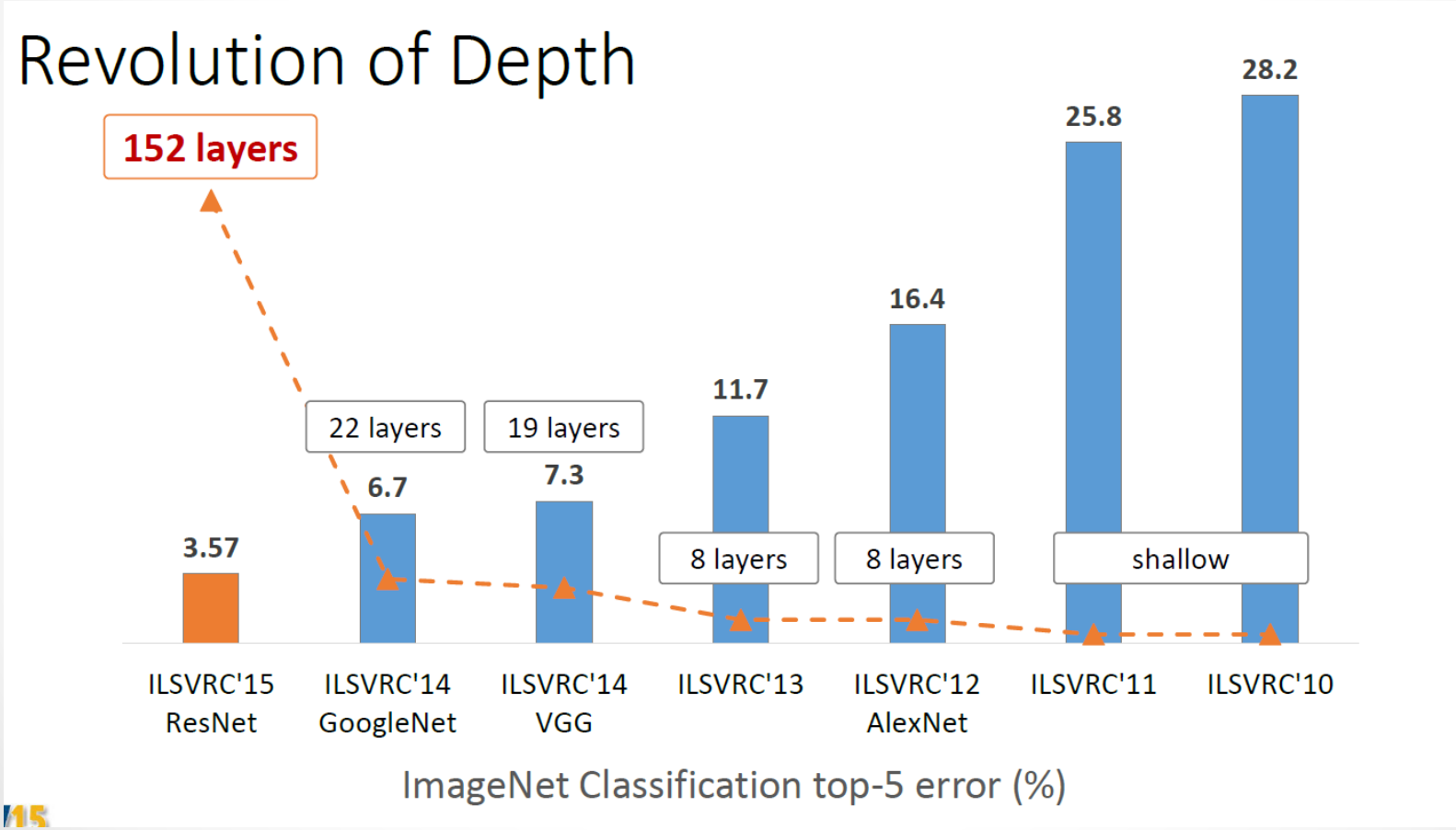
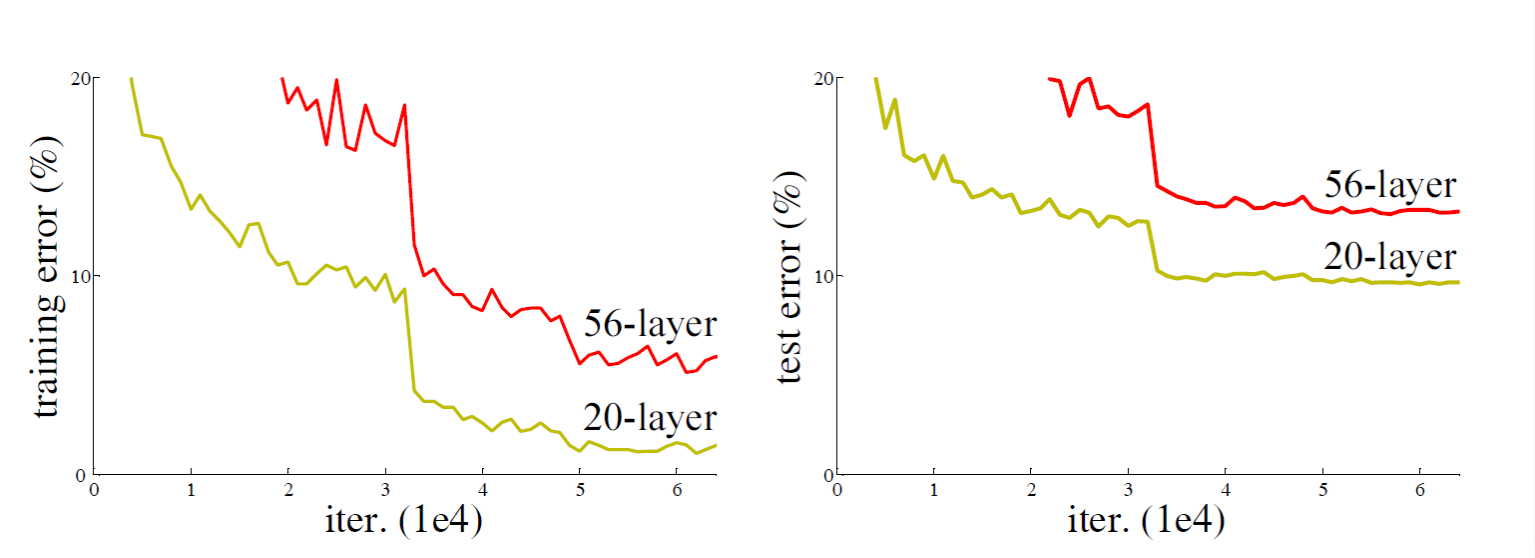
Deep Residual Network and Its Variations

Diyu Yang
(Originally prepared by Kaiming He from Microsoft Research)

Advantages of Depth



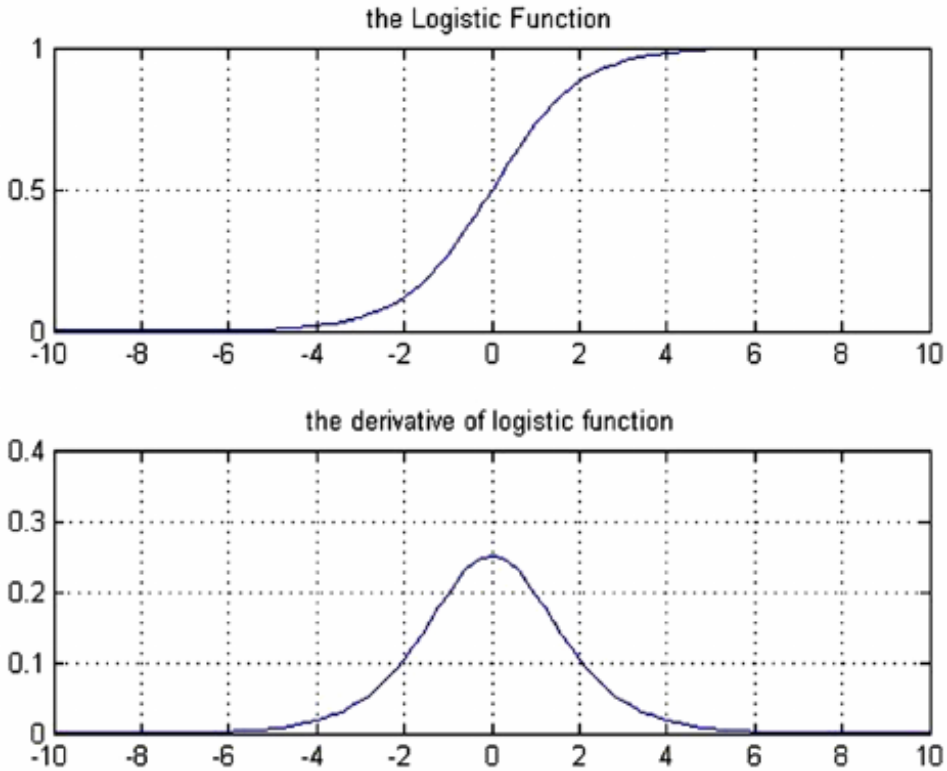
Degradation Problem



Possible Causes ?

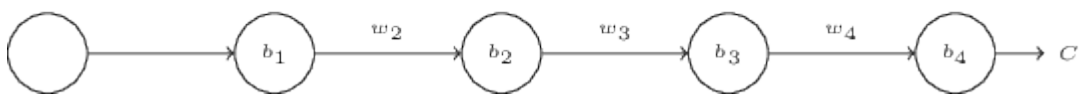
- Vanishing/Exploding Gradients. ?
- Overfitting ✗

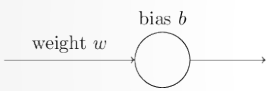
Vanishing/Exploding Gradients



Vanishing Gradients

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$





$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\overbrace{w_2\sigma'(z_2)}^{<\frac{1}{4}}\overbrace{w_3\sigma'(z_3)}^{<\frac{1}{4}}\underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

$|w_k| < 1$ $\quad$ $|\sigma' w_k| < \frac{1}{4}$

common terms

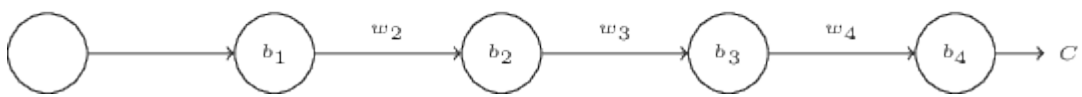
$$\frac{\partial C}{\partial b_3} = \sigma'(z_3)\overbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

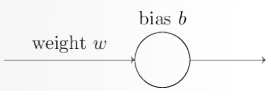
Gradients in the first layer become very small

http://neuralnetworksanddeeplearning.com/

Exploding Gradients

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$





$|w_k| = 100$ $\qquad$ $|\sigma' w_k| \approx 25$

Gradients in the first layer become very large

http://neuralnetworksanddeeplearning.com/

Batch Normalization(1)

- Addresses the problem of vanishing/exploding gradients.
- Increases learning speed and solves many other problems.
- Each activation in every iteration each layer is normalized to have zero mean and variance 1 over a minibatch.
- Integrated into back –propagation algorithm.

Batch Normalization(2)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

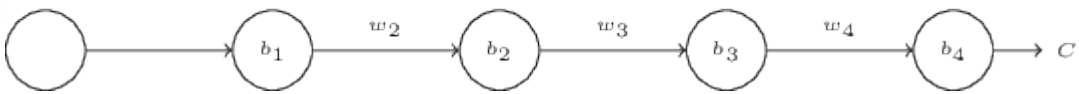
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

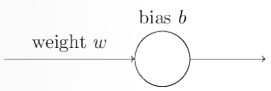
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

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

BN solves vanishing Gradients(1)

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$





$$z_i = \sigma(BN(Wz_{i-1} + b_i))$$

Now u is z:

$$\mathrm{BN}(W\mathrm{u}) = \mathrm{BN}((aW)\mathrm{u})$$

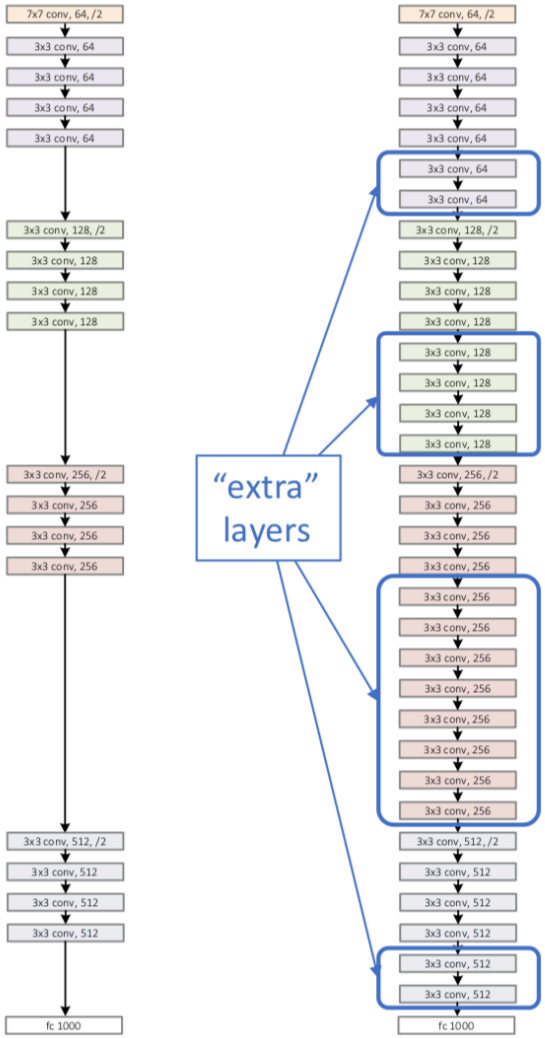
$$\frac{\partial \mathrm{BN}((aW)\mathrm{u})}{\partial \mathrm{u}} = \frac{\partial \mathrm{BN}(W\mathrm{u})}{\partial \mathrm{u}}$$

$$\frac{\partial \mathrm{BN}((aW)\mathrm{u})}{\partial (aW)} = \frac{1}{a} \cdot \frac{\partial \mathrm{BN}(W\mathrm{u})}{\partial W}$$

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

Motivation

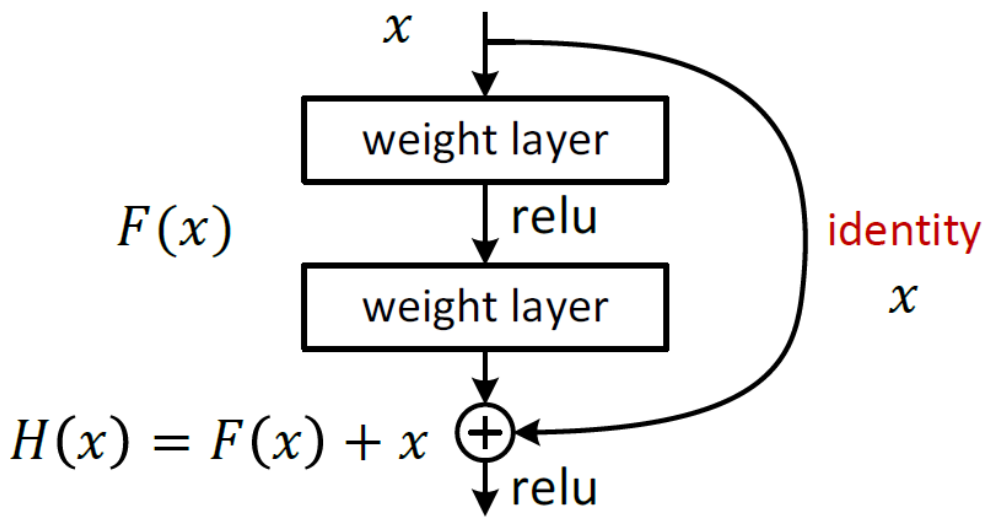
a shallower model (18 layers)



a deeper counterpart (34 layers)

- Richer solution space
- A deeper model should not have **higher training error**
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper…

Residual Learning



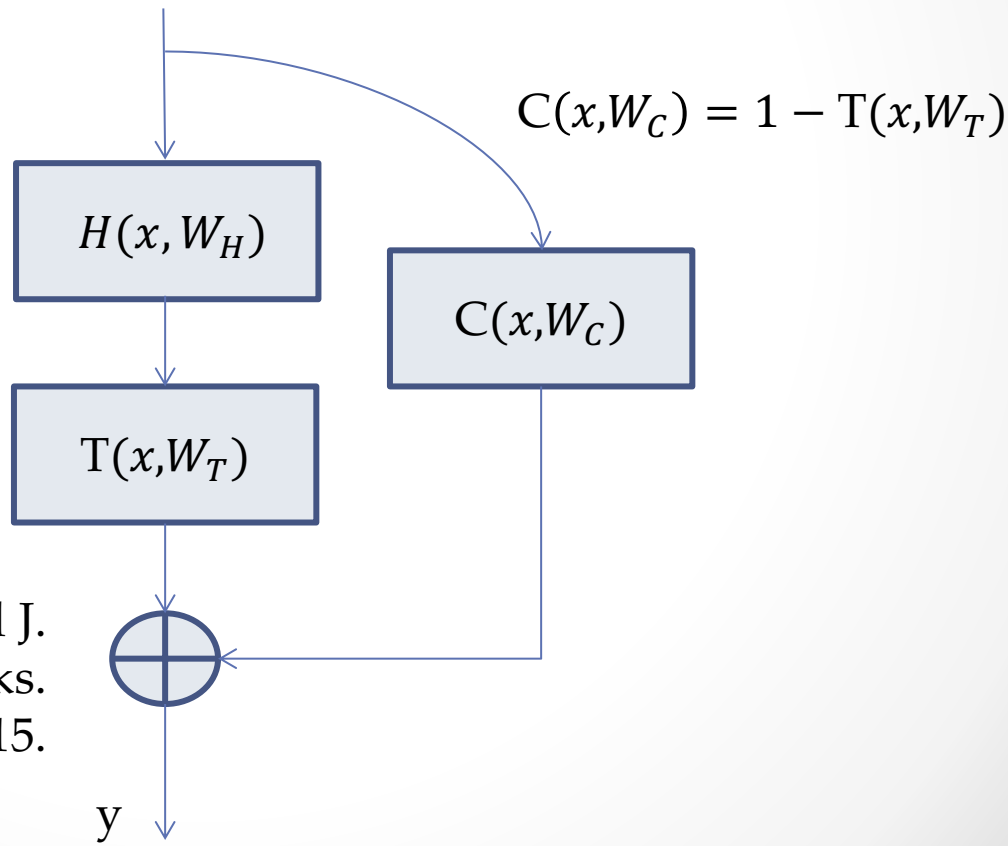
- 2 weight layers fit F(X) = H(x) – X instead of F(X).
- Networks might have "difficulties" fitting the unity mapping.

Very Deep Networks

- Two Similar approaches.
- ResNet.
- Highway Nets (R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks.
- arXiv:1505.00387, 2015.)

Similar Architecture – Highway Net

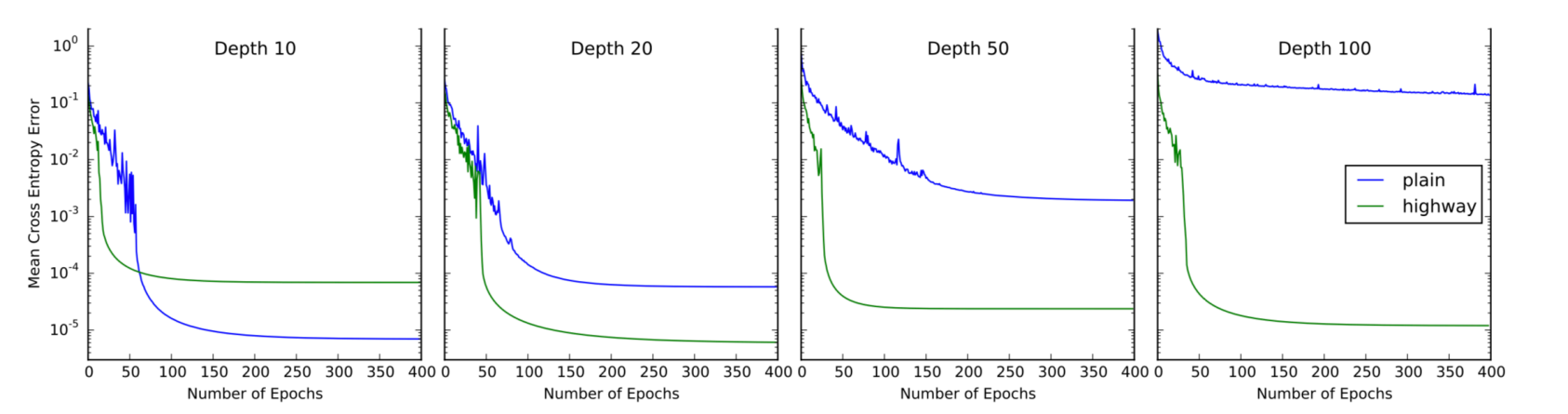
$$\mathrm{y} = H(\mathrm{x}, \mathbf{W_H}) \cdot T(\mathrm{x}, \mathbf{W_T}) + \mathrm{x} \cdot C(\mathrm{x}, \mathbf{W_C}).$$



R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. arXiv:1505.00387, 2015.

Highway Net vs. ResNet

- The gates C and T are data dependent and have parameters.
- When a gated shortcut is "closed" the layers in highway networks represent non-residual functions.
- High-2 way networks have not demonstrated accuracy gains with depth of over 100 layers.



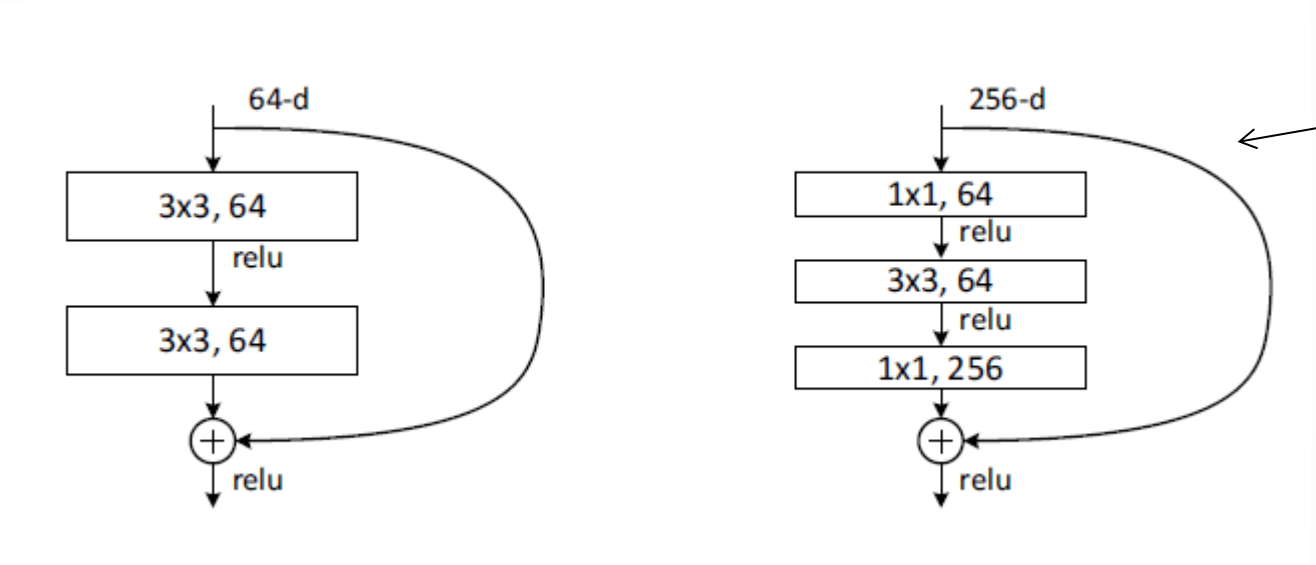
ResNet Design Rules

- 3 X3 Filters.
- Same number of filters for same feature map size.

Feature Map Size halved ⟷ Number of Filters doubled

- Almost no hidden max-pooling.
- Not Hidden FC-Layers.
- No Dropout.

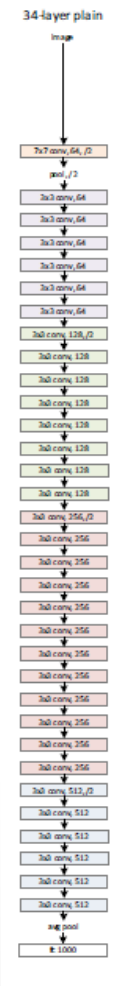
ResNet Buliding Blocks

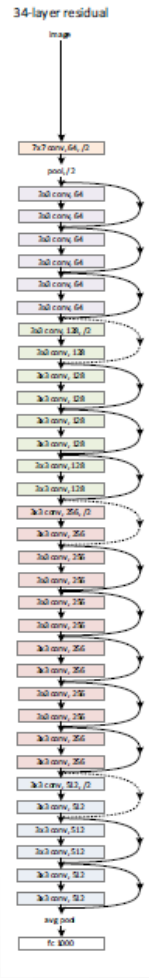


Bottleneck Building Block

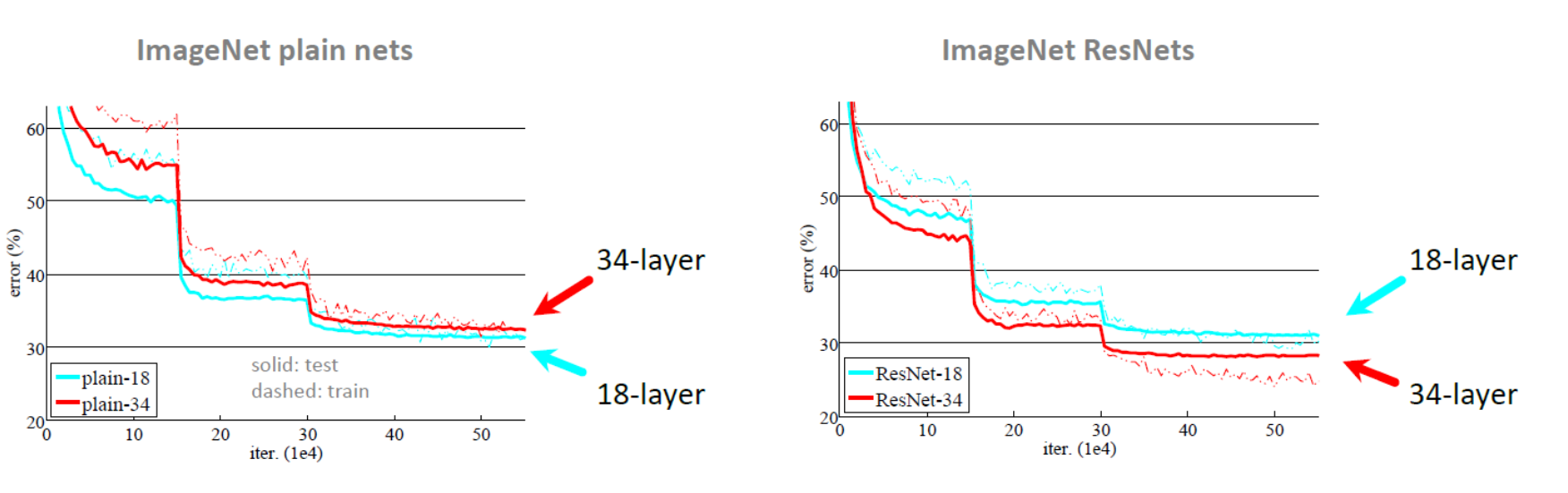
- Building Blocks are stacked to create full networks.

ResNet and Plain Nets



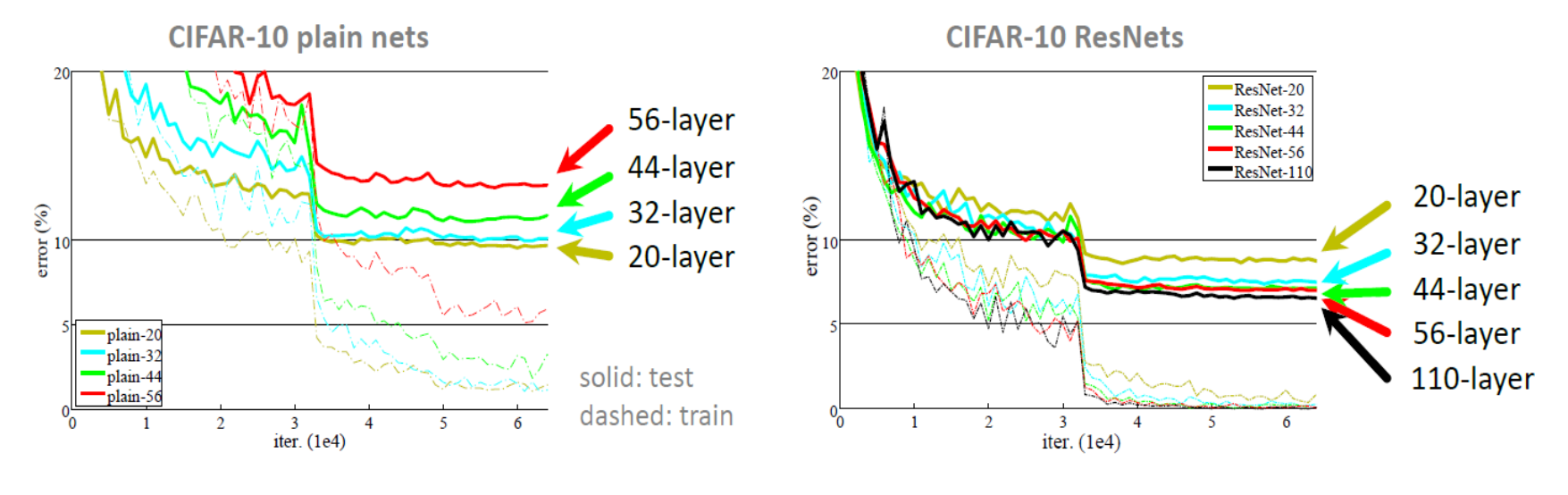


ResNets Vs. PlainNets(1)



- Deeper ResNets has lower train and test errors!

ResNets Vs. PlainNets(2)



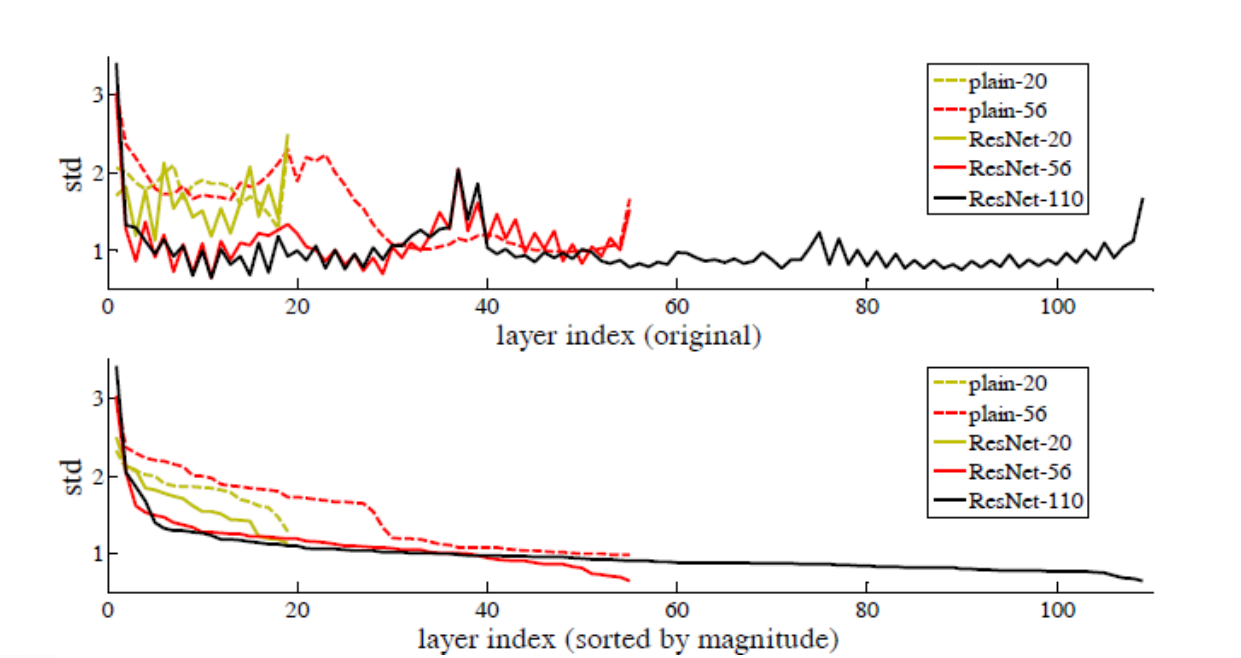
- Deeper ResNets has lower train and test errors!

ResNets Vs. PlainNets(3)

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

- For 34 layers ResNet has a lower error.
- For 18 layers error is roughly the same but convergence is faster.

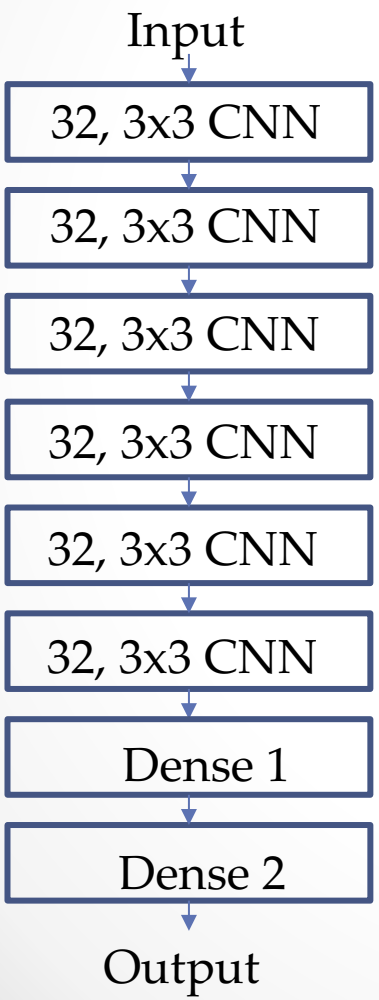
Layer Responses Magnitude

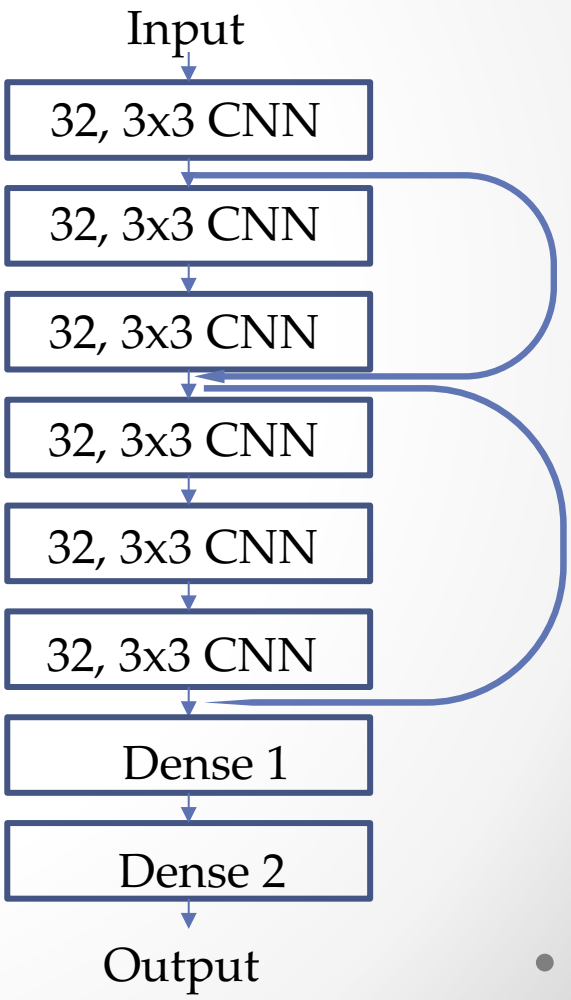


- Responses of 3X3 layers before non-linearity and after BN.
- Residual functions are closer to zero than non-residual.

Exercise 1

- Implement shortcut connections in Keras.





Dimensionality Change

- Shortcut connections assume dimension equality between input X and output F(x).
- If dimensions do not match :

Zero Pad X

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

Projection mapping (Extra Parameters)

Exploring Different Shortcuts Types

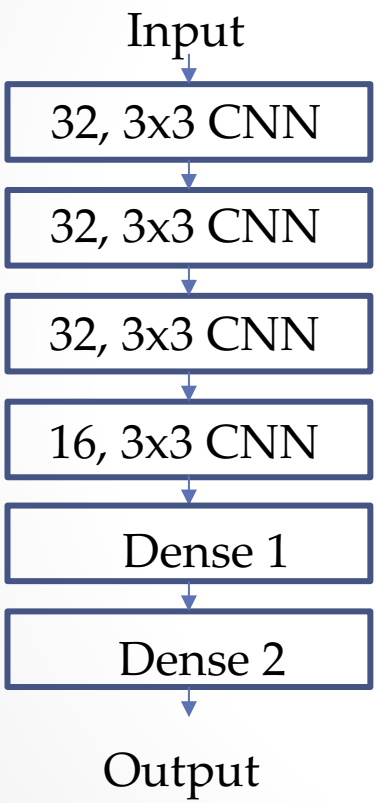
- **Three Options:**
- **A** - Zero padding for increasing dimensions.
- **B** – Projection shortcuts for increasing dimensions; others are identity.
- **C** – All shortcuts are projections.

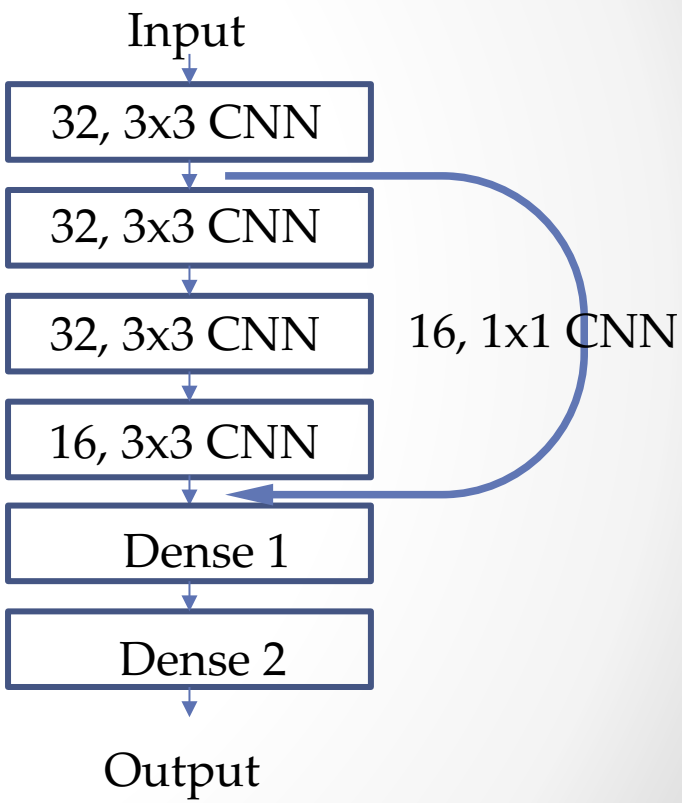
Overall Results

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**

Exercise 2

- Implement projected shortcut connections using CNN.





Uses For Image Detection and localization

- Based on Faster RCNN architecture.
- ResNet-101 architecture is used.
- Obtained best results on MS-COCO, imageNet localization and imageNet Detection datasets.

Conclusions

- Degradation problem is addressed for very deep NN.
- No additional parameter complexity.
- Faster convergence.
- Good for different types of tasks.
- Can be easily trained with existing solvers (Caffe, MatConvNet, etc…).
- Sepp Hochreiter, presumably described the phenomena in 1991.

Bottleneck Architectures

- Bottleneck building block with shortcut identity mapping.
- Similar time complexity to non- bottleneck block.
- 50-layer, 101-layer and 152-layer ResNets were constructed.

ResNet Implementation Details

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

Architectures For CIFAR-10

- First layer is 3X3 convolutional.
- 6n layer stacked . 2n layers for each feature map.
- Feature map size halved => number of filters doubled.
- Only identity shortcuts are used.
- Each pair of layers has a shortcut connection.

output map size	32×32	16×16	8×8
# layers	$1+2n$	$2n$	$2n$
# filters	16	32	64

Localization and Detection Results

Detection:

method	top-5 localization err	
	val	test
OverFeat [40] (ILSVRC'13)	30.0	29.9
GoogLeNet [44] (ILSVRC'14)	-	26.7
VGG [41] (ILSVRC'14)	26.9	25.3
ours (ILSVRC'15)	**8.9**	**9.0**

Localization:

	val2	test
GoogLeNet [44] (ILSVRC'14)	-	43.9
our single model (ILSVRC'15)	60.5	58.8
our ensemble (ILSVRC'15)	**63.6**	**62.1**

Results on detection MS-COCO

training data	COCO train		COCO trainval	
test data	COCO val		COCO test-dev	
mAP	@.5	@[.5, .95]	@.5	@[.5, .95]
baseline Faster R-CNN (VGG-16)	41.5	21.2		
baseline Faster R-CNN (ResNet-101)	48.4	27.2		
+box refinement	49.9	29.9		
+context	51.1	30.0	53.3	32.2
+multi-scale testing	53.8	32.5	**55.7**	**34.9**
ensemble			**59.0**	**37.4**

Overall Results CIFAR-10

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

- ResNet-110 train five times and results are in the format of "best (mean ± std)".
- n= {3,5,7,9,18, 200}.
- For 1202-layer net result degradation due to overfit.

Overall Results ImageNet(2)

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**