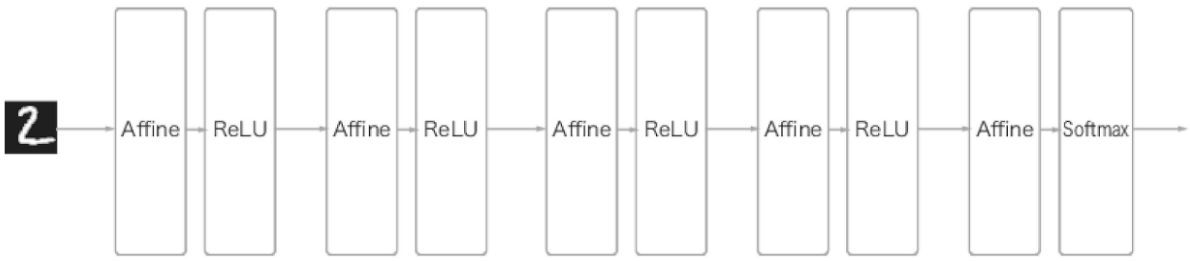
21. CNN (합성곱 신경망)

2020. 8. 7. 11:49

#CNN #Kernel #PADDING #Pooling #stride #스트라이드 #커널 #패딩 #풀링 #합성곱 신경망

MLNN(Multi-Layer Neural Network) : Affine(완전 연결) 계층으로 이뤄진 네트워크

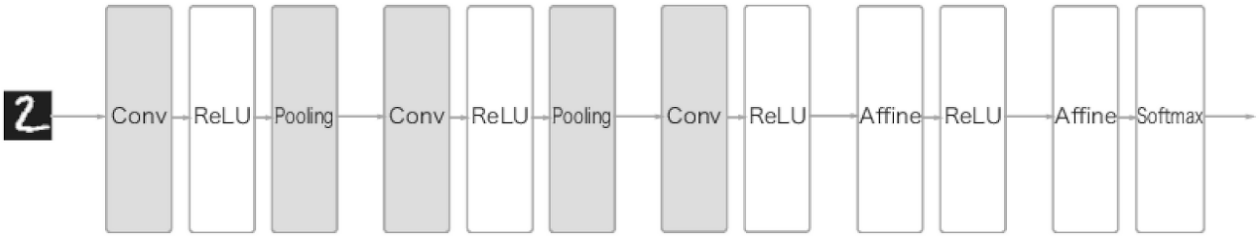
여태 설명했던 신경망은 인접하는 모든 계층의 뉴런과 결합되어 있었다.
이를 완전연결(Fully-Connected)계층(Dense Layer라고도 함)이라고 하며,
완전히 연결된 계층을 Affine 계층이라는 이름으로 구현했다.



Affine 계층으로 이뤄진 신경망

CNN (Convolutional Neural Network)

CNN은 여태 배웠던 신경망이 Affine-ReLU 연결을 한 것과는 달리 ***Conv-ReLU-Pooling의 흐름으로 신경망이 구성***된다.

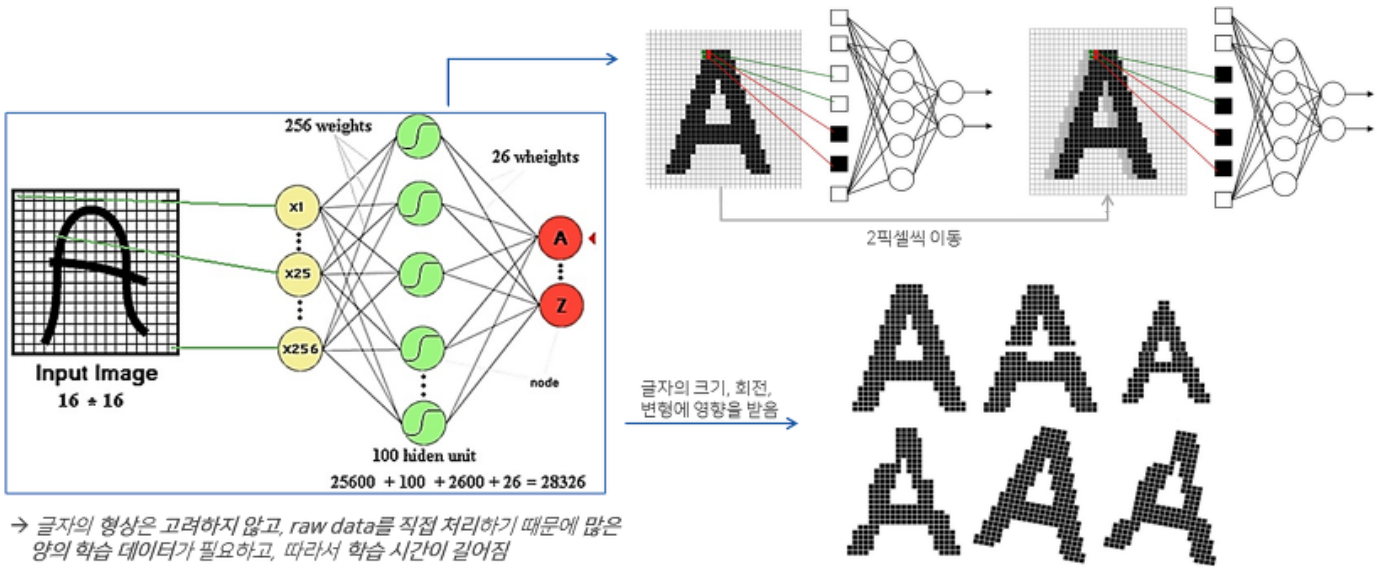


CNN으로 이뤄진 신경망 (합성곱 계층, 풀링 계층 추가)

그런데 왜 CNN을 사용하는걸까?

Affine 계층의 문제점

Affine 계층은 학습을 진행할 때 데이터의 형상을 무시한다. 이미지 데이터는 가로, 세로, 색(RGB) 총 3차원의 데이터인데, Affine 계층으로 학습 시 이 3차원의 데이터를 1차원으로 평평하게 평탄화(Flatten)해야 학습을 할 수 있다. 이렇게 차원을 줄이다보면 ***유의미한 데이터를 정확하게 살릴 수 없다.***



→ *글자의 형상은 고려하지 않고, raw data를 직접 처리하기 때문에 많은 양의 학습 데이터가 필요하고, 따라서 학습 시간이 길어짐*

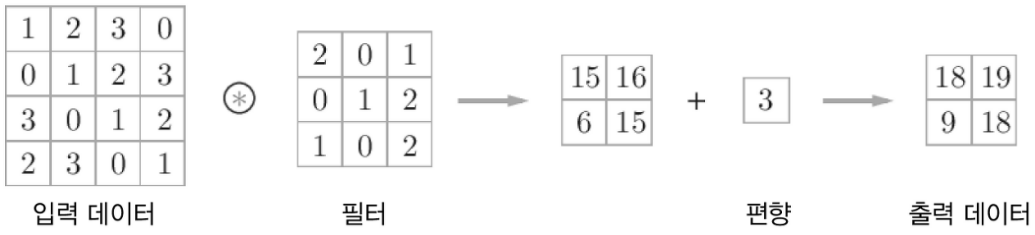
MLNN의 문제점

또한 위의 MLNN의 문제점 이미지를 살펴보면 글자의 형상을 고려하지 않고 학습을하기 때문에 ***데이터가 살짝만 모양이 바뀌어도 새롭게 학습을 시켜야 한다는 문제점***이 있다. 따라서 데이터의 모든 정보를 최대한 손실 없이 학습 시키면서 데이터의 모양 변화에 큰 영향을 받지 않는 새로운 방법이 필요했고, 그에 대한 해답이 CNN이었다.

Conv 계층

Conv(합성곱) 계층의 입출력 데이터를 CNN에서는 특징 맵(Feature Map)이라고도 해서 Conv 계층의 입력 데이터를 입력 특징 맵(Input Feature Map), 출력 데이터를 출력 특징 맵(Output Feature Map)이라고 한다.

Conv 계층은 이름과 일치하게 합성곱 연산을 처리하며, 이는 이미지 처리에서 말하는 필터(커널(Kernel)이라고 주로 표현) 연산에 해당한다. 합성곱 연산은 Kernel의 Window를 일정 간격으로 이동하며 입력 데이터에 적용한다. 이러한 과정은 기존의 Affine 계층으로만 이루어진 신경망과 달리 ***데이터의 형상을 잃지 않으면서 학습을 진행할 수 있도록 (이미지의 특징을 파악할 수 있도록) 도와준다.*** CNN도 기존 신경망과 동일하게 가중치와 편향이 존재하는데 ***필터를 구성하는 매개변수들이 가중치에 해당***하고 ***편향은 항상 1x1로 하나만 존재***한다.

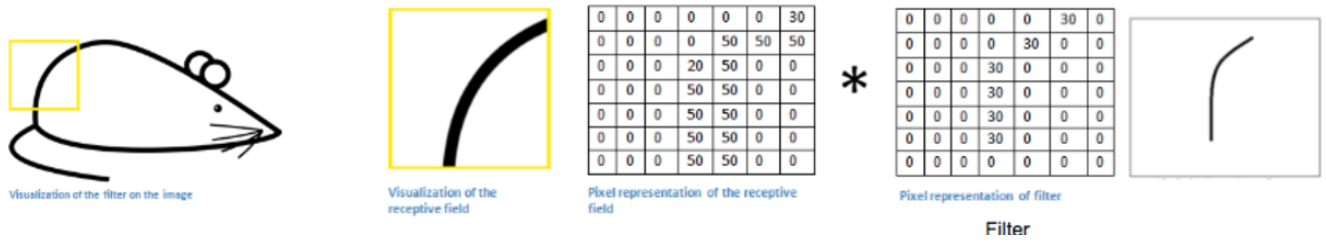


합성곱 연산

필터 (Filter)

필터란 필터에 해당하는 특징이 데이터에 있는지를 검출해주는 함수로, 아래의 그림에서는 곡선이 필터이다.

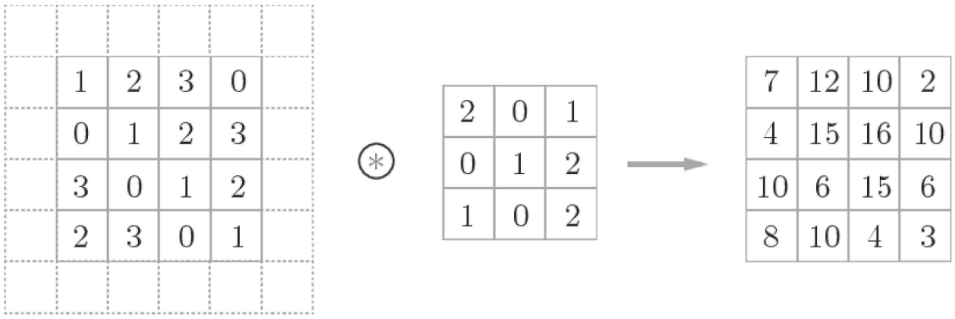
이 필터를 기존의 데이터와 합성곱 연산을 진행하면 필터에 해당하는 특성을 갖고 있으면 결과 값으로 큰 값이 나오고, 특성을 가지지 않으면 결과값이 0에 가깝게 나오면서 ***입력데이터가 필터에 해당하는 특징을 갖고 있는지, 아닌지에 대한 여부를 알 수 있다.***



필터(Filter)에 대한 이해를 돕기 위한 이미지

패딩 (Padding)

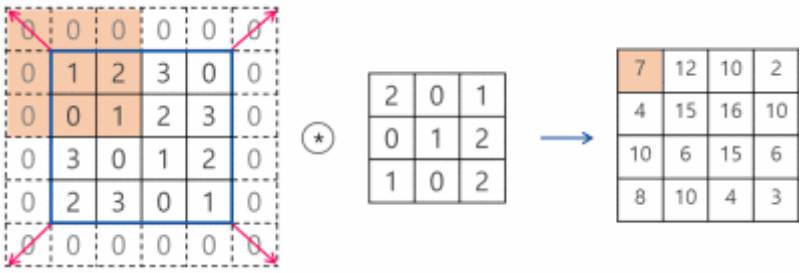
합성곱 연산을 하기 이전, 입력데이터 주변을 특정값으로 채워 늘리는 것을 패딩(Padding)이라고 한다. 패딩(Padding)은 주로 출력 데이터의 크기를 조정할 목적으로 사용한다. 합성곱 연산을 반복하다보면 어느 시점부터는 출력 데이터의 크기가 1이 되어 더이상 합성곱 연산을 진행할 수 없게 된다. (위의 이미지로만 봐도 4x4 크기의 입력 데이터가 합성곱 연산을 진행하면서 2x2의 데이터로 출력되는 것을 확인할 수 있다.) 따라서 ***패딩(Padding)을 사용하여 입력데이터의 외각의 크기를 늘려 출력 데이터의 크기가 줄어드는 것을 방지***한다.



Padding 사용 시 합성곱 연산 : 출력 데이터의 크기가 입력 데이터의 크기와 동일

스트라이드 (Stride)

Stride는 입력데이터에 필터를 적용할 때 이동할 간격을 조절하는, 즉 ***필터가 이동할 간격을 뜻한다.*** Stride의 값을 키우면 키울수록 출력 데이터의 크기는 작아지며 Padding은 크게하면 할수록 출력 크기가 커진다.



Padding, Stride를 적용한 합성곱 연산

이렇게 Padding, Stride를 사용하면 출력 크기가 어떻게 되는지에 대한 수식은 다음과 같으며, 출력 크기에 해당하는 OH와 OW는 정수로 나타내져야 한다.

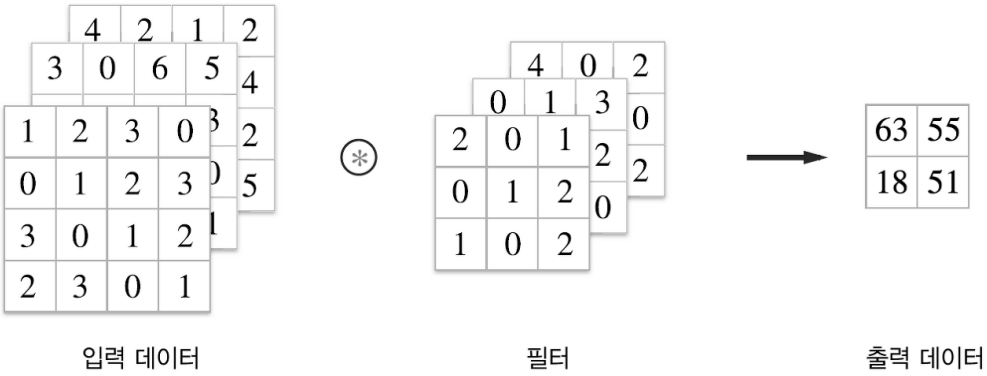
$$(OH, OW) = \left(\frac{H + 2P - FH}{S} + 1\ , \frac{W + 2P - FW}{S} + 1\right)$$

- $(H,\ W)$: 입력크기
- $(FH,\ FW)$: 필터크기
- $(OH,\ OW)$: 출력크기
- P : 패딩
- S : 스트라이드

더보기

3차원 데이터의 합성곱 연산

이미지 데이터는 가로, 세로, 색(RGB) 총 3차원 데이터라고 언급했다. 하지만 위에서 봤던 Conv 계층의 연산은 2차원 데이터로 진행했다. 그렇다면 이미지 데이터와 같은, 혹은 그 이상의 차원의 데이터같은 경우는 어떻게 모든 차원을 살리며 합성곱 연산을 진행할 수 있을까?

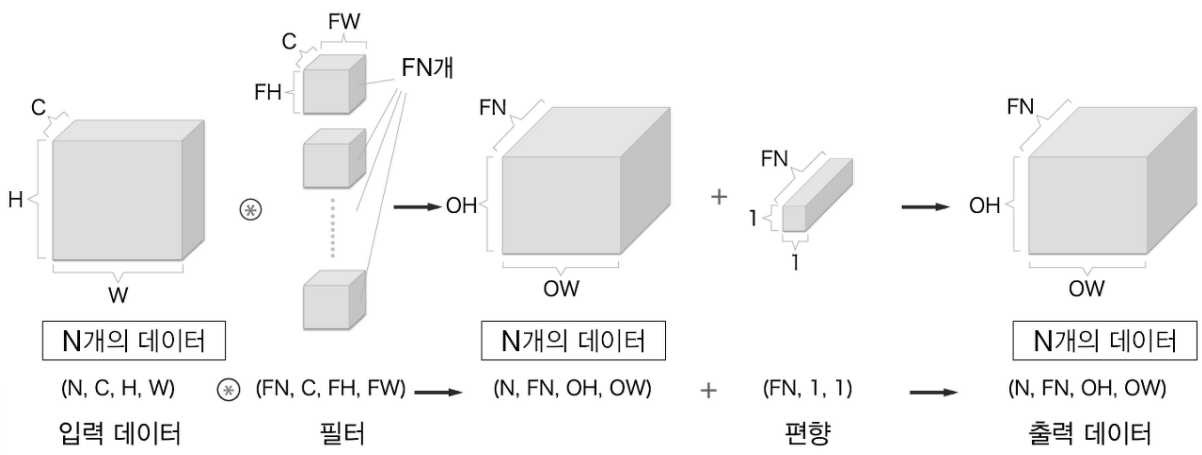


3차원 데이터 합성곱 연산의 예

이에 대한 해답은 위의 그림을 참고하면 된다. ***입력 데이터의 3개의 데이터는 색(RGB) 채널 방향으로 특징 맵이 늘어났다.*** 즉, 각각의 색에 대한 정보를 담은 데이터로 늘어났다고 보면 된다. 그리고 이것을 ***각각의 필터로 합성곱 연산을 진행***하여 출력 데이터로 반환하면 된다. 그러면 ***출력데이터는 2차원의 형태지만 기존 3차원의 데이터를 모두 갖고 있는 데이터로 출력된다.*** 이러한 3차원 합성곱 연산에서 주의할 점은 ***입력 데이터의 채널 수와 필터의 채널 수가 같아야 한다***는 사실이다. 예를 들어 RGB 3개의 채널을 갖고 있다면 필터의 채널 수도 3개여야하고, RB 2개의 채널만 갖고 있다면 필터의 채널 수도 2개여야 한다는 뜻이다.

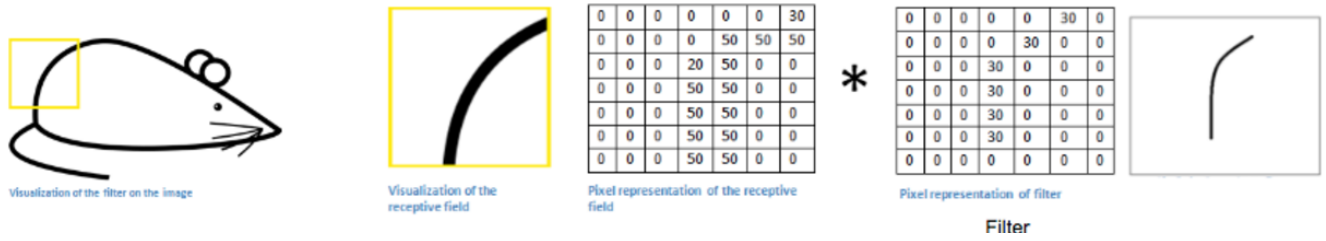
합성곱 연산의 배치 처리

합성곱 연산 시 배치 처리된 데이터를 학습시키는 방법은 기존의 배치와 크게 다르지 않다. 우리가 CNN을 사용하는 이유는 주로 이미지 데이터와 같이 다차원의 데이터를 최대한 기존 데이터의 특징을 모두 학습시키기 위함인데, 배치를 사용하면 3차원이었던 데이터가 4차원으로 늘어나면 된다. 즉, 맨 앞에 배치 사이즈(N)를 추가하여 데이터를 학습시키면 된다.



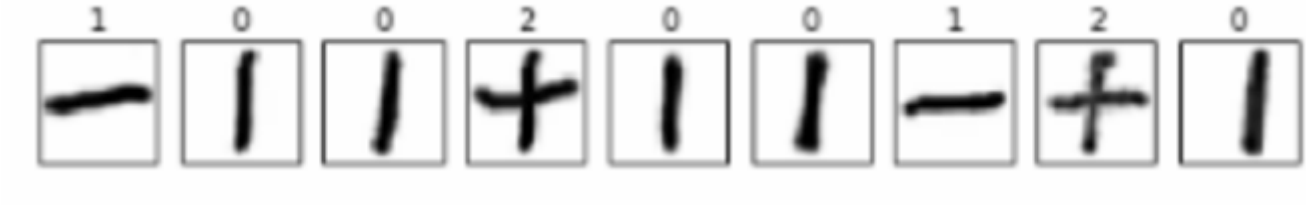
합성곱 연산 처리

위의 이미지는 CNN의 연산 처리 과정으로 지금부터 하나하나 뜯어 설명해보도록 하겠다. 입력 데이터는 배치된 데이터로 N개의 데이터로 구성되어져있어 채널, 가로, 세로로 구성되어져 있는 N개의 데이터 묶음(배치)이다. ***그렇다면 필터의 FN은 정확히 무엇을 의미하는걸까? 필터의 FN은 입력 데이터의 배치 사이즈와는 정확히 다르다.*** 이해를 돕기 위해 다시 한번 맨 위의 필터(Filter) 설명에 사용했던 쥐의 이미지를 가져 오겠다.



1개의 필터에 대한 합성곱 연산

이 그림에 대한 필터는 곡선으로, 오직 1개 뿐이다. 그러면 출력 데이터는 특징 1개에 대한 정보만을 갖고 있는 데이터로 출력된다. 그렇다면 필터의 개수를 FN개로 늘려보면 어떨까?



FN개의 필터들

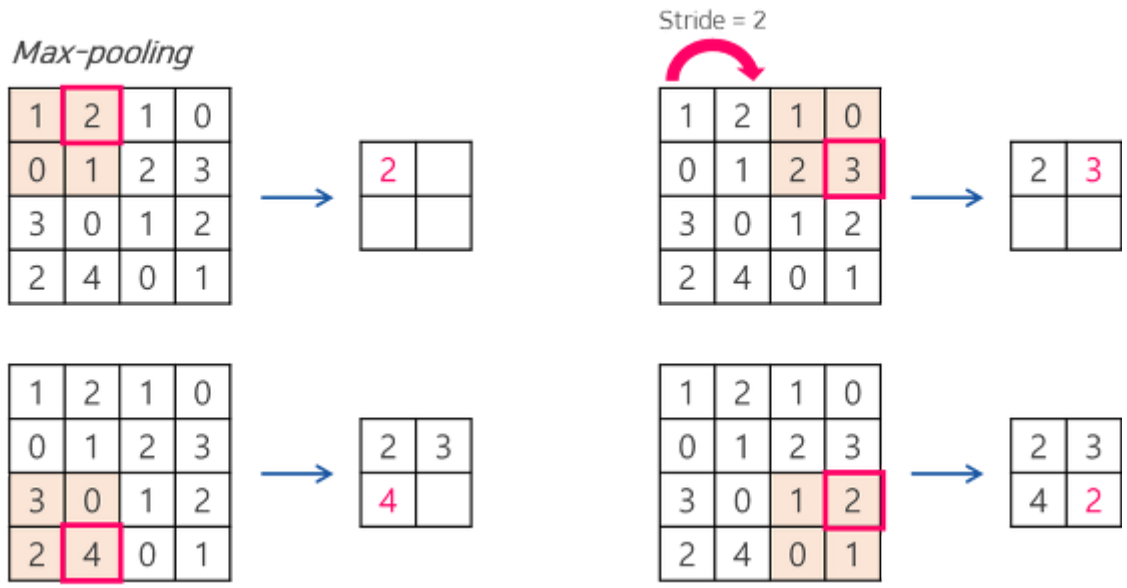
FN개의 필터들로 입력 데이터를 학습시키면 각각의 특징에 대한 출력 데이터를 FN개 얻을 수 있다. 이것들의 묶음을 블록 단위로 하여 다음 계층에 넘기는 것이 필터 (FN, C, FH, FW)의 의미이다. 입력 데이터와 FN개의 필터들과의 합성곱 연산으로 출력된 데이터 (FN, OH, OW)는 필터 개수만큼의 편향 (FN, 1, 1) 과의 연산을 통해 (FN, OH, OW)의 데이터로 출력될 수 있다. 여기서 ***초기 입력 데이터의 C(채널)가 출력 데이터들에 없는 이유는 합성곱을 진행하면서 C에 대한 데이터가 OH, OW에 모두 스며들었기 때문***이다.

풀링(Pooling) 계층

풀링은 ***세로, 가로 방향의 공간을 줄이는 연산***이다.

풀링의 종류로는 크게 max pooling과 average pooling이 있는데,

max pooling은 이미지를 선명하게 만드는 역할을, average pooling은 이미지를 부드럽게 만드는 역할을 한다.



Max-pooling을 사용하는 경우

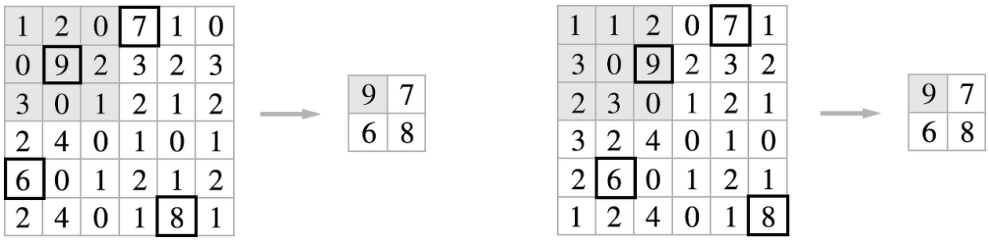
Pooling은 주로 Conv 계층에서 출력데이터의 크기는 입력 데이터의 크기 그대로 유지하고, 풀링 계층에서 크기를 조절한다.

그렇다면 풀링 계층의 특징은 무엇일까?

- 학습해야 할 매개변수가 없다.

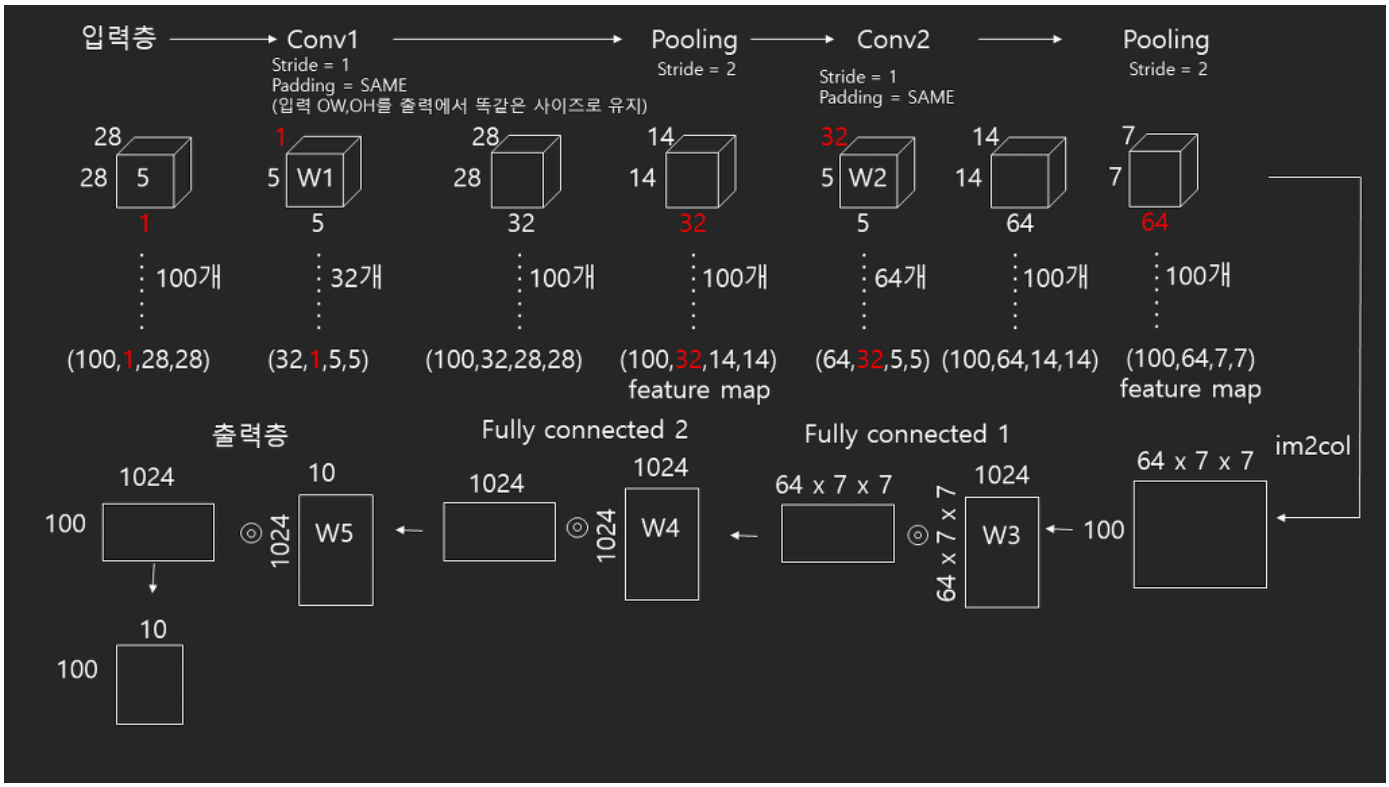
- 채널 수가 변하지 않는다.
- 입력의 변화에 영향을 적게 받는다.

여기서 맨 마지막 입력의 변화에 영향을 적게 받는 이유는 아래의 이미지를 참고하면 이해하기 쉽다.

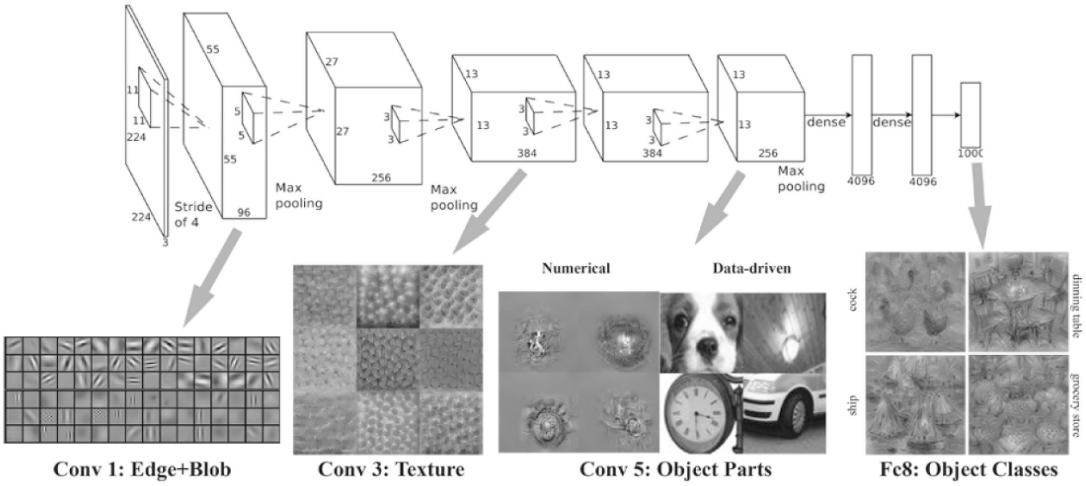


출력 데이터가 오른쪽으로 하나씩 밀려도 풀링 결과는 달라지지 않는다.

또한 Pooling은 ***parameter를 줄이기 때문에 신경망의 표현력이 줄어들어 Overfitting을 억제하고 연산을 비교적 간단하게 만들어준다.***



CNN 구현 흐름 시각화 / 원하는 만큼 Conv, Pooling 계층 진행 후 완전연결계층 진행



층 깊이에 따른 특징 추출 변화

딥러닝의 흥미로운 점은 합성곱 계층이 여러 개 쌓이면서 ***층이 깊어질수록 더 복잡하고 추상화된 정보가 추출된다***는 것이다. 위의 이미지를 보면 Conv1층은 단순하게 Edge에만 반응하지만 Conv3층은 텍스쳐에 반응하고 Conv5는 사물의 일부에 반응하도록 변화된다. 즉, 층이 깊어지면서 뉴런이 반응하는 대상이 단순한 모양에서 고급 정보로 변화해 간다. 따라서 ***합성곱 계층이 깊어지면 깊어질수록 신경망은 사물의 의미를 이해하게 된다.***

'인공지능 > 인공지능 이론' 카테고리의 다른 글

23. im2col 이해하기 - 합성곱 계층(Convolution Layer) (0)	2020.08.12
22. 딥러닝, 전이학습, 강화학습 (0)	2020.08.10
21. CNN (합성곱 신경망) (2)	2020.08.07
20. 배치 정규화 (Batch Normalization), 과적합 (Overfitting) (0)	2020.08.06
19. 가중치 초깃값 (0)	2020.08.06
18. 다양한 최적화 알고리즘 (0)	2020.08.06

SECRET ☐ WRITE

leebywan 2020.11.26 02:40 신고

도움되는 내용 되게 잘 보고 가요

Delete Reply

공부하는 정듀이 2020.11.26 11:06 신고

감사합니다! 블로그한지 1년이 넘어가는데 처음 댓글 받아보네요ㅋㅋㅋ

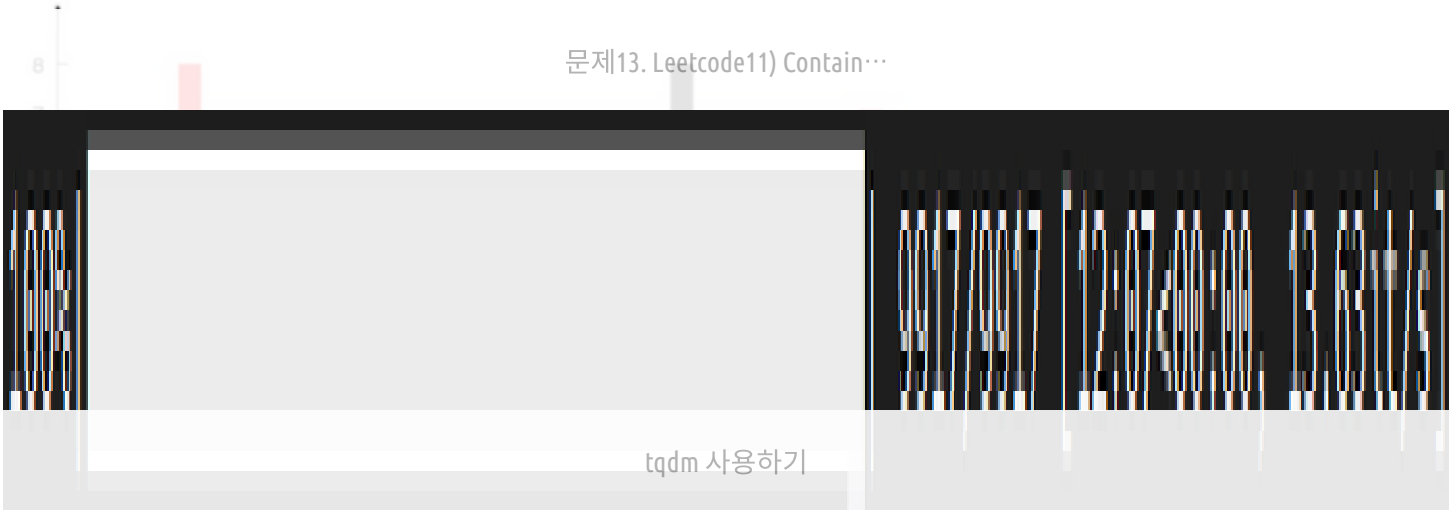
Delete

+ Recent posts

Given `n` non-negative integers $a_1, a_2, \ldots, a_n$, where each represents a point at coordinate (i, a_i). `n` vertical lines are drawn such that the two endpoints of the line `i` is at (i, a_i) and $(i, 0)$. Find two lines, which, together with the x-axis forms a container, such that the container contains the most water.

Notice that you may not slant the container.

Example 1:

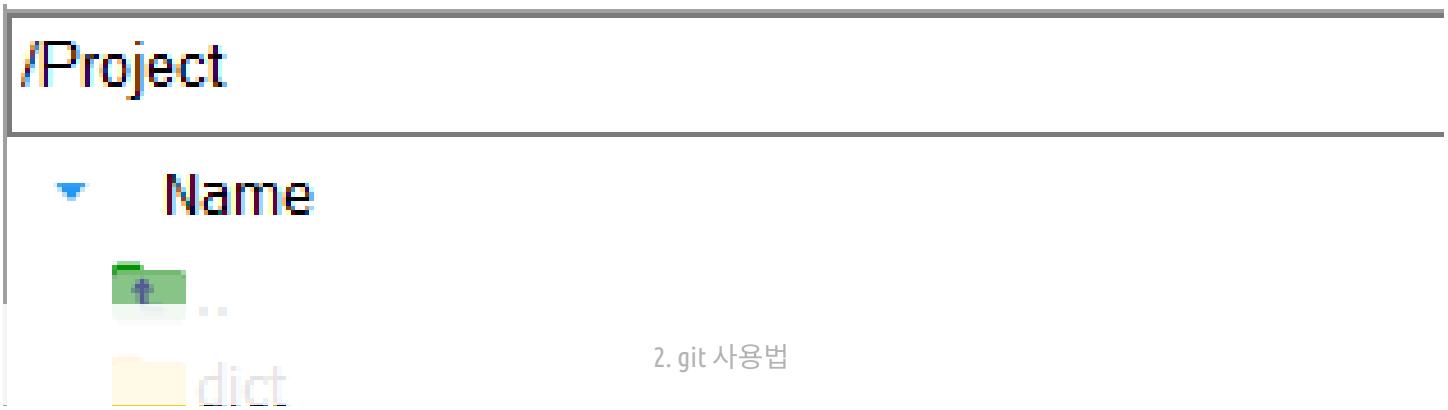


I found this simple solution, PyMuPDF, output to png file. Note the library is imported as "fitz", a historical name for the rendering engine it uses.

```
import fitz

pdffile = "infile.pdf"
doc = fitz.open(pdffile)
page = doc.loadPage(0)  # number of page
pix = page.getPixmap()
```

PyMuPDF : pdf2image



2. git 사용법