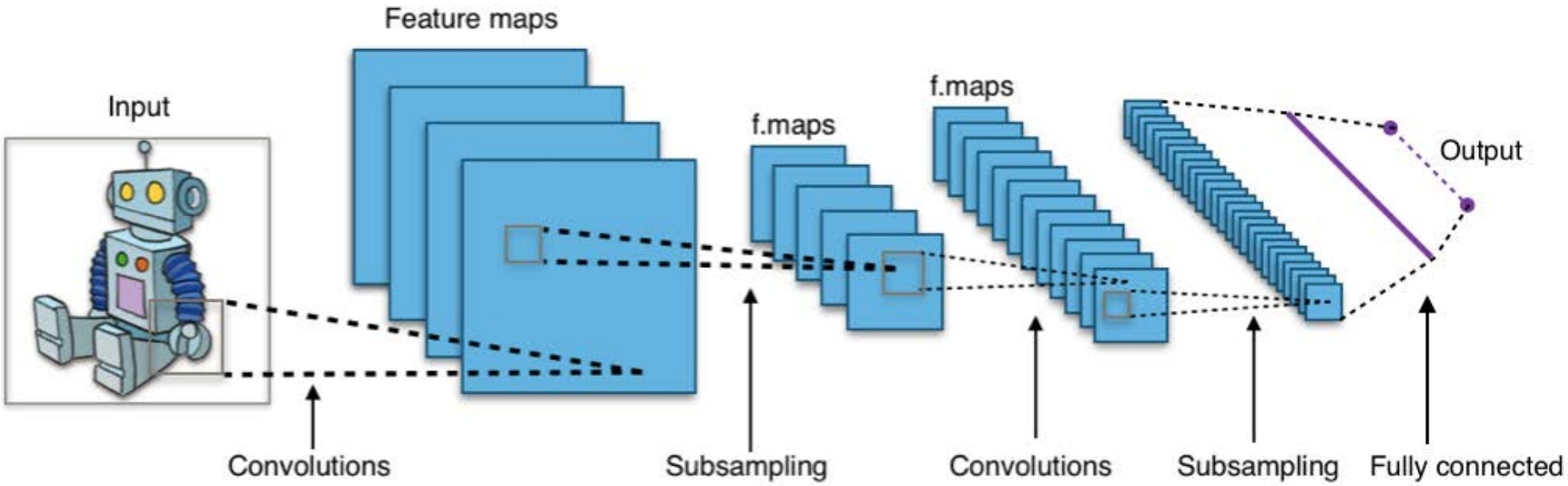
Convolutional Neural Networks

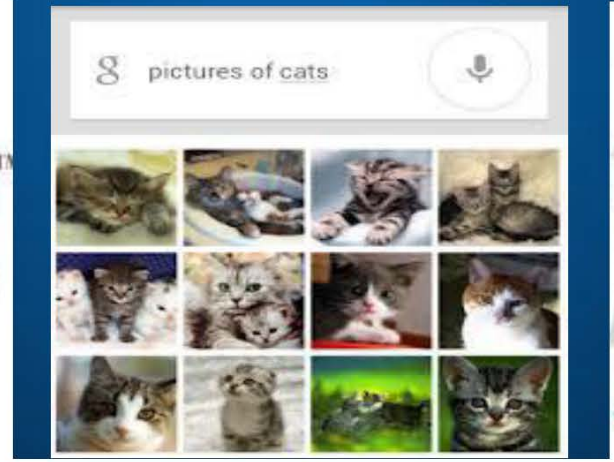


Many slides from Lihi Zelnik, Derek Hoiem, Lana Lazebnik, Jia-bin Huang and Kaiming He, Noah Snavely, Ioannis Gkioulekas, Stanford cs231n

Products

Audio recognition

Image retrieval



Inverse image search

Classification, auto-tagging

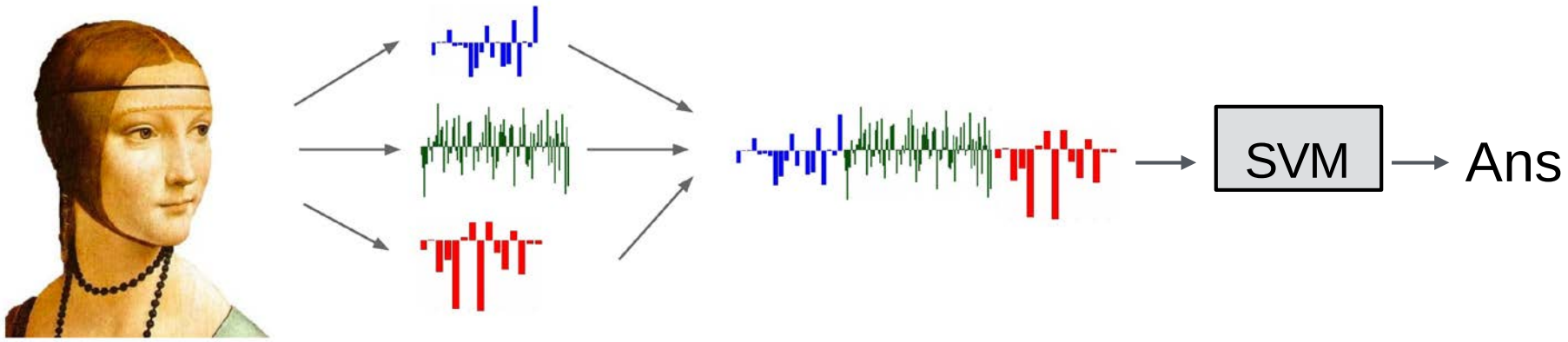
(Unrelated) Dog vs Food

[Karen Zack, @teenybiscuit]

(Unrelated) Dog vs Food

[Karen Zack, @teenybiscuit]

Recap: Before Deep Learning



Input Pixels | *Extract Features* | *Concatenate into a vector **x*** | *Linear Classifier*

Figure: Karpathy 2016

The last layer of (most) CNNs are linear classifiers

Input Pixels

Perform everything with a big neural network, trained end-to-end

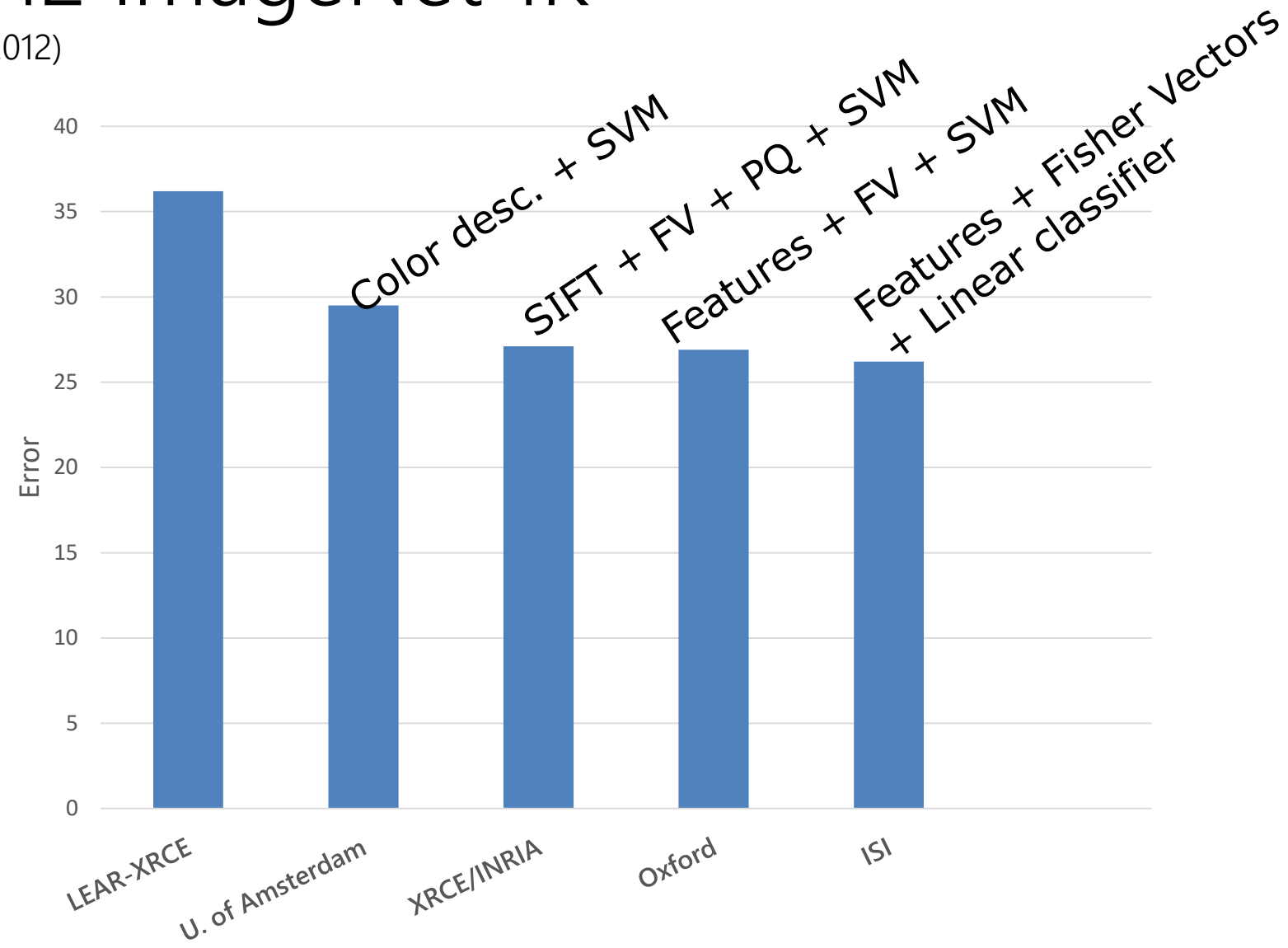
Key: perform enough processing so that by the time you get to the end of the network, the classes are linearly separable

History of deep convolutional nets

- 1950's: neural nets (perceptron) invented by Rosenblatt
- 1980's/1990's: Neural nets are popularized and then abandoned as being interesting idea but impossible to optimize or "unprincipled"
- 1990's: LeCun achieves state-of-art performance on character recognition with convolutional network (main ideas of today's networks)
- 2000's: Hinton, Bottou, Bengio, LeCun, Ng, and others keep trying stuff with deep networks but without much traction/acclaim in vision
- 2010-2011: Substantial progress in some areas, but vision community still unconvinced
  - Some neural net researchers get ANGRY at being ignored/rejected
- 2012: shock at ECCV 2012 with ImageNet challenge

2012 ImageNet 1K

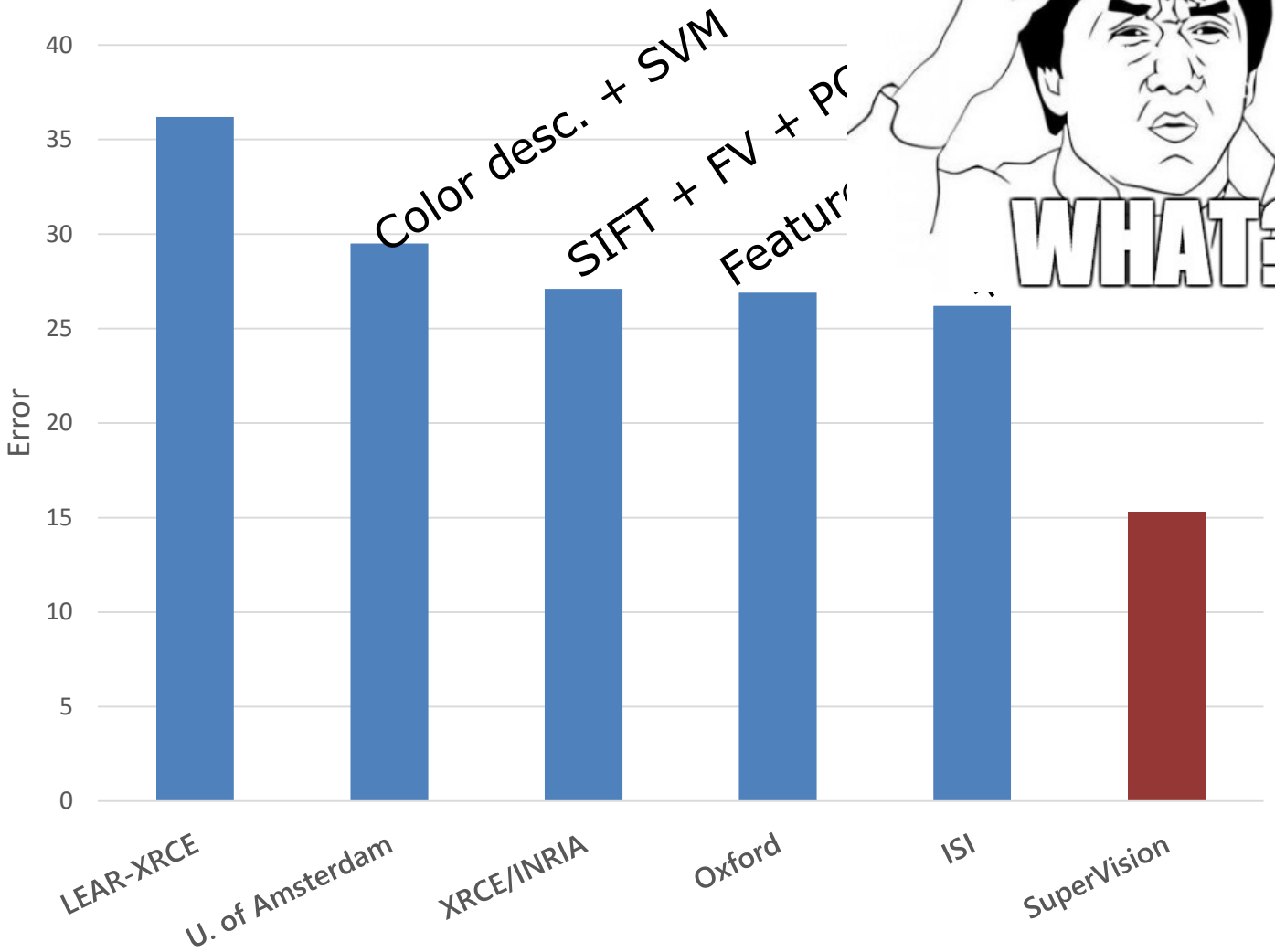
(Fall 2012)

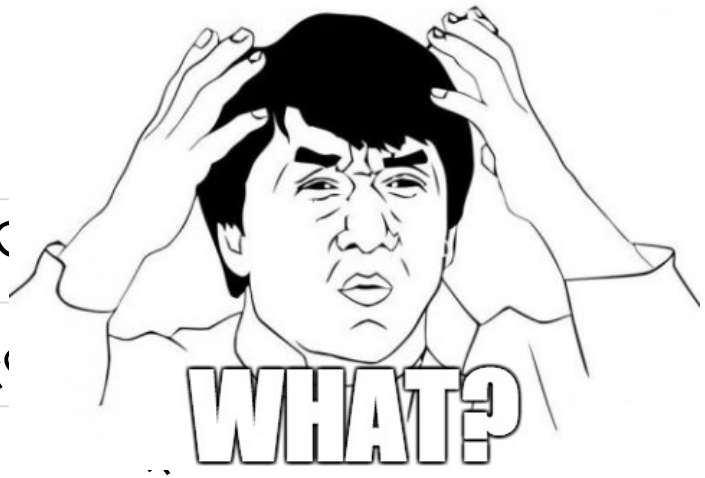


Slide: Jia-bin Huang

2012 ImageNet 1K

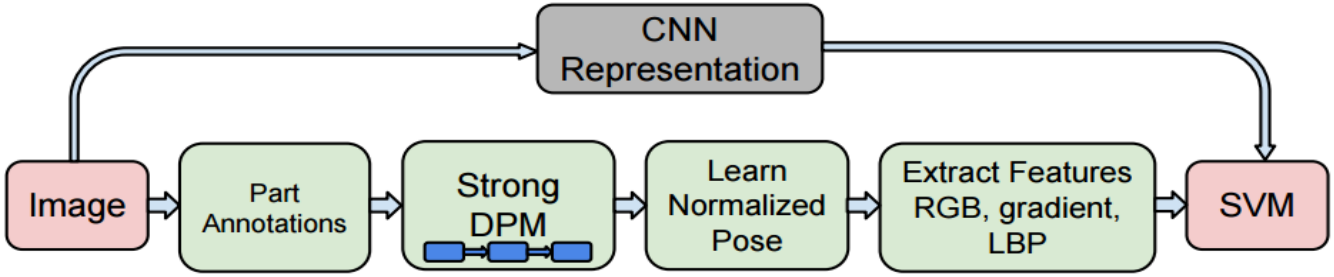
(Fall 2012)

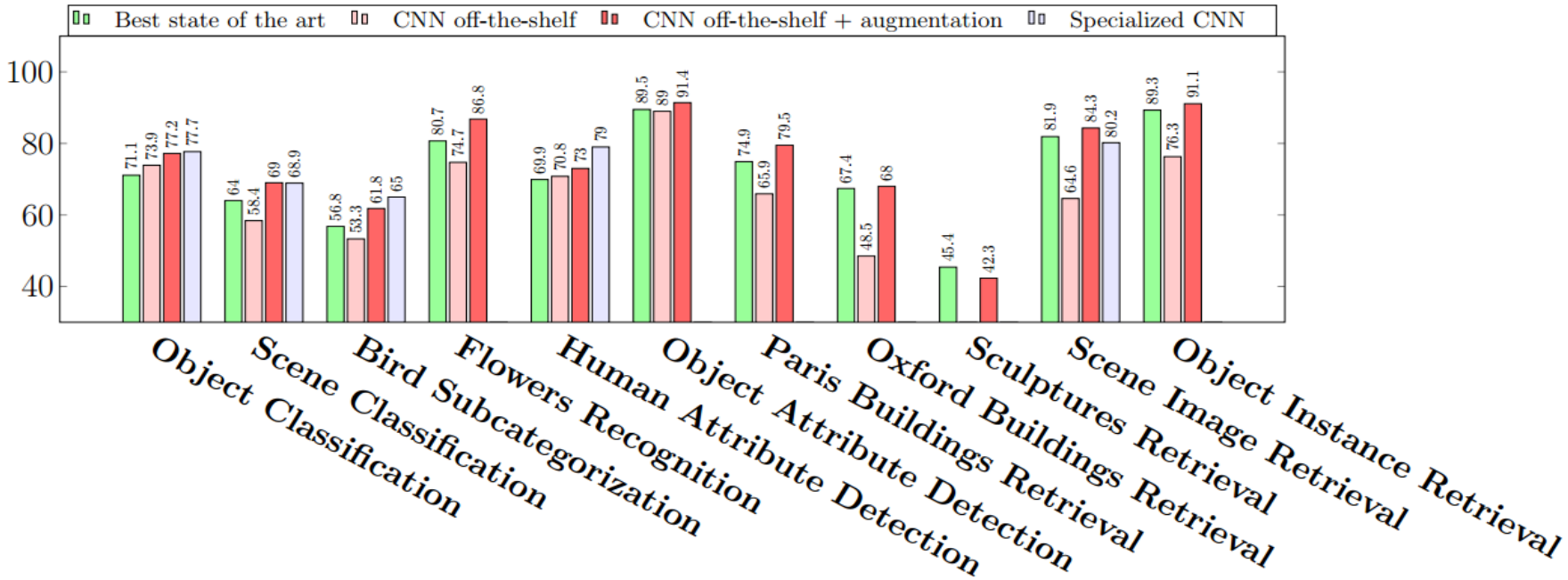




Slide: Jia-bin Huang

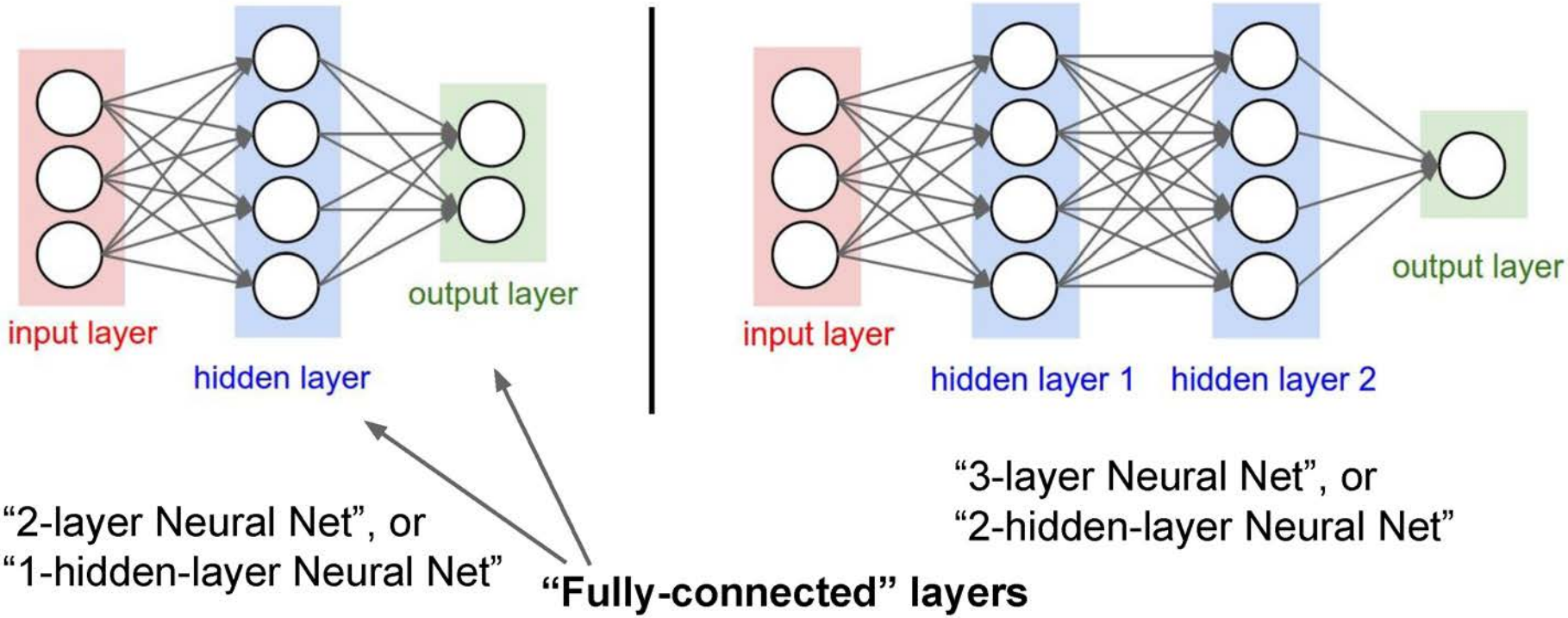
“CNN Features off-the-shelf: an Astounding Baseline for Recognition”





Razavian et al. CVPR 2014

Neural networks: Architectures



- **Deep** networks typically have many layers and potentially millions of parameters

Convolutional neural networks

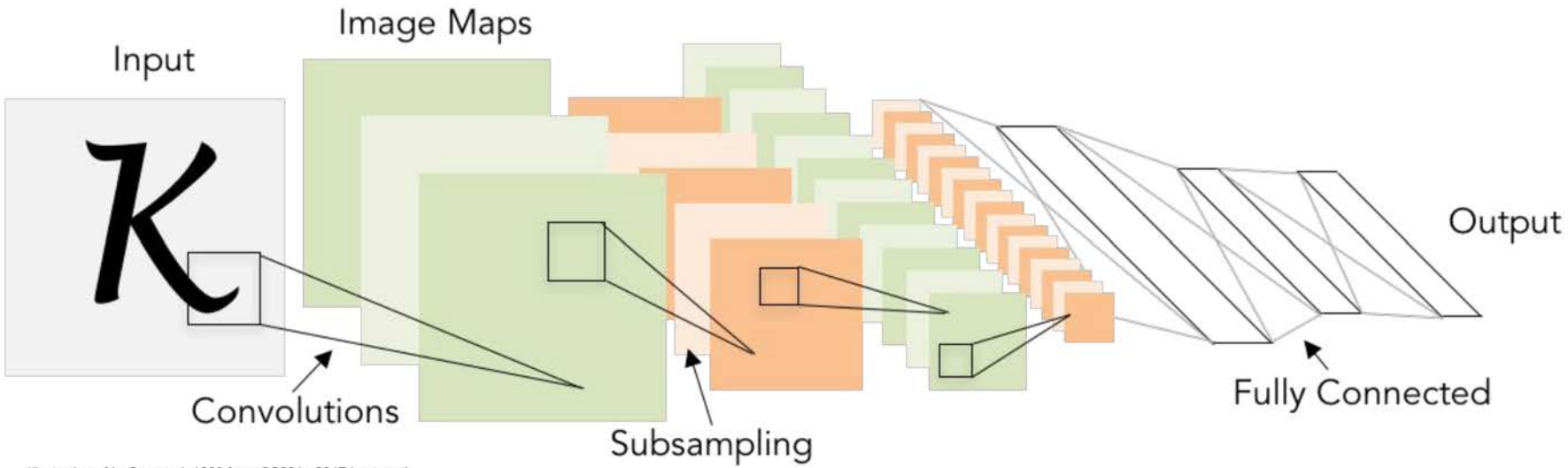


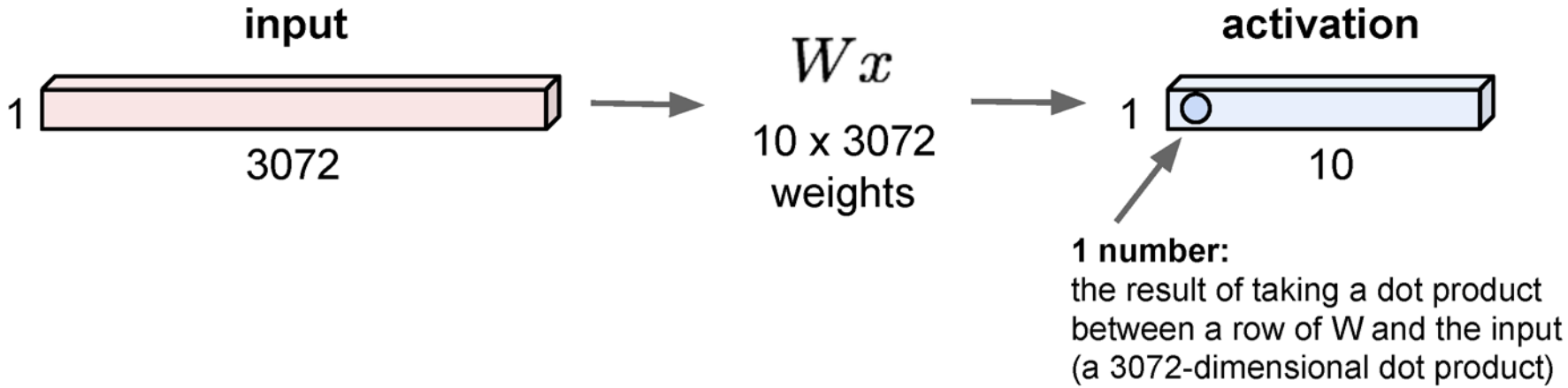
Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Convolutional neural networks

- Layer types:
  - Fully-connected layer
  - *Convolutional layer*
  - Pooling layer

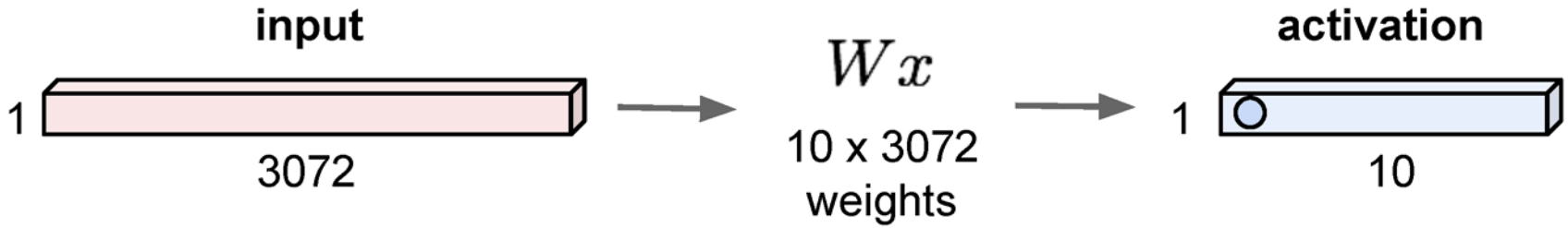
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



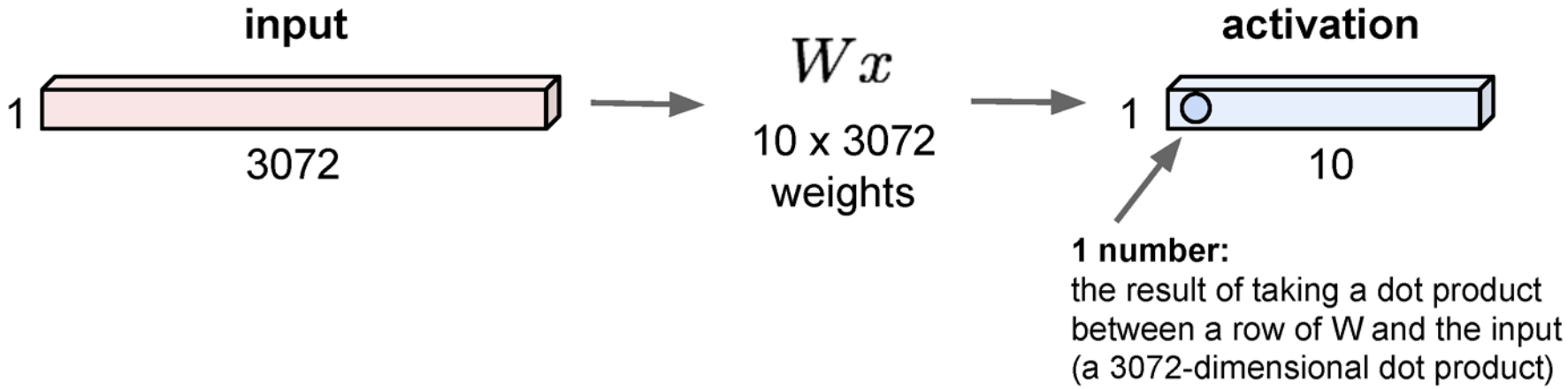
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



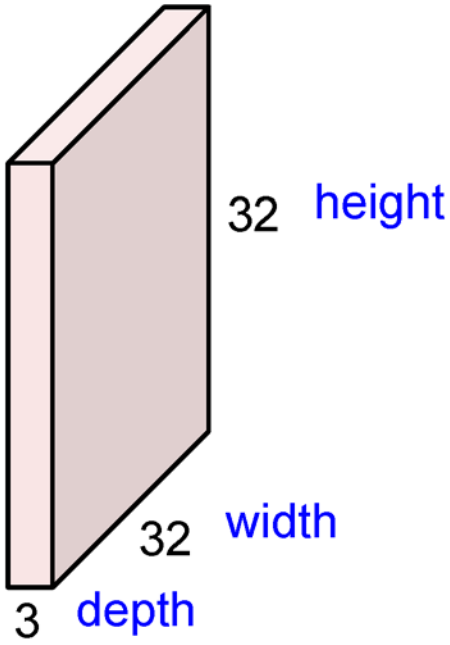
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



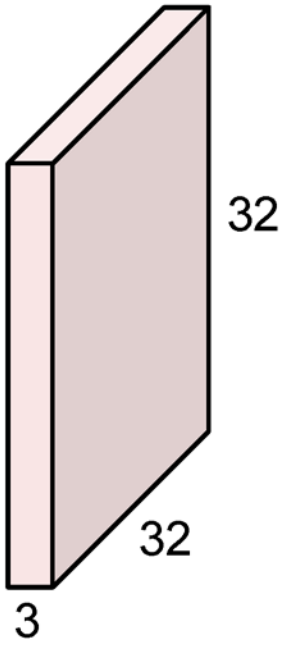
Convolution Layer

32x32x3 image -> preserve spatial structure



Convolution Layer

32x32x3 image



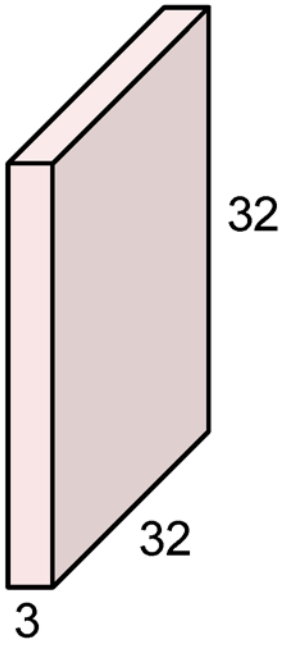
5x5x3 filter

Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

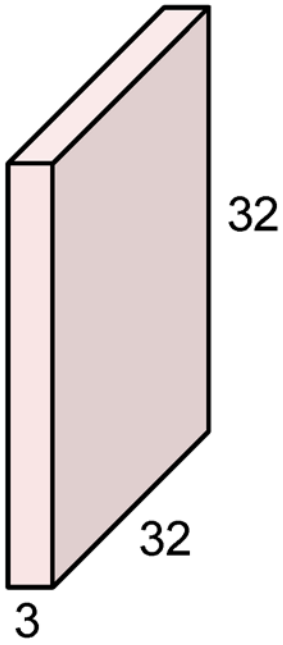


5x5x3 filter

Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer

32x32x3 image

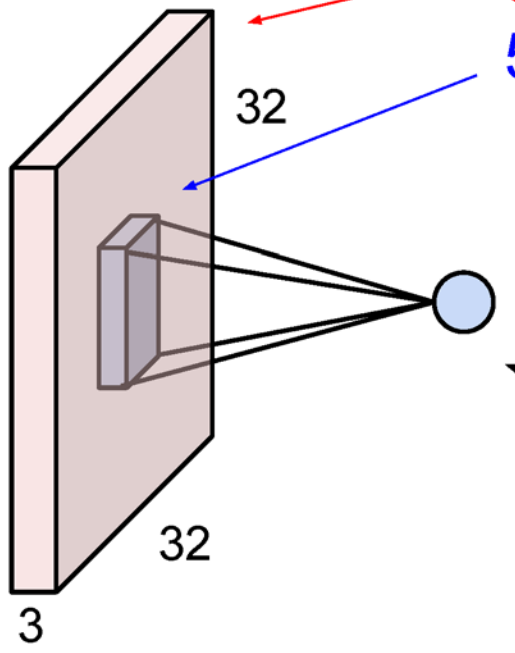


5x5x3 filter

Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Number of weights: 5 x 5 x 3 + 1 = **76**
(vs. 3072 for a fully-connected layer)
(+1 for bias)

Convolution Layer



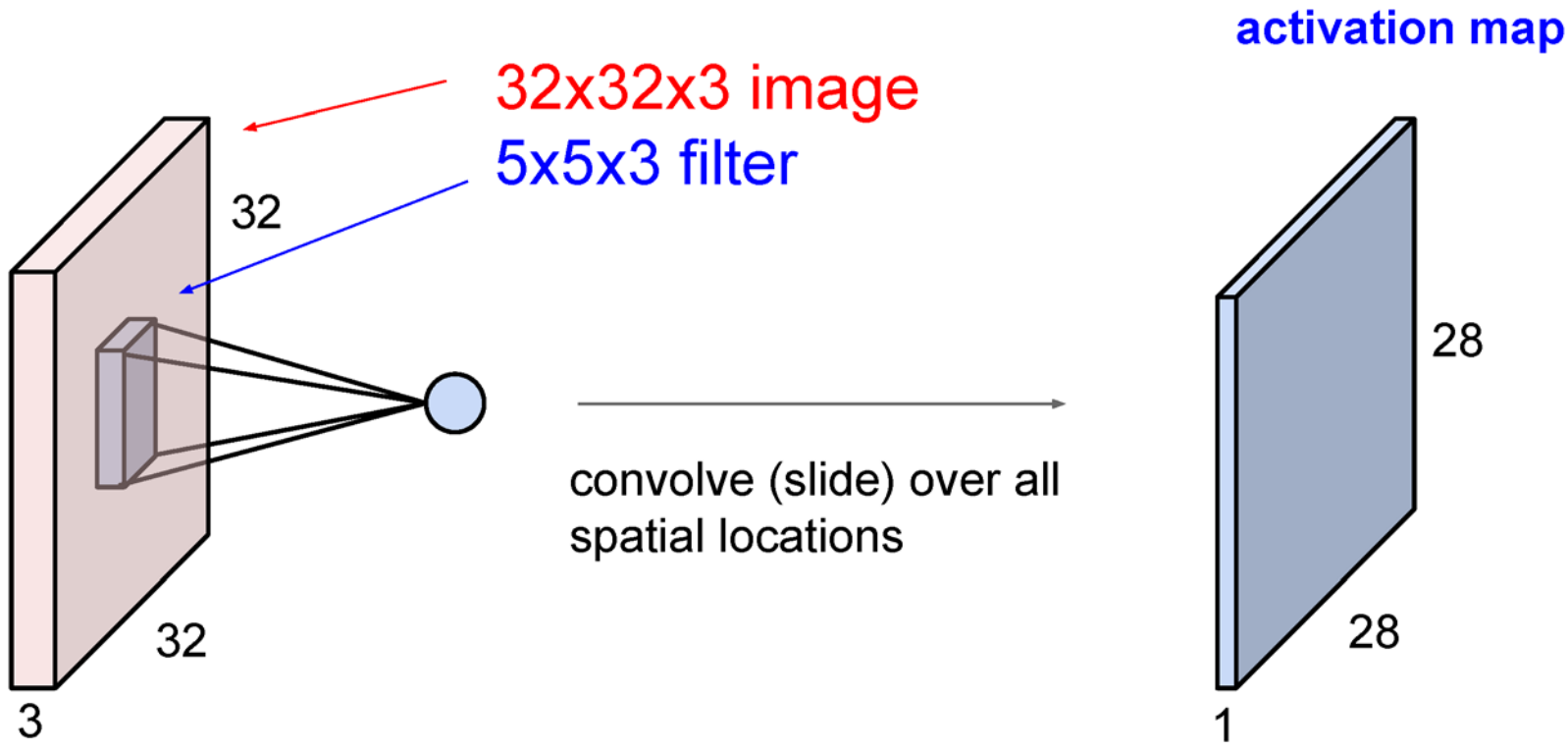
32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

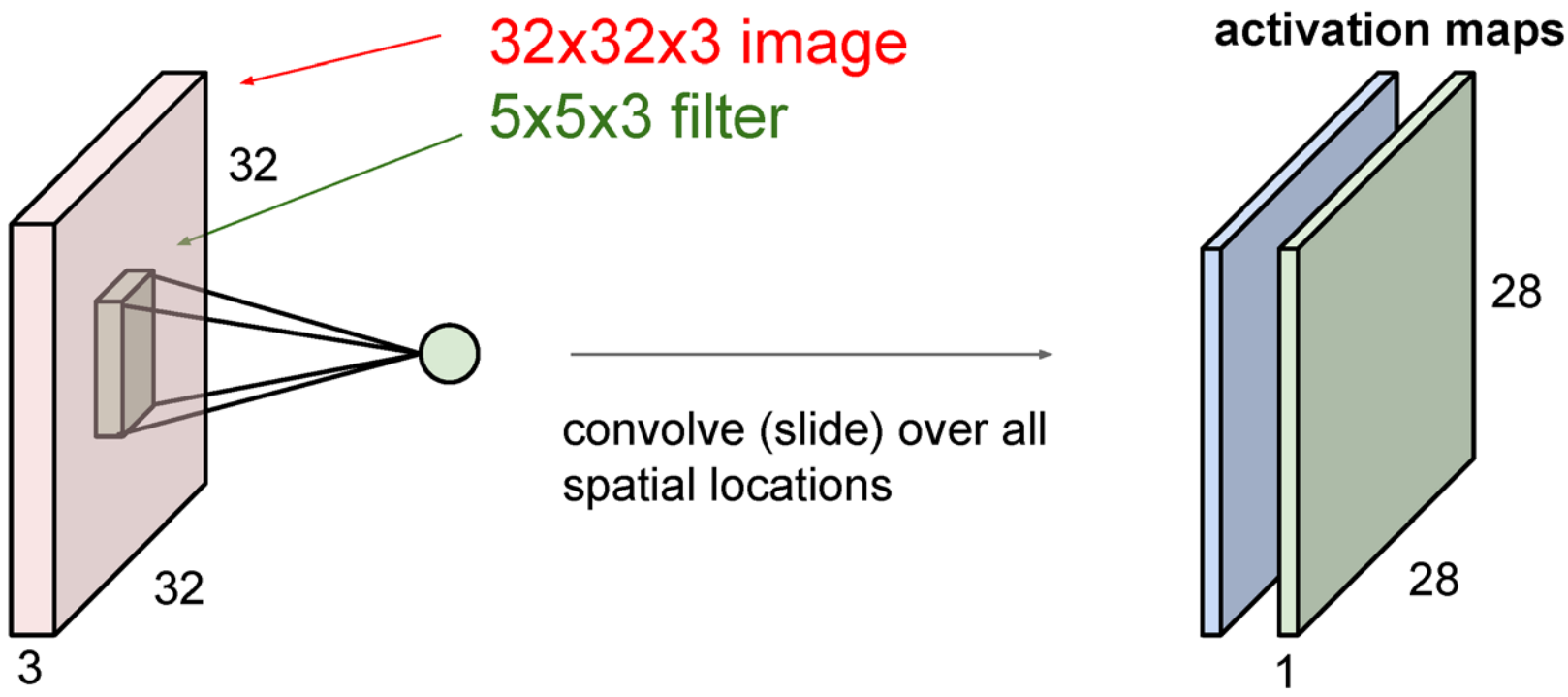
$$w^T x + b$$

Convolution Layer

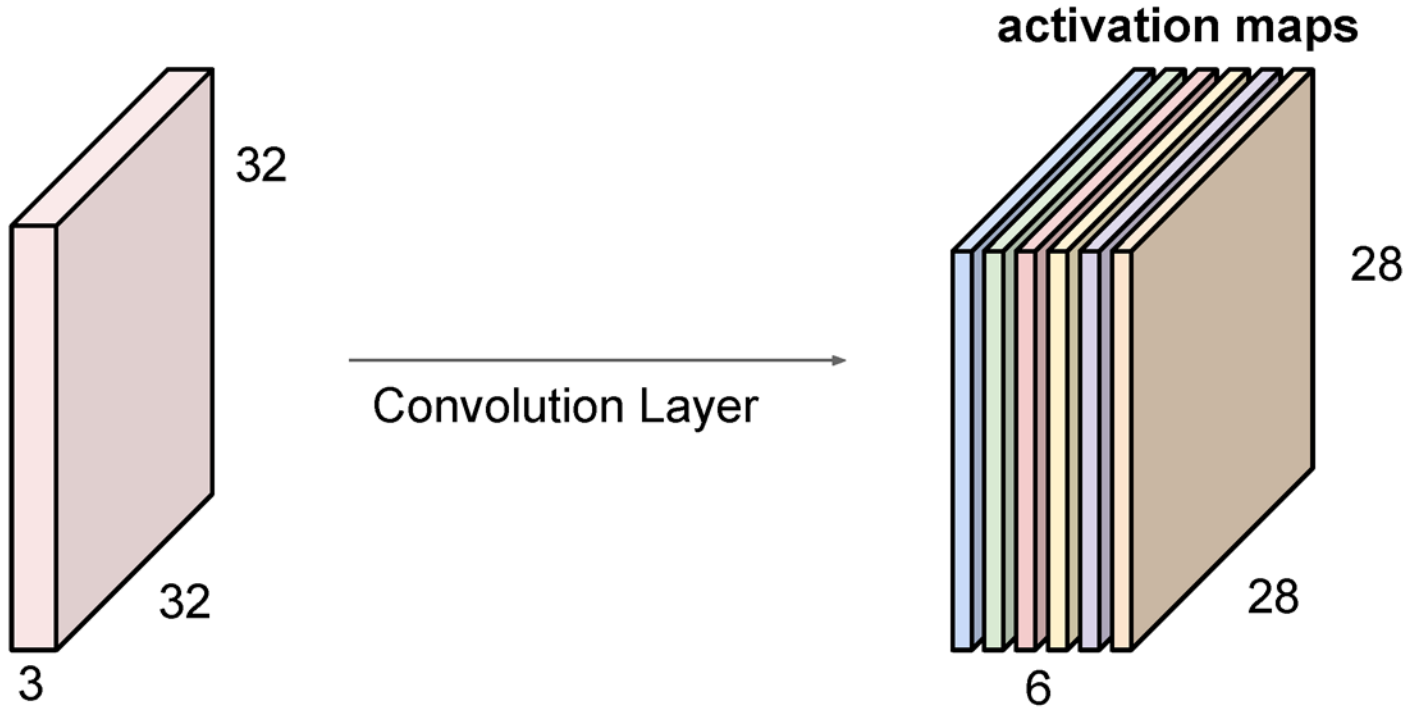


Convolution Layer

consider a second, green filter



For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

(total number of parameters: 6 x (75 + 1) = **456**)

3D Activations

We can unravel the 3D cube and show each layer separately:

(Input)

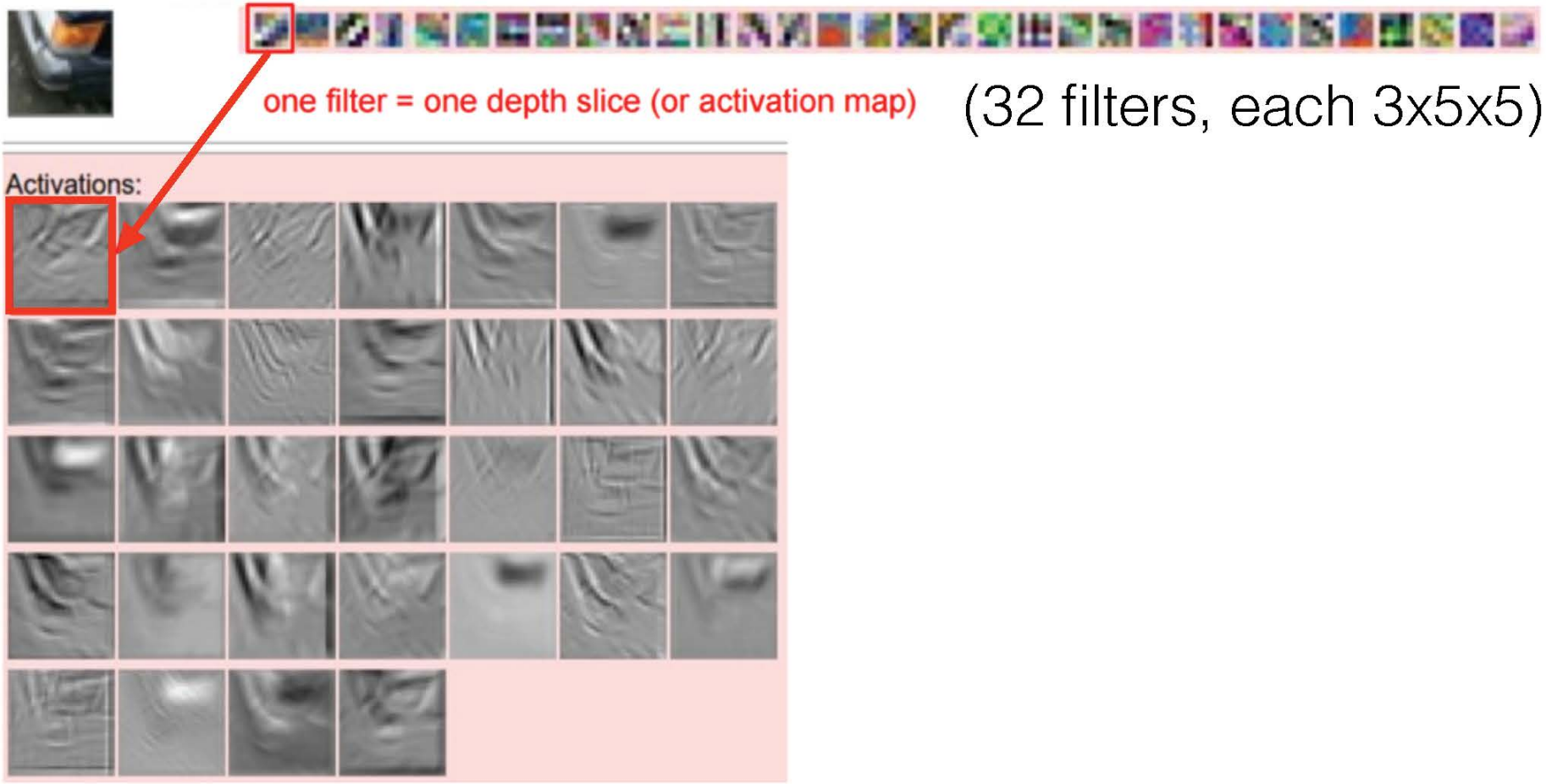


Figure: Andrej Karpathy

3D Activations

We can unravel the 3D cube and show each layer separately:

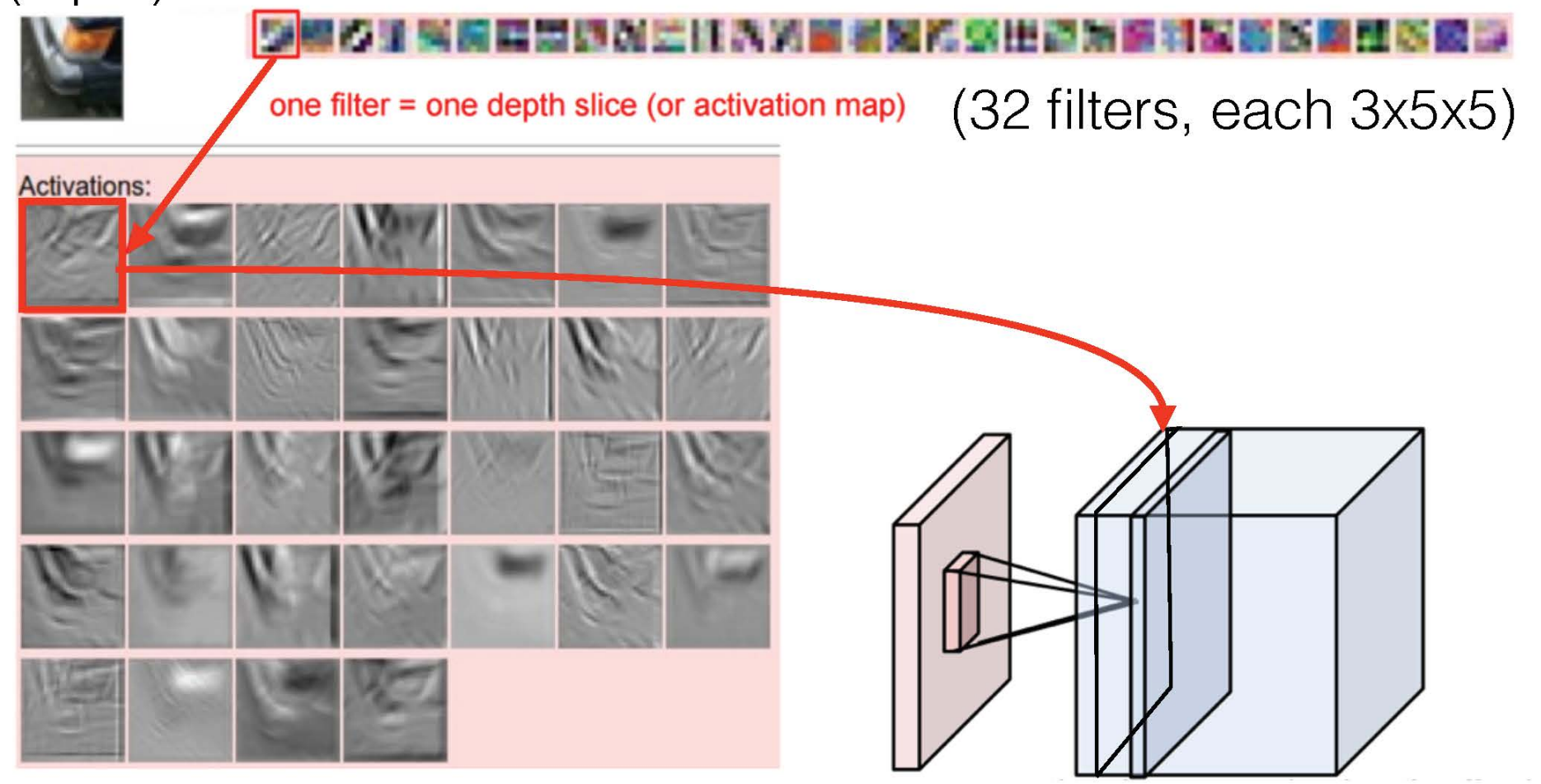


Figure: Andrej Karpathy

3D Activations

We can unravel the 3D cube and show each layer separately:

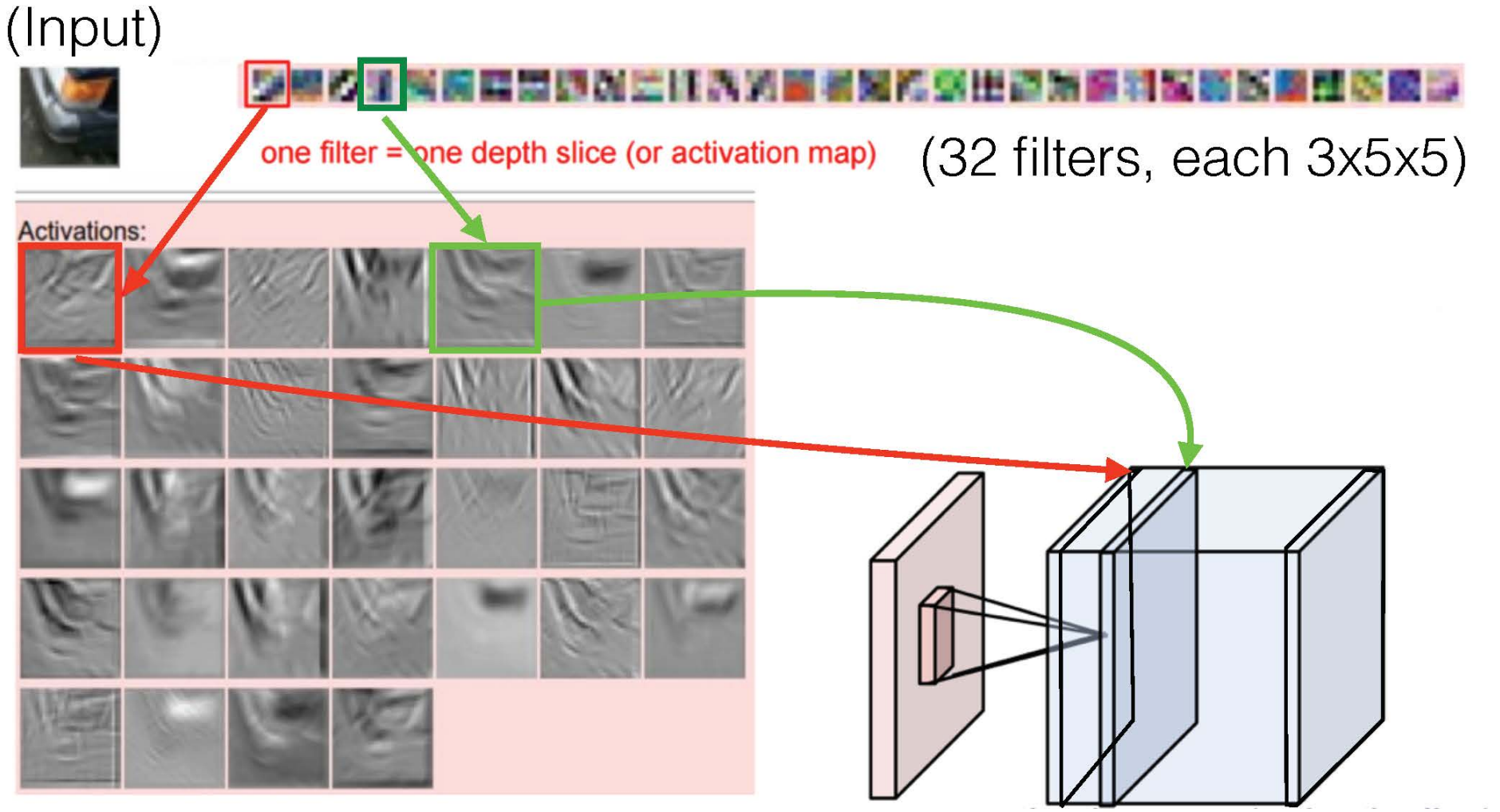


Figure: Andrej Karpathy

3D Activations

We can unravel the 3D cube and show each layer separately:

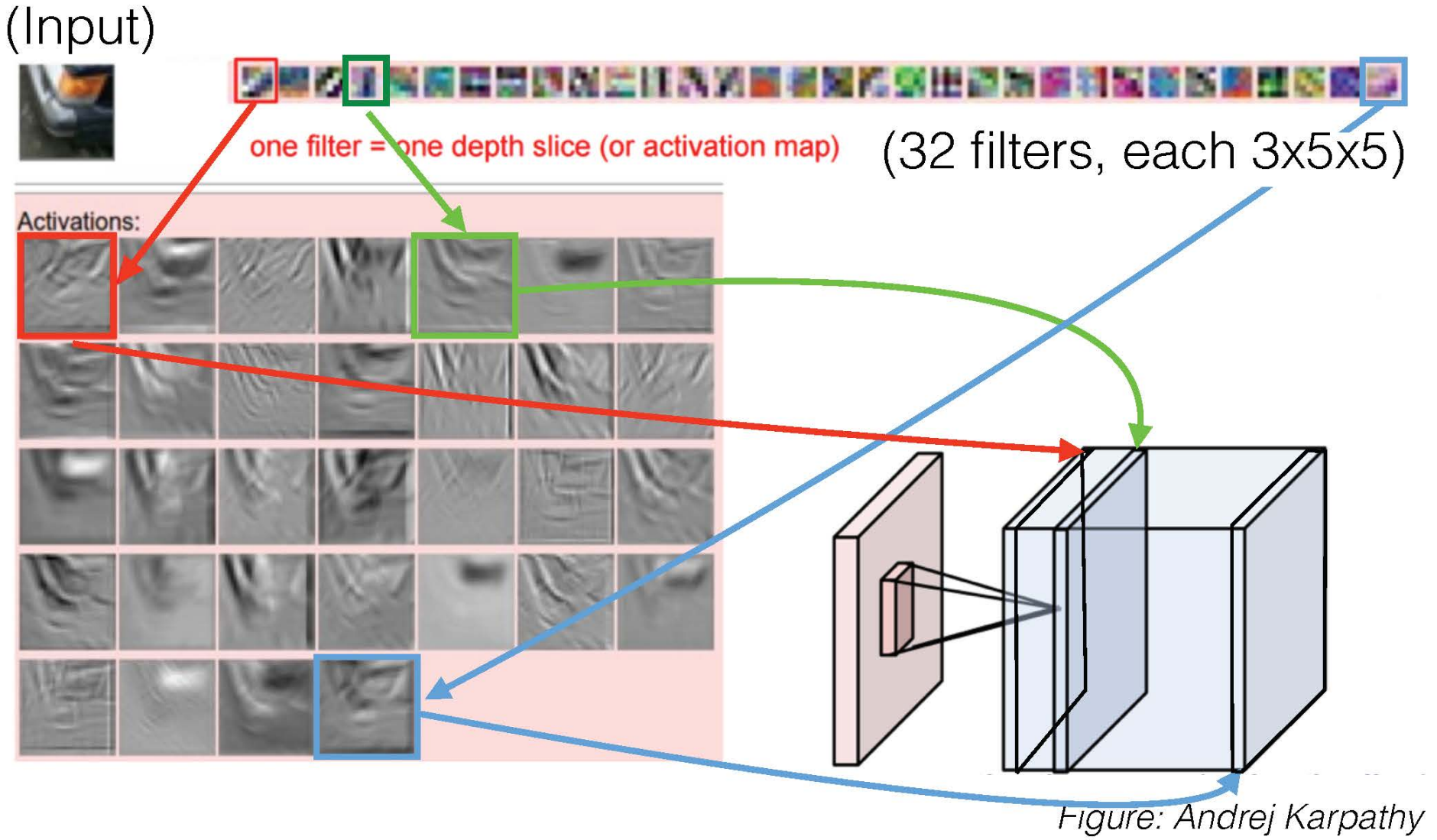
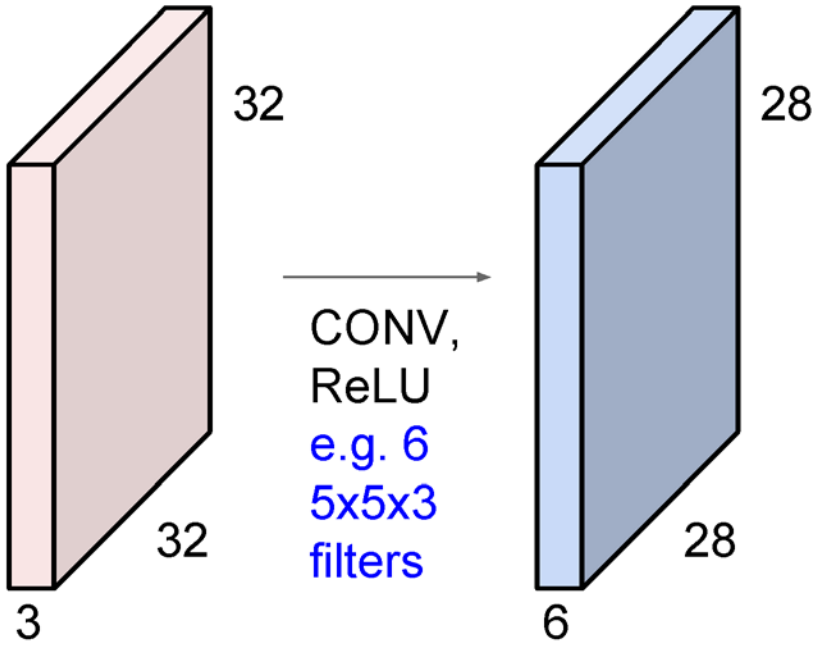
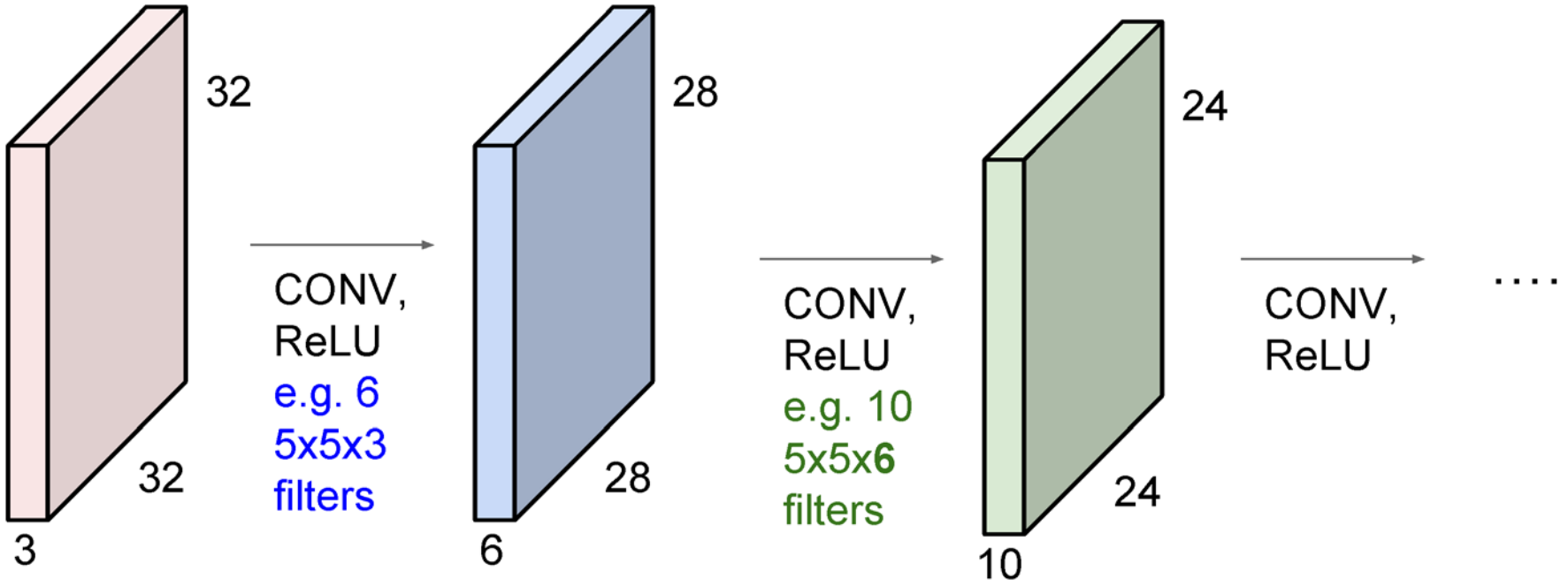


Figure: Andrej Karpathy

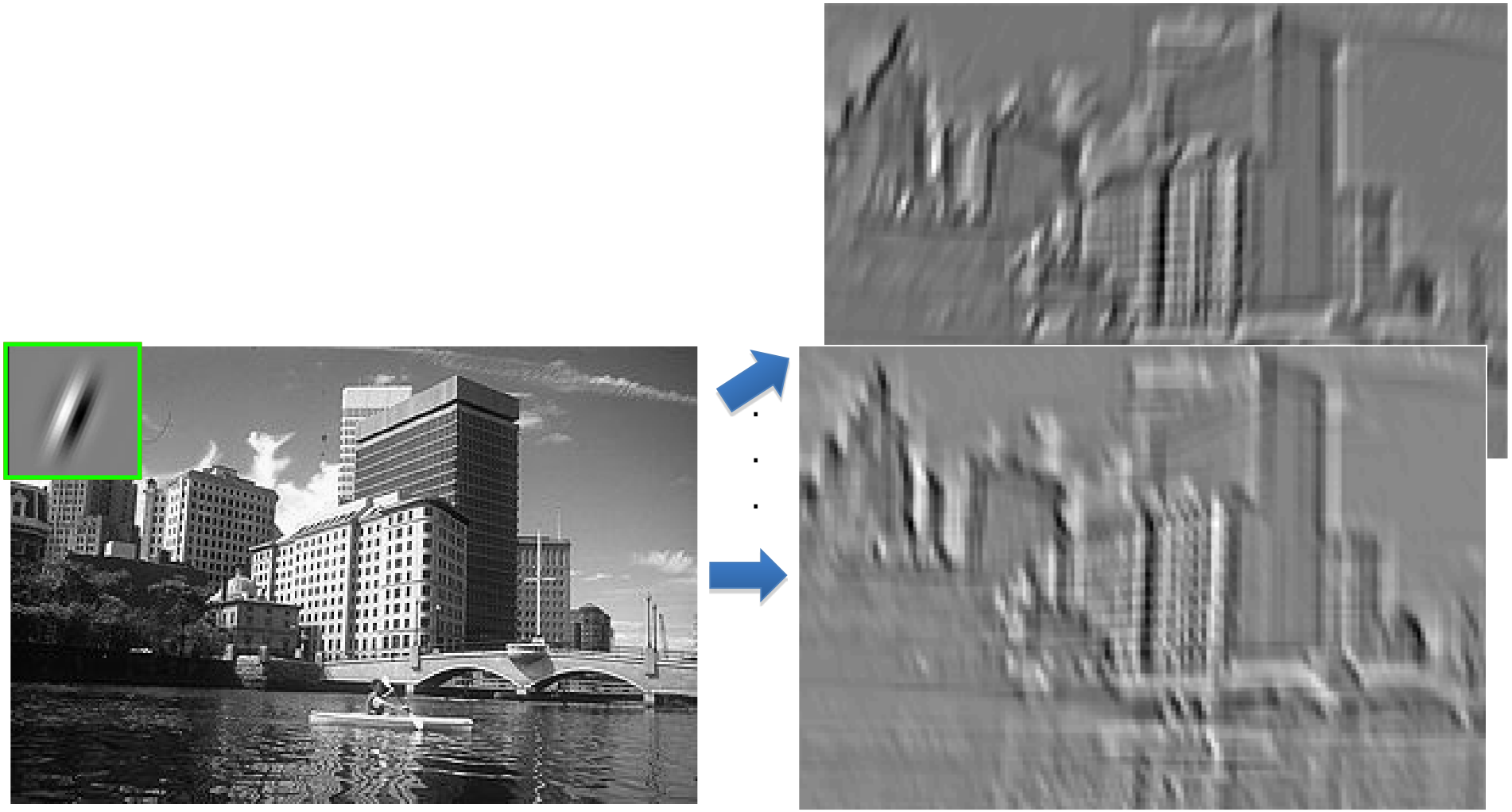
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Convolution as feature extraction

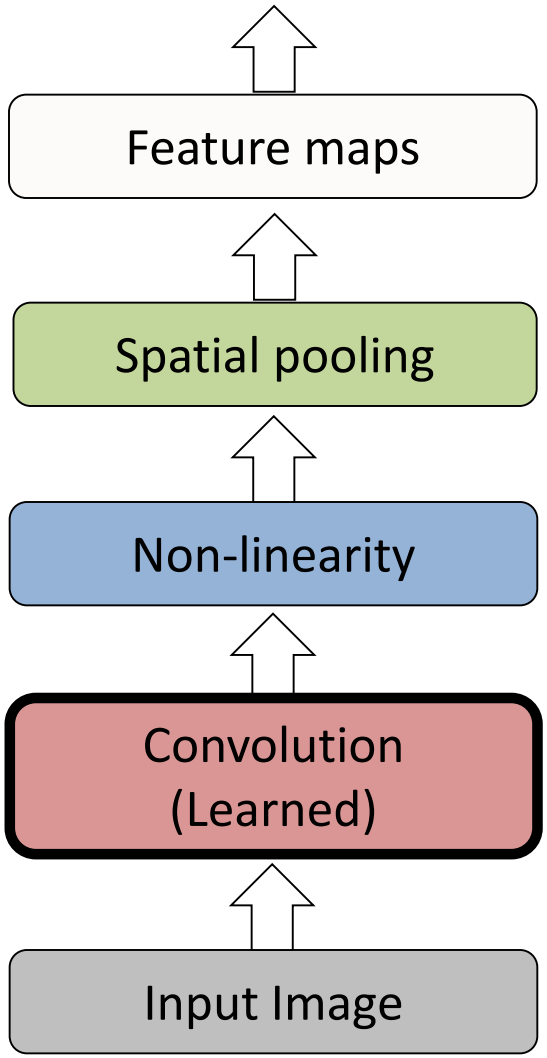


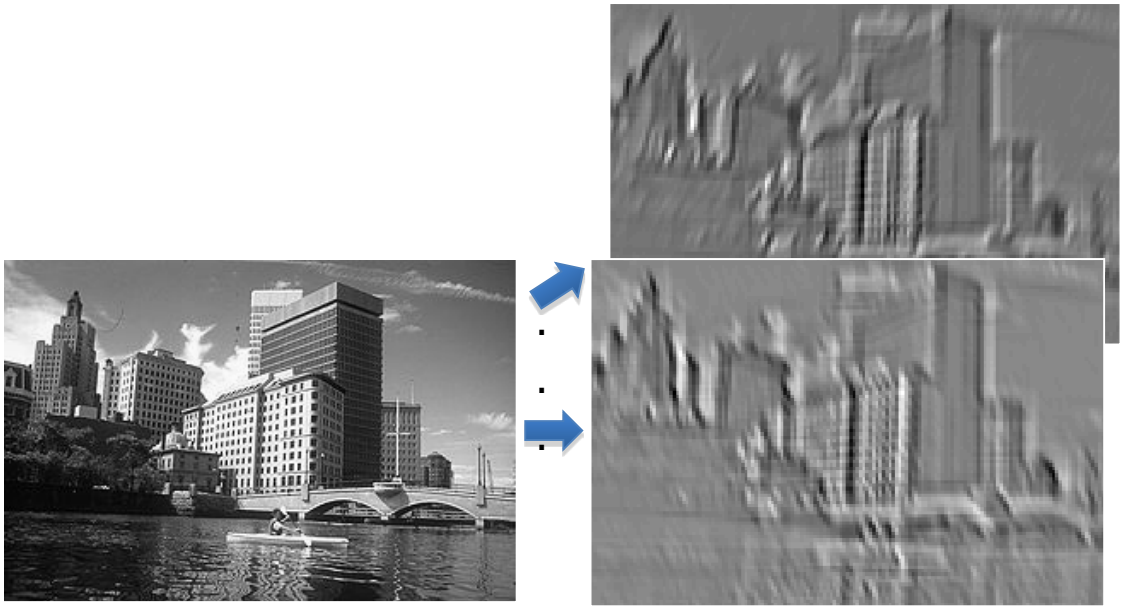
Input

Feature Map

Slide: Lazebnik

Key operations in a CNN

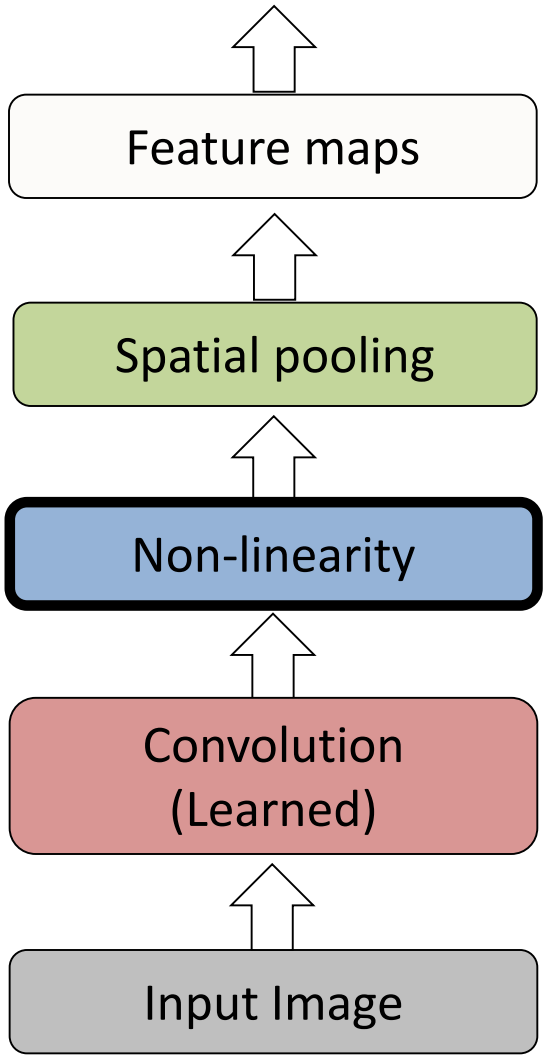


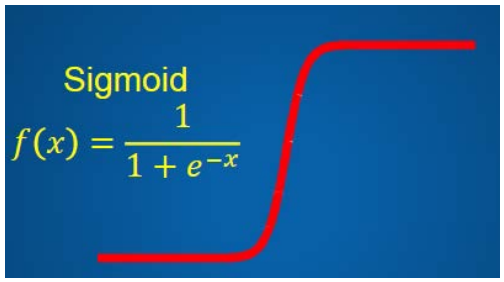


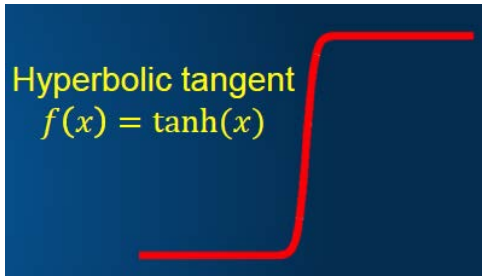
Source: R. Fergus, Y. LeCun

Slide: Lazebnik

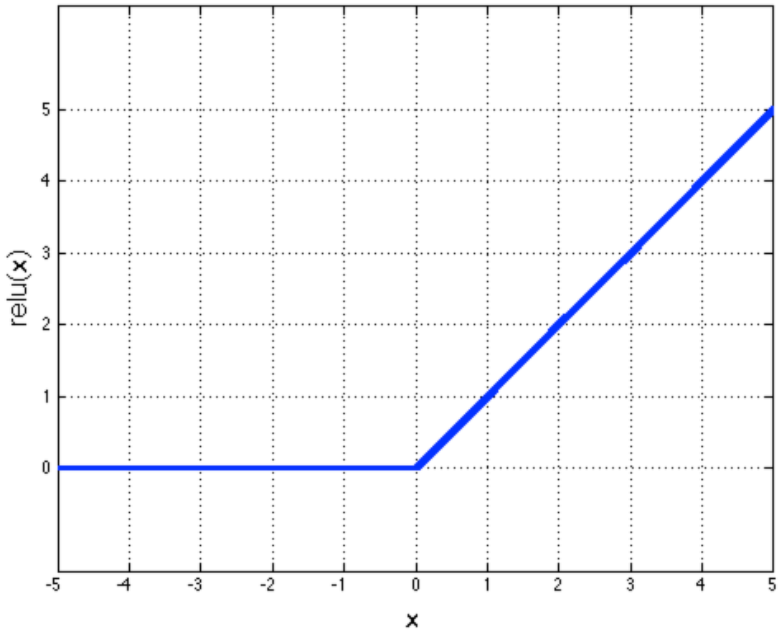
Key operations







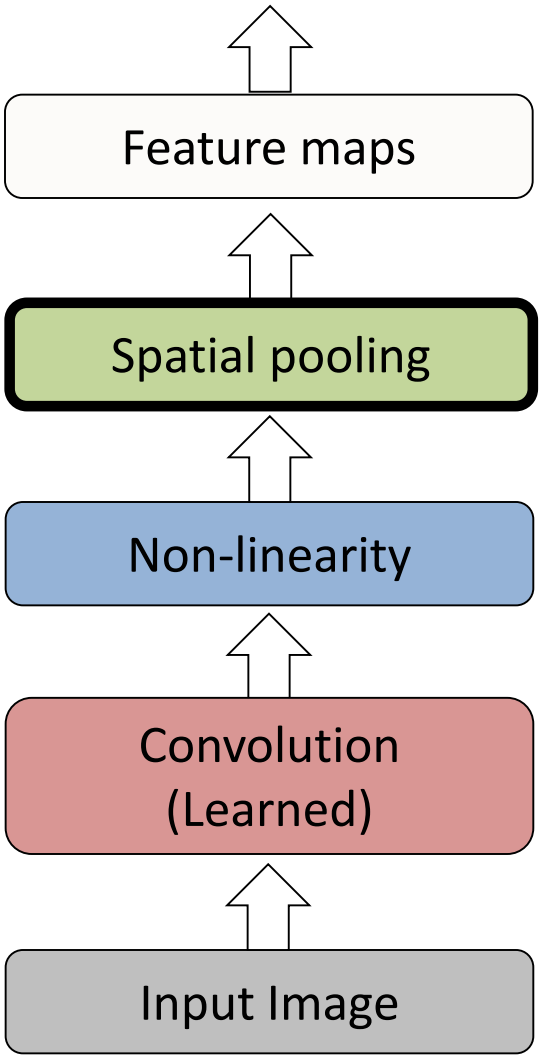
Rectified Linear Unit (ReLU)

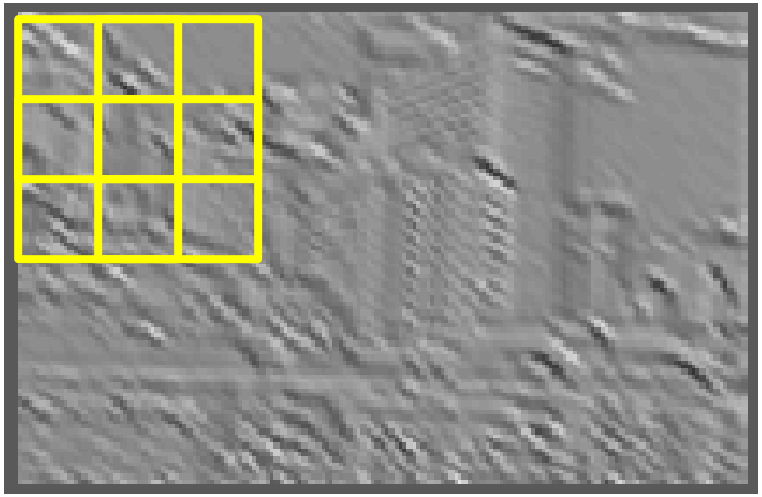


Source: R. Fergus, Y. LeCun

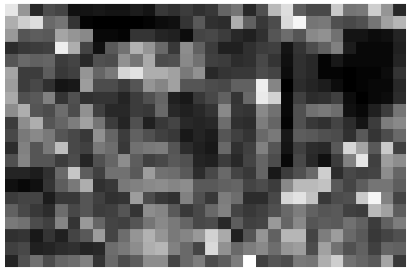
Slide: Lazebnik

Key operations





Max

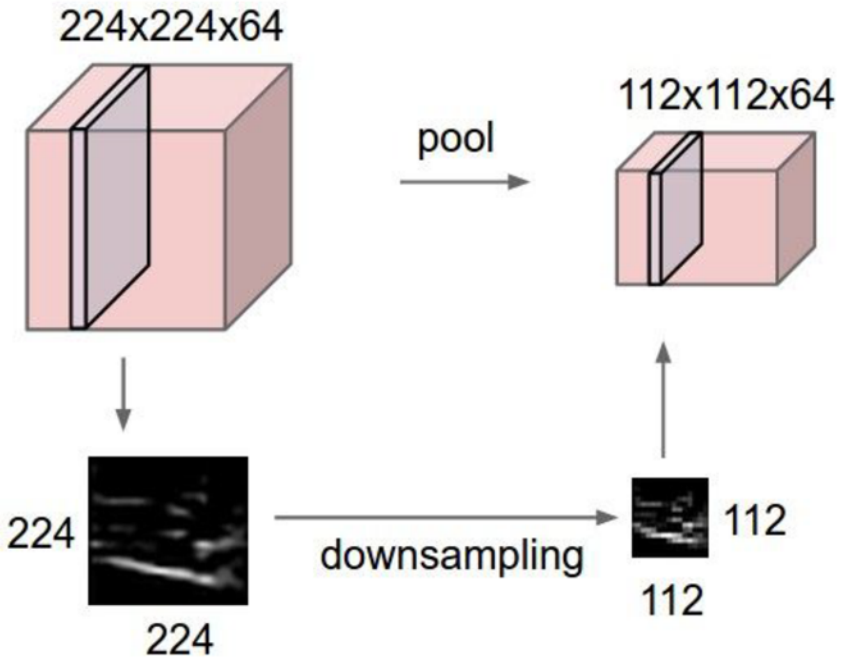


Source: R. Fergus, Y. LeCun

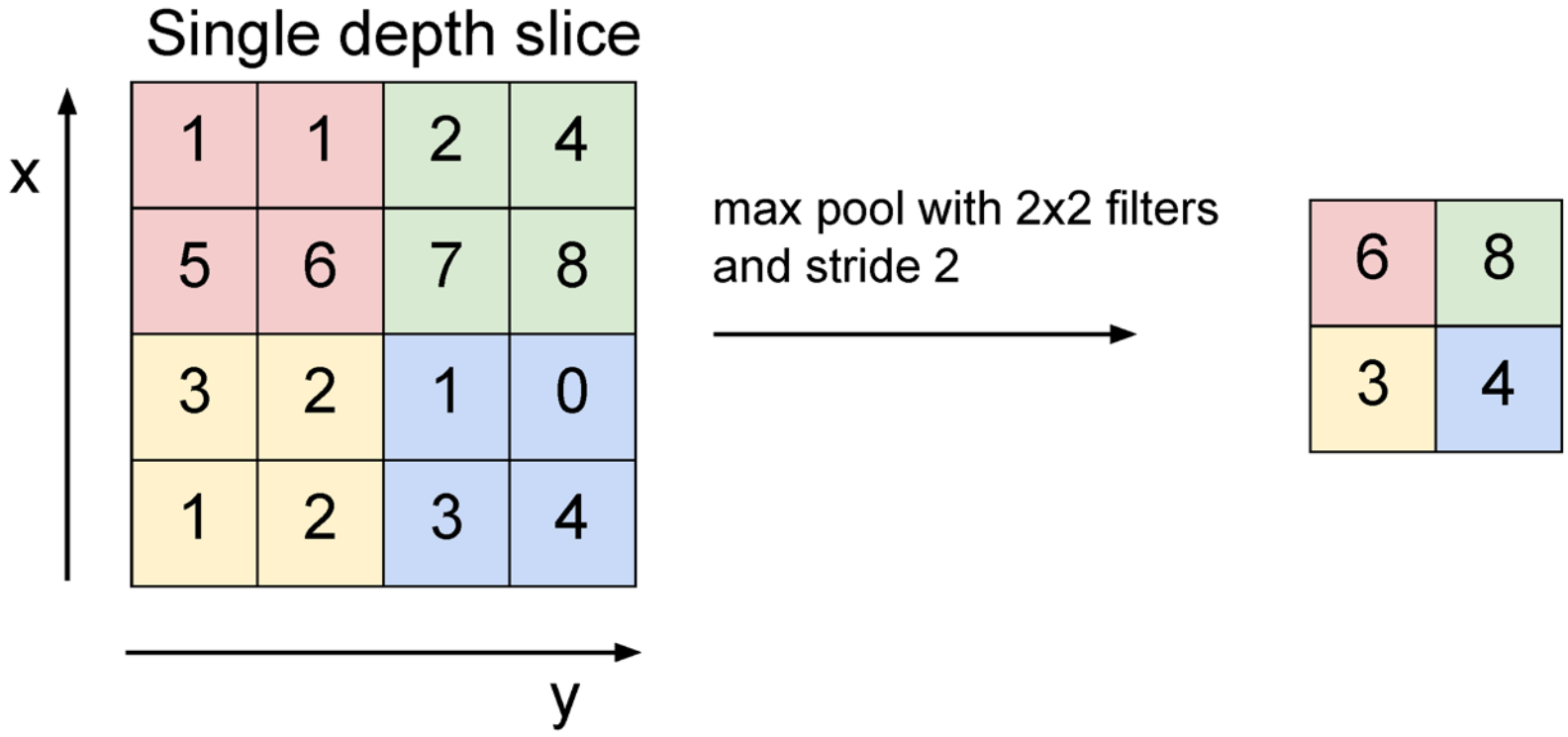
Slide: Lazebnik

Pooling layer

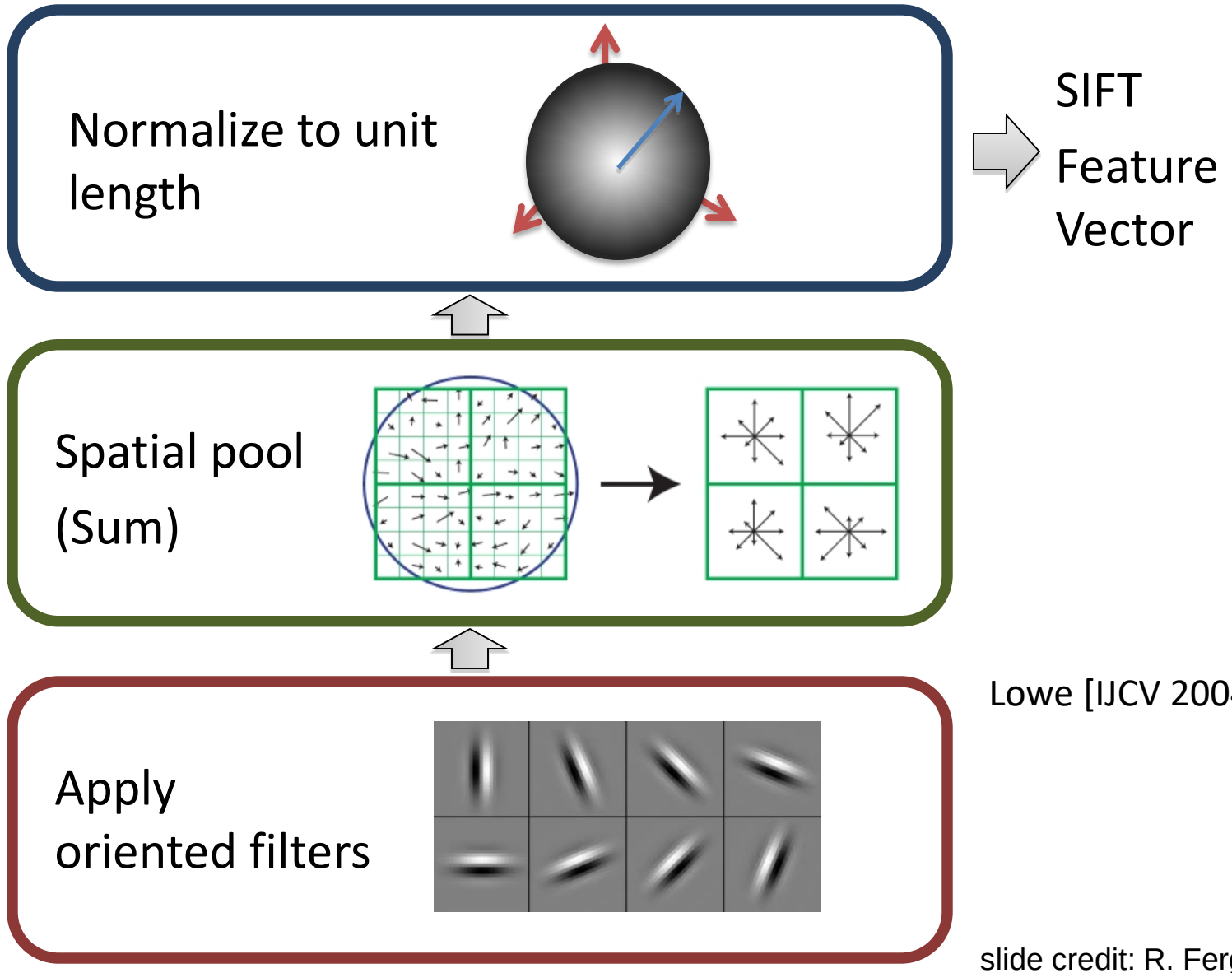
- makes the representations smaller and more manageable
- operates over each activation map independently:



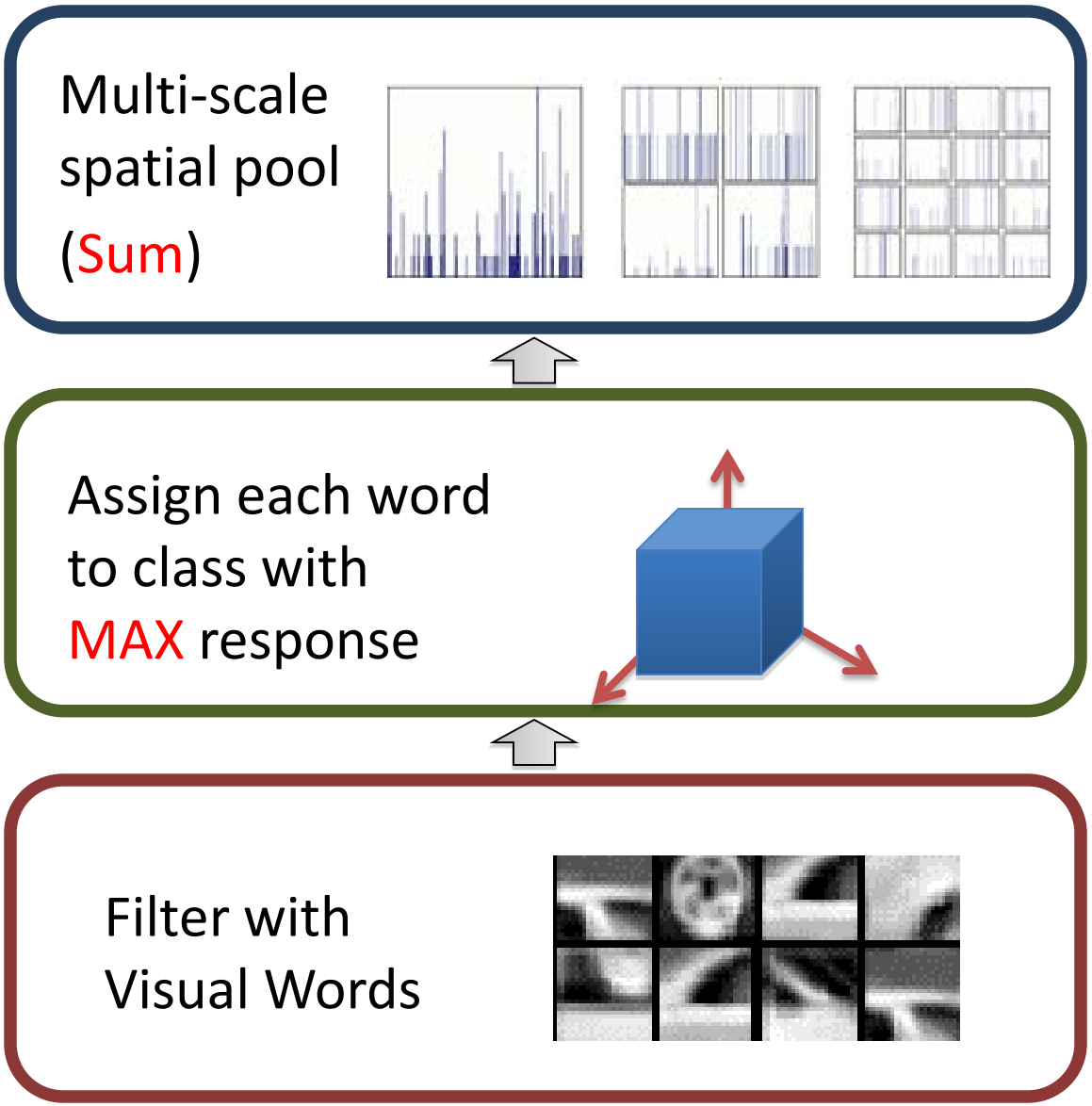
MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Comparison to Pyramids with SIFT



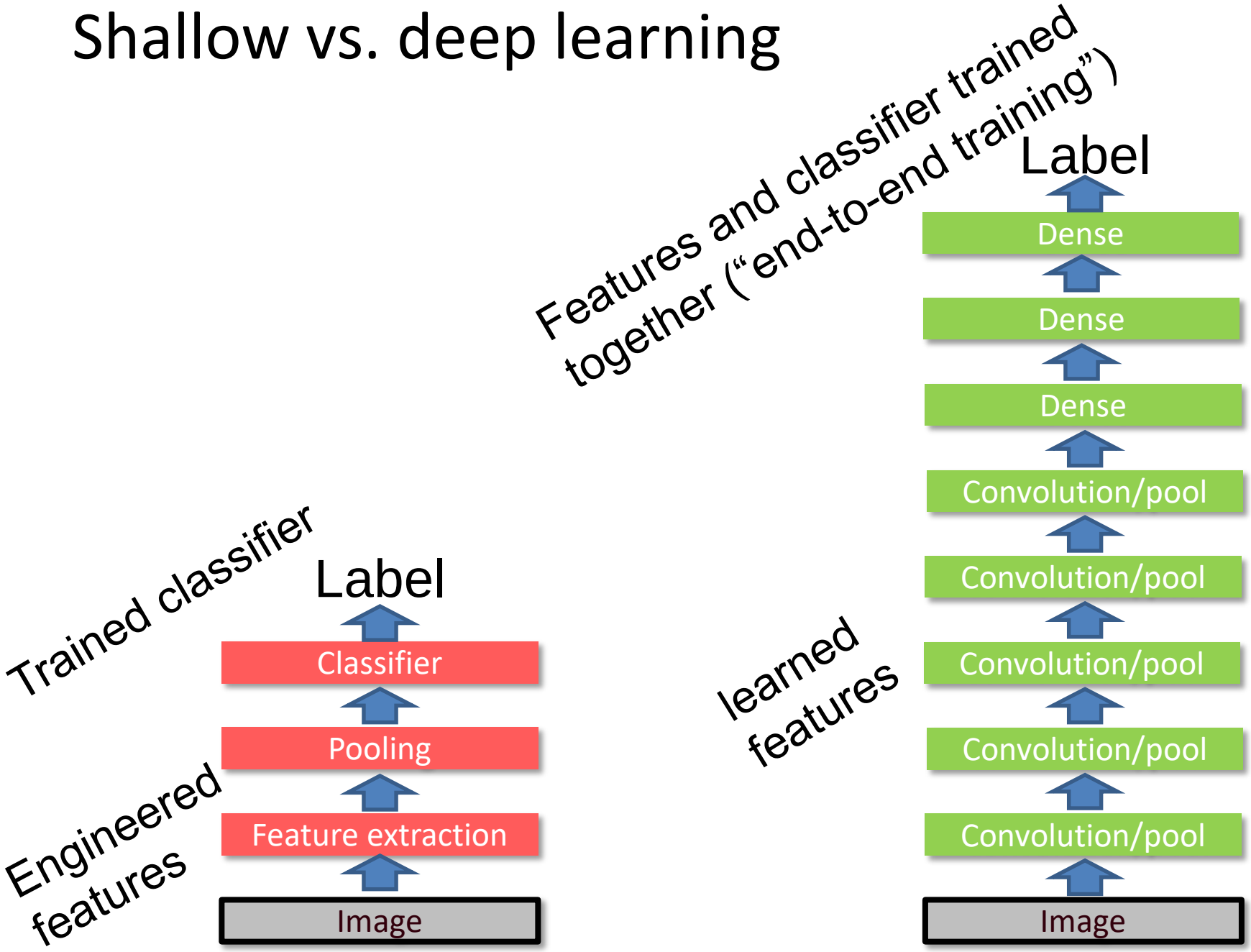
Comparison to Pyramids with SIFT



Lazebnik, Schmid, Ponce [CVPR 2006]

slide credit: R. Fergus

Shallow vs. deep learning



Summary

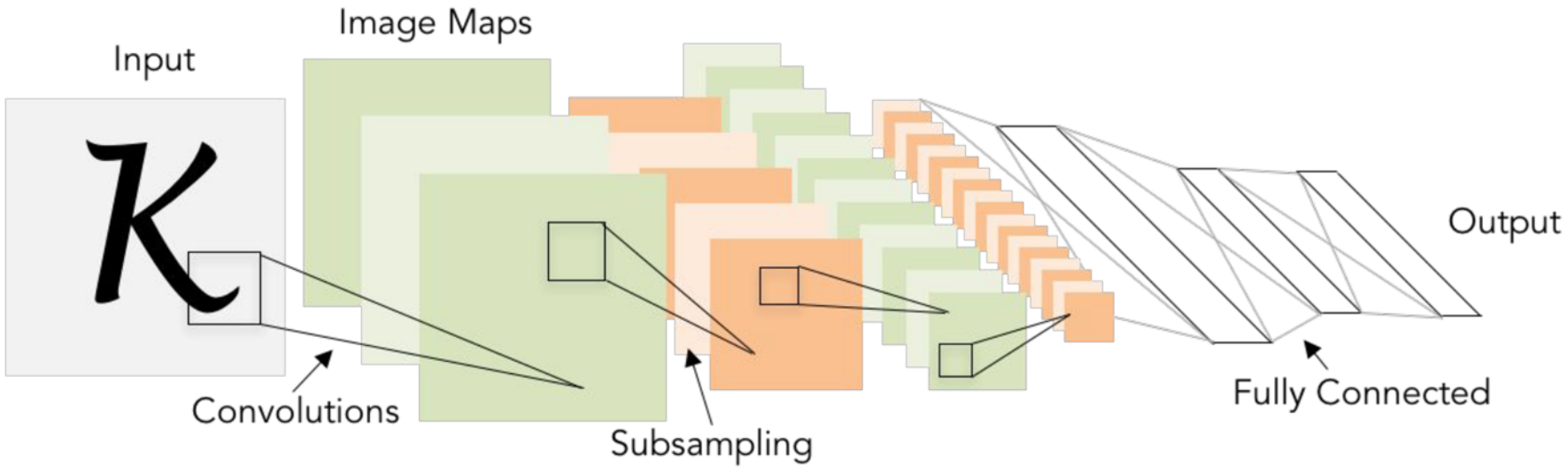
- ConvNets stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like
  [(CONV-RELU)*N-POOL?]*M-(FC-RELU)*K,SOFTMAX
  where N is usually up to ~5, M is large, 0 <= K <= 2.
  - but recent advances such as ResNet/GoogLeNet challenge this paradigm

CNN Architectures

- AlexNet
- VGG
- GoogleNet
- ResNet
- …

LeNet

[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

Fast forward to the arrival of big visual data…

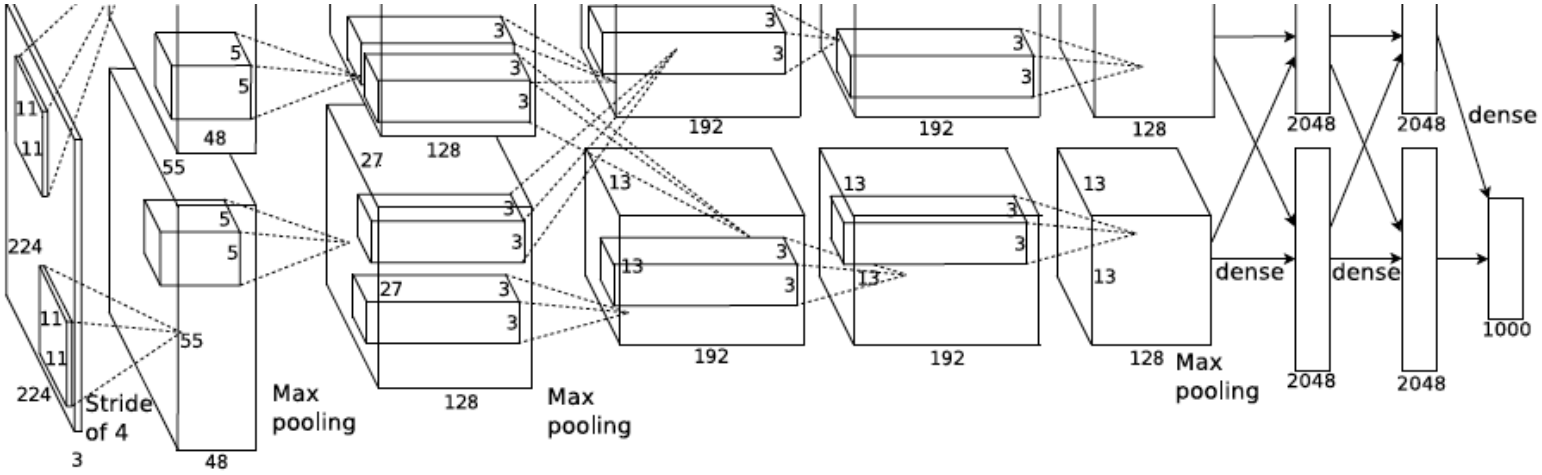
IMAGENET

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC): 1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/

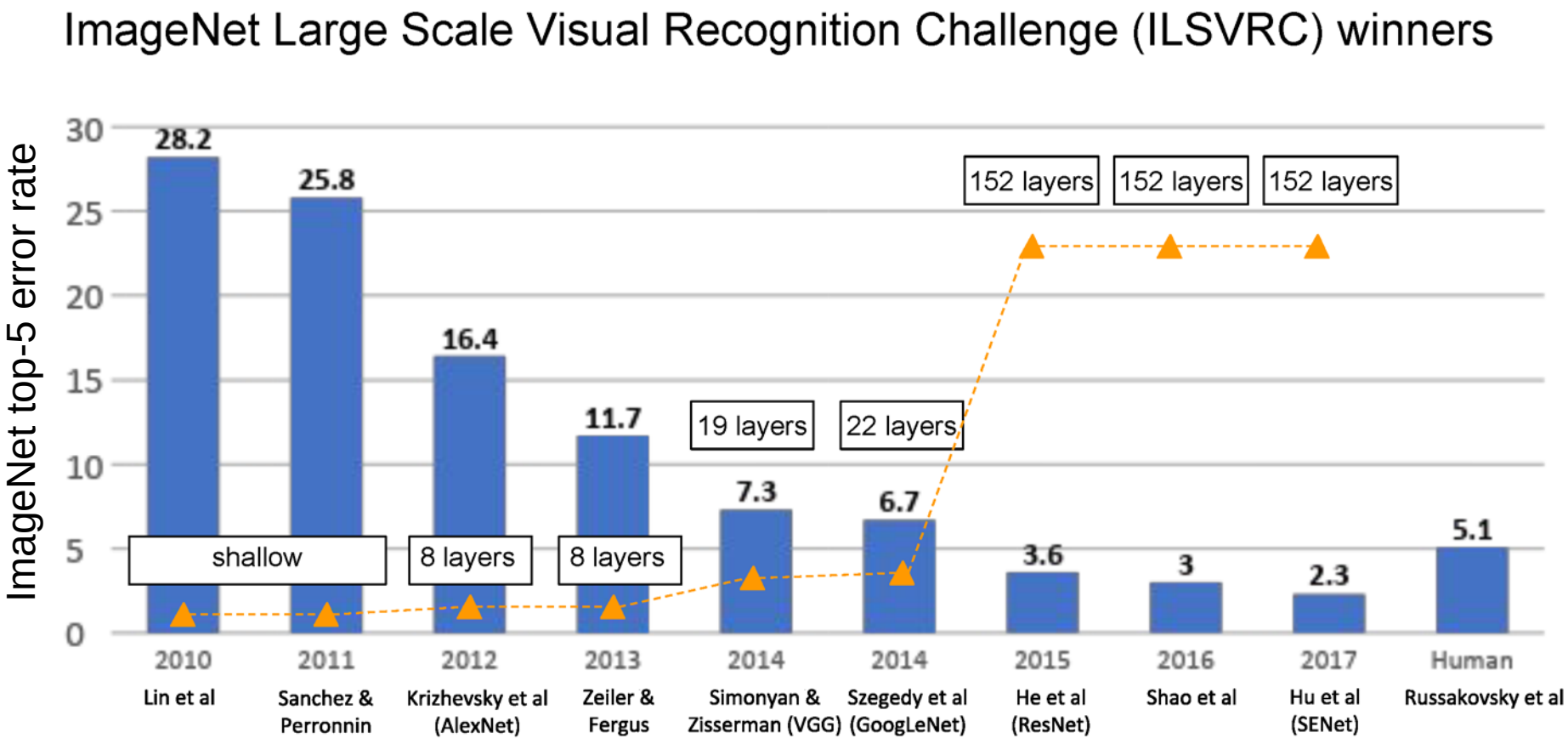
Slide: Lazebnik

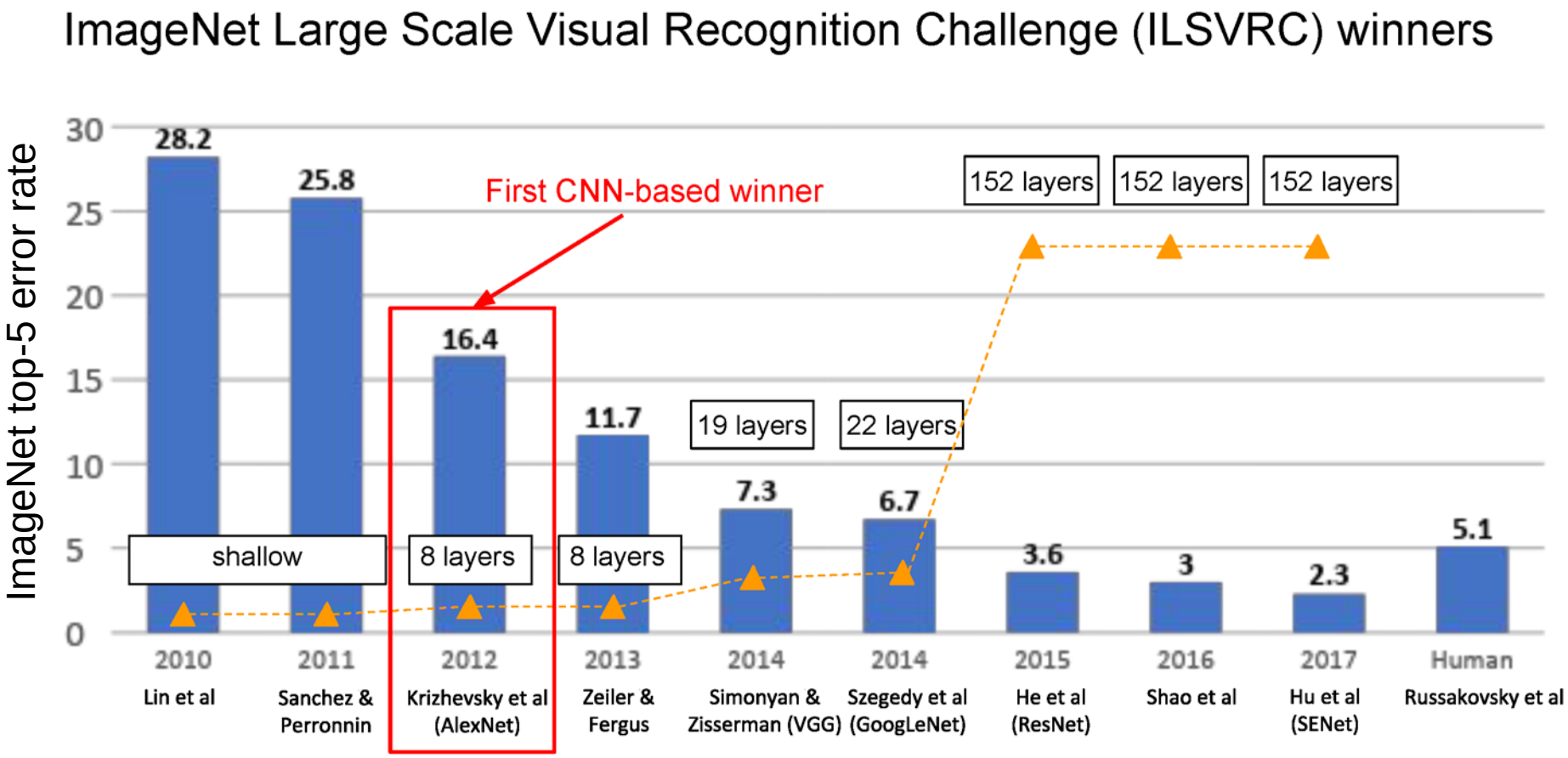
AlexNet: ILSVRC 2012 winner

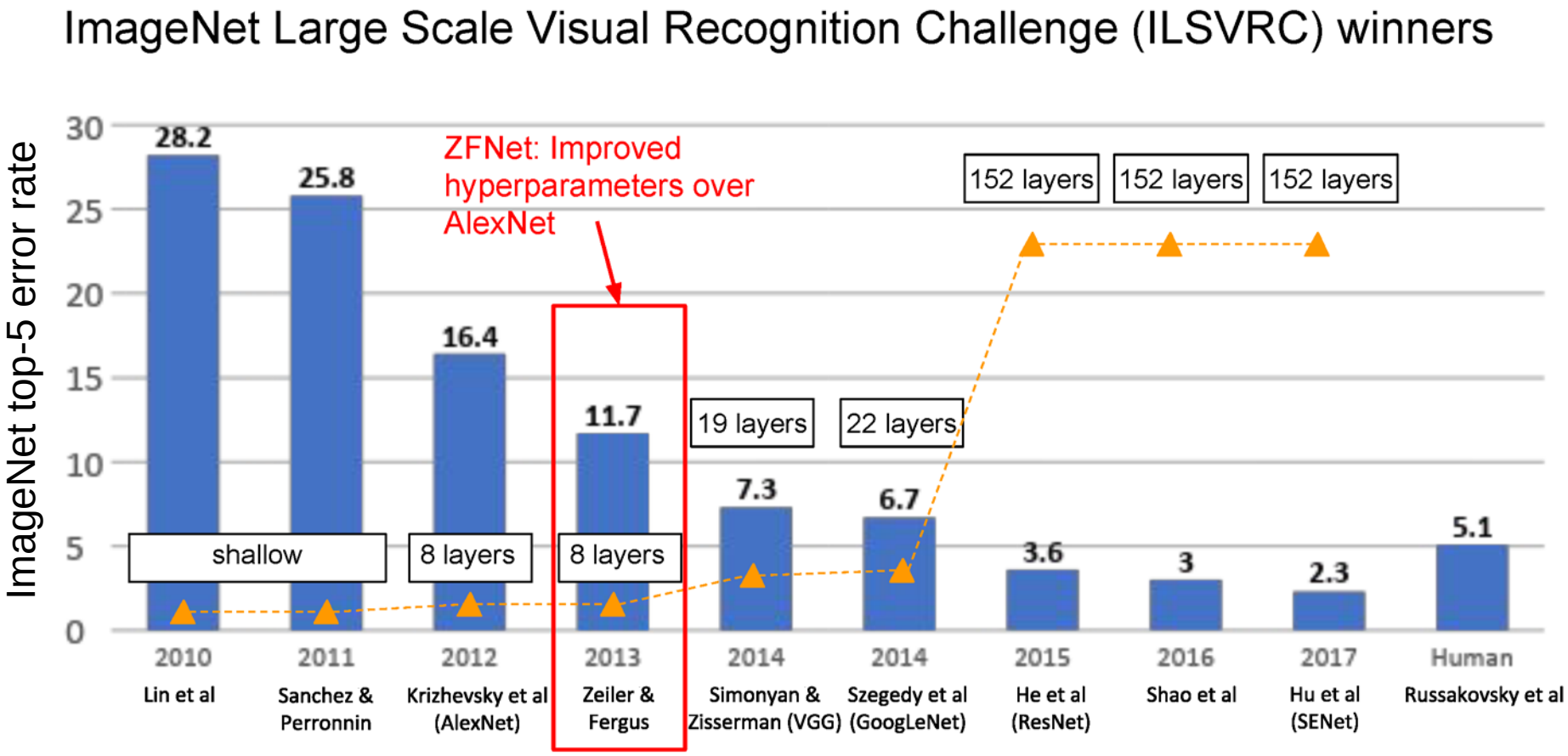


- Similar framework to LeNet but:
  - Max pooling, **ReLU nonlinearity**
  - **More data** and **bigger model** (7 hidden layers, 650K units, 60M params)
  - GPU implementation (**50x speedup** over CPU)
    - Trained on two GPUs for a week
  - Dropout regularization

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

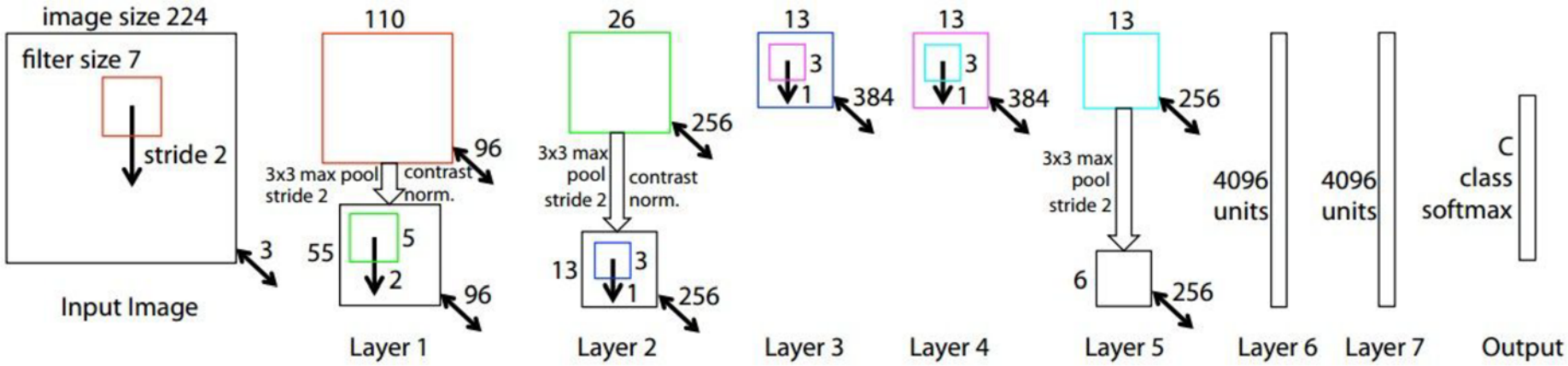

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
ImageNet top-5 error rate
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
3
2.3
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
2016
Shao et al
2017
Hu et al (SENet)
Human
Russakovsky et al


ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
ImageNet top-5 error rate
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
3
2.3
5.1
First CNN-based winner
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
2016
Shao et al
2017
Hu et al (SENet)
Human
Russakovsky et al


ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
ImageNet top-5 error rate
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
3
2.3
5.1
ZFNet: Improved hyperparameters over AlexNet
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers
2010 Lin et al
2011 Sanchez & Perronnin
2012 Krizhevsky et al (AlexNet)
2013 Zeiler & Fergus
2014 Simonyan & Zisserman (VGG)
2014 Szegedy et al (GoogLeNet)
2015 He et al (ResNet)
2016 Shao et al
2017 Hu et al (SENet)
Human Russakovsky et al

ZFNet

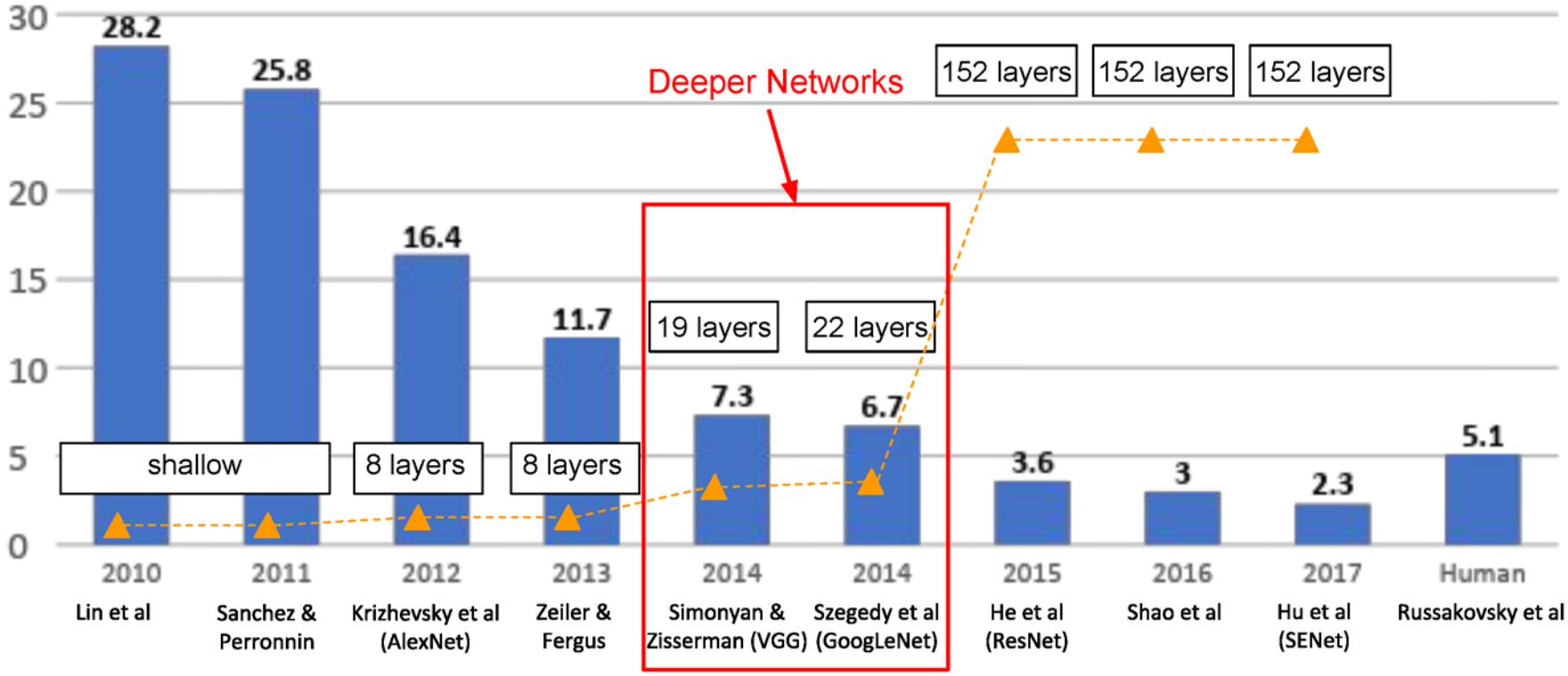
[Zeiler and Fergus, 2013]



AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 16.4% -> 11.7%

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Case Study: VGGNet

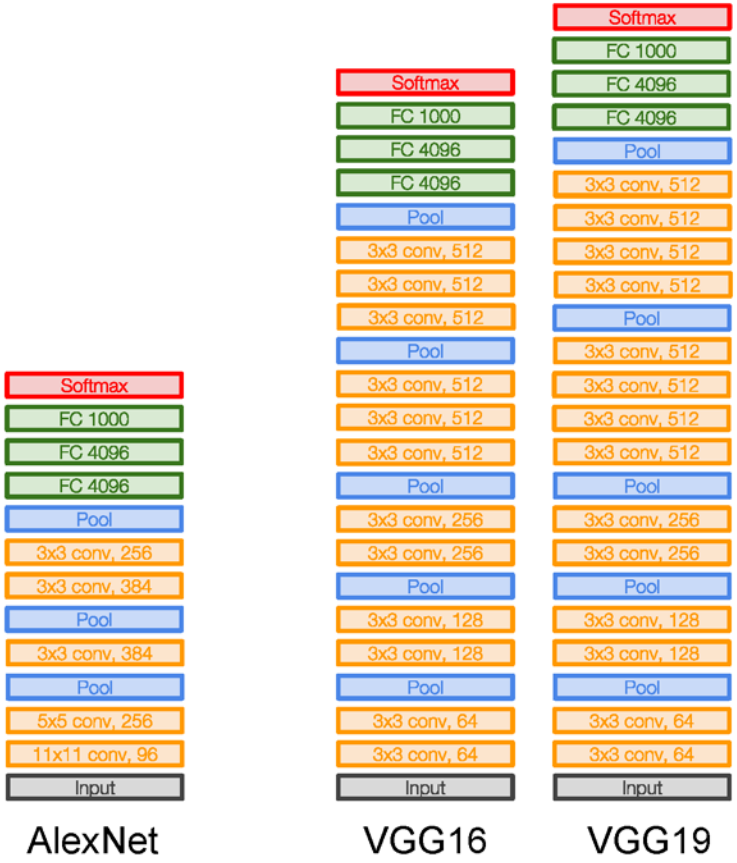
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

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

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

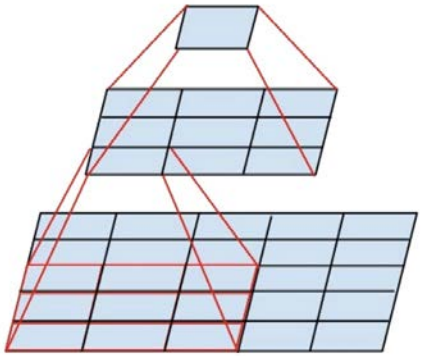
11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



VGGNet

- Sequence of deeper networks trained progressively
- Q: What is the receptive field of 3 layers of 3x3 convolutions?

  A: 7x7



Q: How many parameters are involved?

A: (9+9+9)*C^2

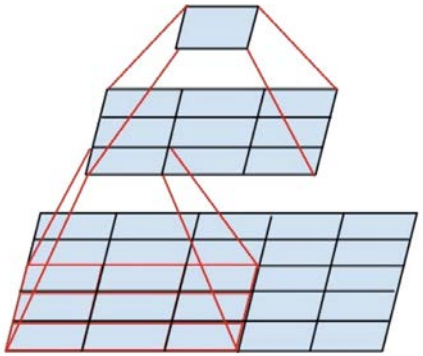
Q: How many parameters in a single 7x7 conv layer?

A: (49)*C^2

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

VGGNet

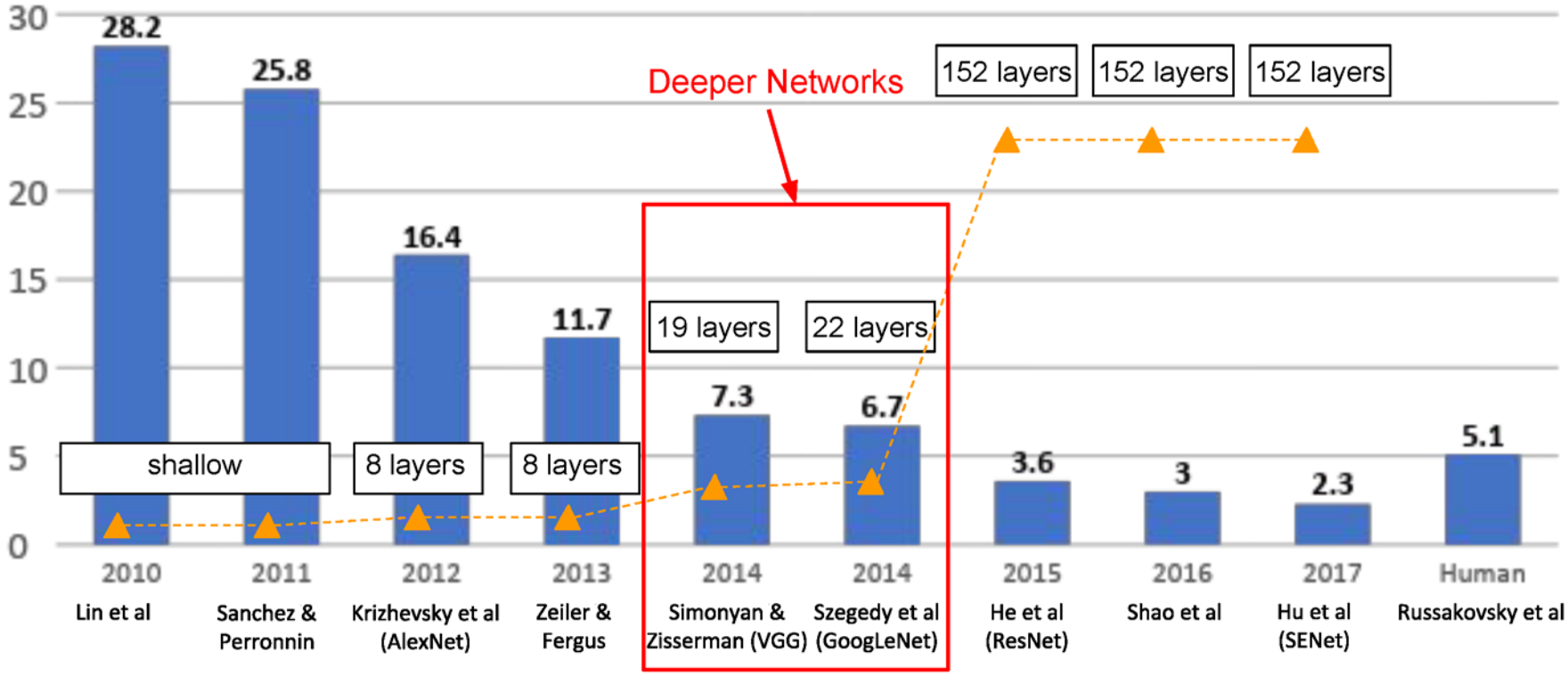
- Sequence of deeper networks trained progressively
- Large receptive fields replaced by successive layers of 3x3 convolutions (with ReLU in between)



  - One 7x7 conv layer with C feature maps needs $49C^2$ weights, three 3x3 conv layers need only $27C^2$ weights

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

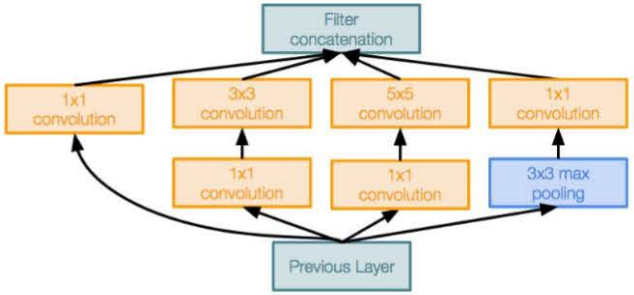
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



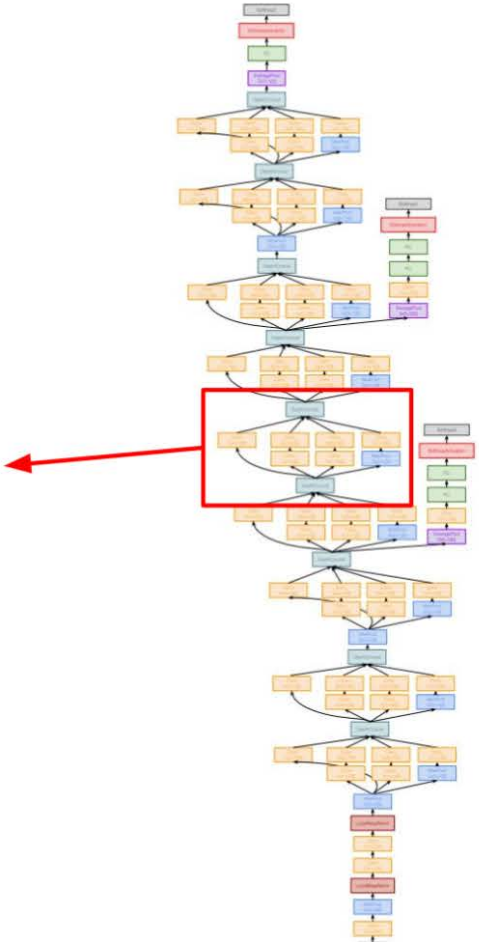
Case Study: GoogLeNet

[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

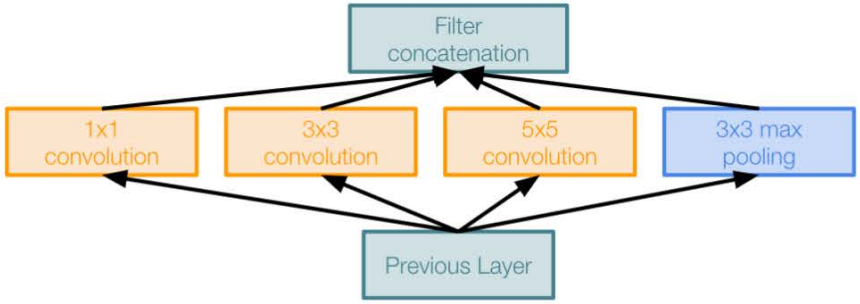


Inception module

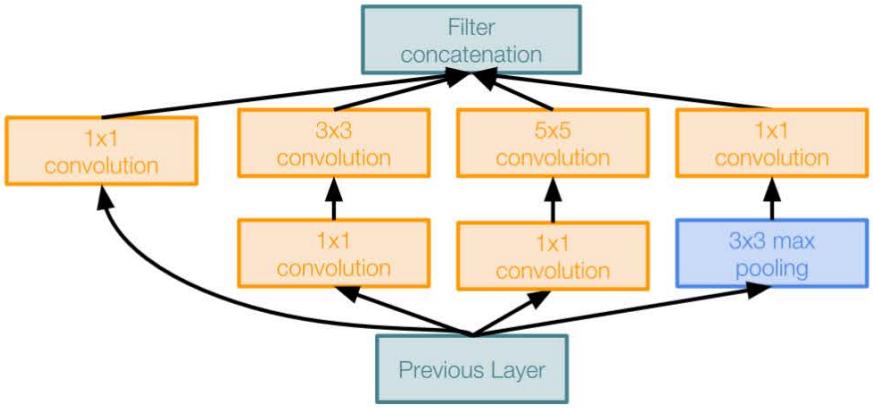


Case Study: GoogLeNet

[Szegedy et al., 2014]

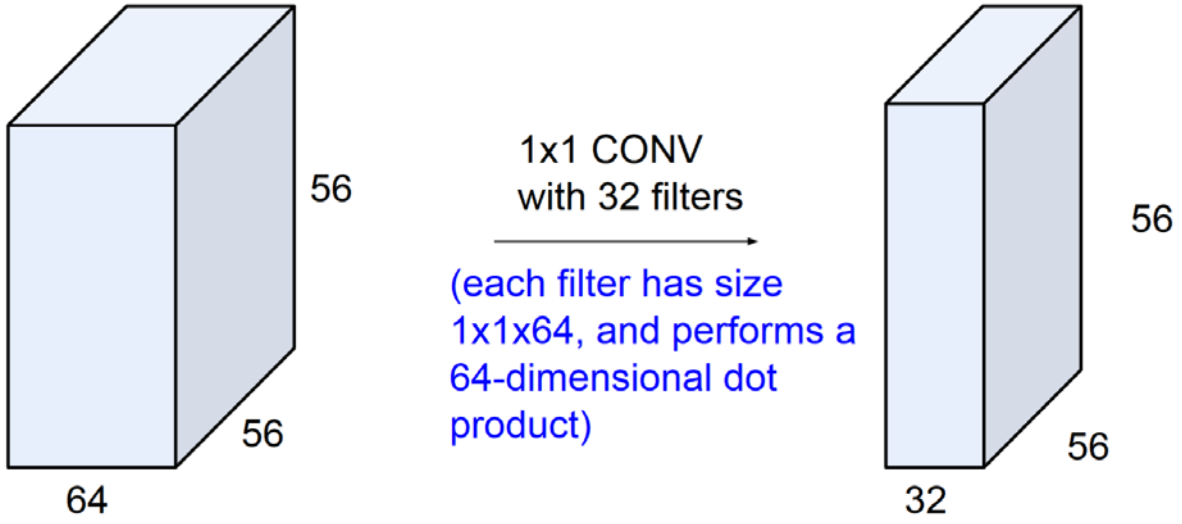


Naive Inception module



Inception module with dimension reduction

1x1 convolutions – dimensionality reduction

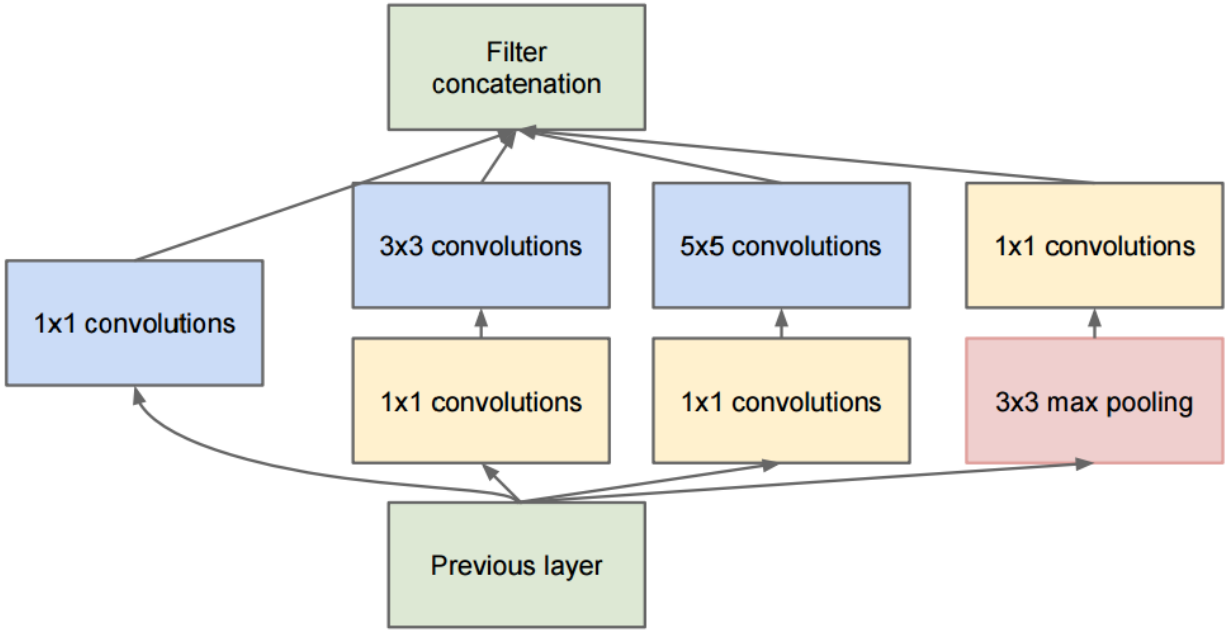


preserves spatial dimensions, reduces depth!

Projects depth to lower dimension (combination of feature maps)

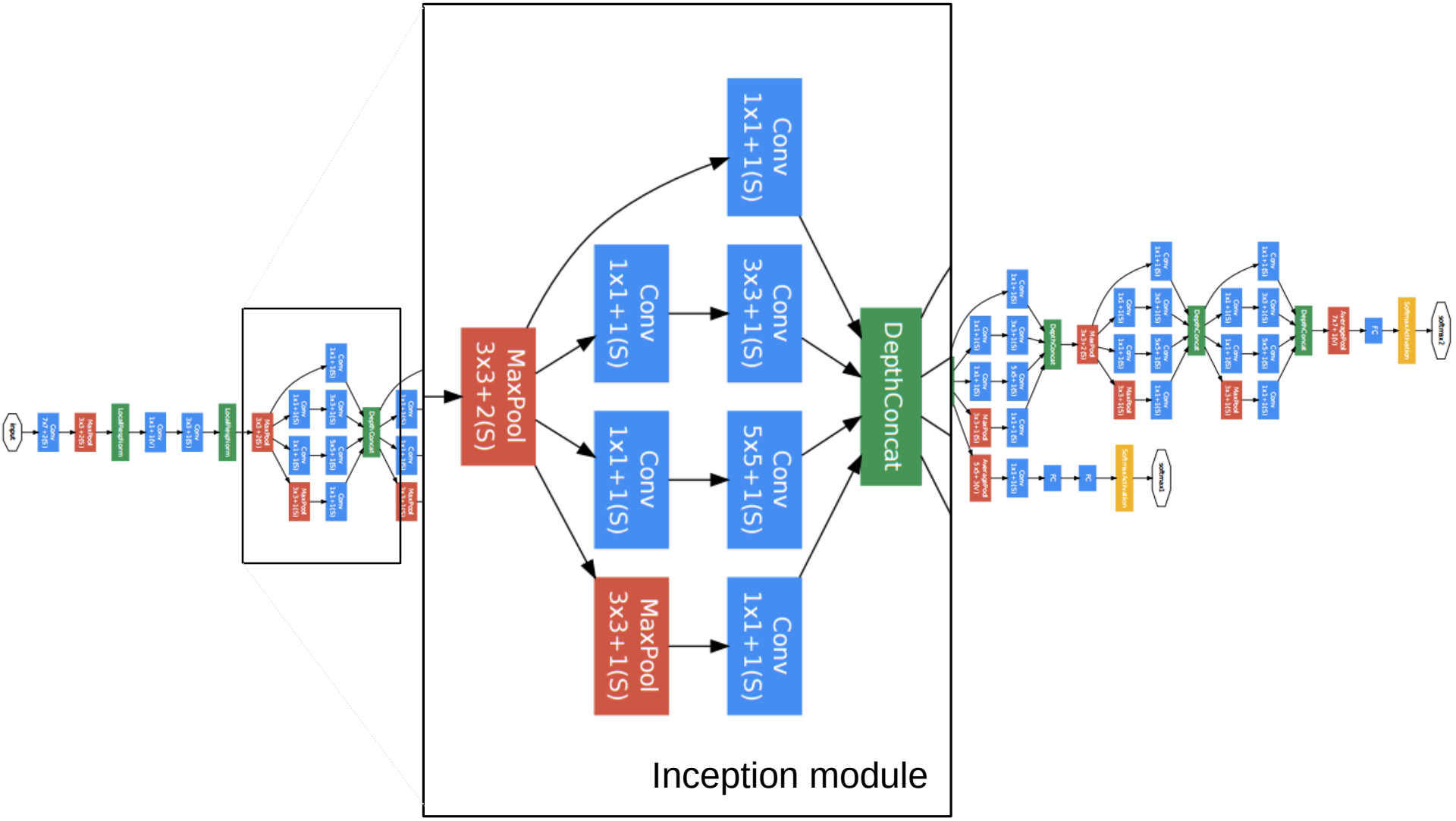
GoogLeNet: Inception module

- Parallel paths with different receptive field sizes and operations to capture sparse patterns of correlations
- 1x1 convolutions for dimensionality reduction before expensive convolutions



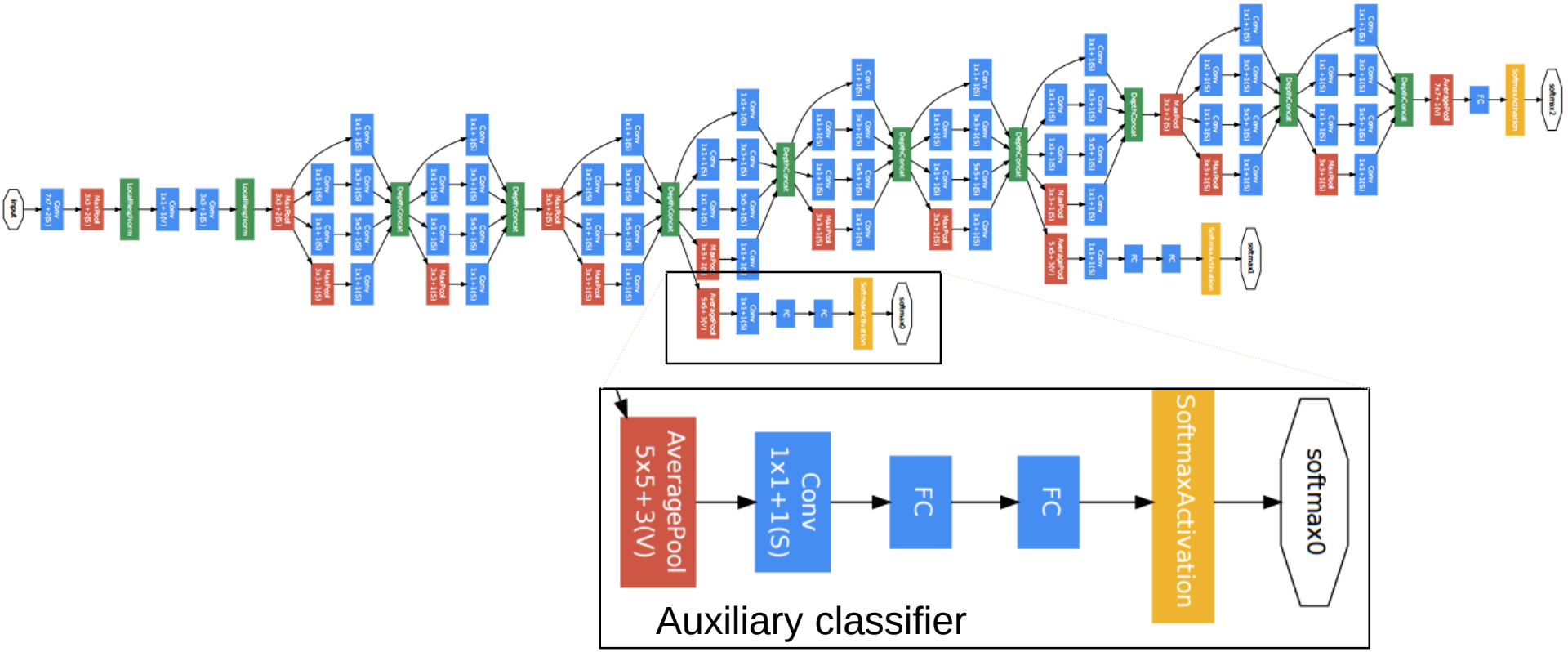
C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet

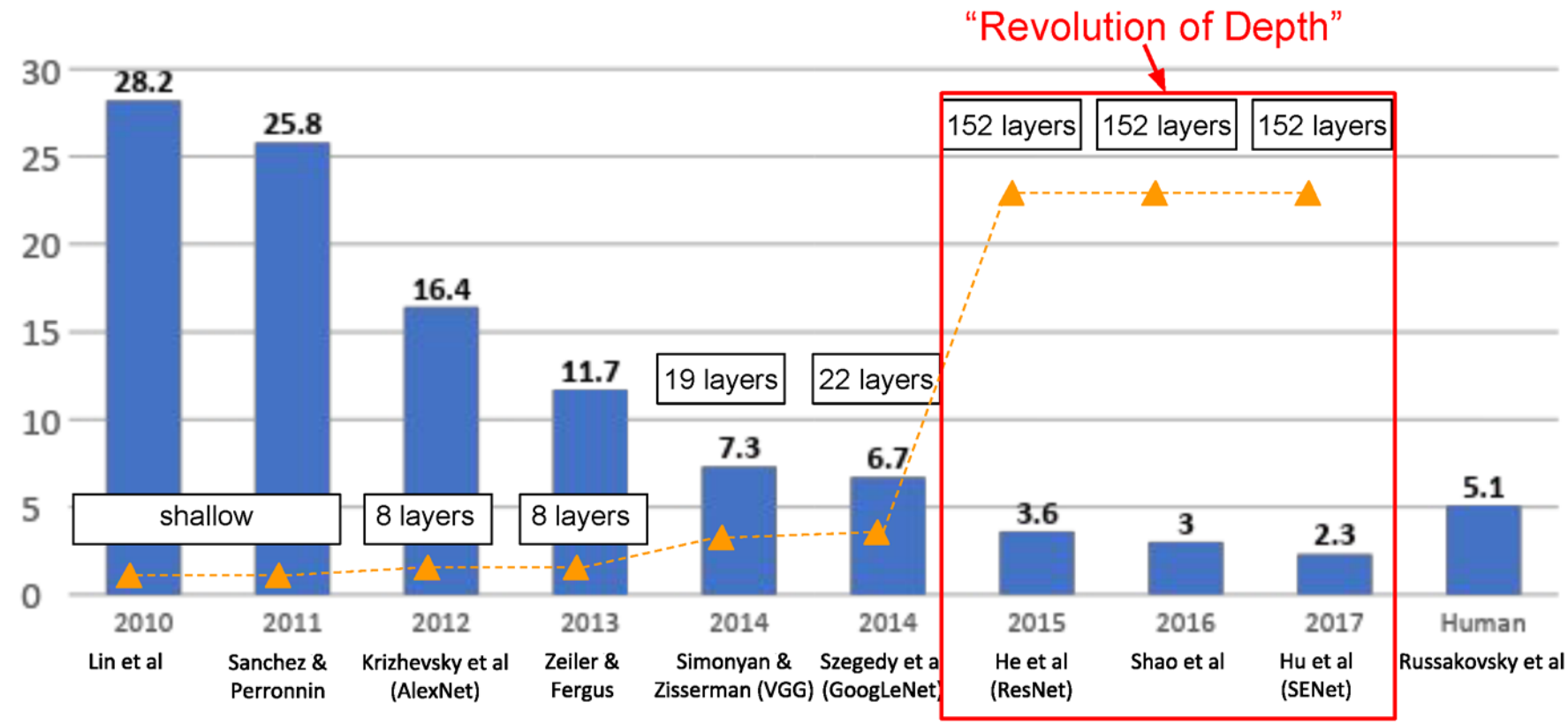


C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet



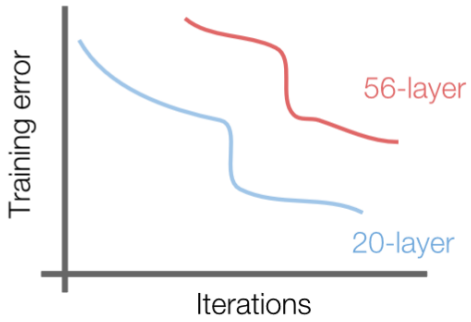
C. Szegedy et al., Going deeper with convolutions, CVPR 2015

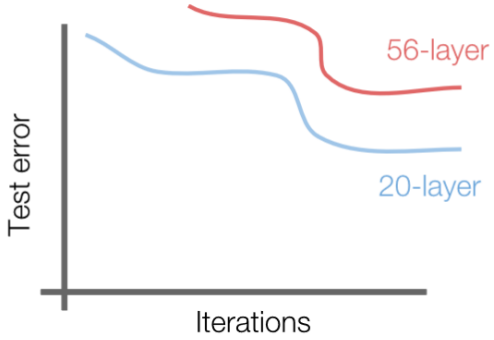

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
"Revolution of Depth"
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
3
2.3
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers
2010
2011
2012
2013
2014
2014
2015
2016
2017
Human
Lin et al
Sanchez & Perronnin
Krizhevsky et al (AlexNet)
Zeiler & Fergus
Simonyan & Zisserman (VGG)
Szegedy et a (GoogLeNet)
He et al (ResNet)
Shao et al
Hu et al (SENet)
Russakovsky et al

Case Study: ResNet

[He et al., 2015]

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





Q: What’s strange about these training and test curves?
[Hint: look at the order of the curves]

56-layer model performs worse on both training and test error
-> The deeper model performs worse, but it’s not caused by overfitting!

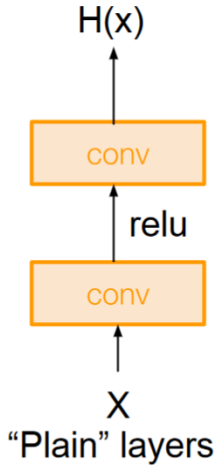
Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize

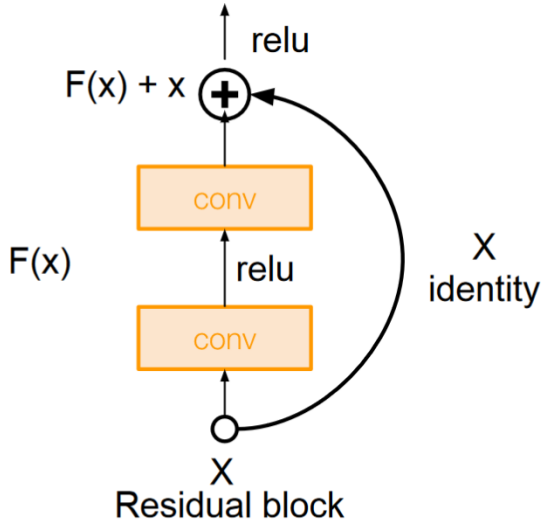
Case Study: ResNet

[He et al., 2015]

Skip connections

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

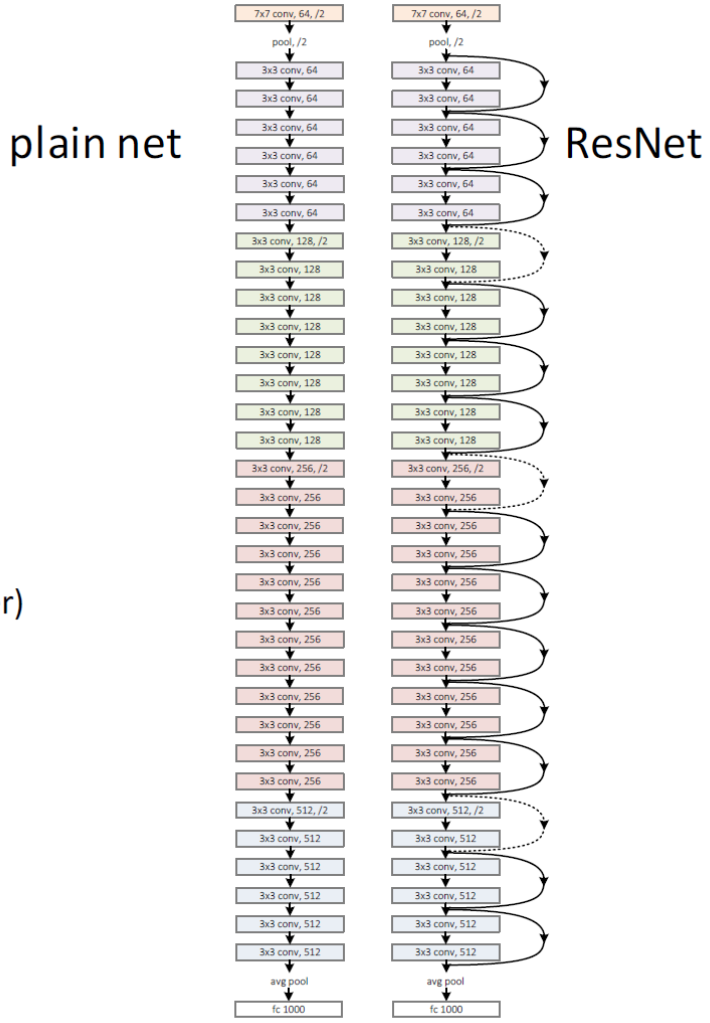




Residual Network

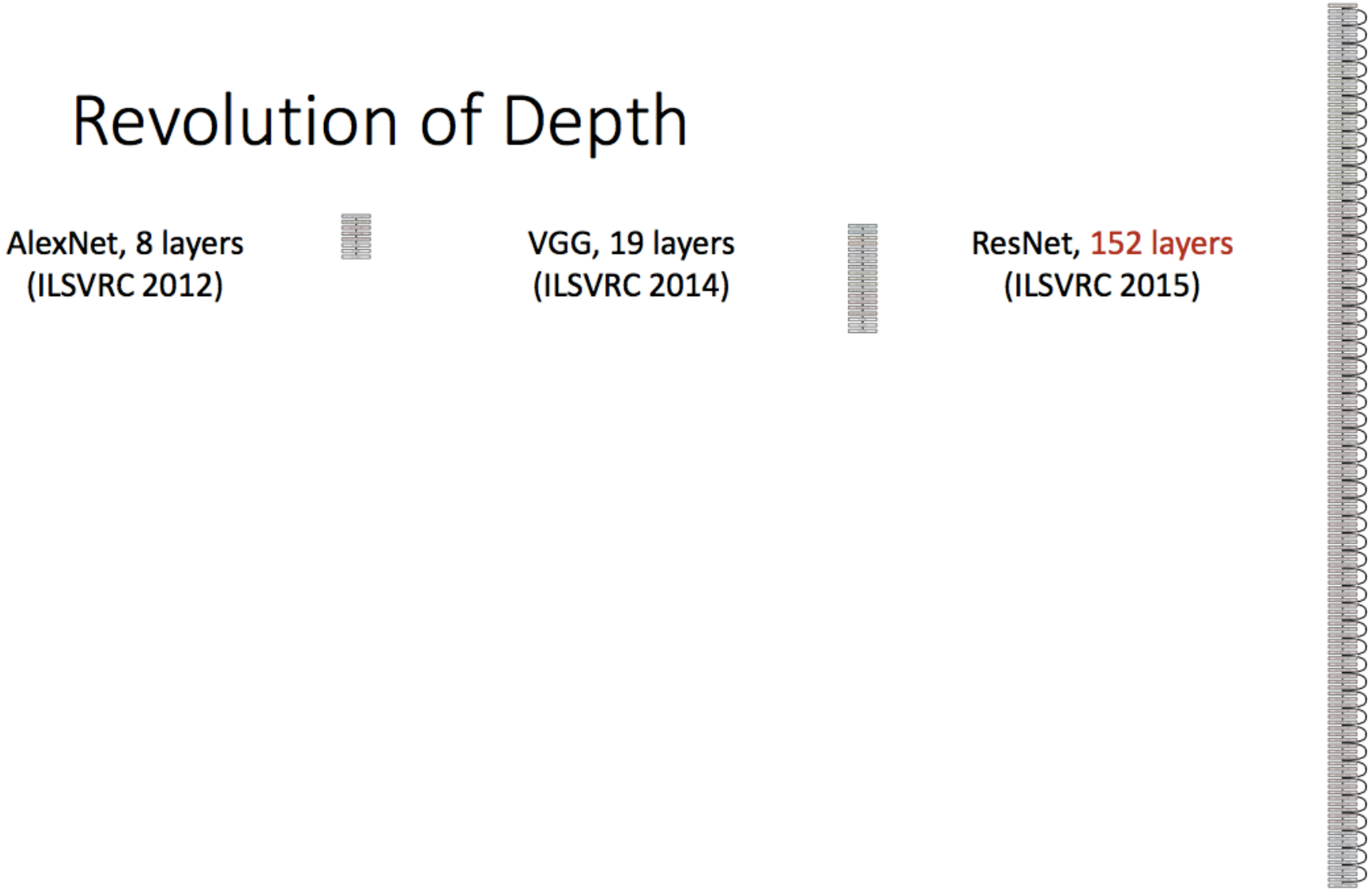
Network “Design”

- Keep it simple
- Our basic design (VGG-style)
  - all 3x3 conv (almost)
  - spatial size /2 => # filters x2 (~same complexity per layer)
  - Simple design; just deep!
- Other remarks:
  - no hidden fc
  - no dropout



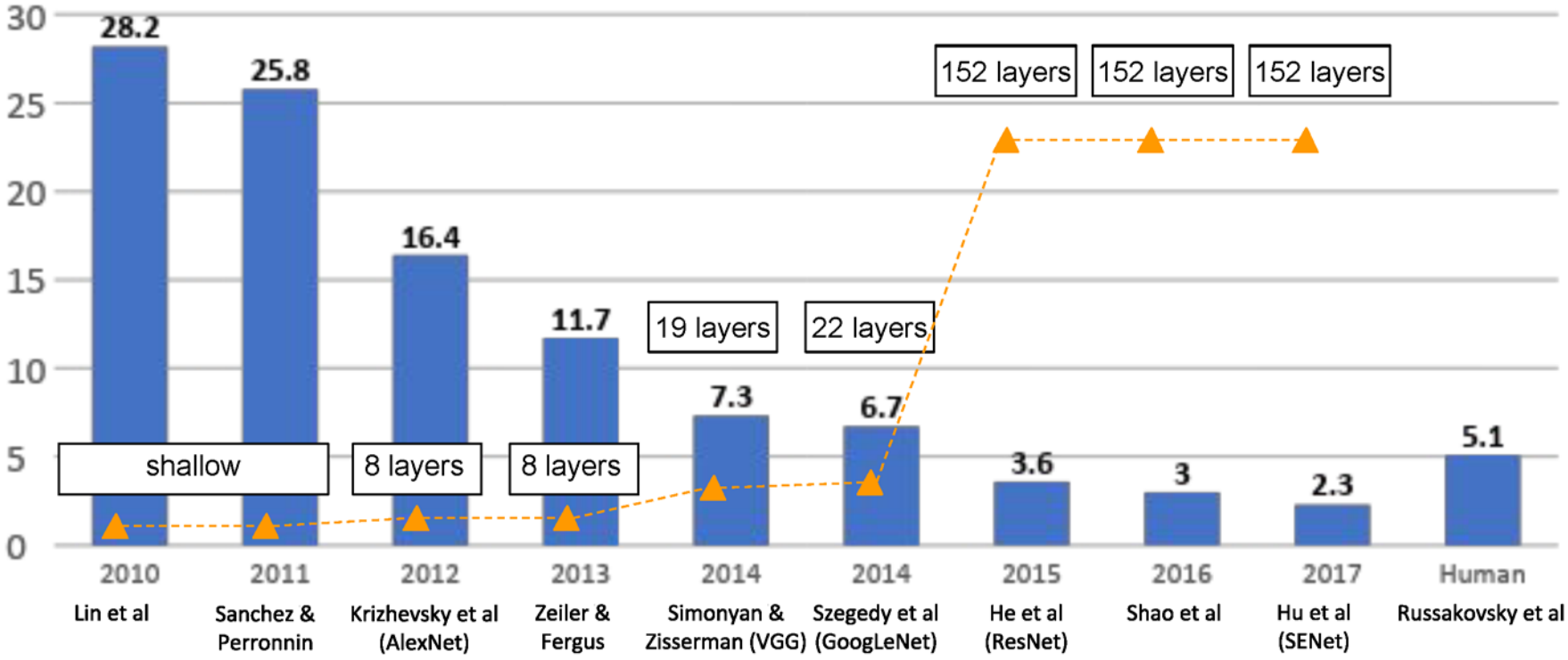
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

ResNet: going real deep



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

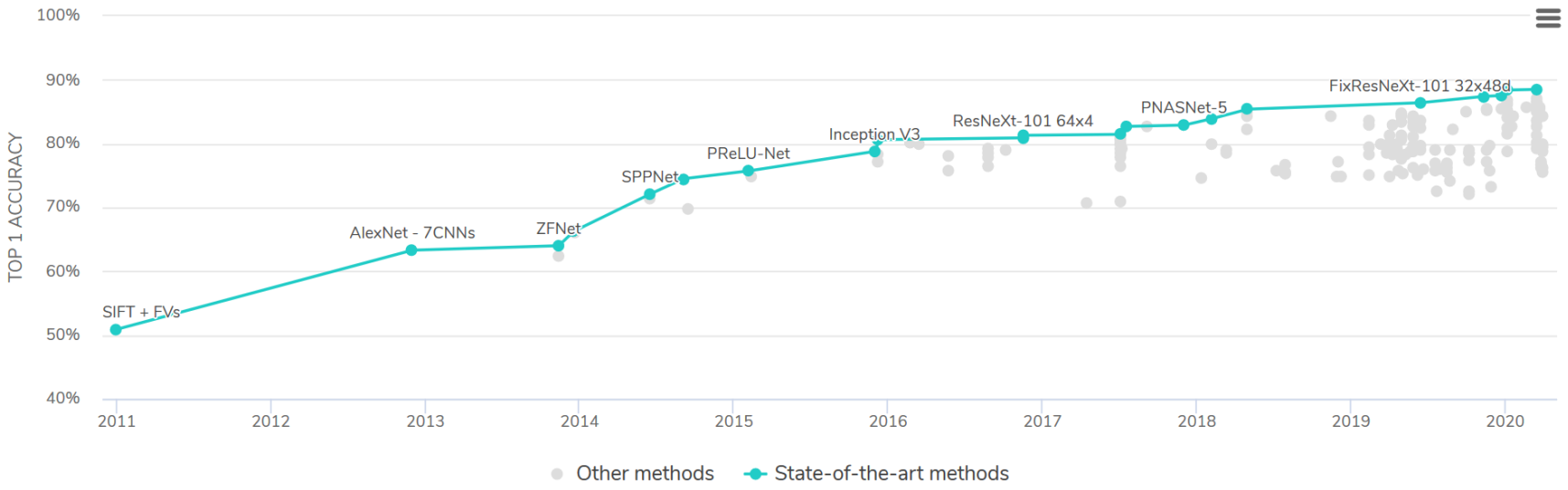
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



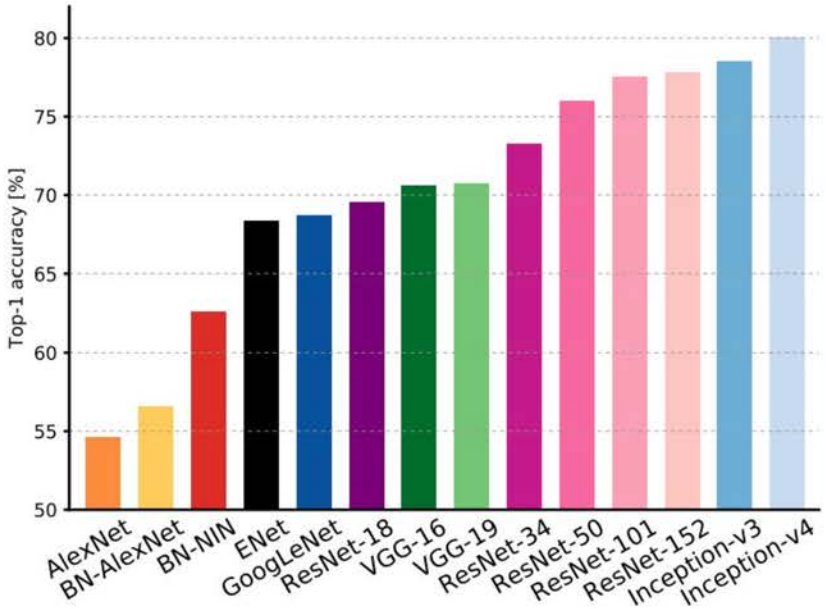
State of the art is improving

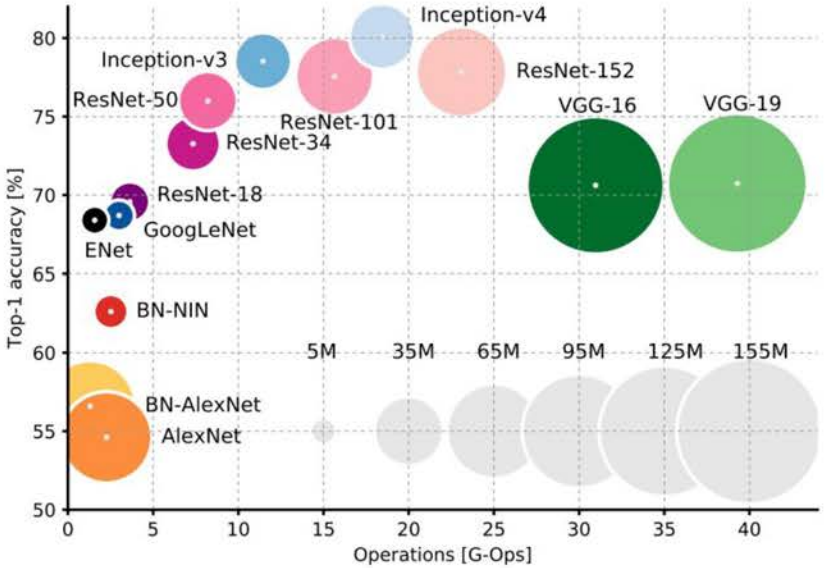
Browse > Computer Vision > Image Classification > ImageNet dataset

Image Classification on ImageNet



Comparing complexity…

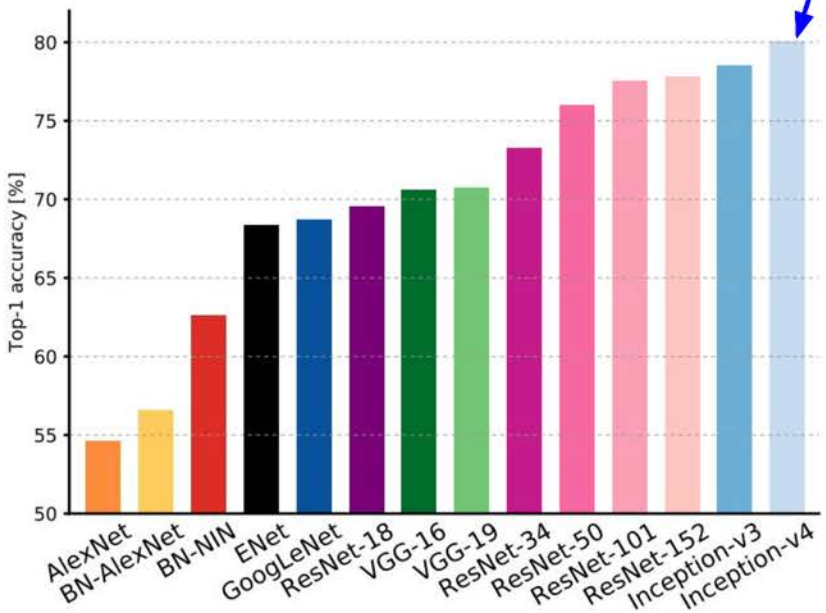


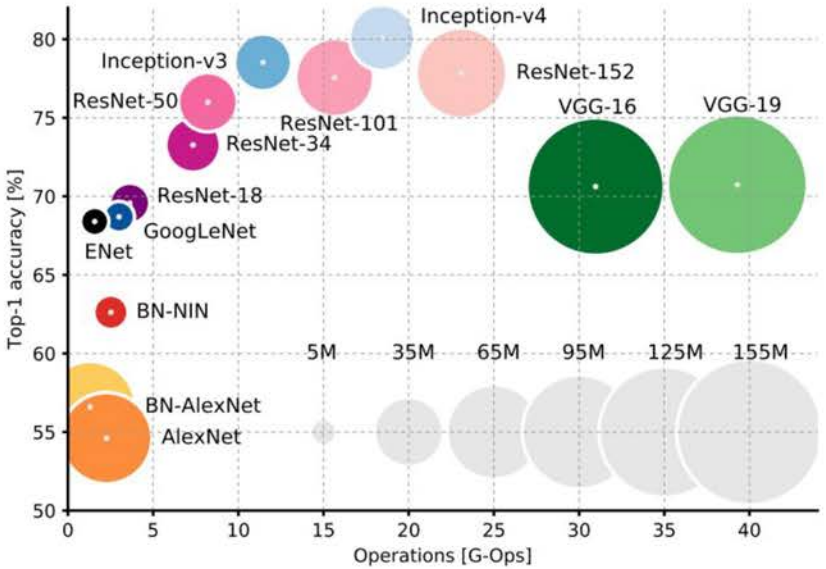


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparing complexity…

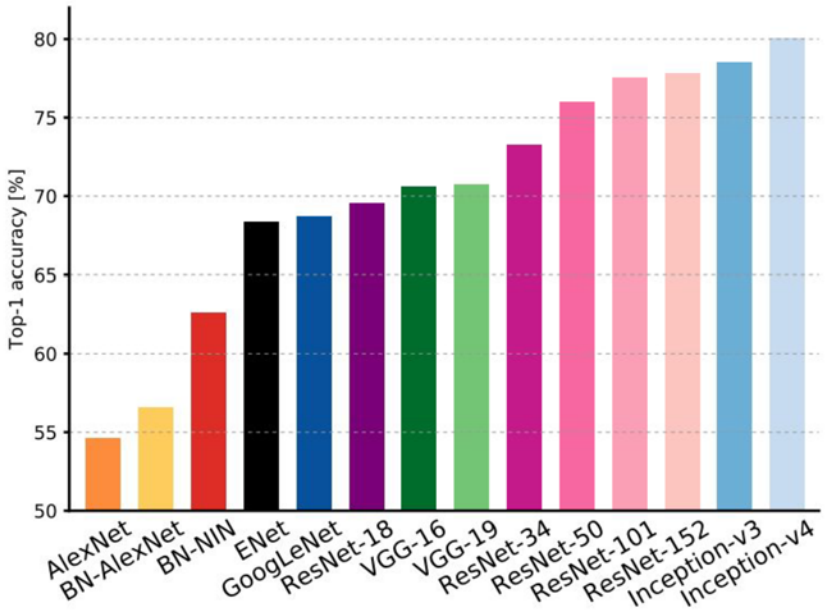
Inception-v4: Resnet + Inception!



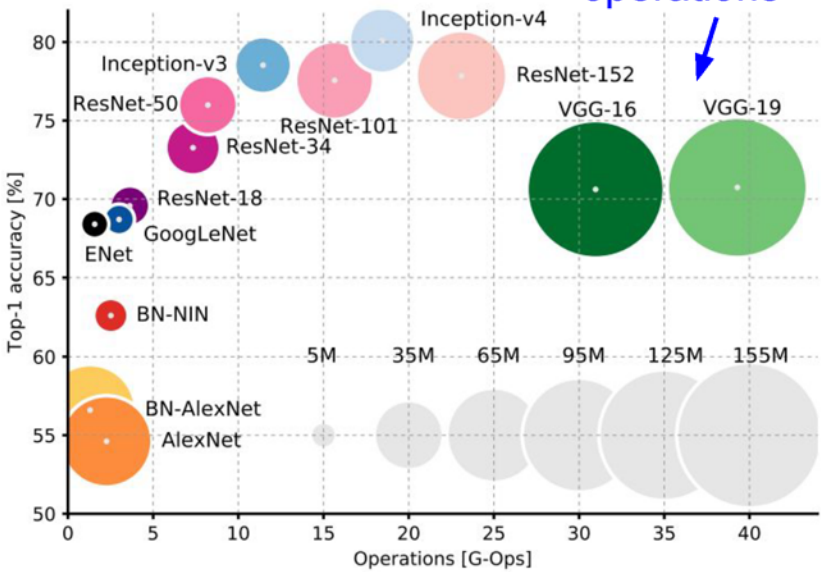


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparing complexity…

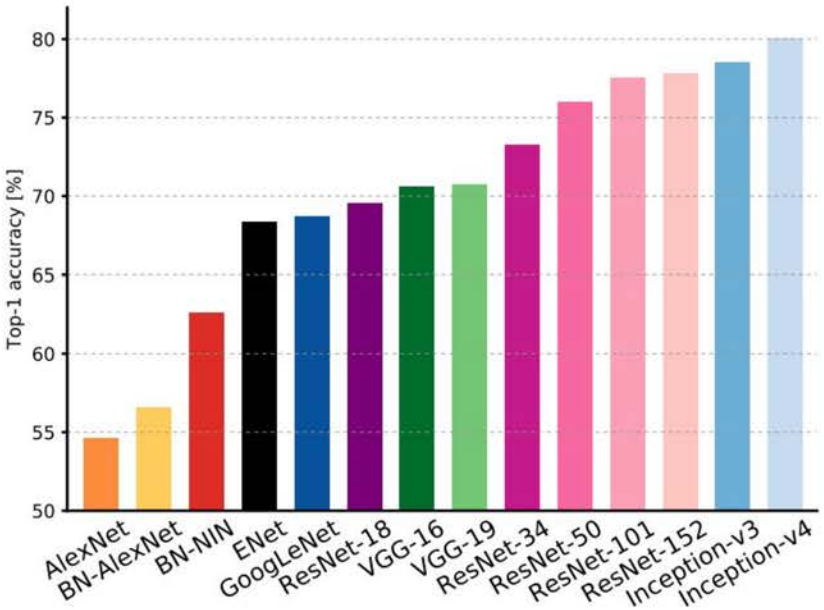


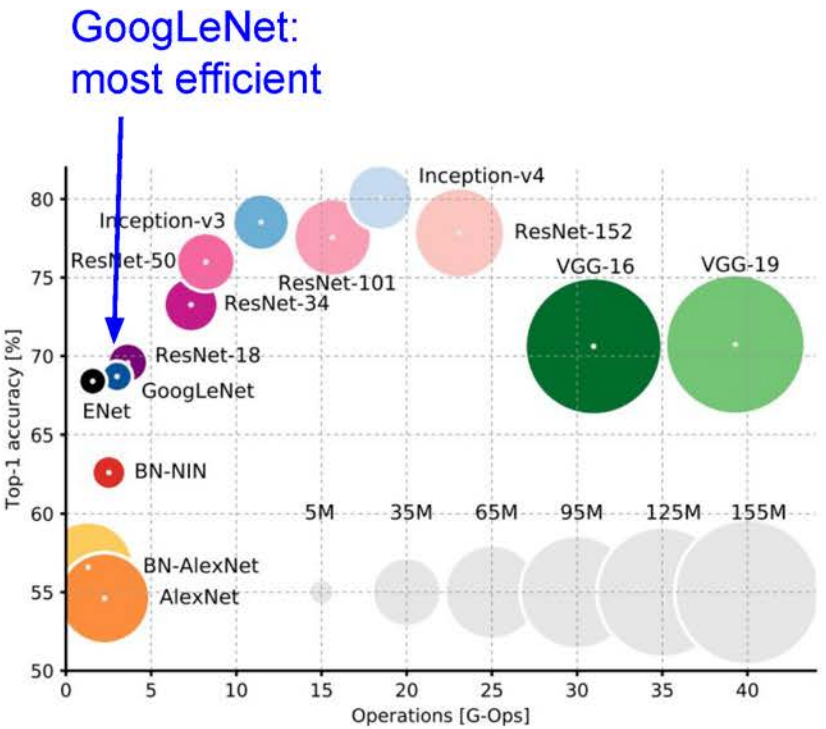
VGG: Highest memory, most operations



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

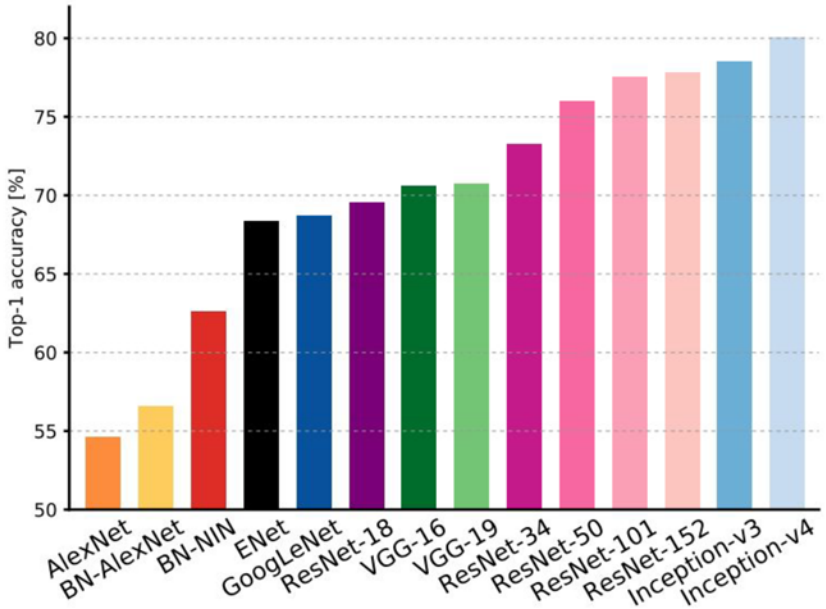
Comparing complexity…



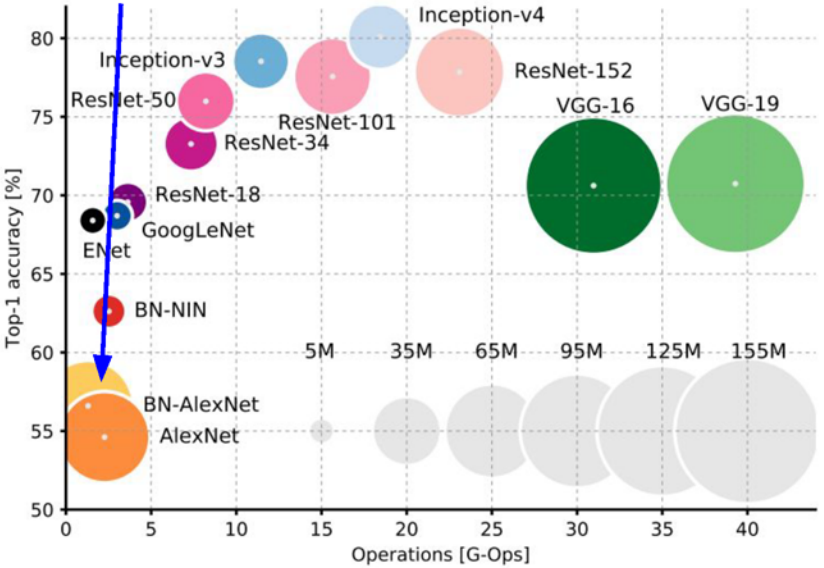


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparing complexity…

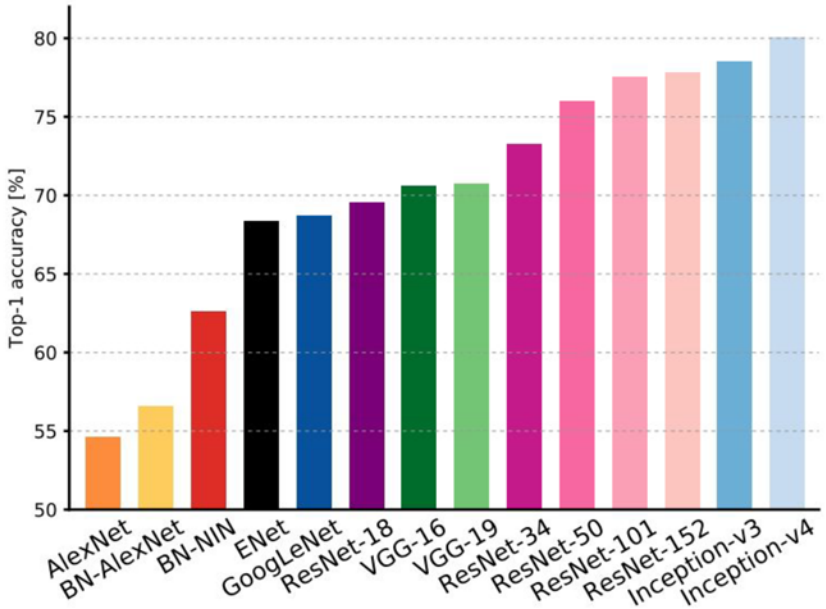


AlexNet:
Smaller compute, still memory heavy, lower accuracy

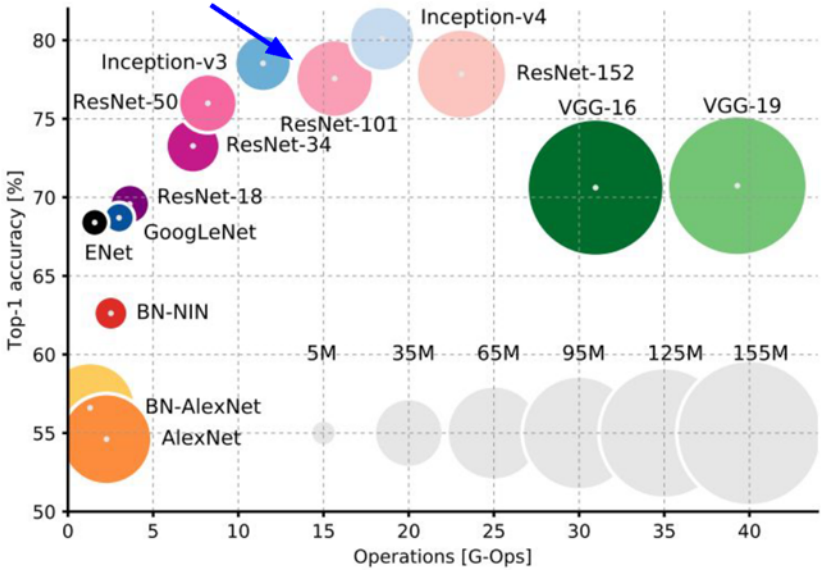


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparing complexity…



ResNet:
Moderate efficiency depending on model, highest accuracy



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

Summary: CNN Architectures

- VGG, GoogLeNet, ResNet all in wide use, available in model zoos
- ResNet current best default, also consider SENet when available
- Trend towards extremely deep networks
- Significant research centers around design of layer / skip connections and improving gradient flow
- Efforts to investigate necessity of depth vs. width and residual connections
- Even more recent trend towards meta-learning

Key ideas of CNN Architectures

- **Convolutional layers**
  - Same local functions evaluated everywhere → much fewer parameters
- **Pooling**
  - Larger receptive field and translational invariance
- **ReLU**:
  - Maintain a gradient signal over large portion of domain
- **Limit parameters**
  - Sequence of 3x3 filters instead of large filters (also encodes that local pixels are more relevant)
  - 1x1 convs to reduce feature dimensions
- **Skip network**
  - Prevents having to maintain early layers (just add residual)
  - Acts as ensemble

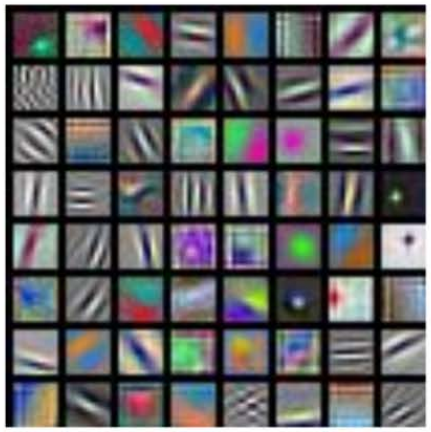
Key ideas of optimization

- **Stochastic gradient descent (SGD) in batches**
  - Batch size 128 or 256 recommended
  - Use ADAM for gradient/momentum
- **Normalize inputs/features (similar idea to whitening)**
  - Batchnorm normalizes inputs to each layer by estimate (e.g. moving average) of mean/std
- **Crazy optimization problem (so many local minima), but**
  - Model capacity is larger than needed to help ensure that important patterns are discovered
  - Many solutions are similarly good (e.g. can permute layers without effect)

Good discussion post on local minima

What does the CNN learn?

First Layer: Visualize Filters

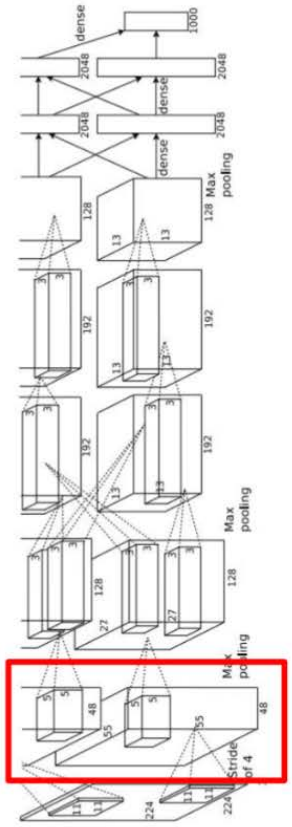


AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7

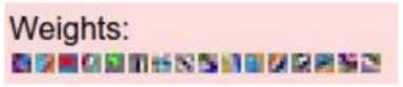


Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Huang et al. "Densely Connected Convolutional Networks", CVPR 2017

Visualize the filters/kernels (raw weights)

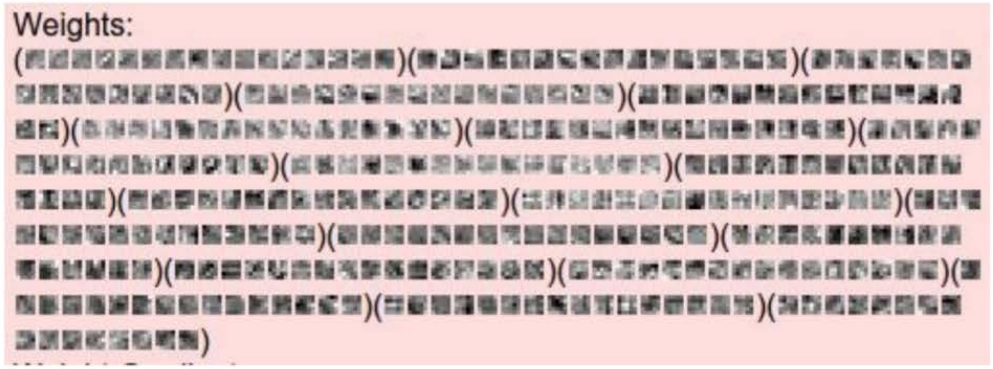
We can visualize filters at higher layers, but not that interesting

(these are taken from ConvNetJS CIFAR-10 demo)



layer 1 weights

16 x 3 x 7 x 7



layer 2 weights

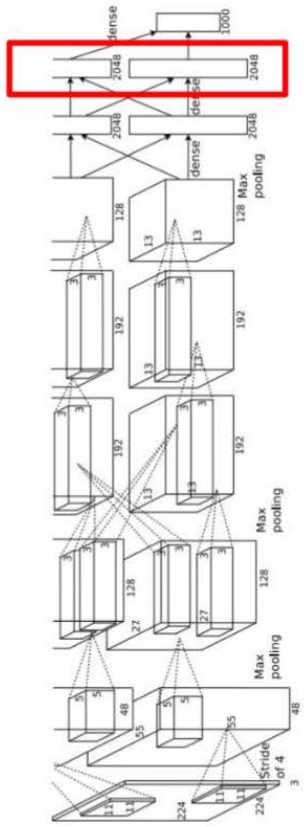
20 x 16 x 7 x 7

layer 3 weights

20 x 20 x 7 x 7

Last Layer

FC7 layer

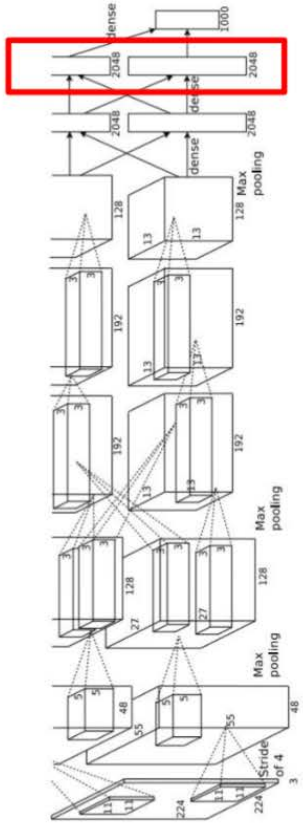


4096-dimensional feature vector for an image (layer immediately before the classifier)

Run the network on many images, collect the feature vectors

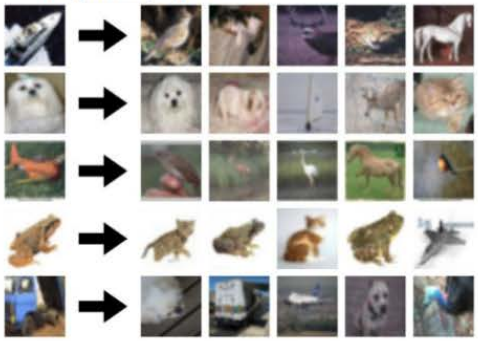
Last Layer: Nearest Neighbors

4096-dim vector



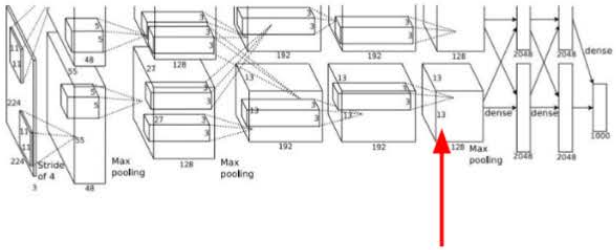
Test image L2 Nearest neighbors in feature space

Recall: Nearest neighbors in pixel space



Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figures reproduced with permission.

Maximally Activating Patches



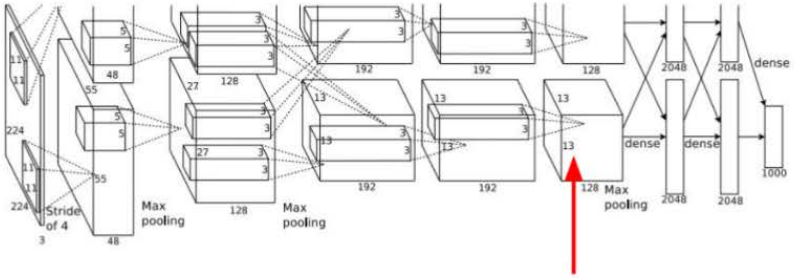
Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations

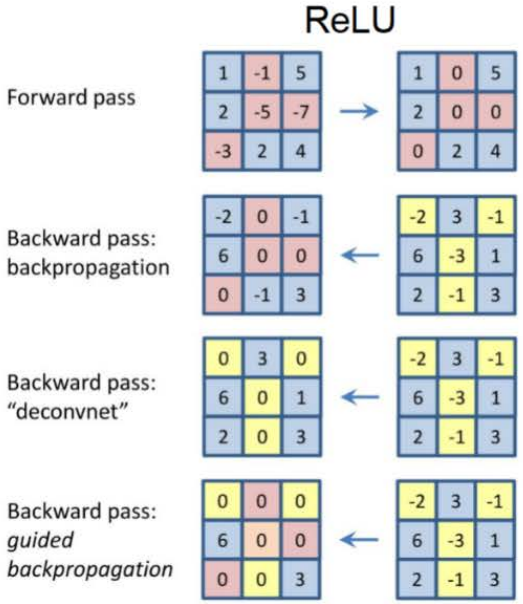
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

Intermediate Features via (guided) backprop



Pick a single intermediate neuron, e.g. one value in 128 x 13 x 13 conv5 feature map

Compute gradient of neuron value with respect to image pixels

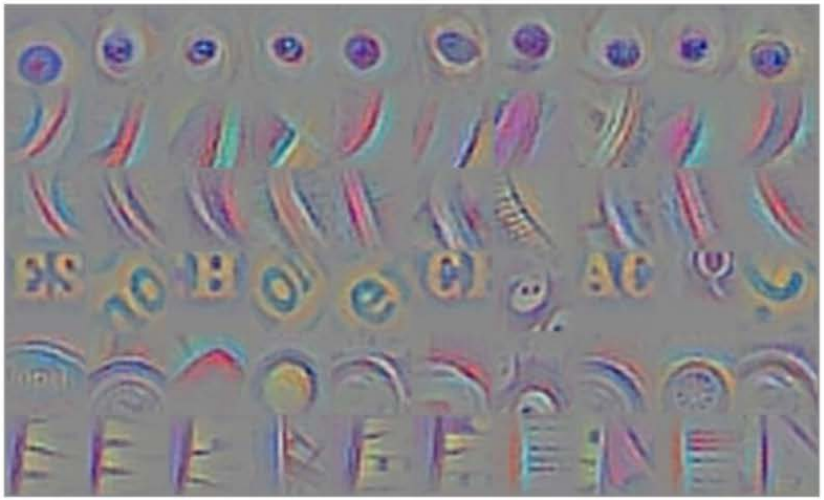


Images come out nicer if you only backprop positive gradients through each ReLU (guided backprop)

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

Intermediate features via (guided) backprop

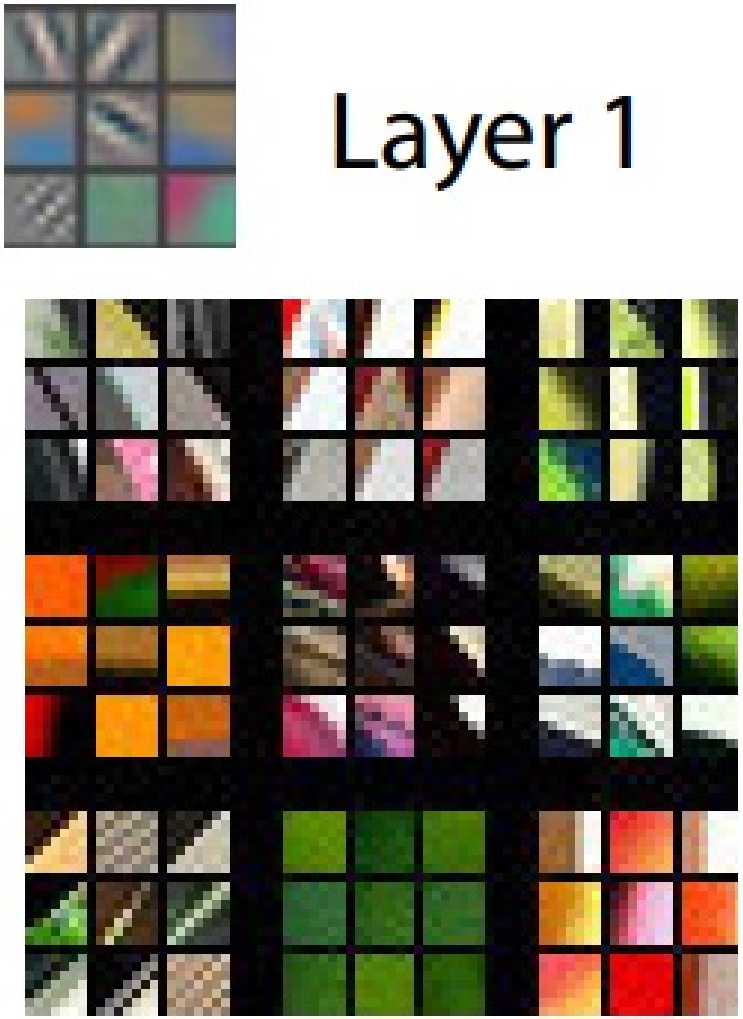
Maximally activating patches
(Each row is a different neuron)



Guided Backprop

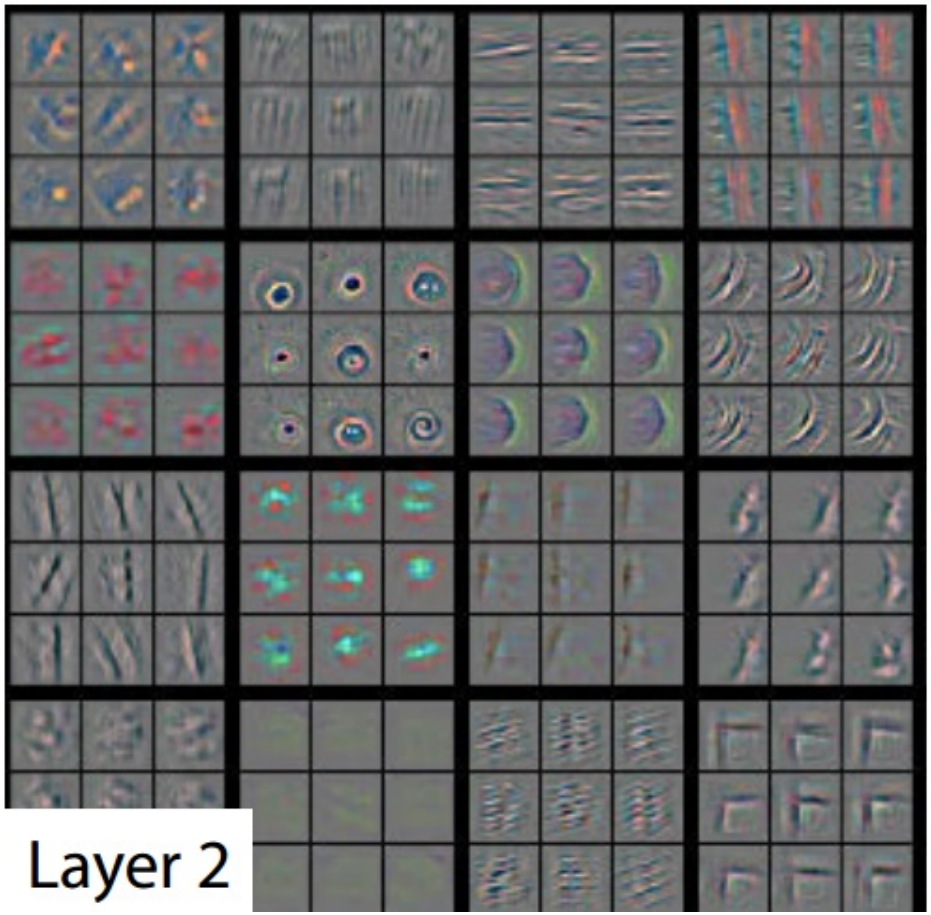
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

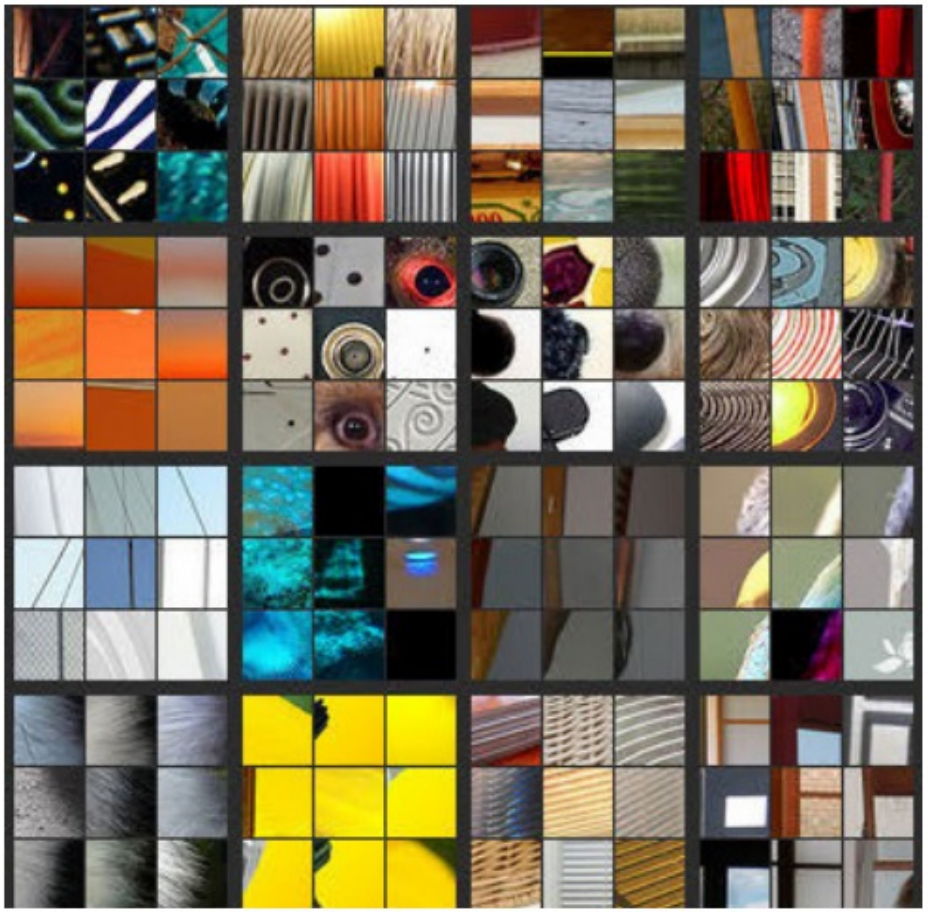
Layer 1



Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

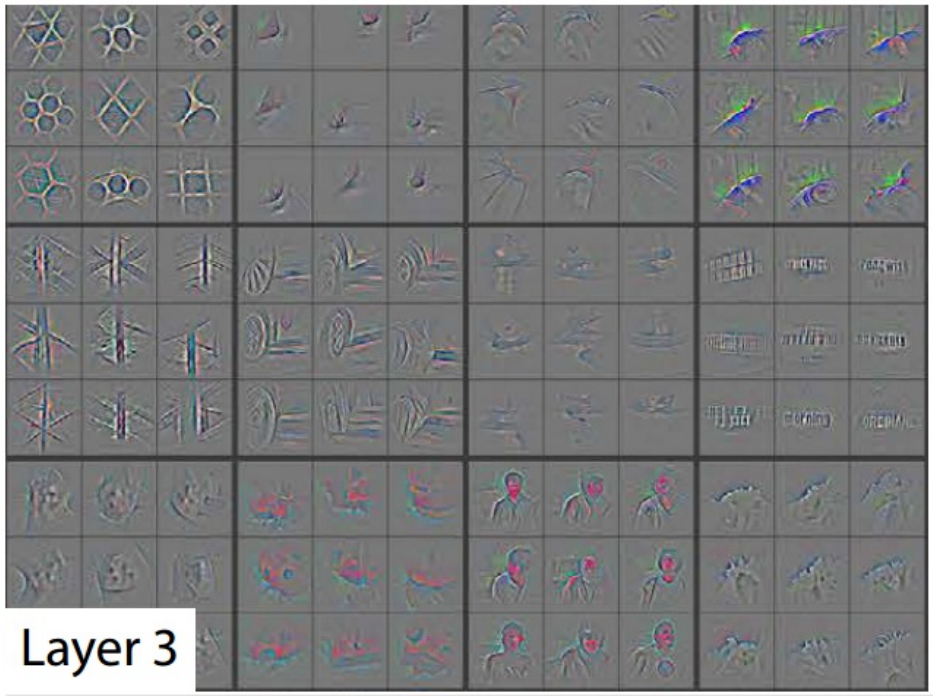
Layer 2





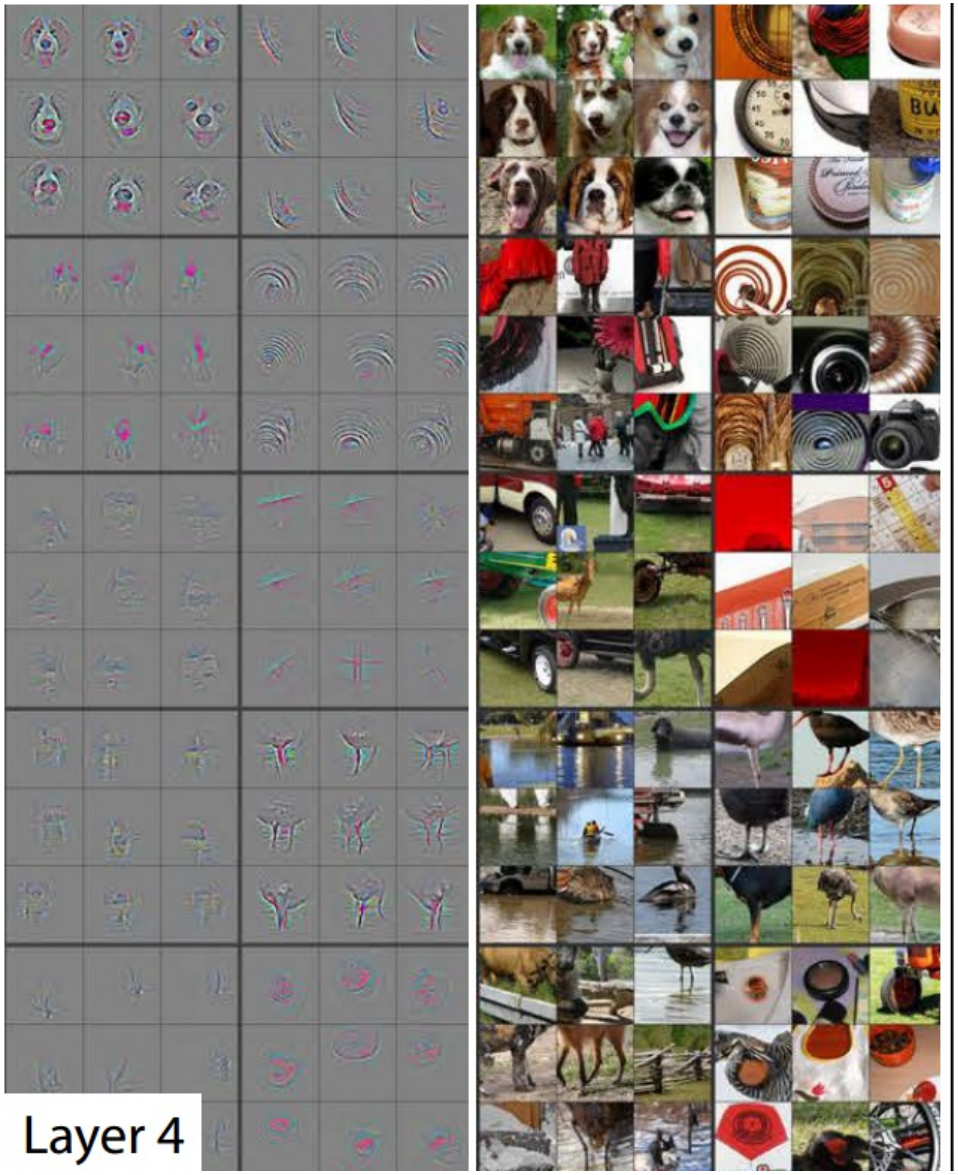
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

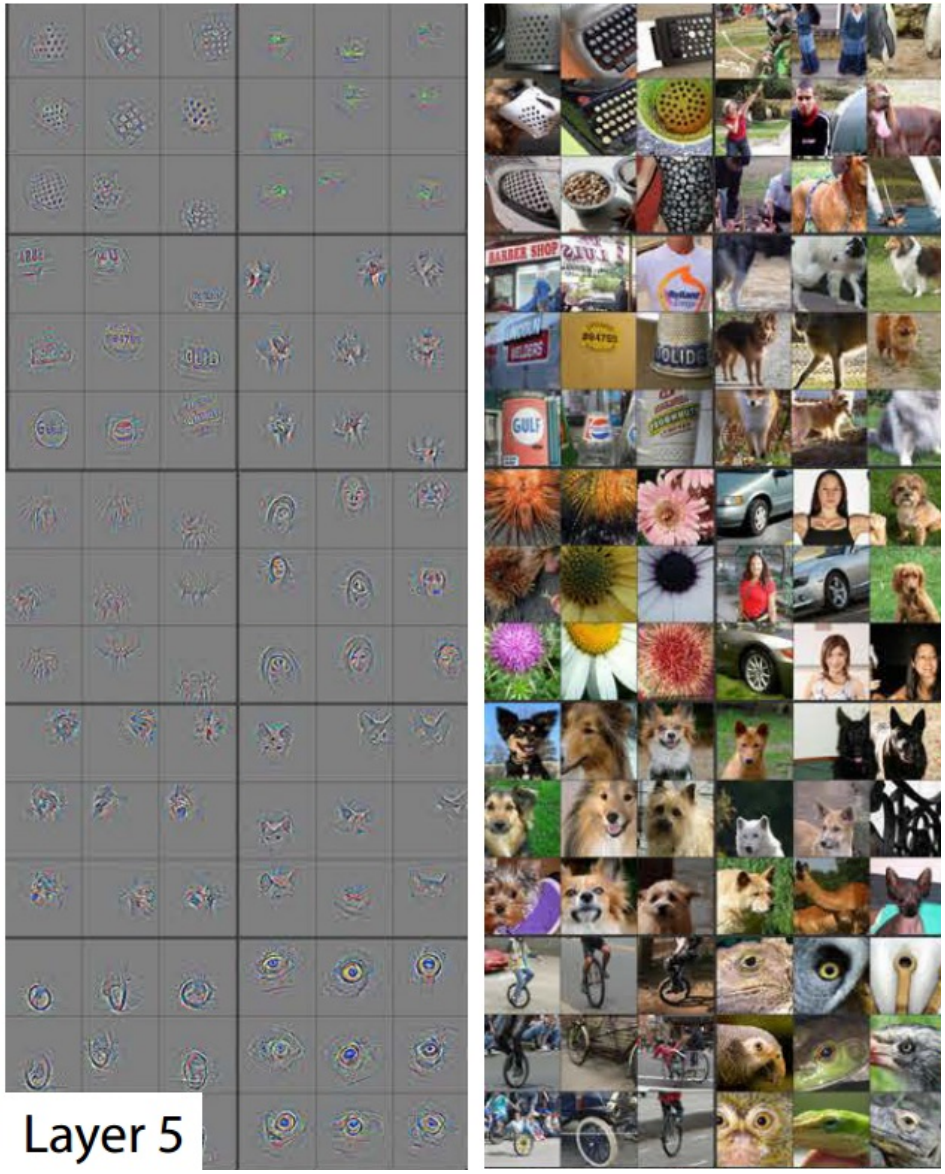
Layer 3



Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 4 and 5





Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Visualizing CNN features: Gradient Ascent

(Guided) backprop:
Find the part of an image that a neuron responds to

Gradient ascent:
Generate a synthetic image that maximally activates a neuron

$$I^* = \arg\max_I f(I) + R(I)$$

Neuron value

Natural image regularizer

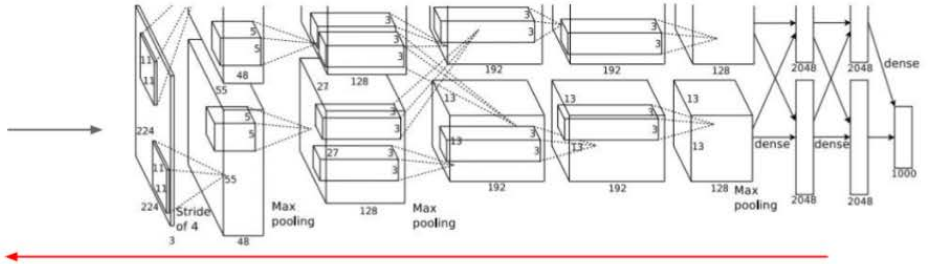
Visualizing CNN features: Gradient Ascent

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

score for class c (before Softmax)

1. Initialize image to zeros

zero image



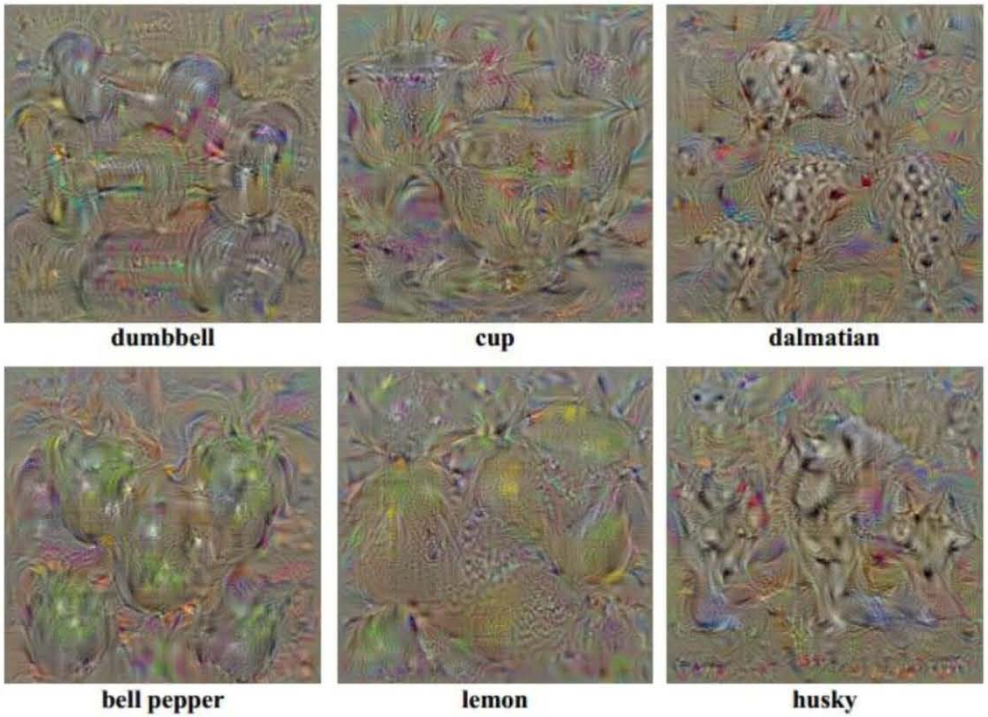
Repeat:

2. Forward image to compute current scores
3. Backprop to get gradient of neuron value with respect to image pixels
4. Make a small update to the image

Visualizing CNN features: Gradient Ascent

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

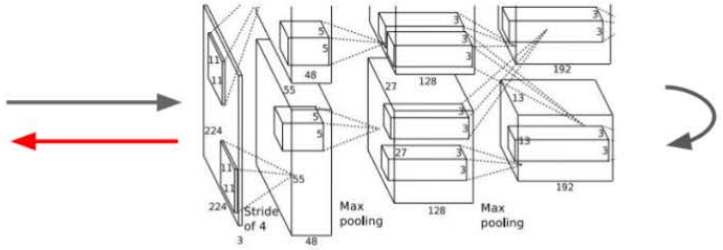
Simple regularizer: Penalize L2 norm of generated image



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

DeepDream: Amplify existing features

Rather than synthesizing an image to maximize a specific neuron, instead try to **amplify** the neuron activations at some layer in the network



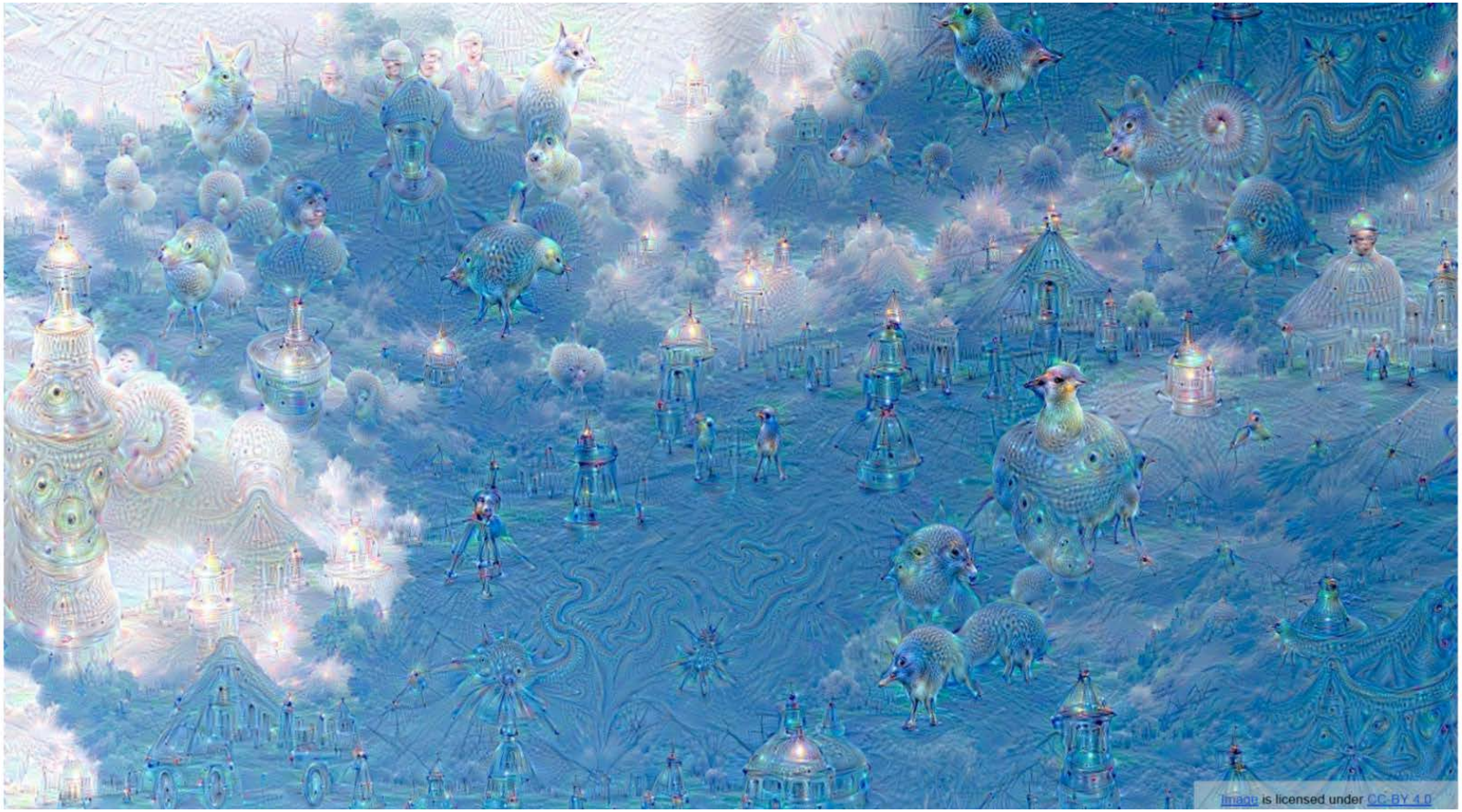
Choose an image and a layer in a CNN; repeat:

1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
3. Backward: Compute gradient on image
4. Update image

Equivalent to:

$$I^* = \arg\max_I \sum_i f_i(I)^2$$

Mordvintsev, Olah, and Tyka, "Inceptionism: Going Deeper into Neural Networks", Google Research Blog. Images are licensed under CC-BY


Imagu is licensed under CC BY 4.0

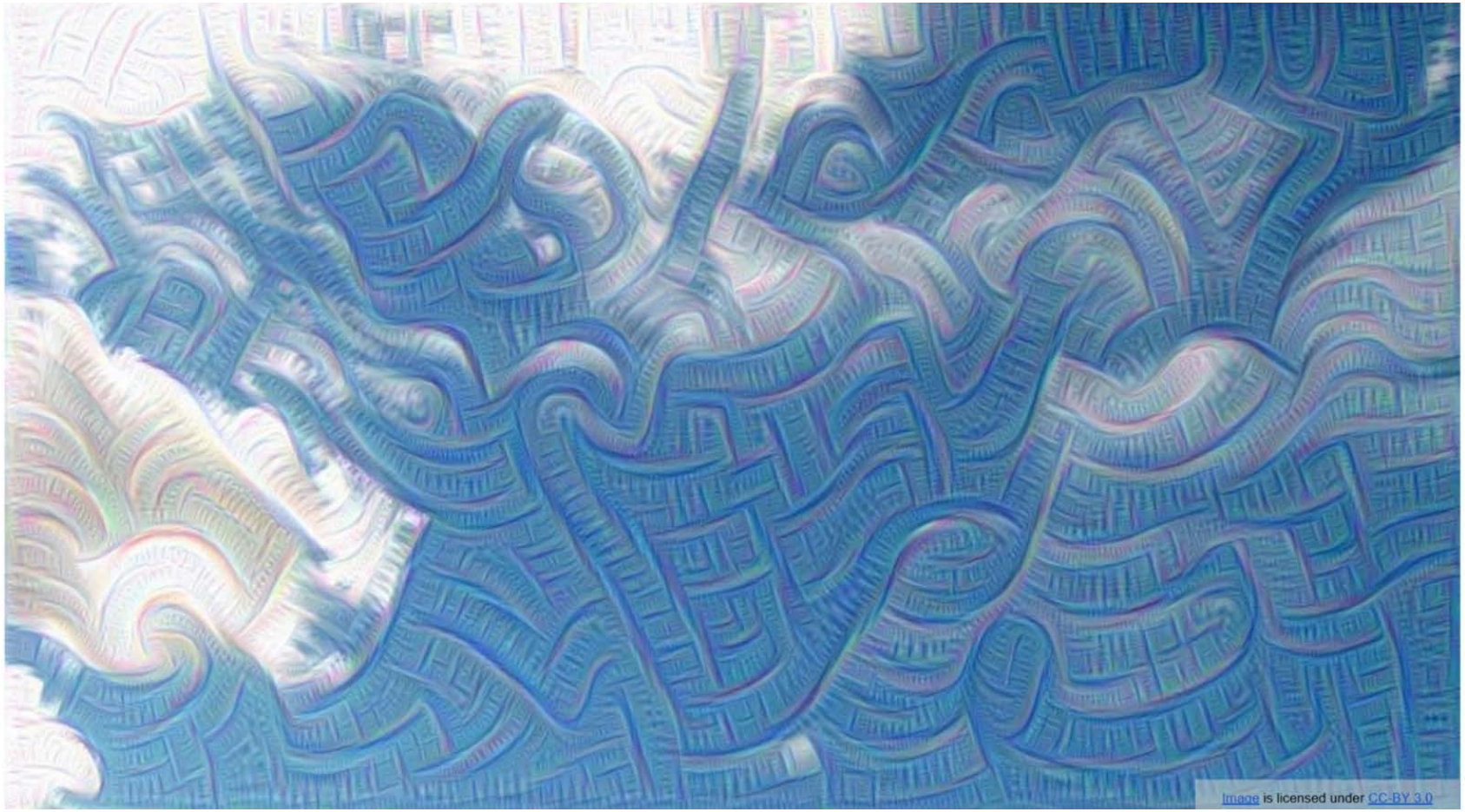
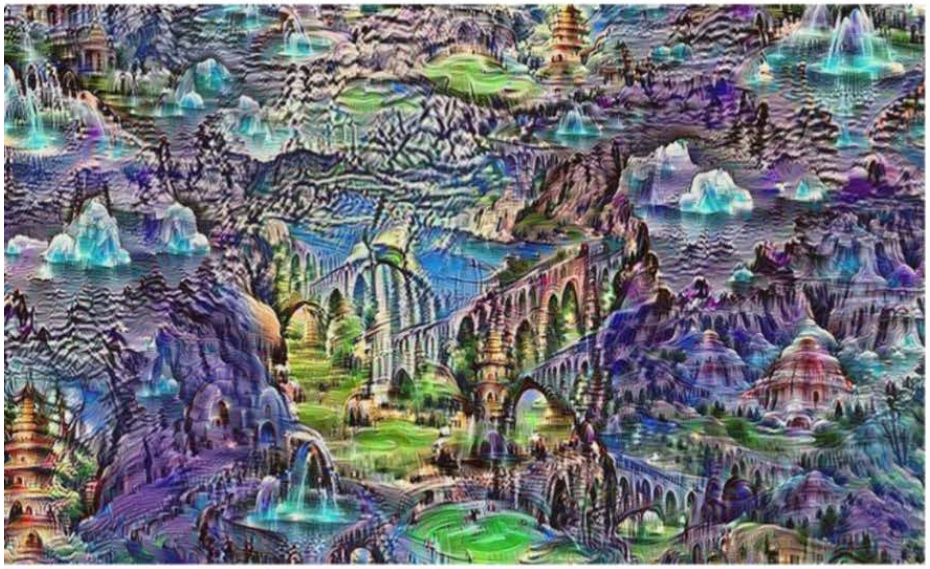
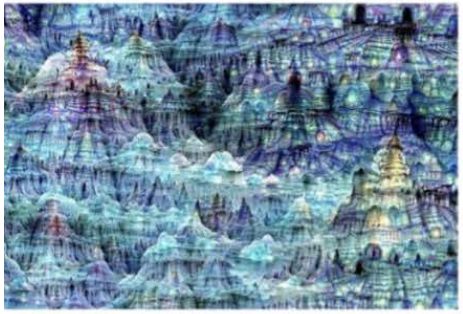
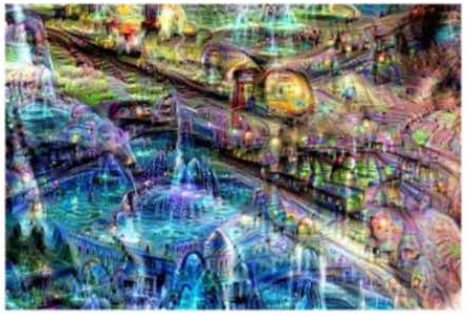
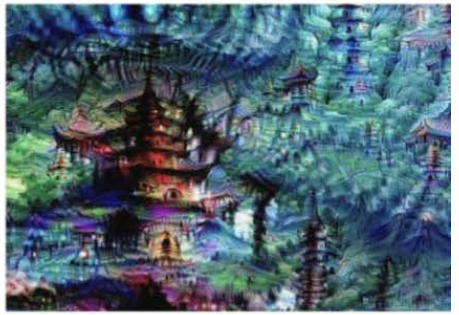


Image is licensed under CC-BY 3.0

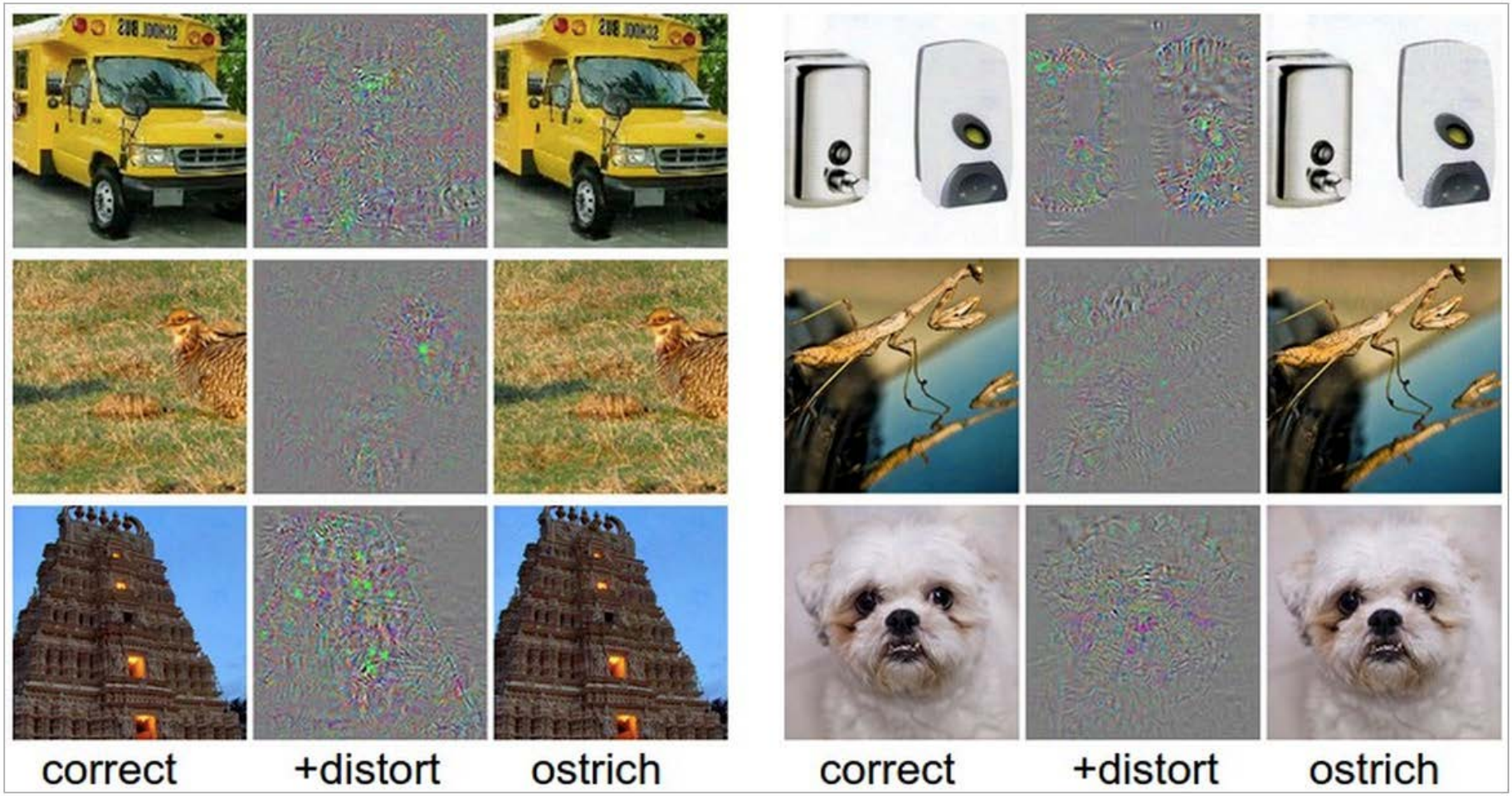

Image is licensed under CC-BY 3.0





Fooling CNNs



Take a correctly classified image (left image in both columns), and add a tiny distortion (middle) to fool the ConvNet with the resulting image (right).

http://arxiv.org/abs/1312.6199

http://karpathy.github.io/2015/03/30/breaking-convnets/

What is going on?

- Recall gradient descent training: modify the weights to reduce classifier error

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$$

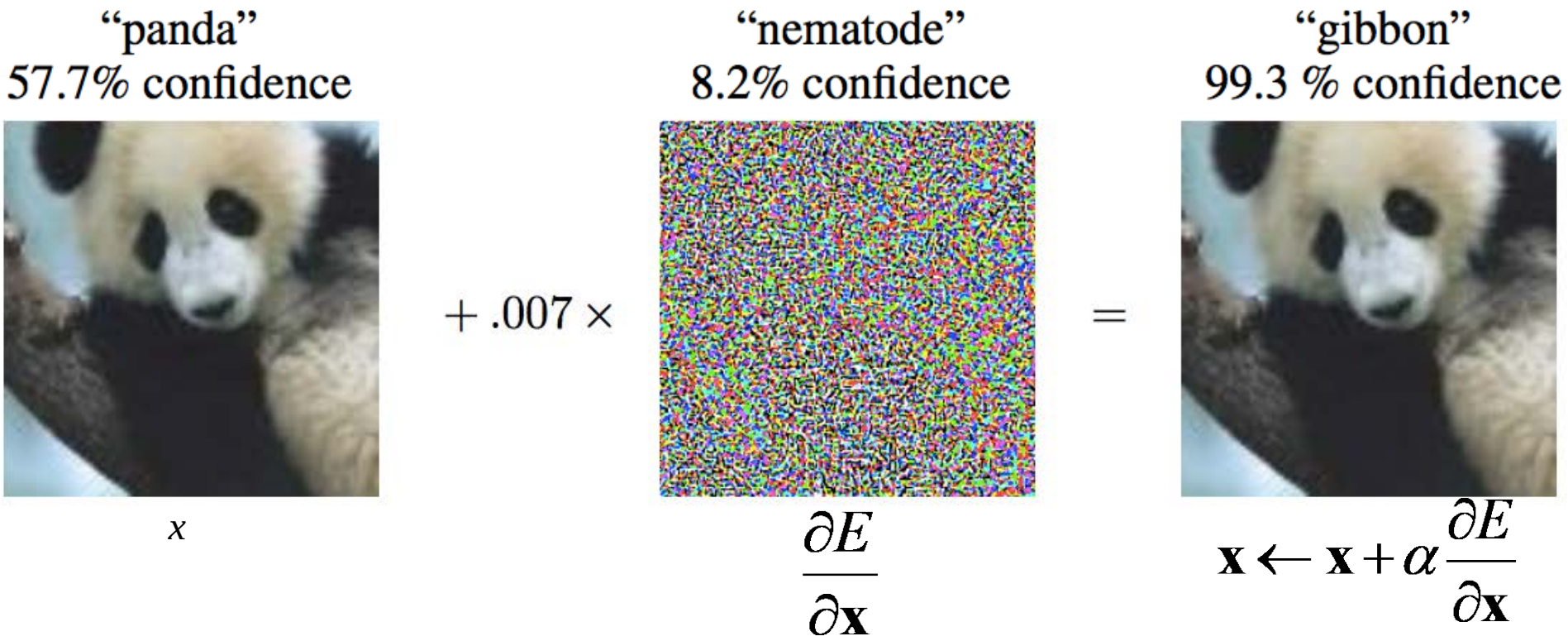
- Adversarial examples: modify the *image* to *increase* classifier error

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \frac{\partial E}{\partial \mathbf{x}}$$

http://arxiv.org/abs/1412.6572

http://karpathy.github.io/2015/03/30/breaking-convnets/

What is going on?



http://arxiv.org/abs/1412.6572

http://karpathy.github.io/2015/03/30/breaking-convnets/

Fooling a linear classifier

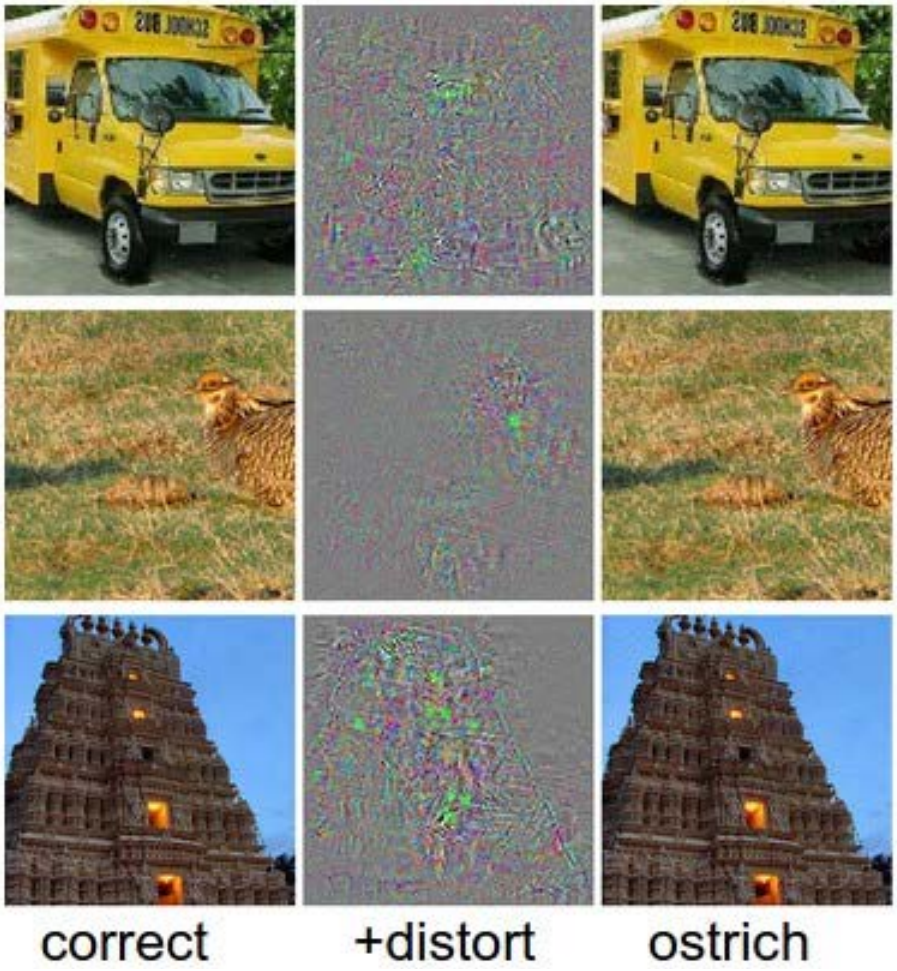
- Perceptron weight update: add a small multiple of the example to the weight vector:
-

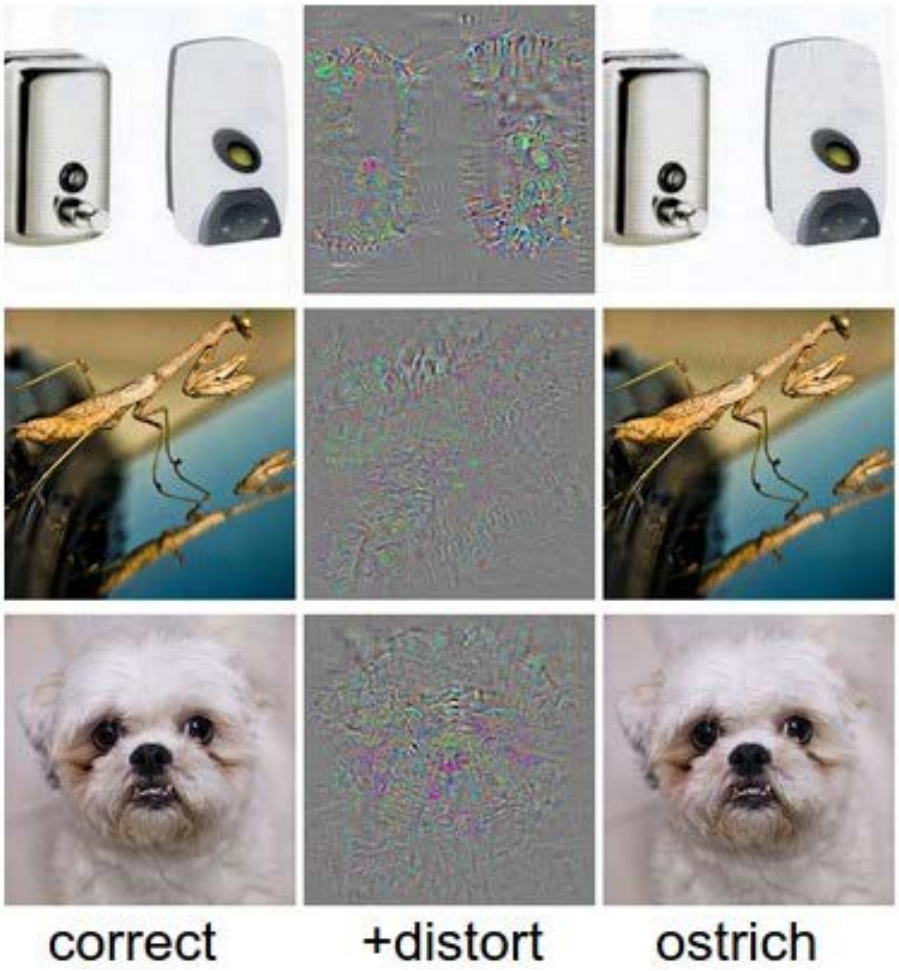
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \mathbf{x}$$

- To fool a linear classifier, add a small multiple of the weight vector to the training example:
-

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \mathbf{w}$$

Fooling a linear classifier





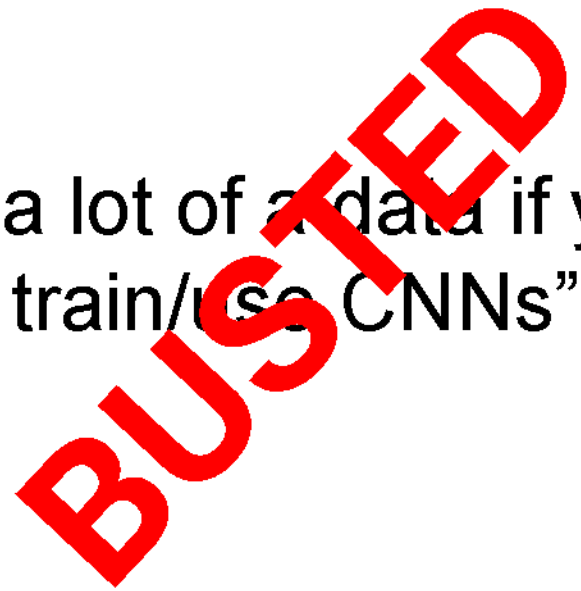
http://karpathy.github.io/2015/03/30/breaking-convnets/

Transfer Learning

“You need a lot of a data if you want to train/use CNNs”

Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

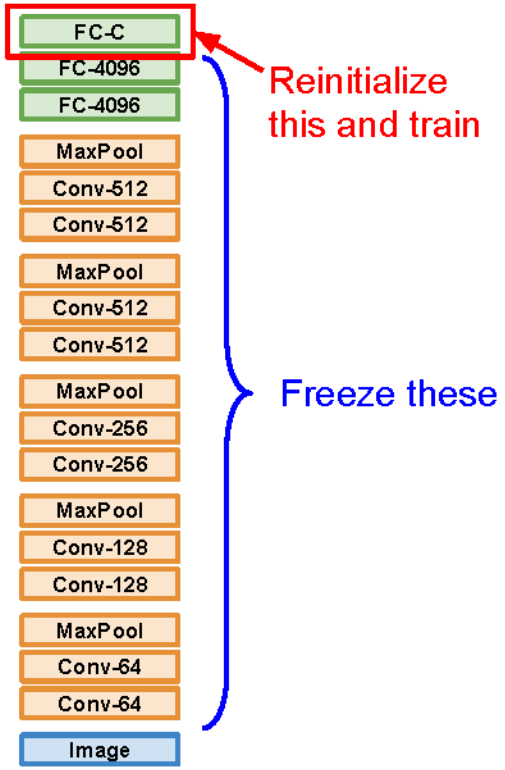


Transfer Learning with CNNs

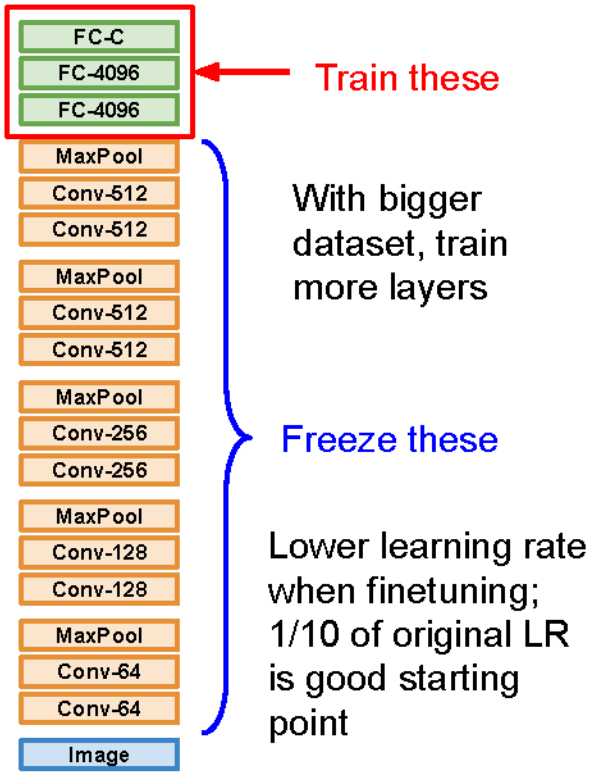
Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

1. Train on Imagenet

2. Small Dataset (C classes)

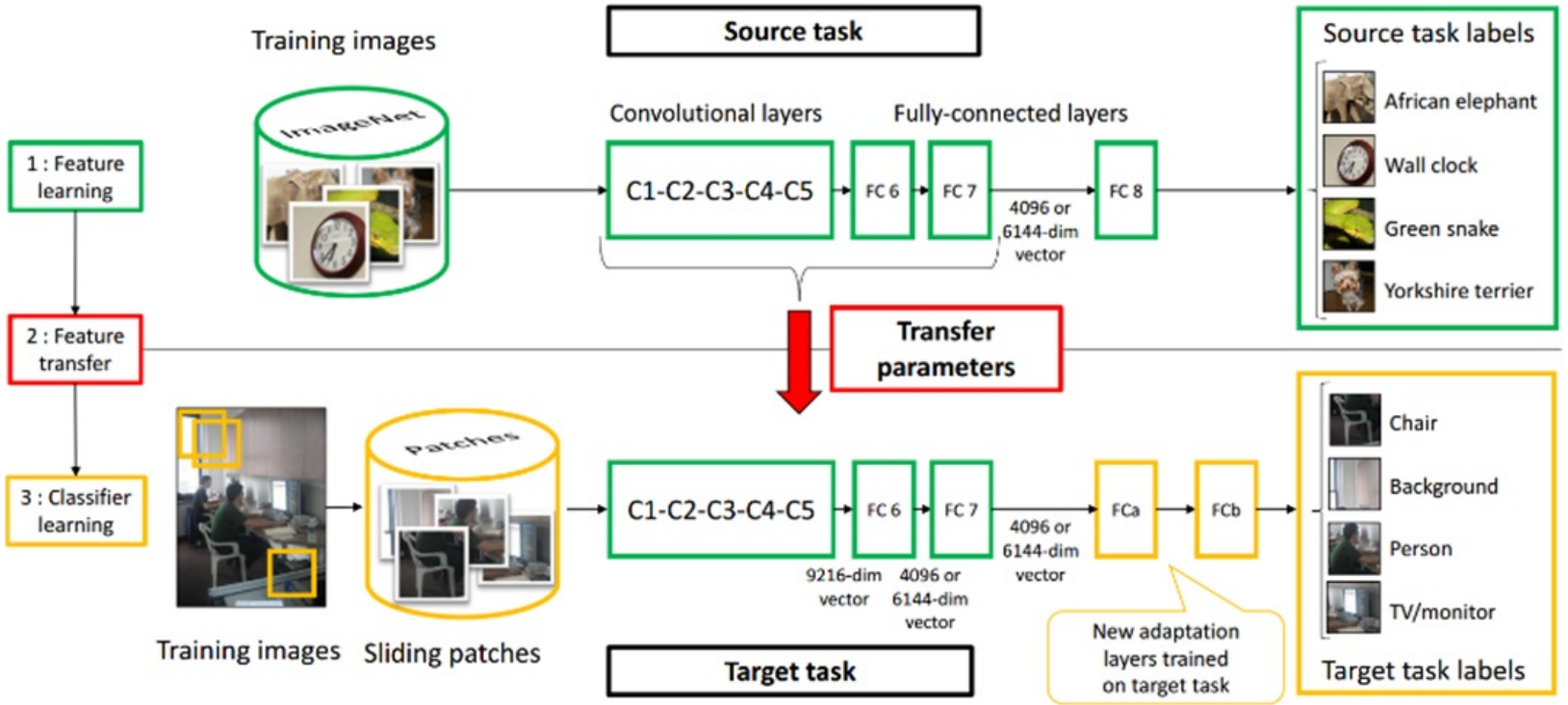


3. Bigger dataset



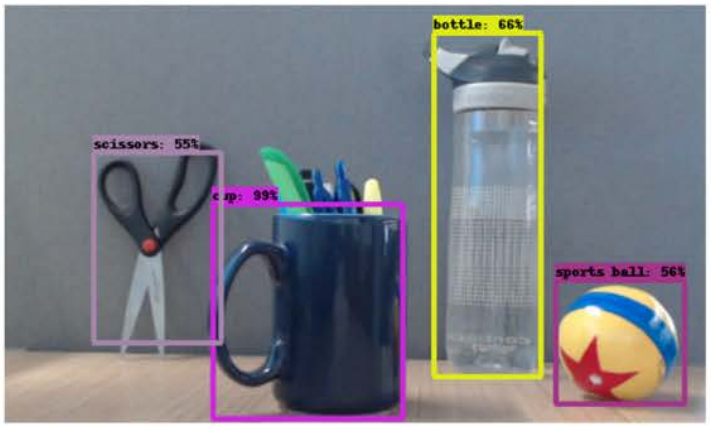
Transfer Learning

- Improvement of learning in a **new** task through the *transfer of knowledge* from a **related** task that has already been learned.
- Weight initialization for CNN

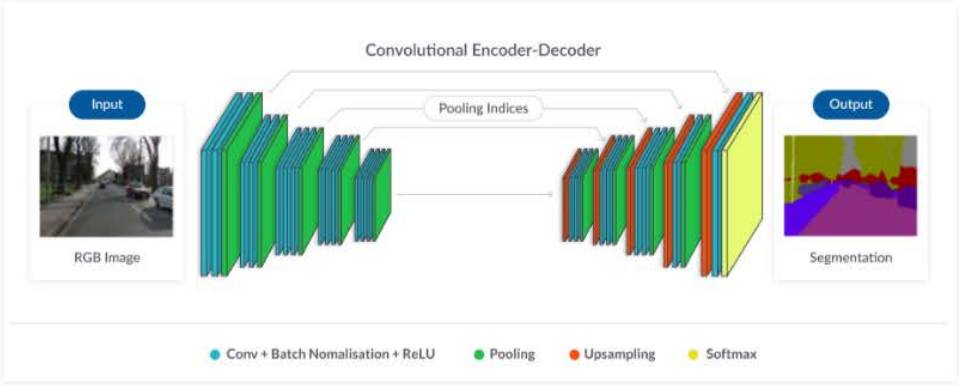


Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks [Oquab et al. CVPR 2014]

CNNs used for many tasks beyond classification

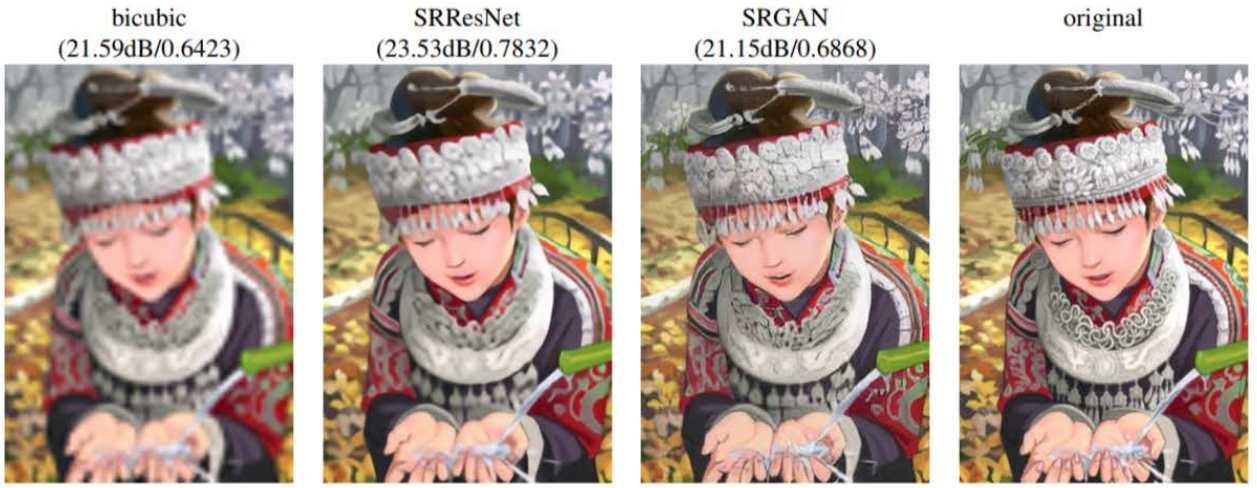
- **Object Detection**
  - 

  - Source (https://medium.com/better-programming/real-time-object-detection-on-gpus-in-10-minutes-6e8c9b857bb3)

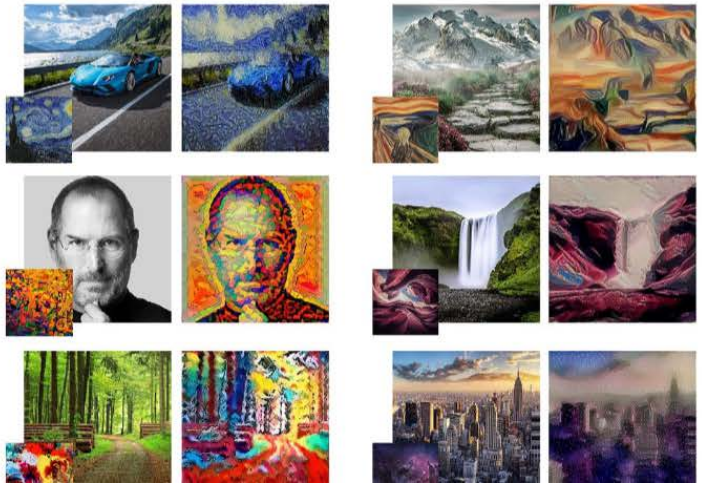
- **Semantic Segmentation**
  - 

  - Source (https://missinglink.ai/guides/computer-vision/image-segmentation-deep-learning-methods-applications/)

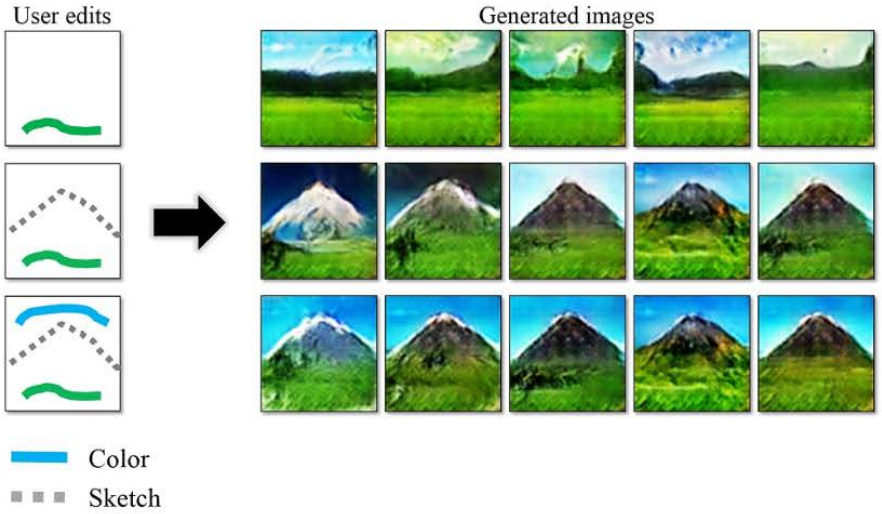
CNNs used for many tasks beyond classification

- **Super Resolution**
  - 

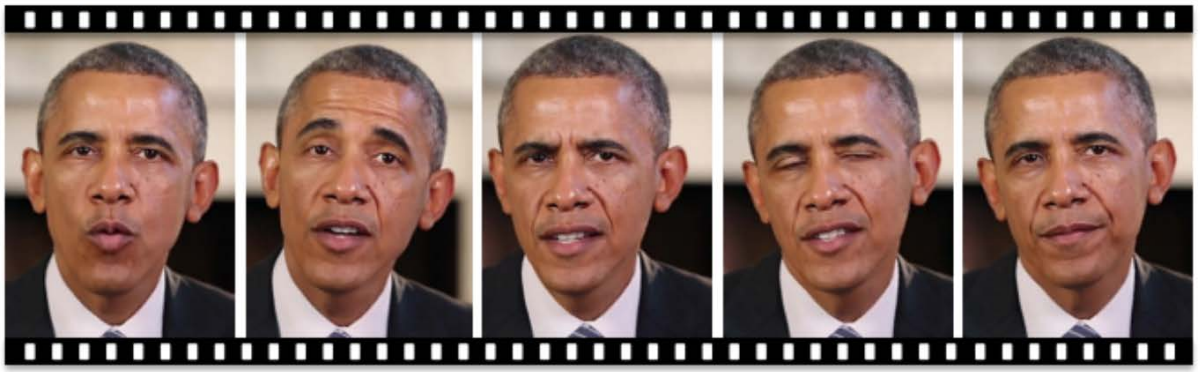
  - Source (https://arxiv.org/pdf/1609.04802.pdf)

- **Style Transfer**
  - 

CNNs used for many tasks beyond classification

- **Image Editing**
  - 

  - Source (http://people.csail.mit.edu/junyanz/projects/gvm/)

- **Multi-Signals**
  - Synthesizing Obama: Learning Lip Sync from Audio
  - 
Output Obama Video
  - Source (http://grail.cs.washington.edu/projects/AudioToObama/)

Reading list

- https://culurciello.github.io/tech/2016/06/04/nets.html
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.
- A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012
- D. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, ICLR 2015
- M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014 (best paper award)
- K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
- M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014
- C. Szegedy et al., Going deeper with convolutions, CVPR 2015
- C. Szegedy et al., Rethinking the inception architecture for computer vision, CVPR 2016
- K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (best paper award)