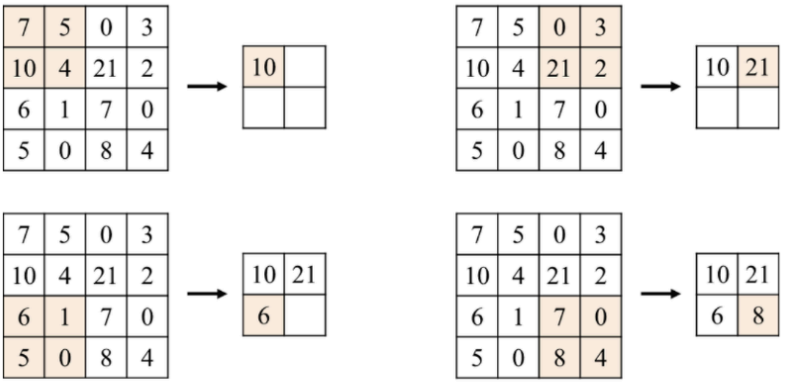


(왼쪽) Padding을 안한 경우, (오른쪽) Zero-padding인 경우

2) Pooling layer

attachment:da2e5f6c-de64-46d3-8922-dee7b9e67095:20250305-1438-27.1516834.mp4

- max-pooling
  - 이미지의 크기를 계속 유지한 채 fully connected layer로 가게 되면 연산량이 기하급수적으로 늘기 때문에 적당히 크기도 줄이며 특정 feature를 강조할 때 pooling layer에서 그 역할을 하게 됨.
  - pooling의 종류는 max pooling, average pooling, min pooling이 있으며 CNN에서는 주로 **max pooling**이 사용되며 노이즈가 감소하고 속도가 빨라지며 영상의 분별력의 좋아짐.
  - <stride=2로 설정된 max pooling 풀링 계층>

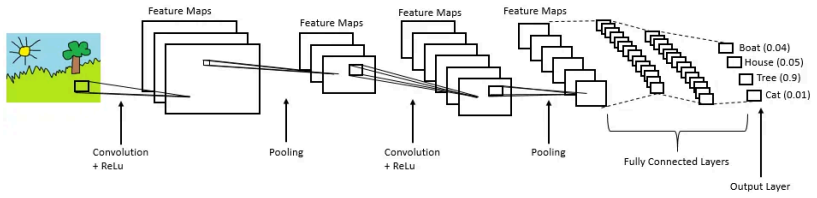


Max-pooling이 적용된 Pooling layer

        - 2×2 크기의 선택 영역마다 max-pooling을 적용했으며 일반적으로 pooling layer의 stride는 선택 영역의 높이 또는 너비의 크기와 동일하게 설정됨.
        - 이미지 데이터 특징은 인접 픽셀들 간 유사도가 매우 높으며 max-pooling의 경우 선택 영역에서 가장 큰 값을 해당 영역의 대표값으로 설정.
    - 이점
        - 선택 영역 내부에서 픽셀들이 이동 및 회전 등에 의해 위치가 변경되더라도 출력값은 동일하기 때문에 이미지를 구성하는 요소들의 이동 및 회전 등에 의해 CNN의 출력값이 영향을 받는 문제 완화.
        - CNN이 처리해야 되는 이미지 크기가 크게 줄어들기 때문에 인공신경망의 model parameter 또한 크게 감소함으로써 CNN의 학습 시간을 크게 절약할 수 있으며, overfitting 문제 역시 완화 가능.

3) Fully Connected Layer

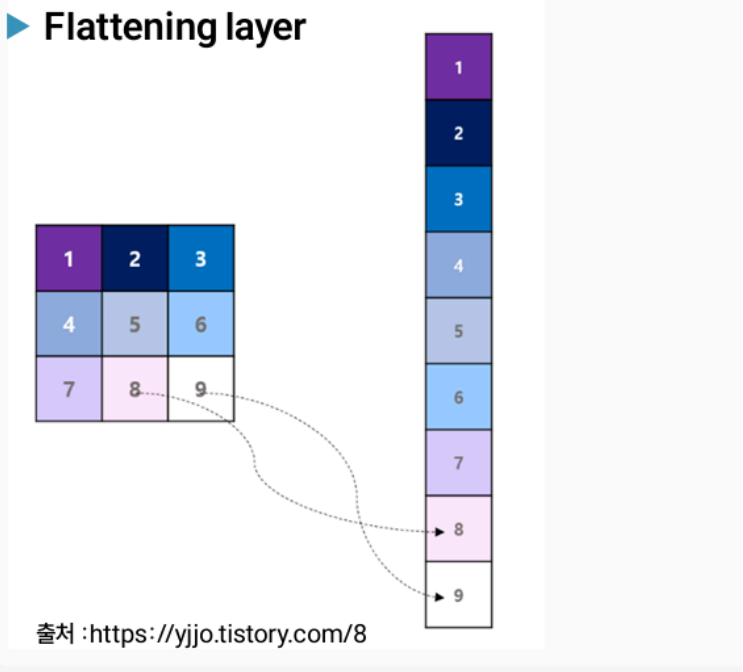
특징 추출을 위한 작업을 완료하였으면 이미지 특징을 추출하여 이것이 무엇을 의미하는 데이터 인지를 분류 작업하는 layer이다.



fully connected layer의 출력 확률의 합 = 1.

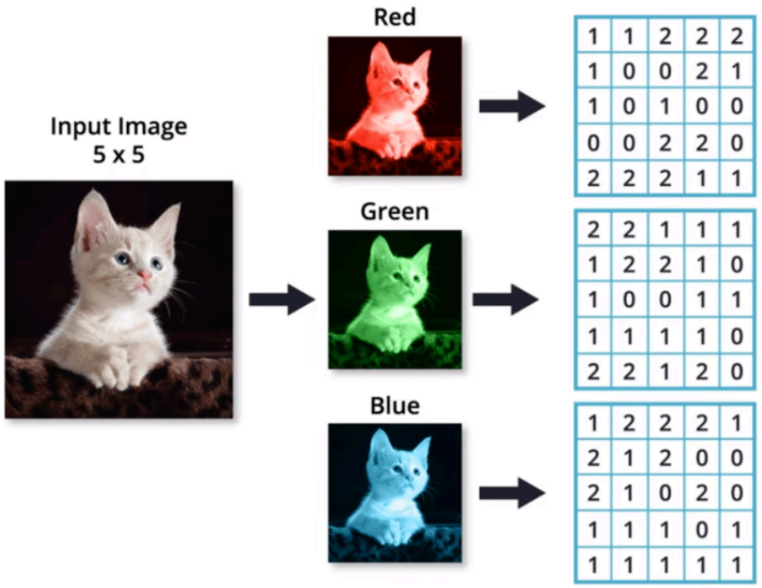
softmax를 fully connected layer의 출력 계층에서 활성화 함수로 활용하여 수행됨.

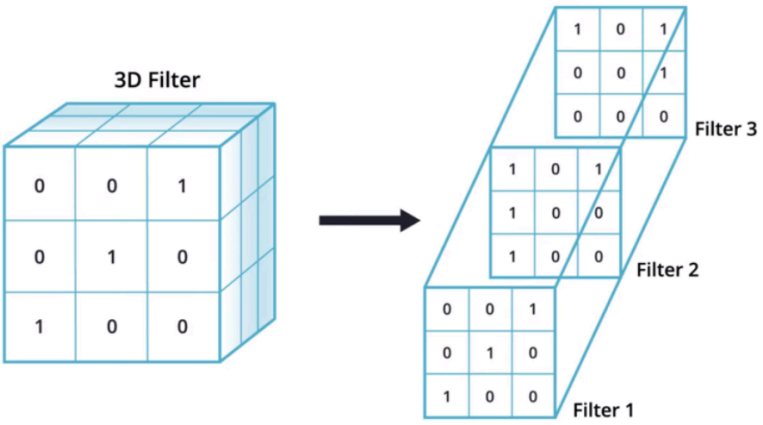
- 종류
    - Flatten Layer: 데이터 타입을 Fully Connected 네트워크 형태로 변경하여 입력 데이터의 shape 변경만 수행.



- Softmax Layer: Classification 수행.

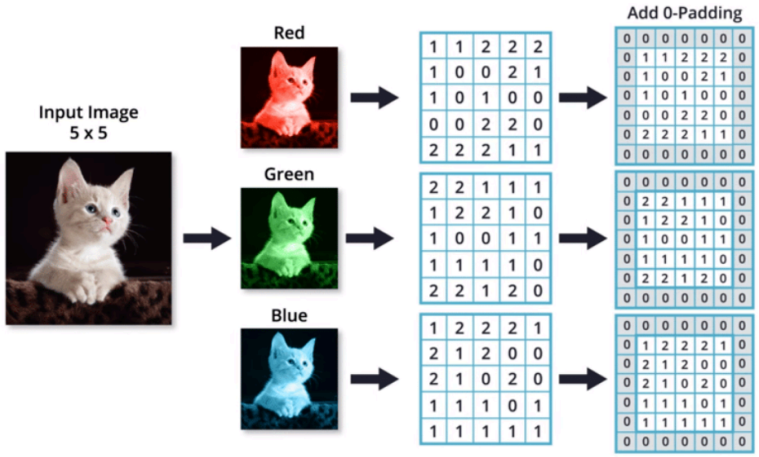
4) Convolution Filter 예시

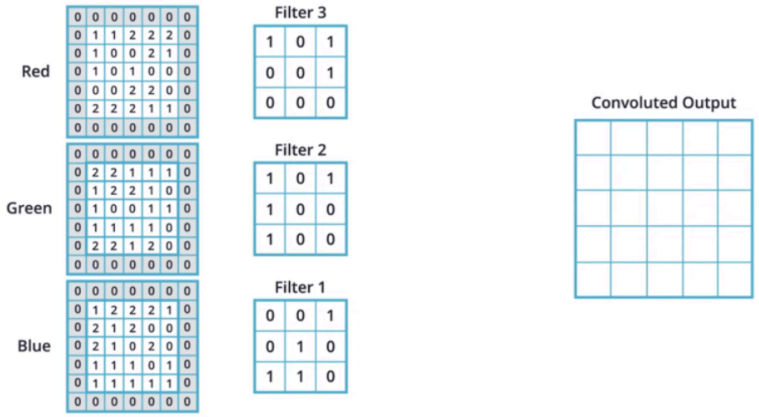




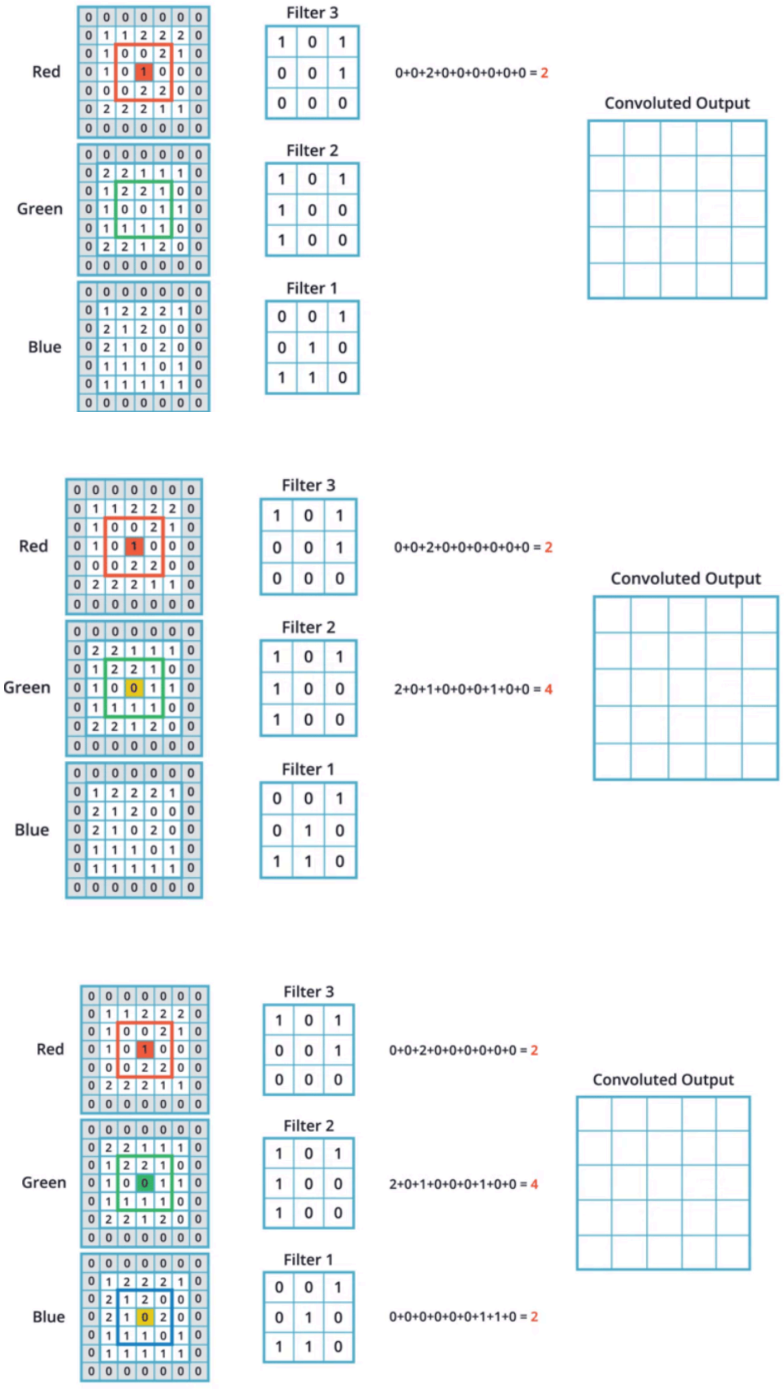
RGB 이미지가 있고 다음과 같은 3D Filter로 Convolution 하고 싶다고 가정해보자. 필터의 depth는 3개의 2D 필터로 구성되어 있으며 RGB 이미지가 5x5 pixel이라고 가정해보자. 이때 입력의 채널 수와 필터의 채널 수가 같아야 한다.

- 정보 손실 방지를 위해 각 배열에 0-padding 추가

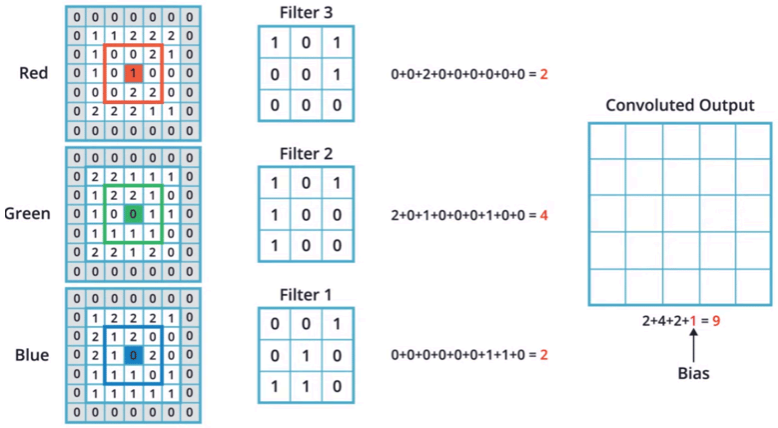


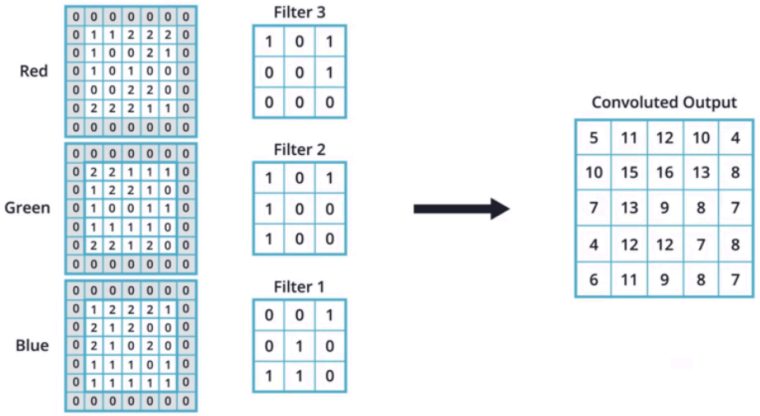


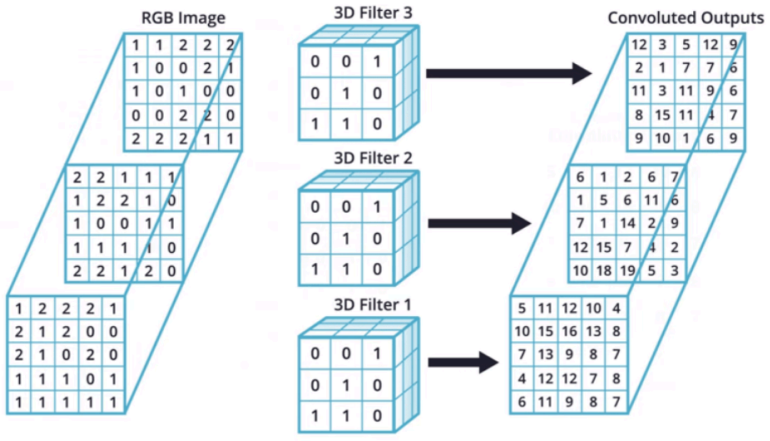
- RGB에 대하여 convolution 진행



- 각 필터를 RGB 각각에 합성곱 연산 진행 후 bias(기본적으로 1 사용)를 더한 최종값 = convoluted output의 한 픽셀 값.


Red
Green
Blue
Filter 3
Filter 2
Filter 1
0+0+2+0+0+0+0+0+0 = 2
2+0+1+0+0+0+1+0+0 = 4
0+0+0+0+0+0+1+1+0 = 2
Convoluted Output
2+4+2+1 = 9
Bias


Red
Green
Blue
Filter 3
Filter 2
Filter 1
Convoluted Output
5 11 12 10 4
10 15 16 13 8
7 13 9 8 7
4 12 12 7 8
6 11 9 8 7


RGB Image
3D Filter 3
3D Filter 2
3D Filter 1
Convoluted Outputs