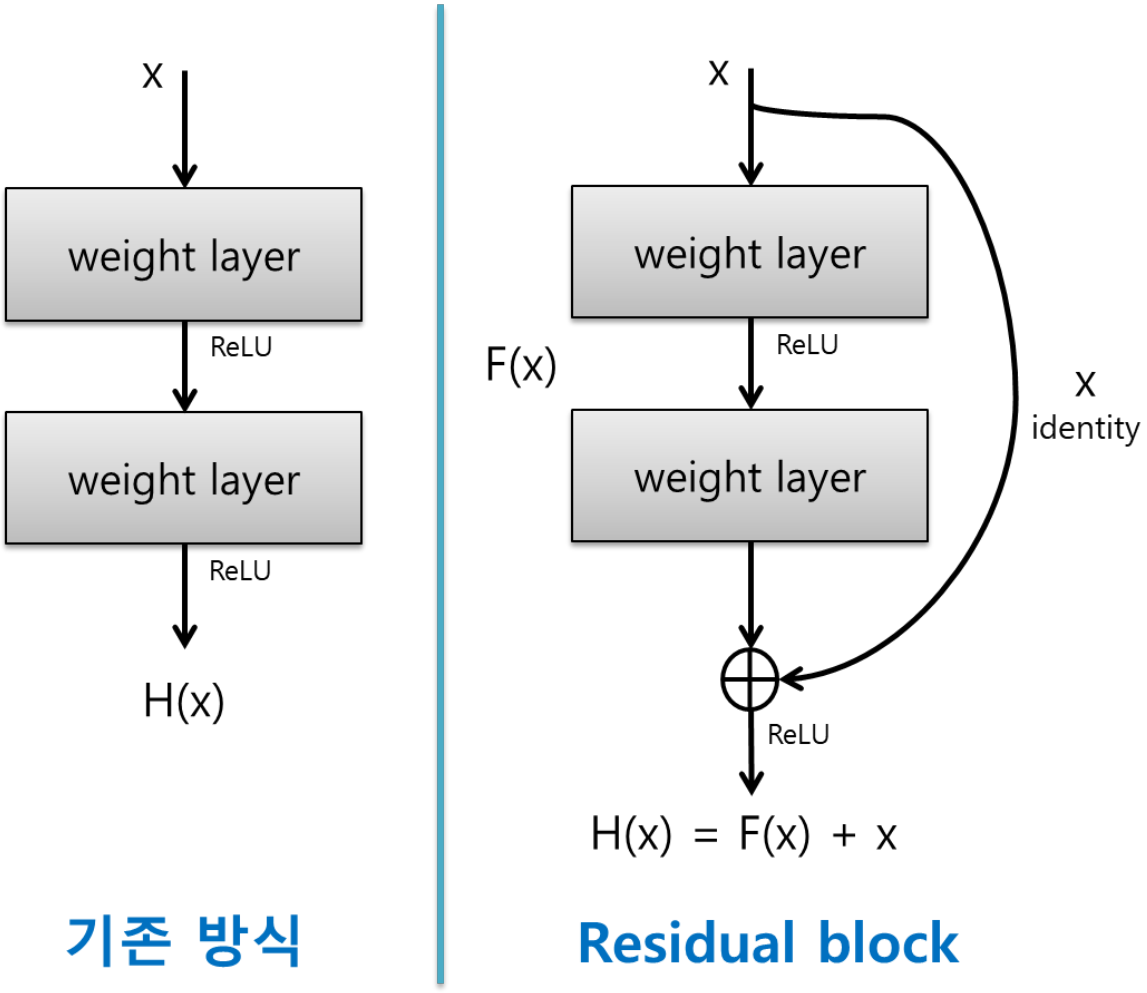
ResNet

네트워크의 깊이가 깊어져도 vanishing gradient가 발생하지 않도록 하는 기법

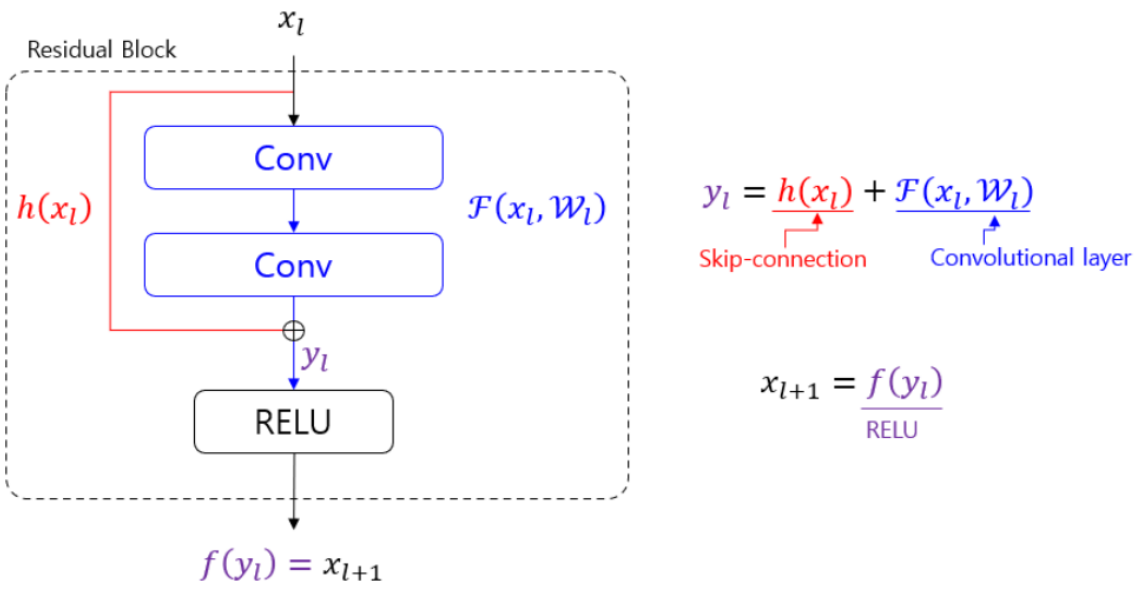
Residual block 방식



기존 방식에서는 y=f(x) 꼴로 direct로 학습하는 반면 resiual block에서는 H(x) = f(x) +x 꼴로 학습하여서

H(x) = f(x) + x 에서 추가 학습량에 해당하는 f(x) = H(x) - x 가 최소가 되도록 학습이 진행된다

→ 이전에 학습했던 내용이 저장되고 해당 layer에서 학습해야 하는 부분만 추가적으로 학습하게됨



장점

1. Residual block

Loss function의 gradient의 계산식은 backward propagation chain rule로 부터 아래 식으로 나타낼 수 있다.

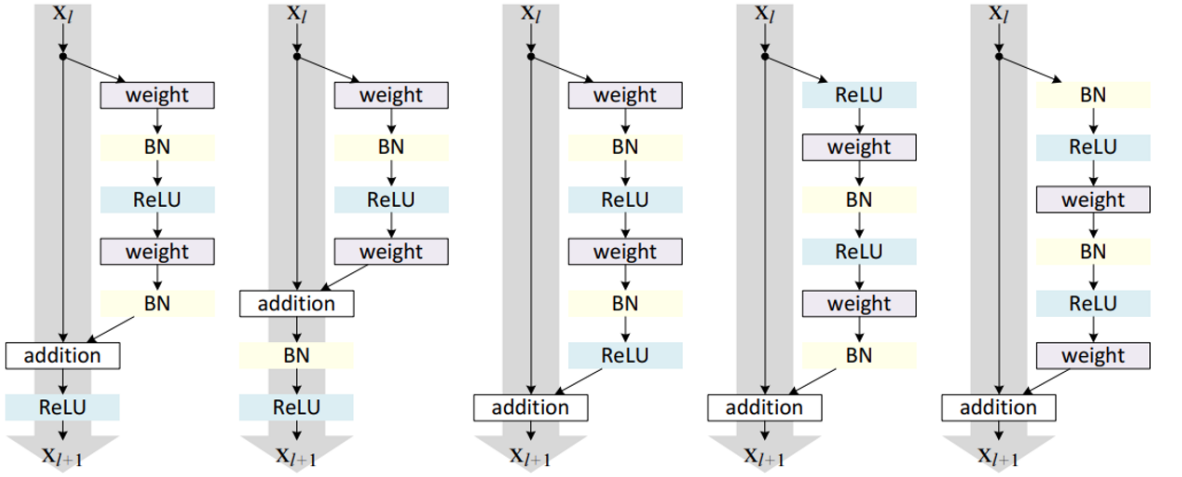
$$\frac{\partial \varepsilon}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i, W_i)\right)$$

추가 계산되는 Convolutional layer가 모든 mini-batch에서 -1이 되는 경우는 매우 희박하기 때문에 총 합인 e / xl = 0 에 가깝게 되는 vanishing gradient가 발생하지 않음

2. Skip-connection

H(x) = f(x) + x 방식으로 어떠한 조작을 가하지 않고 원래 값을 전해줌으로써 정보가 소실되거나 증폭되는 경우를 방지함

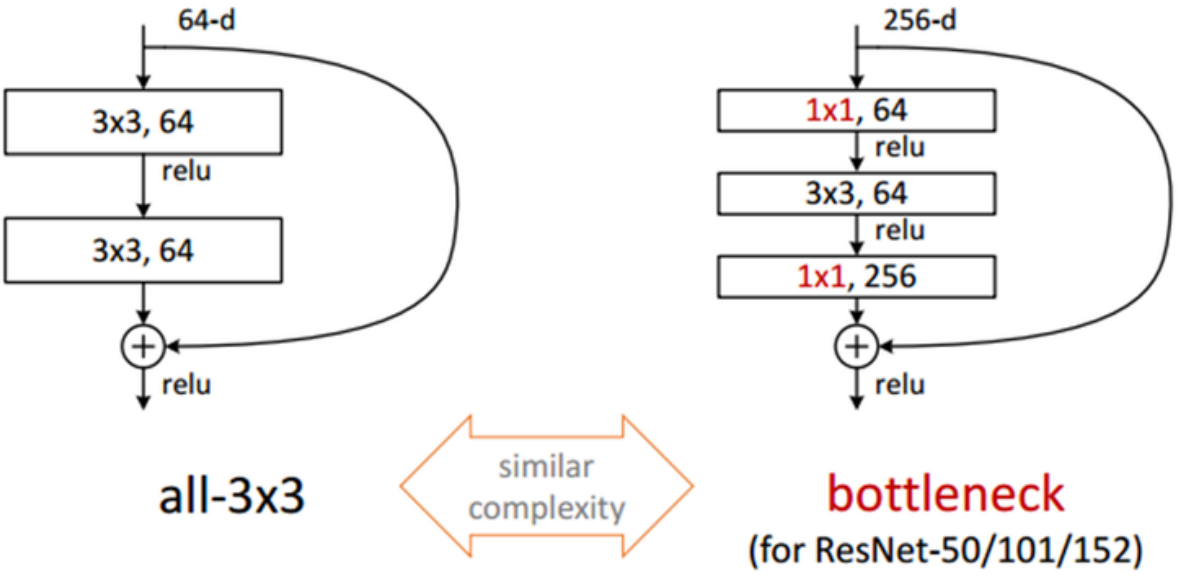
3. Pre-activation



기존 값 X를 f(x)에 더 하는 addition 연산을 모든 과정이 끝난 후에 실행하는 것이 오차율이 가장 낮음

이유는 모름 그냥 해봤을 때 그렇게 나왔대

4. bottle-neck



그냥 연산하는 왼쪽의 경우 3 * 3 * 64 * 2 = 1152

bottleneck 방식의 오른 쪽의 경우 1 * 1 * 64 + 3 * 3 * 64 + 1 * 1 * 256 = 896

연산 속도에서 차이를 보임 (속도 ↔ 정확도 trade off 관계)

convolution 차원이 1*1의 경우 3*3 보다 9배 빠름