ELLab

SUNGKYUNKWAN UNIVERSITY

PreActResNet

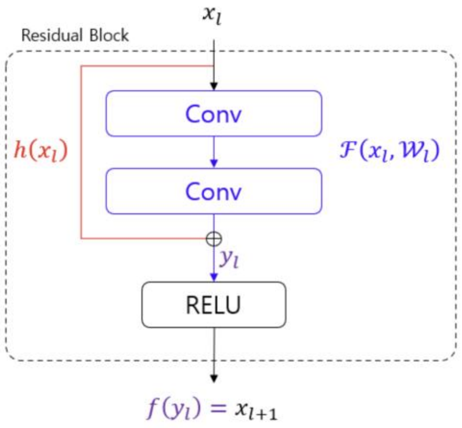
2025.11.12

Introduction

Representation

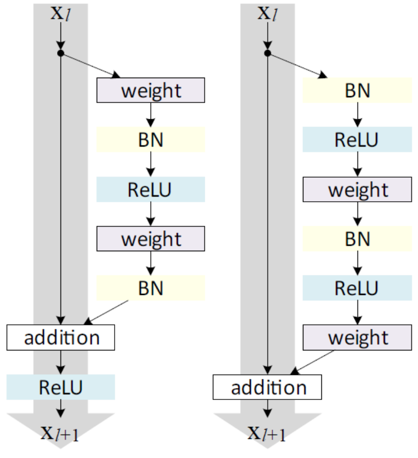
$$\mathcal{F}(x) \coloneqq \mathcal{H}(x) - x$$

Underlying mapping: $H(x)$

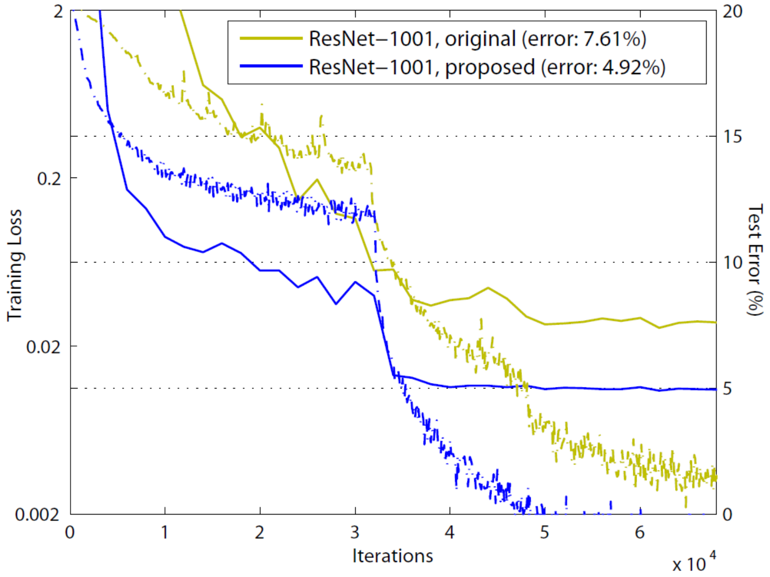


$$y_l = \underbrace{h(x_l)}_{\text{Skip-connection}} + \underbrace{\mathcal{F}(x_l, \mathcal{W}_l)}_{\text{Convolutional layer}}$$

$$x_{l+1} = \underbrace{f(y_l)}_{\text{RELU}}$$



(a) original (b) proposed



- Original ResNet: h is identity, f is relu
- Assumption 1: h is identity, f is also identity
- Assumption 2: h is not identity -> $\mathrm{h(x)} = \lambda \mathrm{x}$, f is identity

- Iter 초기에 original보다 proposed가 더 빨리 수렴함
    - Dashed line: training loss
    - Solid line: test loss

Skip connection

Mathematical Approach

$$\mathbf{y}_l = h(\mathbf{x}_l) + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l),$$
$$\mathbf{x}_{l+1} = f(\mathbf{y}_l),$$

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l).$$

$$\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i), \quad (4)$$

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left(1 + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)\right). \quad (5)$$

- Assumption 1: h is identity, f is also identity
  - (4) summation form
  - (5) gradient can be directly propagated
    - Due to ‘1’, layer does not vanish

Skip connection

Mathematical Approach

$$h(\mathbf{x}_l) = \lambda_l \mathbf{x}_l$$

$$\mathbf{x}_{l+1} = \lambda_l \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l),$$

$$\mathbf{x}_L = (\prod_{i=l}^{L-1} \lambda_i)\mathbf{x}_l + \sum_{i=l}^{L-1} (\prod_{j=i+1}^{L-1} \lambda_j)\mathcal{F}(\mathbf{x}_i, \mathcal{W}_i),$$

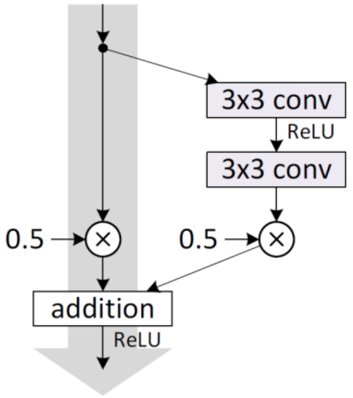
$$\mathbf{x}_L = (\prod_{i=l}^{L-1} \lambda_i)\mathbf{x}_l + \sum_{i=l}^{L-1} \hat{\mathcal{F}}(\mathbf{x}_i, \mathcal{W}_i), \quad (7)$$

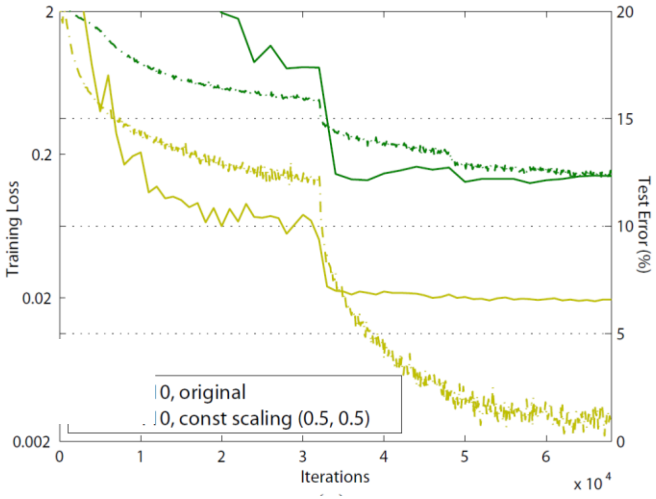
$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left((\prod_{i=l}^{L-1} \lambda_i) + \frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} \hat{\mathcal{F}}(\mathbf{x}_i, \mathcal{W}_i) \right) \quad (8)$$

- Assumption 2: h is not identity -> $\mathrm{h}(\mathrm{x}) = \lambda \mathrm{x}$, f is identity
  - (7) factorial form
  - (8) forces it to flow through the weight layers
    - factor 때문에 L이 커지면(deep network), scalar 값에 의해 gradient exploding/vanishing 발생할 수 있음

Experiment

Constant scaling



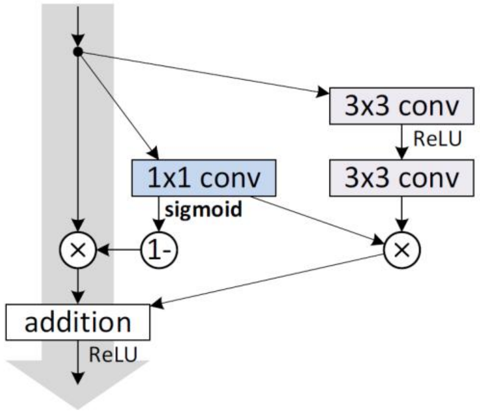


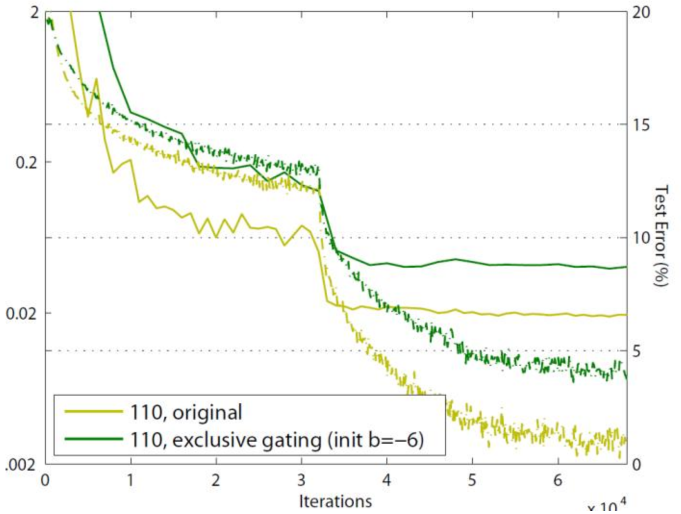
case	Fig.	on shortcut	on $\mathcal{F}$	error (%)	remark
original [1]	Fig. 2(a)	1	1	**6.61**	
constant scaling	Fig. 2(b)	0	1	fail	This is a plain net
		0.5	1	fail	
		0.5	0.5	12.35	frozen gating
exclusive gating	Fig. 2(c)	$1-g(\mathbf{x})$	$g(\mathbf{x})$	fail	init b_g=0 to -5
		$1-g(\mathbf{x})$	$g(\mathbf{x})$	8.70	init b_g=-6
		$1-g(\mathbf{x})$	$g(\mathbf{x})$	9.81	init b_g=-7
shortcut-only gating	Fig. 2(d)	$1-g(\mathbf{x})$	1	12.86	init b_g=0
		$1-g(\mathbf{x})$	1	6.91	init b_g=-6
1×1 conv shortcut	Fig. 2(e)	1×1 conv	1	12.22	
dropout shortcut	Fig. 2(f)	dropout 0.5	1	fail	

- $h(x) = \lambda x$ 에서 λ를 0.5로 고정 -> skip 경로를 original의 절반으로 줄임
  - (i) F is not scaled: $x_{i+1} = 0.5 \cdot h(x_i) + F(x_i, w_i)$ → not converge
  - (ii) F is scaled by constant gate (1- λ): $x_{i+1} = 0.5 \cdot h(x_i) + 0.5 \cdot F(x_i, w_i)$ → converge but higher error

Experiment

Exclusive gating



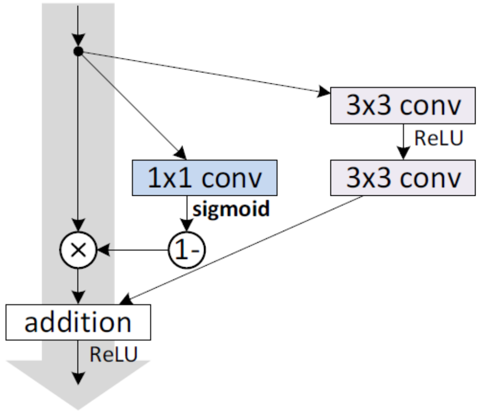


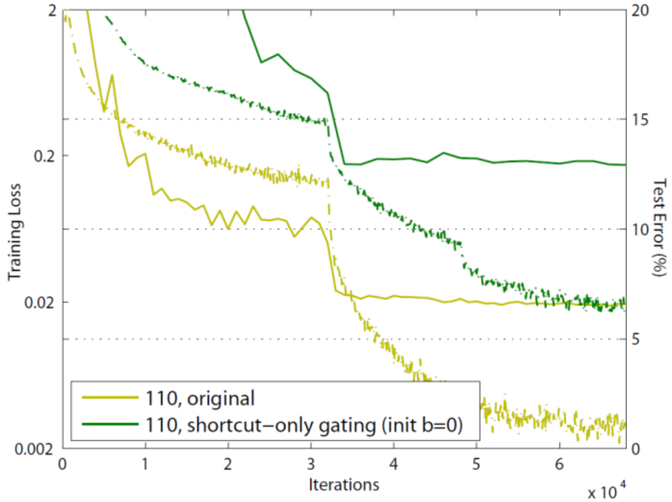
case	Fig.	on shortcut	on $\mathcal{F}$	error (%)	remark
original [1]	Fig. 2(a)	1	1	**6.61**	
constant scaling	Fig. 2(b)	0	1	fail	This is a plain net
		0.5	1	fail	
		0.5	0.5	12.35	frozen gating
exclusive gating	Fig. 2(c)	$1-g(\mathbf{x})$	$g(\mathbf{x})$	fail	init b_g=0 to -5
		$1-g(\mathbf{x})$	$g(\mathbf{x})$	8.70	init b_g=-6
		$1-g(\mathbf{x})$	$g(\mathbf{x})$	9.81	init b_g=-7
shortcut-only gating	Fig. 2(d)	$1-g(\mathbf{x})$	1	12.86	init b_g=0
		$1-g(\mathbf{x})$	1	6.91	init b_g=-6
1×1 conv shortcut	Fig. 2(e)	1×1 conv	1	12.22	
dropout shortcut	Fig. 2(f)	dropout 0.5	1	fail	

- gating fuction: $g(x) = \sigma(W_g x + b_g) \rightarrow$ 1x1 convolution
  - $x_{i+1} = (1 - g(x_i)) \cdot x_i + g(x_i) \cdot F(x_i, w_i)$
  - $(1-g(x))$가 1에 가까워지면 identity에 가까워져서 정보전달 유리하지만 $g(x)$가 0에 수렴하면 잔차함수 F가 억제됨

Experiment

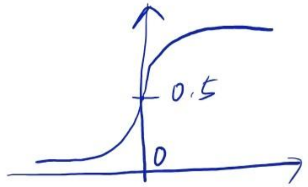
Shortcut-only gating





case	Fig.	on shortcut	on $\mathcal{F}$	error (%)	remark
original [1]	Fig. 2(a)	1	1	**6.61**	
constant scaling	Fig. 2(b)	0	1	fail	This is a plain net
		0.5	1	fail	
		0.5	0.5	12.35	frozen gating
exclusive gating	Fig. 2(c)	$1-g(\mathbf{x})$	$g(\mathbf{x})$	fail	init b_g=0 to -5
		$1-g(\mathbf{x})$	$g(\mathbf{x})$	8.70	init b_g=-6
		$1-g(\mathbf{x})$	$g(\mathbf{x})$	9.81	init b_g=-7
shortcut-only gating	Fig. 2(d)	$1-g(\mathbf{x})$	1	12.86	init b_g=0
		$1-g(\mathbf{x})$	1	6.91	init b_g=-6
1×1 conv shortcut	Fig. 2(e)	1×1 conv	1	12.22	
dropout shortcut	Fig. 2(f)	dropout 0.5	1	fail	

- gating fuction: $g(x) = \sigma(W_g x + b_g) \rightarrow$ 1x1 convolution, short cut에만 적용함
  - $x_{i+1} = (1 - g(x_i)) \cdot x_i + F(x_i, w_i)$
  - Initial b_g에 따라 결과 달라짐
    - $b_g = 0 \rightarrow E(1 - g(x)) = 0.5$: error rate: 12.8%
    - $b_g = -6 \rightarrow E(1 - g(x)) = 1$: almost identity mapping, error rate: 6.61%

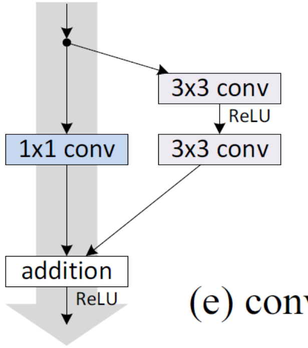


$g(x) = \text{sigmoid}(W_g x + b_g)$

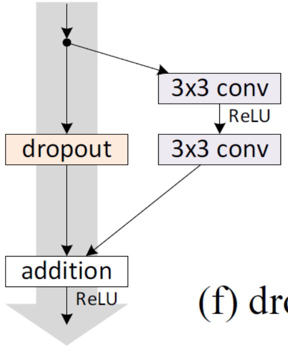
$b_g = 0,\ g(x) \approx 0.5$
$b_g > 0,\ g(x) > 0.5$
$b_g < 0,\ g(x) < 0.5$

Experiment

etc



(e) conv shortcut



(f) dropout shortcut

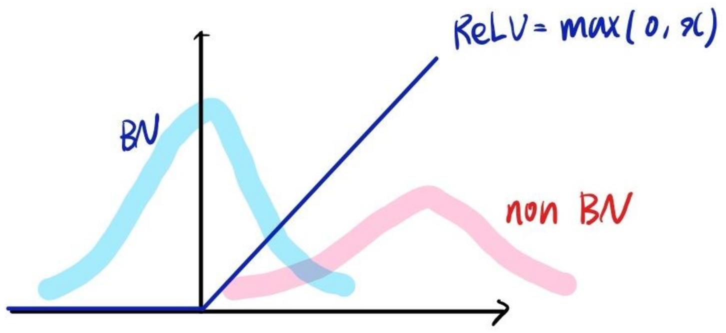
case	Fig.	on shortcut	on $\mathcal{F}$	error (%)	remark
original [1]	Fig. 2(a)	1	1	**6.61**	
constant scaling	Fig. 2(b)	0	1	fail	This is a plain net
		0.5	1	fail	
		0.5	0.5	12.35	frozen gating
exclusive gating	Fig. 2(c)	$1-g(\mathbf{x})$	$g(\mathbf{x})$	fail	init b_g=0 to -5
		$1-g(\mathbf{x})$	$g(\mathbf{x})$	8.70	init b_g=-6
		$1-g(\mathbf{x})$	$g(\mathbf{x})$	9.81	init b_g=-7
shortcut-only gating	Fig. 2(d)	$1-g(\mathbf{x})$	1	12.86	init b_g=0
		$1-g(\mathbf{x})$	1	6.91	init b_g=-6
1×1 conv shortcut	Fig. 2(e)	1×1 conv	1	12.22	
dropout shortcut	Fig. 2(f)	dropout 0.5	1	fail	

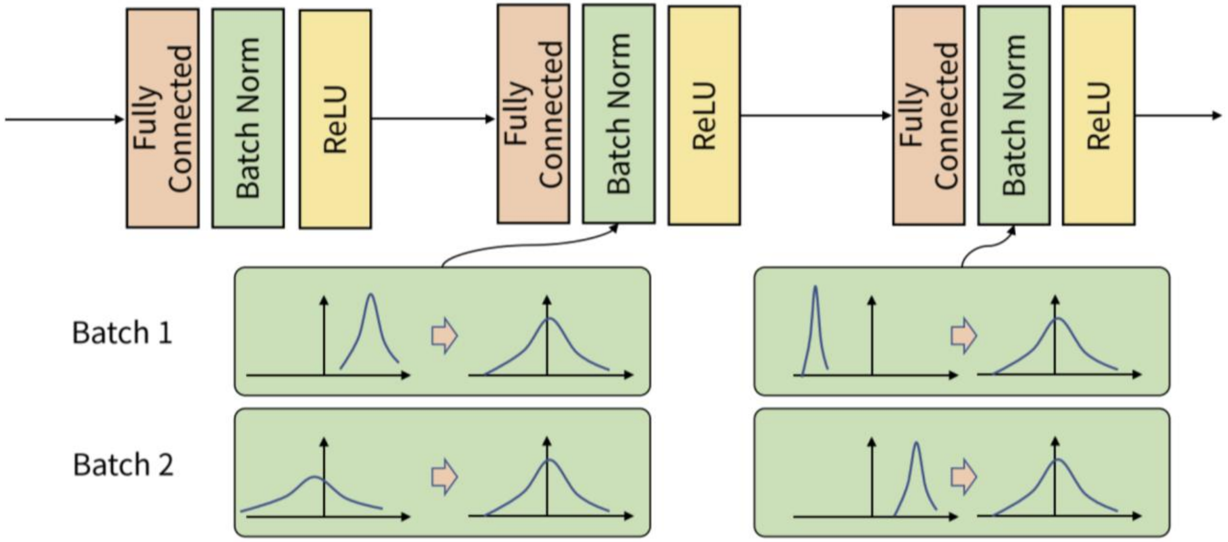
- Conv shortcut
  - Resnet option C
  - Shallow layers(ResNet-34)에서는 유용하지만 deep layer(ResNet-110)에서는 error rate 높음 (파라미터 수 때문)
- Dropout shortcut
  - Short cut에 평균 0.5의 scale을 강제하며 constant scaling(λ = 0.5)와 비슷함
  - Signal propagation을 방해함

Activation Function

$$BN(X) = \gamma\left(\frac{X - \mu_{batch}}{\sigma_{batch}}\right) + \beta$$

ReLU & Batch Normalization



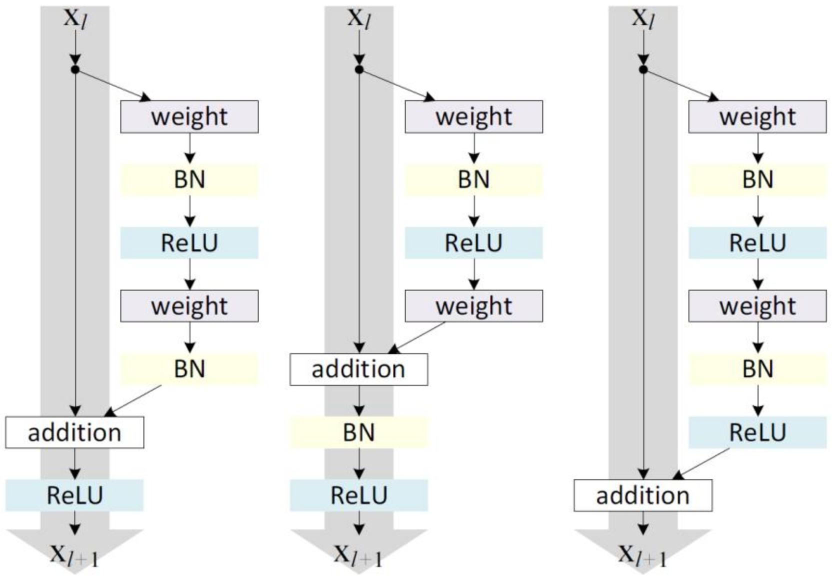


- Internal Covariate Shift 감소
  - Batch가 Layer 통과하면서 각 계층에서 입력 분포가 달라짐
- 학습 과정에서 각 배치 단위 별로 데이터가 다양한 분포를 가지더라도 각 배치별로 평균과 분산을 이용해 정규화함
- Batch Norm with ReLU
  - Non BN: 모든 데이터가 0보다 커져서 ReLU 통과하더라고 non-linearity를 줄 수 없게됨
  - BN: 평균을 0이 되게끔 정규화하므로 음수는 0으로, 양수는 linear하게 전달되도록하는 ReLU의 원래 목적을 잘 반영할 수 있게 도와줌

Experiment

BN after addition & ReLU after addition

case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**



(a) original

(b) BN after addition

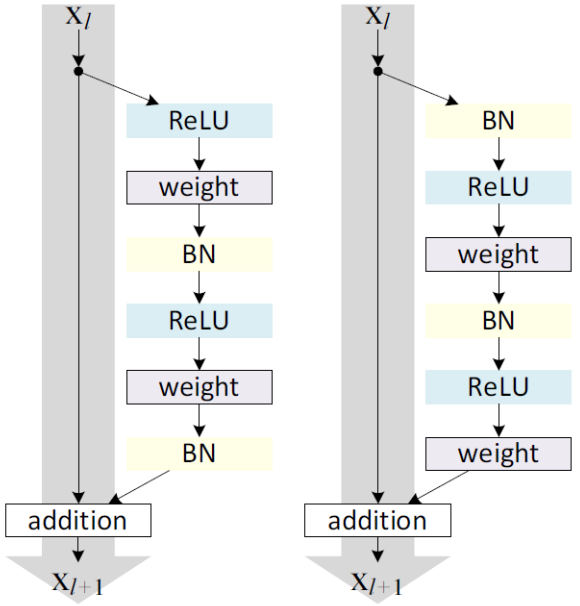
(c) ReLU before addition

- BN after addition
  - BN이 Skip connection 신호를 변경해서 정보 전파 방해함
  - 학습 초기에 train loss 감소 어려워짐
- ReLU after addition
  - $ReLU = max(0, x)$이므로 F의 출력이 비음수가 됨
  - 순전파 신호가 단조증가만 하므로 표현력에 안좋은 영향을 줌 → Residual Function should have values in $(-\infty, \infty)$

Experiment

Pre-Activation model

case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**



(d) ReLU-only pre-activation (e) **full pre-activation**

- (i) ReLU-only pre-activation
  - BN의 이점 충분히 활용하지 못함
- (ii) full pre-activation
  - BN과 ReLU가 붙어있어서 정규화 효과를 줌
  - 유의미한 error rate 감소

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l).$$
$$\mathbf{x}_{l+1} = f(\mathbf{y}_l),$$

Identity ver

asymmetric form

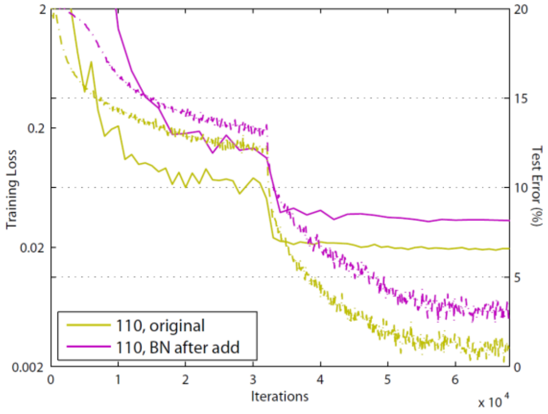
$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\hat{f}(\mathbf{x}_l), \mathcal{W}_l)$$

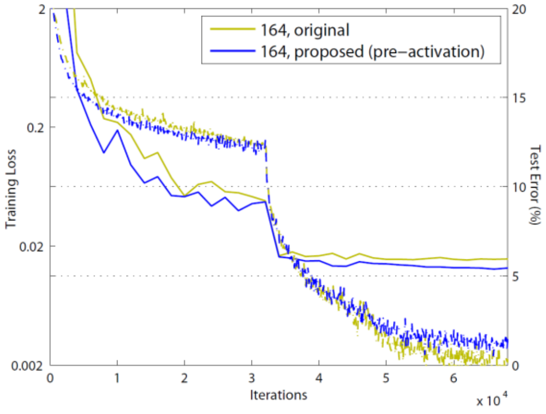
Pre-active ver

Experiment

Pre-Activation model

- 수렴 시점에 training loss는 약간 더 높지만 test error는 더 낮음
  → 일반화성능 향상
- 기존 Residual Unit에서는 BN으로 신호를 정규화 해도 short cut과 더해지면서 다시 비정규화 상태가 됨 → 가중치 층의 입력으로 사용
- Pre-active 구조에서는 모든 가중치의 입력이 BN에 의해 이미 정규화 되어있음
- 모든 실험에서 기존보다 pre-active error rate가 더 낮음



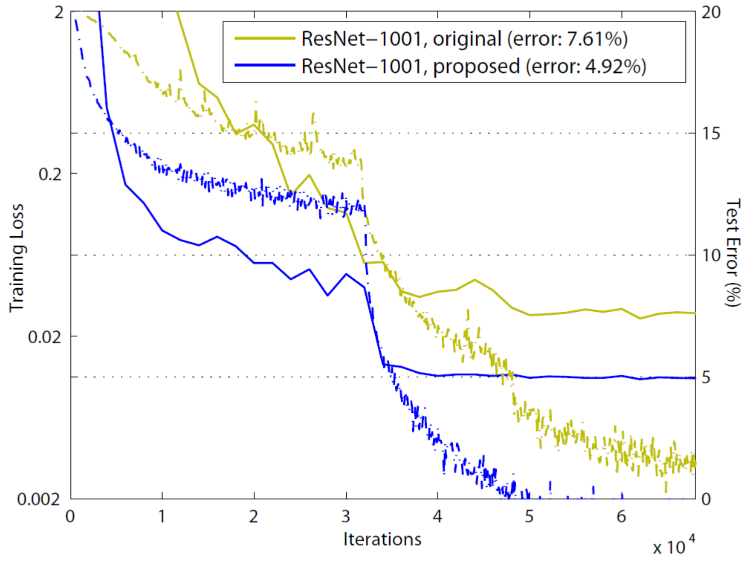


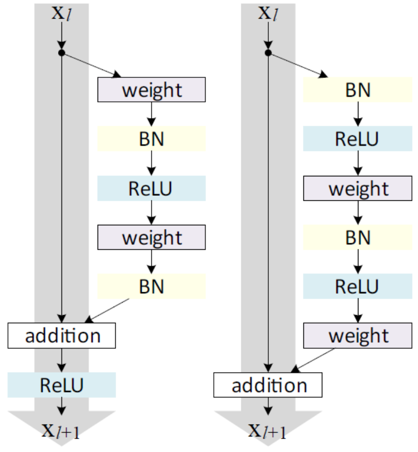
dataset	network	baseline unit	pre-activation unit
CIFAR-10	ResNet-110 (1layer skip)	9.90	8.91
	ResNet-110	6.61	6.37
	ResNet-164	5.93	5.46
	ResNet-1001	7.61	4.92
CIFAR-100	ResNet-164	25.16	24.33
	ResNet-1001	27.82	22.71

Analysis

Pre-Activation model

- Ease of optimization
  - Original ResNet
    - 음수일때 relu에 의해 신호가 잘려나가고 깊을수록(1101) 이 효과 두드러짐
  - Proposed ResNet
    - f 가 항등매핑이므로 두 블록 사이에서 직접 전파 가능
    - Training loss 빠르게 줄임
- Reducing overfitting
  - BN의 정규화효과로 인해 train error는 조금 높아도 test error는 더 낮음(이전 슬라이드 참고)





(a) original (b) proposed

My Code

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])
Warm up	During 400 iter, use 0.01 lr -> but not necessary
Normalization	Conv1 -> activation -> … -> addition -> activation -> avg pooling -> fc
Training epochs	165 epochs (64k iter × 128 / 50k ≈ 164 epochs)

My Code

구현

"For the frst Residual Unit: adopt the first activation right after conv1 and before splitting into two paths"

"for the last Residual Unit: adopt an extra activation right after its element-wise addition"

Conv1 -> activation ->layers -> addition -> activation -> avg pooling -> fc

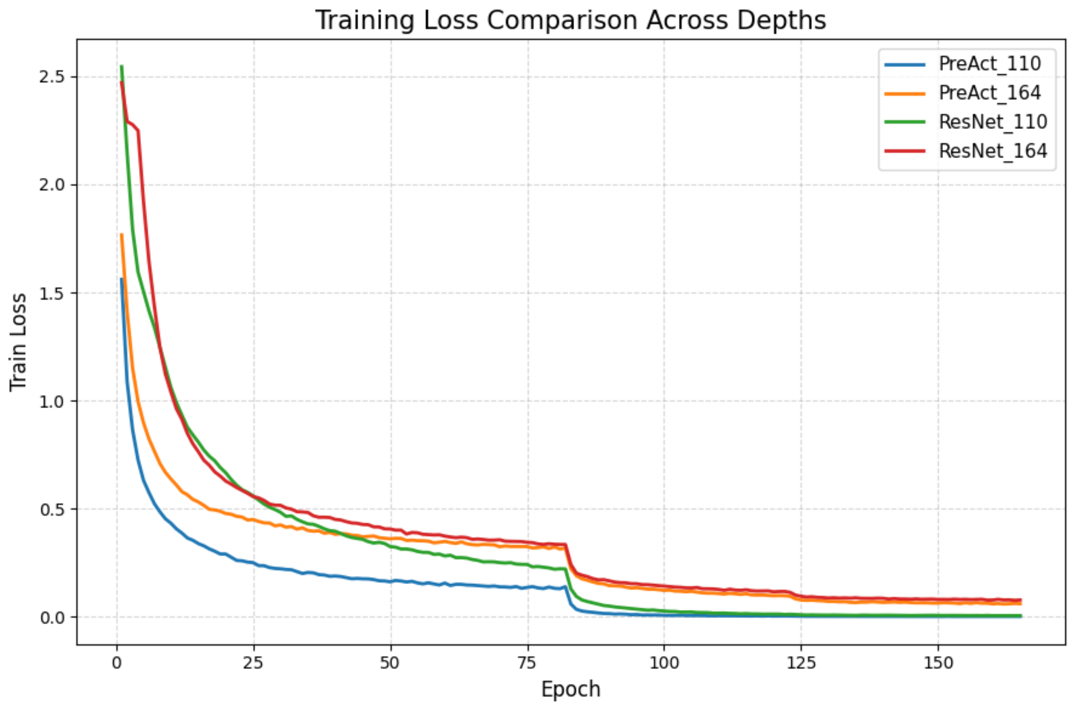
"Bottleneck: Residual Units (for ResNet-164/1001 on CIFAR) [1×1, 16] → [3×3, 16] → [1×1, 64]*"*

https://colab.research.google.com/drive/11duM97_KsVQy-L_VCOK4YhWC4v1jKs8V?usp=sharing

My Code

Experiment

- PreAct가 ResNet보다 학습 초기에 loss가 더 빨리 수렴함
- Bottleneck 구조는 basic block보다 전반적으로 accuracy가 낮음
    - 차원 축소하고 복원하는 과정에서 정보 손실 발생했을 가능성
- 더 깊은 네트워크에서 bottleneck과 basic 구조 실험했다면 bottleneck이 accuracy가 더 높은 결과 나왔을 수도 있음
- Floaps도 비교하려고 했는데 중간에 A100 gpu 다 써서 PreAct_164는 다른 L4로 돌림



Layers	baseline	Pre-active
110 (n=18)	91.52%	93.66%
164B (n=18)	90.04%	91.04%

ELLab