CNN Basic

AI – 스쿨

2022. 06. 07.

Infinyx®

I

CNN Basic

1. CNN ?
2. 주요 컨셉 4가지 개념 정리
3. CNN 실습
   - Convolution Layers
4. 과제 설명

CNN 이 무엇일까 ?

CNN은 **Convolutional Neural Networks**의 약자로 딥러닝에서 주로 이미지나 영상 데이터를 처리할 때 쓰이며 이름에서 알수있다시피 Convolution이라는 전처리 작업이 들어가는 Neural Network 모델입니다.

그렇다면 왜 CNN이라는 방법을 쓰기 시작했을까요?

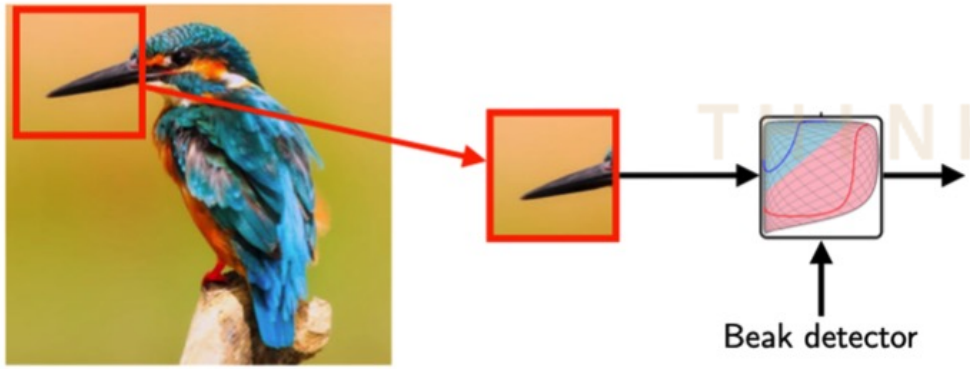
DNN(Deep Neural Network)의 문제점에서부터 출발합니다. 일반 DNN은 기본적으로 1차원 형태의 데이터를 사용합니다.

때문에 (예를들면 1028x1028같은 2차원 형태의)이미지가 입력값이 되는 경우, 이것을 flatten시켜서 한줄 데이터로 만들어야 하는데 이 과정에서 이미지의 공간적/지역적 정보(spatial/topological information)가 손실되게 됩니다.

또한 추상화과정 없이 바로 연산과정으로 넘어가 버리기 때문에 학습시간과 능률의 효율성이 저하됩니다.

DNN 문제를 해결하기 위한 방법

CNN은 이미지를 날것(raw input) 그대로 받음으로써 공간적/지역적 정보를 유지한 채 특성(feature)들의 계층을 빌드업합니다. CNN의 중요 포인트는 이미지 전체보다는 부분을 보는 것, 그리고 이미지의 한 픽셀과 주변 픽셀들의 연관성을 살리는 것입니다.



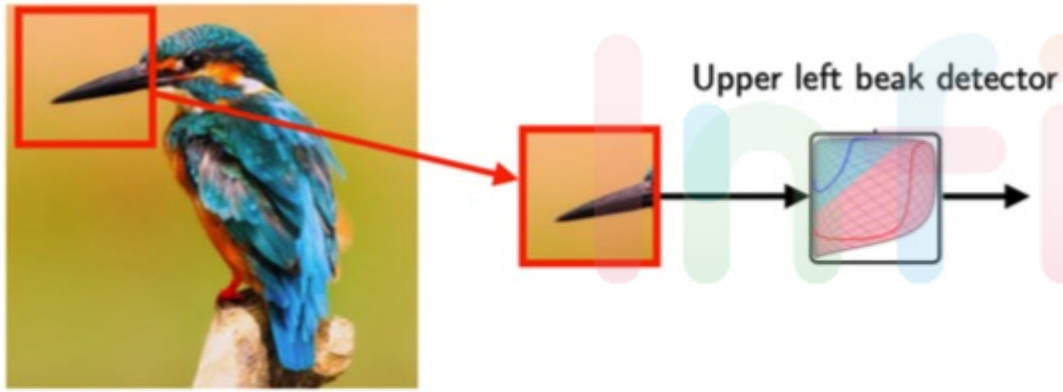
예를 들어 우리는 어떠한 이미지가 주어졌을때 이것이 새의 이미지인지 아닌지 결정할 수 있는 모델을 만들고 싶다고 가정합시다.

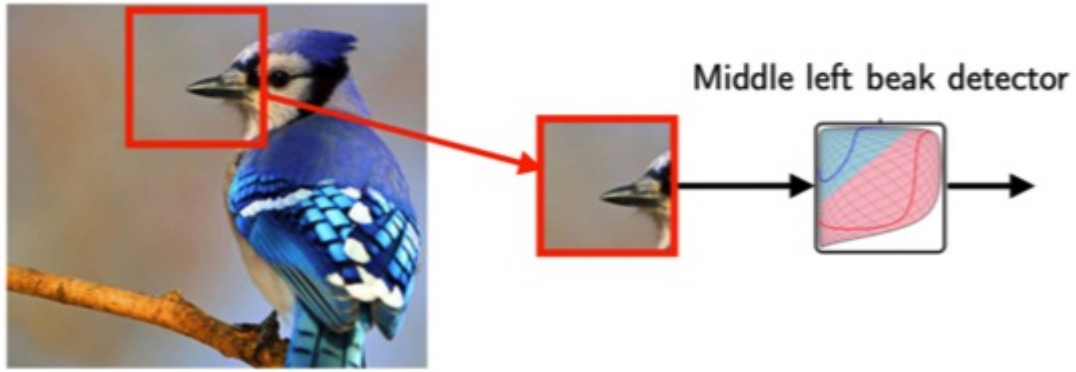
그렇다면 새의 주요 특징인 새의 부리가 중요한 포인트가 될 수 있습니다.

때문에 모델이 주어진 이미지에 새의 부리가 있는지 없는지 판가름 하는것이 중요 척도가 될 것입니다.

하지만 새의 전체 이미지에서 새의 부리 부분은 비교적 작은 부분입니다. 때문에 모델이 전체 이미지를 보는 것 보다는 새의 부리 부분을 잘라 보는게 더 효율적이겠죠? 그것을 해주는것이 CNN입니다.

DNN 문제를 해결하기 위한 방법





또한 위의 두 이미지를 보면 새의 부리부분이 이미지의 다른 부분에 위치해 있다는 것을 알 수 있습니다

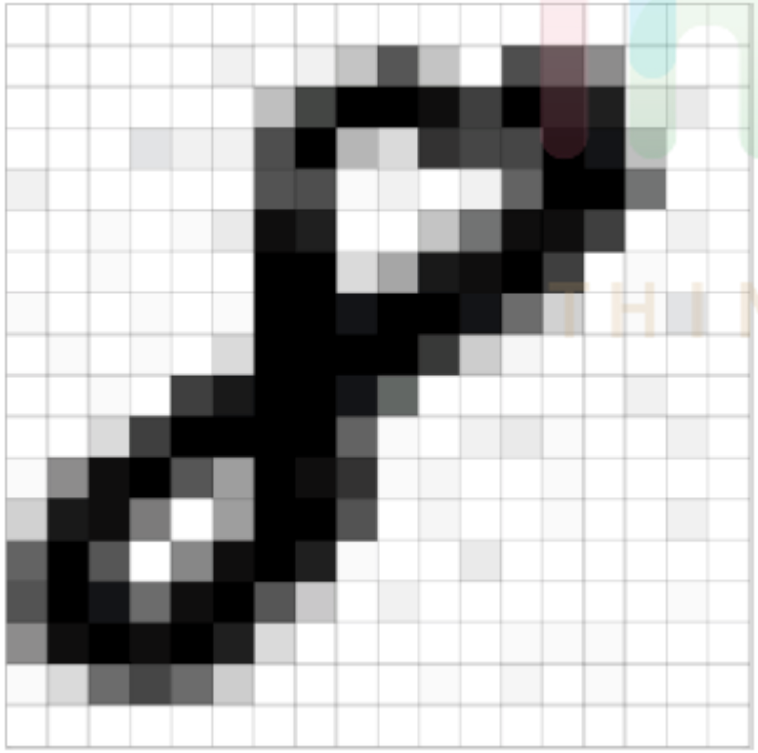
위의 이미지는 이미지의 새의 부리가 왼쪽 상단에 위치한 반면 아래의 이미지는 상단 가운데 부분에서 약간 왼쪽 부분에 위치.

때문에 전체적인 이미지보다는 이미지의 부분 부분을 캐치하는 것이 중요하고 효율적일 수 있습니다.

CNN 주요 컨셉

1. Convolution의 작동 원리

우선 2차원의 이미지를 예로 들어 봅시다. 2차원 이미지는 픽셀 단위로 구성되어 있습니다.



이미지는 MNIST Dataset에서 추출한 샘플중 하나이며 손글씨로 쓰여진 '8'의 gray scale 이미지 입니다.

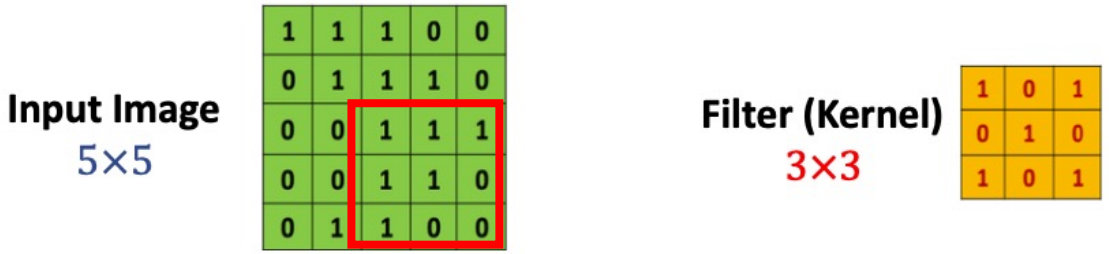
자세히 보면 28x28단위의 픽셀로 구성되어 있습니다.
우리는 이 데이터 입력값을 28x28 matrix(행렬)로 표현할 수 있습니다.

즉, 우리는 2차원 이미지를 matrix로 표현할 수 있다는 뜻입니다.

우리가 CNN에 넣어줄 입력값은 이렇게 *matrix로 표현된 이미지* 입니다.

CNN 주요 컨셉

1. Convolution의 작동 원리



조금 더 간략한 예를 위해 위와 같이 5x5의 matrix로 표현된 이미지 입력값이 있다고 가정합시다.
그리고 CNN에는 필터(커널)가 존재합니다.
위의 예시의 경우에는 3x3크기의 필터입니다. 쉽게 표현하면 이 하나의 필터를 이미지 입력값에 전체적으로 훑어준다고 생각하면 됩니다.
즉 ***우리의 입력값 이미지의 모든 영역에 같은 필터를 반복 적용해 패턴을 찾아 처리하는 것이 목적입니다.***

그렇다면 이미지 위에 이 필터를 훑어줄때 어떤 일이 일어날까요? 정확히 말하면 이 필터를 이용해 연산처리를 해주는 것입니다.

그리고 이 연산처리는 matrix와 matrix간의 *Inner Product*라는 것을 사용합니다.

CNN 주요 컨셉

Inner product(내적 공간)

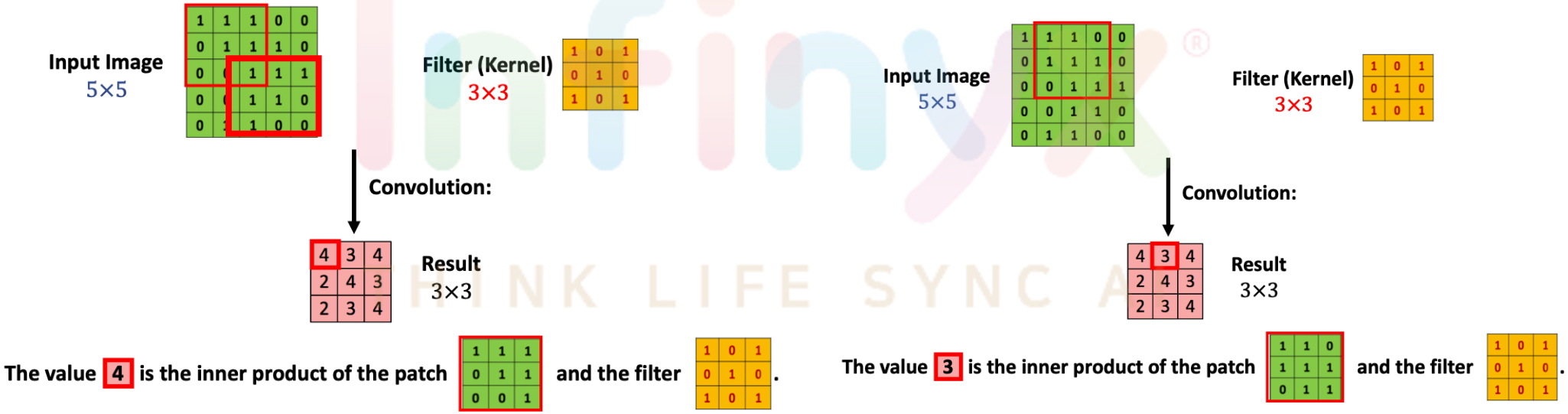
- $\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$.
- Inner product:

$$\langle \mathbf{A}, \mathbf{B} \rangle = \sum_i \sum_j a_{ij} b_{ij} = 70.$$

쉽게 말해서 크기의 두 matrix를 놓고 각 위치에 있는 숫자를 곱해 모두 더해주는 것입니다.

CNN 주요 컨셉

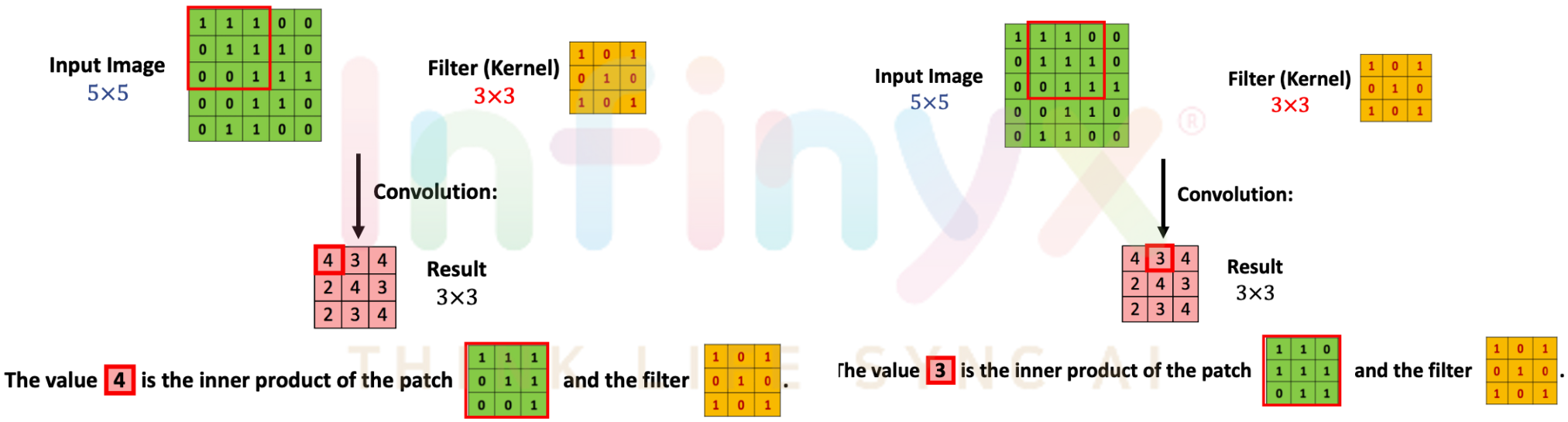
어떻게 이 이미지 입력값 matrix의 각 부분과 필터가 연산처리 되는지 그림으로 살펴보겠습니다.



우선 빨간 테두리 부분의 matrix와 필터의 Inner product연산을 해주면 그 결과값은 4가 됩니다.

이번엔 이 빨간 테두리 부분의 matrix와 필터의 연산처리를 해주어 결과값 3을 얻어냅니다.
이러한 과정을 전체 이미지 matrix에 필터 matrix를 조금씩 움직이며 반복해 줍니다.
그렇게 반복하여 얻어낸 결과값이 분홍색으로 표시된 matrix입니다.

CNN 주요 컨셉



자세히 보면 결과값의 크기(단어 선택의 혼동이 있을 수 있습니다.
여기서 크기란 matrix의 dimension을 가르킵니다)는 3x3이라는 것을 파악할 수 있습니다.

왜 그런지 다들 눈치 채셨나요? 5x5로 구성된 칸들에 3x3크기의 블록을 움직인다고 생각해보면
너비와 높이 둘다 3번씩 놓을 수 있기 때문입니다.

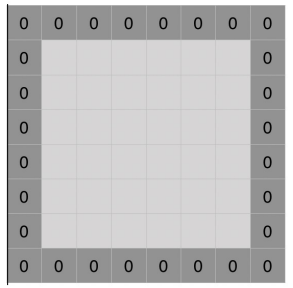
CNN 주요 컨셉

2. Zero Padding

Convolution처리를 보면 5x5크기의 이미지에 필터처리를 해주었더니 결과값의 크기가 3x3로 줄어들었다는 것을 알 수 있습니다. 즉 손실되는 부분이 발생한다는 뜻입니다.

이런 문제점(?)을 해결하기 위해 Padding이라는 방법을 사용할 수 있습니다.

쉽게 말해 0로 구성된 테두리를 이미지 가장자리에 감싸 준다고 생각하면 됩니다.



Zero-padding added to image

위의 경우 5x5크기의 이미지 입력값이 7x7크기가 될겁니다.

이 상태에서 3x3필터를 적용해줄 경우 똑같이 5x5크기의 결과값을 얻을수 있겠죠?

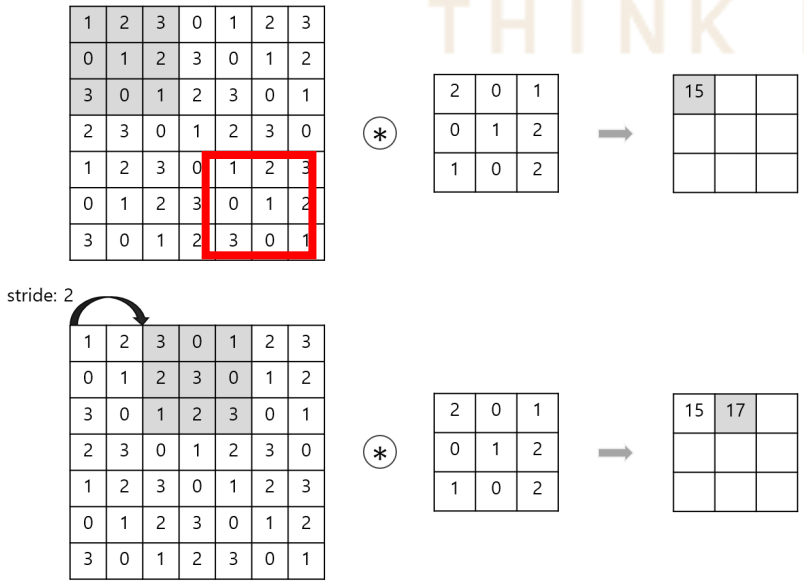
이렇게 되면 *입력값의 크기와 결과값의 크기가 같아졌음*, 즉 손실이 없어졌음을 볼 수 있습니다.

CNN 주요 컨셉

3. Stride

Stride는 쉽게 말해 필터를 얼마만큼 움직여 주는가에 대한 것입니다.
stride값이 1일경우 필터를 한칸씩 움직여 준다고 생각하면 됩니다

참고로 stride-1이 기본값이고 stride는 1보다 큰 값이 될 수도 있습니다. stride값이 커질 경우 필터가 이미지를 건너뛰는 칸이 커짐을 의미하므로 결과값 이미지의 크기는 작아짐을 의미합니다.



참고로 입력 데이터 크기와 패딩, 스트라이드를 고려하여 출력 데이터의 크기를 계산하는 수식은 아래와 같다.

- 입력 크기=(H, W)
- 필터 크기=(FH, FW)
- 출력 크기=(OH, OW)
- 패딩=P
- 스트라이드=S

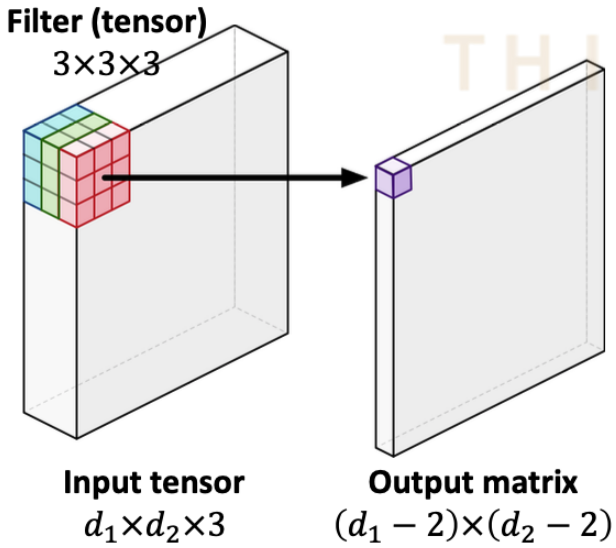
$$OH = \frac{H + 2P - FH}{S} + 1$$

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

CNN 주요 컨셉

4. The Order-3 Tensor 3차원 형성의 합성곱 연산

2차원 이미지의 입력값을 살펴보았습니다. 물론 입력값이 3차원인 경우도 존재합니다.
예를들면 컬러이미지는 R, G, B의 세가지 채널로 구성되어 있기 때문에 d_1 x d_2 x 3 과 같은 삼차원의 크기를 갖습니다.
이러한 모양을 order-3 텐서라고 칭합니다.
유사하게 익숙한 2차원의 이미지는 order-2 텐서(즉 matrix)라고도 칭합니다.

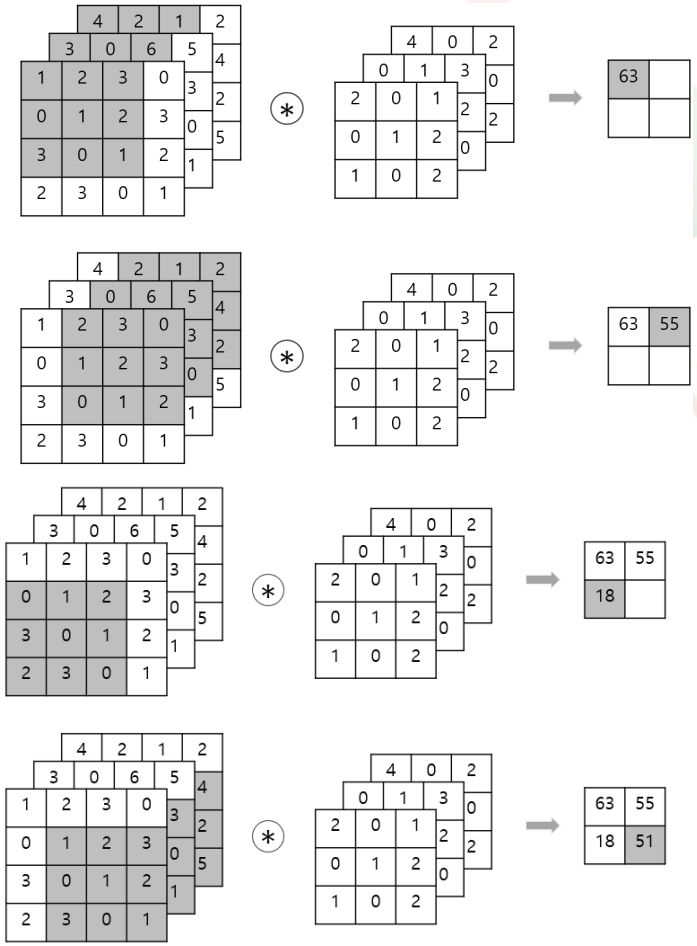


이미지와 같이 이 경우에 필터는 k1xk2x3의 크기를 가진 order-3 텐서가 됩니다. 연산처리는 똑같이 Inner product를 사용합니다.

그러면 당연히 결과값의 모양은 matrix가 되겠죠?

CNN 주요 컨셉

4. The Order-3 Tensor 3차원 형성의 합성곱 연산



3차원 데이터의 합성곱 연산에서 주의할 점은 입력 데이터의 채널 수와 필터의 채널 수가 같아야 한다는 것이다. 필터 크기는 무관하다.

합성곱 연산의 출력을 다음 layer에 3차원 형상의 데이터로 전달하기 위해서는 출력 데이터의 채널도 여러개가 필요하다. 출력 데이터가 다수의 채널을 갖도록 하려면 필터를 여러개 사용하면 된다.

CNN In Pytorch 실습

Pytorch에는 CNN을 개발 하기 위한 API들이 있습니다. 다채널로 구현 되어 있는 CNN 신경망을 위한 Layers CNN을 위한 API를 알아 보겠습니다

Convolution Layers

Convolution 연산을 위한 레이어들은 다음과 같습니다.

- Conv1d (Text-CNN에서 많이 사용)
- Conv2d (이미지 분류에서 많이 사용)
- Conv3d (3d 모델에서 많이 사용)

Parameters

```
Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,
groups=1, bias=True, padding_mode='zeros')
```

CNN In Pytorch 실습

Parameters

```
Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,
groups=1, bias=True, padding_mode='zeros')
```

- in_channels: 입력 채널 수을 뜻합니다. 흑백 이미지일 경우 1, RGB 값을 가진 이미지일 경우 3을 가진 경우가 많습니다.
- out_channels: 출력 채널 수을 뜻합니다.
- kernel_size: 커널 사이즈를 뜻합니다. int 혹은 tuple이 올 수 있습니다.
- stride: stride 사이즈를 뜻합니다. int 혹은 tuple이 올 수 있습니다. 기본 값은 1입니다.
- padding: padding 사이즈를 뜻합니다. int 혹은 tuple이 올 수 있습니다. 기본 값은 0입니다.
- padding_mode: padding mode를 설정할 수 있습니다. 기본 값은 'zeros' 입니다. 아직 zero padding만 지원 합니다.
- dilation: 커널 사이 간격 사이즈를 조절 합니다.
- groups: 입력 층의 그룹 수을 설정하여 입력의 채널 수를 그룹 수에 맞게 분류 합니다. 그 다음, 출력의 채널 수를 그룹 수에 맞게 분리하여, 입력 그룹과 출력 그룹의 짝을 지은 다음 해당 그룹 안에서만 연산이 이루어지게 합니다.
- bias: bias(편향) 값을 설정 할 지, 말지를 결정합니다. 기본 값은 True 입니다.

Bias -> 예측값과 실제 정답과의 차이의 평균

CNN In Pytorch 실습

Shape

Input Tensor$(N, C_{in}, H_{in}, W_{in})$의 모양과 Output Tensor$(N, C_{out}, H_{out}, W_{out})$의 모양은 다음과 같습니다.

Input Tensor$(N, C_{in}, H_{in}, W_{in})$

- N: batch의 크기
- C_{in}: `in_channels` 에 넣은 값과 일치하여야 함.
- H_{in}: 2D Input Tensor의 높이
- W_{in}: 2D Input Tensor의 너비

Output Tensor$(N, C_{out}, H_{out}, W_{out})$

- N: batch의 크기
- C_{out}: `out_channels` 에 넣은 값과 일치 함.
- $H_{out} = \lfloor \frac{H_{in}+2\times padding[0]-dilation[0]\times(kernel\_size[0]-1)-1}{stride[0]} + 1 \rfloor$
- $W_{out} = \lfloor \frac{W_{in}+2\times padding[1]-dilation[1]\times(kernel\_size[1]-1)-1}{stride[1]} + 1 \rfloor$

CNN In Pytorch 실습

Convolution Layers – 실습

- 과제 다음과 같음 -> 손으로 계산 하여 캡처 하여 제출 !! Or 문서 제출 하셔도 됩니다.

conv1 연산 후 torch.size -> 어떻게 변경된건가 ?
Conv2 연산 후 torch.size -> 어떻게 변경된건가 ?
차원 감소 후 torch.size -> 어떻게 변경된건가 ?
FC1 연산 후 torch.size -> 어떻게 변경된건가 ?
FC2 연산 후 torch.size -> 어떻게 변경된건가 ?


감사합니다.
THINK LIFE SYNC AI
Infinyx