

Convolutional Neural Networks

Fully connected layer

모든 차원을 펼쳐 하나의 긴 벡터로 나타냄

⇒ 층마다 모든 뉴런이 서로 연결

- ex. $32 \times 32 \times 3$ image → stretch to 3072×1
: input(1×3072) → weights
 $Wx(10 \times 3072) \rightarrow$ activation (1×10)
 - Wx 의 결과로 하나의 숫자 산출

Convolution Layer

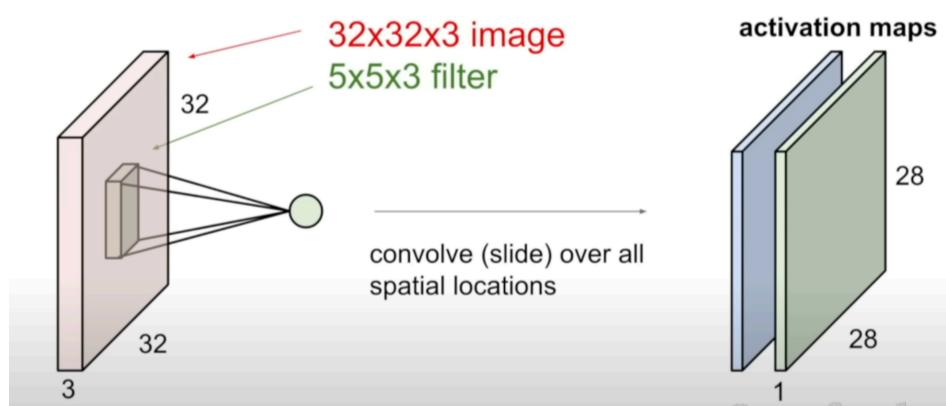
이미지 구조 유지

→ 작은 사이즈의 filter(w)를 사용해서 image 위를 슬라이드하며 모든 공간과의 dot product를 계산하는 방식으로 작동

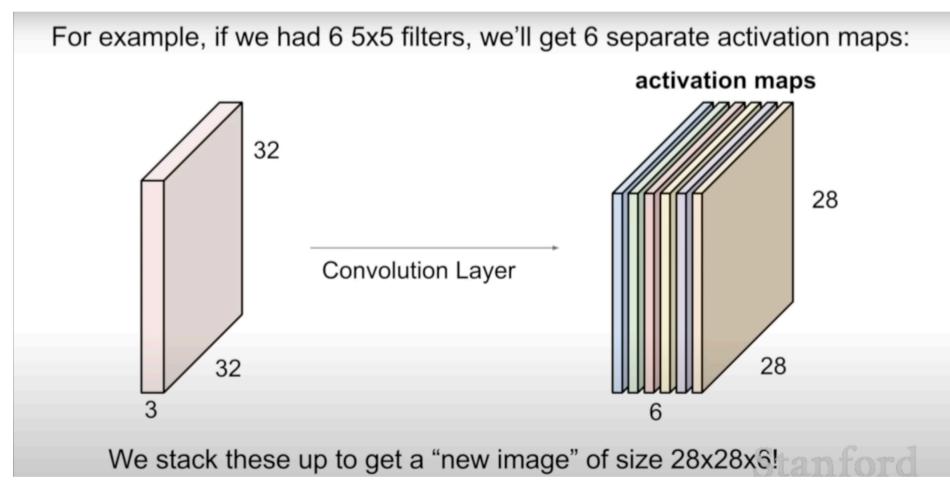
⇒ 각 filter은 input의 small region에 연결되고 모두 같은 region을 보지만, 서로 다른 패턴을 찾음

Convolution Layer 작동 방식

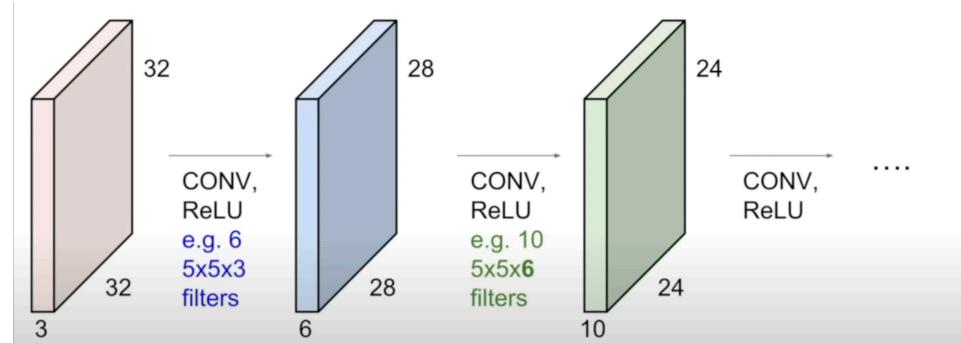
- 각 $n \times n$ 사이즈의 image, filter(w)의 dot product 결과마다 하나의 숫자 산출
→
 $\text{depth} * n^2 + \text{bias} \Rightarrow w^T x + b$
 - +) image depth = filter depth
- image의 모든 위치에 슬라이딩하며 모든 연산이 끝나면 activation maps 생성



위와 같이 이미지의 모든 위치에서 연산이 더해져 activation maps 생성



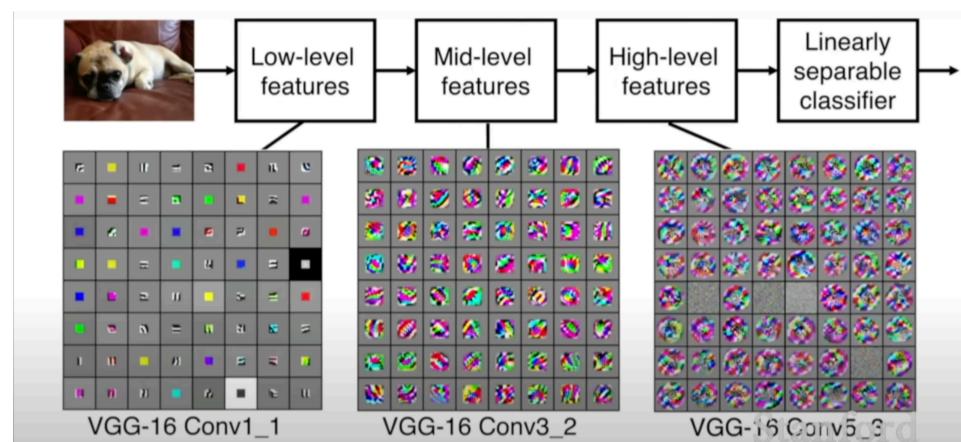
ex. 5*5의 필터가 6개 있을 때



그리고 이 연산은 위와 같이 (Conv + Activation function + @)을 하나의 block으로 특정 횟수만큼 반복됨

Convolution layer 층에 따른 특성 차이

Convolution layer이 초기 단계일 때는 단순한 특성 추출, 층이 깊어질수록 복잡한 특성 추출 가능



그림과 같이 Low-level features은 단순한 패턴을 띠지만, High-level에 가까워질수록 더 복잡한 양상의 특성 확인 가능
→ 아래 visualization은 각 뉴런을 가장 크게 활성화하는 이미지 패턴 (모서리, 색, ...)

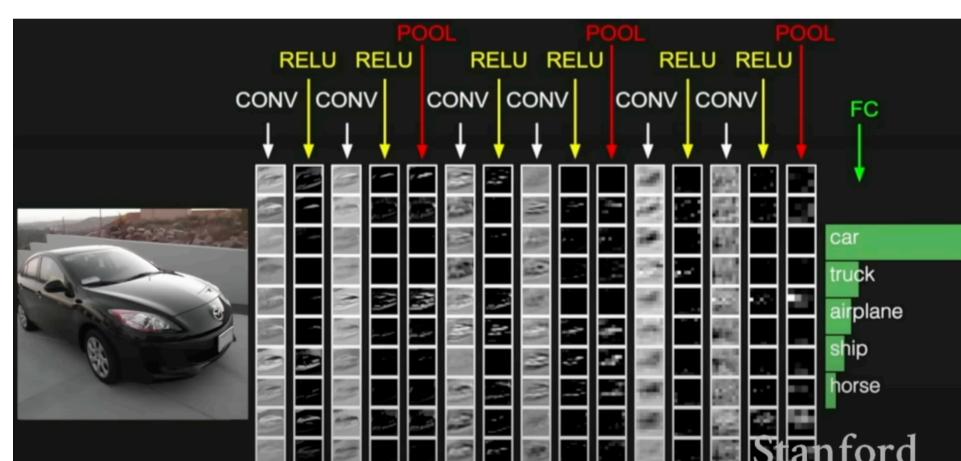
Convolution layer인 이유

두 신호의 합성곱과 연관되어있기 때문

- filter와 신호(image) 합의 elementwise multiplication

$$f|x, y| * g|x, y| = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f|n_1, n_2| \cdot g|x - n_1, y - n_2|$$

Convolution network 과정



위에서 언급했듯이 (Conv + activation 층)을 기본적으로 하나의 block으로 함. 이 block과 이후에 언급할 pooling 층과 결합해 정보를 압축하는 과정을 반복. 이후 이를 기반으로 FC층을 통해 주어진 Task를 진행

Convolution hyperparameter

Stride

: image sliding 간격

- Default: 1
- output size $\Rightarrow (N - F)/\text{stride} + 1$

Padding

: (image size 유지를 위해) 모서리에 임의의 숫자(0)를 넣은 칸을 추가하는 기법

- Default: 2

Filter size

: 필터 크기 $F \times F$

input image size를 유지하는 이유

: 층을 거듭할수록 이미지 크기가 빠르게 줄어듦

- 네트워크 층 깊게 설정 불가
- image corner에 있는 정보 유실

문제

input volume이 $32 \times 32 \times 3$, 5×5 필터가 10개, stride=1, pad=2일 때,

- Q1. Output volume size는?
A. $(32(H) + 2 * 2(\text{pad}) - 5(F)) / \text{stride} + 1 = 32$ 이므로 $\Rightarrow 32 \times 32 \times 10$
- Q2. 이 layer의 parameter 개수는?
A. $5(F) * 5(F) * 3(C=\text{Depth}) + 1(\text{bias term}) = 760$ 이므로 $\Rightarrow 760$ 개

Convolution layer summary

Input volume $\Rightarrow W_1 \times H_1 \times D_1$

Hyperparameter

- filter 개수 $\Rightarrow K$
- filter 크기 $\Rightarrow F$
- stride $\Rightarrow S$
- padding 개수 $\Rightarrow P$

Output volume $\Rightarrow W_2 \times H_2 \times D_2$

- $W_2 = (W_1 - F + 2P) / S + 1$
- $H_2 = (H_1 - F + 2P) / S + 1$
- $D_2 = K$

Parameter

- 한 filter의 parameter 개수 $\Rightarrow F \cdot F \cdot D_1$
- 전체 filter의 parameter 개수 $\Rightarrow (F \cdot F \cdot D_1) \cdot K$

Output volume에서 d -th depth slice는 input volume에

1. d -th filter를 stride 적용한 합성곱으로 계산하고
2. d -th bias를 더한 결과

```

module = nn.SpatialConvolution(nInputPlane, nOutputPlane, kW, kH, [dW], [dH], [padW], [padH])

# nInputPlane: forward()로 입력할 input 평면의 수
# nOutputPlane: conv layer의 결과로 나올 output 평면의 수
# kW: conv의 커널 width
# kH: conv의 커널 height
# dW: width 차원에서 conv의 step(간격) = stride. Default = 1
# dH: height 차원에서 conv의 step(간격) = stride. Default = 1
# padW: input plane에 width마다 추가할 zero pad의 수. Default = 0. 최적은 (kW-1)/2
# padH: input plane에 height마다 추가할 zero pad의 수. Default = padW. 최적은 (kH-1)/2

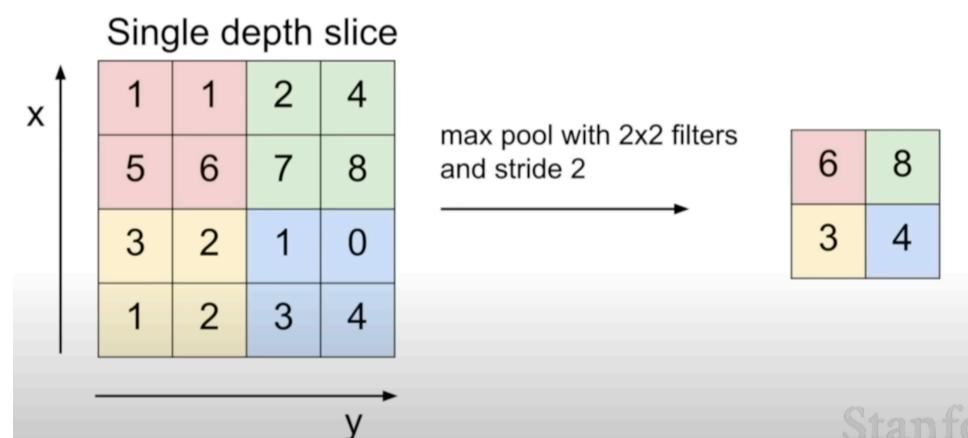
```

Pooling Layer

- : Representation을 더 작고 다루기 쉽게 만들 \Rightarrow downsampling
- \rightarrow 각 activation map에 대해서 독립적으로 계산
- \rightarrow 대체로 overlap하지 않는 방식으로 input size(H, W) 축소
- \rightarrow 최근에는 Pooling층이 아닌 stride를 통해 downsampling을 하는 방식을 채용하기도 함 + 서로 장단점이 존재

Max pooling

- : 가장 큰 값으로 요약
- \rightarrow 우리가 image에서 찾고자하는 패턴은 높은 값에 위치해있는 경우가 많기에 가장 흔하게 사용



Pooling layer summary

Input volume $\Rightarrow W_1 \times H_1 \times D_1$

Hyperparameter

- filter 크기 $\Rightarrow F$
- stride $\Rightarrow S$

Output volume $\Rightarrow W_2 \times H_2 \times D_2$

- $W_2 = (W_1 - F)/S + 1$
- $H_2 = (H_1 - F)/S + 1$
- $D_2 = D_1$

+) Pooling layer에 pad를 적용하는 것은 흔치않음

\Rightarrow pooling layer에서는 overlap을 지양하기에 image corner의 정보 소실 문제가 없기 때문

Fully connected layer in Conv

: Conv층을 모두 지난 후, $W \times H \times D$ 꼴의 행렬을 flatten해서 1-D input으로 전환

→ 마지막으로 FC layer를 통해 conv map outputs을 모두 결합, 연결해 score 산출