CSE5194: ResNet and ResNeXt

The Ohio State University
Zhengqi Dong
E-mail: dong.760@osu.edu
Date: 09/20/2019

The Ohio State University

Part1: (ResNet) Deep Residual Learning for Image Recognition

1. Introduction: Background and Key Contribution

- Background
  - Published in 2015 by Kaiming He, and etc, with 56517 citation so far
  - Won 1st place on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation at the ILSVRC & COCO 2015 competition.
  - Achieved 3.57% error on the ImageNet test dataset
  - A 28% improvement on the COCO obejct detection dataset
- Key contribution:
  - Solved the **degradation problem** -- with the network depth increasing, accuracy gets saturated and degraded rapidly.
  - The residual networks are easier to optimize and can gain higher accuracy as increased depth.

1. Introduction: Residual Neural Network

- Question: Shouldn't building better neural networks as easy as stacking more layers?

- Problem: Vanishing/Exploding gradients, degradation problem.
- Old Solution: Normalized initialization and normalized the intermediate layers
- New Solution: Residual Learning Block

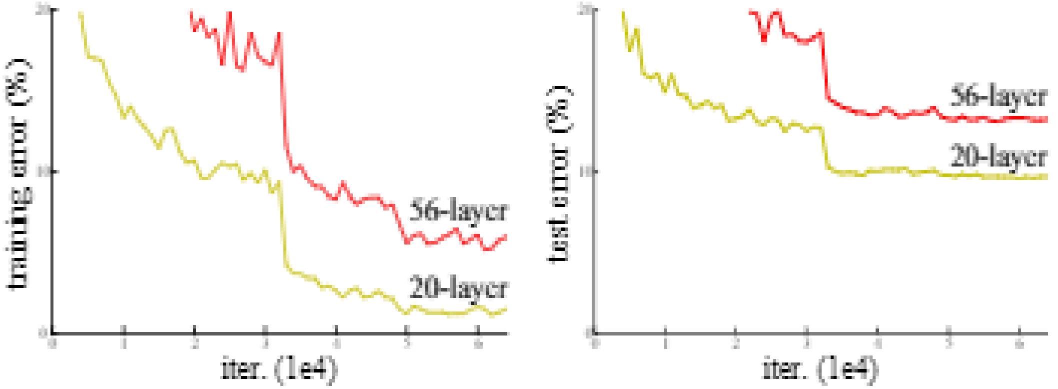


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Reference: https://arxiv.org/abs/1512.03385

1. Introduction: Problem Definition

- Vanishing Gradient
  - Your gradient/derivative can get very very very small
  - Even exponentially small
  - Make the training difficult to converge, or not converge.
- Exploding Gradients
  - Your gradient/derivative can get very very very larger
  - Make the gradient exploded/diverge…
- Degradation problem:
  - With the network depth increasing, accuracy get saturated

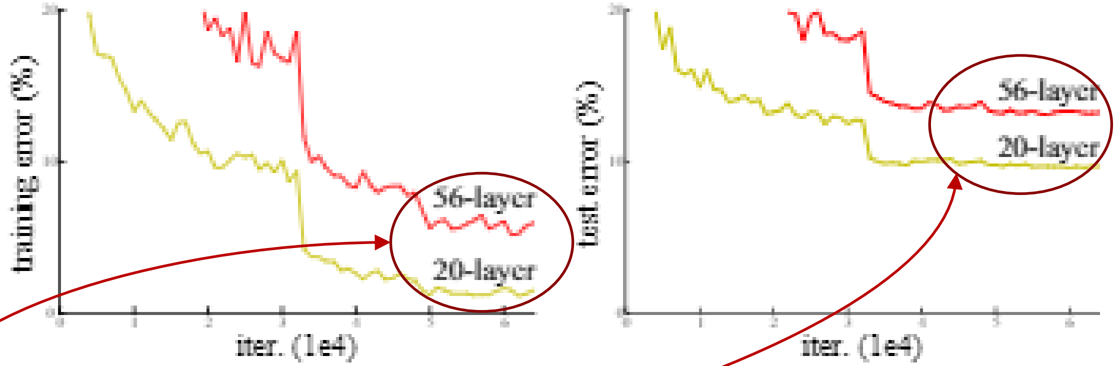


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

2. Network Design: Residual Learning Block

- Better solution: ResNet
- Inspired by VGG nets: stacking building block of the same shape
- Residual Block
  - Insert “identity shortcuts”, aka shortcut connection, or skip connection
  - Allow the information directly pass to deeper layer.
  - Add neither extra parameter nor computational complexity
- ResNet = a stack of Residual Block

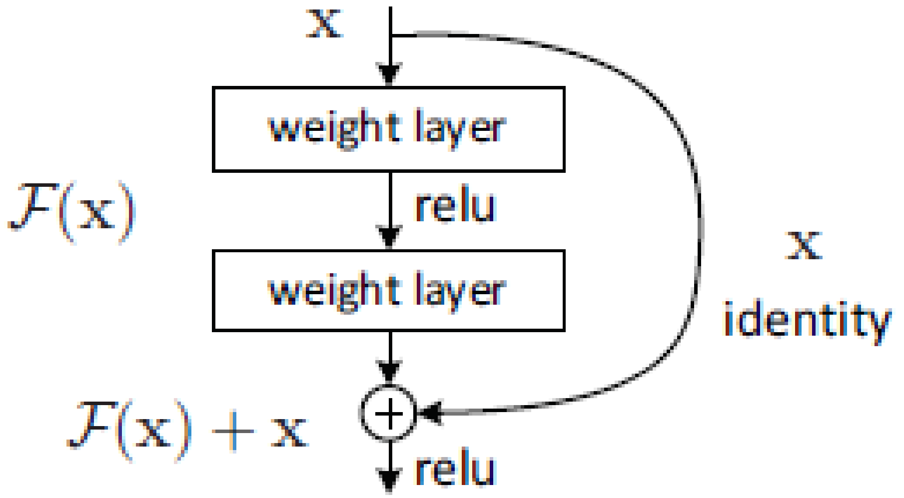
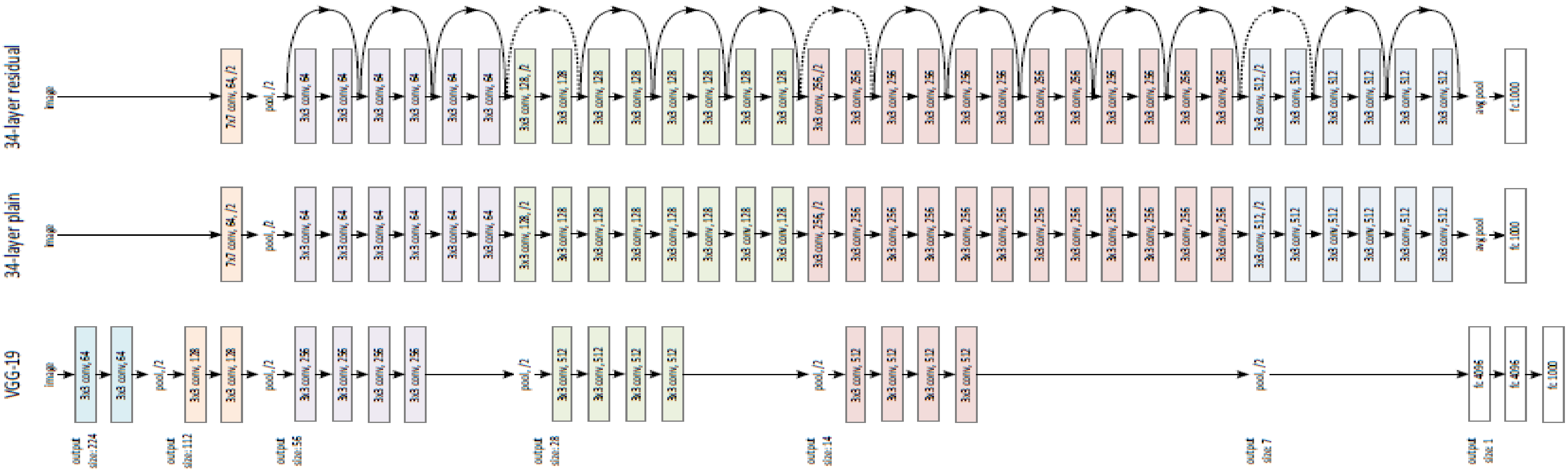


Figure 2. Residual learning: a building block.

2. Network Design: Plain vs ResNet vs VGG

THE OHIO STATE UNIVERSITY

- ResNet = Plain Network + Short Connection
  - Residual network can gain accuracy from considerably increased depth.
- Top: a ResNet with 34 parameter layers (3.6 billion FLOPs).
- Middle: a plain network with 34 parameter layer (3.6 billion FLOPs).
- Bottom: VGG-19 model (19.6 billion FLOPs).



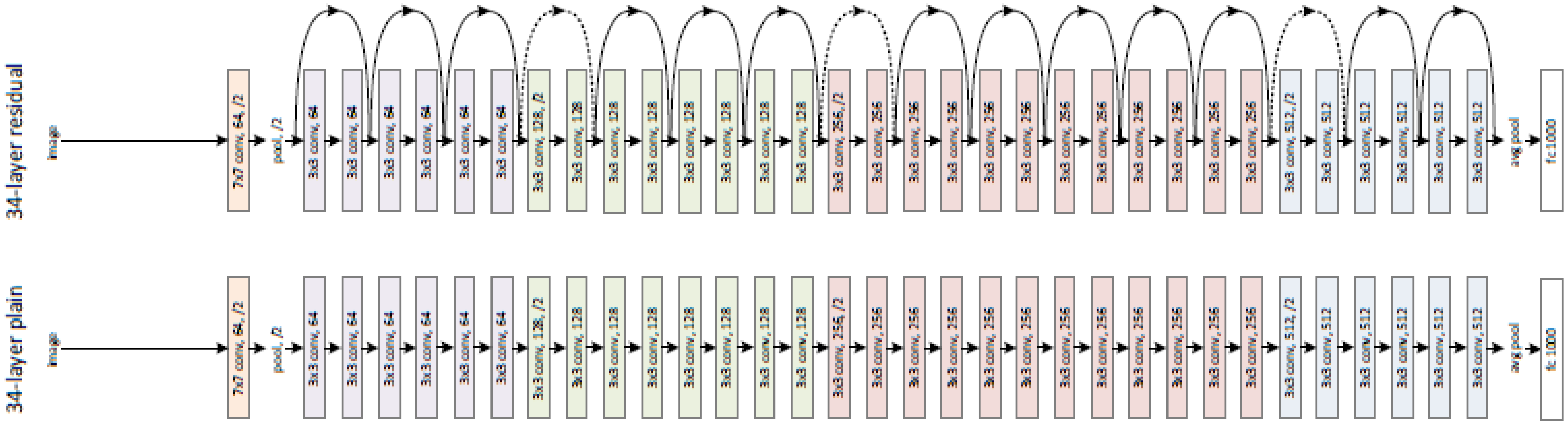
2. Network Design: Shortcut Connections

- Identity Mapping and Projection W_s:
  - If input and output has same dimensions (denoted by solid line):

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}. \qquad (1)$$

  - If input and output has different dimensions (denoted by dotted line):
    - Option A: Zero padded for extra dimension
    - Option B: Perform the projection shortcut to match the dimension (done by 1*1 Conv).

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}. \qquad (2)$$



- Comparison
  - Type A: zero-padding for increasing dimensions, and rest are identity shortcut(parameter free)
  - Type B: Projection for increasing dimensions only
  - Type C: Projection for all shortcut
- Conclusion:
  - Type C is marginally better as extra parameters introduced, but time complexity and model size are doubled
  - Type A is used for rest of paper

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

Identity Shortcut:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}. \qquad (1)$$

Projection Shortcut:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}. \qquad (2)$$

- Dataset: ImageNet 2012 classification dataset
  - Training dataset: 1.28M train images
  - Validation dataset: evaluated on 50K validation images
  - Testing dataset: final resulted (top1 and top5 error rate) tested on 100K test images.
  - Classes: consist of 1000 classes
- Larger network architectures evaluated in ImageNet: <u>"Bottleneck" building block</u>

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				
		[3×3, 64 3×3, 64] ×2	[3×3, 64 3×3, 64] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3
conv3_x	28×28	[3×3, 128 3×3, 128] ×2	[3×3, 128 3×3, 128] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×8
conv4_x	14×14	[3×3, 256 3×3, 256] ×2	[3×3, 256 3×3, 256] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×23	[1×1, 256 3×3, 256 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512 3×3, 512] ×2	[3×3, 512 3×3, 512] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3
	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

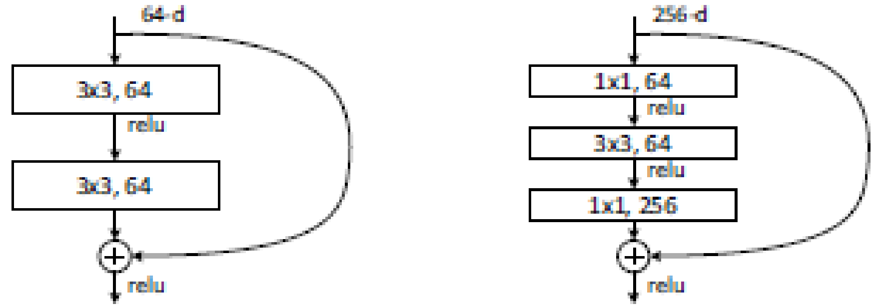


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

4. ImageNet Classification: Performance analysis

- Plain network vs ResNet
  - Obvious degradation problem
  - Plain net has higher training error throughout the whole training procedure
  - Situation reversed with ResNets

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

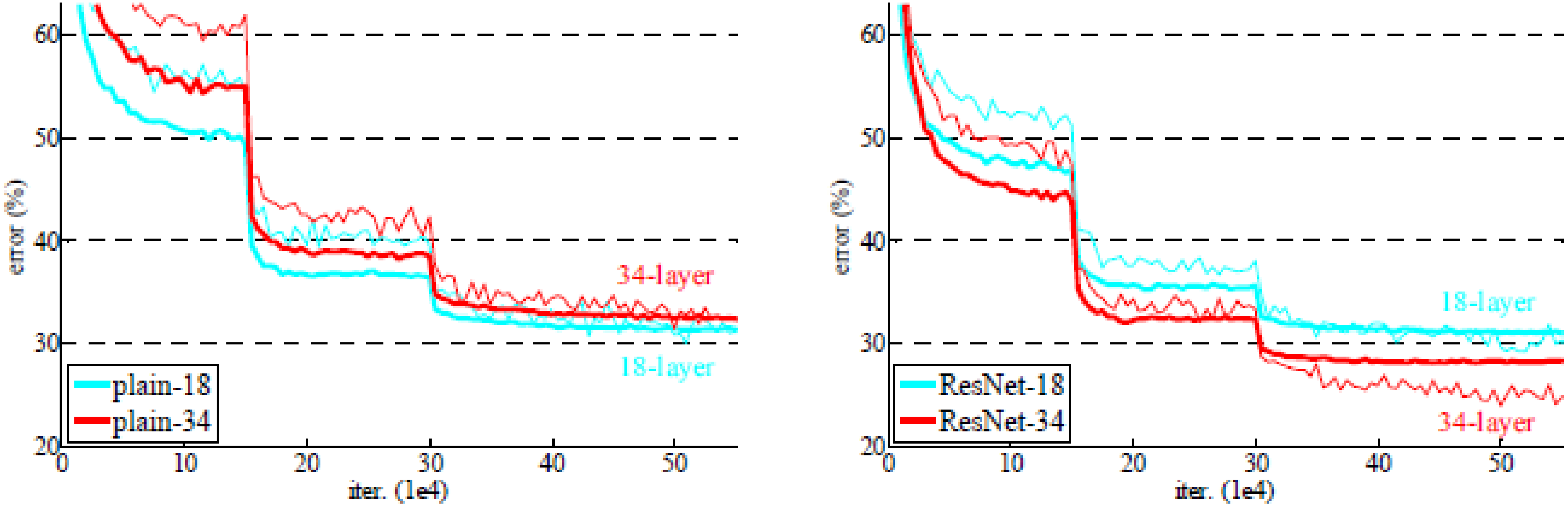


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

5. Implementation Tricks (read offline)

- Augmentation follow the practice [8, 9]
  - Resize the images with its shorter side randomly sampled in between [256; 480] for scale augmentation. [9]
  - Standard color augmentation. [8]
  - A 224x224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted [8]
- Other techniques follow [5, 6, 7]
  - Batch normalization right after each Conv and before activation. [7]
  - Initialized the weight as in [5] and train all plain/residual nets from scratch.
  - Use SGF with a mini-batch size of 256.
  - The learning rate starts from 0.1 and is divided by 10 when the error stagnated, and the models are trained for up to 60*10^4 iterations.
  - Use a weight decay of 0.0001 and a momentum of 0.9.
  - No dropout. [6] [7]
- Code is available: https://github.com/KaimingHe/deep-residual-networks
  - [TF version] https://github.com/tensorpack/tensorpack/tree/master/examples/ResNet

Part2: (ResNeXt) Aggregated Residual Transformations for Deep Neural Networks

1. Introduction: Background and Key Contribution

THE OHIO STATE UNIVERSITY

- Background:
  - Won second place in the 2016 ILSVRC image classification task
  - A simpler design: a 101-layer ResNeXt achieved better accuracy than ResNet-200 but has only 50% complexity.
  - The transition from "Feature Engineering" to "Network Engineering": In contrast to traditional hand-designed features (e.g. SIFT and HOG), human effort has been shifted to designing better neural network architecture for learning representation.
- Main Contribution:
  - Adopted similar strategy inherited from VGG/ResNets: stack modules of same topology.
  - Exploited the split-transform-merge (aka multi-path) strategy in an easy and extensible way.
  - Introduces a new dimension for gaining the accuracy: Cardinality

Reference: https://arxiv.org/abs/1611.05431

1. Introduction: Next Dimension

Question: Shouldn't building better neural networks as easy as stacking more layers?

- Old approach: going deeper(increase #layers) and wider(increase bottleneck width)
- New approach: increase cardinality C
- **Cardinality**: the size of the set of transformation (or # of branches/paths/groups)

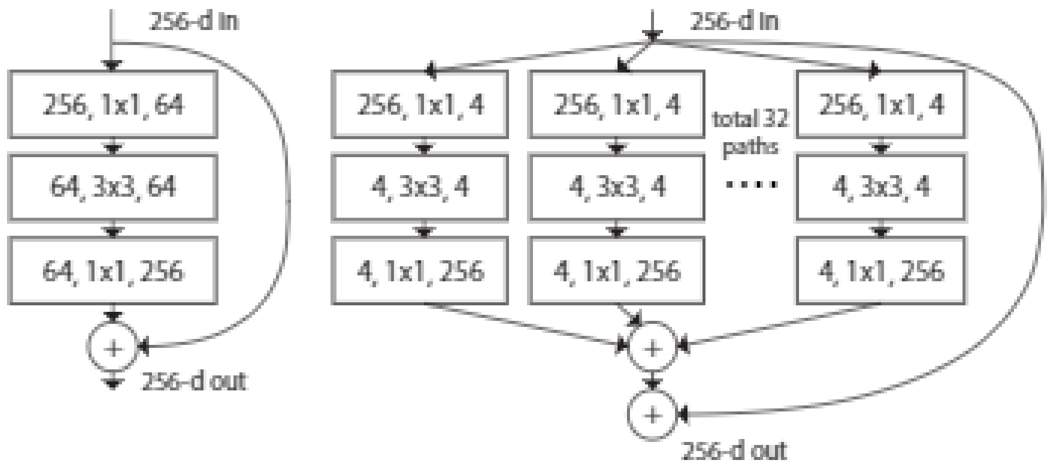
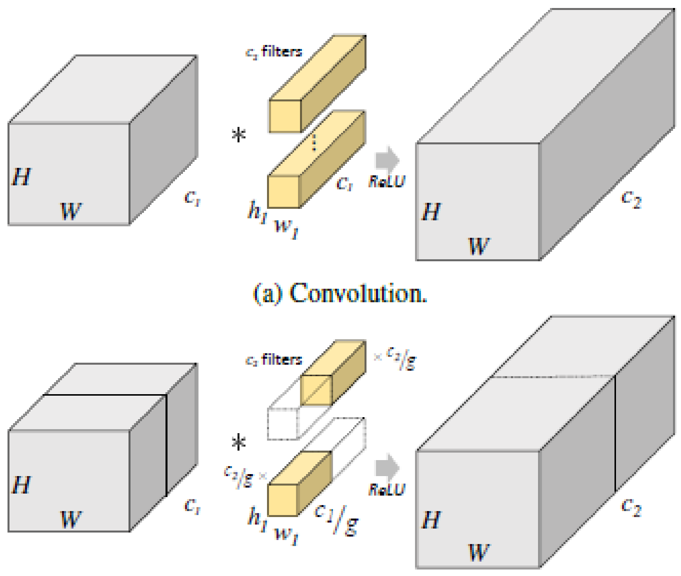


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

2. Related work: Grouped Convolutions

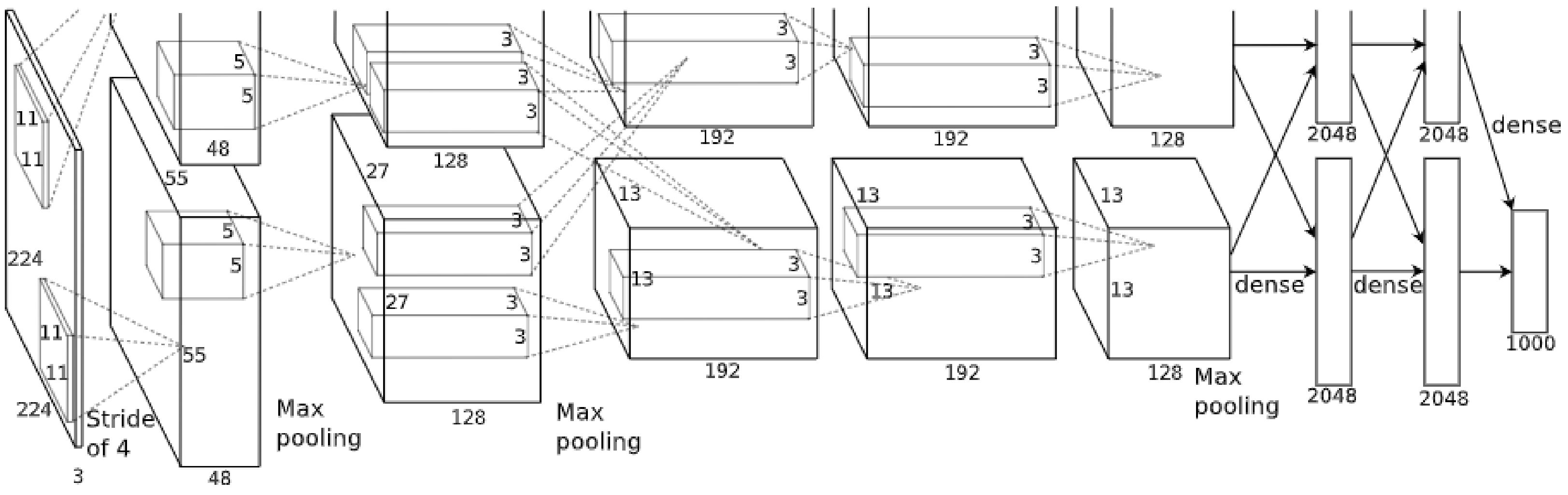
THE OHIO STATE UNIVERSITY

- Grouped Convolutions:
  - a process of applying multiple kernels/filters per layer on same images
  - Allow the training of network across multiple GPUs, and thus results more efficient parallelized training.
  - Learned better representations, https://blog.yani.io/filter-group-tutorial/



(a) Convolution.

(b) Convolution with filter groups.



The architecture of AlexNet as illustrated in the original paper, showing two separate convolutional filter groups across most of the layers (Alex Krizhevsky et al. 2012).

Reference:Deep Roots: Improving CNN Efficiency with Hierarchical Filter Groups, https://arxiv.org/abs/1605.06489

Reference: ImageNet Classification with Deep Convolutional Neural Networks, http://www.cs.toronto.edu/~hinton/absps/imagenet.pdf

3. Method: Two Template rules

- Two simple rules:
  1. If producing spatial maps of the same size, the blocks share the same hyper-parameters (width and filter size)
  2. Each time when the spatial map is downsampled by a factor of 2, the width of the blocks is multiplied by a factor of 2.

stage	output	ResNet-50	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2 [1×1, 64 3×3, 64 1×1, 256] ×3	3×3 max pool, stride 2 [1×1, 128 3×3, 128, C=32 1×1, 256] ×3
conv3	28×28	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 256 3×3, 256, C=32 1×1, 512] ×4
conv4	14×14	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 512 3×3, 512, C=32 1×1, 1024] ×6
conv5	7×7	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 1024 3×3, 1024, C=32 1×1, 2048] ×3
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**$\times 10^6$	**25.0**$\times 10^6$
FLOPs		**4.1**$\times 10^9$	**4.2**$\times 10^9$

Table 1. (**Left**) ResNet-50. (**Right**) ResNeXt-50 with a 32×4d template (using the reformulation in Fig. 3(c)). Inside the brackets are the shape of a residual block, and outside the brackets is the number of stacked blocks on a stage. "C=32" suggests grouped convolutions [24] with 32 groups. *The numbers of parameters and FLOPs are similar between these two models.*

3. Method: Aggregated Transformations

- Simple Neurons:

$$\sum_{i=1}^{D} w_i x_i, \qquad (1)$$

- Aggregated Transformation:

$$\mathbf{y} = \mathbf{x} + \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x}), \qquad (3)$$

  - C: cardinality, size of the set of transformations to be aggregated
  - T_i(x): arbitrary transformation function, e.g. linear transformation

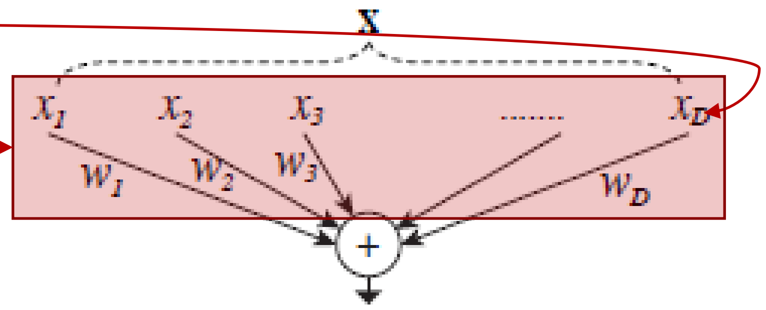
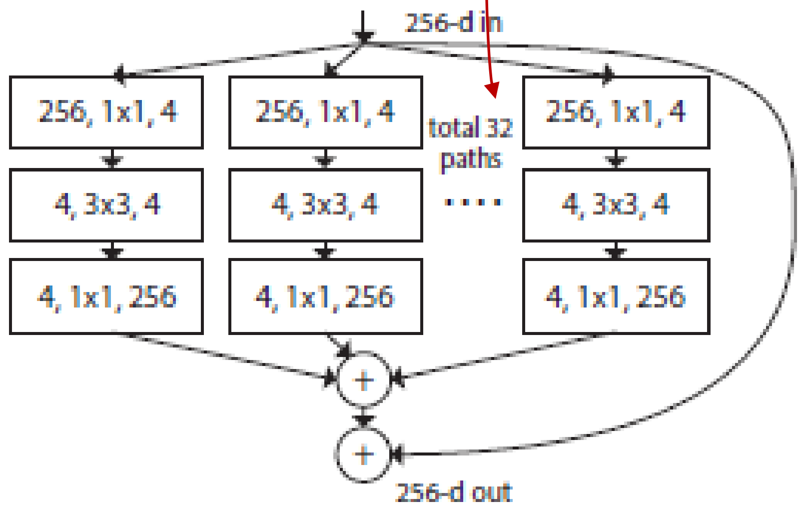


Figure 2. A simple neuron that performs inner product.



(a)

3. Method: Equivalent building block of ResNeXt

THE OHIO STATE UNIVERSITY

- Fig.3 a): $$\mathbf{y} = \mathbf{x} + \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x}), \qquad (3)$$
- Fig.3 b): Similar to Inception-ResNet block, but the same topology shared amount the multiple paths.
- Fig.3 c): applied grouped convolutions

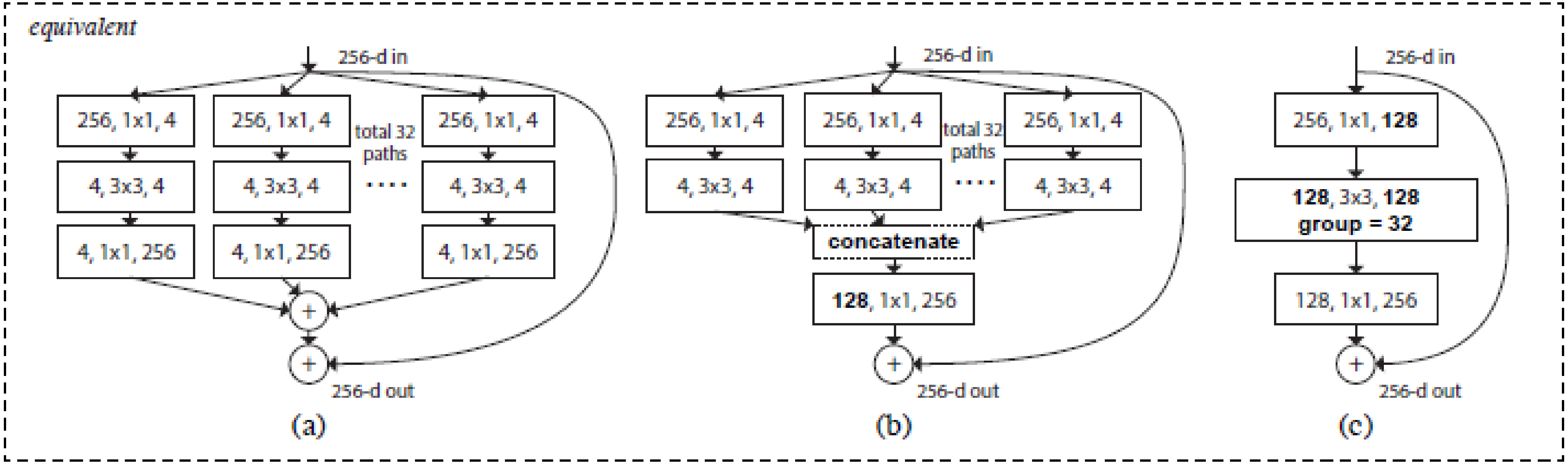


Figure 3. Equivalent building blocks of ResNeXt. **(a)**: Aggregated residual transformations, the same as Fig. 1 right. **(b)**: A block equivalent to (a), implemented as early concatenation. **(c)**: A block equivalent to (a,b), implemented as grouped convolutions [24]. Notations in **bold** text highlight the reformulation changes. A layer is denoted as (# input channels, filter size, # output channels).

3. Method: Inception-ResNet vs ResNeXt

- Inception-ResNet
  - Many hyper-parameters need to be tailored for each individual transformation
  - Hard to adapt to a new dataset/task
- ResNeXt:
  - Use the same topology among all paths
  - Proved a better accuracy over all Inception model

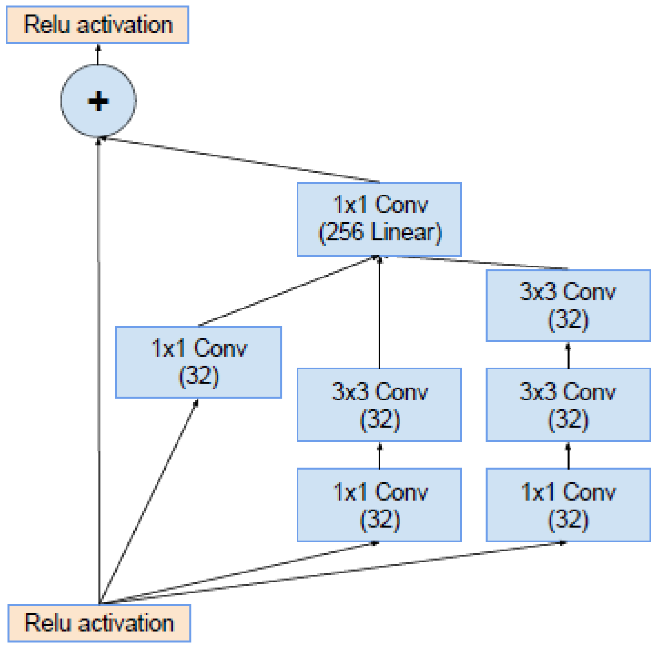


Figure: Inception-ResNet-v2 module

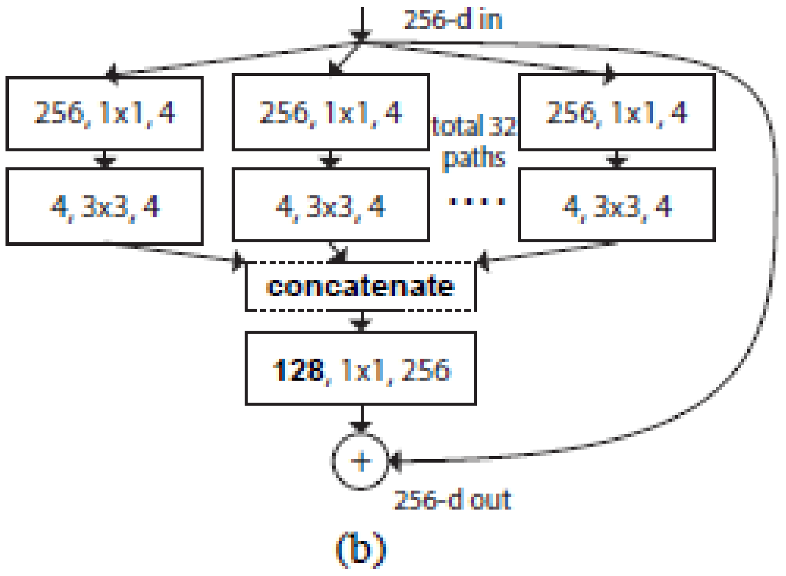


Figure: ResNeXt building block with 32 cardinality

Reference: https://arxiv.org/pdf/1602.07261.pdf

4. ImageNet1K: Model Capacity vs Width

- For evaluating different cardinalities C, the complexity(# params) is preserved by adjusting the width of bottleneck.
  - Calculate the #params for original network
    - **ResNet-50 (1x64d)** = 256*64+3*3*64*64+64*256~=70k params
  - Calculate the #params for bottleneck width d:
    - **ResNeXt-50 (32x4d)** = C*(256*d+3*3*d*d+d*256)~=70k params

Equivalent Complexity

cardinality C	1	2	4	8	32
width of bottleneck d	64	40	24	14	4
width of group conv.	64	80	96	112	128

Table 2. Relations between cardinality and width (for the template of conv2), with roughly preserved complexity on a residual block. The number of parameters is ~70k for the template of conv2. The number of FLOPs is ~0.22 billion (# params×56×56 for conv2).

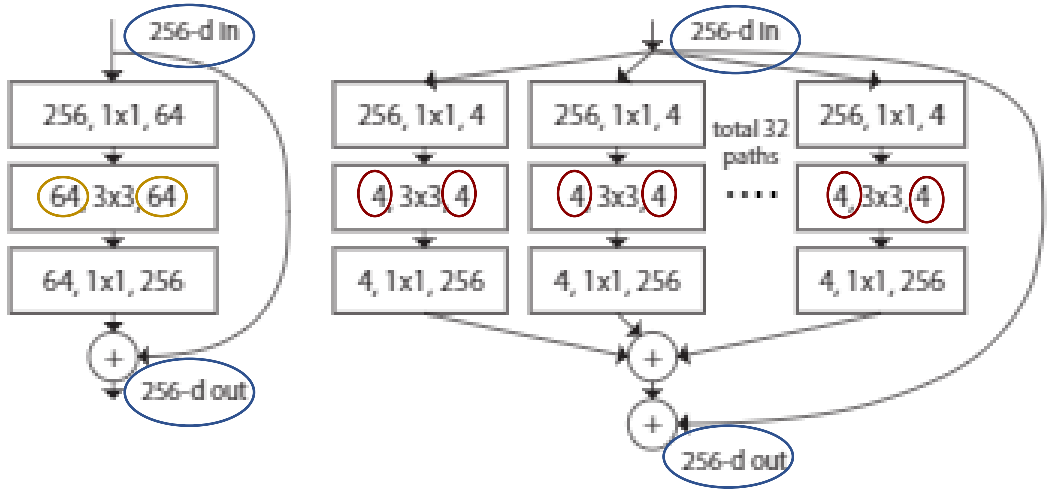


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

Table 3. Ablation experiments on ImageNet-1K. (**Top**): ResNet-50 with preserved complexity (~4.1 billion FLOPs); (**Bottom**): ResNet-101 with preserved complexity (~7.8 billion FLOPs). The error rate is evaluated on the single crop of 224×224 pixels.

4. ImageNet-1K: Increasing Cardinality Vs Deeper/Wider

THE OHIO STATE UNIVERSITY

- Original approach:
  - Going Deeper: 0.3% improvement
  - Going Wider: 0.7% improvement
- New approach:
  - Increasing Cardinality(C): 1.3% improvement
- Conclusion: Increasing cardinality C shows much better results than going deeper or wider

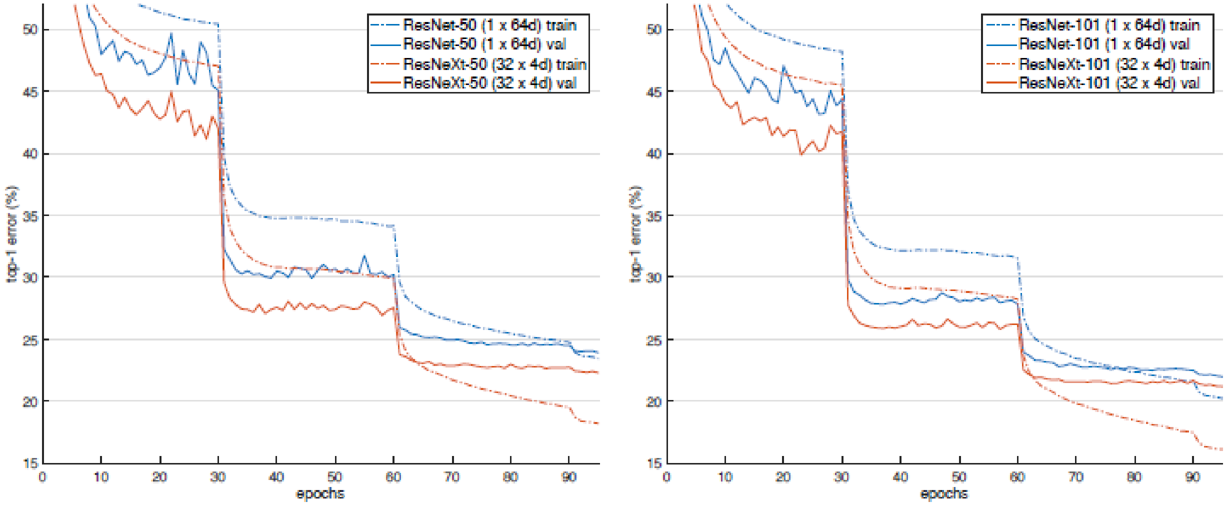


Figure 5. Training curves on ImageNet-1K. (**Left**): ResNet/ResNeXt-50 with preserved complexity (~4.1 billion FLOPs, ~25 million parameters); (**Right**): ResNet/ResNeXt-101 with preserved complexity (~7.8 billion FLOPs, ~44 million parameters).

Baseline

Going Deeper

Going Wider

Increase Cardinality

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-200 [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100**d	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

Table 4. Comparisons on ImageNet-1K when the number of FLOPs is increased to 2× of ResNet-101's. The error rate is evaluated on the single crop of 224×224 pixels. The highlighted factors are the factors that increase complexity.

5. Implementation Details (read offline)

THE OHIO STATE UNIVERSITY

- A 224x224 crop is randomly cropped from a resized image using the scale and aspect ratio augmentation [13] [10]
- The shortcuts connection for different input-output dimension are project, type B in [12]
- Downsampling of conv3, 4, and 5 is done by stride-2 convolutions in the 3x3 layer of the first block in each stage, as suggested in [10]
- Use SGD with a mini-batch size of 256 on 8 GPUs (32 samples per GPUs for Data parallelism)
- The weight decay is 0.0001 and the momentum is 0.9
- Start from a learning rate of 0.1, and divide it by 10 for three times using the schedule in [10].
- Adopt the weight initialization of [12]
- evaluate the error on the single 224x224 center crop from an image whose shorter side is 256.
- Choose Fig.3 c) ResNeXt block, grouped convolutions.
- Batch normalization(BN) is performed right after the convolutions, and ReLU is performed right after BN, except the output of the block [12]
- Code is available of https://github.com/facebookresearch/ResNeXt
  - [PyTorch version]: https://pytorch.org/hub/pytorch_vision_resnext/

Conclusion

- ResNet
  - Vanishing gradient, Exploding gradient, and degradation problem
  - Residual building block, Bottleneck Building block
  - Shortcut connection, Projection shortcut
  - Deep residual network are easy to optimize and can gain a better accuracy as the increased of network depth.
- ResNeXt
  - multi-branch/path (split-transform-merge in Inception net) strategy
  - Two template rule, Aggregated transformation
  - Trade-off between Cardinality(C) and Bottleneck width(d)
  - Increasing cardinality is more effective than going deeper/wider.

Thank You!
Any Question?

Reference

[1] Deep Residual Learning for Image Recognition, https://arxiv.org/abs/1512.03385

[2] Aggregated Residual Transformations for Deep Neural Networks, https://arxiv.org/abs/1611.05431

[3] Deeplearning.ai, https://www.youtube.com/watch?v=ZILIbUvp5lk&list=PLpFsSf5Dm-pd5d3rjNtIXUHT-v7bdaEIe&index=113

[4] Cellstart, VANISHING / EXPLODING GRADIENT PROBLEM IN DEEP NEURAL NETWORKS, https://www.cellstrat.com/2018/05/17/vanishing-exploding-gradient-problem-in-deep-neural-networks/

[5] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In ICCV, 2015.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing coadaptation of feature detectors. arXiv:1207.0580, 2012.

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

[8] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[10] S. Gross and M. Wilber. Training and investigating Residual Nets. https://github.com/facebook/fb.resnet.torch, 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In ICCV, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015.