ELLab

SUNG KYUN KWAN UNIVERSITY

ResNet

2025.10.30

Review

기존 문제점

Is learning better networks as easy as stacking more layers?

↓

Problem: gradient vanishing & exploding

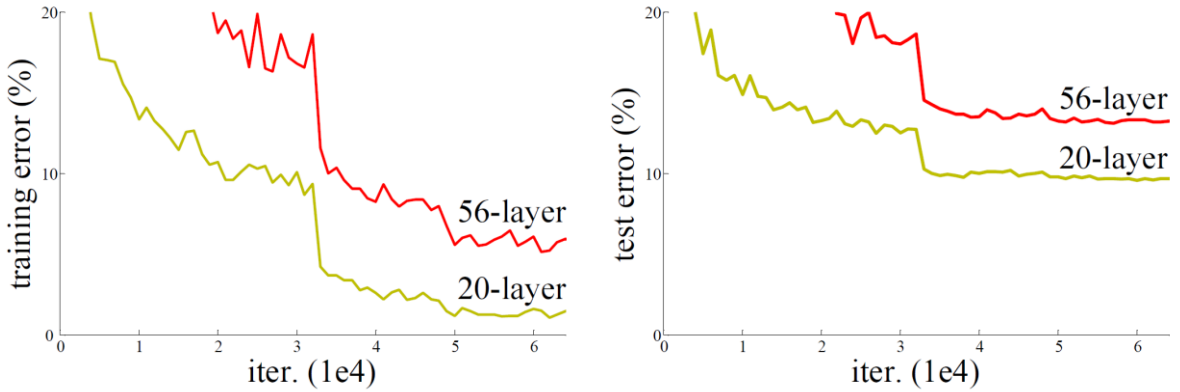
Solution: normalized

↓

Problem: train error 大 → degradation

Solution: constructed identity model

But cannot resolve



Residual Learning

- 비선형 활성함수있는 상태에서 여러층을 거쳐 identity 만드는것 어려움
- Degradation 문제 해결 안됨
- Identity가 optimal일때 모델은 단순히 각 비선형 층의 weight를 0으로 수렴시키는게 더 쉬움

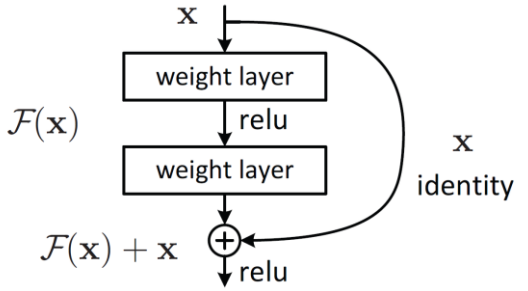
Review

Residual learning

"To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers"

Residual mapping

- 학습하고자하는 목표 함수 H(x) → F(x)
- 전체 함수가 아니라 입력 x로 부터의 차이를 학습함
- Reference가 있어서 더 쉬움

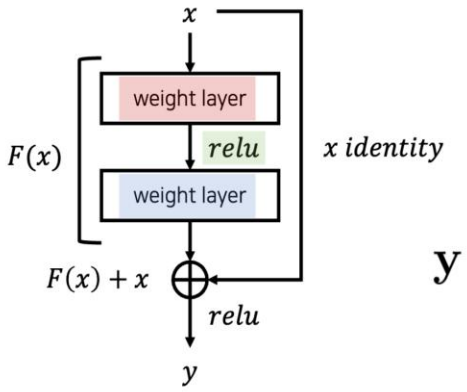


$$\mathcal{F}(x) \coloneqq \mathcal{H}(x) - x$$

Underlying mapping: $H(x)$

Shortcut connection

- Layer를 건너 뛰면서 identity mapping 수행함
- 파라미터 증가시키지 않음 (option A), Plain과 공정한 비교 가능
- Residual blocks의 input과 output의 차원이 다른 경우
  Projection mapping으로 수행



$$\mathcal{F} = W_2\sigma(W_1\mathbf{x})$$

일반적인 형태

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}$$

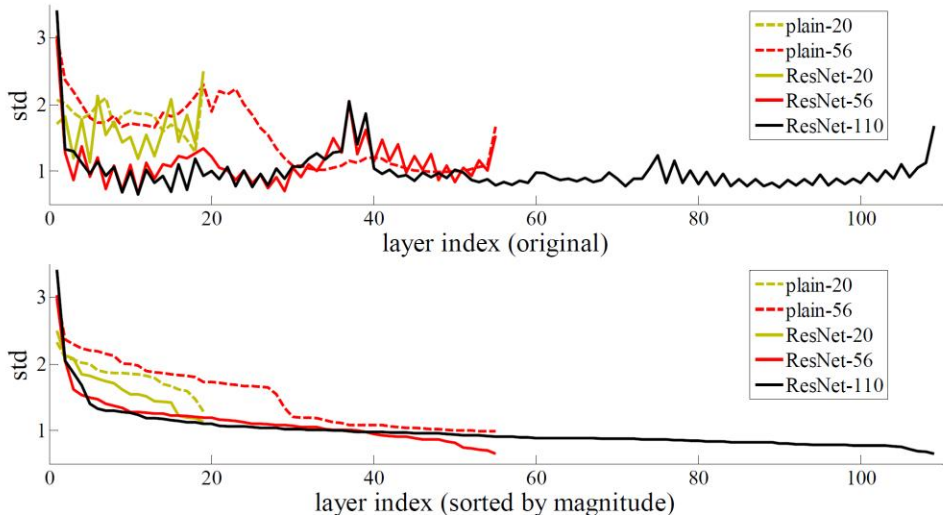
multiple convolutional layers　shortcut

Review

Fig. 7

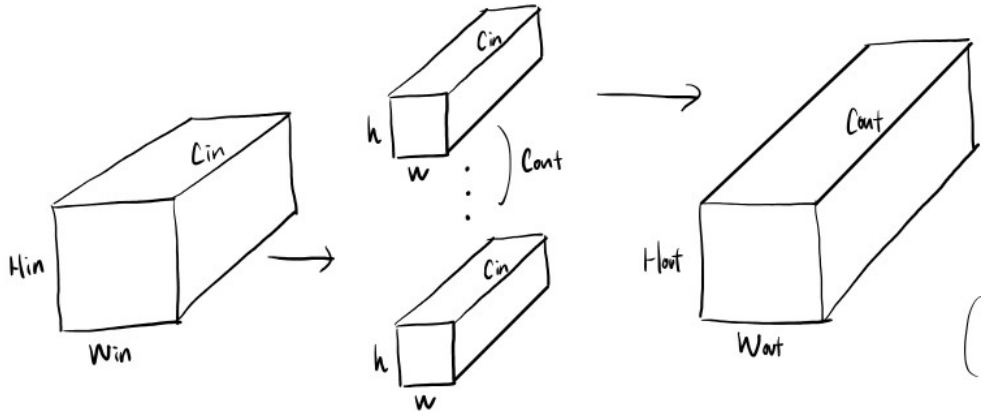
"In real cases, it is unlikely that identity mappings are optimal, but our reformulation may help to precondition the problem."

- Plain(점선): 전체적으로 표준편차가 더 큼
- Residual(실선): 대부분의 층에서 출력 분산이 훨씬 작음
  - $y = \mathcal{F}(x) + x$ 에서 더해지는 잔차 F가 거의 0에 가까운 값을 내고 있고 대부분의 layer는 입력 x값을 거의 그대로 통과시키고 아주 작은 보정(F)만 수행함
  - Identity가 학습과정의 좋은 출발점(reference) 역할을 하고 있음
- Resnet이 깊어질수록 response 크기가 점점 작아짐
  - 층이 많을수록 개별 layer가 입력 신호를 더 적게 변경함



Review

Bottleneck 연산량 비교



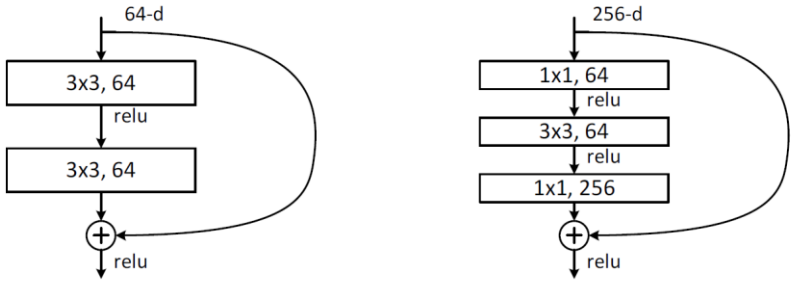


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

Convolution 연산량 $\propto H_{\text{out}} \times W_{\text{out}} \times \text{h} \times \text{w} \times \text{C}_{\text{in}} \times \text{C}_{\text{out}}$

- Basic ResNet: $2\left(56^2 \times 3^2 \times 64^2\right) = 231{,}211{,}008$
- Bottleneck: $\left(56^2 \times 1^2 \times 256 \times 64\right) + \left(56^2 \times 3^2 \times 64\right) + \left(56^2 \times 1^2 \times 64 \times 256\right) = 218{,}365{,}952$
- both designs have similar time complexity

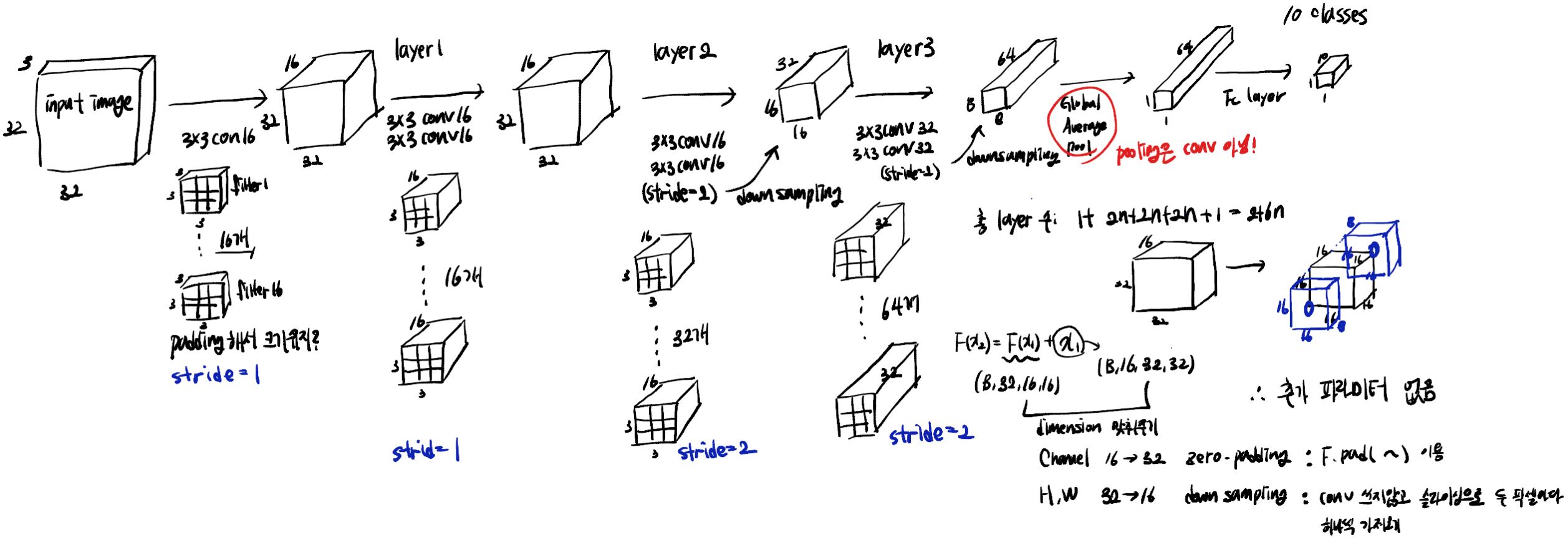
CIFAR-10

모델 구조

- ✓ 첫번째 층은 3x3 convolution
- ✓ 이후에 총 6n개의 3x3 conv층 사용함
- ✓ 필터수 {16, 32, 64}
- ✓ Downsampling은 stride=2의 convolution
- ✓ 네트워크의 끝은 Global Average Pooling → 10-way FC layer → Softmax 구조
- ✓ Short cut은 항등연결만 사용 -> option A

CIFAR-10

모델 구조



CIFAR-10

학습 세팅

항목	논문 설정값
데이터셋	CIFAR-10 (50k train / 10k test)
이미지 전처리	32×32, 4픽셀 zero-padding 후 32×32 랜덤 crop, horizontal flip
모델 깊이	n={3,5,7,9} → 20, 32, 44, 56층
Optimizer	SGD (momentum=0.9)
Batch size	128
Weight decay	1e-4
Learning rate	0.1 → 1/10 at 32k, 48k iter → stop at 64k iter
Scheduler	step decay (epoch ≈ [82, 123, 164])
Normalization	BatchNorm after each Conv, before ReLU
Training epochs	165 epochs (64k iter × 128 / 50k ≈ 164 epochs)

CIFAR-10

구현

"On this dataset we use identity shortcuts in all cases (i.e., option A)"

- channel zero padding: F.pad 함수 이용
- downsampling: conv 쓰지 않고 슬라이싱으로 구현, plain과 파라미터 수 똑같음

"We start with a learning rate of 0.1, divide it by 10 at 32k and 48k iterations, and terminate training at 64k iterations"

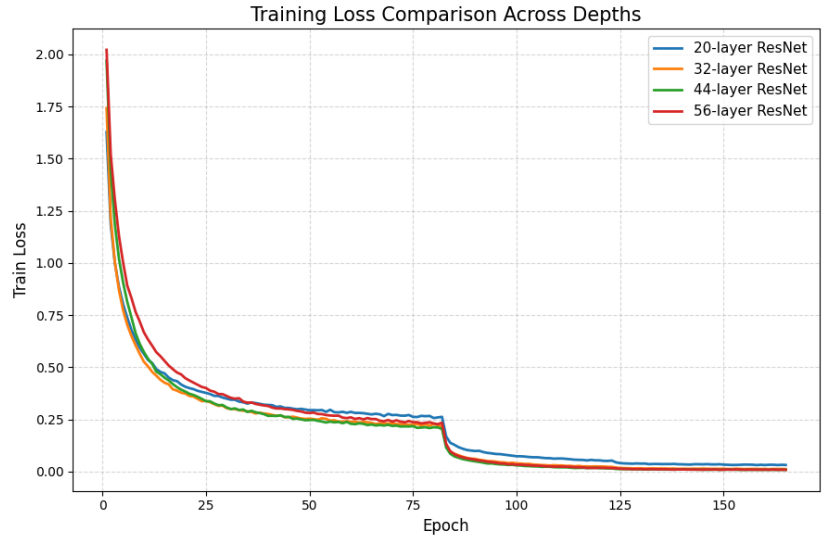
- Learning Rate Scheduler (MultiStepLR) 사용, 학습 도중 지정한 epoch에 도달하면 learning rate를 gamma배로 줄임
  - train 5k, batch size 128: 한 epoch의 iteration 수 – $5\text{k}/128 \approx 391$
  - 32k iterations: $32\text{k}/391 \approx 82(\text{epoch})$
  - 48k iterations: $48\text{k}/391 \approx 123(\text{epoch})$

https://colab.research.google.com/drive/1qQBfloj0_J6uZyoKyqkUoKBm-1pjphwm?usp=sharing

CIFAR-10

결과

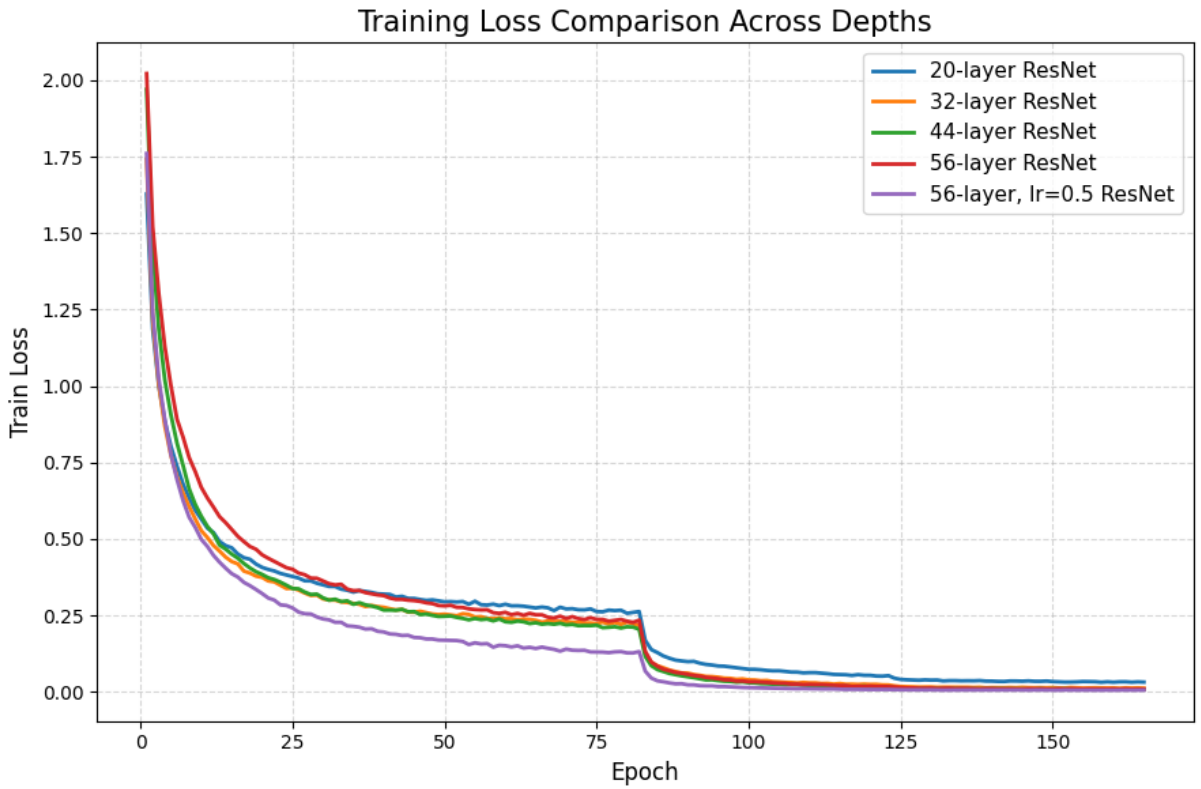
- 20,32,44층에서는 층이 깊어질수록 train loss가 작아지고 validation acuuracy가 높아짐 → good
- 반면 56층에서는 train loss가 44층에 비해 커졌고 validation accuracy도 낮아짐 → under fitting
  - 초반에 train loss가 다른 층들에 비해서 많이 컸는데 초기 학습률이 커서 손해를 보고 시작한 것으로 생각됨 (논문에서도 110층은 warm up을 거쳤다고 나옴)



Layers	Train Loss (last epoch)	Validation Accuracy
20 (n=3)	0.0319	90.61%
32 (n=5)	0.0123	91.28%
44 (n=7)	0.0084	91.34%
56 (n=9)	0.0088	90.45%

CIFAR-10

결과



Layers	Train Loss (last epoch)	Validation Accuracy
20 (n=3)	0.0319	90.61%
32 (n=5)	0.0123	91.28%
44 (n=7)	0.0084	91.34%
56 (n=9)	0.0088	90.45%
56 (lr=0.5)	0.0059	91.73%

- 56 layer 초기 learning rate를 0.05로 변경해서 돌려봄
  - 83에서 scheduler로 lr 조정해주기 전까지, 학습이 이전보다 확연히 나아진 것을 볼 수 있음

CIFAR-10

코드 수정할점

- ✓ 처음 train5만장에서 45k/5k로 train, val set 분리 했어야 했는데 5만장 전체를 학습하고 test 데이터 1만장을 검증 데이터로 사용함
- ✓ Layer1 직전에 실수로 maxpool 진행함, basic resnet-34에서는 쓰는게 맞았는데 CIFAR-10 설명에는 쓰라는 말 없음
- ✓ Short cut 진행할때 layer1에 적용이 안되는 것 같음, layer 1에는 직접 shortcut 적용하게끔 수정해야함
- ✓ 데이터 normalize 할때 평균, 분산 어떻게 지정해야할지 몰라서 일단 classifier 연습 코드에서 쓴것 그대로 사용함

https://colab.research.google.com/drive/1v0VfDbI0F6RKzHROAMeY6OYzPnmv3ghM?usp=sharing

Layers	Train Loss (last epoch)	Validation Accuracy
20 (n=3)	0.0319	90.61%
32 (n=5)	0.0123	91.28%
44 (n=7)	0.0084	91.34%
56 (n=9)	0.0088	90.45%
56 (lr=0.5)	0.0059	91.73%
20 수정본	0.0236	91.44%

ELLab