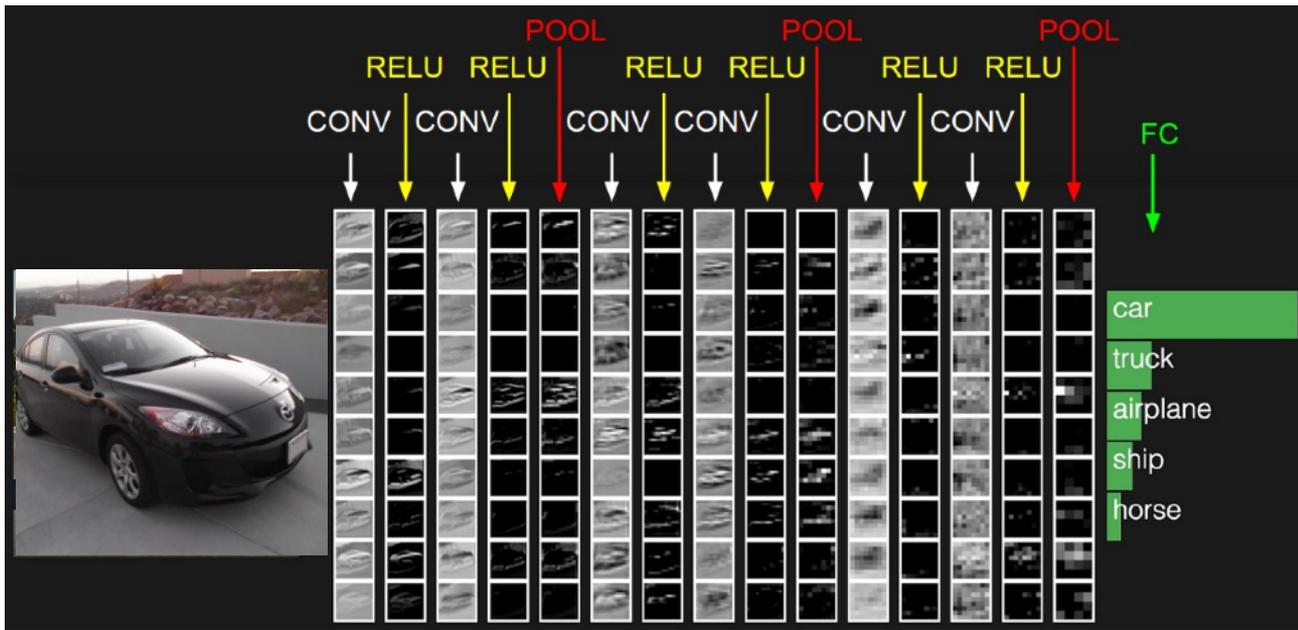# Lecture 11:
CNN Architectures …
Project overview

Agenda

Wrap up the discussion of CNN architectures

Project overview and ideas (led by Rangel and Max)

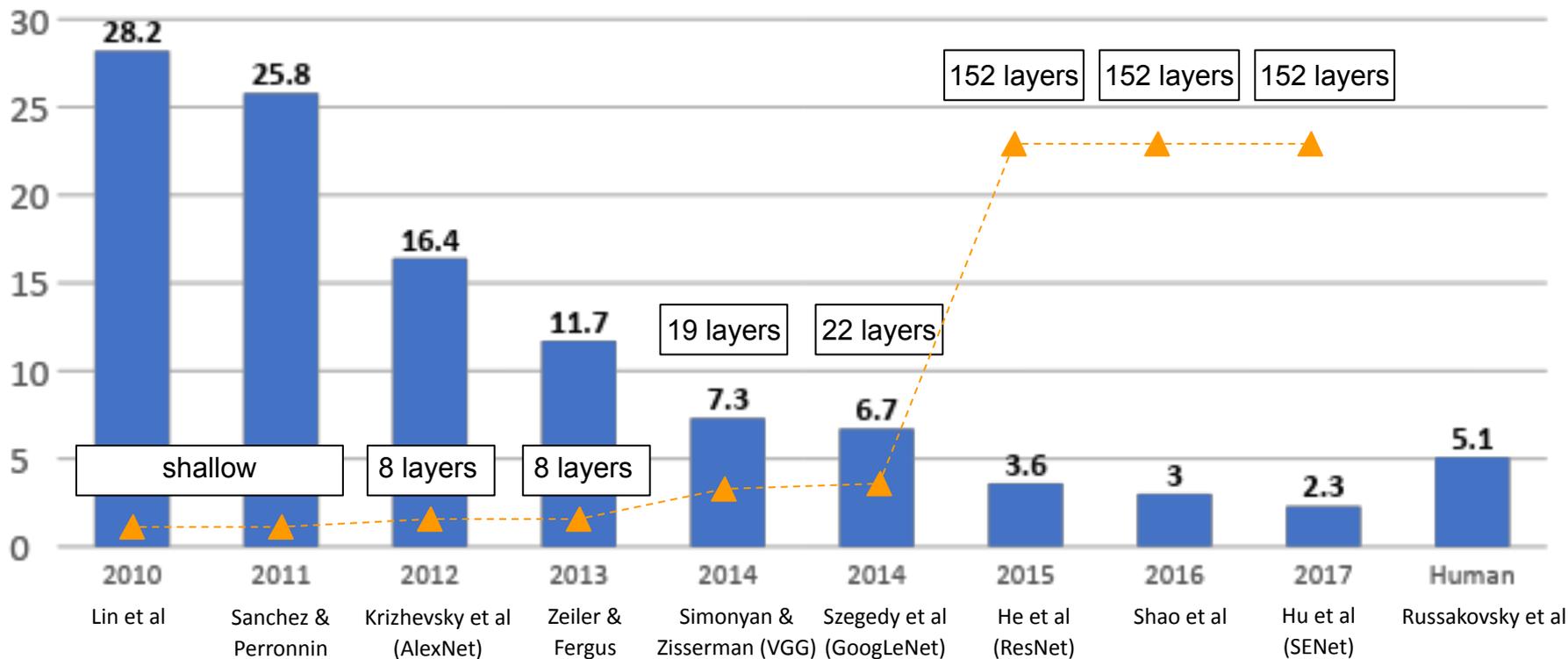# Recap: Convolutional Neural Networks

Recap: CNN Architectures

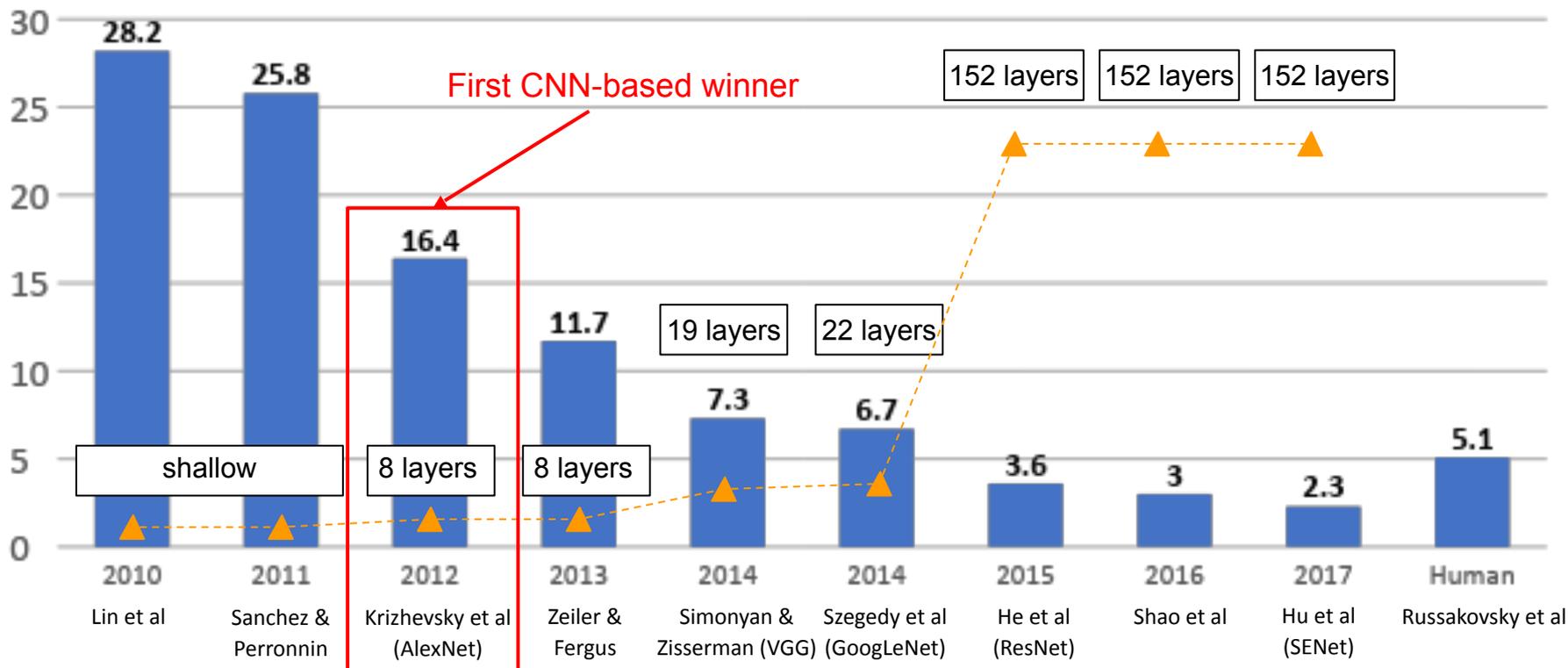## Case Studies

- AlexNet
- VGG
- GoogLeNet
- ResNet

Also....

- SENet
- Wide ResNet
- ResNeXT

- DenseNet
- MobileNets
- NASNet
- EfficientNet

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
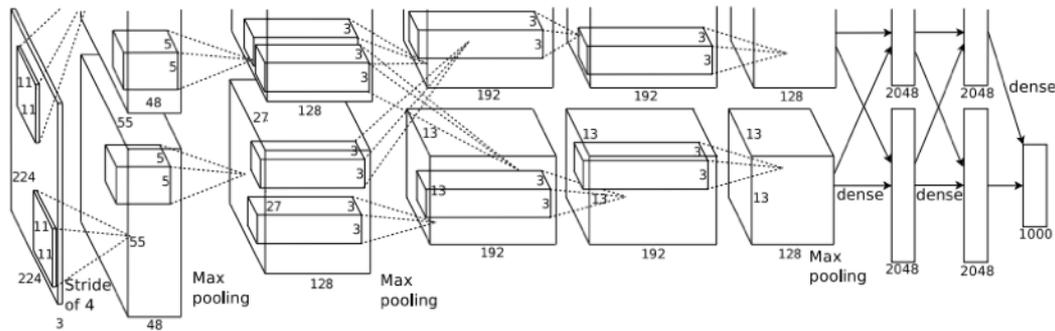[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
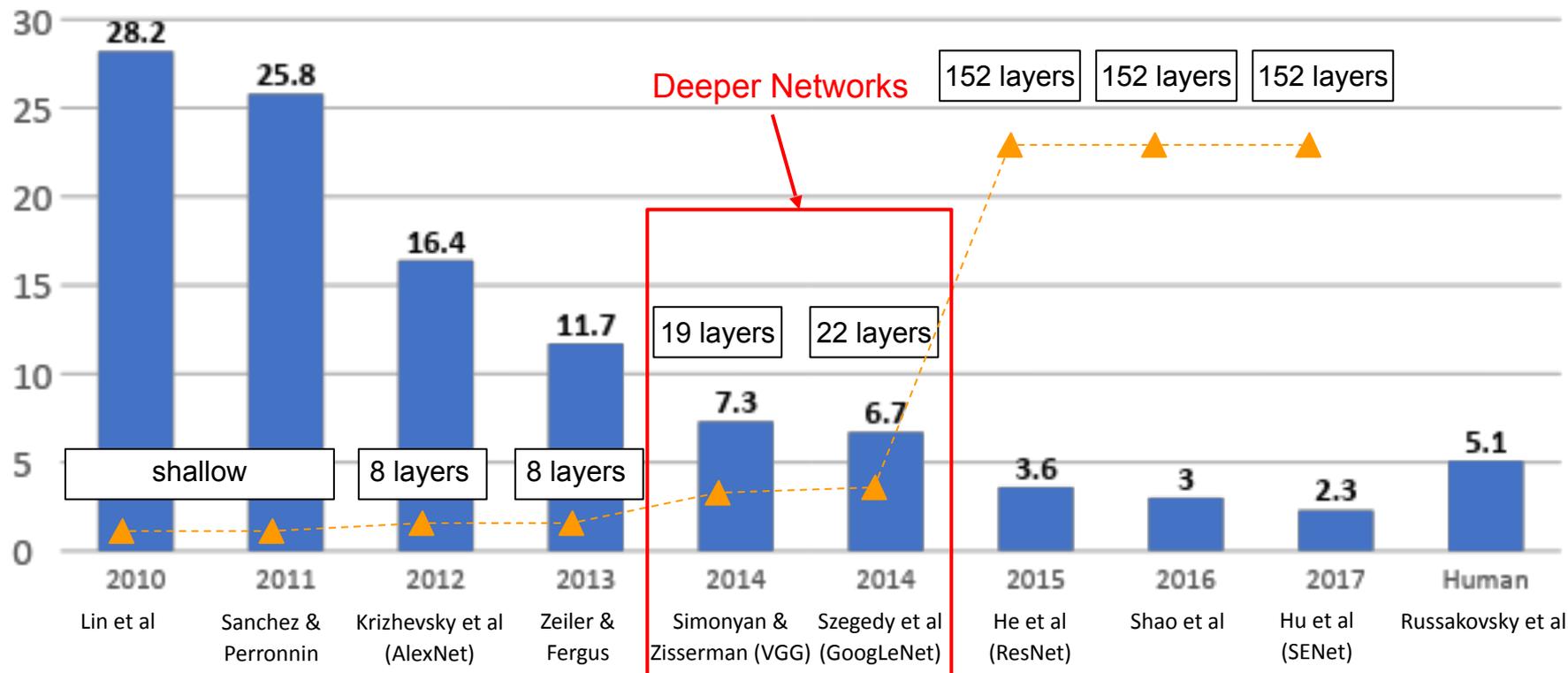[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used LRN layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

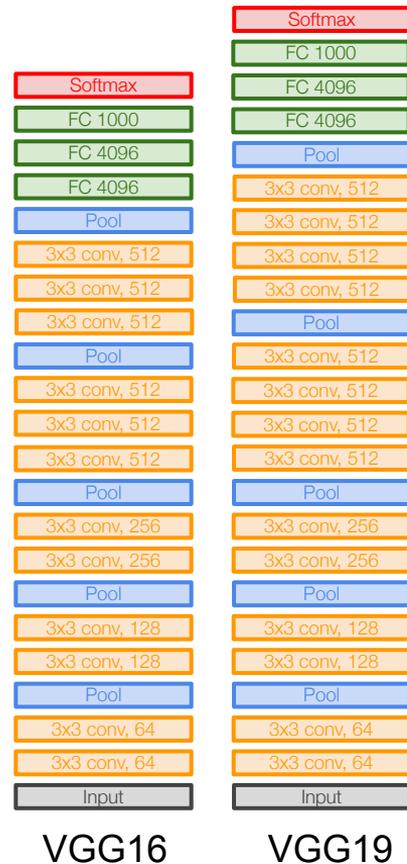# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14
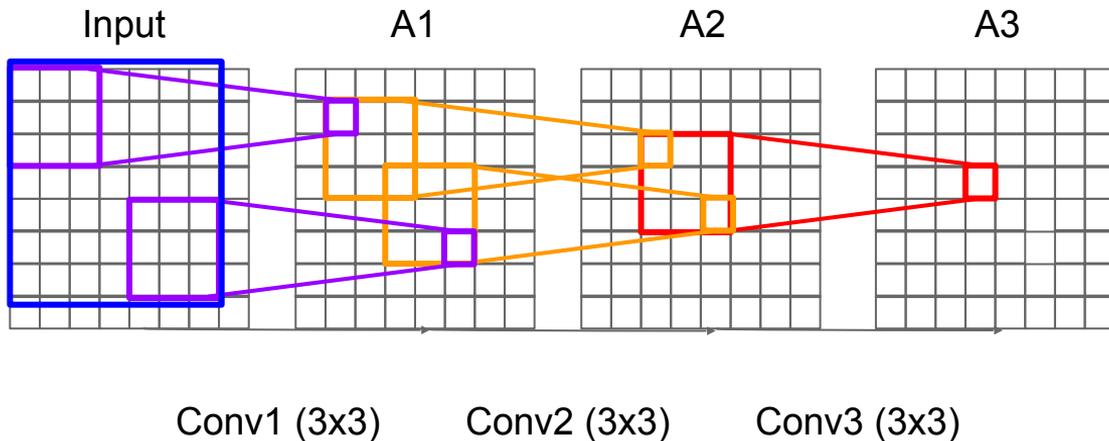

AlexNet

VGG16

VGG19

Case Study: VGGNet

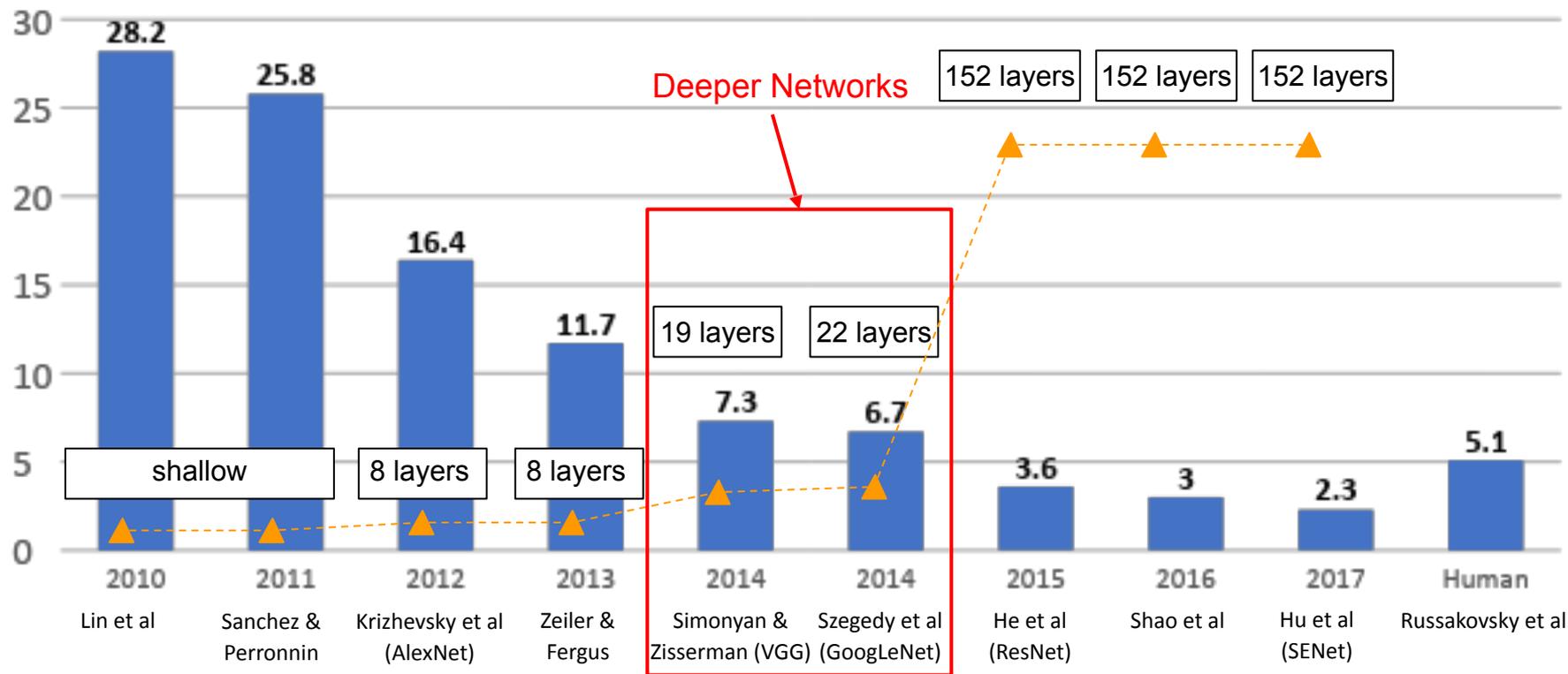*[Simonyan and Zisserman, 2014]*

Q: What is the effective receptive field
of three 3x3 conv (stride 1) layers?

Input   A1   A2   A3

Conv1 (3x3)   Conv2 (3x3)   Conv3 (3x3)

VGG16   VGG19

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Details:
- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012
- No Local Response Normalisation (LRN)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks

AlexNet   VGG16   VGG19

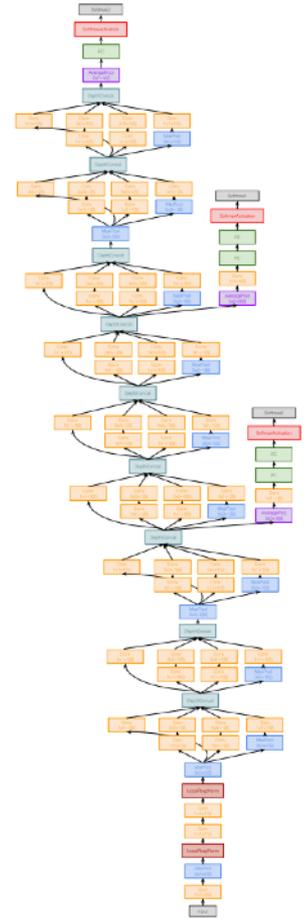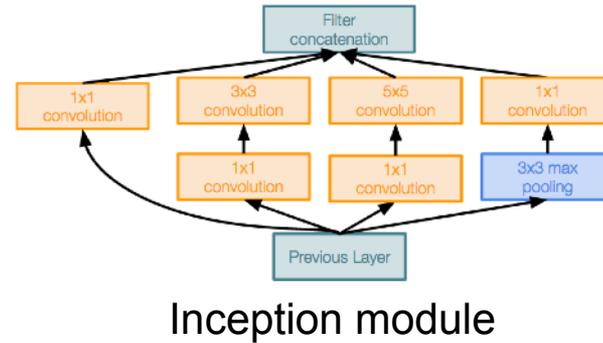# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

Case Study: GoogLeNet

[Szegedy et al., 2014]

**Deeper networks, with computational efficiency**

- ILSVRC'14 classification winner (6.7% top 5 error)
- 22 layers
- Only 5 million parameters!
  12x less than AlexNet
  27x less than VGG-16
- Efficient "Inception" module
- No FC layers

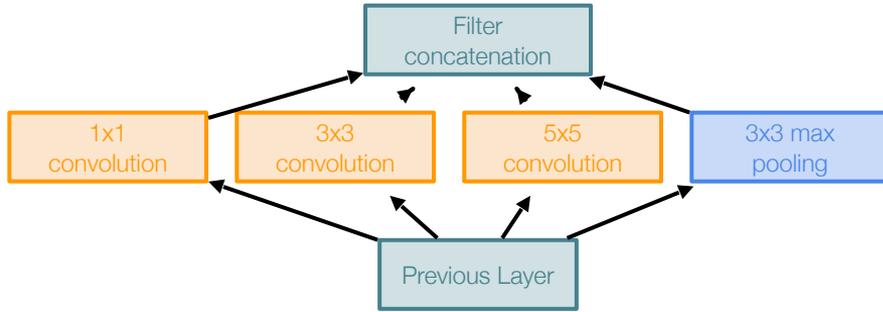Inception module

Case Study: GoogLeNet

[Szegedy et al., 2014]

"Inception module": design a good local network topology (network within a network) and then stack these modules on top of each other
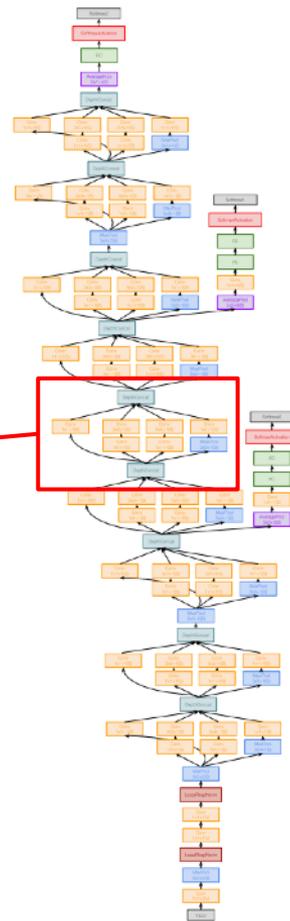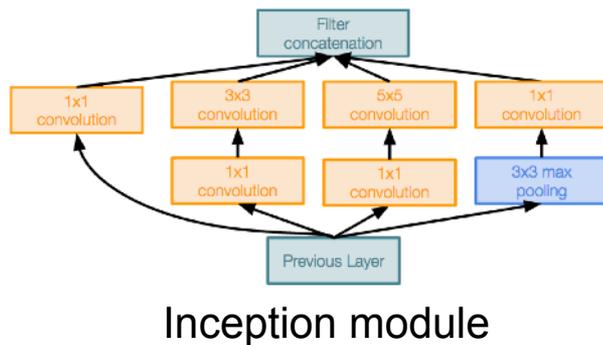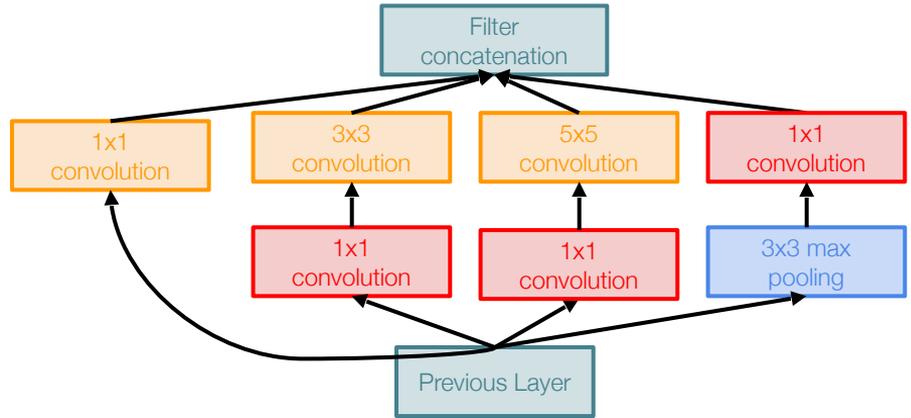

Inception module

Case Study: GoogLeNet

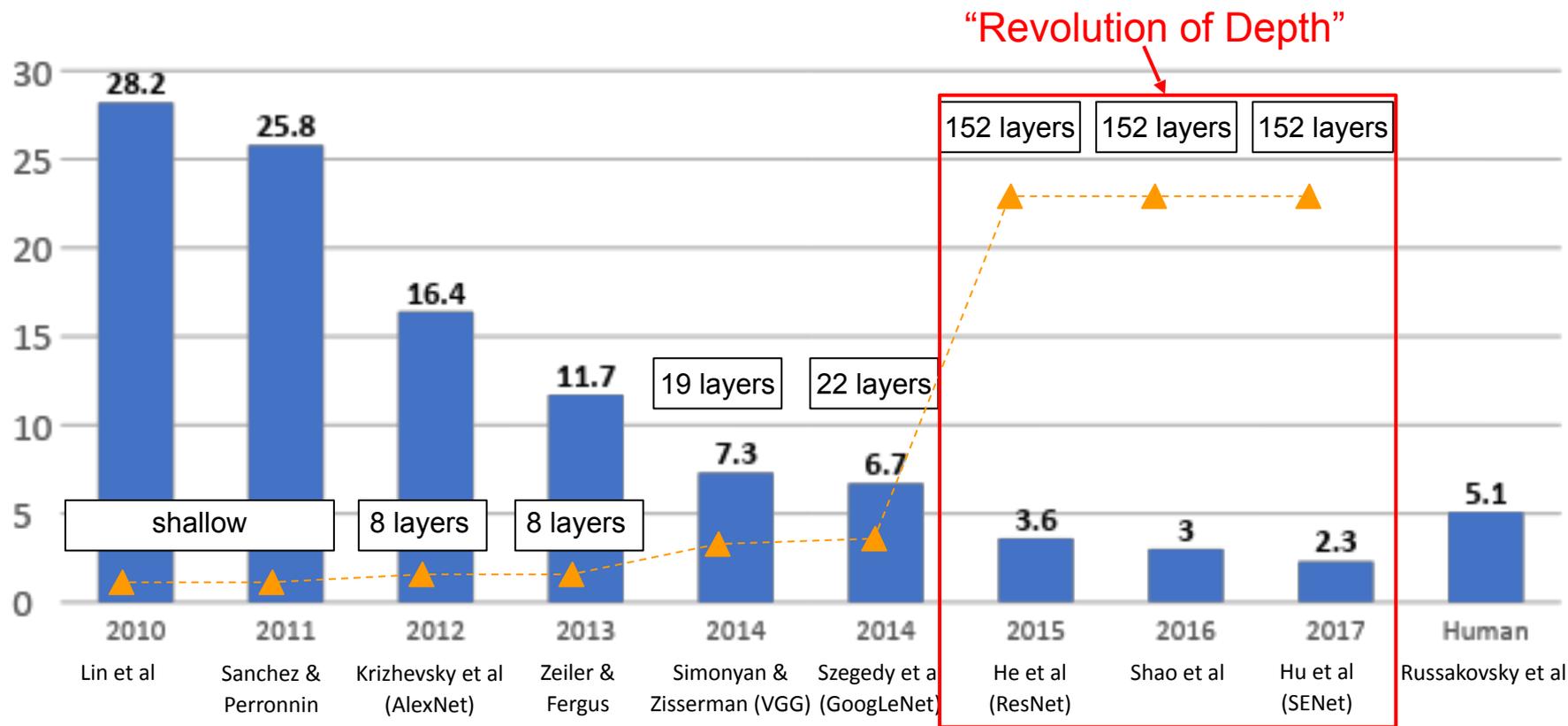*[Szegedy et al., 2014]*

Naive Inception module

1x1 conv "bottleneck" layers

Inception module with dimension reduction

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

"Revolution of Depth"

Case Study: ResNet

[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!
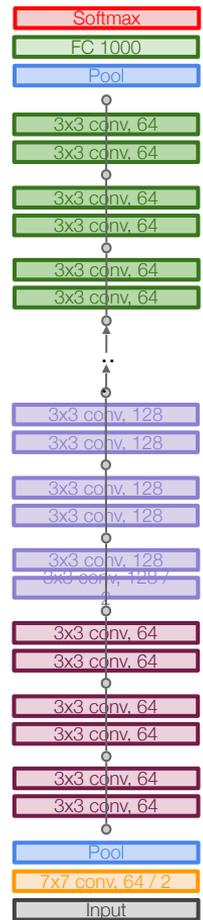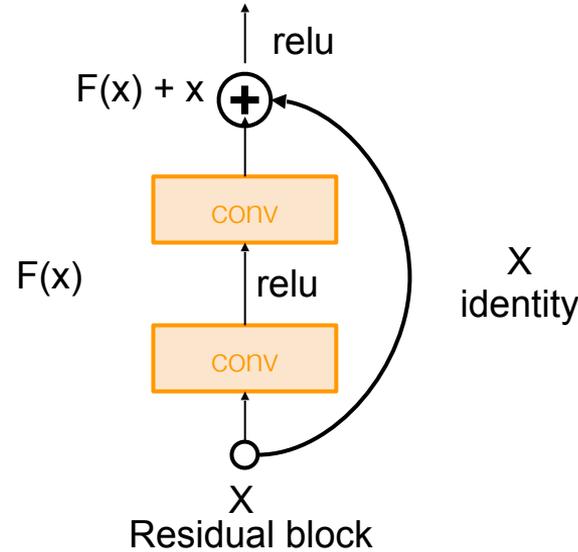
F(x) + x

relu

conv

F(x)   relu

conv

X
identity

X
Residual block

Case Study: ResNet

[He et al., 2015]

What happens when we continue stacking deeper layers on a "plain" convolutional neural network?

Case Study: ResNet

[He et al., 2015]

What happens when we continue stacking deeper layers on a "plain" convolutional neural network?
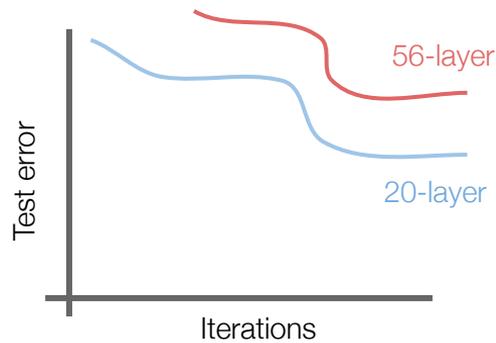
Case Study: ResNet

[He et al., 2015]

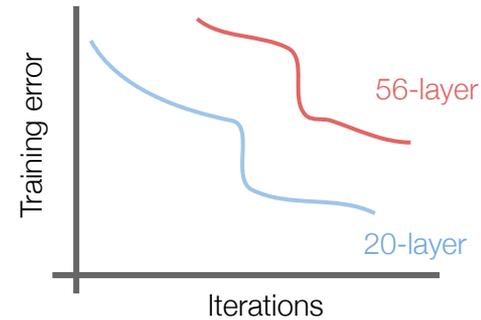What happens when we continue stacking deeper layers on a "plain" convolutional neural network?
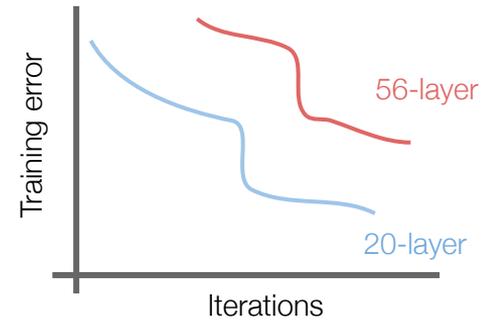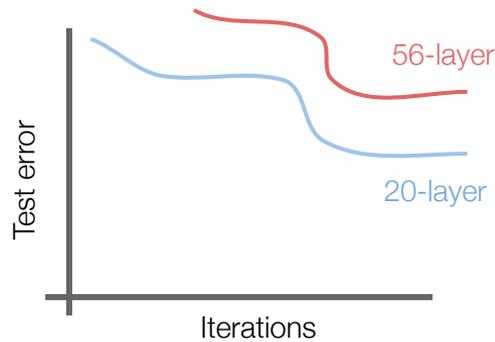
56-layer model performs worse on both test and training error
-> The deeper model performs worse, but it's not caused by overfitting!

Case Study: ResNet

[He et al., 2015]

Fact: Deep models have more representation power (more parameters) than shallower models.

Hypothesis: the problem is an *optimization* problem, **deeper models are harder to optimize**

Case Study: ResNet

[He et al., 2015]

Fact: Deep models have more representation power (more parameters) than shallower models.

Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize

What should the deeper model learn to be at least as good as the shallower model?

A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.
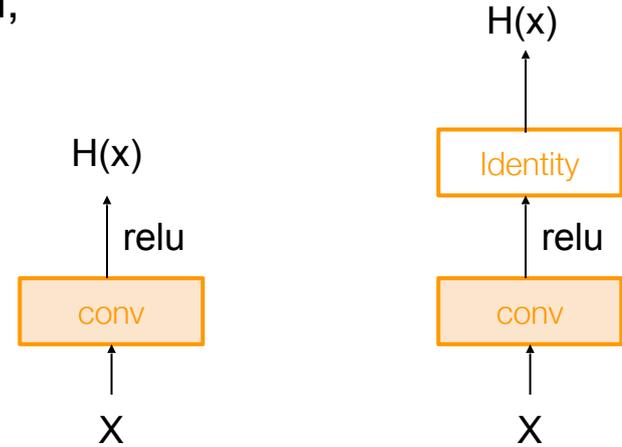
H(x)

↑

relu

| conv |

↑

X

H(x)

↑

| Identity |

↑

relu

| conv |

↑

X

Case Study: ResNet

[He et al., 2015]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

H(x)

conv

relu

conv

X
"Plain" layers
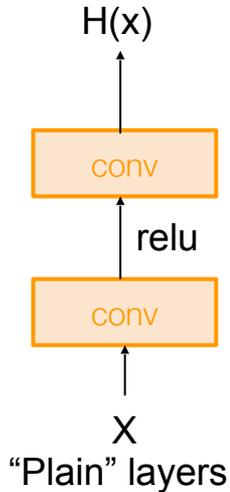
Case Study: ResNet

[He et al., 2015]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

Identity mapping:
H(x) = x if F(x) = 0

Case Study: ResNet

[He et al., 2015]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

$H(x) = F(x) + x$

H(x)

conv

relu

conv

X

"Plain" layers

$H(x) = F(x) + x$

relu

conv

F(x)

relu

conv

X
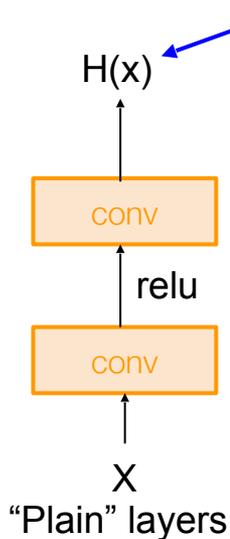
Residual block
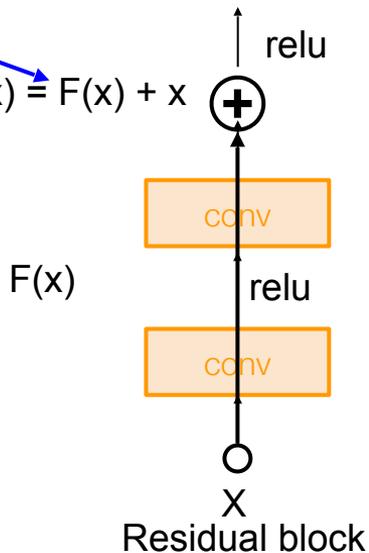
X identity

Identity mapping: H(x) = x if F(x) = 0

Use layers to fit **residual** F(x) = H(x) - x instead of H(x) directly
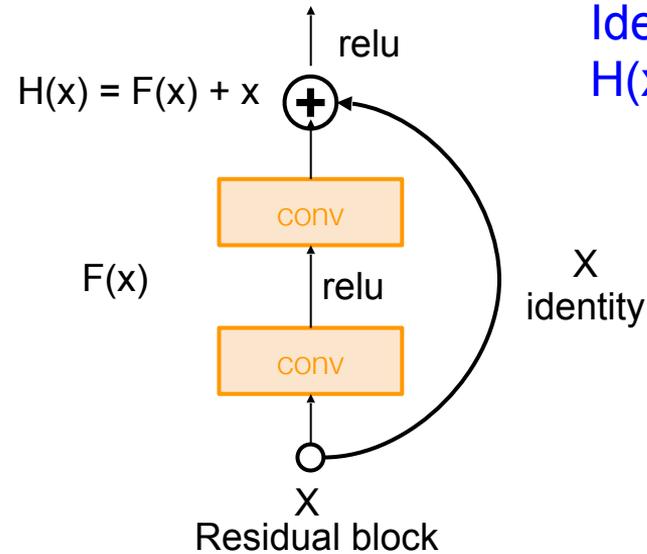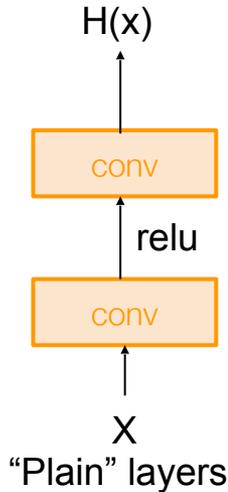
Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers

F(x) + x

relu

3x3 conv

F(x)

relu

3x3 conv

X
identity

X

Residual block

Softmax
FC 1000
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512, / 2
...
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128, / 2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
Pool
7x7 conv, 64, / 2
Input
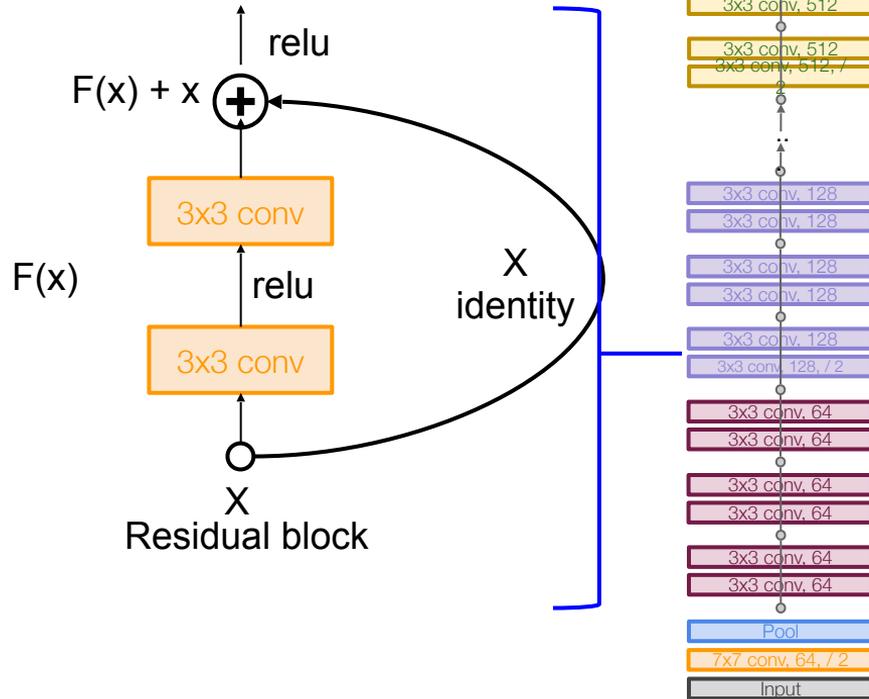
Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/ 2 in each dimension) Reduce the activation volume by half.

relu

F(x) + x ⊕

relu

F(x)

3x3 conv

3x3 conv

X

Residual block

X identity

Softmax

FC 1000

Pool

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512, /2

...

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128, / 2

3x3 conv, 128 filters, /2 spatially with stride 2

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64 filters
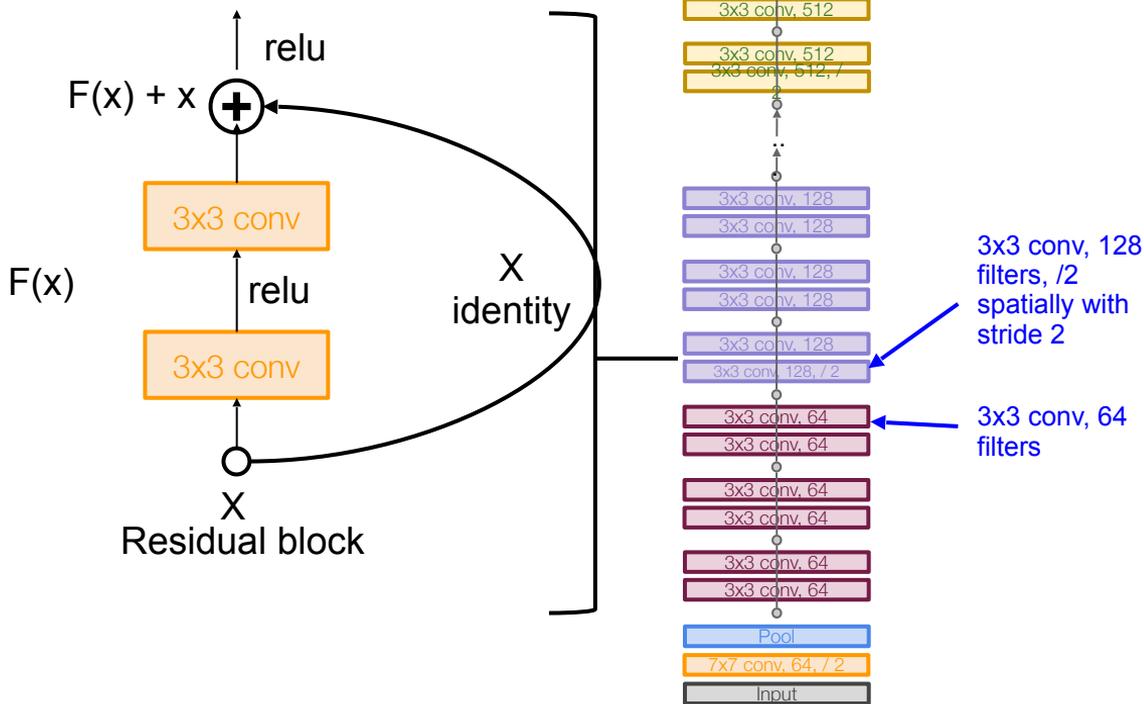
Pool

7x7 conv, 64, / 2

Input

Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/ 2 in each dimension)
- Additional conv layer at the beginning (stem)

relu

$F(x) + x$

3x3 conv

$F(x)$ relu

3x3 conv

X identity

X

Residual block

Softmax
FC 1000
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512, / 2
...
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128, / 2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
Pool
7x7 conv, 64, / 2
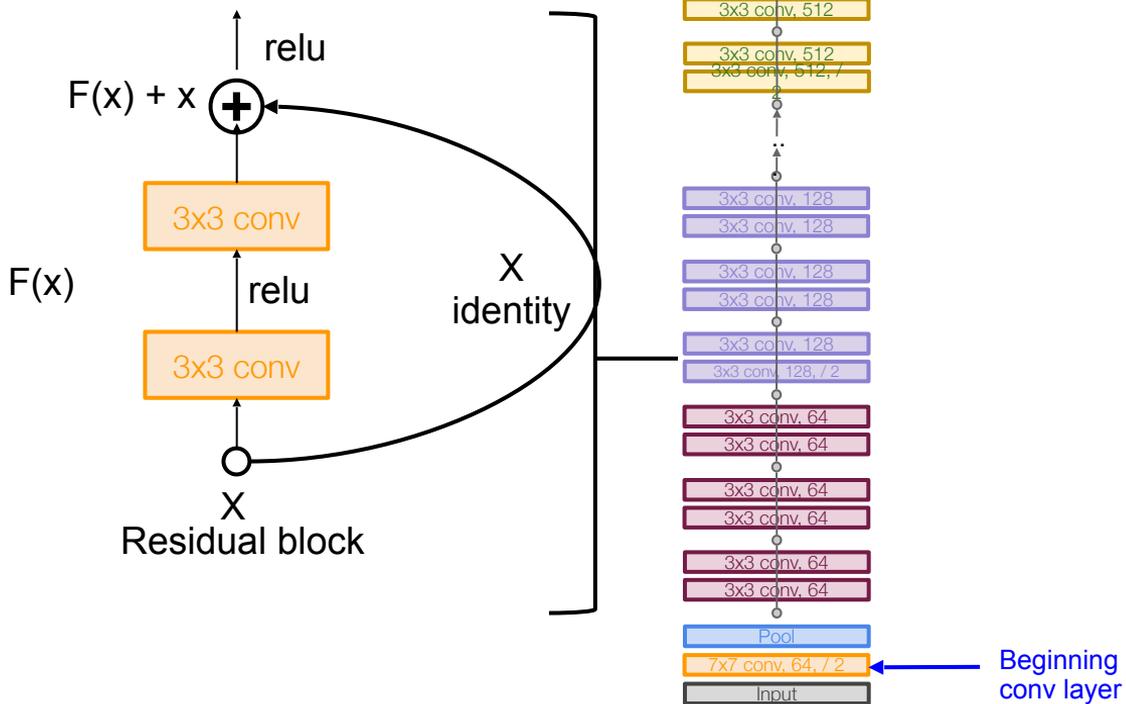Input

Beginning conv layer

Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/ 2 in each dimension)
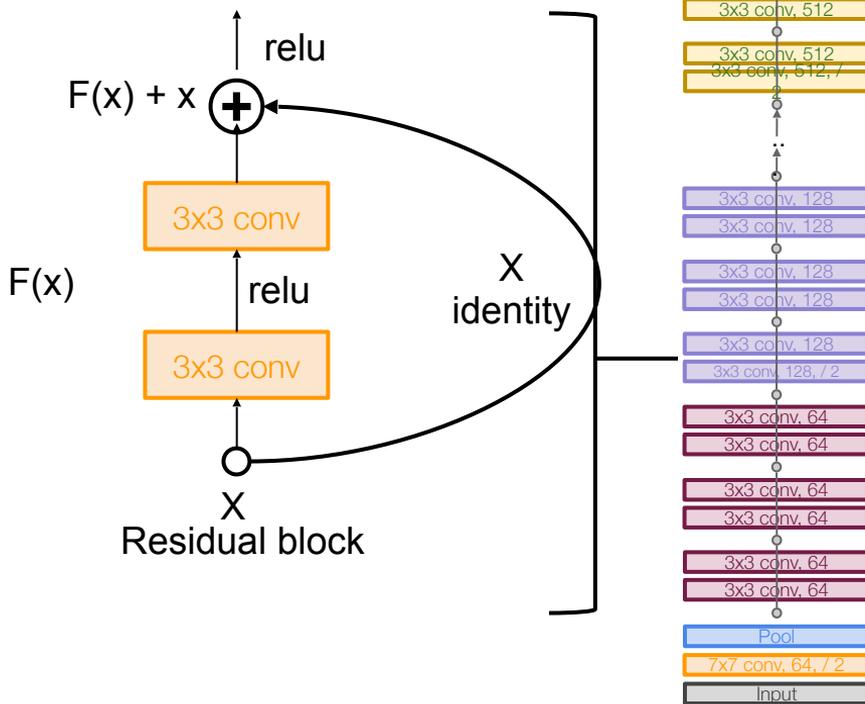- Additional conv layer at the beginning (stem)
- No FC layers at the end (only FC 1000 to output classes)
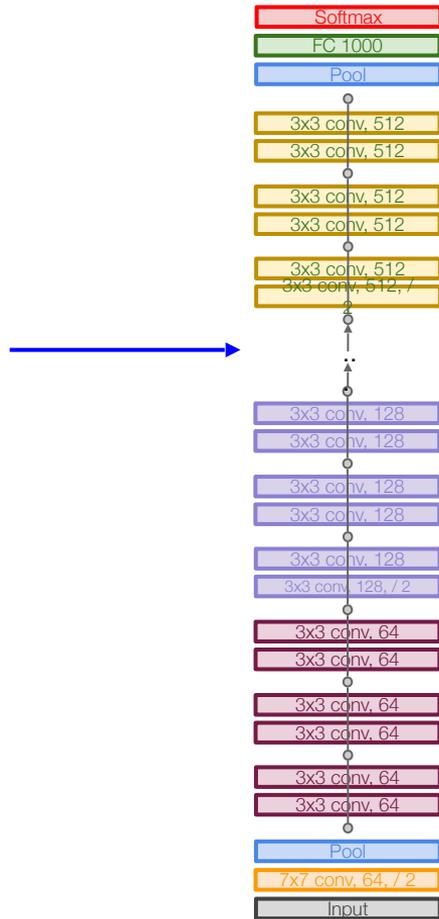- (In theory, you can train a ResNet with input image of variable sizes)

relu

$F(x) + x$ ⊕

3x3 conv

relu

$F(x)$

3x3 conv

X identity

X
Residual block

No FC layers besides FC 1000 to output classes

Global average pooling layer after last conv layer

Softmax
FC 1000
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512, / 2
...
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128, / 2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
Pool
7x7 conv, 64, / 2
Input

Case Study: ResNet

[He et al., 2015]

Total depths of 18, 34, 50, 101, or 152 layers for ImageNet

Softmax
FC 1000
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512, /

...

3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128, / 2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
Pool
7x7 conv, 64, / 2
Input

Case Study: ResNet

[He et al., 2015]

For deeper networks
(ResNet-50+), use "bottleneck"
layer to improve efficiency
(similar to GoogLeNet)

28x28x256
output

$\oplus$

1x1 conv, 256

BN, relu

3x3 conv, 64

BN, relu

1x1 conv, 64

28x28x256
input

Case Study: ResNet

[He et al., 2015]

For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)

1x1 conv, 256 filters projects back to 256 feature maps (28x28x256)

3x3 conv operates over only 64 feature maps

1x1 conv, 64 filters to project to 28x28x64

28x28x256 output

28x28x256 input

1x1 conv, 256

BN, relu

3x3 conv, 64

BN, relu

1x1 conv, 64
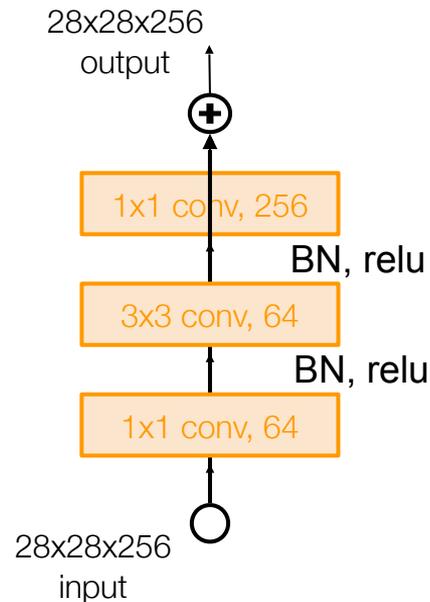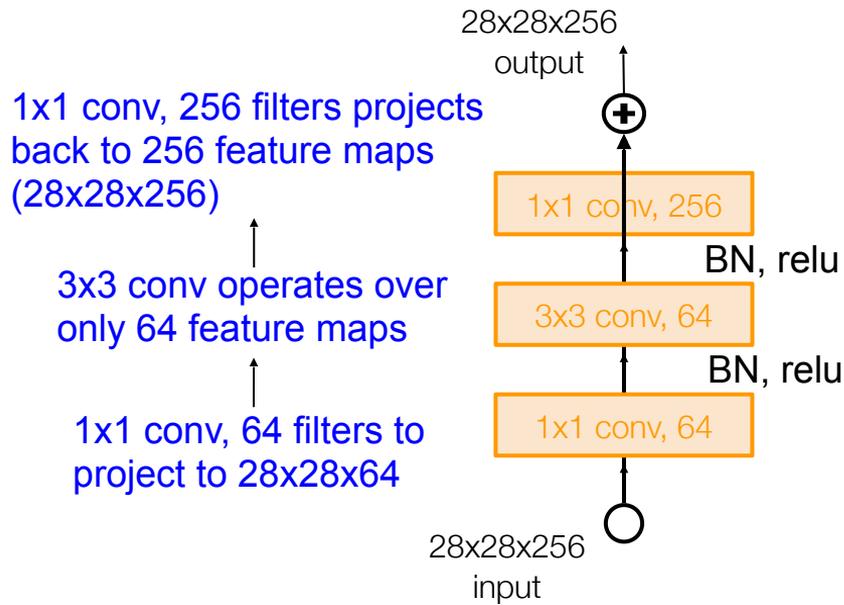
Case Study: ResNet

[He et al., 2015]

Training ResNet in practice:

- Batch Normalization after every CONV layer
- Xavier initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

Case Study: ResNet

[He et al., 2015]

Experimental Results
- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lower training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

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

Case Study: ResNet

[He et al., 2015]

Experimental Results
- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lower training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

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

ILSVRC 2015 classification winner (3.6% top 5 error) -- better than "human performance"! (Russakovsky 2014)
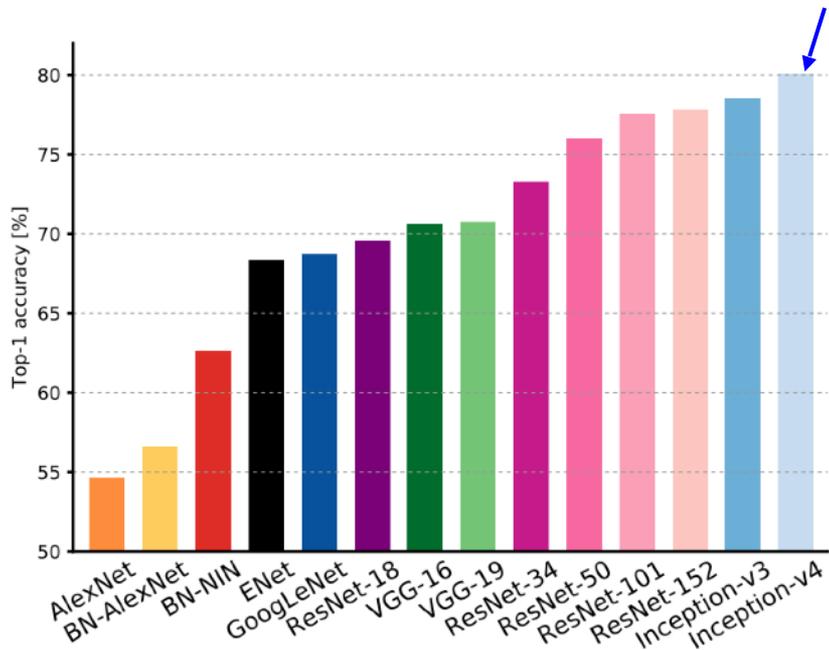
Comparing complexity...
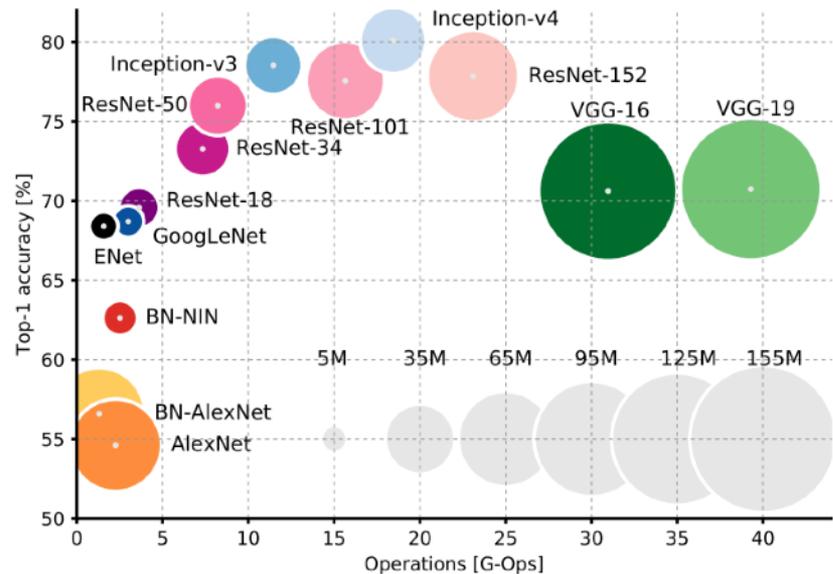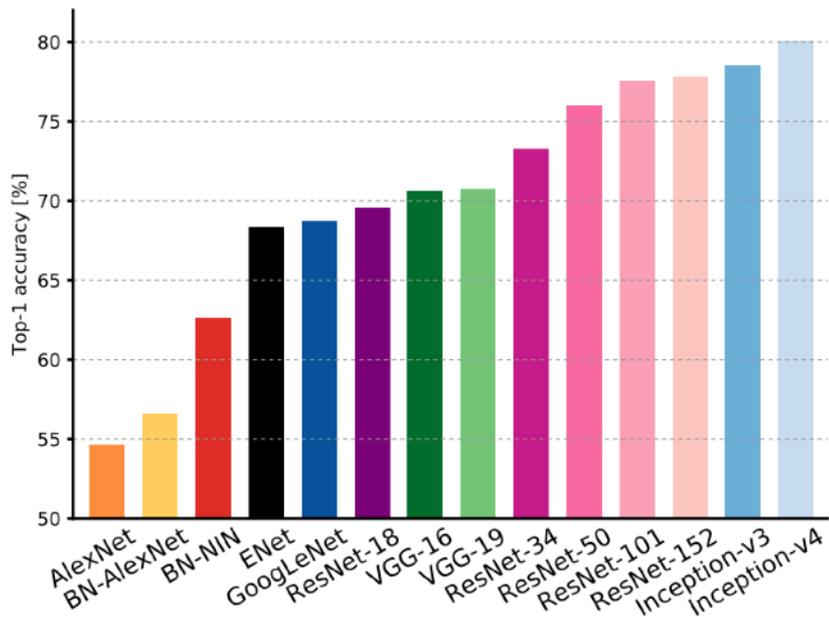
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

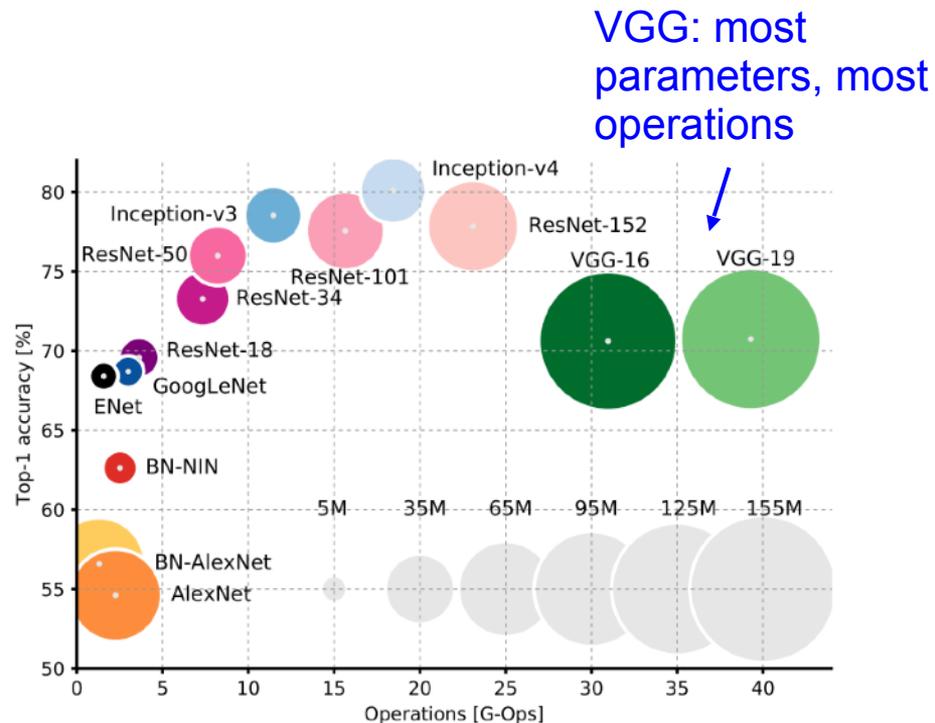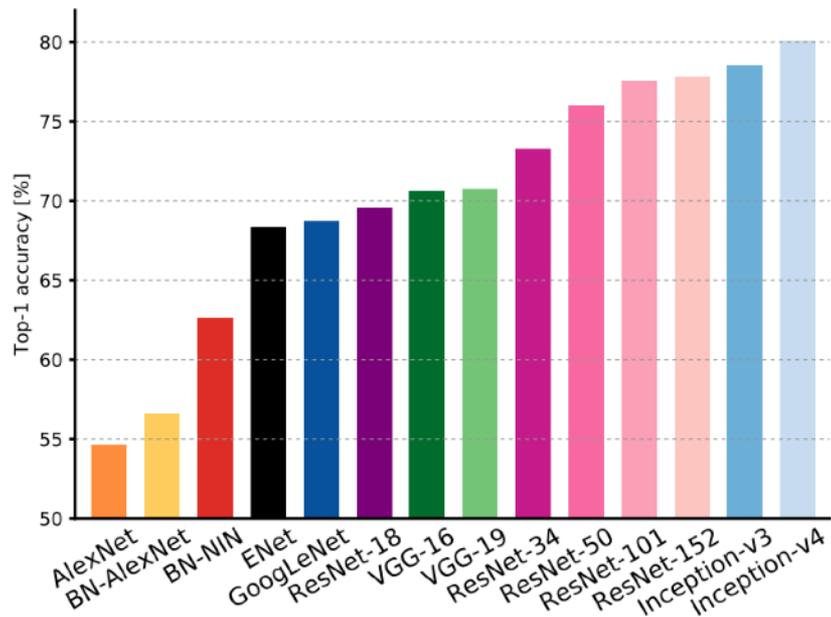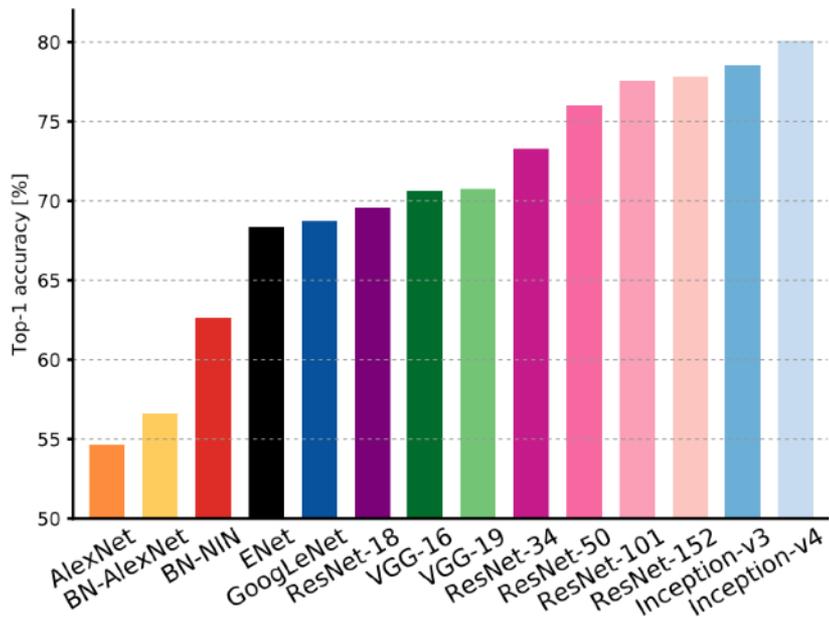# Comparing complexity...
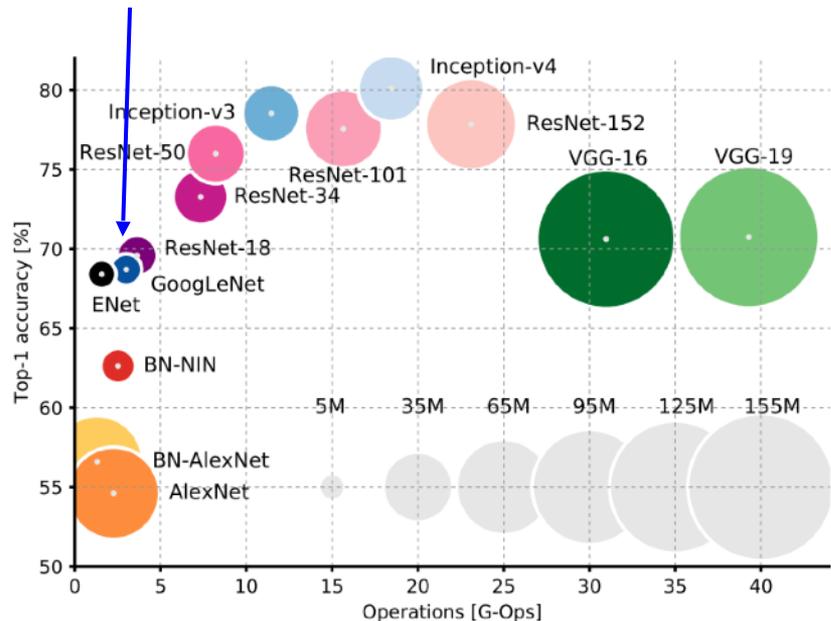
Inception-v4: Resnet + Inception!

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

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

Comparing complexity...

VGG: most parameters, most operations

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

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.
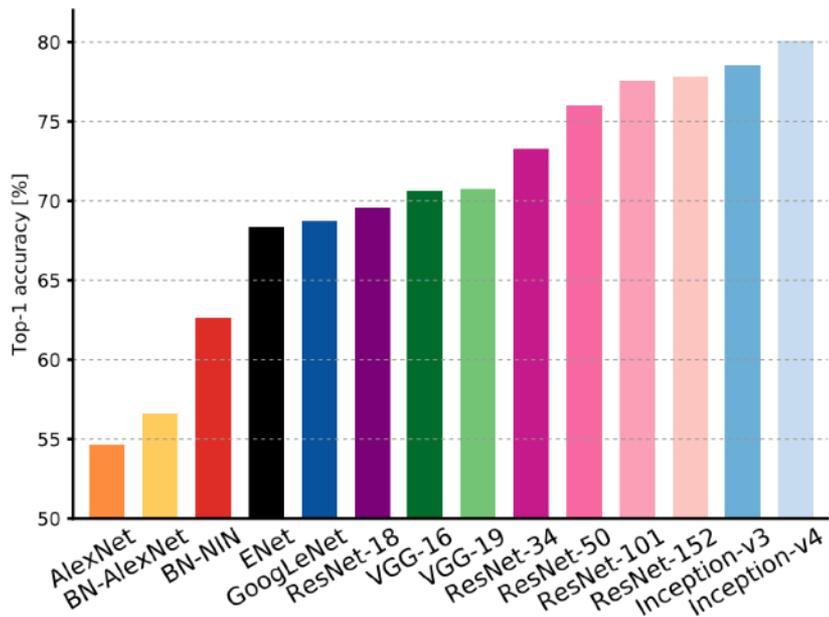
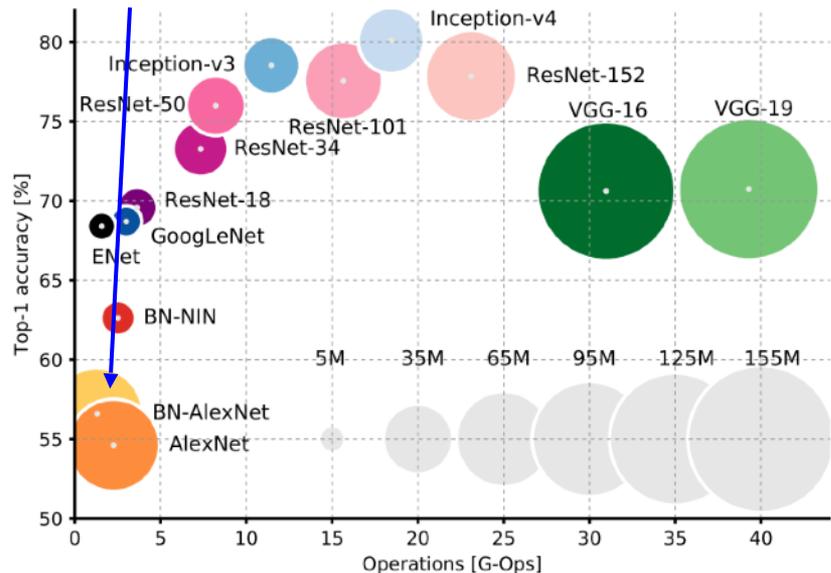# Comparing complexity...

GoogLeNet: most efficient

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

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.
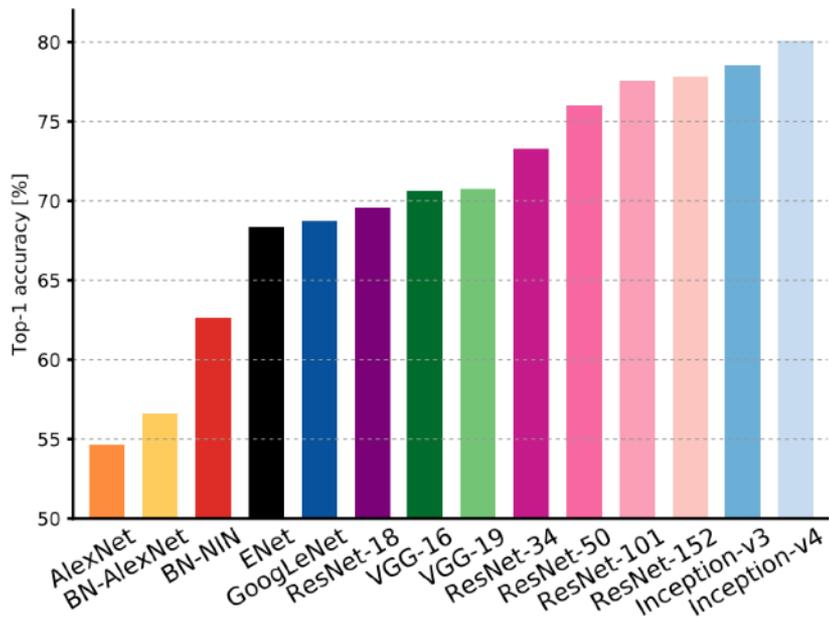
Comparing complexity...

AlexNet:
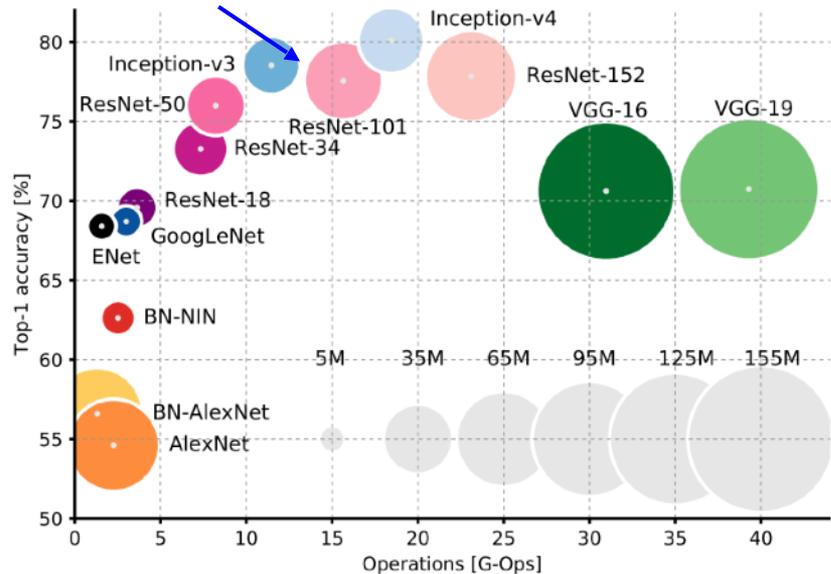Smaller compute, still memory heavy, lower accuracy

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

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.
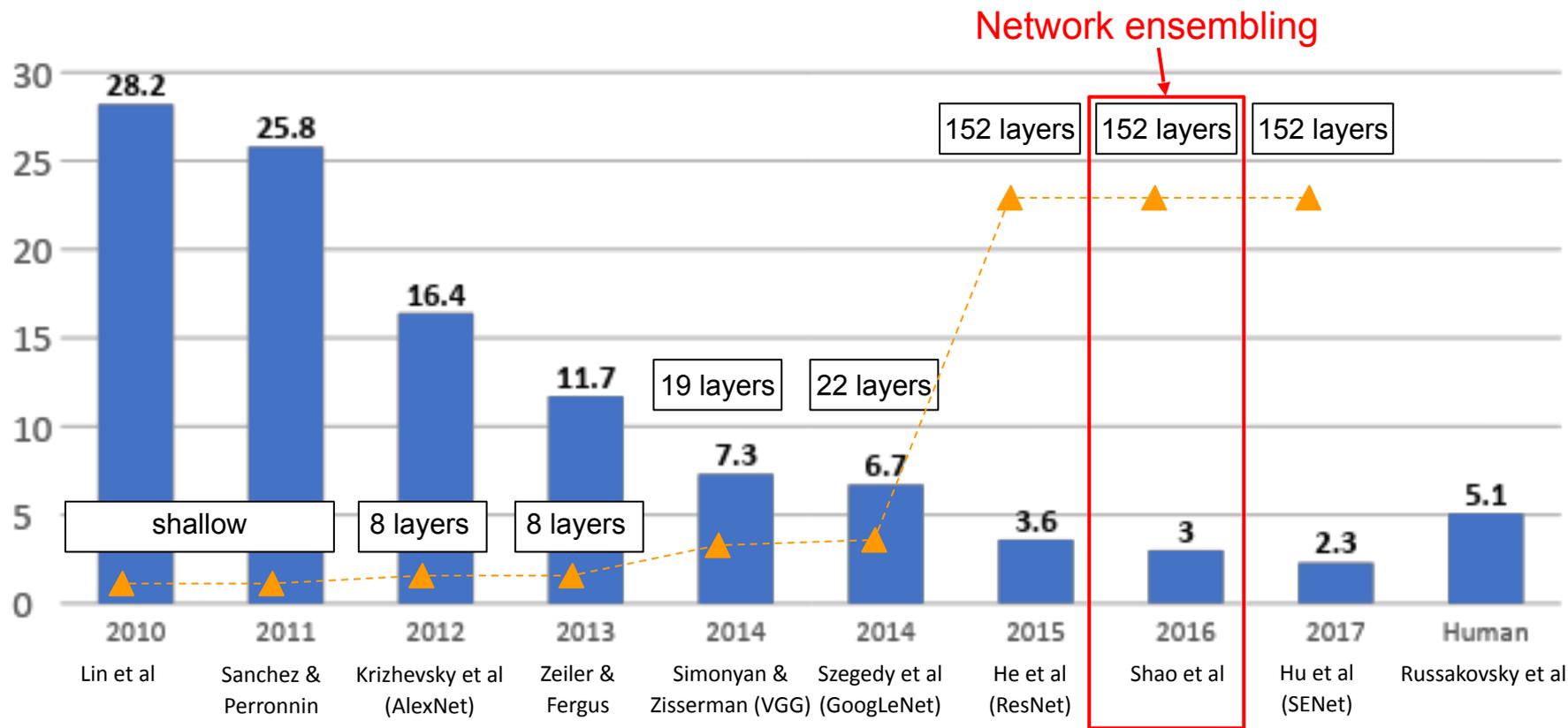
Comparing complexity...

ResNet:
Moderate efficiency depending on model, highest accuracy

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

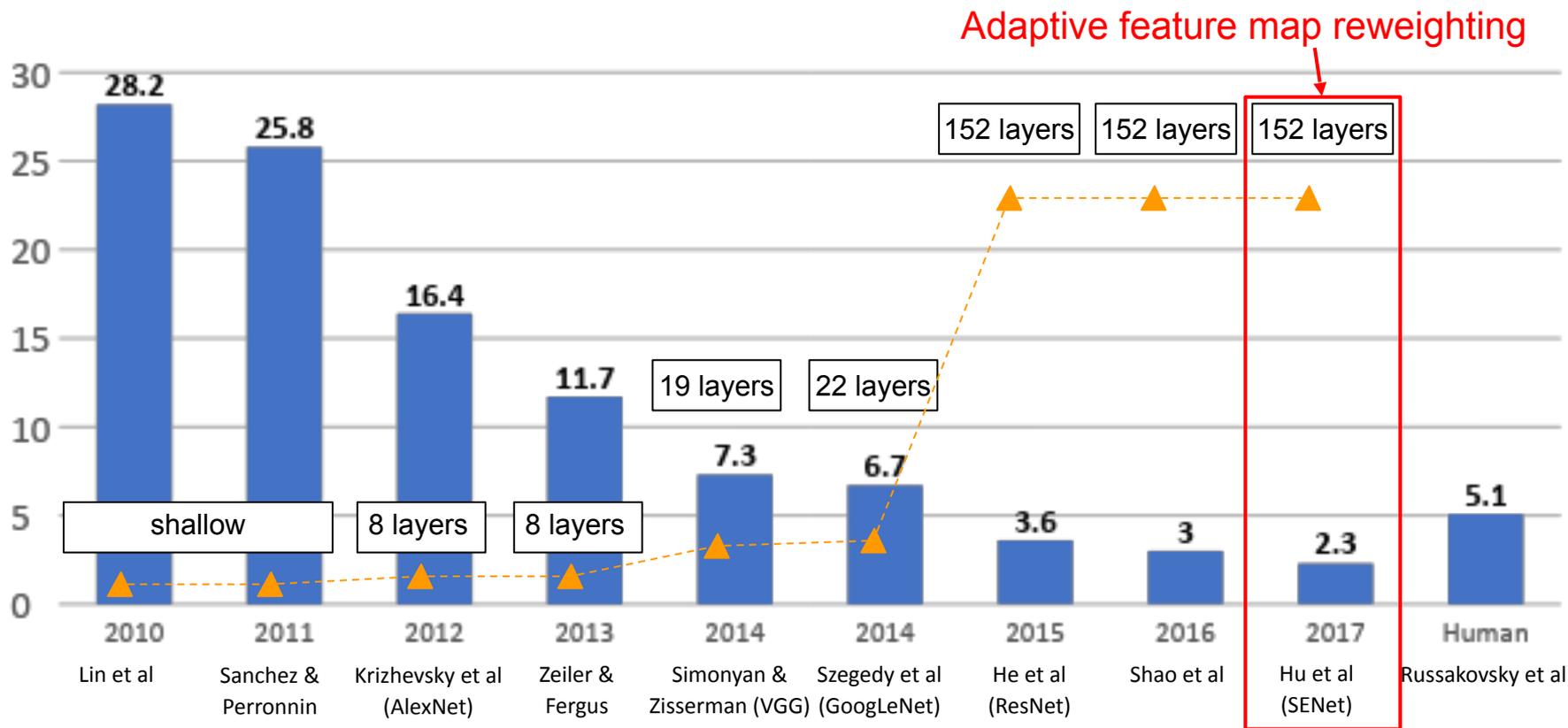# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

"Good Practices for Deep Feature Fusion"

[Shao et al. 2016]

- Multi-scale ensembling of Inception, Inception-Resnet, Resnet, Wide Resnet models
- ILSVRC'16 classification winner

	Inception-v3	Inception-v4	Inception-Resnet-v2	Resnet-200	Wrn-68-3	Fusion（Val.）	Fusion（Test）
Err. (%)	4.20	4.01	3.52	4.26	4.65	2.92 (-0.6)	2.99

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

Adaptive feature map reweighting

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

Completion of the challenge:
Annual ImageNet competition no longer held after 2017 -> now moved to Kaggle.
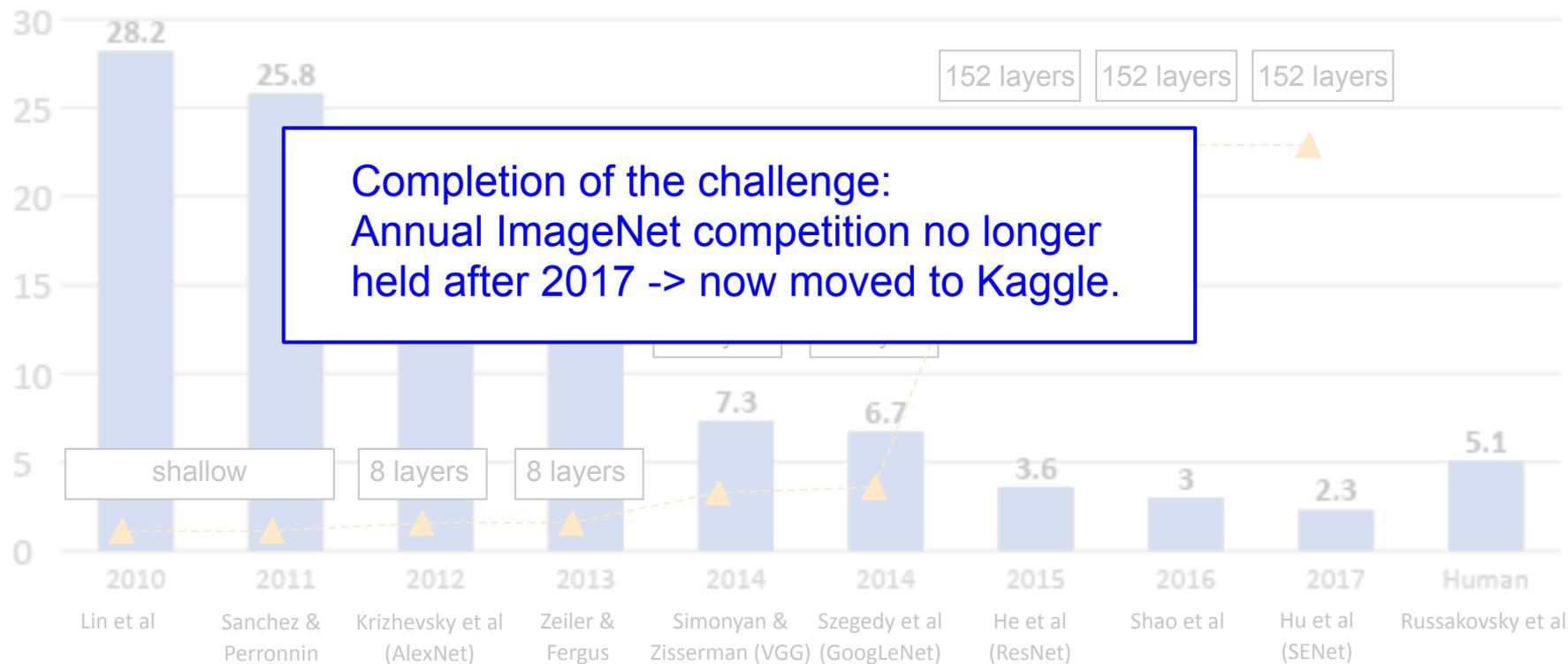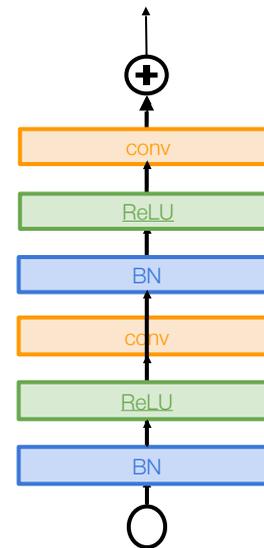
But research into CNN architectures is still flourishing

Improving ResNets...

Identity Mappings in Deep Residual Networks

[He et al. 2016]

- Improved ResNet block design from creators of ResNet
- Creates a more direct path for propagating information throughout network
- Gives better performance

Improving ResNets...
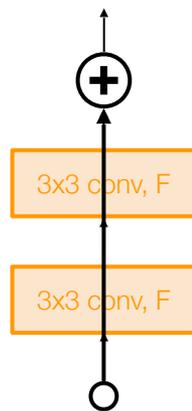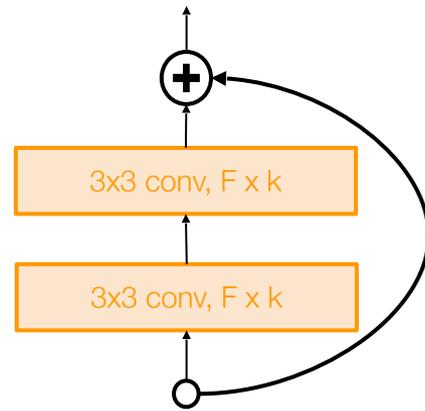
Wide Residual Networks

[Zagoruyko et al. 2016]

- Argues that residuals are the important factor, not depth
- User wider residual blocks (F x k filters instead of F filters in each layer)
- 50-layer wide ResNet outperforms 152-layer original ResNet
- Increasing width instead of depth more computationally efficient (parallelizable)

Basic residual block   Wide residual block

Improving ResNets...

Aggregated Residual Transformations for Deep Neural Networks (ResNeXt)

[Xie et al. 2016]

- Also from creators of ResNet
- Increases width of residual block through multiple parallel pathways ("cardinality")
- Parallel pathways similar in spirit to Inception module

256-d out

| 1x1 conv, 256 |

| 3x3 conv, 64 |

| 1x1 conv, 64 |

256-d in

256-d out

| 1x1 conv, 256 | | 1x1 conv, 256 | | 1x1 conv, 256 |

| 3x3 conv, 4 | | 3x3 conv, 4 | | 3x3 conv, 4 |

| 1x1 conv, 4 | | 1x1 conv, 4 | | 1x1 conv, 4 |

32 paths ...

256-d in
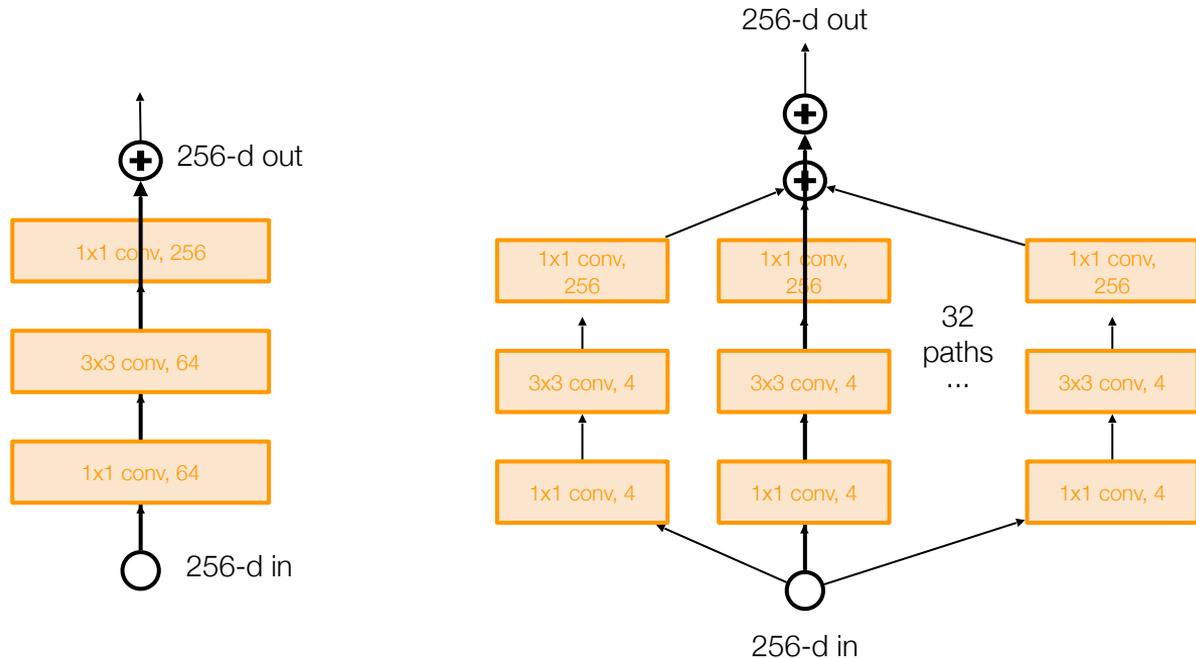
Other ideas...

Densely Connected Convolutional Networks (DenseNet)

[Huang et al. 2017]

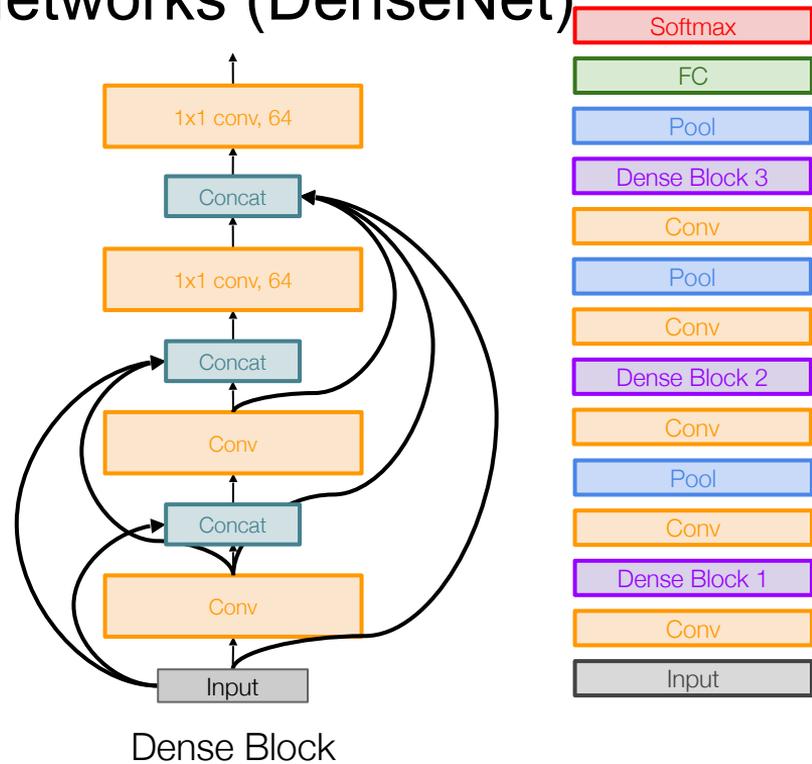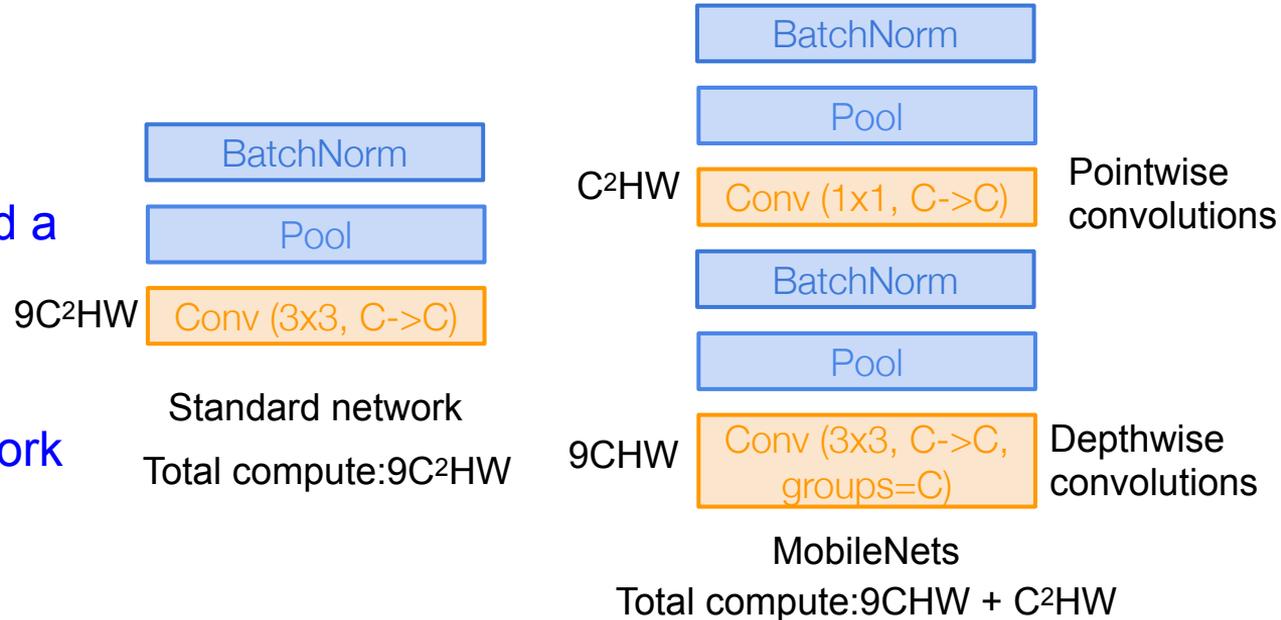- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse
- Showed that shallow 50-layer network can outperform deeper 152 layer ResNet

Dense Block

Efficient networks...

MobileNets: Efficient Convolutional Neural Networks for Mobile Applications *[Howard et al. 2017]*

- Depthwise separable convolutions replace standard convolutions by factorizing them into a depthwise convolution and a 1x1 convolution
- Much more efficient, with little loss in accuracy
- Follow-up MobileNetV2 work in 2018 (Sandler et al.)
- ShuffleNet: Zhang et al, CVPR 2018

BatchNorm

Pool

$9C^2HW$ — Conv (3x3, C->C)

Standard network

Total compute: $9C^2HW$

BatchNorm

Pool

C^2HW — Conv (1x1, C->C)   Pointwise convolutions

BatchNorm

Pool

$9CHW$ — Conv (3x3, C->C, groups=C)   Depthwise convolutions

MobileNets

Total compute: $9CHW + C^2HW$

Learning to search for network architectures...

Neural Architecture Search with Reinforcement Learning (NAS)

[Zoph et al. 2016]

- "Controller" network that learns to design a good network architecture (output a string corresponding to network design)
- Iterate:
    1) Sample an architecture from search space
    2) Train the architecture to get a "reward" R corresponding to accuracy
    3) Compute gradient of sample probability, and scale by R to perform controller parameter update (i.e. increase likelihood of good architecture being sampled, decrease likelihood of bad architecture)
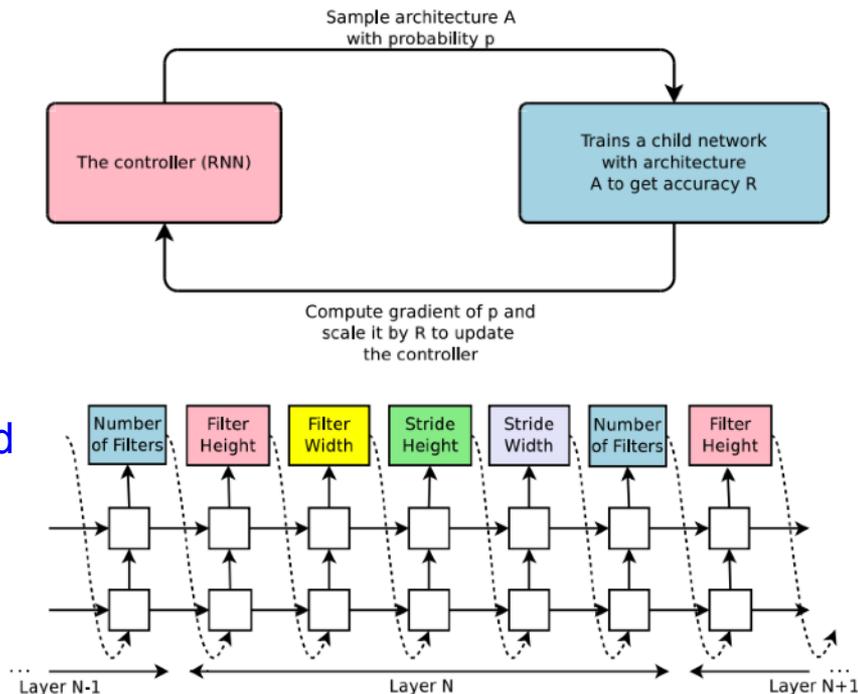
But sometimes smart heuristic is better than NAS ...

EfficientNet: Smart Compound Scaling

[Tan and Le. 2019]

- Increase network capacity by scaling width, depth, and resolution, while balancing accuracy and efficiency.
- Search for optimal set of compound scaling factors given a compute budget (target memory & flops).
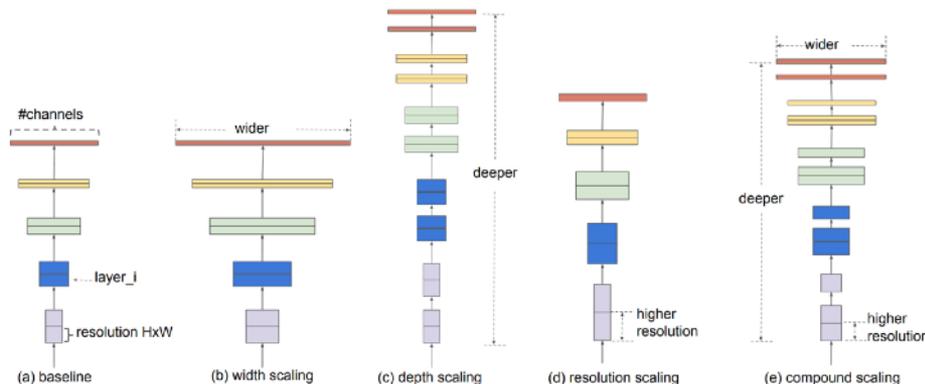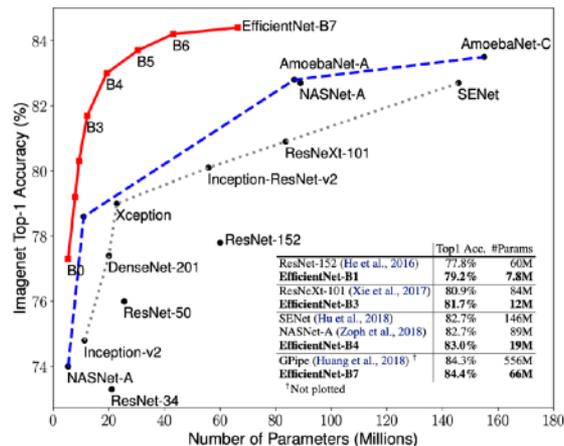- Scale up using smart heuristic rules

$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

Summary: CNN Architectures

Case Studies

- AlexNet
- VGG
- GoogLeNet
- ResNet

Also....

- SENet
- Wide ResNet
- ResNeXT

- DenseNet
- MobileNets
- NASNet

Main takeaways

AlexNet showed that you can use CNNs to train Computer Vision models.
ZFNet, **VGG** shows that bigger networks work better
GoogLeNet is one of the first to focus on efficiency using 1x1 bottleneck convolutions and global avg pool instead of FC layers
ResNet showed us how to train extremely deep networks
  - Limited only by GPU & memory!
  - Showed diminishing returns as networks got bigger
After ResNet: CNNs were better than the human metric and focus shifted to Efficient networks:
  - Lots of tiny networks aimed at mobile devices: **MobileNet**, **ShuffleNet**
Neural Architecture Search can now automate architecture design

Summary: CNN Architectures

- Many popular architectures are available in model zoos.
- ResNets are currently good defaults to use.
- Networks have gotten increasingly deep over time.
- Many other aspects of network architectures are also continuously being investigated and improved.