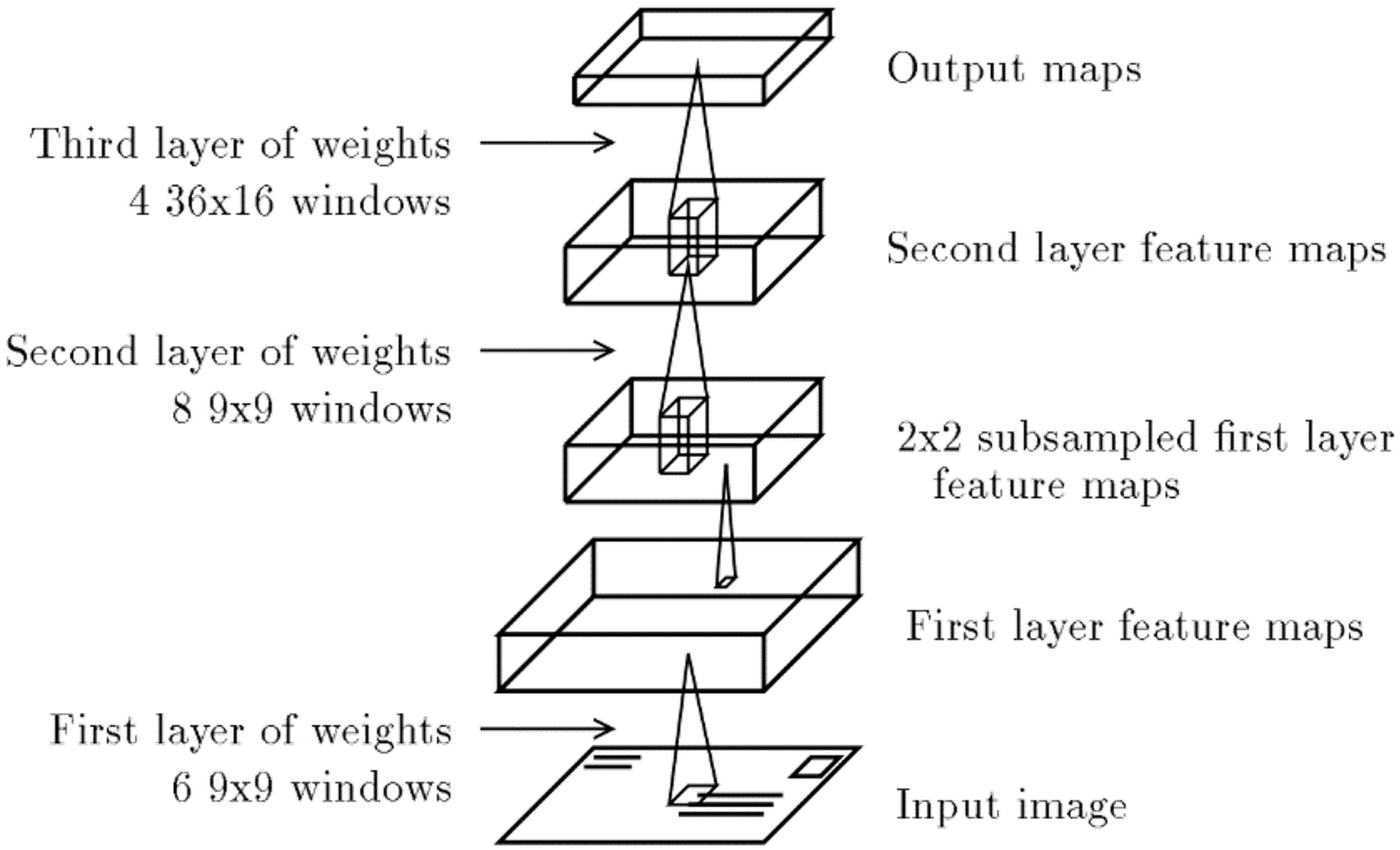
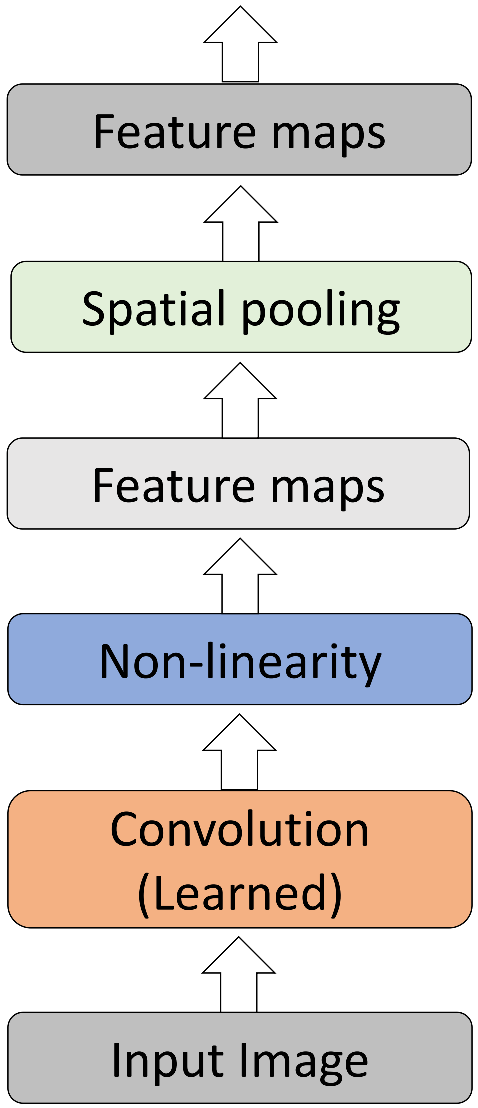
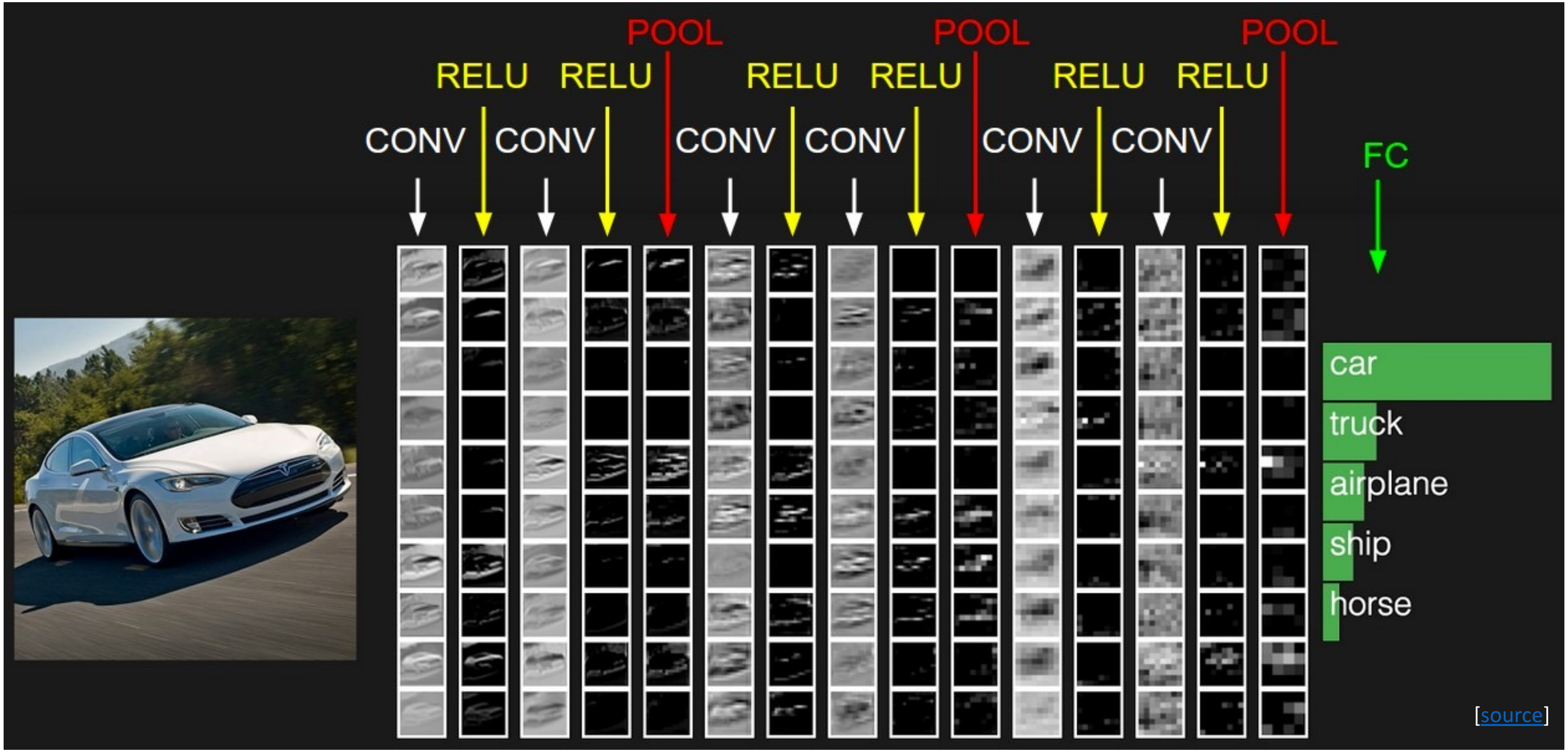
Recap: Network Hierarchy



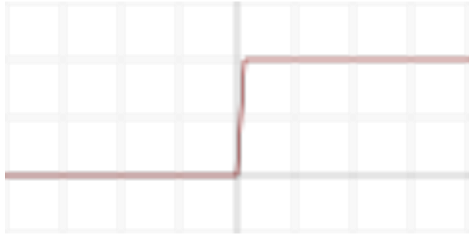


Recap: Network Hierarchy



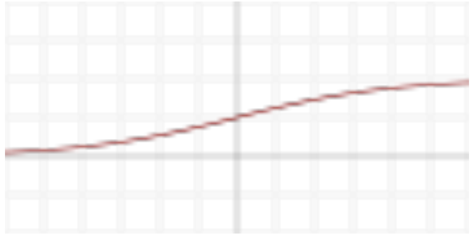
[source]

Recap: Activation Functions



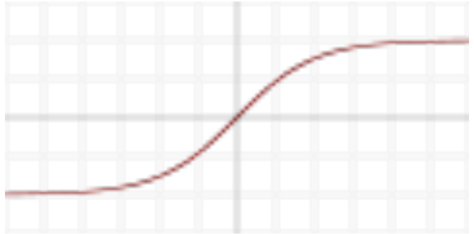
Binary

$[x > 0]$



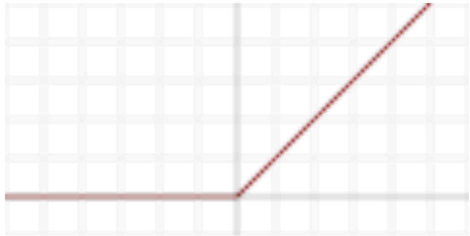
Logistic

$\frac{1}{1+\exp(-x)}$



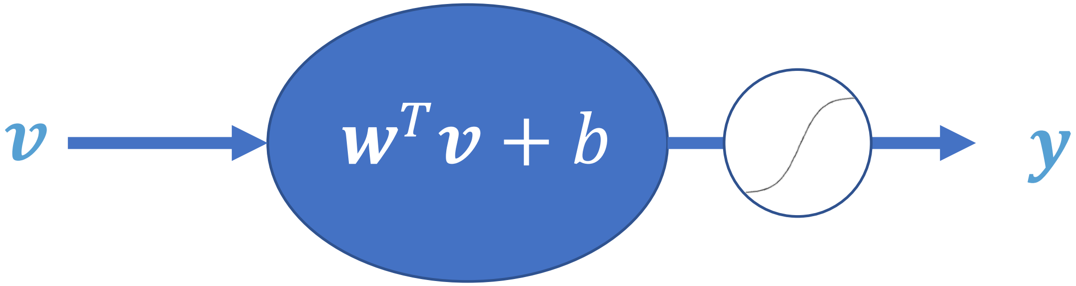
tanh

$\frac{2}{1+\exp(-2x)} - 1$



ReLU

$\max(x, 0)$



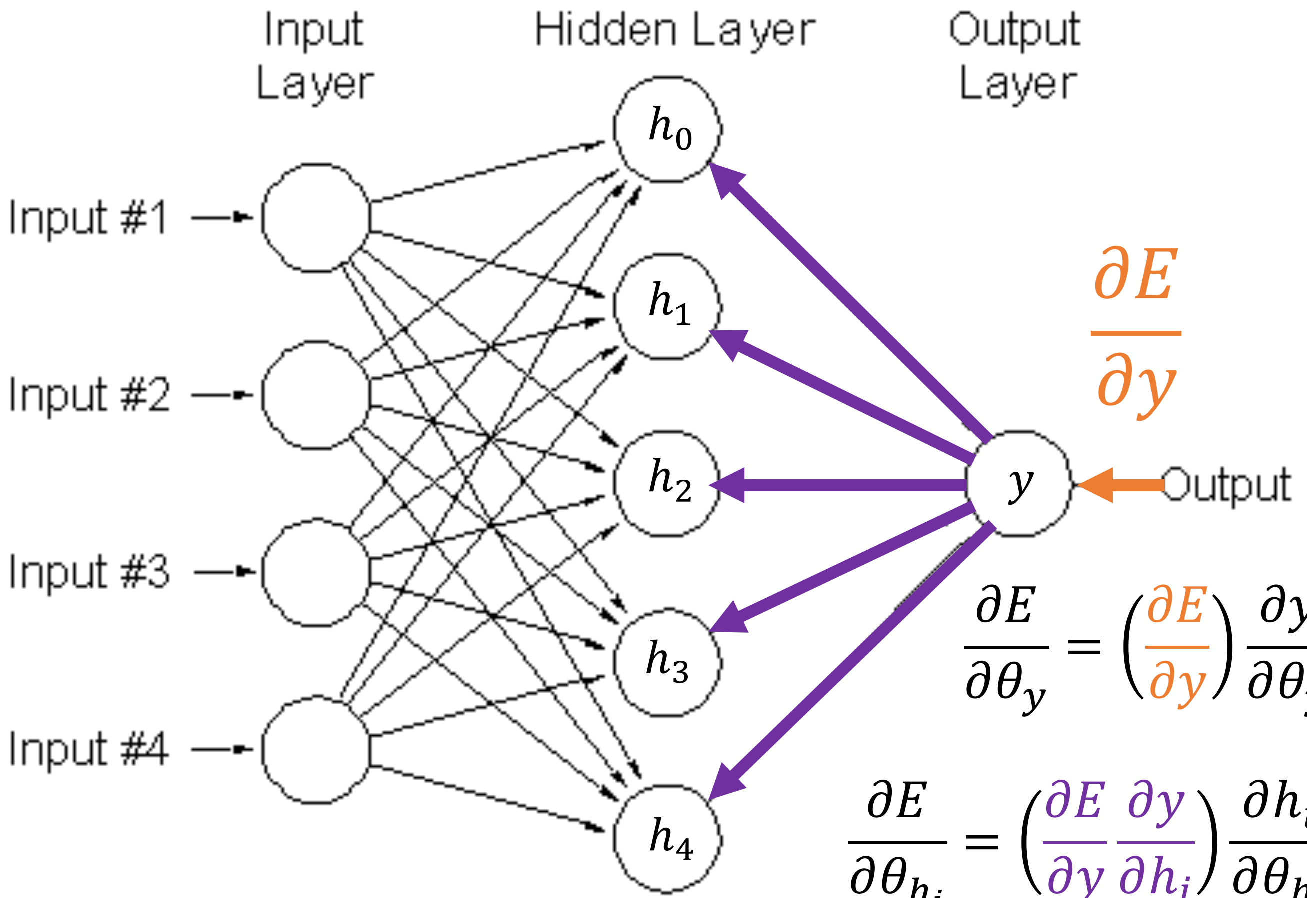
[source]

Recap: Backpropagation

- Starting at the end, we can calculate derivatives and pass these values back to previous layers

- Within each neuron, we're also backpropagating along the operation graph

- Takeaway: Network components must be differentiable



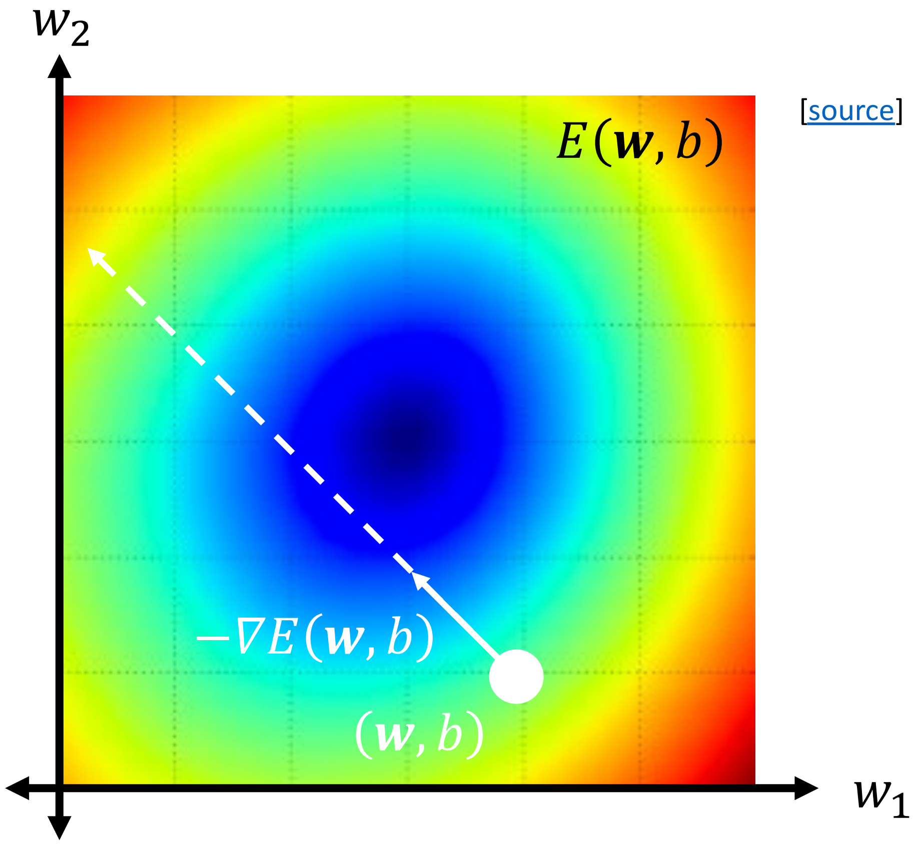
$$\frac{\partial E}{\partial \theta_y} = \left(\frac{\partial E}{\partial y}\right)\frac{\partial y}{\partial \theta_y}$$

$$\frac{\partial E}{\partial \theta_{h_i}} = \left(\frac{\partial E}{\partial y}\frac{\partial y}{\partial h_i}\right)\frac{\partial h_i}{\partial \theta_{h_i}}$$

Stochastic Gradient Descent

- Stochastic Gradient Descent
  - Evaluate only a single data-point at a time, and march accordingly
  - Randomly order training data such that index $i(t)$ is given at descent iteration t

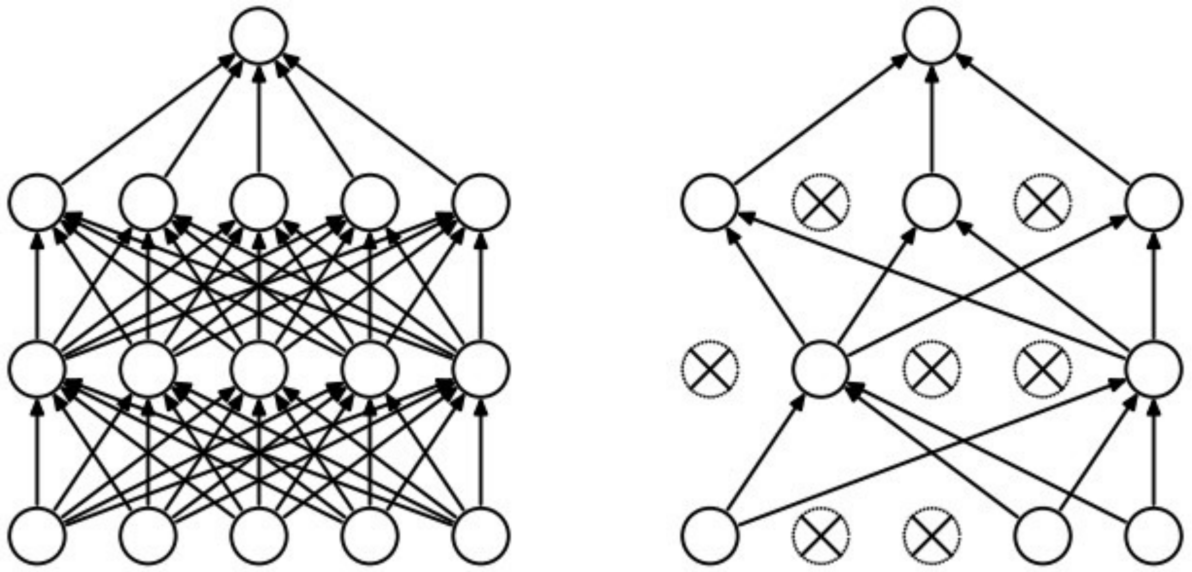
$$E_t(\boldsymbol{w}, b) = \frac{1}{2}\left(y_{i(t)}(\boldsymbol{w}, b) - \hat{y}_{i(t)}\right)^2$$



[source]

Dropout

- Increase sparsity: By randomly removing activations, neurons can more quickly build up strong relationships with certain inputs
- Results in increased neuron specialization
- Simulates averaging over a variety of models



(a) Standard Neural Net

(b) After applying dropout.

[source]

Loss Functions

- Regression: If our target values are continuous, a common approach is to use mean squared error (MSE)

$$E(\boldsymbol{\theta}) = \frac{1}{2N}\sum_i (y_i(\boldsymbol{\theta}) - \widehat{y_i})^2$$

- Two-class Classification: Binary cross-entropy

$$E(\boldsymbol{\theta}) = -\frac{1}{N}\sum_i \begin{array}{r} \widehat{y_i}\ \ln y_i(\boldsymbol{\theta})) + \\ (1 - \widehat{y_i})\ln(1 - y_i(\boldsymbol{\theta})) \end{array}$$

- Multi-class Classification: Categorical cross-entropy

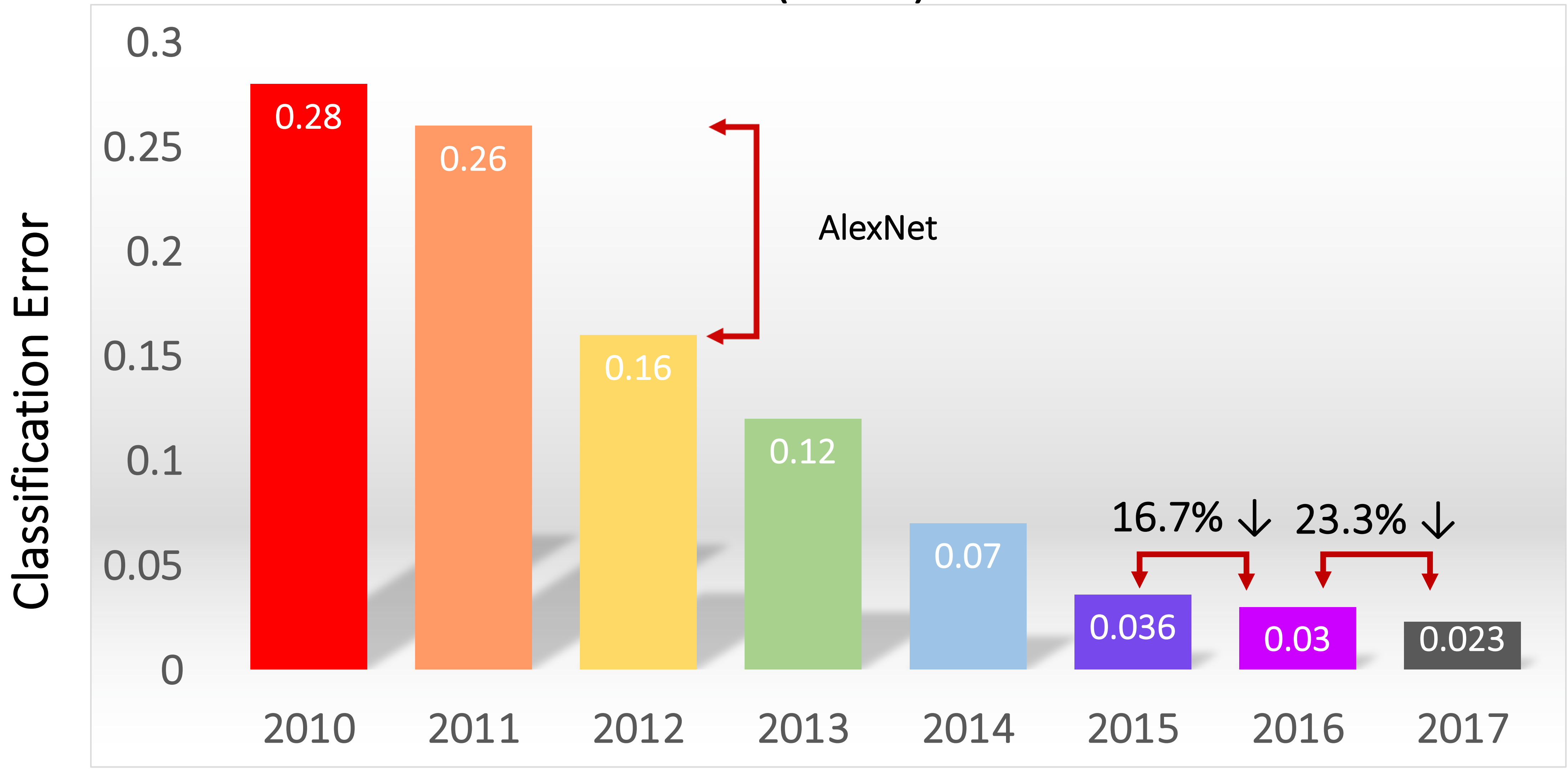
$$E(\boldsymbol{\theta}) = -\sum_i \sum_c \widehat{y_{i,c}} \ln y_{i,c}(\boldsymbol{\theta})$$

Convolutional Neural Networks

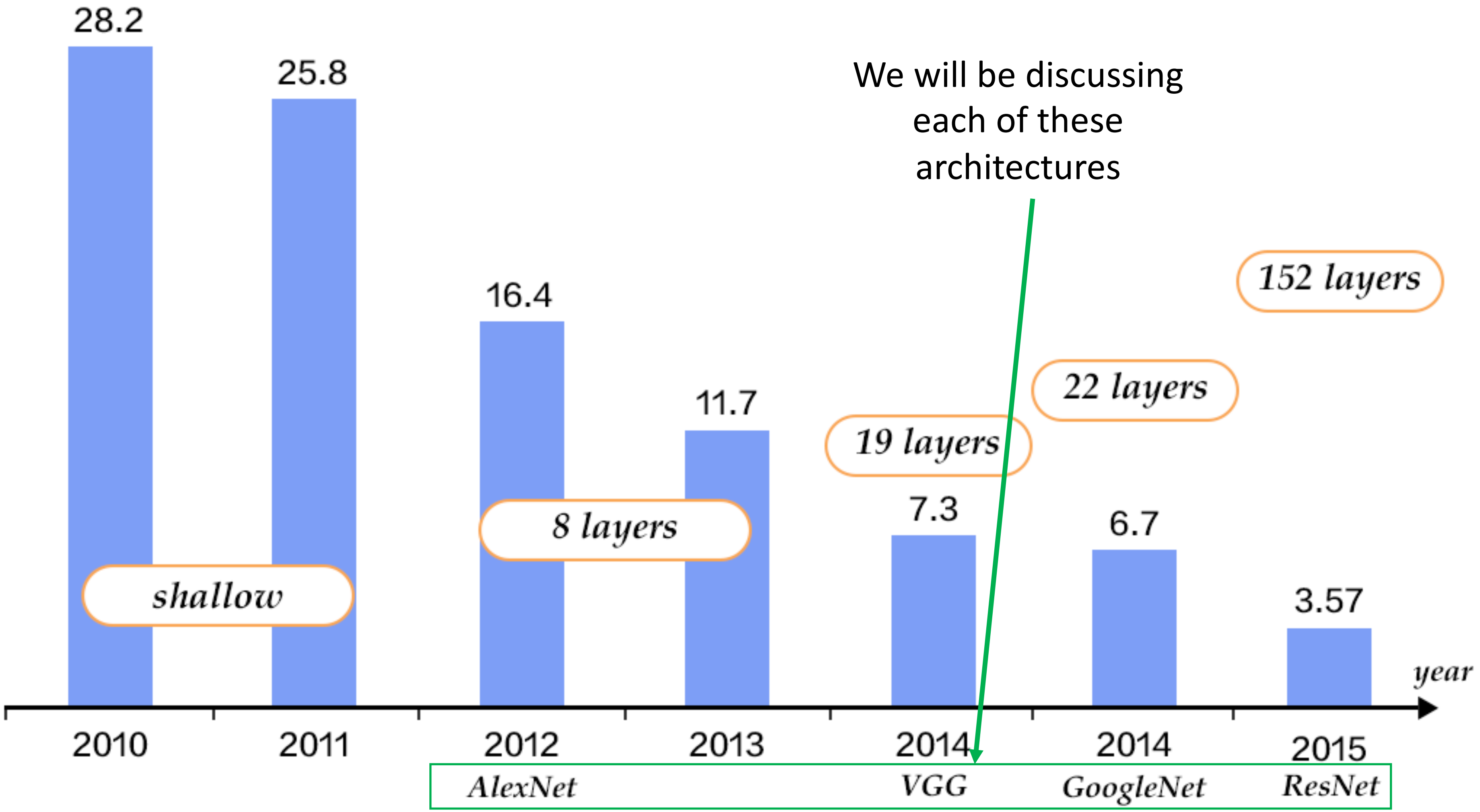
ILSVRC Challenge

- International detection and classification challenge begun in 2010 and completed in 2017
    - [Link]
- Provide 1.2M labelled image subset of “ImageNet” (a 15M+ image dataset) for training statistical models for various recognition tasks
- 2012 saw deep learning burst into the collective consciousness when a deep neural network called “AlexNet” **<u>absolutely crushed</u>** the previous top results for image classification
- Since then, deep neural networks have led to even more dramatic improvements

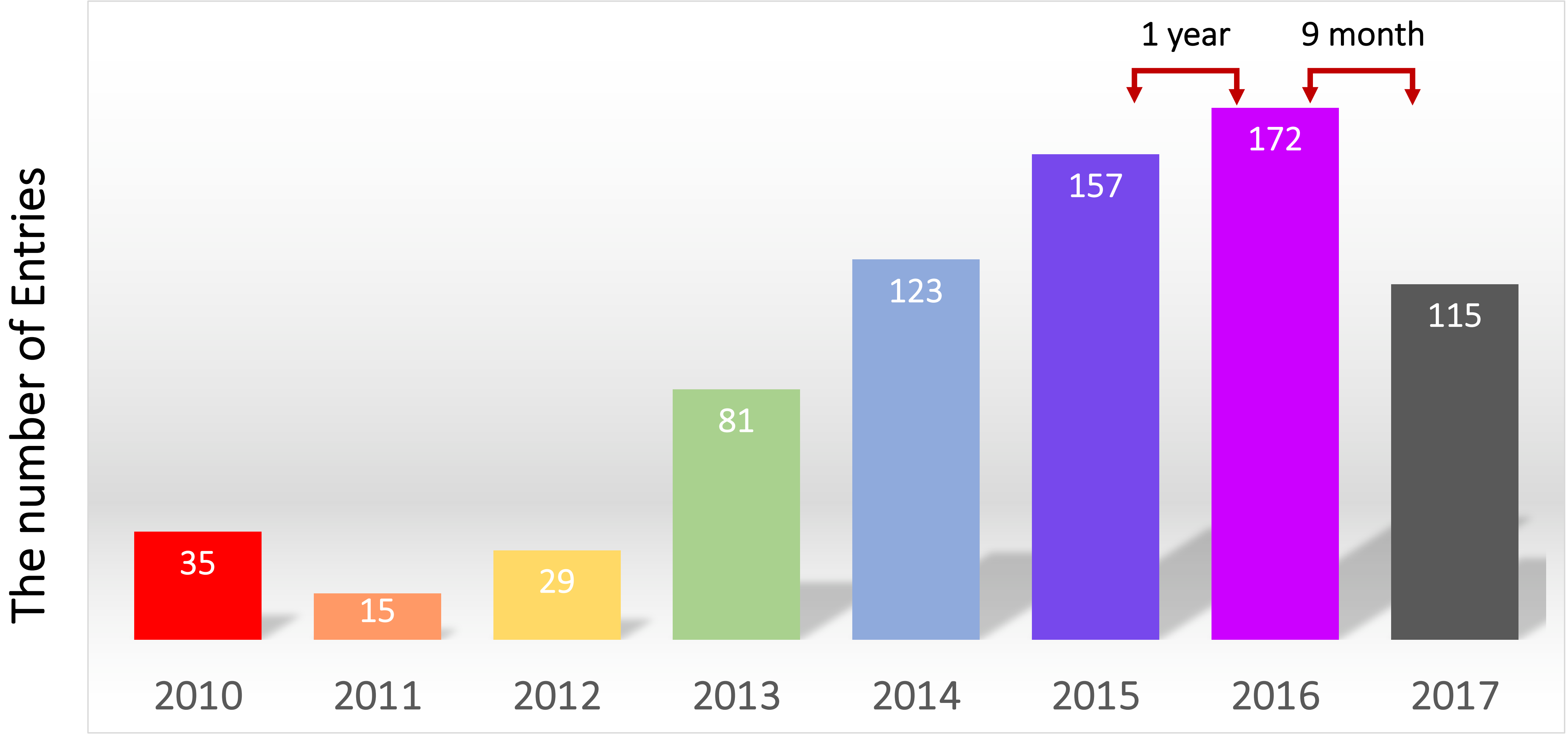
Classification Results (CLS)



ILSVRC Challenge



Participation in ILSVRC over the years



Frameworks for Deep Learning

- Theano
  - One of the first frameworks to pop up began as an numerical optimization and machine learning library from University of Montreal in 2010
  - Admins recently announced they would not longer be maintaining Theano due to competition from industry frameworks
- Caffe
  - Developed at Berkeley in 2013, was the framework of choice for many in computer vision from around 2014-2016
  - Still maintained but has fallen out of favor recently
- TensorFlow
  - Google's horse in the game, released 2015
  - Fairly low level control of network design, but can be used with Keras to give higher-level control
  - Supports C++ and Python interfaces
- Torch/PyTorch
  - Torch is another framework that predates AlexNet, being released way back in 2002 for scientific computing tasks
  - Gained popularity for use with neural networks, but was hampered by its language of choice: Lua
  - A Python interface, PyTorch, was recently released (early 2017) and has seen a rapid increase in popularity do to the high-level of control it grants developers
- Many others (DeepLerning4j, MatConvNet, Lasagne, Paddle, etc.)

AlexNet (2012)

The one that started it all.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

AlexNet

- 8 layer convolutional neural network for image classification
- Massive improvement over the then-state-of-the-art in the 2012 ILSVRC Challenge

Label: Steel drum

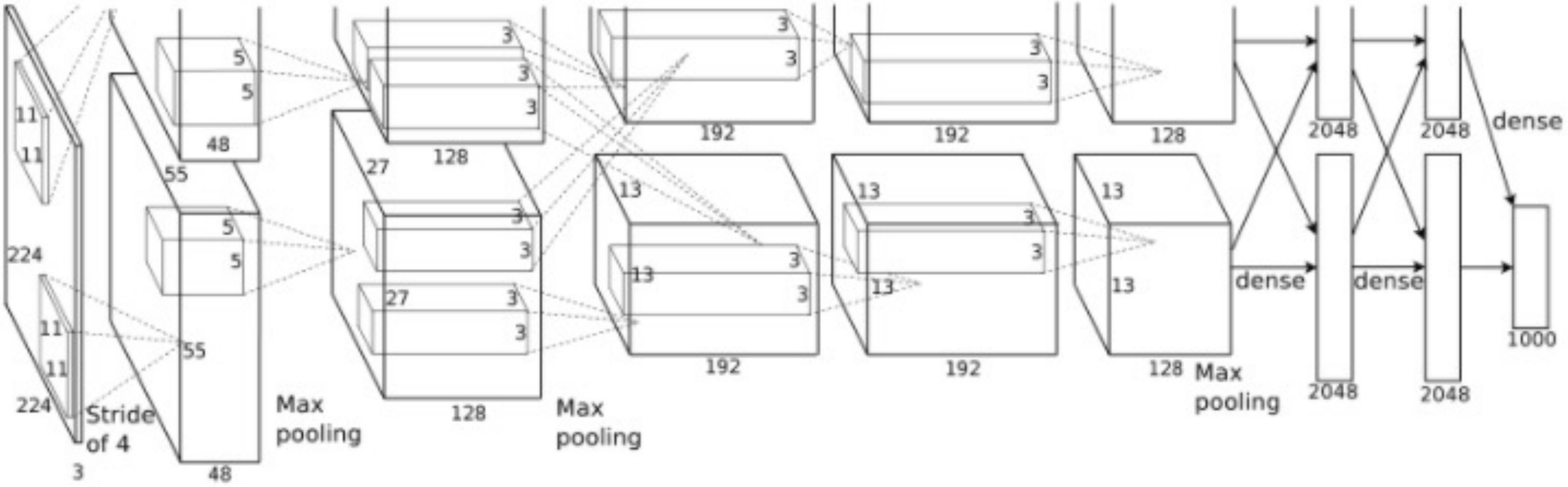
1000 object classes

1,431,167 images

AlexNet

- Designed with 8 layers:
  - 5 convolutional layers
  - 3 fully connected (FC) layers
- Final FC layer fed into 1000 node “prediction” layer
  - 1 node for each class
- Output a classification probability (in range [0, 1]) for each class
- Used many of the hallmarks of modern networks
  - Training set mean-subtraction pre-processing
  - ReLU nonlinearity
  - Overlapping pooling
  - Local response normalization layers
  - Strided kernels
  - Dropout

AlexNet



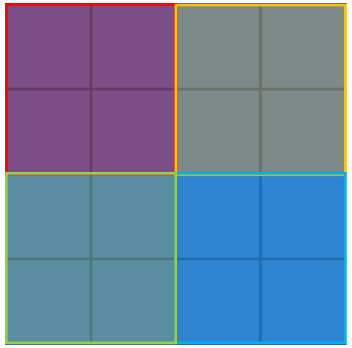
11
11
224
224
11
11
3
Stride
of 4
5
5
48
55
5
5
55
48
Max
pooling
3
3
3
128
27
3
3
27
3
3
128
Max
pooling
3
3
192
13
13
3
3
192
3
3
192
13
13
3
3
192
128
13
13
128
Max
pooling
dense
2048
2048
dense
2048
2048
dense
1000

AlexNet Insights: Data Augmentation

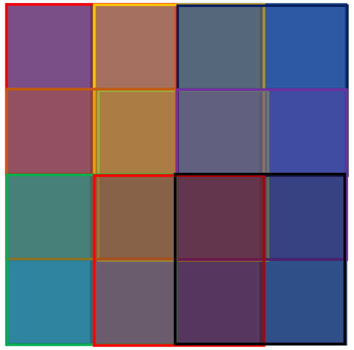
- Deep networks have many, many, many parameters
  - Remember: When $data\ size \approx parameter\ size$ we overfit statistical models
- Image set in dataset are resized to 256x256
- Random 224x224 crops are taken, along with their horizontal reflection
  - Increases dataset by factor of 2048
- Test-time predictions are the average scores for 5 crops and their reflections

AlexNet Insights: Overlapping Pooling

- Spatial pooling useful for translational invariance (aggregating information from regions)



- Traditional local pooling used grids



- AlexNet found improved generalization with overlapping pooling

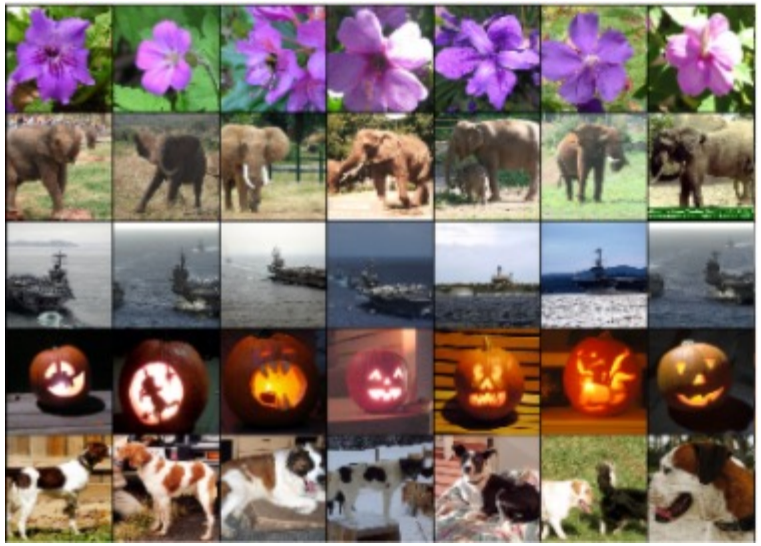
AlexNet Insights: Other

- Less over-fitting through 50% dropout
- Weight-decay not only serves to regularize (i.e. keep the model general), but also reduced training error
- Local response normalization (normalizing layer activations by a function over all activations at that layer) not required with ReLU, but it can aid generalization

AlexNet Insights: Other

- High classification accuracy even for cropped and/or off-center objects

- Last “hidden layer” (i.e. 4096 dimensional layer before prediction is made) is a good feature encoding
  - Images can be grouped together based on Euclidean distance using this vector



VGG (2014)

Size matters…in multiple ways

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

VGG

- Deeper networks (11-19 layers, depending on the design)

- **Main idea: deeper layers, smaller kernels**

- 8-layer AlexNet had 60M parameters, but successors that added more layers had 140M+

- VGG models offer 130-140M parameter designs but with as many as 19 layers
  - Largest filter size is 3x3 (as opposed to the 5x5, 7x7, and larger kernels of earlier networks)

VGG

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					

Most commonly used

VGG Insights: Training

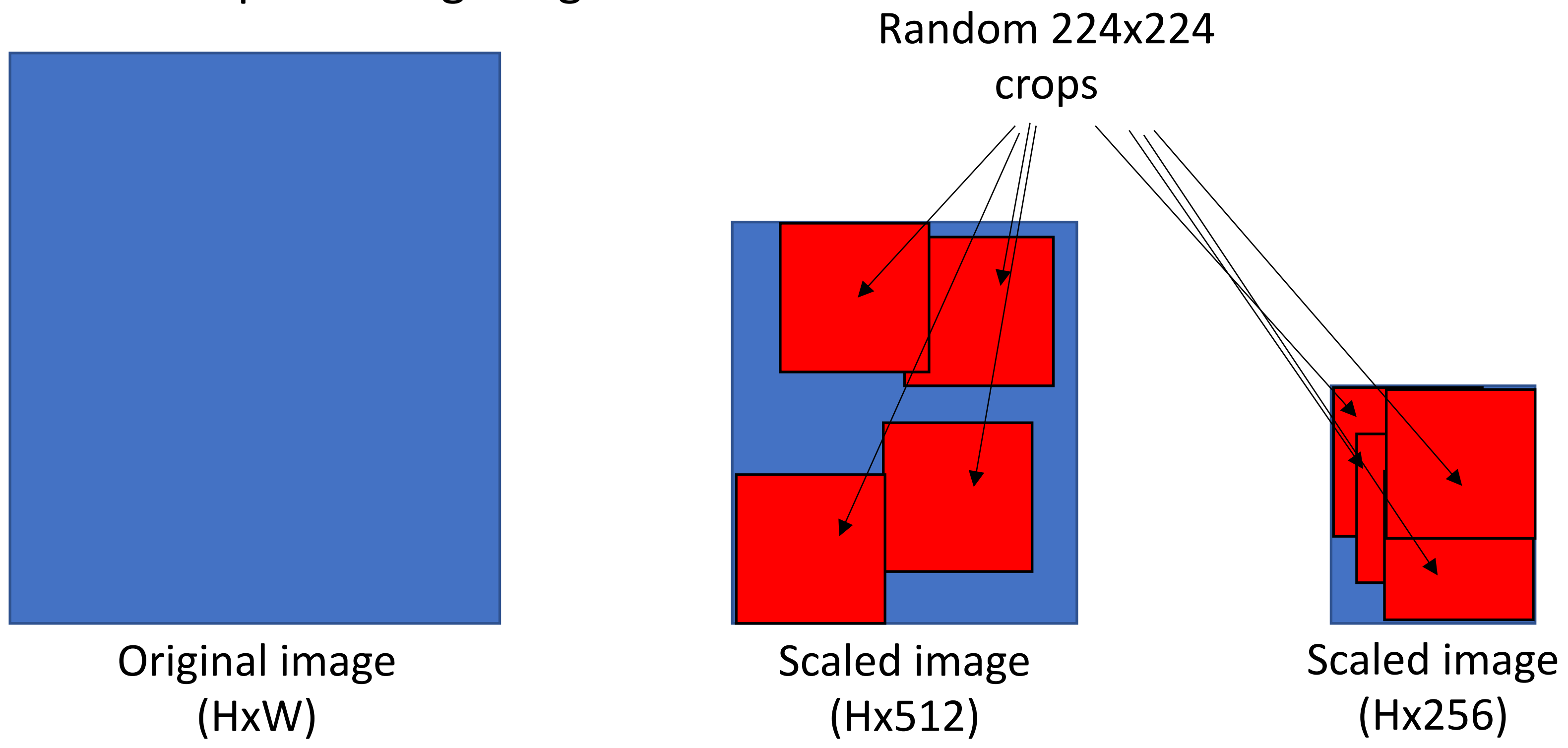
- Training followed same practices as AlexNet

- Found that weight initializations had larger effect for deeper networks
  - In such complex models, initial conditions can often lead to getting stuck in local minima during optimization

- Trained subset of network first and use those parameters to initialize deeper networks
  - First trained 11-layer network, then used those weights to initialize deeper networks

VGG Insights: Data Augmentation

- Augmented data as in AlexNet, but added "scale jittering"
  - Resize to difference scales (e.g. shortest side of image = {256, 512}) and randomly crop from within those
  - Leads to processing images at different scales



VGG Insights: Evaluation

- Error decreases with extra depth
  - 16-layer Network C consistently better than 13-layer Network B
- Network D (16 layers, all 3x3 kernels) consistently outperforms Network C (16 layers, last few are 1x1 kernels)
  - Implication: non-trivial receptive fields are important for spatial context
- Scale jittering at test time appears to lead to better results due to robustness to scale changes
- Deeper network with small kernels (e.g. 3x3) perform better than shallower networks with larger kernels (e.g. 7x7)

GoogleNet (2015)

Going deeper with convolutions

Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

GoogleNet

- 22 layer network
- **Main idea: deeper and wider network, without extra computational costs**
- Introduce “inception module,” a network-in-network approach
  - Idea is that these “micro-networks” enhance local modeling
- Well established principle: deeper is generally better
  - But it rapidly increases parameters which need more (hard to obtain) labeled data
  - Dramatically increases computation requirements too
    - Ex. For two, chained convolutional layers computation increasing quadratically with increase in number of filters
  - Deeper is not always better if the dense filter weights are often 0’s

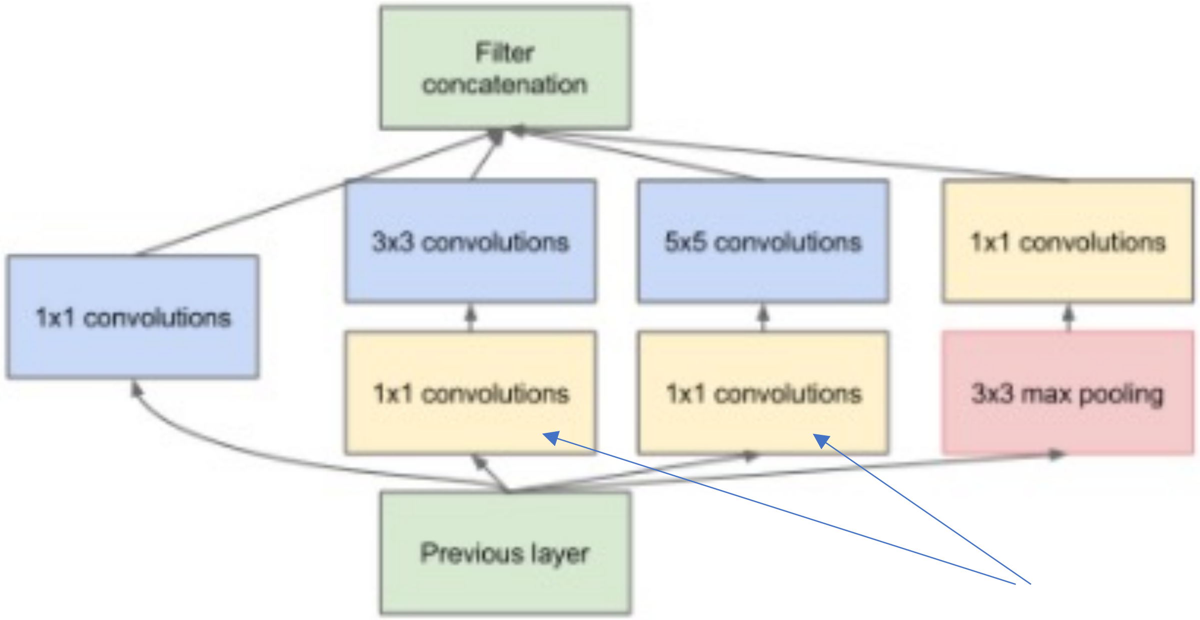
GoogleNet

- Intuition: neural net theory suggests sparsity is better for learning, but quirks of parallel computation have led folks to use dense networks

- Goal: mimic locally sparse structure with dense components

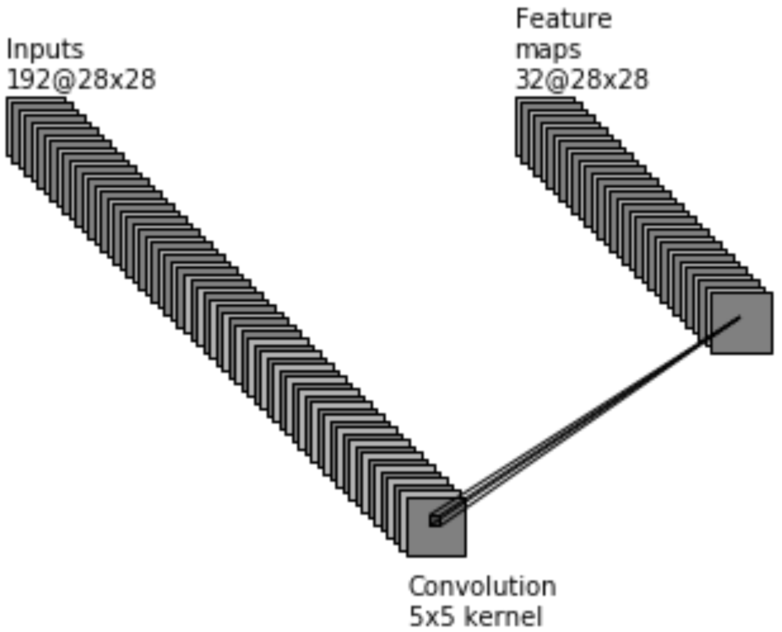
- Design “inception models” that first compress information and then learn to aggregate

GoogleNet: Inception Module

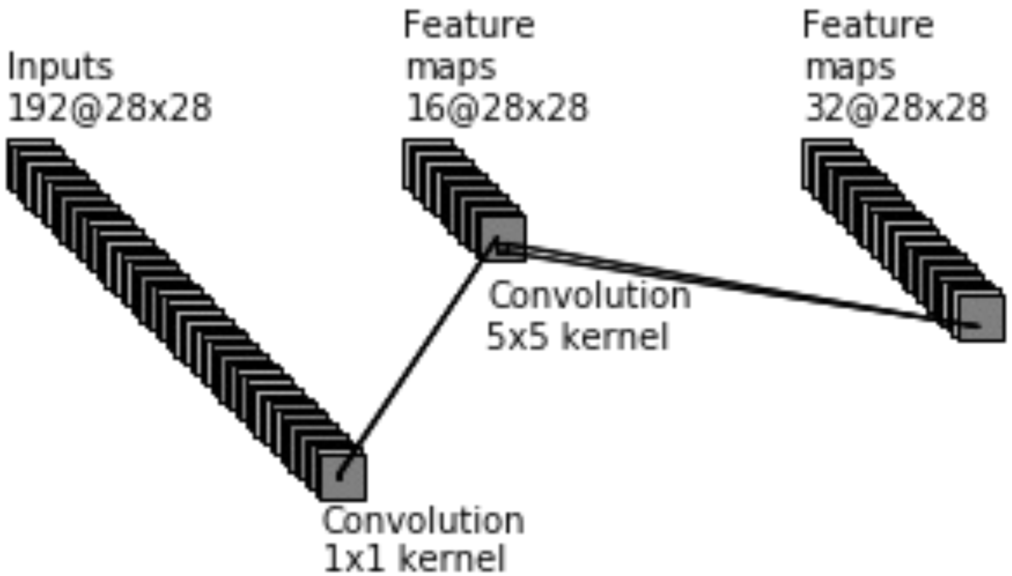


1x1 convolutions reduce dimensionality (embedding) before more expensive filter computations

GoogleNet: Inception Module



$$5^2(28)^2(192)(32) = 120,422,400 \text{ operations.}$$



$$[(1^2)(28^2)(192)(16)]+[(5^2)(28^2)(16)(32)] =$$
$$2,408,448+10,035,200 = 12,443,648 \text{ operations.}$$

GoogleNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

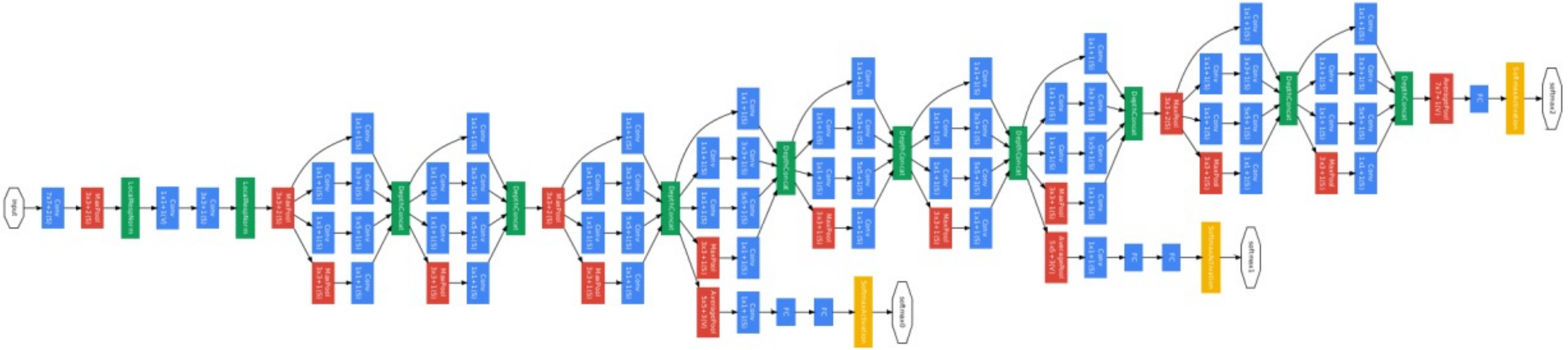
GoogleNet Insights

- Trained 7 models and made predictions as ensemble to achieve higher results

- Used a more varied scale jittering and formal selection of crops for input images
  - Suggest that cropping has diminishing returns

- Results suggest that replicating sparsity with dense components **works**
  - Significant gains with only modest additional computation for deeper network

GoogleNet Structure



ResNet (2016)

How deep can we go?

He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Deep Residual Networks (ResNets)

- A simple and clean framework of training “very” deep nets
- State-of-the-art performance for
  - Image classification
  - Object detection
  - Semantic segmentation
  - and more...

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

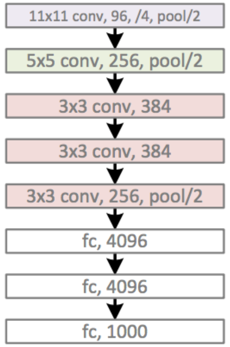
ResNets @ ILSVRC & COCO 2015 Competitions

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

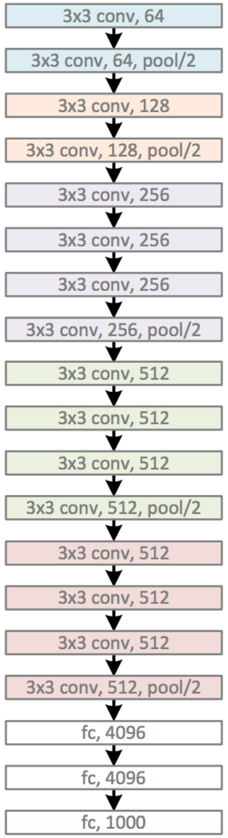
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Revolution of Depth

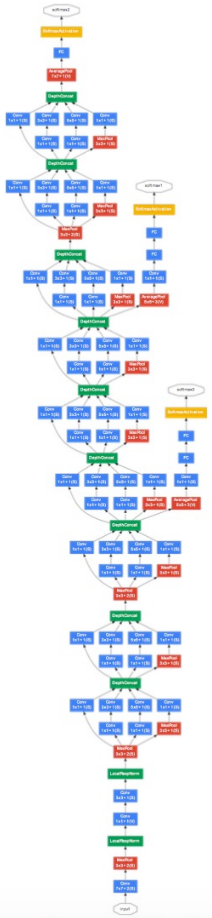
AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



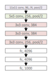
GoogleNet, 22 layers
(ILSVRC 2014)



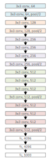
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Revolution of Depth

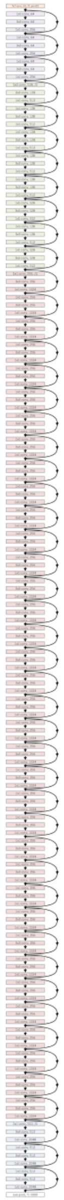
AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



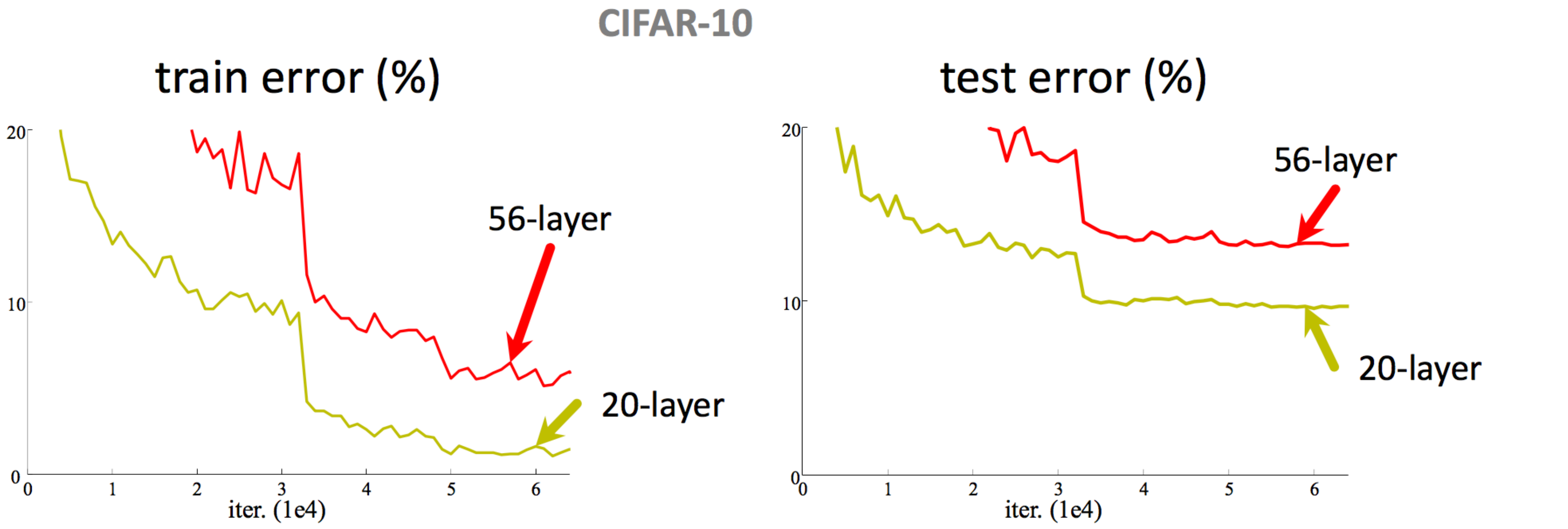
ResNet, 152 layers
(ILSVRC 2015)



Is learning better networks as simple as stacking more layers? ……No

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

Simply stacking layers?



- Plain nets: stacking 3x3 Convlayers...
- 56-layer net has higher training error and test error than 20-layer net

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

ResNet

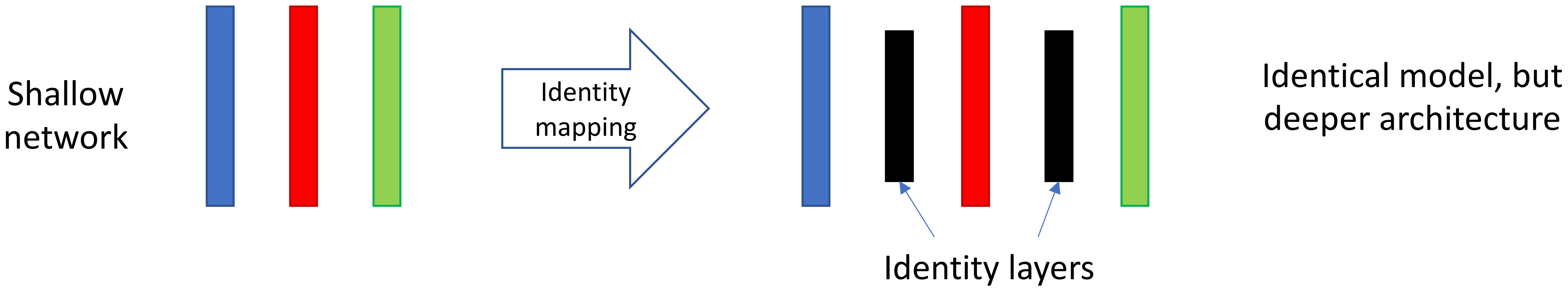
- Opening line: “Deeper neural networks are more difficult to train.”

- Accuracy in deeper networks degrades with depth
  - **Not due to overfitting** (training error increases)

- Due to poor optimization strategies for such complex models

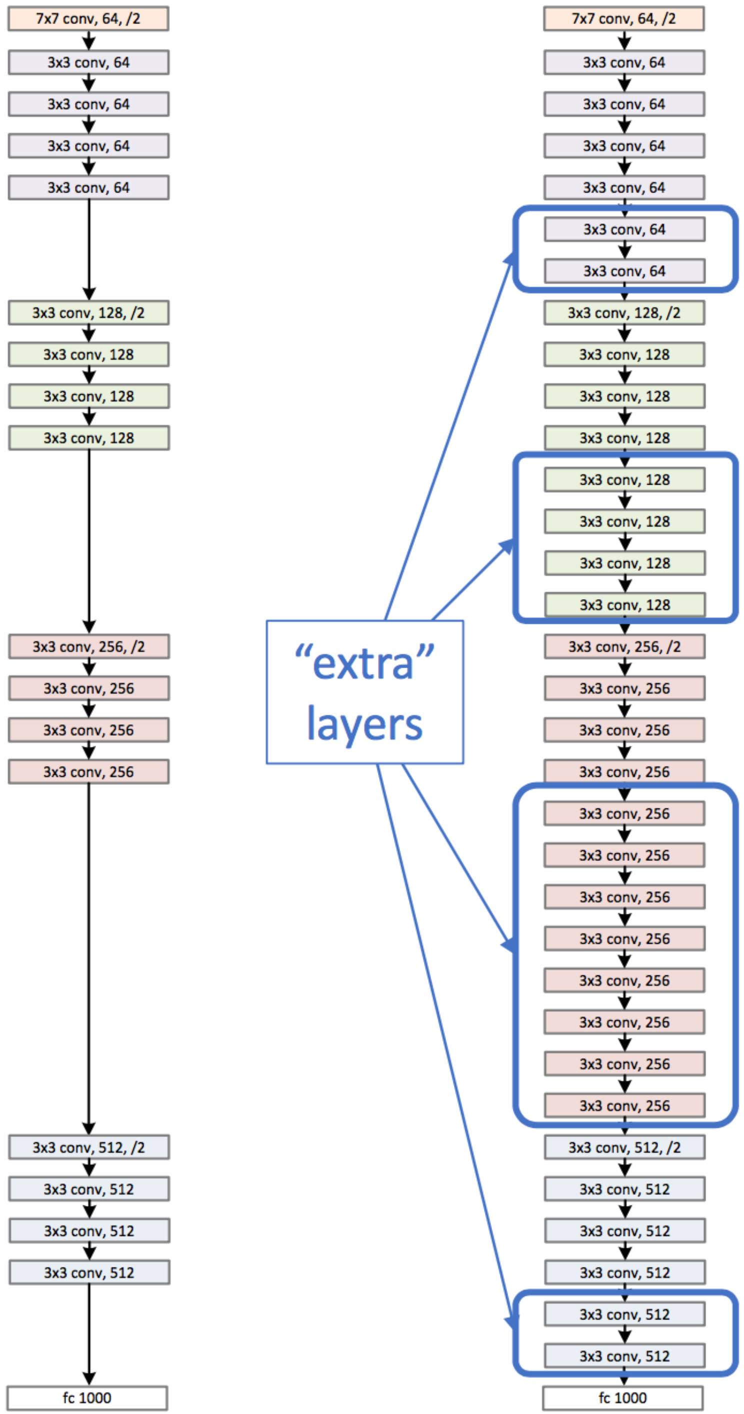
ResNet

- **Thought experiment:** a deep network constructed by “mapping” a shallower network should not have these problems



- In theory, this deeper network could be learned through optimization
  - Hence training should be equivalent irrespective of depth

- Experiments suggest this doesn’t happen



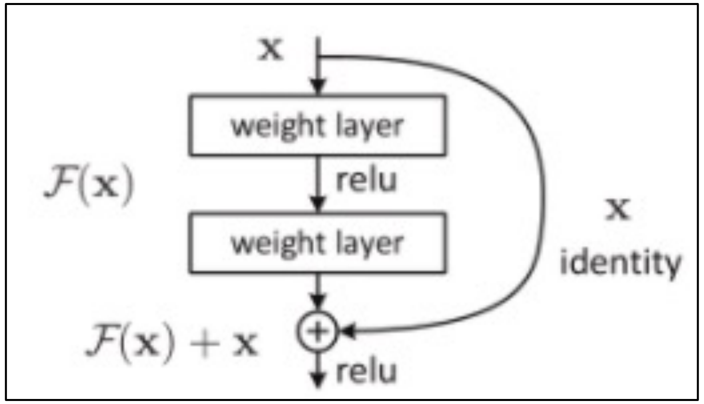
Left : a shallower model (18 layers)
Right : a deeper counterpart (34 layers)

- A deeper model should not have **higher training error**
- A solution by construction:
  - original layers: copied from a learned shallower model
  - extra layers: set as **identity**
  - at least the same training error
- **Optimization difficulties**: solvers cannot find the solution when going deeper…

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

ResNet

- Solution: explicitly construct this shallow-to-deep mapping



- Represent the desired mapping as $H(x)$
- Define stacked layers mapping as $F(x) = H(x) - x$
- Result is $H(x) = F(x) + x$

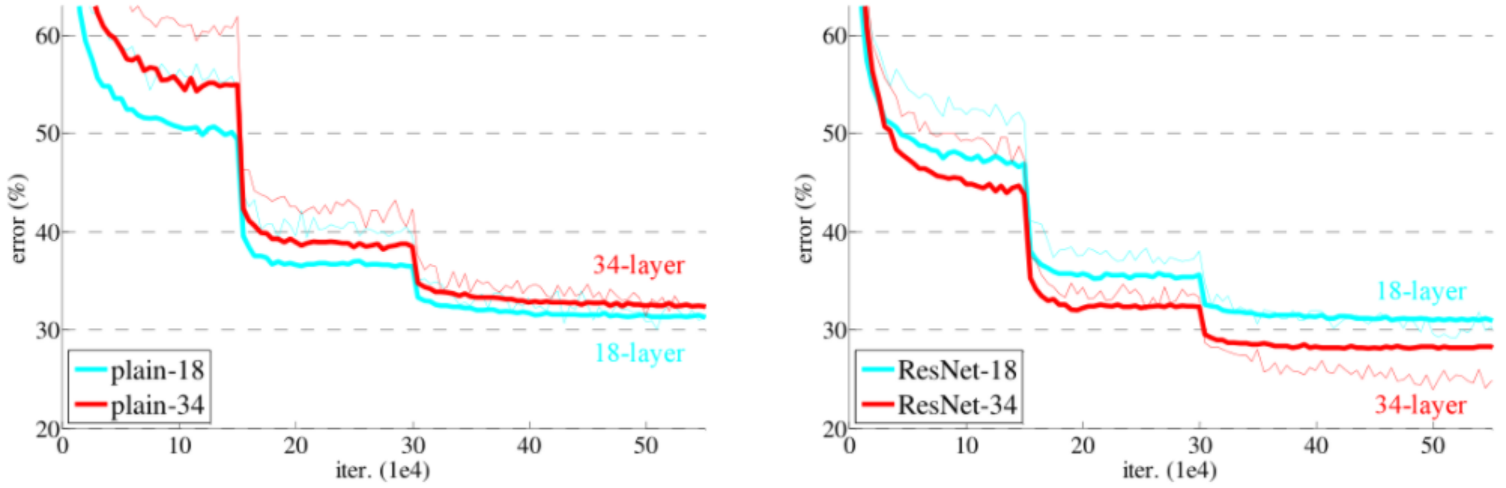
- Intuition: if identity mapping is optimal, residual $F(x)$ will go to 0
  - Easier optimization problem than "hoping" certain layers will become identity

ResNet

- Roughly the same design principles as VGG
  - Compare to VGG-19
- Standard SGD training
- Batch normalization after each convolution
- Cropping and augmentation follows VGG

ResNet Insights: Performance

- 34-layer "plain" network suffers from performance degradation at depth while 34-layer does not



Thin curves are training error
Bold curves are validation error

- Not likely due to vanishing gradients as the plain network is also batch normalized

ResNet Insights: Evaluation

- Design is less computationally heavy
  - 34 layer ResNet has 3.6B FLOPS (multiply-adds)
  - VGG-19 has 19.6B
- Can get very deep without performance degradation
  - In fact, 152-layer ResNet ensemble won ILSVRC 2015
- Authors got aggressive: trained a 1,202-layer network
  - No problems with training, but poor results
  - This time, likely due to overfitting

Summary

What have we learned?

AlexNet

- Deep convolutional neural networks work for recognition tasks

- Data augmentation is a valid training technique for deep CNNs to prevent overfitting

- Weight-decay is useful for generalization as well as regularization

- Overlapping pooling (stride < half-width) is useful

VGG

- Better performance with more layers, smaller kernels
- Weight initialization plays a role in performance of deep networks
- Scale robustness can be achieved through scale jittering

GoogleNet

- Inception modules can help you get deeper without extra computation
- Local sparsity can be modeled with dense operations
- Ensemble methods are applicable to neural networks

ResNet

- Hard to get very deep with traditional methods due to local optimization shortcomings
- Designing to an explicit residual mapping can resolve training problems
- Very, very deep networks can be trained…if only we had the data to prevent overfitting

References and Helpful Links

- Stanford CS231n course on CNNs for Visual Recognition
  by Andrej Karpathy and Justin Johnson
- Deep Learning in Neural Networks: An Overview
  by Jürgen Schmidhuber (many good references)
- A 'Brief' History of Neural Nets and Deep Learning
  by Andrey Kurenkov (blog series)
- A Brief History of Neural Network Architectures
  by Eugenio Culurciello (blog post)