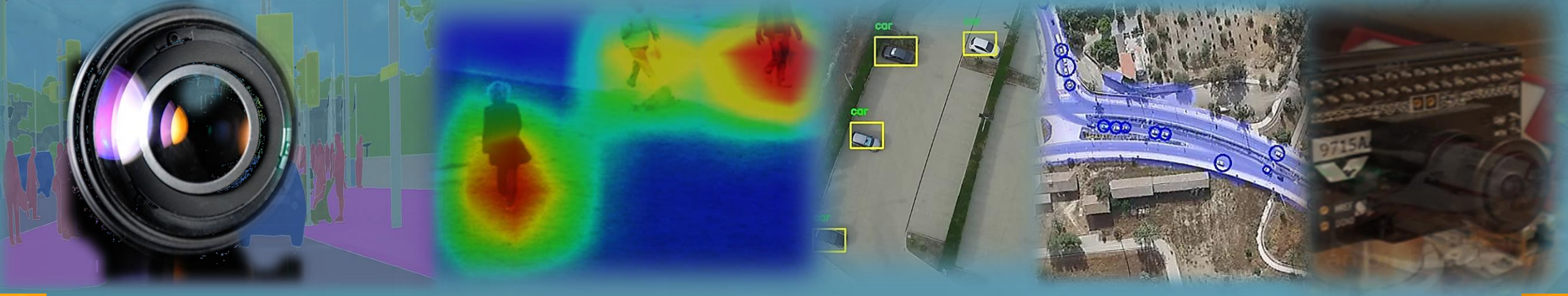


Lecture 13: Classification with Convolutional Neural Networks

Dr Christos Kyrkou

KIOS Research and Innovation Center of Excellence,

Electrical and Computer Engineering Department, University of Cyprus

kyrkou.christos@ucy.ac.cy

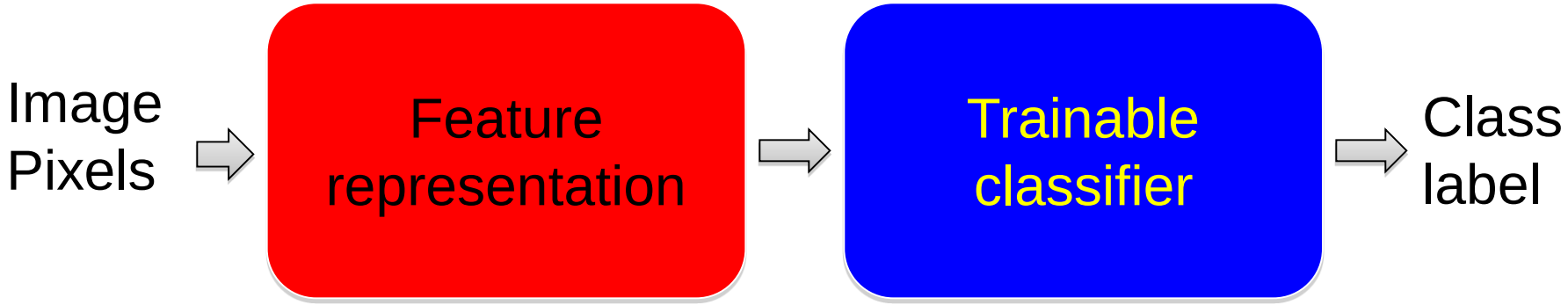
Outline

- Motivation and history
  - From shallow to deep networks
- Convolutional neural networks (CNN)
  - CNN building blocks
- Classical CNN Architectures:
  - 1st generation (2012-2013): AlexNet
  - 2nd generation (2014): VGGNet, GoogLeNet
  - 3rd generation (2015): ResNet
- Training Considerations
- Transfer Learning

- Goodfellow et al. "Deep Learning" (2016),
  – Chapter 9: Convolutional Networks

- http://cs231n.github.io/convolutional-networks/

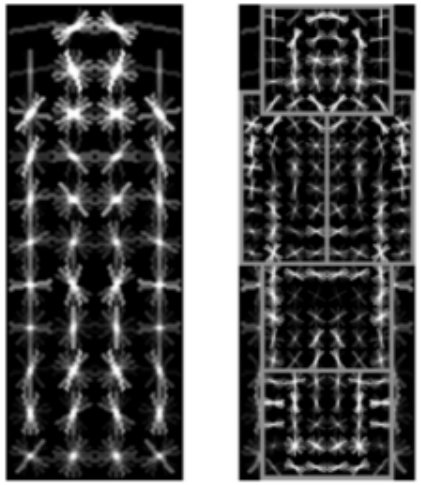
Traditional recognition pipeline



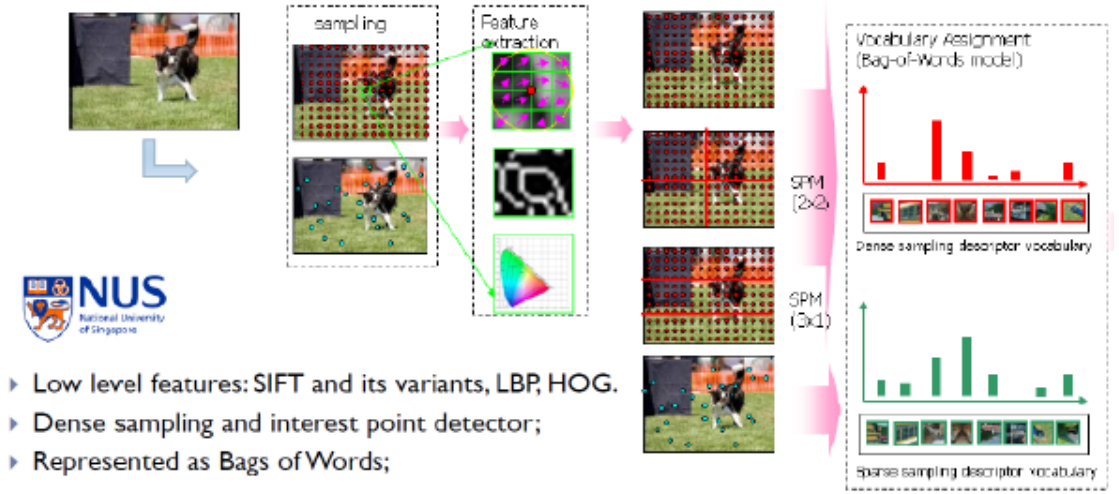
- Hand-crafted feature representation. Features are not learned
- Off-the-shelf trainable classifier, often generic (e.g., SVM)

What's wrong with this?

- Features are key to recent progress in recognition
- Multitude of hand-designed features
  - SIFT, HOG,…
- Where next? Better classifiers? Or Keep building more features?



Felzenszwalb, Girshick, McAllester and Ramanan, PAMI 2007



Yan & Huang
(Winner of PASCAL 2010 classification competition)

“Deep” recognition pipeline

- What are the right features?
  - Why not let the algorithm decide
  - Deep Neural networks: Feature extraction + linear model

- Learn a feature hierarchy from pixels to classifier
- Each layer extracts features from the output of previous layer
- Train all layers jointly

A very brief history…

- Geoffrey Hinton et al. (**2006**) **proposed learning a high-level representation** using successive layers of binary or real-valued latent variables with a restricted Boltzmann machine to model each layer.
- In 2012, Ng and Dean created a network that **learned to recognize higher-level concepts, such as cats, only from watching unlabeled images**. Unsupervised pre-training and increased computing power from GPUs and distributed computing allowed the use of larger networks, particularly in image and visual recognition problems, which became known as "deep learning".
- Ciresan and colleagues (2010) showed that despite the vanishing gradient problem, **GPUs make backpropagation feasible** for many-layered feedforward neural networks.
- **Between 2009 and 2012, ANNs began winning prizes** in ANN contests, approaching human level performance on various tasks, initially in pattern recognition and machine learning.

Deep Learning Breakthroughs. What Changed!

- Compute – Moore's Law
  - CPUs, GPUs, ASICs
- Organized large datasets:
  - Imagenet, COCO
- Algorithms and research:
  - Backprop, CNN, LSTM
- Software and Infrastructure:
  - Git, ROS, PR2, AWS, Amazon Mechanical Turk, TensorFlow, …
- Financial backing of large companies
  - Google, Facebook, Amazon, …

- But nothing has really changed!!!!

Neural networks vs. SVMs (a.k.a. “deep” vs. “shallow” learning)

IMAGENET Large Scale Visual Recognition Challenge (ILSVRC)

Date: June 2, 2015

Dear ILSVRC community,

This is a follow up to the announcement on May 19, 2015 with some more details and the status of the test server.

During the period of November 28th, 2014 to May 13th, 2015, there were at least 30 accounts used by a team from Baidu to submit to the test server at least 200 times, far exceeding the specified limit of two submissions per week. This includes short periods of very high usage, for example with more than 40 submissions over 5 days from March 15th, 2015 to March 19th, 2015. Figure A below shows submissions from ImageNet accounts known to be associated with the team in question. Figure B shows a comparison to the activity from all other accounts.

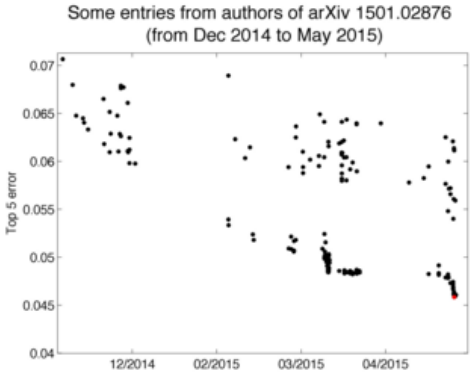


Figure A

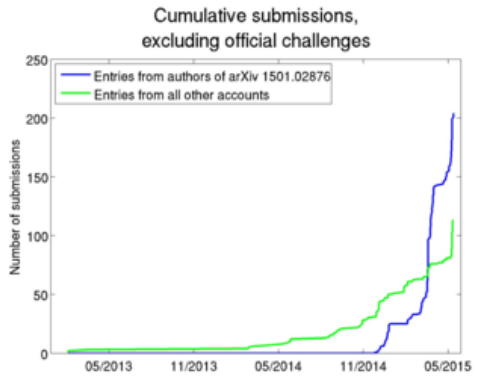
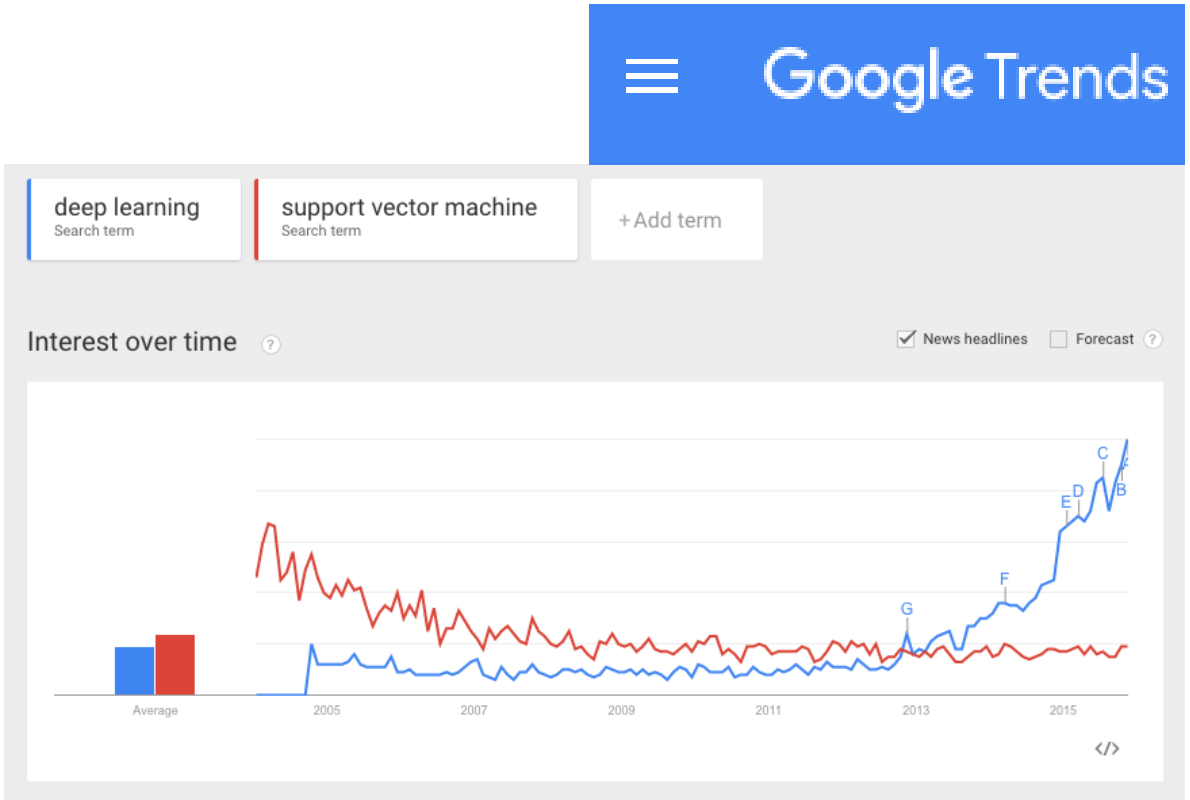


Figure B

The results obtained during this period are reported in a recent arXiv paper. Because of the violation of the regulations of the test server, these results may not be directly comparable to results obtained and reported by other teams. To make this clear, by exploiting the ability to test many slightly different solutions on the test server it is possible to 1) select the best out of a set of very similar solutions based on test performance and achieve a small but potentially significant advantage and 2) choose methods for further research and development based directly on the test data instead of using only the training and validation data for such choices.

http://www.image-net.org/challenges/LSVRC/announcement-June-2-2015



Some thoughts…

So, 1. **what exactly is deep learning** ?

And, 2. **why is it generally better** than other methods on image, speech and certain other types of data?

Some thoughts…

So, 1. **what exactly is deep learning** ?

And, 2. **why is it generally better** than other methods on image, speech and certain other types of data?

The short answers

1. **'Deep Learning' means** using a **neural network** with **several layers of nodes** between input and output

2. **the series of layers between input & output do feature identification and processing in a series of stages, just as our brains seem to.**

Some thoughts…

hmmm… OK, but:

3. **multilayer neural networks have been around for 25 years.**

What's actually new?

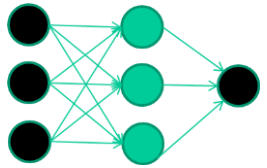
Some thoughts…

hmmm… OK, but:

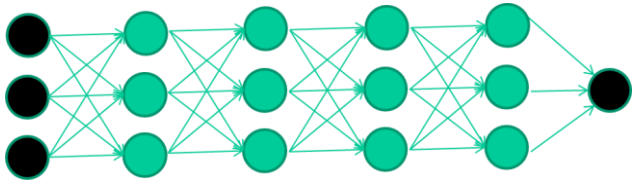
3. **multilayer neural networks have been around for 25 years.**

What's actually new?

we have always had good algorithms for learning the weights in networks with 1 hidden layer

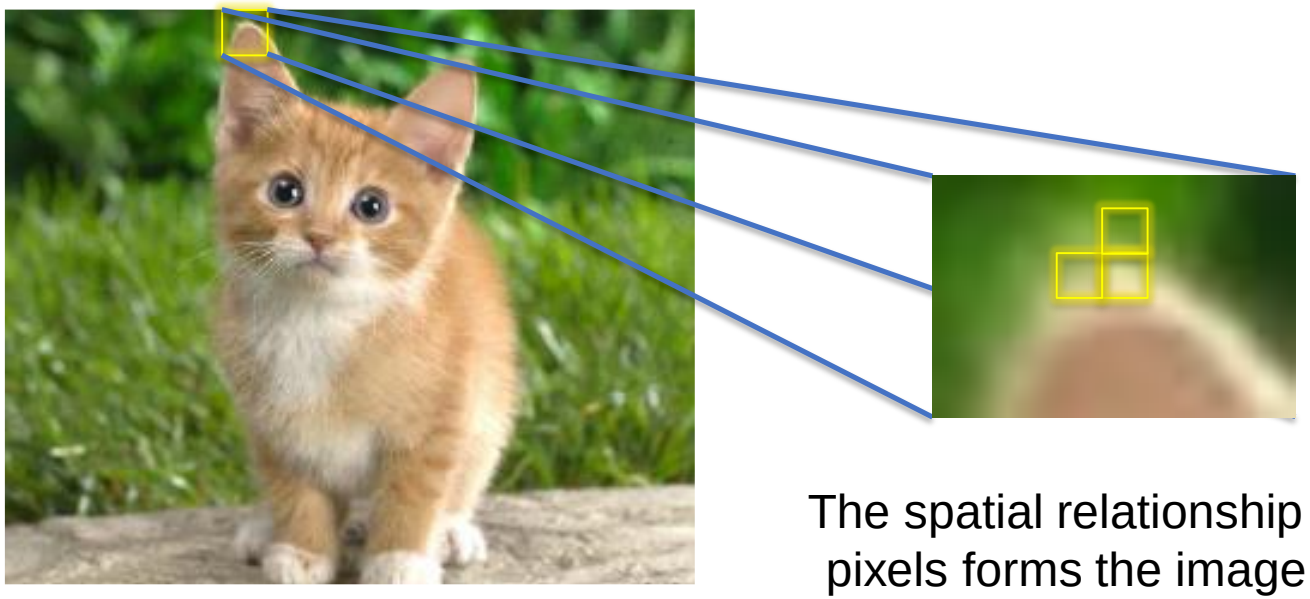


but these algorithms are not good at learning the weights for networks with more hidden layers

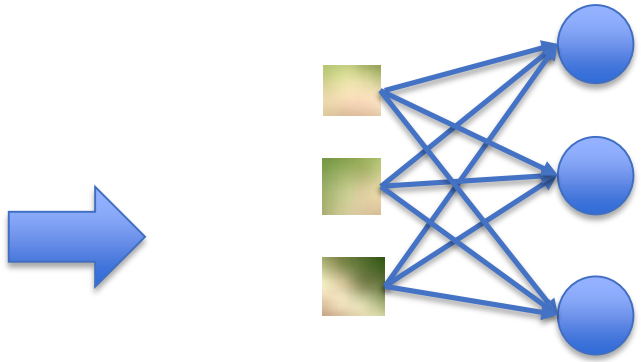


what's new is: algorithms for training many-layer networks

4. What is wrong with dense layers for images?



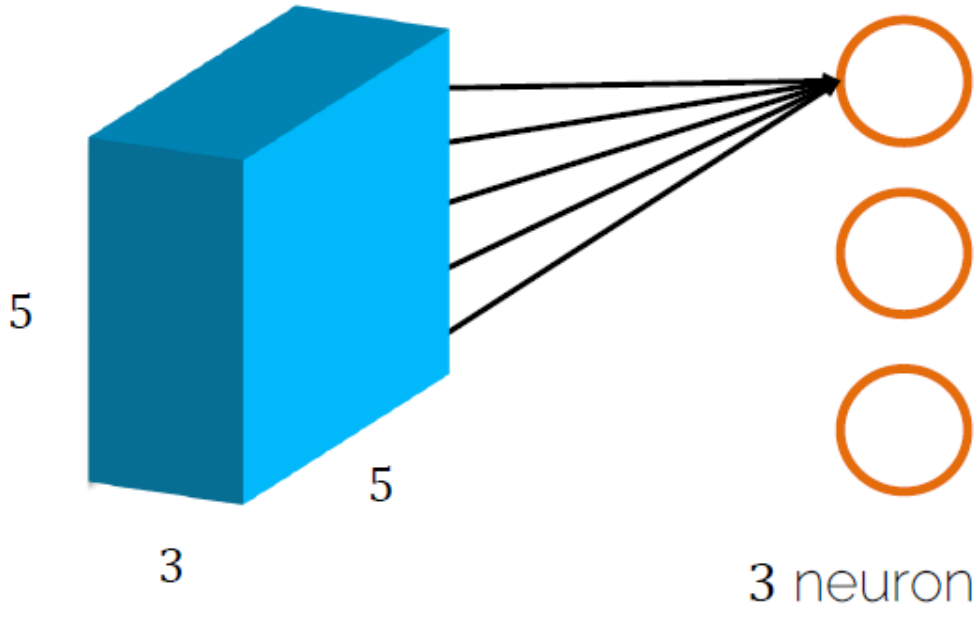
The spatial relationship between pixels forms the image content

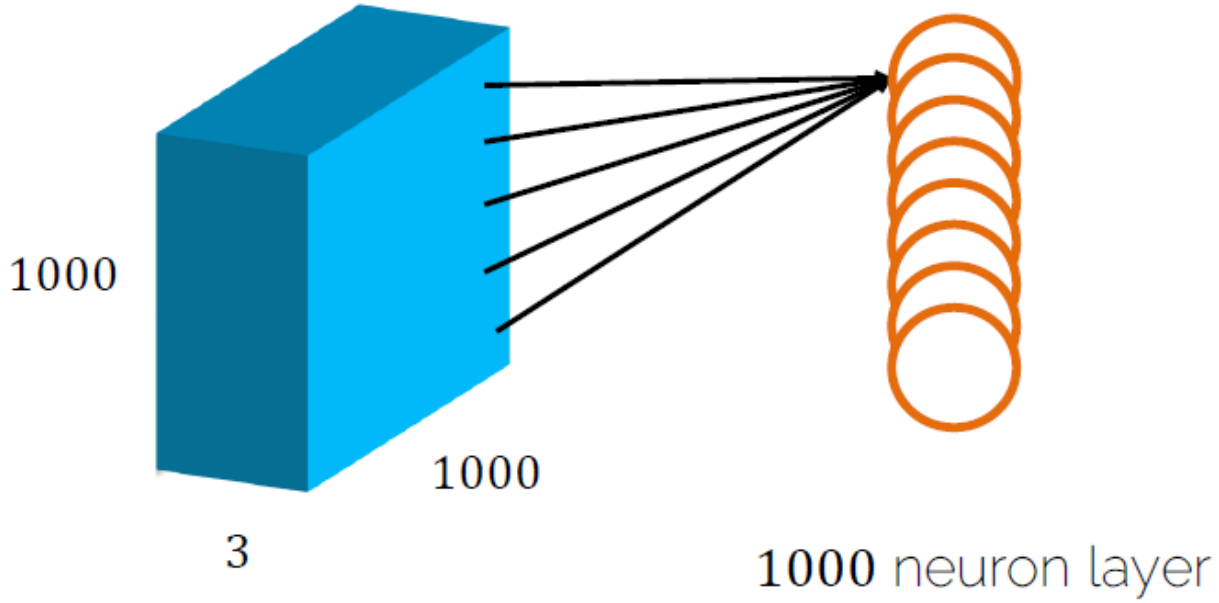


Dense layers (in general) do not make any assumptions regarding the structure of the images

- But images have structure:
  - We know that neighbouring pixels are correlated and distant ones not so much.
  - Also image features are repeatable, if we detect a line in one region, it might be useful to detect it in another region as well.

4. What is wrong with dense layers for images?





- 75 weights for the whole 5×5 image on the 3 channels
- 3×75 for all three neurons→ 225

- $3 \times 10^9 \rightarrow 3$ billion weights!
- **IMPRACTICAL!**

4. What is wrong with dense layers for images?

- We cannot make networks arbitrary complex

- Why don't just go deeper to get better?
  - No Structure!!
  - It is just brute force.
  - Optimization becomes hard
  - Performance plateaus/drops!

- We want to restrict the degrees of freedom
  - **We want a layer with structure**
  - **Weight sharing** → using the same weights for the same image

Spoiler ahead!

- Convolution operation with **filters** searches for the **same pattern everywhere** in the image
- It uses a filter of **with the same weights** to detect the same pattern across the image
- **Different filters** search for **different features**
- **Learn these Filters!**

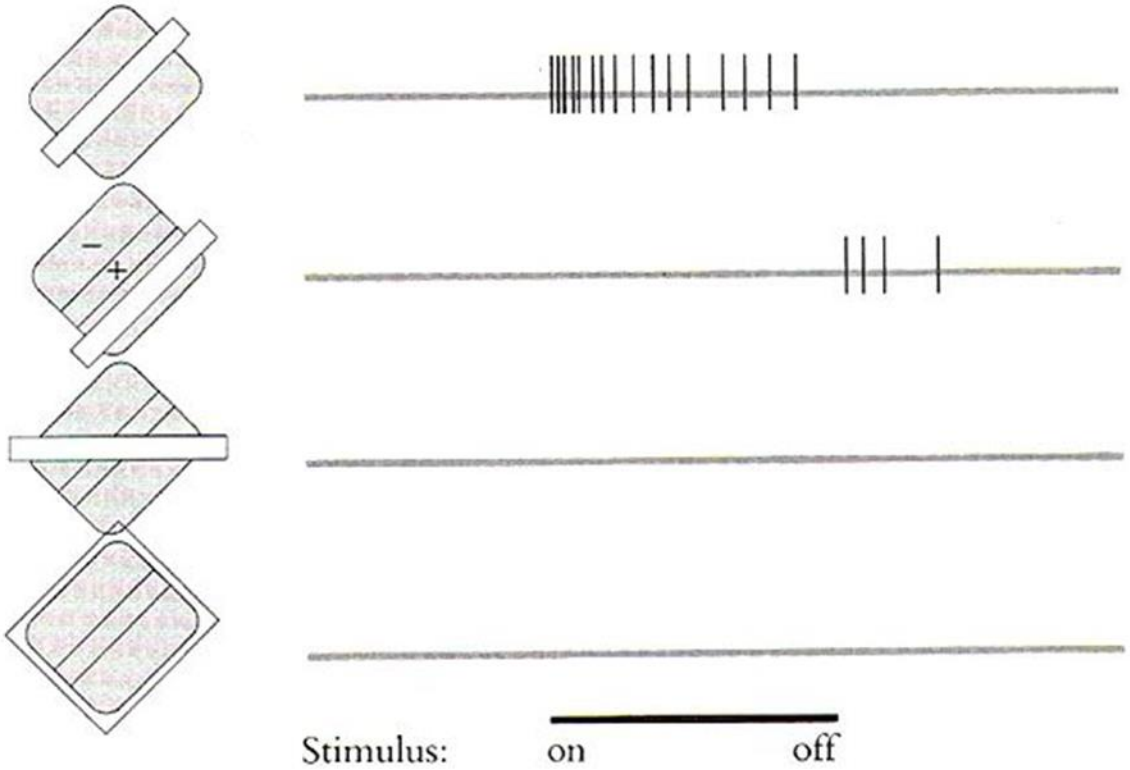
Input

Edge detection

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

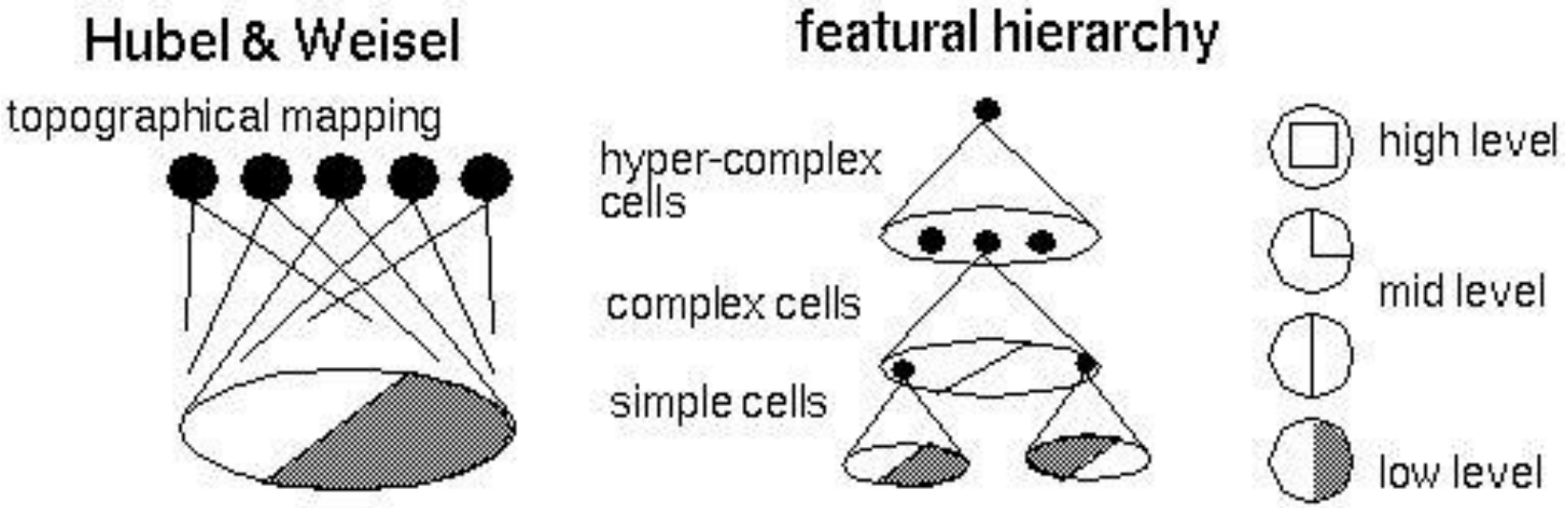
Hubel and Weisel (1962)

- Discovered orientation sensitive neurons in V1



Inspiration: Biological visual system

- D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981)
  - Visual cortex consists of a hierarchy of simple, complex, and hyper-complex cells

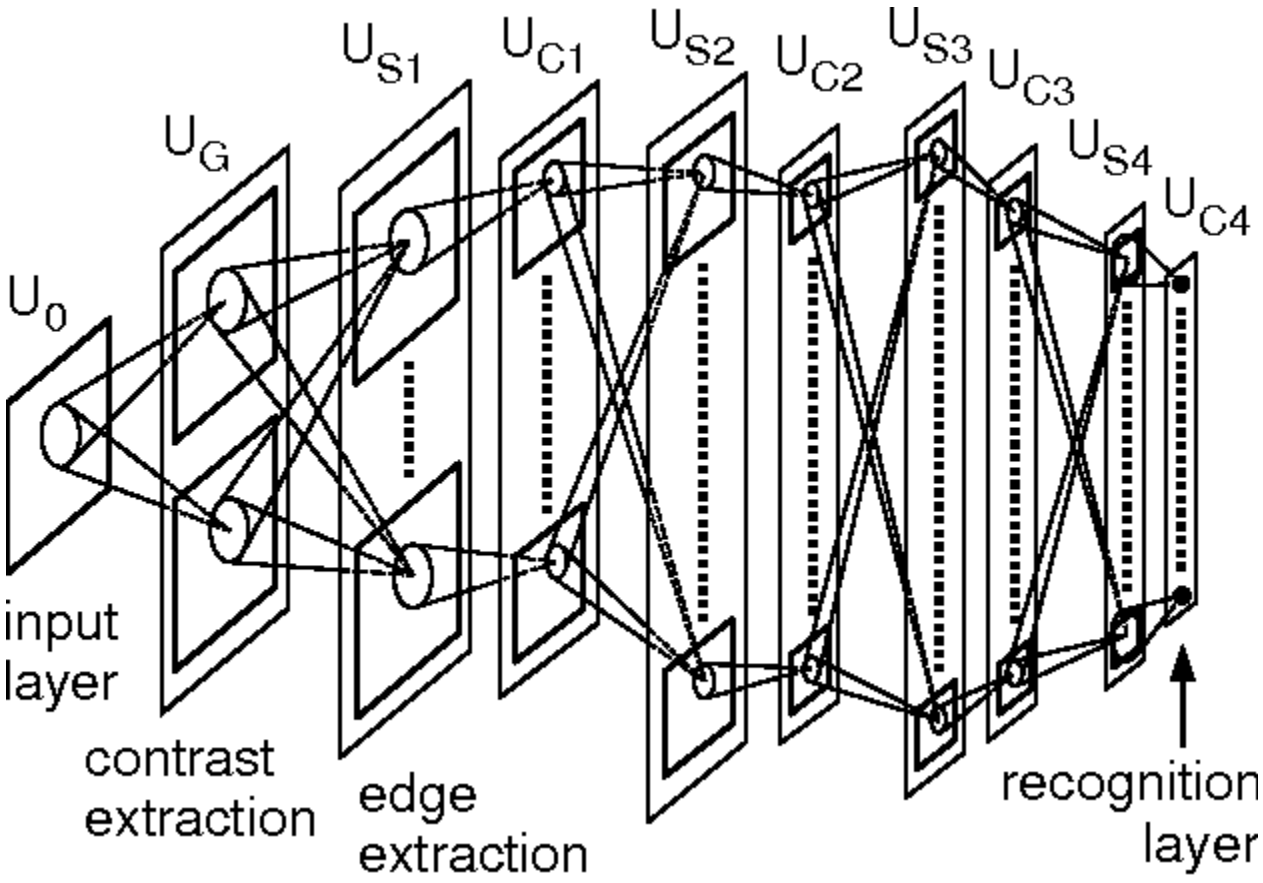


Source

History: Neocognitron

- First network architecture
- Took inspiration from Hubel and Wiesel's Neuroscience research and designed two types of neurons:
  - simple cells: contains weights that need to be learned
  - complex cells: pools activations from previous layers as a way of modeling the hierarchical nature of our visual system

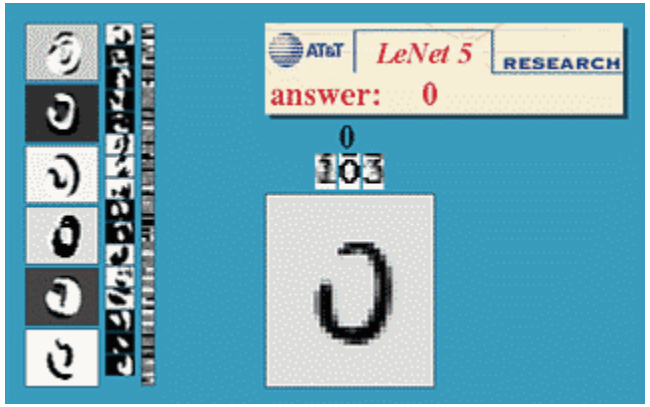
K. Fukushima, 1980s

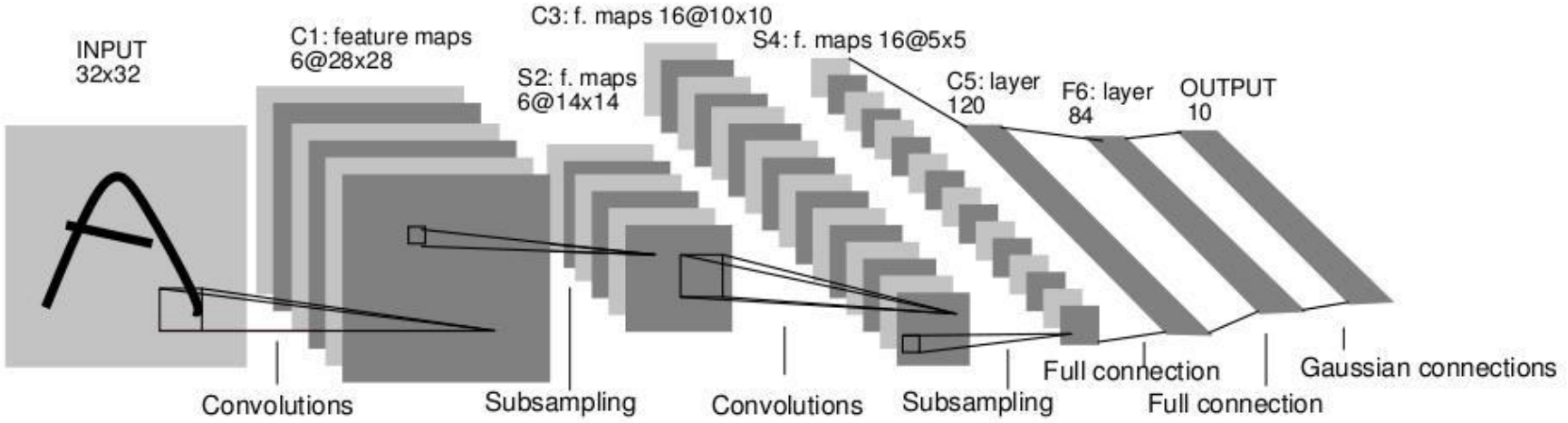


https://en.wikipedia.org/wiki/Neocognitron

History: LeNet-5 (Yann LeCun et al)

- First demonstrated by LeCun et al for handwritten digit recognition (1998)
- Convolutional Neural Networks (CNN): Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
  - Added learning to Neocognitron
  - Higher stages compute more global, more invariant, more abstract features
  - Classification layer at the end
- Trained on MNIST digit dataset with 60K training examples

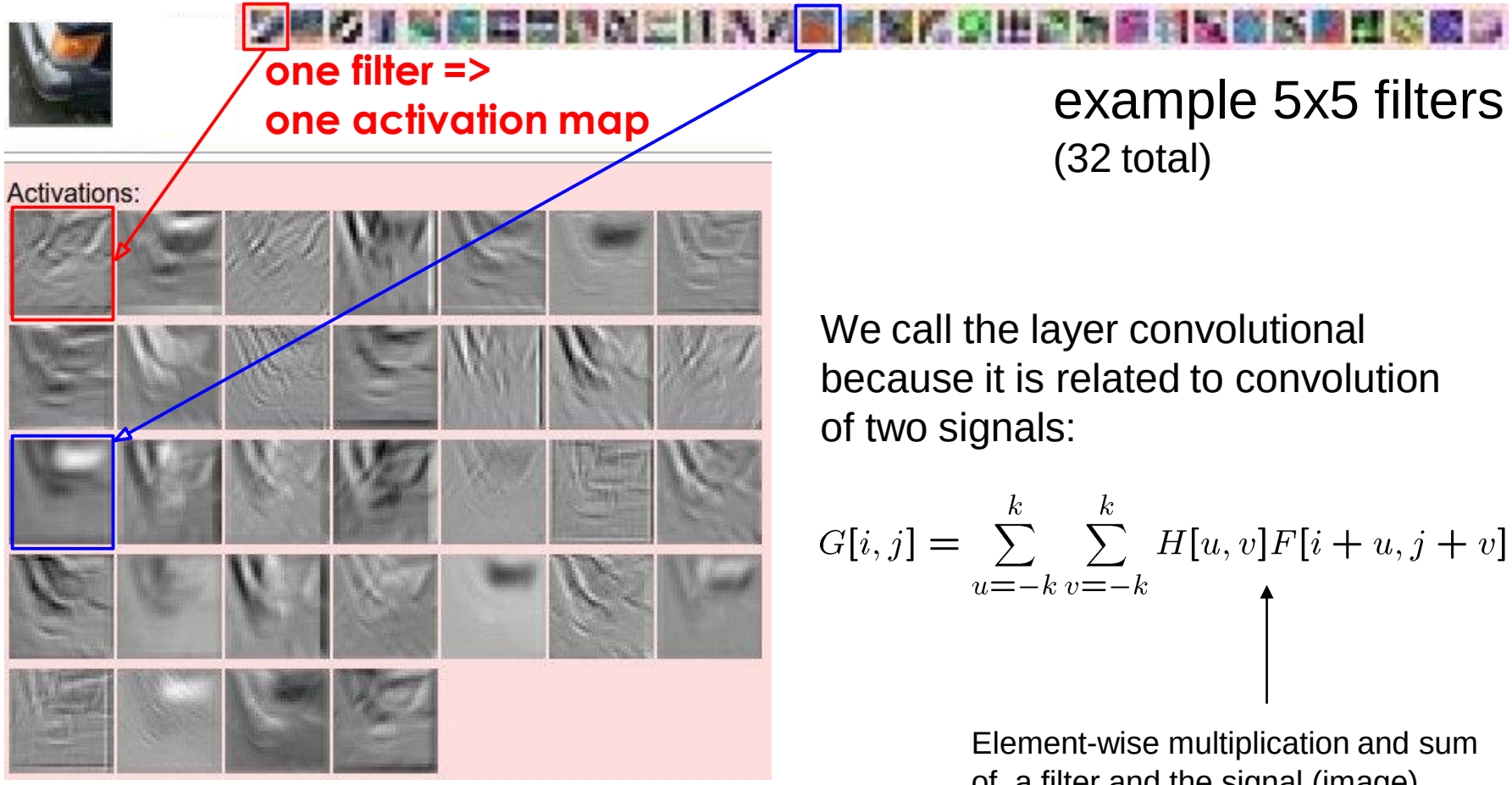




Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Adapted from Rob Fergus

Convolution Layer



example 5x5 filters
(32 total)

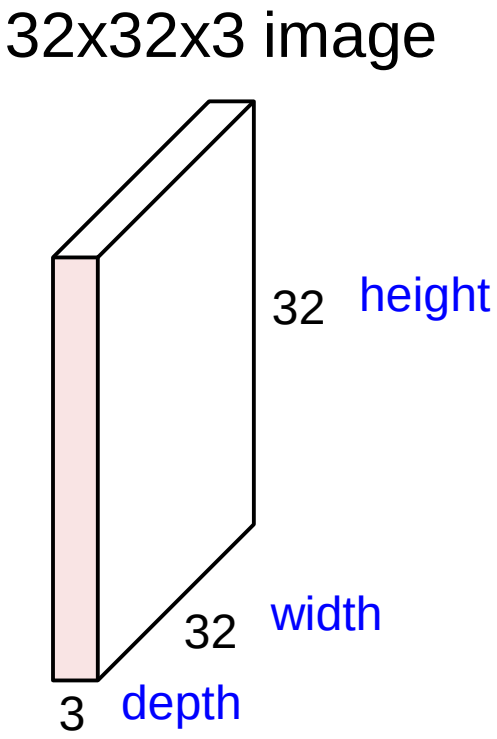
We call the layer convolutional because it is related to convolution of two signals:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

Element-wise multiplication and sum of a filter and the signal (image)

Adapted from Andrej Karpathy, Kristen Grauman

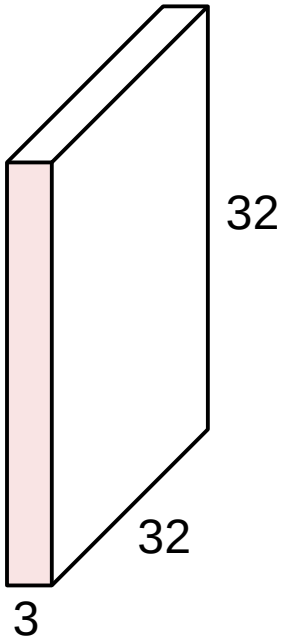
Convolution Layer



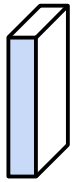
Credit: Andrej Karpathy

Convolution Layer

32x32x3 image



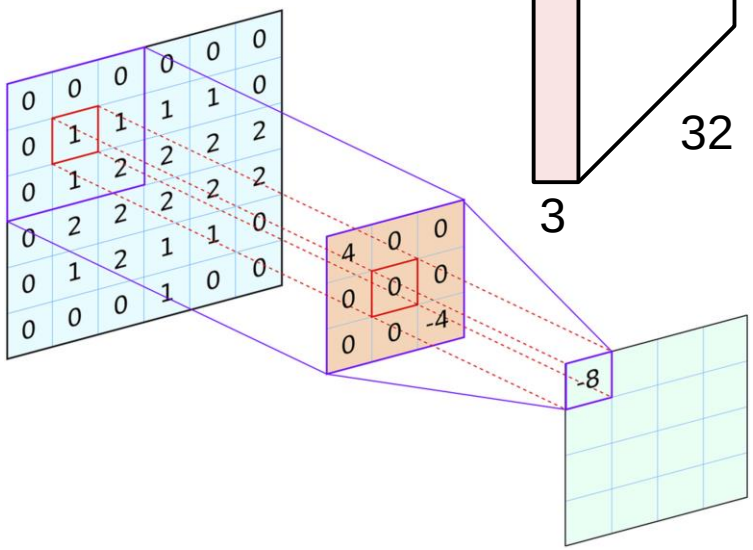
5x5x3 filter

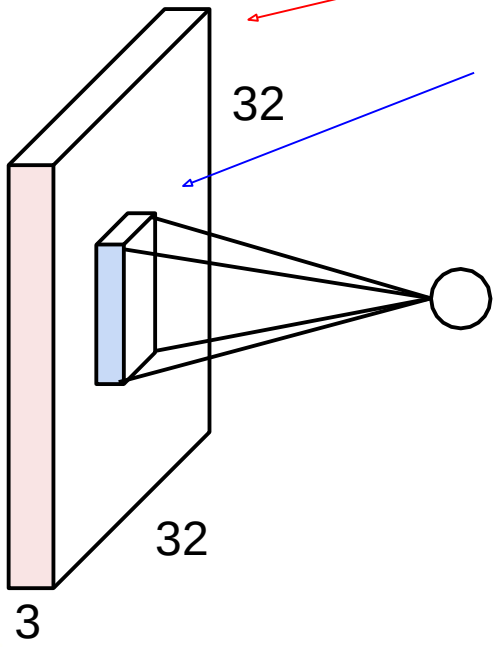


Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

Credit: Andrej Karpathy

Convolution Layer





32x32x3 image

5x5x3 filter w

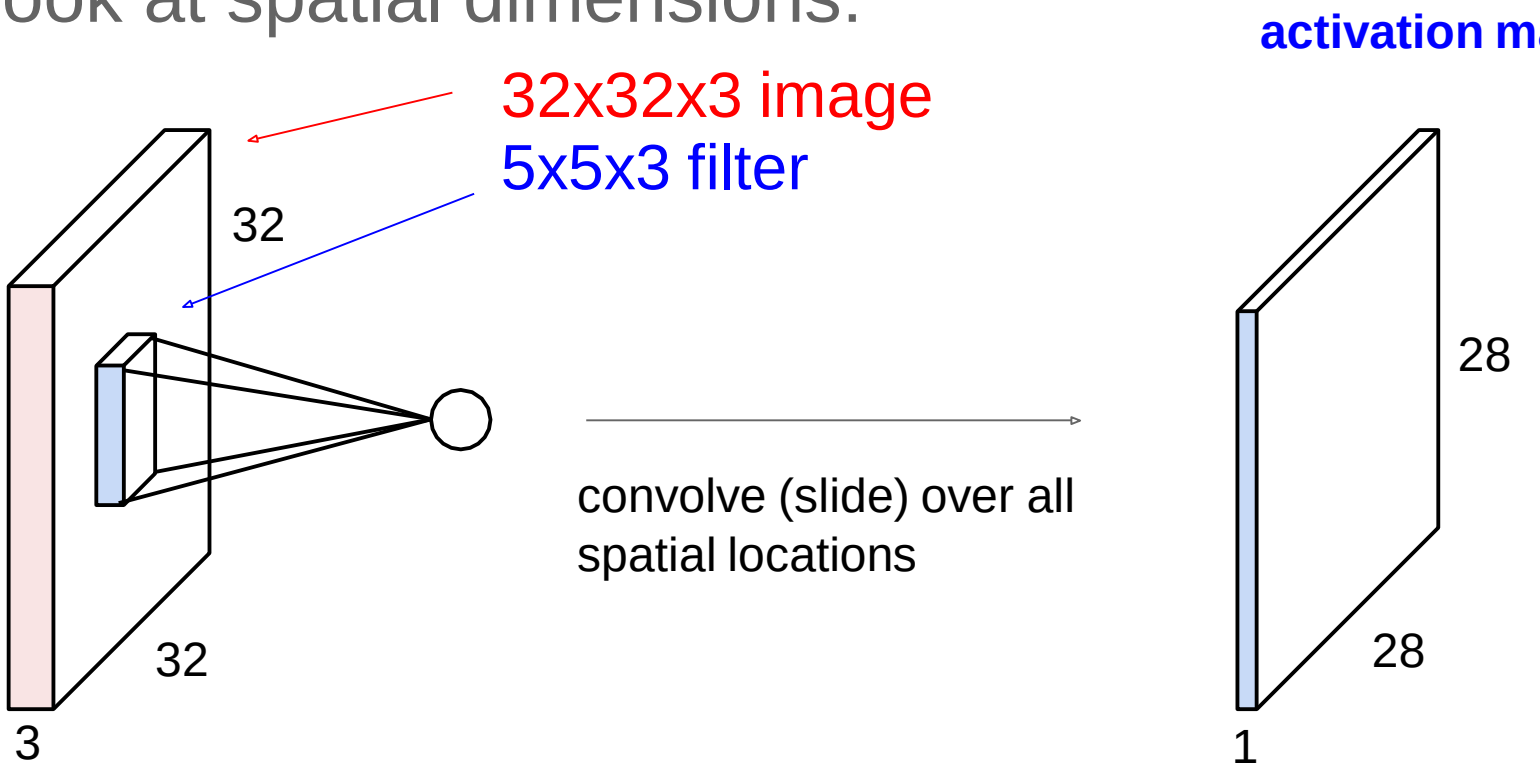
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Credit: Andrej Karpathy

Convolution Layer

- A closer look at spatial dimensions:



Credit: Andrej Karpathy

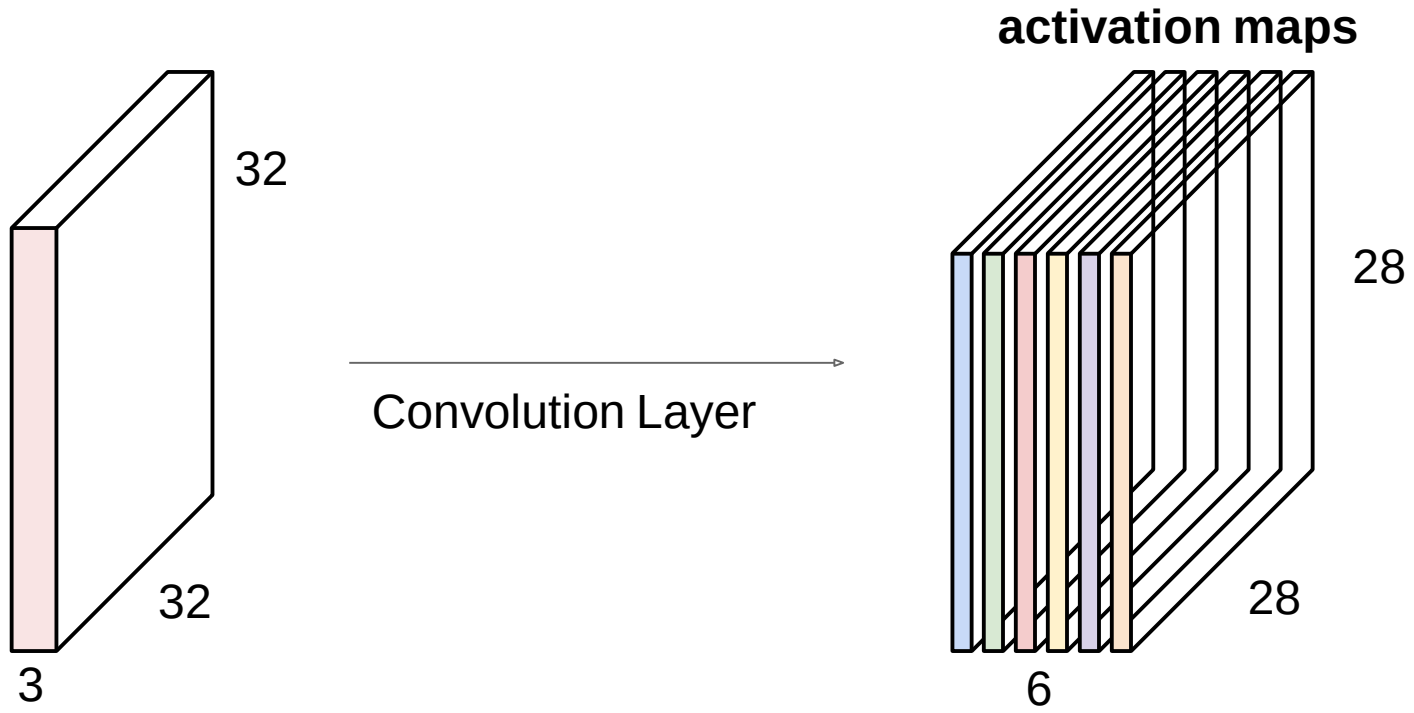
Convolution Layer

- Consider a second, green filter

Credit: Andrej Karpathy

Convolution Layer

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

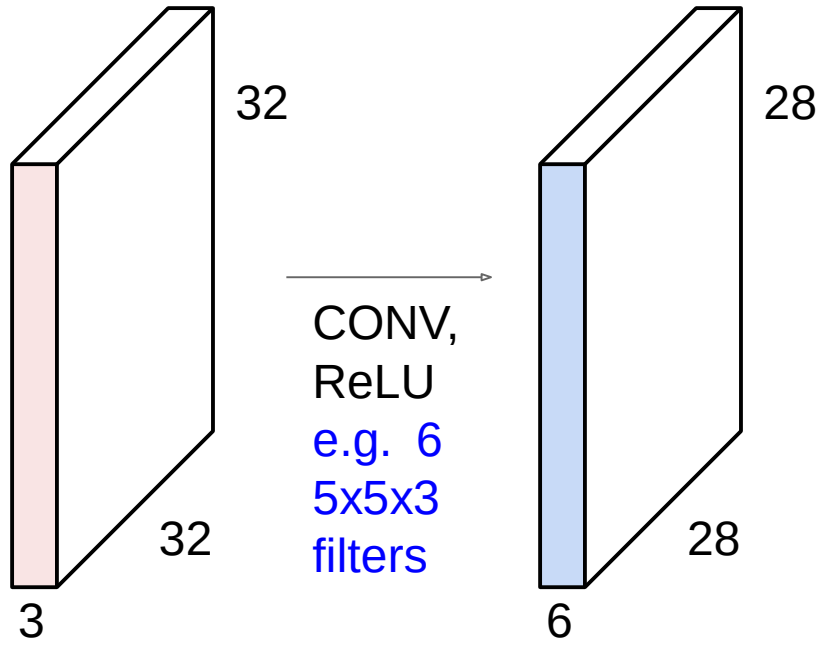


We stack these up to get a "new image" of size 28x28x6!

Credit: Andrej Karpathy

Convolution Layer

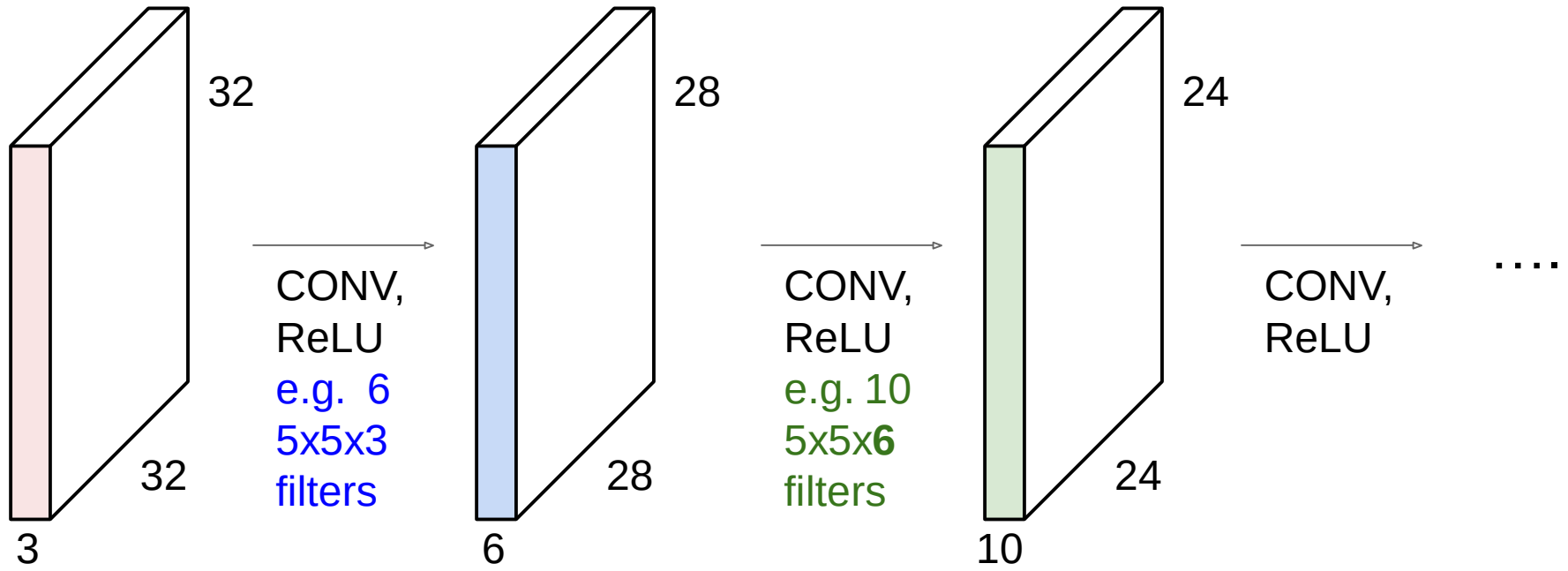
- **Preview**: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Credit: Andrej Karpathy

Convolution Layer

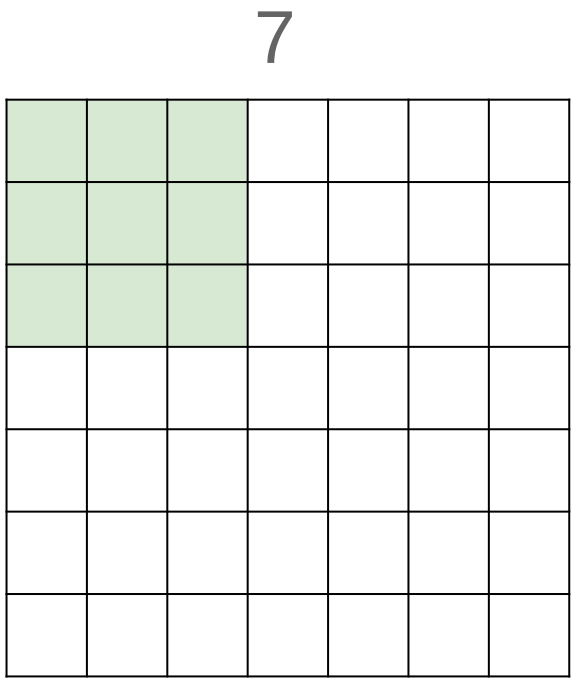
- **Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



Credit: Andrej Karpathy

Convolution Layer: A closer look

- A closer look at spatial dimensions:

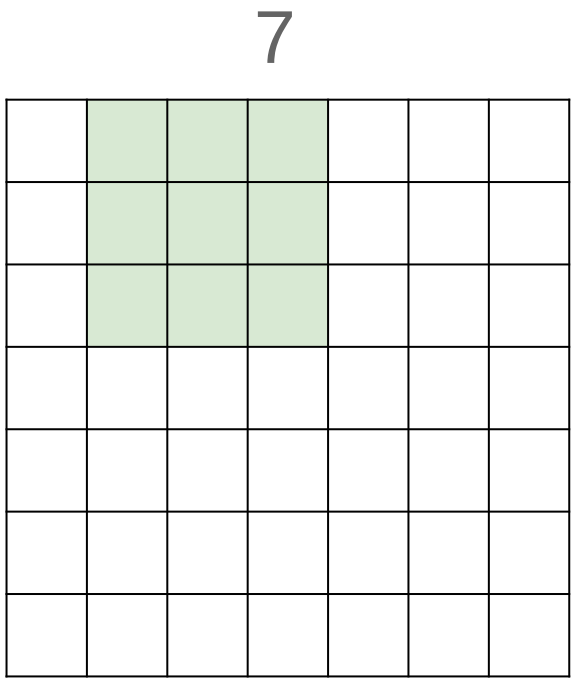


- 7x7 input (spatially) assume 3x3 filter

Credit: Andrej Karpathy

Convolution Layer: A closer look

- A closer look at spatial dimensions:

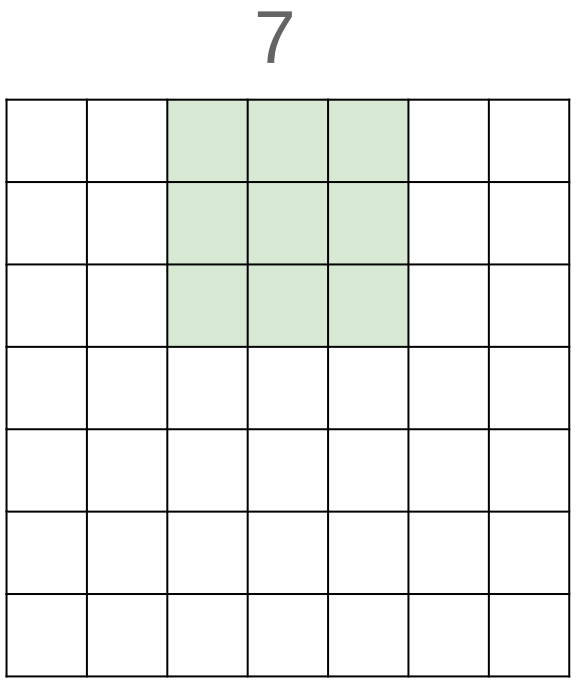


- 7x7 input (spatially) assume 3x3 filter

Credit: Andrej Karpathy

Convolution Layer: A closer look

- A closer look at spatial dimensions:

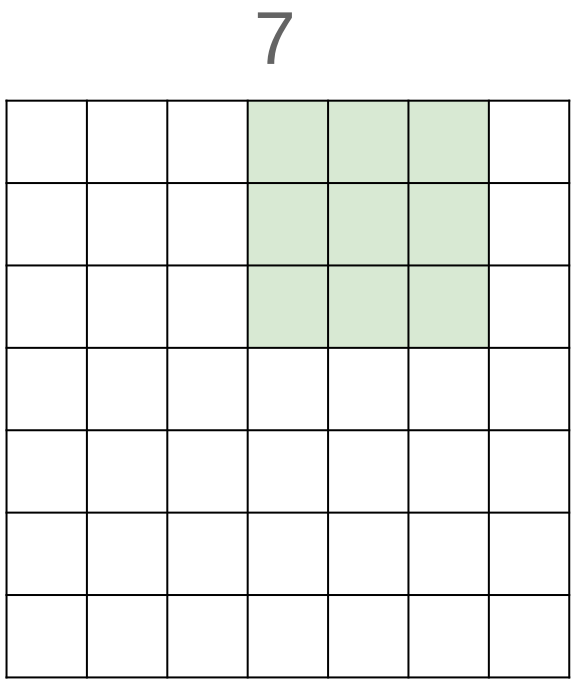


- 7x7 input (spatially) assume 3x3 filter

Credit: Andrej Karpathy

Convolution Layer: A closer look

- A closer look at spatial dimensions:

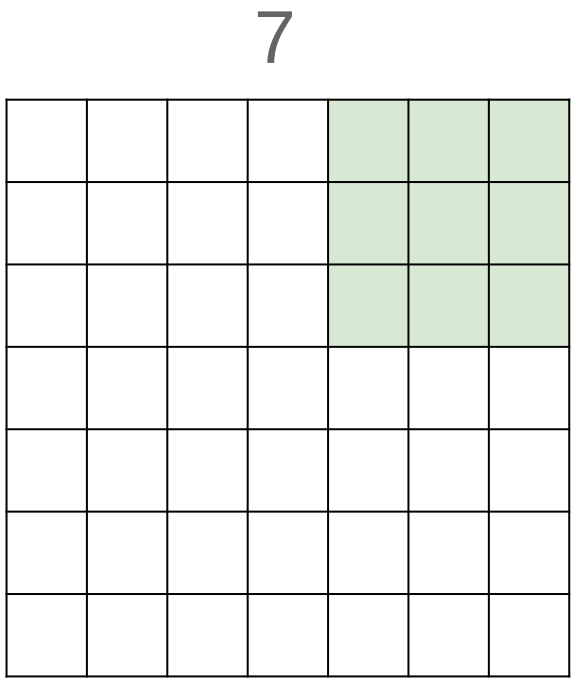


- 7x7 input (spatially) assume 3x3 filter

Credit: Andrej Karpathy

Convolution Layer

- A closer look at spatial dimensions:



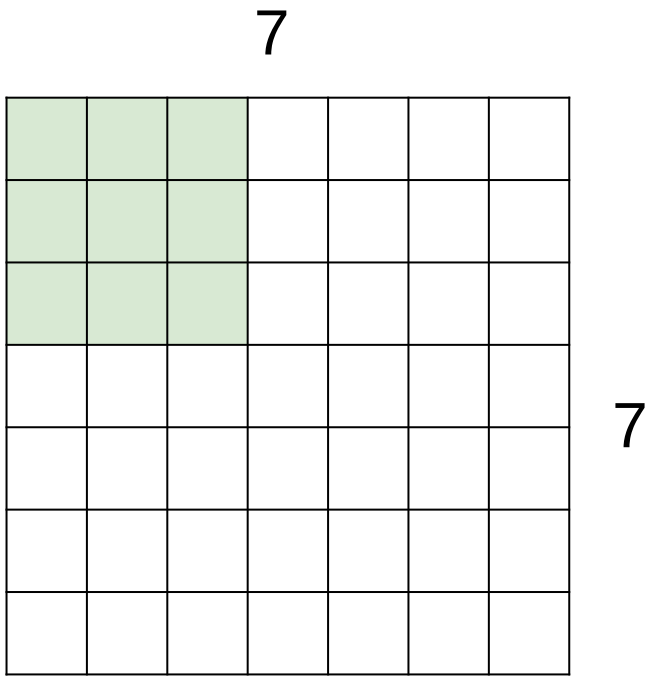
- 7x7 input (spatially) assume 3x3 filter

=> 5x5 output

Credit: Andrej Karpathy

Convolution Layer

- A closer look at spatial dimensions:

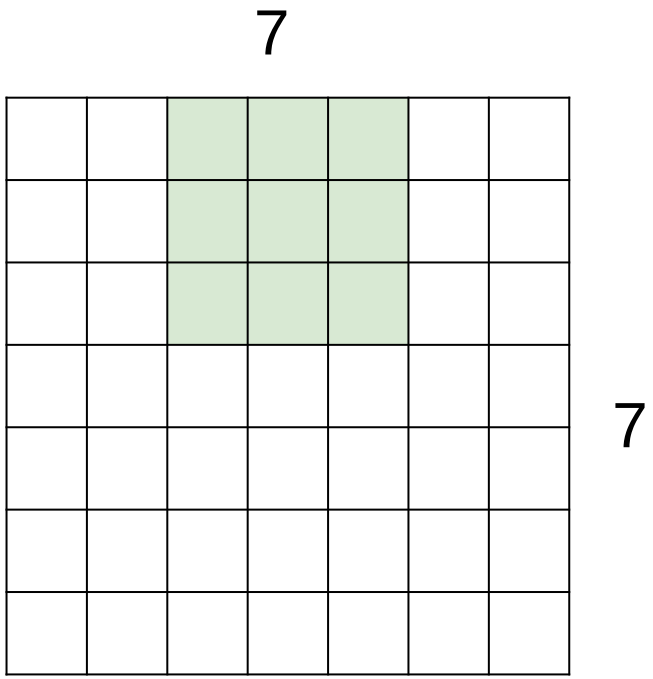


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Credit: Andrej Karpathy

Convolution Layer

- A closer look at spatial dimensions:

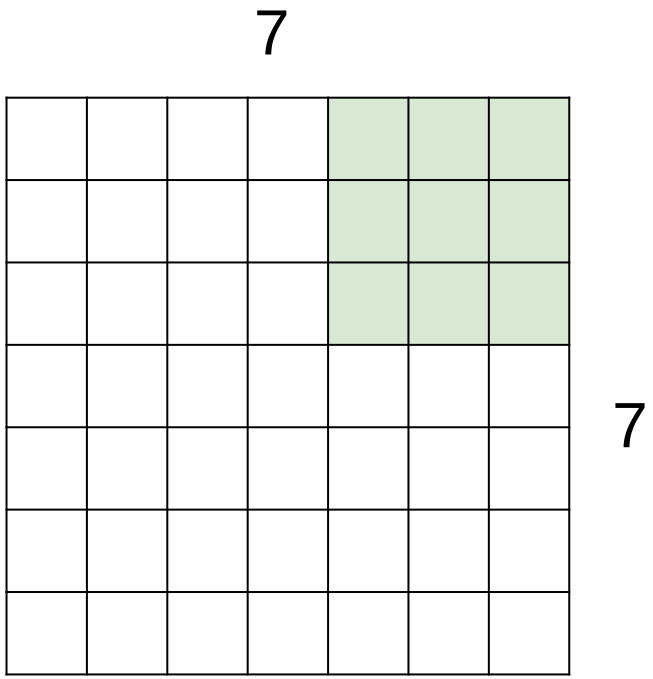


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Credit: Andrej Karpathy

Convolution Layer

- A closer look at spatial dimensions:



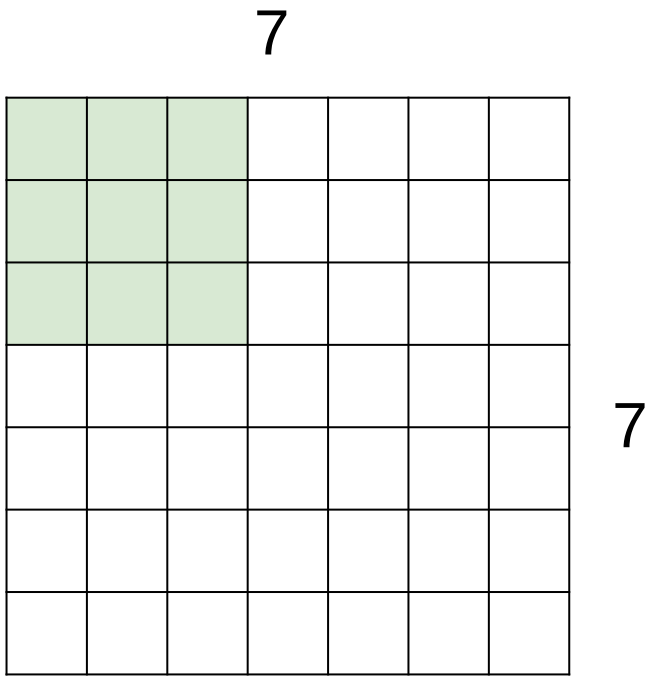
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

A stride higher than 1 significantly reduces the spatial resolution.

Credit: Andrej Karpathy

Convolution Layer

- A closer look at spatial dimensions:

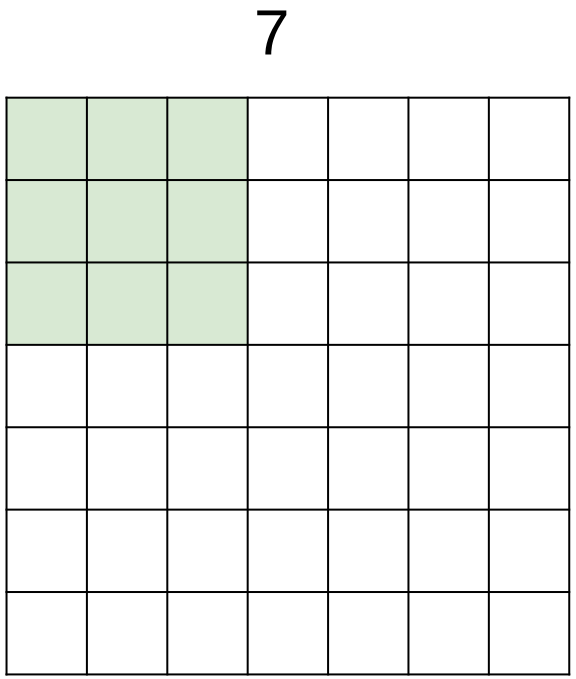


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Credit: Andrej Karpathy

Convolution Layer

- A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

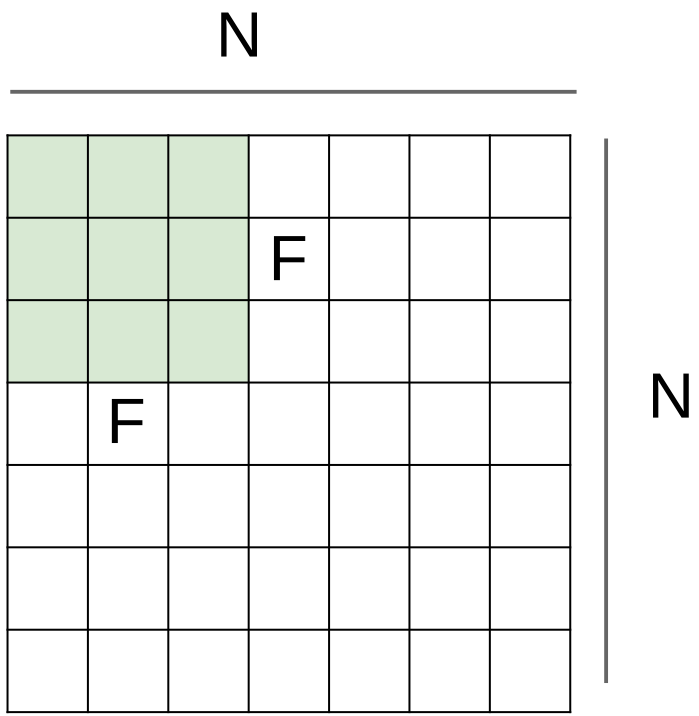
doesn't fit!

cannot apply 3x3 filter on 7x7 input with stride 3.

Credit: Andrej Karpathy

Convolution Layer

- A closer look at spatial dimensions:



Output size:
(N - F) / stride + 1

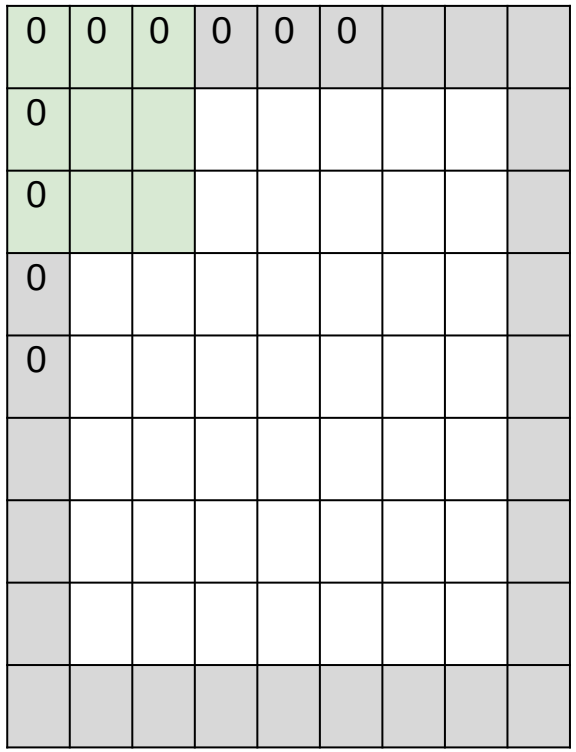
e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

- **Stride**: The number of pixels that the filter is shifted by.

Credit: Andrej Karpathy

Convolution Layer

- In practice: Common to zero pad the border



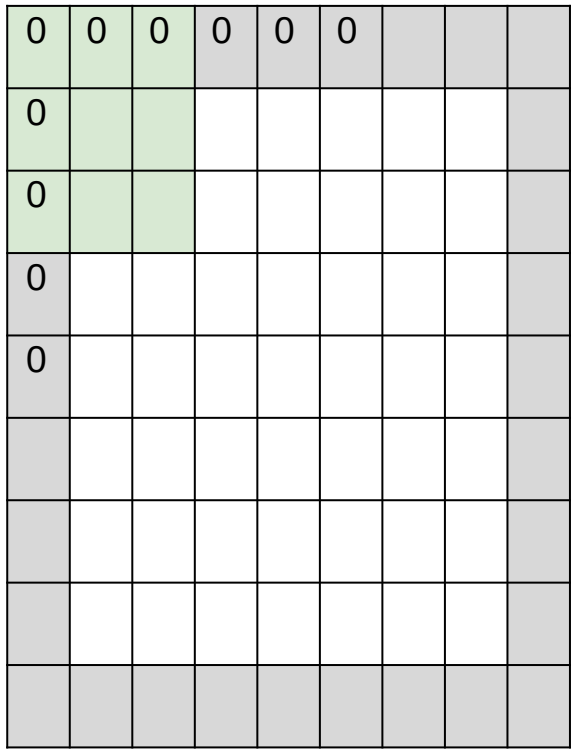
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

Credit: Andrej Karpathy

Convolution Layer

- In practice: Common to zero pad the border



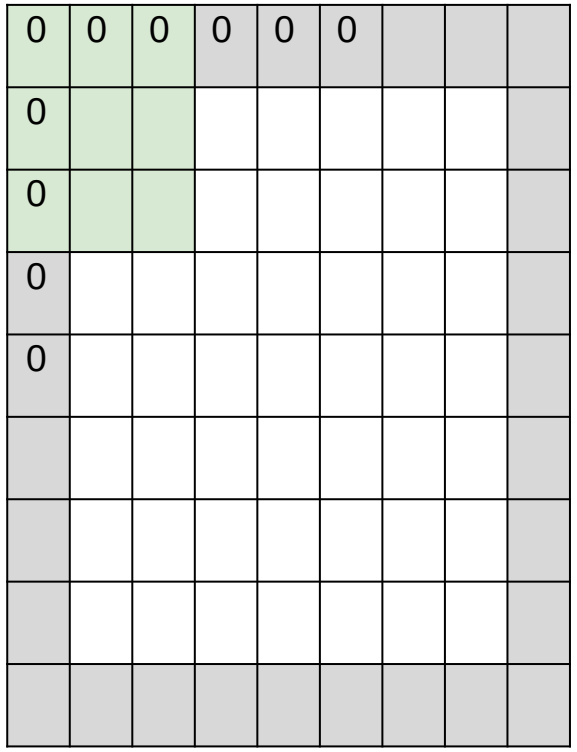
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

Credit: Andrej Karpathy

Convolution Layer

- In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

(N + 2*padding - F) / stride + 1

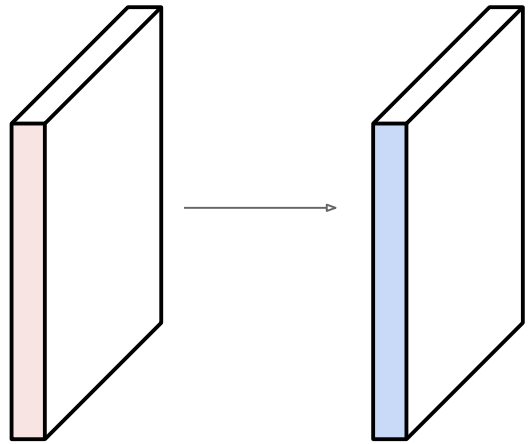
Credit: Andrej Karpathy

Convolutions: More Details

- Examples time:

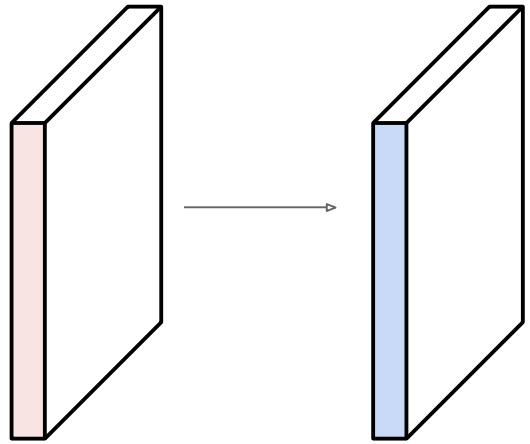
- Input volume: **32x32x3**
- 10 5x5 filters with stride 1, pad 2

- Output volume size: ?



Convolutions: More Details

- Examples time:



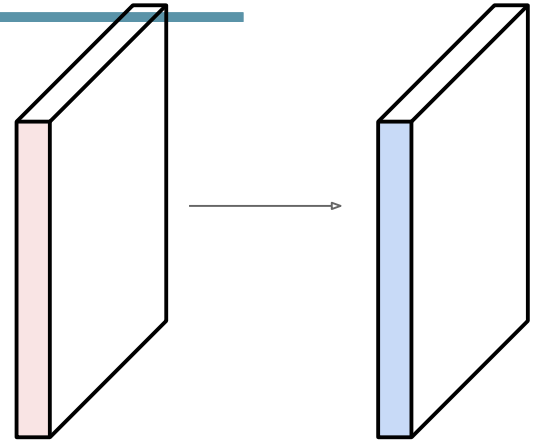
- Input volume: **32x32x3**
- 10 5x5 filters with stride 1, pad 2

- Output volume size:
- (32+2*2-5)/1+1 = 32 spatially, so
- **32x32x10**

Credit: Andrej Karpathy

Convolutions: More Details

- Examples time:



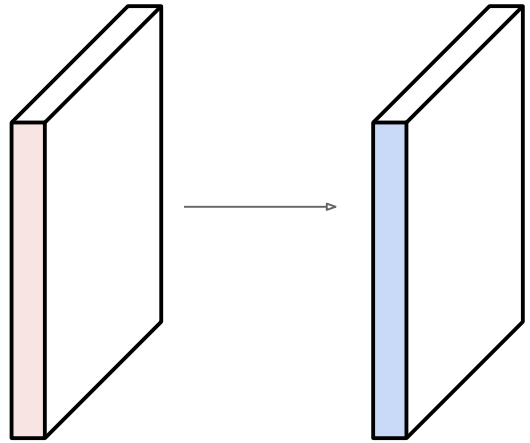
- Input volume: **32x32x3**
- 10 5x5 filters with stride 1, pad 2

- Number of parameters in this layer?

Credit: Andrej Karpathy

Convolutions: More Details

- Examples time:
- Input volume: **32x32x3**
- 10 5x5 filters with stride 1, pad 2

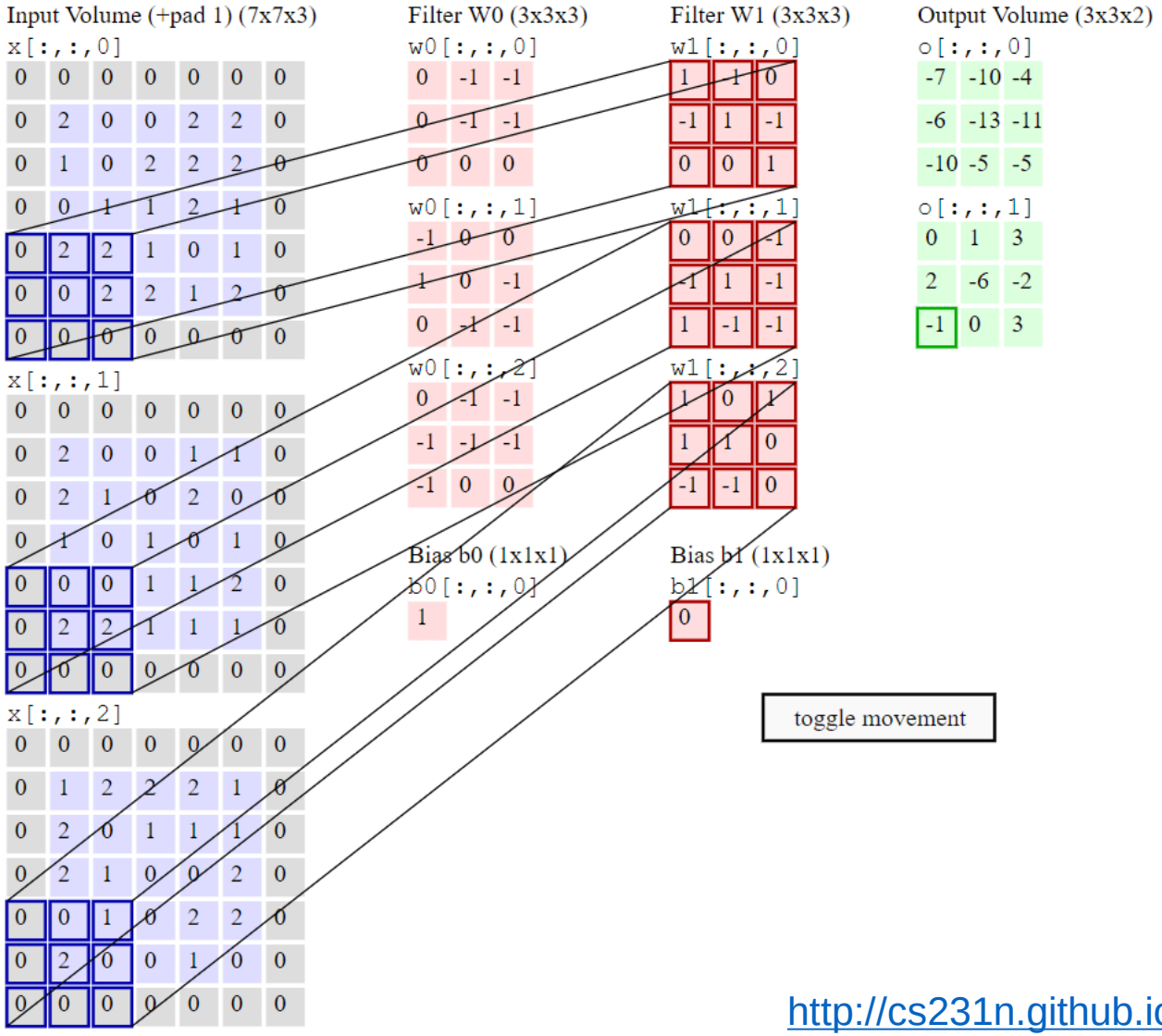


- Each filter has 5*5*3 + 1 = 76 params
- => 76*10 = **760** (+1 for bias)

- Considering the example with image 1000×1000 RGB image and 1000 neurons:
  - Only 76,000 parameters in the layer!→ Massive reduction!

Credit: Andrej Karpathy

Convolutional layer Visualization

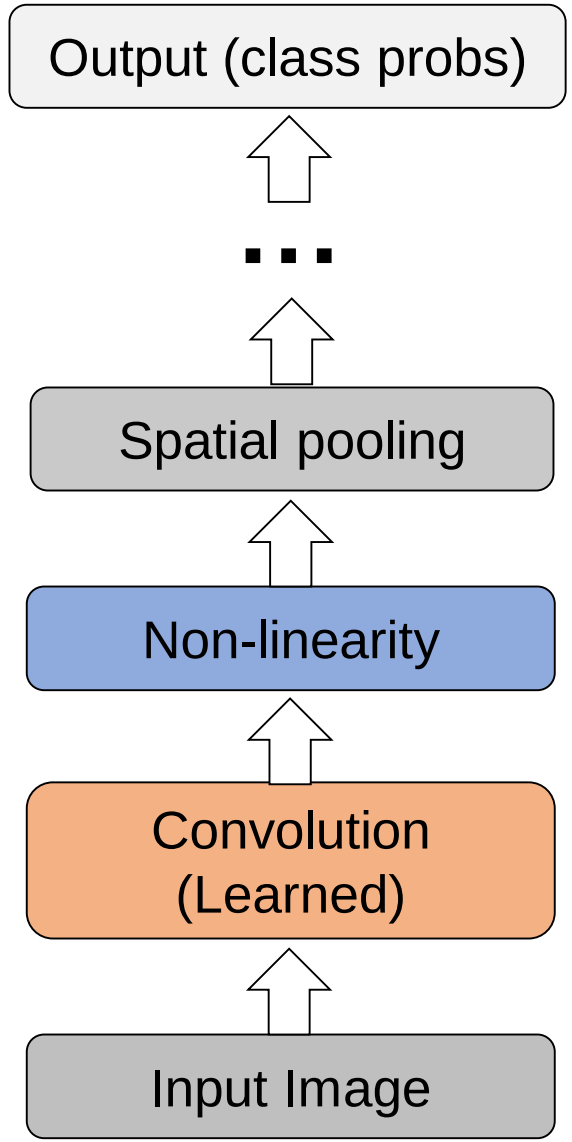


http://cs231n.github.io/convolutional-networks/#conv

Convolutional Neural Networks (CNN)

- Feed-forward feature extraction:
  - Convolve input with learned filters
  - Apply non-linearity
  - Spatial pooling (downsample)

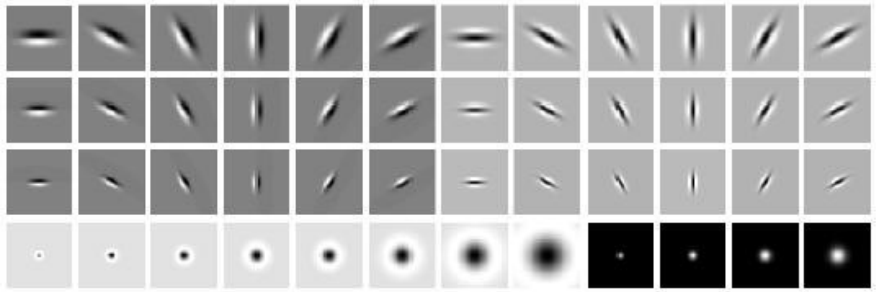
- Supervised training of convolutional filters by back-propagating classification error



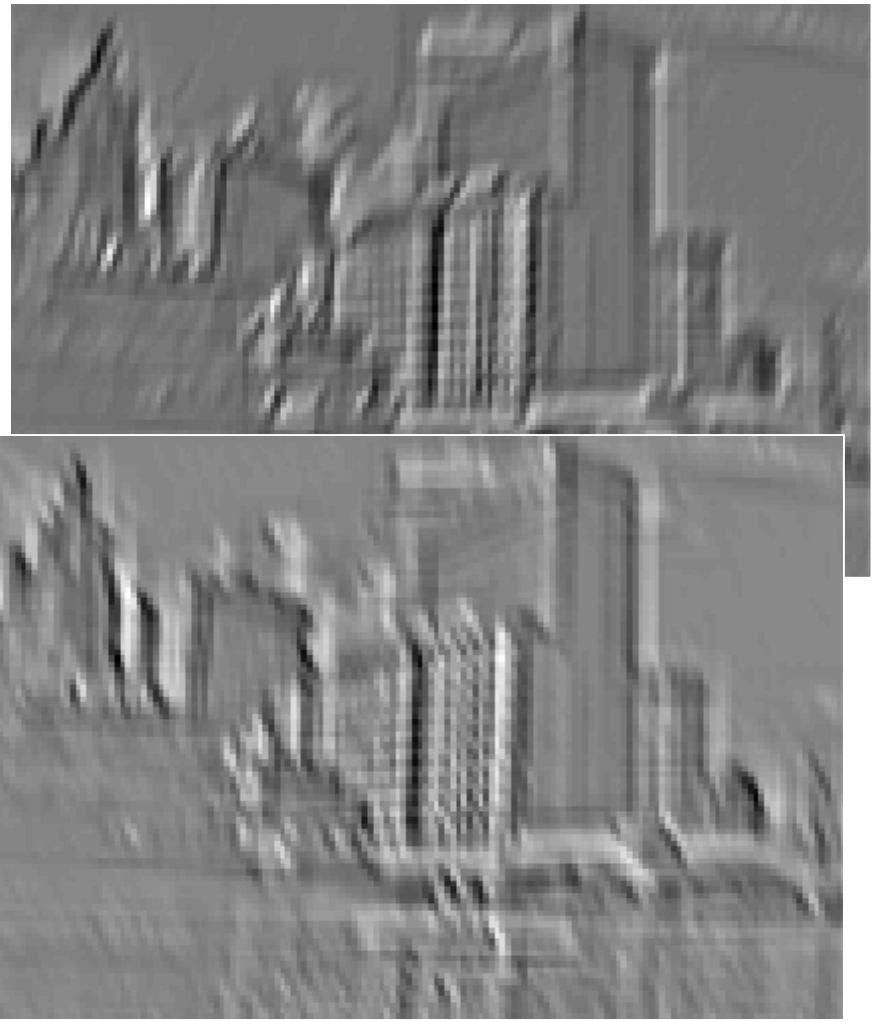
Adapted from Lana Lazebnik

1. Convolution

bank of K filters



Input



Feature Map

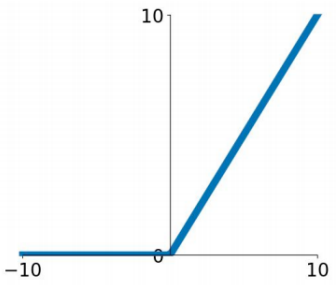
Adapted from Rob Fergus

2. Non-Linearity

- Per-element (independent)
- Options:
  - Tanh
  - Sigmoid: 1/(1+exp(-x))
  - Rectified linear unit (ReLU)
    - Avoids saturation issues

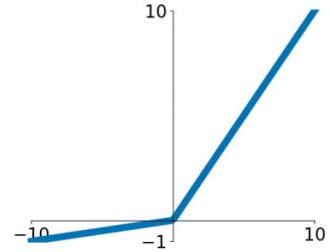
ReLU

$$\max(0, x)$$



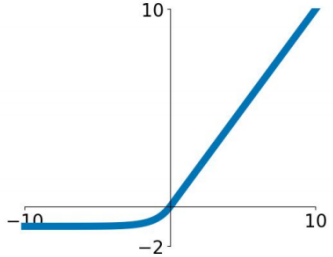
Leaky ReLU

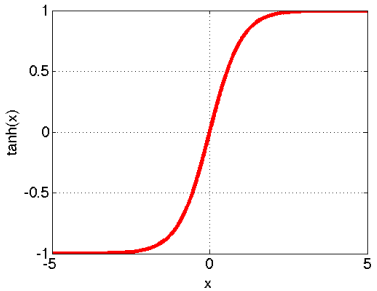
$$\max(0.1x, x)$$

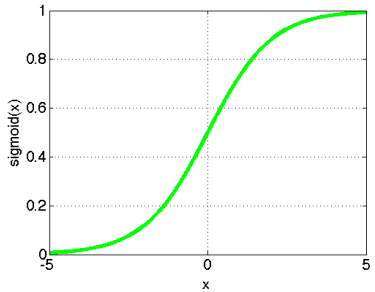


ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$





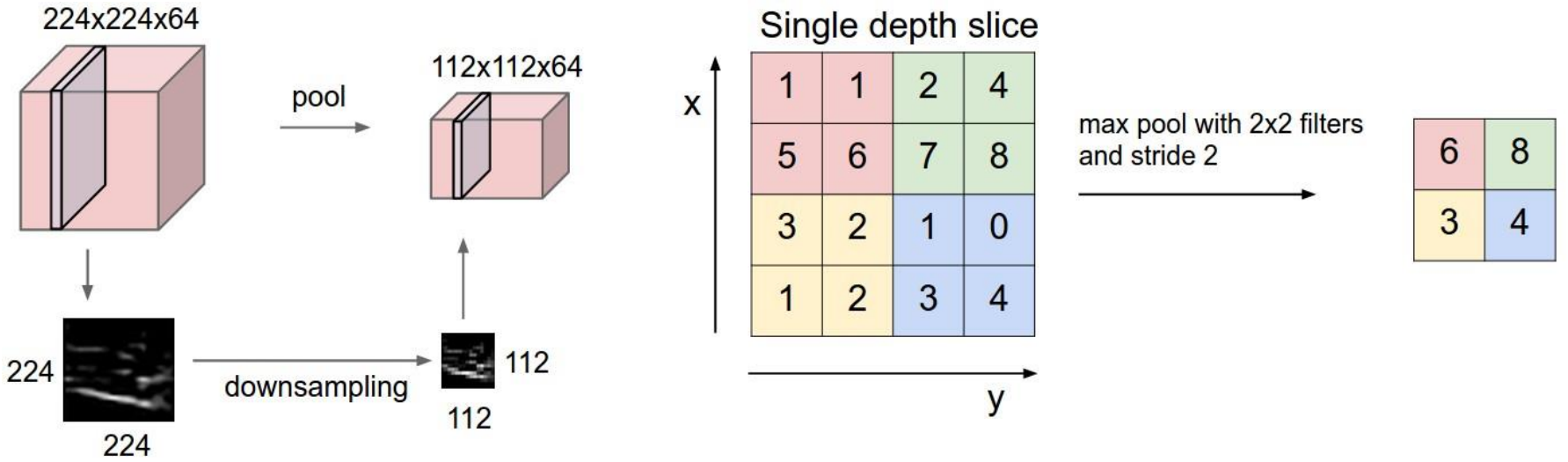


Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification [He et al. 2015]

Adapted from Rob Fergus

3. Spatial Pooling

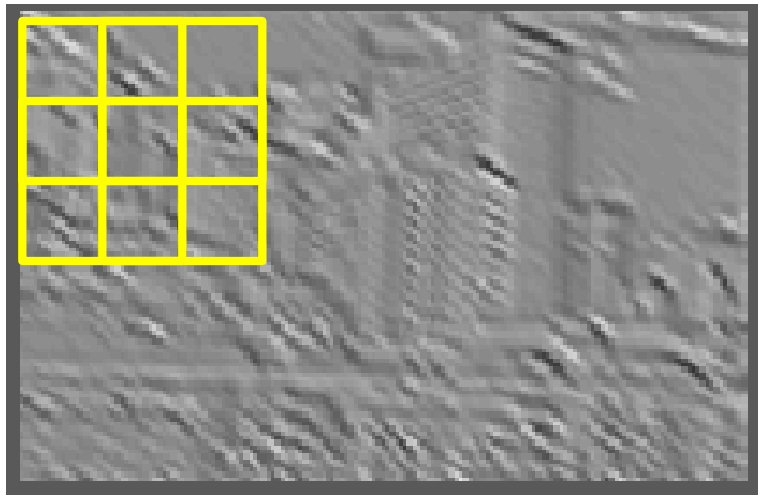
- Sum or max over non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (neurons see more of input)

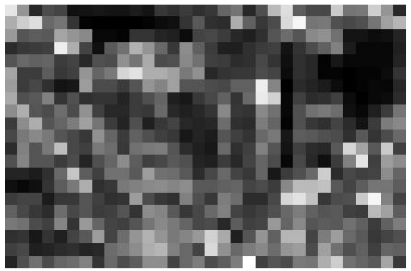


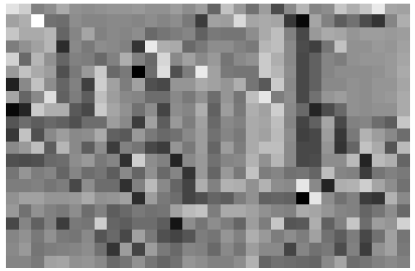
Rob Fergus, figure from Andrej Karpathy

3. Spatial Pooling

- Sum or max over non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (neurons see more of input)

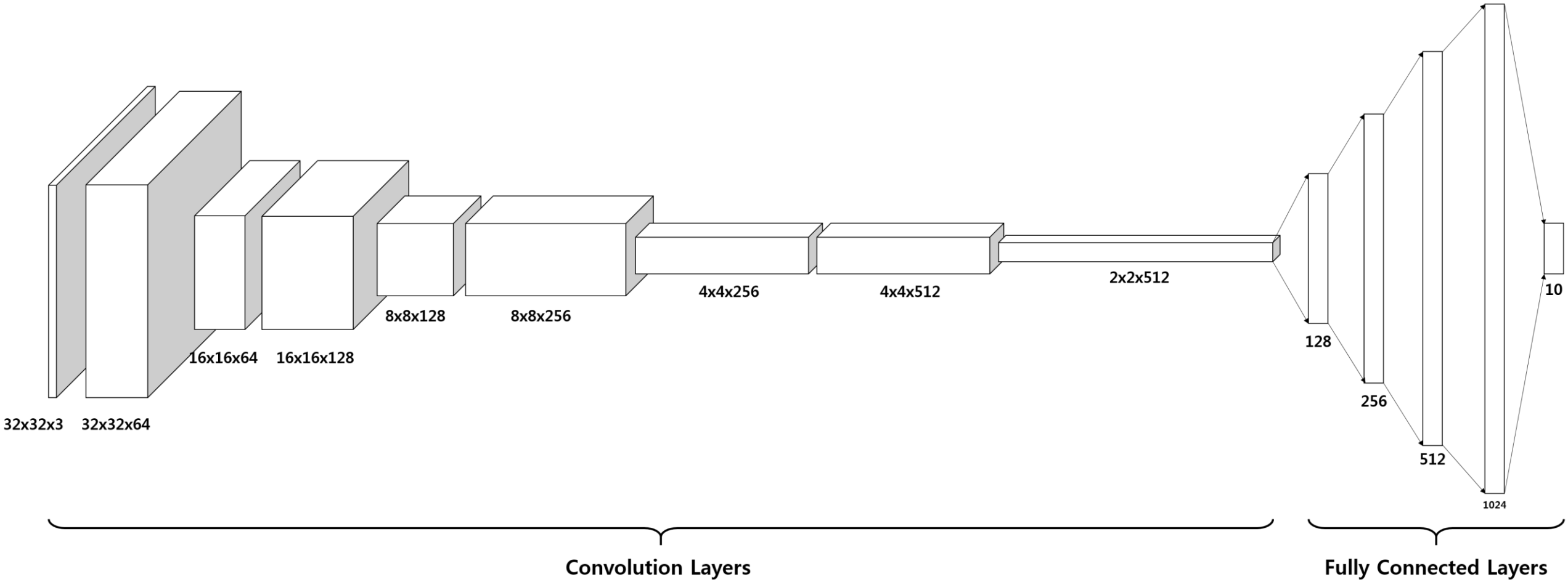




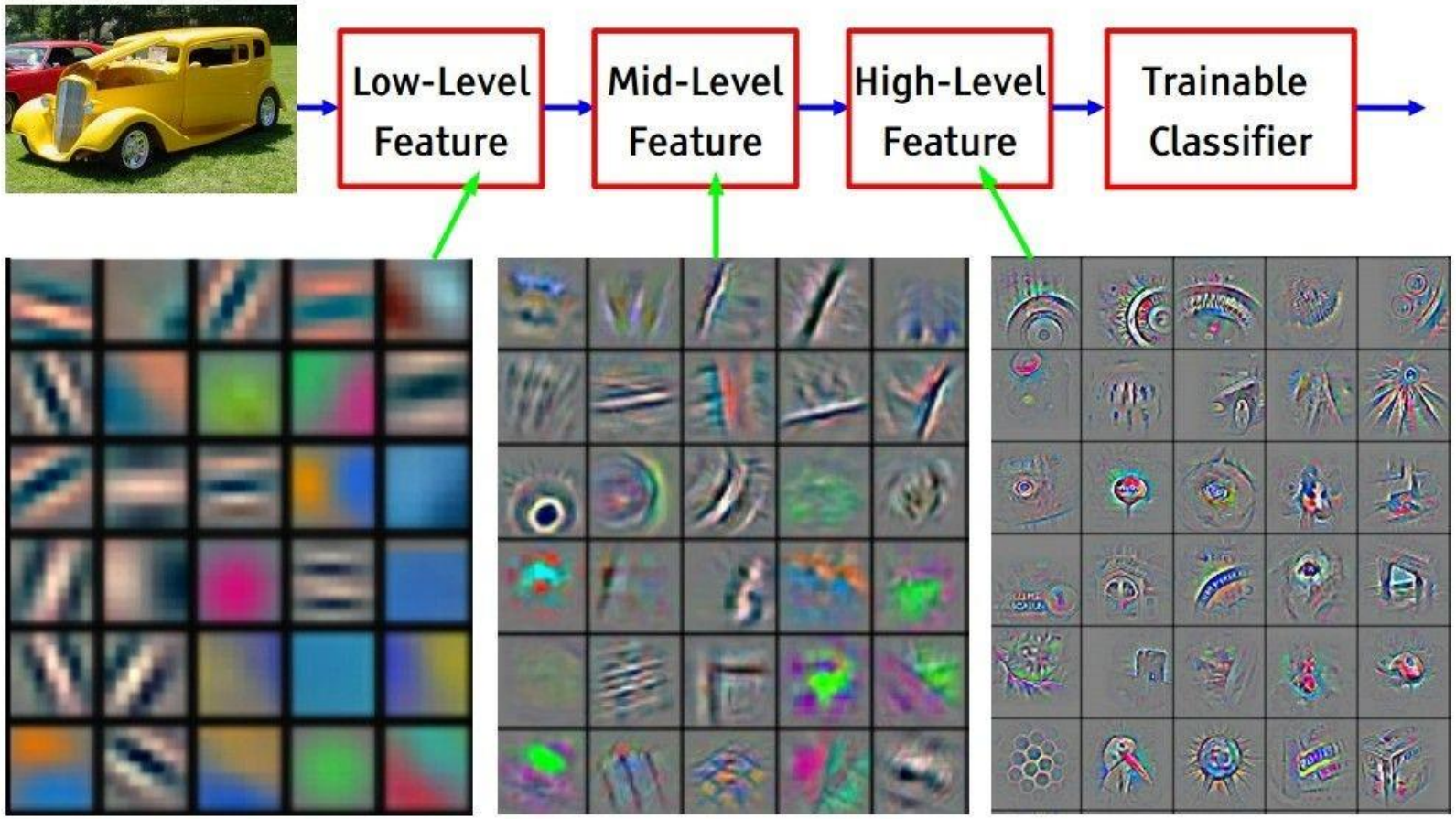


Adapted from Rob Fergus

Convolutional Network

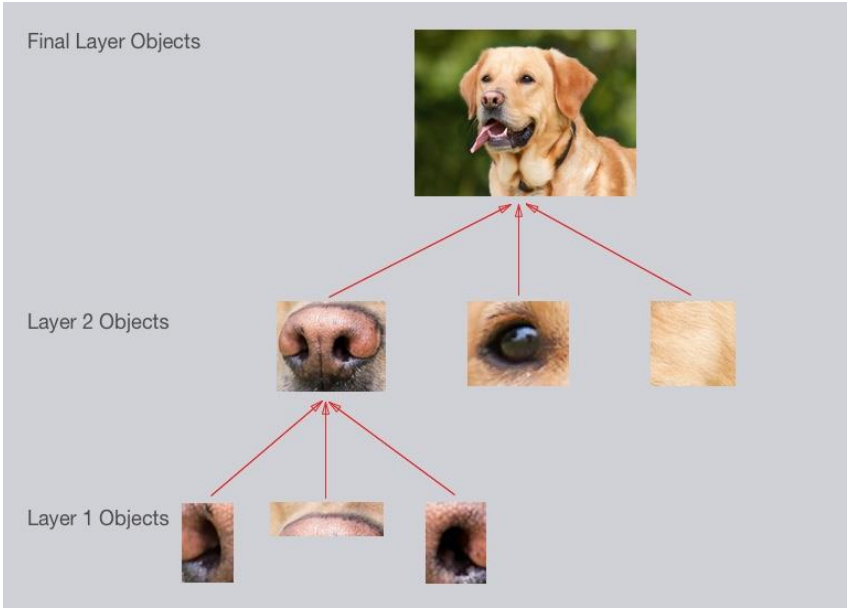


Convolutional Network



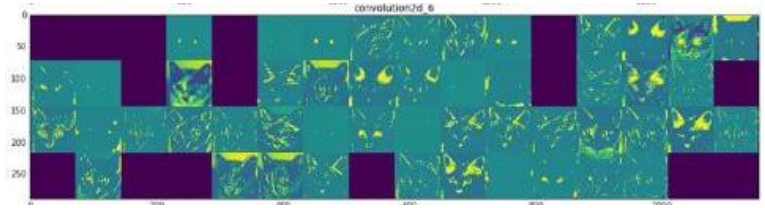
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Credit: Andrej Karpathy

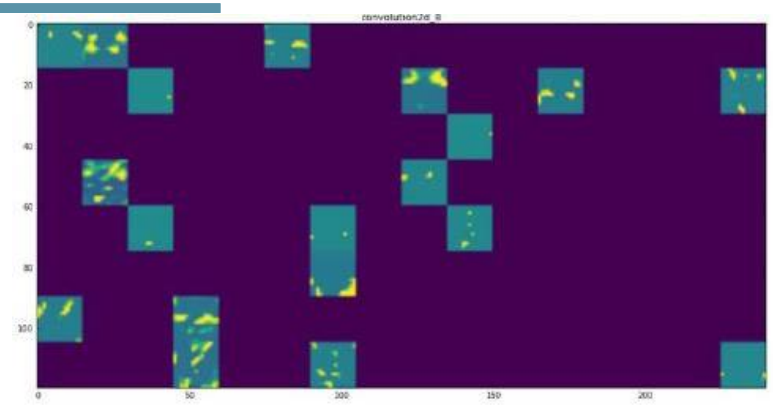


[From recent Yann LeCun slides]

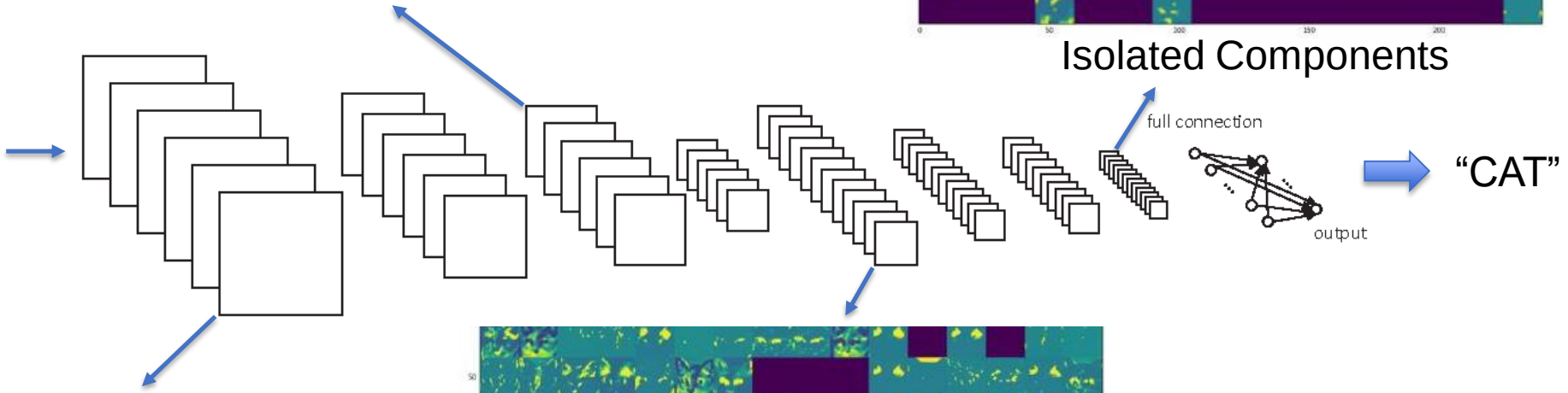
Visualizing what convnets learn

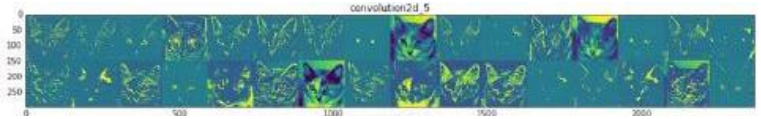


Blobs & Combinations of Features

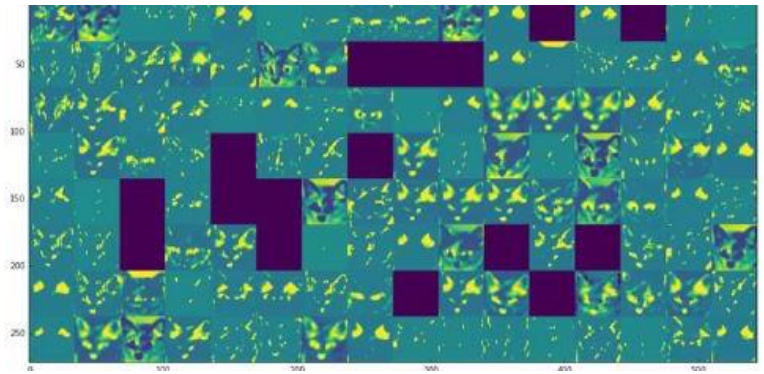


Isolated Components



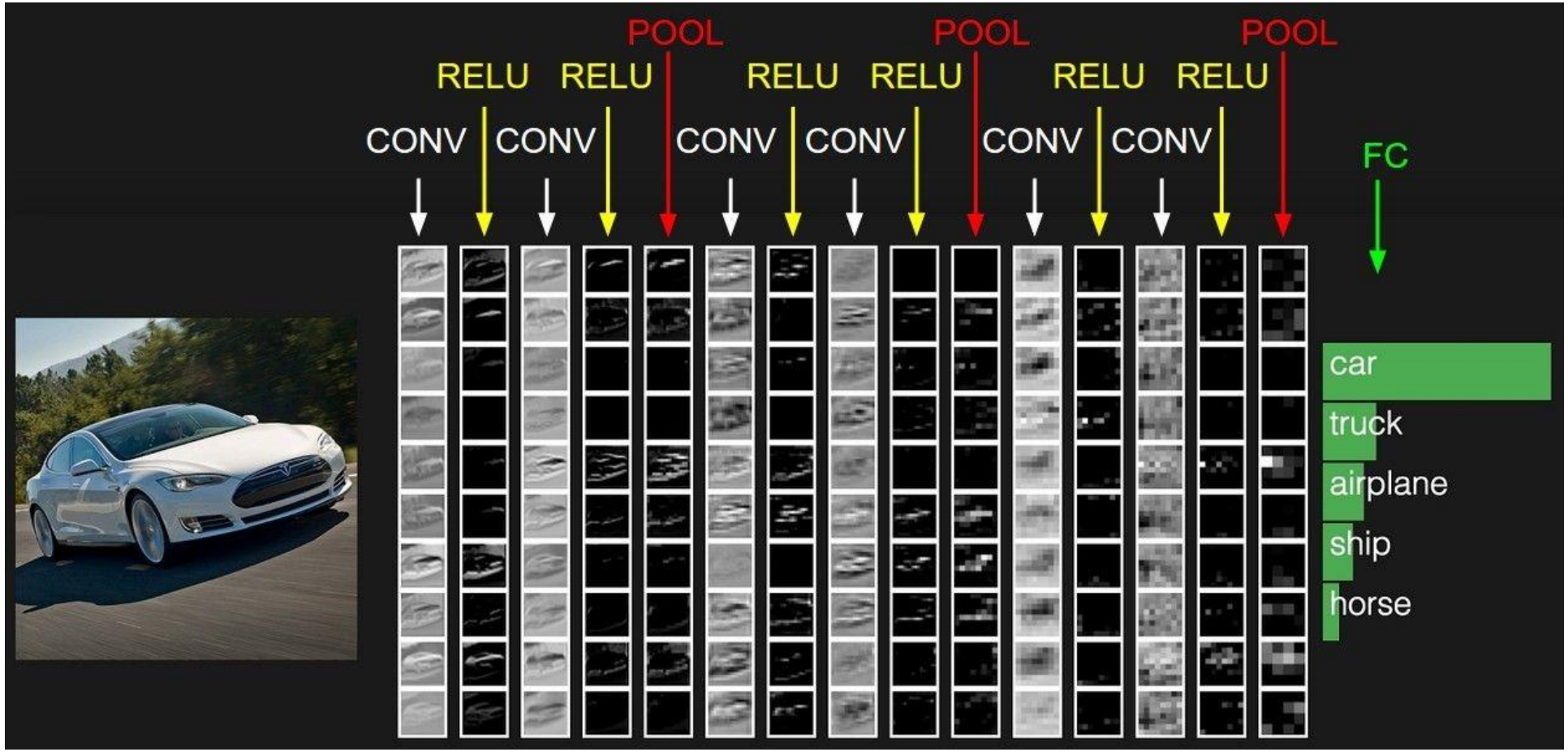


Edges - Colors



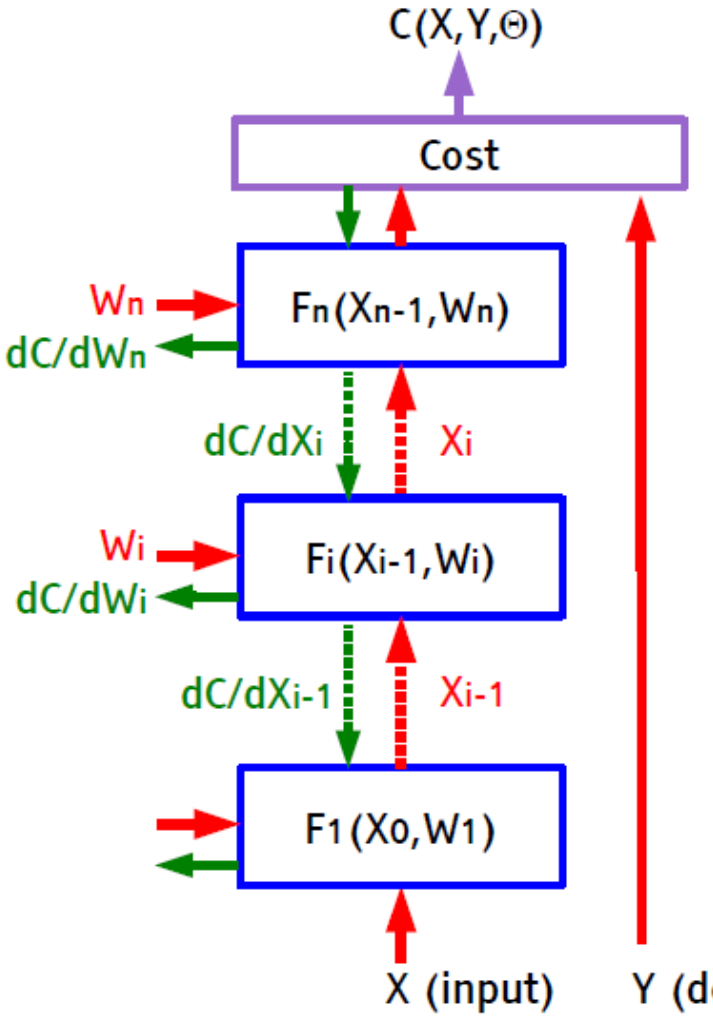
Object Parts-Motifs

Convolutional Network



Andrej Karpathy

Convolutional Network



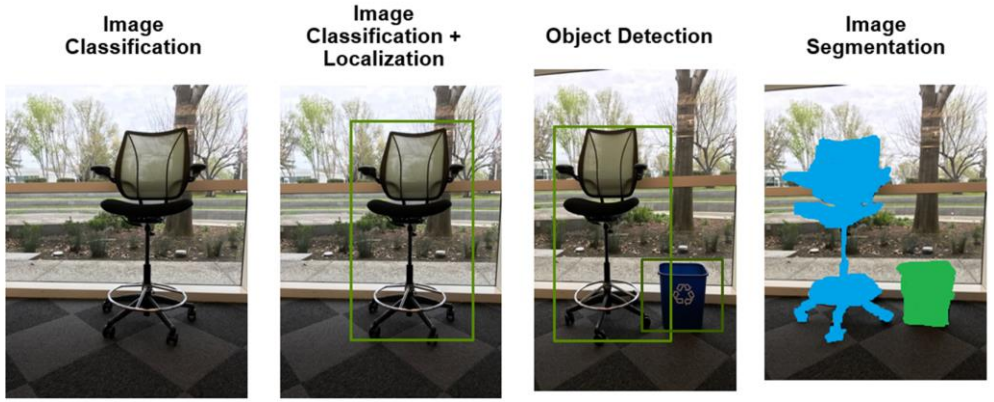
- A practical Application of Chain Rule

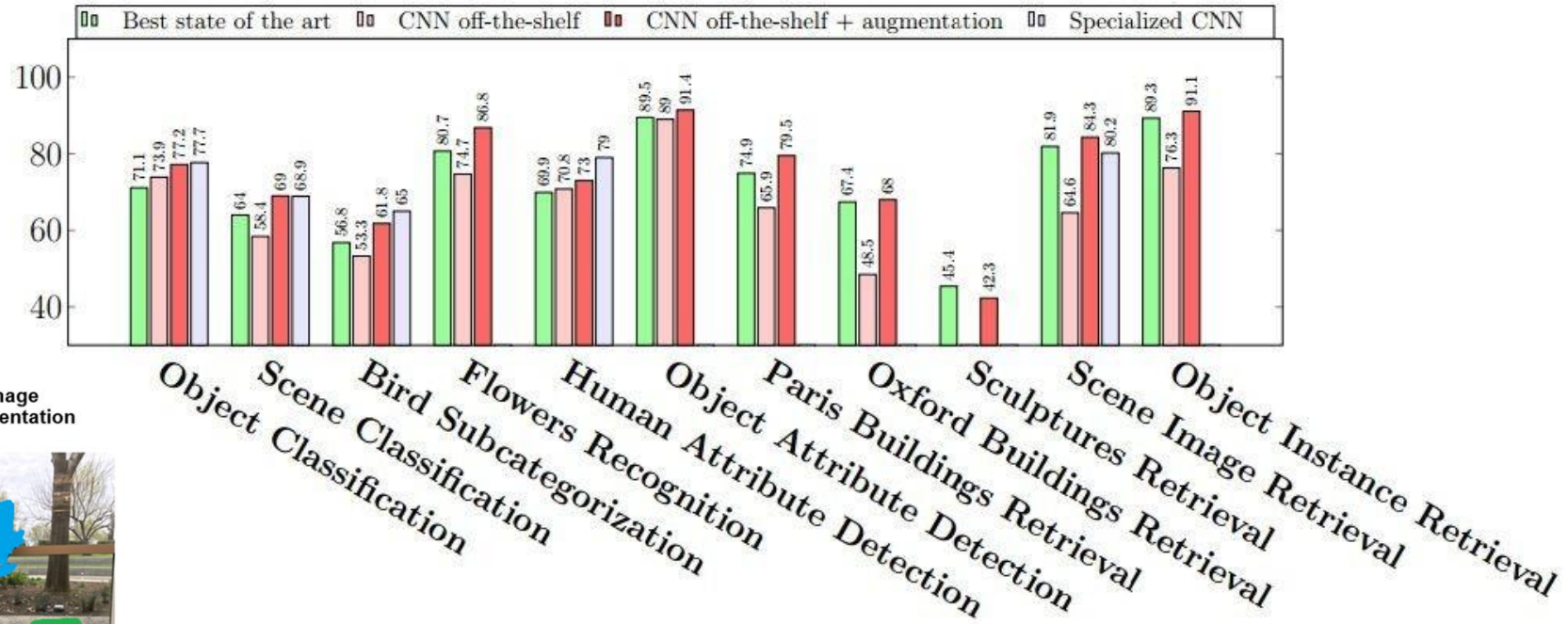
- Backprop for the state gradients:
- $dC/dX_{i-1} = dC/dX_i \cdot dX_i/dX_{i-1}$
- $dC/dX_{i-1} = dC/dX_i \cdot dF_i(X_{i-1},W_i)/dX_{i-1}$

- Backprop for the weight gradients:
- $dC/dW_i = dC/dX_i \cdot dX_i/dW_i$
- $dC/dW_i = dC/dX_i \cdot dF_i(X_{i-1},W_i)/dW_i$

Convolutional neural networks for Vision

- State of the art performance on many problems
- Most papers in recent vision conferences use deep neural networks





Razavian et al., CVPR 2014 Workshops

ImageNet Challenge

- ~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
  - Test set of 50,000 images

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

ImageNet Challenge 2012-2014

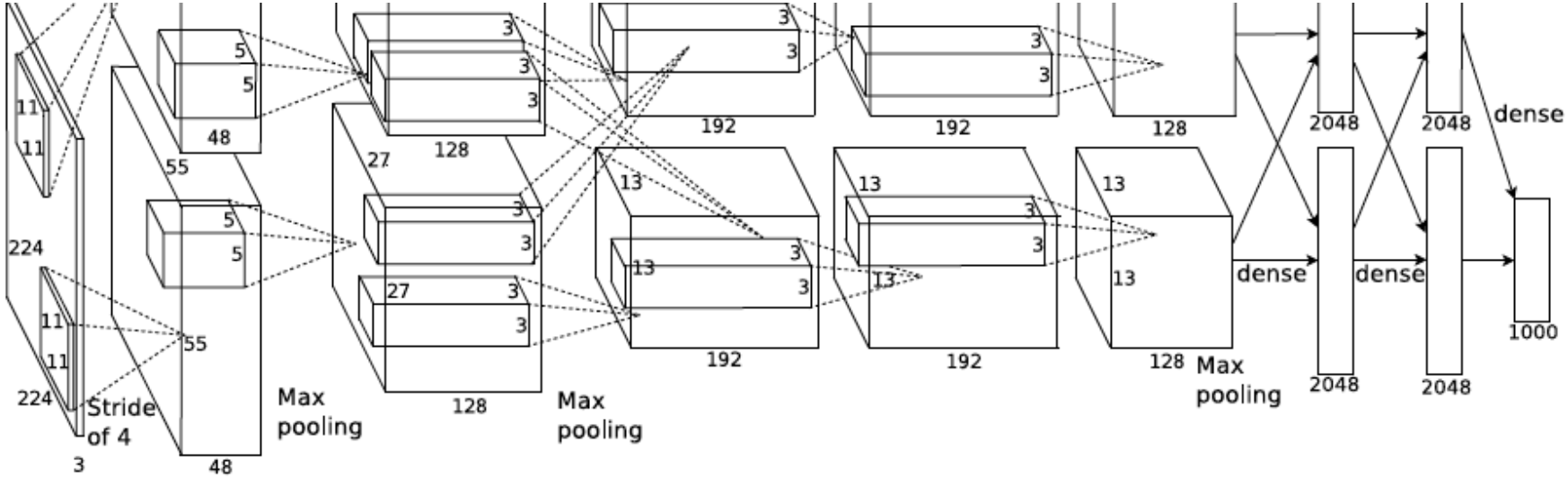
Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
Human expert*			5.1%	

Best non-convnet in 2012: 26.2%

- Top-5 error: The correct answer is amongst the top 5 predictions.

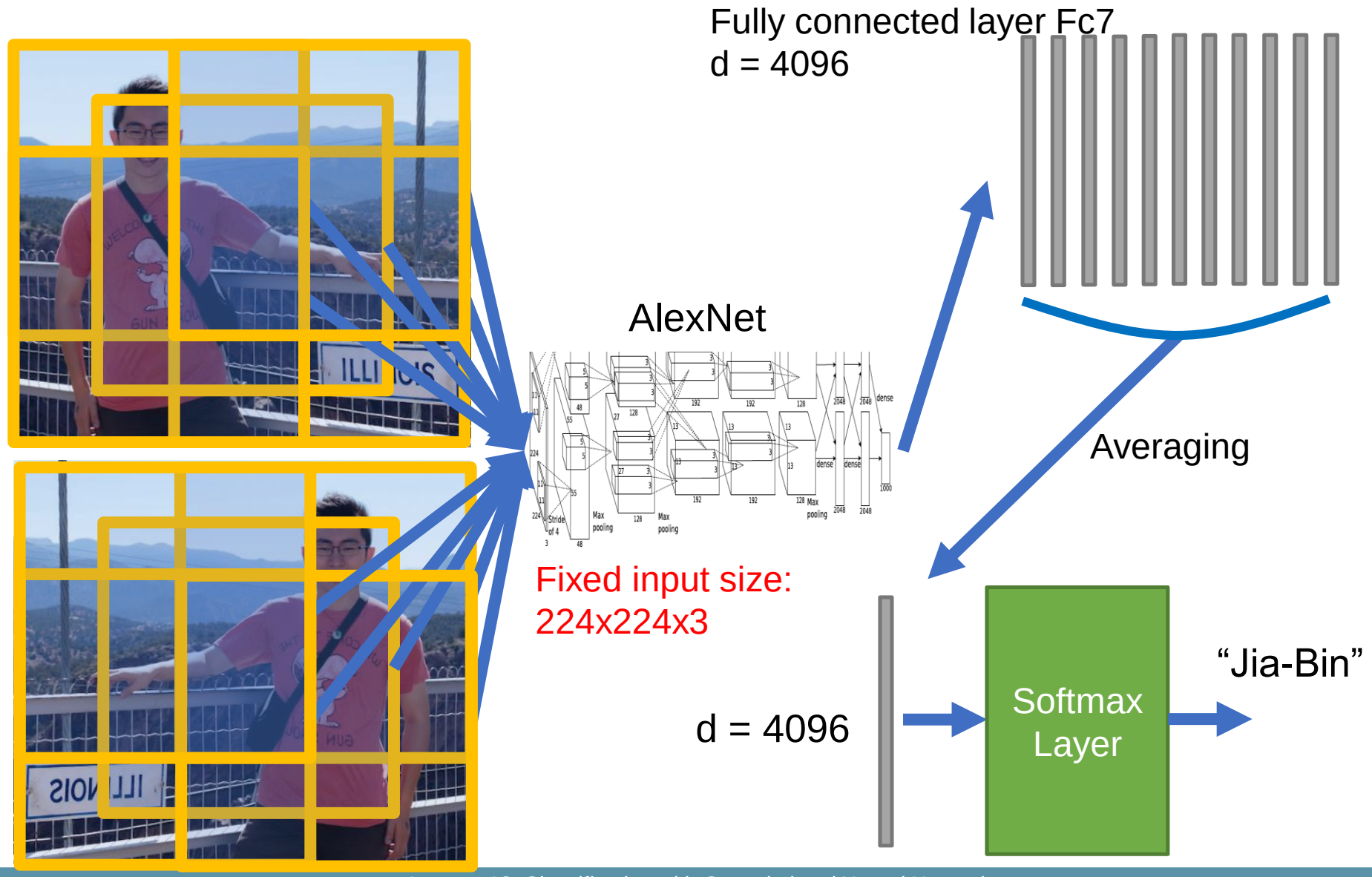
AlexNet: ILSVRC 2012 winner

- 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

Using CNN for Image Classification



Case Study: VGGNet

[Simonyan and Zisserman, 2014]

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

best model

11.2% top 5 error in ILSVRC 2013
->
7.3% top 5 error

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 **conv1-256**	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 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Andrej Karpathy

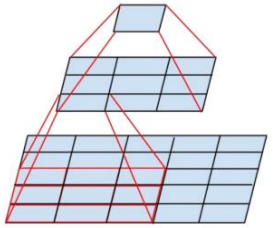
VGGNet: ILSVRC 2014 2nd place

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 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

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



- One 7x7 conv layer with K feature maps needs 49K2 weights, three 3x3 conv layers need only 27K2 weights
- Experimented with 1x1 convolutions

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

VGGNet: ILSVRC 2014 2nd place

Image source

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

GoogLeNet: ILSVRC 2014 winner

- The Inception Module

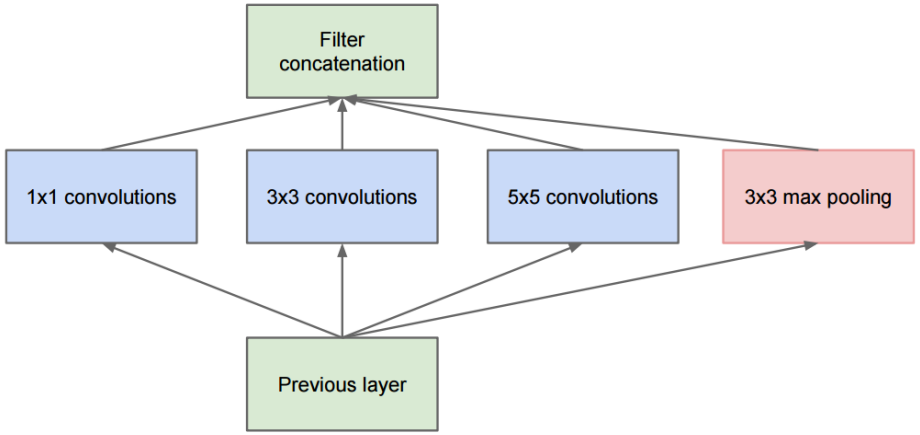
http://knowyourmeme.com/memes/we-need-to-go-deeper

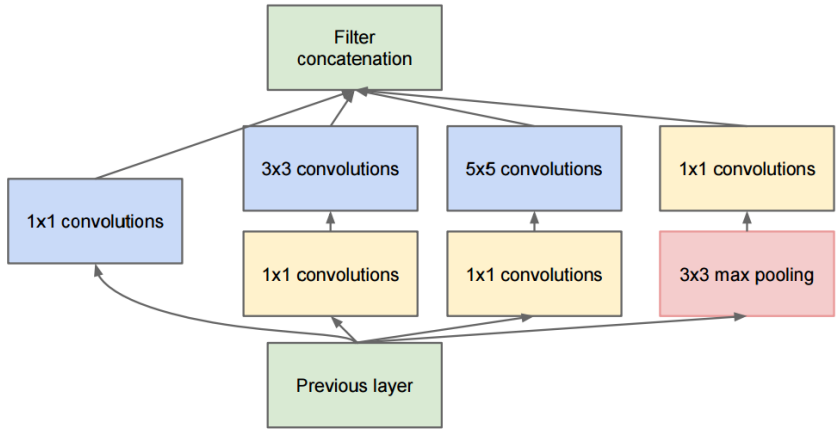
C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet

The Inception Module

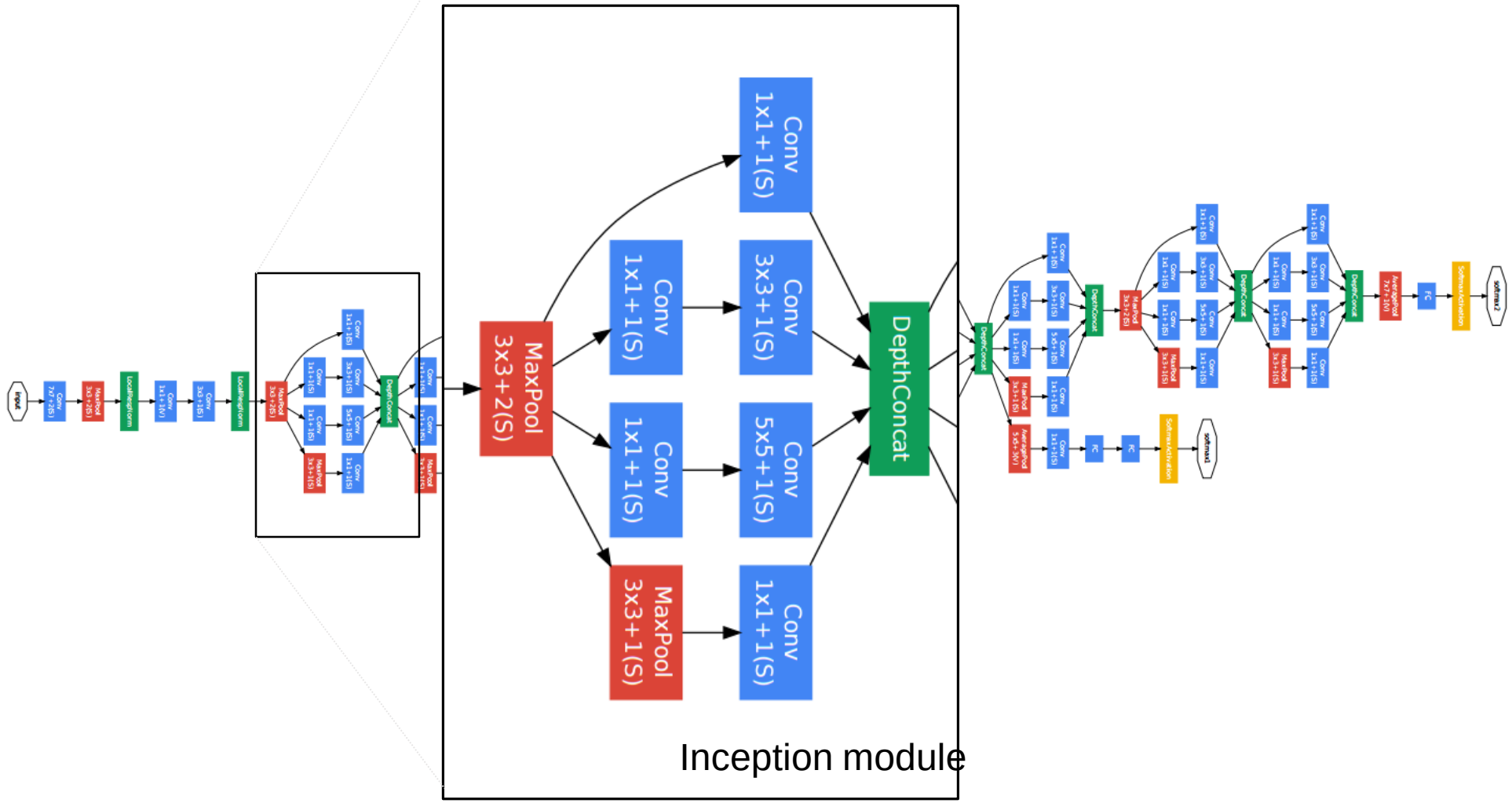
- Parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps
- Use 1x1 convolutions for dimensionality reduction before expensive convolutions





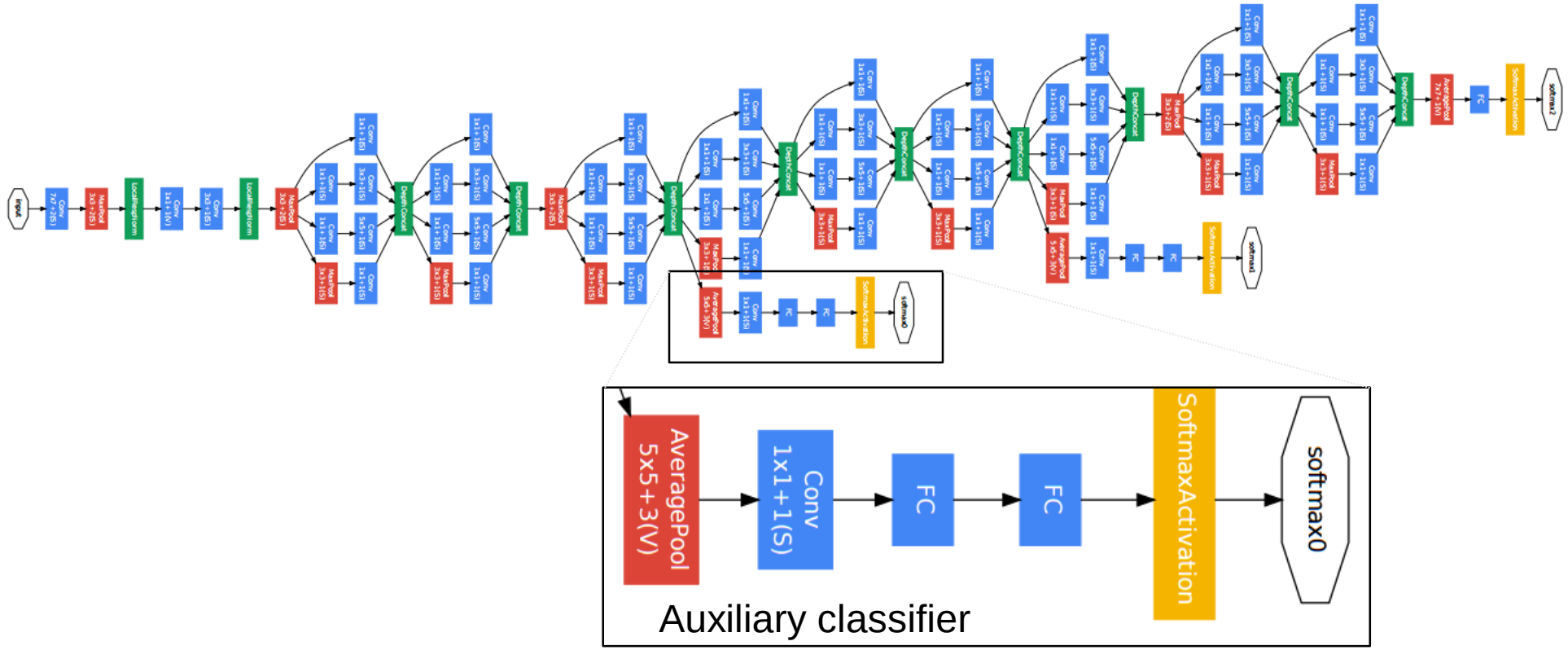
C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet



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

GoogLeNet



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

GoogLeNet

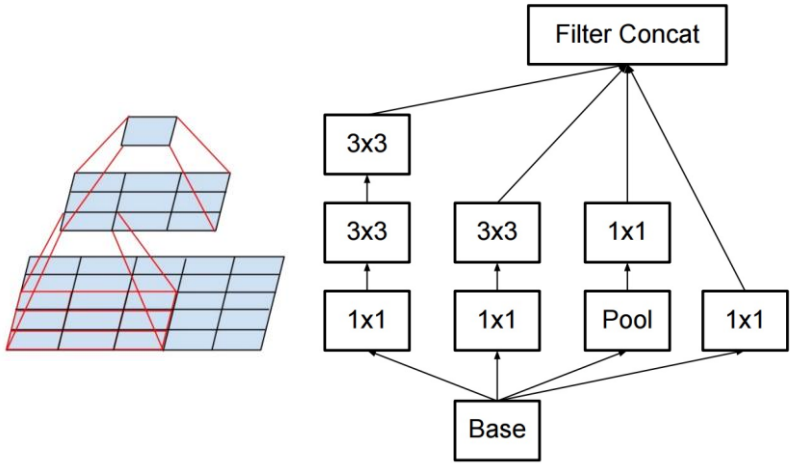
- GoogLeNet achieved 6.67% error

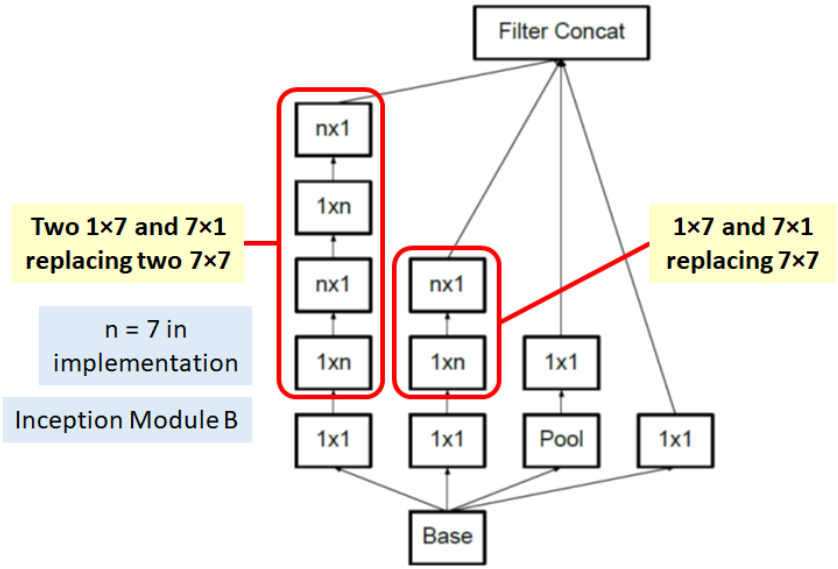
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	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								

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

Inception v2, v3

- Improve training with batch normalization, reducing importance of auxiliary classifiers
- More variants of inception modules with aggressive factorization of filters
- Increase the number of feature maps while decreasing spatial resolution (pooling)



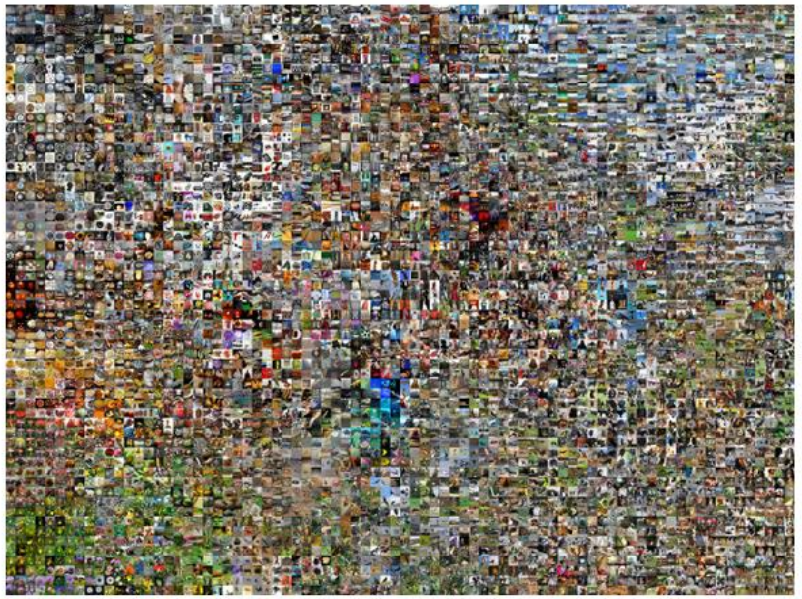


C. Szegedy et al., Rethinking the inception architecture for computer vision, CVPR 2016

Human Benchmark

ROBERT MCMILLAN 01.21.15 6:30 AM

THIS GUY BEAT GOOGLE'S SUPER-SMART AI—BUT IT WASN'T EASY



Human expert*			5.1%	

https://cs.stanford.edu/people/karpathy/ilsvrc/

ImageNet Challenge 2012-2014

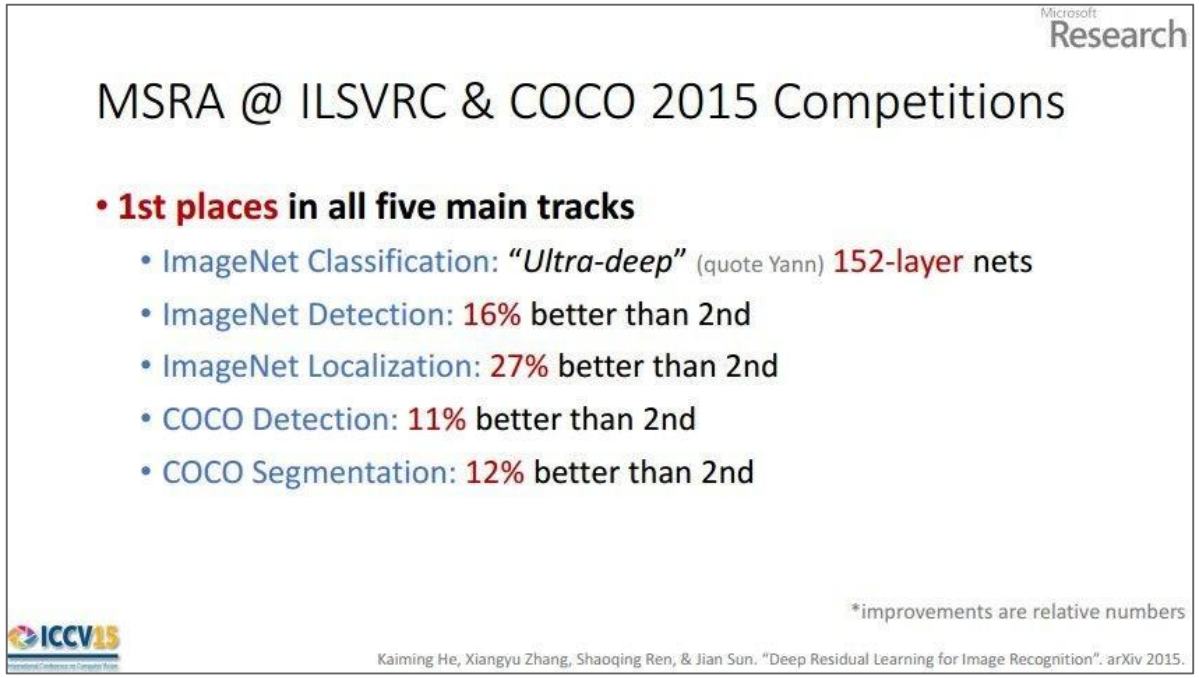
Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
Human expert*			5.1%	

http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)



2-3 weeks of training on 8 GPU machine

at runtime: faster than a VGGNet! (even though it has 8x more layers)

Slide from Kaiming He's recent presentation https://www.youtube.com/watch?v=1PGLj-uKT1w

https://d2l.ai/chapter_convolutional-modern/resnet.html

Andrej Karpathy

ResNet: ILSVRC 2015 winner

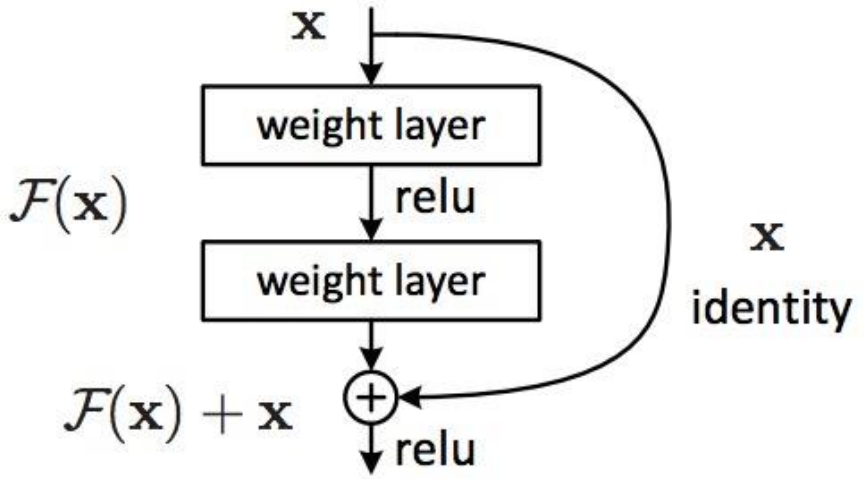
K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

Residual Networks Intuition

- Only if larger function classes contain the smaller ones are we guaranteed that increasing them strictly increases the expressive power of the network.
- This is the question that He et al, 2016 considered when working on very deep computer vision models.
- At the heart of ResNet is the idea that every additional layer should contain the identity function as one of its elements
- his means that if we can train the newly-added layer into an identity mapping $f(x) = x$, the new model will be as effective as the original model.

ResNet

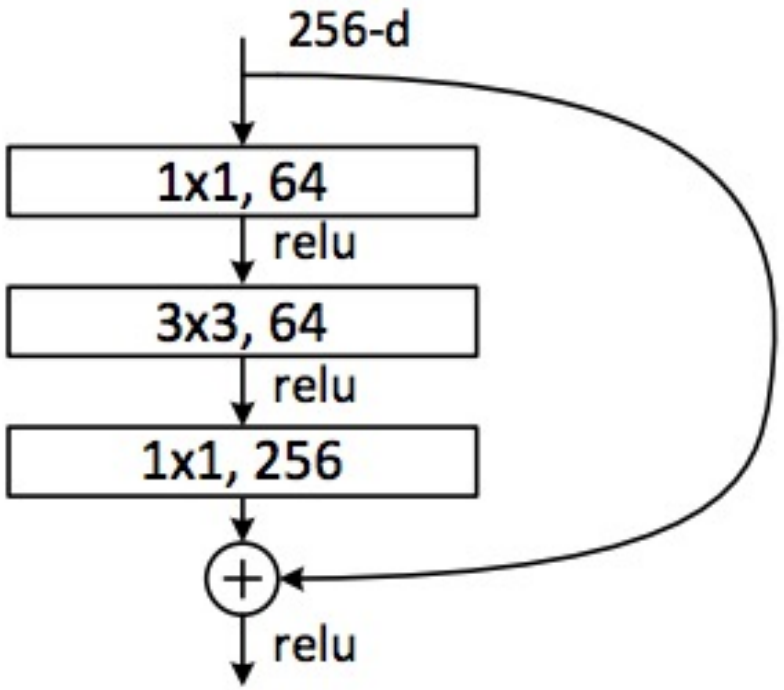
- The residual module
  - Introduce skip or shortcut connections (existing before in various forms in literature)
  - Make it easy for network layers to represent the identity mapping



K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

ResNet

Deeper residual module (bottleneck)



- Directly performing 3x3 convolutions with 256 feature maps at input and output:
  256 x 256 x 3 x 3 ~ 600K operations
- Using 1x1 convolutions to reduce 256 to 64 feature maps, followed by 3x3 convolutions, followed by 1x1 convolutions to expand back to 256 maps:
  256 x 64 x 1 x 1 ~ 16K
  64 x 64 x 3 x 3 ~ 36K
  64 x 256 x 1 x 1 ~ 16K
  Total: ~70K

K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

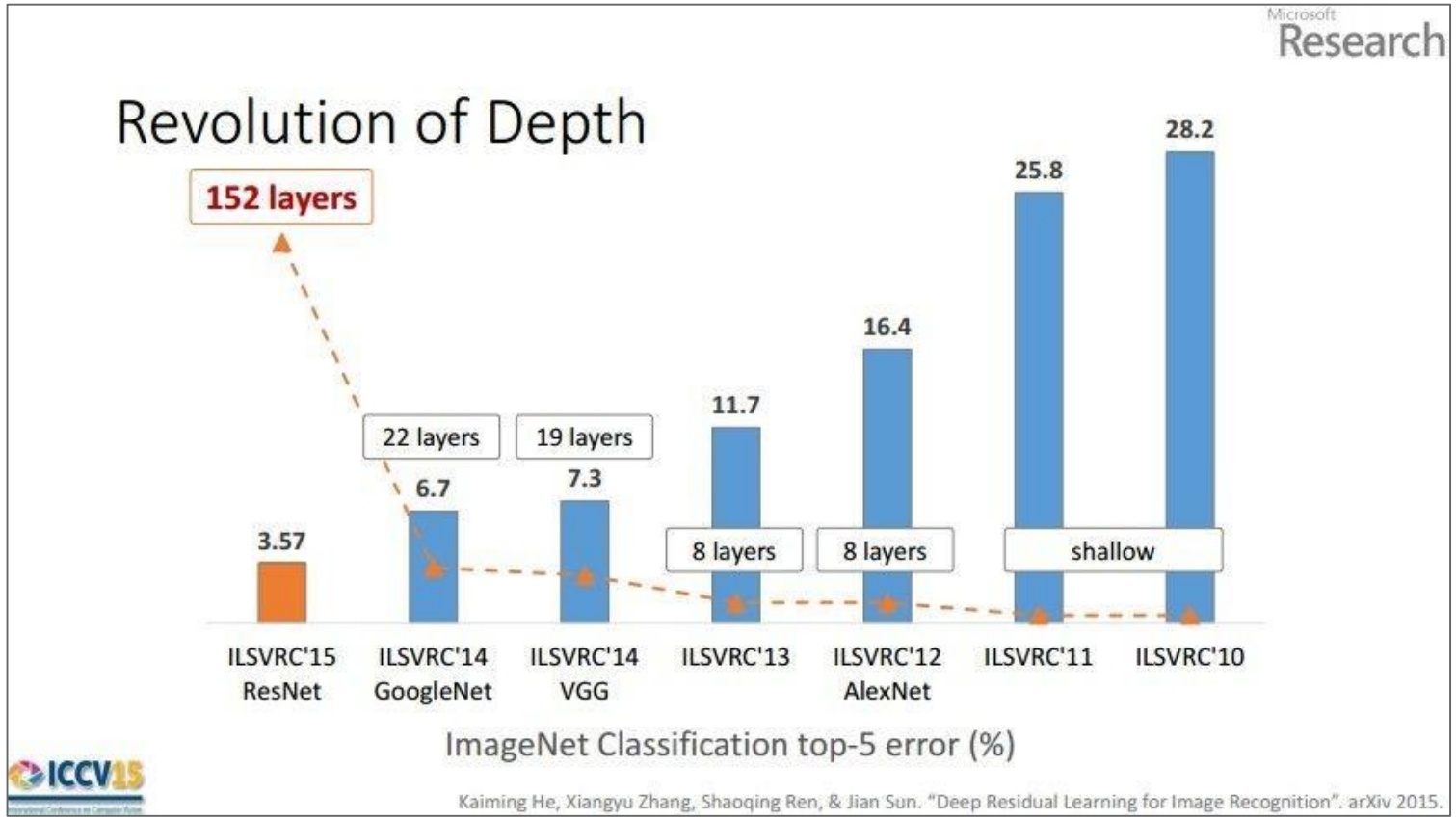
ResNet

- Architectures for ImageNet:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

ResNet

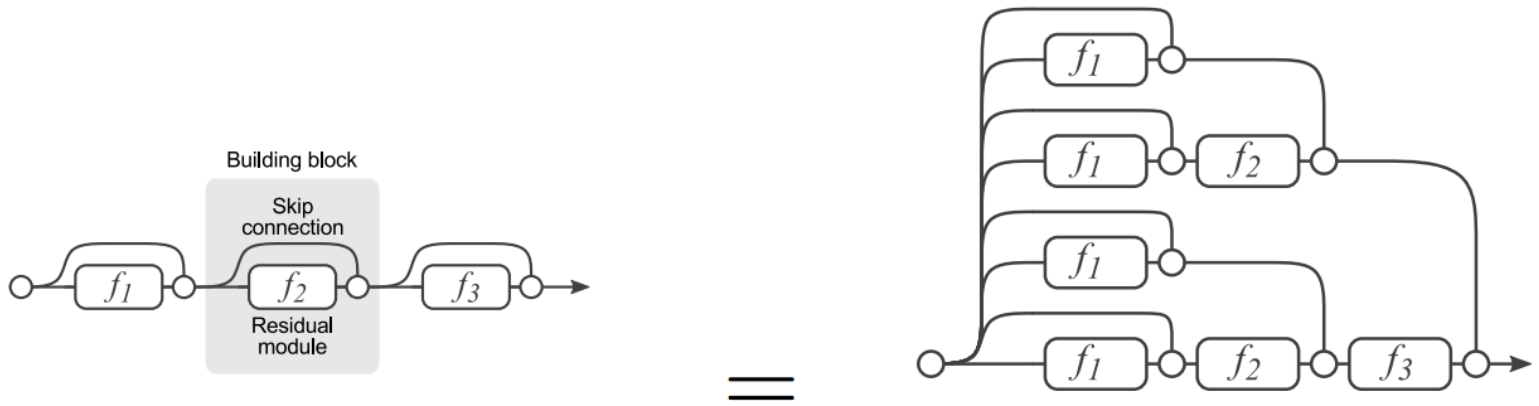


(slide from Kaiming He's recent presentation)

Andrej Karpathy

Interpreting ResNets

- ResNets are collections of many paths of different length, and shorter paths predominantly contribute to training

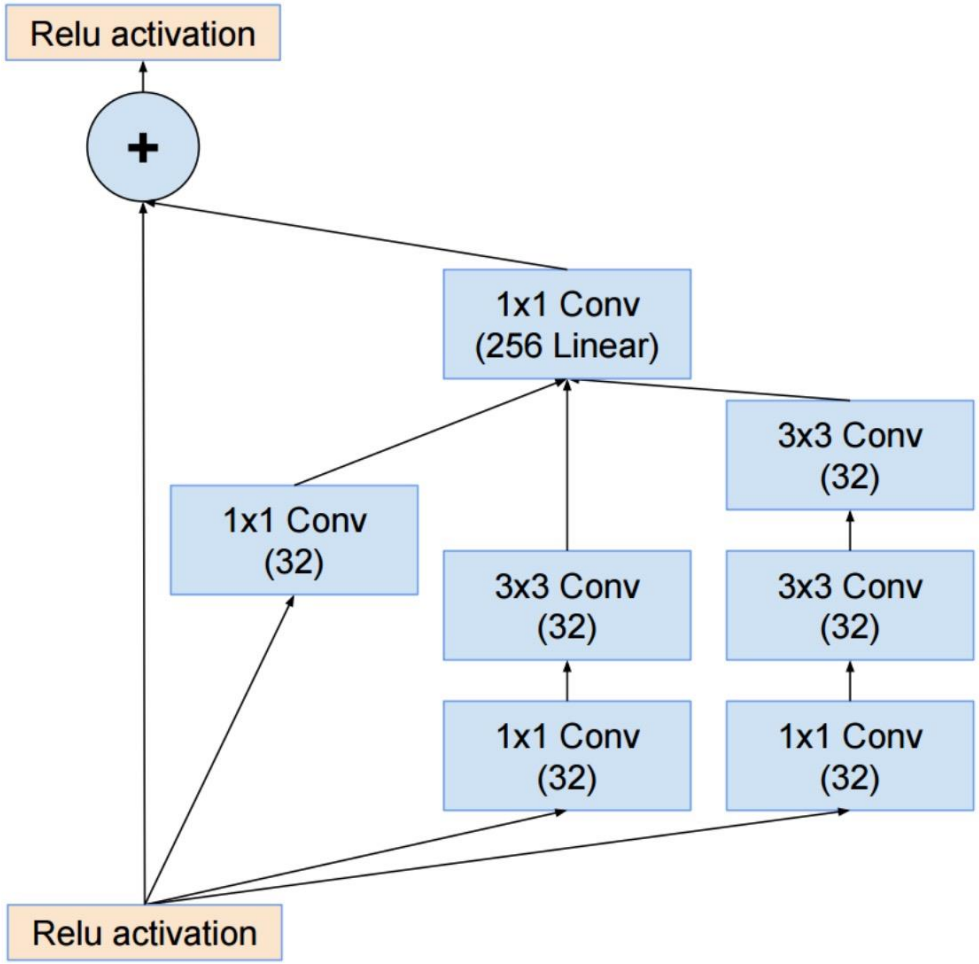


(a) Conventional 3-block residual network (b) Unraveled view of (a)

Figure 1: Residual Networks are conventionally shown as (a), which is a natural representation of Equation (1). When we expand this formulation to Equation (6), we obtain an *unraveled view* of a 3-block residual network (b). Circular nodes represent additions. From this view, it is apparent that residual networks have $O(2^n)$ implicit paths connecting input and output and that adding a block doubles the number of paths.

A. Veit, M. Wilber, S. Belongie, Residual Networks Behave Like Ensembles of Relatively Shallow Networks, NIPS 2016

Inception v4



C. Szegedy et al., Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, arXiv 2016

Summary: ILSVRC 2012-2015

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (AlexNet, 7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
ResNet (152 layers)	2015	1st	3.57%	
Human expert*			5.1%	

http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

What's missing from the picture?

- Training tricks and details: initialization, regularization, normalization
- Training data augmentation
- Averaging classifier outputs over multiple crops/flips
- Ensembles of networks

- Officially, starting with 2015, image classification is not part of ILSVRC challenge, but people continue to benchmark on the data

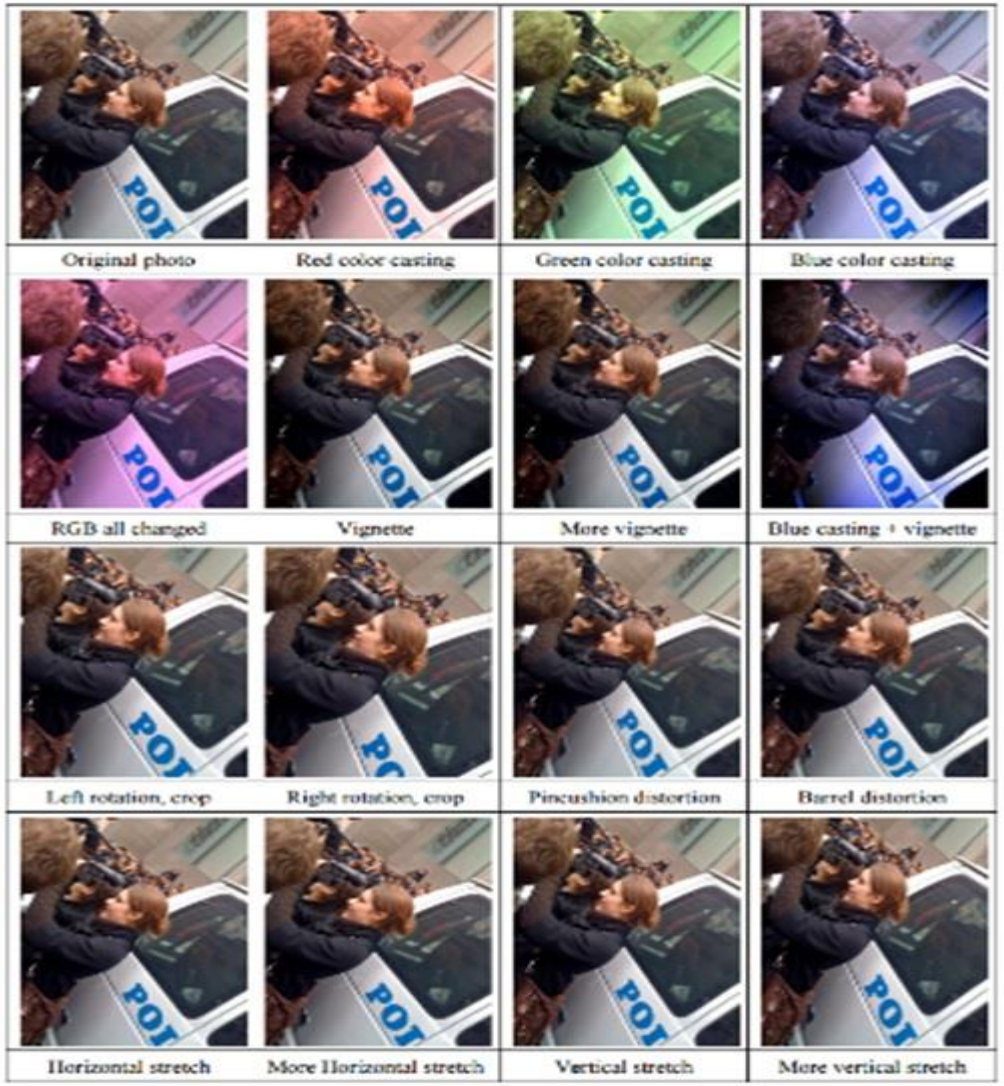
Training Convolutional Neural Networks

- Backpropagation + stochastic gradient descent with momentum
  - Neural Networks: Tricks of the Trade

- Advanced Regularization
  - Data augmentation
  - Dropout
  - Batch normalization

Data Augmentation (Jittering)

- Create virtual training samples
  - Horizontal flip
  - Random crop
  - Color casting
  - Geometric distortion

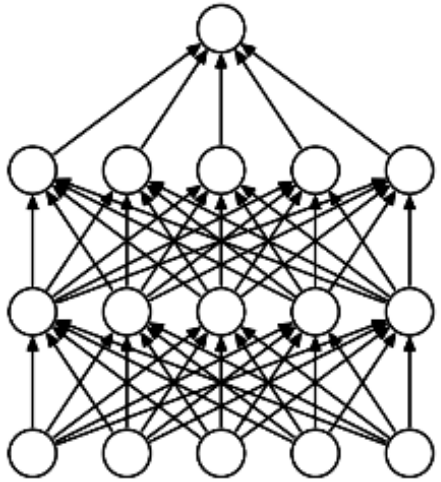


Deep Image [Wu et al. 2015]

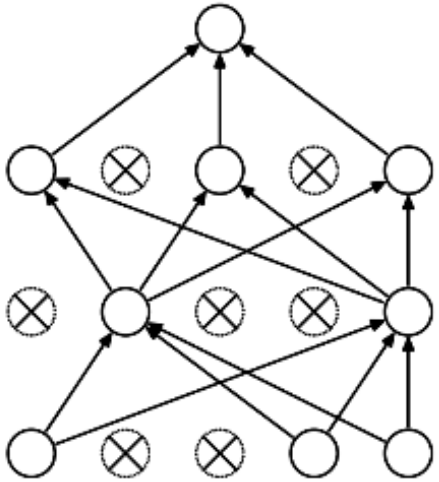
Jia-bin Huang

Regularization: Dropout

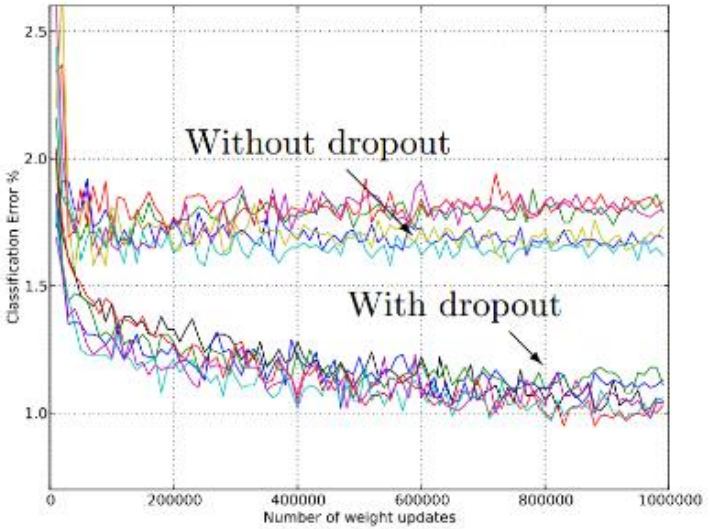
- Randomly turn off some neurons
- Allows individual neurons to independently be responsible for performance
- We can specify the drop rate (or keep rate) per layer, e.g., 0.5



(a) Standard Neural Net



(b) After applying dropout.



Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Adapted from Jia-bin Huang

Batch Normalization

- It was motivated by the problem of internal covariance shift, where changes in the distribution of the inputs of each layer affects the affect the learning rate of the network.
- It tries to make the output of a layer have a mean of 0 and unit variance by normalizing the output based on statistics of the batch earning rate of the network.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

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

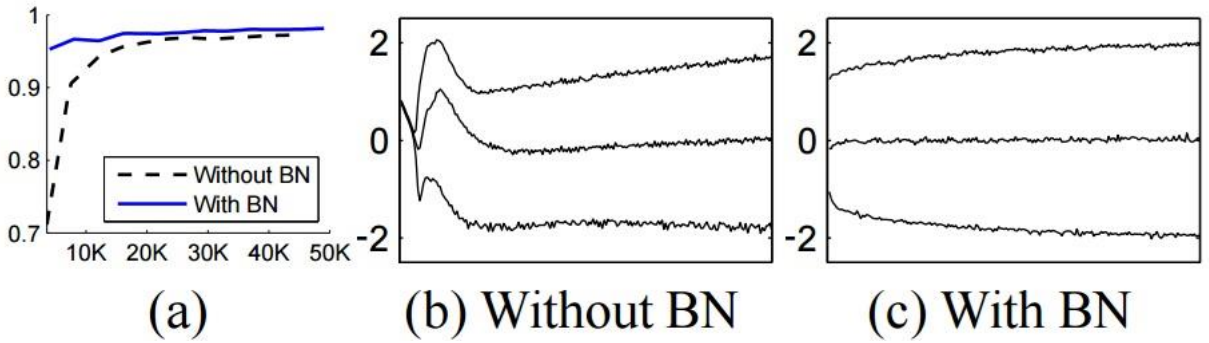
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

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

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

Used internally at that layer

Propagated to Next layer



Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [Ioffe and Szegedy 2015]

Design principles

- Make networks parameter-efficient
  - Reduce filter sizes, factorize filters
  - Use 1×1 convolutions to reduce number of feature maps before more expensive operations
  - Minimize reliance on FC layers
- Reduce spatial resolution gradually, within each level of resolution replicate a given “block” multiple times
- Use skip connections and/or create multiple redundant paths through the network
- Play around with depth vs. width vs. “cardinality”

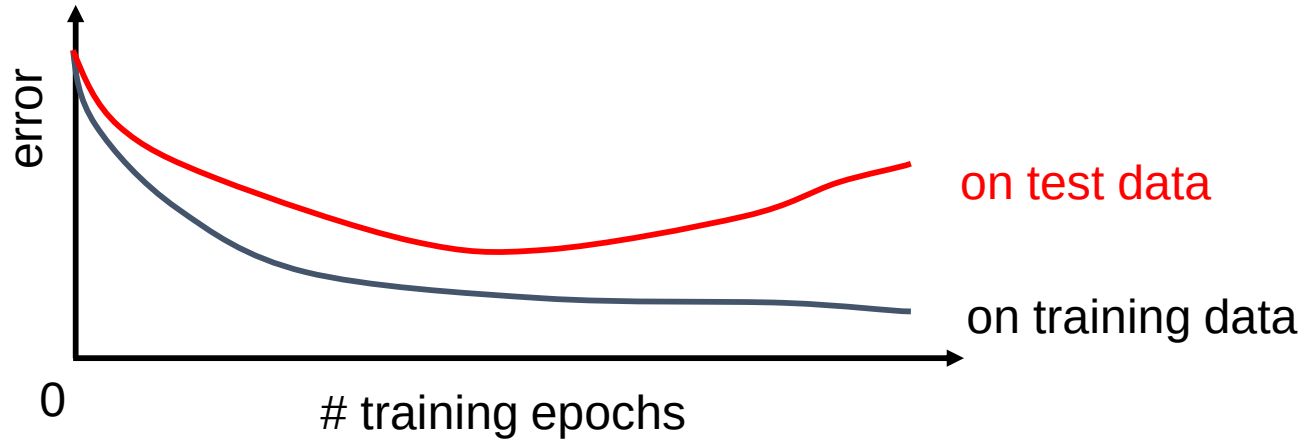
Comments on training algorithm

- Not guaranteed to converge to zero training error, may converge to local optima or oscillate indefinitely.
- However, in practice, does converge to low error for many large networks on real data.
- Thousands of epochs (epoch = network sees all training data once) may be required, hours or days to train.
- To avoid local-minima problems, run several trials starting with different random weights (random restarts), and take results of trial with lowest training set error.
- May be hard to set learning rate and to select number of hidden units and layers.
- Neural networks had fallen out of fashion in 90s, early 2000s; back with a new name and significantly improved performance (deep networks trained with dropout and lots of data).

Ray Mooney, Carlos Guestrin, Dhruv Batra

Over-training prevention

- Running too many epochs can result in over-fitting.



- Keep a hold-out validation set and test accuracy on it after every epoch. Stop training when additional epochs actually increase validation error.

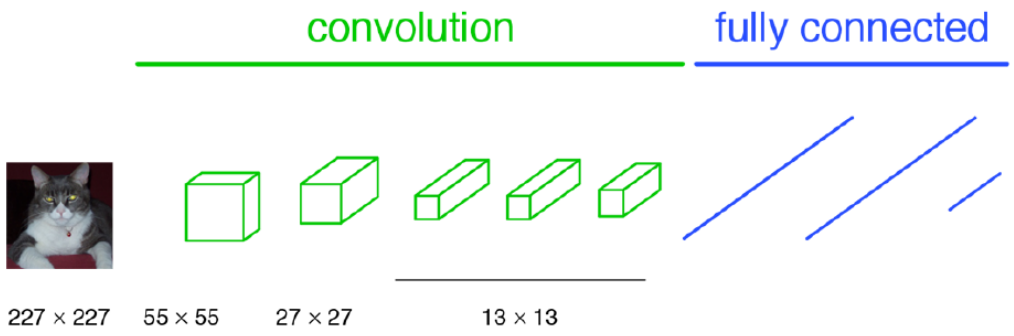
Adapted from Ray Mooney

Training: Best practices

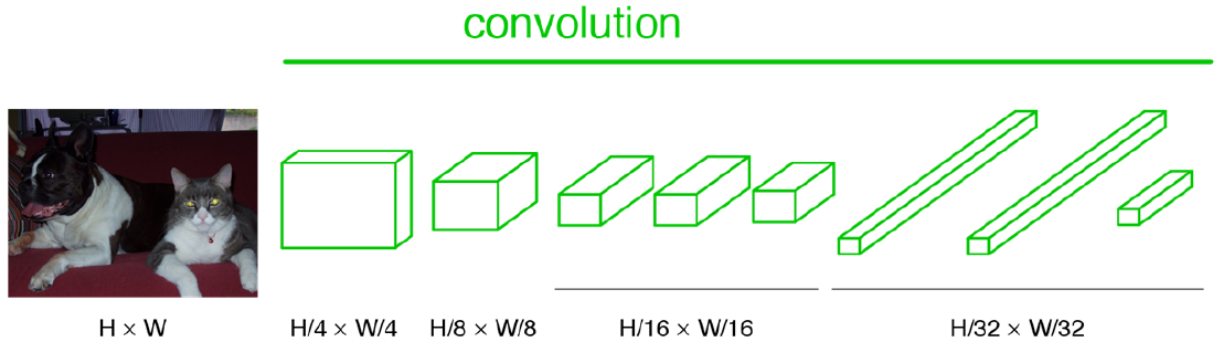
- Use mini-batch
- Use regularization (L1,L2, dropout)
- Use gradient checks (especially for custom functionality)
- Use cross-validation for your parameters
- Use RELU or leaky RELU or ELU, don’t use sigmoid
- Center (subtract mean from) your data
- To initialize, use “Xavier initialization”
- Learning rate: too high? Too low?
  - Cosine annealing

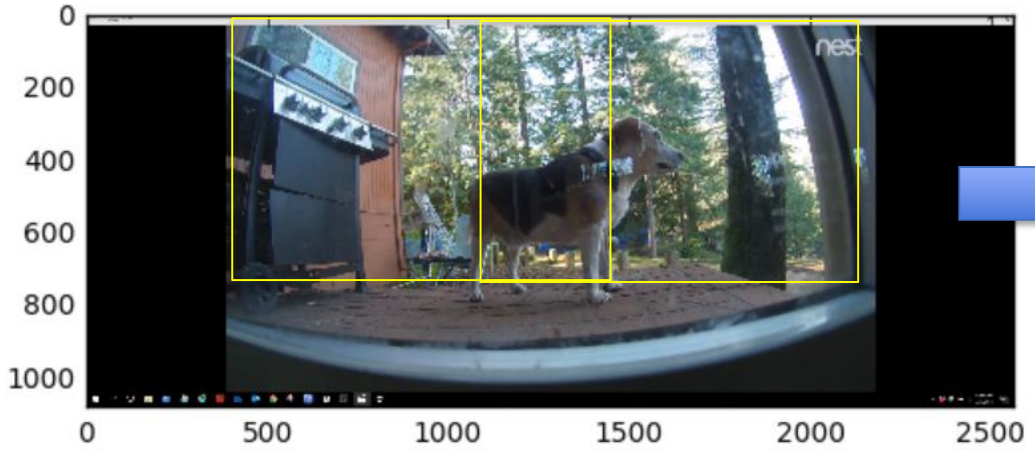
FCN: Becoming Fully Convolutional

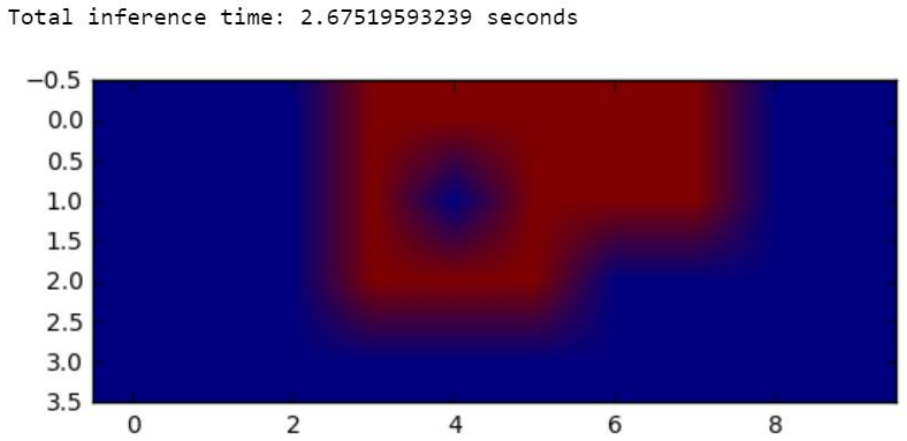
- Convert fully connecter layers to convolutional layers. Can input any size image and get an array output



"tabby cat"







Transfer Learning

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

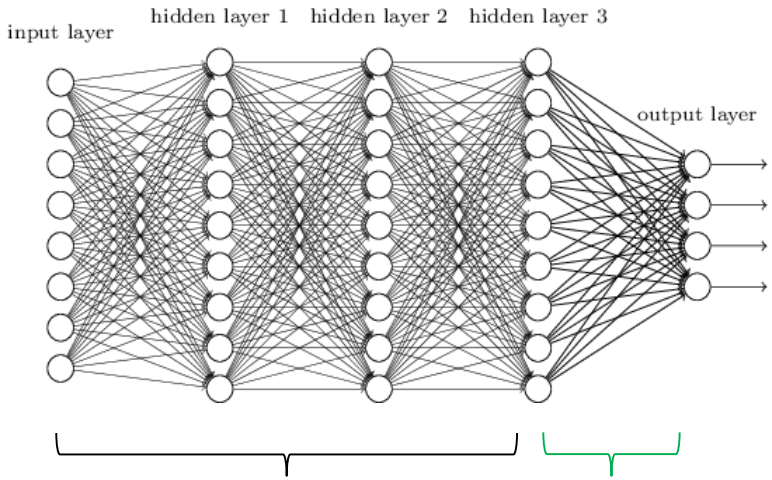
- Transfer learning is a machine learning (ML) technique where knowledge gained during the training of one set of ML problems can be used to train other similar types of problems.

Andrej Karpathy

https://github.com/dipanjanS/hands-on-transfer-learning-with-python

Transfer Learning with CNNs

- The more weights you need to learn, the more data you need
- That's why with a deeper network, you need more data for training than for a shallower network
- One possible solution:

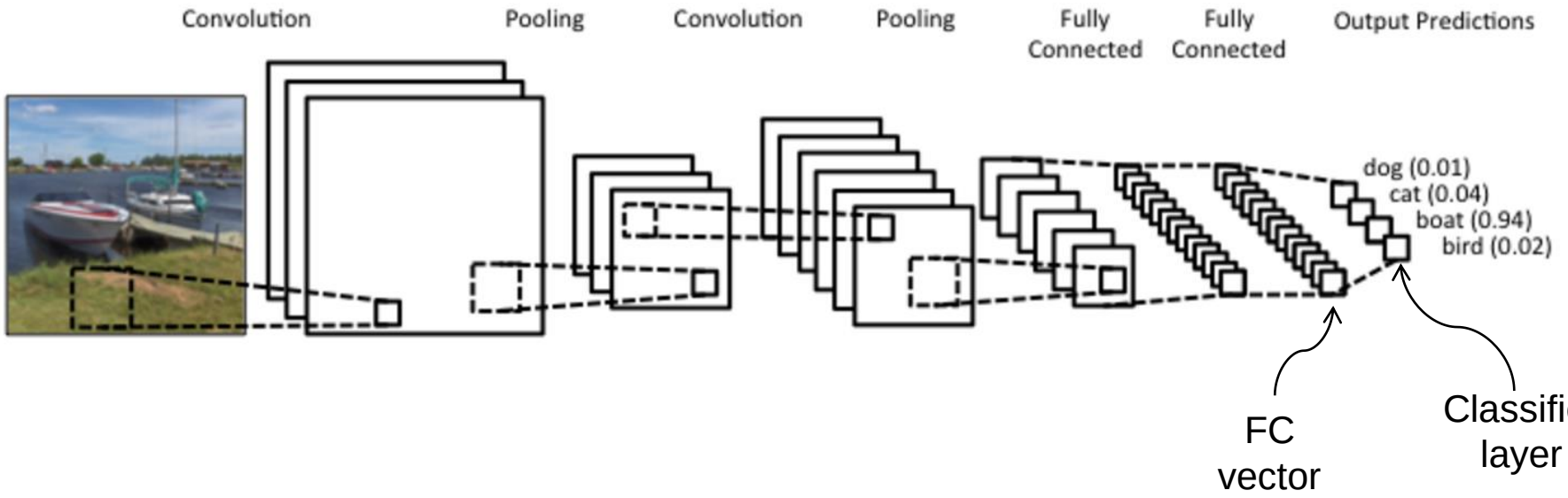


Set these to the already learned weights from another network

Learn these on your own task

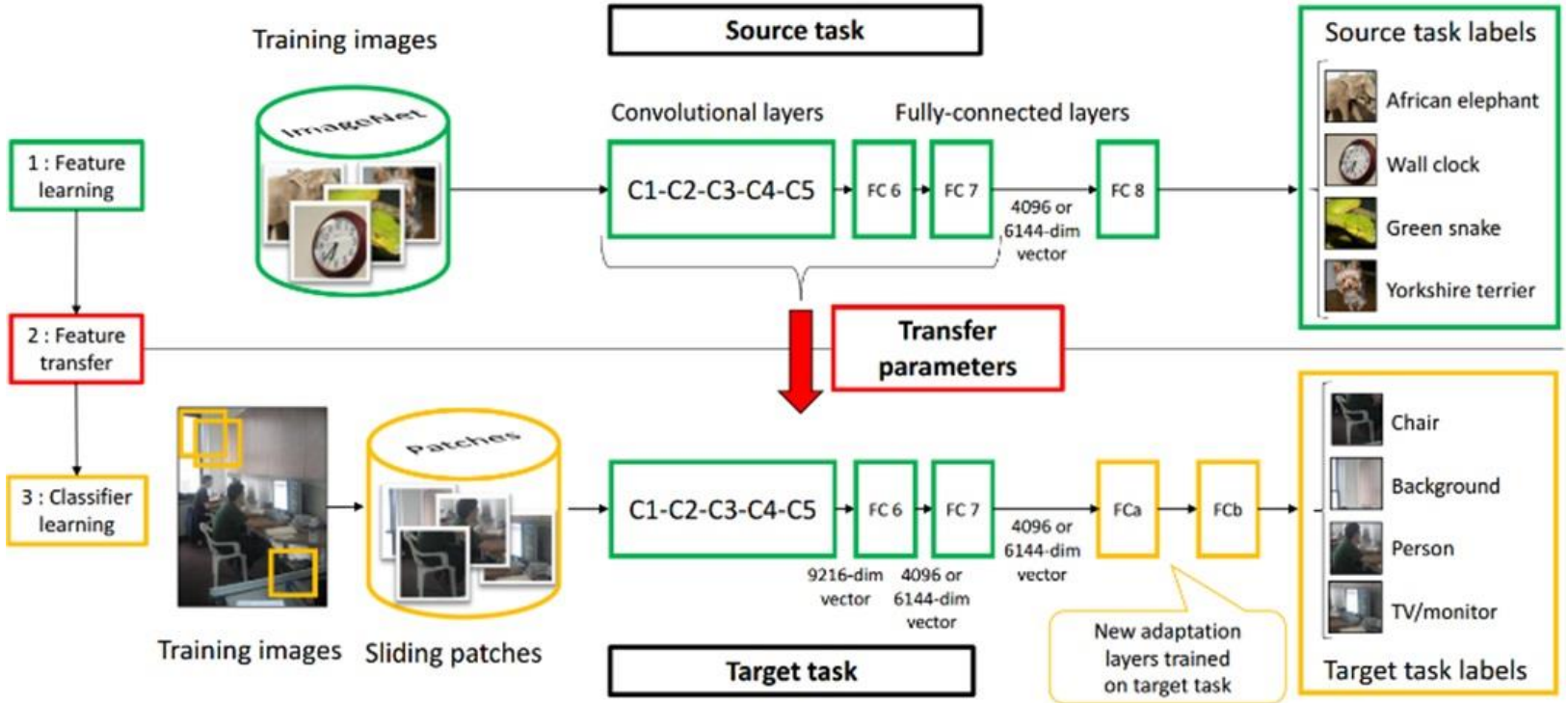
How to use a trained network for a new task?

- Networks trained on ImageNet learn general features that can be used across image domains
  - Take the vector of activations from one of the fully connected (FC) layers and treat it as an off-the-shelf feature
  - Train a new classifier layer on top of the FC layer
  - Fine-tune the whole network



A framework for 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

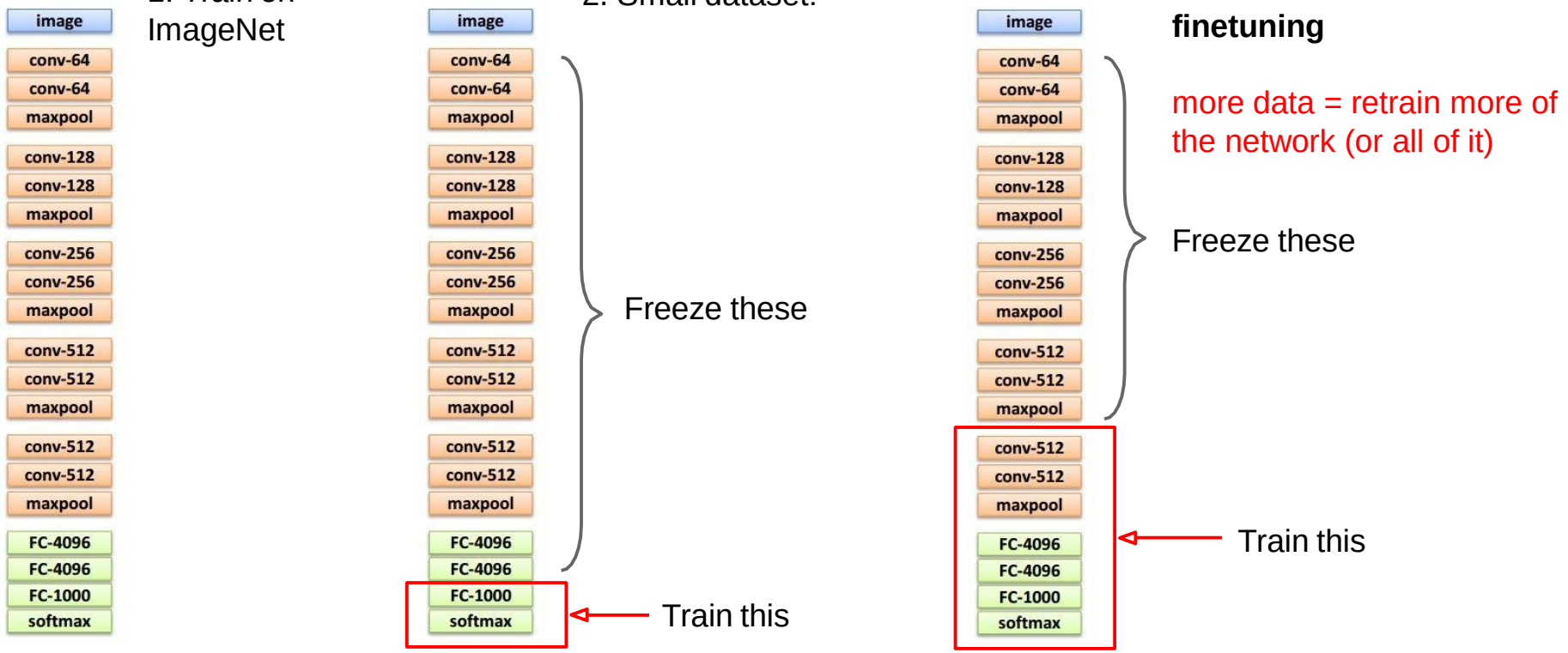


Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks [Oquab et al. CVPR 2014]

Transfer Learning with CNNs

Source: classification on ImageNet

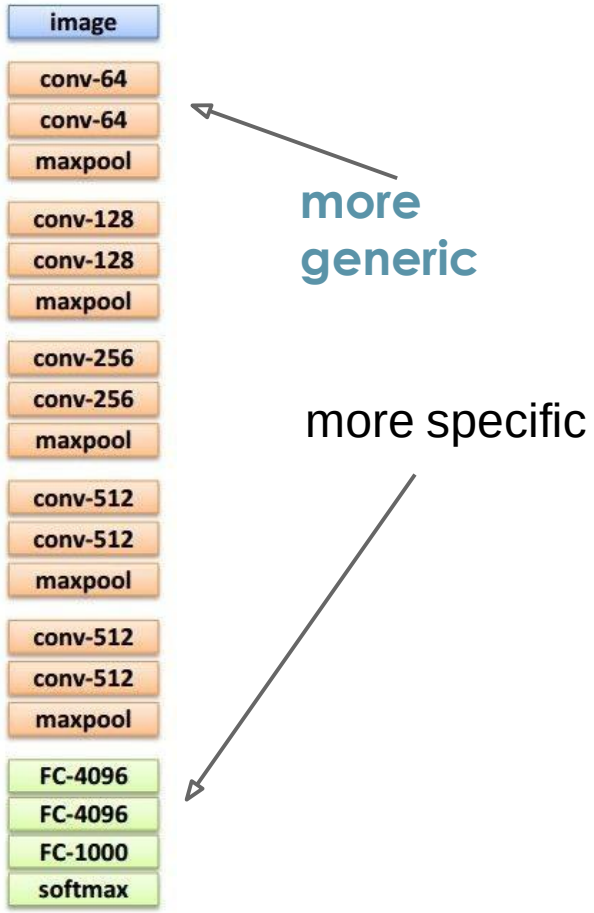
Target: some other task/data



Another option: use network as feature extractor, train SVM on extracted features for target task

Adapted from Andrej Karpathy

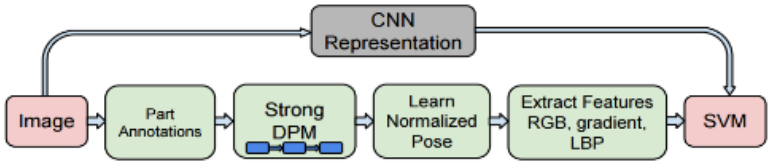
Transfer Learning with CNNs

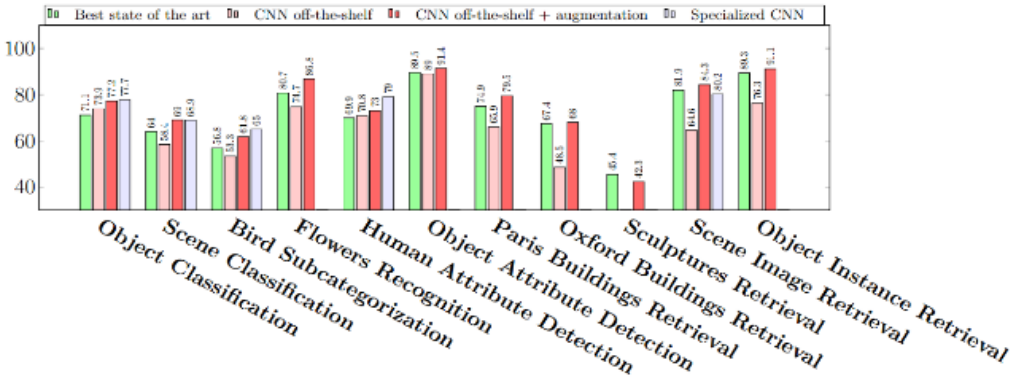


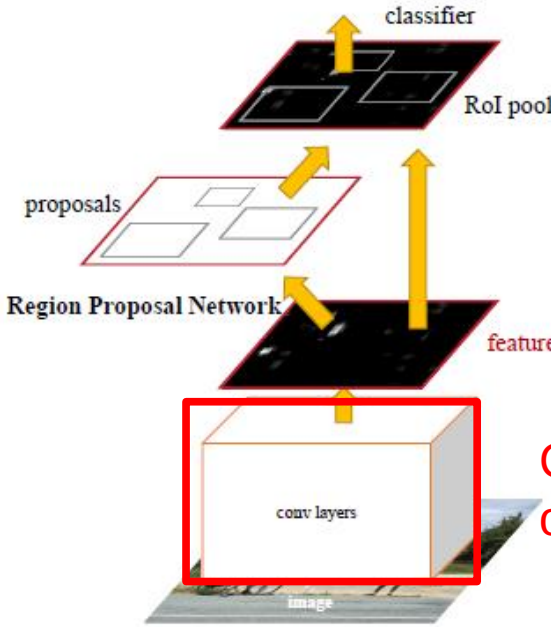
	very similar dataset	**very different dataset**
very little data	Use linear classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Andrej Karpathy

Transfer learning with CNNs is pervasive…



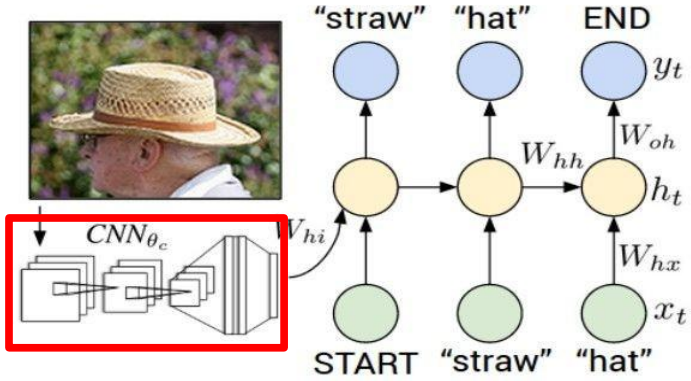




CNN pretrained on ImageNet

Object Detection
Ren et al., "Faster R-CNN", NIPS 2015

Image Captioning
Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015



CNN Features off-the-shelf:
an Astounding Baseline for Recognition
[Razavian et al. 2014]

Adapted from Andrej Karpathy

What are ConvNets good for

- **Signals that comes to you in the form of (multidimensional) arrays.**
- **Signals that have strong local correlations**
- **Signals where features can appear anywhere**
- **Signals in which objects are invariant to translations and distortions**

- **1D ConvNets: sequential signals, text**
  - Text Classification
  - Musical Genre Recognition
  - Acoustic Modeling for Speech Recognition
  - Time-Series Prediction
- **2D ConvNets: images, time-frequency representations (speech and audio)**
  - Object detection, localization, recognition
- **3D ConvNets: video, volumetric images, tomography images**
  - Video recognition / understanding
  - Biomedical image analysis
  - Hyperspectral image analysis

Resources

- http://deeplearning.net/
- https://github.com/ChristosChristofidis/awesome-deep-learning
- https://github.com/terryum/awesome-deep-learning-papers
- https://d2l.ai/index.html

Reading list (incomplete)

- 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
- M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014
- 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
- G. Huang, Z. Liu, and L. van der Maaten, Densely Connected Convolutional Networks, CVPR 2017

What we have learned today

- Overview
  - Neuroscience, Perceptron, multi-layer neural networks
- Convolutional neural network (CNN)
  - Convolution, nonlinearity, max pooling
  - CNN for classification and beyond
- Understanding and visualizing CNN
  - Find images that maximize some class scores; visualize individual neuron activation, input pattern and images; breaking CNNs
- Training CNN
  - Dropout; data augmentation; batch normalization; transfer learning