

Deep Residual Networks

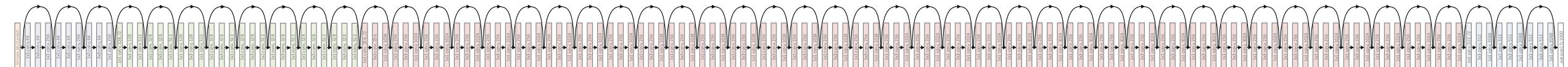
Deep Learning Gets Way Deeper

8:30-10:30am, June 19
ICML 2016 tutorial

Kaiming He

Facebook AI Research*

*as of July 2016. Formerly affiliated with Microsoft Research Asia



Overview

- Introduction
- Background
 - From shallow to deep
- Deep Residual Networks
 - From 10 layers to 100 layers
 - From 100 layers to 1000 layers
- Applications
- Q & A

Introduction

Introduction

Deep Residual Networks (ResNets)

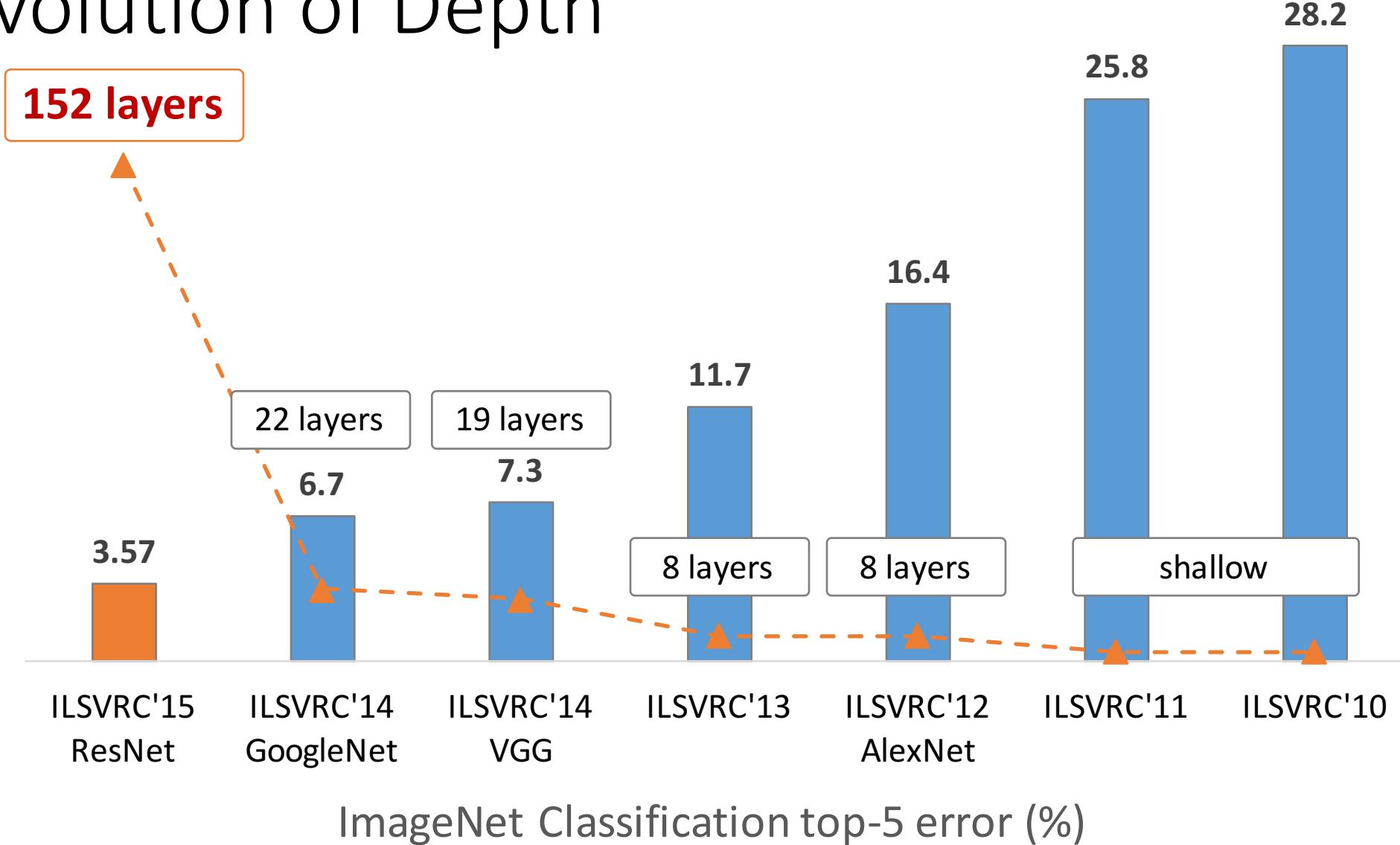
- “Deep Residual Learning for Image Recognition”. CVPR 2016 (next week)
- A simple and clean framework of training “very” deep nets
- State-of-the-art performance for
 - Image classification
 - Object detection
 - Semantic segmentation
 - and more...

ResNets @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
 - ImageNet Classification: “Ultra-deep” **152-layer** nets
 - ImageNet Detection: **16%** better than 2nd
 - ImageNet Localization: **27%** better than 2nd
 - COCO Detection: **11%** better than 2nd
 - COCO Segmentation: **12%** better than 2nd

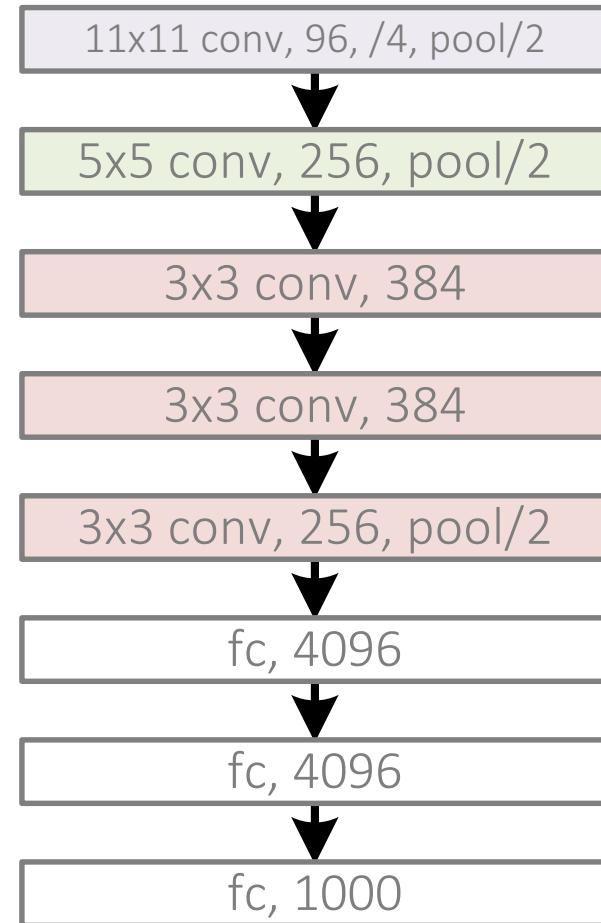
*improvements are relative numbers

Revolution of Depth



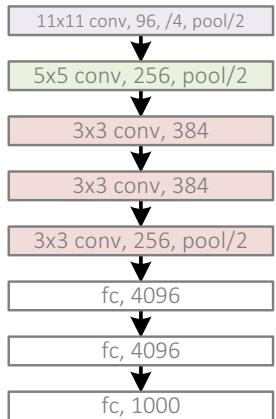
Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

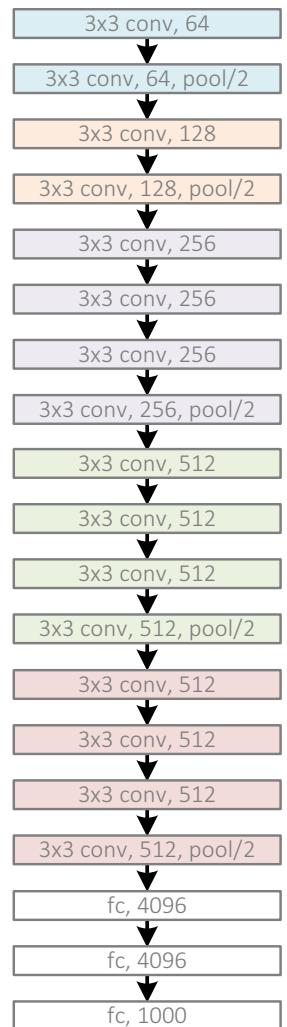


Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)



Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



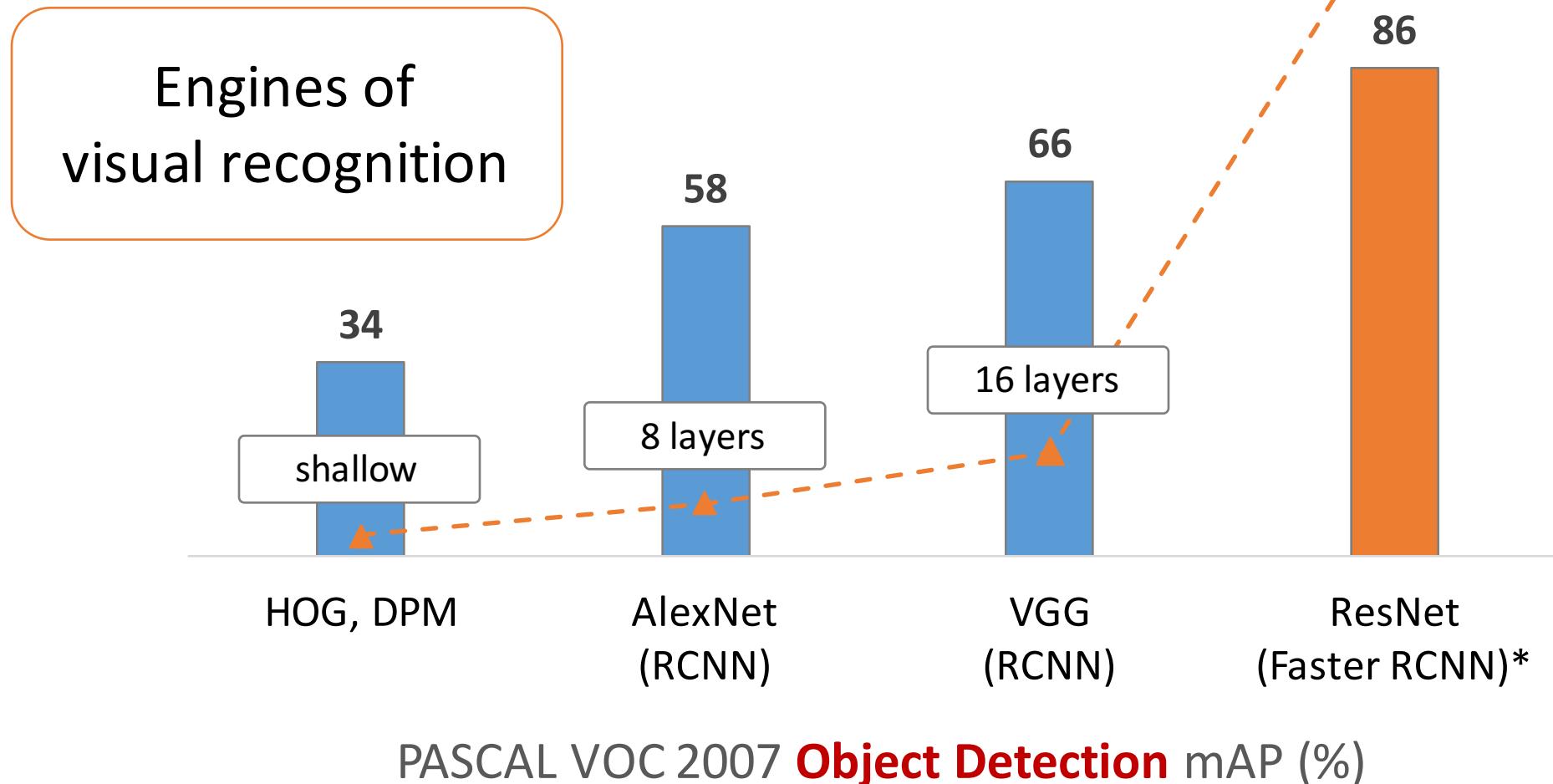
VGG, 19 layers
(ILSVRC 2014)



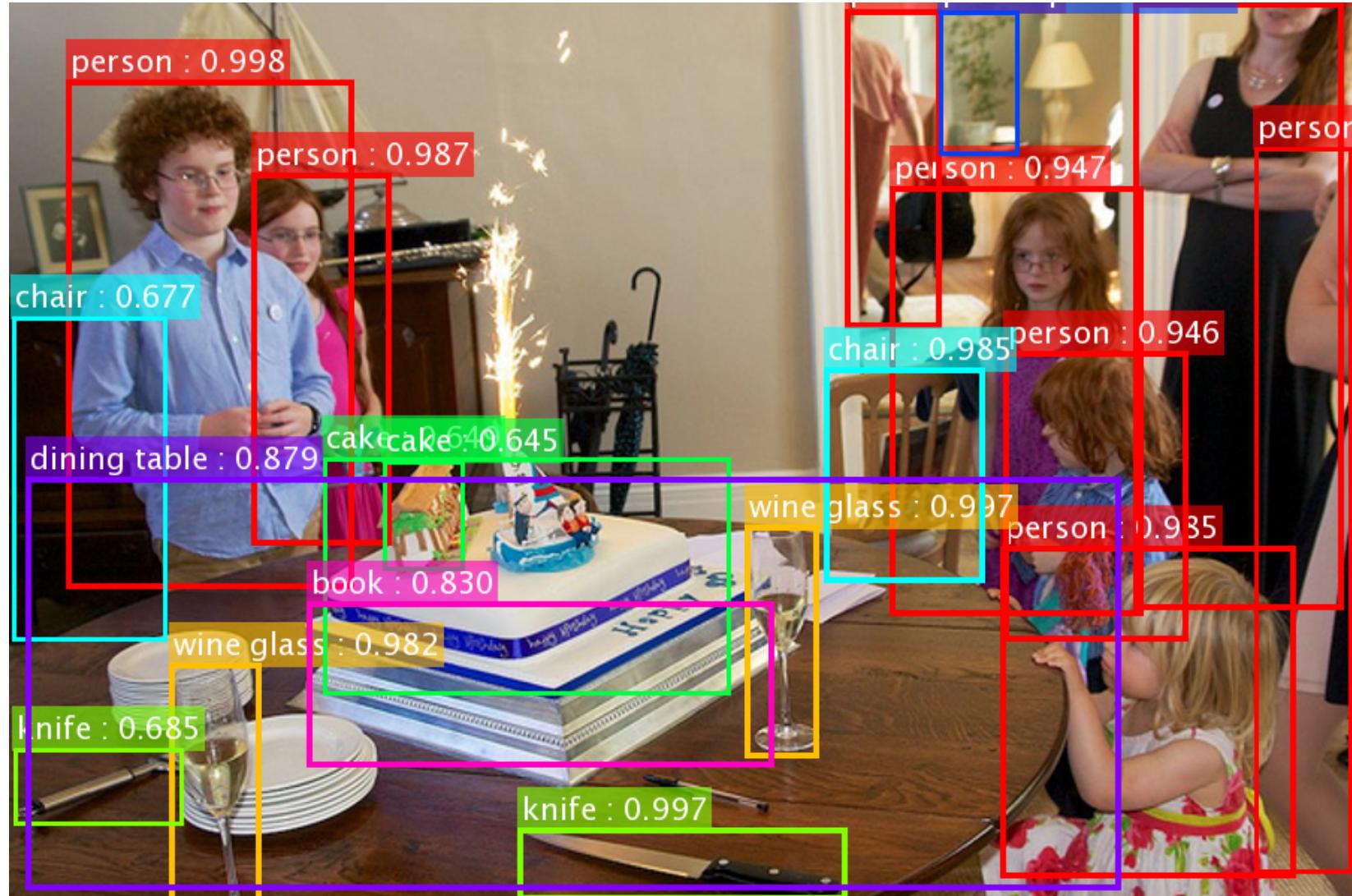
ResNet, **152 layers**
(ILSVRC 2015)



Revolution of Depth



*w/ other improvements & more data



ResNet's object detection result on COCO

*the original image is from the COCO dataset

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

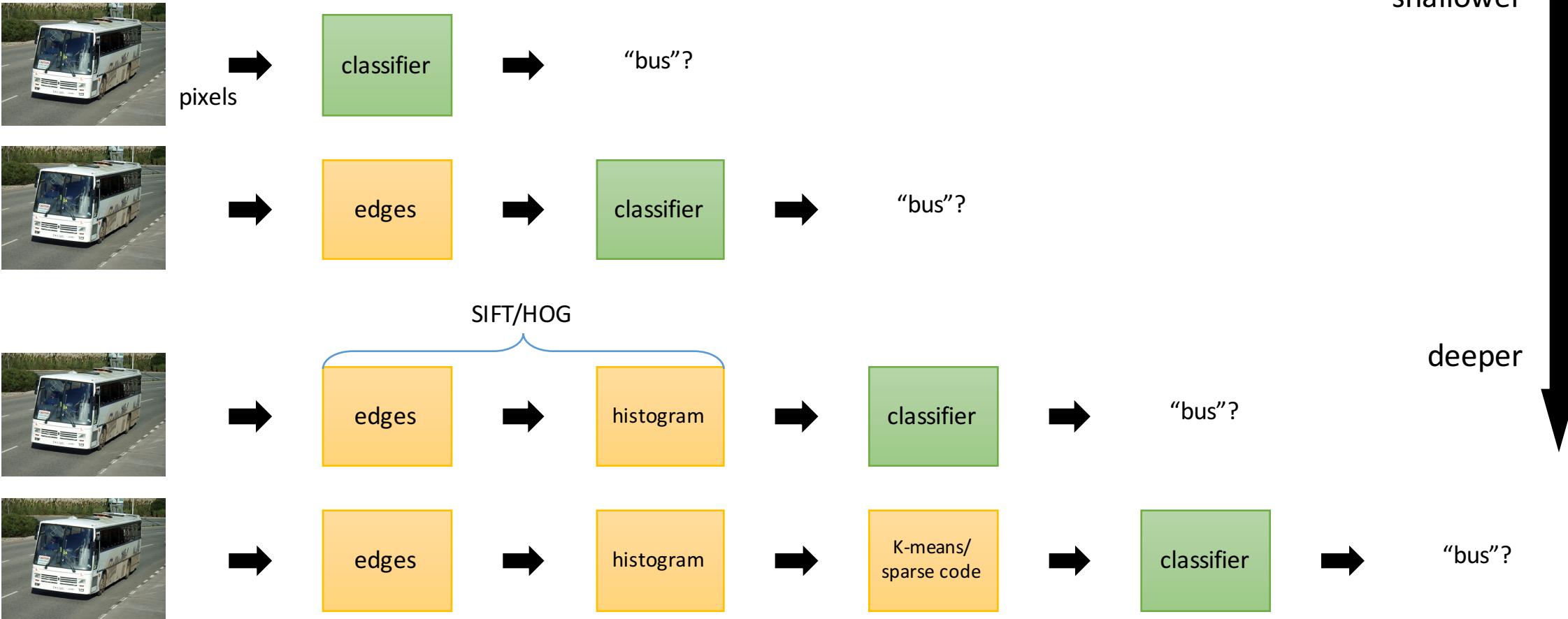
Very simple, easy to follow

- **Many third-party implementations** (list in <https://github.com/KaimingHe/deep-residual-networks>)
 - Facebook AI Research's Torch ResNet:
 - Torch, CIFAR-10, with ResNet-20 to ResNet-110, training code, and curves: code
 - Lasagne, CIFAR-10, with ResNet-32 and ResNet-56 and training code: code
 - Neon, CIFAR-10, with pre-trained ResNet-32 to ResNet-110 models, training code, and curves: code
 - Torch, MNIST, 100 layers: blog, code
 - A winning entry in Kaggle's right whale recognition challenge: blog, code
 - Neon, Place2 (mini), 40 layers: blog, code
 - ...
- **Easily reproduced results** (e.g. Torch ResNet: <https://github.com/facebook/fb.resnet.torch>)
- **A series of extensions and follow-ups**
 - > 200 citations in 6 months after posted on arXiv (Dec. 2015)

Background

From shallow to deep

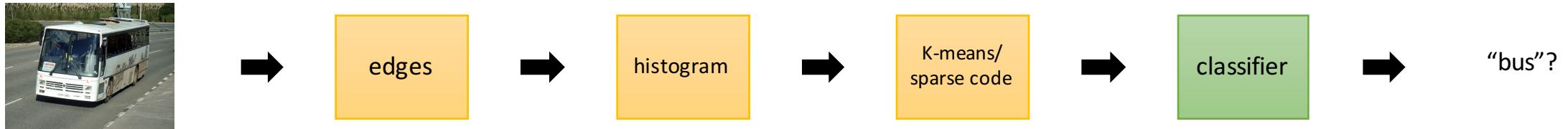
Traditional recognition



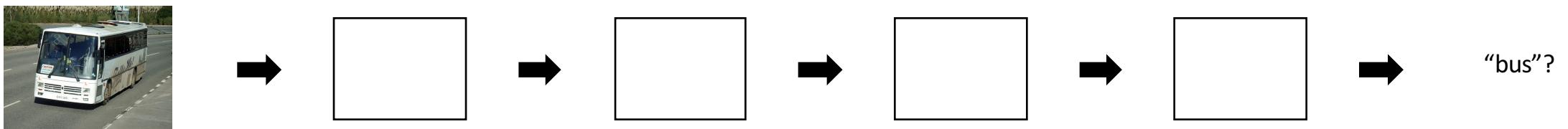
But what's next?

Deep Learning

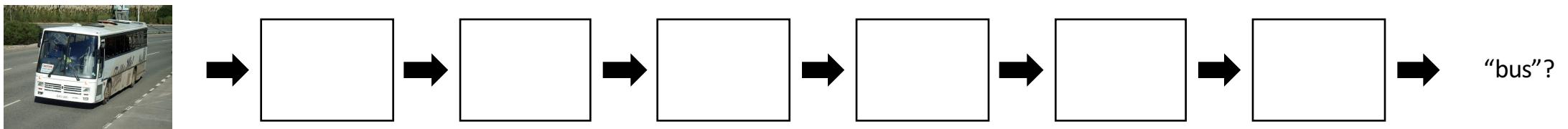
Specialized components, domain knowledge required



Generic components (“layers”), less domain knowledge

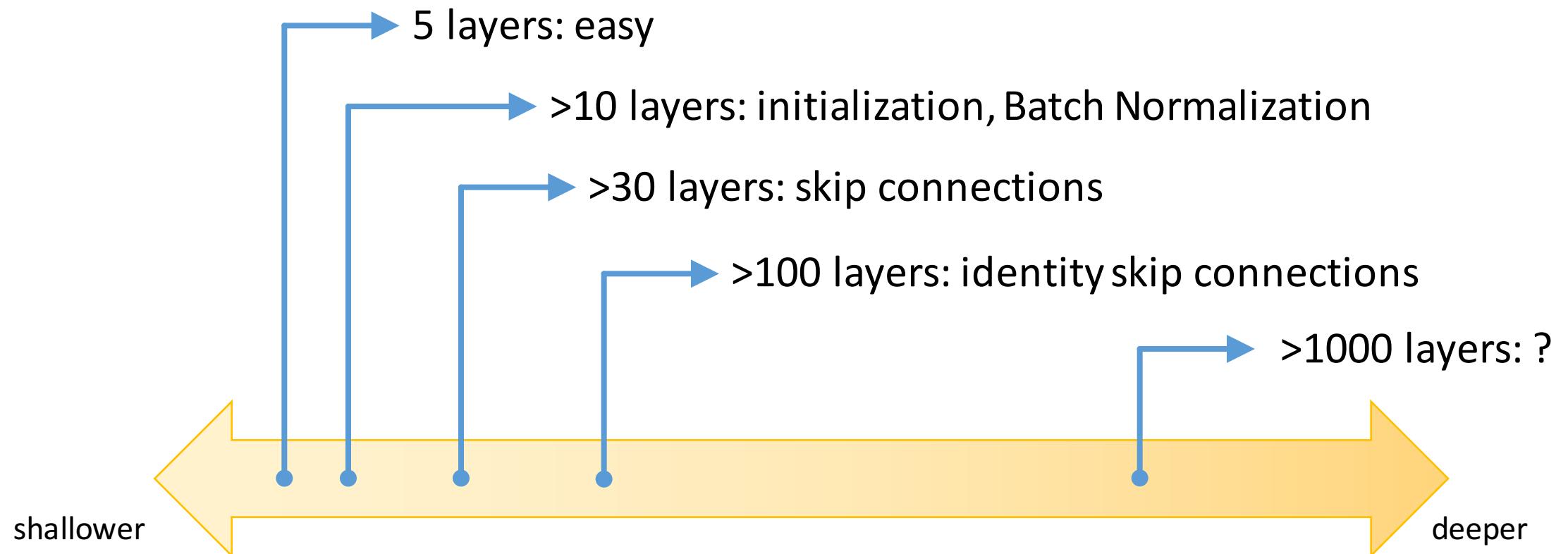


Repeat elementary layers => Going deeper

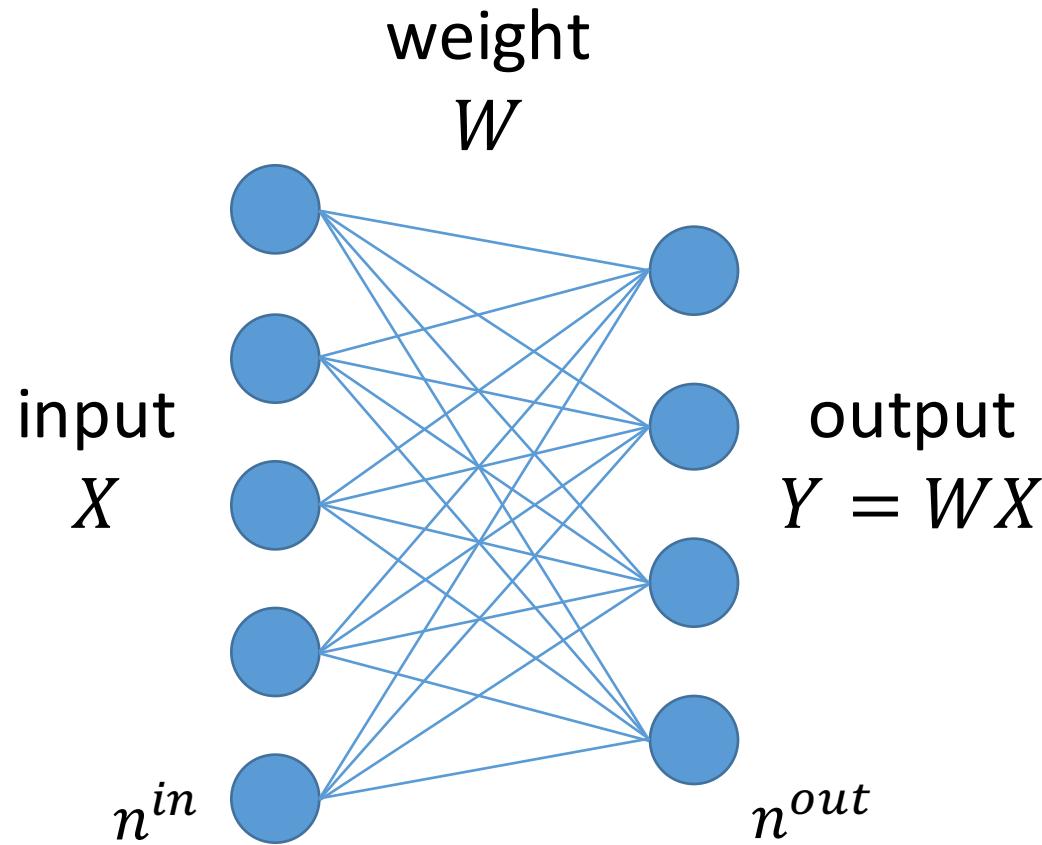


- End-to-end learning
- Richer solution space

Spectrum of Depth



Initialization



If:

- Linear activation
- x, y, w : independent

Then:

1-layer:

$$Var[y] = (n^{in} Var[w]) Var[x]$$

Multi-layer:

$$Var[y] = \left(\prod_d n_d^{in} Var[w_d] \right) Var[x]$$

Initialization

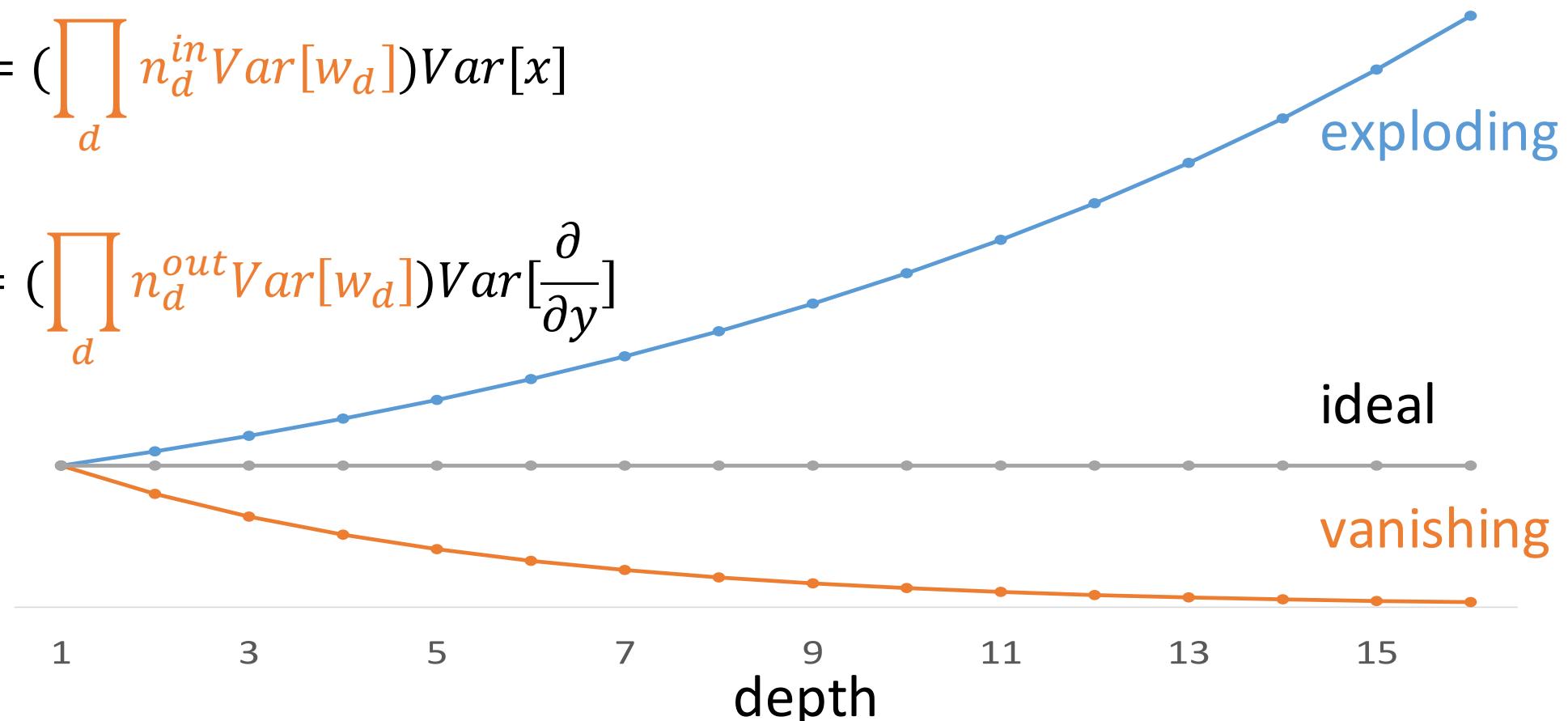
Both forward (response) and backward (gradient) signal can vanish/explode

Forward:

$$Var[y] = \left(\prod_d n_d^{in} Var[w_d] \right) Var[x]$$

Backward:

$$Var\left[\frac{\partial}{\partial x}\right] = \left(\prod_d n_d^{out} Var[w_d] \right) Var\left[\frac{\partial}{\partial y}\right]$$



LeCun et al 1998 "Efficient Backprop"

Glorot & Bengio 2010 "Understanding the difficulty of training deep feedforward neural networks"

Initialization

- Initialization under **linear** assumption

$$\prod_d n_d^{in} \text{Var}[w_d] = \text{const}_{fw} \text{ (healthy forward)}$$

and

$$\prod_d n_d^{out} \text{Var}[w_d] = \text{const}_{bw} \text{ (healthy backward)}$$



$$n_d^{in} \text{Var}[w_d] = 1$$

or*

$$n_d^{out} \text{Var}[w_d] = 1$$

*: $n_d^{out} = n_{d+1}^{in}$, so $\frac{\text{const}_{bw}}{\text{const}_{fw}} = \frac{n_{last}^{out}}{n_{first}^{in}} < \infty$.

It is sufficient to use either form.

“Xavier” init in Caffe

LeCun et al 1998 “Efficient Backprop”

Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”

Initialization

- Initialization under **ReLU** activation

$$\prod_d \frac{1}{2} n_d^{in} Var[w_d] = const_{fw} \text{ (healthy forward)}$$

and

$$\prod_d \frac{1}{2} n_d^{out} Var[w_d] = const_{bw} \text{ (healthy backward)}$$

$$\frac{1}{2} n_d^{in} Var[w_d] = 1$$

or

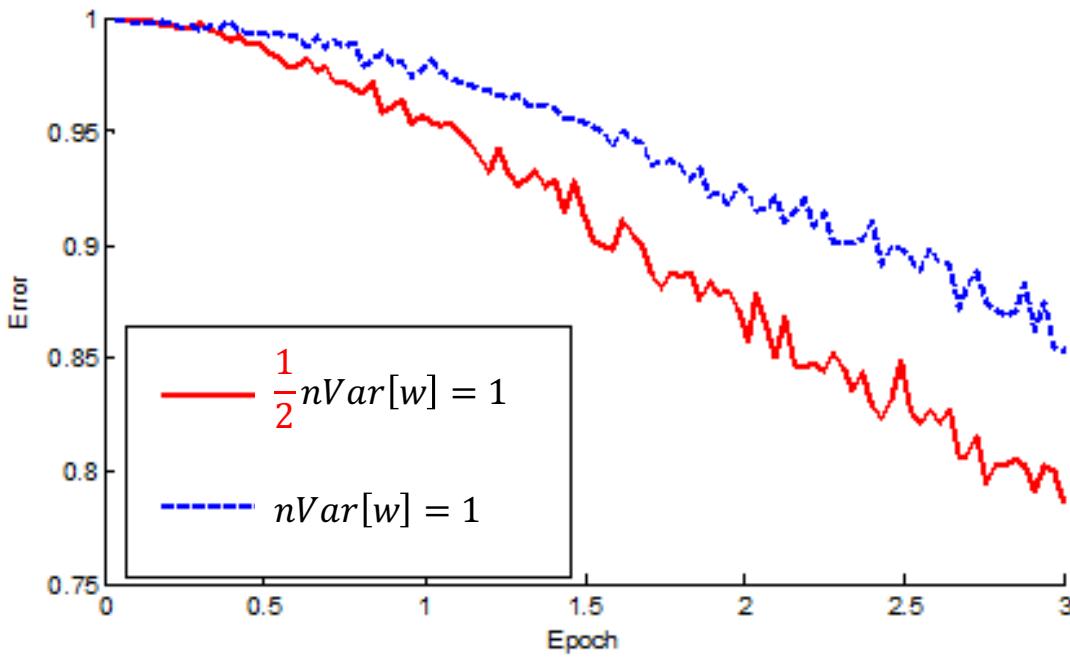
$$\frac{1}{2} n_d^{out} Var[w_d] = 1$$

With D layers, a factor of 2 per layer has exponential impact of 2^D

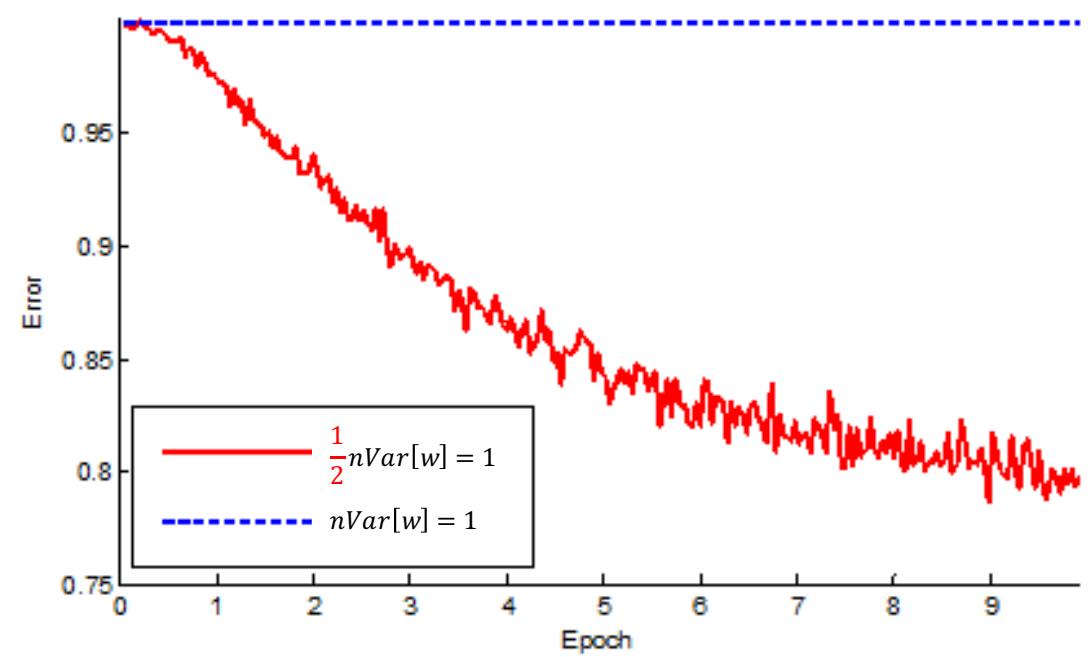
“MSRA” init in Caffe

Initialization

22-layer ReLU net:
good init converges faster



30-layer ReLU net:
good init is able to converge

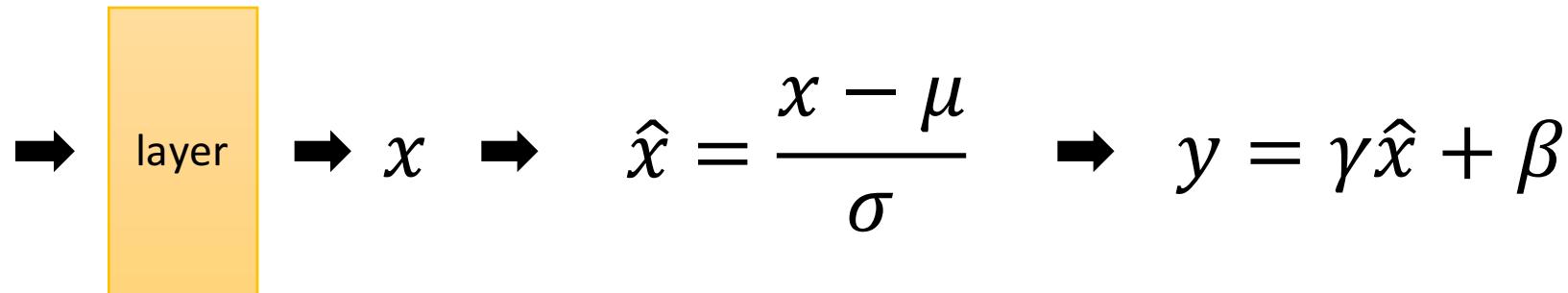


*Figures show the beginning of training

Batch Normalization (BN)

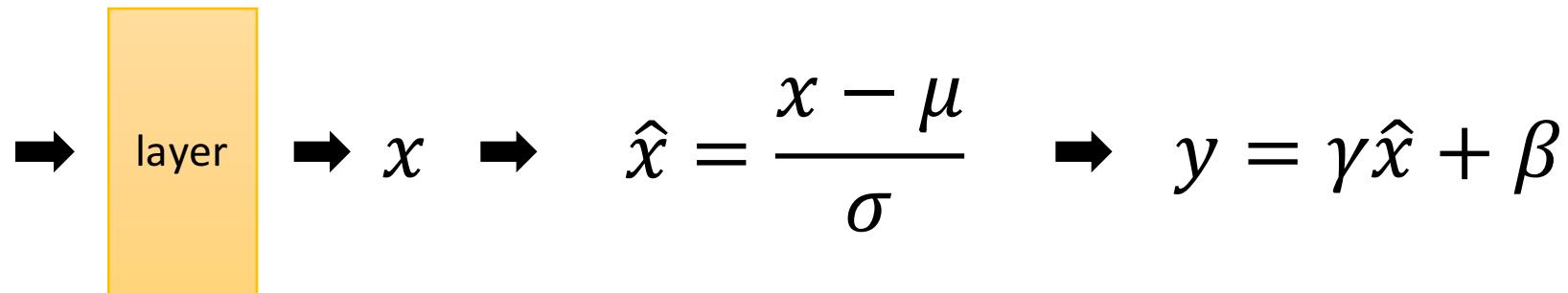
- Normalizing input (LeCun et al 1998 “Efficient Backprop”)
- BN: normalizing **each layer**, for **each mini-batch**
- Greatly accelerate training
- Less sensitive to initialization
- Improve regularization

Batch Normalization (BN)



- μ : mean of x in mini-batch
- σ : std of x in mini-batch
- γ : scale
- β : shift
- μ, σ : functions of x ,
analogous to responses
- γ, β : parameters to be learned,
analogous to weights

Batch Normalization (BN)



2 modes of BN:

- Train mode:
 - μ, σ are functions of x ; backprop gradients
- Test mode:
 - μ, σ are pre-computed* on training set

Caution: make sure your BN
is in a correct mode

*: by running average, or post-processing after training

Batch Normalization (BN)

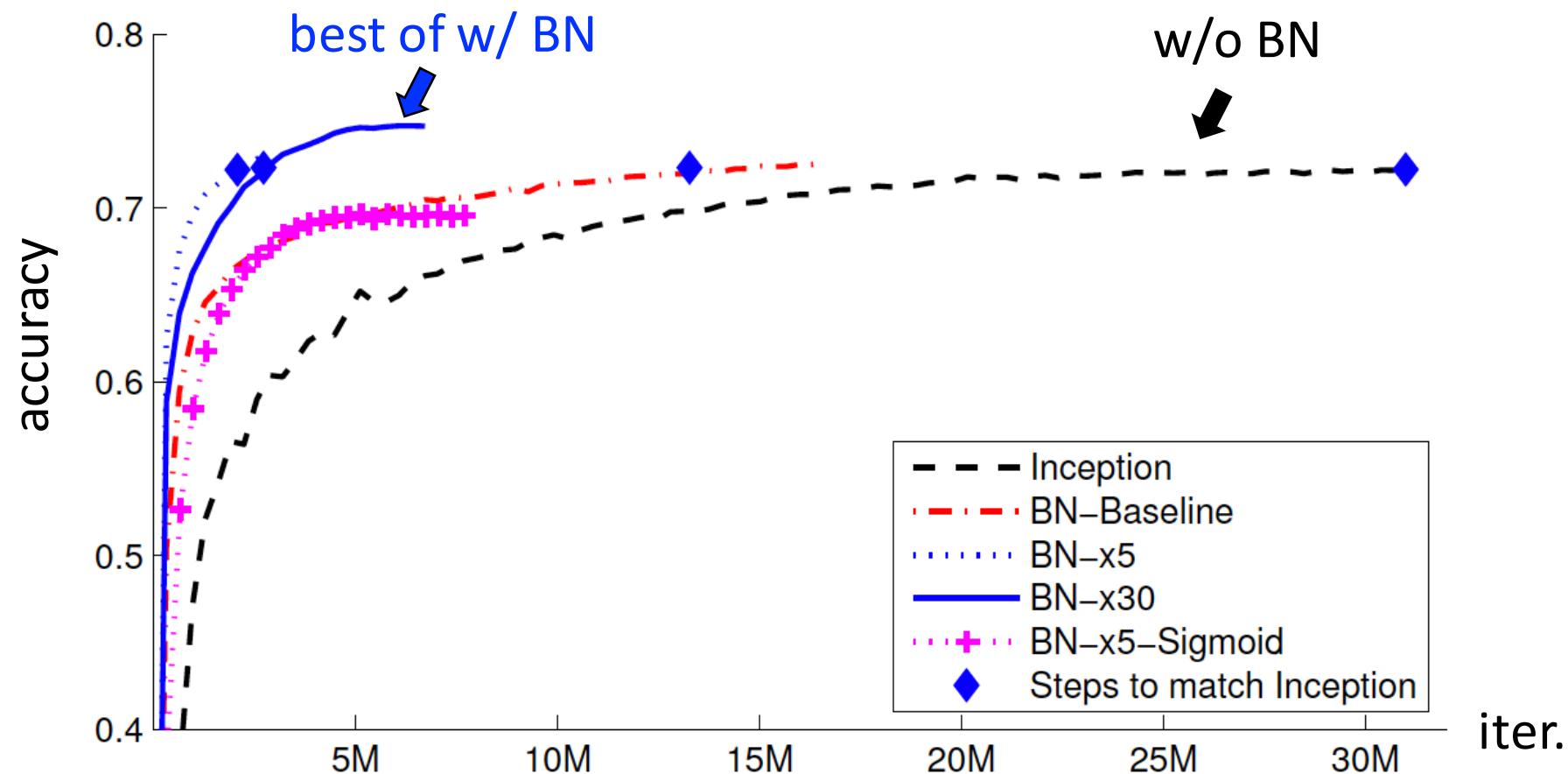


Figure taken from [S. Ioffe & C. Szegedy]

Deep Residual Networks

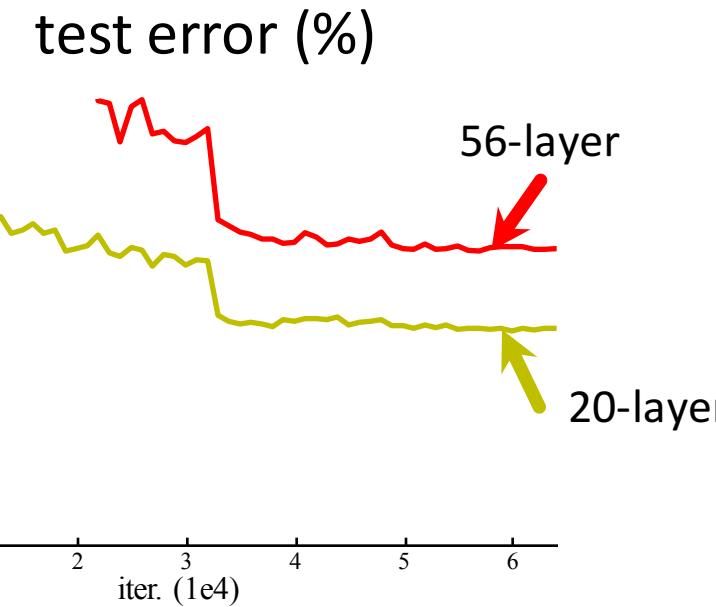
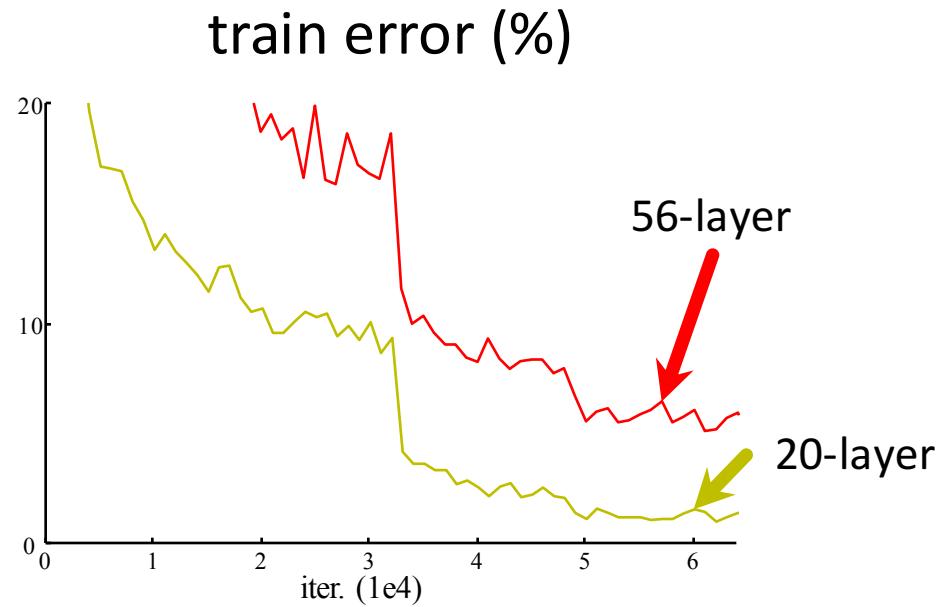
From 10 layers to 100 layers

Going Deeper

- Initialization algorithms ✓
- Batch Normalization ✓
- **Is learning better networks as simple as stacking more layers?**

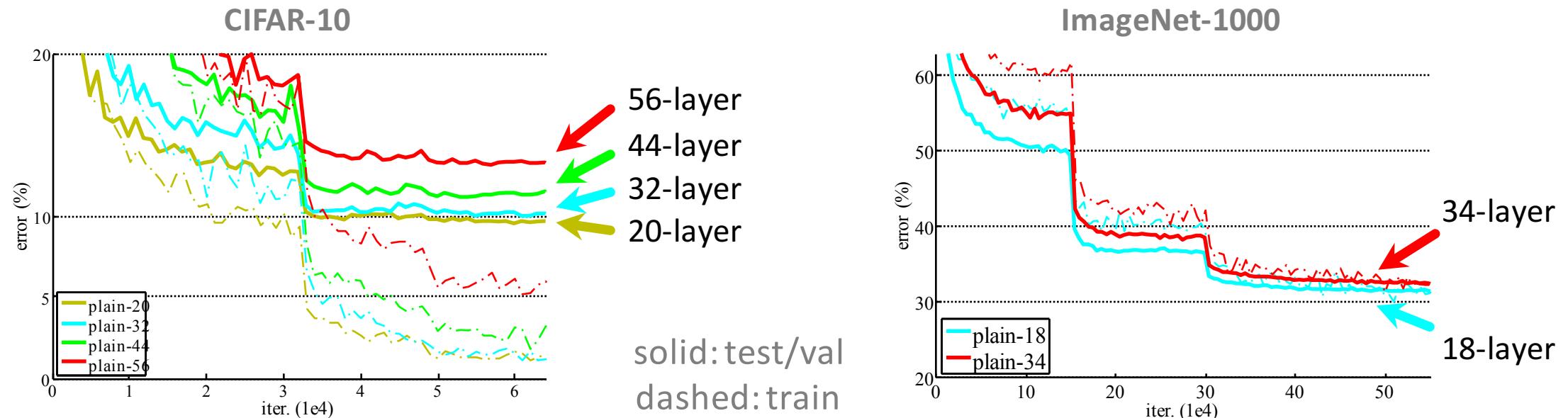
Simply stacking layers?

CIFAR-10



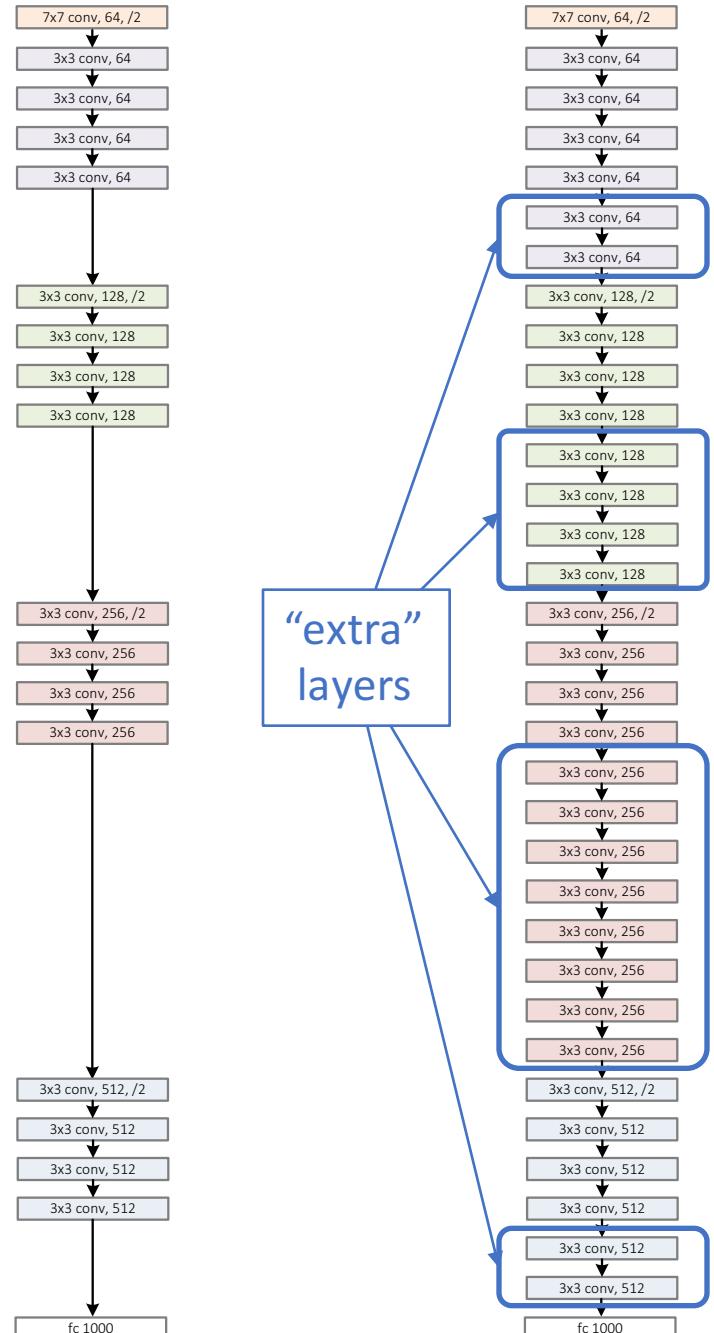
- *Plain* nets: stacking 3×3 conv layers...
- 56-layer net has **higher training error** and test error than 20-layer net

Simply stacking layers?



- “Overly deep” plain nets have **higher training error**
- A general phenomenon, observed in many datasets

a shallower
model
(18 layers)

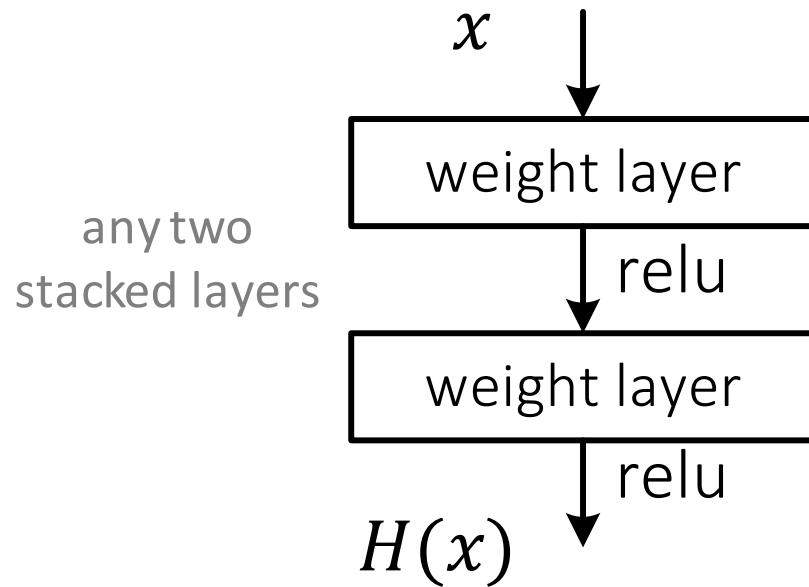


a deeper
counterpart
(34 layers)

- 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...

Deep Residual Learning

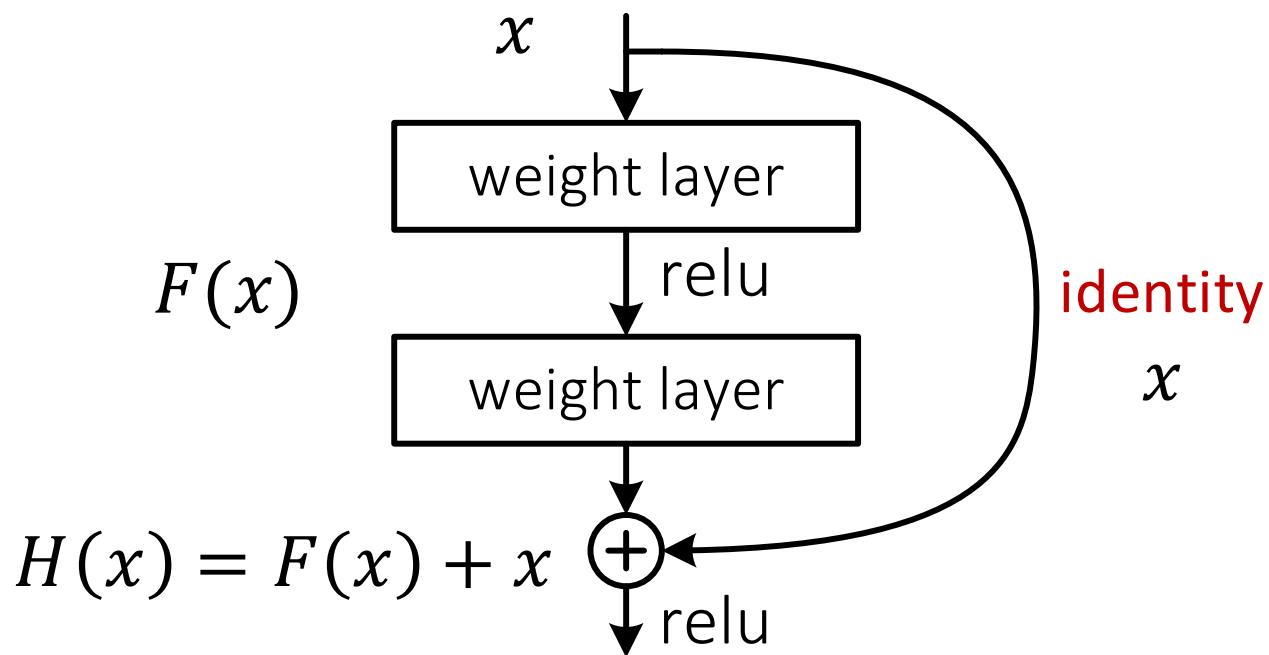
- Plain net



$H(x)$ is any desired mapping,
hope the 2 weight layers fit $H(x)$

Deep Residual Learning

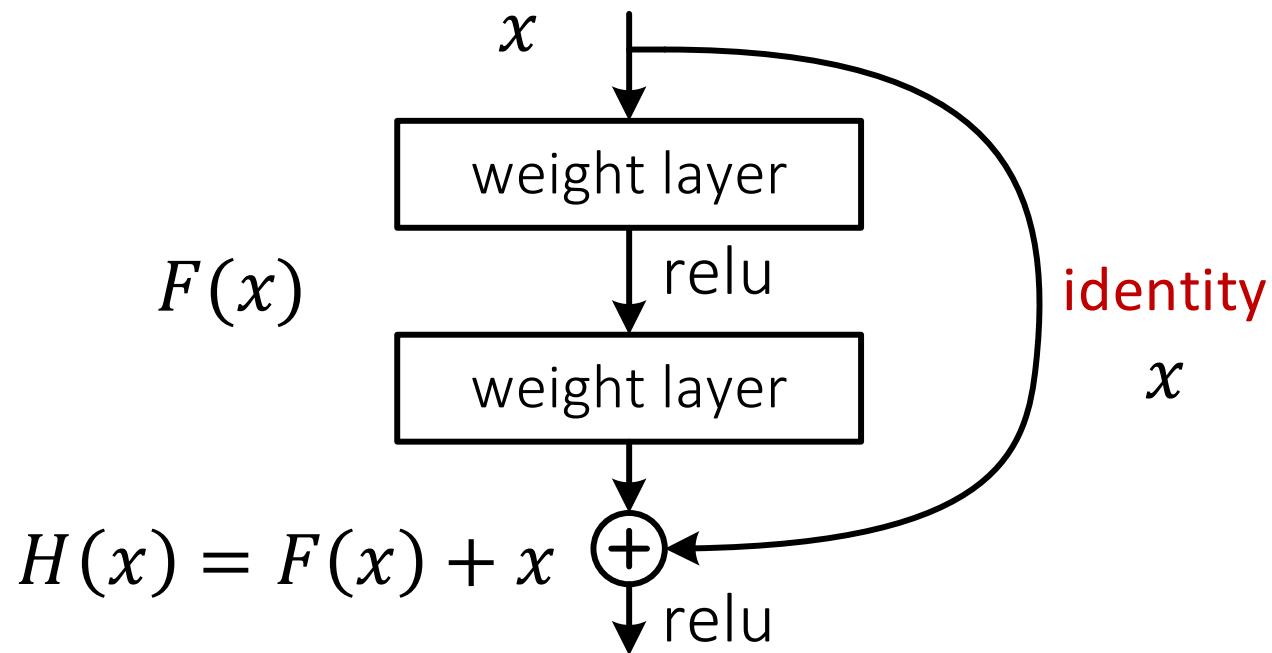
- **Residual net**



$H(x)$ is any desired mapping,
hope the 2 weight layers fit $H(x)$
hope the 2 weight layers fit $F(x)$
let $H(x) = F(x) + x$

Deep Residual Learning

- $F(x)$ is a **residual mapping w.r.t. identity**



- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

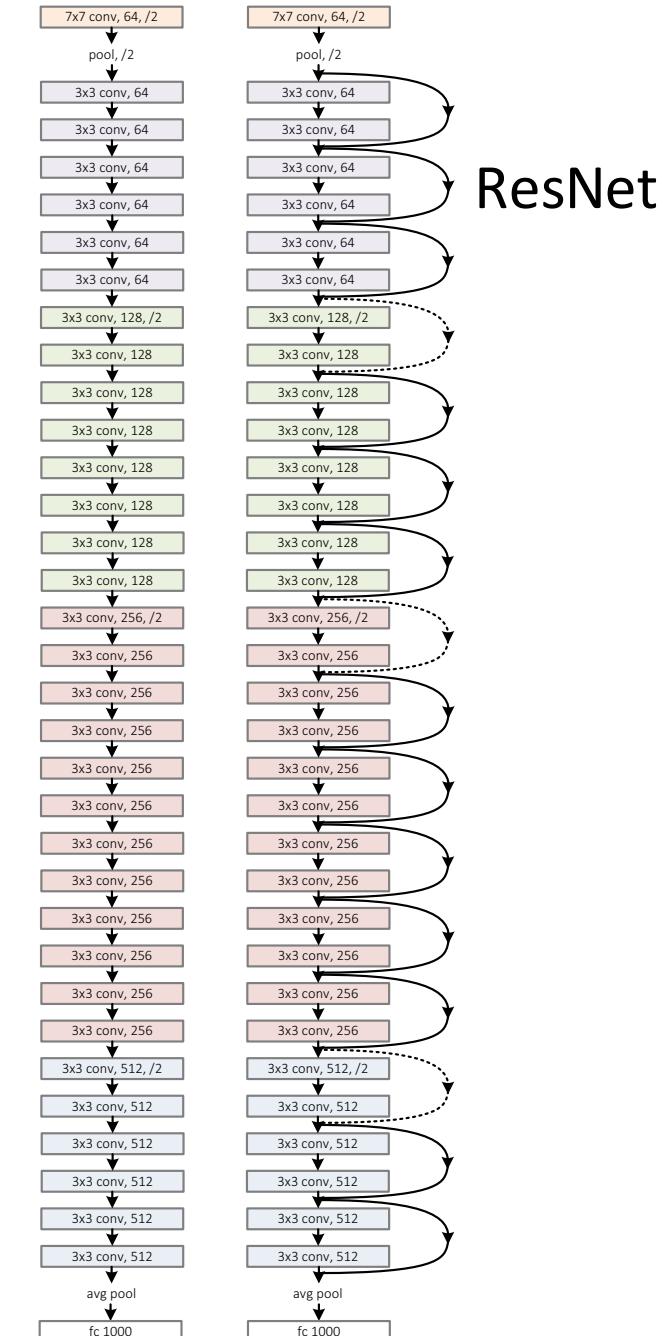
Related Works – Residual Representations

- VLAD & Fisher Vector [Jegou et al 2010], [Perronnin et al 2007]
 - Encoding **residual** vectors; powerful shallower representations.
- Product Quantization (IVF-ADC) [Jegou et al 2011]
 - Quantizing **residual** vectors; efficient nearest-neighbor search.
- MultiGrid & Hierarchical Precondition [Briggs, et al 2000], [Szeliski 1990, 2006]
 - Solving **residual** sub-problems; efficient PDE solvers.

Network “Design”

- Keep it simple
- Our basic design (VGG-style)
 - all 3x3 conv (almost)
 - spatial size /2 => # filters x2 (~same complexity per layer)
 - **Simple design; just deep!**
- Other remarks:
 - no hidden fc
 - no dropout

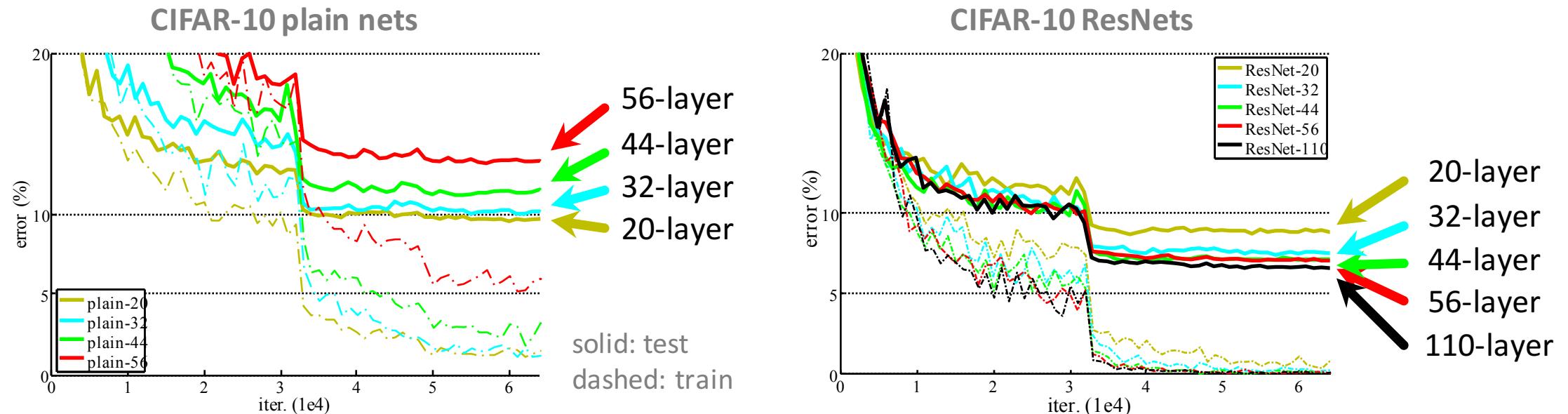
plain net



Training

- All plain/residual nets are trained **from scratch**
- All plain/residual nets use Batch Normalization
- Standard hyper-parameters & augmentation

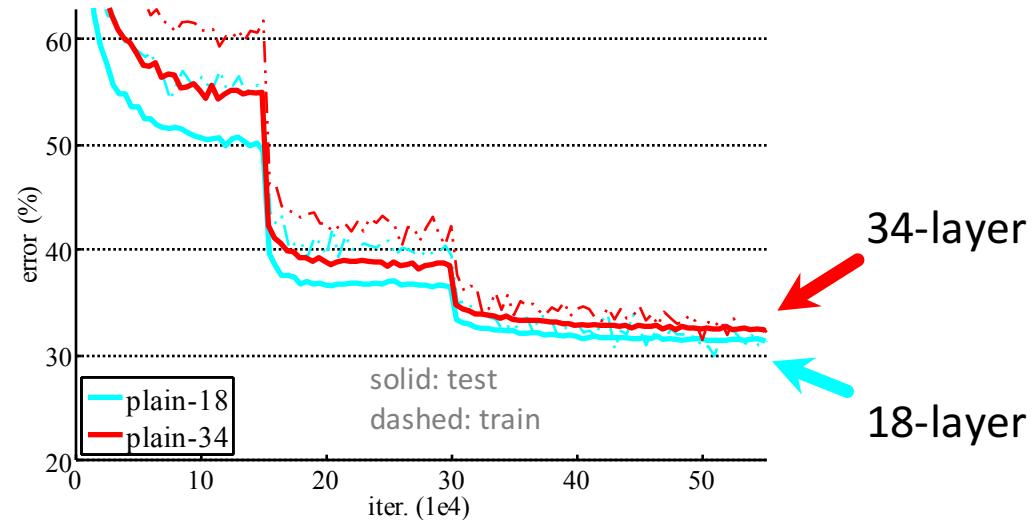
CIFAR-10 experiments



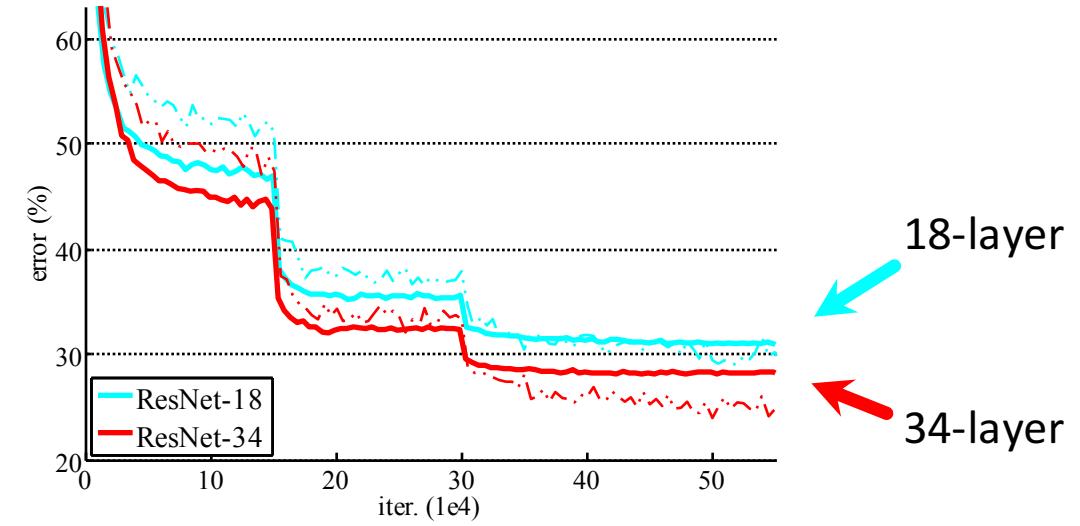
- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

ImageNet experiments

ImageNet plain nets



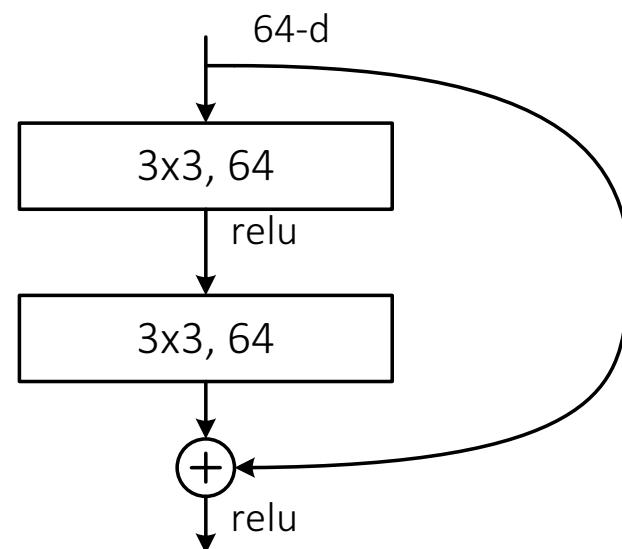
ImageNet ResNets



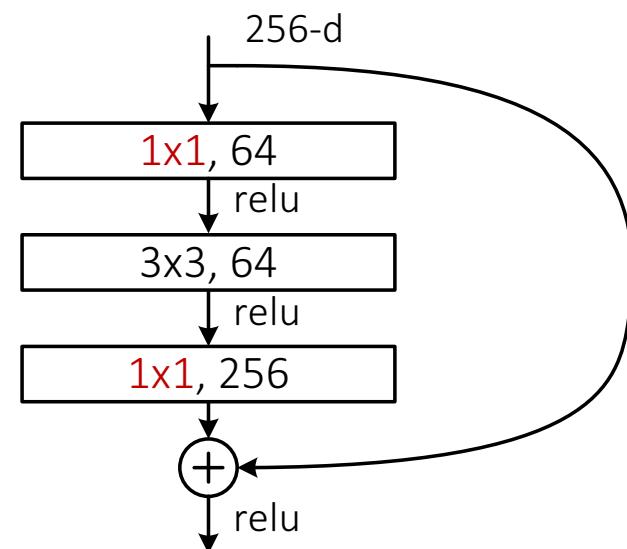
- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

ImageNet experiments

- A practical design of going deeper

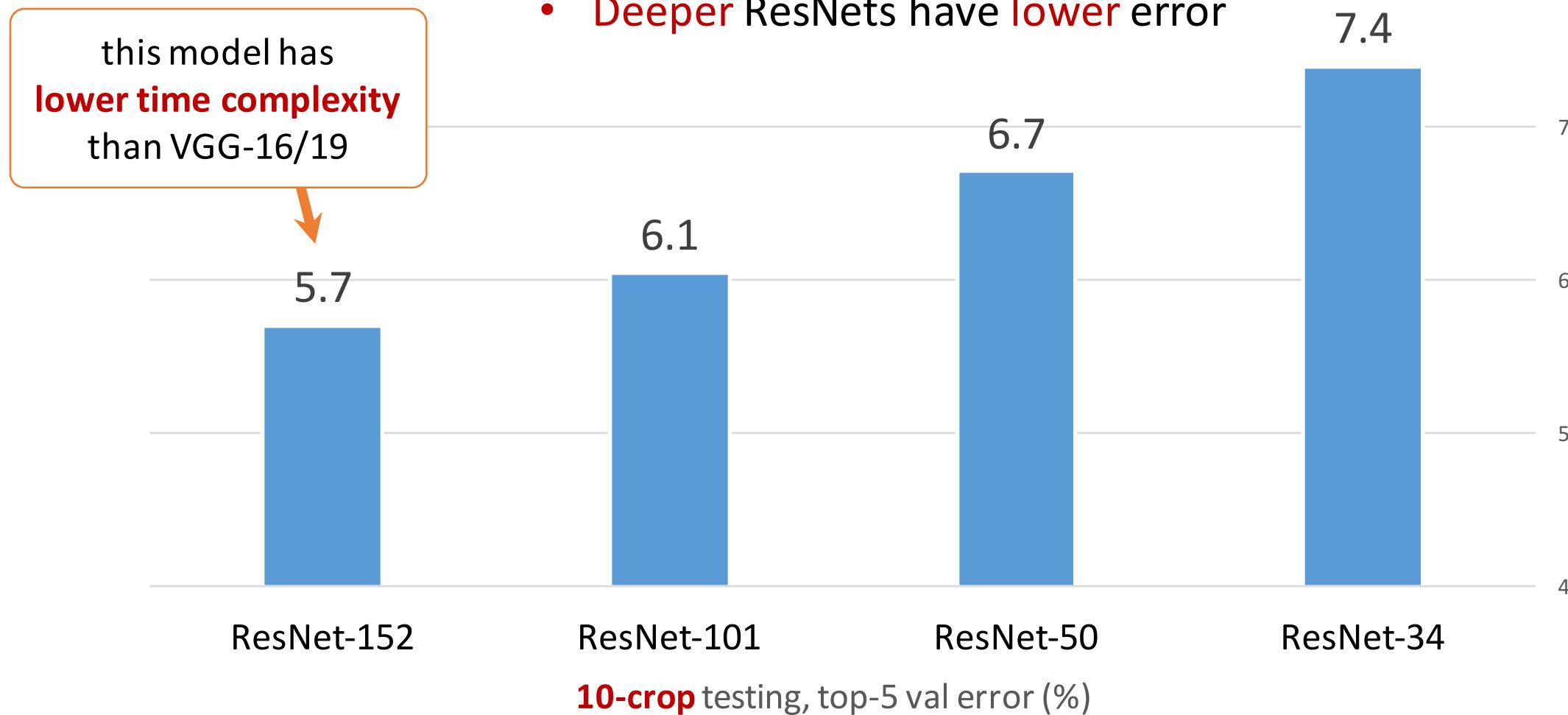


all-3x3

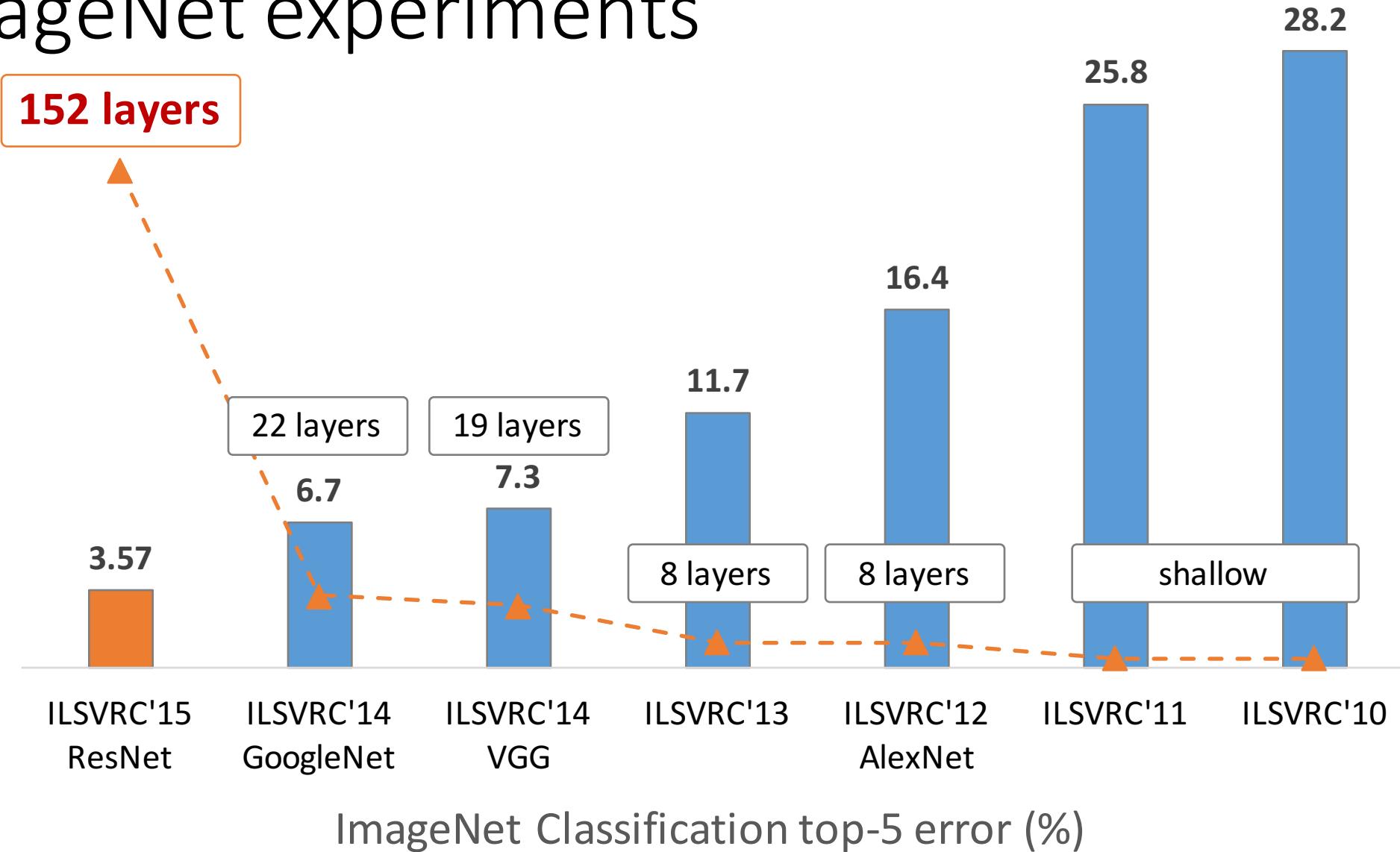


bottleneck
(for ResNet-50/101/152)

ImageNet experiments



ImageNet experiments



Discussions

Representation, Optimization, Generalization

Issues on learning deep models

- **Representation** ability

- Ability of model to fit training data, if optimum could be found
- If model A's solution space is a superset of B's, A should be better.

- **Optimization** ability

- Feasibility of finding an optimum
- Not all models are equally easy to optimize

- **Generalization** ability

- Once training data is fit, how good is the test performance

How do ResNets address these issues?

- **Representation** ability

- No explicit advantage on representation (only re-parameterization), but
- Allow models to go **deeper**

- **Optimization** ability

- Enable very smooth forward/backward prop
- Greatly ease optimizing **deeper** models

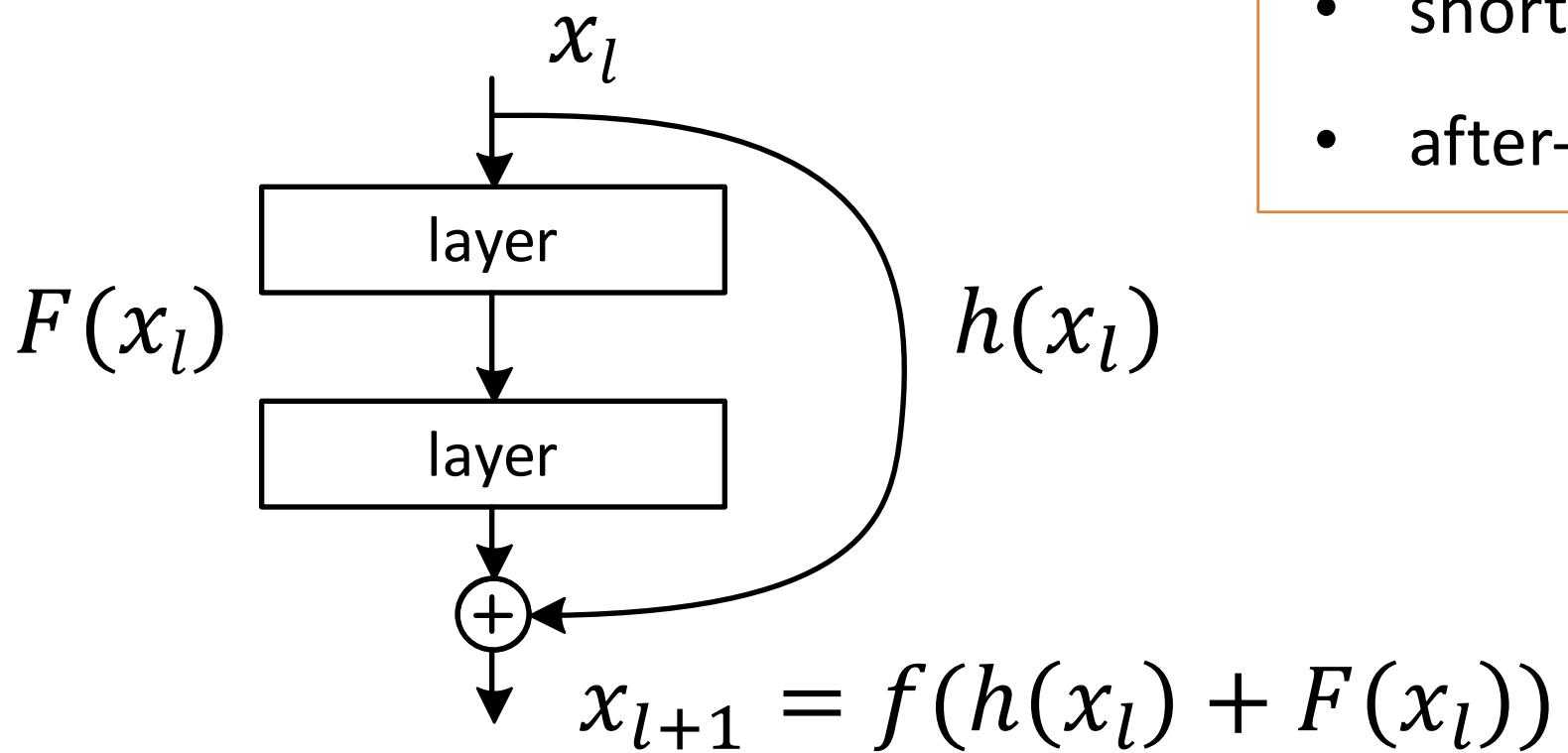
- **Generalization** ability

- Not explicitly address generalization, but
- **Deeper**+thinner is good generalization

On the Importance of Identity Mapping

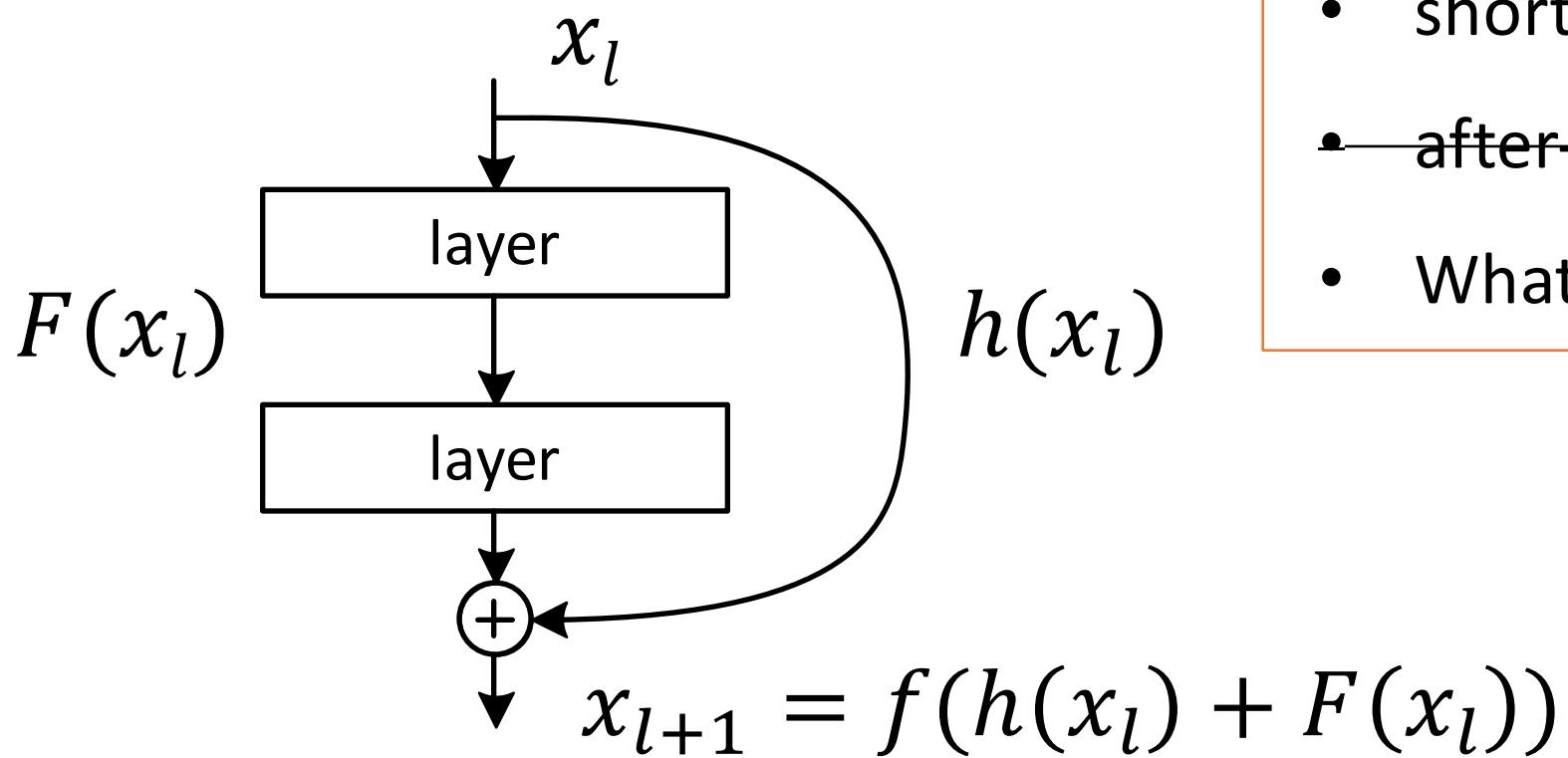
From 100 layers to 1000 layers

On identity mappings for optimization



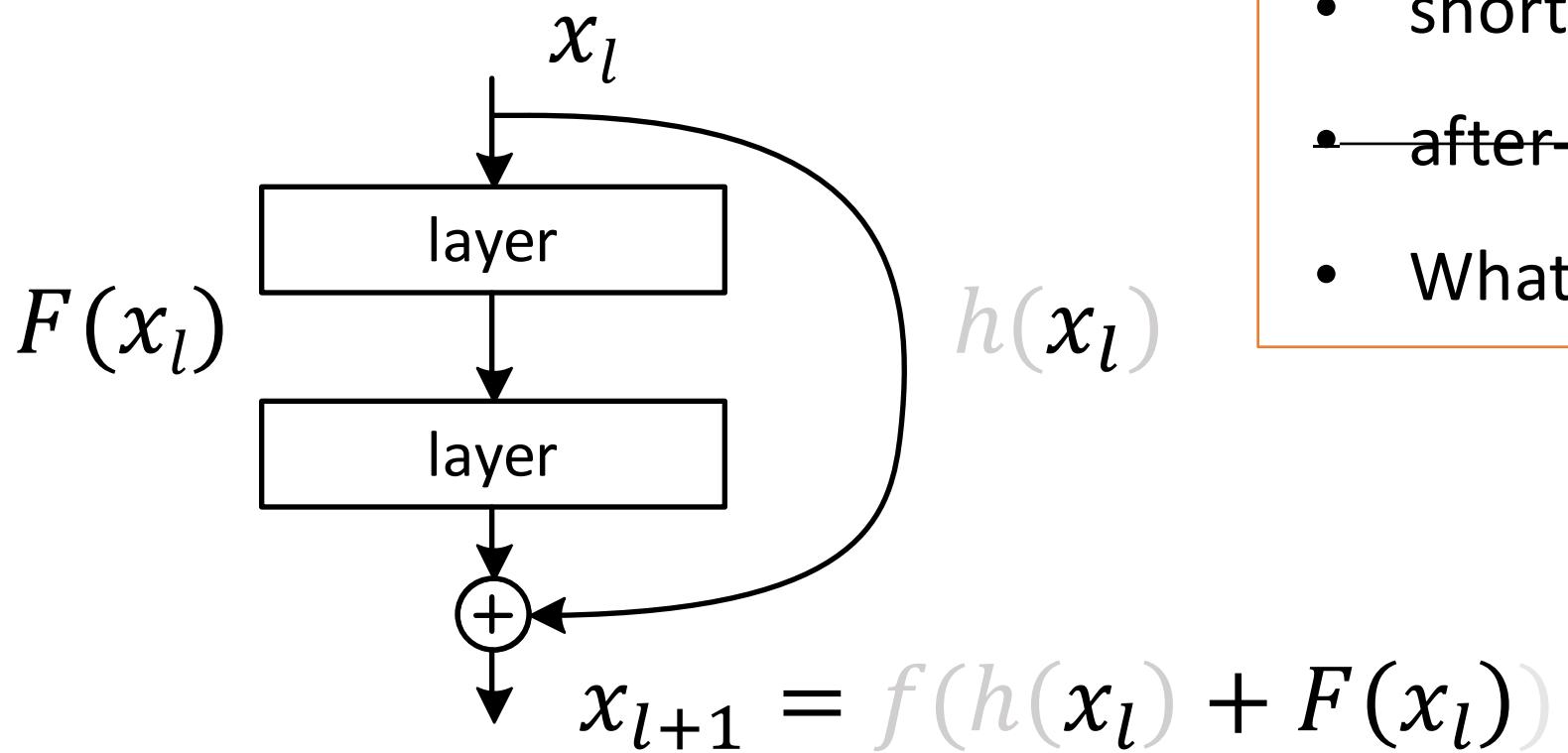
- shortcut mapping: $h = \text{identity}$
- after-add mapping: $f = \text{ReLU}$

On identity mappings for optimization



- shortcut mapping: $h = \text{identity}$
- after-add mapping: $f = \text{ReLU}$
- What if $f = \text{identity}$?

On identity mappings for optimization



- shortcut mapping: $h = \text{identity}$
- after-add mapping: $f = \text{ReLU}$
- What if $f = \text{identity}$?

Very smooth forward propagation

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



$$x_{l+2} = x_{l+1} + F(x_{l+1})$$

Very smooth forward propagation

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



$$x_{l+2} = x_{l+1} + F(x_{l+1})$$

$$x_{l+2} = x_l + F(x_l) + F(x_{l+1})$$

Very smooth forward propagation

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



$$x_{l+2} = x_{l+1} + F(x_{l+1})$$

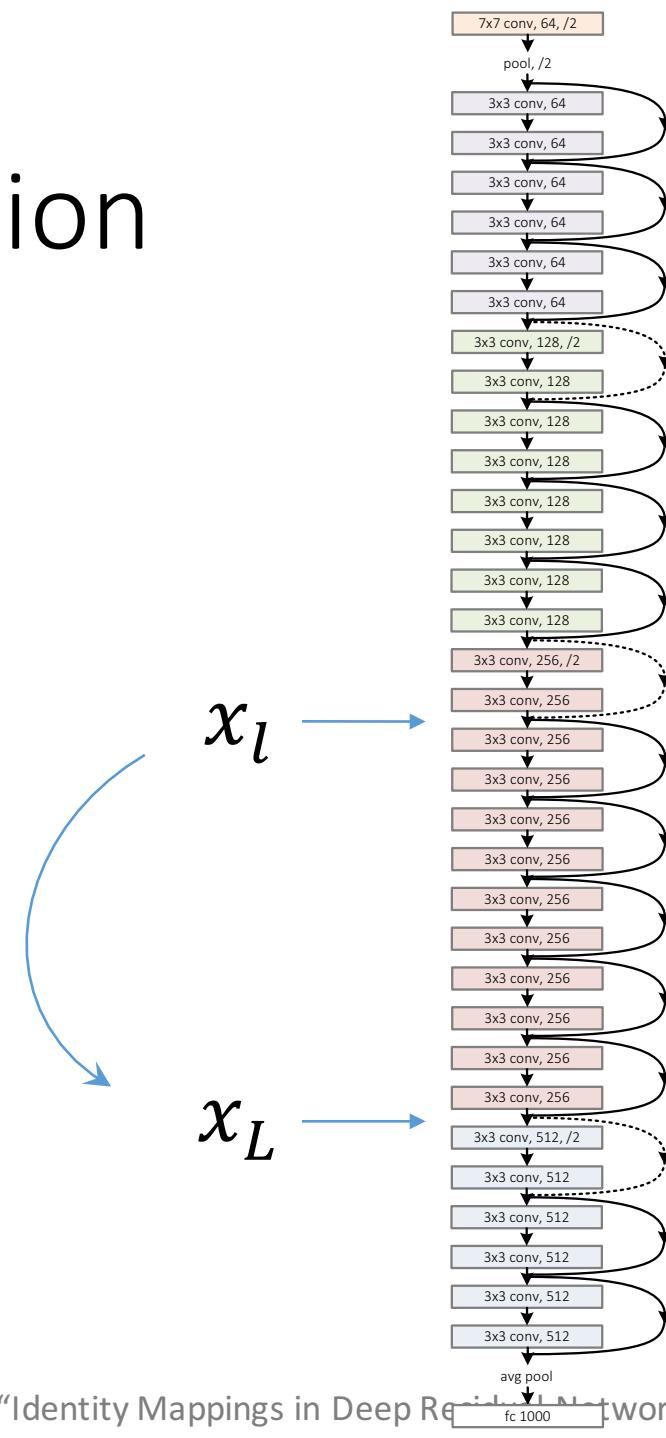
$$x_{l+2} = x_l + F(x_l) + F(x_{l+1})$$

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

Very smooth forward propagation

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

- Any x_l is **directly** forward-prop to any x_L , plus **residual**.
- Any x_L is an **additive** outcome.
 - in contrast to **multiplicative**: $x_L = \prod_{i=l}^{L-1} W_i x_l$



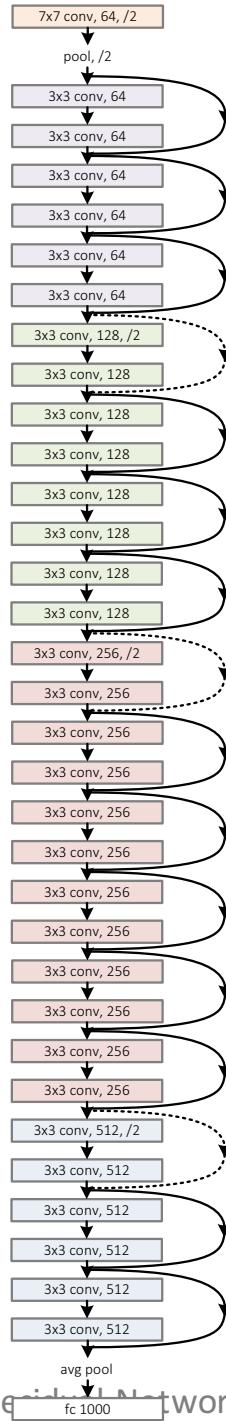
Very smooth backward propagation

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



$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L} \frac{\partial x_L}{\partial x_l} = \frac{\partial E}{\partial x_L} \left(1 + \frac{\partial}{\partial x_l} \sum_{i=1}^{L-1} F(x_i) \right)$$

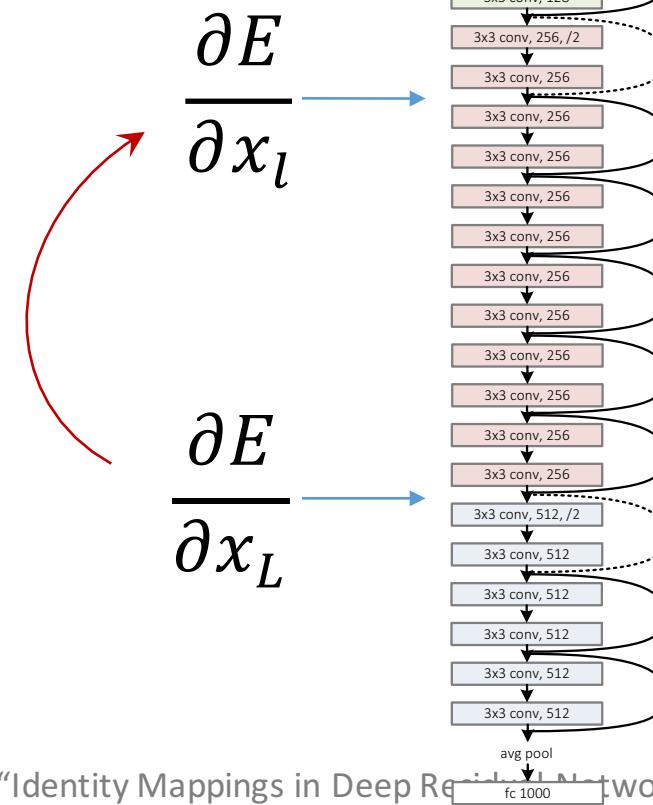
$$\frac{\partial E}{\partial x_l} \quad \frac{\partial E}{\partial x_L}$$



Very smooth backward propagation

$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L} \left(1 + \frac{\partial}{\partial x_l} \sum_{i=1}^{L-1} F(x_i) \right)$$

- Any $\frac{\partial E}{\partial x_L}$ is **directly** back-prop to any $\frac{\partial E}{\partial x_l}$, plus **residual**.
- Any $\frac{\partial E}{\partial x_l}$ is **additive**; unlikely to vanish
 - in contrast to **multiplicative**: $\frac{\partial E}{\partial x_l} = \prod_{i=l}^{L-1} W_i \frac{\partial E}{\partial x_L}$



Residual for every layer

$$\text{forward: } x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

Enabled by:

- shortcut mapping: $h = \text{identity}$
- after-add mapping: $f = \text{identity}$

$$\text{backward: } \frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L} \left(1 + \frac{\partial}{\partial x_l} \sum_{i=1}^{L-1} F(x_i) \right)$$

Experiments

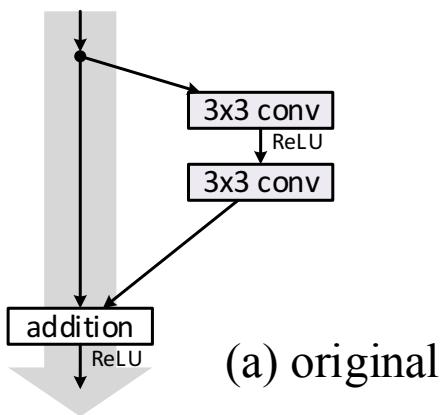
- Set 1: what if shortcut mapping $h \neq$ identity
- Set 2: what if after-add mapping f is identity
- Experiments on ResNets with more than 100 layers
 - deeper models suffer more from optimization difficulty

Experiment Set 1:
what if shortcut mapping $h \neq$ identity?

* ResNet-110 on CIFAR-10

$$h(x) = x$$

error: 6.6%

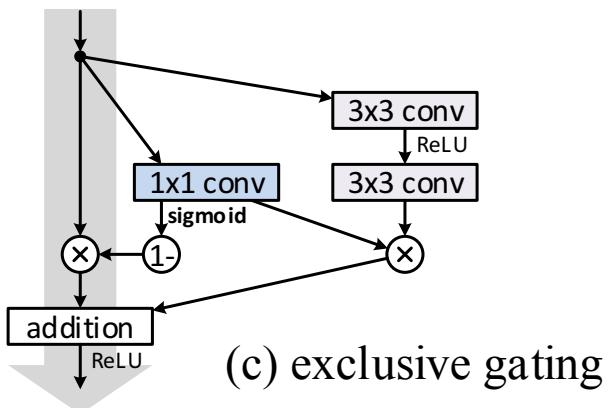


(a) original

$$h(x) = \text{gate} \cdot x$$

error: 8.7%

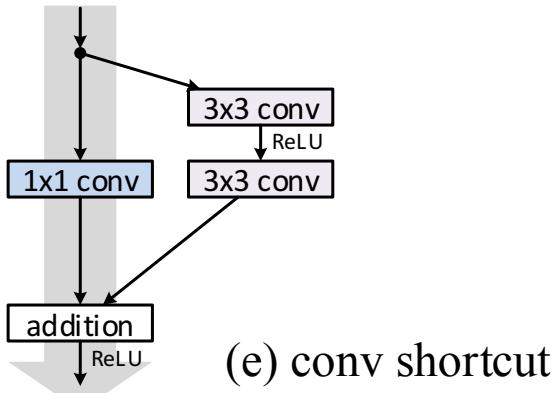
*similar to "Highway Network"



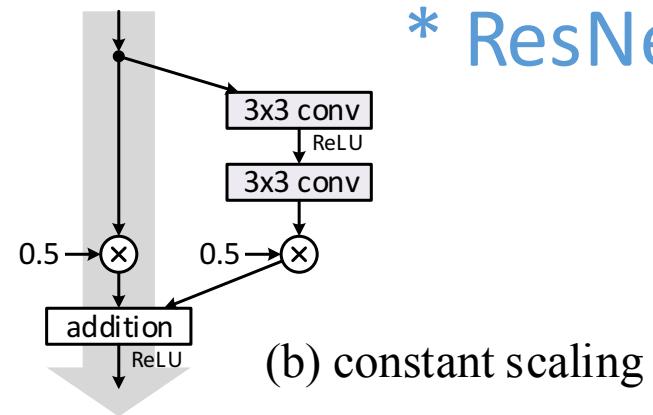
(c) exclusive gating

$$h(x) = \text{conv}(x)$$

error: 12.2%



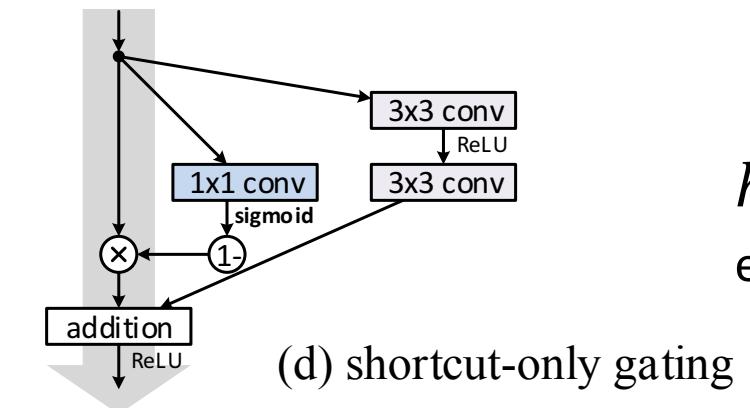
(e) conv shortcut



(b) constant scaling

$$h(x) = 0.5x$$

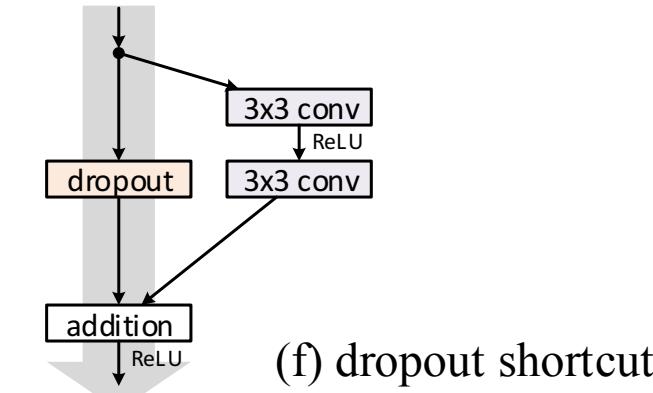
error: 12.4%



(d) shortcut-only gating

$$h(x) = \text{gate} \cdot x$$

error: 12.9%



(f) dropout shortcut

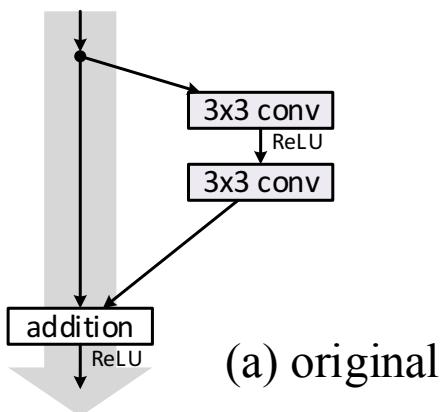
$$h(x) = \text{dropout}(x)$$

error: > 20%

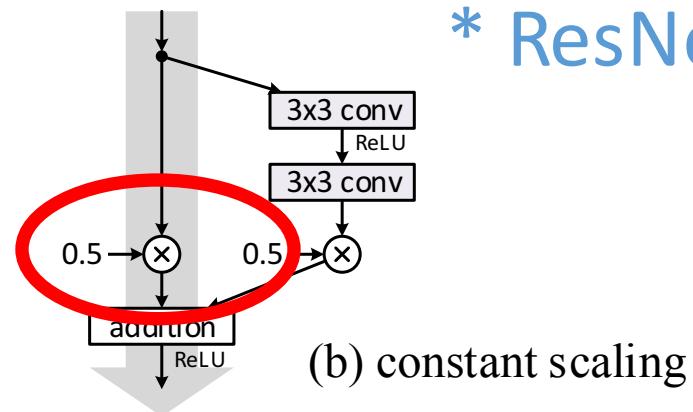
* ResNet-110 on CIFAR-10

$$h(x) = x$$

error: 6.6%



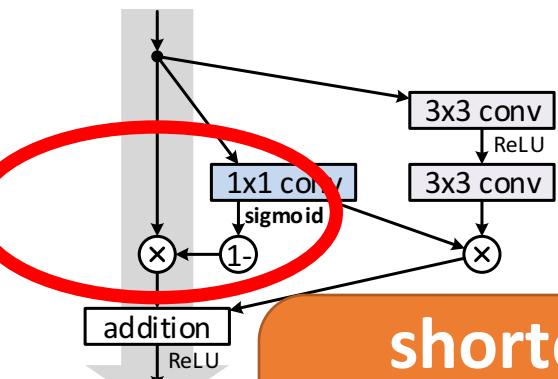
(a) original



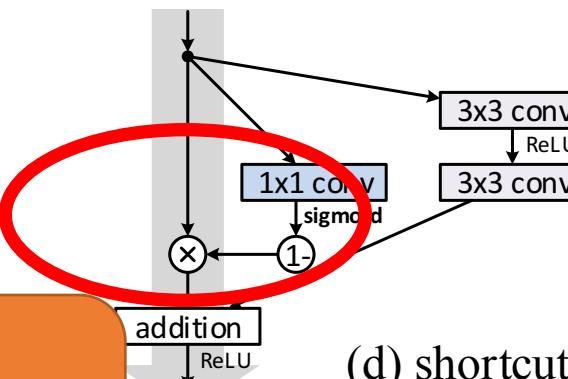
(b) constant scaling

$$h(x) = \text{gate} \cdot x$$

error: 8.7%



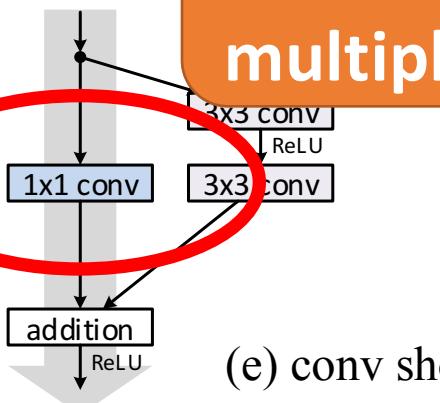
shortcuts
blocked by
multiplications



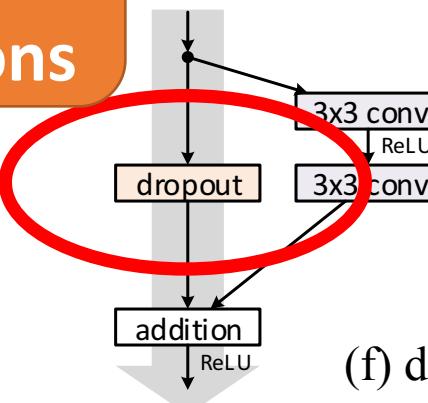
(d) shortcut-only gating

$$h(x) = \text{conv}(x)$$

error: 12.2%



(e) conv shortcut



(f) dropout shortcut

$$h(x) = 0.5x$$

error: 12.4%

$$h(x) = \text{gate} \cdot x$$

error: 12.9%

$$h(x) = \text{dropout}(x)$$

error: > 20%

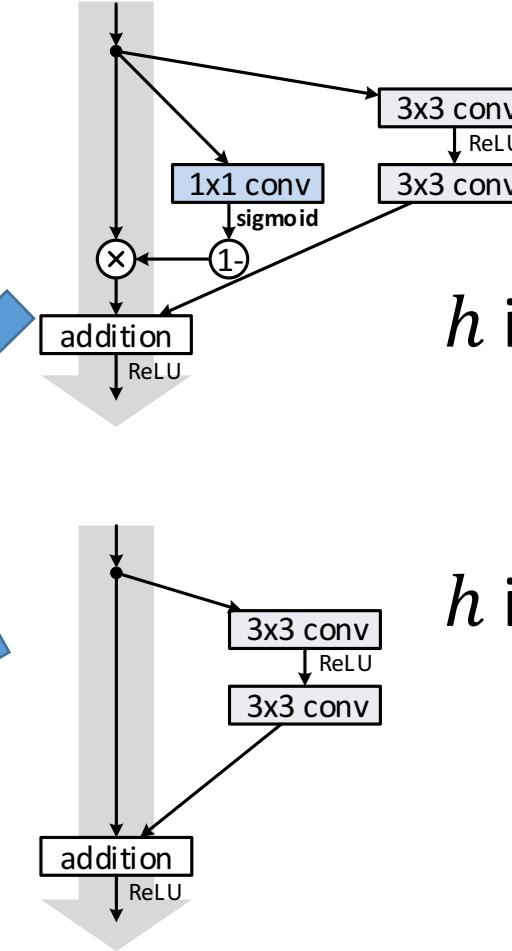
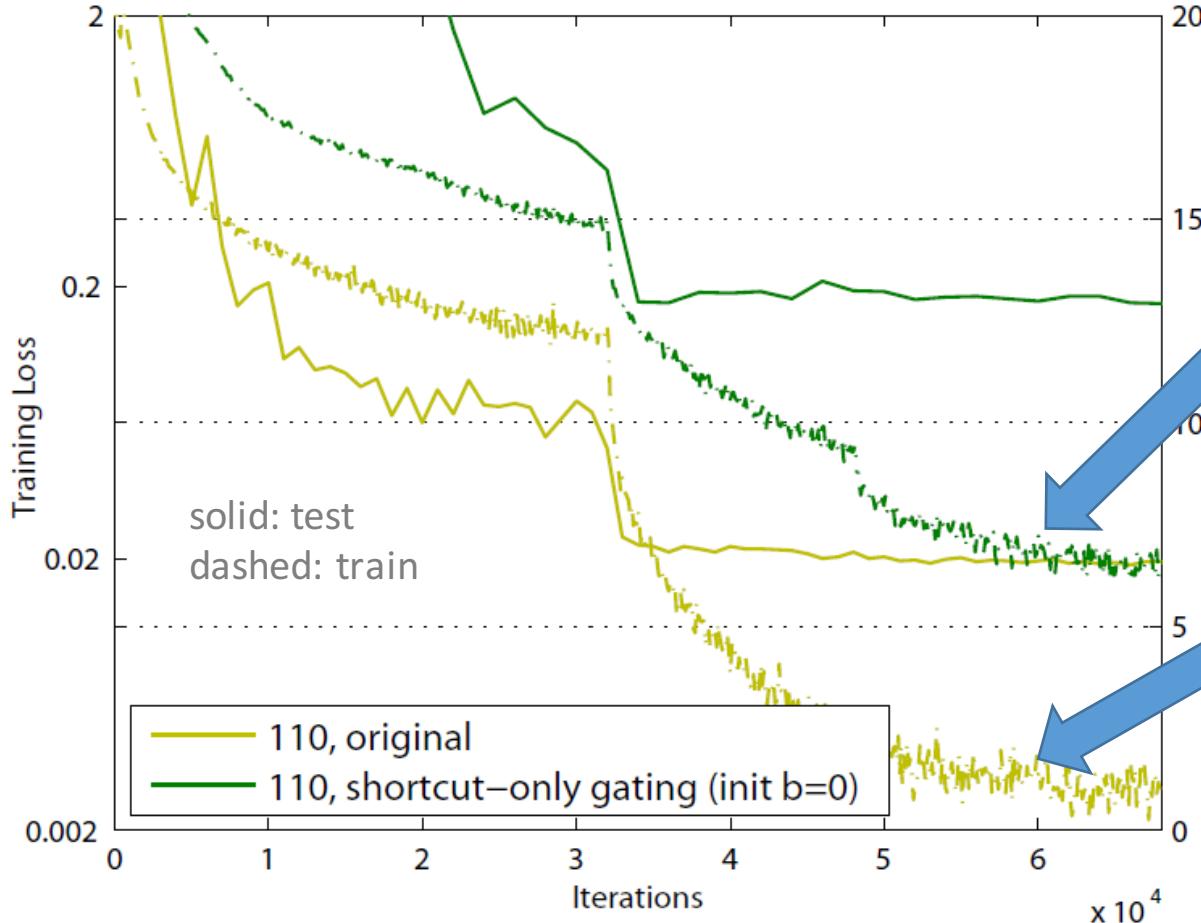
If h is multiplicative, e.g. $h(x) = \lambda x$

forward: $x_L = \lambda^{L-l} x_l + \sum_{i=l}^{L-1} \hat{F}(x_i)$

- if h is multiplicative, shortcuts are blocked
- direct propagation is decayed

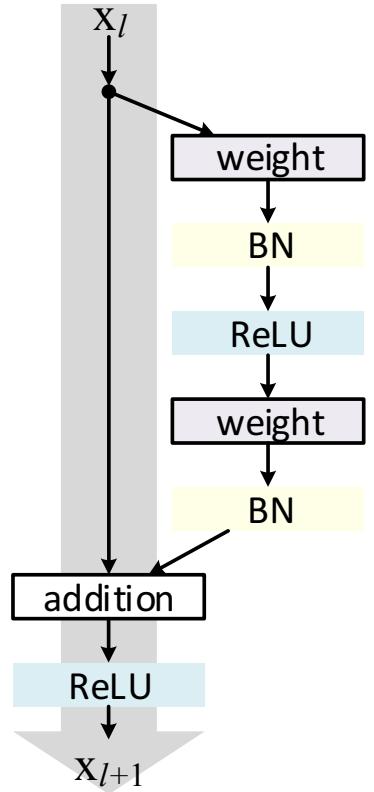
backward: $\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L} (\lambda^{L-l} + \frac{\partial}{\partial x_l} \sum_{i=1}^{L-1} \hat{F}(x_i))$

*assuming f = identity

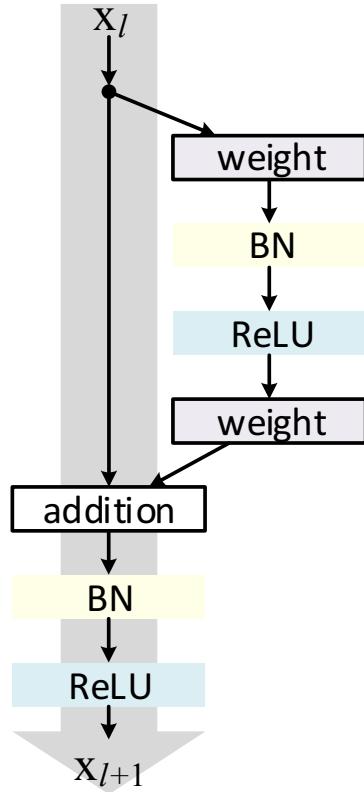


- gating should have better representation ability (identity is a special case), but
- optimization difficulty dominates results

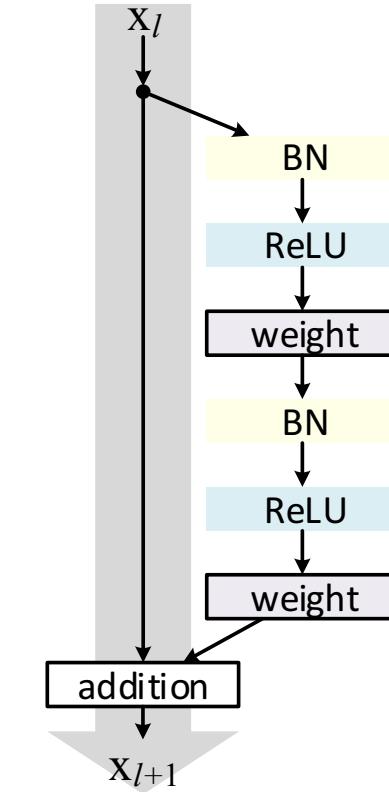
Experiment Set 2:
what if after-add mapping f is identity



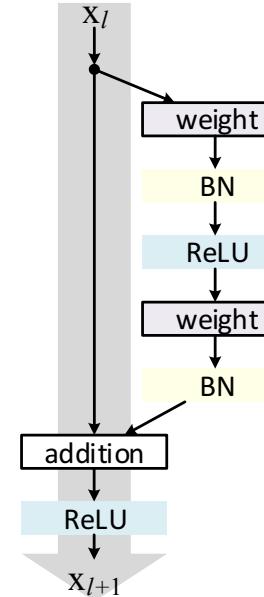
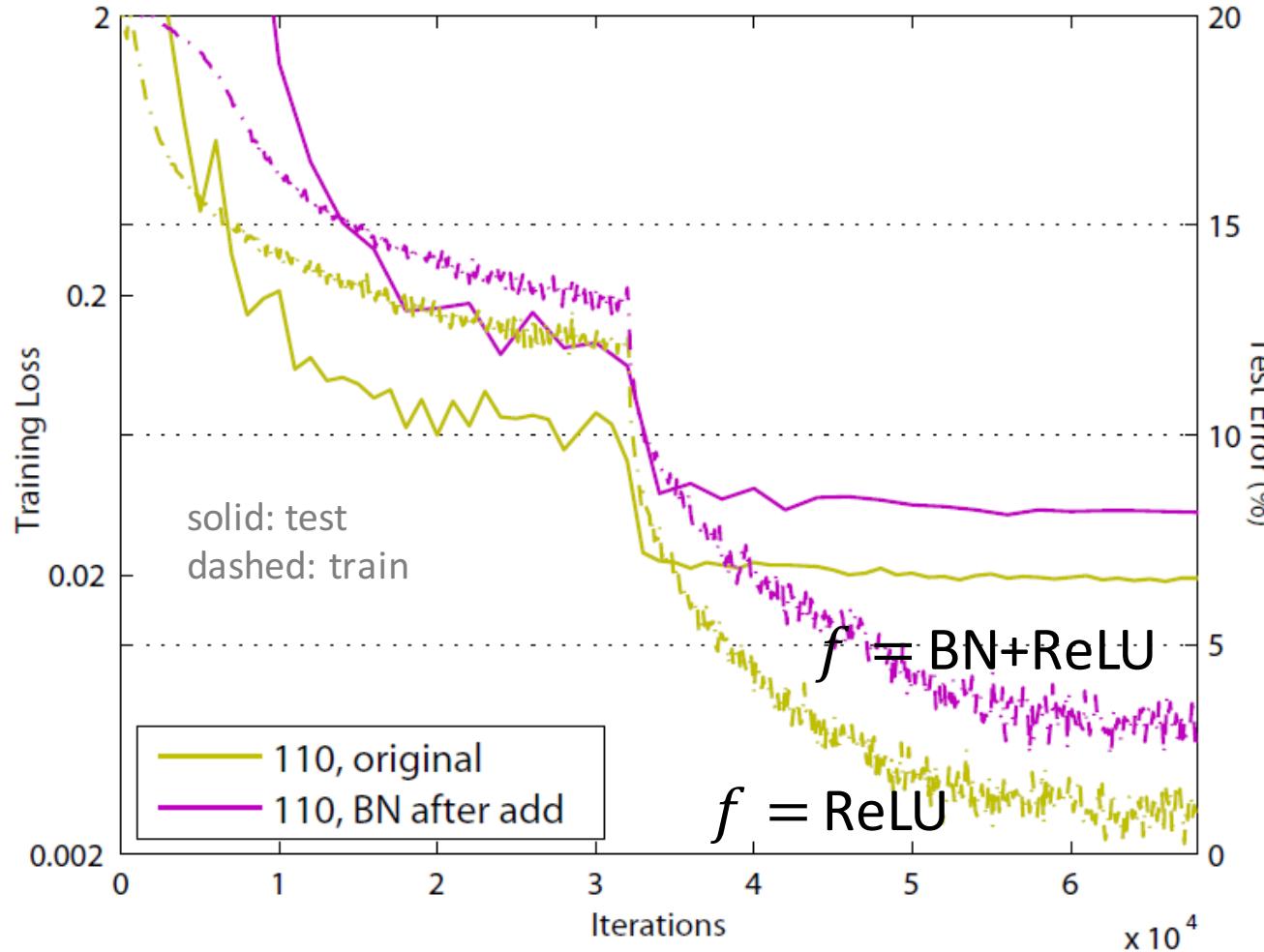
f is ReLU
(original ResNet)



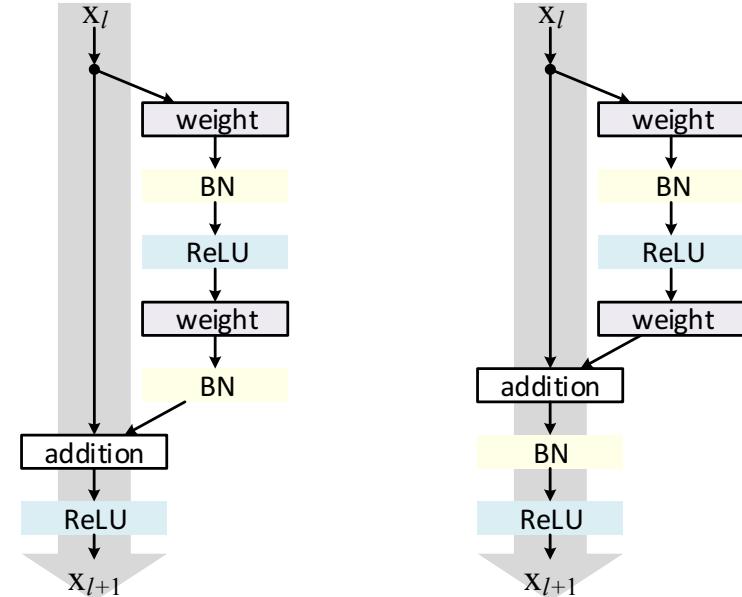
f is BN+ReLU



f is identity
(pre-activation ResNet)

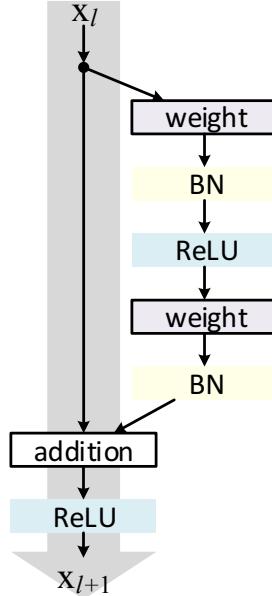
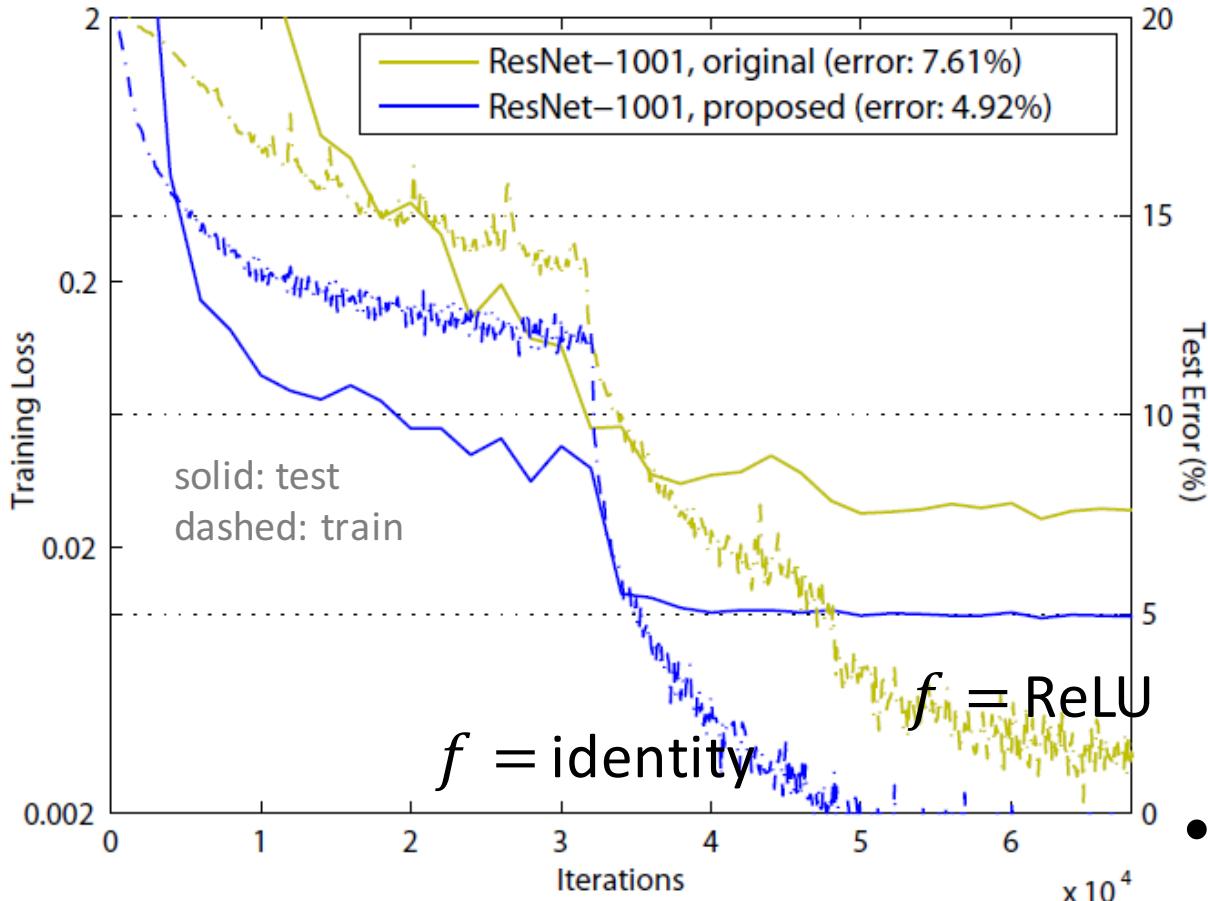


$$f = \text{ReLU} \quad f = \text{BN} + \text{ReLU}$$



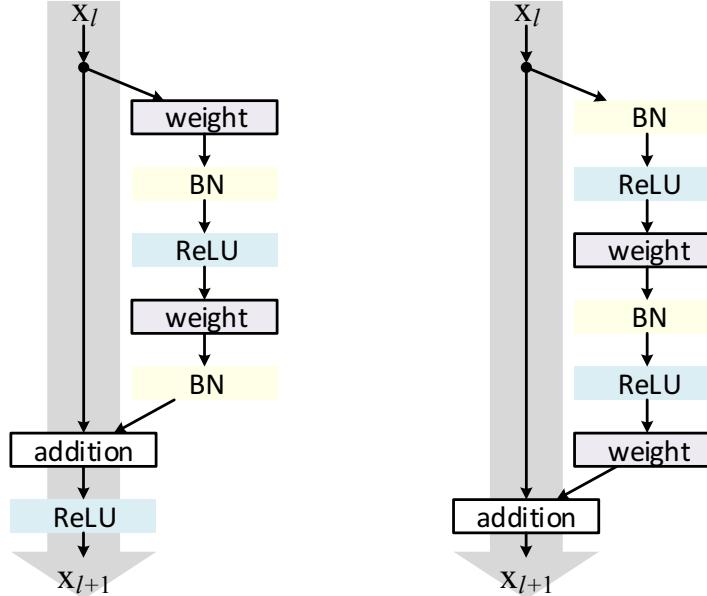
- BN could block prop
- Keep the shortest pass as smooth as possible

1001-layer ResNets on CIFAR-10



$$f = \text{ReLU}$$

$$f = \text{identity}$$



- ReLU could block prop when there are 1000 layers
- pre-activation design eases optimization (and improves generalization; see paper)

Comparisons on CIFAR-10/100

CIFAR-10

method	error (%)
NIN	8.81
DSN	8.22
FitNet	8.39
Highway	7.72
ResNet-110 (1.7M)	6.61
ResNet-1202 (19.4M)	7.93
ResNet-164, pre-activation (1.7M)	5.46
ResNet-1001 , pre-activation (10.2M)	4.92 (4.89 ± 0.14)

CIFAR-100

method	error (%)
NIN	35.68
DSN	34.57
FitNet	35.04
Highway	32.39
ResNet-164 (1.7M)	25.16
ResNet-1001 (10.2M)	27.82
ResNet-164, pre-activation (1.7M)	24.33
ResNet-1001 , pre-activation (10.2M)	22.71 (22.68 ± 0.22)

*all based on moderate augmentation

ImageNet Experiments

ImageNet single-crop (320x320) val error

method	data augmentation	top-1 error (%)	top-5 error (%)
ResNet-152, original	scale	21.3	5.5
ResNet-152, pre-activation	scale	21.1	5.5
ResNet-200, original	scale	21.8	6.0
ResNet-200, pre-activation	scale	20.7	5.3
ResNet-200, pre-activation	scale + aspect ratio	20.1*	4.8*

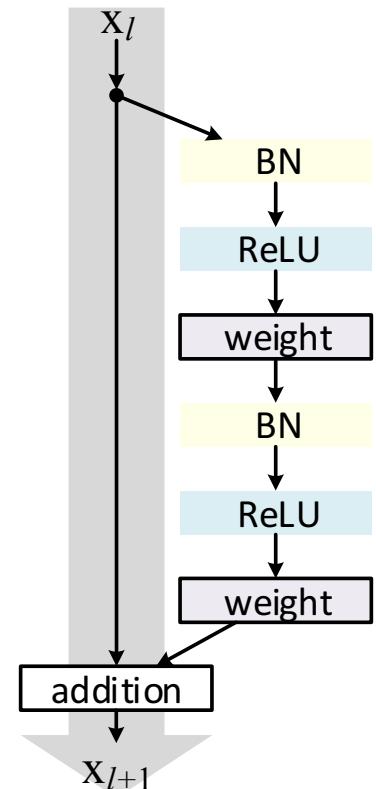
*independently reproduced by:

<https://github.com/facebook/fb.resnet.torch/tree/master/pretrained#notes>

training code and models available.

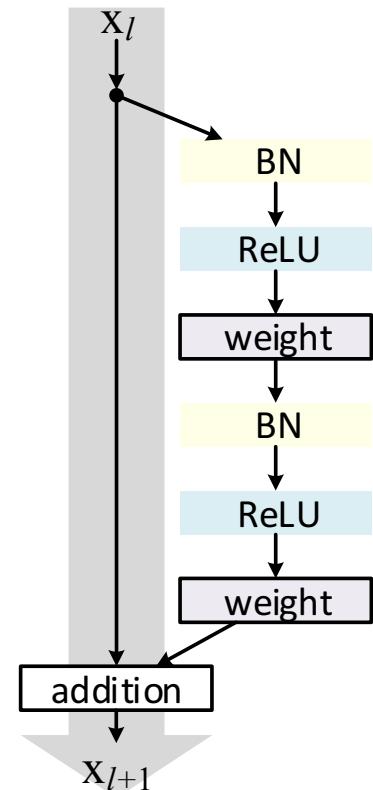
Summary of observations

- Keep the shortest path as smooth as possible
 - by making h and f identity
 - forward/backward signals directly flow through this path
- Features of any layers are additive outcomes
- **1000-layer** ResNets can be easily trained and have better accuracy



Future Works

- **Representation**
 - skipping 1 layer vs. multiple layers?
 - Flat vs. Bottleneck?
 - Inception-ResNet [Szegedy et al 2016]
 - ResNet in ResNet [Targ et al 2016]
 - Width vs. Depth [Zagoruyko & Komodakis 2016]
- **Generalization**
 - DropOut, MaxOut, DropConnect, ...
 - Drop Layer (Stochastic Depth) [Huang et al 2016]
- **Optimization**
 - Without residual/shortcut?



Applications

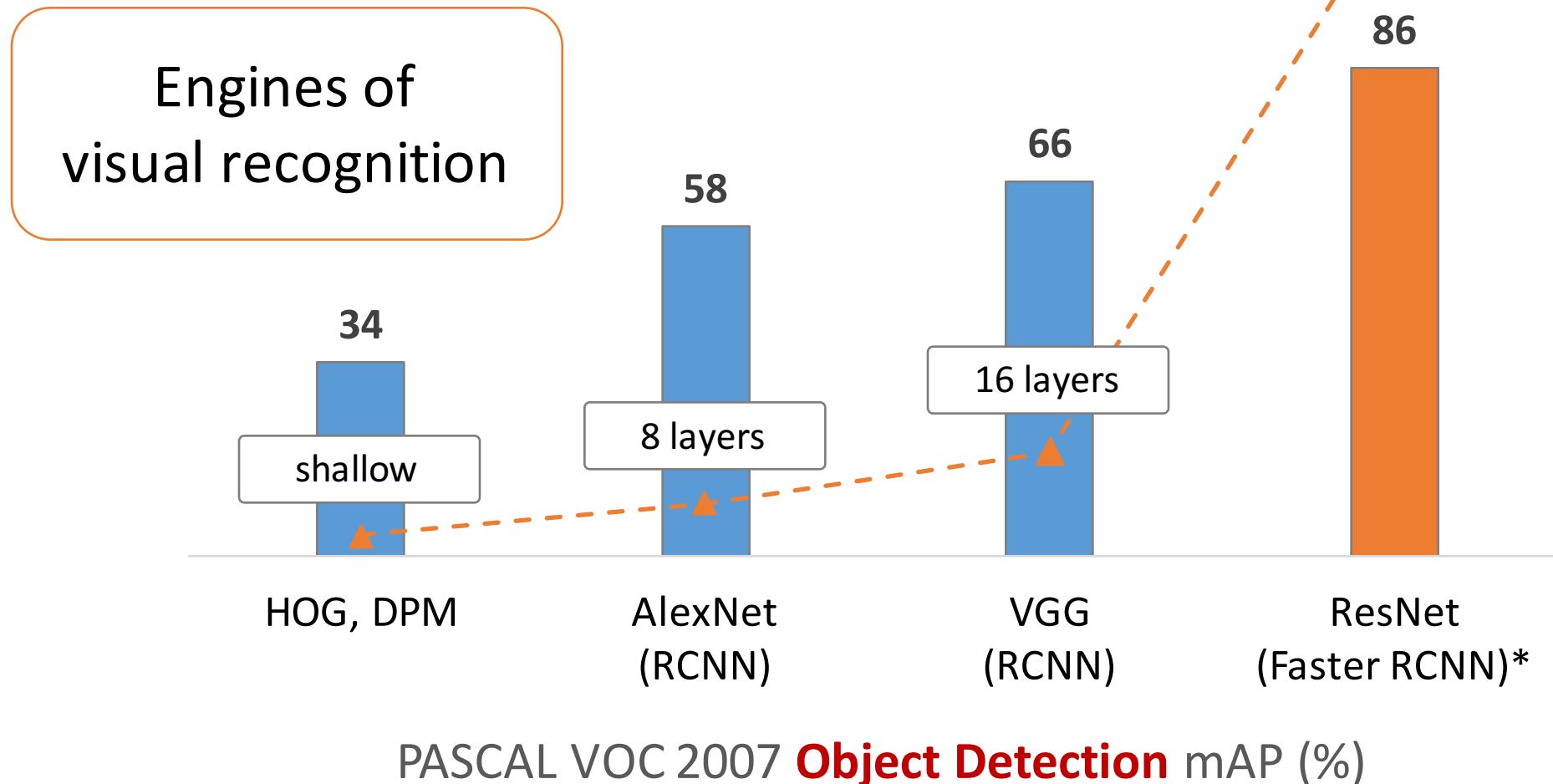
“Features matter”

“Features matter.” (quote [Girshick et al. 2014], the R-CNN paper)

task	2nd-place winner	ResNets	margin (relative)
ImageNet Localization <small>(top-5 error)</small>	12.0	9.0	27%
ImageNet Detection <small>(mAP@.5)</small>	53.6	62.1	16%
COCO Detection <small>(mAP@.5:.95)</small>	33.5	37.3	11%
COCO Segmentation <small>(mAP@.5:.95)</small>	25.1	28.2	12%

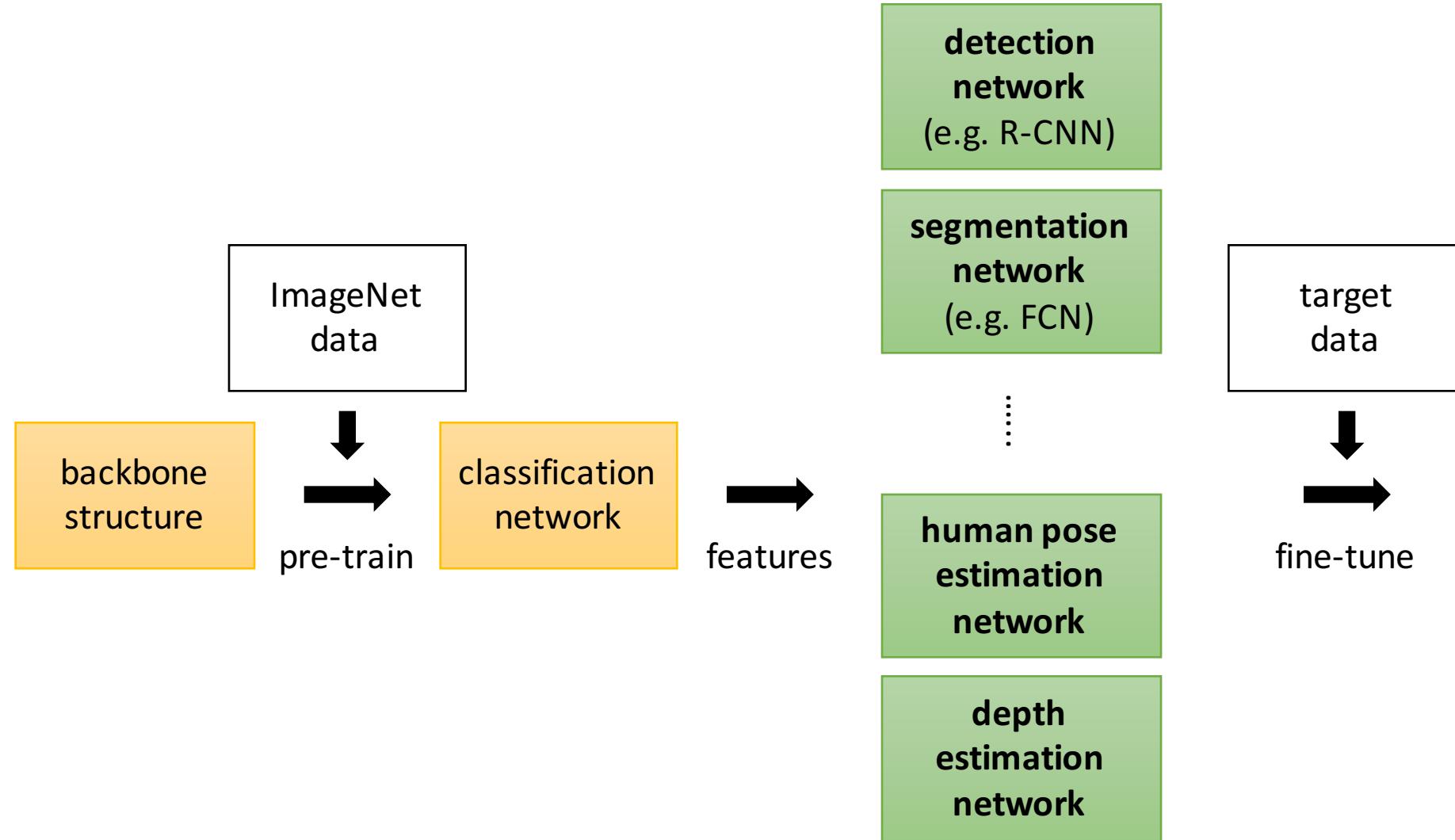
- Our results are all based on **ResNet-101**
- Deeper features are **well transferrable**

Revolution of Depth



*w/ other improvements & more data

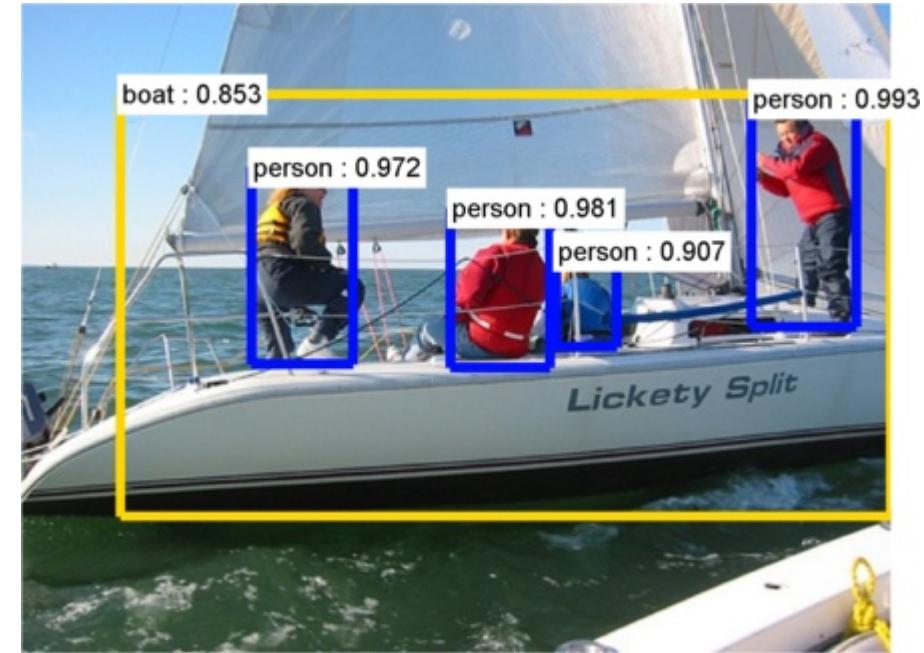
Deep Learning for Computer Vision



Example: Object Detection



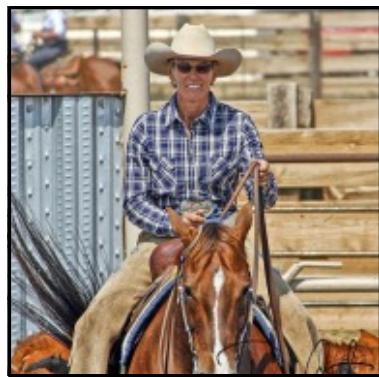
Image Classification
(what?)



Object Detection
(what + where?)

Object Detection: R-CNN

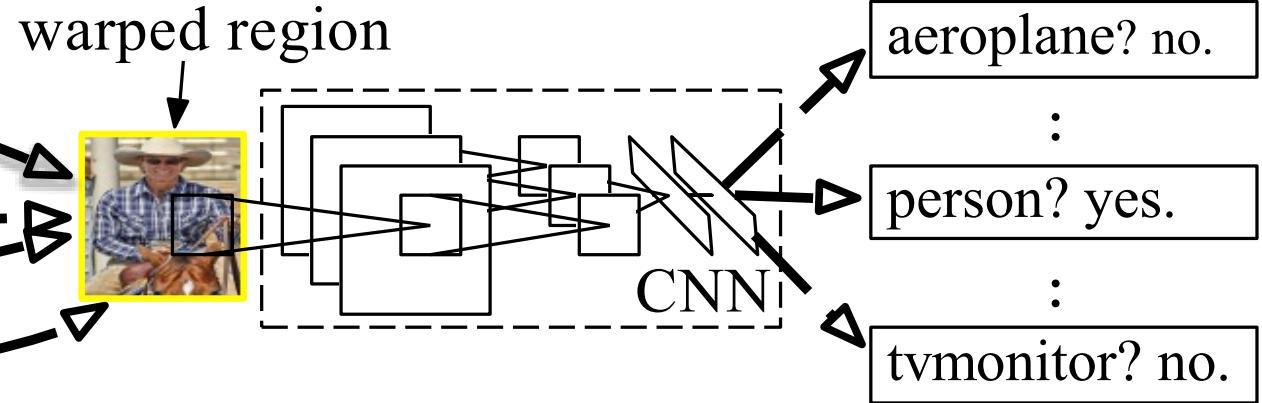
figure credit: R. Girshick et al.



input image



region proposals
~2,000



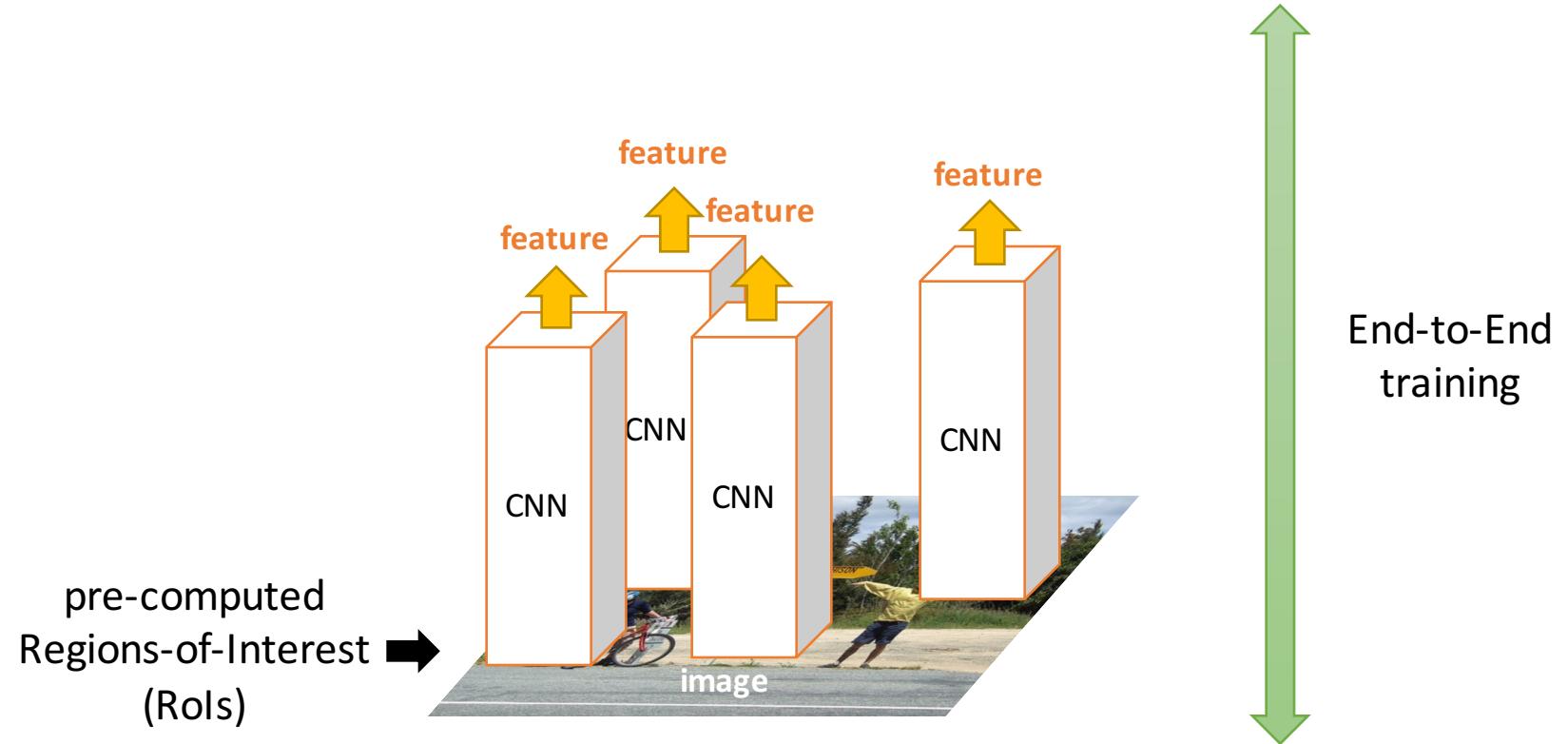
1 CNN for each region

classify regions

Region-based **CNN** pipeline

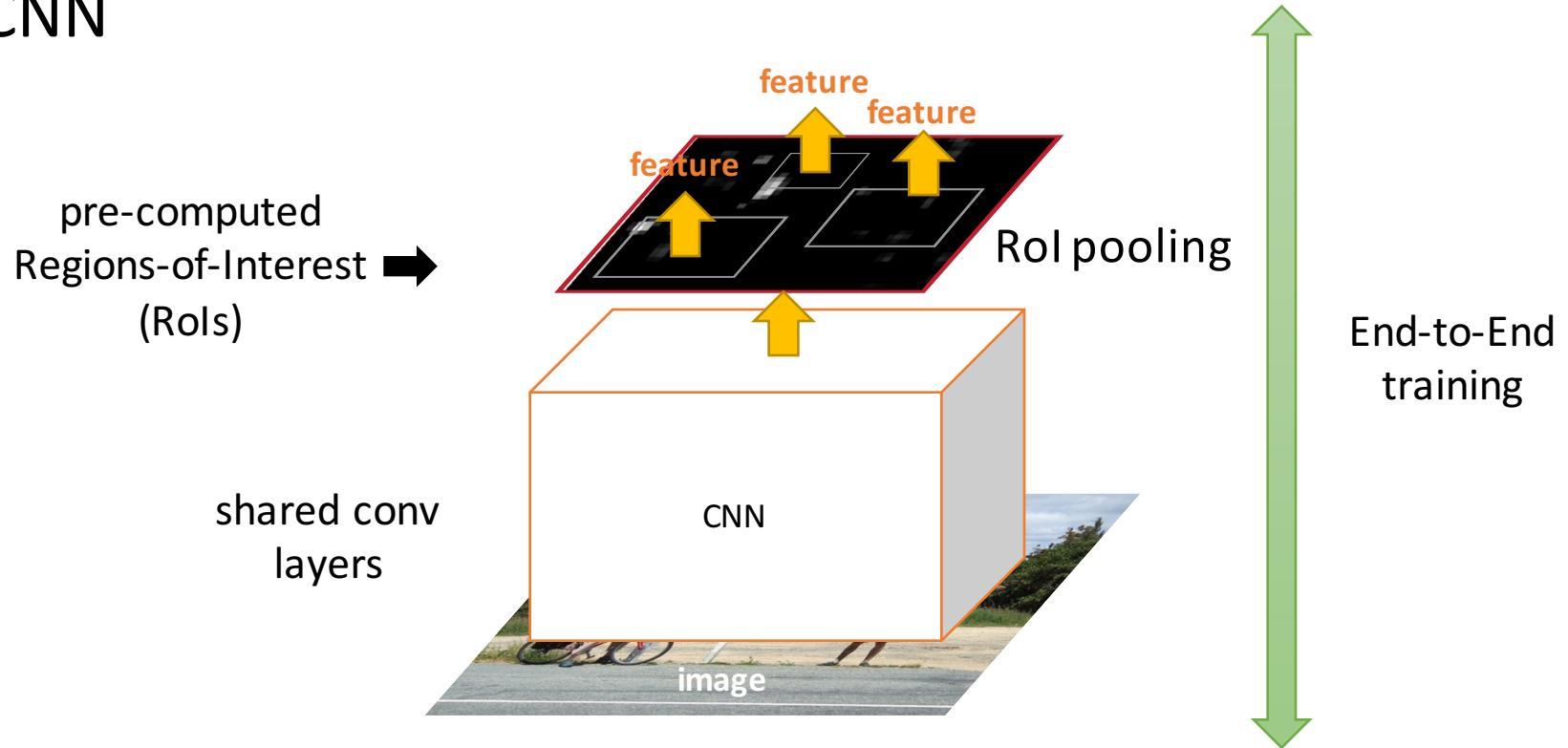
Object Detection: R-CNN

- R-CNN



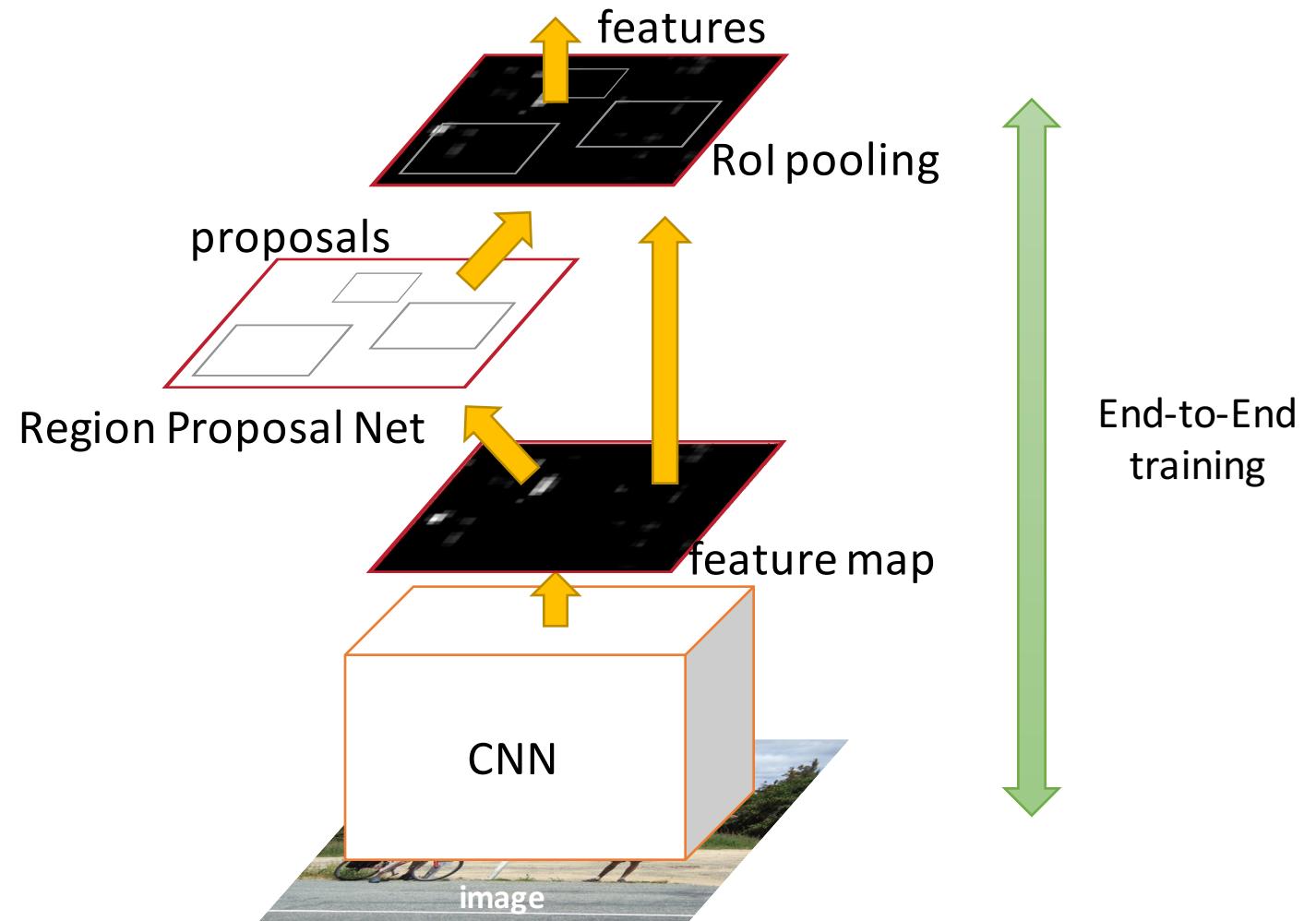
Object Detection: Fast R-CNN

- Fast R-CNN

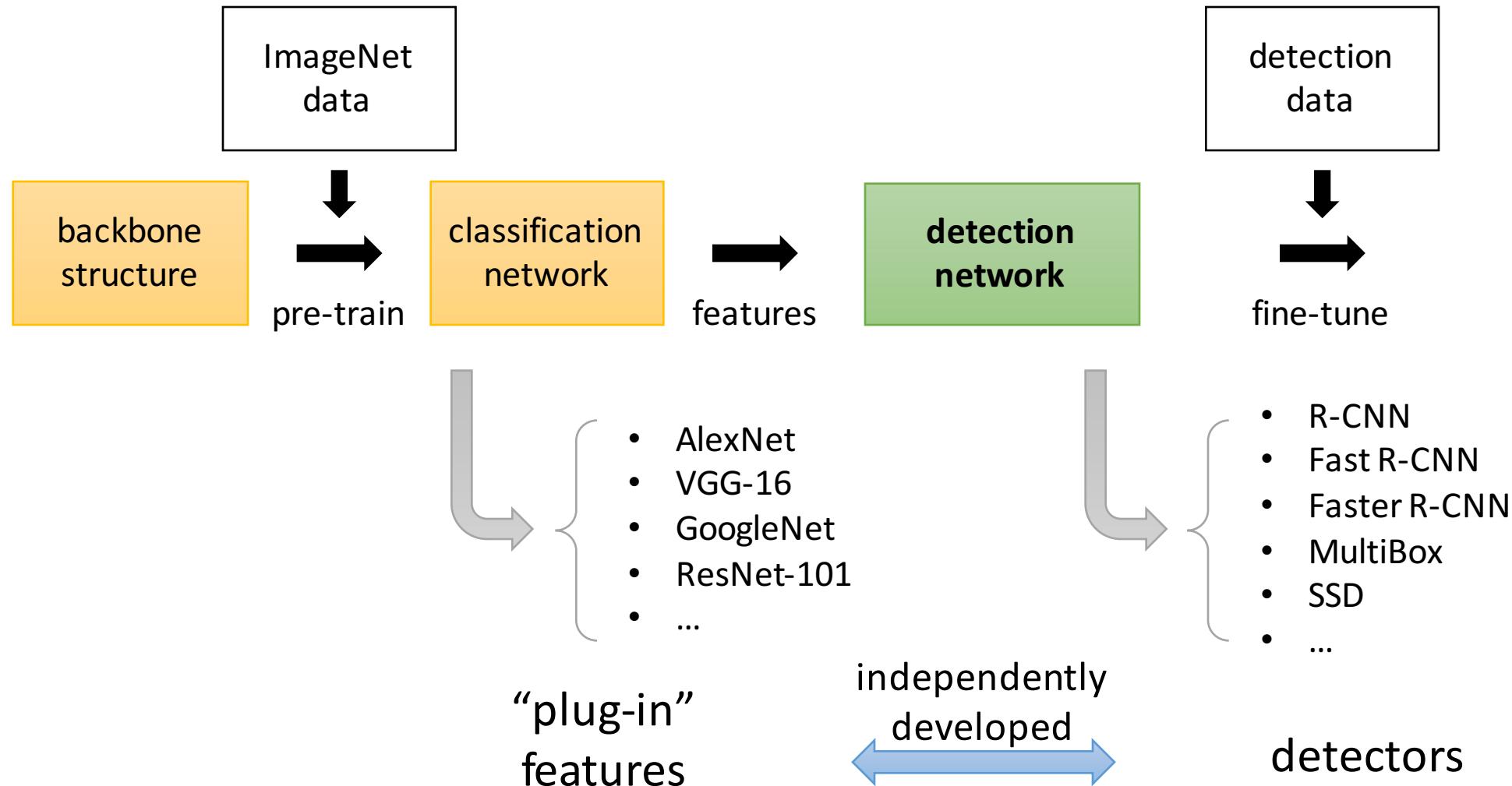


Object Detection: Faster R-CNN

- Faster R-CNN
 - Solely based on CNN
 - No external modules
 - Each step is end-to-end



Object Detection



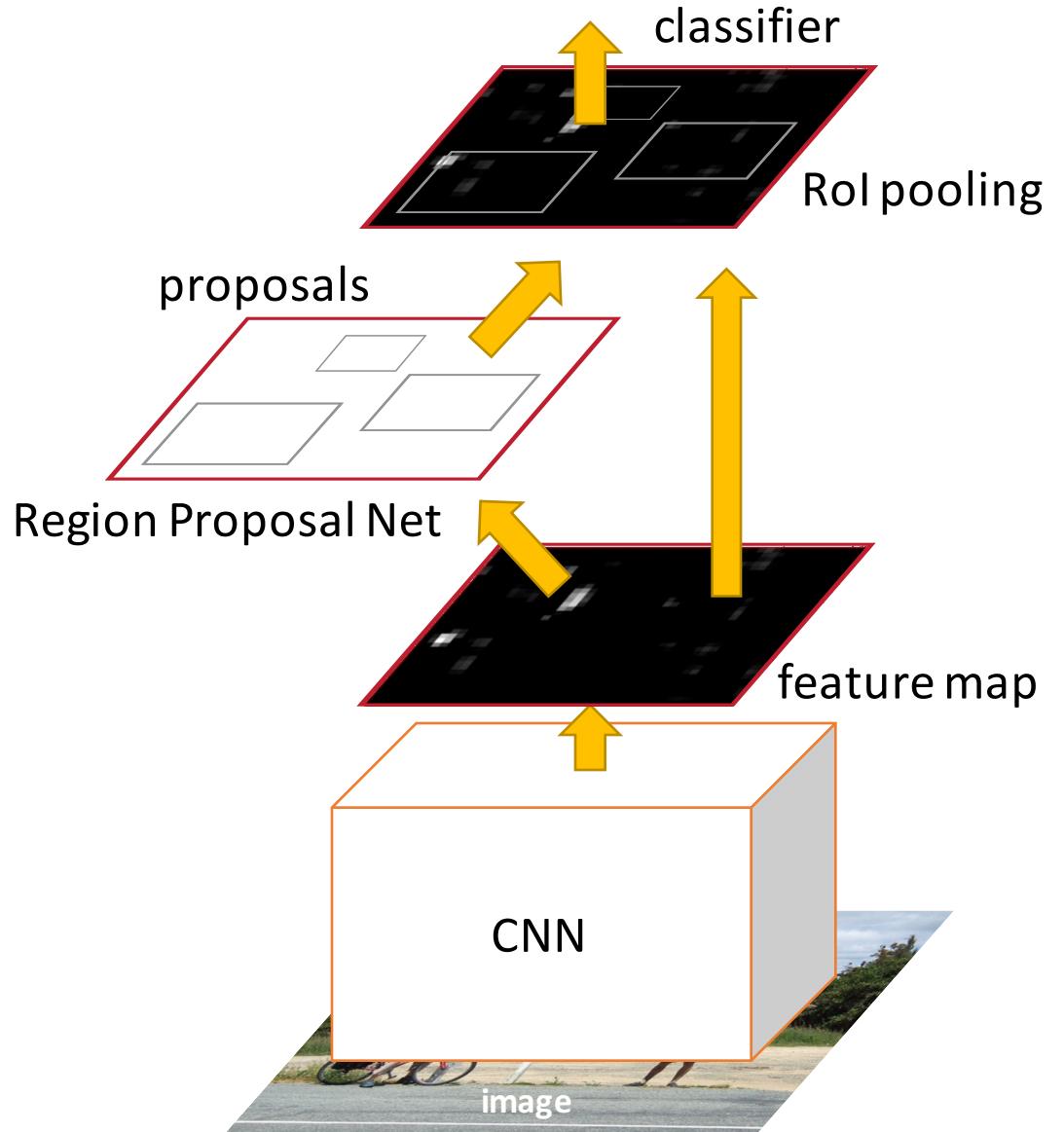
Object Detection

- Simply “Faster R-CNN + ResNet”

Faster R-CNN baseline	mAP@.5	mAP@.5:.95
VGG-16	41.5	21.5
ResNet-101	48.4	27.2

coco detection results

**ResNet-101 has 28% relative gain
vs VGG-16**

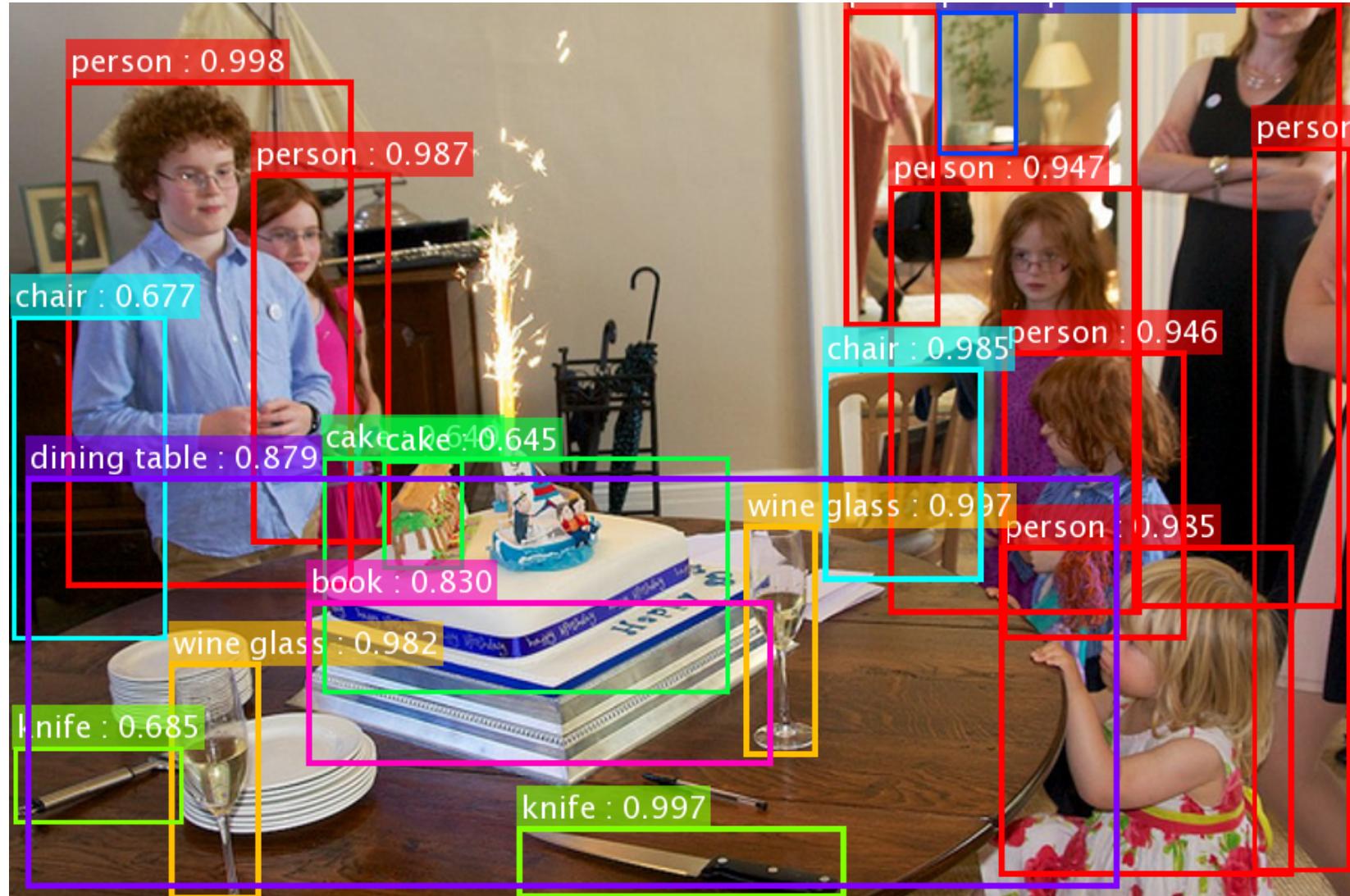


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. NIPS 2015.

Object Detection

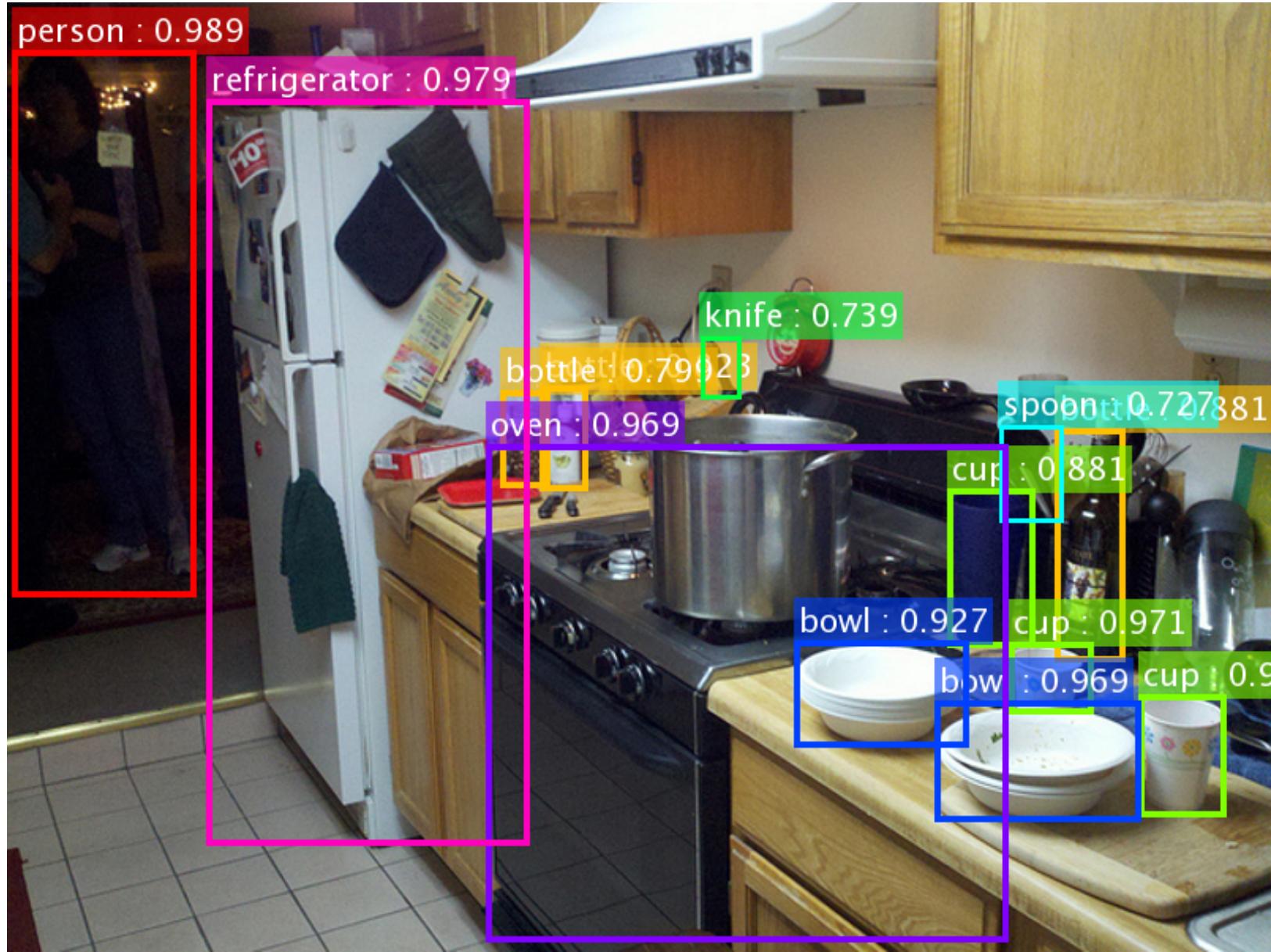
- RPN **learns** proposals by extremely deep nets
 - We use **only 300 proposals** (no hand-designed proposals)
- Add components:
 - Iterative localization
 - Context modeling
 - Multi-scale testing
- All are based on CNN features; all are end-to-end
- All benefit **more** from **deeper** features – cumulative gains!



ResNet's object detection result on COCO

*the original image is from the COCO dataset

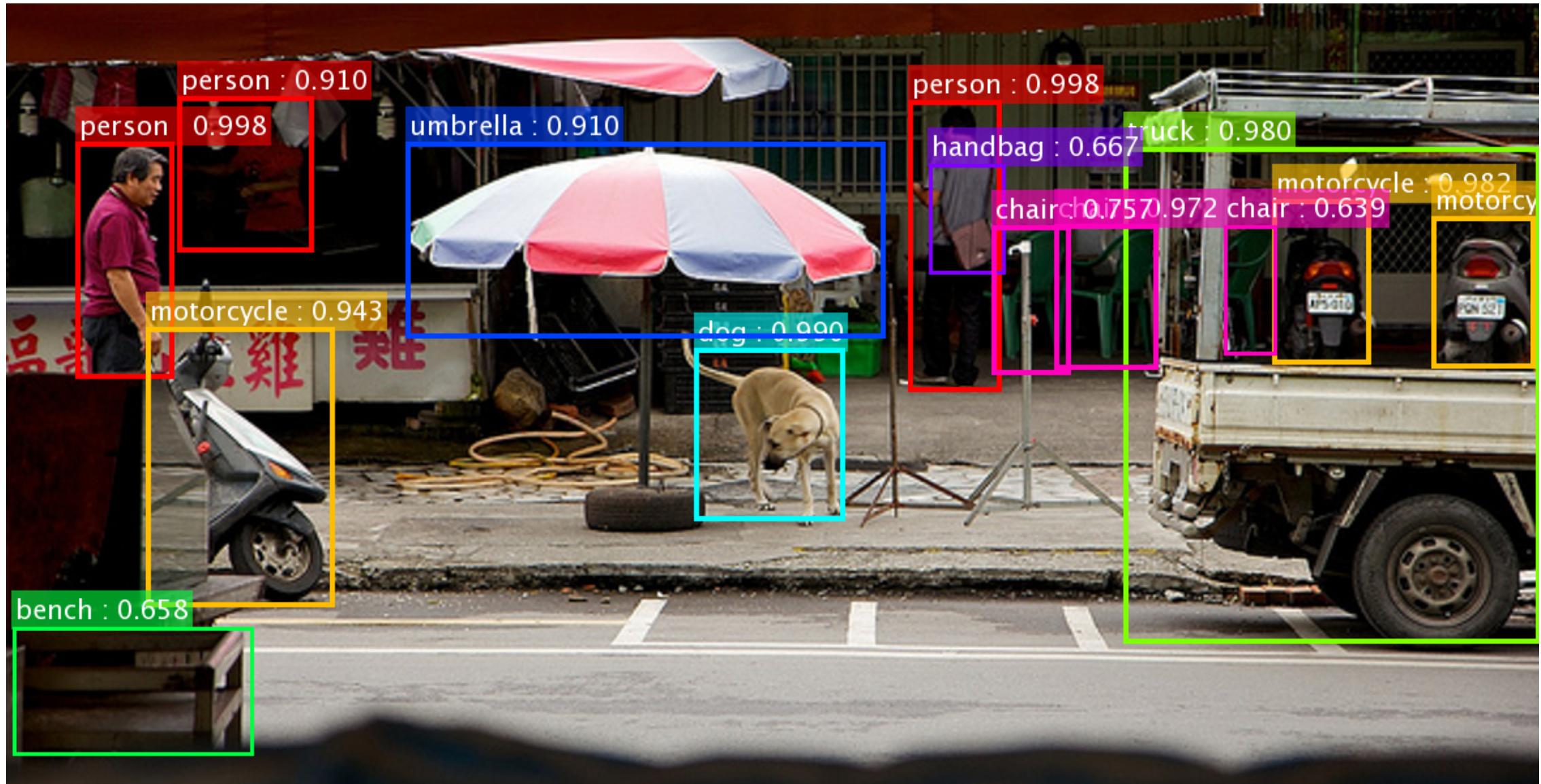
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.



*the original image is from the COCO dataset

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.



*the original image is from the COCO dataset

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.



this video is available online: <https://youtu.be/WZmSMkK9VuA>

Results on real video. Models trained on MS COCO (80 categories).
(frame-by-frame; no temporal processing)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.

More Visual Recognition Tasks

ResNet-based methods lead on these benchmarks (incomplete list):

- ImageNet classification, detection, localization
- MS COCO detection, segmentation
- PASCAL VOC detection, segmentation
- Human pose estimation [Newell et al 2016]
- Depth estimation [Laina et al 2016]
- Segment proposal [Pinheiro et al 2016]
- ...

	mean	aero	bicycle	bird	boat	bottle	bus	car	cat	chair	couch	motorcycle	sofa	train	tv	
DeepLabv2-CRF [?]	79.7	92.6	60.4	91.6	63.4	76.3	95.0	88.4	92.1	89.1	93.5	85.1	95.1	98.1	97.1	
CASIA_SegResNet_CRF_COCO [?]	79.3	93.8	42.1	81.1	65.6	55.1	95.1	95.1	95.1	95.1	95.1	95.1	95.1	95.1	95.1	95.1
Adelaide_VeryDeep_FCN_VOC [?]	79.1	91.9	48.1	93.4	69.3	75.5	94.2	87.5	97.1	89.1	95.1	85.1	95.1	98.1	97.1	97.1
LRR_4x_COCO [?]	78.7	93.2	44.2	89.4	65.4	74.9	95.5	87.0	94.1	89.1	95.1	85.1	95.1	98.1	97.1	97.1
CASIA_IVA_OASeg [?]	78.3	93.8	41.9	89.4	67.5	71.5	94.6	85.3	89.1	89.1	95.1	85.1	95.1	98.1	97.1	97.1
Oxford_TVGV_HO_CRF [?]	77.9	92.5	59.1	90.3	70.6	74.4	92.4	84.1	88.1	88.1	95.1	85.1	95.1	98.1	97.1	97.1
Adelaide_Context_CNN_CRF_COCO [?]	77.8	92.9	39.6	84.0	67.9	75.3	92.7	83.8	90.1	89.1	95.1	85.1	95.1	98.1	97.1	97.1

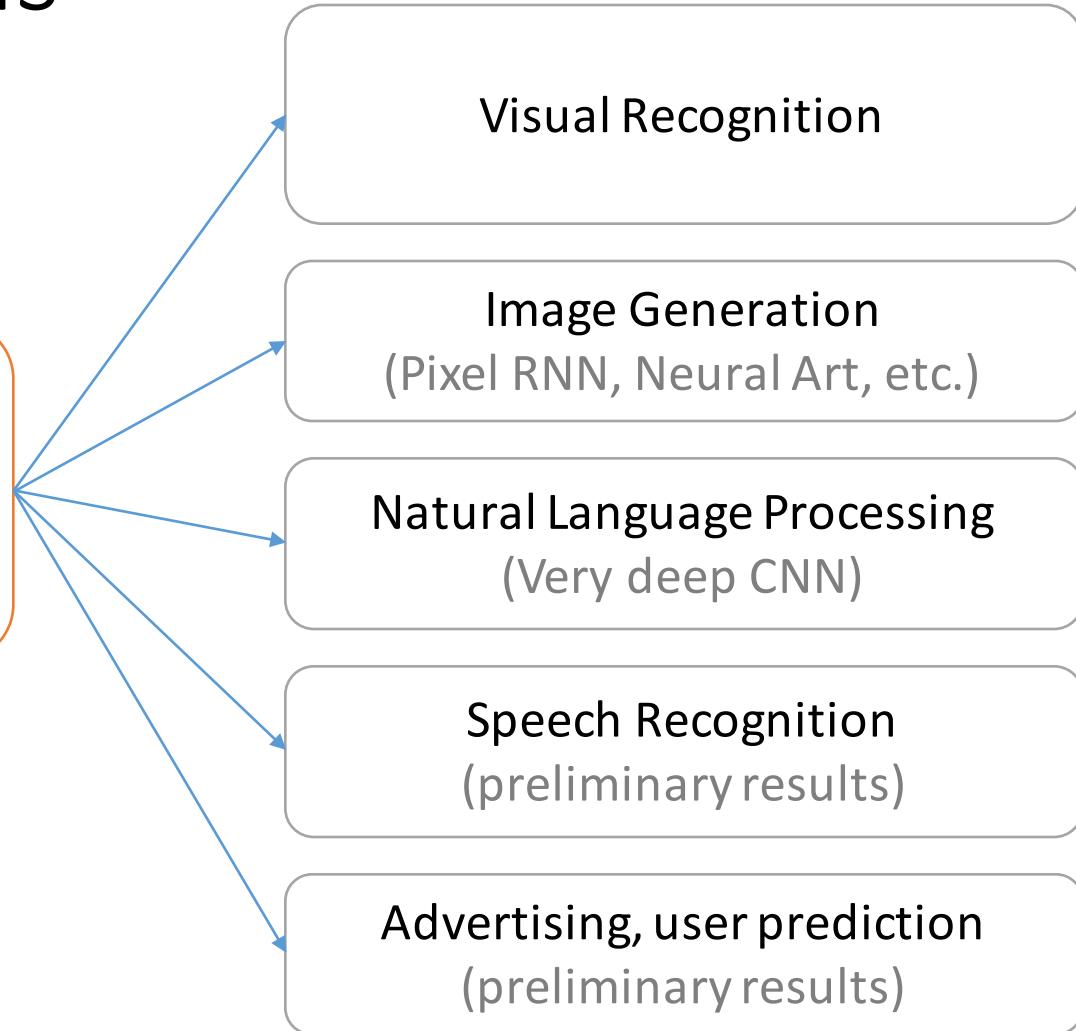
PASCAL segmentation leaderboard

	mean	aero	bicycle	bird	boat	bottle	bus	car	cat	chair	couch	motorcycle	sofa	train	tv	
Faster RCNN, ResNet (VOC+COCO) [?]	83.8	92.1	88.4	84.9	75.9	71.4	86.3	87.8	94.2	89.1	93.5	85.1	95.1	98.1	97.1	97.1
R-FCN, ResNet (VOC+COCO) [?]	82.0	89.5	88.3	83.1	75.9	71.7	86.1	86.3	94.1	89.1	93.5	85.1	95.1	98.1	97.1	97.1
OHEN+FRCN, VGG16, VOC+COCO [?]	80.1	90.1	87.7	79.5	65.6	60.5	80.1	80.1	93.0	89.1	93.5	85.1	95.1	98.1	97.1	97.1
SSD500 VGG16 VOC + COCO [?]	78.7	89.1	85.7	78.9	63.3	57.0	85.3	84.1	92.3	89.1	93.5	85.1	95.1	98.1	97.1	97.1
HFM_VGG16 [?]	77.5	88.8	85.1	76.8	64.8	61.4	85.0	84.1	90.0	89.1	93.5	85.1	95.1	98.1	97.1	97.1
IFRN_07+12 [?]	76.6	87.8	83.9	79.0	64.5	58.9	82.2	82.0	91.4	89.1	93.5	85.1	95.1	98.1	97.1	97.1
ION [?]	76.4	87.5	84.7	76.8	63.8	58.3	82.6	79.0	90.9	89.1	93.5	85.1	95.1	98.1	97.1	97.1

PASCAL detection leaderboard

Potential Applications

ResNets have shown outstanding or promising results on:



Conclusions of the Tutorial

- Deep Residual Learning:
 - Ultra deep networks can be easy to train
 - Ultra deep networks can gain accuracy from depth
 - Ultra deep representations are well transferrable
 - Now 200 layers on ImageNet and 1000 layers on CIFAR!

Resources

- Models and Code
 - Our ImageNet models in Caffe: <https://github.com/KaimingHe/deep-residual-networks>
- Many available implementation
 - (list in <https://github.com/KaimingHe/deep-residual-networks>)
 - Facebook AI Research's Torch ResNet:
<https://github.com/facebook/fb.resnet.torch>
 - Torch, CIFAR-10, with ResNet-20 to ResNet-110, training code, and curves: code
 - Lasagne, CIFAR-10, with ResNet-32 and ResNet-56 and training code: code
 - Neon, CIFAR-10, with pre-trained ResNet-32 to ResNet-110 models, training code, and curves: code
 - Torch, MNIST, 100 layers: blog, code
 - A winning entry in Kaggle's right whale recognition challenge: blog, code
 - Neon, Place2 (mini), 40 layers: blog, code
 -

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Identity Mappings in Deep Residual Networks". arXiv 2016.