Convolutional Neural Networks V

CS7150, Spring 2025

Prof. Huaizu Jiang

Northeastern University

Recap

Batch Normalization for ConvNets

Batch Normalization for **fully-connected** networks

$$x : N \times D$$

Normalize

$$\mu, \sigma : 1 \times D$$

$$\gamma, \beta : 1 \times D$$

$$y = \frac{(x - \mu)}{\sigma}\gamma + \beta$$

Batch Normalization for **convolutional** networks
(Spatial Batchnorm, BatchNorm2D)

$$x : N \times C \times H \times W$$

Normalize

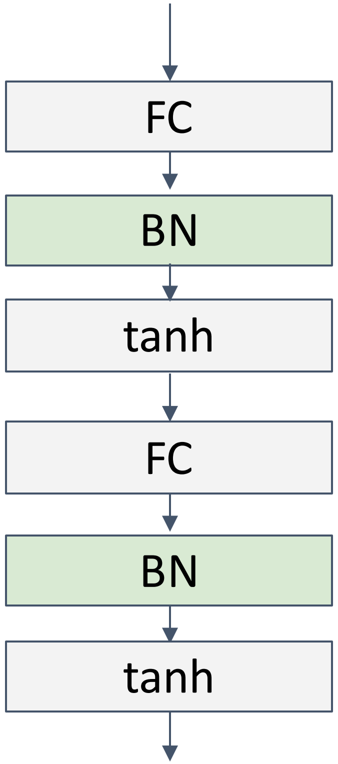
$$\mu, \sigma : 1 \times C \times 1 \times 1$$

$$\gamma, \beta : 1 \times C \times 1 \times 1$$

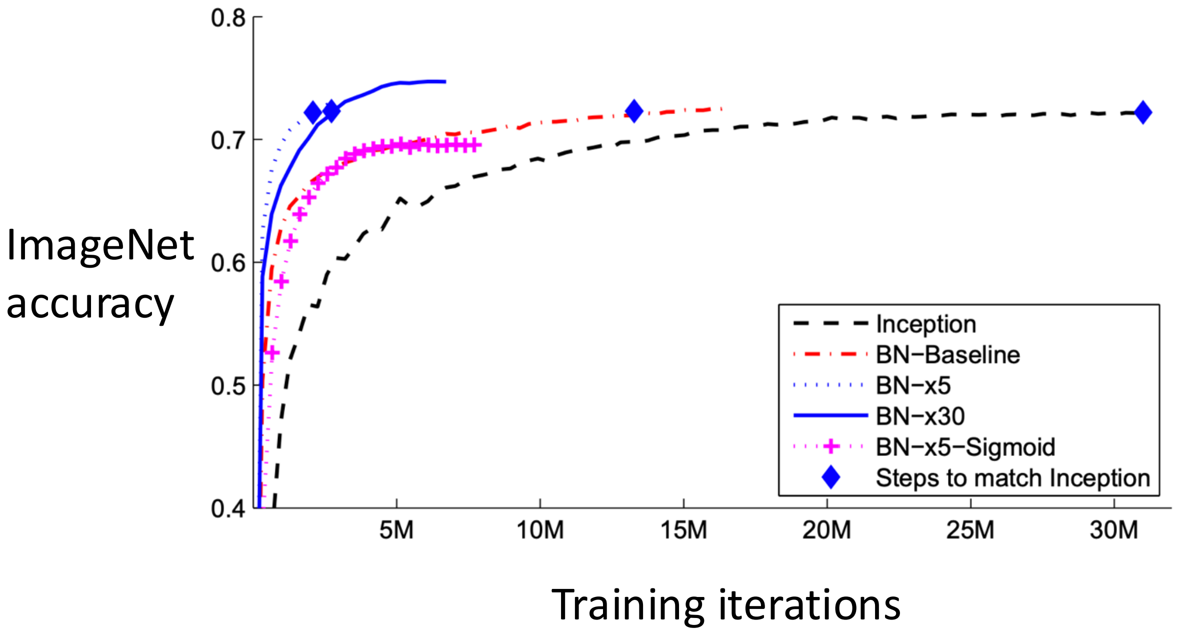
$$y = \frac{(x - \mu)}{\sigma}\gamma + \beta$$

Slide credit: D Fouhey & J Johnson

Batch Normalization

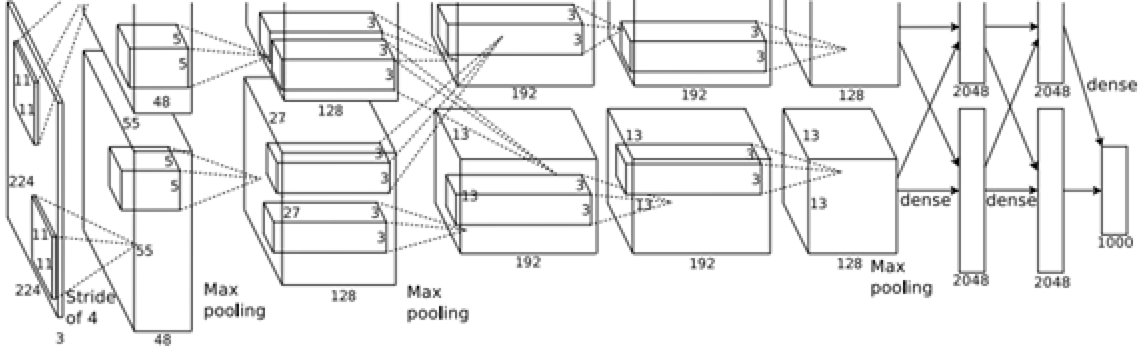


- Makes deep networks **much** easier to train!
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as regularization during training
- Free at test-time: can be fused with conv!



Slide credit: D Fouhey & J Johnson

AlexNet

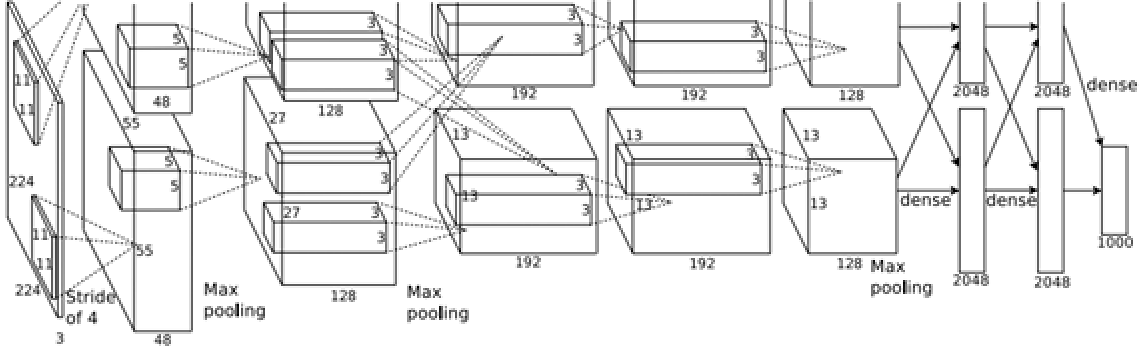


How to choose this?
Trial and error =(

	Input size		Layer				Output size				
Layer	C	H / W	filters	kernel	stride	pad	C	H / W	memory (KB)	params (k)	flop (M)
conv1	3	227	64	11	4	2	64	56	784	23	73
pool1	64	56		3	2	0	64	27	182	0	0
conv2	64	27	192	5	1	2	192	27	547	307	224
pool2	192	27		3	2	0	192	13	127	0	0
conv3	192	13	384	3	1	1	384	13	254	664	112
conv4	384	13	256	3	1	1	256	13	169	885	145
conv5	256	13	256	3	1	1	256	13	169	590	100
pool5	256	13		3	2	0	256	6	36	0	0
flatten	256	6					9216		36	0	0
fc6	9216		4096				4096		16	37,749	38
fc7	4096		4096				4096		16	16,777	17
fc8	4096		1000				1000		4	4,096	4

Slide credit: D Fouhey & J Johnson

AlexNet

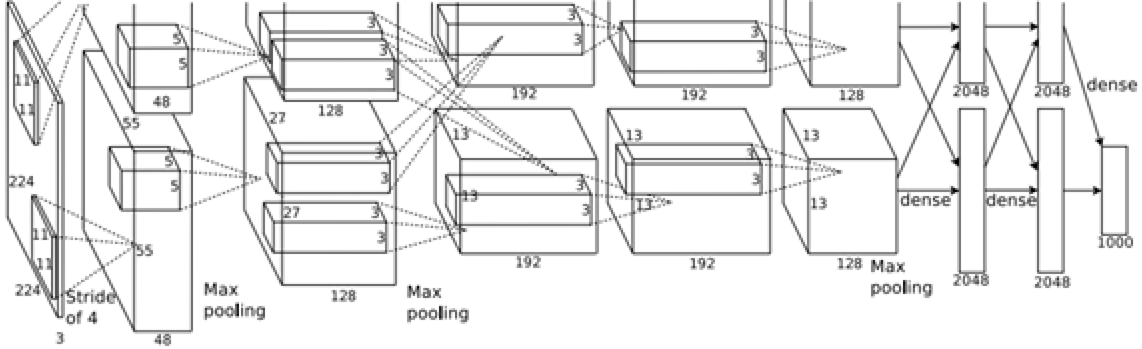


Interesting trends here!

	Input size		Layer				Output size				
Layer	C	H / W	filters	kernel	stride	pad	C	H / W	memory (KB)	params (k)	flop (M)
conv1	3	227	64	11	4	2	64	56	784	23	73
pool1	64	56		3	2	0	64	27	182	0	0
conv2	64	27	192	5	1	2	192	27	547	307	224
pool2	192	27		3	2	0	192	13	127	0	0
conv3	192	13	384	3	1	1	384	13	254	664	112
conv4	384	13	256	3	1	1	256	13	169	885	145
conv5	256	13	256	3	1	1	256	13	169	590	100
pool5	256	13		3	2	0	256	6	36	0	0
flatten	256	6					9216		36	0	0
fc6	9216		4096				4096		16	37,749	38
fc7	4096		4096				4096		16	16,777	17
fc8	4096		1000				1000		4	4,096	4

Slide credit: D Fouhey & J Johnson

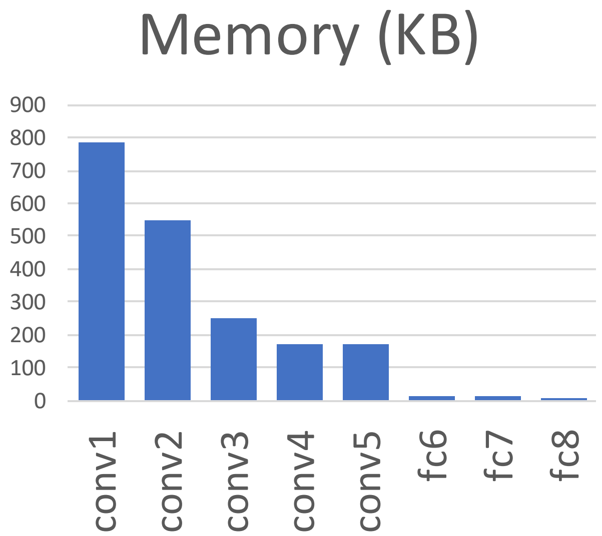
AlexNet

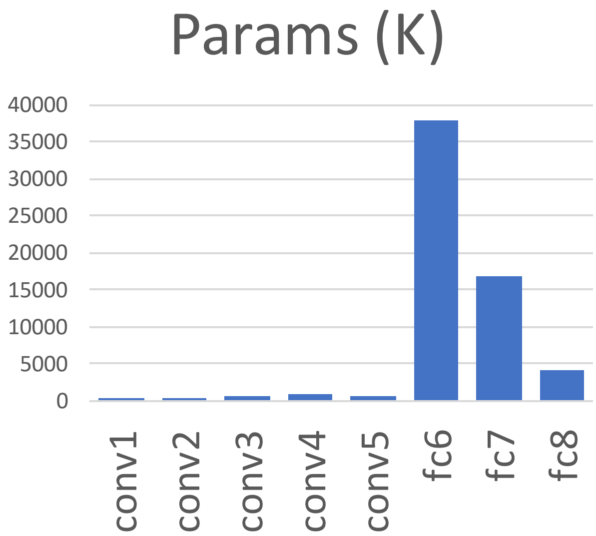


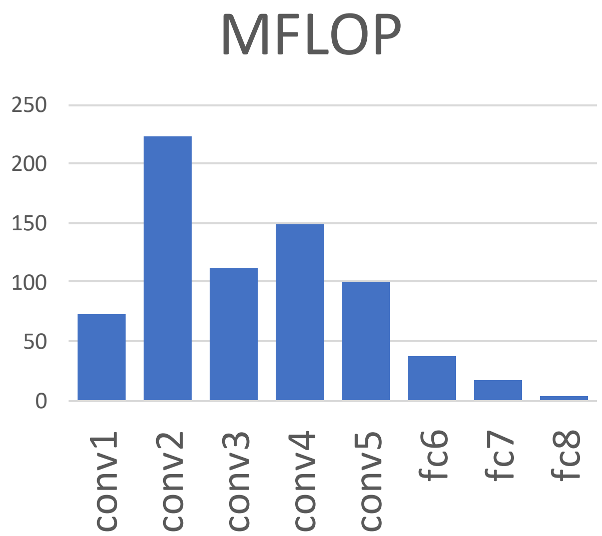
Most of the **memory usage** is in the early convolution layers

Nearly all **parameters** are in the fully-connected layers

Most **floating-point ops** occur in the convolution layers







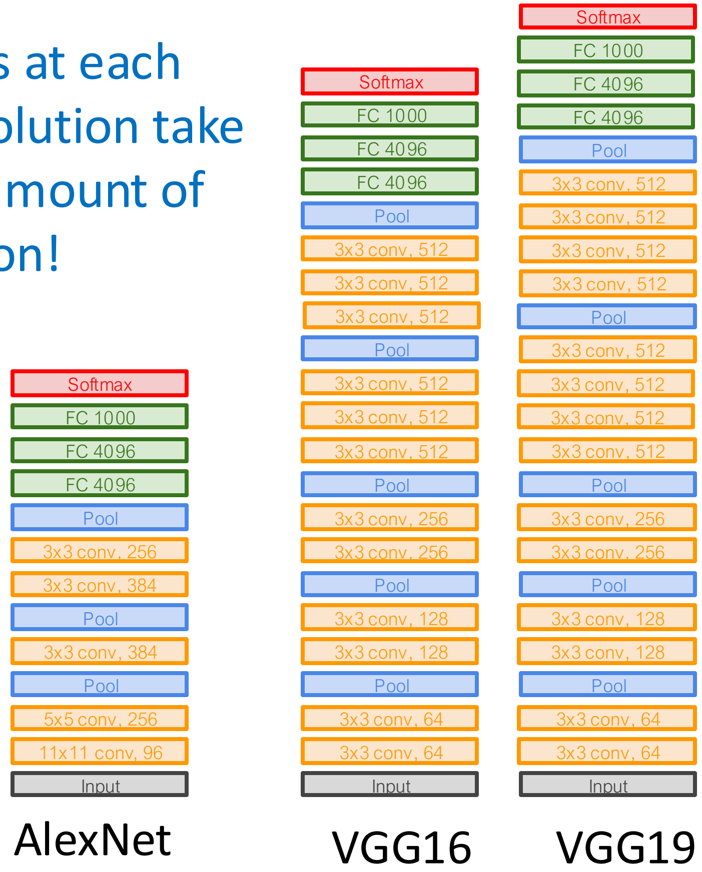
Slide credit: D Fouhey & J Johnson

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!

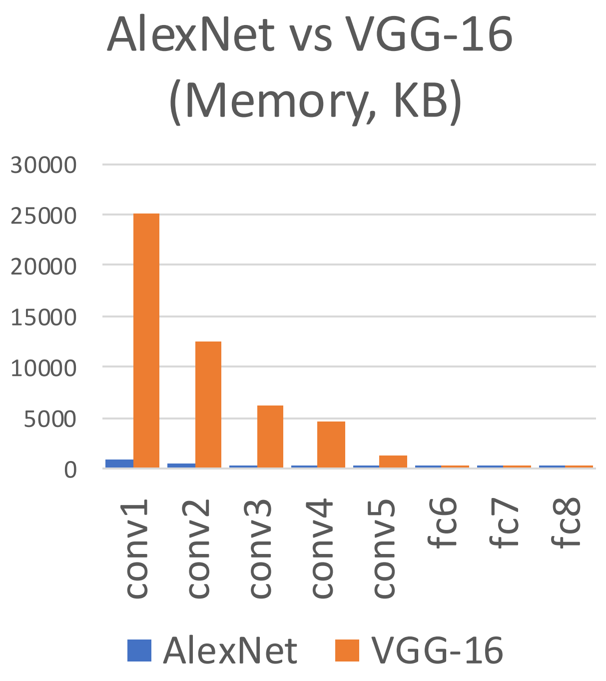
Input: C x 2H x 2W	Input: 2C x H x W
Layer: Conv(3x3, C->C)	Conv(3x3, 2C -> 2C)
Memory: 4HWC	Memory: 2HWC
Params: $9C^2$	Params: $36C^2$
FLOPs: $36HWC^2$	FLOPs: $36HWC^2$

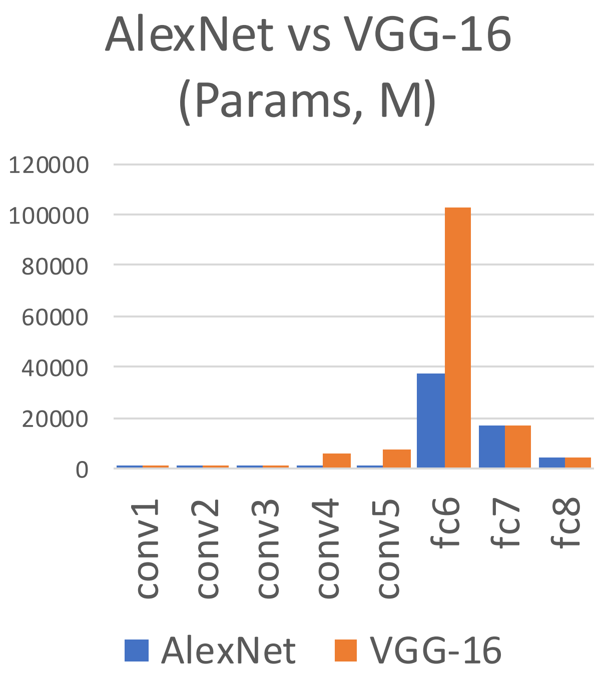


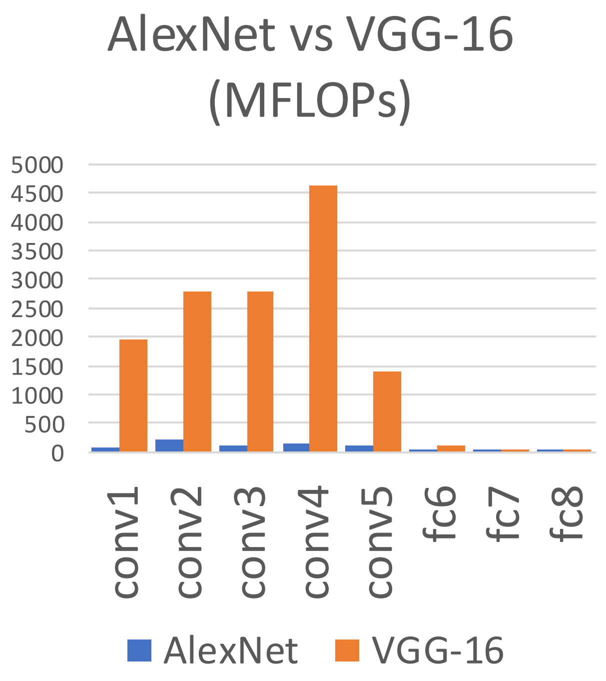
Simonyan and Zissermann, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, ICLR 2015

Slide credit: D Fouhey & J Johnson

AlexNet vs VGG-16: Much Bigger!







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

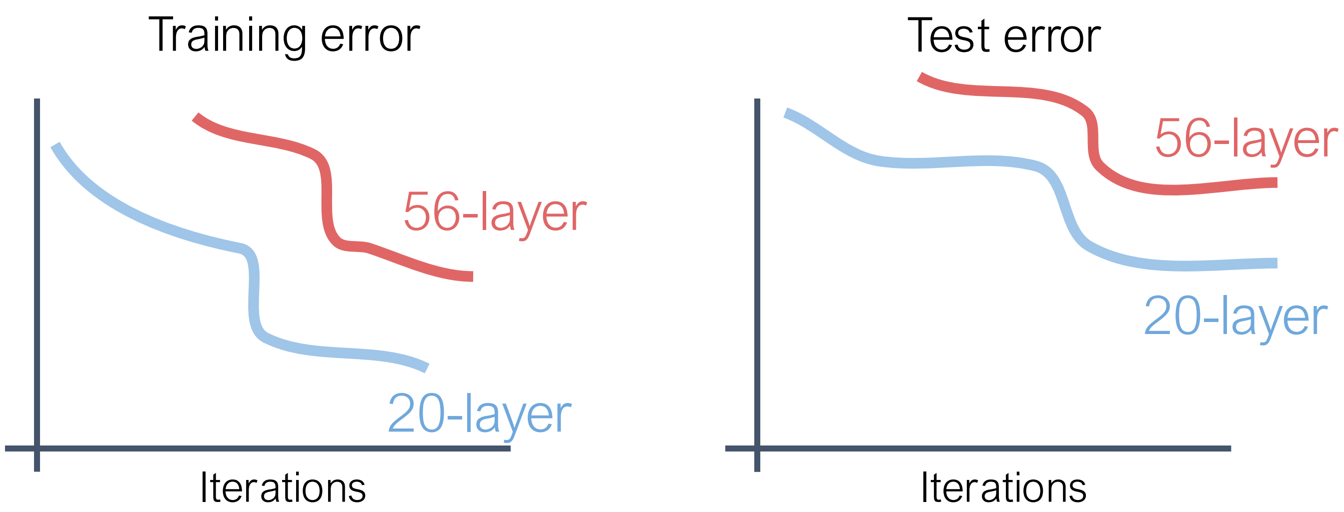
AlexNet total: 61M
VGG-16 total: 138M (2.3x)

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

Slide credit: D Fouhey & J Johnson

Residual Networks

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



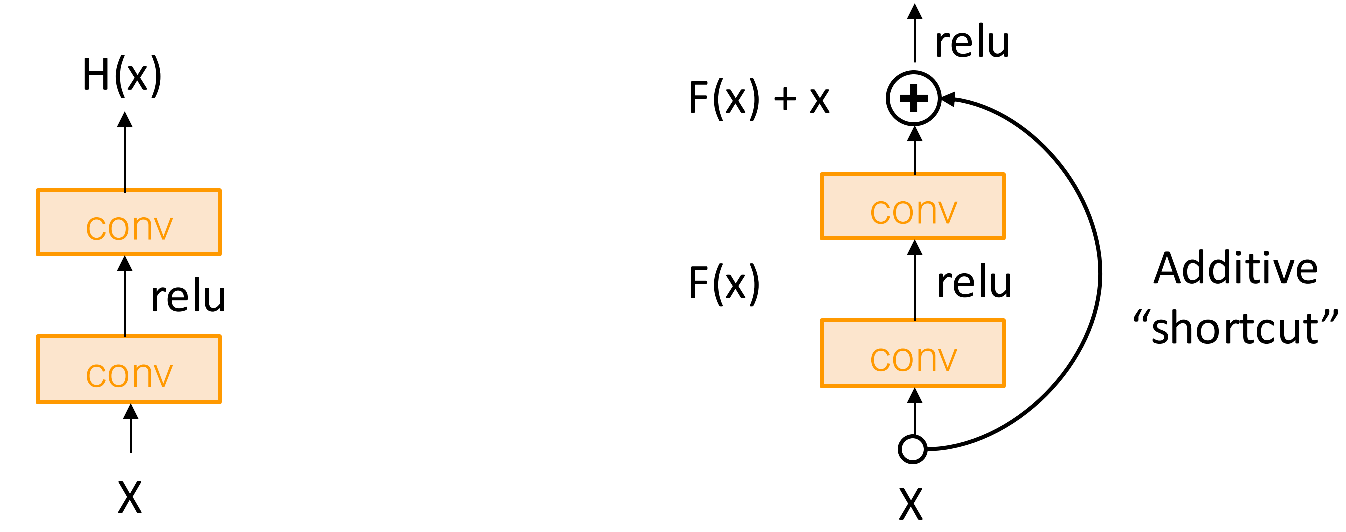
In fact the deep model seems to be **underfitting** since it also performs worse than the shallow model on the training set! It is actually **underfitting**

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

Slide credit: D Fouhey & J Johnson

Residual Networks

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



"Plain" block

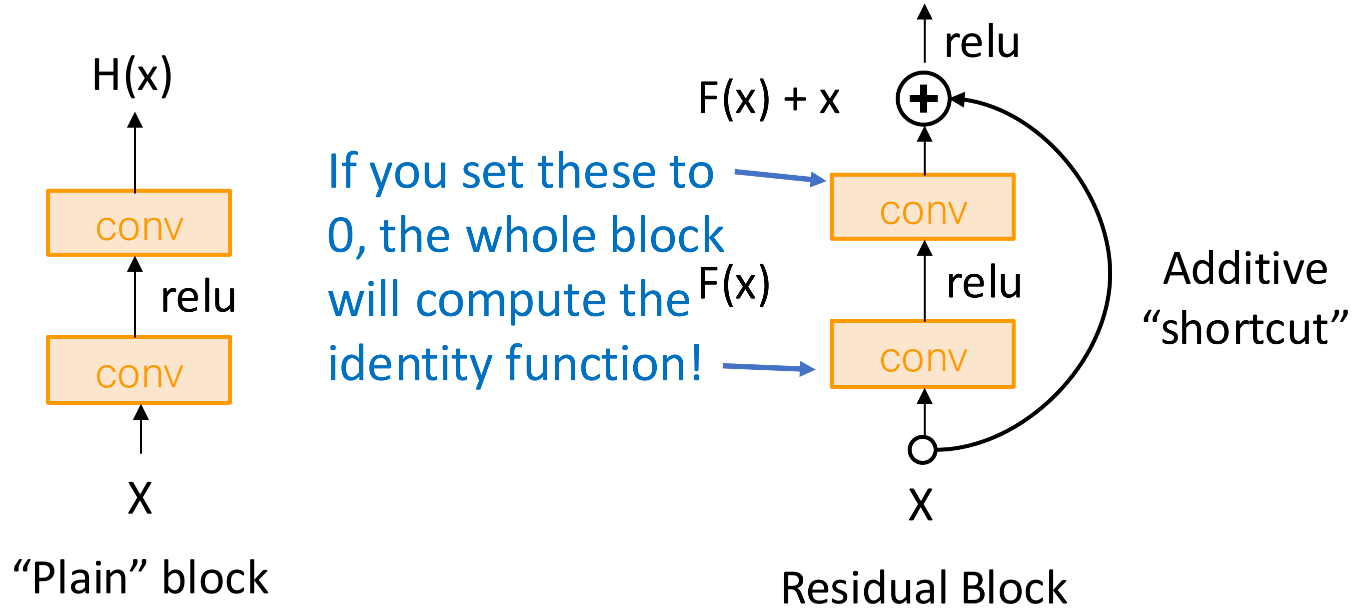
Residual Block

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

Slide credit: D Fouhey & J 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

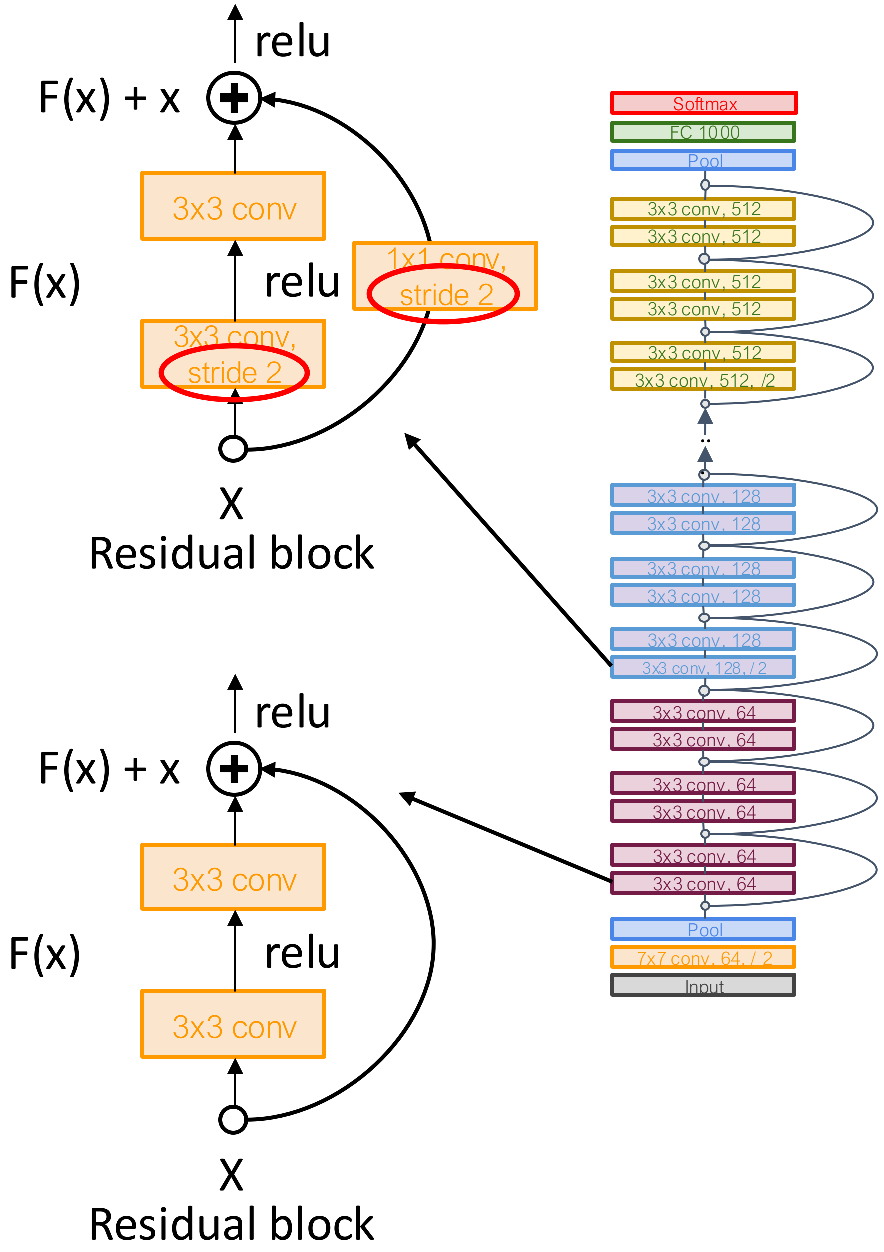
Slide credit: D Fouhey & J Johnson

Residual Networks

A residual network is a stack of many residual blocks

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

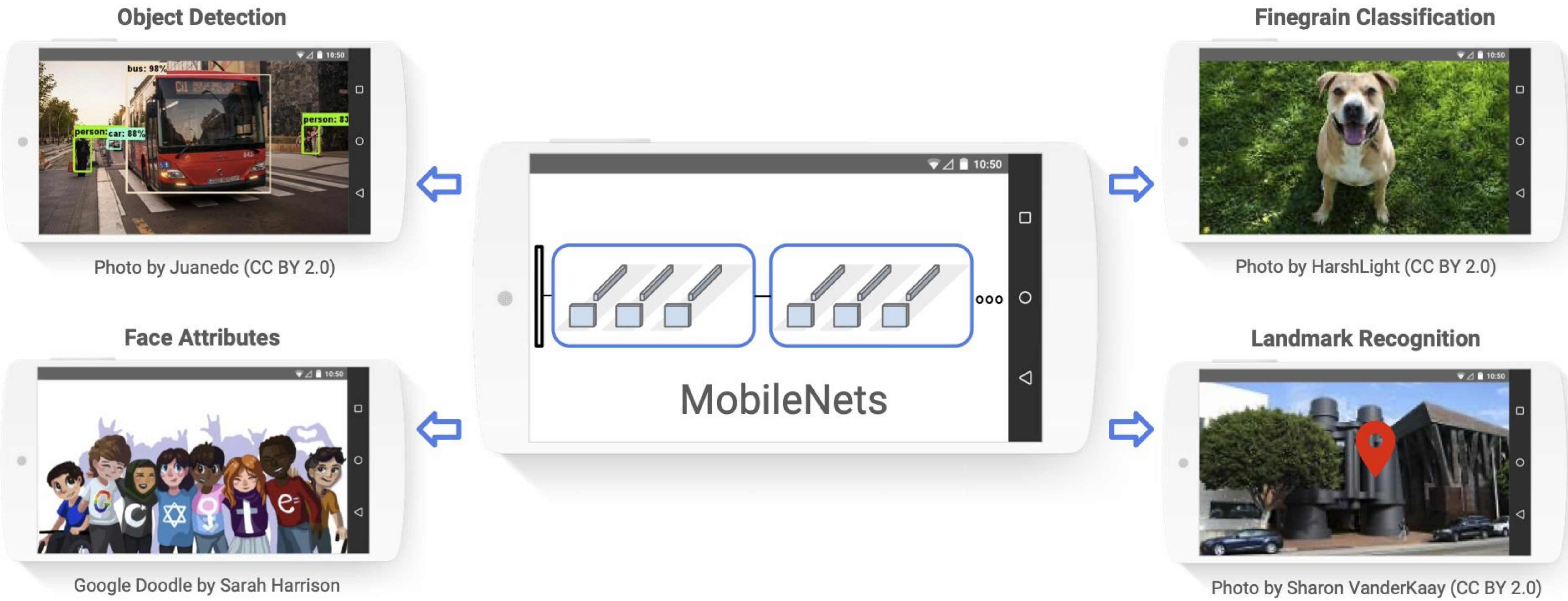
Network is divided into **stages**: the first block of each stage halves the resolution (with stride-2 conv) and doubles the number of channels



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

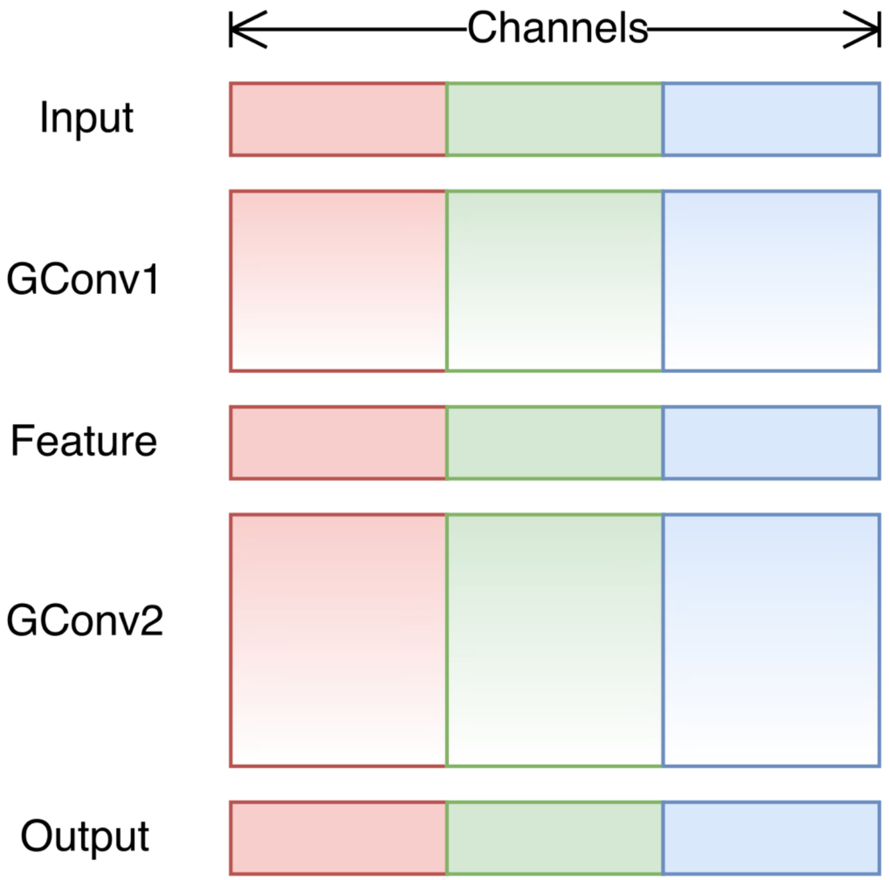
Adapted from: D Fouhey & J Johnson

Tiny Networks for Mobile Devices



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

Group-based Convolution



Input: C_{in} x H x W

Hyperparameters:

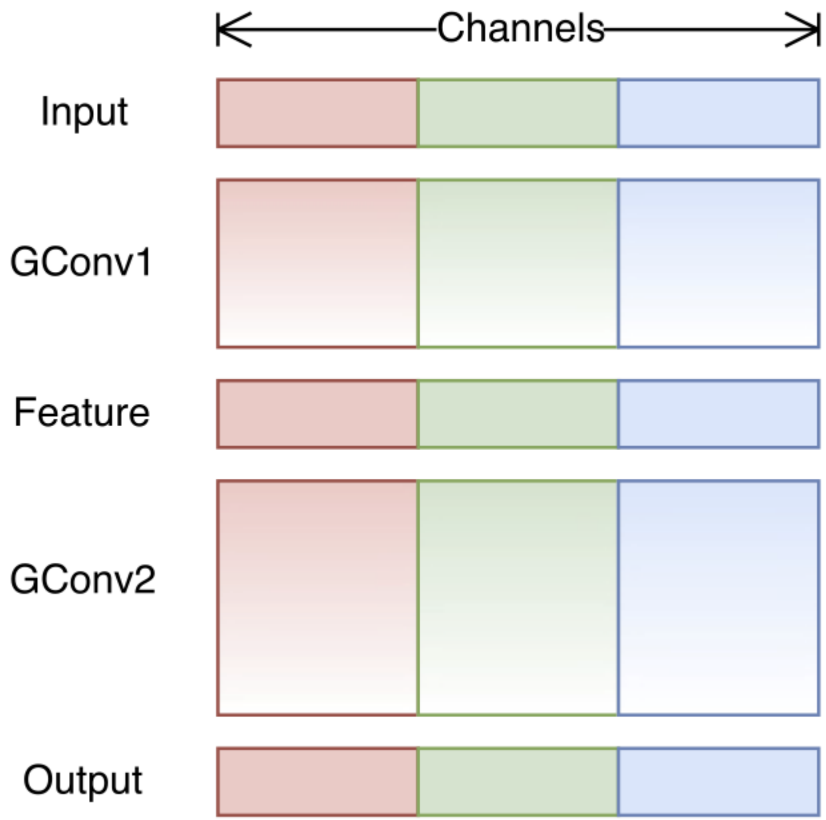
- **Kernel size**: K_H x K_W
- **Number filters**: C_{out}
- **Padding**: P
- **Stride**: S
- **Groups:** G

Weight matrix: C_{out} /G x C_{in} /G x K_H x K_W x G

Bias vector: C_{out} /G

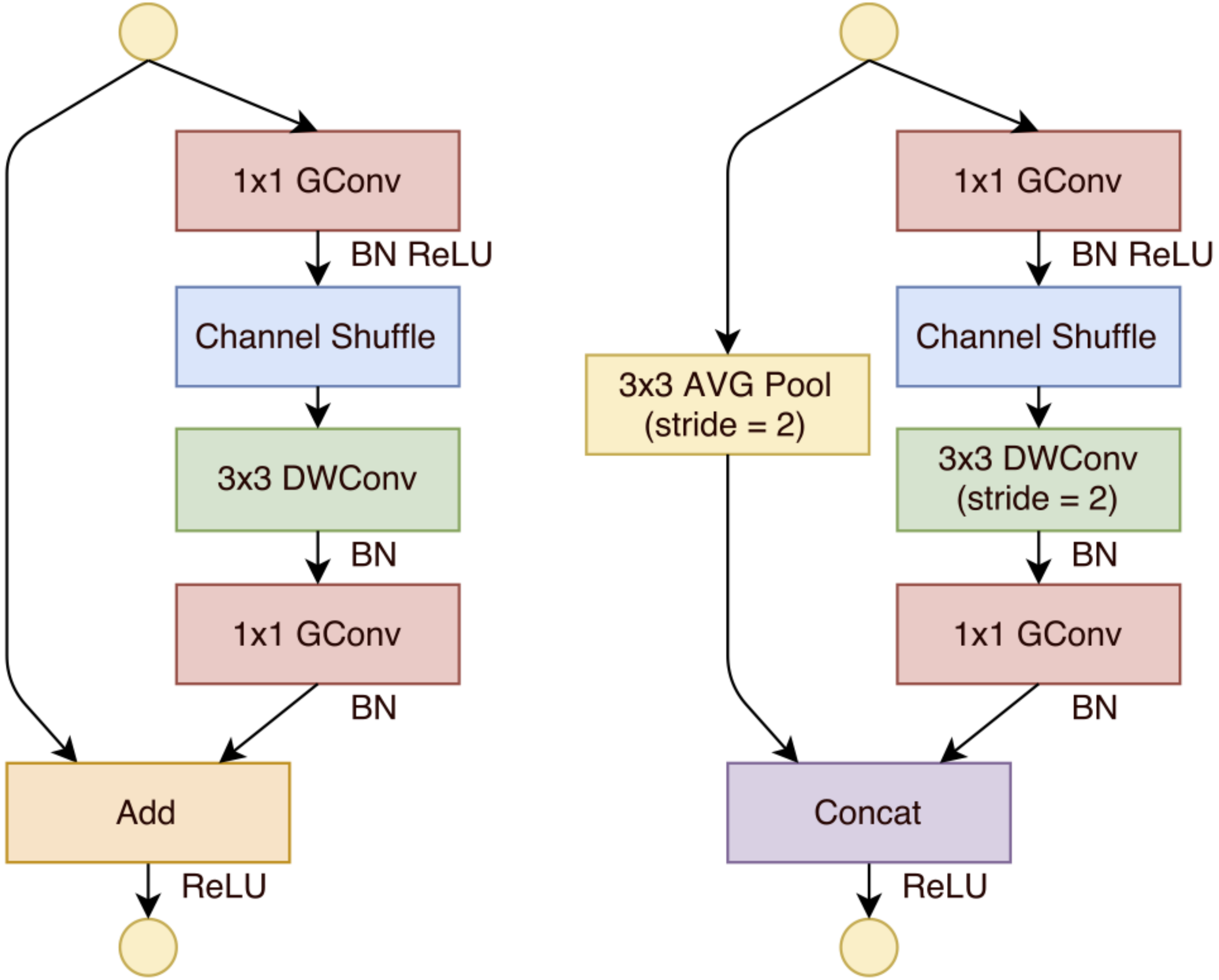
FLOPS: C_{out} /G x C_{in} /G x K_H x K_W x G x H x W

ShuffleNet



[Zhang et al., ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. CVPR 2018]

ShuffleNet Units

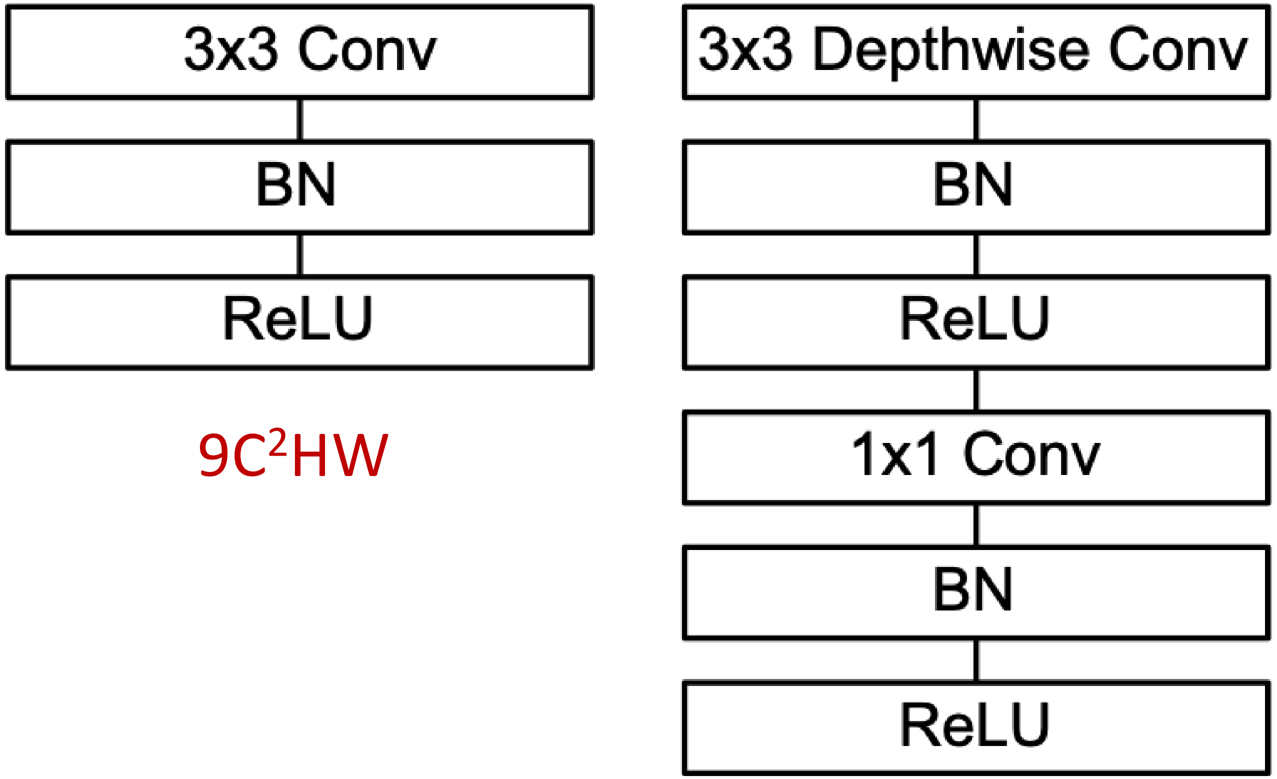


[Zhang et al., ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. CVPR 2018]

Today’s Class

- Lightweight convolutional neural networks
- Tips of training deep convolutional neural networks
- PyTorch tutorial

MobileNet



Channel: C, Groups: C

$9CHW$

$9C^2HW$

Pointwise Conv

C^2HW

Computation reduction: $9C^2HW/(9CHW + C^2HW) = 9C/(9+C)$

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

Training Convolutional Networks

1. Download big datasets
2. Design CNN architecture
3. Initialize Weights
4. For t = 1 to T:
    1. Form minibatch
    2. Compute loss + gradient
    3. Update Weights
5. Apply trained model to task

Slide credit: D Fouhey & J Johnson

Training Convolutional Networks

1. Download big datasets
2. Design CNN architecture
3. Initialize Weights
4. For t = 1 to T:
    1. Form minibatch
    2. Compute loss + gradient
    3. Update Weights
5. Apply trained model to task

Slide credit: D Fouhey & J Johnson

Training Convolutional Networks

1. Download big datasets
2. Design CNN architecture
3. Initialize Weights
4. For t = 1 to T:
    1. Form minibatch
    2. Compute loss + gradient
    3. Update Weights
5. Apply trained model to task

If the model is big, won't we overfit?

Regularizing CNNs: Weight Decay

$$L_{reg} = \frac{1}{2}\sum\nolimits_{\ell} \|W_\ell\|^2 \qquad \frac{\partial L_{reg}}{\partial W_\ell} = W_\ell$$

Add L2 regularization term L_{reg} to the loss penalizing large weight matrices

Usually don't regularize bias terms, or BatchNorm scale / shift params

*Technical note: Adding an explicit term to the loss is "L2 Regularization"; "Weight decay" adds a term to the gradient. Theyare equivalent for SGD, but not quite the same for other optimizers like Adam

Slide credit: D Fouhey & J Johnson

Regularizing CNNs: Data Augmentation

Hippo

Hippo?

Horizontal Flip

Hippo?

Color Jitter

Hippo?

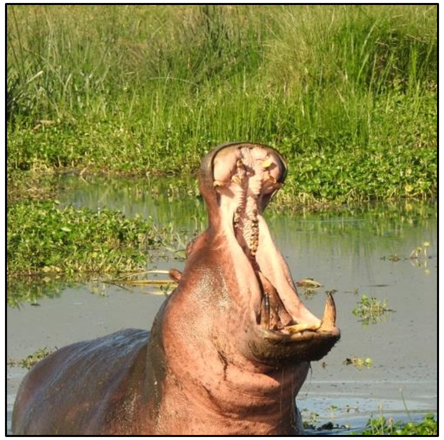


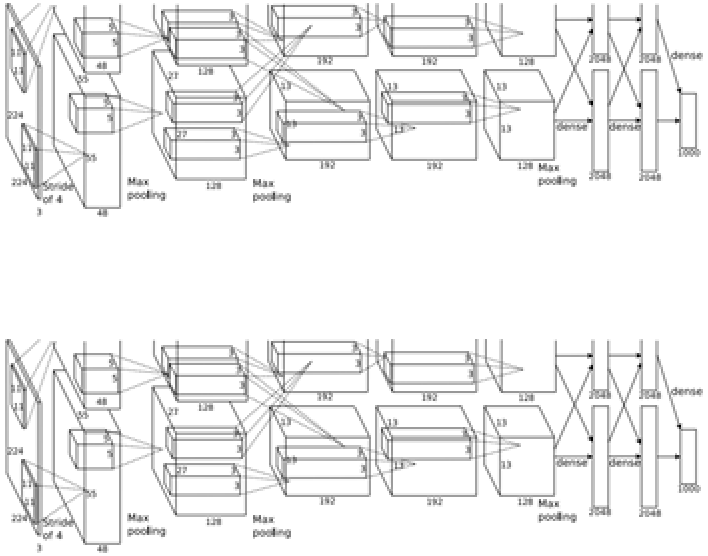
Image Cropping

Regularizing CNNs: Data Augmentation

Apply random transformations to input images during training
Artificially “inflate” the size of your dataset

Training
Image

Augmented
images



Hippo

Hippo

Slide credit: D Fouhey & J Johnson

Training Convolutional Networks

1. Download big datasets
2. Design CNN architecture
3. Initialize Weights
4. For t = 1 to T:
    1. Form minibatch
    2. Compute loss + gradient
    3. Update Weights
5. Apply trained model to task

If the model is big, won't we overfit?

Slide credit: D Fouhey & J Johnson

Training Convolutional Networks

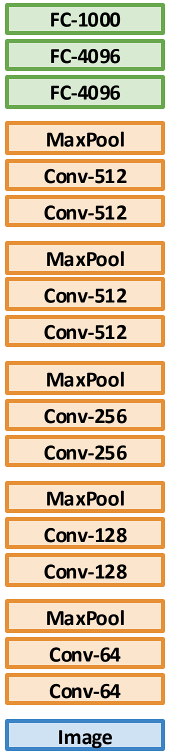
1. Download big datasets
2. Design CNN architecture
3. Initialize Weights
4. For t = 1 to T:
    1. Form minibatch
    2. Compute loss + gradient
    3. Update Weights
5. Apply trained model to task

What if we can't find one?

Slide credit: D Fouhey & J Johnson

Transfer Learning: Feature Extraction

1. Train on ImageNet



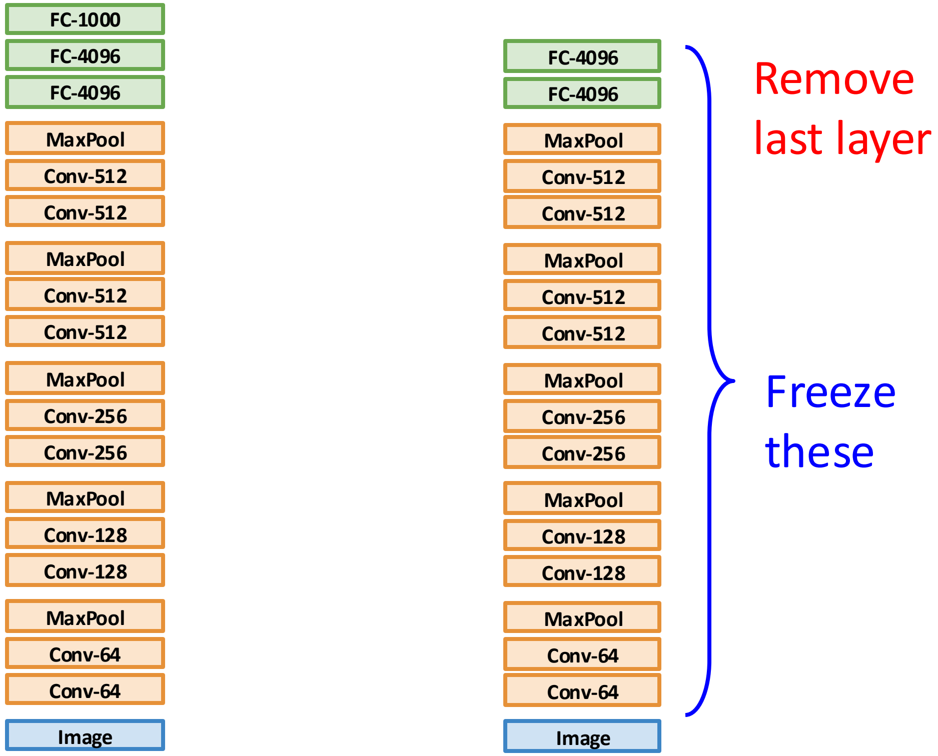
Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

Slide credit: D Fouhey & J Johnson

Transfer Learning: Feature Extraction

1. Train on ImageNet

2. CNN as feature extractor



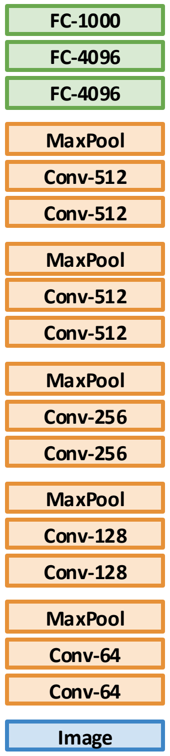
Use your small dataset to train a **linear classifier** on top of pretrained CNN features

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

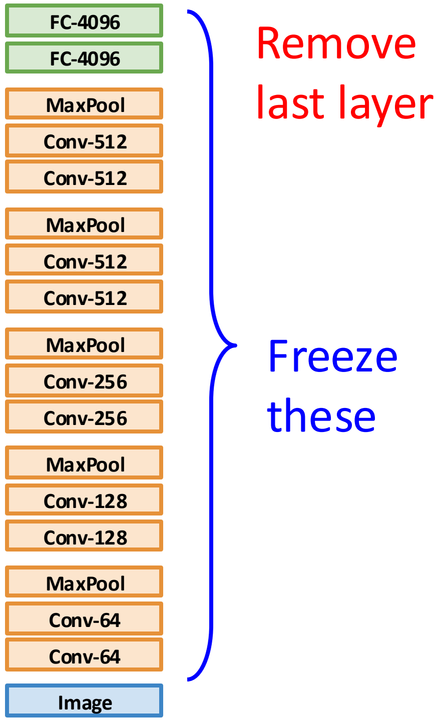
Slide credit: D Fouhey & J Johnson

Transfer Learning: Fine-Tuning

1. Train on ImageNet

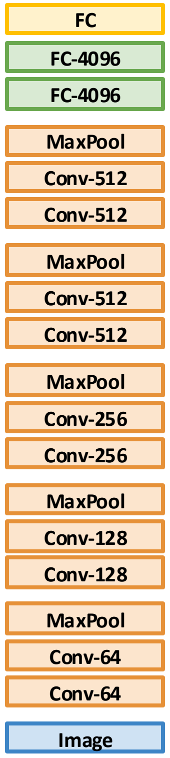


2. CNN as feature extractor



3. Bigger dataset: **Fine-Tuning**

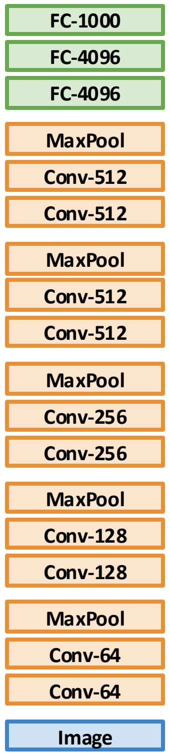
Reinitialize last layer and continue training whole network on your dataset



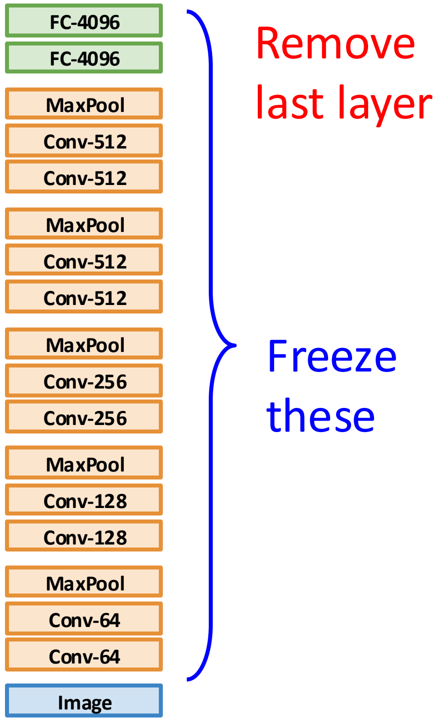
Slide credit: D Fouhey & J Johnson

Transfer Learning: Fine-Tuning

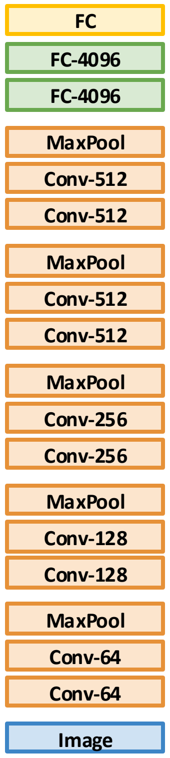
1. Train on ImageNet



2. CNN as feature extractor



3. Bigger dataset: **Fine-Tuning**



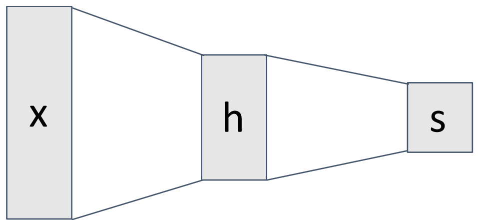
Reinitialize last layer and continue training whole network on your dataset

Some tricks:
- Train with feature extraction first before fine-tuning
- Lower the learning rate: use ~1/10 of LR used in original training
- Sometimes freeze lower layers to save computation

Slide credit: D Fouhey & J Johnson

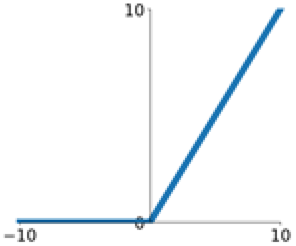
Recap: Convolutional Networks

Fully-Connected Layers



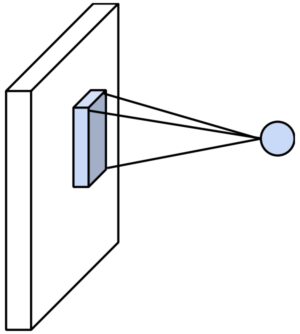
$$y = Wx + b$$

Activation Function

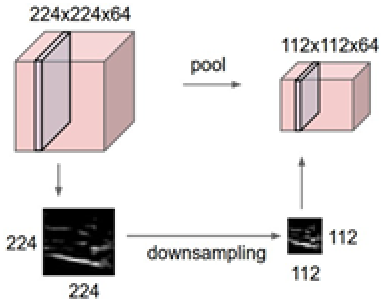


$$y = \max(0, x)$$

Convolution Layers



Pooling Layers

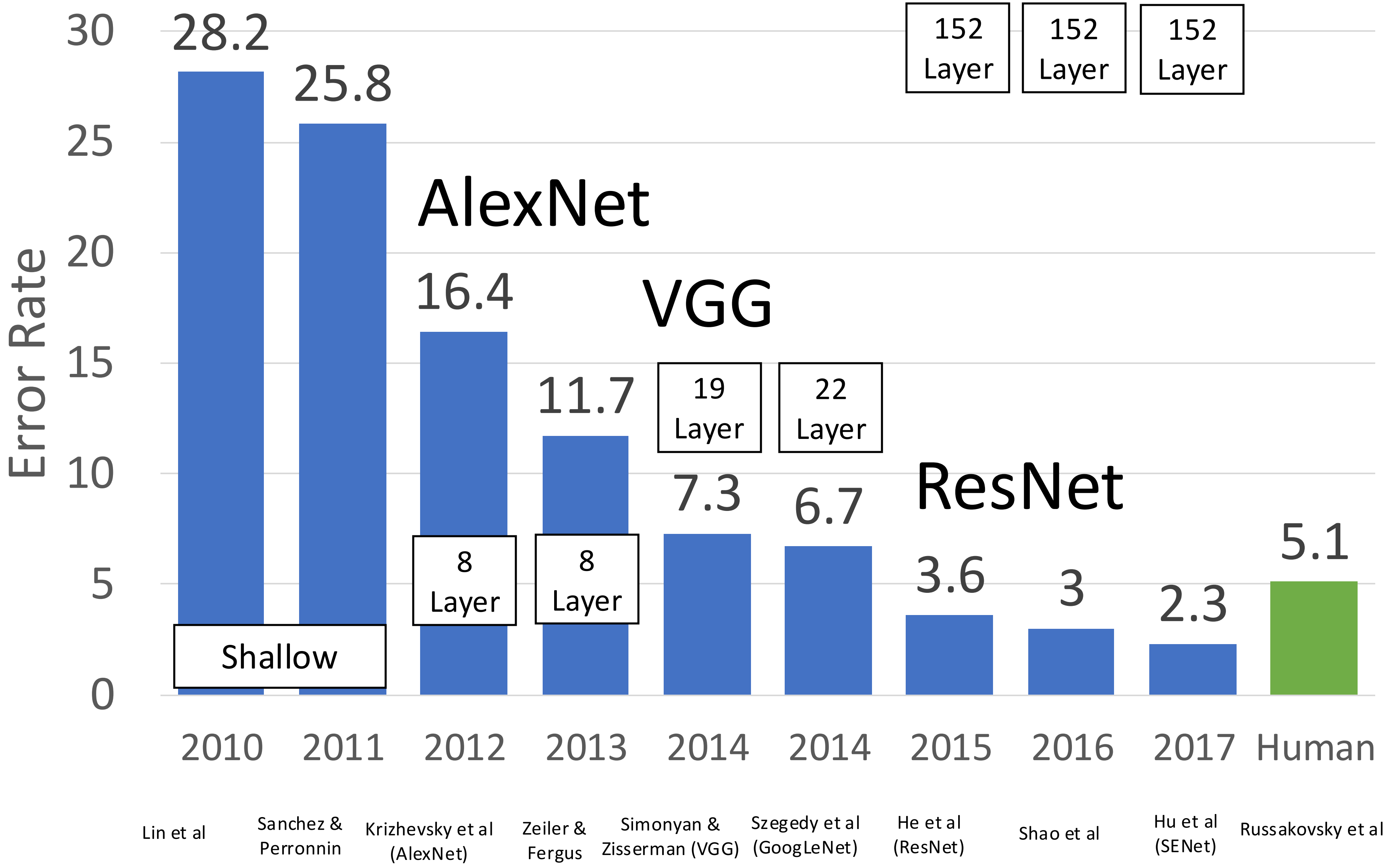


Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Slide credit: D Fouhey & J Johnson

Recap: CNN Architectures



Slide credit: D Fouhey & J Johnson

Recap: Training CNNs

1. Download big datasets   Transfer Learning
2. Design CNN architecture
3. Initialize Weights   Xavier / MSRA Init
4. For t = 1 to T:
    1. Form minibatch
    2. Compute loss + gradient
    3. Update Weights
5. Apply trained model to task

Regularization + Data Augmentation

Slide credit: D Fouhey & J Johnson

Next Class

More about

Convolutional Neural Networks