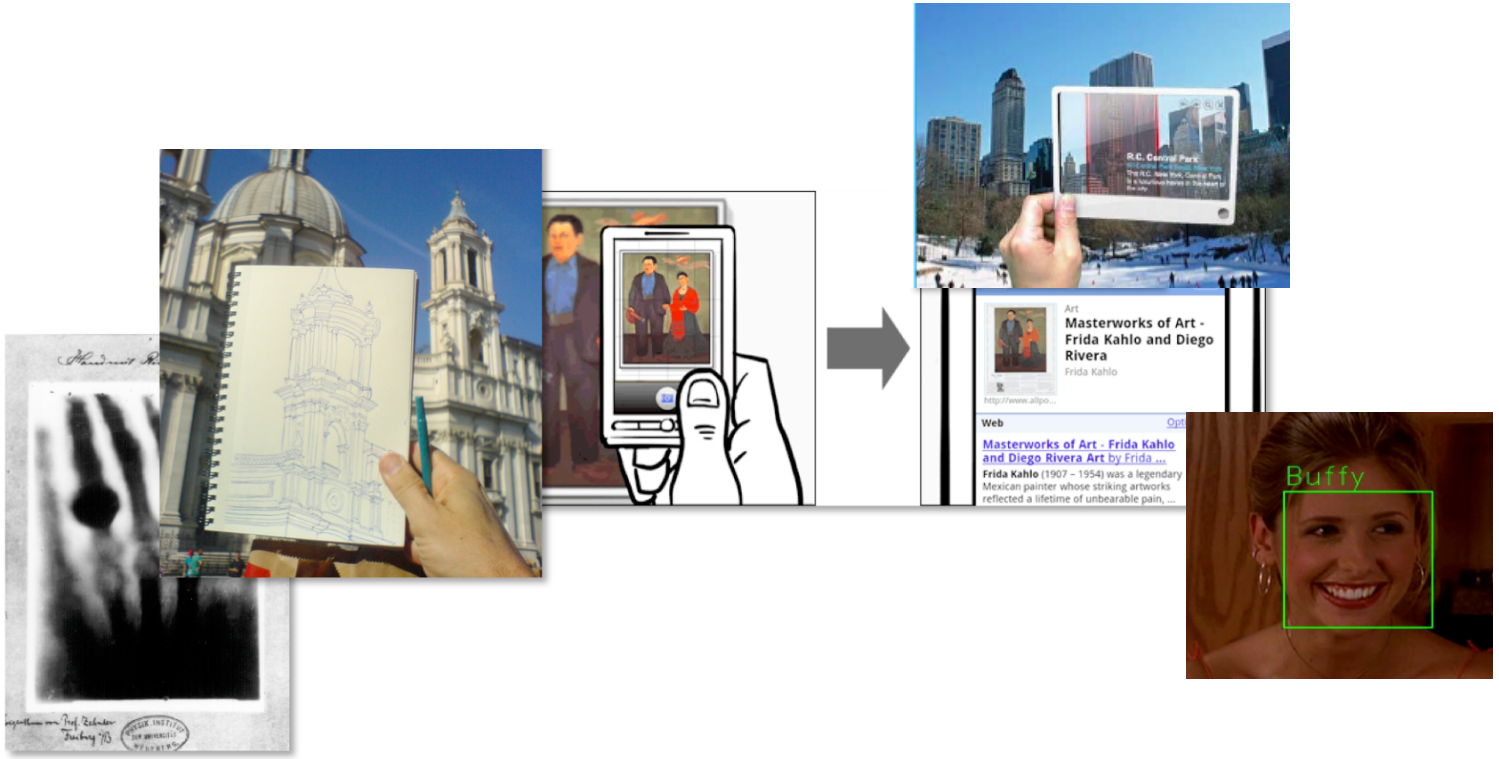


Deep (3)

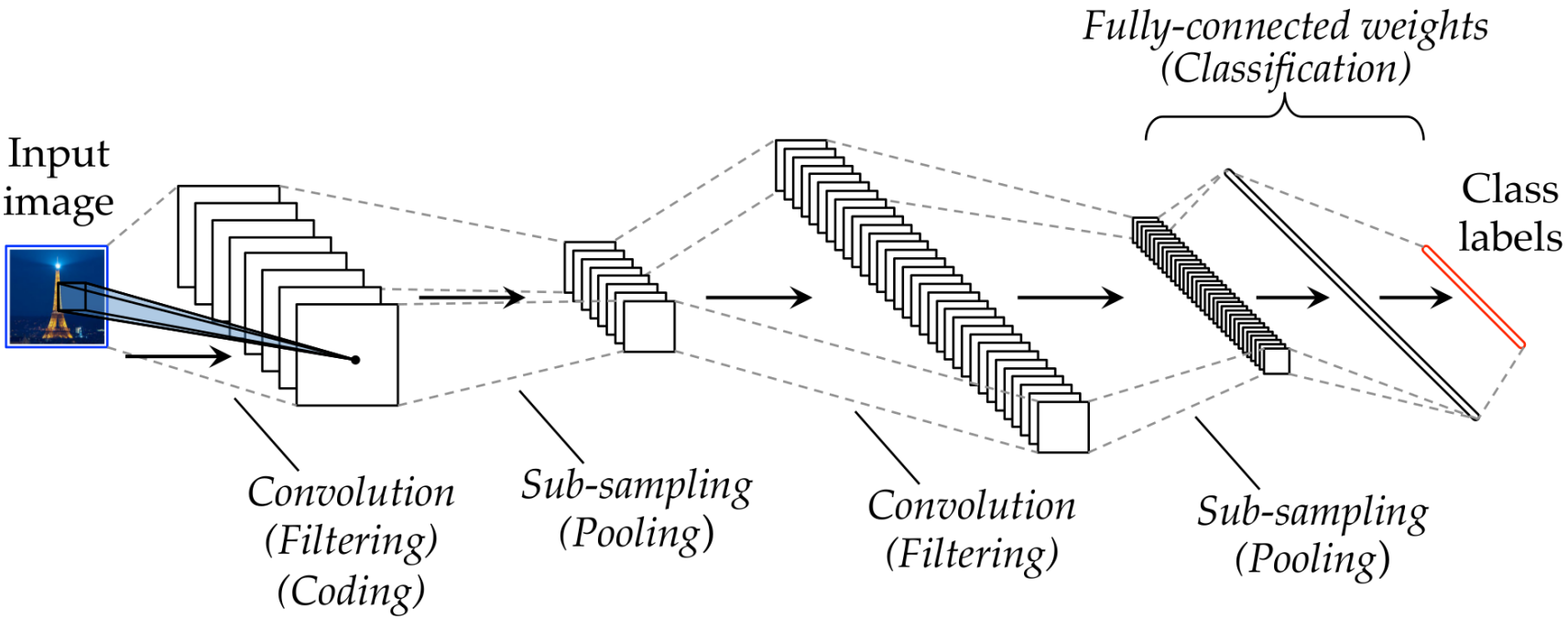
Matthieu Cord LIP6 / SU

Outline

ConvNets as Deep Neural Networks for Vision

1. Neural Nets
2. Deep Convolutional Neural Networks
3. **Modern Deep Architectures**
   - **The big picture**

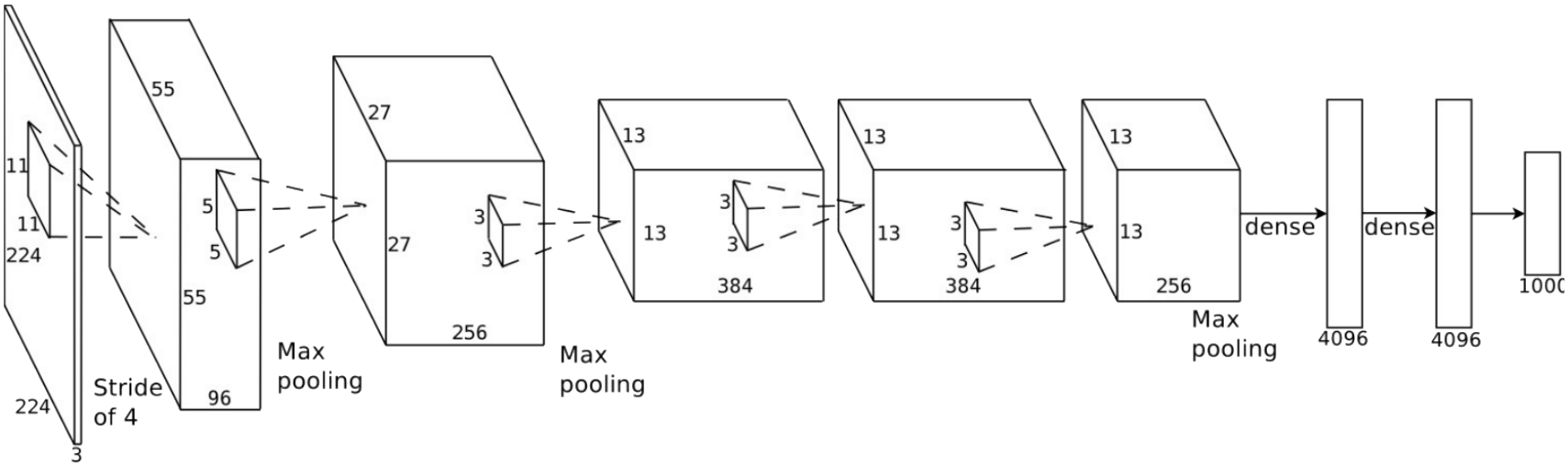
Deep Convolutional Neural Networks (Deep ConvNets)



- **Convolution** uses local weights shared across the whole image
- **Pooling** shrinks the spatial dimensions
- **Activation** (ReLU, Sigmoid, Tanh,..)
- **Fully connected**

Deep ConvNets for image classification

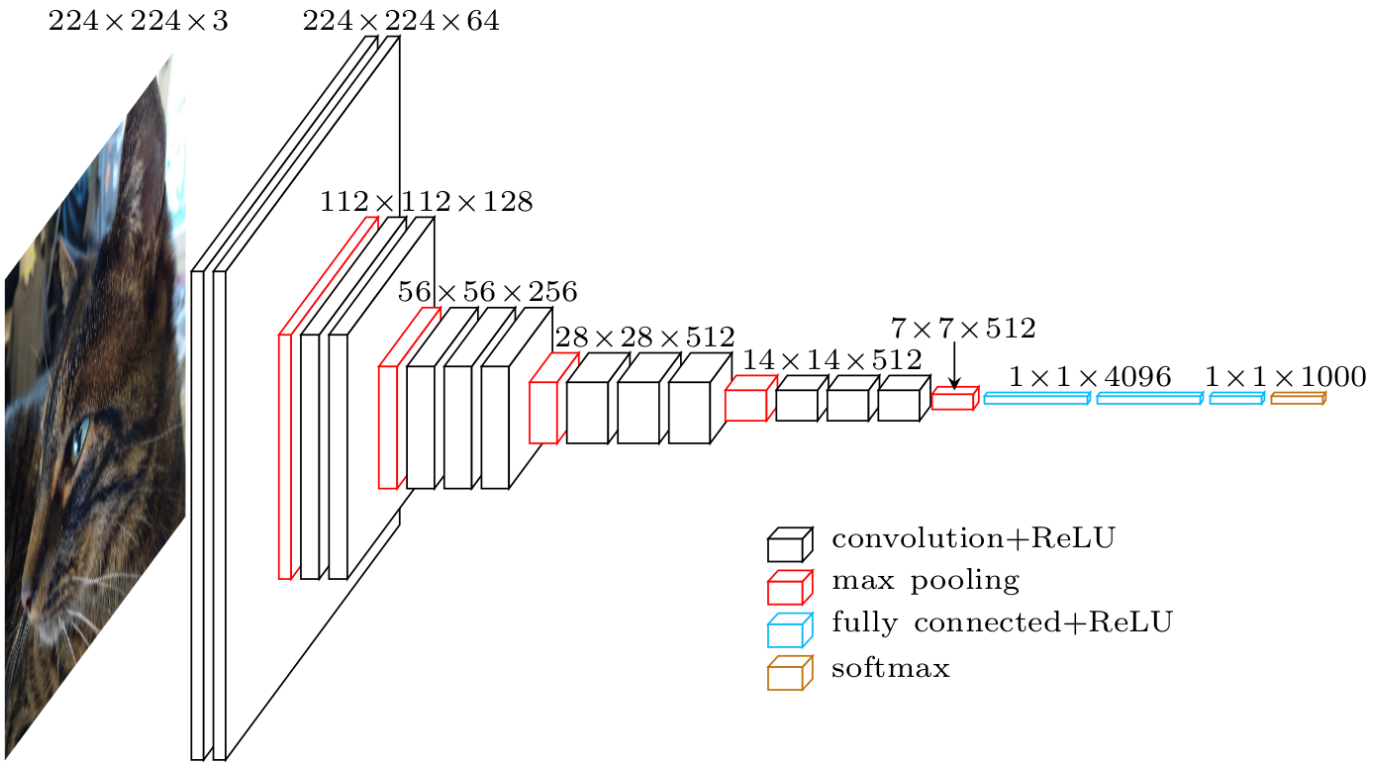
- **AlexNet** 8 layers, 62M parameters



Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton
ImageNet Classification with Deep Convolutional Neural Networks.
In *NIPS*, 2012.

Deep ConvNets for image classification

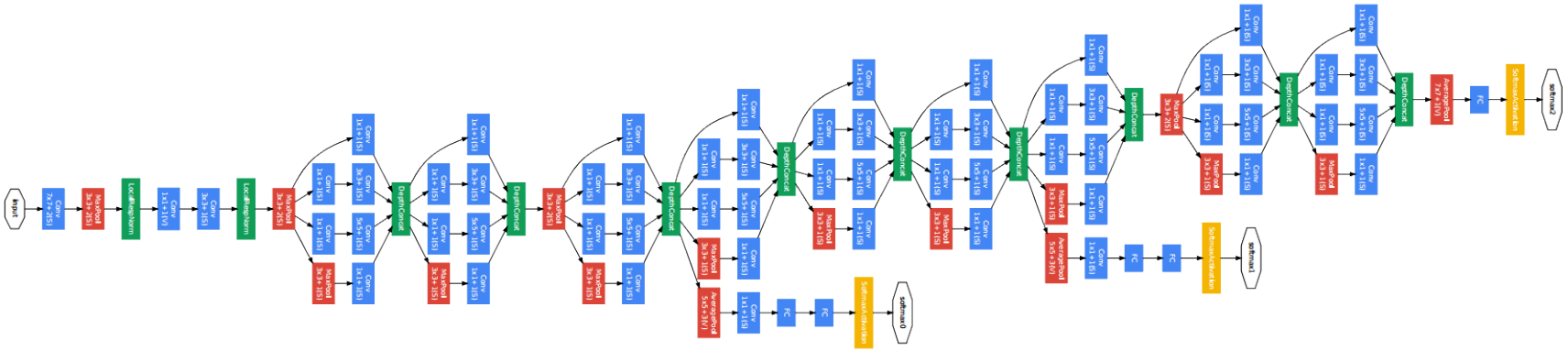
- **VGG16** (very deep) 16 layers, 138M parameters
  - Increasing the depth



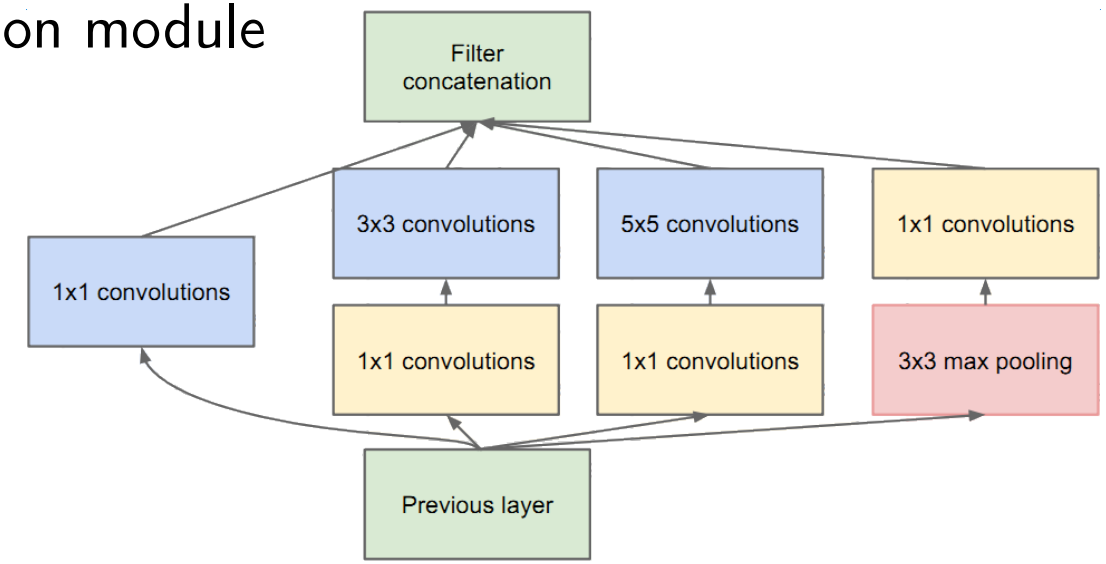
Karen Simonyan and Andrew Zisserman
Very Deep Convolutional Networks for Large-Scale Image Recognition.
In *ICLR*, 2015.

Deep ConvNets for image classification

- **GoogLeNet** / **Inception** 22 layers, 7M parameters



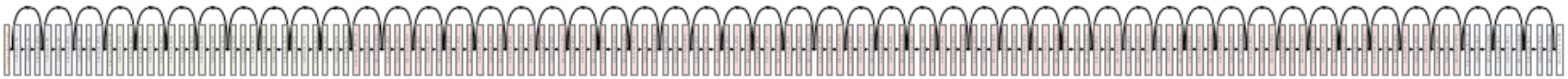
- Inception module

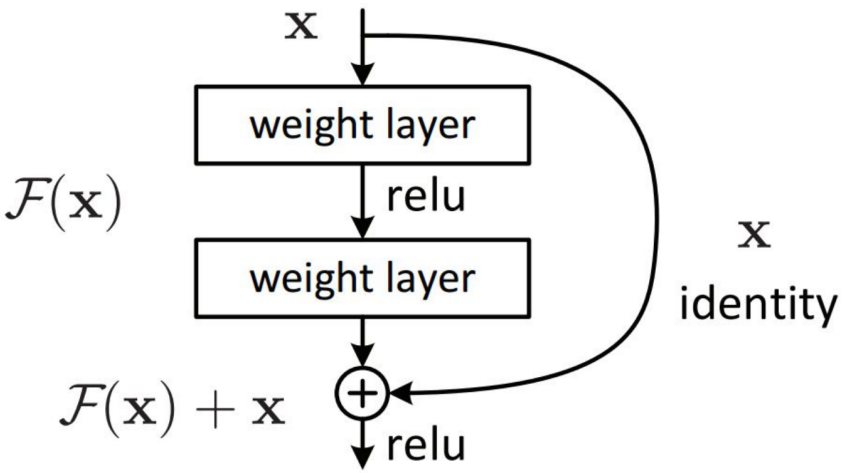


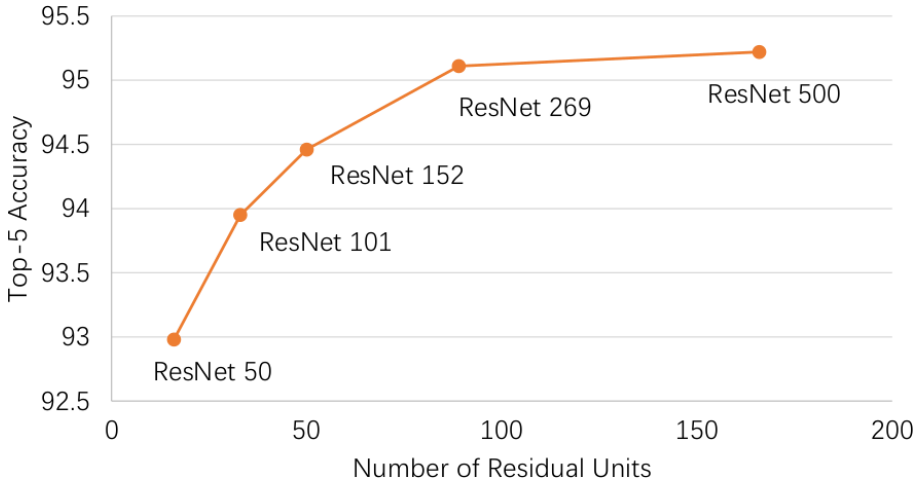
Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke and Rabinovich
Going Deeper with Convolutions.
In *CVPR*, 2015.

Deep ConvNets for image classification

- **ResNet** 152 layers, 60M parameters

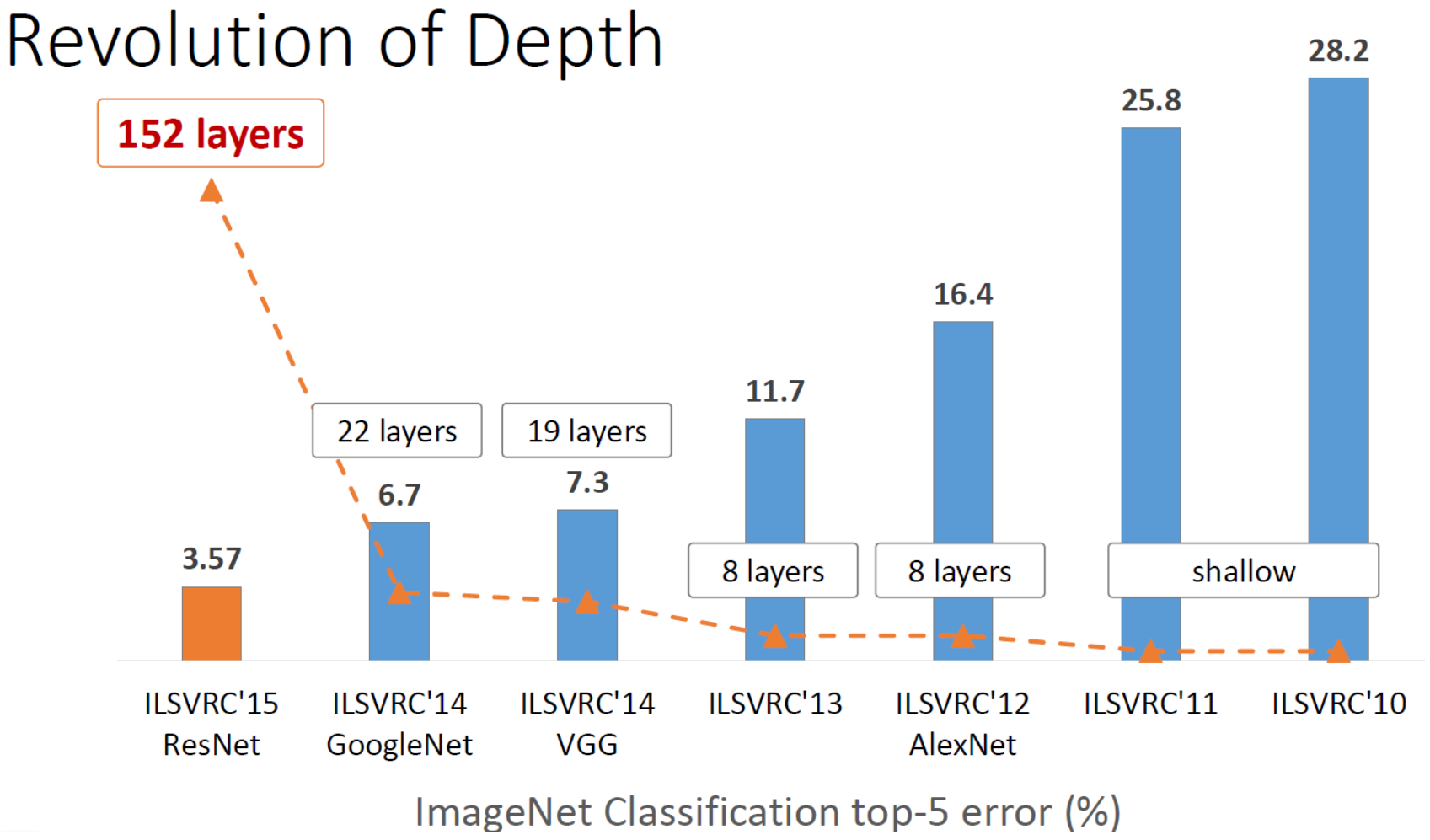






Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun
Deep Residual Learning for Image Recognition.
In *CVPR*, 2016.

Deep ConvNets for image classification



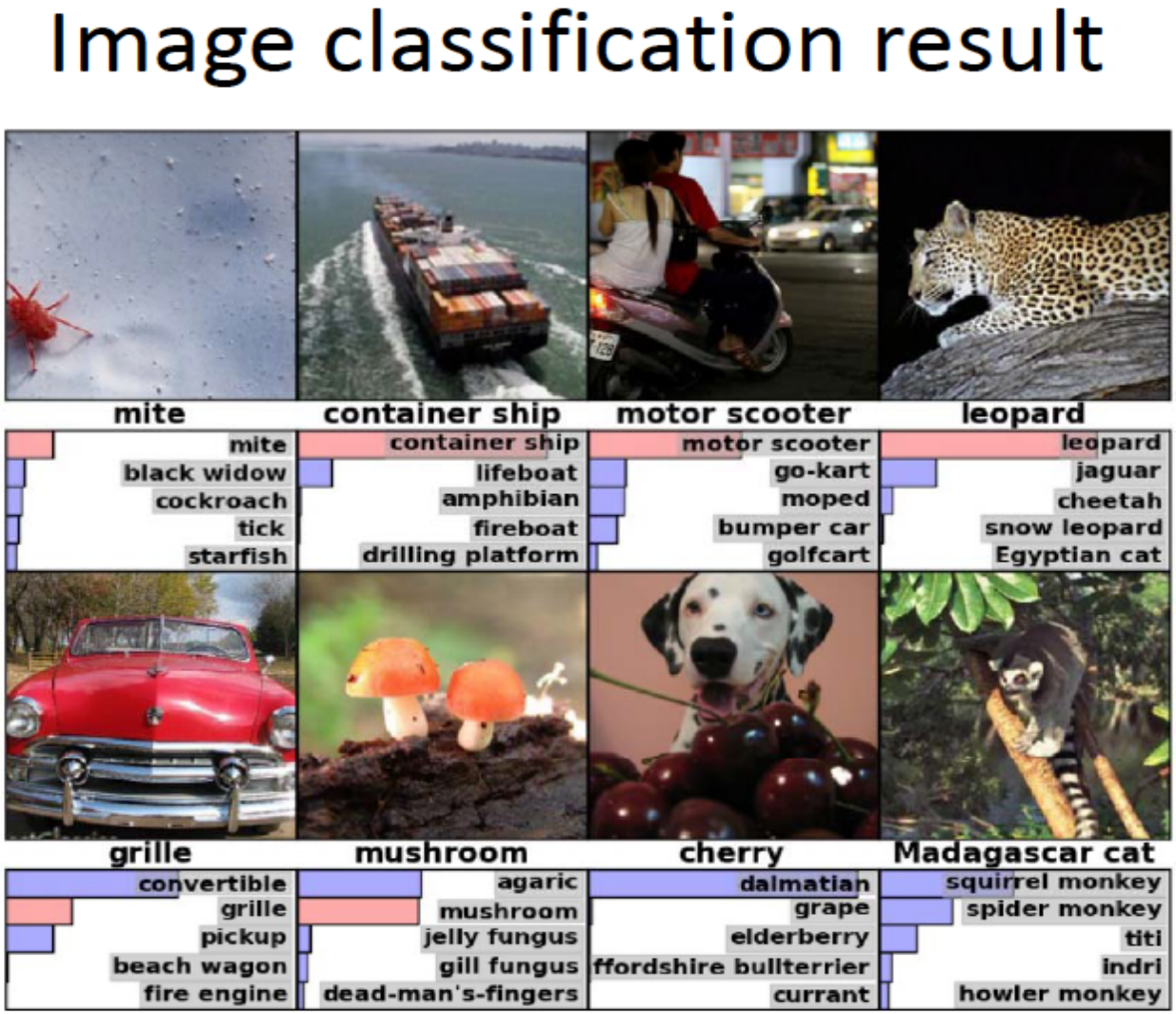
Outline

ConvNets as Deep Neural Networks for Vision

1. Neural Nets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
   - The big picture
   - **AlexNet: 2012 the (deep) revolution**
     - **Archi and learning**
     - **Ablation study**

ImageNet 2012: the (deep) revolution

- 1.2 million labeled images
- 1000 classes
- Mono-class
- TOP5

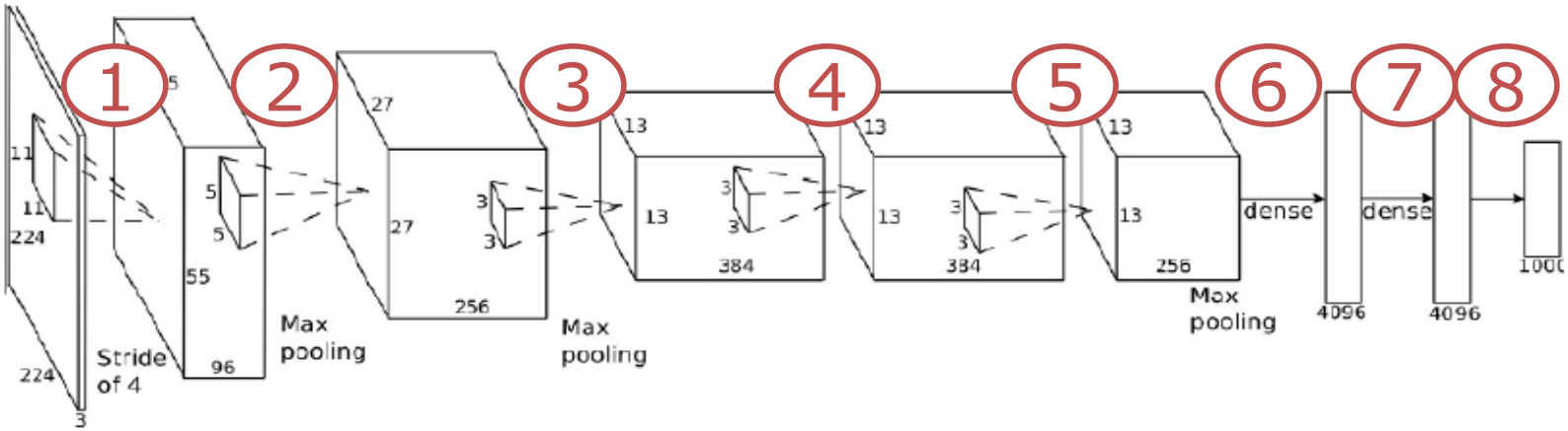


Architecture of the IMAGENET Challenge 2012 Winner:

A Large CNN [@Fergus NIPS 2013]

Krizhevsky et al. [NIPS2012]

- Same model as LeCun'98 but:
  - Bigger model (8 layers)
  - More data (10^6 vs 10^3 images)
  - GPU implementation (50x speedup over CPU)
  - Better regularization (DropOut)

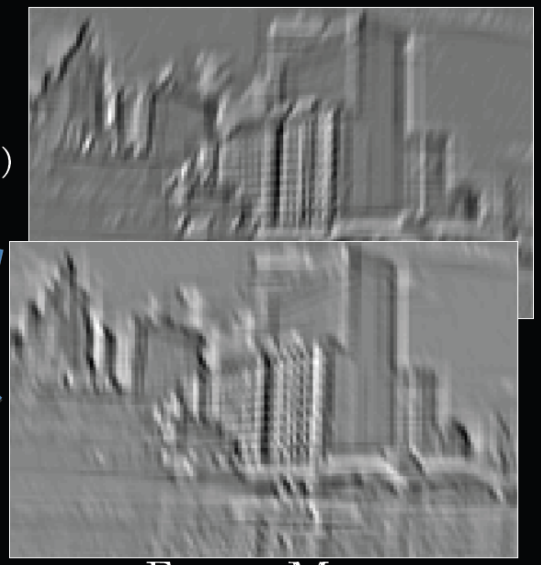


- 7 hidden layers, 650,000 neurons, 60,000,000 parameters
- Trained on 2 GPUs for a week

Filtering

- Convolutional
  - Dependencies are local
  - Translation equivariance
  - Tied filter weights (few params)
  - Stride 1,2,… (faster, less mem.)

Input

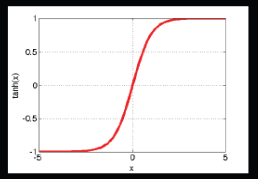


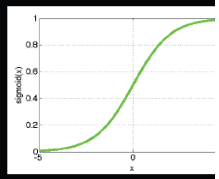
Feature Map

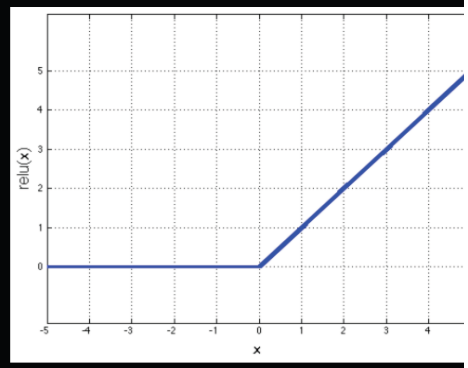
Non-Linearity

- Non-linearity
  - Per-feature independent
  - Tanh
  - Sigmoid: 1/(1+exp(-x))
  - Rectified linear
    - Simplifies backprop
    - Makes learning faster
    - Avoids saturation issues

→ Preferred option

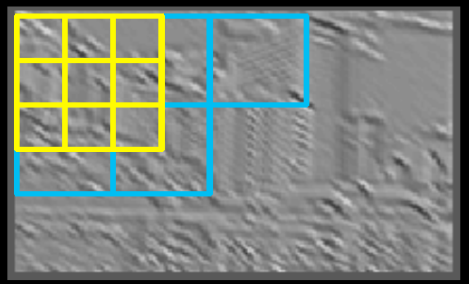




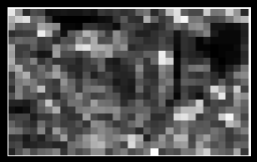


Pooling

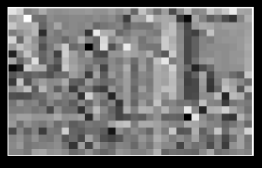
- Spatial Pooling
  - Non-overlapping / overlapping regions
  - Sum or max
  - Boureau et al. ICML'10 for theoretical analysis



Max

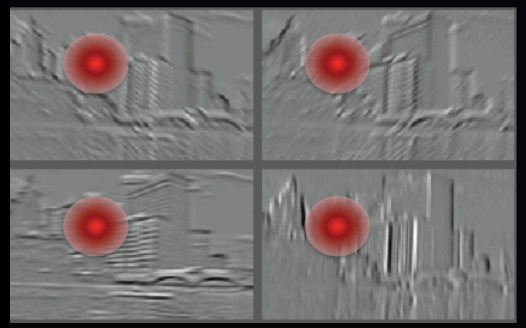


Sum

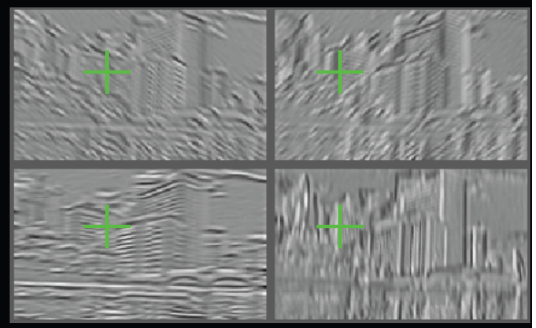


Normalization

- Contrast normalization (between/across feature maps)
  - Local mean = 0, local std. = 1, "Local" → 7x7 Gaussian
  - Equalizes the features maps



Feature Maps



Feature Maps
After Contrast Normalization

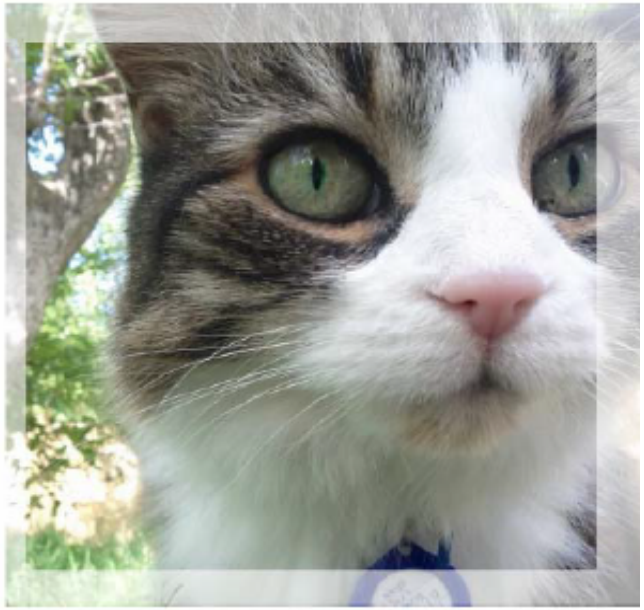
Learning the deep CNN 2012

- Basics:
  - SGD, Backprop
  - Cross Validation
  - Grid search
- "New"
  - Huge computational resources (GPU)
  - Huge training set (1 million images)
  - Data augmentation - Pre-processing
  - Dropout
  - ReLu
  - *Contrast normalization*
  - And (dixit LeCun) few hacks to initialize the weights, prevent the weights from blowing up for large values of the learning rate ...

Data Augmentation

lots of jittering, mirroring, and color perturbation of the original images generated on the fly to increase the size of the training set

Crop, flip,.. in train AND in test

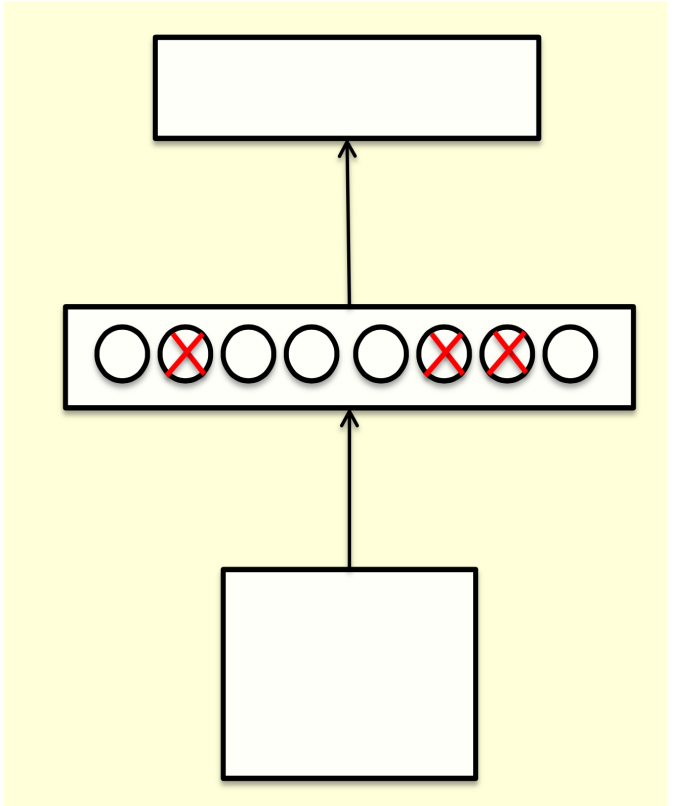


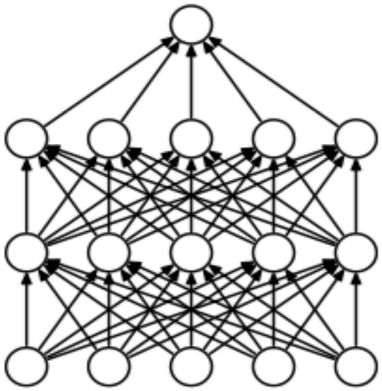
Dropout: an efficient way to average many large neural nets

For each training example, randomly omit each hidden unit with probability 0.5

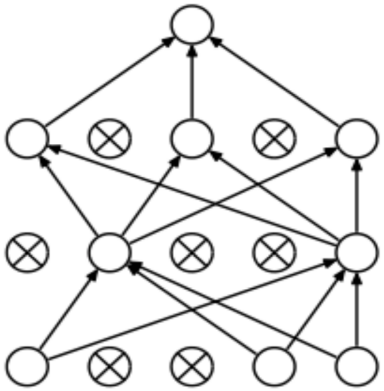
Due to sharing of weights, model strongly regularized

- Pulls the weights towards what other models want.
- Better than L2 and L1 regularization that pull weights towards zero





Standard Neural Net



After applying dropout.

@Hinton, NIPS 2012

Dropout: what do we do at test time?

Option 1:

Sample many different architectures and take the geometric mean of their output distributions

Option 2: (Faster way)

Use all the hidden units

but after **halving their outgoing weights**

Rq: In case of single hidden layer, this is equivalent to the geometric mean of the predictions of all models

For multiple layers, it's a pretty good approximation and its fast

How well does dropout work?

Significantly improve generalization:

- For very deep nets, or at least when there are huge fully connected layers (eg. AlexNet first FC layer, VGG next, …)
- Less useful for fully convolutional nets

Useful to prevent feature co-adaptation (feature only helpful when other specific features present)

Later in course (week 11)

$\Rightarrow$ Dropout as a Bayesian Approximation

$\Rightarrow$ Representing Model Uncertainty in Deep Learning

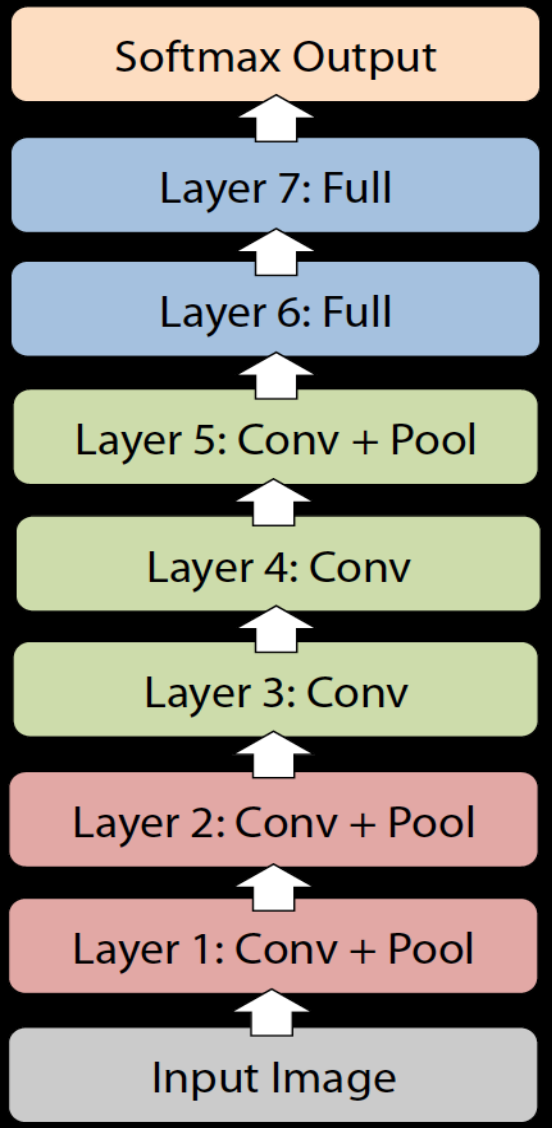
Outline

ConvNets as Deep Neural Networks for Vision

1. From MLP to ConvNets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
   - The big picture
   - AlexNet : 2012 the (deep) revolution
     - Archi and learning
     - **Ablation study**
       - **Number of layers**
       - ***Tapping off features at each layer***
       - **Transfo Robustness vs layers**

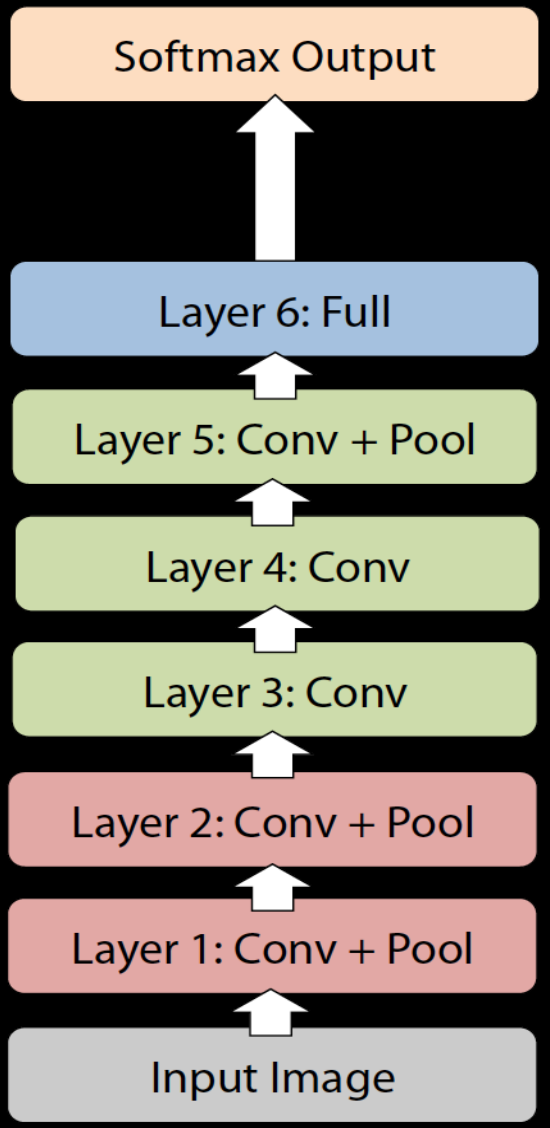
Architecture of Krizhevsky et al.

- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.2% top-5 error
- Our reimplementation:
  18.1% top-5 error



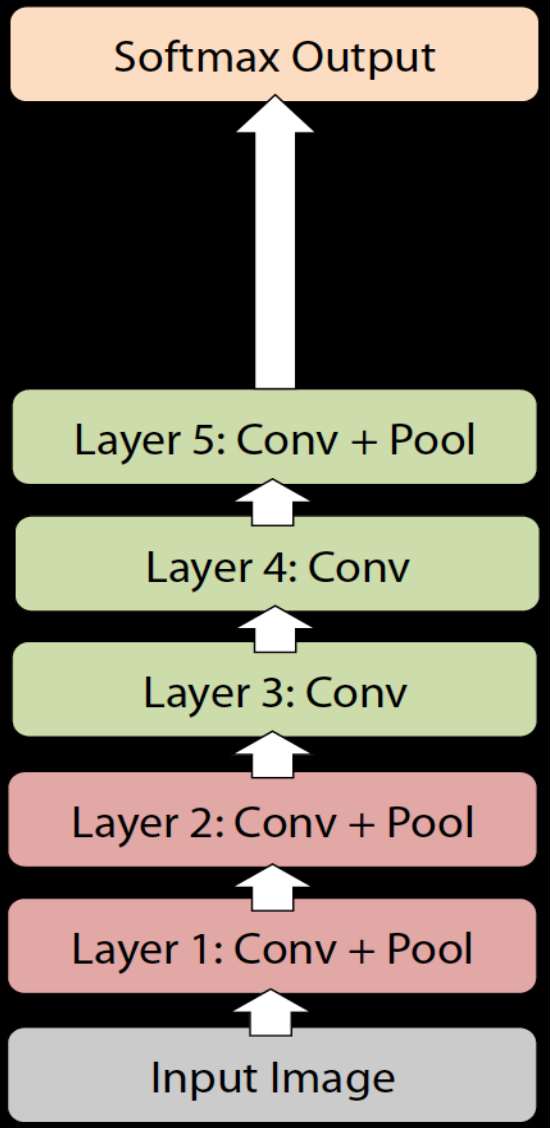
Architecture of Krizhevsky et al.

- Remove top fully connected layer
  - Layer 7
- Drop 16 million parameters
- Only 1.1% drop in performance!



Architecture of Krizhevsky et al.

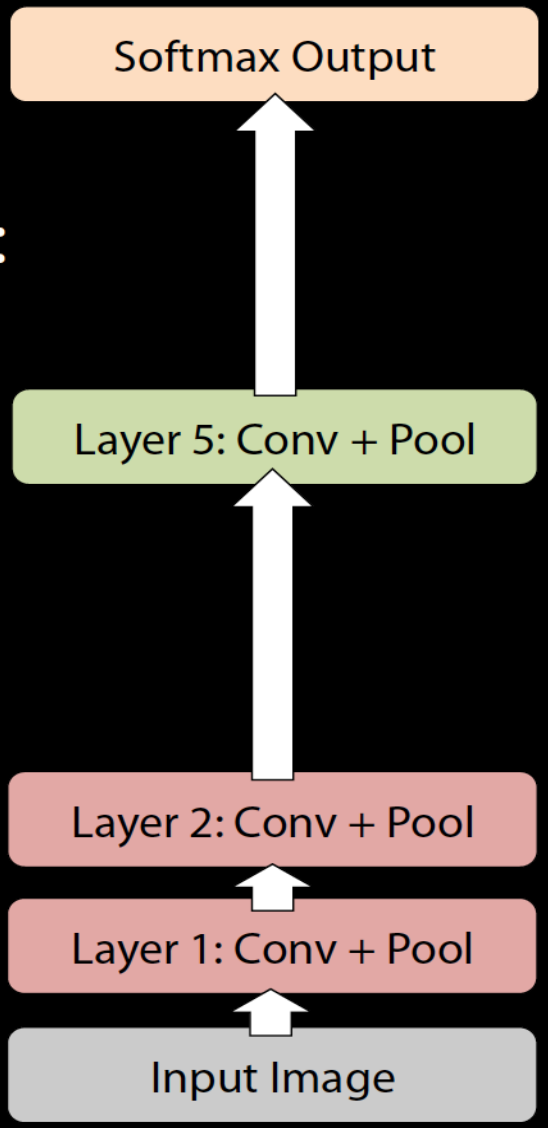
- Remove both fully connected layers
  - Layer 6 & 7
- Drop ~50 million parameters
- 5.7% drop in performance



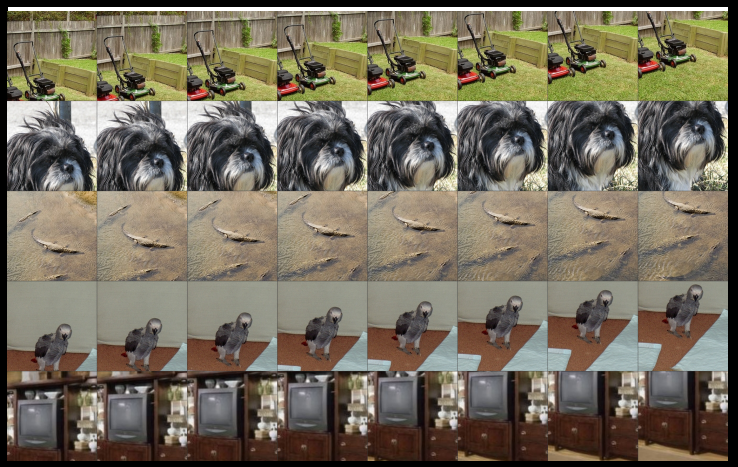
Architecture of Krizhevsky et al.

- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6 ,7
- Now only 4 layers
- 33.5% drop in performance

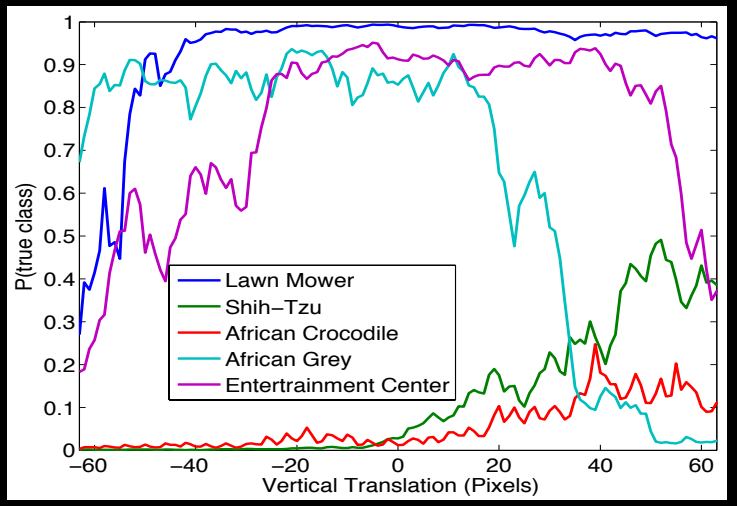
→Depth of network is key



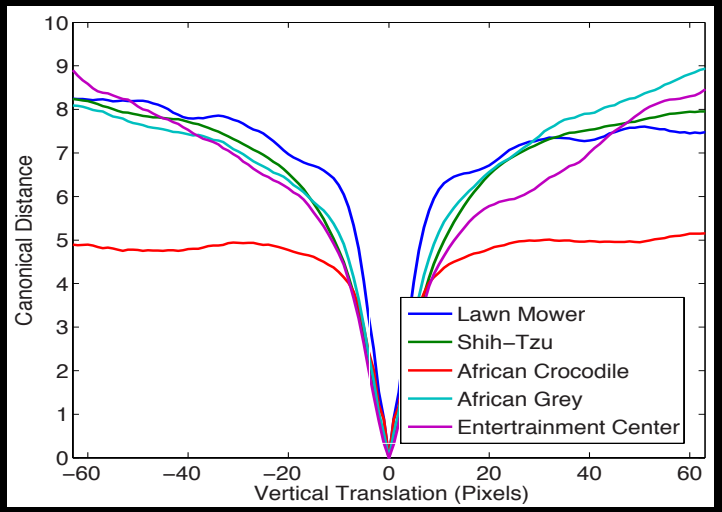
Translation (Vertical)



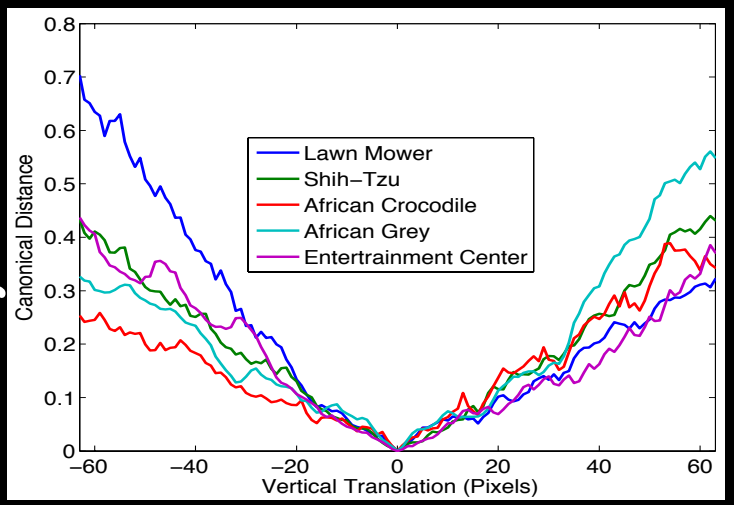
Output



Layer 1

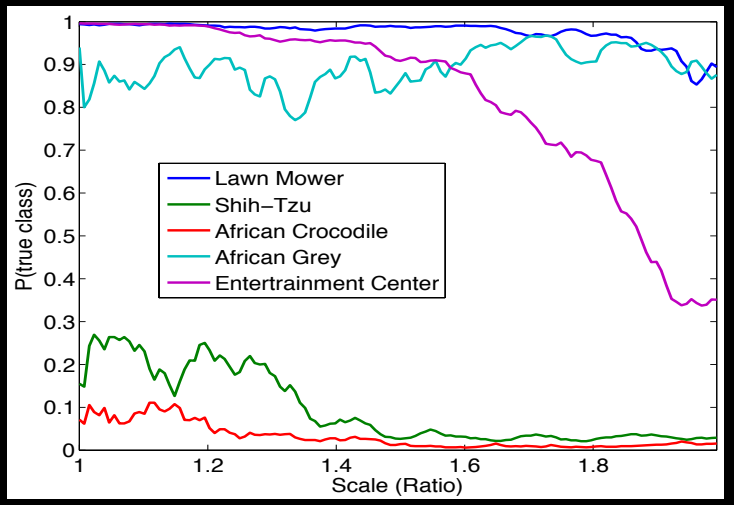


Layer 7

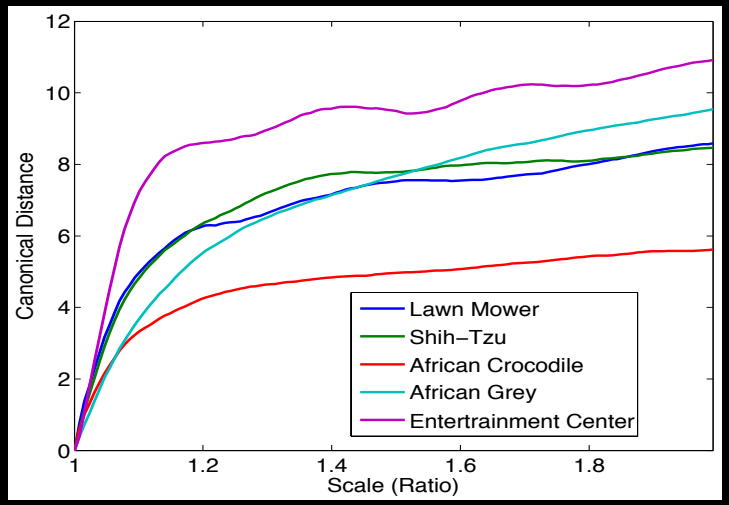


Scale Invariance

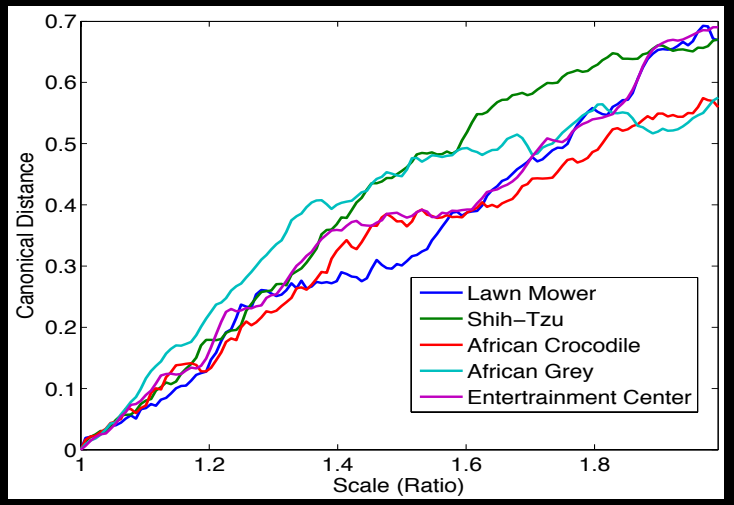
Output



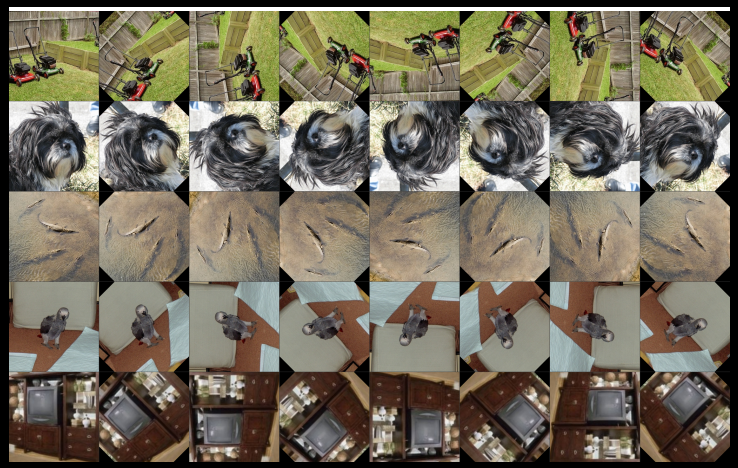
Layer 1



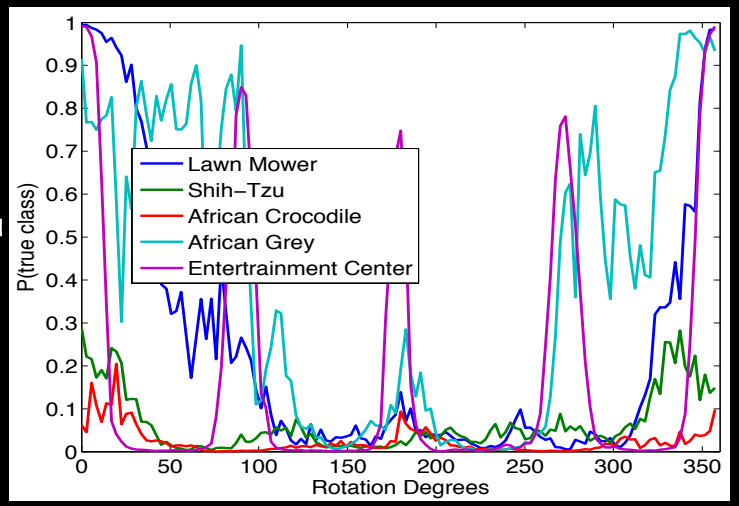
Layer 7



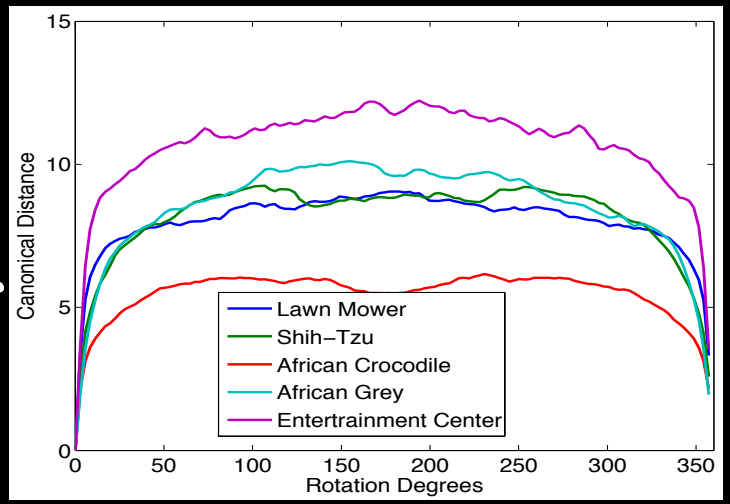
Rotation Invariance



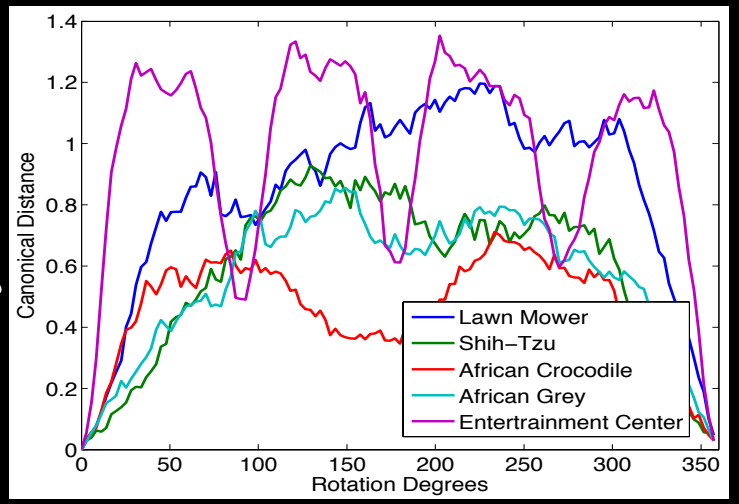
Output



Layer 1



Layer 7



What's next AlexNet?

How to improve AlexNet architecture?

+++Deep?

+++Convolutional?

+++Fully connected?

All?

⇒A lot of empirical studies

⇒Tuning various design parameters

⇒what really works?

⇒Winners: VGG, GoogleNets, ResNet

Outline

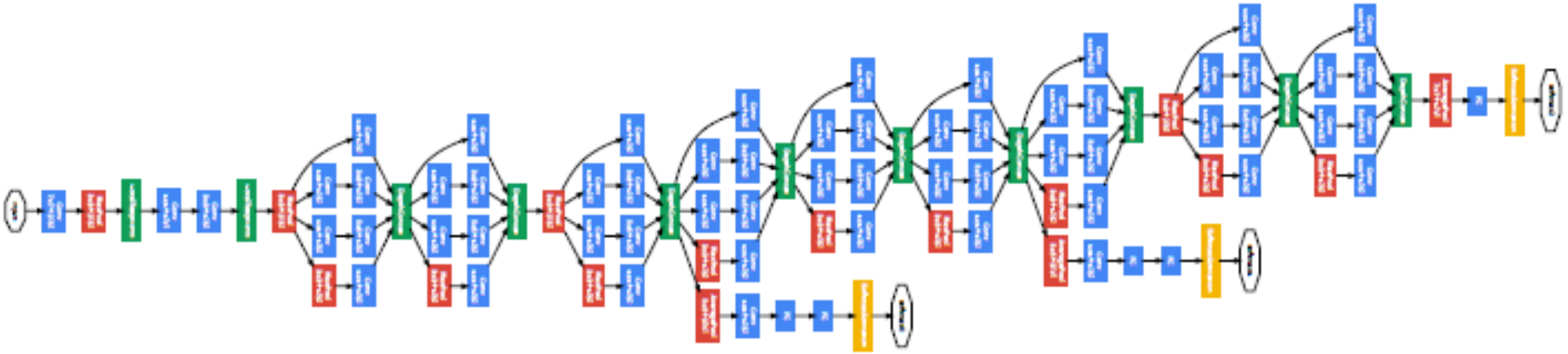
ConvNets as Deep Neural Networks for Vision

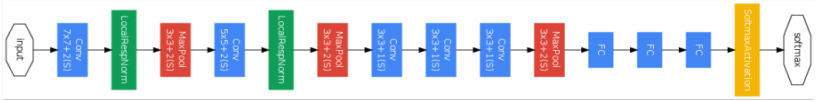
1. Neural Nets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
   - The big picture
   - AlexNet
   - **GoogLeNet**

GoogLeNet (2014)

Winner of ILSVRC -2014. Very deep network with 22 layers:

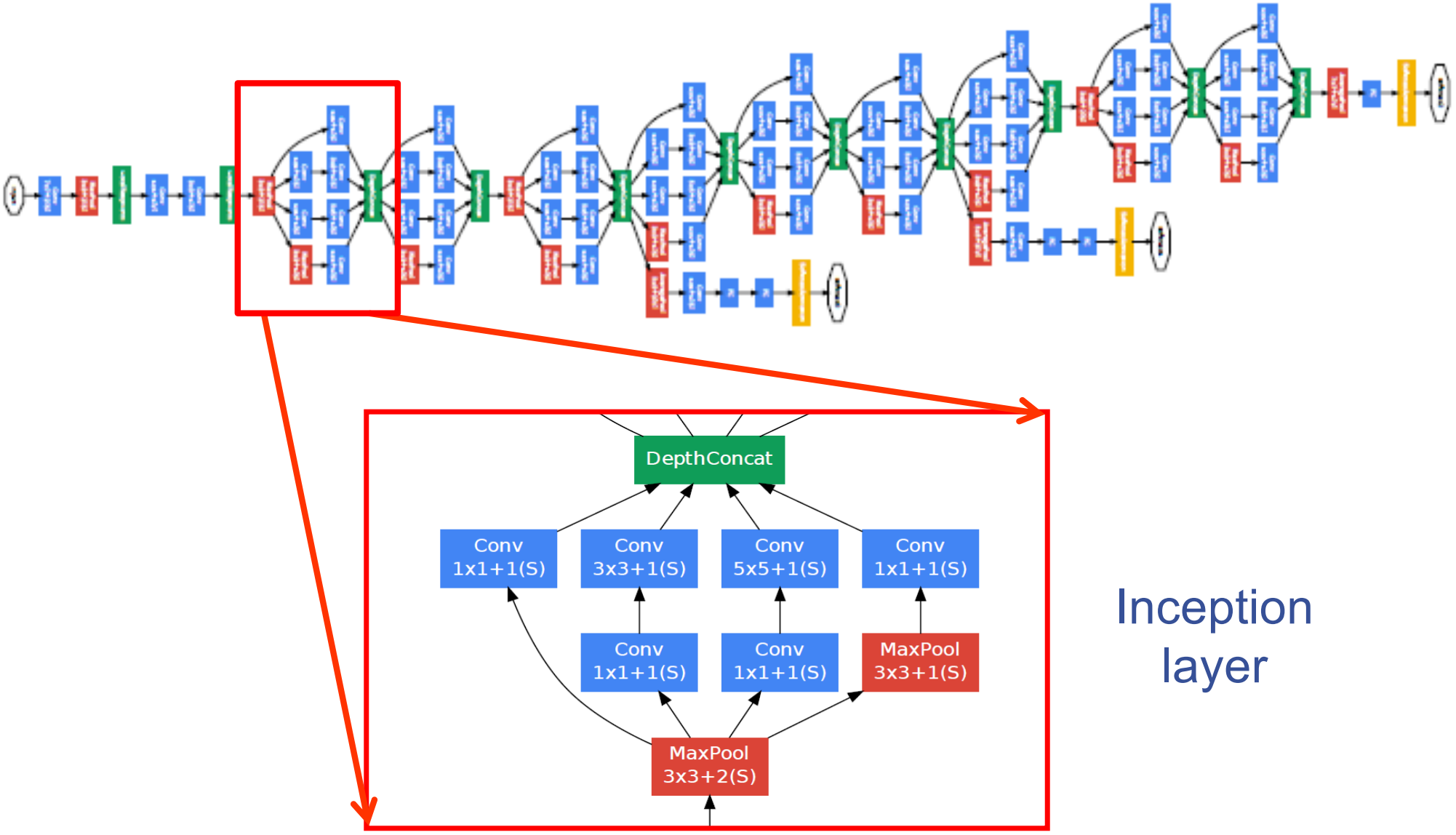
- Network–in-network–in-network
- Removed fully connected layers → small # of parameters (5M weights)





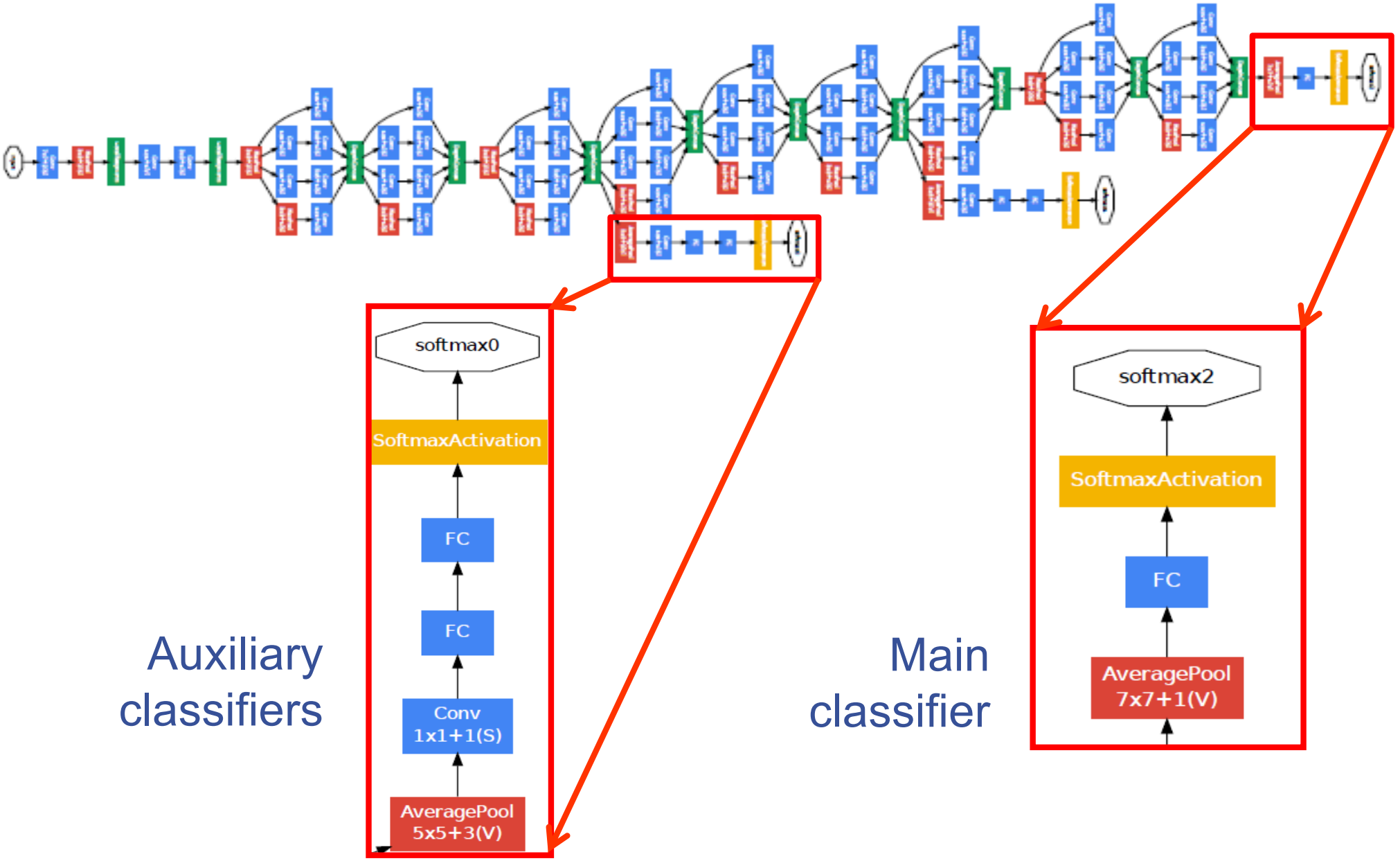
Convolution
Pooling
Softmax
Other

GoogLeNet (2014)



Inception layer

GoogLeNet (2014)



Auxiliary classifiers

Main classifier

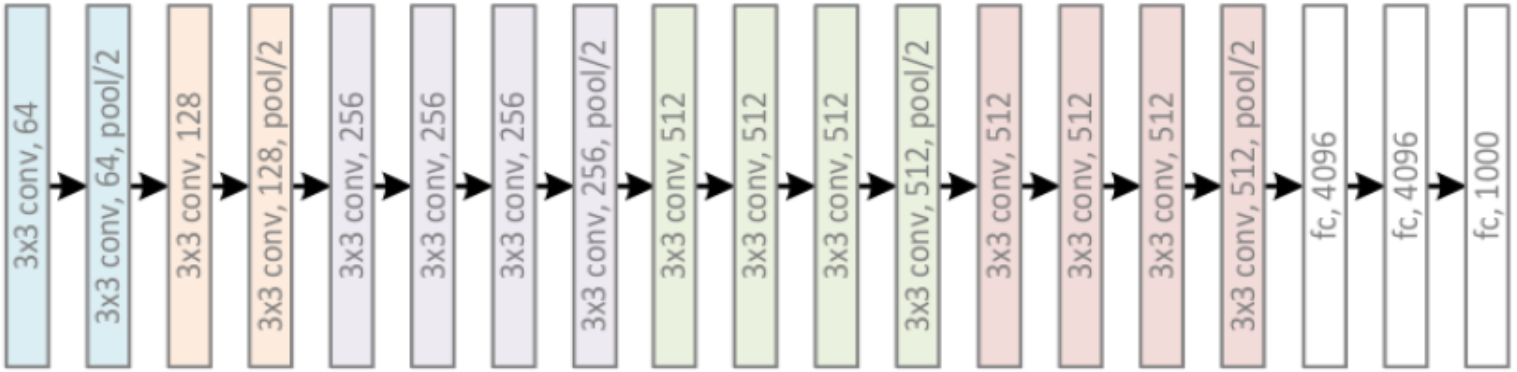
Outline

ConvNets as Deep Neural Networks for Vision

1. Neural Nets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
   - The big picture
   - AlexNet
   - GoogleNet
   - **VGG**

VGG Net: Archi post-2012 revolution

VGG, 16/19 layers, 2014



K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

VGG Net

Basic Idea: Investigate the **effect of depth** in large scale image recognition

- **Fix other parameters** of architecture, and steadily increase depth

Fixed configuration:

- Convolutional Layers: from 8 to 16
- Fully Connected Layers: 3
- Stride: 1
- ReLu: Follow all hidden layers
- Max-Pooling: 2x2 window
- Padding: s/t spatial resolution is preserved
- #Convolutional filters: Starting from 64, double after each max-pooling layer until 512
- Filter sizes: 3x3 and 1x1

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

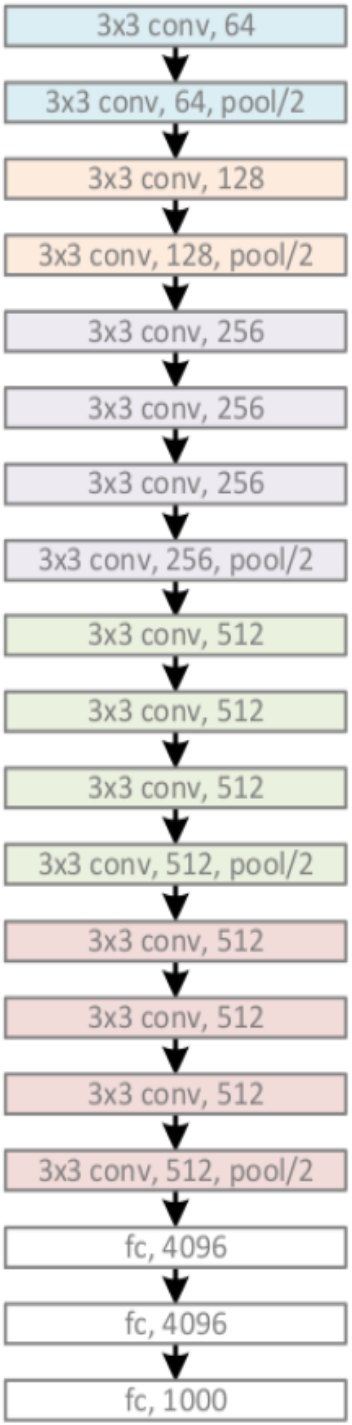


TABLE CREDIT:VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, ICLR2015

VGG Net

Results:

- First place in localization (25.3% error), second in classification (7.3% error) in ILSVRC 2014 using ensemble of 7 networks
- Outperforms Szegedy et.al (GoogLeNet) in terms of single network classification accuracy (7.1% vs 7.9%)

Observations with VGG testing:

- Deepnets with small filters outperform shallow networks with large filters
  - Shallow version of B: 2 layers of 3x3 replaced with single 5x5 performs worse
- LRN does not improve performance
- Classification error decreases with increases ConvNet depth
- Important to capture more spatial context (config D vs C)
- Error rate saturated at 19 layers
- Scale jittering at training helps capturing multiscale statistics and leads to better performance

Outline

ConvNets as Deep Neural Networks for Vision

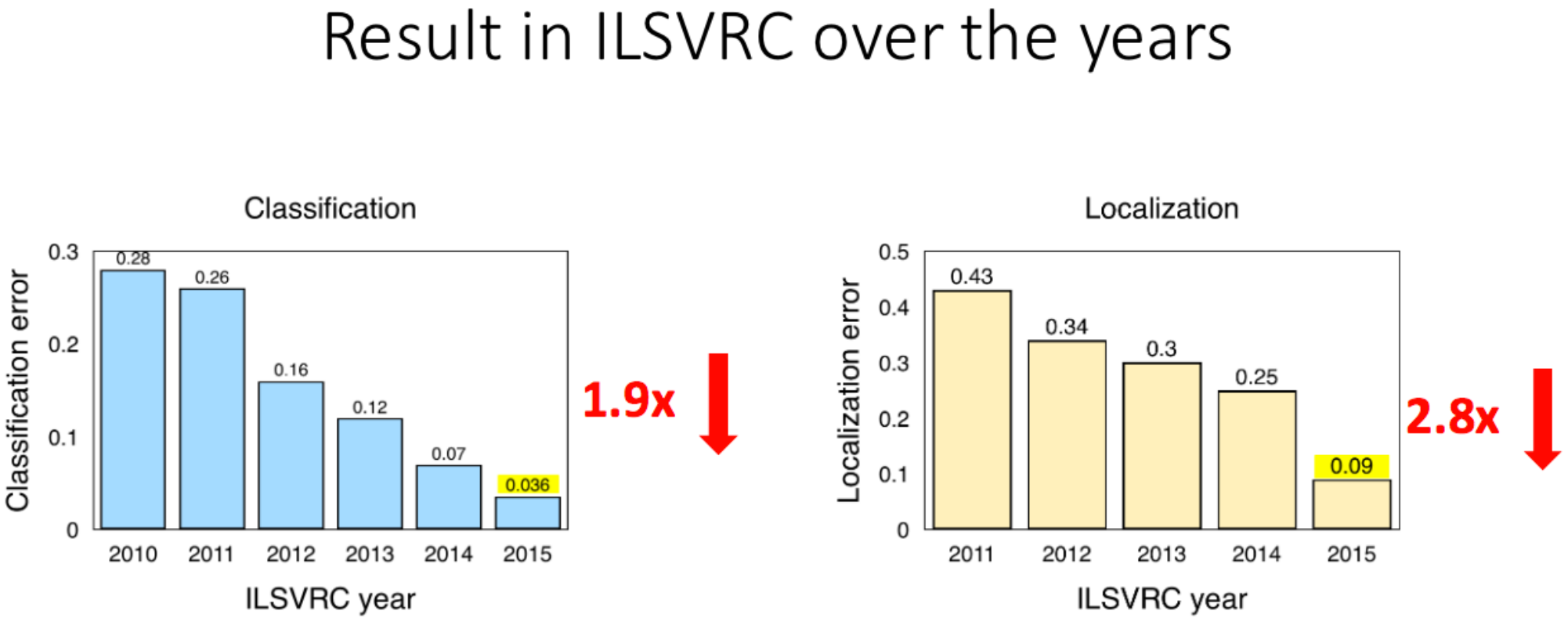
1. Neural Nets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
   - The big picture
   - GoogleNet
   - VGG
   - **ResNet**

Outline

ConvNets as Deep Neural Networks for Vision

1. Neural Nets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
   - The big picture
   - AlexNet
   - GoogleNet
   - VGG
   - **ResNet**

Results 2015: 3.6% top5 error



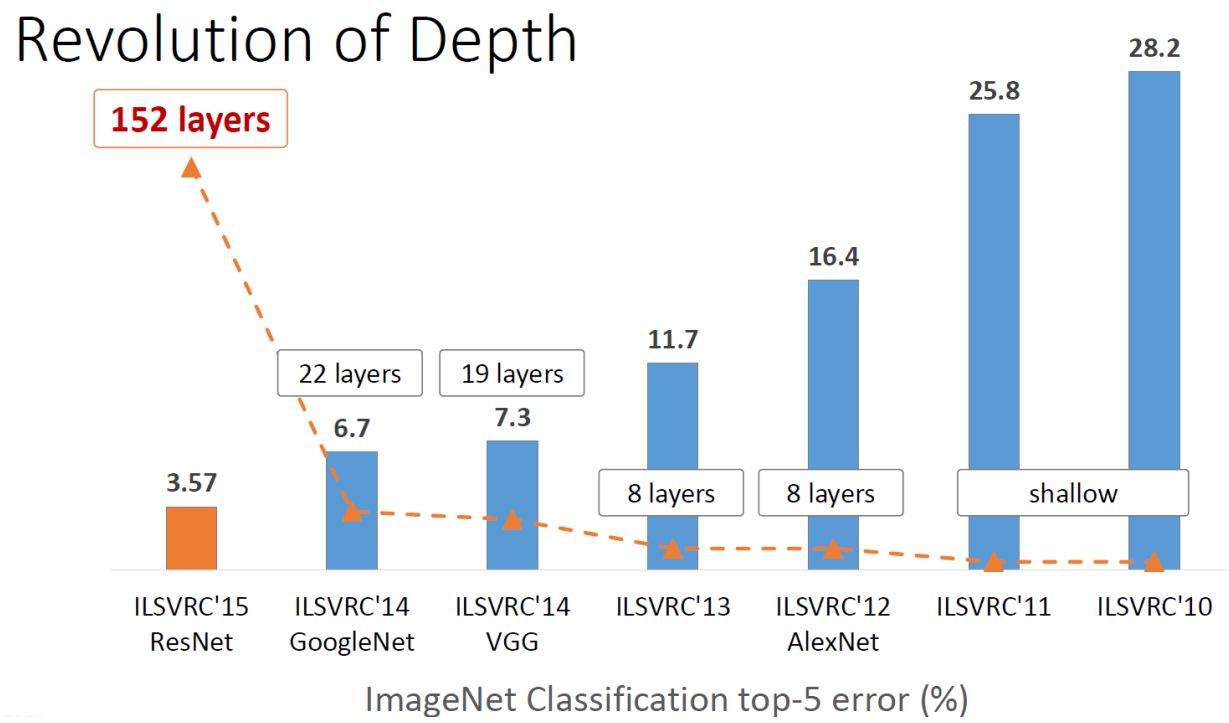
ResNet
The deeper, the better

+ Deeper network covers more complex problems
  - Receptive field size ↑
  - Non-linearity ↑
- Training deeper network more difficult because of vanishing/exploding gradients problem

@ Kaiming He ILSVRC & COCO 2015

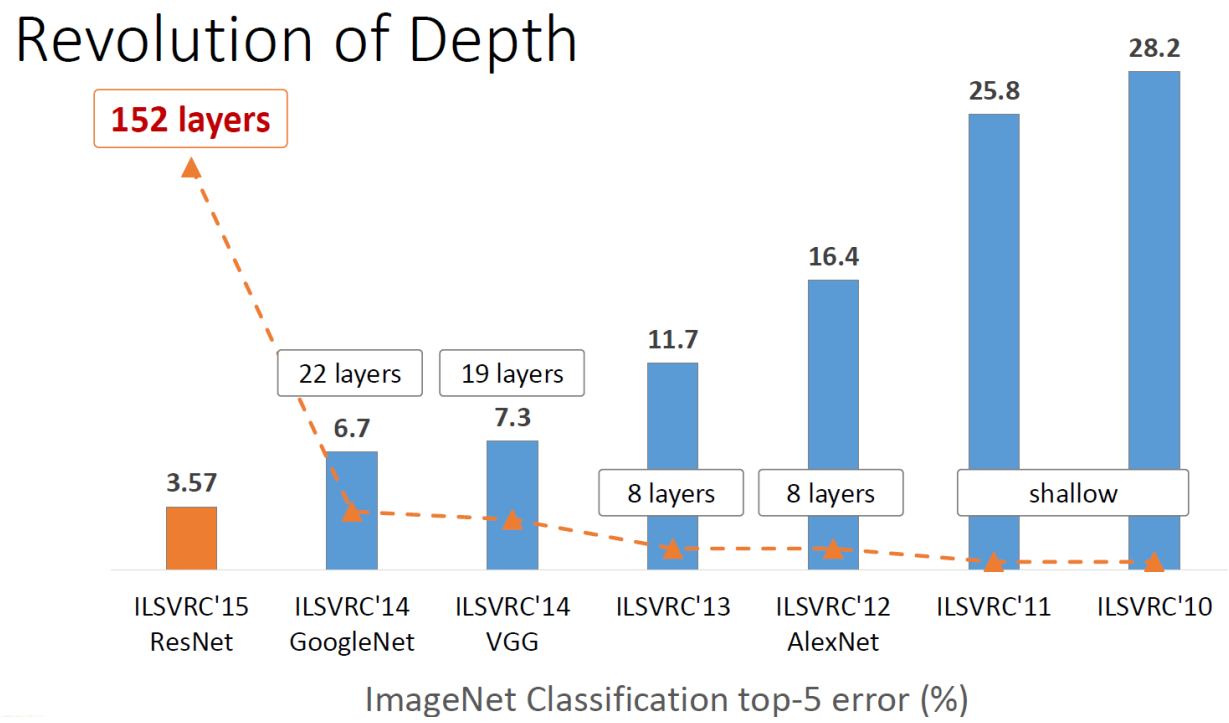
Deep Neural Network

- Escape from few layers
  - ReLU for solving gradient vanishing problem
  - Dropout …



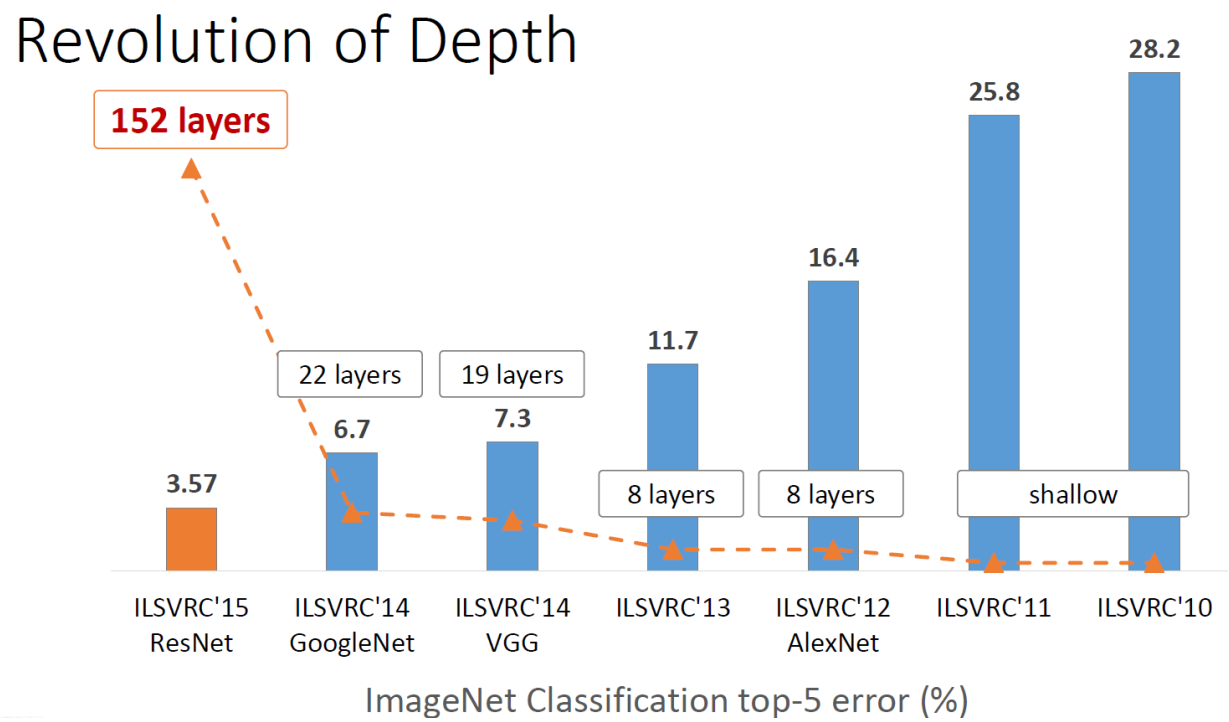
Deep Neural Network

- Escape from 10 layers
  - Normalized initialization
  - Intermediate normalization layers



Deep Neural Network

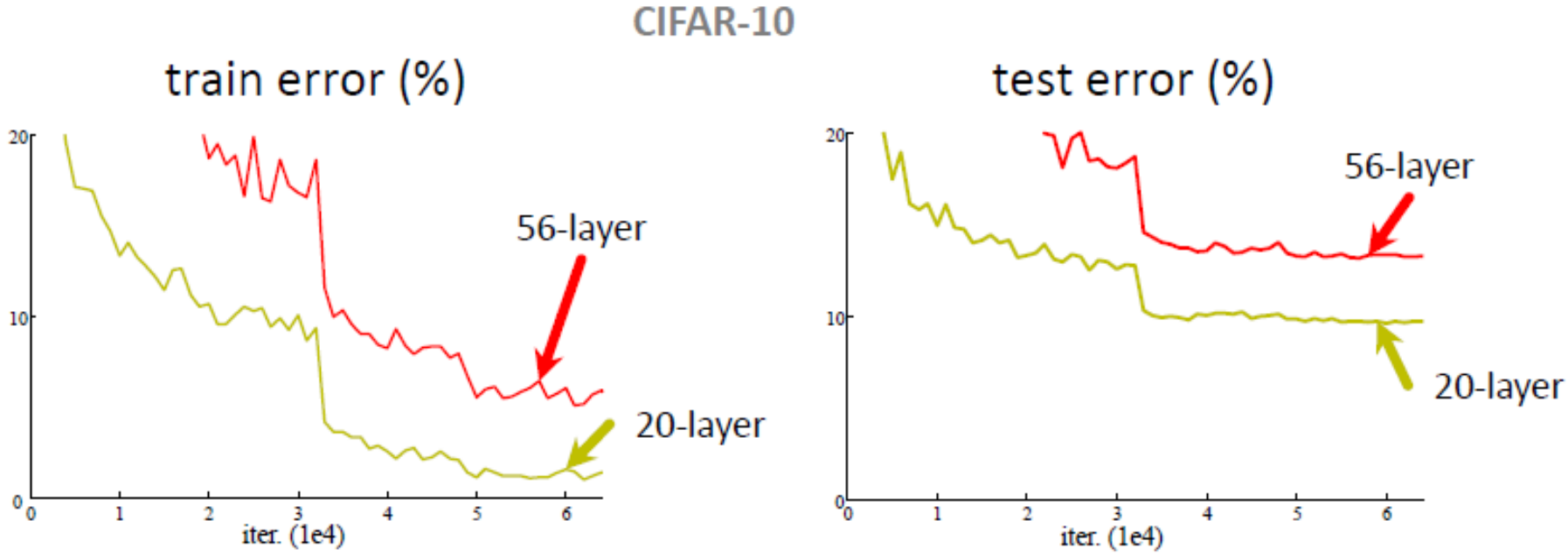
- Escape from 100 layers
  - Residual network



Deeper VGG: 56 Plain Network

Plain nets: stacking 3x3 conv layers

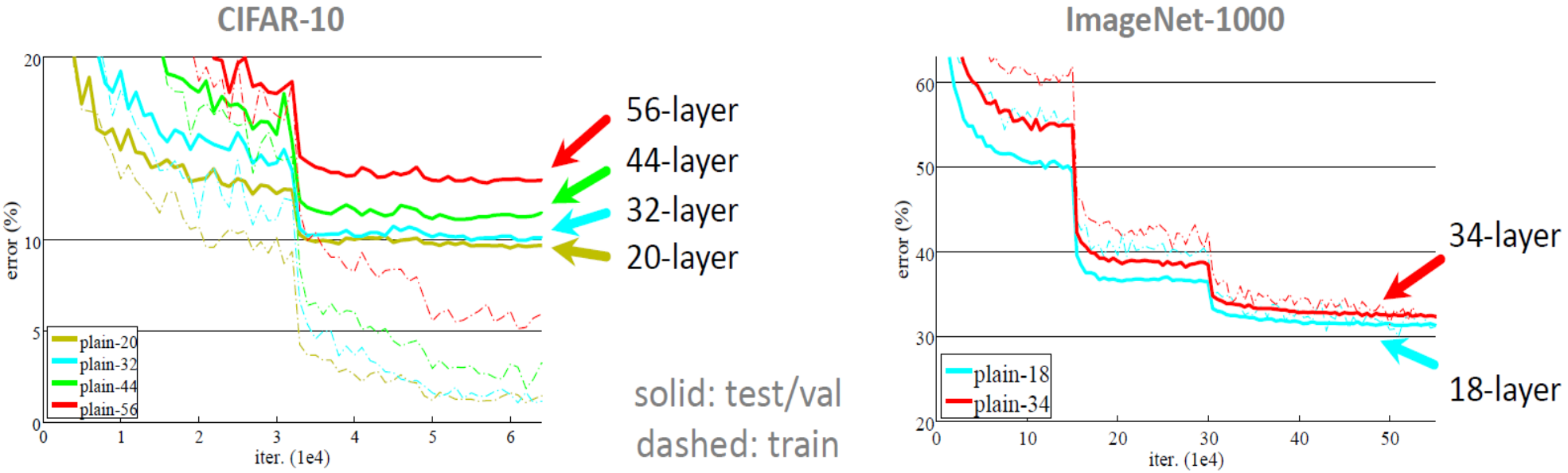
- 56-layer net has higher training error and test error than 20-layers net



Deeper VGG:

"Overly deep" plain nets have higher training error

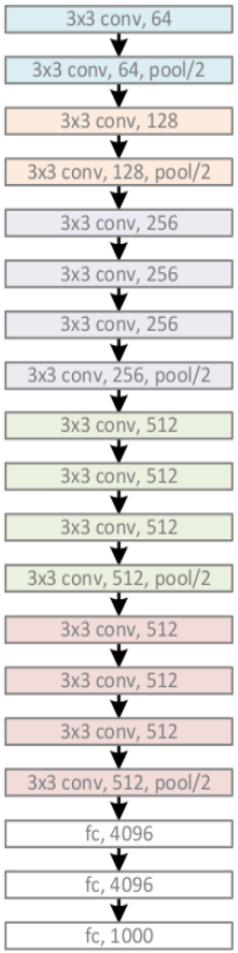
A general phenomenon, observed in many datasets



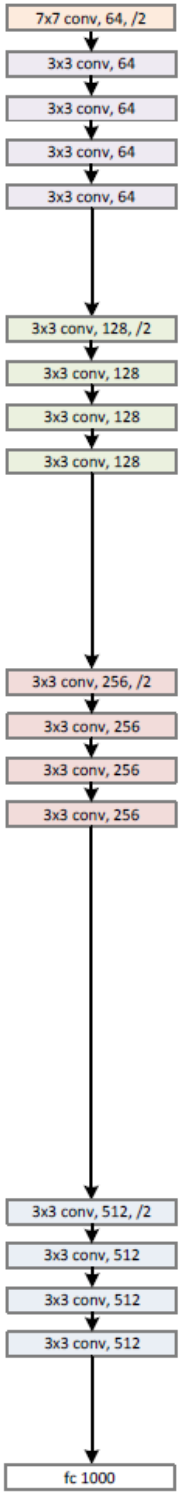
Residual Network

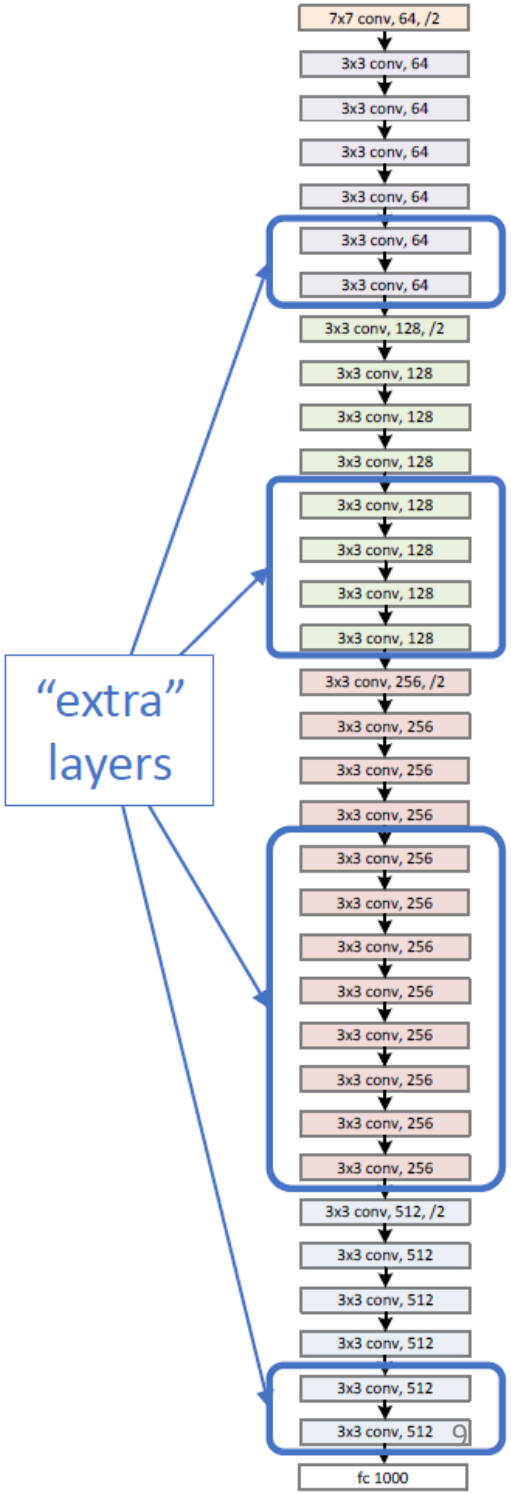
Naïve solution

If extra layers identity mapping, training error not increase



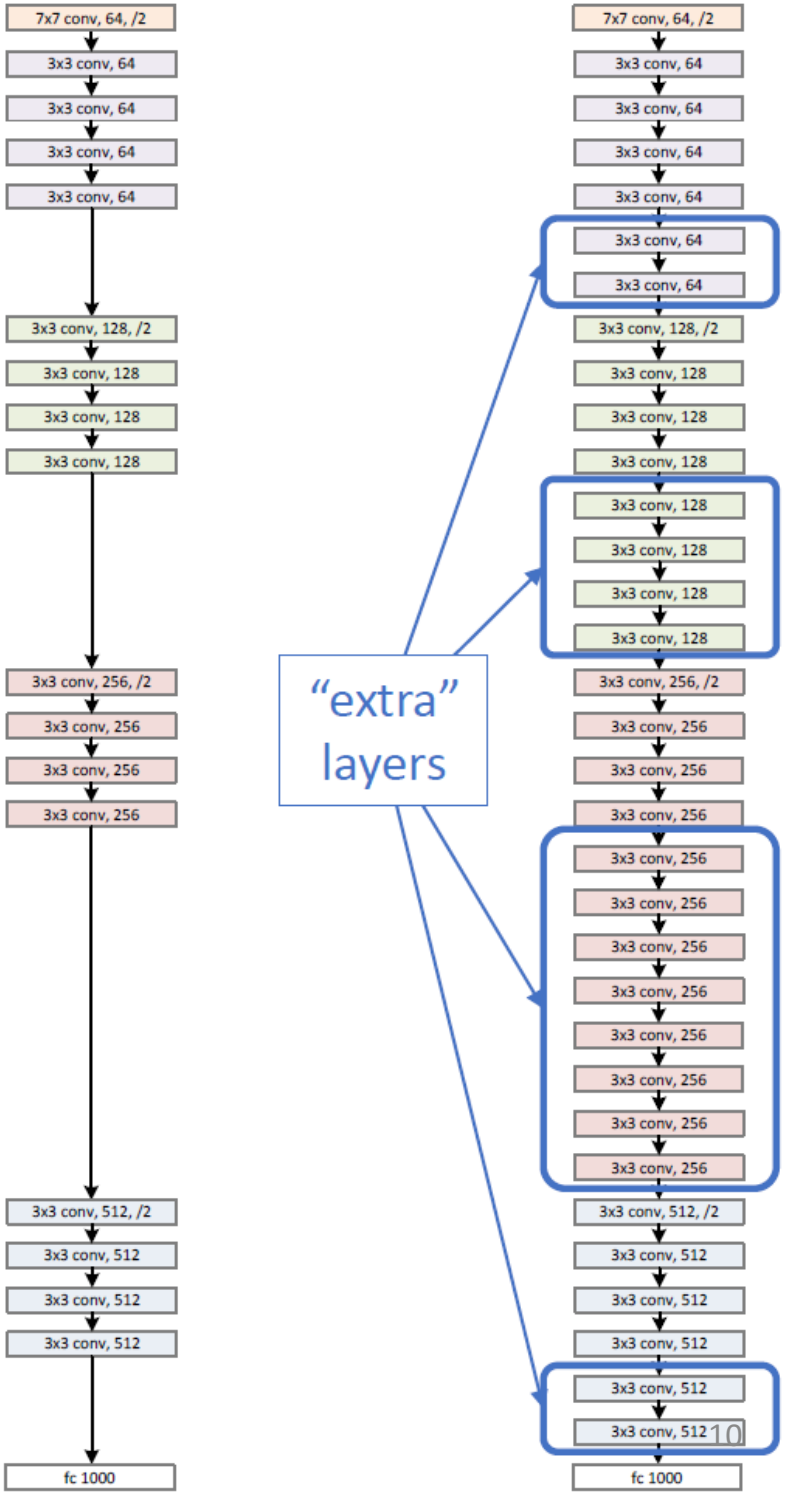
VGG, 16/19 layers, 2014





Residual Network

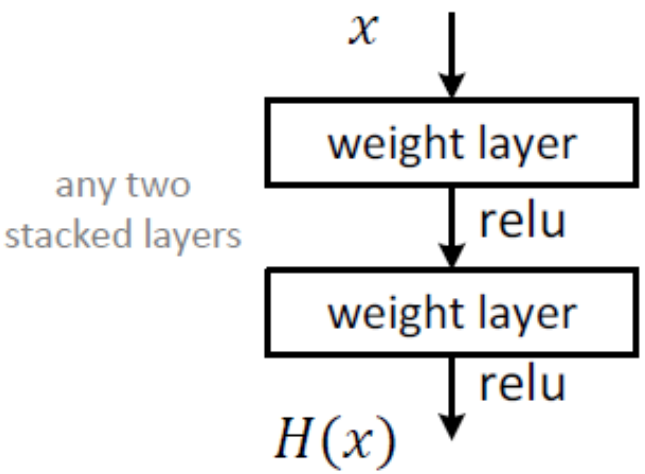
- Deeper networks maintain the tendency of results
  - Features in same level will be almost same
  - An amount of changes is fixed
  - Adding layers make smaller differences
  - Optimal mappings closer to an identity



Residual Network

Plain block

- Difficult to make identity mapping because of multiple non-linear layers

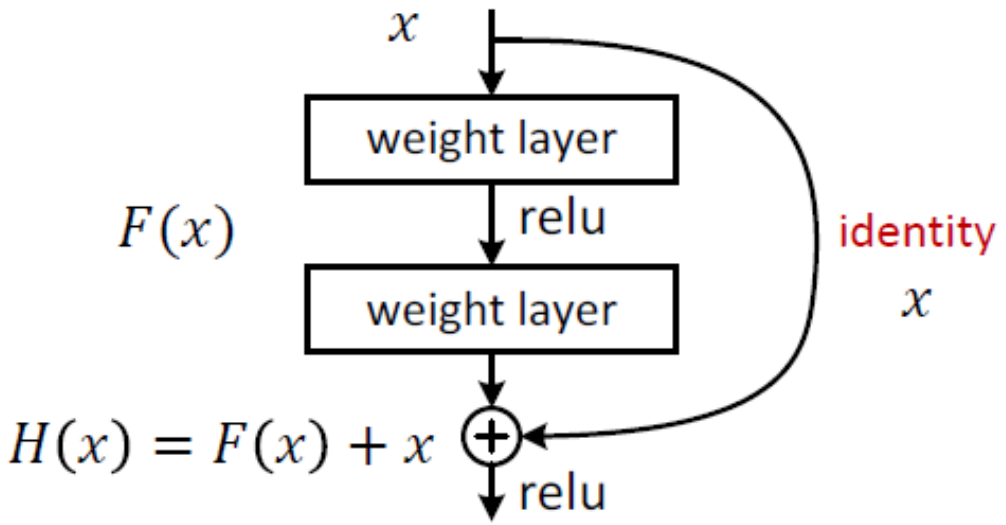


Residual Network

Residual block

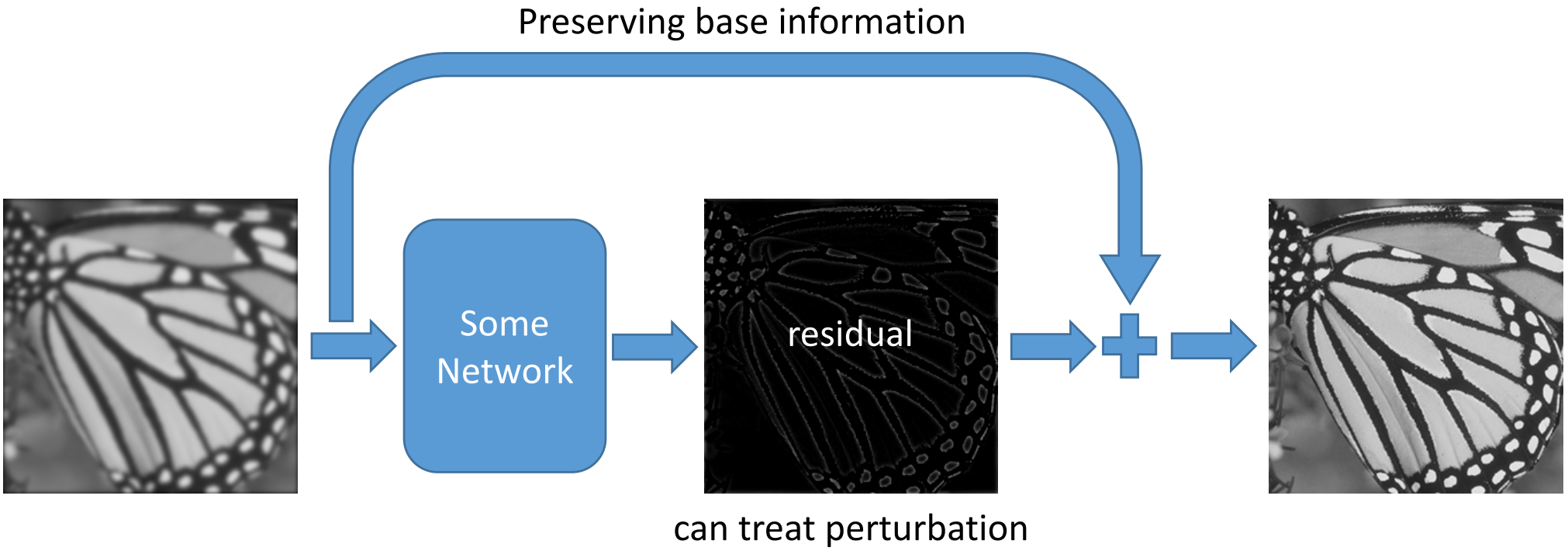
If identity were optimal,
easy to set weights as 0
If optimal mapping is closer
to identity, easier to find
small fluctuations

-> Appropriate for treating
perturbation as keeping a
base information



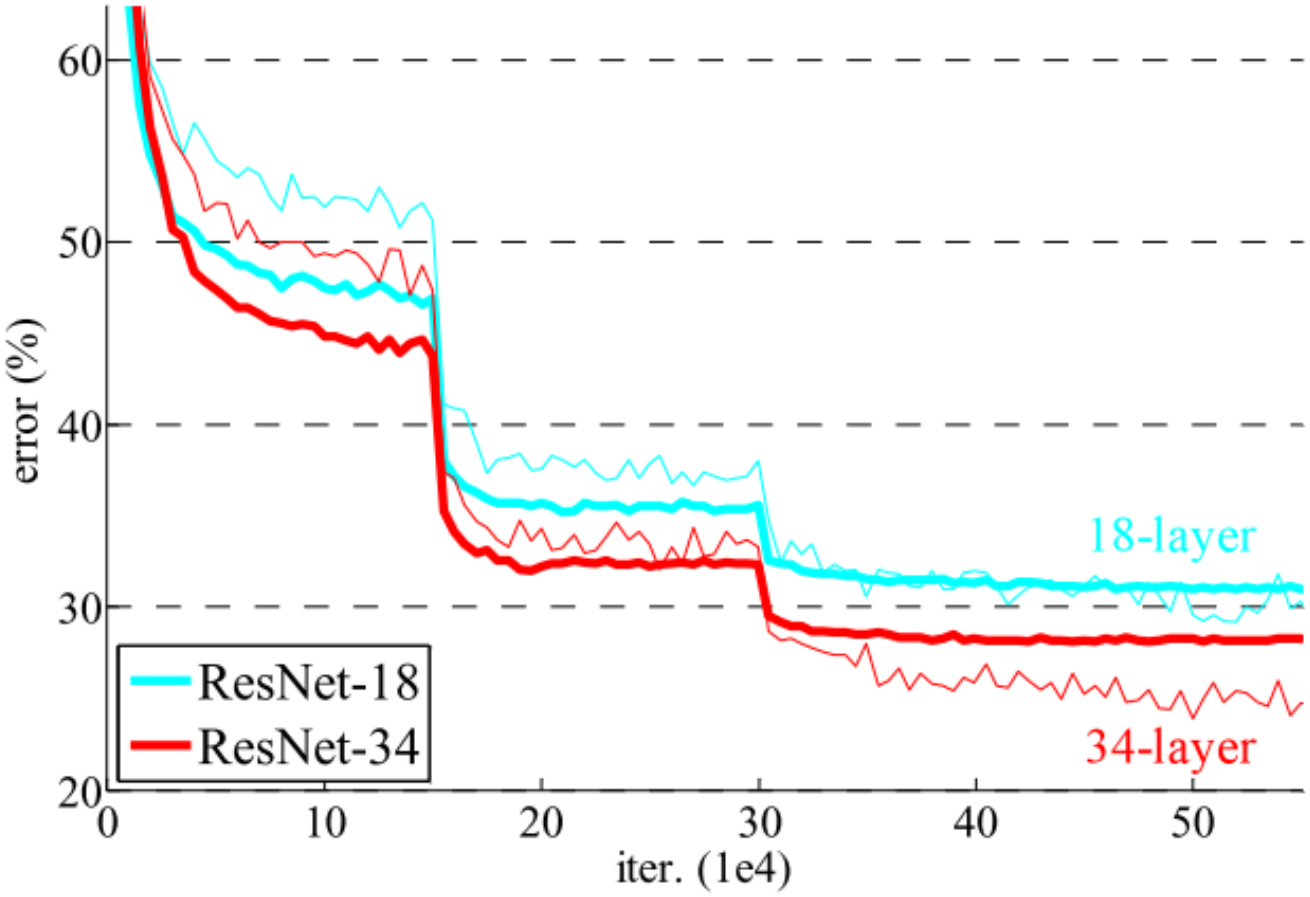
Residual Network

- Difference between an original image and a changed image



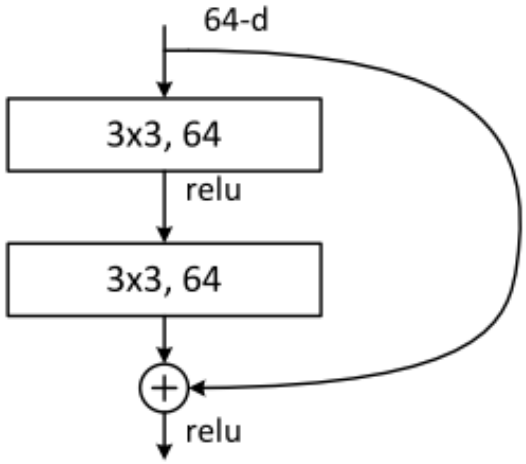
Residual Network

Deeper ResNets have lower training error

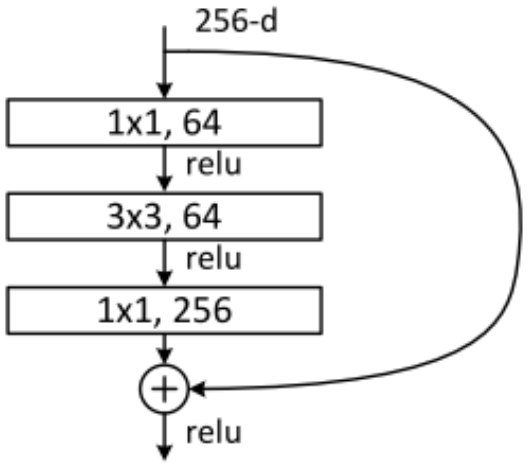


Residual Network

- Residual block
  - Very simple
  - Parameter-free



A naïve residual block



"bottleneck" residual block
(for ResNet-50/101/152)

Residual Network

- Shortcuts connections
  - Identity shortcuts

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

  - Projection shortcuts

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

Network Design

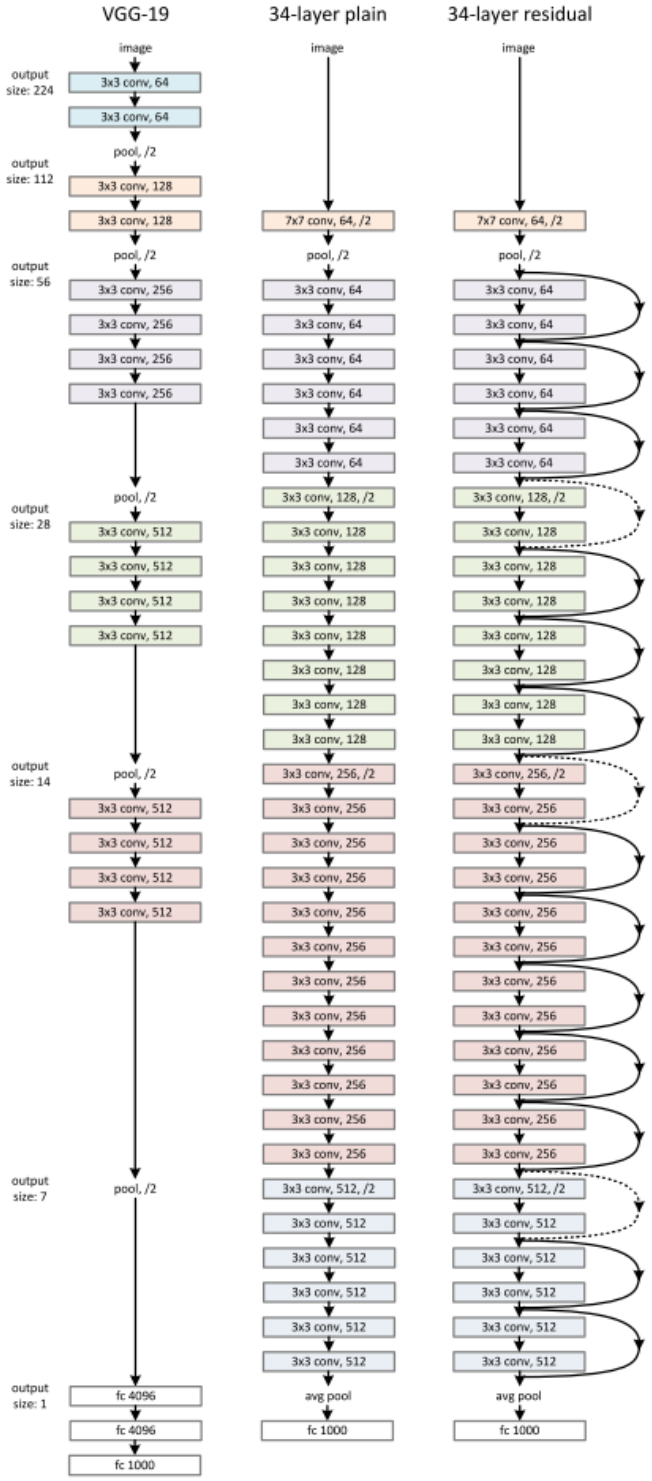
Basic design (VGG-style)

- All 3x3 conv (almost)
- Spatial size/2 => #filters x2
- Batch normalization
- Simple design, just deep

Other remarks

- No max pooling (almost)
- No hidden fc
- No dropout

ConvNet Configuration			
B	C	D	E
13 weight layers	16 weight layers	16 weight layers	19 weight layers
ut (224 × 224 RGB image)			
conv3-64	conv3-64	conv3-64	conv3-64
conv3-64	conv3-64	conv3-64	conv3-64
maxpool			
conv3-128	conv3-128	conv3-128	conv3-128
conv3-128	conv3-128	conv3-128	conv3-128
maxpool			
conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256
	conv1-256	**conv3-256**	conv3-256
			conv3-256
maxpool			
conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512
	conv1-512	**conv3-512**	conv3-512
			conv3-512
maxpool			
conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512
	conv1-512	**conv3-512**	conv3-512
			conv3-512
maxpool			
FC-4096			
FC-4096			
FC-1000			
soft-max			

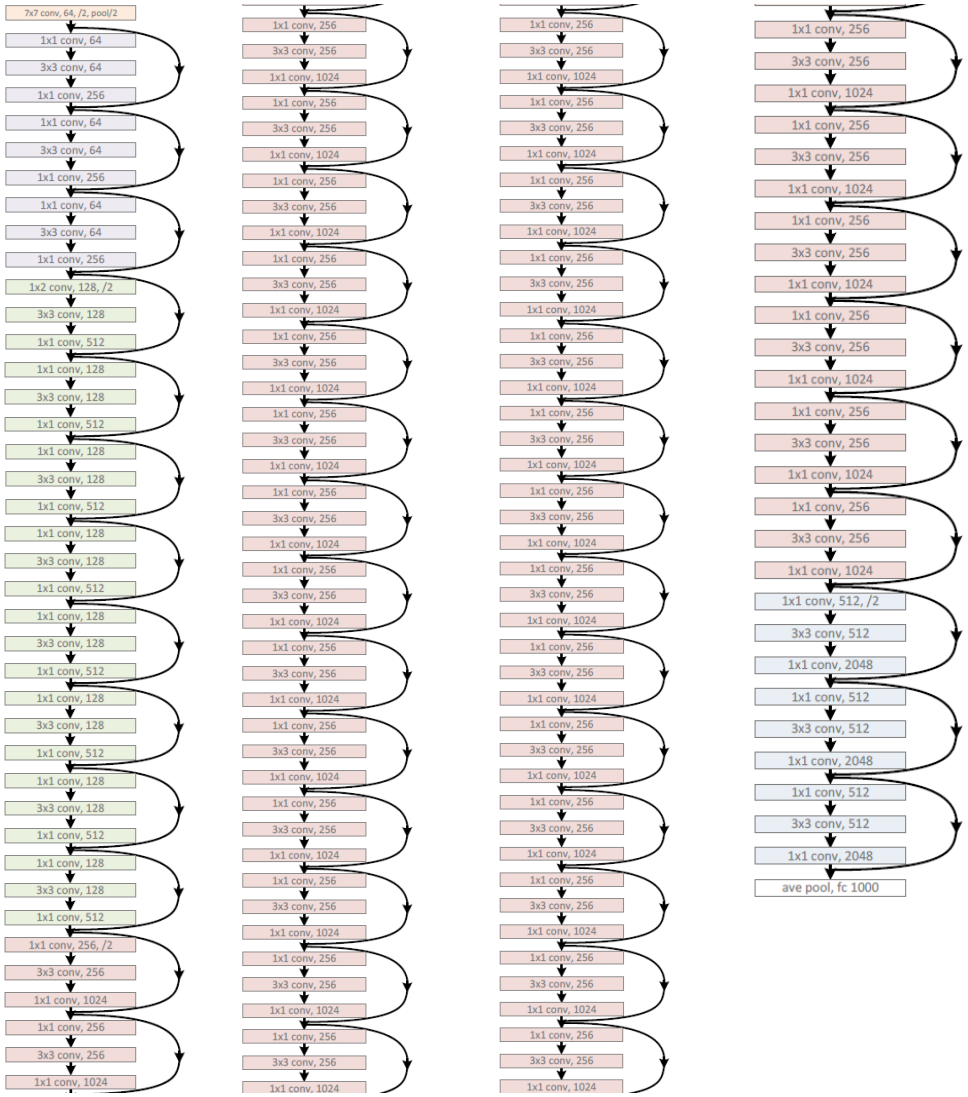


Network Design

ResNet-152

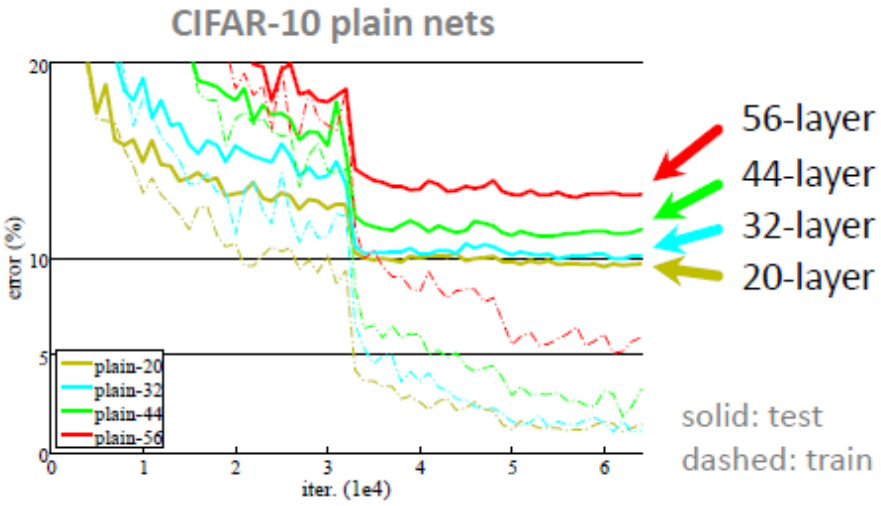
- Use bottlenecks
- ResNet-152 (11.3 billion FLOPs) lower complexity than VGG-16/19 nets (15.3/19.6 billion FLOPs)

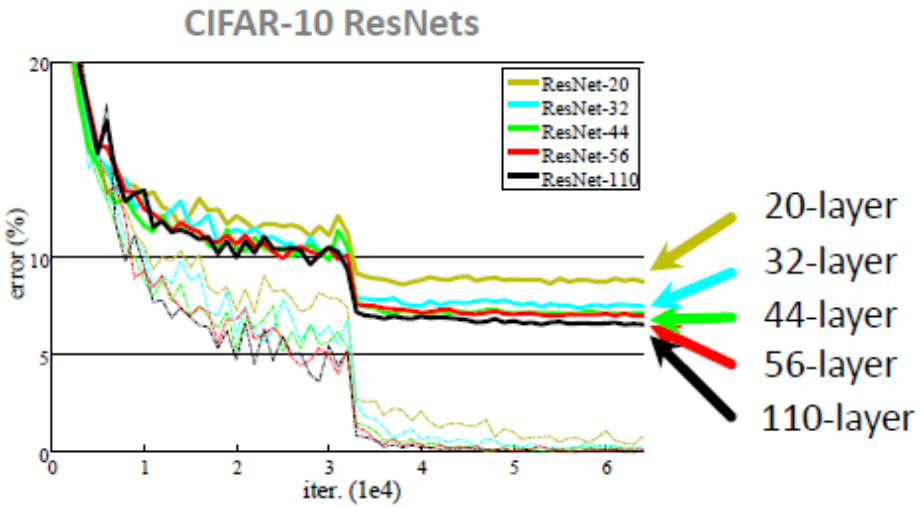
About 64M parameters



Results

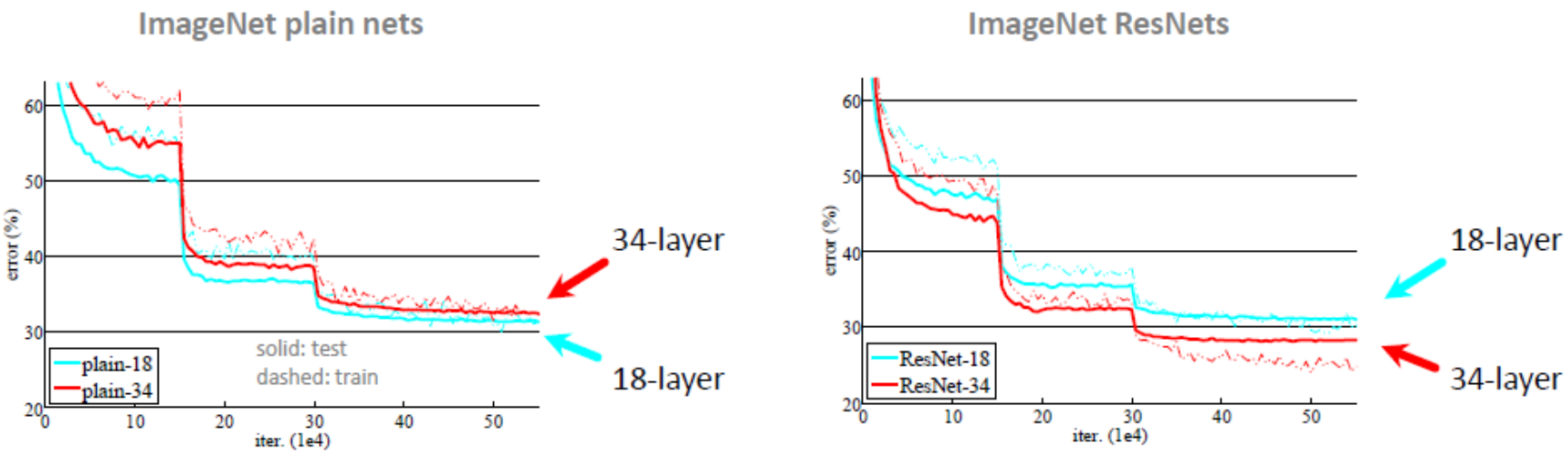
- Deep Resnets can be trained without difficulties
- Deeper ResNets have lower training error, and also lower test error





Results

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

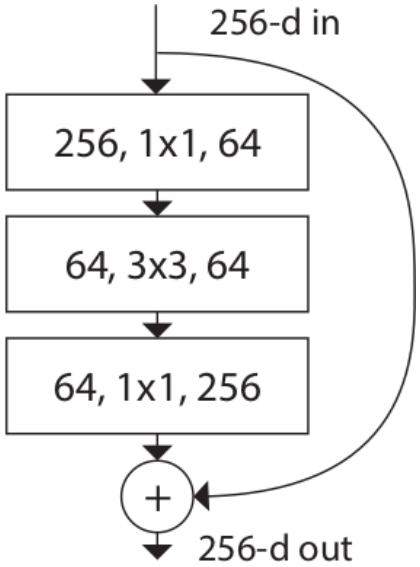


Results

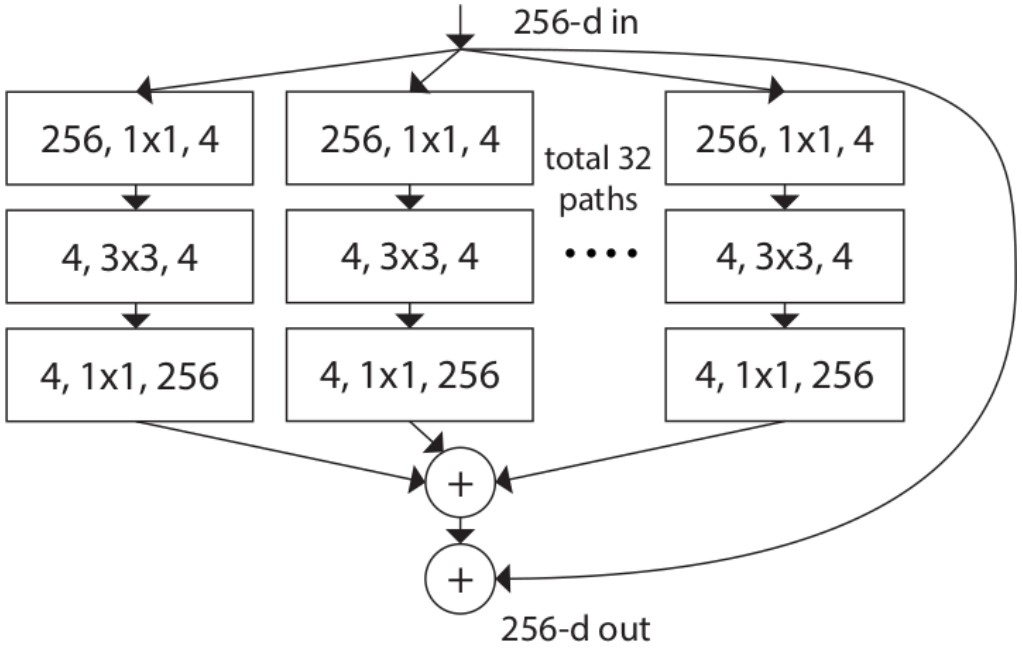
- 1st places in all five main tracks in “ILSVRC & COCO 2015 Competitions”
  - ImageNet Classification
  - ImageNet Detection
  - ImageNet Localization
  - COCO Detection
  - COCO Segmentation

Deep ConvNets for image classification

- **ResNeXt**
  - Multi-branch architecture



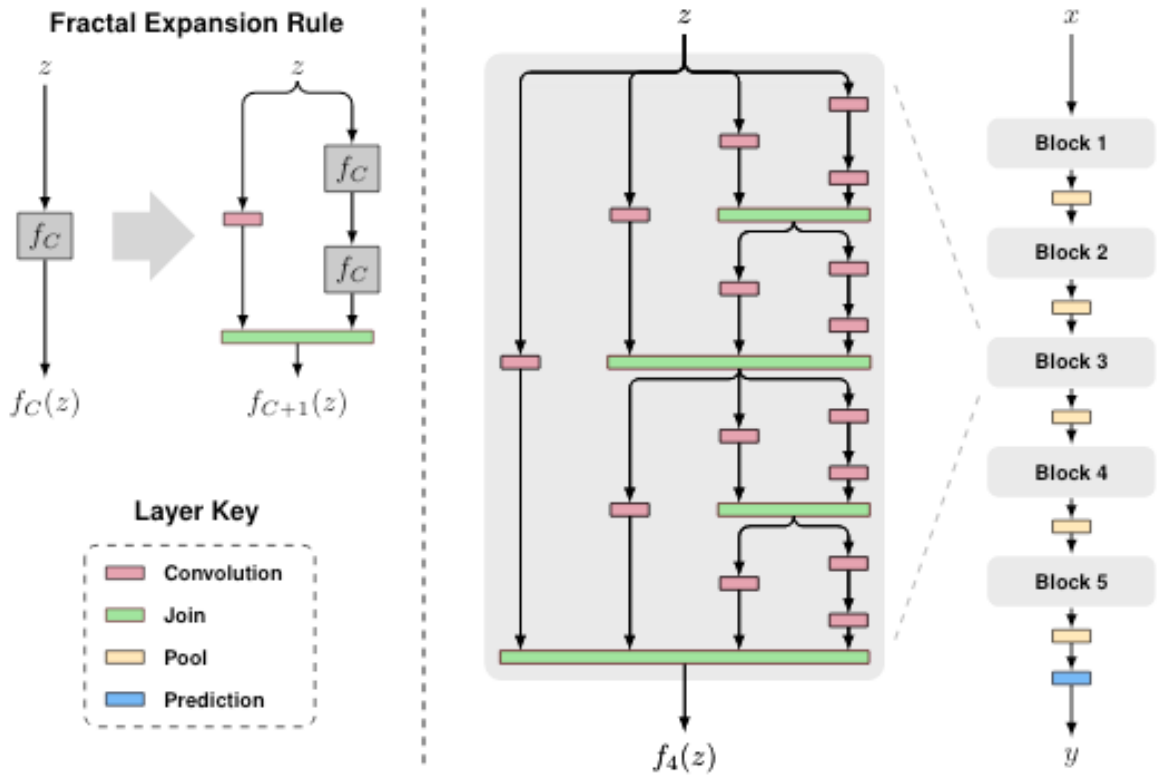
ResNet



ResNeXt

Saining Xie, Ross Girshick, Piotr Dollàr, Zhuowen Tu and Kaiming He
Aggregated Residual Transformations for Deep Neural Networks.
In *CVPR*, 2017.

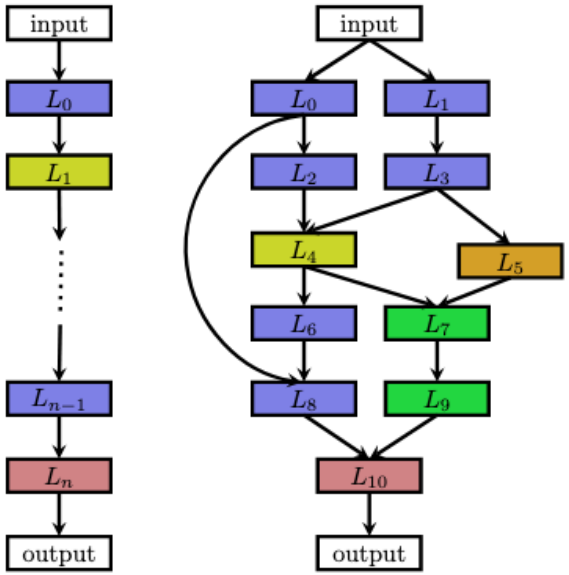
FractalNet: Ultra-Deep Neural Networks without Residuals

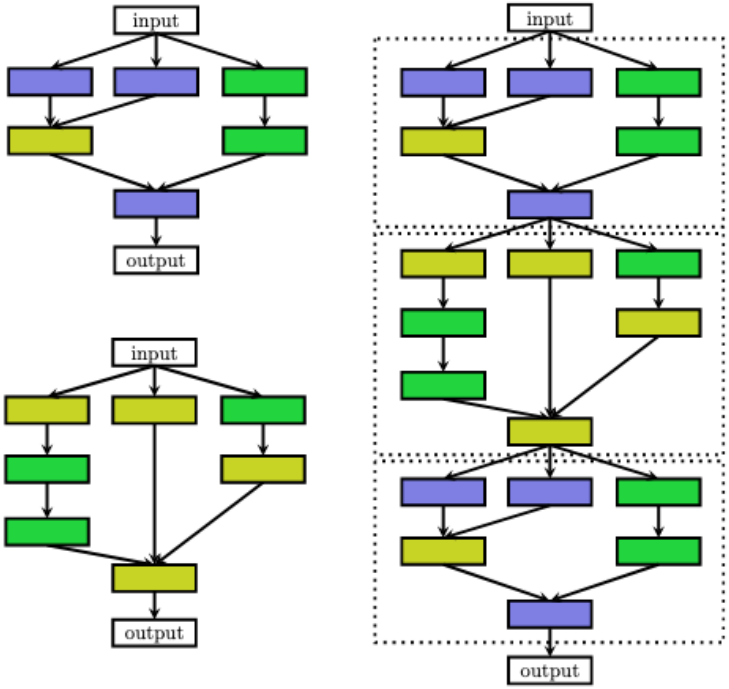


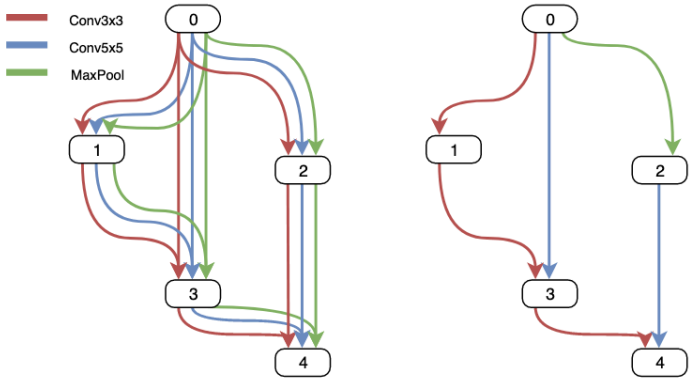
Comments from paper: *"Our experiments demonstrate that **residual representation is not fundamental to the success of extremely deep** convolutional neural networks. A fractal design achieves an error rate of 22.85% on CIFAR-100, matching the state-of-the-art held by residual networks."*

Exploring type of deep modules in Neural Nets

Neural Architecture Search







Conclusion

- ResNet: currently the best archi for large scale image classification
- Not yet consensus about the design of the Net (cf. FractalNet), Neural Architecture Search

- Fully Convolutional Net (FCN) very interesting option
  Beyond classification!

COMPLEMENTS

Batch normalization

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

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

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

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

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

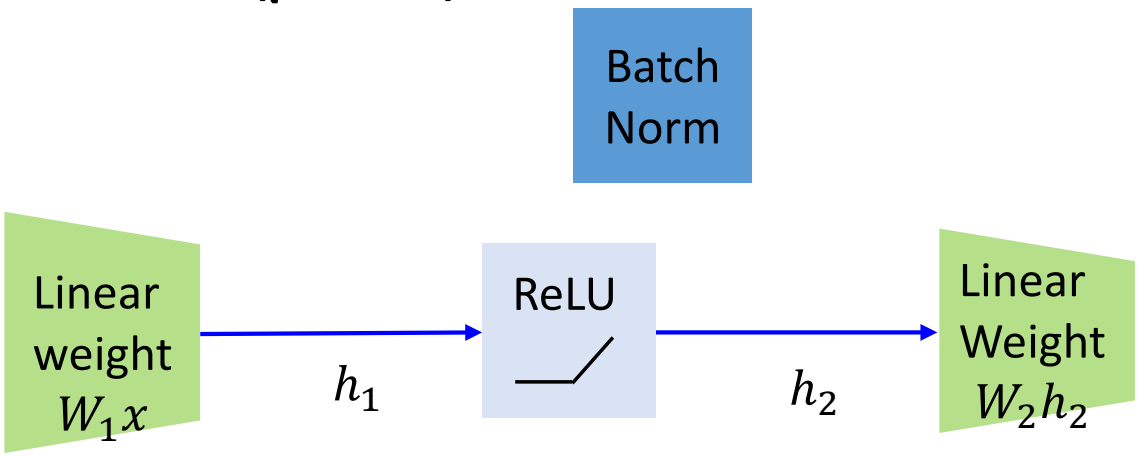
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- Improves gradient flow through the network
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Reduces need for dropout

Un-normalization!! Re-compute and apply the optimal scaling and bias for each neuron!
Learn γ and β (same dims as μ and σ^2).
It can (should?) learn the identity mapping!

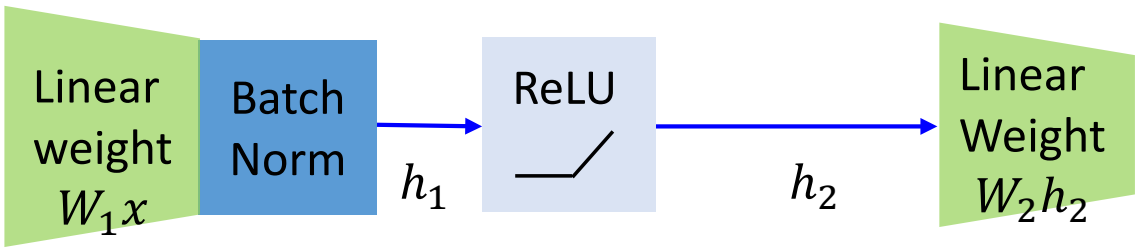
Where to do BatchNorm ?

BatchNorm is just a linear scale/bias layer in the limit of large batch sizes (μ, σ^2)



Where to do BatchNorm ?

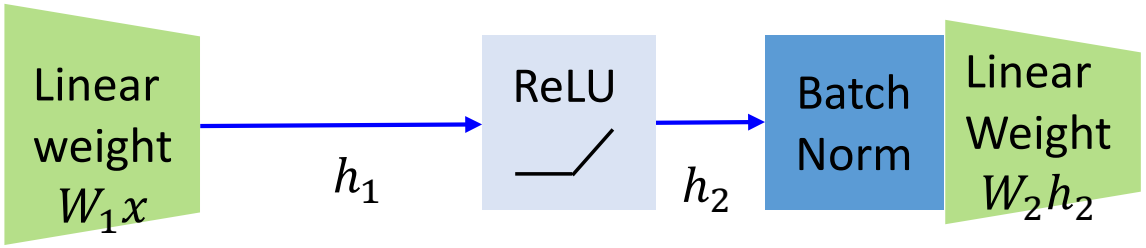
BatchNorm is just a linear scale/bias layer in the limit of large batch sizes (μ, σ^2).



?? Linear weight layer should already have optimal scale/bias in its output

Where to do BatchNorm ?

BatchNorm is just a linear scale/bias layer in the limit of large batch sizes (μ, σ^2).



?? Any bias/scaling by the batch norm layer should be over-ruled by the second linear weight layer.

Multitask learning / auxiliary loss in GoogLeNet