

④ Resnet (Residual Network)

1. 사용: 딥러닝에서 사용하는 신경망 구조. 이미지 분류 / 다른 컴퓨터 vision 작업에서 효과적으로 사용. (residual connection: 잔여 연결을 사용함)

a. 일반적인 딥러닝 모델은 여러 층 쌓아 복잡한 특징 학습.

∴ 층이 많을수록 기울기 소실 발생. \Rightarrow network 학습 잘못 함.

7. skip connection: 층을 쌓는 대신 입력을 직접 다음 층에 함.

$y = f(x) + x$ 각 층이 입력을 직접 넘겨줌. 네트워크가 깊어져도 학습 빠르.

i) 깊은 network에서는 역전파 과정에서 기울기가 0에 가까워짐.

ii) 표현력의 저하 발생

④ motion prediction에서 skip connection 사용하면

① 차량의 과거 위치정보 전달 빠르 ② network가 깊어져도 학습 안정적

③ 정확한 미래경로 예측 ④ CNN 기반 모델에서 과거 프레임 정보 유지하여 features 학습 빠르.

2. Residual Block



Plain layers

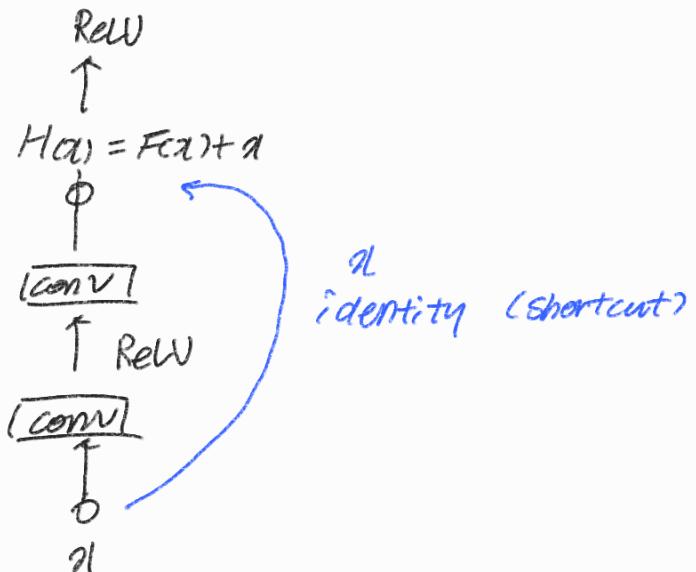
input x 만더함

$$y = f(x)$$

y (output)는 기울기를 새롭게 학습하는 정보.

∴ 기존 학습 정보 보포Ex. 변형시켜 새롭게 생성.

층 깊어질수록 한번에 학습할 mapping 증가.



Residual Block

input x 더함

$$y = f(x) + x$$

x가 그대로 보존된 y

∴ 기존 학습 정보 보포 + 추가적 학습정보

output 이전에 layer에서 학습한 정보 연결. 추가 학습 정보만 mapping

- a. layer depth가 깊어질수록 (학습이 많을수록) 차는 점점 증가함 $H(x)$ 에 근접.
 추가 학습량 $F(x)$ 가 점점 작아지며 0에 근접하는 최고값으로 수렴.
 $\therefore H(x) = F(x) + g$ 에서 추가 학습량에 해당하는 $F(x) = H(x) - g$ 가
 최소값 0 이 되도록 학습

b. Resnet 구조 : VGG-19 대체 기본틀.

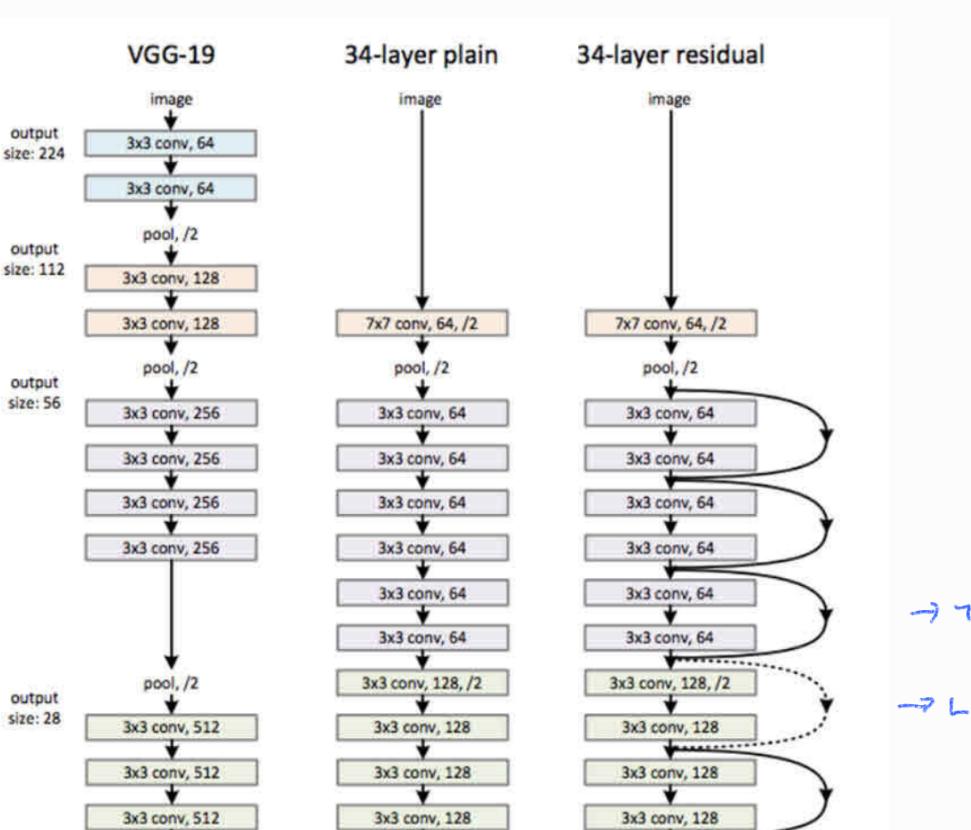
a. VGG-19 : 여러개의 3×3 합성곱 (convolution) 쌓고 중간에 pooling 씀.

ex) conv → conv → pooling 하여 깊어짐.

깊어지면 계산량↑ 학습 어려움(기울기 소실 문제)

b. Residual connection (잔차연결)

ex) conv → conv 쌓고 skip connection 추가로 입력을 다음 블록에 직접 더함.



ㄱ. 입출력의 차원이 같을 때. skip connection 그대로 사용

$$H(x) = F(x) + g$$

같은 크기의 층계 (3x3 convolution)

ㄴ. 입출력의 차원이 다를 때. 1x1 convolution 사용 → 차원 맞추고 더함.

$$\leftarrow H(x) = F(x) + W_S \times g \text{ 형태. } \text{ 1x1 conv } \text{ 사용 후 더함.}$$

마운Pooling (Stride=2) 같은 연산 들어가면 정선포함.

c. 다운샘플링 : 특성 맵의 사이즈가 반으로 줄어들어. (Stride=2)

ㄱ. Pooling : ex) 2x2 Max Pooling → 가로, 세로 2가지 계산.

ㄴ. Stride=2 : ex) 3x3 Convolution, Stride=2. → 가로, 세로 2가지 계산.

projection
shortcut
path.

CNN에서 특성 맵의 크기를 줄이면 연산량 감소 → 더 큰(넓은) 영역을 한번에 봄.

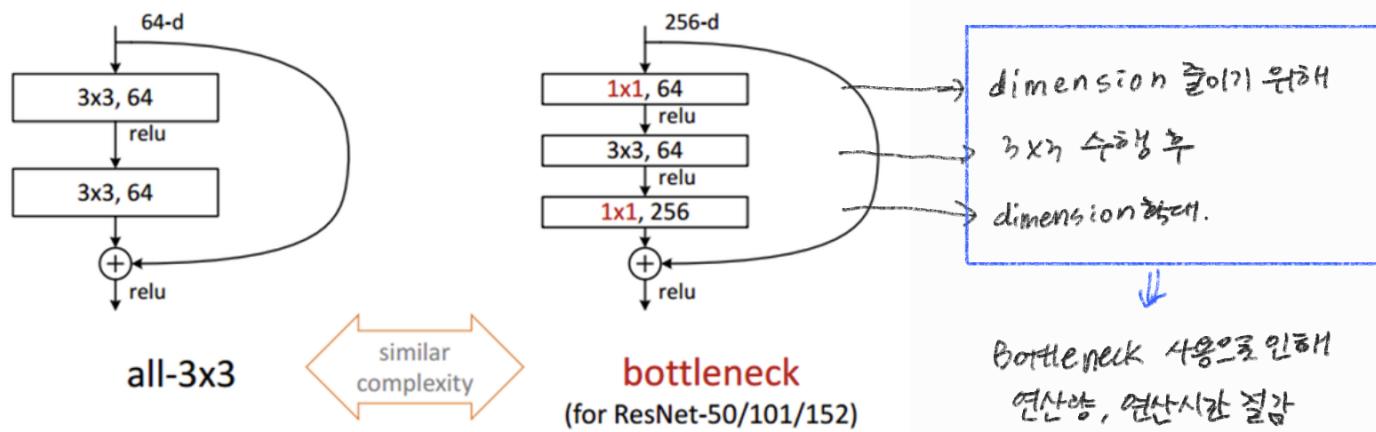
d. 특성맵의 depth를 2배로 늘림 (채널 수 증가)

ex) $(64, 64, 64) \rightarrow (32, 32, 128)$
기존, 세로 채널수
다운샘플링

∴ CNN에서 특성맵의 크기를 줄이면 채널수(depth)가 증가함. 하지만 이 과정에서 정보가 손실 될 수 있으므로 더 많은 채널에서 풍부한 특징 학습 유도.
깊은 신경망에서 추상적 고수준 특징 학습 필요 → 채널 수 증가로 더 많은 정보 취득.

⊕ Convolution parameters = kernel size(기존) + kernel size(세로) + Input channel 수 × Output channel 수

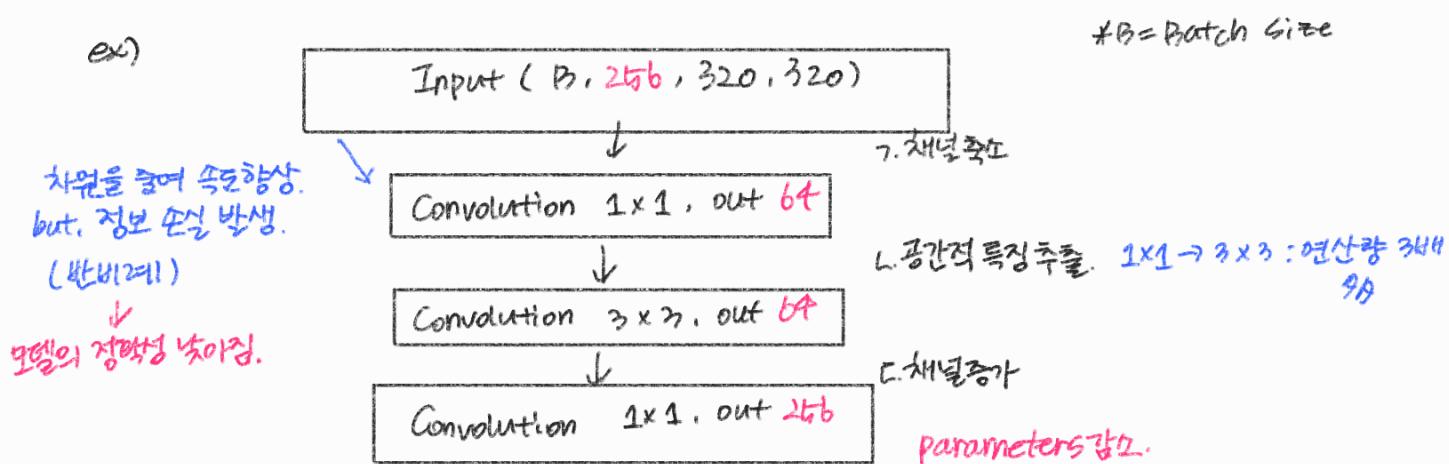
f. Bottleneck



a. 기본 구조 : 1*1 convolution. (= pointwise convolution)

= $1 \times 1 \times \text{Input channel} \times \text{output channel}$.
Feature map(output channel)을 개우거나 줄일 때 사용

ex)



ㄱ. 채널축소: 연산량 감소 목적

L. 공간적 특징추출(feature map): 1x1 convolution에서는 spatial feature X. Kernel $>= 2$.

ㄷ. 채널증가: CNN은 feature map의 특성이 많을수록 학습이 잘됨.