✳ 케이스 스터디 ①

why use case study?

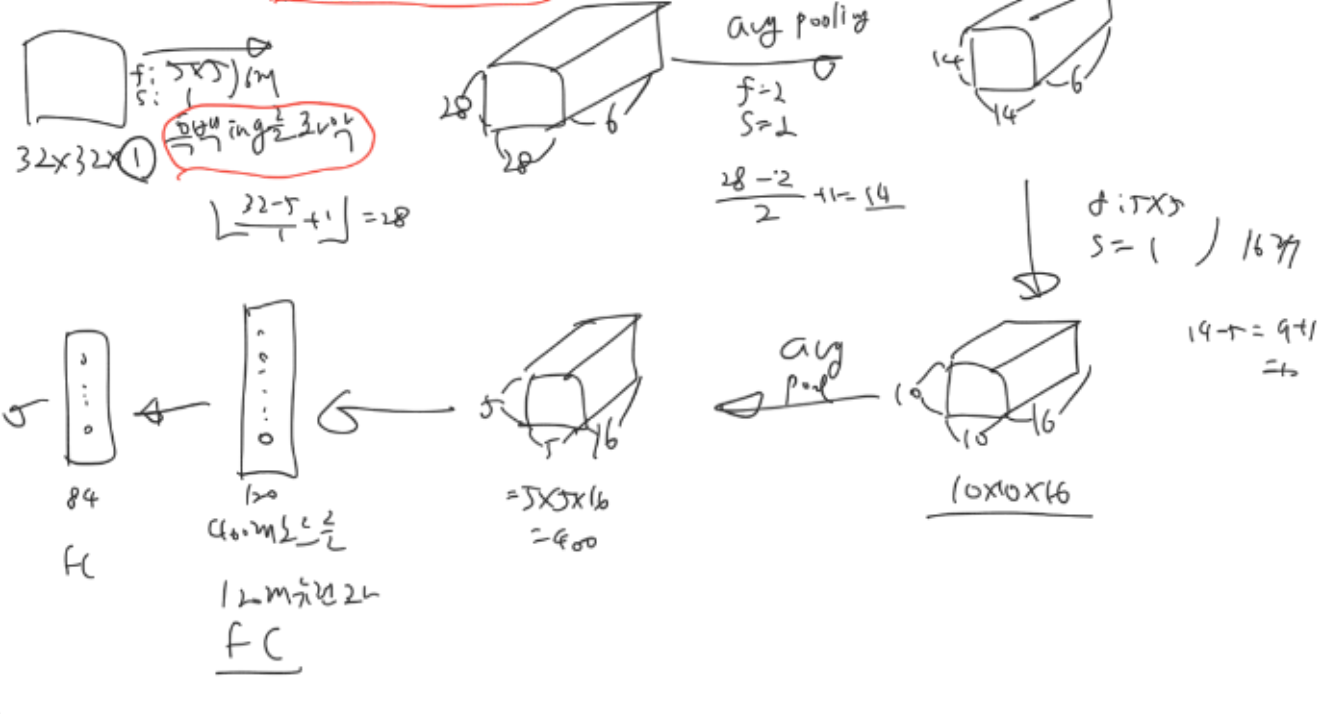
classic network
- LeNet-5
- AlexNet
- VGG

ResNet → more deep → 152개 layer train

Inception → 효과적으로 신경망을 구축하는 방법을 알려줌

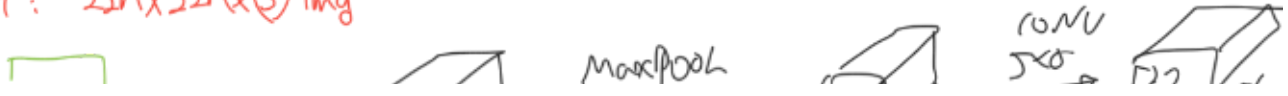
② 고전적인 네트워크 (Classic Network)

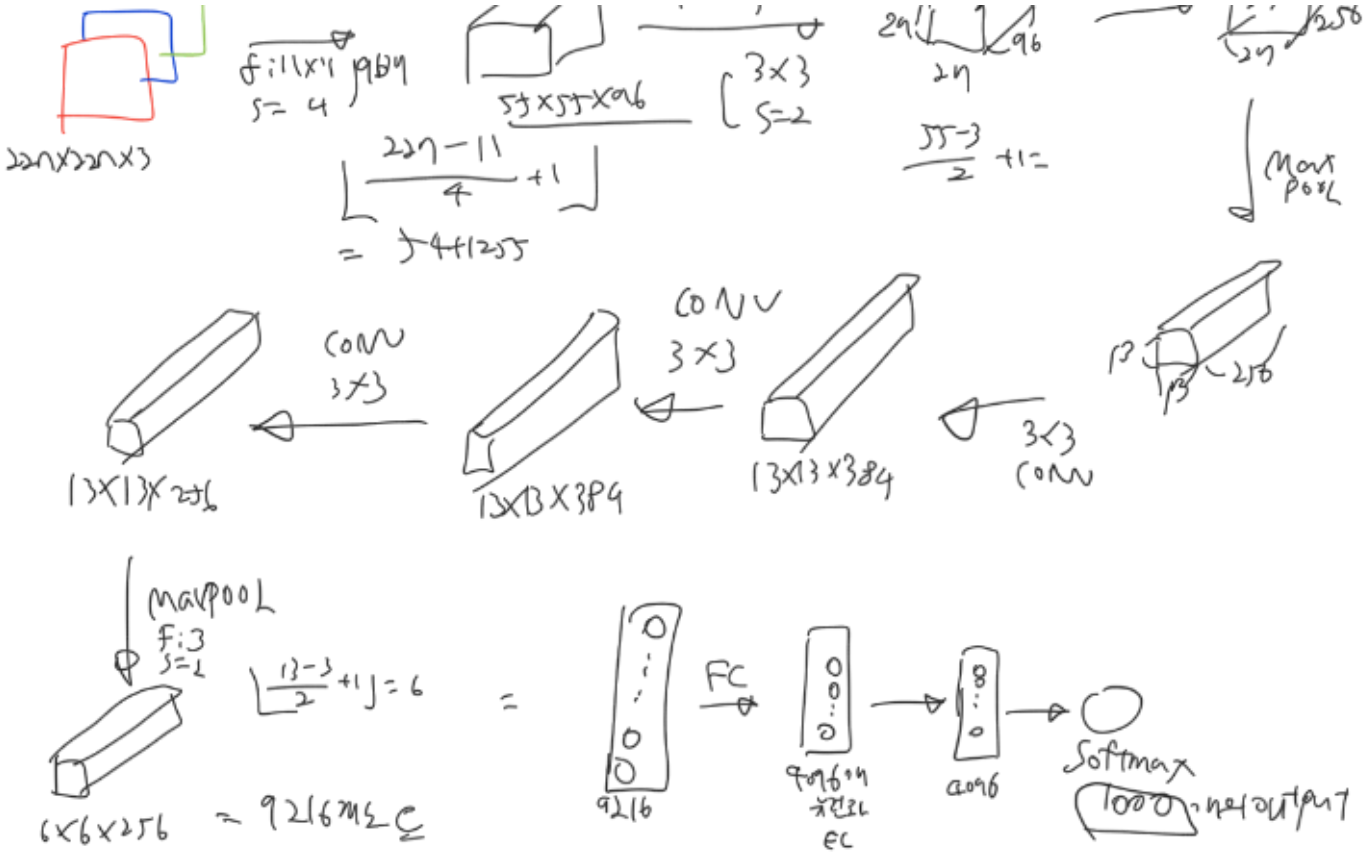
✳ LeNet-5 (목적 : 손글씨 판독)



③ AlexNet

input : 227×227×3 img (RGB)





=> 알렉스넷은 LeNet과 매우 유사하나 훨씬 큰 크기를 가진다.

#C: 3

(흑백) 채널 1개

LeNet은 6만개 정도 매개변수를 가지지만
AlexNet은 6천만개 ,, 를 가진다. => 더 많은 hidden unit과 더 많은 data를 통해 train 하기때문에 훨씬 더 좋은 성능을 보인다.

AlexNet이

AlexNet이 LeNet과 구별되는 특징은

- ReLu Activation 함수를 사용한다

$= \begin{cases} 0 & (x<0) \\ x & (x \geq 0) \end{cases}$ 로 다음 layer로 그래도 넘겨 vanishing gradient를 방지하기 위함

당시는 GPU가 느려서 2개 CPU를 돌려서 사용했다.

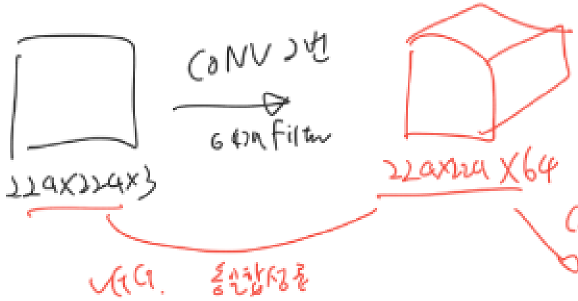
즉, Layer들을 2개로 나눠서 각 GPU에 담고 2개 GPU가 소통하는 형식으로 이루어짐

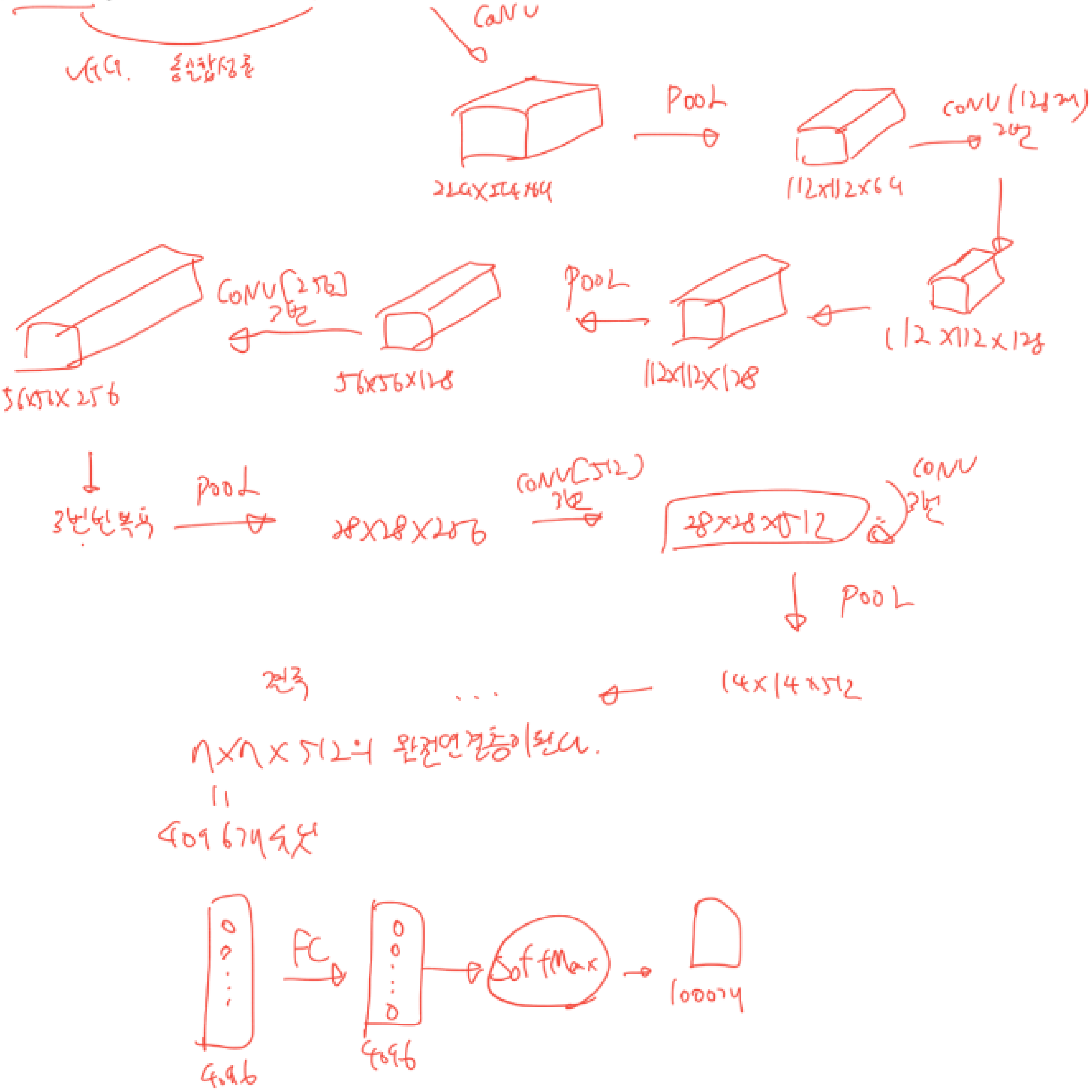
AlexNet에는 Local Response Normalization이 포함됨.

(3) VGG-16 특징: 많은 hyper parameter를 가지는 대신 CONV에서

(학습 X 고정변수 하지만 과정에는 학습가능)

간단 | stride가 1인 3×3 filter만 사용해 동일 합성곱을 하고 (output 크기 동일) max pool에서는 f: 2×2, s: 2로 고정





※ VGG-16 16의 의미는 16개의 가중치를 가지는 층이 존재한다는 뜻.

ㄴ 1억 3천만개 변수를 갖는 매우 큰 Network

2배씩 CONV filter 개수 늘리는 게 특징

VGG 단점 (training 할 변수가 많아져서 네트워크 크기가 커진다.)

특징: layer 층이 깊어질수록 n_H와 n_W는 절반으로 줄어드는데 #channel은 2배씩 증가
=> 체계적이다.

* Lenet은 변수의수가 상대적 적다. Pool 뒤에 비선형함수를 적용한지 오래되지 않았다 + 비선형함수로 Relu가 아닌 Sigmoid를 사용

* AlexNet은 이미지를 1000개의 클래스로 분류
=> Lenet에 비해 많은 변수를 갖고 Relu를 사용
+ 동일합성곱을 이용하여 input과 output의 n_H, n_W를 동일하게 함

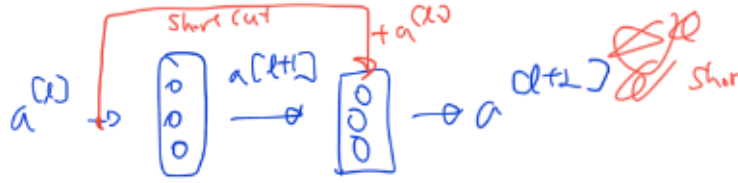
* VGG-16 : 간편한 구조 { CONV층은 filter 3×3에 동일합성곱.
Pool층은 2×2 filter에 S=2
=> CONV시에 n_H, n_W는 동일, 채널수 2,3배 증가
Pool 시 n_H, n_W를 절반 감소.
즉, 체계적으로 줄어든다. (단점 parameter수가 정말 많아 training할 네트워크가 매우 커진다.

* ResNet · (목표) Residual Networks을 학습.

Q 아주 deep한 신경망을 학습시키기 어려운 이유는? ≈ Vanishing gradient or gradient가 폭발적 증가

* 한 층의 비선형 함수를 가지고 깊은 층을 학습시켜보자! = ResNet.

Q ResNet은 "Residual Block"이라는 작은 블록으로 구성

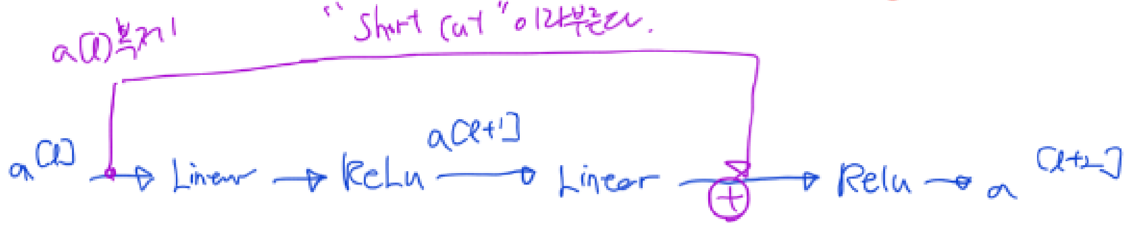


Short cut은 'Linear'연산 뒤에, 'Relu'연산 전에 ⊕된다.

a → Linear → ReLU

cost함수 $z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$, $a^{[l+1]} = g(z^{[l+1]})$

*g는 Relu 비선형성을 나타냄



Relu 비선형성을 적용해주기전에 a[l]을 더해주고 Relu

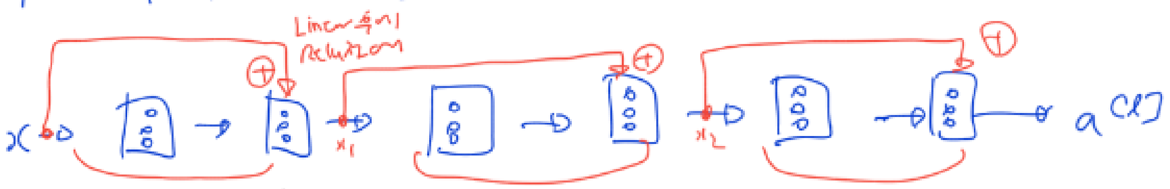
∴ $a^{[l+2]} = Relu = g(z^{[l+2]} + a^{[l]})$

a[l]을 잔여블록이라한다.

이렇게 a[l]을 복제해서 Linear 후, Relu 전에 더해주는 것을 "short cut"또는 층을 skip한다고해서 "skip connection"이라한다.

∴ Residual Block (잔여블록)를 사용하면 훨씬 깊은 신경망을 훈련시킬수 있다.

"Plain Network" (평형망)



이 평형망을 Resnet으로 바꾸려면 Scip connection을 더해주면된다.

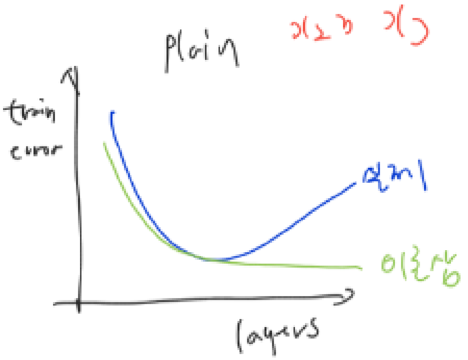
총 3개의 잔여블록이 합쳐진것.

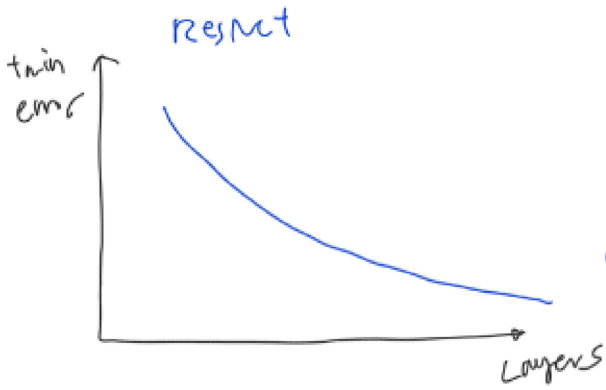
$x \to x_1$ 에 더해지고

$x_1 \to x_2$ 에서 더해지고

$x_2 \to x_3$ ''

$a^{[l]}$은 x_2가 $a^{[l-1]}$에 더해져서 생성





100개이상 층에서도 훈련 error를 신경망이 깊어지더라도 감소 시킬수 있다.

즉 Resnet은 잔여블록을 Linear 연산 후, Relu연산 전에 더해줌으로써

질문 why? Relu 전에 해야하지 ⊕ Residual Block을?

⇒ 논문이랑 구글링해볼결과 (original Resnet과 앞의 위치에서 residual Block을 더해준 것 학습 비교

→ original의 결과가 가장 좋음 (Relu 전)

이런변형 시도

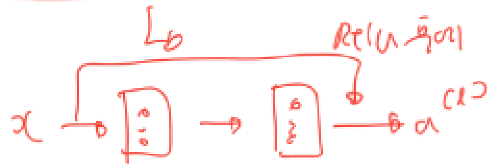
(1X1 CONV를 추가하거나 / 기존절반은 그냥보내고 절반은 ⊕)

→ Relu가 multiplication에서 blocking된 형태이기때문에 정보가 곧바로 전달되는게 아니라 곱셈에의해 변형된형태로 전달된다.

이로인해 최적화 문제가 생길수 있게됨.

또는 Relu를 먼저 해보아도 → train error 훨씬 더 떨어짐.

↳ Relu 후에 더하면 결과가 항상 non-negative가 나옴.



∴ original Resnet이 가장 성능이 좋아서

$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right)$$

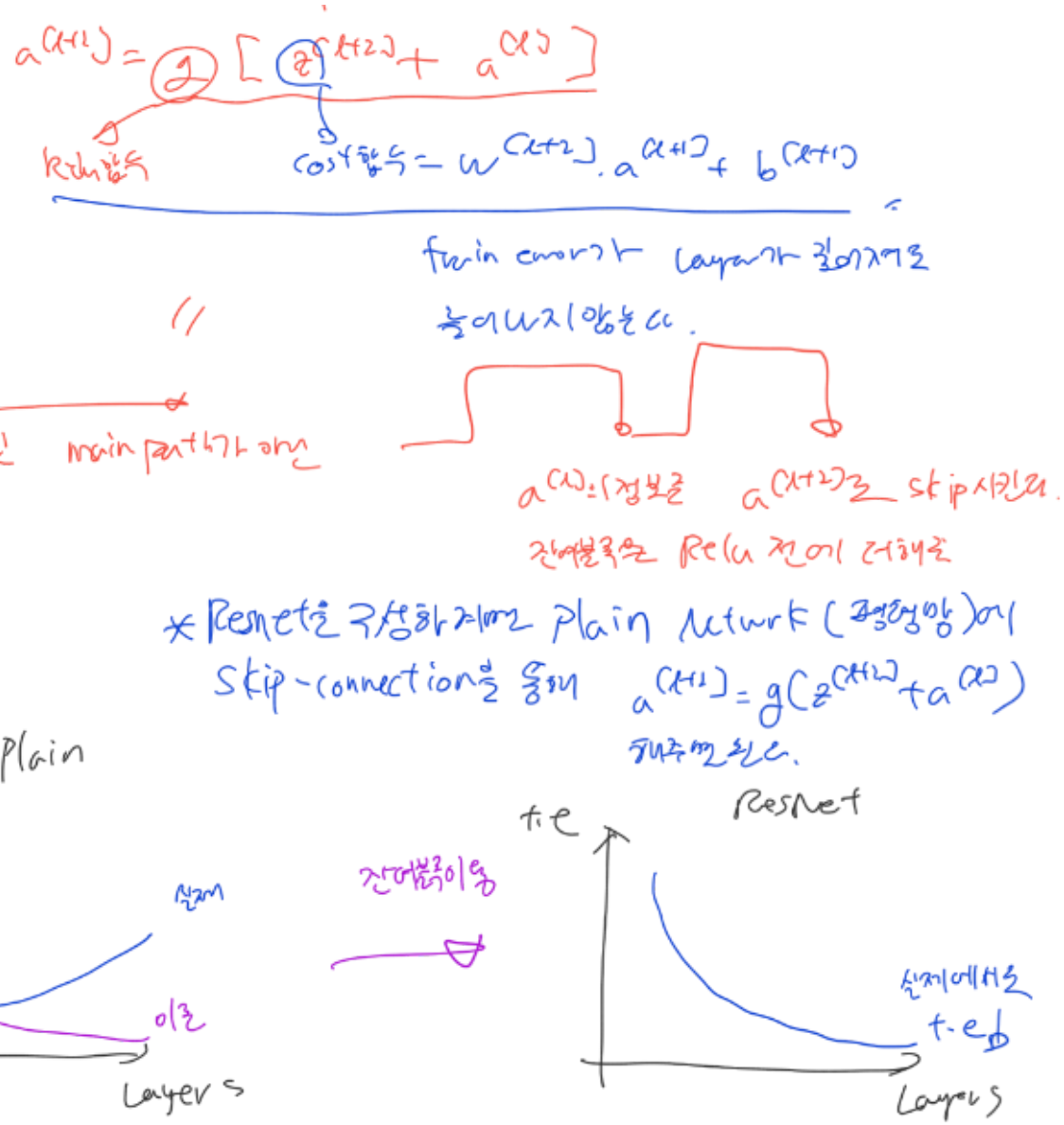
g: Relu함수

z: cost함수 $z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$

이로인해 Resnet이 깊어지는 신경망을 학습시키는데 좋은 효과를 가짐

⇒ 'Residual Block'을 Relu전에 더해주자

skip-connection 또는 short-cut



* ResNet이 잘 작동하는 이유는??

마지막 수정: 오후 5:49