Paper presentation

Deep Residual Learning for Image Recognition[1]

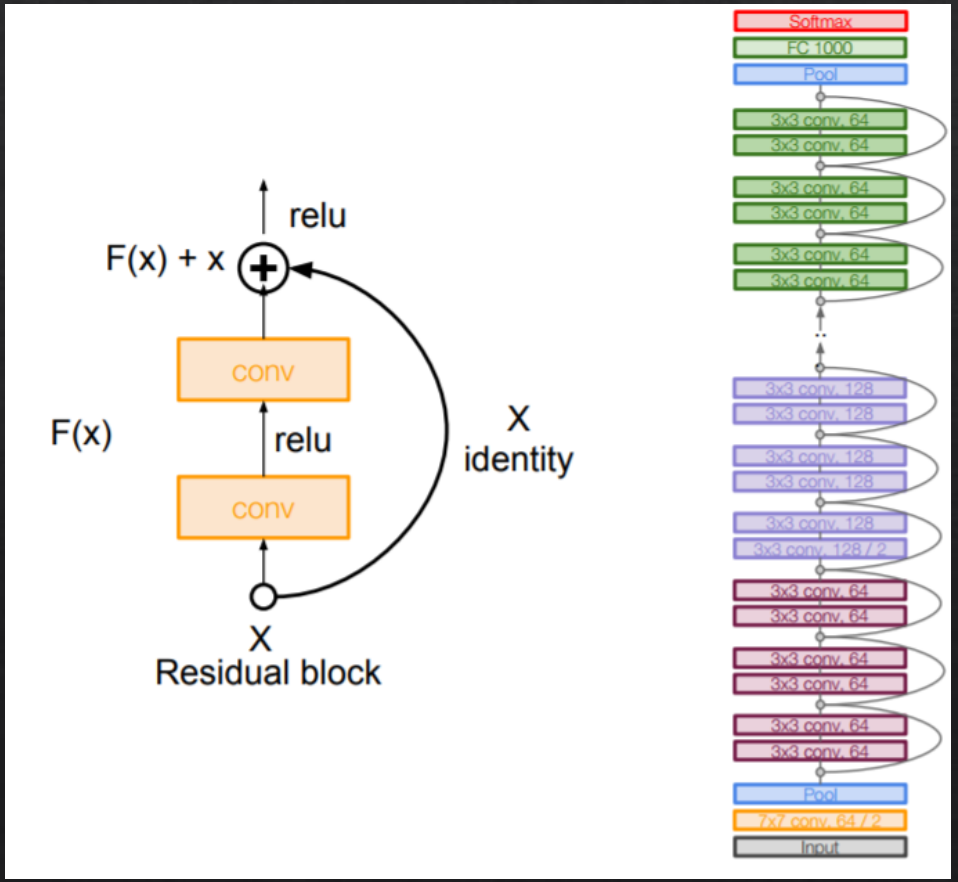
Identity Mappings in Deep Residual Networks[2]

Densely Connected Convolutional Networks[3]

Zayn Liu

Deep Residual Learning for Image Recognition

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015)



Is learning better networks as easy as stacking more layers ?

Is learning better networks as easy as stacking more layers ?

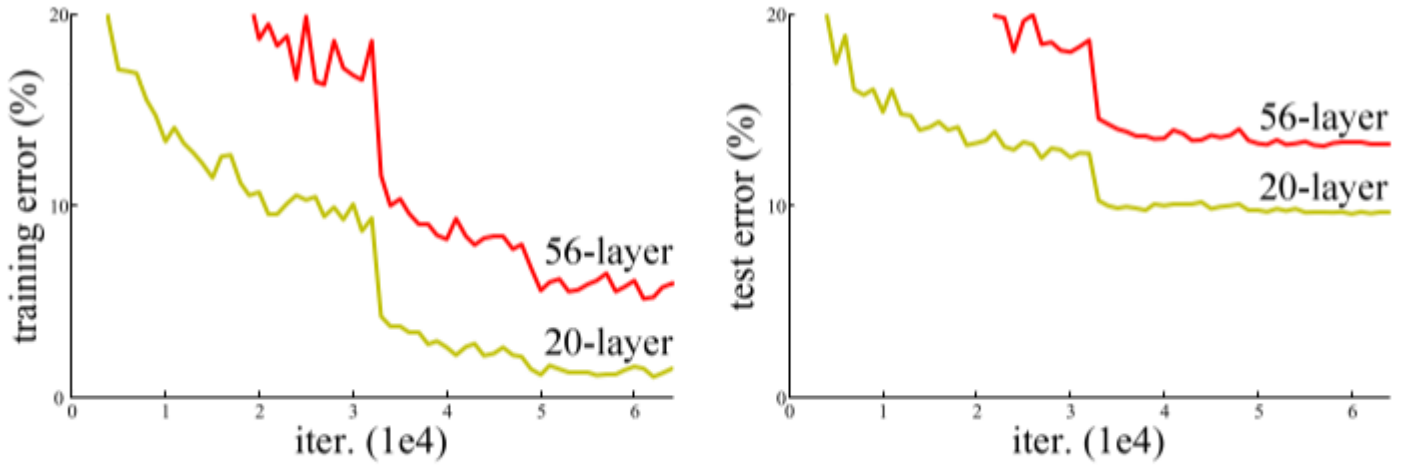


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Is learning better networks as easy as stacking more layers ?

- Gradient vanish ?
  - Solved by batch normalization
- Overfitting ?
  - Higher training error

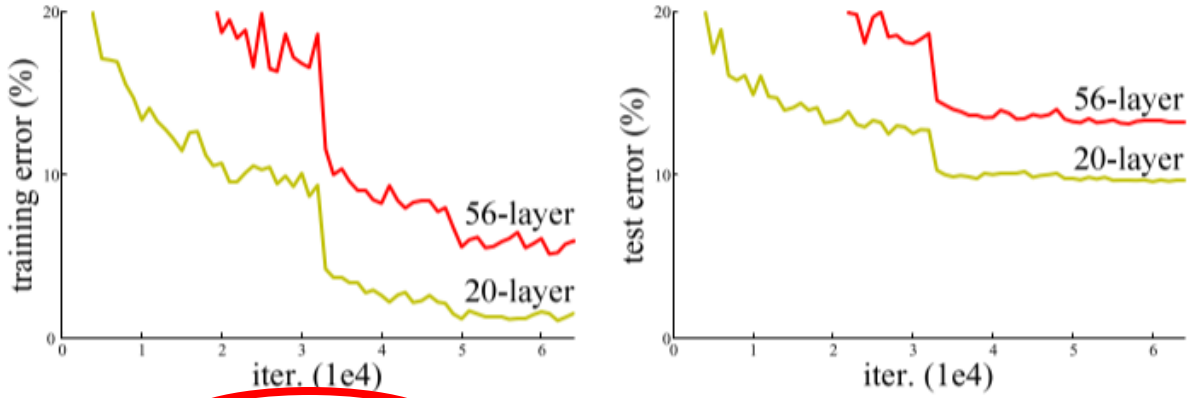
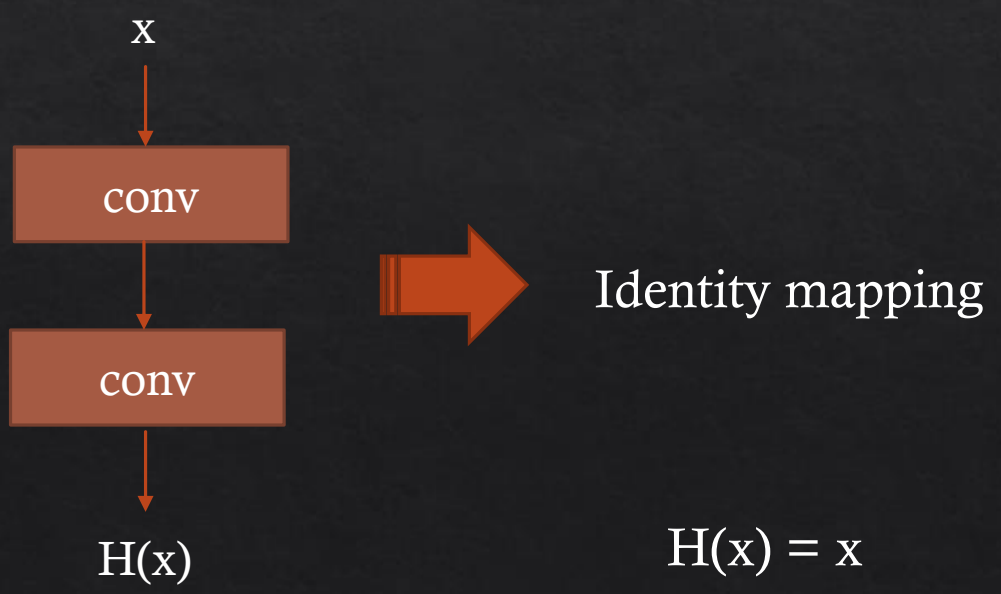


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

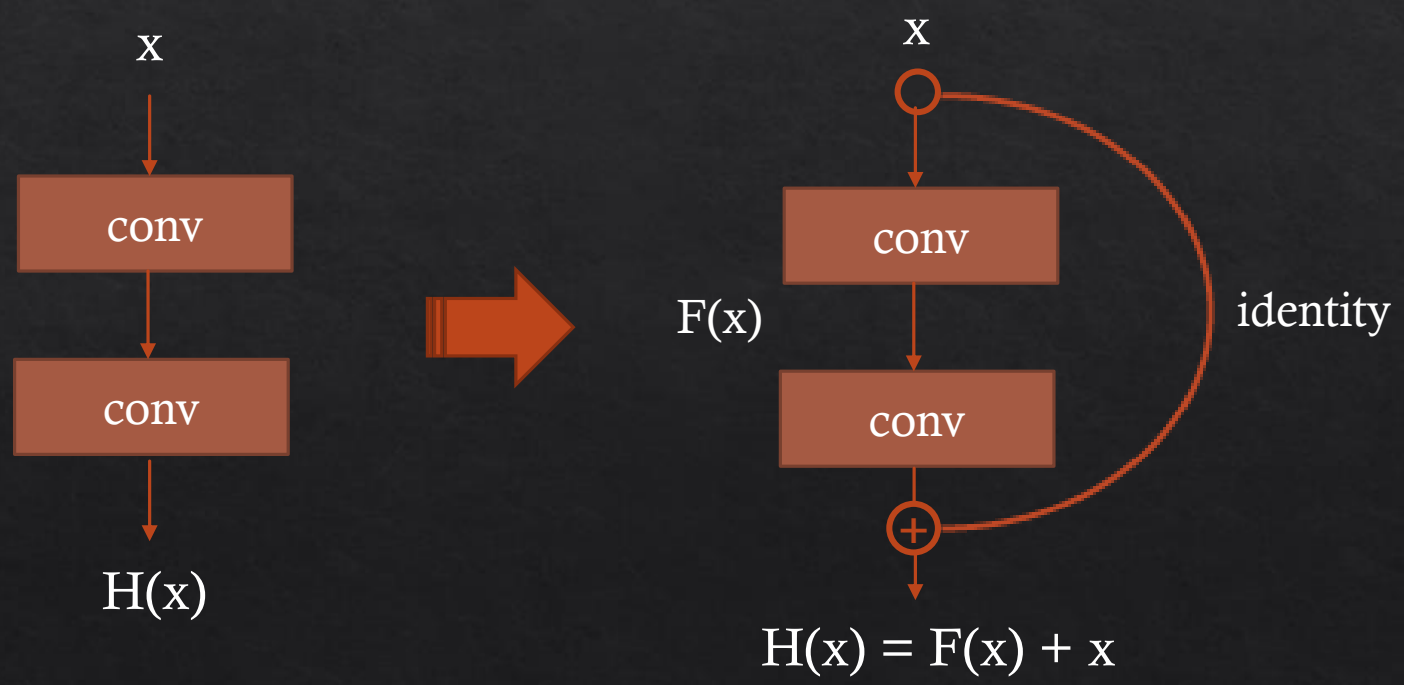
Degradation Problem

- Degradation problem : with the network depth increasing, accuracy gets saturated
- If the added layers are identity mapping, and the other layers are copied from the learned shallower model. -> less or equal training error
- The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize.

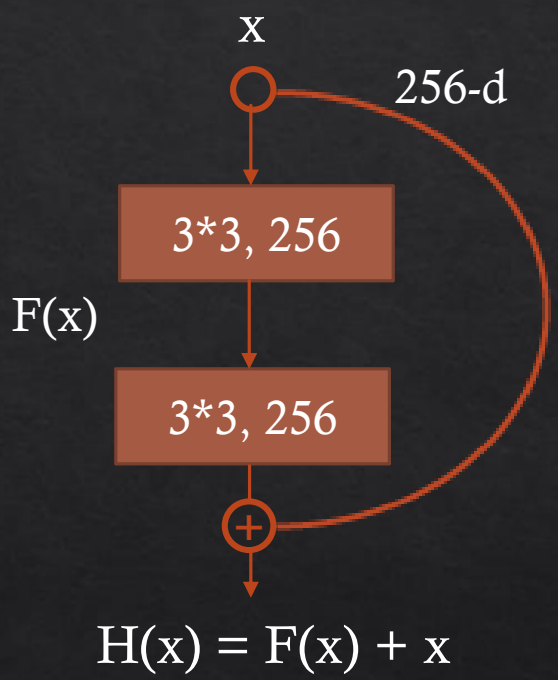


Residual Learning

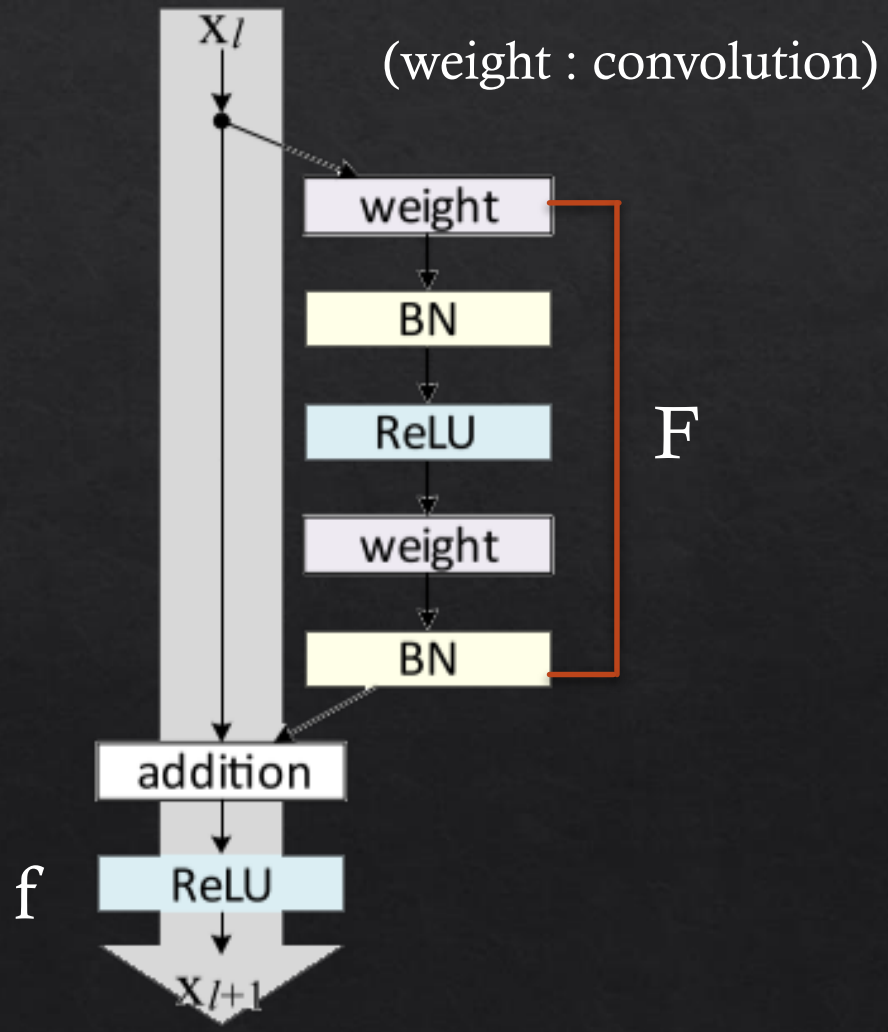
Add a shortcut from previous output -> easy to learn identity mapping



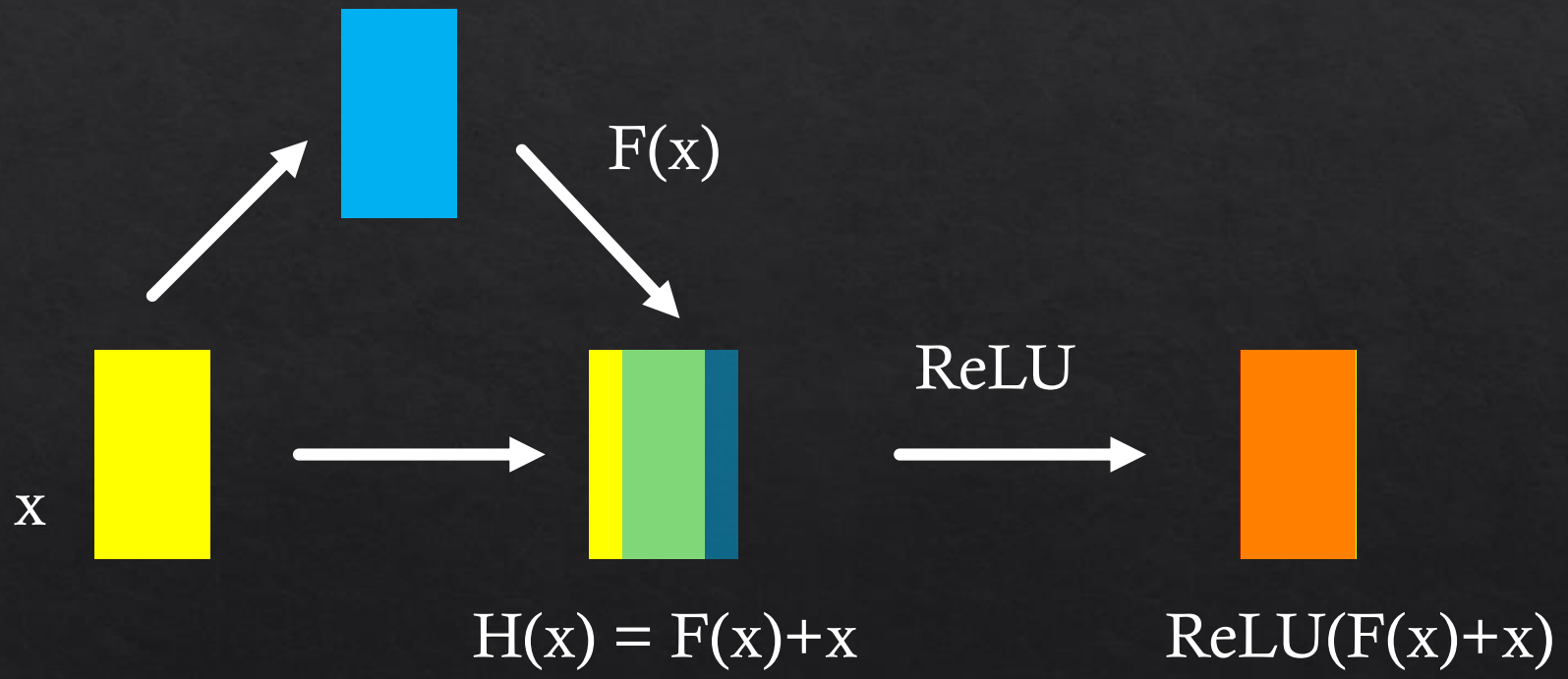
Residual block



inner design



Residual block

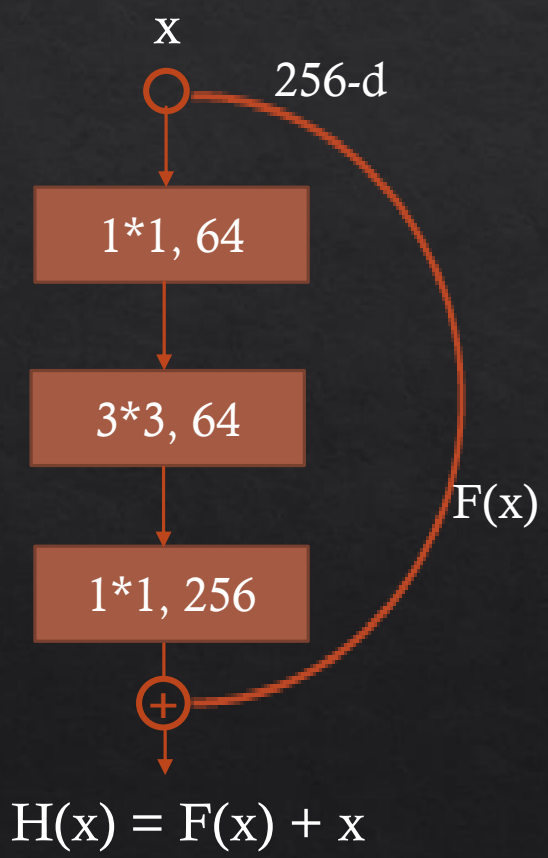


Bottleneck design

For deeper model (layers>50)

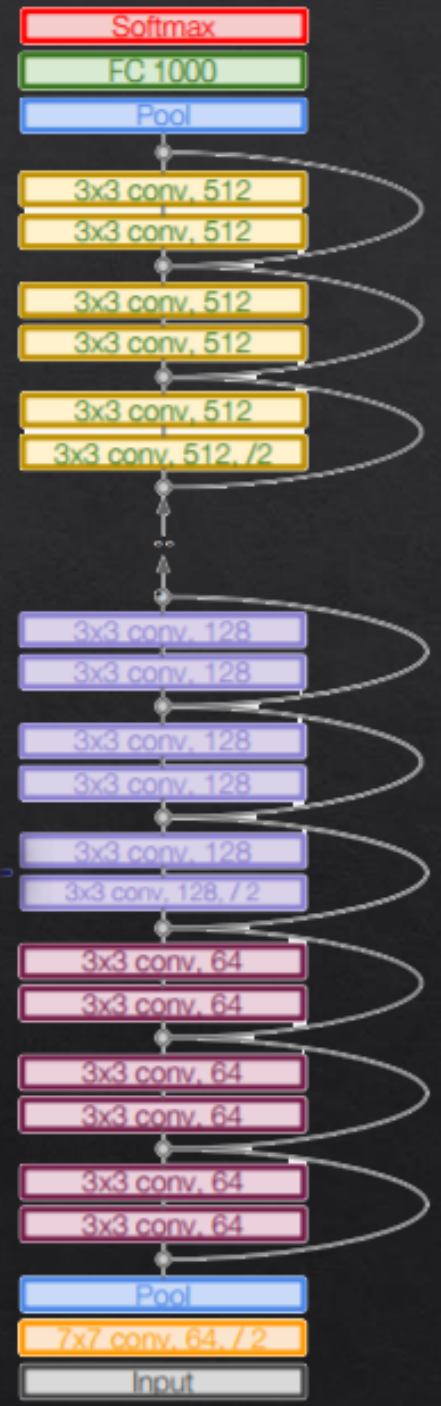
Reduce computing cost

Similar to Inception model



Layer design

- Shortcut connects between 2 residual block
- Double # of filters and downsample (stride 2) between different stage
- The model ends with a global average pooling and fully connected layer



Model Architecture

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

Experiment setting

- Optimizer: SGD with momentum of 0.9
- Learning rate: starts from 0.1 and is divided by 10 when the error plateaus
- Weight decay of 0.0001
- No Dropout
- Dataset: ImageNet
- Iteration:60×10e4 iterations.
- Plain network: same architecture with ResNet, but without shortcut connection

Experiment Result

- Plain network vs ResNet

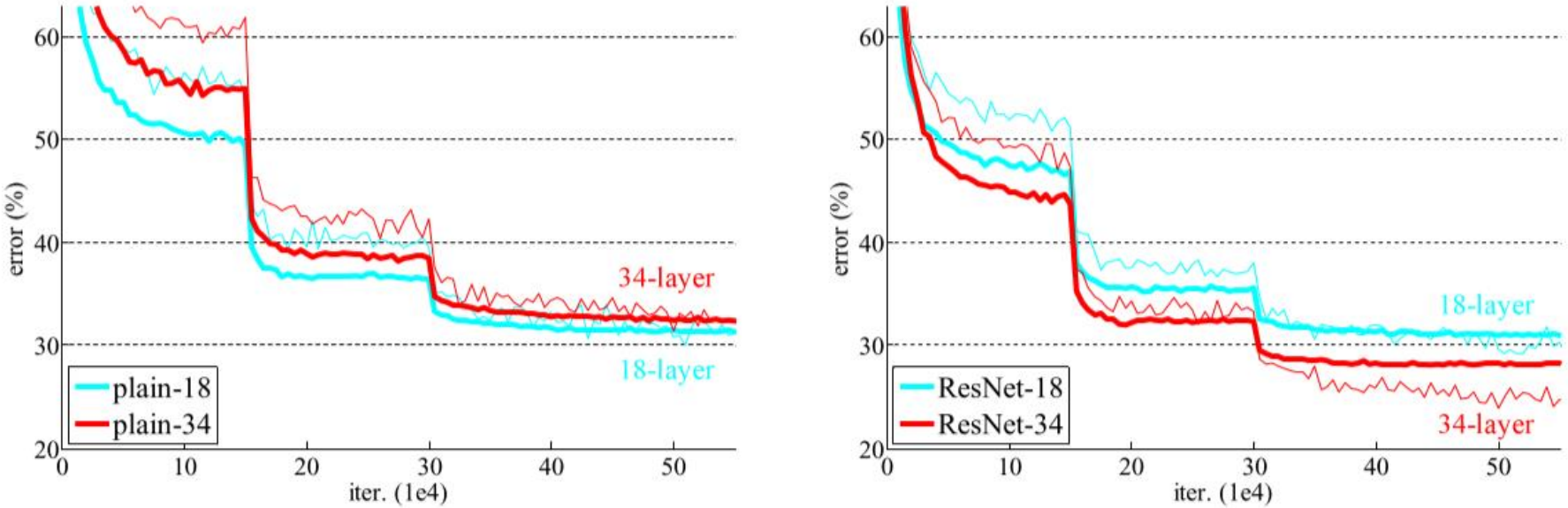


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

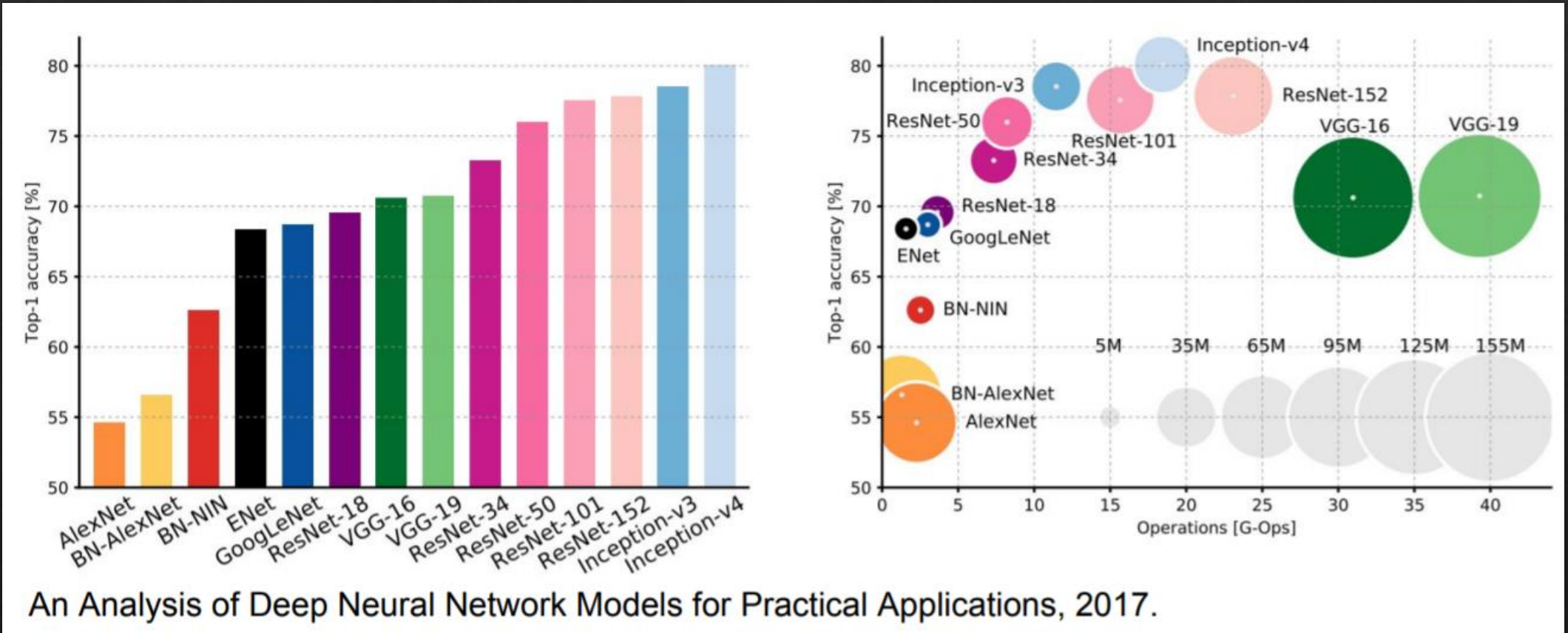
Experiment Result

- ResNet compare with other models

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

Computation cost



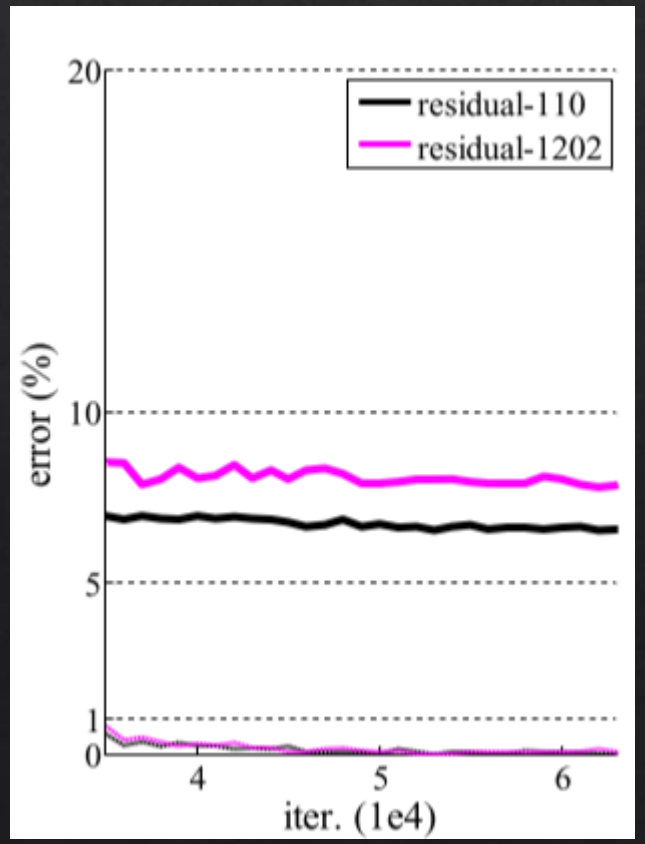
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Deeeeeeeeeper model

- Error rate of ResNet1202 increased -> may be overfitting

method			error (%)
Maxout [10]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 (7.72±0.16)
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Table 6. Classification error on the **CIFAR-10** test set. All methods are with data augmentation. For ResNet-110, we run it 5 times and show "best (mean±std)" as in [43].

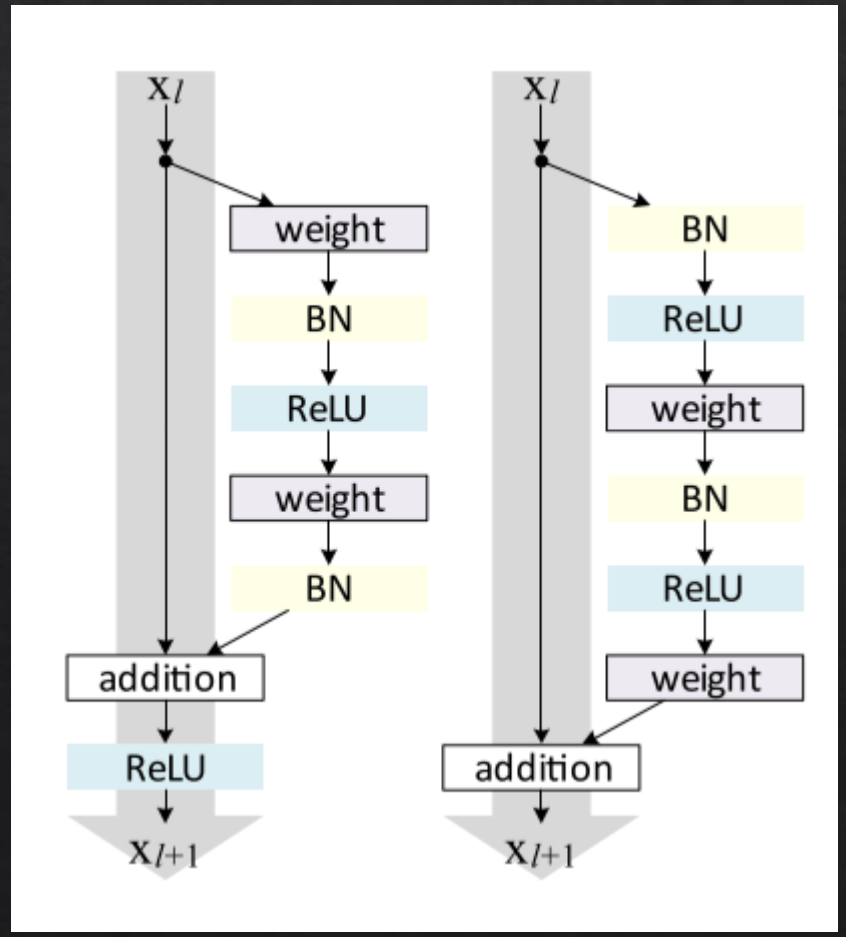


Summary

- Through residual learning, they make model be easier to learn identity mapping.
- Improve performance on different dataset.(ILSVRC 2015 champion)
- Improve gradient flow.
- Cost efficiency.

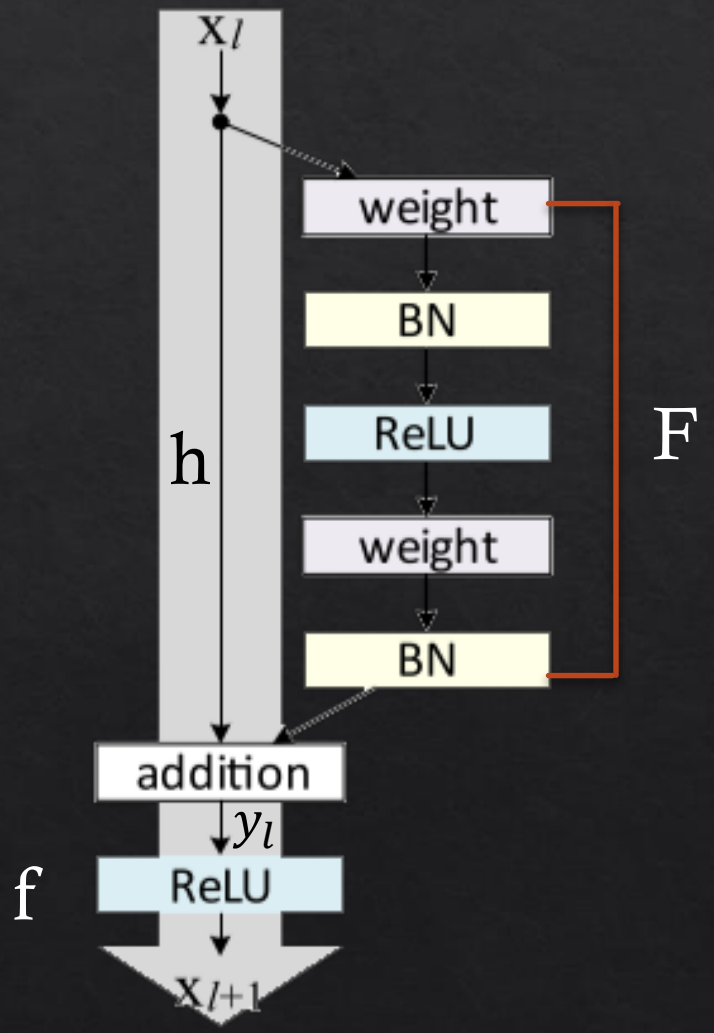
Identity Mappings in Deep Residual Networks

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016)



Not only identity in residual block, but also through the network

Original Residual Unit



$$y_l = h(x_l) + F(x_l, W_l)$$

$$x_{l+1} = f(y_l)$$

Let $h(x_l) = x_l$, we can get identity mapping in residual block.

Direct path through the entire network

$$y_l = h(x_l) + F(x_l, W_l)$$

$$x_{l+1} = f(y_l)$$

Let $h(x_l) = x_l, f(y_l) = y_l$
we can get identity mapping
through whole model.

Theory

$$x_{l+1} = f(h(x_l) + F(x_l, W_l))$$

$$h(x_l) = x_l$$
$$f(y_l) = y_l$$

$$x_{l+1} = x_l + F(x_l, W_l)$$

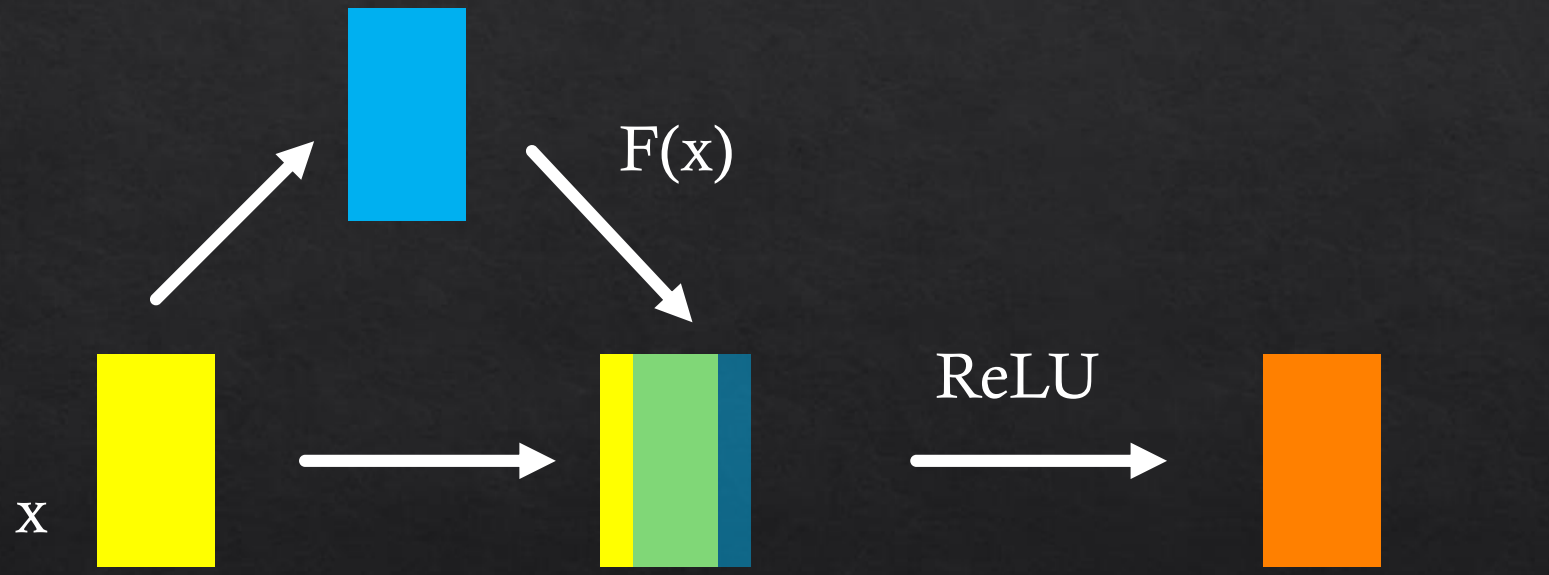
$$x_L = x_l + \sum_{i=l}^{L-1} F(x_l, W_l)$$

Benefit of direct path through the entire network

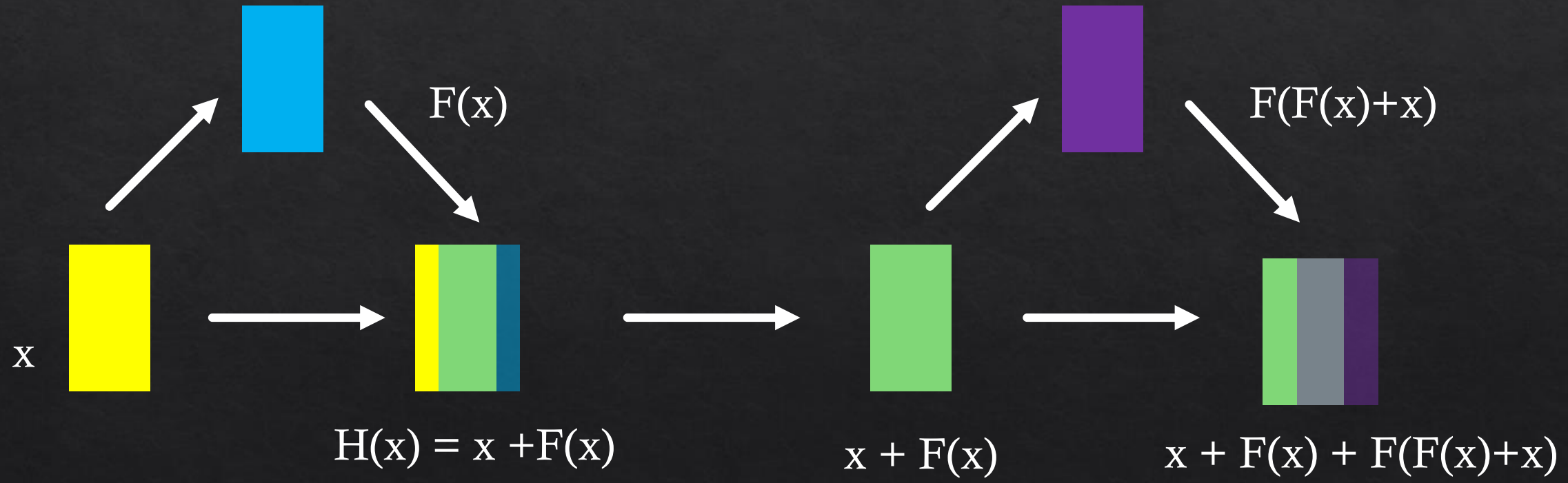
$$x_L = x_l + \sum_{i=l}^{L-1} F(x_l, W_l) \qquad \frac{\partial \varepsilon}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} F(x_l, W_l)\right)$$

- The feature of any deeper unit L can be represented as the feature of any shallower unit l plus a residual function.
- The information $(\frac{\partial \varepsilon}{\partial x_L})$ can be directly propagated to any shallow unit l to avoid gradient vanishing.

Residual block

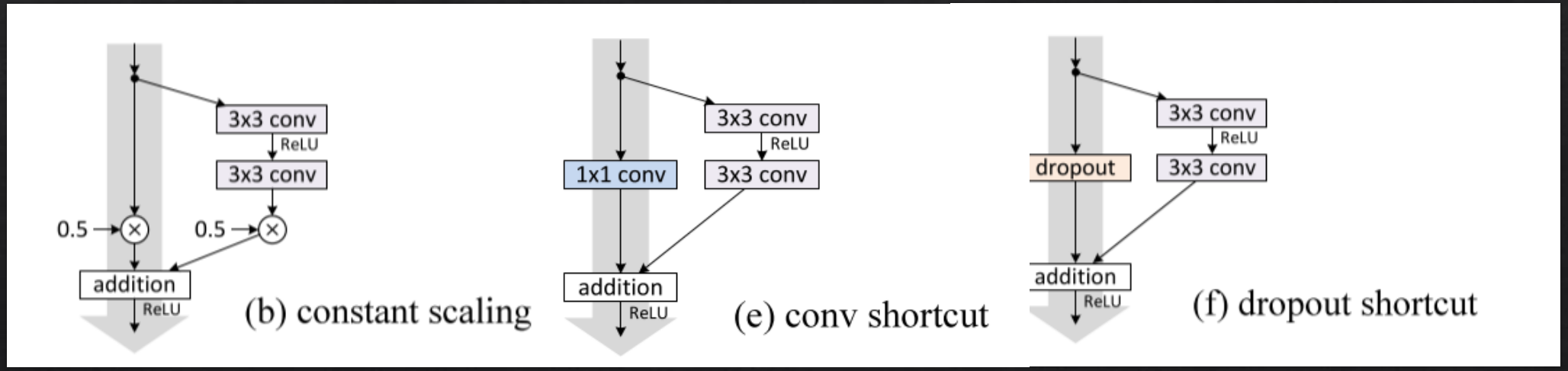


Direct path through the entire network



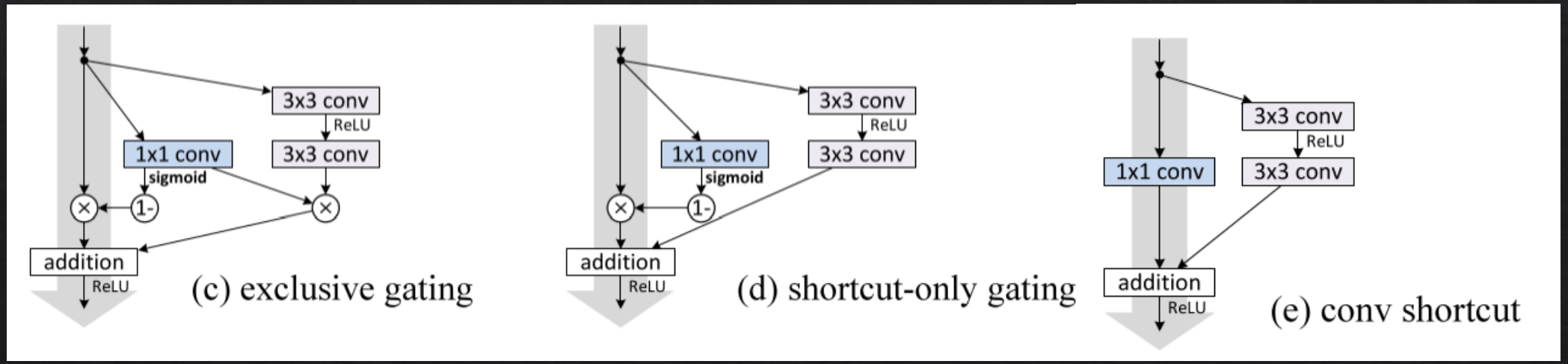
Importance of Identity Skip Connections

- Destroy Identity : Multiplicative manipulations (scaling, gating, 1×1 convolutions, and dropout) on the shortcuts can hamper information propagation and lead to optimization problems.



Importance of Identity Skip Connections

- These models have stronger representational abilities than identity shortcuts and cover the solution space of identity shortcuts. But still get higher training error. The degradation is caused by optimization issues, instead of representational abilities.



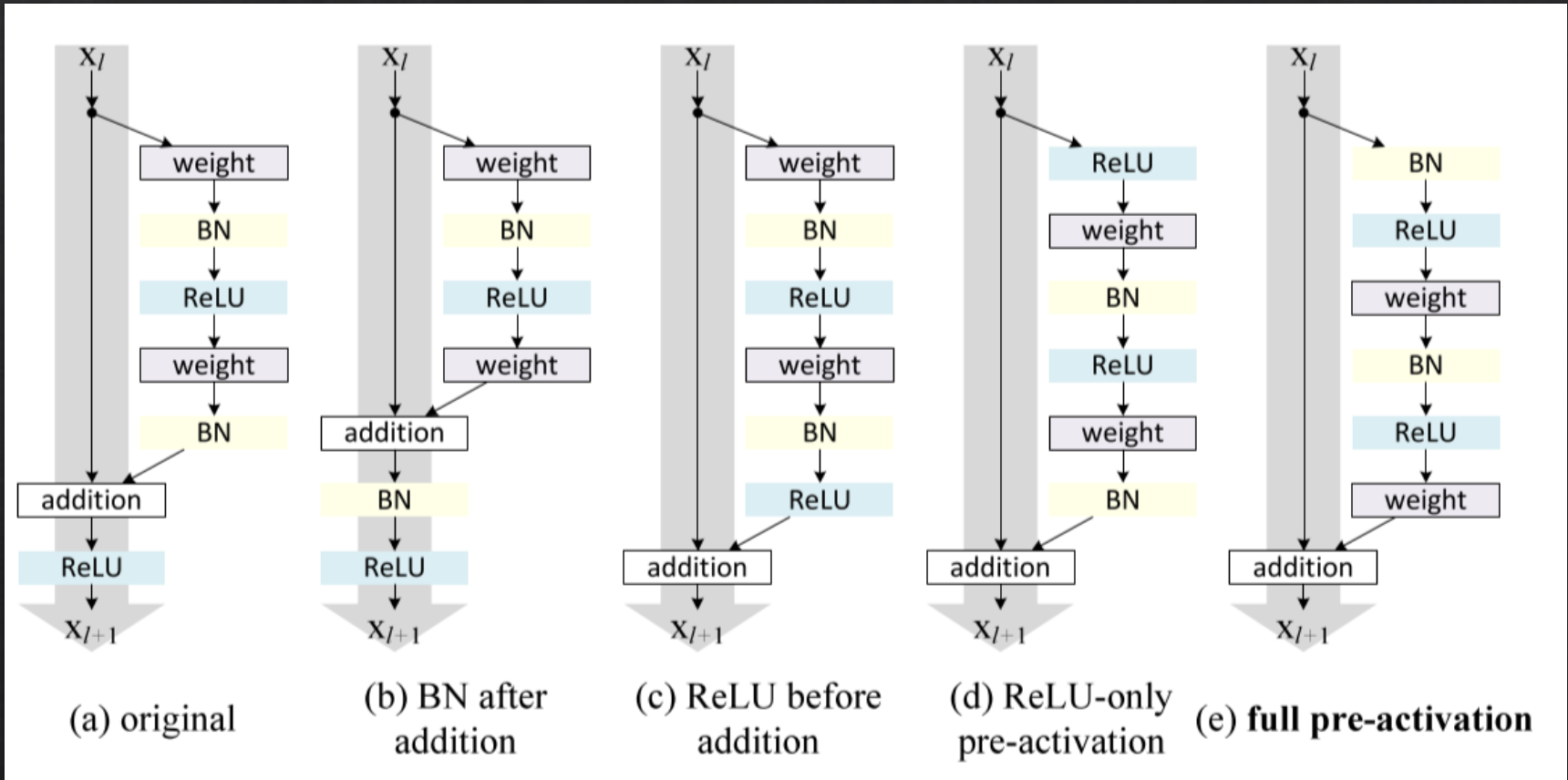
Importance of Identity Skip Connections

Table 1. Classification error on the CIFAR-10 test set using ResNet-110 [1], with different types of shortcut connections applied to all Residual Units. We report "fail" when the test error is higher than 20%.

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	

* fail: test error is higher than 20%.

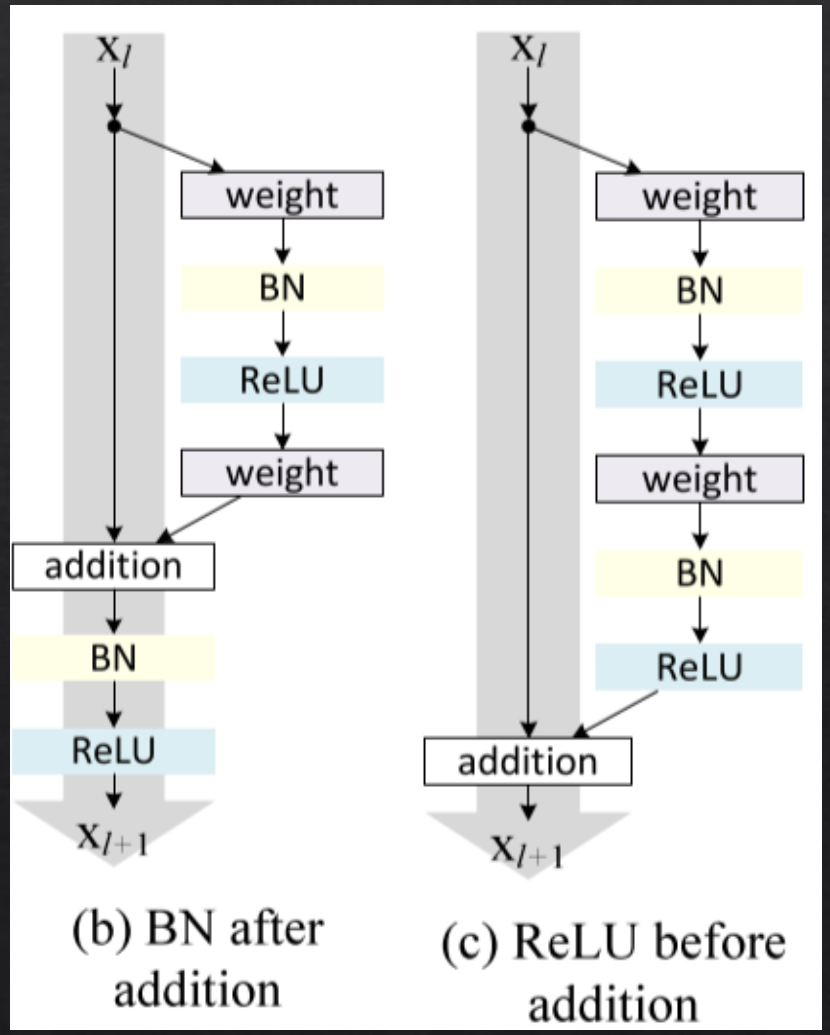
Usage of Activation Functions



Usage of Activation Functions

- (b) BN impedes information propagation

- (c) ReLU cause non-negative output from the transform F. While intuitively a "residual" function should take values in $(-\infty, +\infty)$.



Usage of Activation Functions

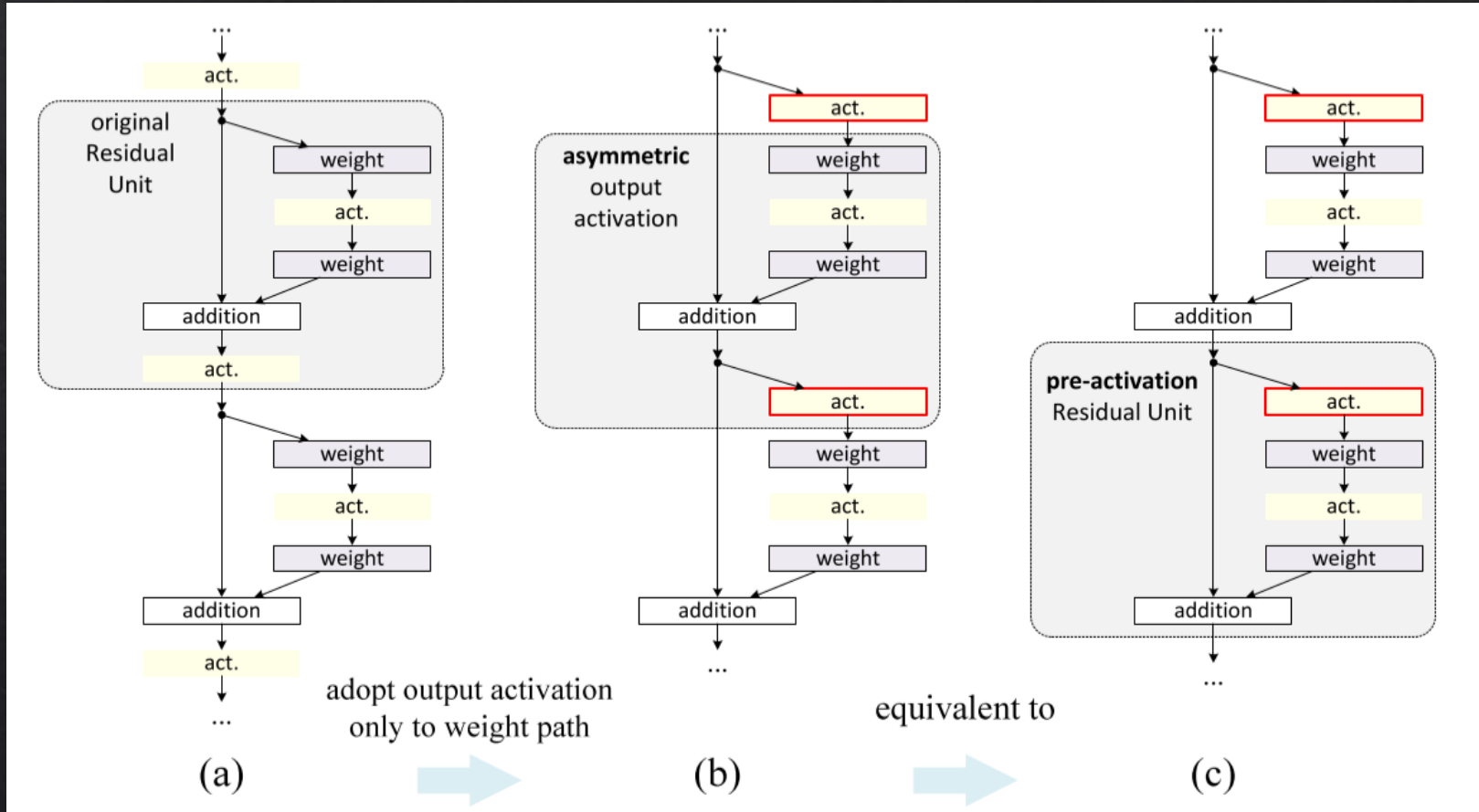
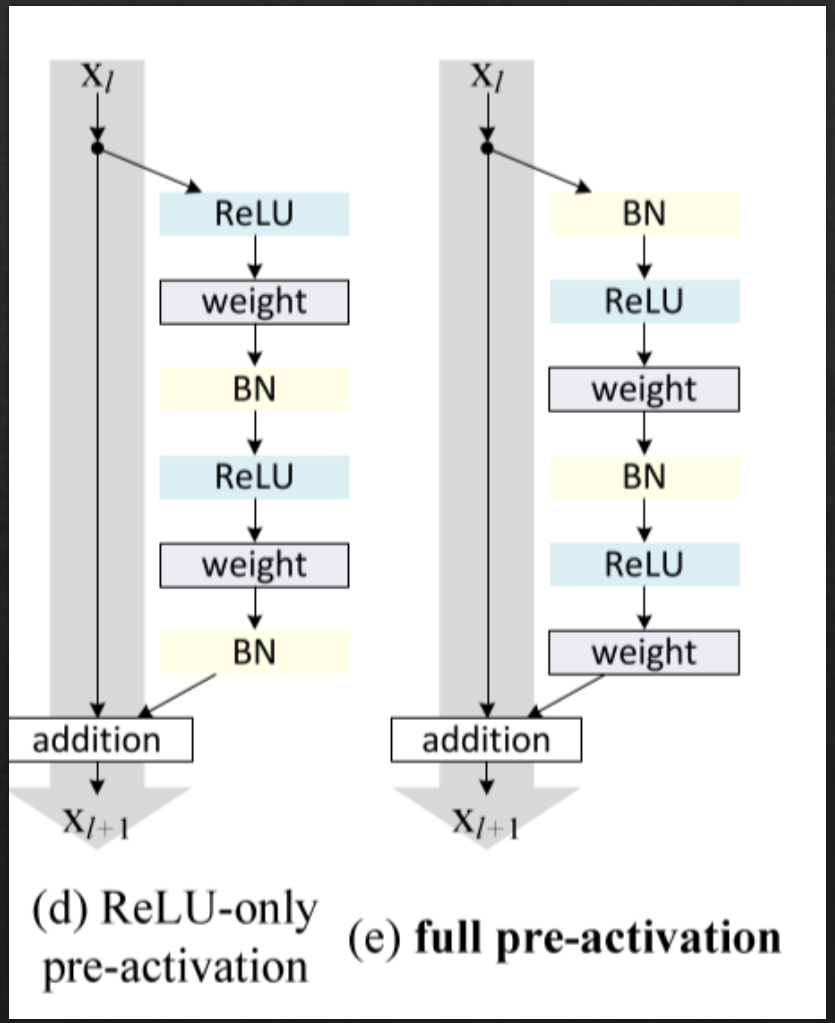


Figure 5. Using asymmetric after-addition activation is equivalent to constructing a *pre-activation* Residual Unit.

Usage of Activation Functions

- (d) Using previous after-addition activation is equivalent to constructing a pre-activation Residual Unit.

- (e) After addition, the merged signal is not normalized. On the contrary, in pre-activation version, the inputs to all weight layers have been normalized



Usage of Activation Functions

Table 2. Classification error (%) on the CIFAR-10 test set using different activation functions.

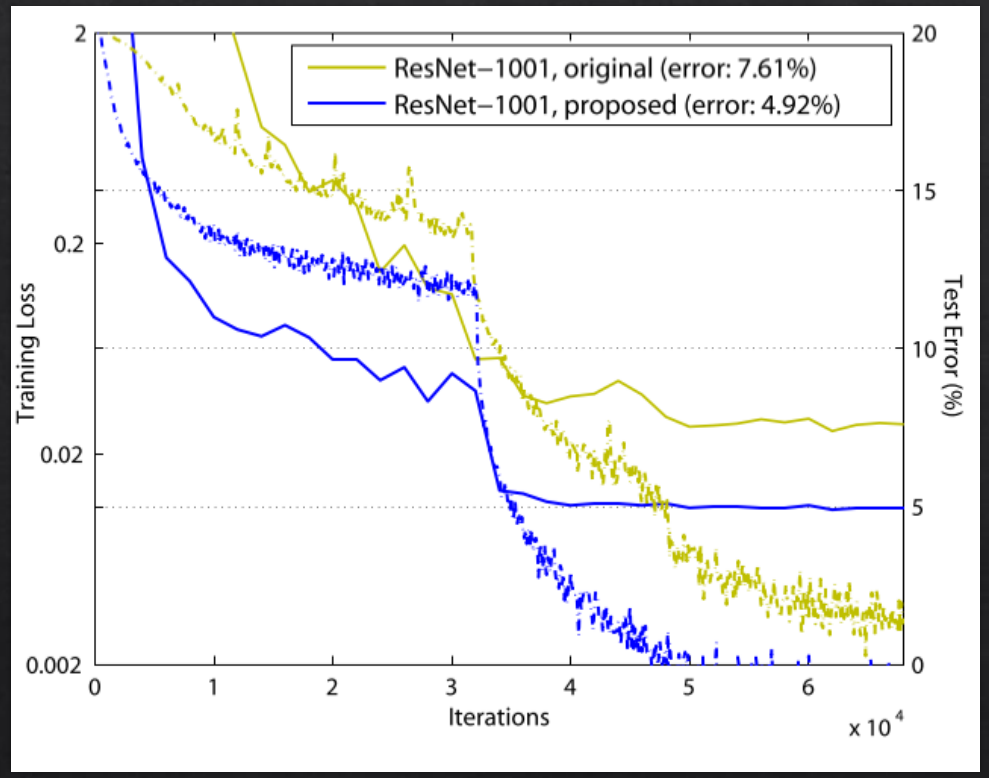
case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**

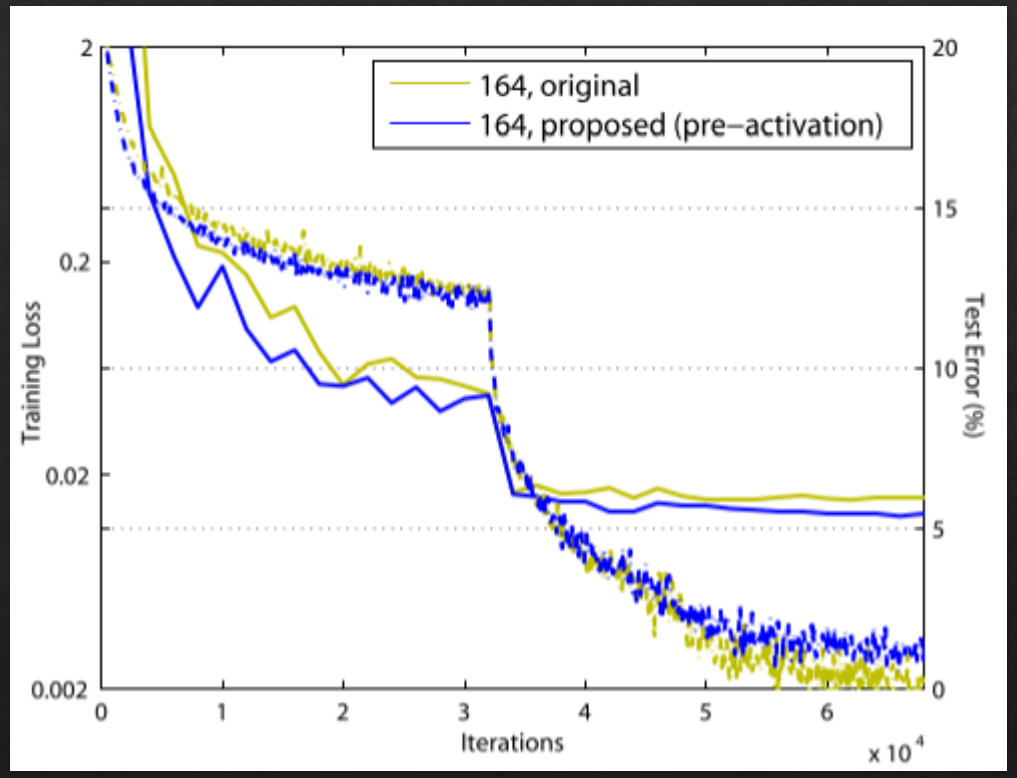
Experiment Result

Table 3. Classification error (%) on the CIFAR-10/100 test set using the original Residual Units and our pre-activation Residual Units.

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 - Ease of optimization





Training curves on CIFAR-10. Solid lines denote test error, and dashed lines denote training loss.

Analysis - Ease of optimization

- Because of identity mapping, the signal can be propagated directly between any two units. 1001-layer network reduces the training loss very quickly.

- Observation: Shallow network only suffer a little bit in the beginning, but goes into a healthy status soon.
  - After some training, the weights are adjusted into a status such that y_l is more frequently above zero and f (ReLU) does not truncate it. The truncation, is more frequent when there are 1000 layers.

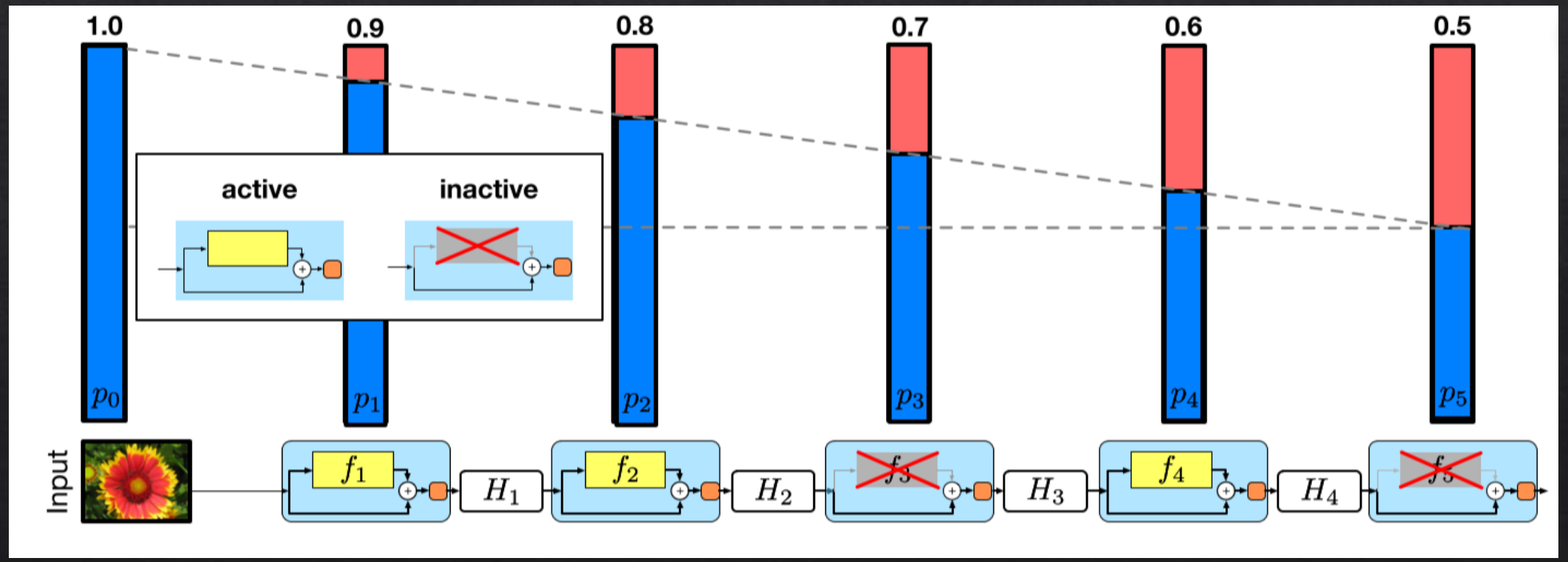
Analysis - Reducing overfitting

- The pre-activation version reaches slightly higher training loss at convergence, but produces lower test error.

- In the original Residual Unit, the signal is soon added to the shortcut and thus the merged signal is not normalized. However, in pre-activation version, the inputs to all weight layers have been normalized by BN.

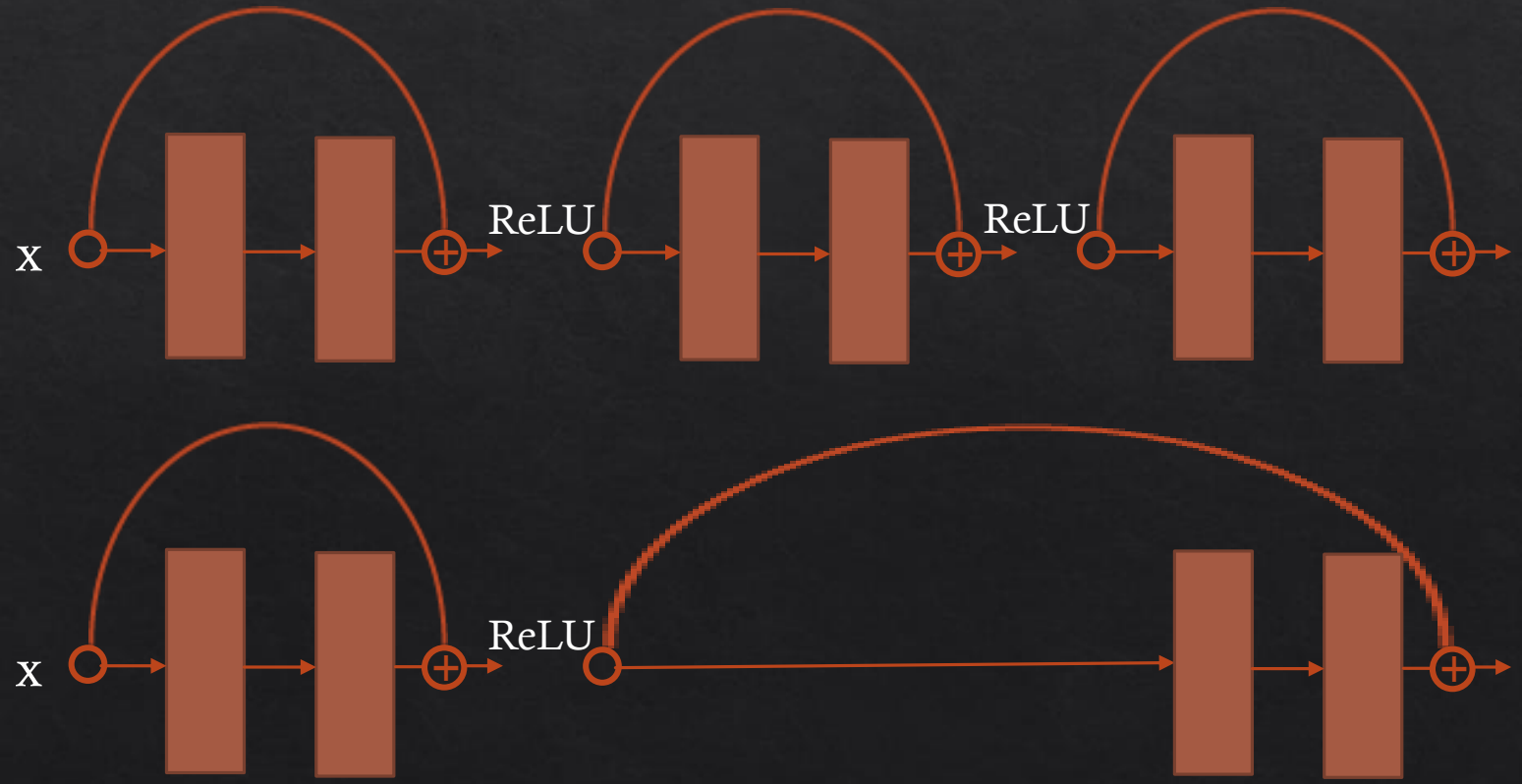
Deep Networks with Stochastic Depth

- Gao Huang*, Yu Sun*, Zhuang Liu†, Daniel Sedra, Kilian Q. Weinberger (2016)



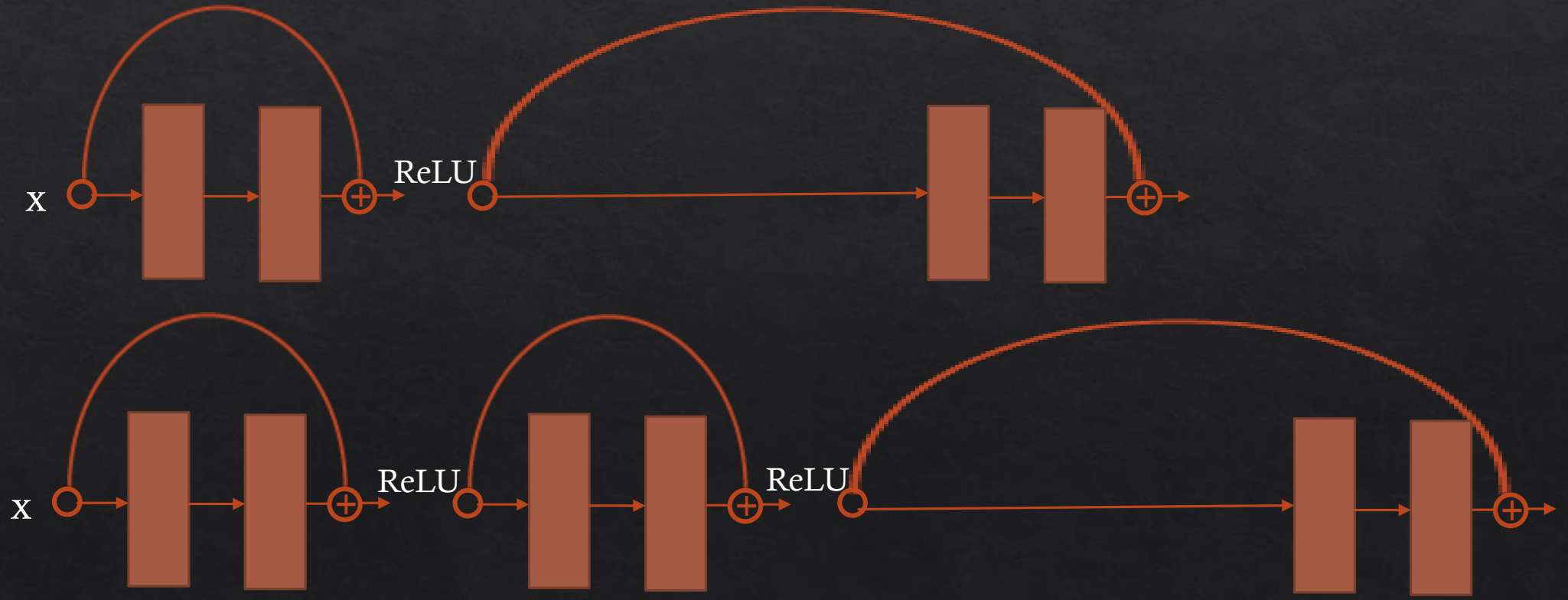
Randomly drop layers

- Shorten the network significantly by randomly removing layers independently for each mini-batch.



Randomly drop layers

- Ensemble of networks of different depths



Advantages

- Training time savings
  - Shorter path with expectation
- Higher Accuracy
  - Implicit model ensemble
- Regularization
  - Training an ensemble of networks, but with different depths, possibly achieving higher diversity among ensemble members.

Densely Connected Convolutional Networks

- Gao Huang, Zhuang Liu, Kilian Q. Weinberger, Laurens van der Maaten (CVPR 2017)

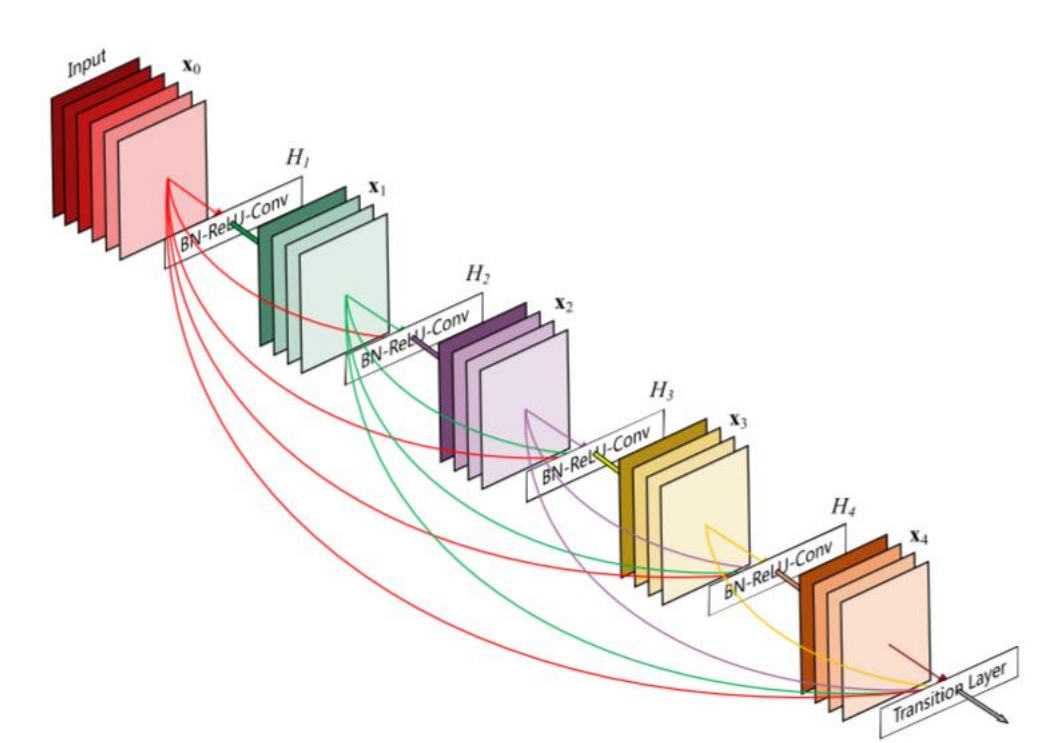


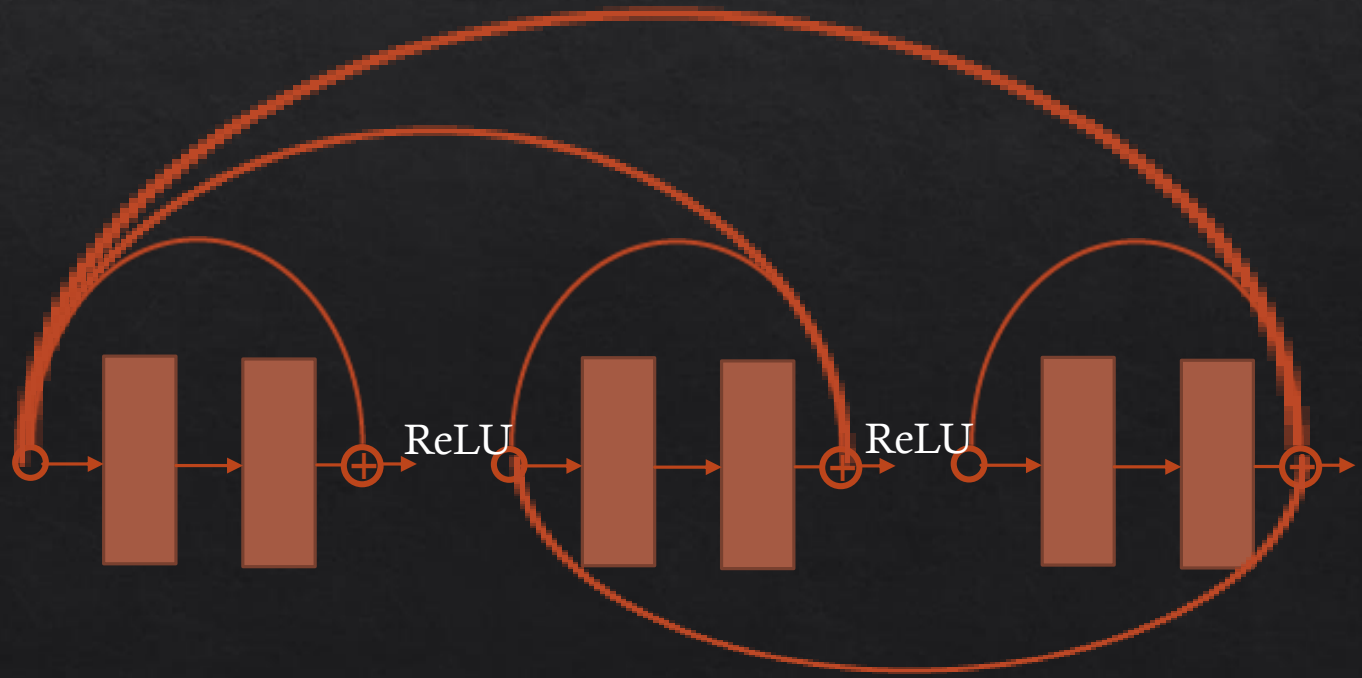
Figure 1. A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

To ensure maximum information flow, we connect all layers directly.

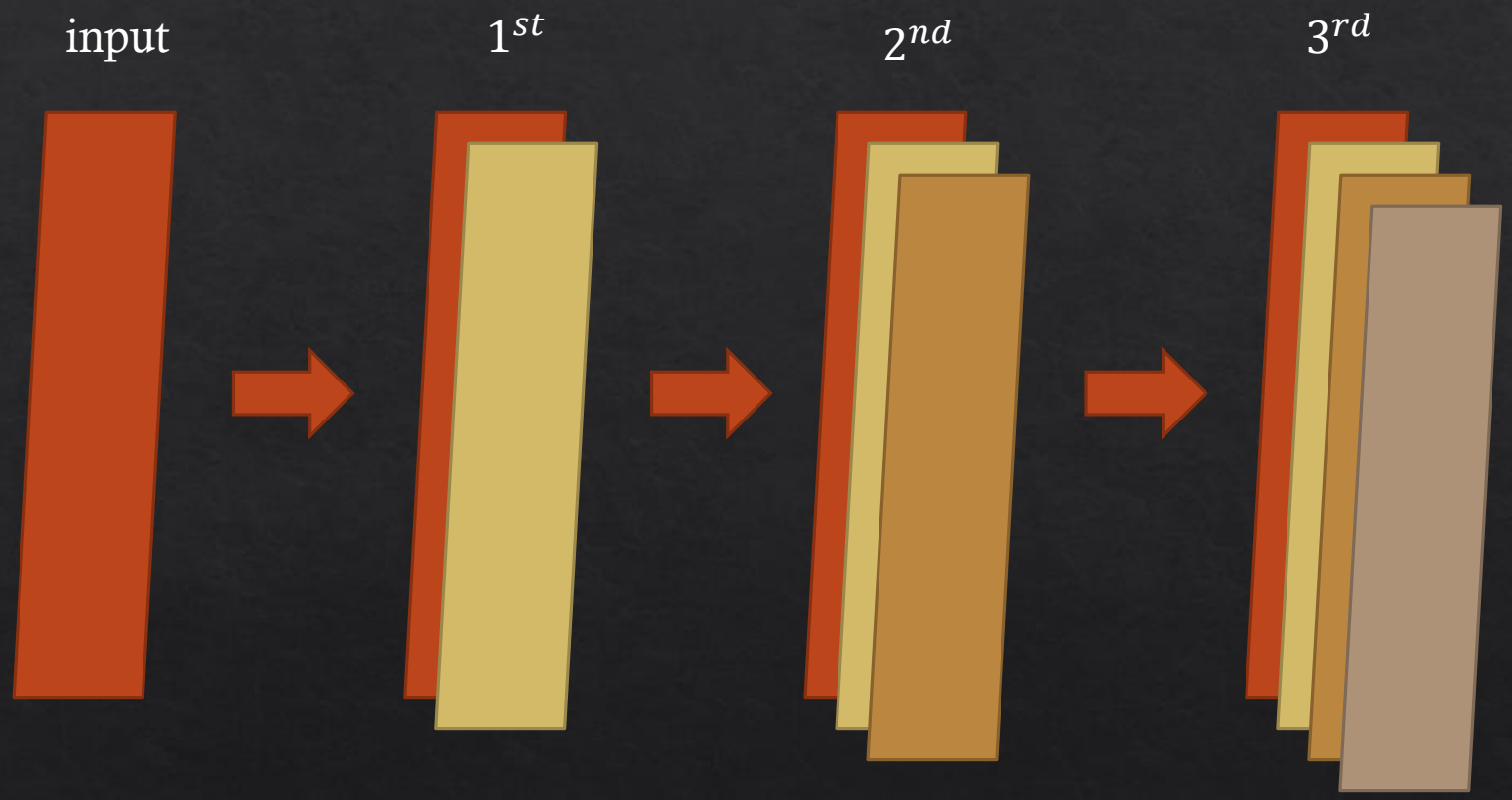
Dense connectivity

Stochastic Depth

DenseNet



Dense connectivity



Dense connectivity

- Concatenate all the features before this layer. Hence, the l^{th} layer has l inputs, consisting of the feature-maps of all preceding convolutional blocks.
- ResNet: $x_l = H(x_{l-1}) + x_{l-1}$

  DenseNet: $x_l = H([x_0, x_1, \dots, x_{l-1}])$
- Each layer has direct access to the gradients from the loss function, make gradient flow improved.

Layer design

- Conv block: BN -> ReLU -> 3*3 Conv
- Transition block: BN -> ReLU ->1*1 Conv -> 2*2 AveragePooling
  - Change feature map size

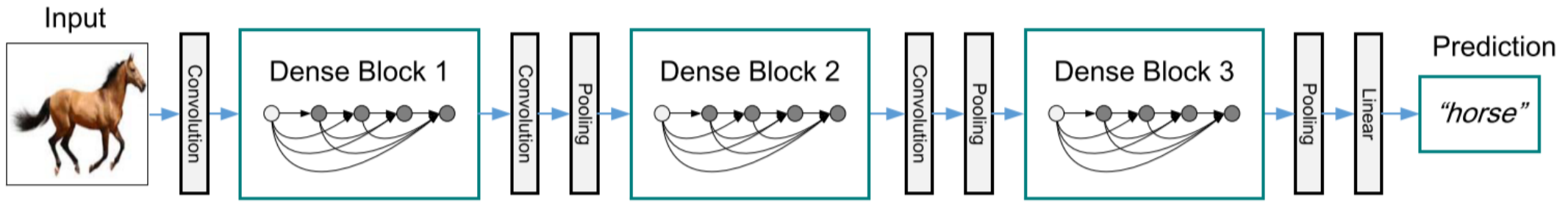
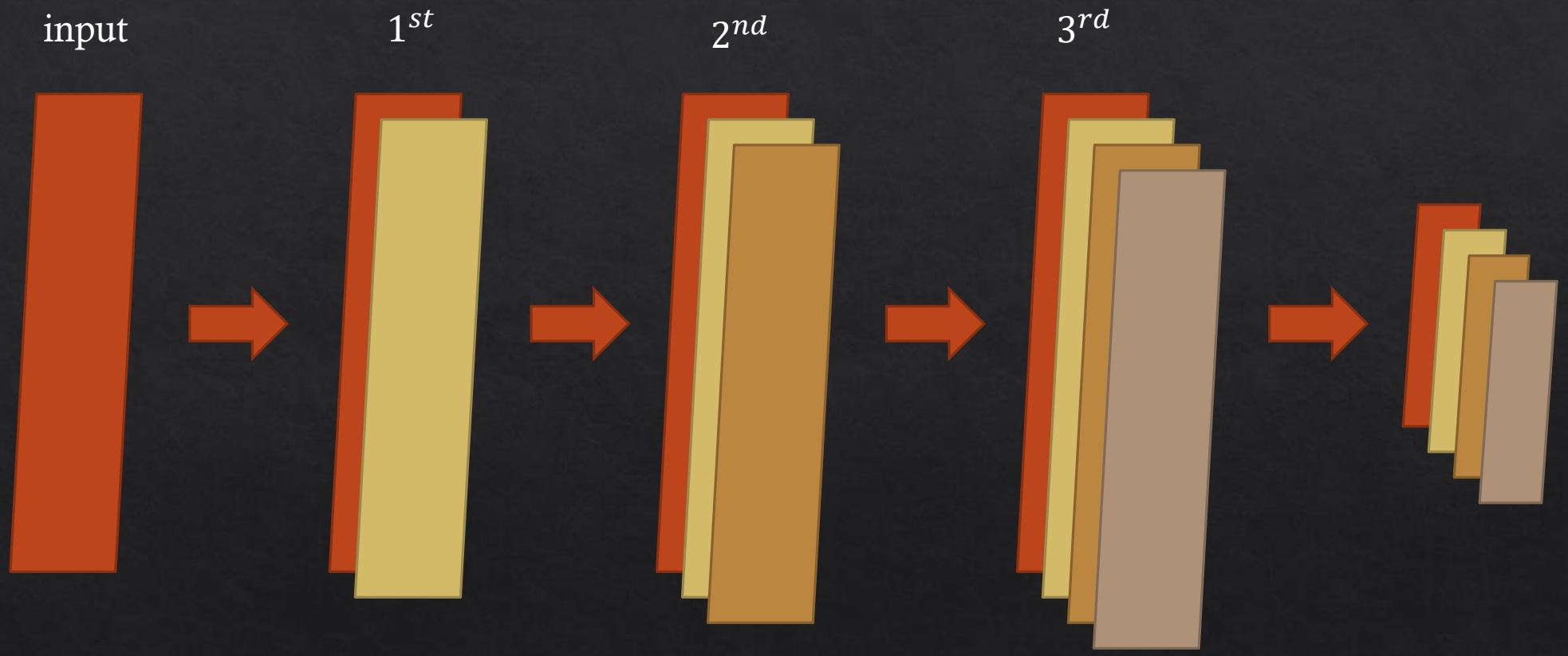


Figure 2. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature map sizes via convolution and pooling.

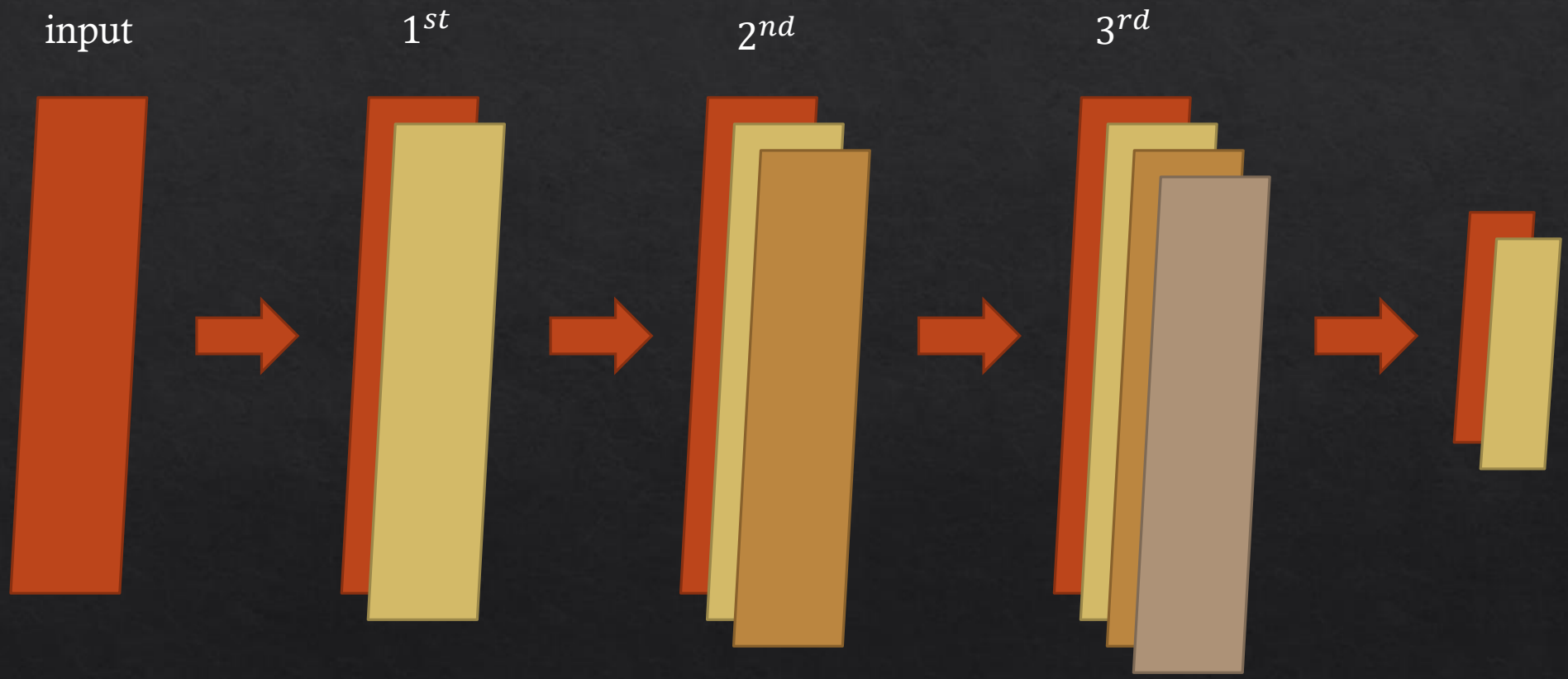
Transition



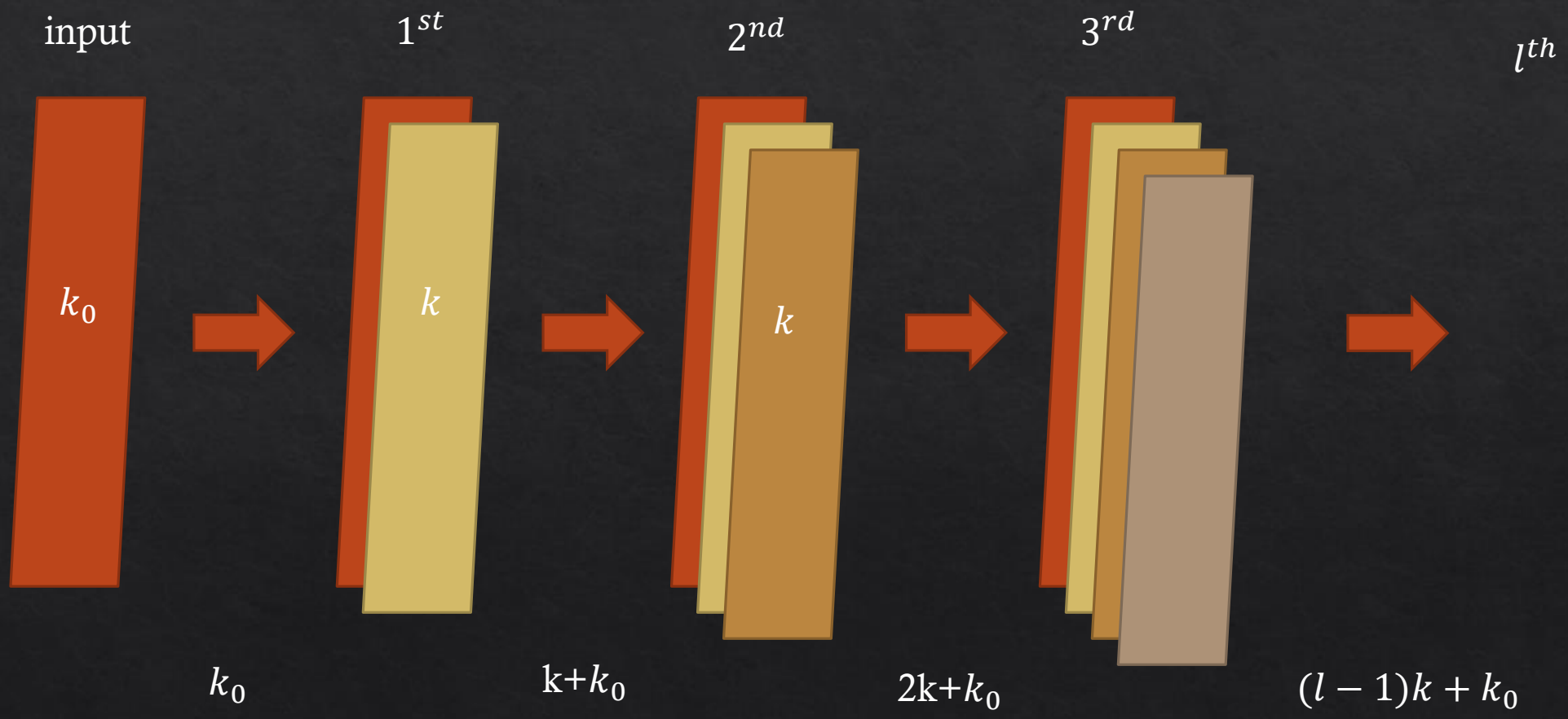
Variation design

- Bottleneck design(DenseNet-B): BN -> ReLU -> 1*1 Conv -> BN -> ReLU -> 3*3 Conv
  - Reduce computing cost
- Compression design(DenseNet-C): Reduce the number of feature-maps at transition layers.
  - If a dense block contains m feature-maps, we let the following transition layer generate θm output feature-maps, where $0 < \theta \leq 1$

DenseNet-C



Growth rate



Growth rate

- growth rate : filters number produced in each layer.
- If each function H_l produces k feature maps as output, it follows that the l th layer has $k \times (l-1) + k_0$ input feature-maps, where k_0 is first conv filters number.
- Limit k to a small integer to improve parameter efficiency.
- Explanation: Every layer contribute k feature-maps as new information, we can see the concatenative feature-maps as global “collective knowledge.”

Model Architecture

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 36
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 24
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 1. DenseNet architectures for ImageNet. The growth rate for the first 3 networks is $k = 32$, and $k = 48$ for DenseNet-161. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

CIFAR Experiment

- Improve accuracy: C10: 3.46%, C100: 17.18%
- Parameter efficiency
  - DenseNet-BC(l=100,k=12): C10: 4.51%, C100: 22.27%, parameter #:0.8M
  - ResNet-1001(pre-activation): C10: 4.62%, C100: 22.71%, parameter #:10.2M
- Regularization : without data augmentation, DenseNet particularly outperforms other architectures.

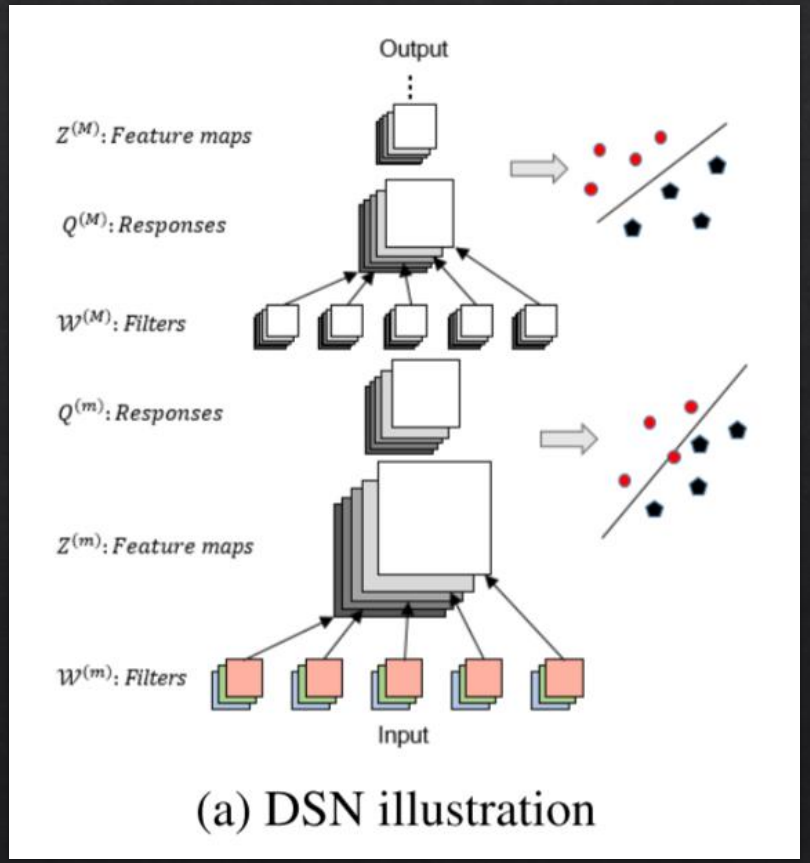
Experiment Result

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [31]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [33]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [41]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

Table 2. Error rates (%) on CIFAR and SVHN datasets. L denotes the network depth and k its growth rate. Results that surpass all competing methods are **bold** and the overall best results are **blue**. "+" indicates standard data augmentation (translation and/or mirroring). $*$ indicates results run by ourselves. All the results of DenseNets without data augmentation (C10, C100, SVHN) are obtained using Dropout. DenseNets achieve lower error rates while using fewer parameters than ResNet. Without data augmentation, DenseNet performs better by a large margin.

Analysis - Implicit Deep Supervision

- DSN(deeply-supervised nets[8]): classifiers attached to every hidden layer, enforcing the intermediate layers to learn discriminative features.



(a) DSN illustration

Analysis - Implicit Deep Supervision

- Every individual layers receive additional supervision from the loss function through the shorter connection.
- DenseNets perform a similar deep supervision : a single classifier on top of the network provides direct supervision to all layers through at most two or three transition layers.

Analysis - Parameter efficiency

- As a direct consequence of the input concatenation, it encourages feature reuse throughout the network.
- Hence, each layer only need to produces small number of feature-maps. DenseNet will collect every feature-maps as global state.

Analysis - Parameter efficiency

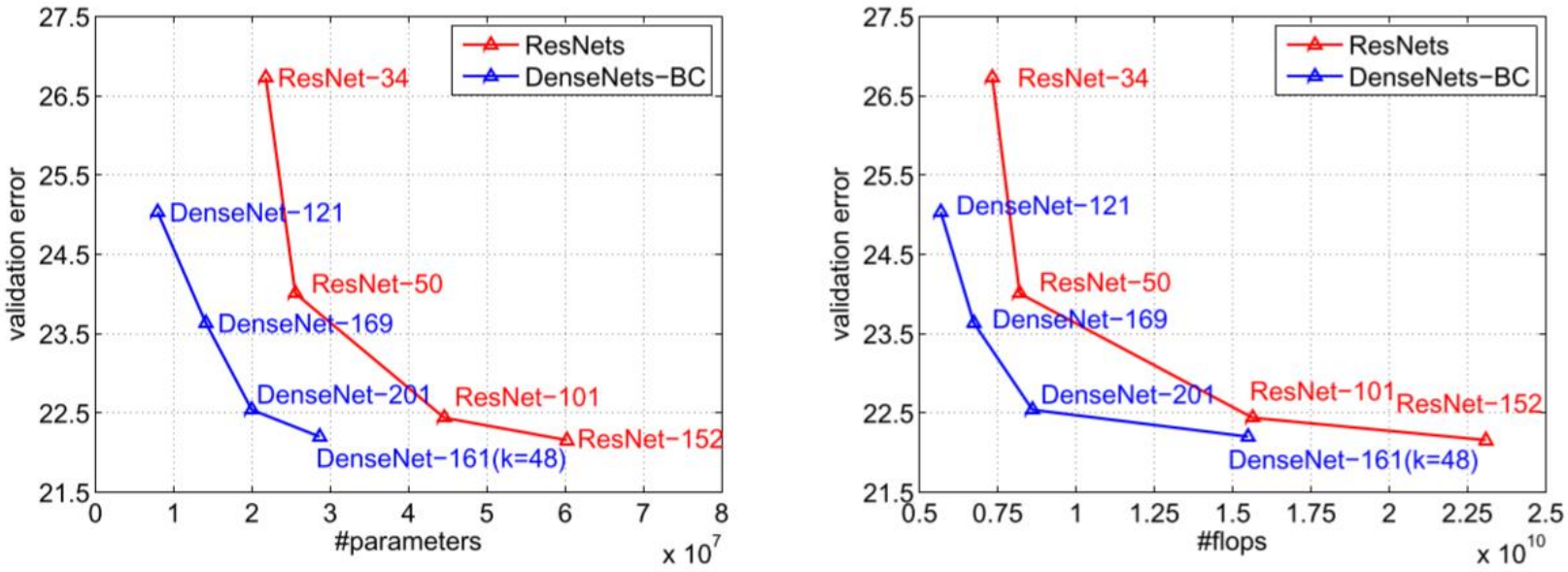


Figure 3. Comparison of the DenseNet and ResNet Top-1 (single model and single-crop) error rates on the ImageNet classification dataset as a function of learned parameters (*left*) and flops during test-time (*right*).

Analysis - Regularization

- Stochastic depth regularization of residual networks: layers are randomly dropped, which creates direct connections between layers. It provide regularization effect.
- DenseNet has similar connective pattern as Stochastic depth regularization, but more deterministic.
- DenseNet interpretation of stochastic depth may provide insights into the success of this regularizer.

Analysis - Feature Reuse

- With DenseNet on C10+ with L=40 and k=12, for each convolutional layer l within a block, we compute the average (absolute) weight assigned to connections with layer s.
- An red dot in position (l, s) indicates that the layer l makes, on average, strong use of feature maps produced s-layers before.

Analysis - Feature Reuse

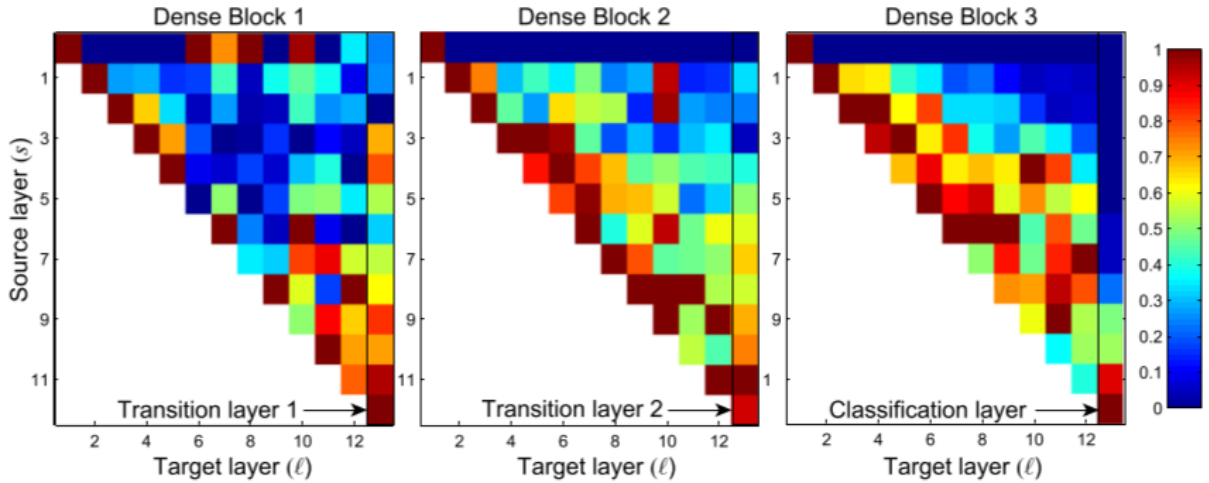
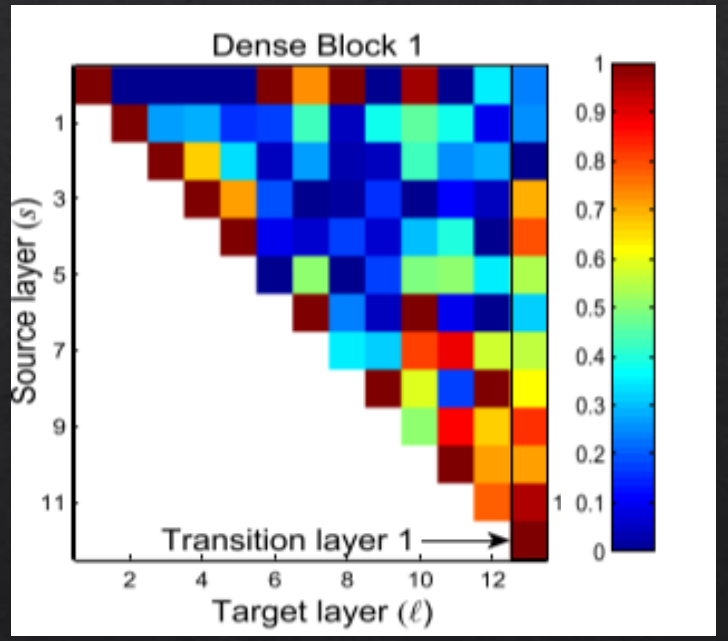


Figure 5. The average absolute filter weights of convolutional layers in a trained DenseNet. The color of pixel (s, ℓ) encodes the average $L1$ norm (normalized by the number of input feature maps) of the weights connecting convolutional layer s to layer ℓ within a dense block. The three columns highlighted by black rectangles correspond to the two transition layers and the classification layer. The first row encodes those weights connected to the input layer of the dense block.

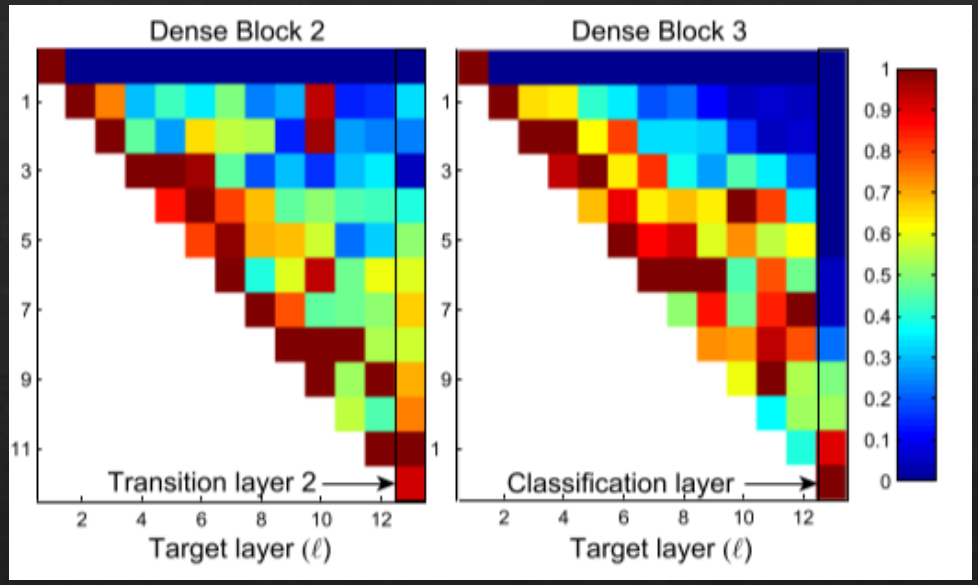
Analysis - Feature Reuse

- All layers spread their weights over many inputs within the same block. This indicates that features extracted by very early layers.



Analysis - Feature Reuse

- The layers in the second and third dense block assign the least weight to the outputs of the transition layer, indicating that the transition layer outputs many redundant feature.
- This is in keeping with the strong results of DenseNet-BC where exactly these outputs are compressed.



Summary

- DenseNet introduces direct connections between any two layers with the same feature-map size, it provides higher accuracy on several datasets
- DenseNet require substantially fewer parameters and less computation to achieve state-of-the-art performances.
- DenseNet has a strong regularization ability to avoid overfitting.

Reference

- [1] Deep Residual Learning for Image Recognition - https://arxiv.org/abs/1512.03385
- [2] Identity Mappings in Deep Residual Networks - https://arxiv.org/abs/1603.05027
- [3] Densely Connected Convolutional Networks - https://arxiv.org/abs/1608.06993
- [4] Deep Networks with Stochastic Depth - https://arxiv.org/abs/1603.09382
- [5] Stanford cs231n - http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf
- [6] Going Deeper with Convolutions - https://arxiv.org/abs/1409.4842
- [7] ResNet50 Architecture - http://ethereon.github.io/netscope/#/gist/db945b393d40bfa26006
- [8] Deeply-Supervised Nets - https://arxiv.org/abs/1409.5185
- [9] keras Resnet - https://github.com/fchollet/keras/blob/master/keras/applications/resnet50.py
- [10] DenseNets Lua implement - https://github.com/liuzhuang13/DenseNet

Thanks for listening