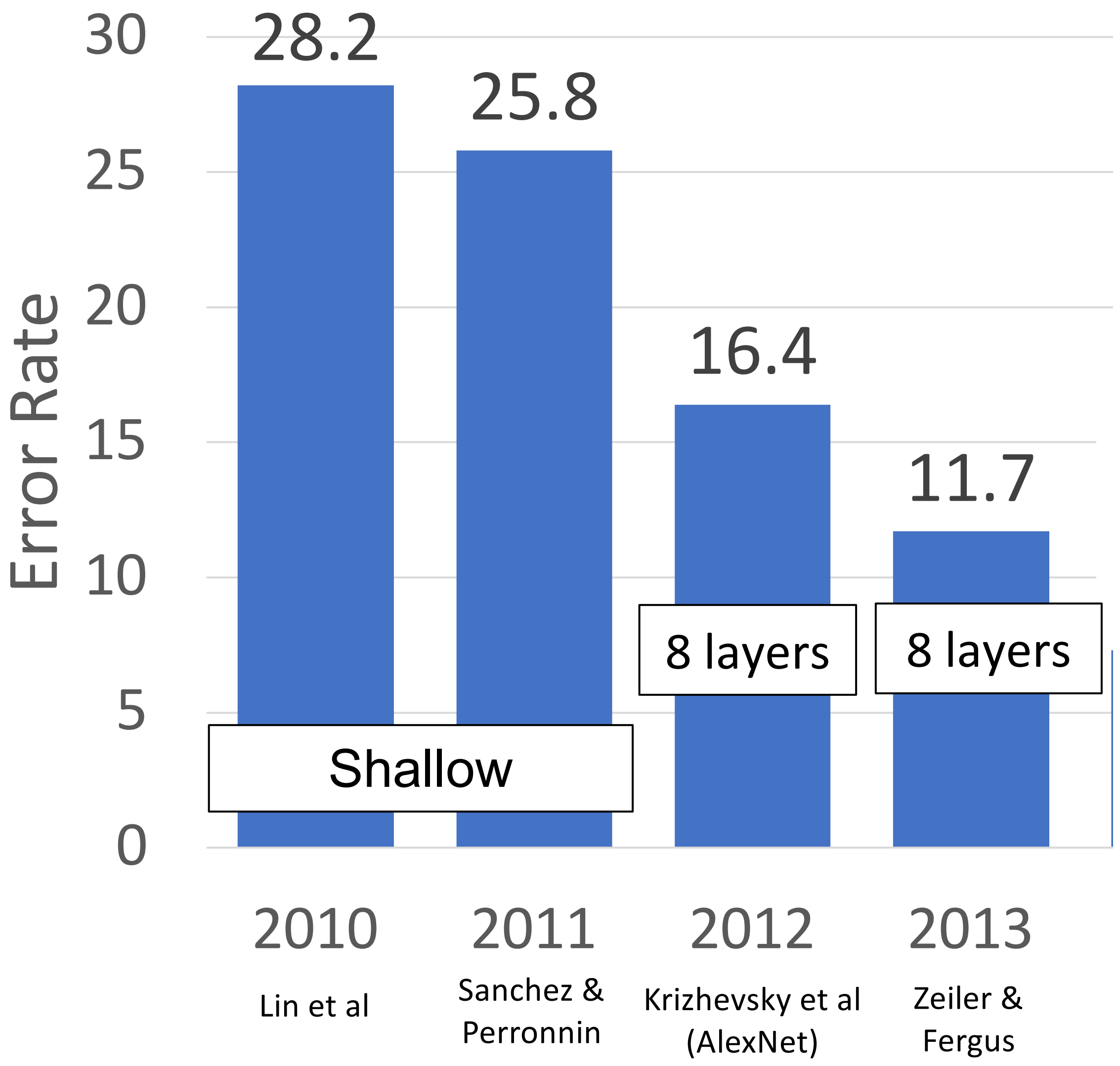
Vision Architectures Since AlexNet

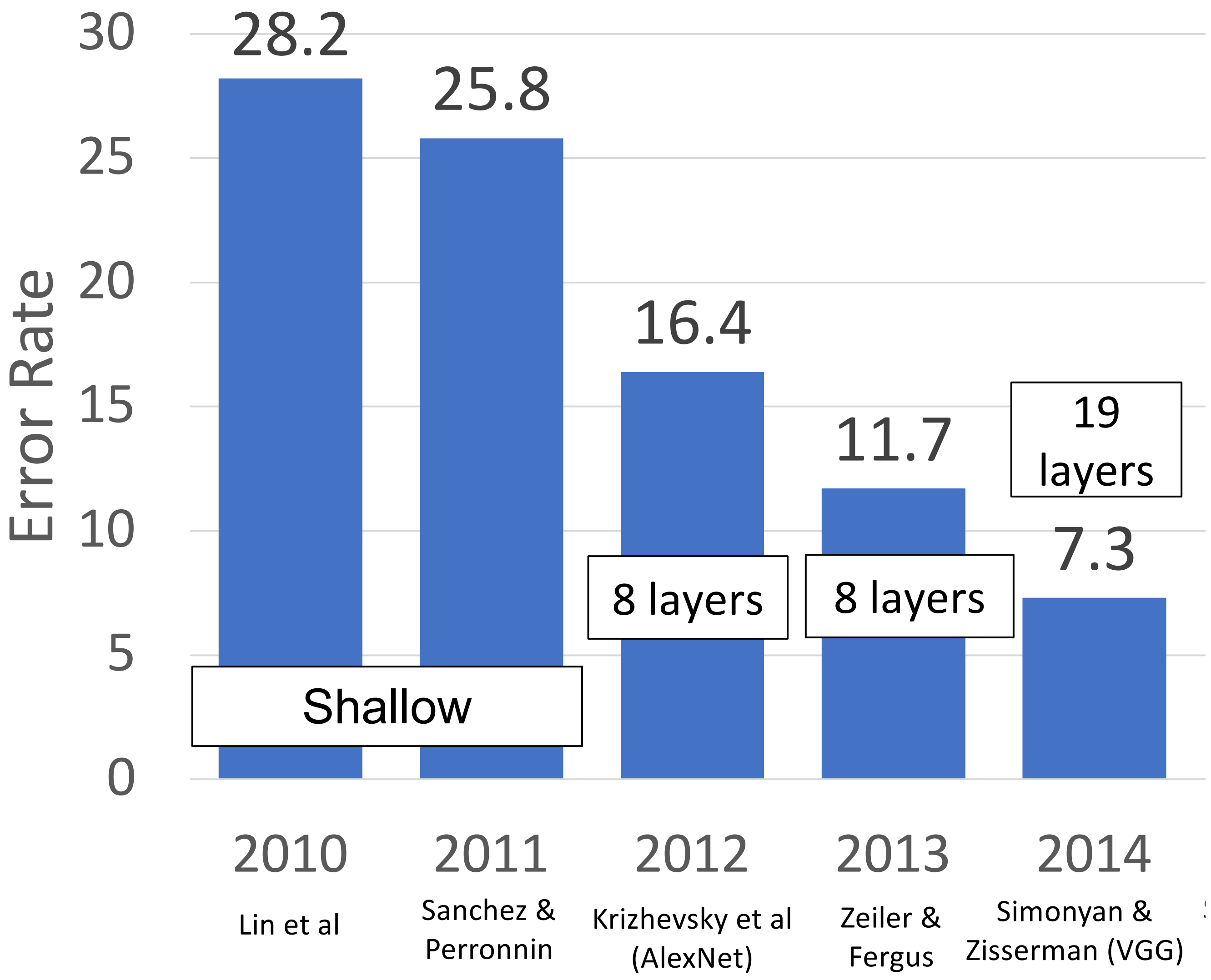
Slides from Justin Johnson

ImageNet Classification Challenge



Slide from Justin Johnson

ImageNet Classification Challenge



Slide from Justin Johnson

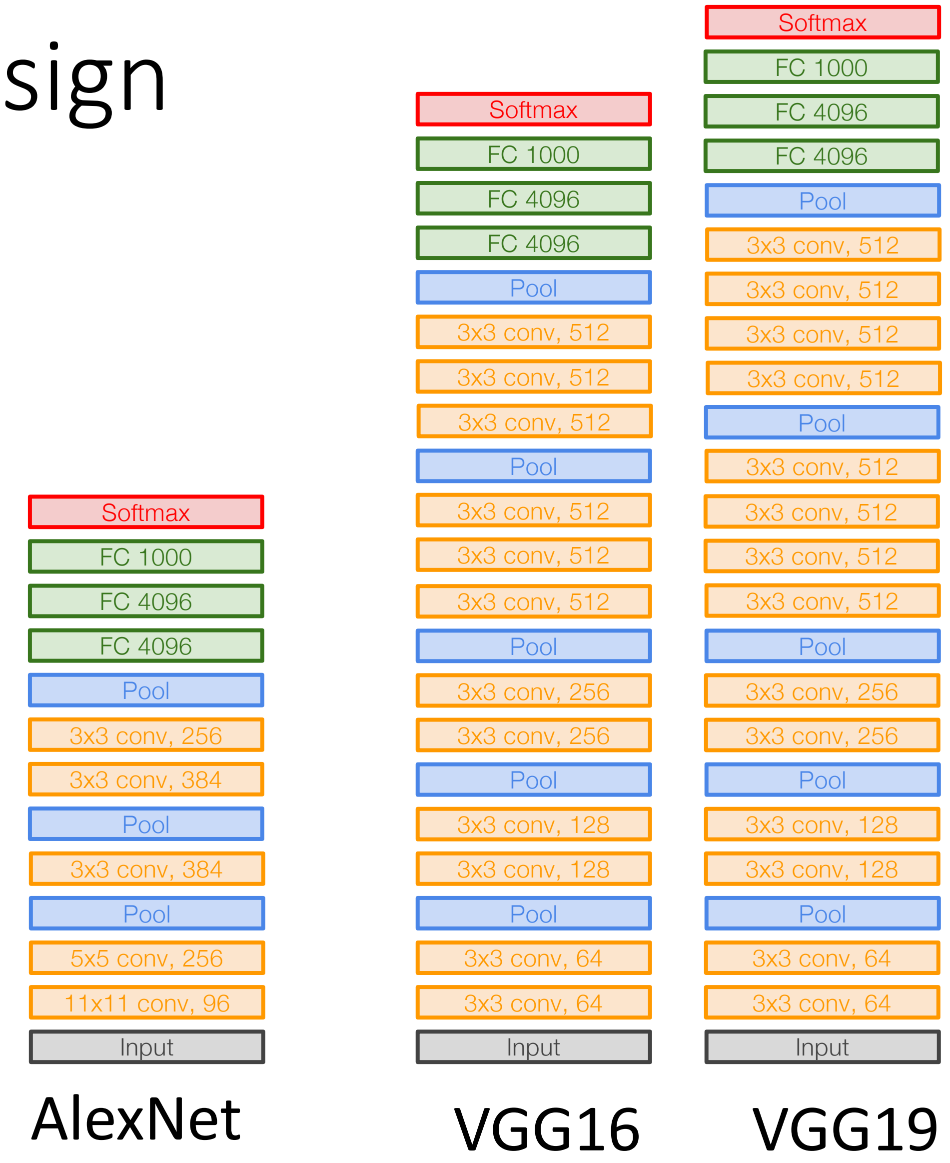
VGG: Deeper Networks, Regular Design

<u>VGG Design rules:</u>

All conv are 3x3 stride 1 pad 1

All max pool are 2x2 stride 2

After pool, double #channels



Simonyan and Zissermann, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, ICLR 2015

Slide from Justin Johnson

VGG: Deeper Networks, Regular Design

<u>VGG Design rules:</u>

All conv are 3x3 stride 1 pad 1

All max pool are 2x2 stride 2

After pool, double #channels

Network has 5 convolutional **stages**:

Stage 1: conv-conv-pool

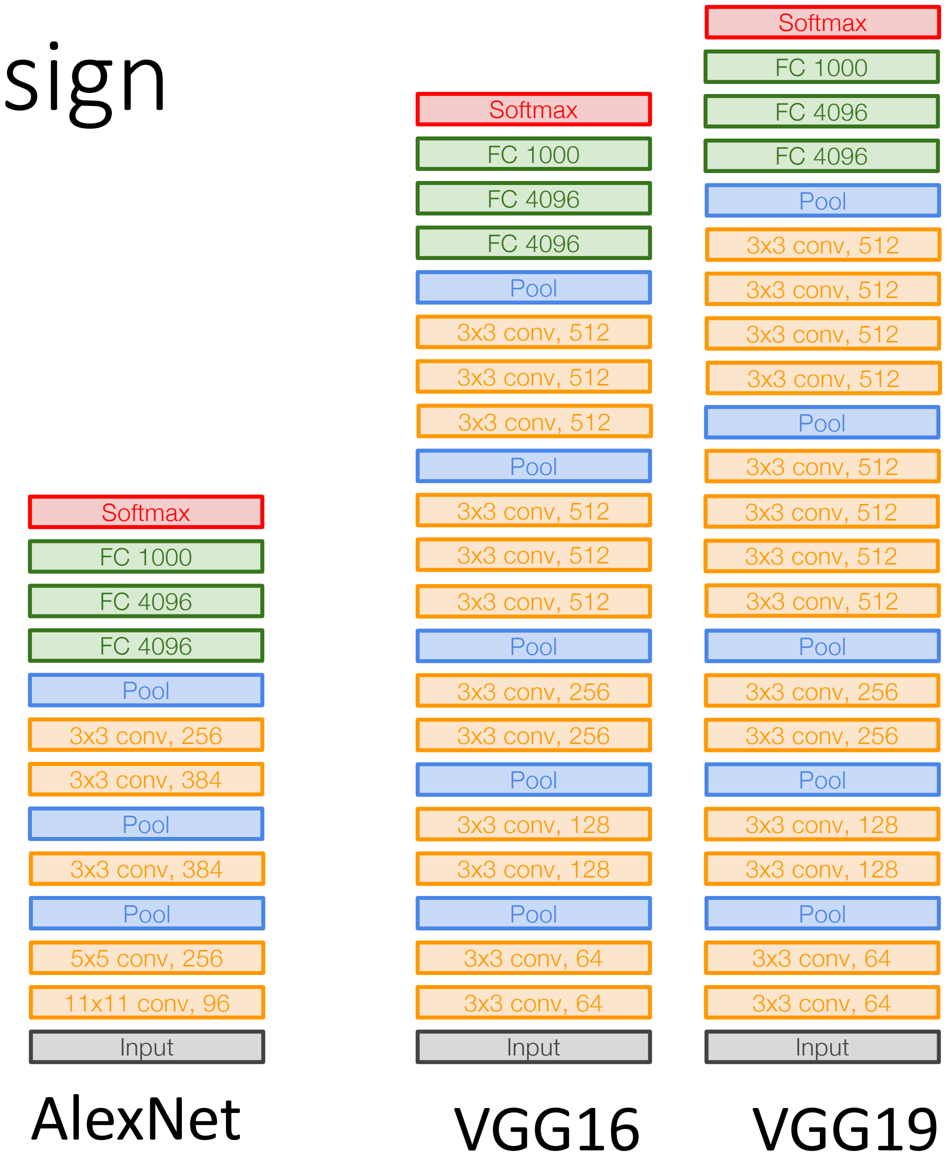
Stage 2: conv-conv-pool

Stage 3: conv-conv-pool

Stage 4: conv-conv-conv-[conv]-pool

Stage 5: conv-conv-conv-[conv]-pool

(VGG-19 has 4 conv in stages 4 and 5)



Simonyan and Zissermann, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, ICLR 2015

Slide from Justin Johnson

VGG: Deeper Networks, Regular Design

VGG Design rules:

All conv are 3x3 stride 1 pad 1

All max pool are 2x2 stride 2

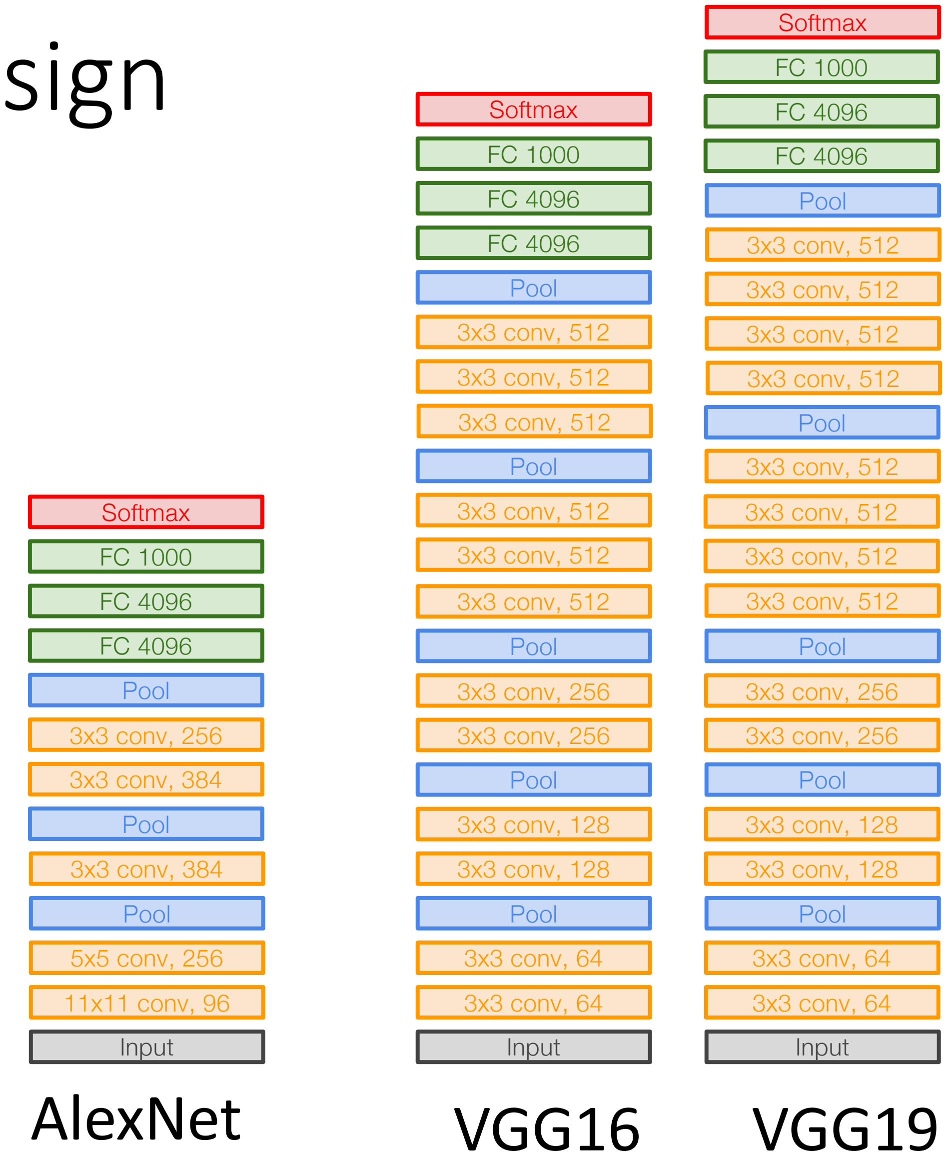
After pool, double #channels

Option 1:

Conv(5x5, C -> C)

Params: $25C^2$

FLOPs: $25C^2HW$



Simonyan and Zissermann, "Very Deep Convolutional Networks for Large-Scale Image Recognition", ICLR 2015

Slide from Justin Johnson

VGG: Deeper Networks, Regular Design

<u>VGG Design rules:</u>

All conv are 3x3 stride 1 pad 1

All max pool are 2x2 stride 2

After pool, double #channels

<u>Option 1:</u>

Conv(5x5, C -> C)

Params: $25C^2$

FLOPs: $25C^2HW$

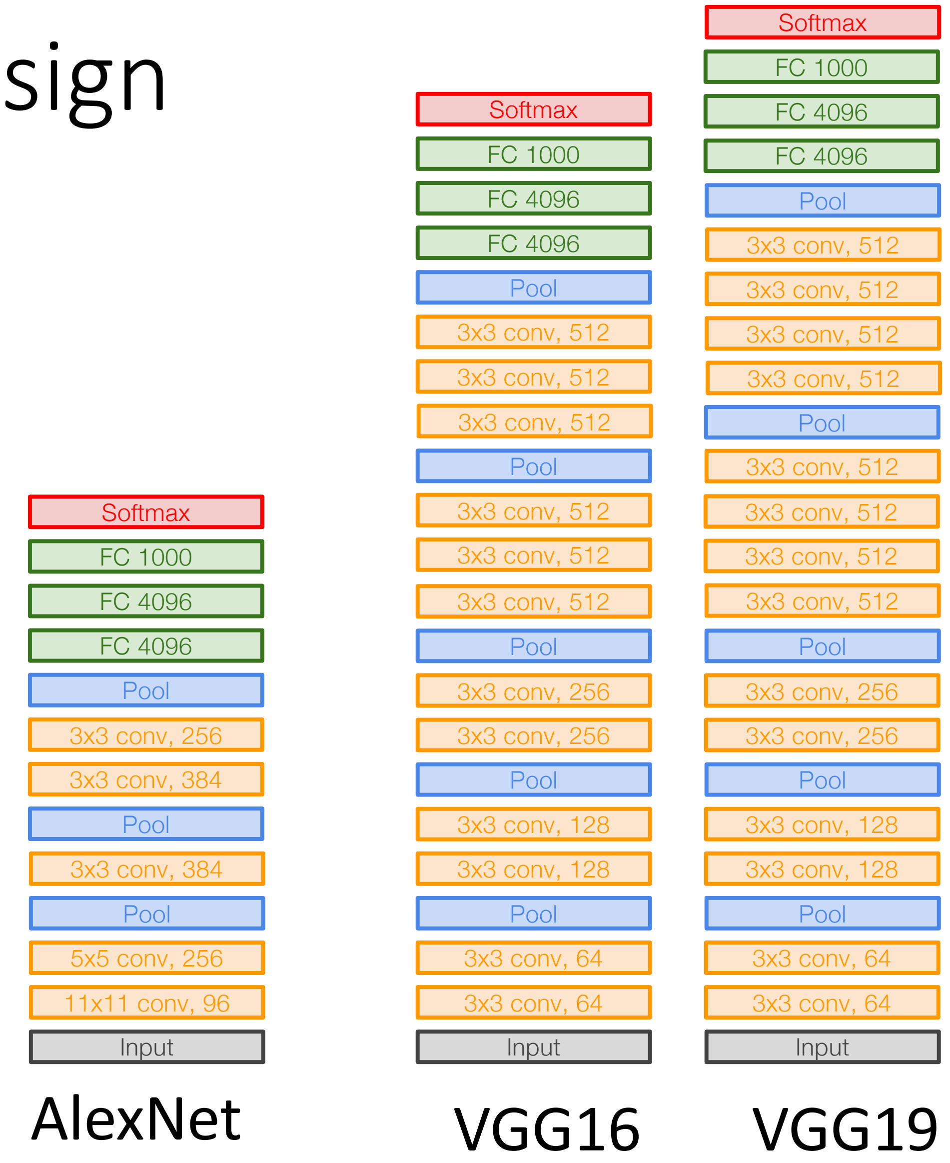
<u>Option 2:</u>

Conv(3x3, C -> C)

Conv(3x3, C -> C)

Params: $18C^2$

FLOPs: $18C^2HW$



Simonyan and Zissermann, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, ICLR 2015

Slide from Justin Johnson

VGG: Deeper Networks, Regular Design

<u>VGG Design rules:</u>

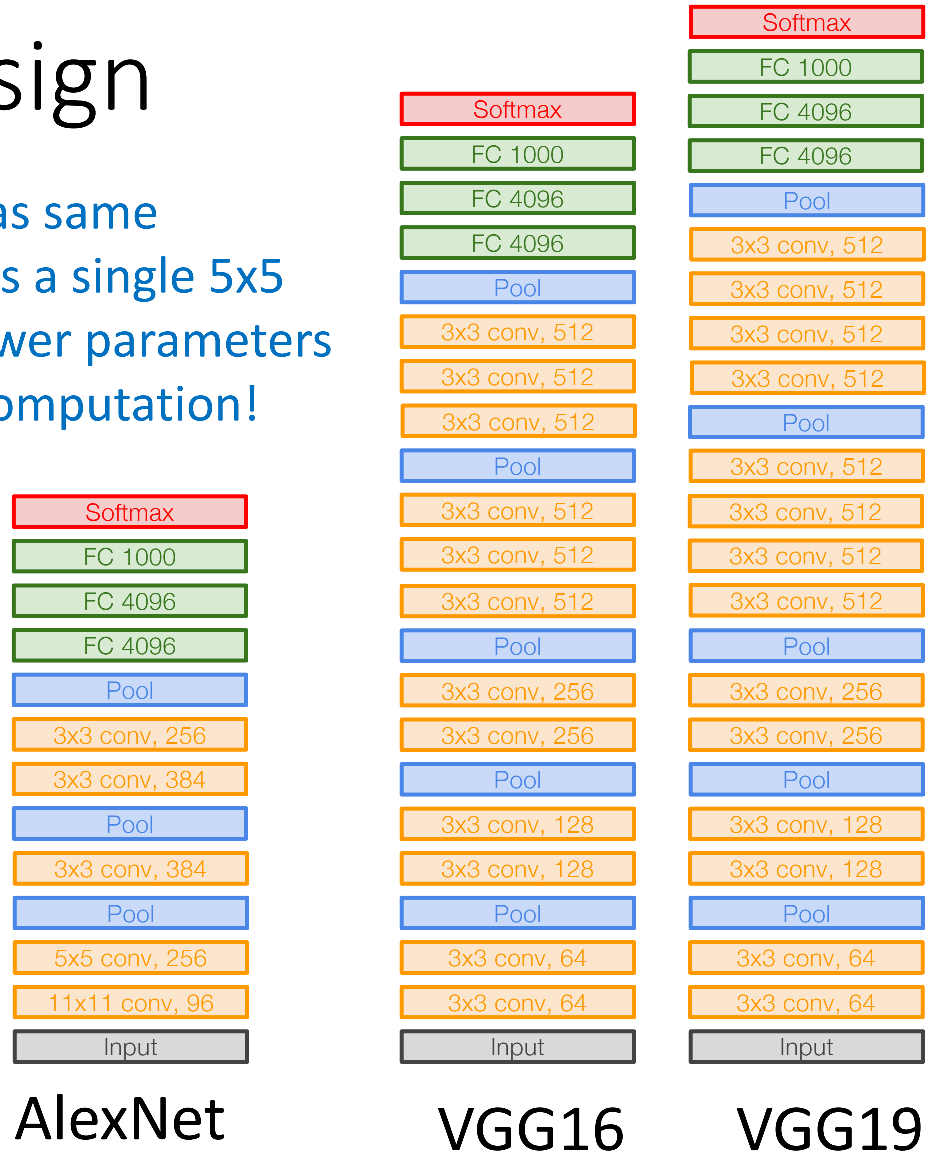
All conv are 3x3 stride 1 pad 1

All max pool are 2x2 stride 2

After pool, double #channels

Two 3x3 conv has same receptive field as a single 5x5 conv, but has fewer parameters and takes less computation!

<u>Option 1:</u>	<u>Option 2:</u>
Conv(5x5, C -> C)	Conv(3x3, C -> C)
	Conv(3x3, C -> C)
Params: $25C^2$	Params: $18C^2$
FLOPs: $25C^2HW$	FLOPs: $18C^2HW$



Simonyan and Zissermann, "Very Deep Convolutional Networks for Large-Scale Image Recognition", ICLR 2015

Slide from Justin Johnson

VGG: Deeper Networks, Regular Design

<u>VGG Design rules:</u>

All conv are 3x3 stride 1 pad 1

All max pool are 2x2 stride 2

After pool, double #channels

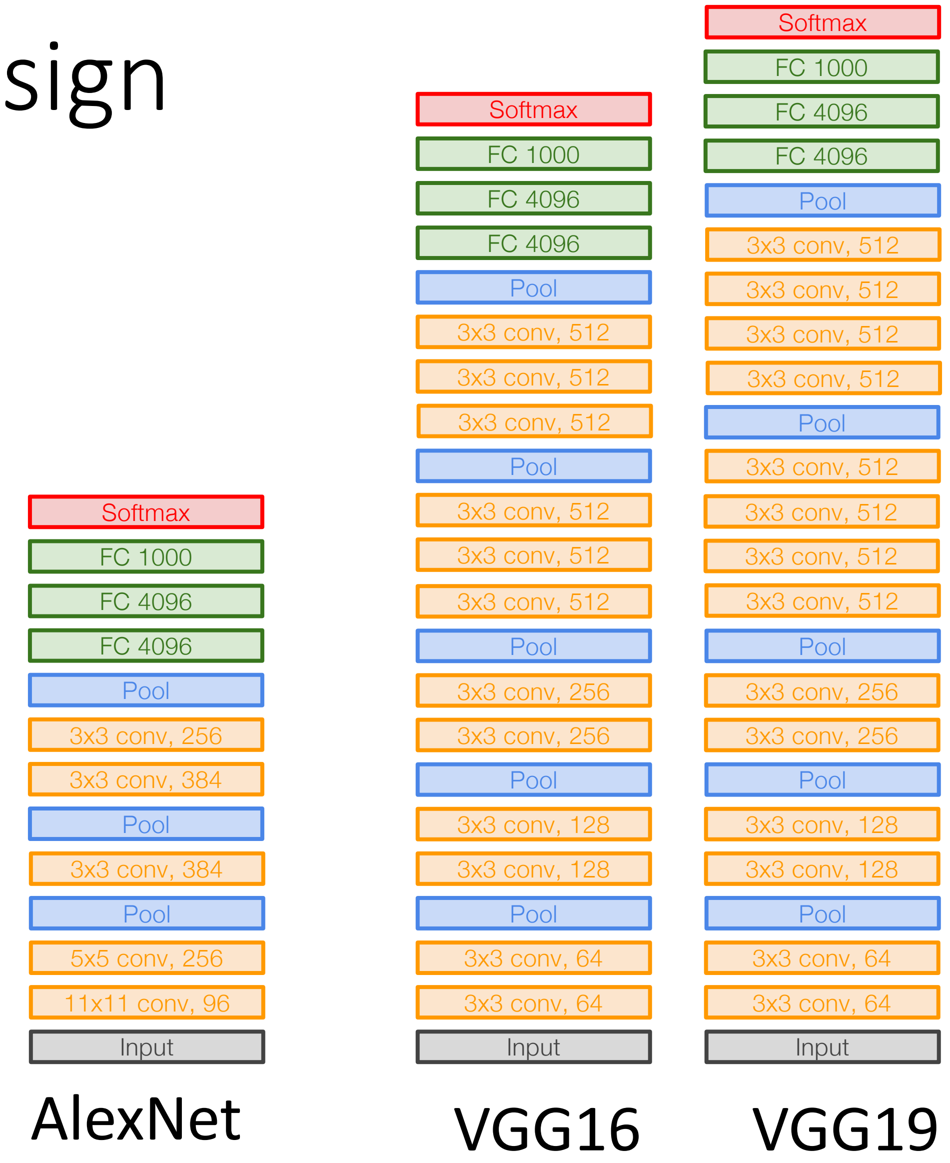
Input: C x 2H x 2W

Layer: Conv(3x3, C->C)

Memory: 4HWC

Params: $9C^2$

FLOPs: $36HWC^2$



Simonyan and Zissermann, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, ICLR 2015

Slide from Justin Johnson

VGG: Deeper Networks, Regular Design

<u>VGG Design rules:</u>

All conv are 3x3 stride 1 pad 1

All max pool are 2x2 stride 2

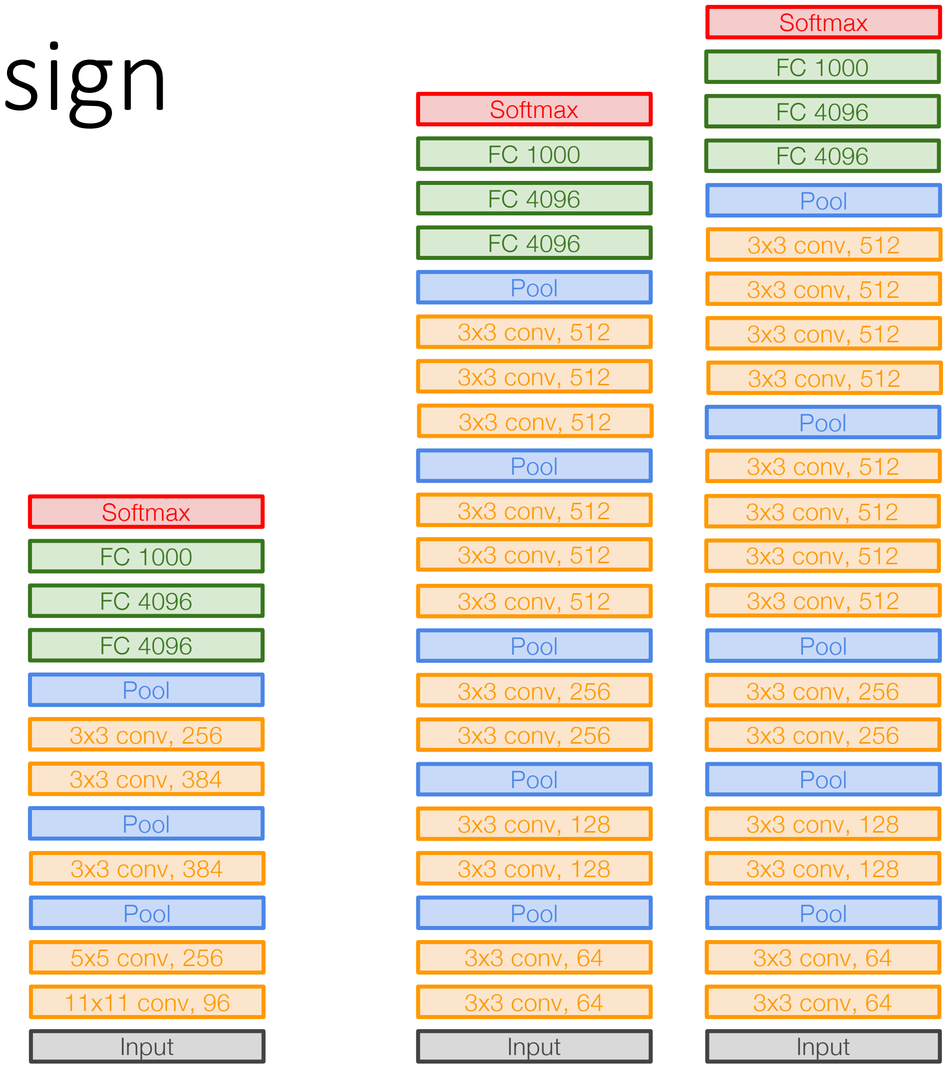
After pool, double #channels

Input: C x 2H x 2W
Layer: Conv(3x3, C->C)

Memory: 4HWC
Params: $9C^2$
FLOPs: $36HWC^2$

Input: 2C x H x W
Conv(3x3, 2C -> 2C)

Memory: 2HWC
Params: $36C^2$
FLOPs: $36HWC^2$



AlexNet VGG16 VGG19

Simonyan and Zissermann, "Very Deep Convolutional Networks for Large-Scale Image Recognition", ICLR 2015

Slide from Justin Johnson

VGG: Deeper Networks, Regular Design

<u>VGG Design rules:</u>

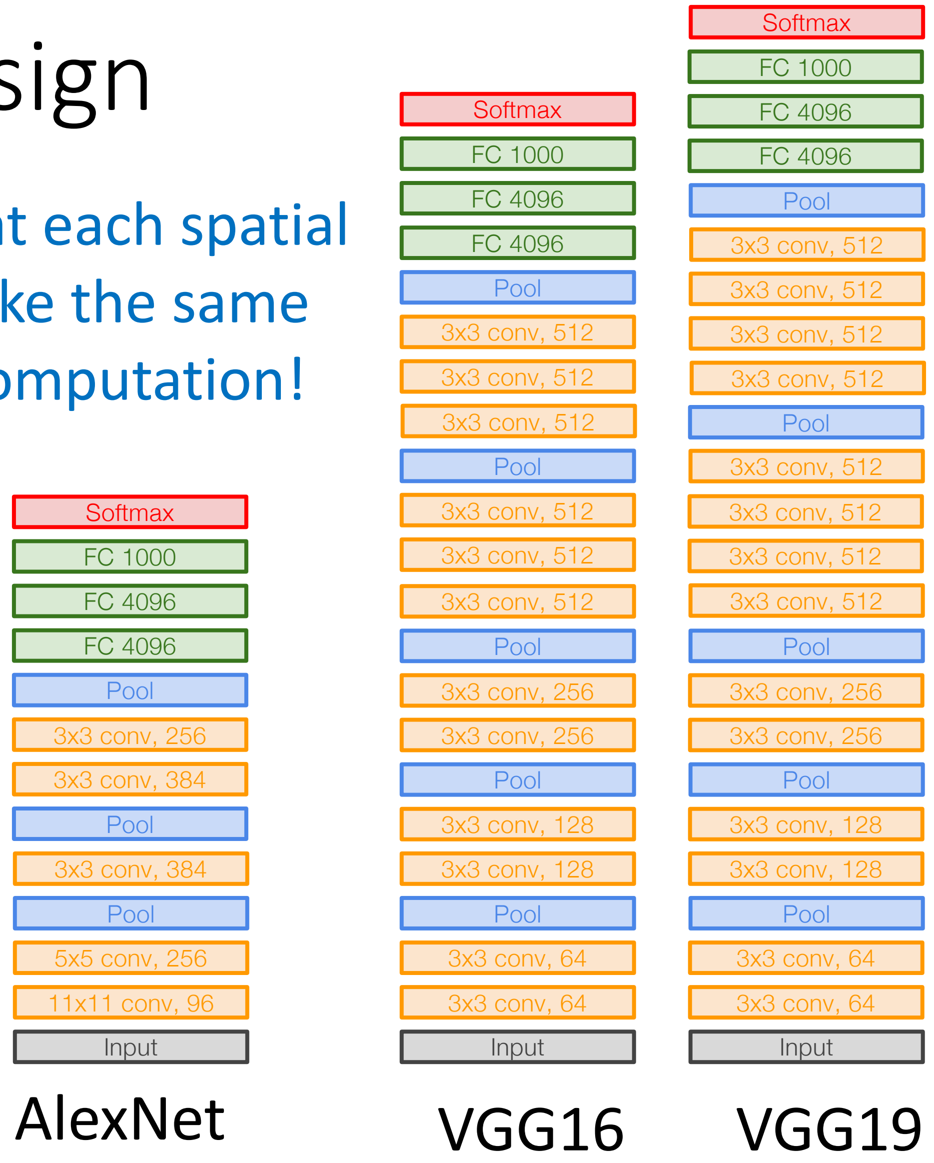
All conv are 3x3 stride 1 pad 1

All max pool are 2x2 stride 2

After pool, double #channels

Conv layers at each spatial resolution take the same amount of computation!

Input: C x 2H x 2W	Input: 2C x H x W
Layer: Conv(3x3, C->C)	Conv(3x3, 2C -> 2C)
Memory: 4HWC	Memory: 2HWC
Params: $9C^2$	Params: $36C^2$
FLOPs: $36HWC^2$	FLOPs: $36HWC^2$

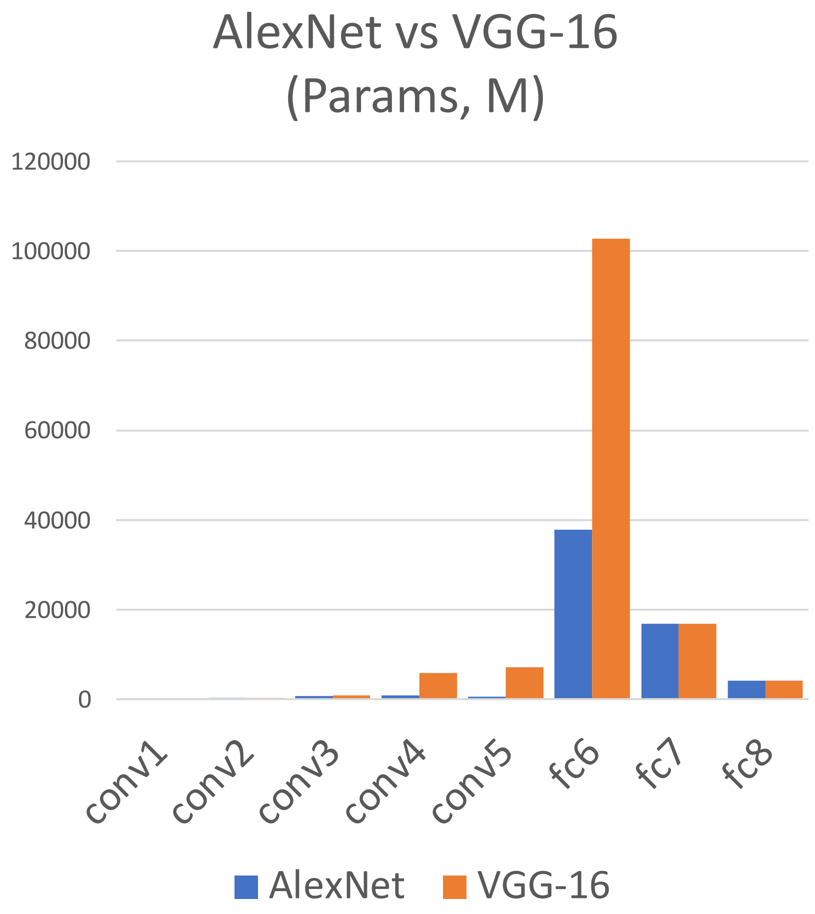


Simonyan and Zissermann, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, ICLR 2015

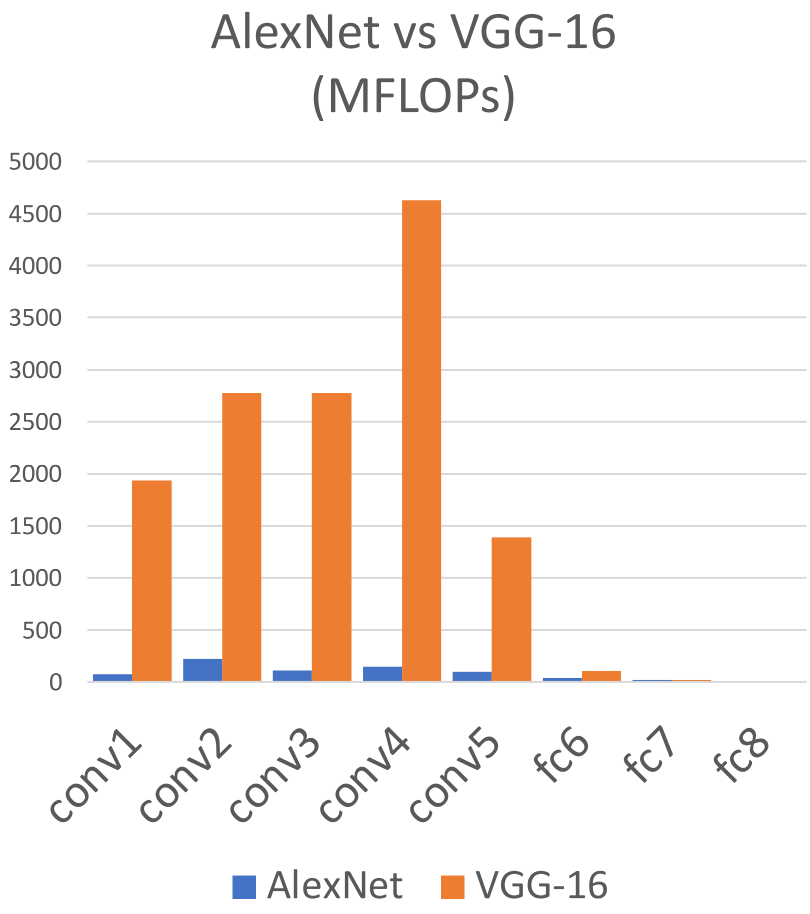
Slide from Justin Johnson

AlexNet vs VGG-16: Much bigger network!

AlexNet total: 1.9 MB
VGG-16 total: 48.6 MB (25x)



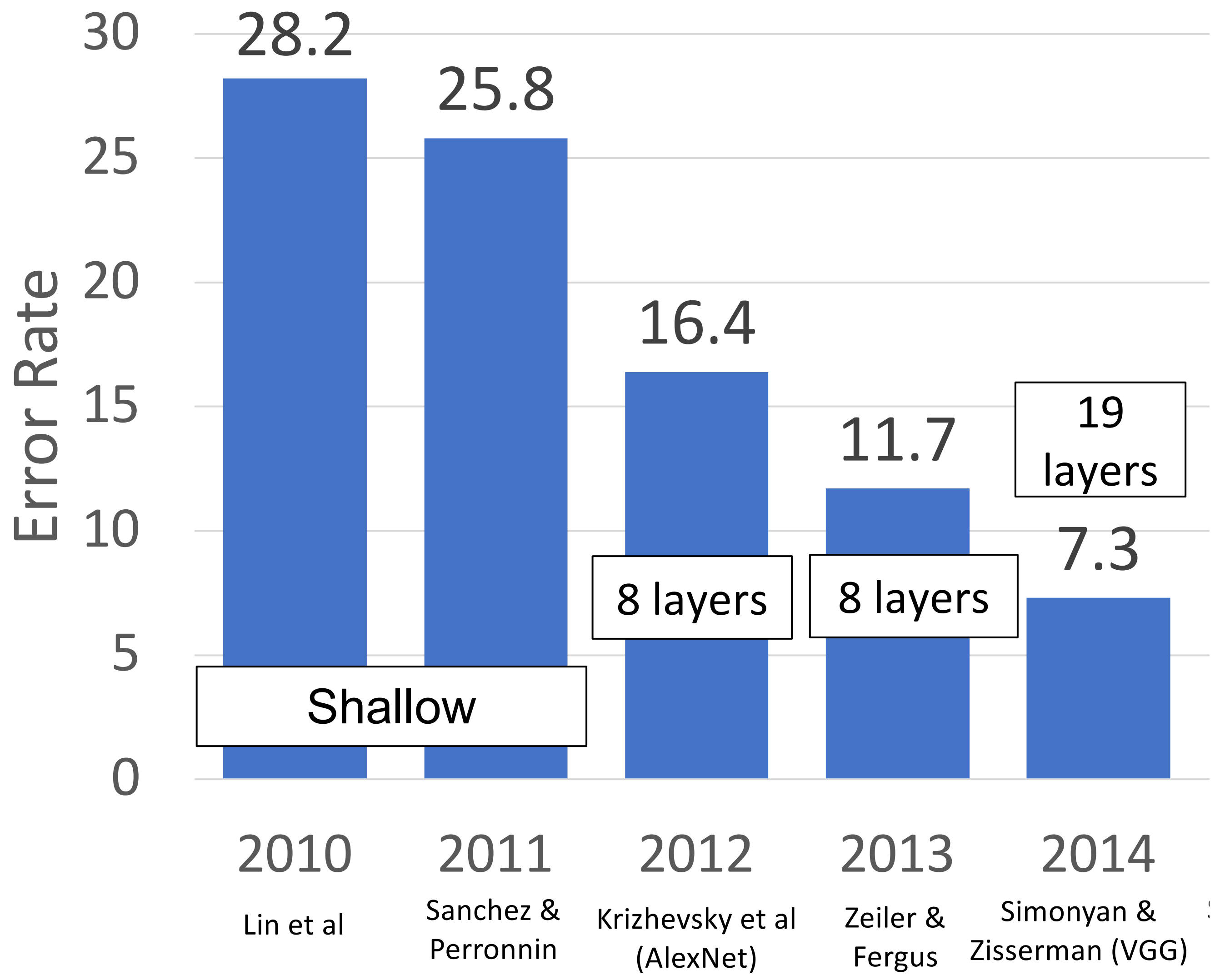
AlexNet total: 61M
VGG-16 total: 138M (2.3x)



AlexNet total: 0.7 GFLOP
VGG-16 total: 13.6 GFLOP (19.4x)

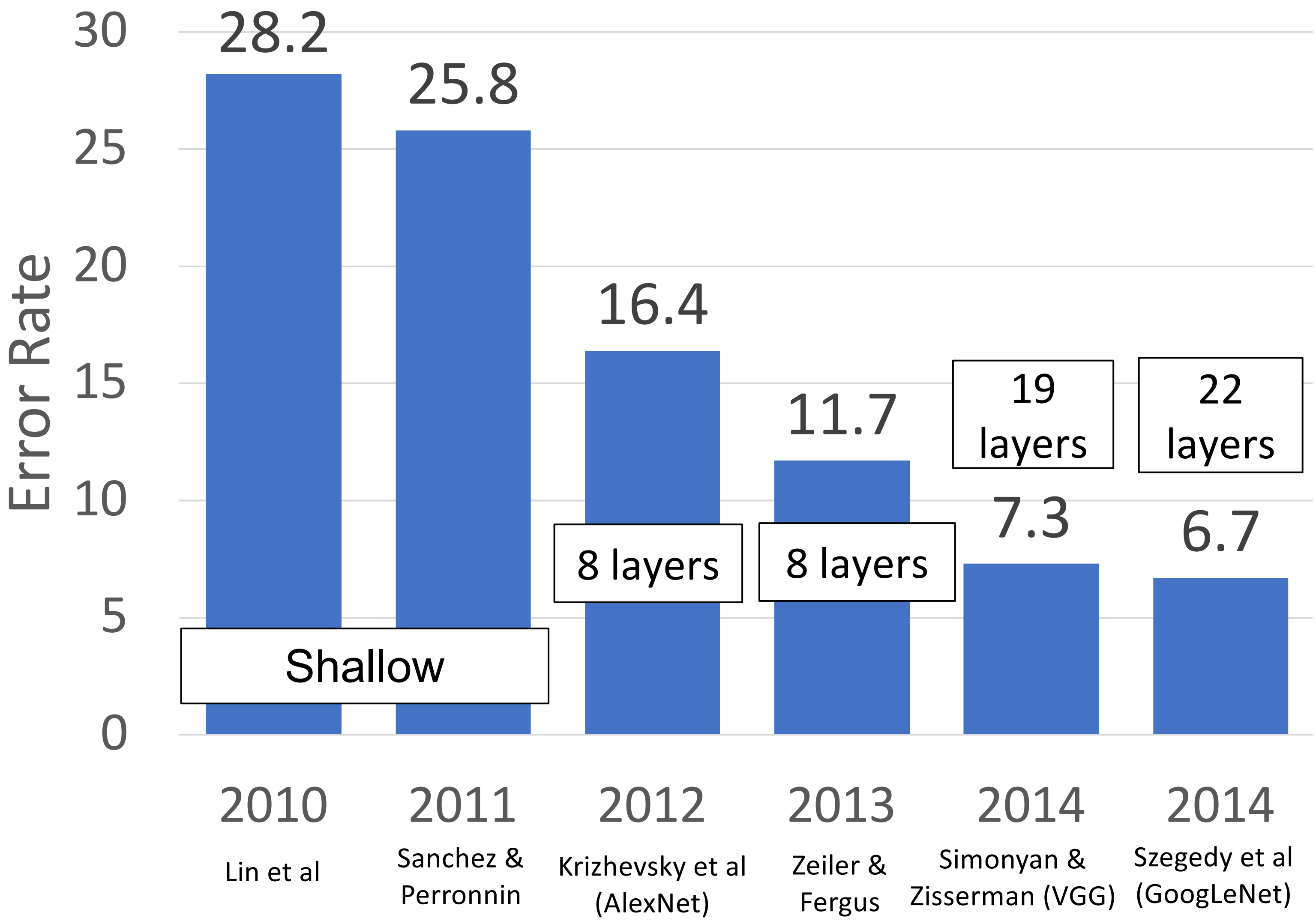
Slide from Justin Johnson

ImageNet Classification Challenge



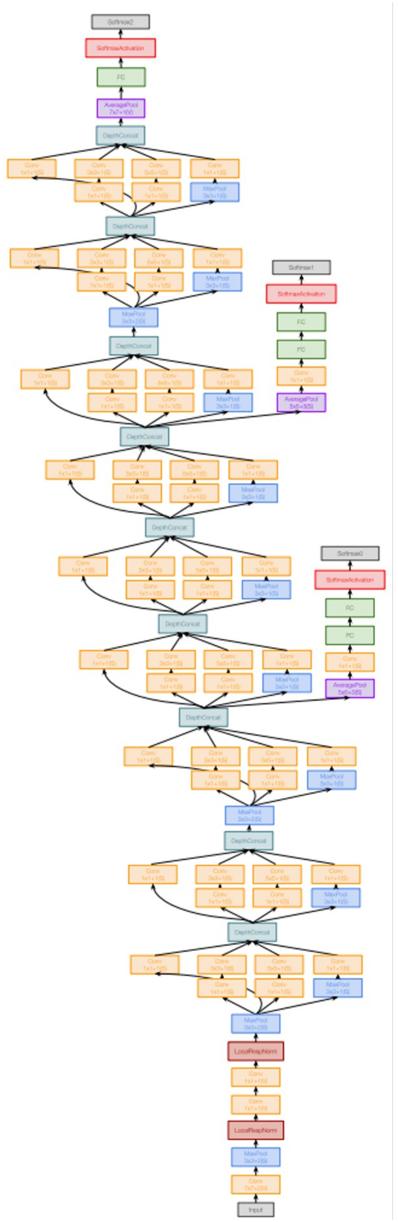
Slide from Justin Johnson

ImageNet Classification Challenge



Slide from Justin Johnson

GoogLeNet: Focus on Efficiency



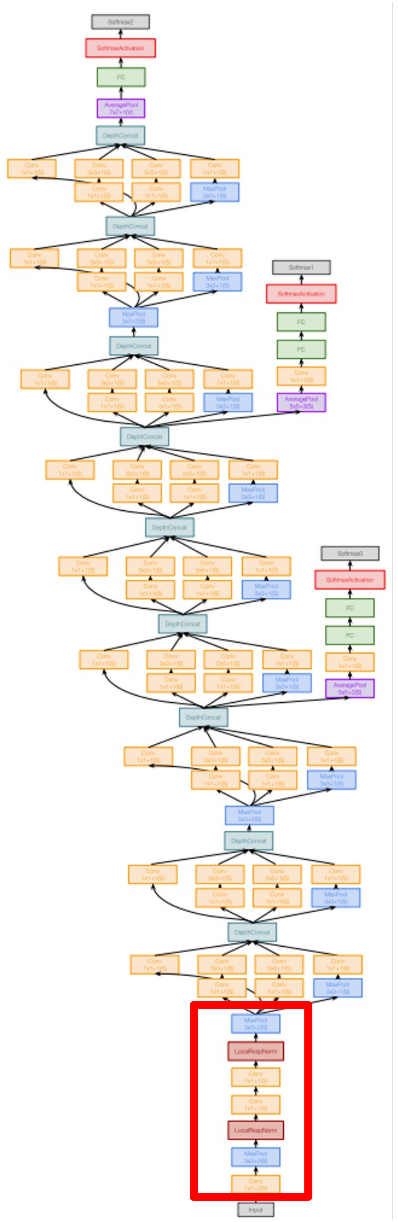
Many innovations for efficiency: reduce parameter count, memory usage, and computation

Szegedy et al, "Going deeper with convolutions", CVPR 2015

Slide from Justin Johnson

GoogLeNet: Aggressive Stem

Stem network at the start aggressively downsamples input
(Recall in VGG-16: Most of the compute was at the start)



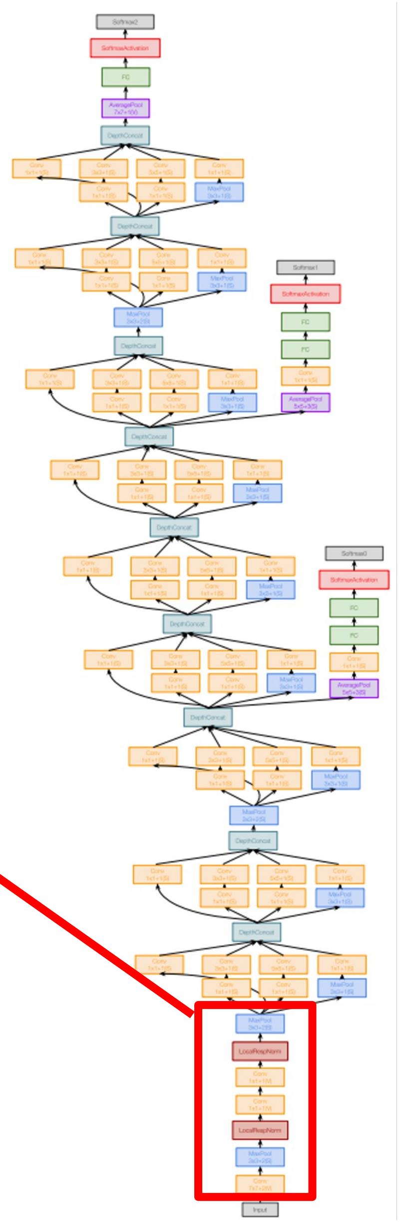
Szegedy et al, “Going deeper with convolutions”, CVPR 2015

Slide from Justin Johnson

GoogLeNet: Aggressive Stem

Stem network at the start aggressively downsamples input
(Recall in VGG-16: Most of the compute was at the start)

	Input size		Layer				Output size				
Layer	C	H / W	filters	kernel	stride	pad	C	H/W	memory (KB)	params (K)	flop (M)
conv	3	224	64	7	2	3	64	112	3136	9	118
max-pool	64	112		3	2	1	64	56	784	0	2
conv	64	56	64	1	1	0	64	56	784	4	13
conv	64	56	192	3	1	1	192	56	2352	111	347
max-pool	192	56		3	2	1	192	28	588	0	1



Total from 224 to 28 spatial resolution:
Memory: 7.5 MB
Params: 124K
MFLOP: 418

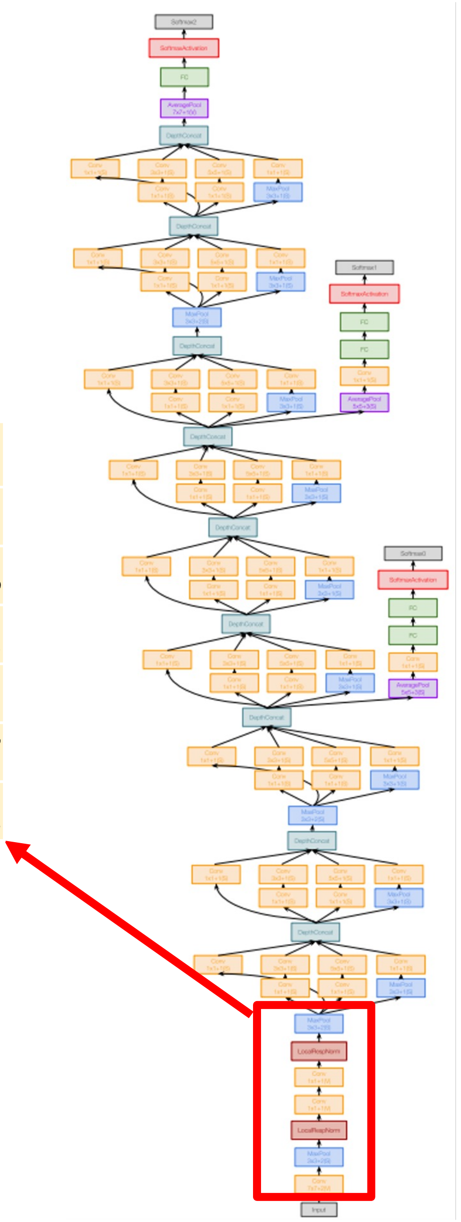
Szegedy et al, "Going deeper with convolutions", CVPR 2015

Slide from Justin Johnson

GoogLeNet: Aggressive Stem

Stem network at the start aggressively downsamples input
(Recall in VGG-16: Most of the compute was at the start)

	Input size		Layer				Output size				
Layer	C	H / W	filters	kernel	stride	pad	C	H/W	memory (KB)	params (K)	flop (M)
conv	3	224	64	7	2	3	64	112	3136	9	118
max-pool	64	112		3	2	1	64	56	784	0	2
conv	64	56	64	1	1	0	64	56	784	4	13
conv	64	56	192	3	1	1	192	56	2352	111	347
max-pool	192	56		3	2	1	192	28	588	0	1



Total from 224 to 28 spatial resolution:
Memory: 7.5 MB
Params: 124K
MFLOP: 418

Compare VGG-16:
Memory: 42.9 MB (5.7x)
Params: 1.1M (8.9x)
MFLOP: 7485 (17.8x)

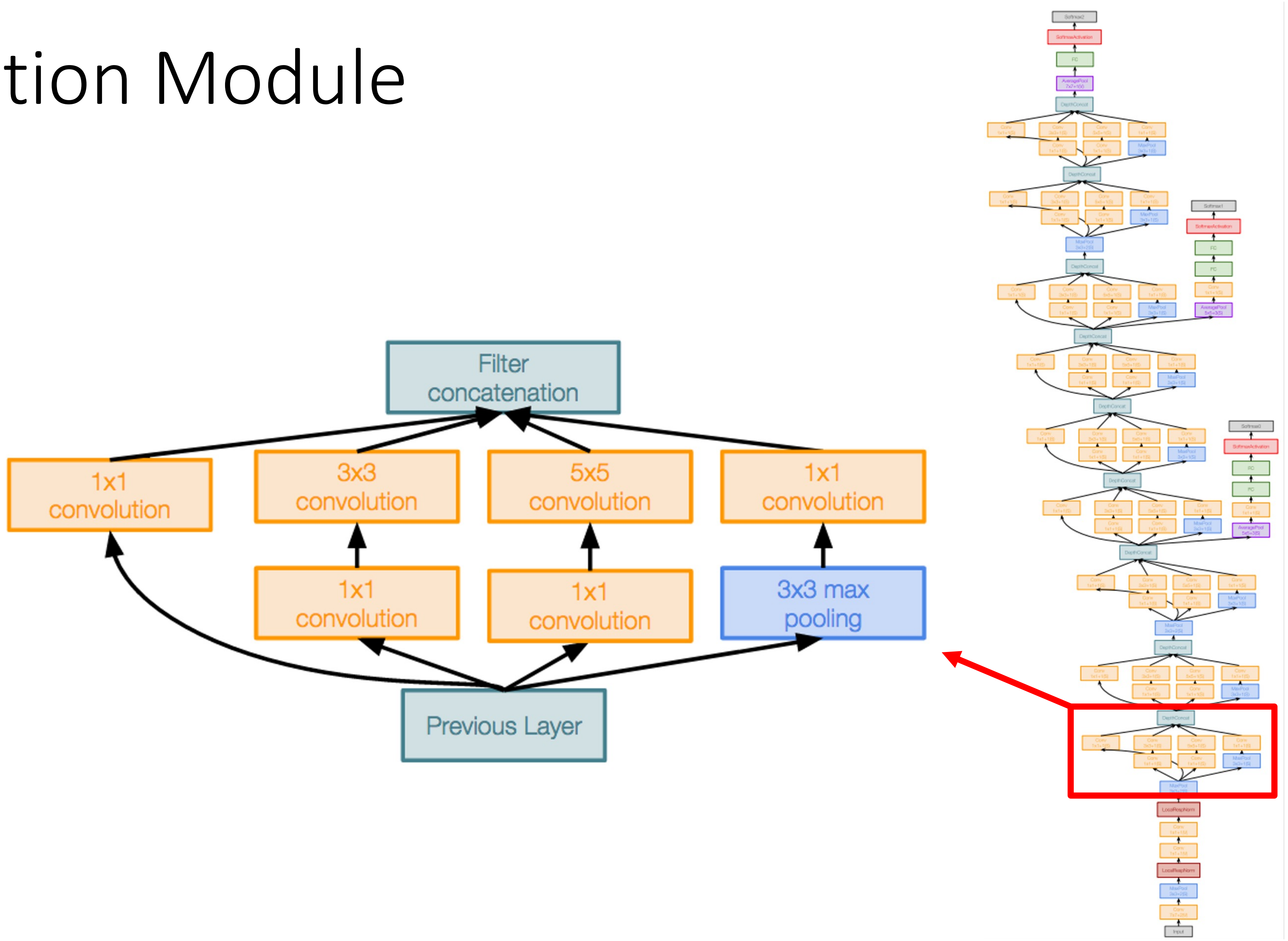
Szegedy et al, "Going deeper with convolutions", CVPR 2015

Slide from Justin Johnson

GoogLeNet: Inception Module

Inception module
Local unit with
parallel branches

Local structure repeated
many times throughout the
network



Szegedy et al, “Going deeper with convolutions”, CVPR 2015

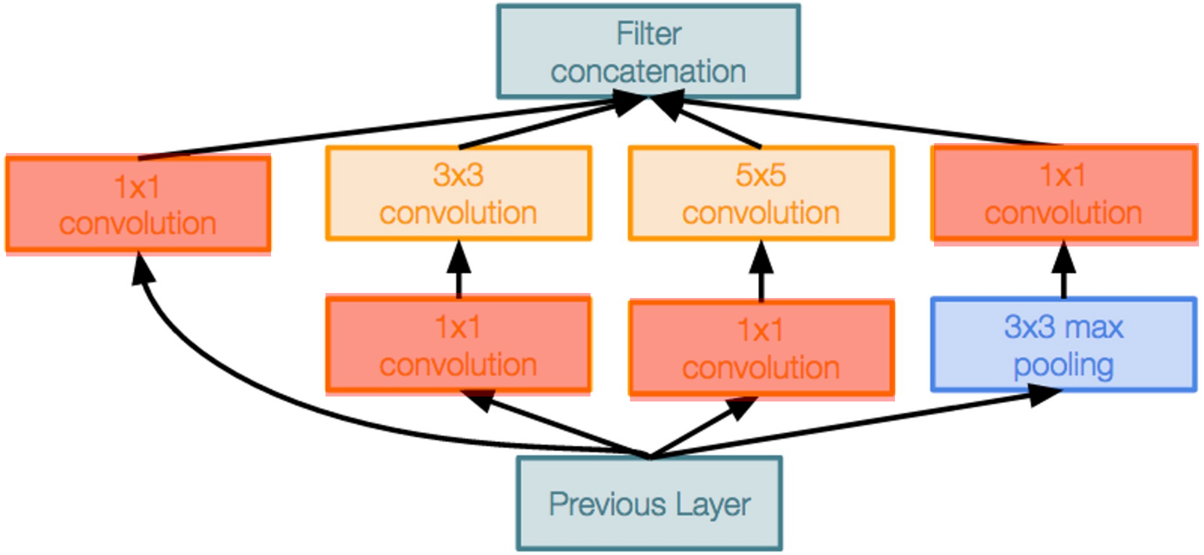
Slide from Justin Johnson

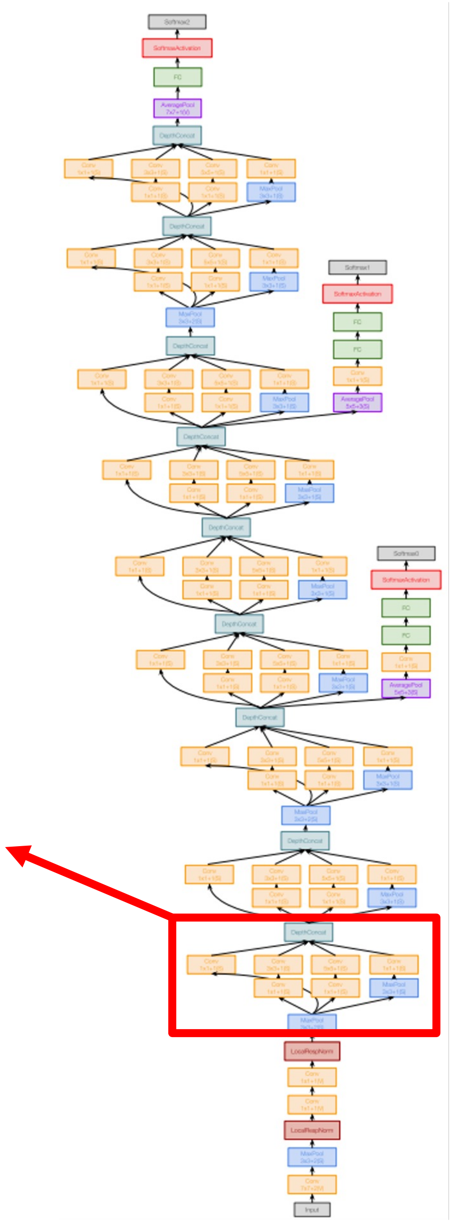
GoogLeNet: Inception Module

Inception module
Local unit with parallel branches

Local structure repeated many times throughout the network

Uses 1x1 "Bottleneck" layers to reduce channel dimension before expensive conv (we will revisit this with ResNet!)





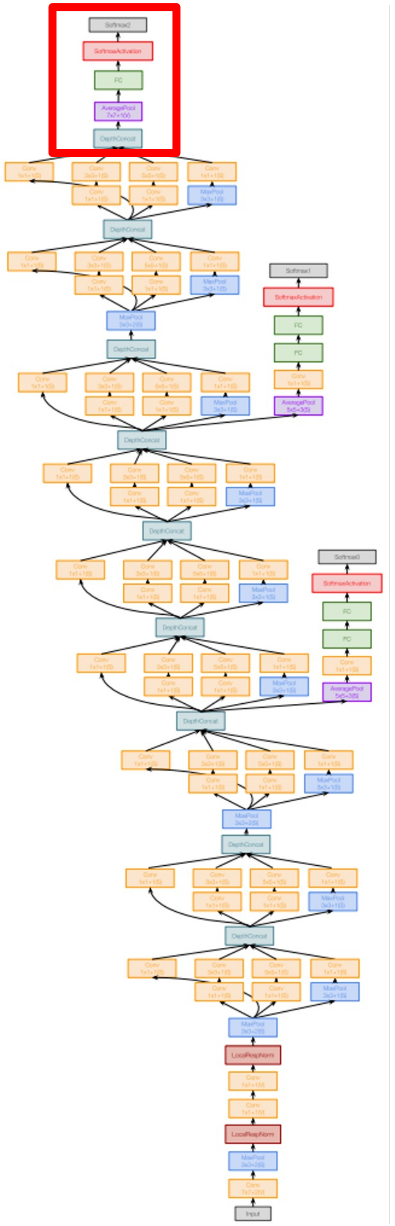
Szegedy et al, "Going deeper with convolutions", CVPR 2015

Slide from Justin Johnson

GoogLeNet: Global Average Pooling

No large FC layers at the end! Instead uses **global average pooling** to collapse spatial dimensions, and one linear layer to produce class scores
(Recall VGG-16: Most parameters were in the FC layers!)

	Input size		Layer				Output size				
Layer	C	H/W	filters	kernel	stride	pad	C	H/W	memory (KB)	params (k)	flop (M)
avg-pool	1024	7		7	1	0	1024	1	4	0	0
fc	1024		1000				1000		0	1025	1



Slide from Justin Johnson

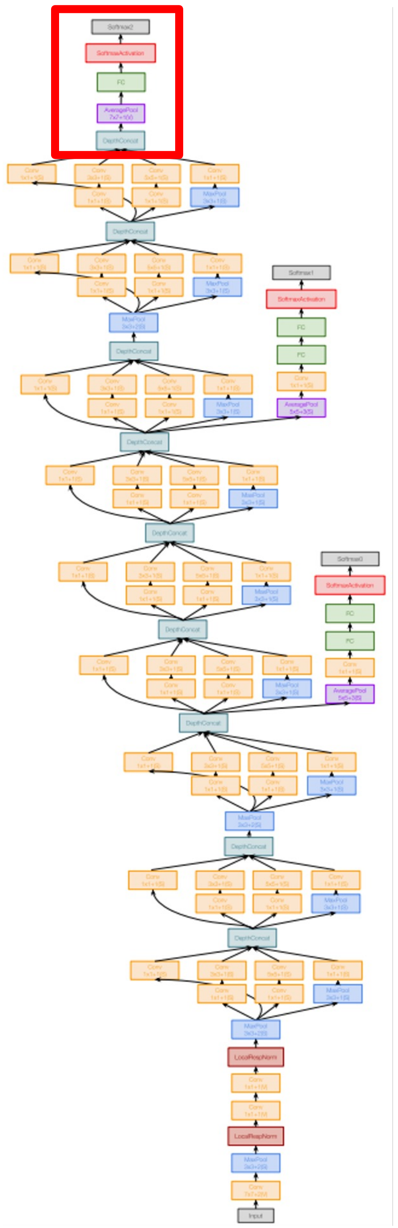
GoogLeNet: Global Average Pooling

No large FC layers at the end! Instead uses **global average pooling** to collapse spatial dimensions, and one linear layer to produce class scores
(Recall VGG-16: Most parameters were in the FC layers!)

	Input size		Layer				Output size				
Layer	C	H/W	filters	kernel	stride	pad	C	H/W	memory (KB)	params (k)	flop (M)
avg-pool	1024	7		7	1	0	1024	1	4	0	0
fc	1024		1000				1000		0	1025	1

Compare with VGG-16:

Layer	C	H/W	filters	kernel	stride	pad	C	H/W	memory (KB)	params (K)	flop (M)
flatten	512	7					25088		98		
fc6	25088			4096			4096		16	102760	103
fc7	4096			4096			4096		16	16777	17
fc8	4096			1000			1000		4	4096	4



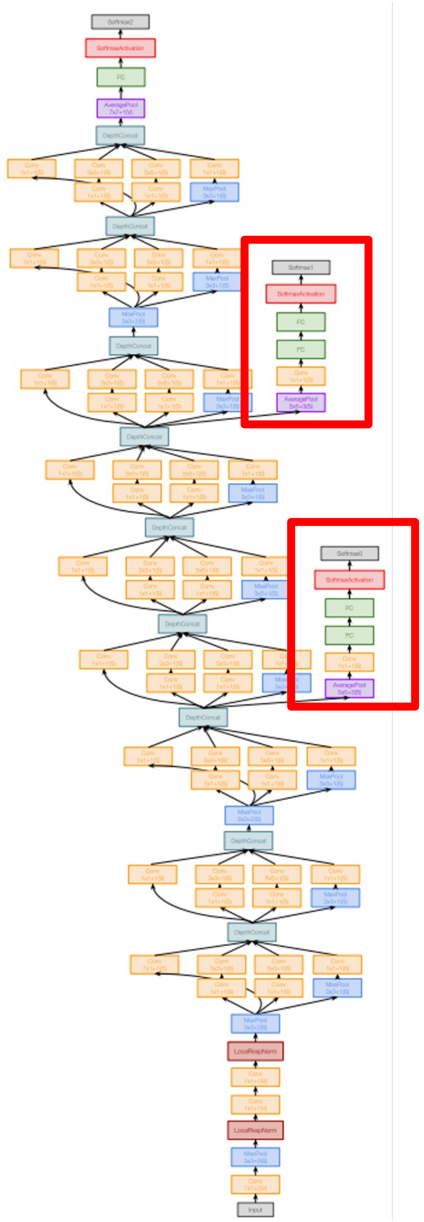
Slide from Justin Johnson

GoogLeNet: Auxiliary Classifiers

Training using loss at the end of the network didn’t work well: Network is too deep, gradients don’t propagate cleanly

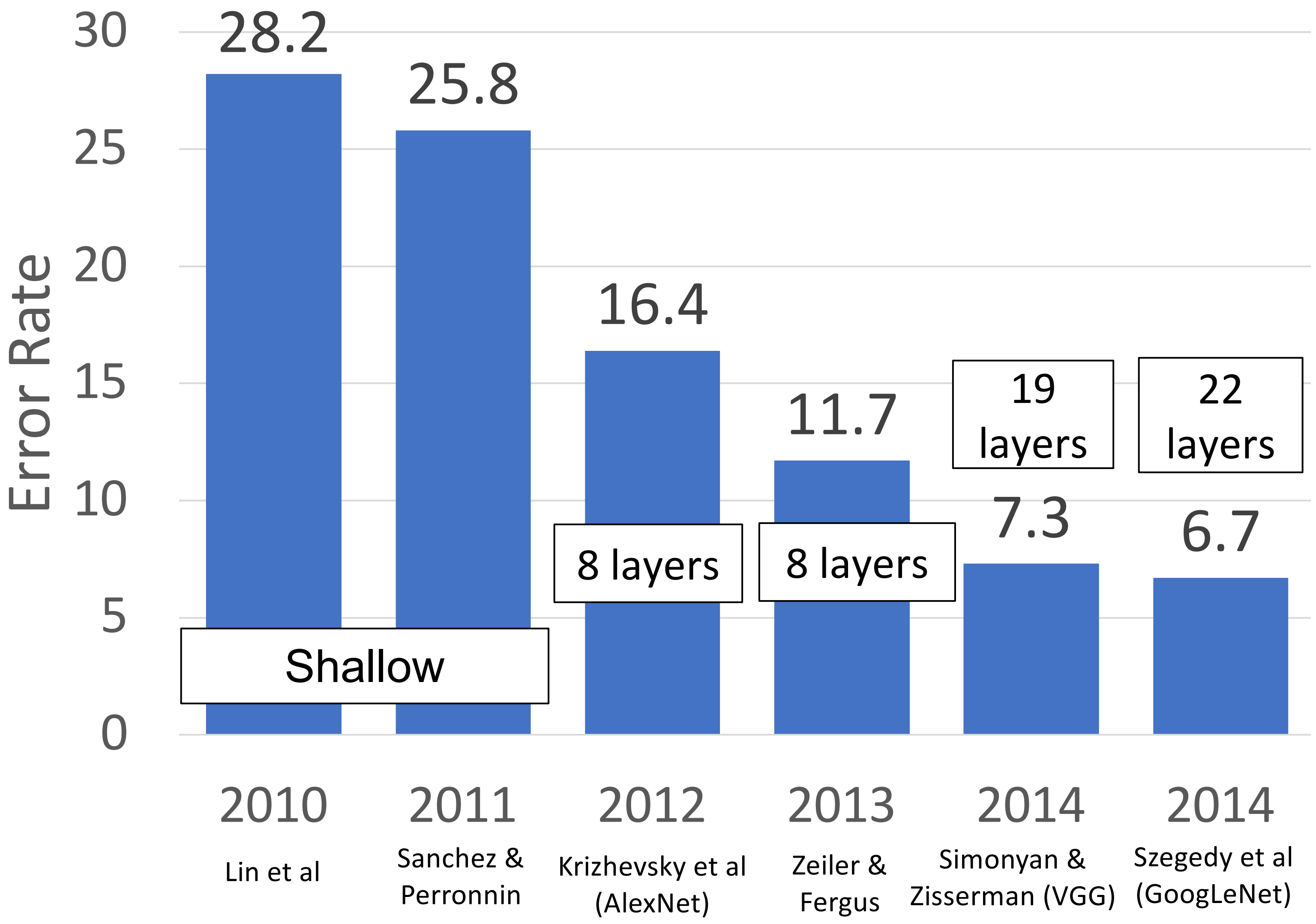
As a hack, attach “auxiliary classifiers” at several intermediate points in the network that also try to classify the image and receive loss

GoogLeNet was before batch normalization! With BatchNorm no longer need to use this trick



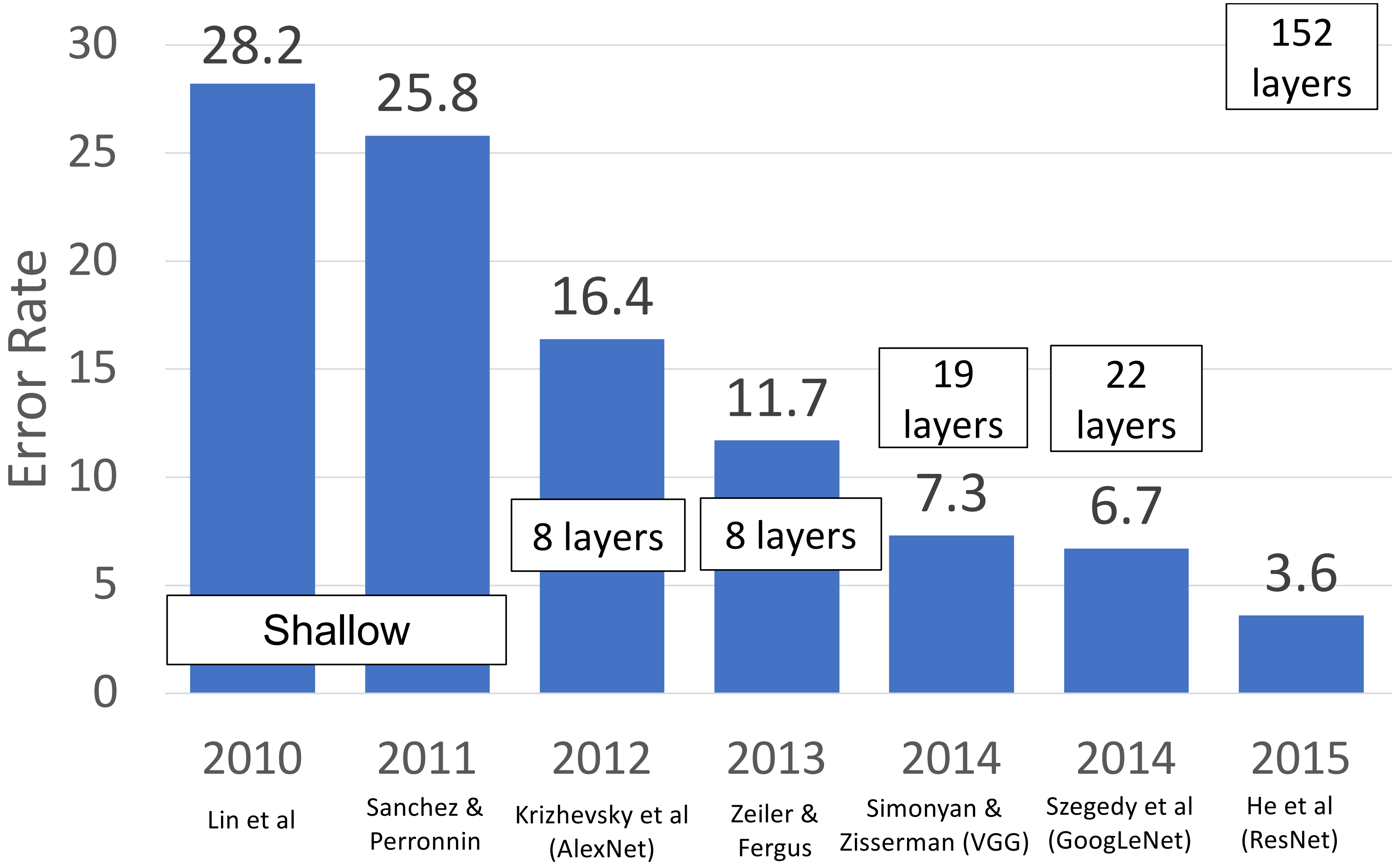
Slide from Justin Johnson

ImageNet Classification Challenge



Slide from Justin Johnson

ImageNet Classification Challenge



Slide from Justin Johnson

Residual Networks

Once we have Batch Normalization, we can train networks with 10+ layers.
What happens as we go deeper?

He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016

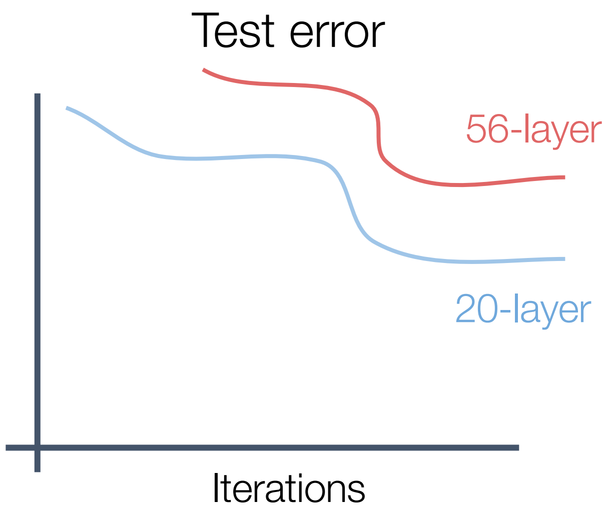
Slide from Justin Johnson

Residual Networks

Once we have Batch Normalization, we can train networks with 10+ layers. What happens as we go deeper?

Deeper model does worse than shallow model!

Initial guess: Deep model is **overfitting** since it is much bigger than the other model

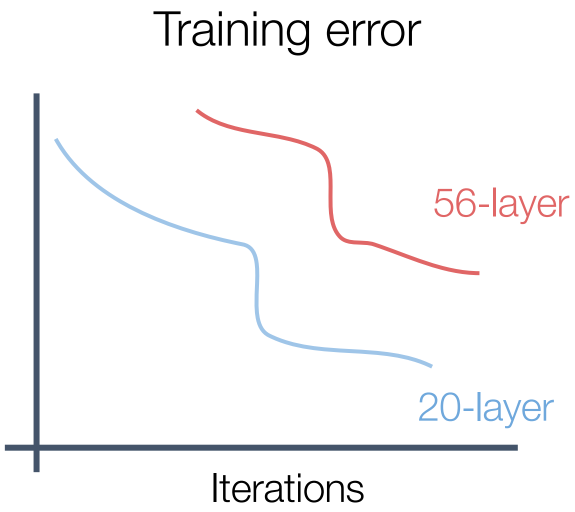


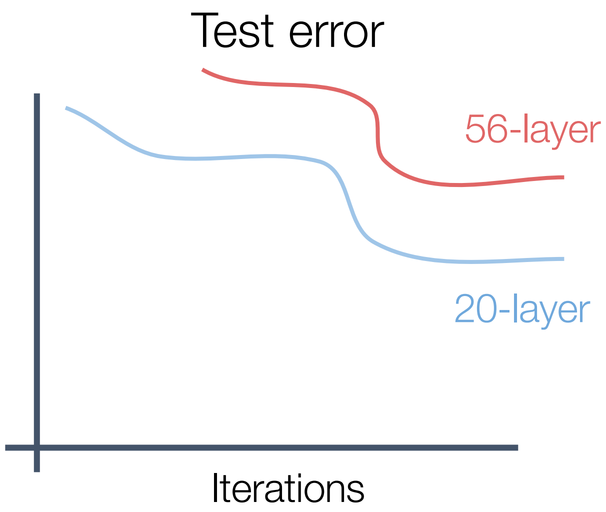
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016

Slide from Justin Johnson

Residual Networks

Once we have Batch Normalization, we can train networks with 10+ layers.
What happens as we go deeper?





In fact the deep model seems to be **underfitting** since it also performs worse than the shallow model on the training set! It is actually **underfitting**

He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

Slide from Justin Johnson

Residual Networks

A deeper model can emulate a shallower model: copy layers from shallower model, set extra layers to identity

Thus deeper models should do at least as good as shallow models

Hypothesis: This is an optimization problem. Deeper models are harder to optimize, and in particular don't learn identity functions to emulate shallow models

He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

Slide from Justin Johnson

Residual Networks

A deeper model can <u>emulate</u> a shallower model: copy layers from shallower model, set extra layers to identity

Thus deeper models should do at least as good as shallow models

Hypothesis: This is an <u>optimization</u> problem. Deeper models are harder to optimize, and in particular don't learn identity functions to emulate shallow models

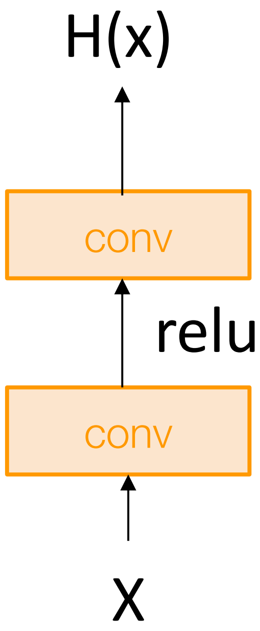
Solution: Change the network so learning identity functions with extra layers is easy!

He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

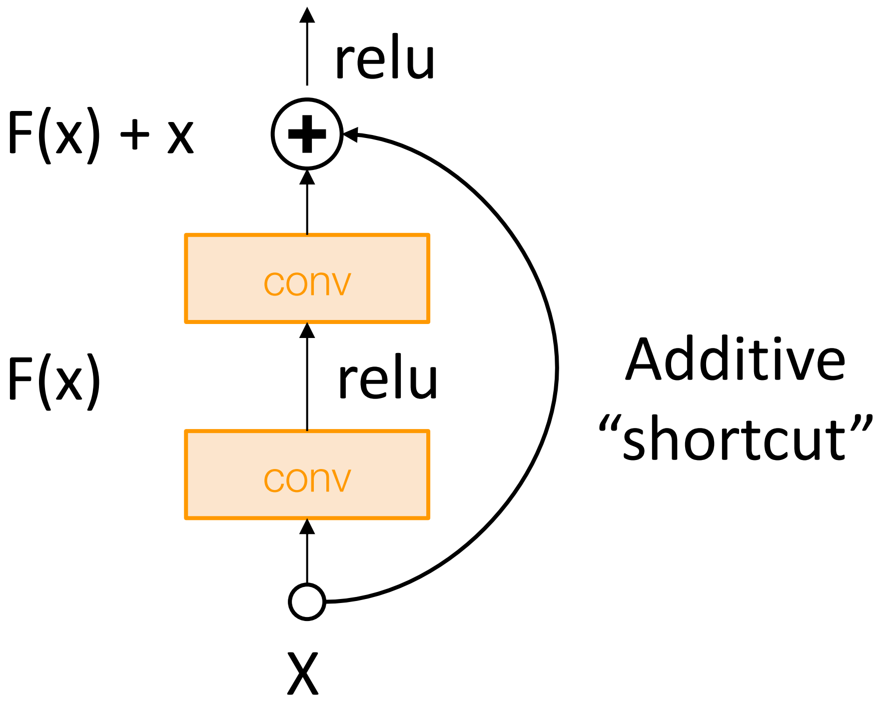
Slide from Justin Johnson

Residual Networks

Solution: Change the network so learning identity functions with extra layers is easy!



"Plain" block



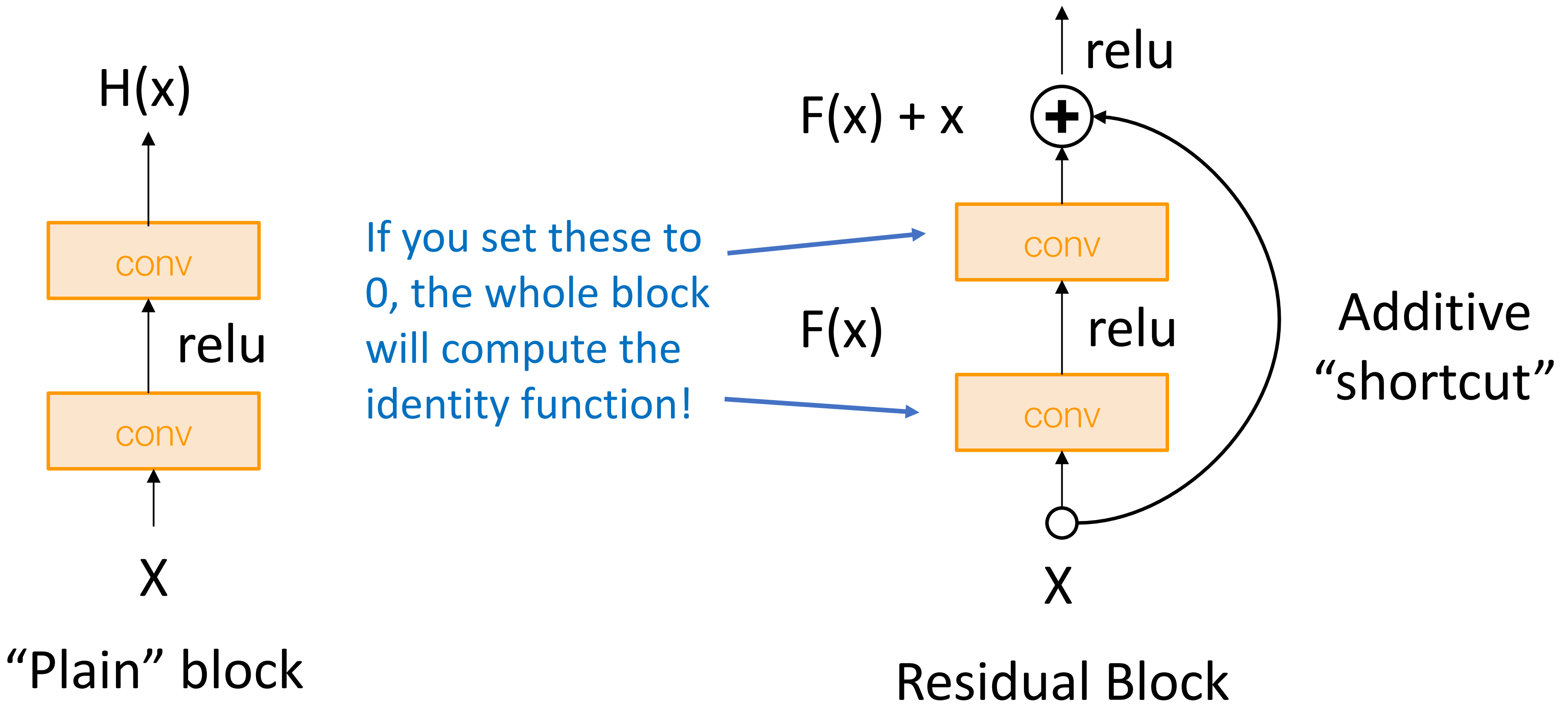
Residual Block

He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

Slide from Justin Johnson

Residual Networks

Solution: Change the network so learning identity functions with extra layers is easy!



He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

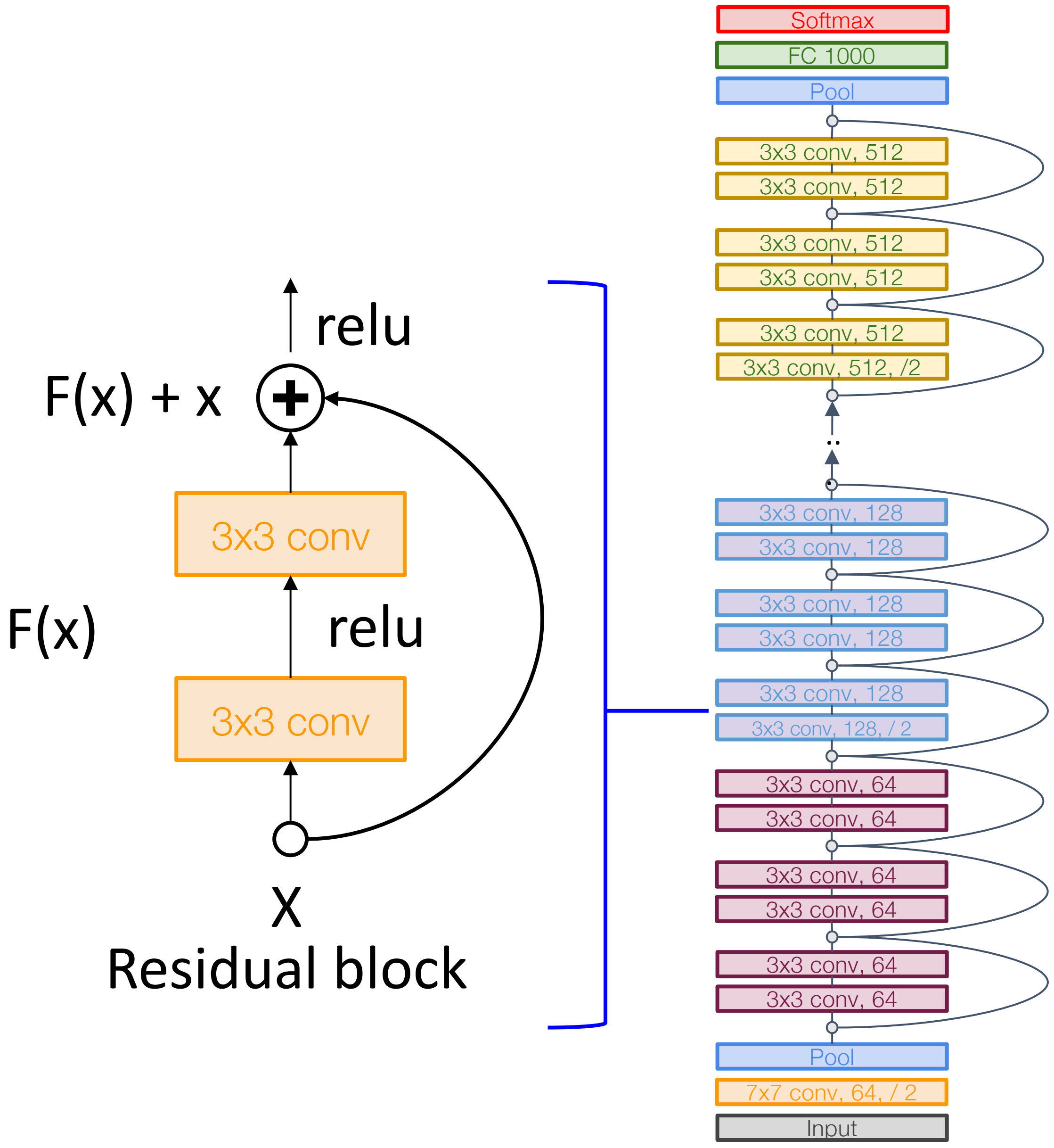
Slide from Justin Johnson

Residual Networks

A residual network is a stack of many residual blocks

Regular design, like VGG: each residual block has two 3x3 conv

Network is divided into **stages**: the first block of each stage halves the resolution (with stride-2 conv) and doubles the number of channels



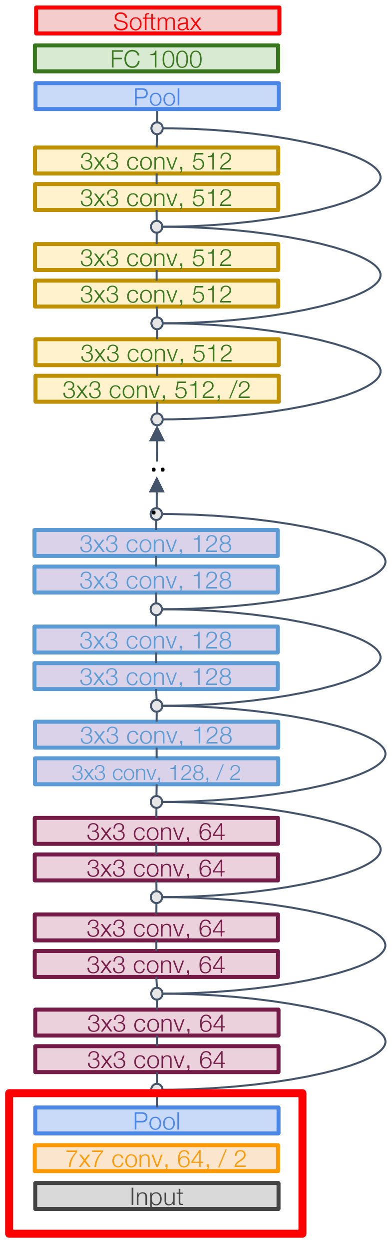
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016

Slide from Justin Johnson

Residual Networks

Uses the same aggressive **stem** as GoogleNet to downsample the input 4x before applying residual blocks:

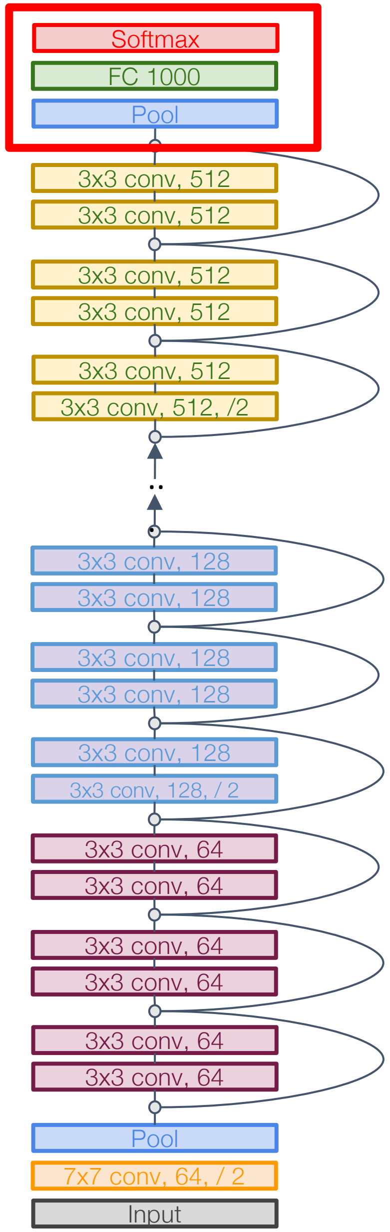
	Input size		Layer				Output size				
Layer	C	H/W	filters	kernel	stride	pad	C	H/W	memory (KB)	params (k)	flop (M)
conv	3	224	64	7	2	3	64	112	3136	9	118
max-pool	64	112		3	2	1	64	56	784	0	2



He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

Slide from Justin Johnson

Residual Networks



Like GoogLeNet, no big fully-connected-layers: instead use **global average pooling** and a single linear layer at the end

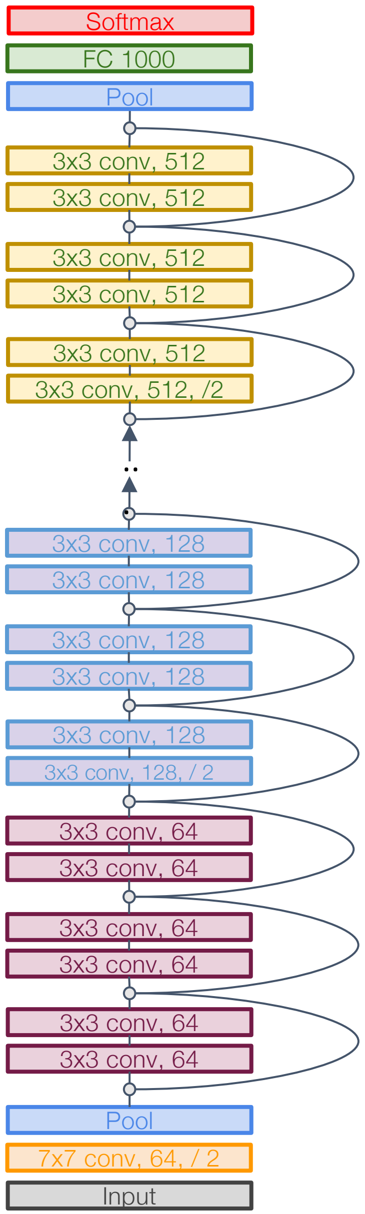
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

Slide from Justin Johnson

Residual Networks

ResNet-18:
Stem: 1 conv layer
Stage 1 (C=64): 2 res. block = 4 conv
Stage 2 (C=128): 2 res. block = 4 conv
Stage 3 (C=256): 2 res. block = 4 conv
Stage 4 (C=512): 2 res. block = 4 conv
Linear

ImageNet top-5 error: 10.92
GFLOP: 1.8



He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Error rates are 224x224 single-crop testing, reported by [torchvision]

Slide from Justin Johnson

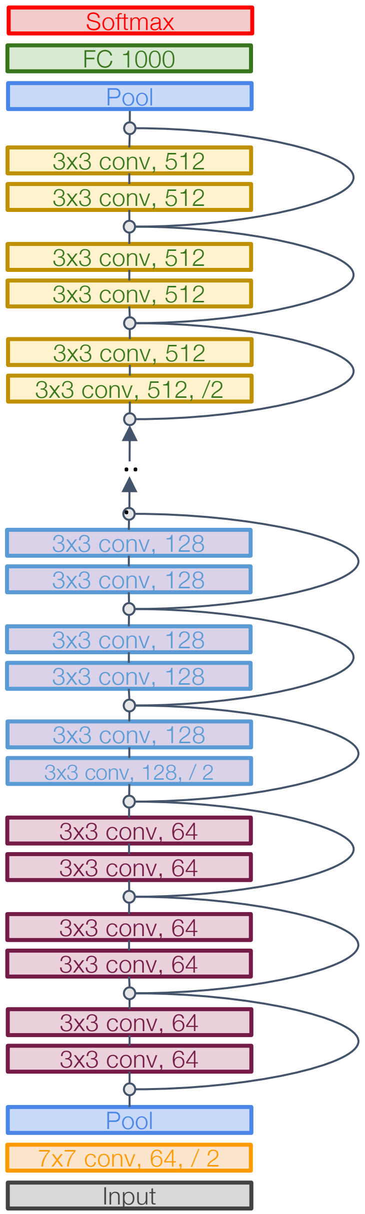
Residual Networks

ResNet-18:
Stem: 1 conv layer
Stage 1 (C=64): 2 res. block = 4 conv
Stage 2 (C=128): 2 res. block = 4 conv
Stage 3 (C=256): 2 res. block = 4 conv
Stage 4 (C=512): 2 res. block = 4 conv
Linear

ImageNet top-5 error: 10.92
GFLOP: 1.8

ResNet-34:
Stem: 1 conv layer
Stage 1: 3 res. block = 6 conv
Stage 2: 4 res. block = 8 conv
Stage 3: 6 res. block = 12 conv
Stage 4: 3 res. block = 6 conv
Linear

ImageNet top-5 error: 8.58
GFLOP: 3.6



He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Error rates are 224x224 single-crop testing, reported by torchvision

Slide from Justin Johnson

Residual Networks

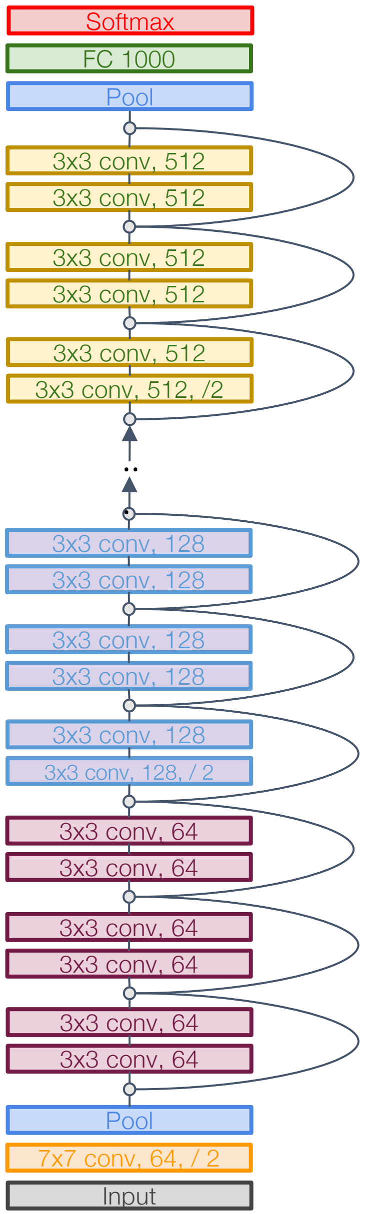
ResNet-18:
Stem: 1 conv layer
Stage 1 (C=64): 2 res. block = 4 conv
Stage 2 (C=128): 2 res. block = 4 conv
Stage 3 (C=256): 2 res. block = 4 conv
Stage 4 (C=512): 2 res. block = 4 conv
Linear

ImageNet top-5 error: 10.92
GFLOP: 1.8

ResNet-34:
Stem: 1 conv layer
Stage 1: 3 res. block = 6 conv
Stage 2: 4 res. block = 8 conv
Stage 3: 6 res. block = 12 conv
Stage 4: 3 res. block = 6 conv
Linear

ImageNet top-5 error: 8.58
GFLOP: 3.6

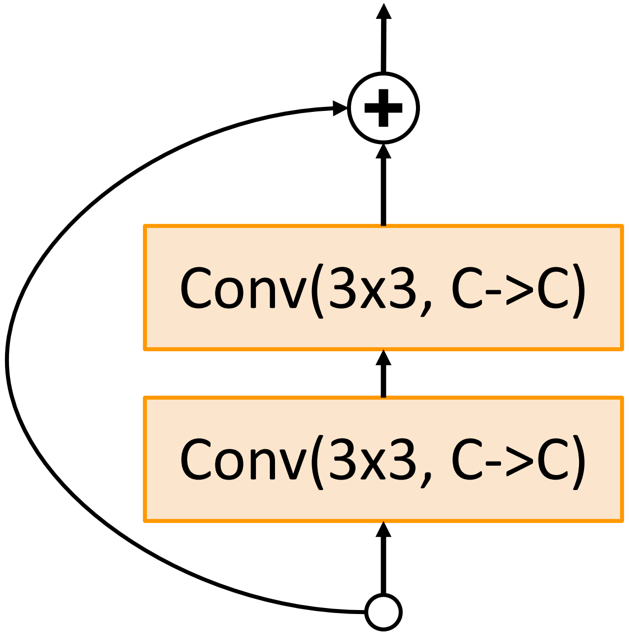
VGG-16:
ImageNet top-5 error: 9.62
GFLOP: 13.6



He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Error rates are 224x224 single-crop testing, reported by torchvision

Slide from Justin Johnson

Residual Networks: Basic Block

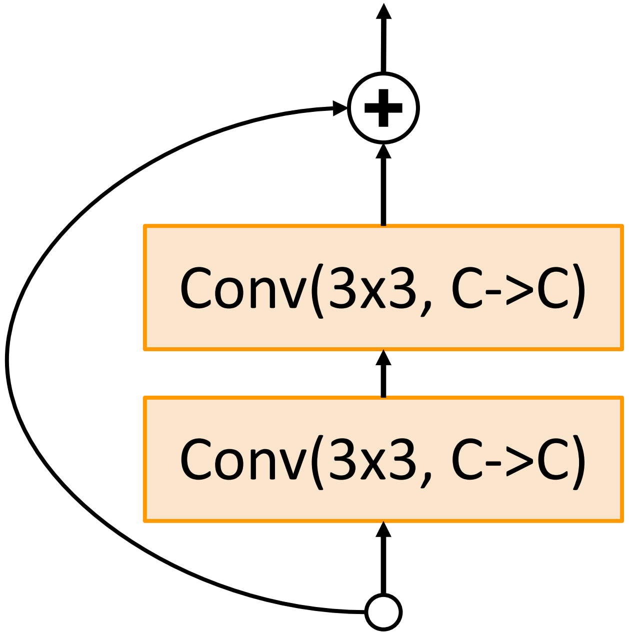


"Basic"
Residual block

He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

Slide from Justin Johnson

Residual Networks: Basic Block



FLOPs: $9HWC^2$

FLOPs: $9HWC^2$

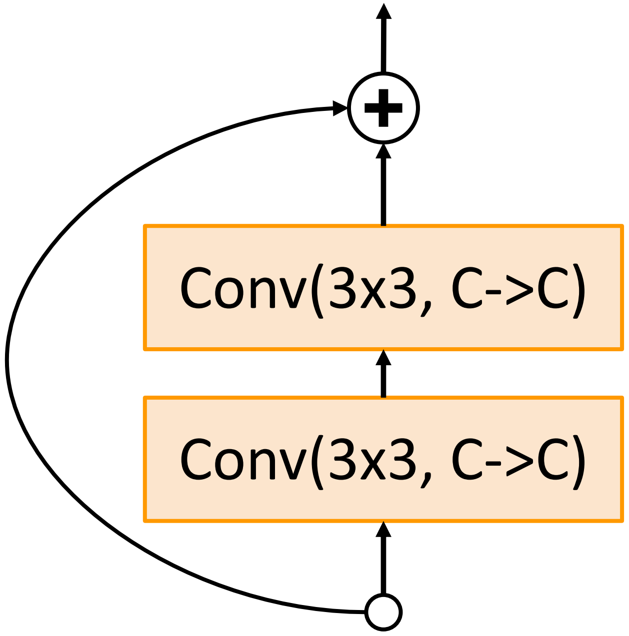
"Basic"
Residual block

Total FLOPs:
$18HWC^2$

He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

Slide from Justin Johnson

Residual Networks: Bottleneck Block

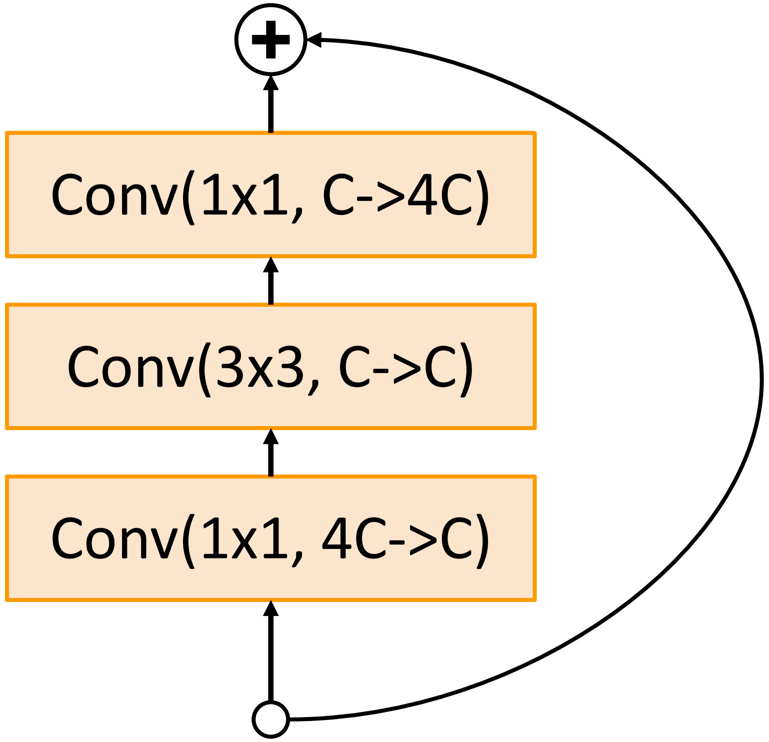


FLOPs: $9HWC^2$

FLOPs: $9HWC^2$

“Basic”
Residual block

Total FLOPs:
$18HWC^2$



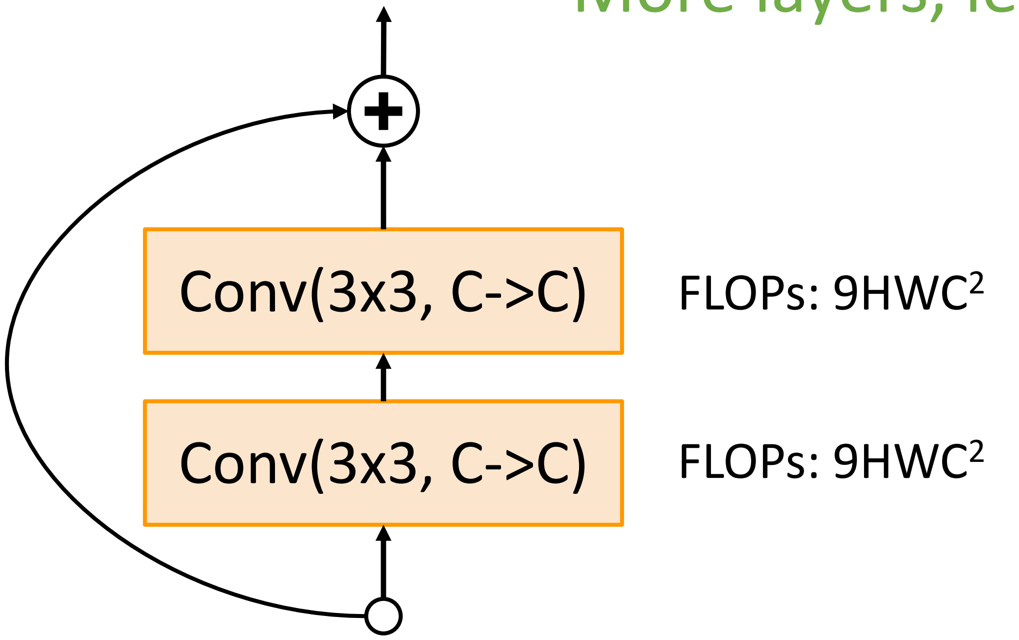
“Bottleneck”
Residual block

He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016

Slide from Justin Johnson

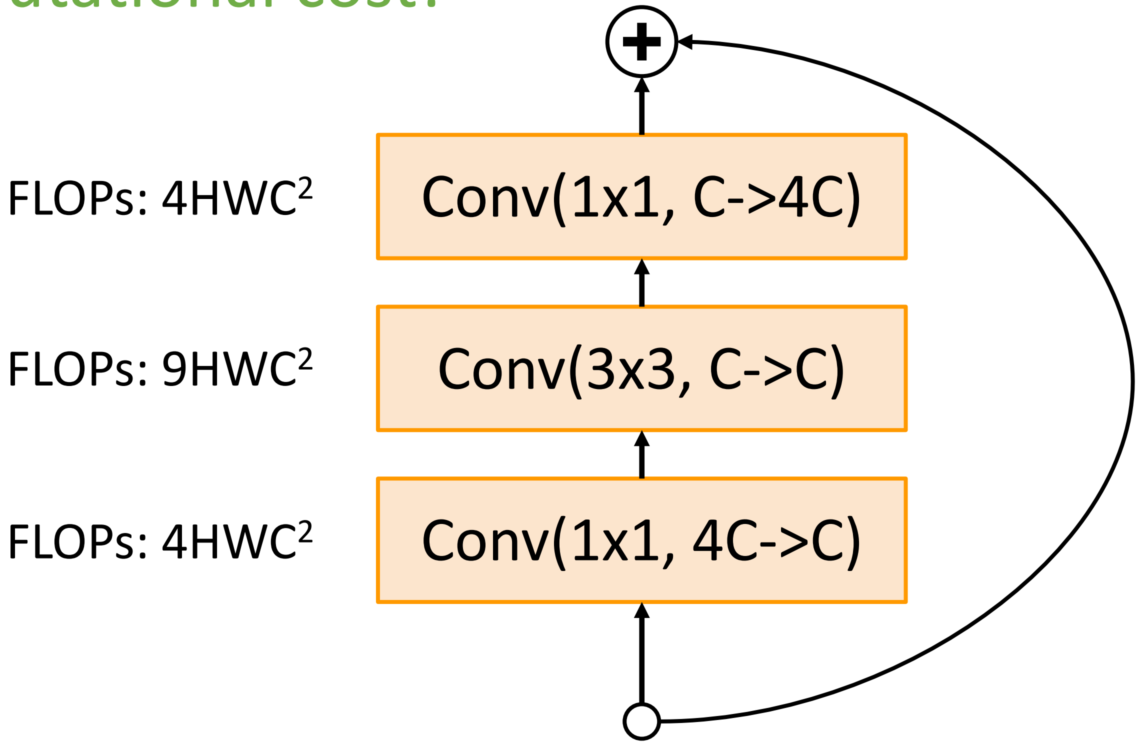
Residual Networks: Bottleneck Block

More layers, less computational cost!



"Basic"
Residual block

Total FLOPs:
$18HWC^2$



Total FLOPs:
$17HWC^2$

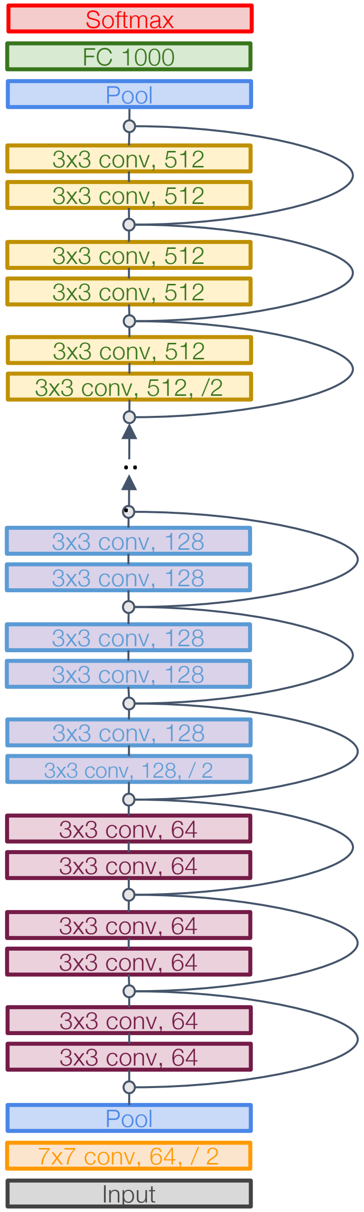
"Bottleneck"
Residual block

He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

Slide from Justin Johnson

Residual Networks

			Stage 1		Stage 2		Stage 3		Stage 4				
	Block type	Stem layers	Blocks	Layers	Blocks	Layers	Blocks	Layers	Blocks	Layers	FC layers	GFLOP	ImageNet top-5 error
ResNet-18	Basic	1	2	4	2	4	2	4	2	4	1	1.8	10.92
ResNet-34	Basic	1	3	6	4	8	6	12	3	6	1	3.6	8.58



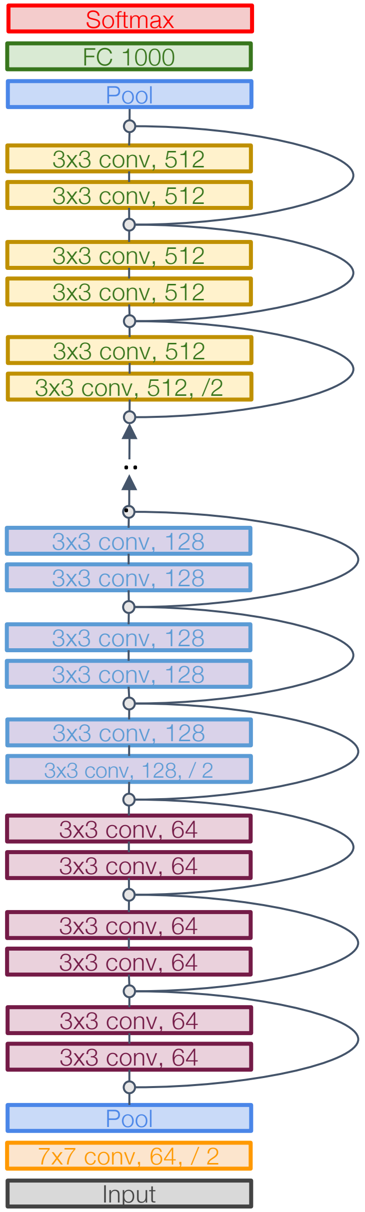
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Error rates are 224x224 single-crop testing, reported by torchvision

Slide from Justin Johnson

Residual Networks

ResNet-50 is the same as ResNet-34, but replaces Basic blocks with Bottleneck Blocks.
This is a great baseline architecture for many tasks even today!

			Stage 1		Stage 2		Stage 3		Stage 4				
	Block type	Stem layers	Blocks	Layers	Blocks	Layers	Blocks	Layers	Blocks	Layers	FC layers	GFLOP	ImageNet top-5 error
ResNet-18	Basic	1	2	4	2	4	2	4	2	4	1	1.8	10.92
ResNet-34	Basic	1	3	6	4	8	6	12	3	6	1	3.6	8.58
ResNet-50	Bottle	1	3	9	4	12	6	18	3	9	1	3.8	7.13



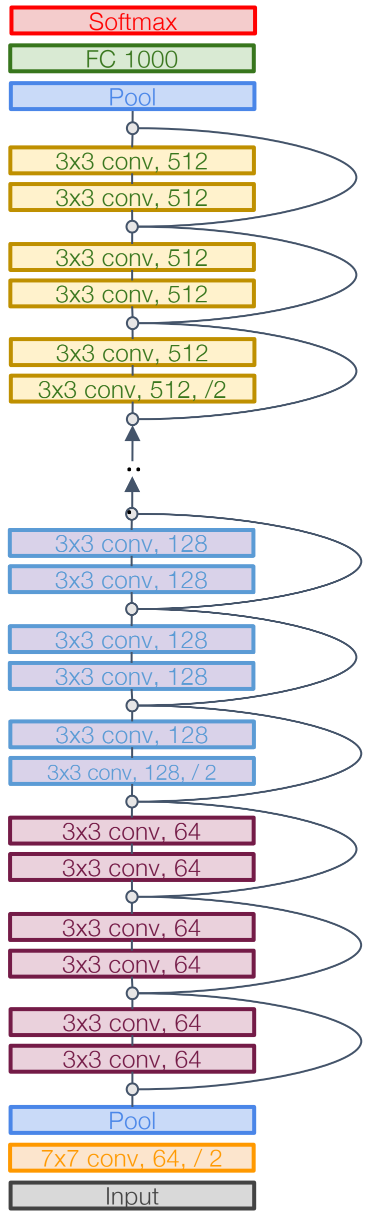
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Error rates are 224x224 single-crop testing, reported by torchvision

Slide from Justin Johnson

Residual Networks

Deeper ResNet-101 and ResNet-152 models are more accurate, but also more computationally heavy

			Stage 1		Stage 2		Stage 3		Stage 4				
	Block type	Stem layers	Blocks	Layers	Blocks	Layers	Blocks	Layers	Blocks	Layers	FC layers	GFLOP	ImageNet top-5 error
ResNet-18	Basic	1	2	4	2	4	2	4	2	4	1	1.8	10.92
ResNet-34	Basic	1	3	6	4	8	6	12	3	6	1	3.6	8.58
ResNet-50	Bottle	1	3	9	4	12	6	18	3	9	1	3.8	7.13
ResNet-101	Bottle	1	3	9	4	12	23	69	3	9	1	7.6	6.44
ResNet-152	Bottle	1	3	9	8	24	36	108	3	9	1	11.3	5.94



He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Error rates are 224x224 single-crop testing, reported by torchvision

Slide from Justin Johnson

Residual Networks

- Able to train very deep networks
- Deeper networks do better than shallow networks (as expected)
- Swept 1st place in all ILSVRC and COCO 2015 competitions
- Still widely used today!

MSRA @ ILSVRC & COCO 2015 Competitions

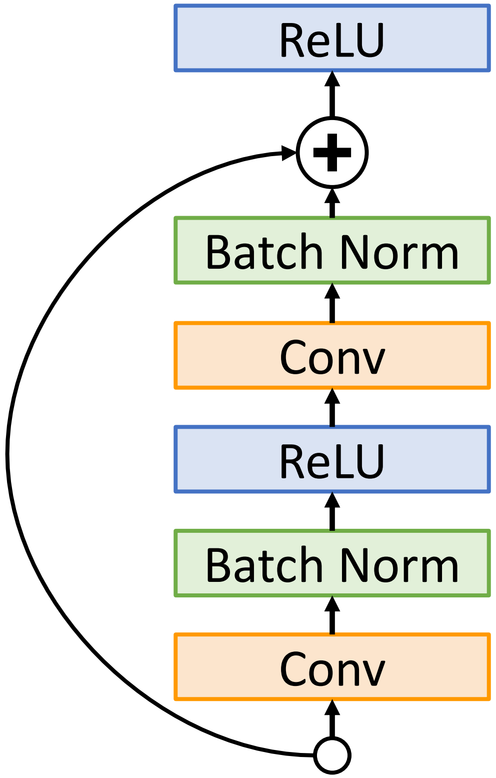
- **1st places in all five main tracks**
  - ImageNet Classification: "*Ultra-deep*" (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

He et al, "Deep Residual Learning for Image Recognition", CVPR 2016

Slide from Justin Johnson

Improving Residual Networks: Block Design

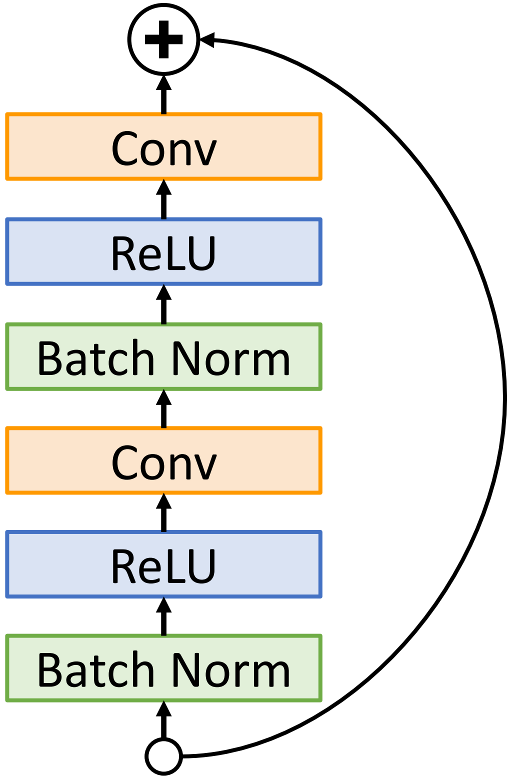
Original ResNet block



Note ReLU **after** residual:

Cannot actually learn identity function since outputs are nonnegative!

"Pre-Activation" ResNet Block



Note ReLU **inside** residual:

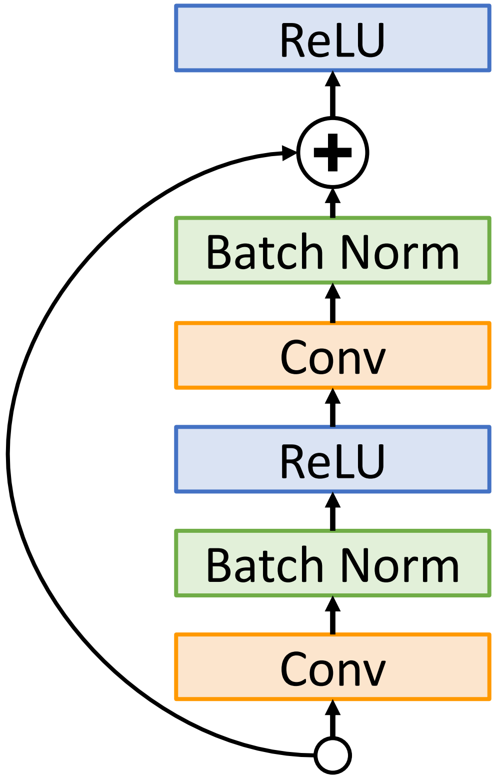
Can learn true identity function by setting Conv weights to zero!

He et al, "Identity mappings in deep residual networks", ECCV 2016

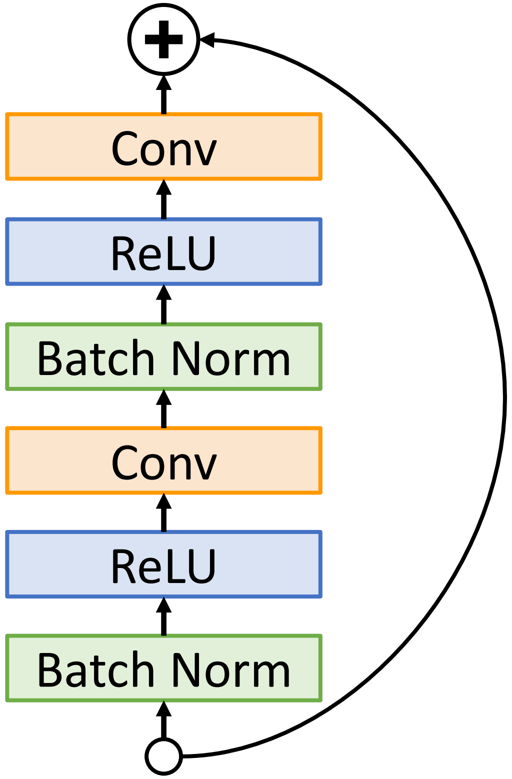
Slide from Justin Johnson

Improving Residual Networks: Block Design

Original ResNet block



"Pre-Activation" ResNet Block



Slight improvement in accuracy (ImageNet top-1 error)

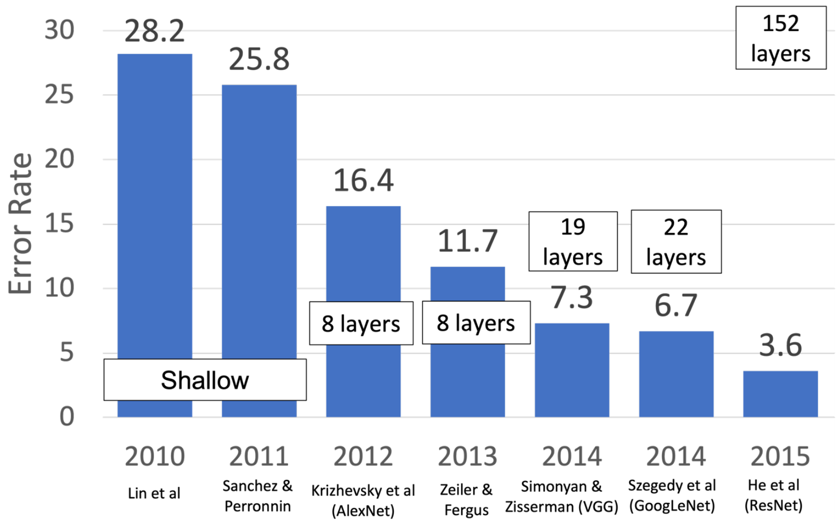
ResNet-152: 21.3 vs **21.1**
ResNet-200: 21.8 vs **20.7**

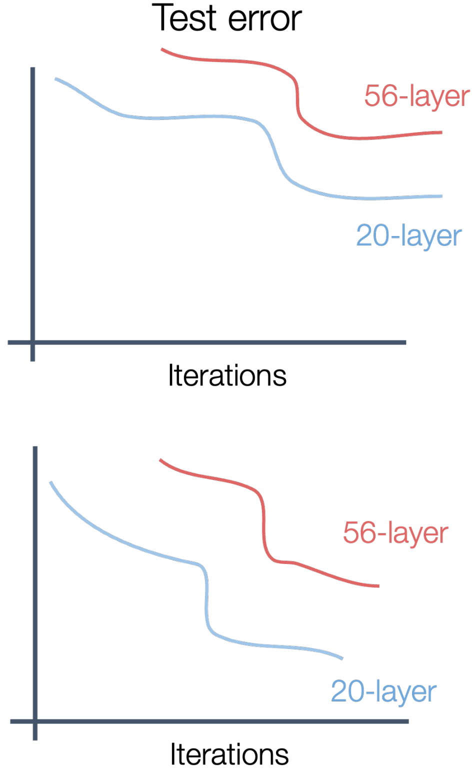
Not actually used that much in practice

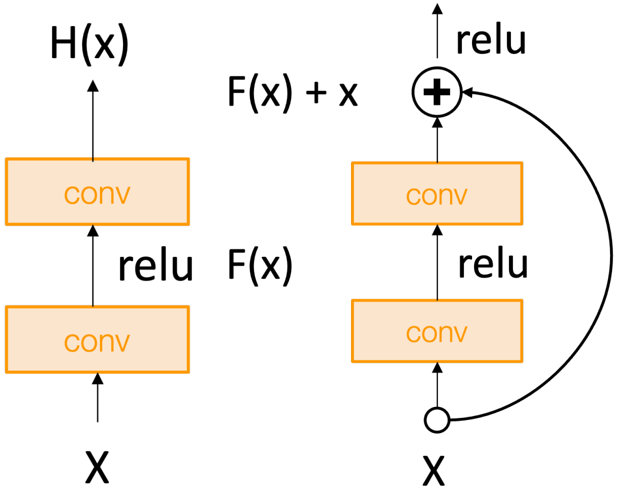
He et al, "Identity mappings in deep residual networks", ECCV 2016

Slide from Justin Johnson

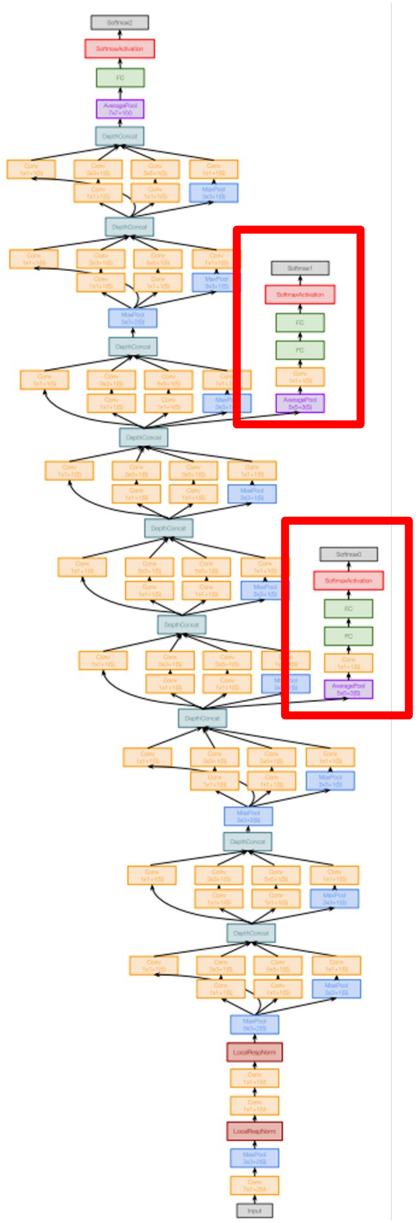
Recap



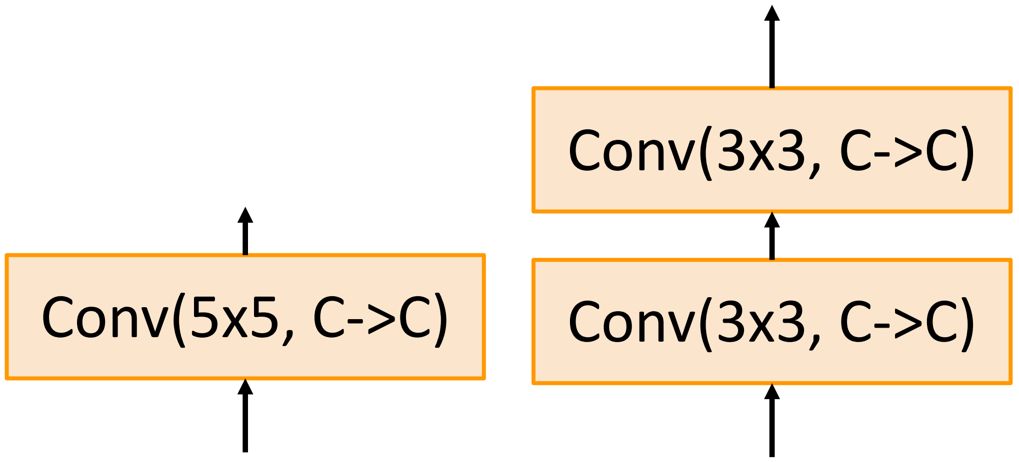




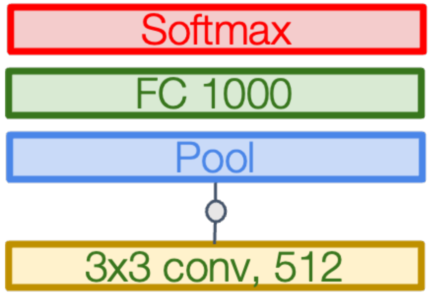
Residual Connections



Auxiliary Supervision



Multiple smaller convs v/s a single large conv

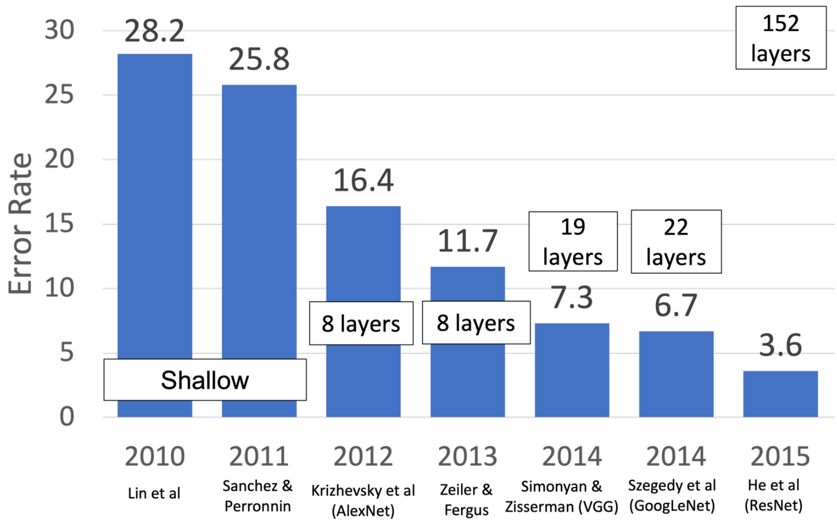


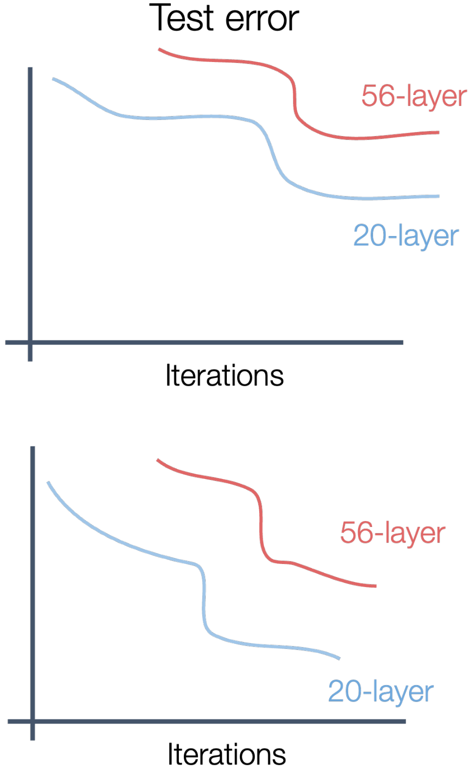
Global average pooling vs fully connected heads

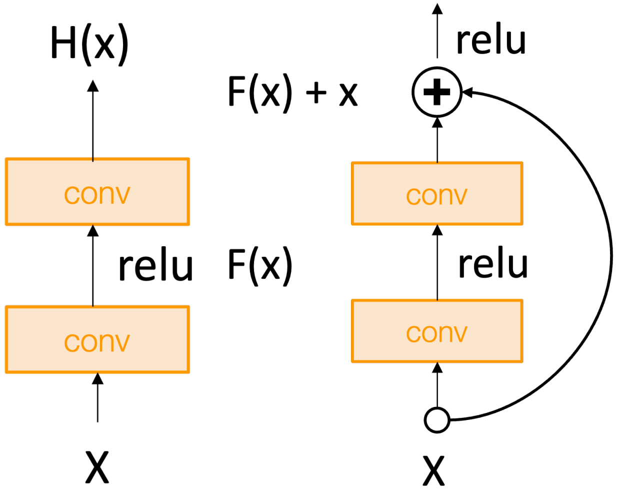
Today’s plan

- Finish talking about other architectures since AlexNet
- Designing neural networks for vision problems
  - Segmentation
  - …

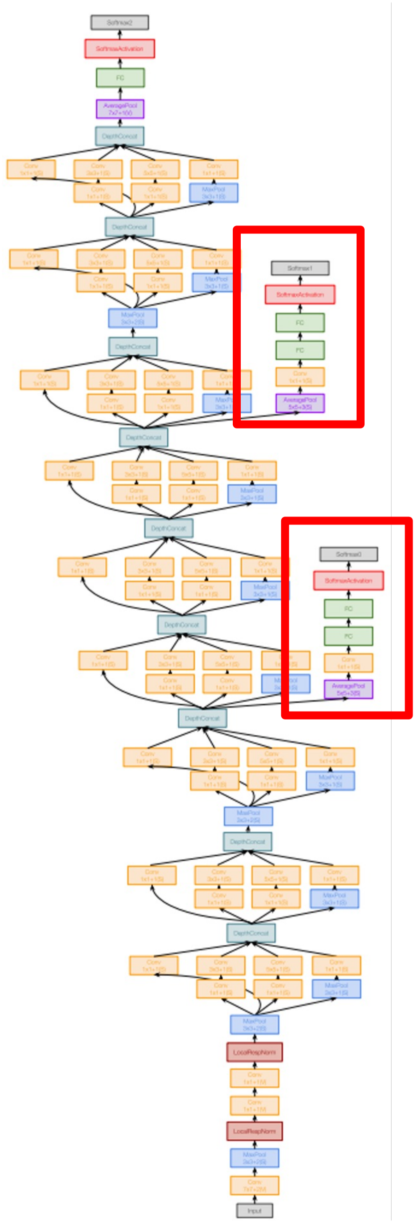
Recap

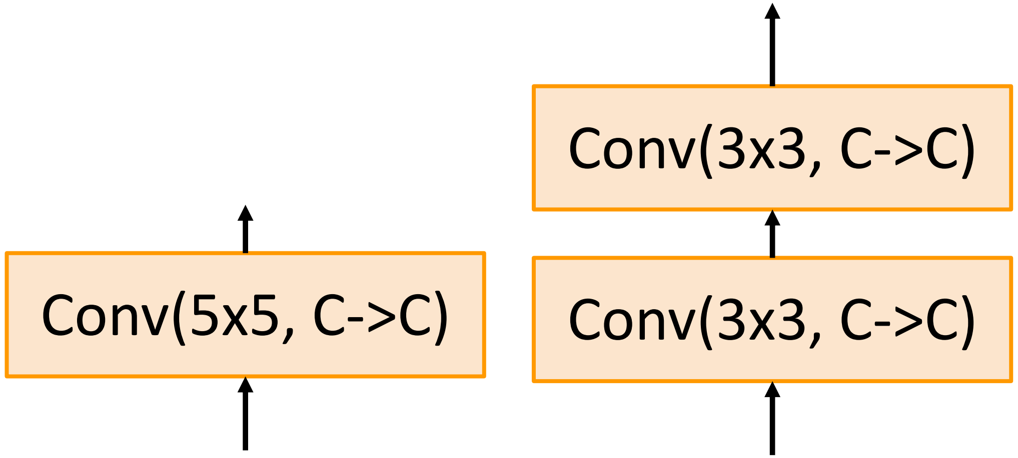




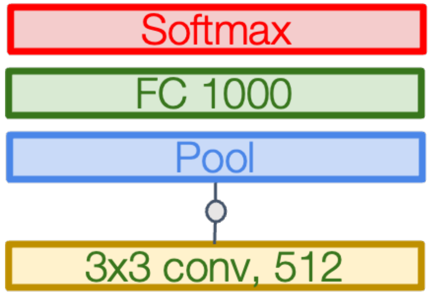


Residual Connections





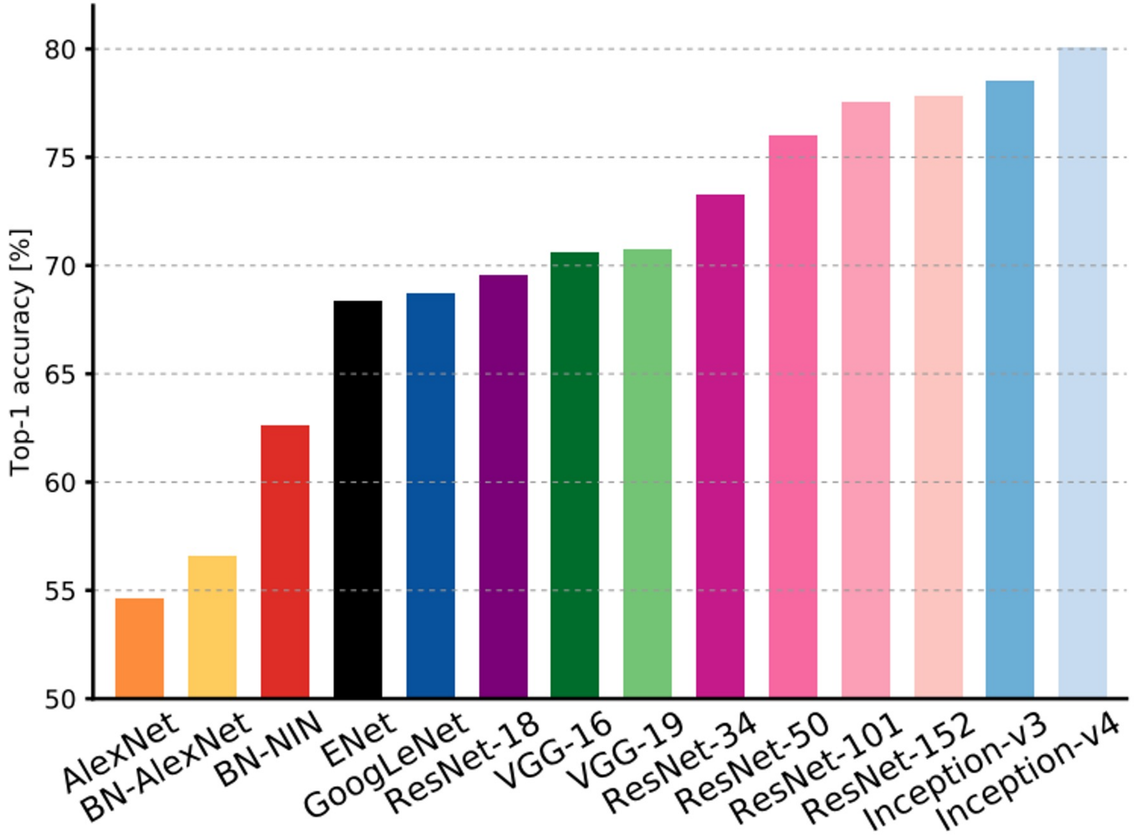
Multiple smaller convs v/s a single large conv

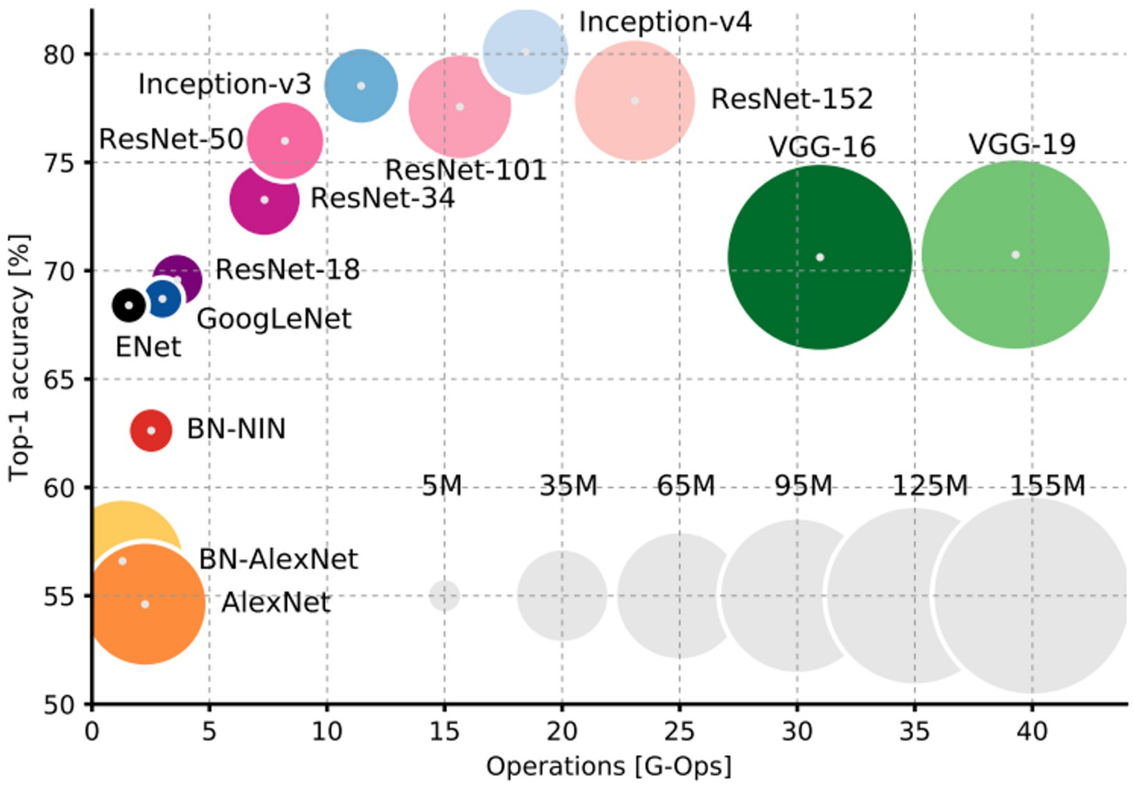


Global average pooling vs fully connected heads

Auxiliary Supervision

Comparing Complexity



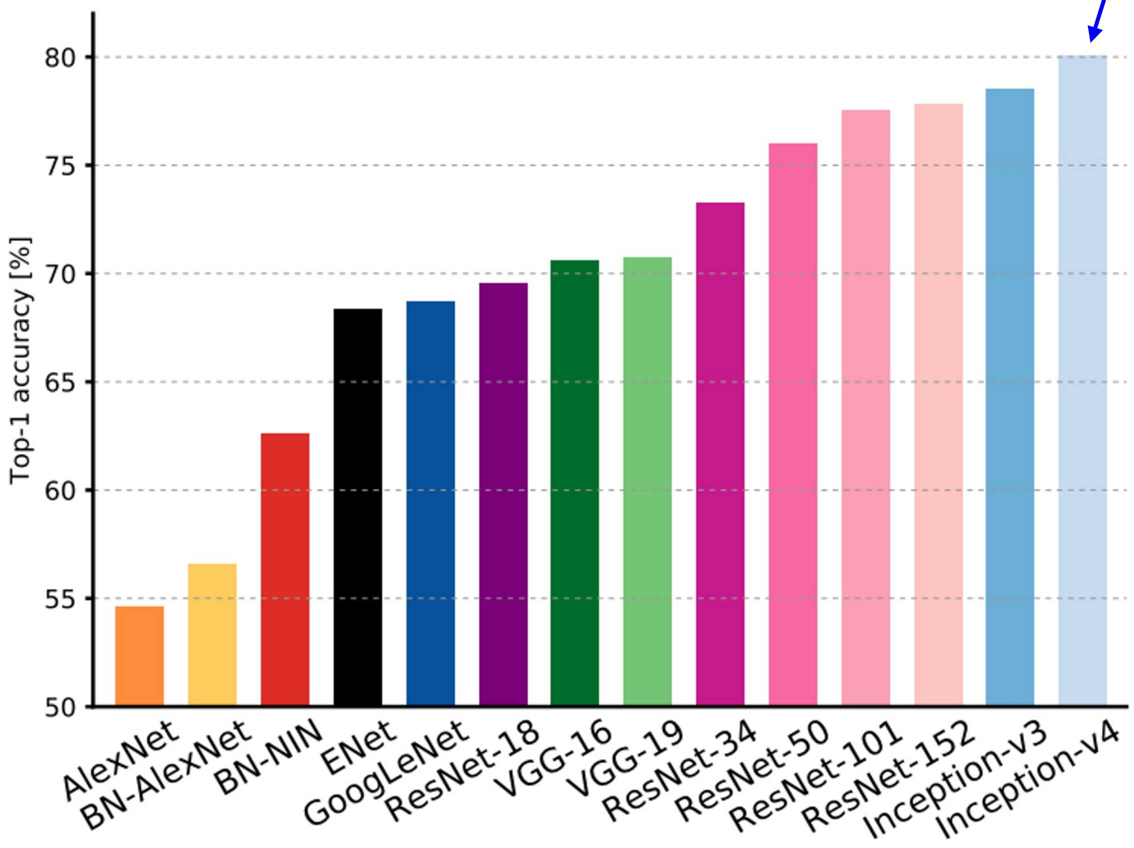


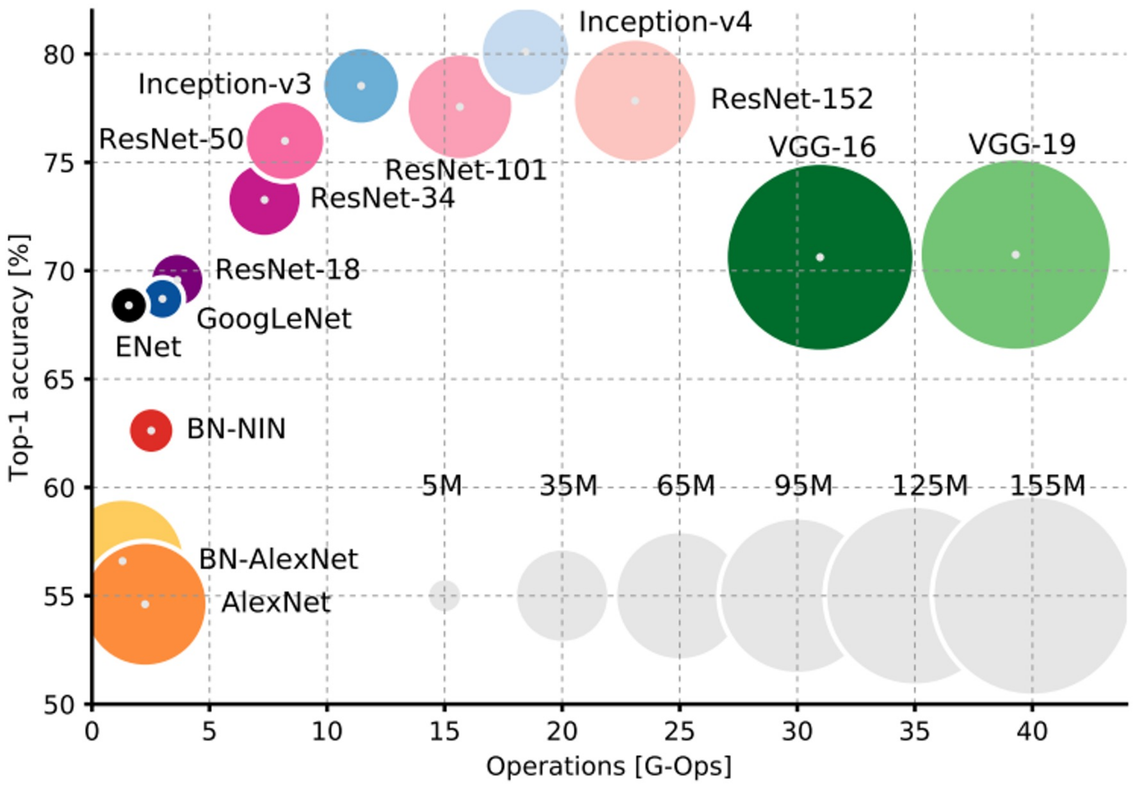
Canziani et al, "An analysis of deep neural network models for practical applications", 2017

Slide from Justin Johnson

Comparing Complexity

Inception-v4: Resnet + Inception!



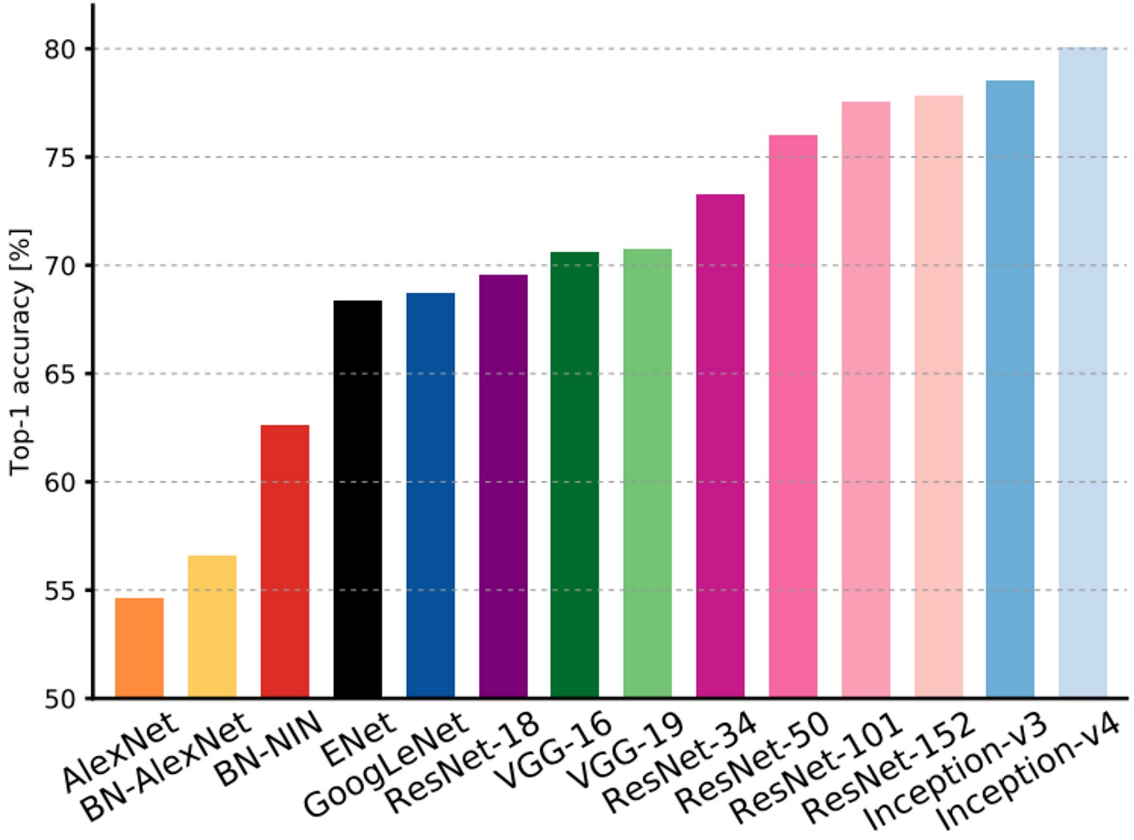


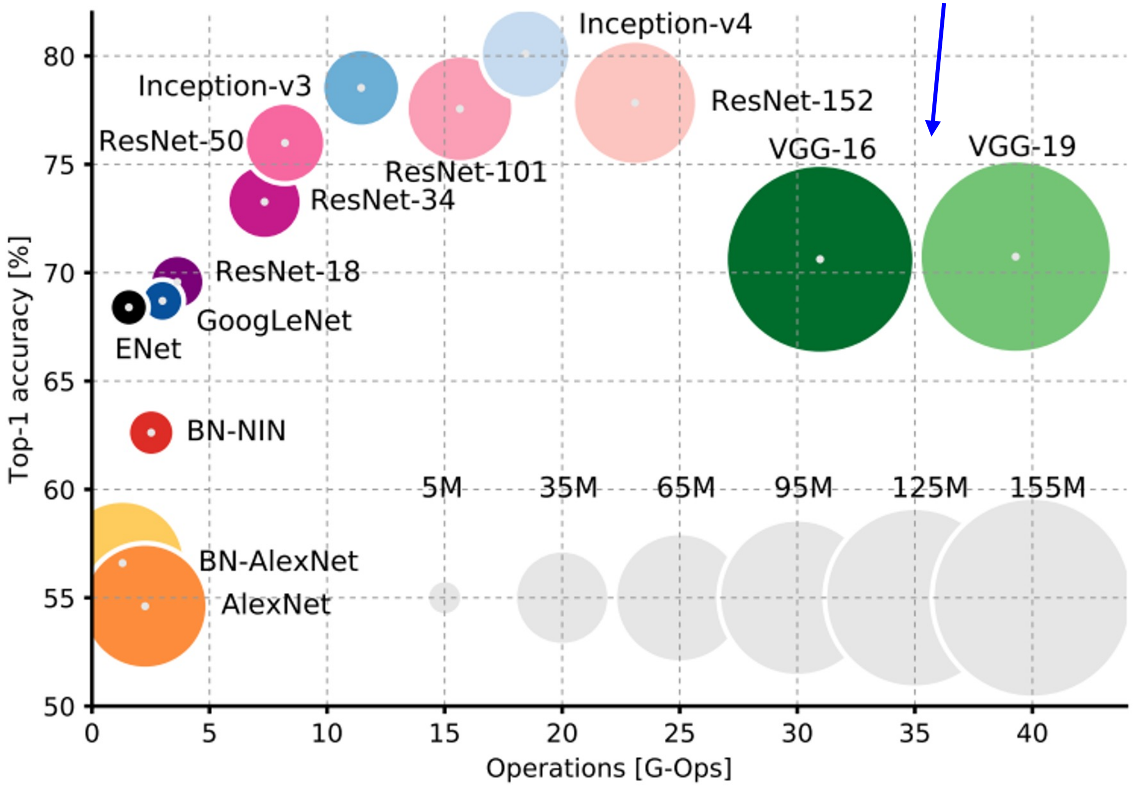
Canziani et al, “An analysis of deep neural network models for practical applications”, 2017

Slide from Justin Johnson

Comparing Complexity

VGG: Highest memory, most operations



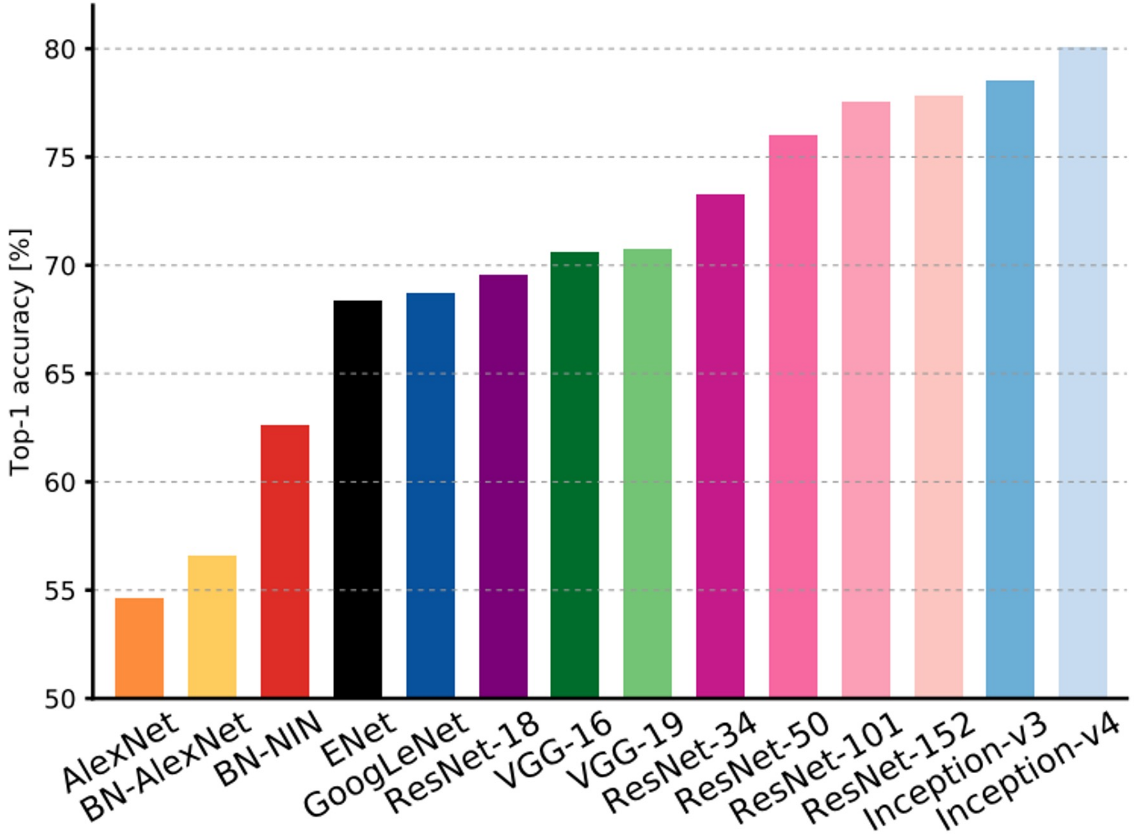


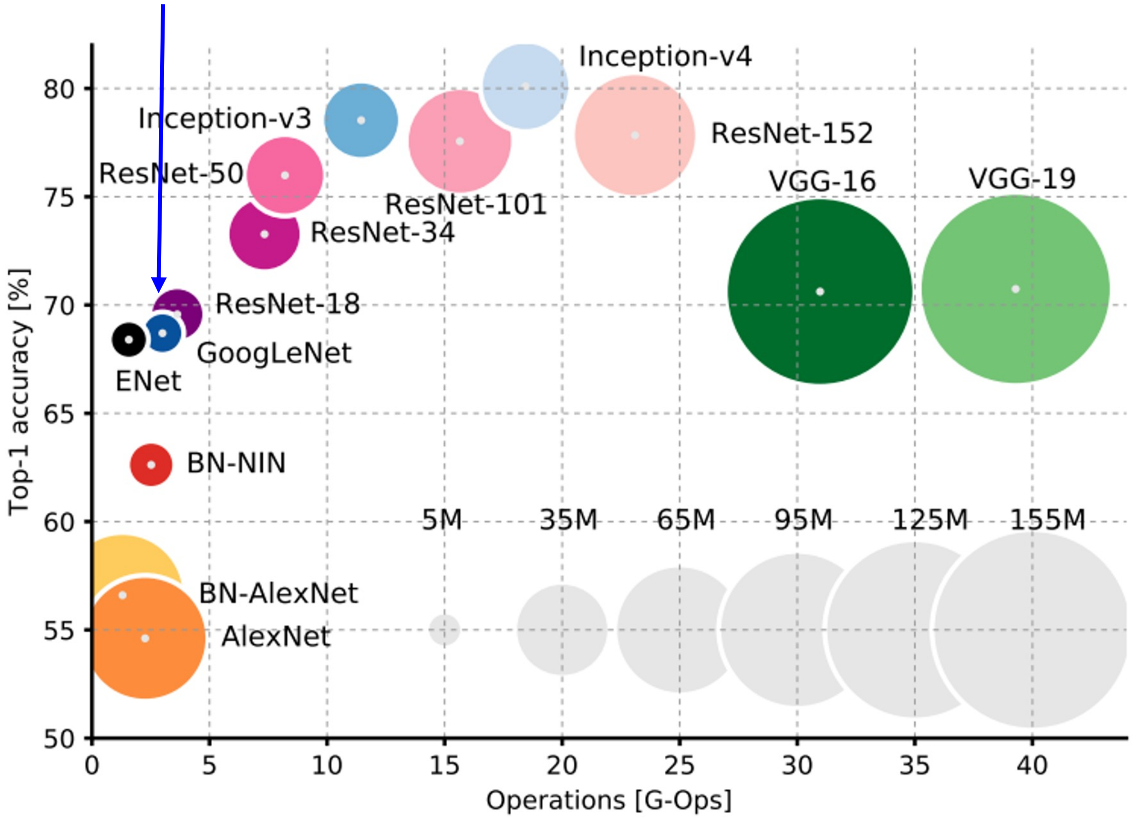
Canziani et al, "An analysis of deep neural network models for practical applications", 2017

Slide from Justin Johnson

Comparing Complexity

GoogLeNet:
Very efficient!



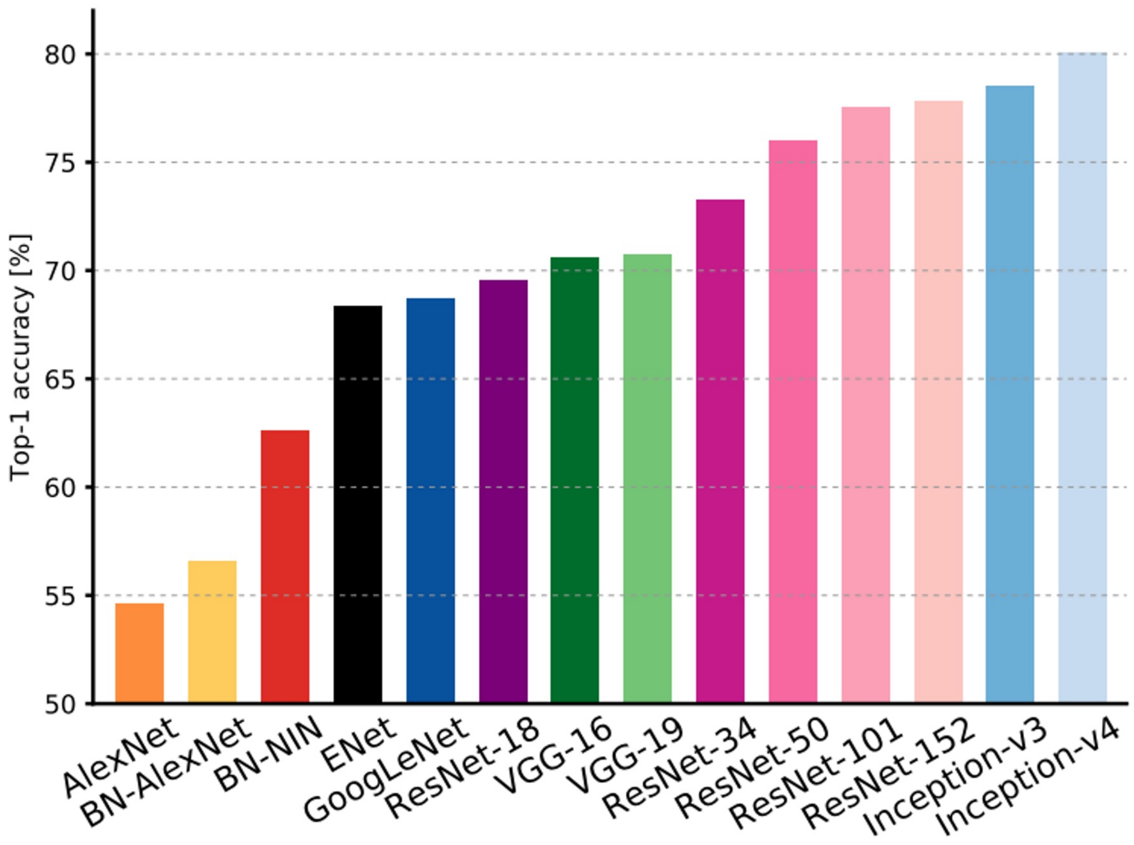


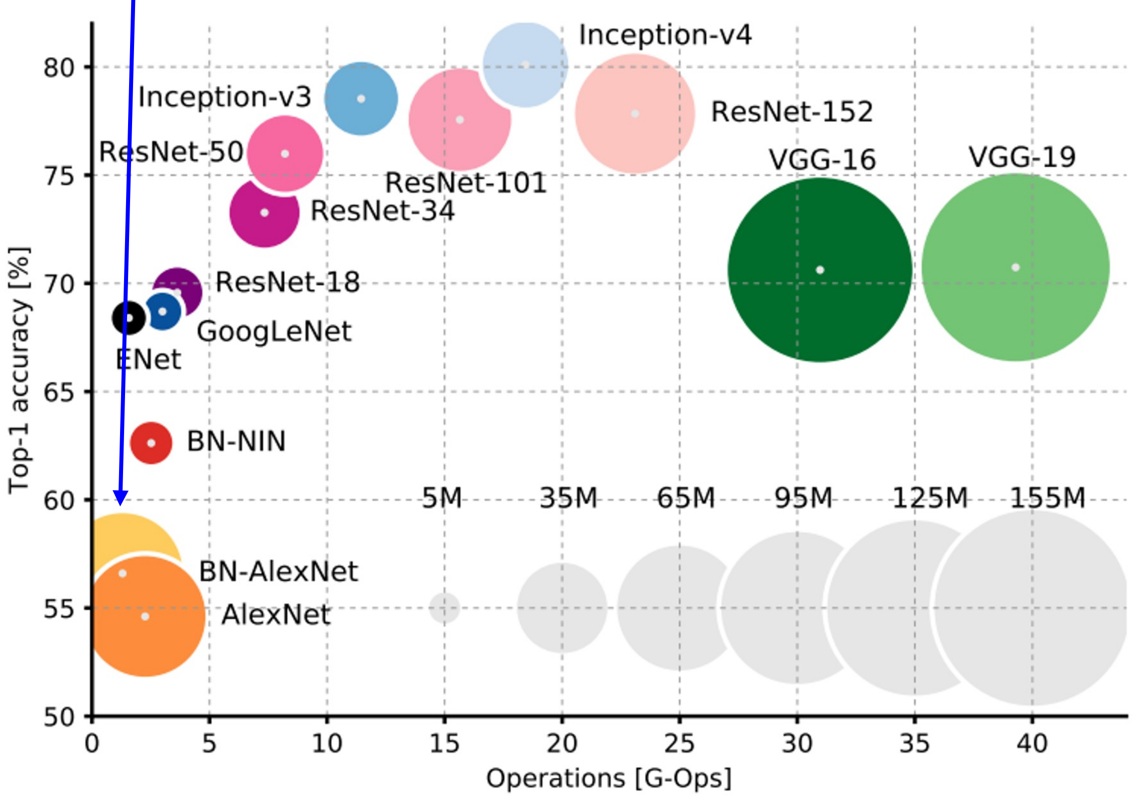
Canziani et al, "An analysis of deep neural network models for practical applications", 2017

Slide from Justin Johnson

Comparing Complexity

AlexNet: Low compute, lots of parameters



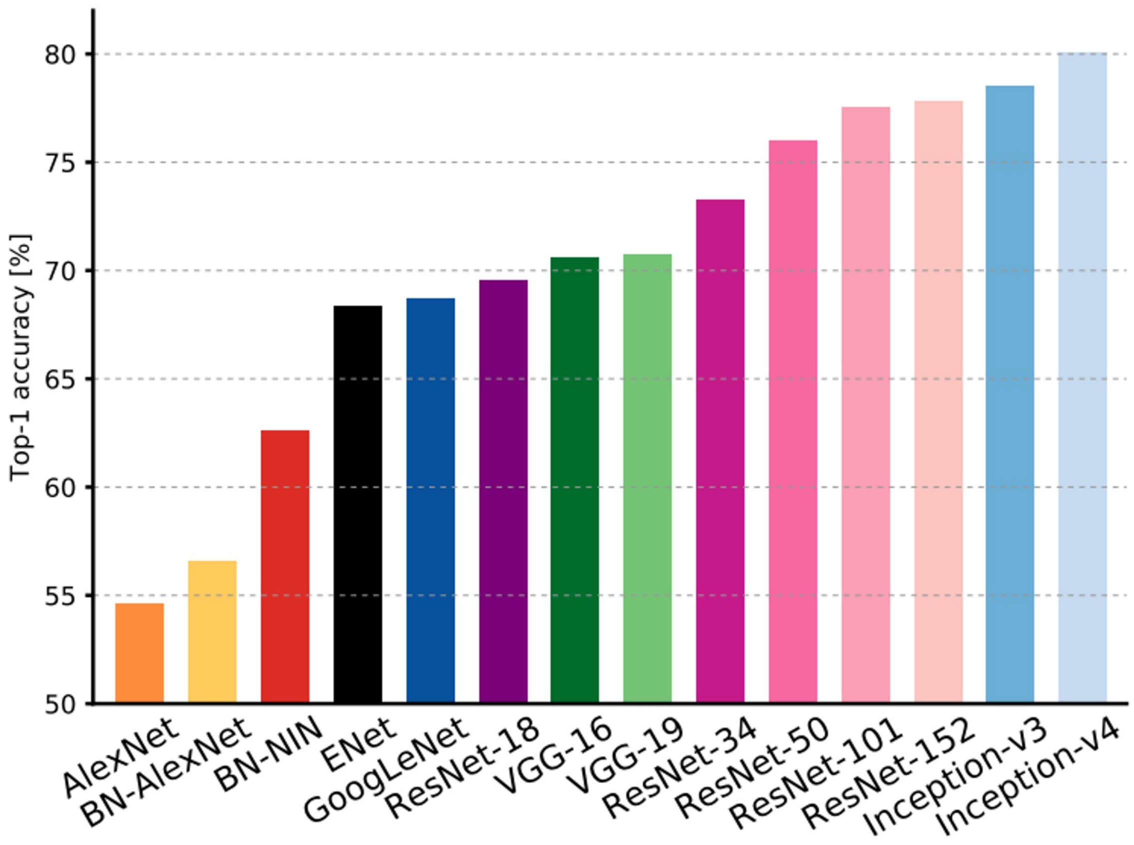


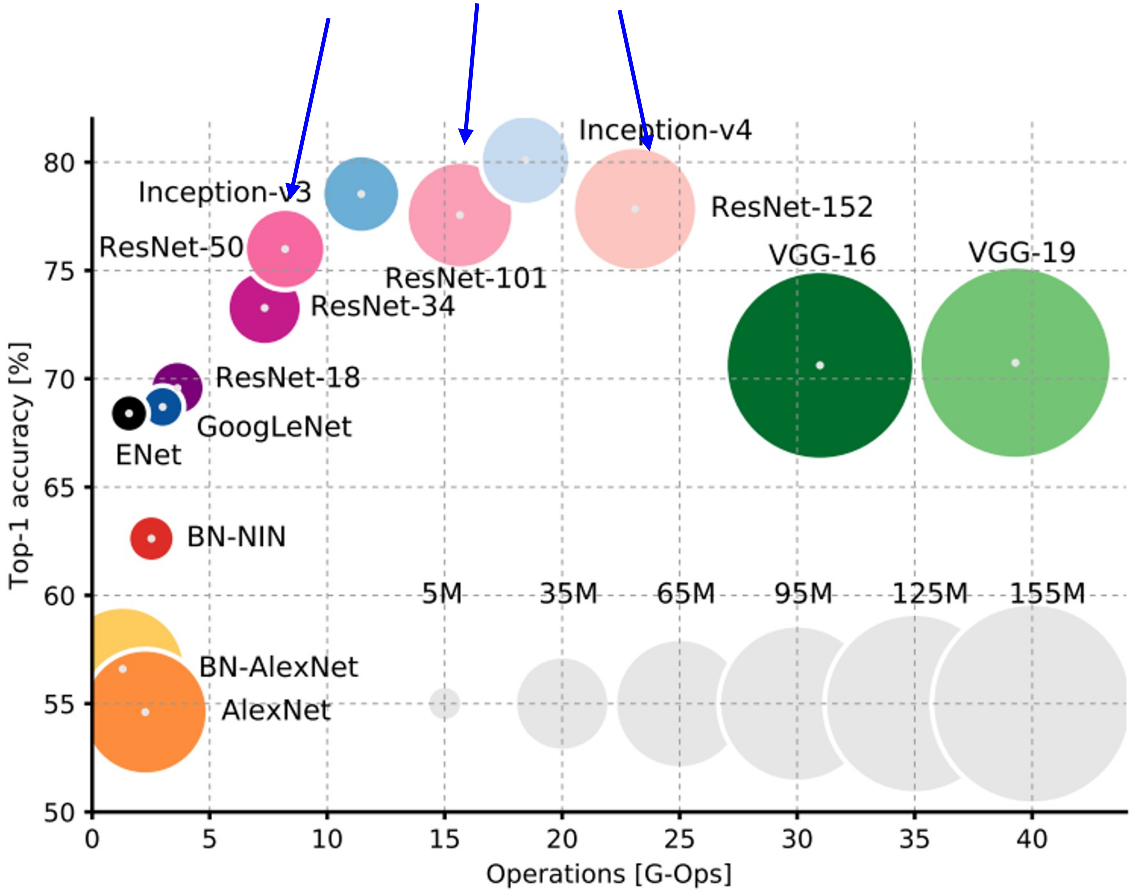
Canziani et al, "An analysis of deep neural network models for practical applications", 2017

Slide from Justin Johnson

Comparing Complexity

ResNet: Simple design, moderate efficiency, high accuracy

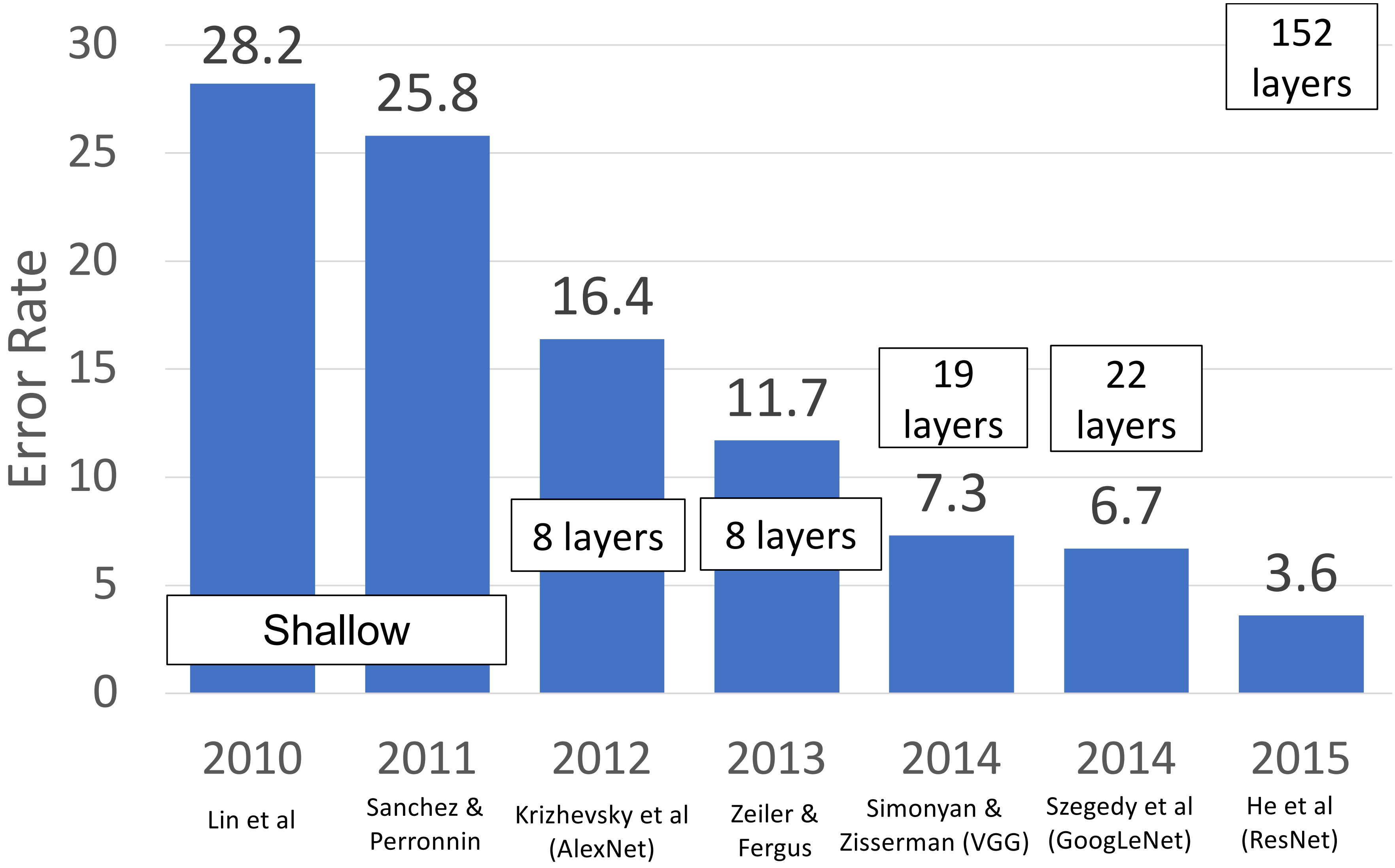




Canziani et al, “An analysis of deep neural network models for practical applications”, 2017

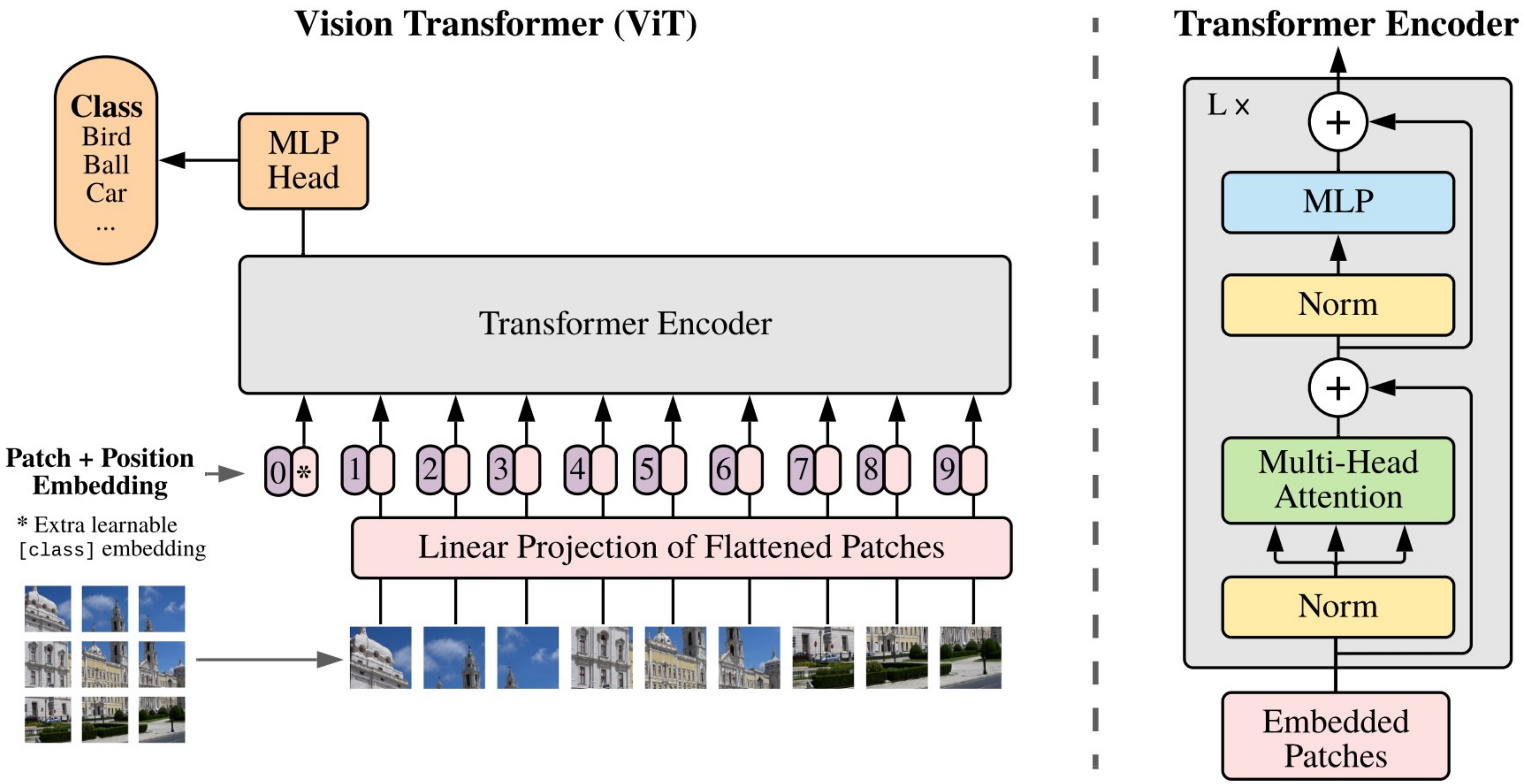
Slide from Justin Johnson

ImageNet Classification Challenge



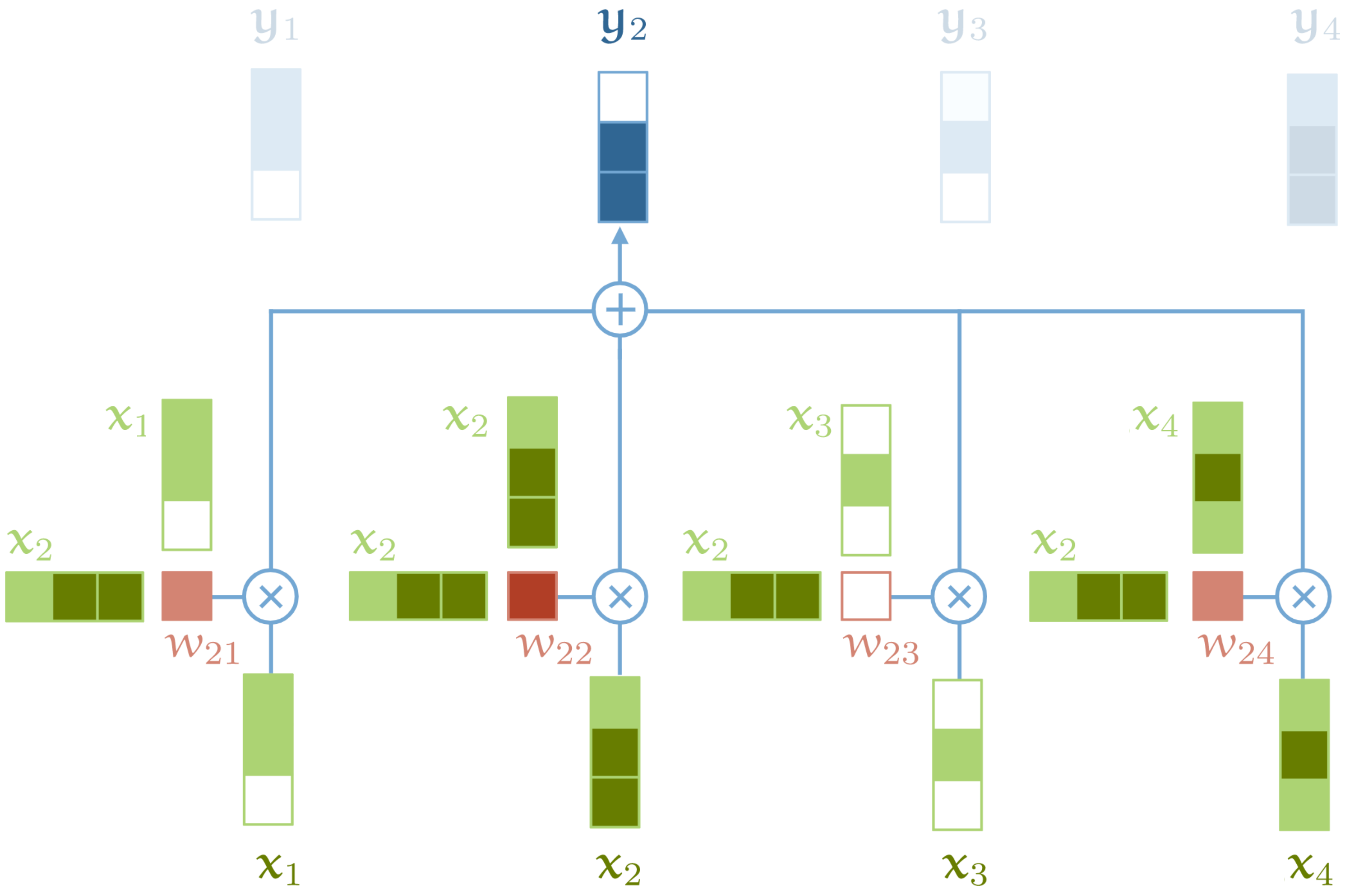
Slide from Justin Johnson

Attention (Vision Transformers)



A. Dosovitskiy et al., [An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale](#).

Attention



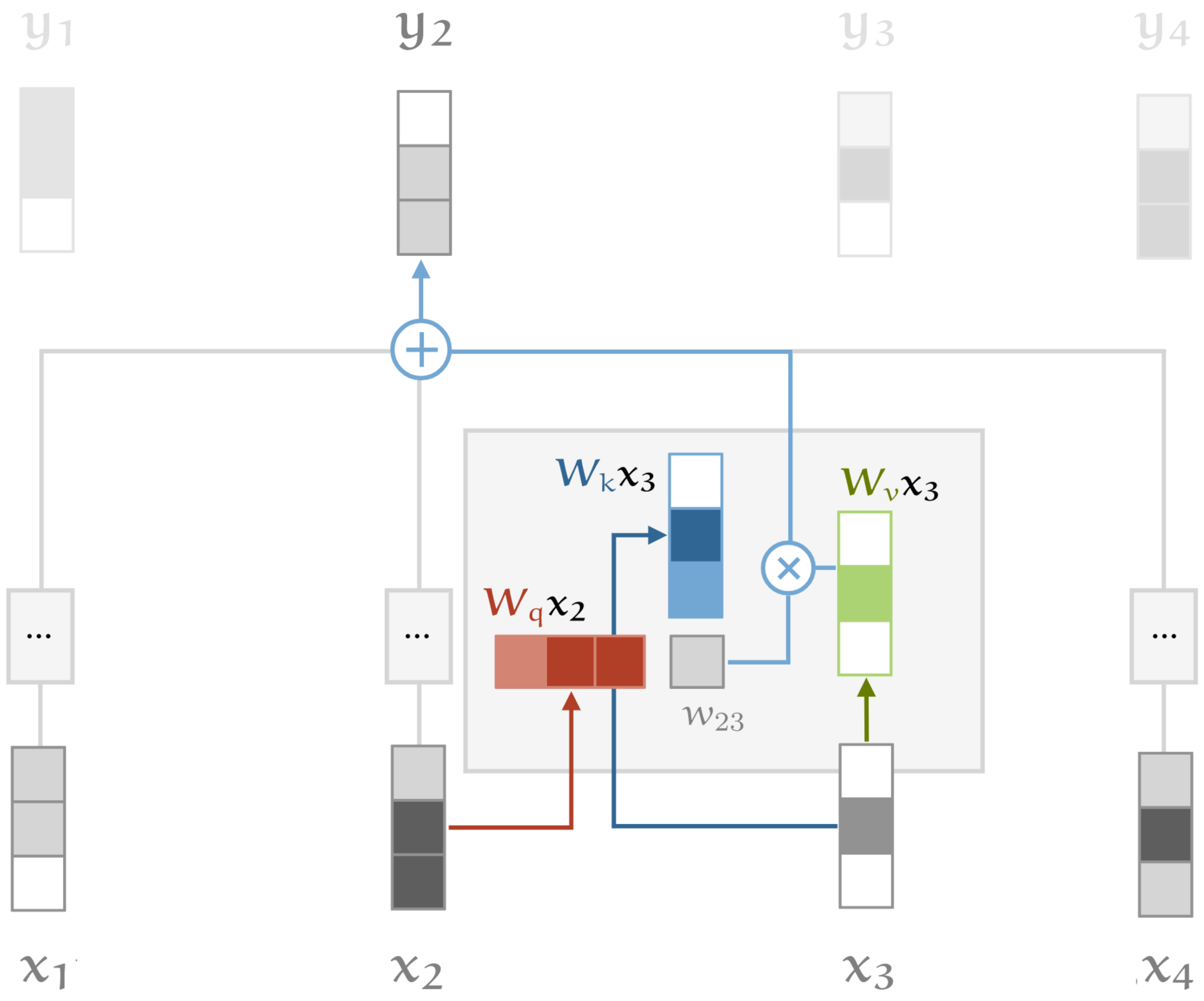
$$y_i = \sum_j w_{ij} x_{ij}$$

$$w_{ij} = softmax_j(x_i^T x_j / \sqrt{d_k})$$

$$w_{ij} = \frac{e^{x_i^T x_j}}{\sum_j e^{x_i^T x_j}}$$

Source: http://peterbloem.nl/blog/transformers See also: [Attention is all you need]()

Attention (with key, query and value)



$$y_i = \sum_j w_{ij} \boldsymbol{W}_v \boldsymbol{x}_{ij}$$

$$w_{ij} = softmax_j((\boldsymbol{W}_q \boldsymbol{x}_i)^T \boldsymbol{W}_k \boldsymbol{x}_j / \sqrt{d_k})$$

More in a later class

Source: http://peterbloem.nl/blog/transformers See also: [Attention is all you need]()

Attention (Vision Transformers)

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21K (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Table 2: Comparison with state of the art on popular image classification benchmarks. We report mean and standard deviation of the accuracies, averaged over three fine-tuning runs. Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train. ViT pre-trained on the smaller public ImageNet-21k dataset performs well too. *Slightly improved 88.5% result reported in Touvron et al. (2020).

A. Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Scale Better with More Data