Vision and Cognitive Systems

SCQ1097939 - LM CS,DS,CYB,PD

Convolutional Neural Networks for Visual Recognition

Prof. Lamberto Ballan

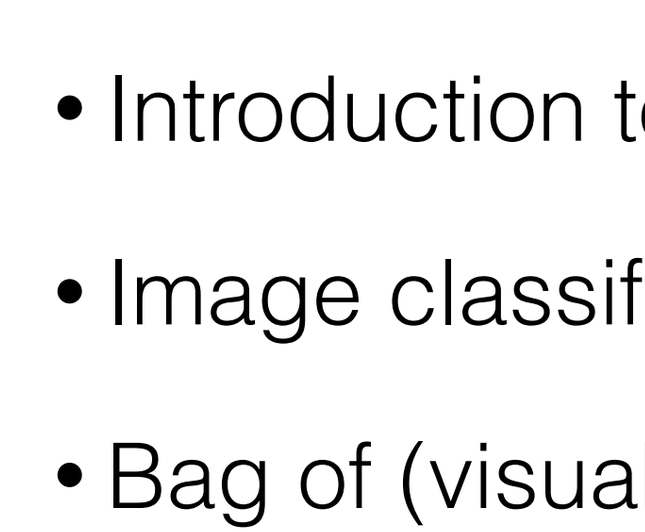
University of Padova, A.Y. 2023/24 November 27, 2023

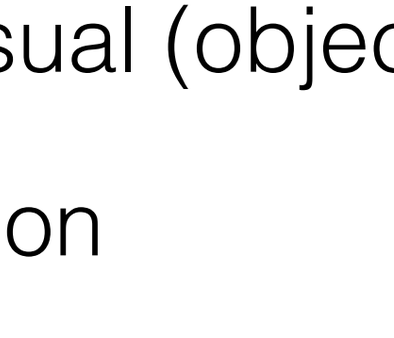
DIPARTIMENTO MATEMATICA
Department of Mathematics “Tullio Levi-Civita”

What we learned until now

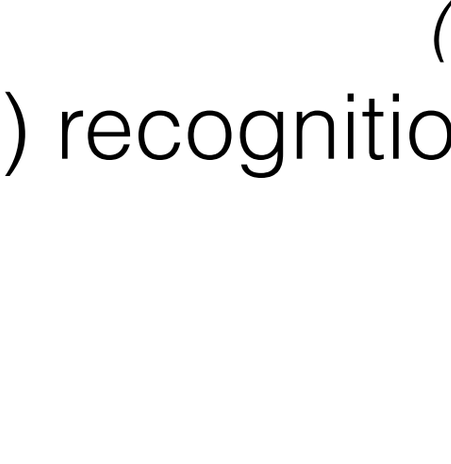
(a brief summary)

- Introduction to visual (object) recognition
- Image classification
- Bag of (visual) words
- Spatial pyramids

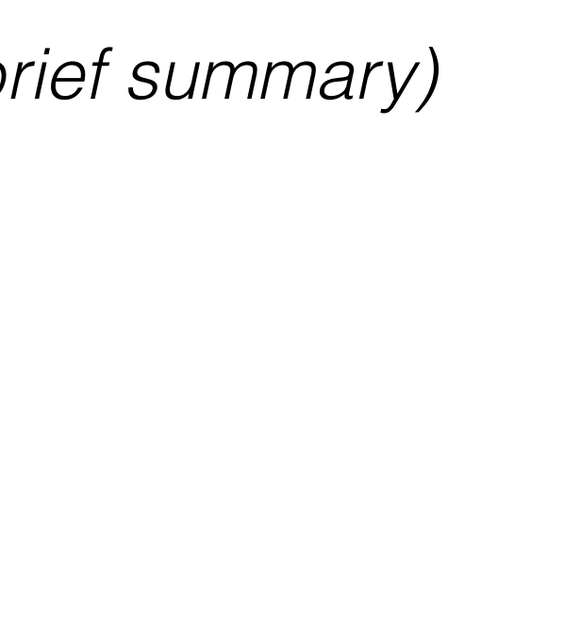




Input image



Bag of image patches



Feature space

Bag of features: pipeline

- Feature extraction:

- Codebook formation:

- Quantize features using visual vocabulary

- Represent images by frequencies of “visual words”:

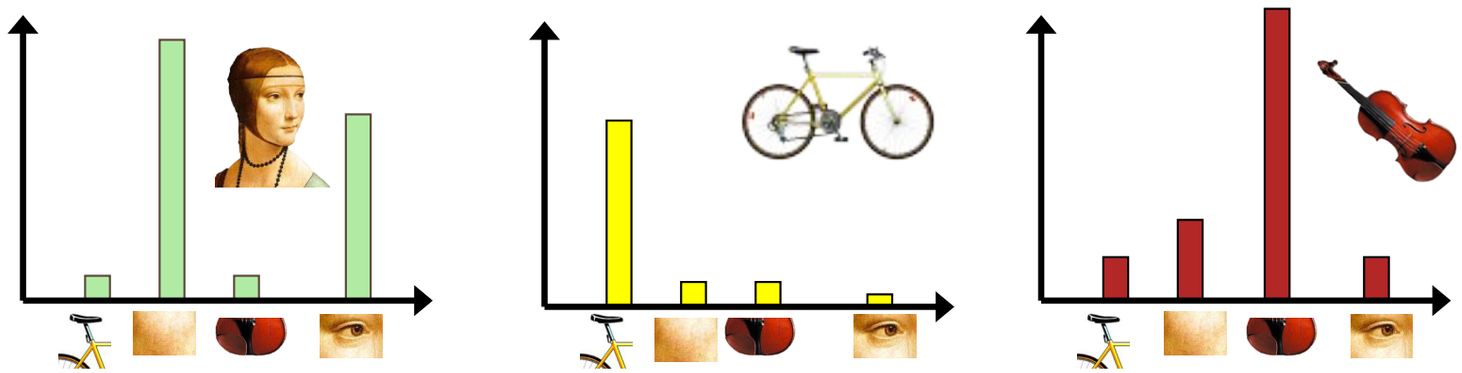


Image classification

- We need two major “ingredients”:
  - A way to *describe* images (e.g bow-histograms)
  - A procedure to *compare* images and *learn* a statistical model for a class (i.e. a classifier: kNN, SVM, etc.)

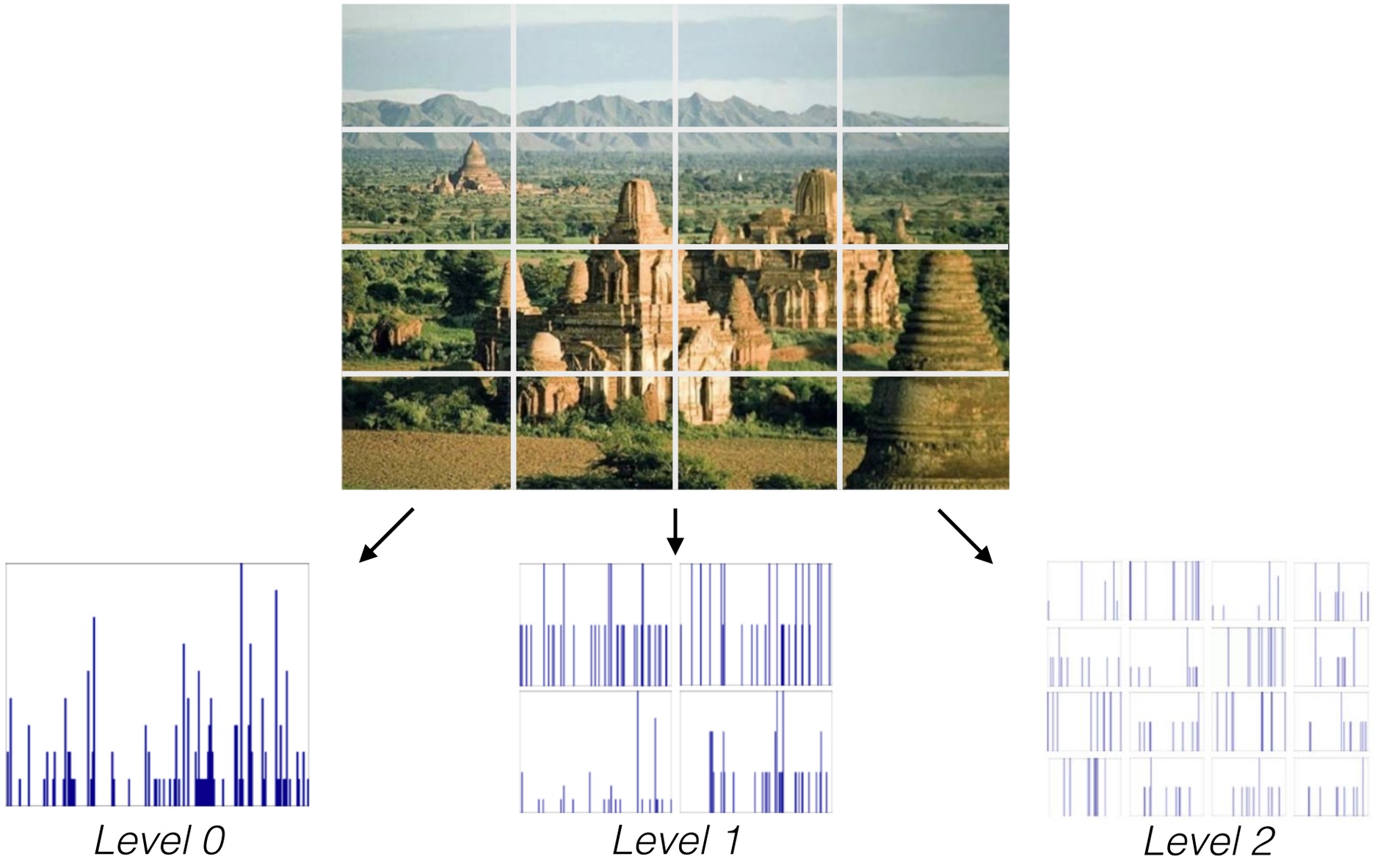
Image

img_feat: $[c_1,\ldots,c_N]$

bird

Bag of features++: spatial pyramids

- Intuition: locally orderless representation at several levels of spatial resolution



Slide credit: S. Lazebnik

What we will learn today?

- From handcrafted features to representation learning
- Intro to Convolutional Neural Networks
- Conv, Pool and Fully Connected layers; a closer look at spatial dimensions

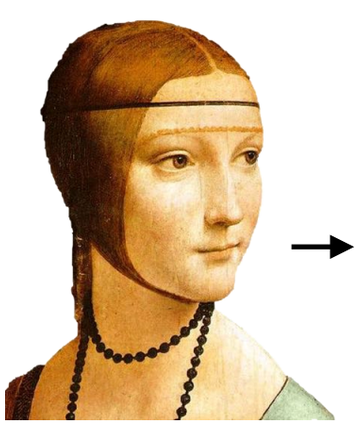
Several slides adapted from Stanford CS231n by L. Fei-Fei, A. Karpathy, J. Johnson, S. Yeung

Resources and background material:

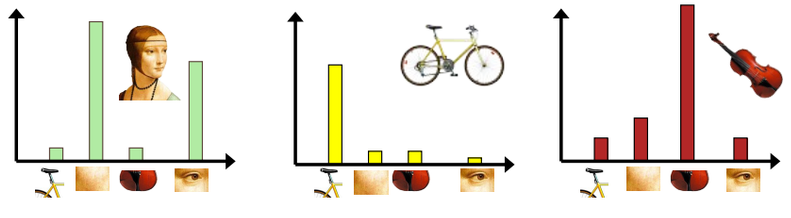
- **http://cs231n.stanford.edu/** (slides, videos)
- **http://cs231n.github.io/** (notes and tutorials)

From handcrafted representations to…

- Since now we have seen hand-crafted feature representations
- Let's take a look (again) at our bow pipeline:



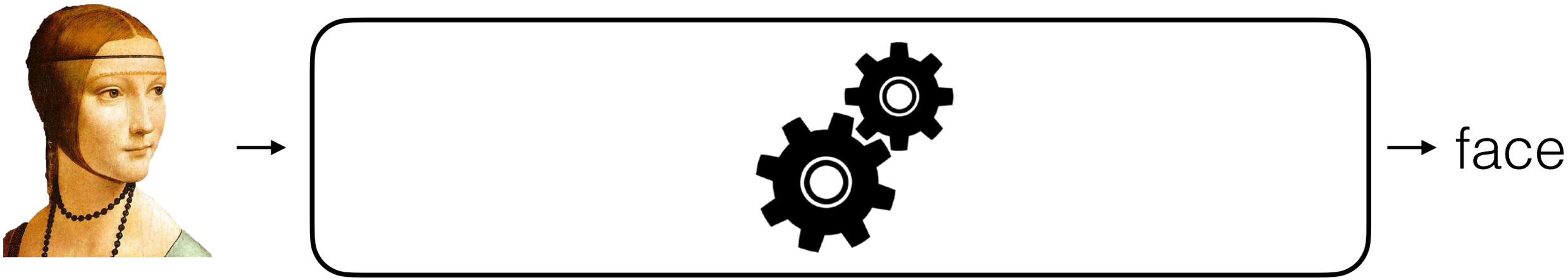
Feature Extraction → Codebook Formation → Feature Quantization (pooling) → Trainable Classifier → face



Representation learning

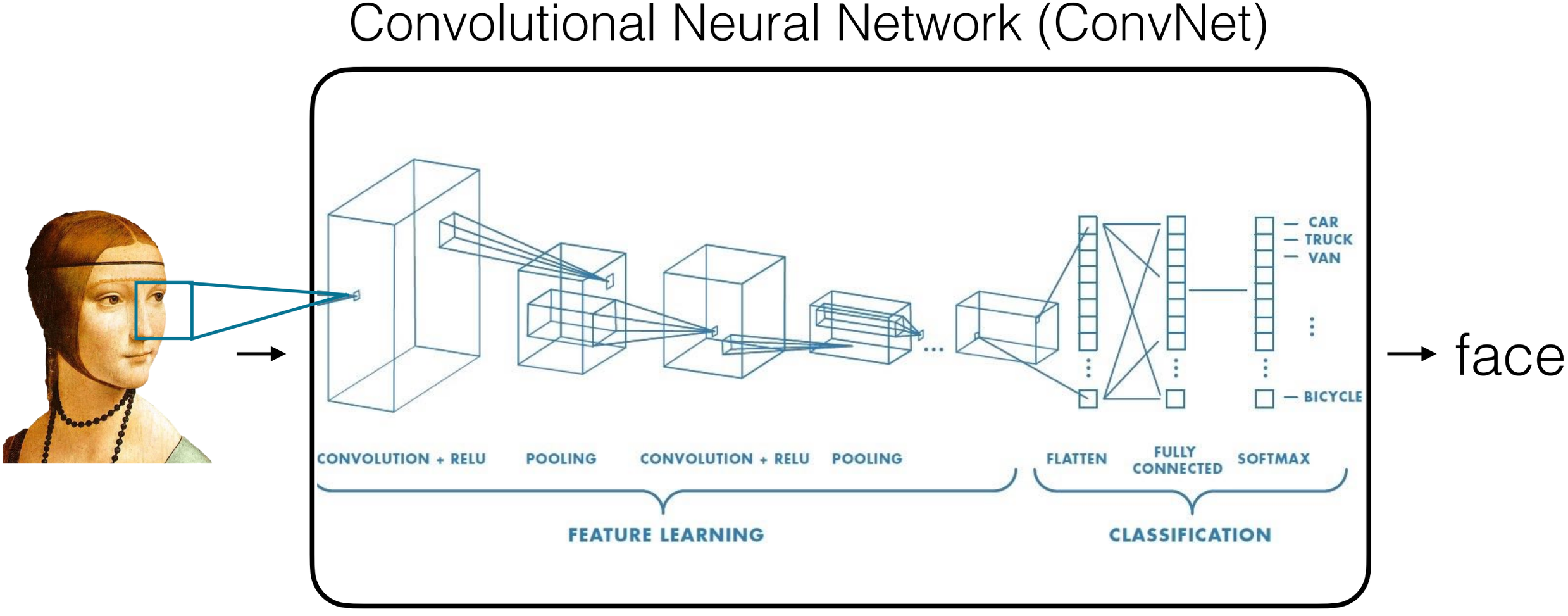
- Intuition: learn also the (image) representation directly from the data (*end-to-end learning)*

Deep Learning can be summarized as learning both the representation and the classifier out of data



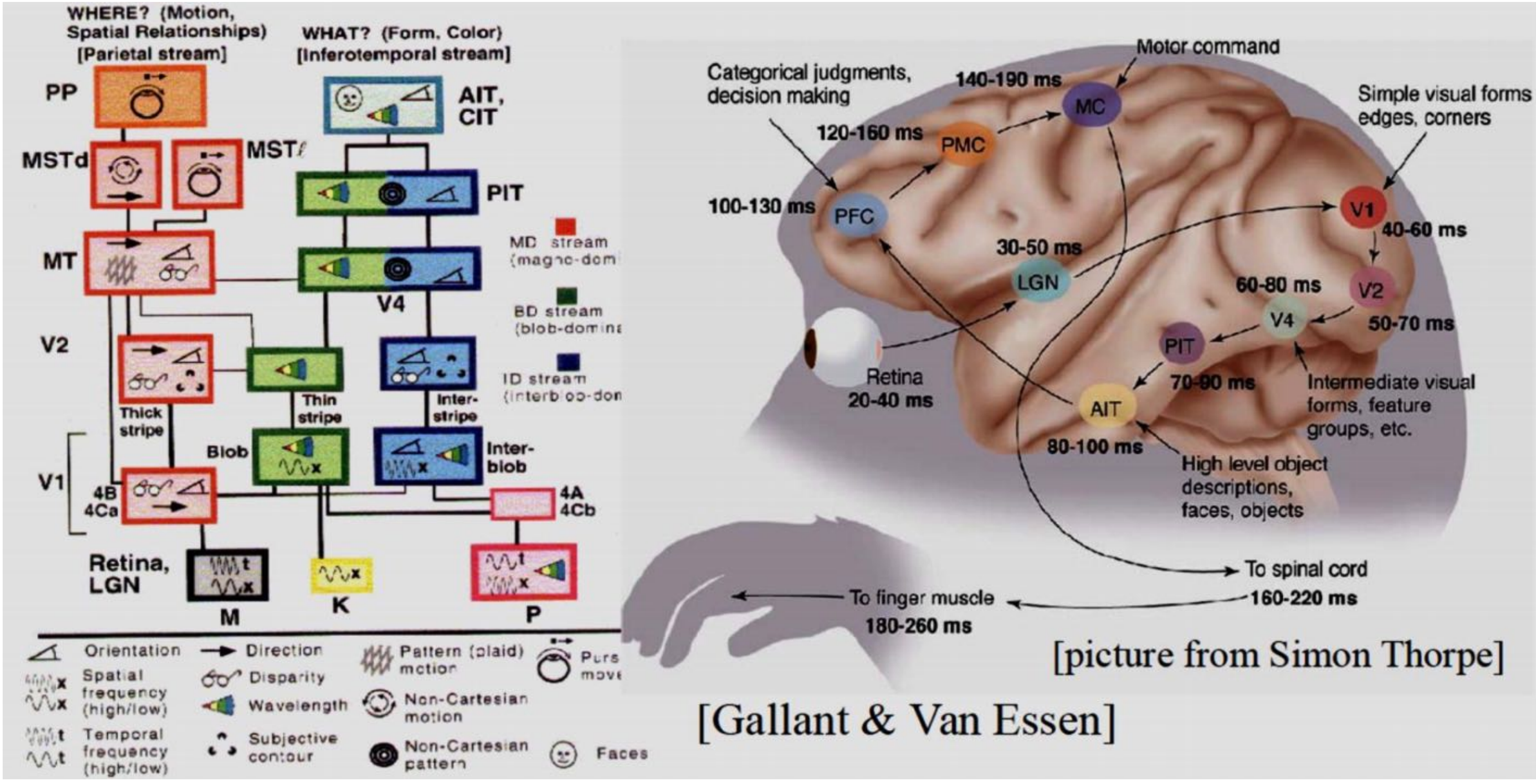
Representation learning

- Intuition: learn also the (image) representation directly from the data



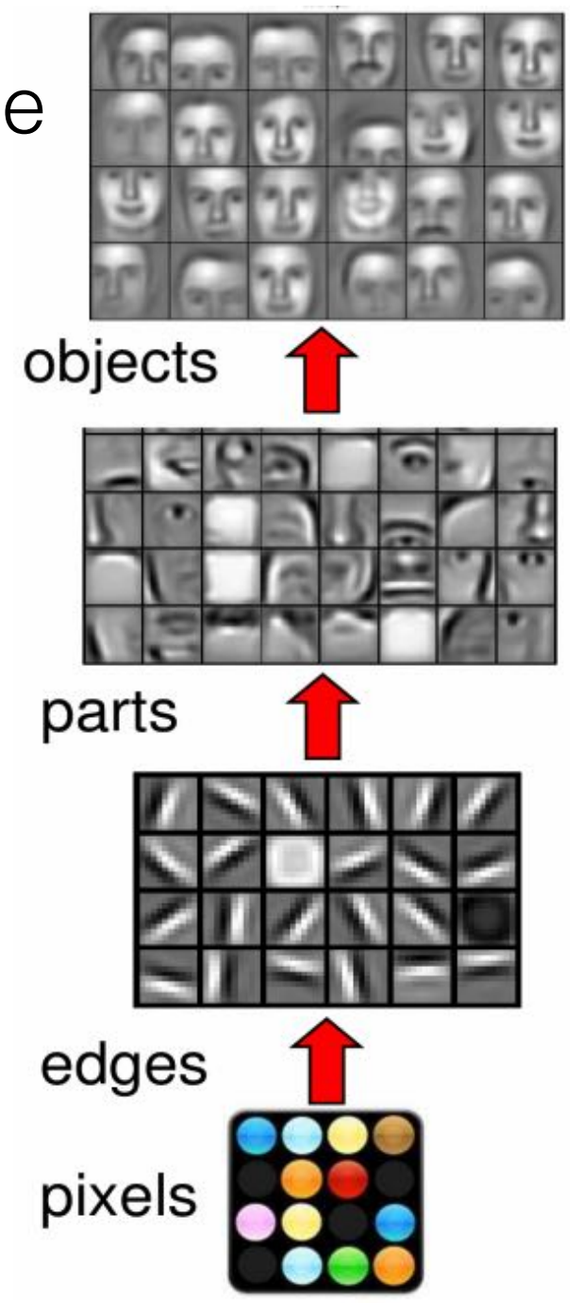
Inspiration / Biological analogy

- The ventral (recognition) pathway in the visual cortex has multiple stages



Representation learning (in broader terms)

- Learning the representation is a challenging problem for AI, Neuroscience, Cognitive Science
- Cognitive perspective:
  - How can a perceptual system build itself by looking at the external world?
  - How much prior "structure" is needed?
- Neuroscience perspective:
  - Does the cortex run a single general learning algorithm or multiple simpler ones?
- AI/ML perspective:
  - What is the fundamental principle?
  - What are the learning algorithm and architecture?



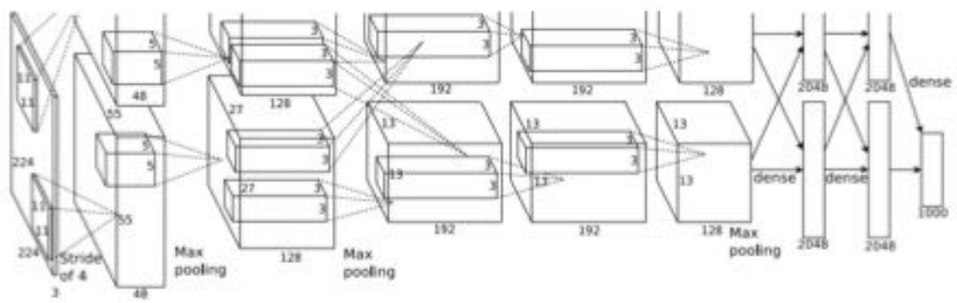
Slide credit: Y. LeCun

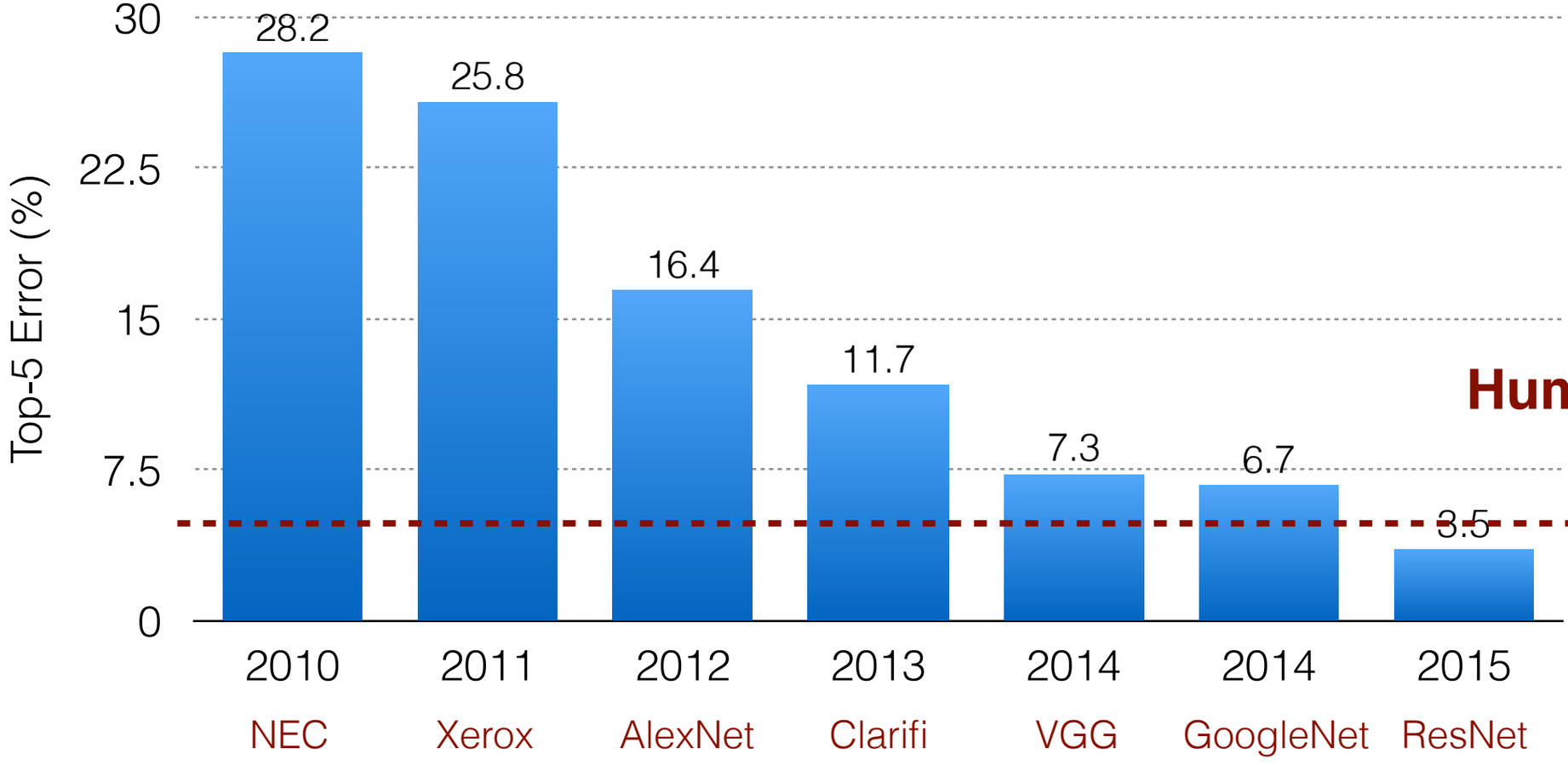
Neural networks are cool again

- Datasets drive computer vision progress

Neural networks are cool again

- ImageNet-ILSVRC (classification) over the years





Neural networks are cool again

NNs had an hard time in 2000s/early 2010s…

CVPR 2012 submission

Then an ICML 2012 paper:
https://arxiv.org/abs/1202.2160

Hi Serge,

We decided to withdraw our paper #[ID no.] from CVPR "[Paper Title]" by [Author Name] et al.

We posted it on ArXiv: http://arxiv.org/ [Paper ID] .

We are withdrawing it for three reasons: 1) the scores are so low, and the reviews so ridiculous, that I don't know how to begin writing a rebuttal without insulting the reviewers; 2) we prefer to submit the paper to ICMLwhere it might be better received; 3) with all the fuss I made, leaving the paper in would have looked like I might have tried to bully the program committee into giving it special treatment.

Getting papers about feature learning accepted at vision conference has always been a struggle, and I've had more than my share of bad reviews over the years. Thankfully, quite a few of my papers were rescued by area chairs.

This time though, the reviewers were particularly clueless, or negatively biased, or both. I was very sure that this paper was going to get good reviews because: 1) it has two simple and generally applicable ideas for segmentation ("purity tree" and "optimal cover"); 2) it uses no hand-crafted features (it's all learned all the way through. Incredibly, this was seen as a negative point by the reviewers!); 3) it beats all published results on 3 standard datasets for scene parsing; 4) it's an order of magnitude faster than the competing methods.

If that is not enough to get good reviews, I just don't know what is.

So, I'm giving up on submitting to computer vision conferences altogether. CV reviewers are just too likely to be clueless or hostile towards our brand of methods. Submitting our papers is just a waste of everyone's time (and incredibly demoralizing to my lab members)

I might come back in a few years, if at least two things change:

- Enough people in CV become interested in feature learning that the probability of getting a non-clueless and non-hostile reviewer is more than 50% (hopefully [Computer Vision Researcher]'s tutorial on the topic at CVPR will have some positive effect).

- CV conference proceedings become open access.

We intent to resubmit the paper to ICML, where we hope that it will fall in the hands of more informed and less negatively biased reviewers (not that ML reviewers are generally more informed or less biased, but they are just more informed about our kind of stuff). Regardless, I actually have a keynote talk at [Machine Learning Conference], where I'll be talking about the results in this paper.

Be assured that I am not blaming any of this on you as the CVPR program chair. I know you are doing your best within the traditional framework of CVPR.

I may also submit again to CV conferences if the reviewing process is fundamentally reformed so that papers are published before they get reviewed.

You are welcome to forward this message to whoever you want.

I hope to see you at NIPS or ICML.

Cheers,

- [Author]

Neural networks are cool again

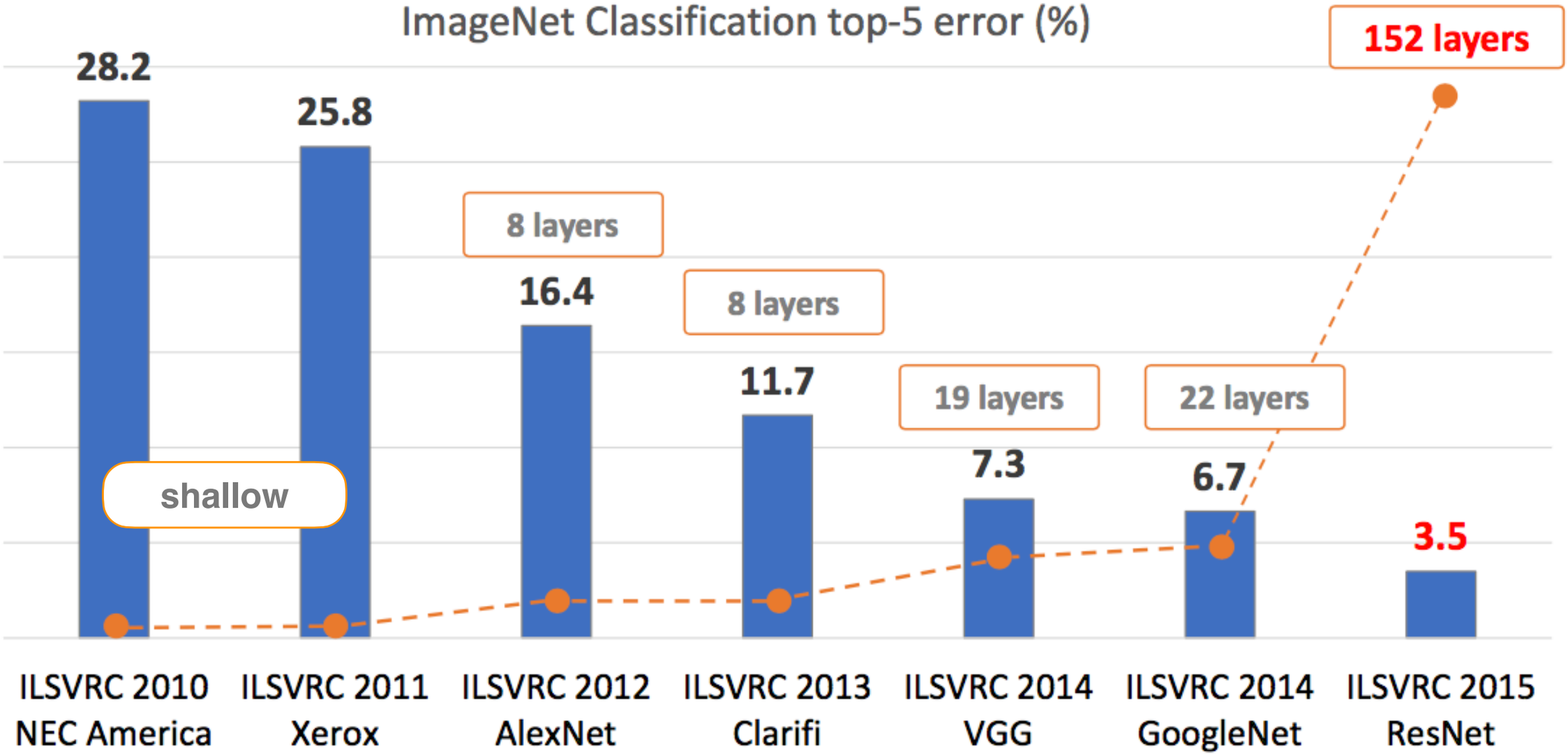
- The “ImageNet moment” in 2012

IMAGENET

ILSVRC’12: https://www.image-net.org/challenges/LSVRC/2012/

The rise of deep learning

- Revolution of depth: how deep is deep?



Scale drive deep learning progress

Performance

This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Revisiting Unreasonable Effectiveness of Data in Deep Learning Era

Chen Sun[1], Abhinav Shrivastava[1,2], Saurabh Singh[1], and Abhinav Gupta[1,2]

[1]Google Research
[2]Carnegie Mellon University

Abstract

The success of deep learning in vision can be attributed to: (a) models with high capacity; (b) increased computational power; and (c) availability of large-scale labeled data. Since 2012, there have been significant advances in representation capabilities of the models and computational capabilities of GPUs. But the size of the biggest dataset has surprisingly remained constant. What will happen if we increase the dataset size by 10× or 100×? This paper takes a step towards clearing the clouds of mystery surrounding the relationship between 'enormous data' and visual deep learning. By exploiting the JFT-300M dataset which has more than 375M noisy labels for 300M images, we investigate how the performance of current vision tasks would change if this data was used for representation learning. Our paper delivers some surprising (and some expected) findings. First, we find that the performance on vision tasks increases logarithmically based on volume of training data size. Second, we show that representation learning (or pre-training) still holds a lot of promise. One can improve performance on many vision tasks by just training a better base model. Finally, as expected, we present new state-of-the-art results for different vision tasks including image classification, object detection, semantic segmentation and human pose estimation. Our sincere hope is that this inspires vision community to not undervalue the data and develop collective efforts in building larger datasets.

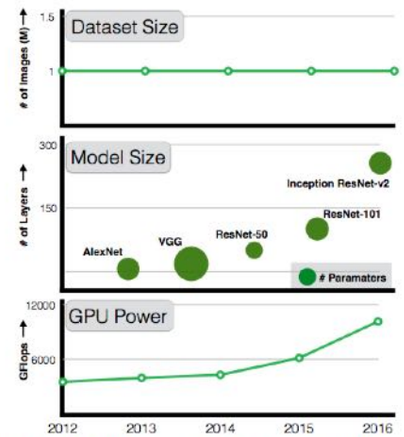


Figure 1. The Curious Case of Vision Datasets: While GPU computation power and model sizes have continued to increase over the last five years, size of the largest training dataset has surprisingly remained constant. Why is that? What would have happened if we have used our resources to increase dataset size as well? This paper provides a sneak-peek into what could be if the dataset sizes are increased dramatically.

ously, while both GPUs and model capacity have continued to grow, datasets to train these models have remained

works

tworks

rks

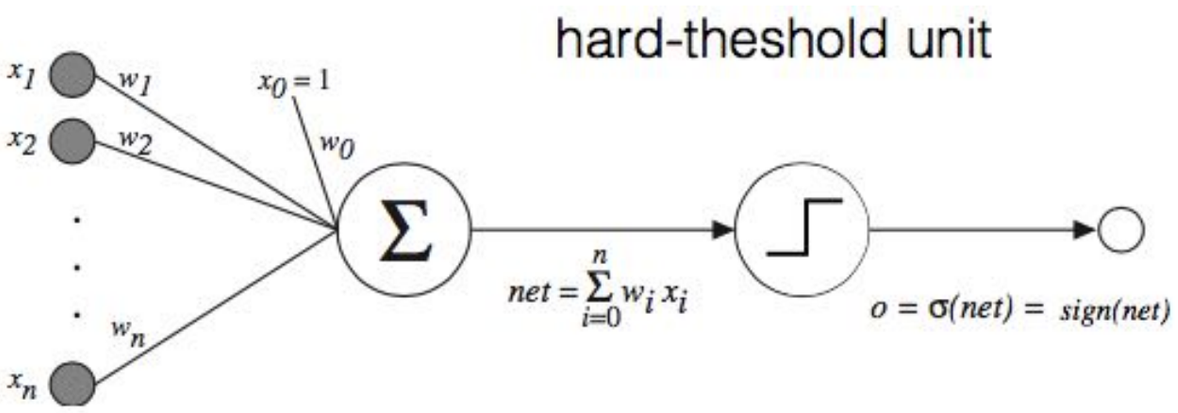
Pascal-VOC (2007)

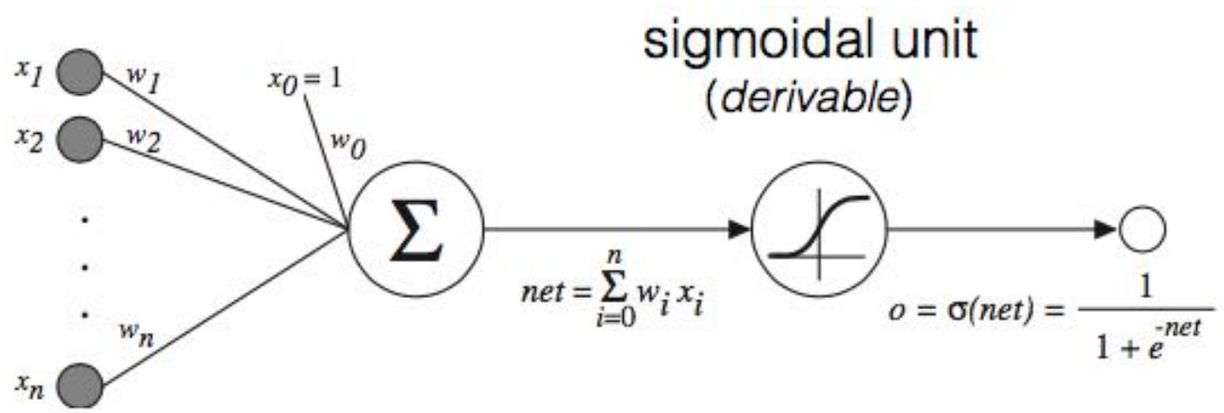
Caltech101 (2005)

Amount of data

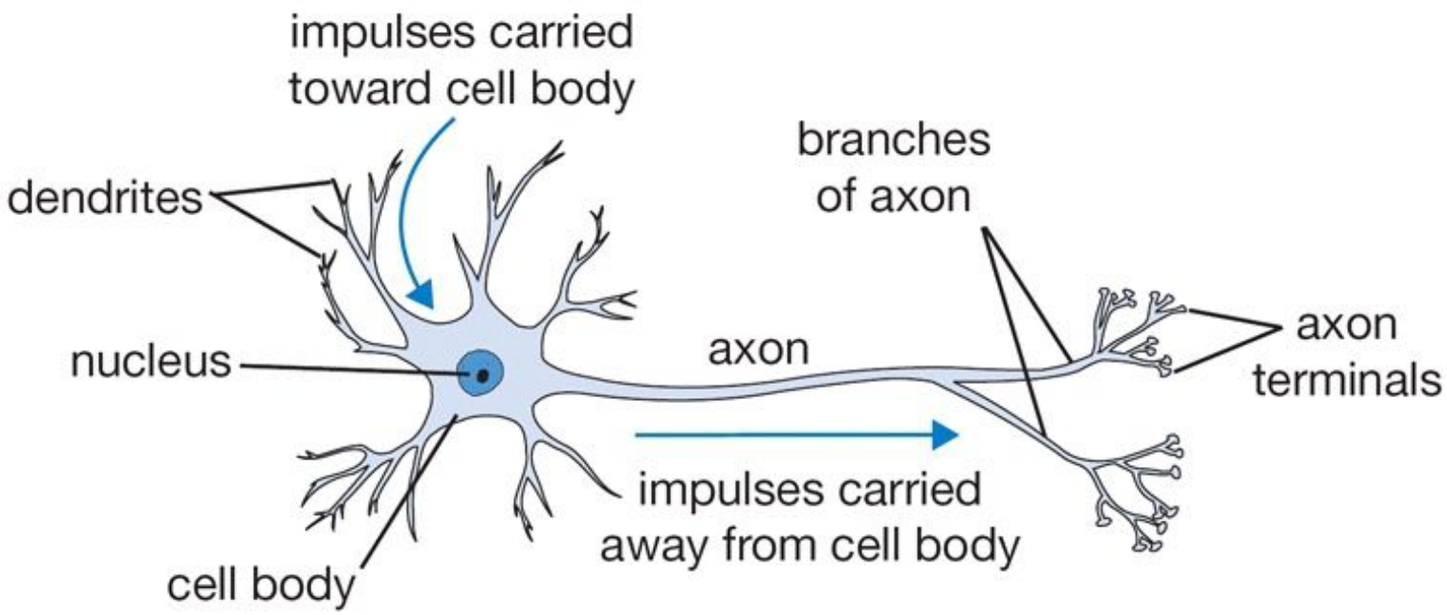
Neural Networks recap

- Artificial Neural Network: it is a system consisting of interconnected units that compute *nonlinear functions*
  - *input* units represent input variables
  - *output* units represent output variables
  - *hidden* units (if present) represent internal variables that codify (after learning) correlations among input and desired output variables
  - *weights* are associated to connections among units



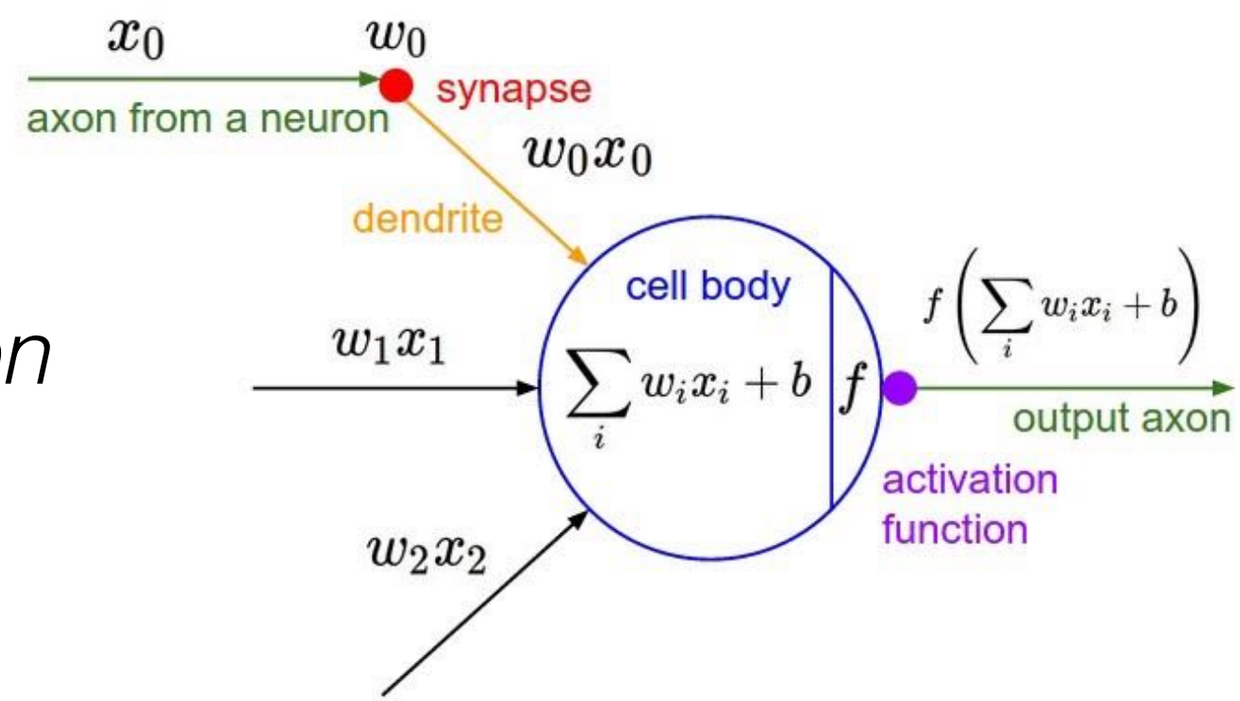


Neural Networks recap: biological analogy



Biological Neuron

A simple Artificial Neuron (perceptron)



e.g. sigmoid activation

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

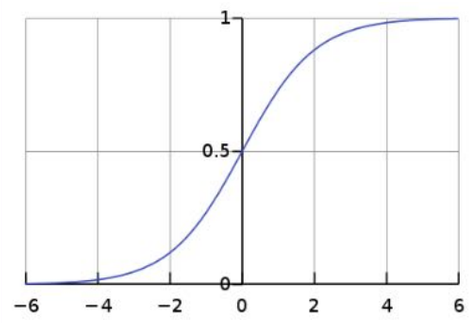


Image credit: Fei-Fei, Karpathy, Johnson

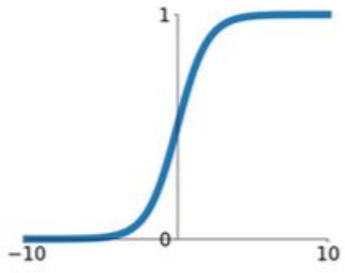
Neural Networks recap: activation functions

- **Feed-forward**: each neuron performs a dot product with the input and its weights, adds the bias and applies the activation function (or *non-linearity*)

- There are several activation functions you may encounter:

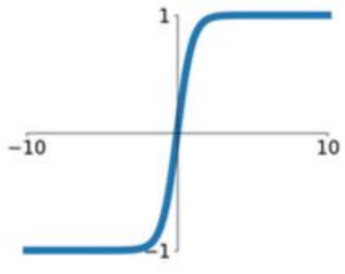
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



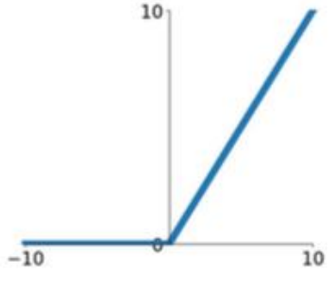
tanh

$$\tanh(x)$$



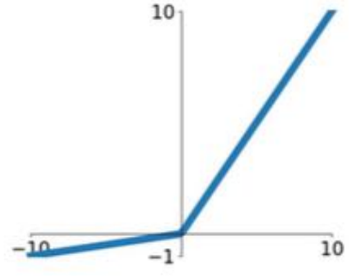
ReLU

$$\max(0, x)$$



Leaky ReLU

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

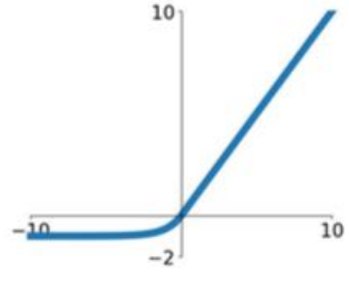


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

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

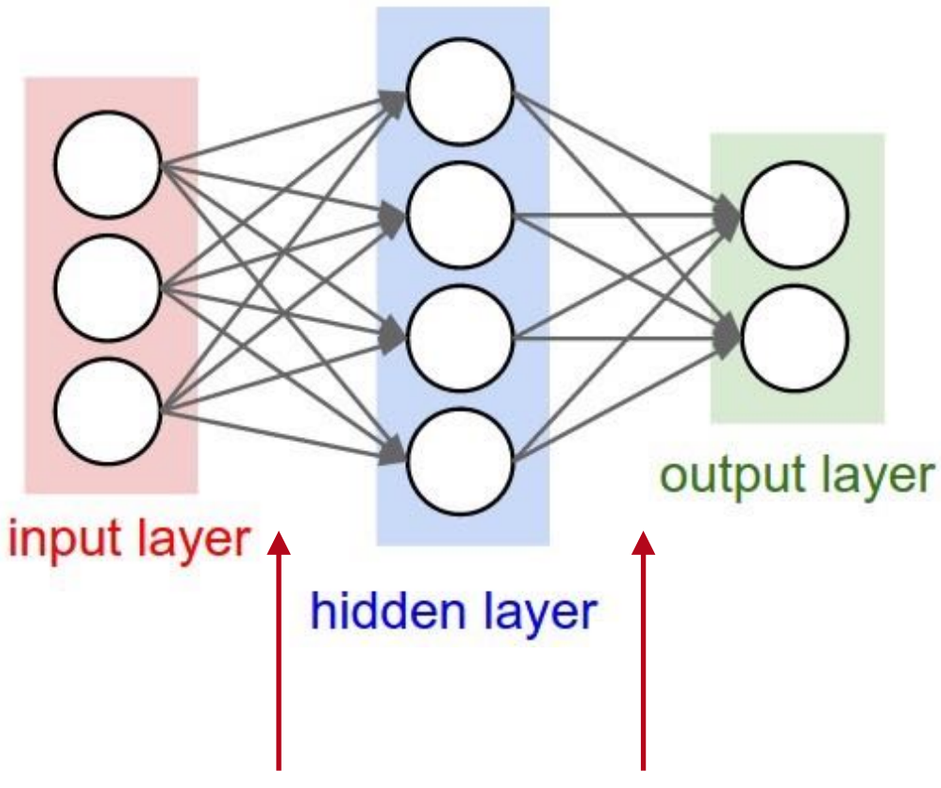


Neural Networks recap: architectures

- Note: when we say N-layer neural network, we do not count the input layer

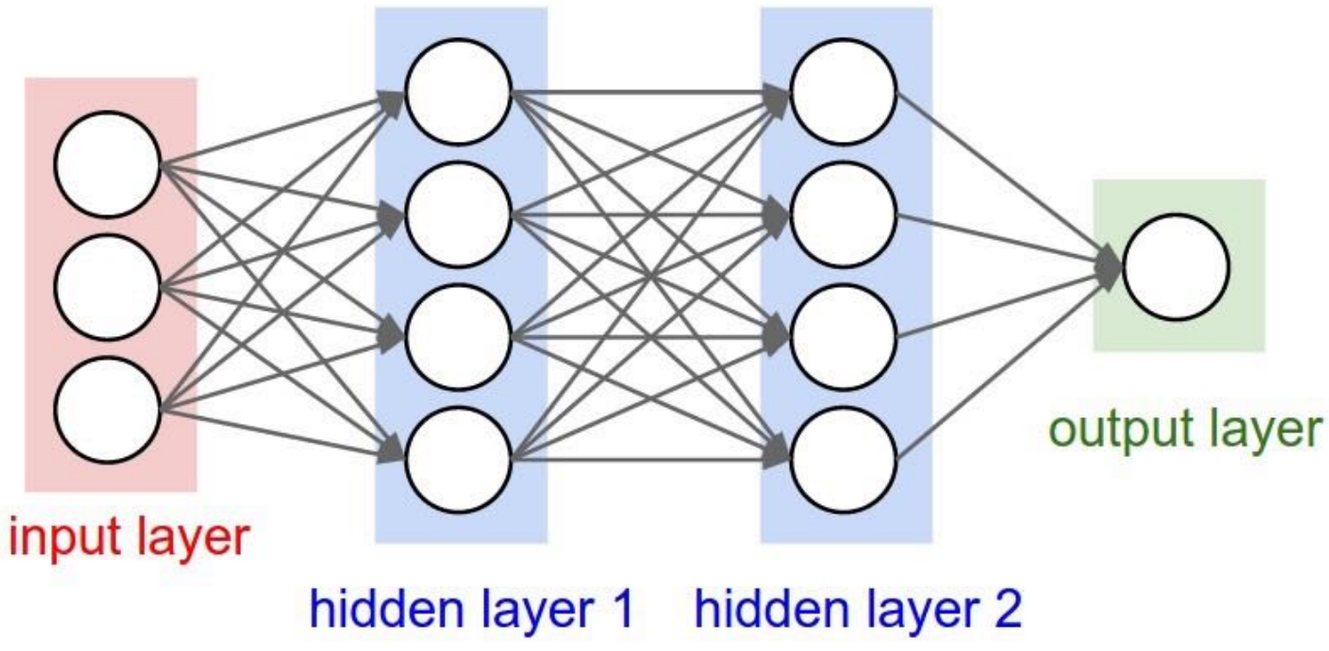
2-layer neural network

or 1-hidden-layer neural net



3-layer neural network

or 2-hidden-layer neural net



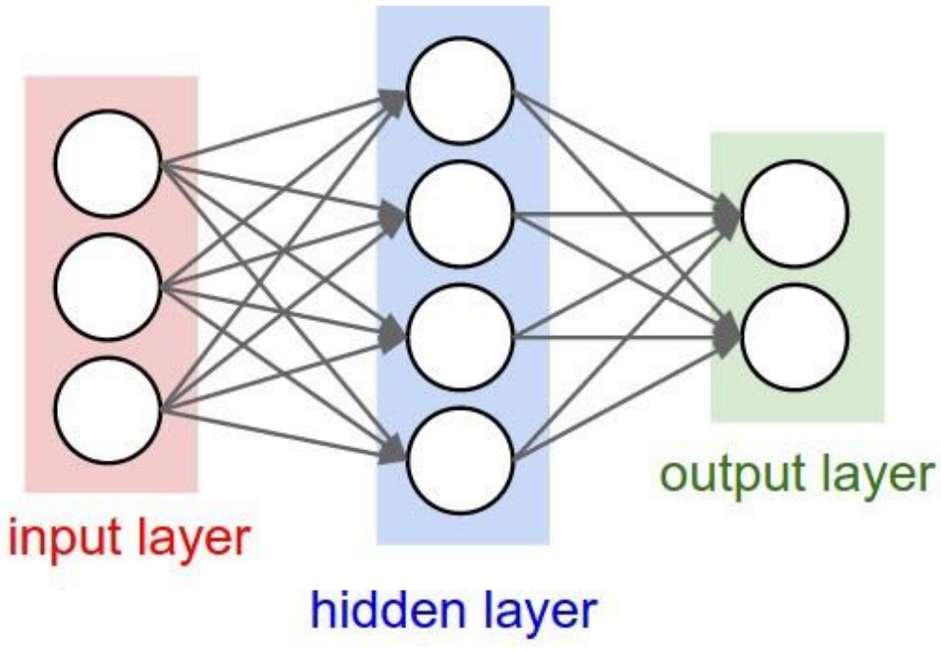
Fully Connected layers

Image credit: Fei-Fei, Karpathy, Johnson

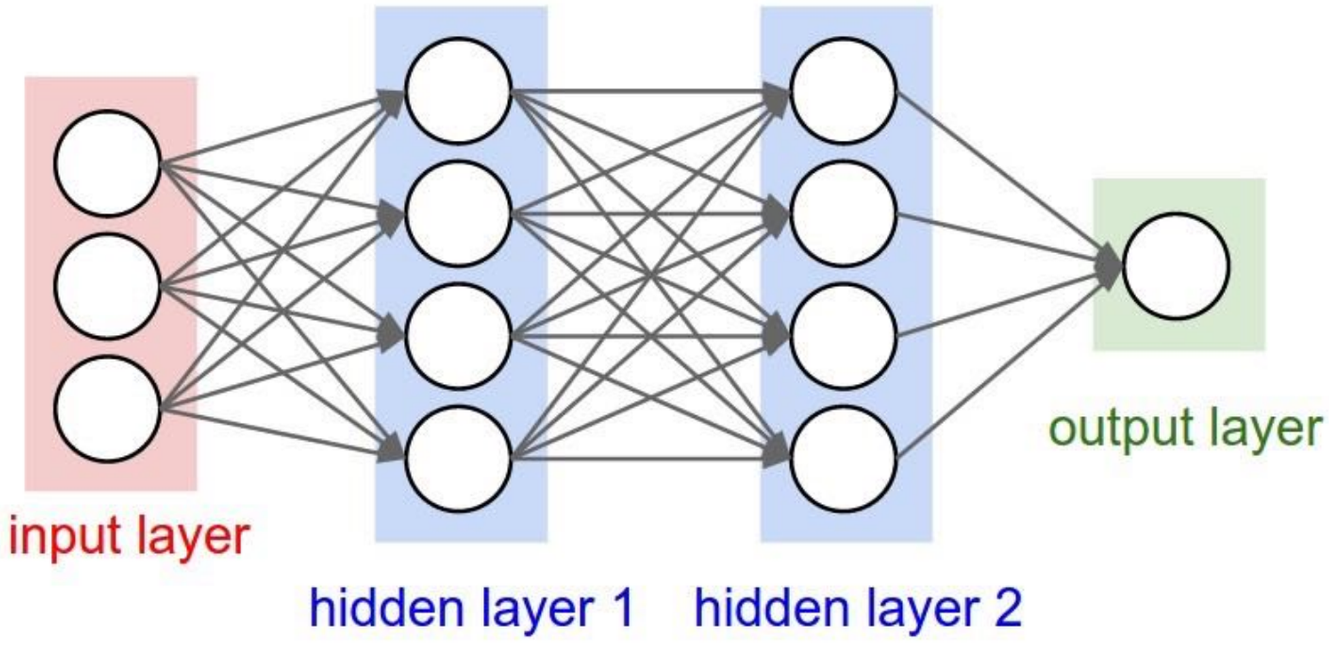
Neural Networks recap: architectures

- **Sizing neural networks:** the two metrics that are commonly used to measure the size of a NeuralNet are the #neurons or (more commonly) the #parameters

(a) **2-layer neural network**



(b) **3-layer neural network**

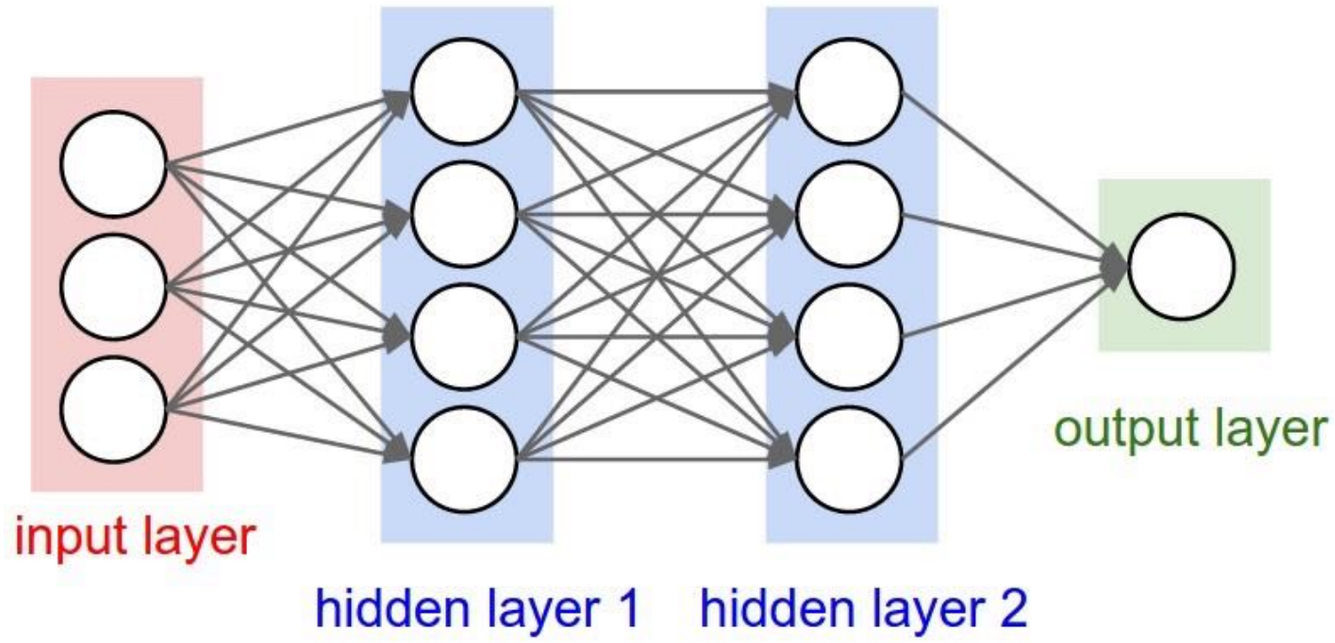


(b) has 4+4+1 neurons; how many (learnable) parameters?

(b) #parameters: 3x4 + 4x4 + 4x1 = 32 + 9 *(biases)* = 41

Neural Networks recap: computation

- Example feed-forward computation of a NeuralNet:



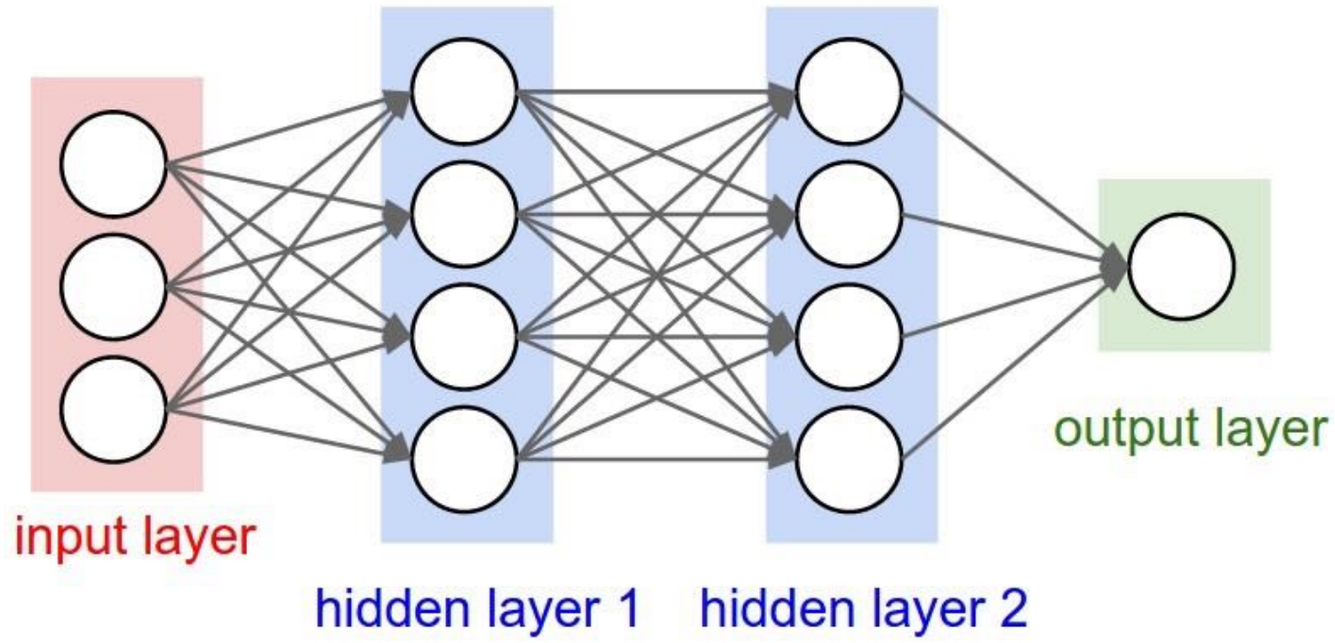
We can efficiently evaluate an entire layer of neurons

```
class Neuron:
  # ...
  def neuron_tick(inputs):
    """ assume inputs and weights are 1-D numpy arrays and bias is a number """
    cell_body_sum = np.sum(inputs * self.weights) + self.bias
    firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum)) # sigmoid activation function
    return firing_rate
```


Slide credit: Fei-Fei, Karpathy, Johnson

Neural Networks recap: computation

- Example feed-forward computation of a NeuralNet:



We can efficiently evaluate an entire layer of neurons

```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```


Slide credit: Fei-Fei, Karpathy, Johnson

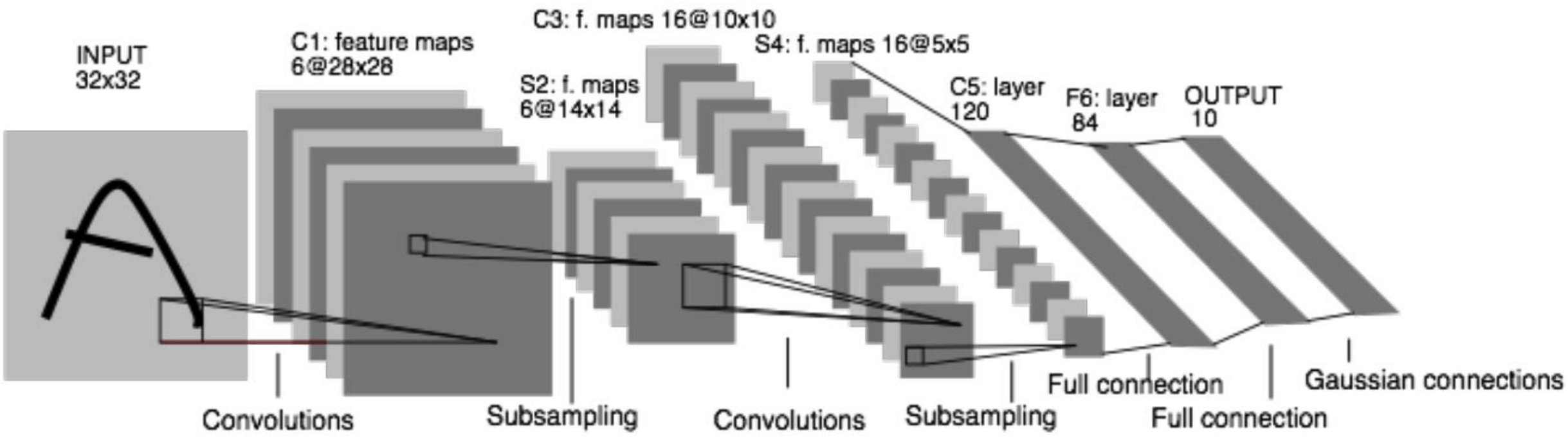
Neural Networks recap

- Architectures: we arrange neurons (units) into Fully Connected layers
  - Note: the abstraction of a layer has the nice property that it allows us to use efficient vectorized code (e.g. matrix multiplies)
- The forward pass of a FC layer corresponds to one matrix multiplication followed by a bias offset and an activation function
- Representational power: a neural network can approximate any continuous function

Convolutional Neural Networks

- CNNs: the assumption that the inputs are images allows us to encode certain local properties into the architecture

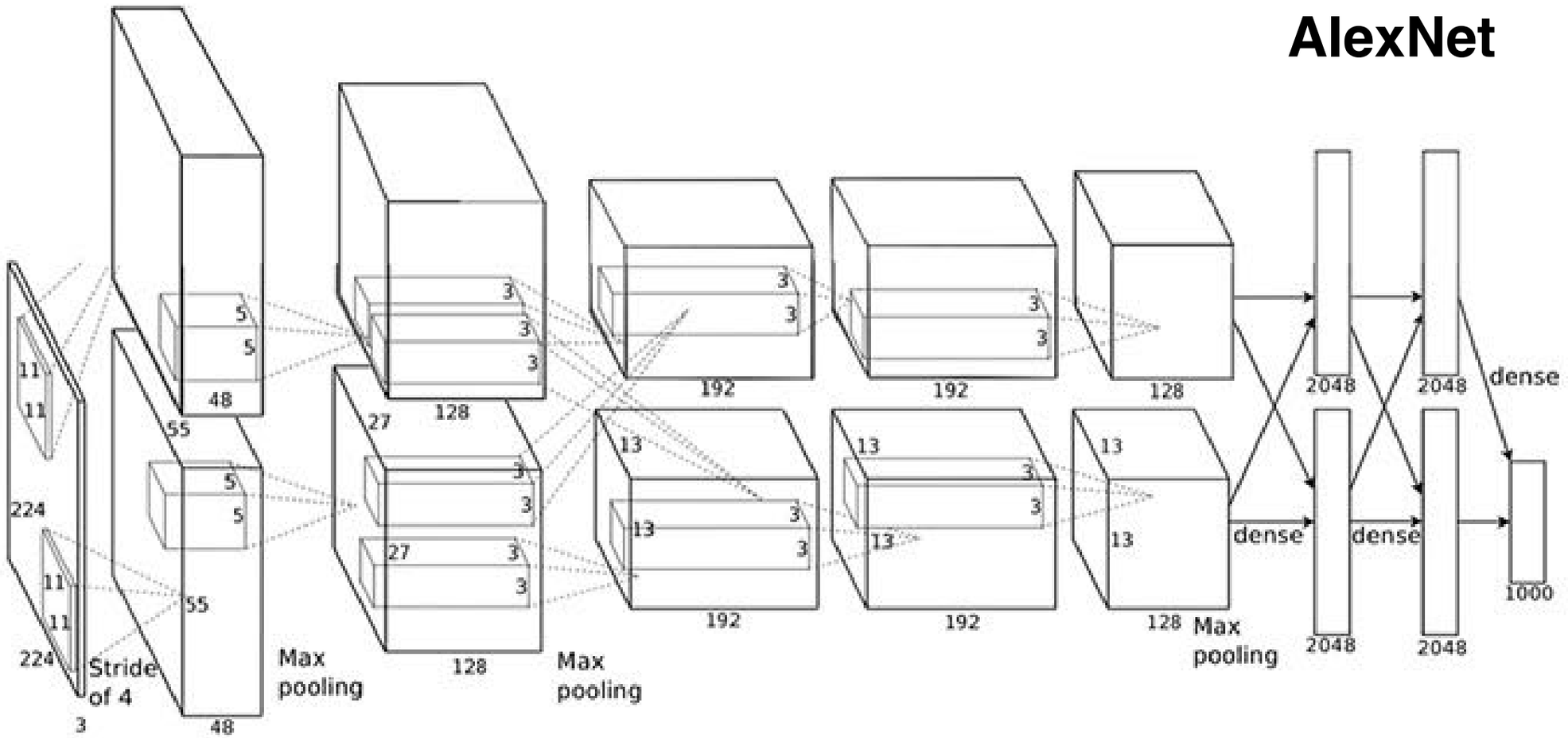
LeNet-5



Y.LeCun, L.Bottou, Y.Bengio, P.Haffner, “Gradient-based learning applied to document recognition”, Proc.IEEE 1998

Convolutional Neural Networks

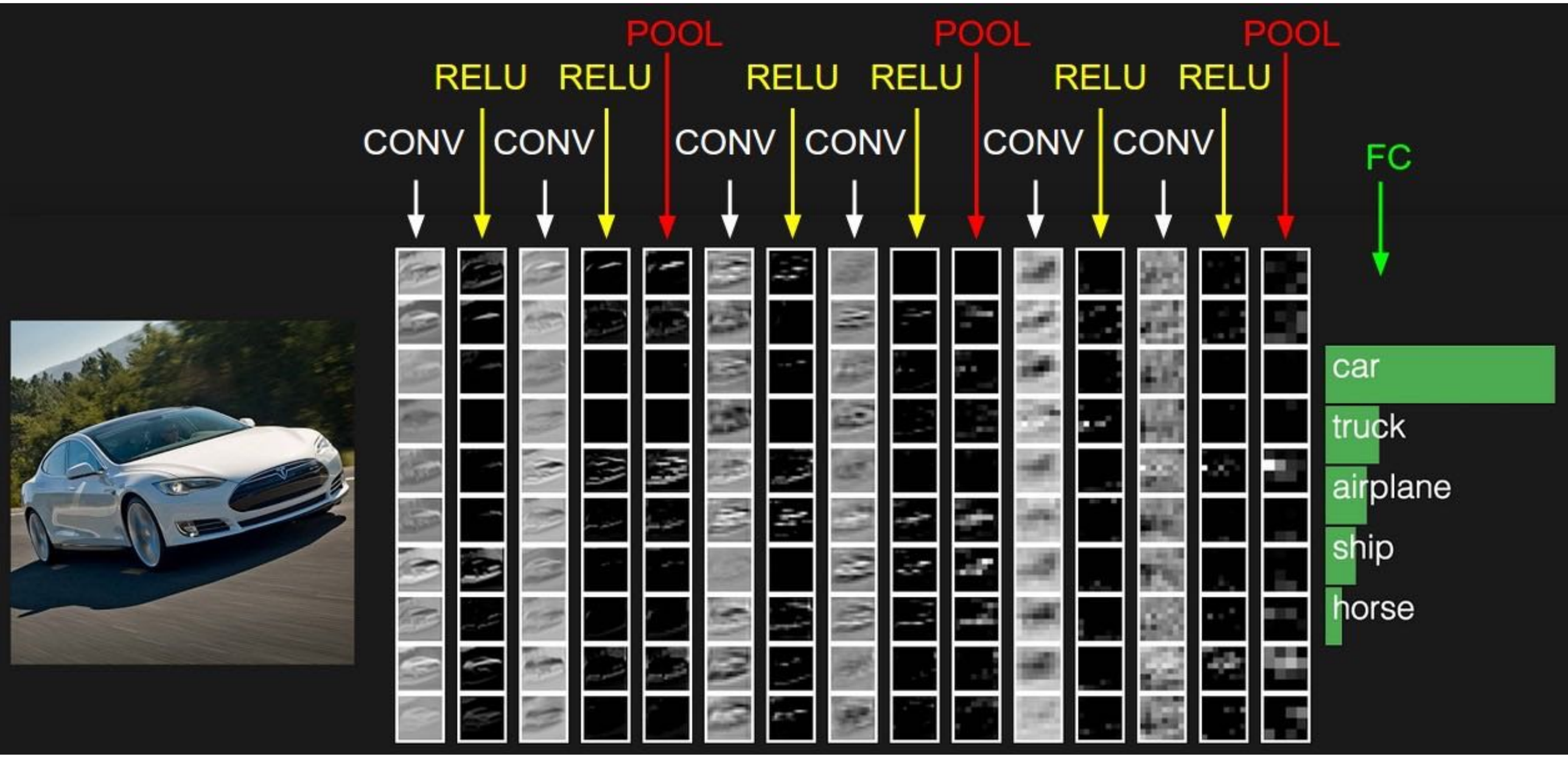
- Return of the CNN: first strong “modern” results



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

Convolutional Neural Networks

- A ConvNet is a sequence of layers that transform the input image (volume) to the final class scores

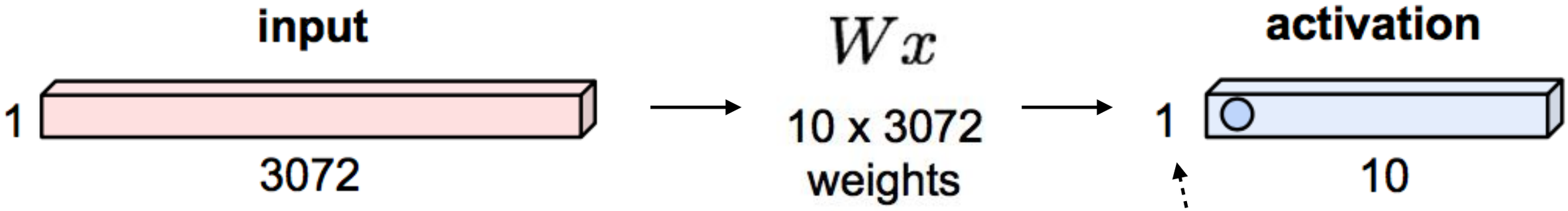


Convolutional Neural Networks

- ConvNet (CNN): a sequence of layers where each layer transforms one volume of activations to another through a *differentiable* function
- We have three main types of layers:
  - Fully Connected Layer (as in “regular” ANN)
  - Convolutional Layer
  - Pooling Layer
- We stack these layers to form a full CNN architecture

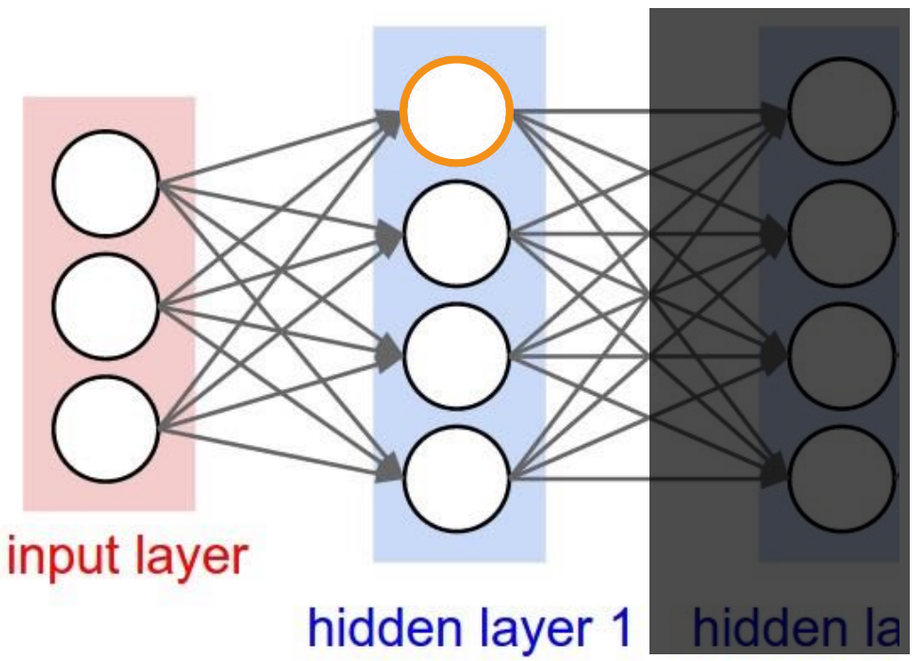
Fully Connected layer

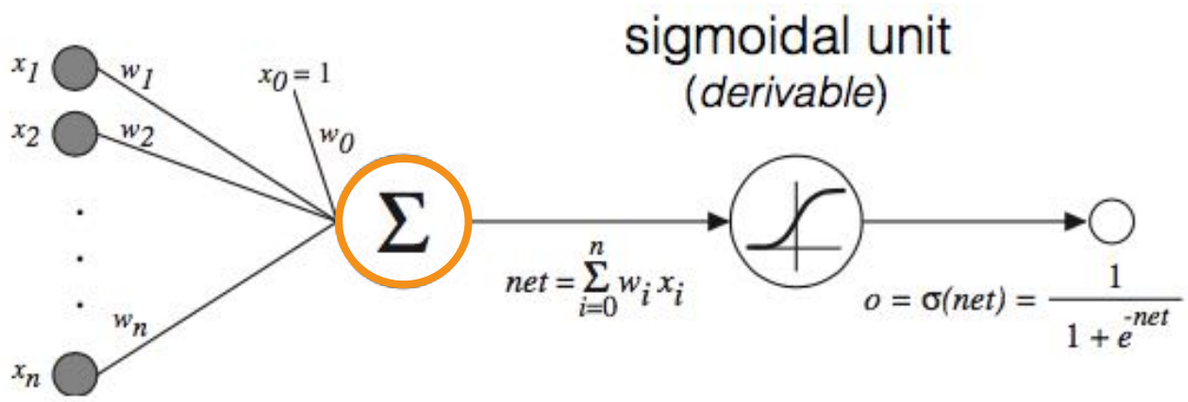
- input: 32x32x3 image -> *stretch to* 3072x1



activation: *1 number, i.e. the result of taking a dot product between a row of W and the input (a 3072-d dot product)*

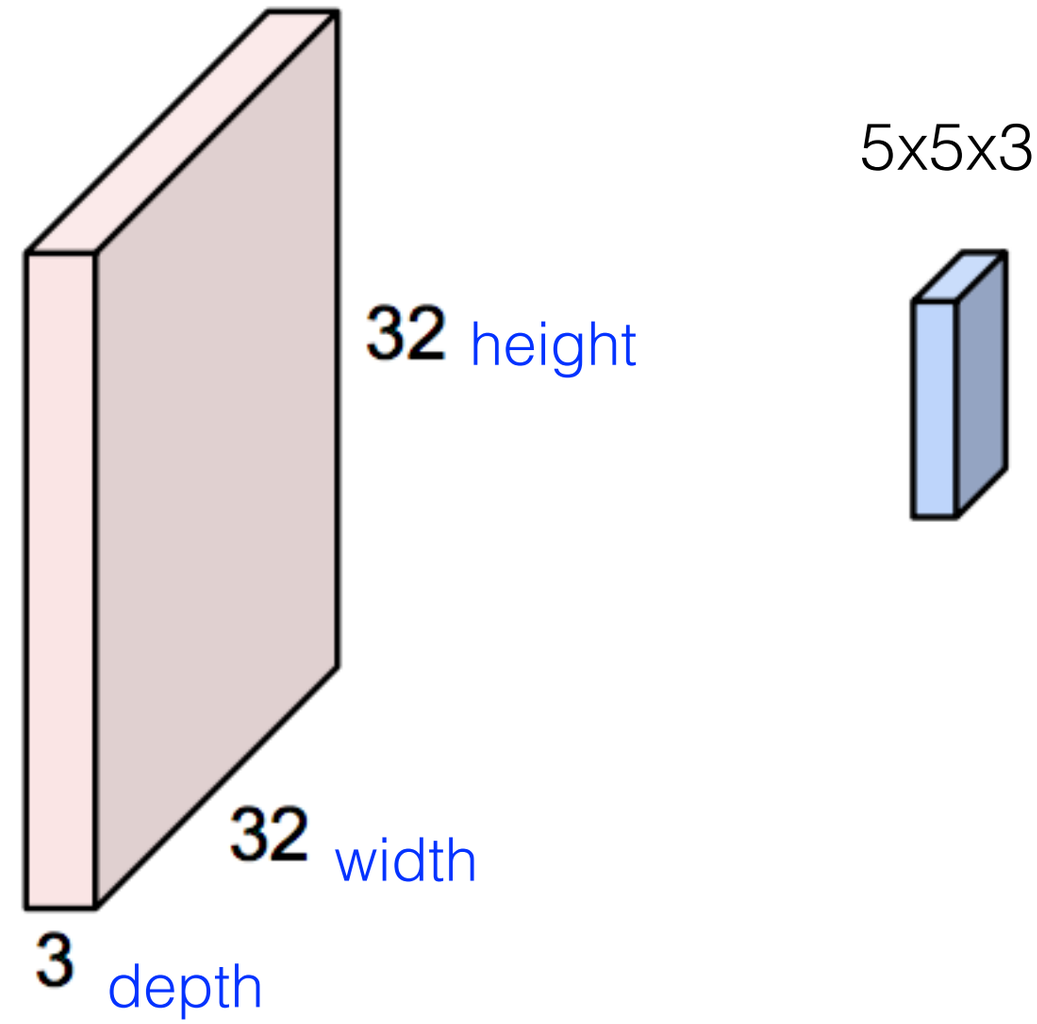
Artificial Neural Network





Convolutional layer

- input: 32x32x3 image -> *preserve spatial structure*



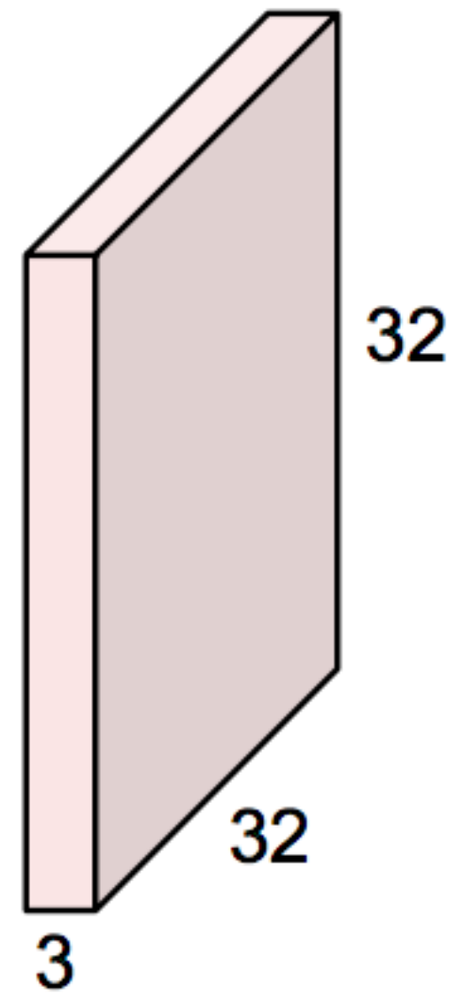
Convolve the filter with the image,
i.e. “slide over the image spatially,
computing dot products”

Convolutional layer

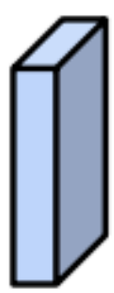
- input: 32x32x3 image -> *preserve spatial structure*

32x32x**3** image

Note: filters always extend the full depth of the input volume



5x5x**3** filter



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

Recall: image filtering and convolution

- Image filtering example: Gaussian smoothing

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

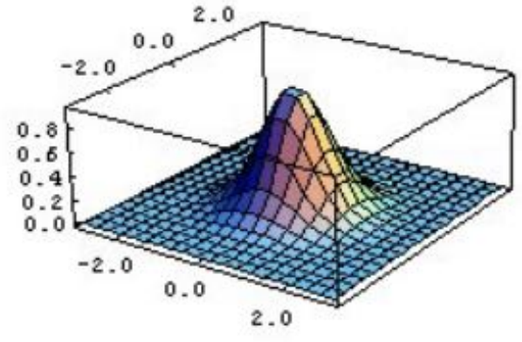
$F[x, y]$

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$

$H[x, y]$

This kernel is an approximation of a 2D Gaussian function

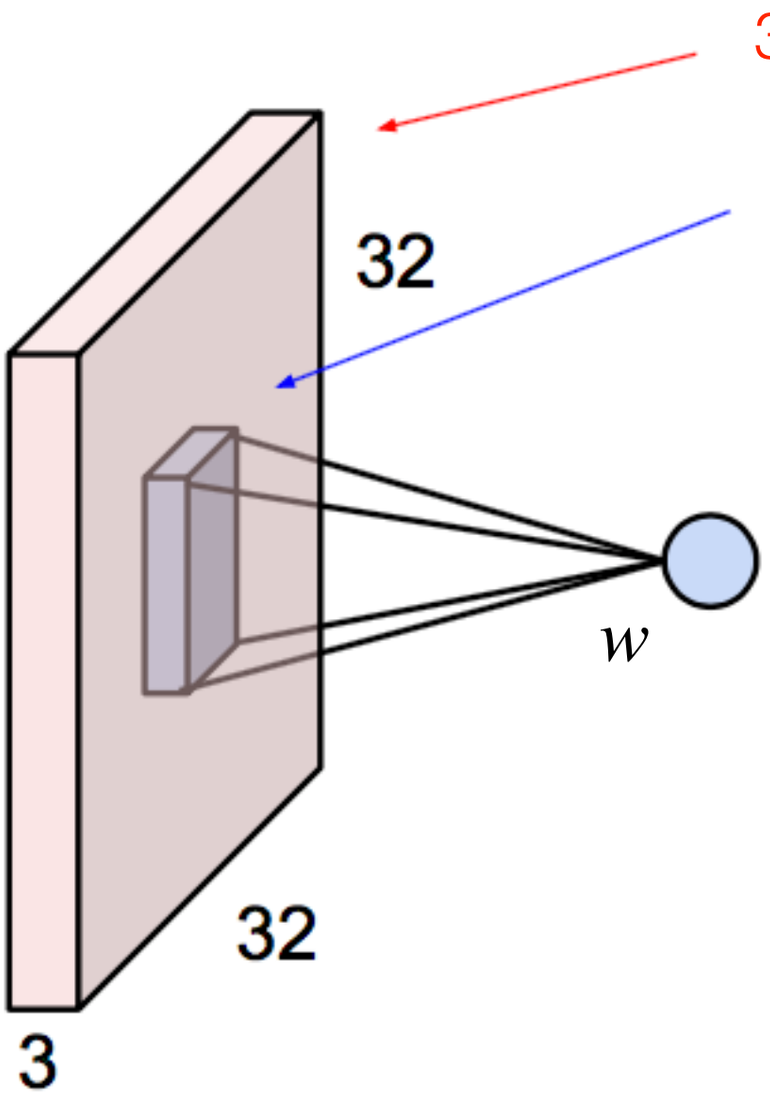
$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Convolution: $$(f * h)[n, m] = \sum_{k,\, l} f[k, l]\, h[n\text{-}k, m\text{-}l]$$

Convolutional layer

- input: 32x32x3 image -> *preserve spatial structure*



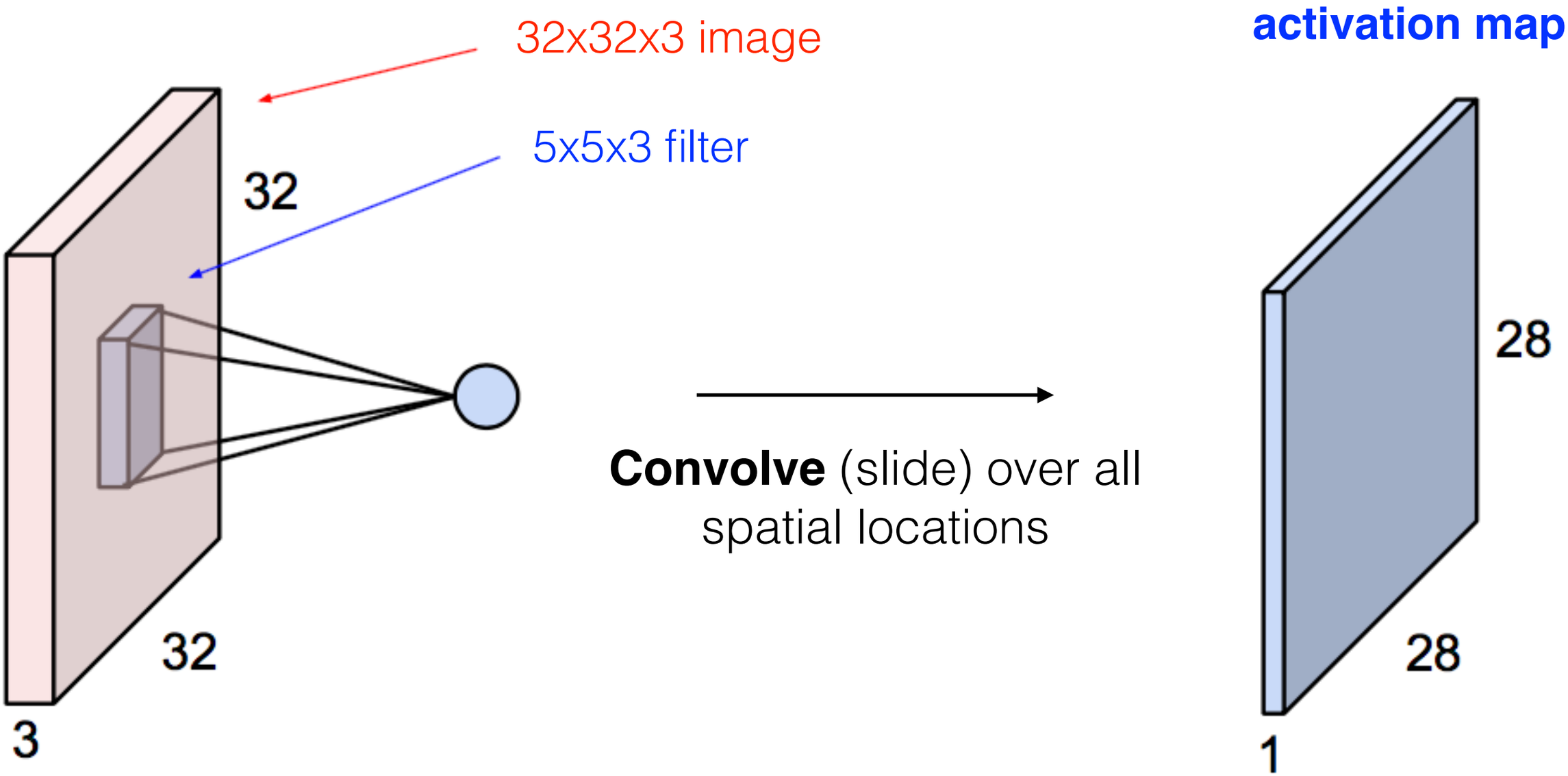
activation: *1 number, i.e. the result of taking a dot product between the filter and a 5x5x3 chunk of the image*

*(i.e. 5*5*3=75-d dot product + bias)*

$$w^T x + b$$

Convolutional layer

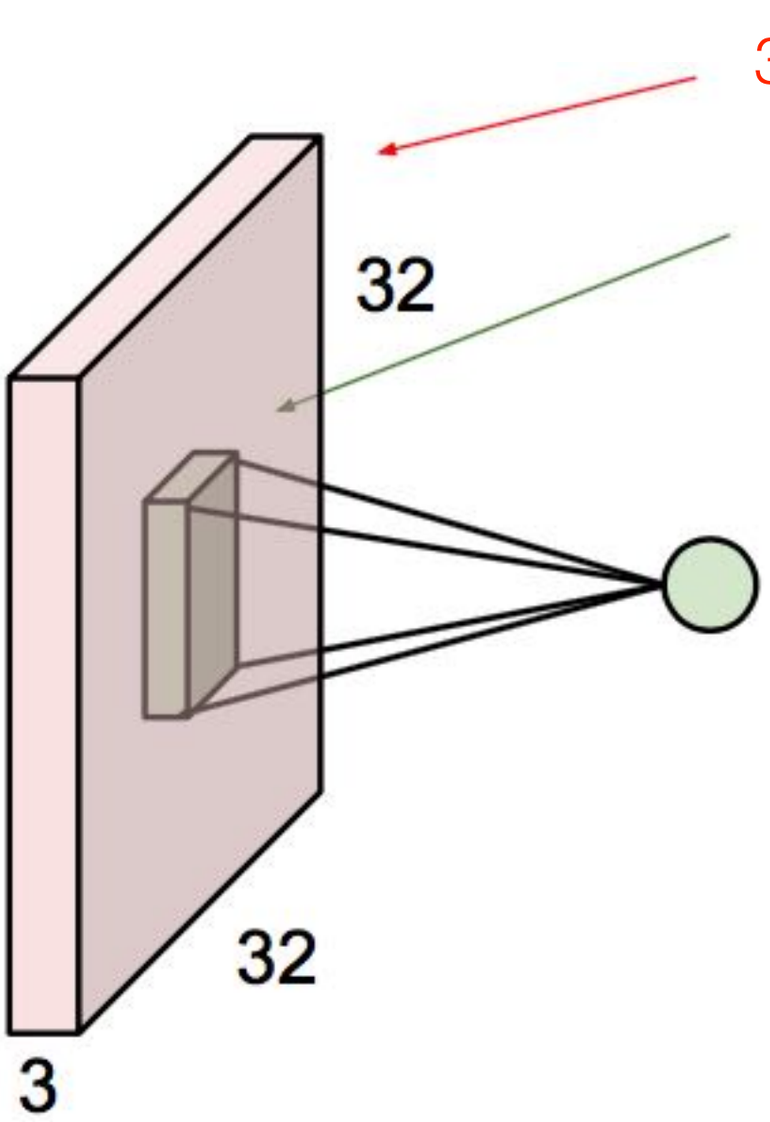
- input: 32x32x3 image -> *preserve spatial structure*



Slide credit: Fei-Fei, Karpathy, Johnson

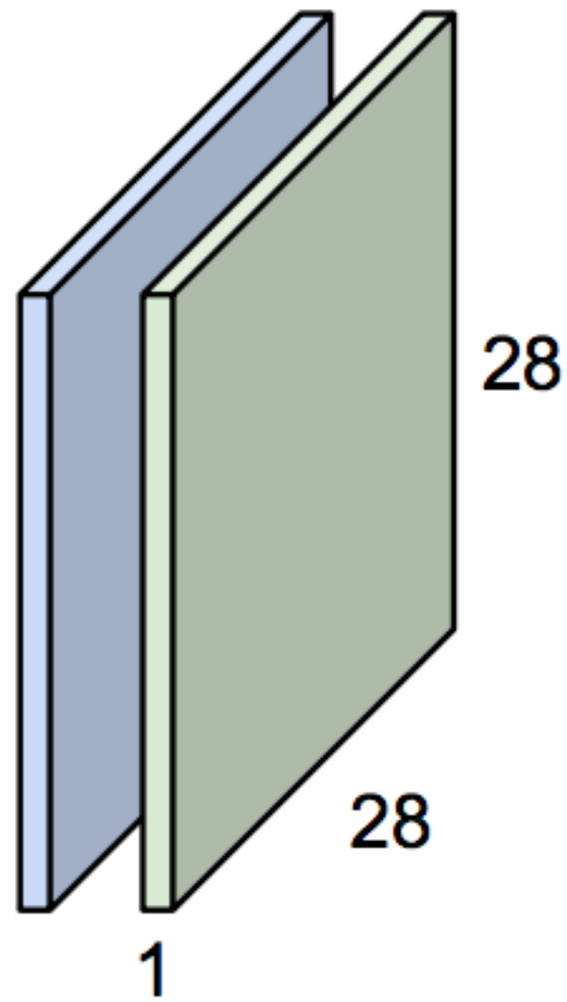
Convolutional layer

- Consider a second, green filter



Convolve (slide) over all spatial locations

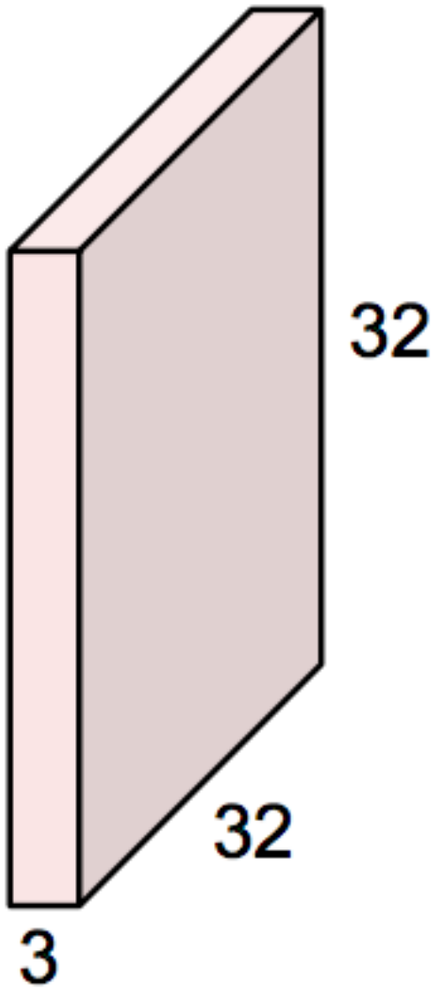
activation maps



Convolutional layer

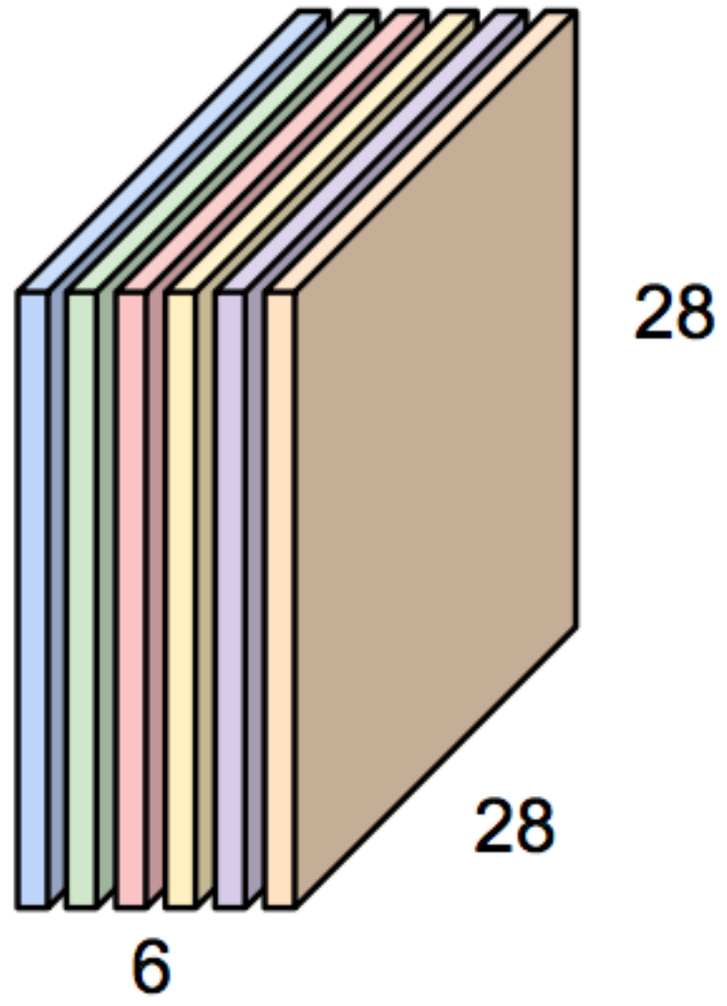
- For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

activation maps



We stack these up to get a "new image" (volume) of size 28x28x6

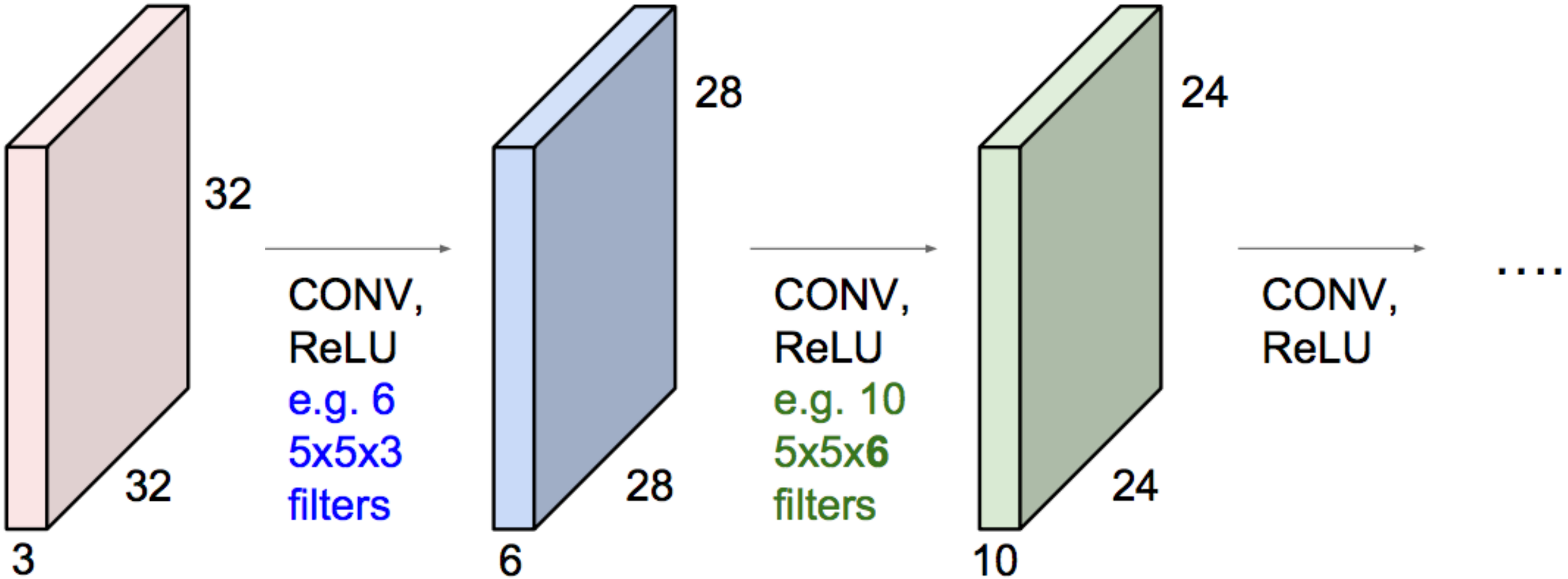
Convolution layer



Slide credit: Fei-Fei, Karpathy, Johnson

Convolutional Neural Networks

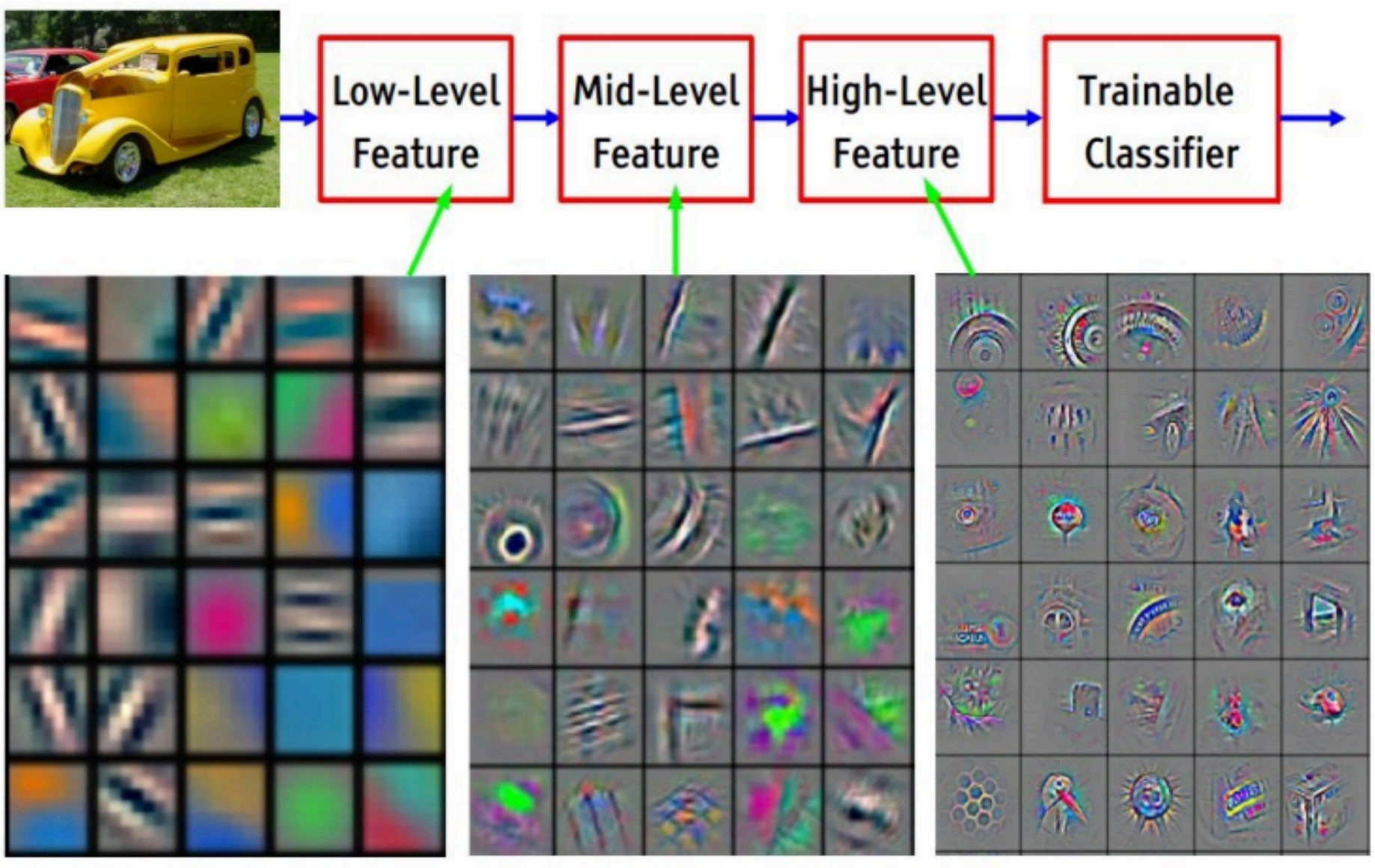
- A ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



Slide credit: Fei-Fei, Karpathy, Johnson

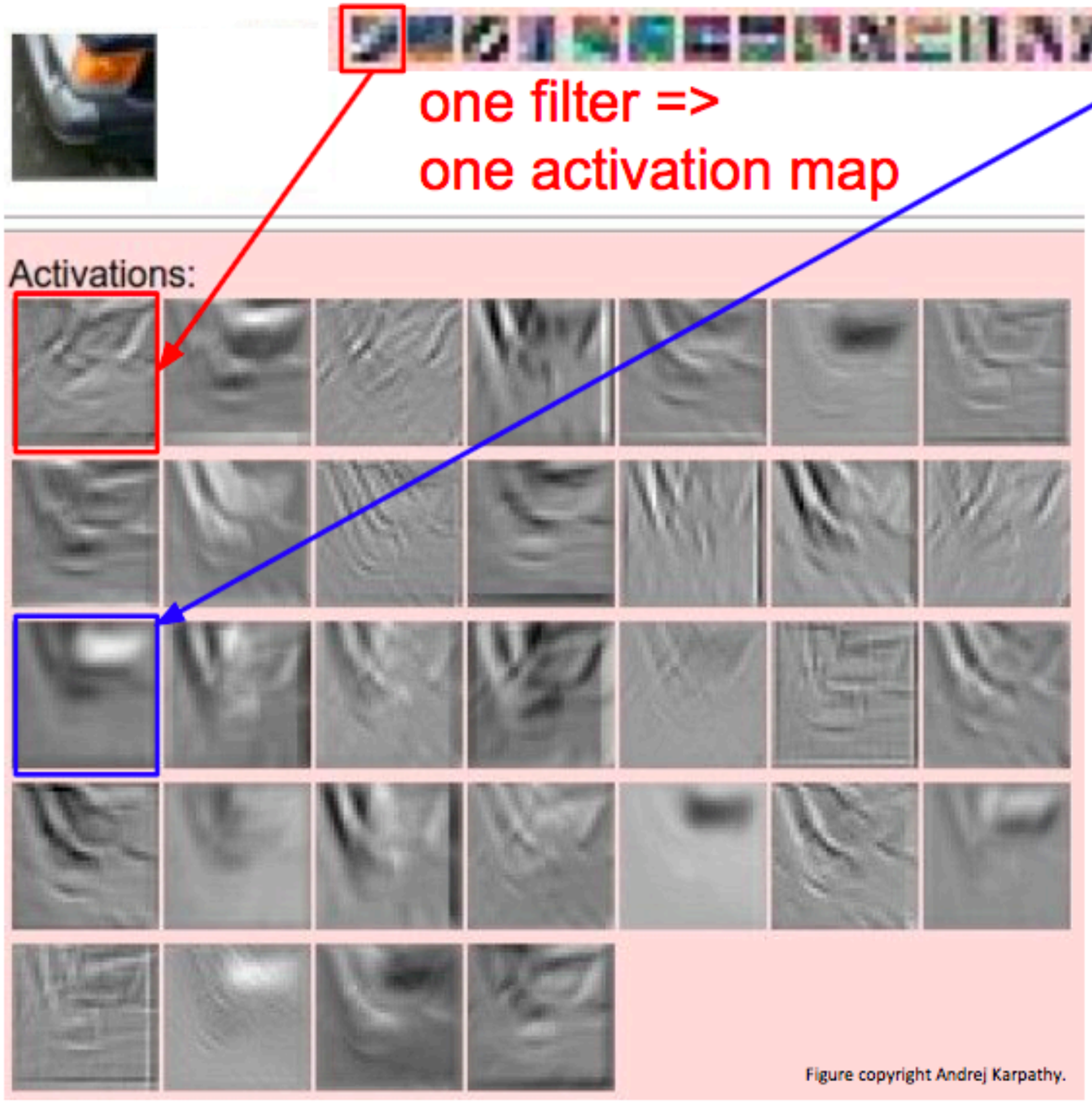
Convolutional Neural Networks

- Preview: feature visualization of CNN trained on ImageNet



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

Convolutional Neural Networks



example 5x5 filters
(32 total)

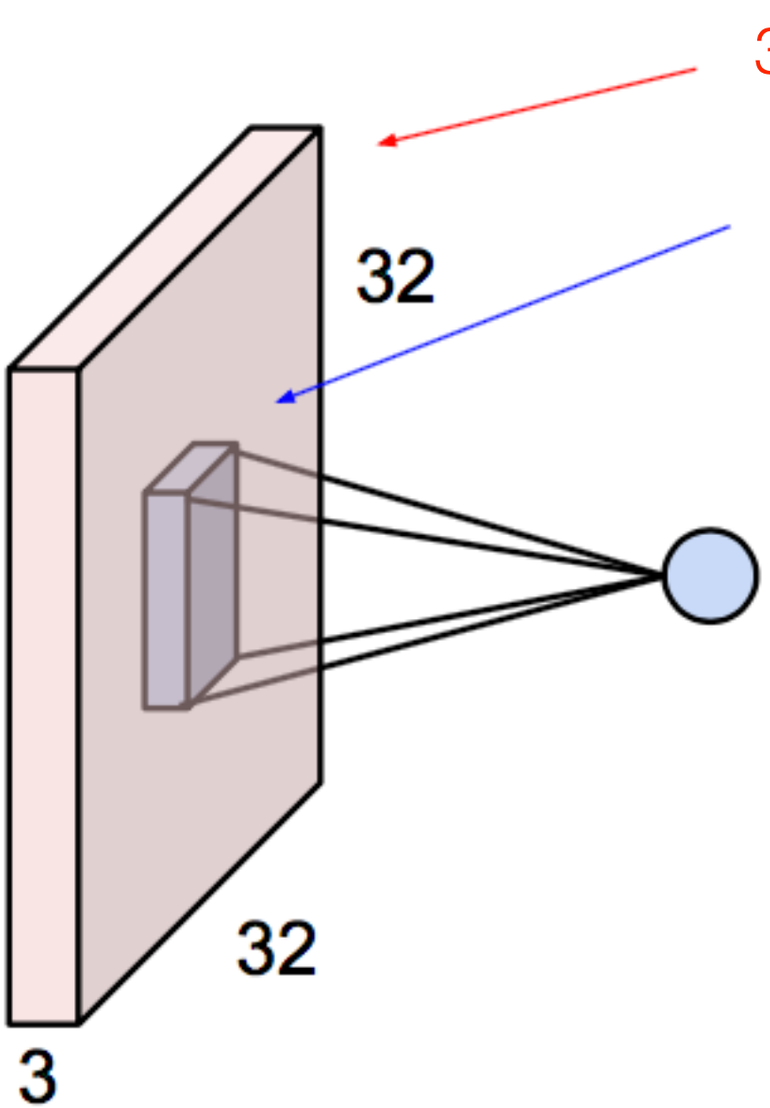
We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

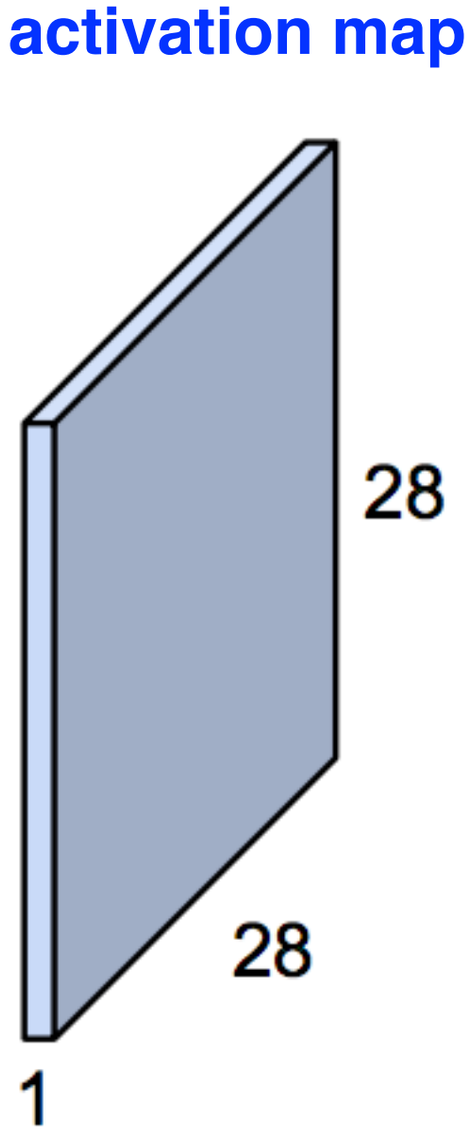
elementwise multiplication and sum of a filter and the signal (image)

Convolutional layer

- A closer look at spatial dimensions:

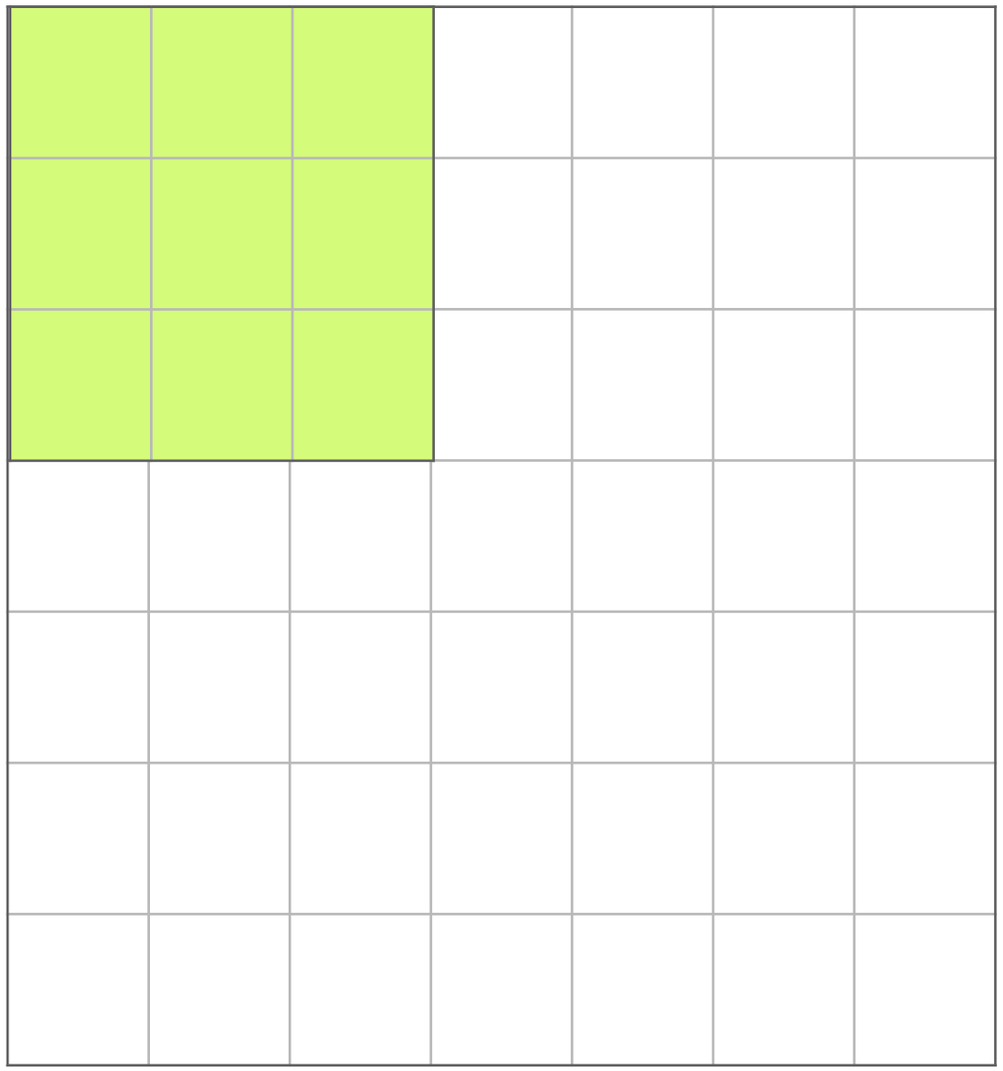


Convolve (slide) over all spatial locations



Convolutional layer

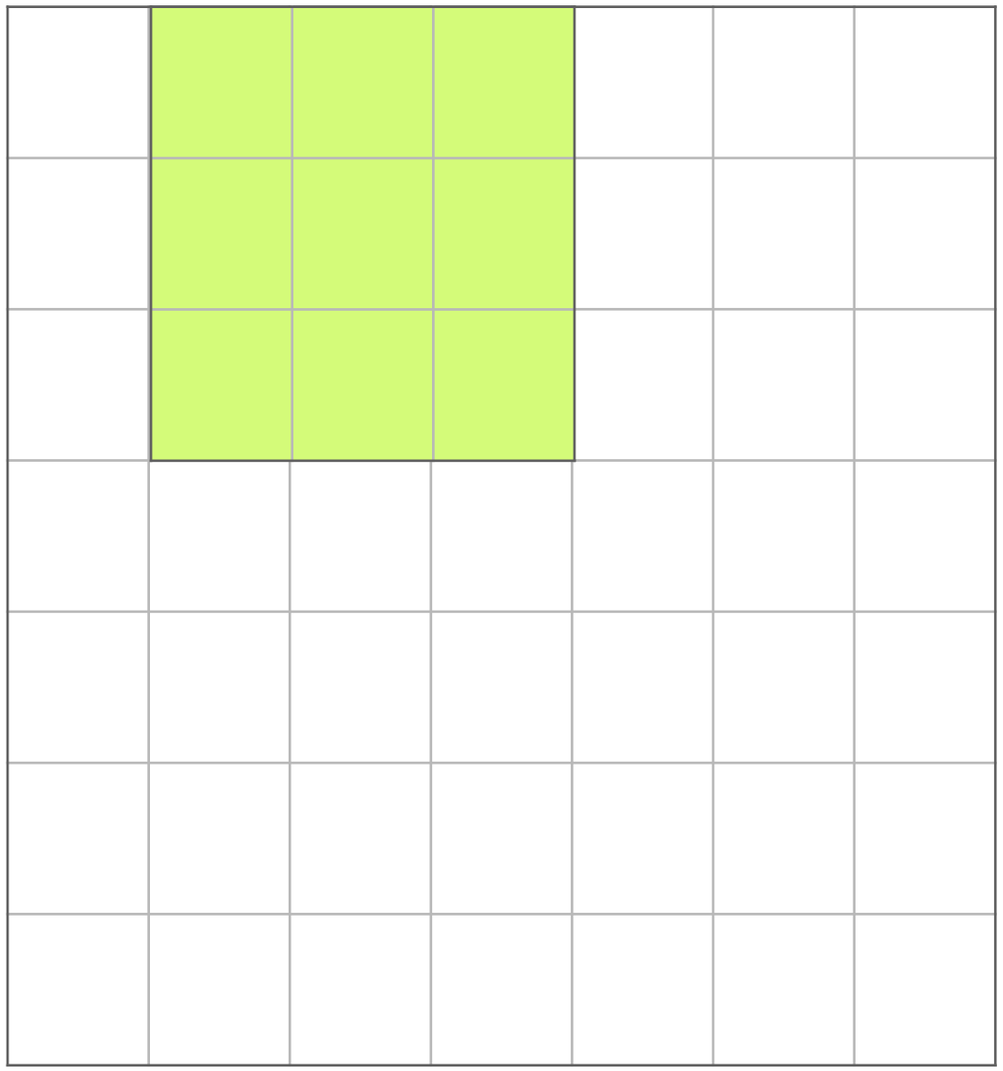
- A closer look at spatial dimensions:



7x7 input (spatially)
Assume 3x3 filter

Convolutional layer

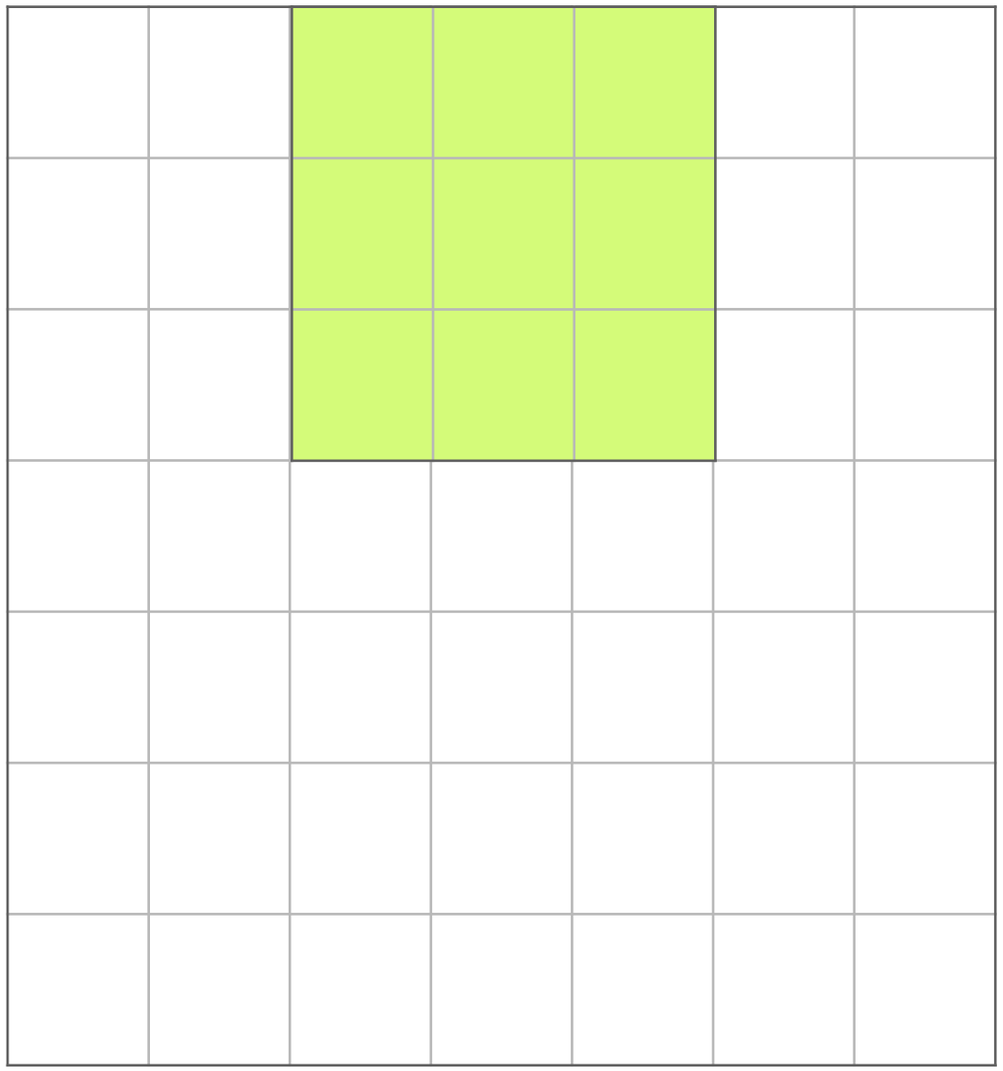
- A closer look at spatial dimensions:



7x7 input (spatially)
Assume 3x3 filter

Convolutional layer

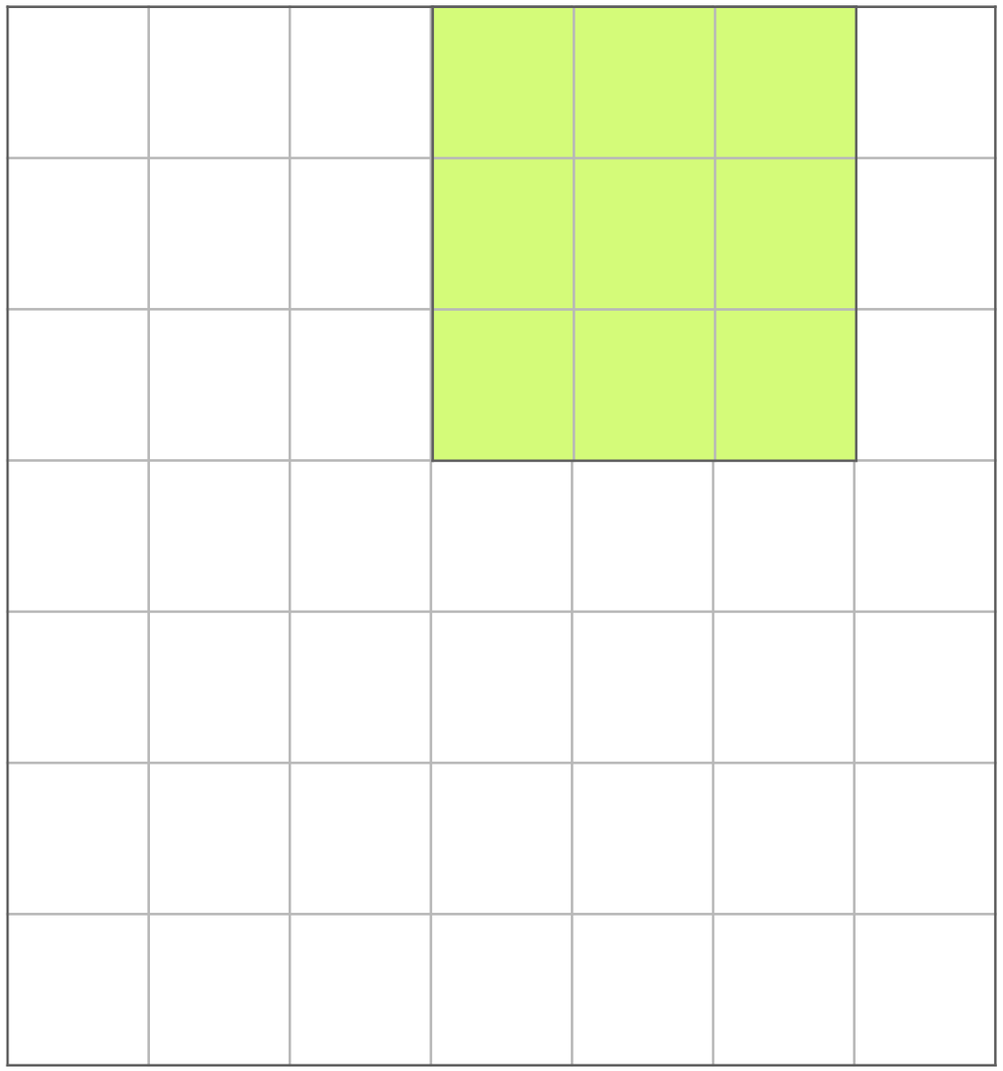
- A closer look at spatial dimensions:



7x7 input (spatially)
Assume 3x3 filter

Convolutional layer

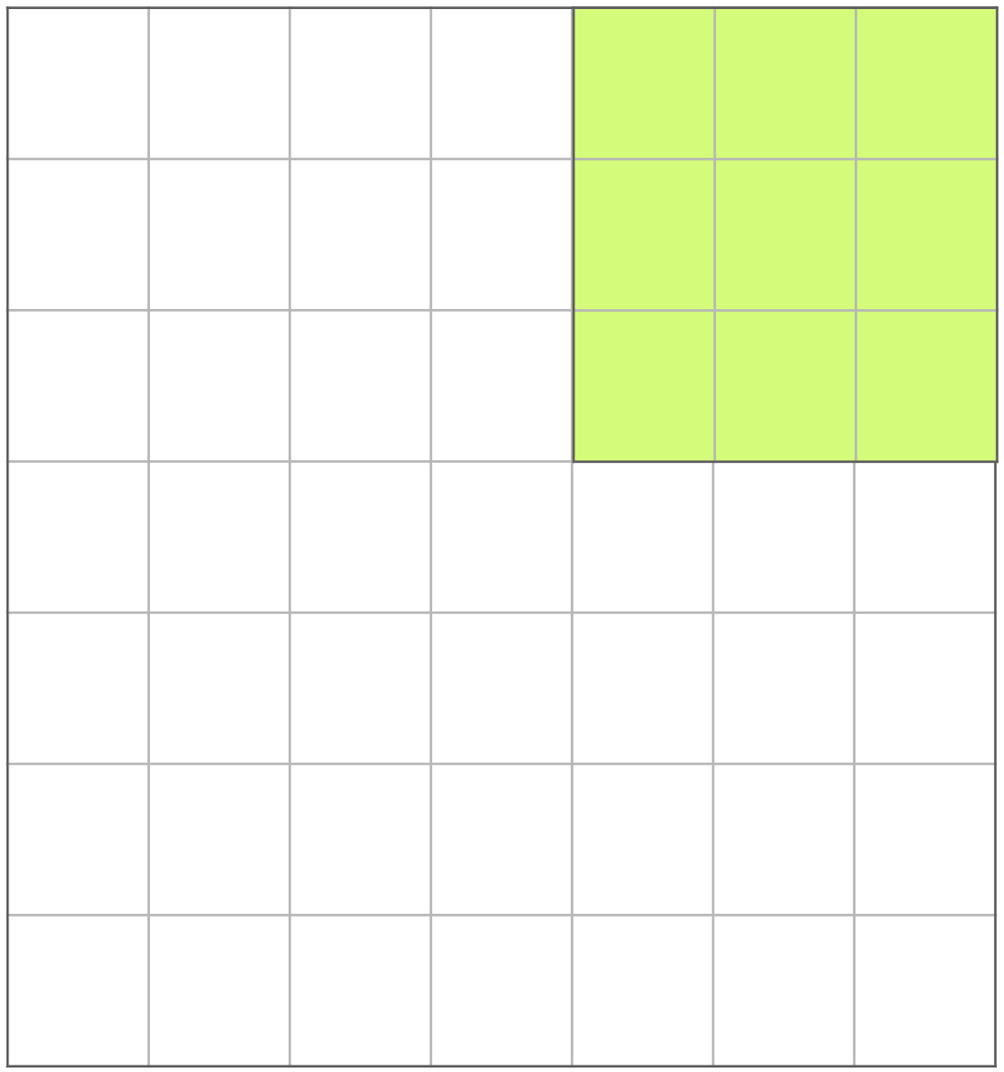
- A closer look at spatial dimensions:



7x7 input (spatially)
Assume 3x3 filter

Convolutional layer

- A closer look at spatial dimensions:

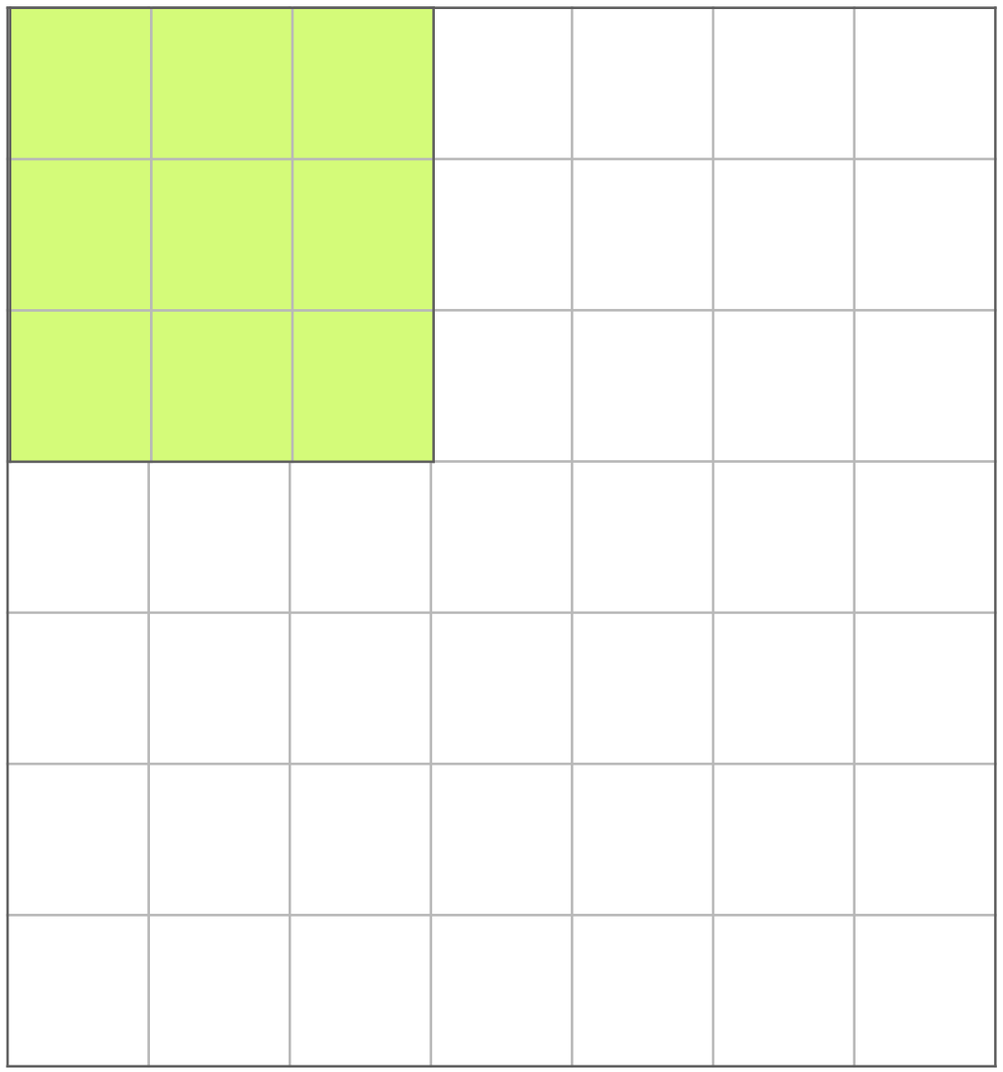


7x7 input (spatially)
Assume 3x3 filter

→ **5x5 output**

Convolutional layer

- A closer look at spatial dimensions:



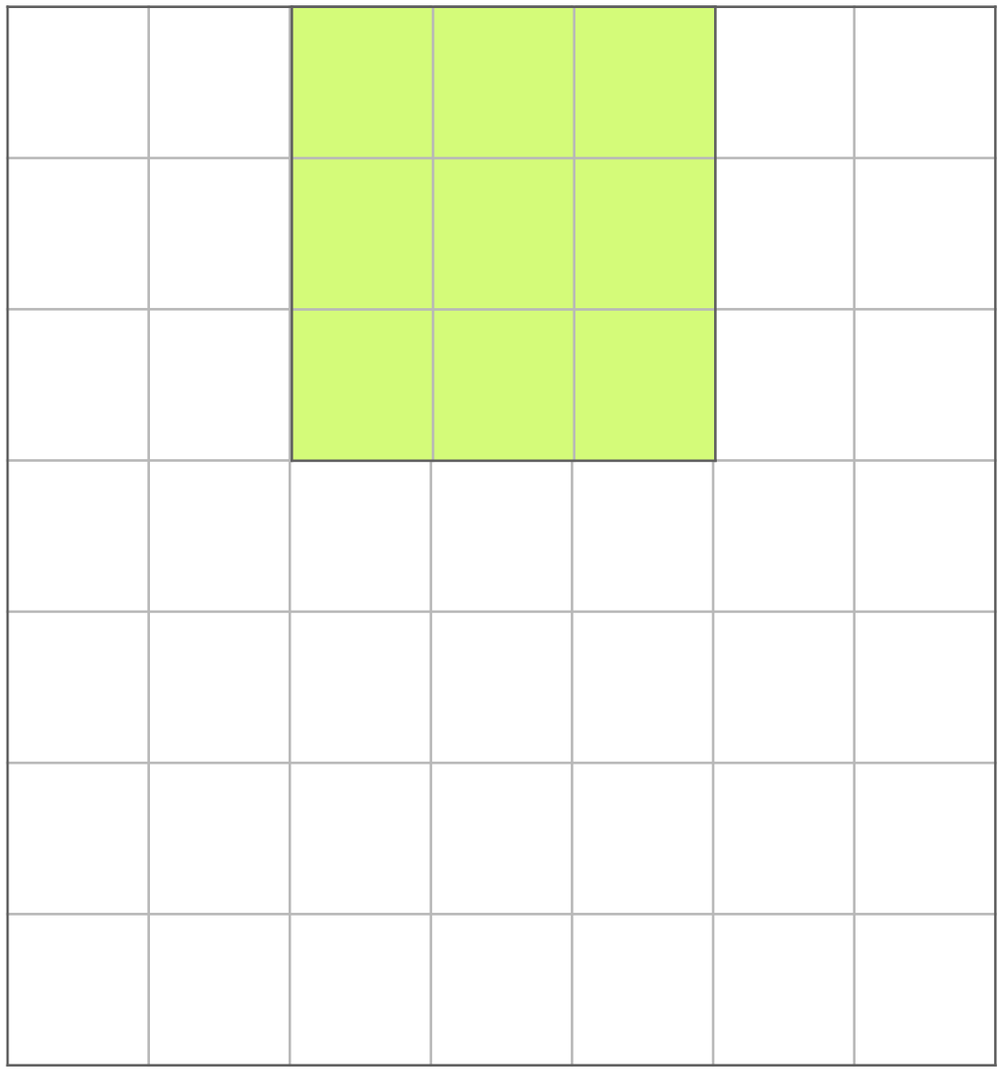
7x7 input (spatially)

Assume 3x3 filter

applied with **stride 2**

Convolutional layer

- A closer look at spatial dimensions:



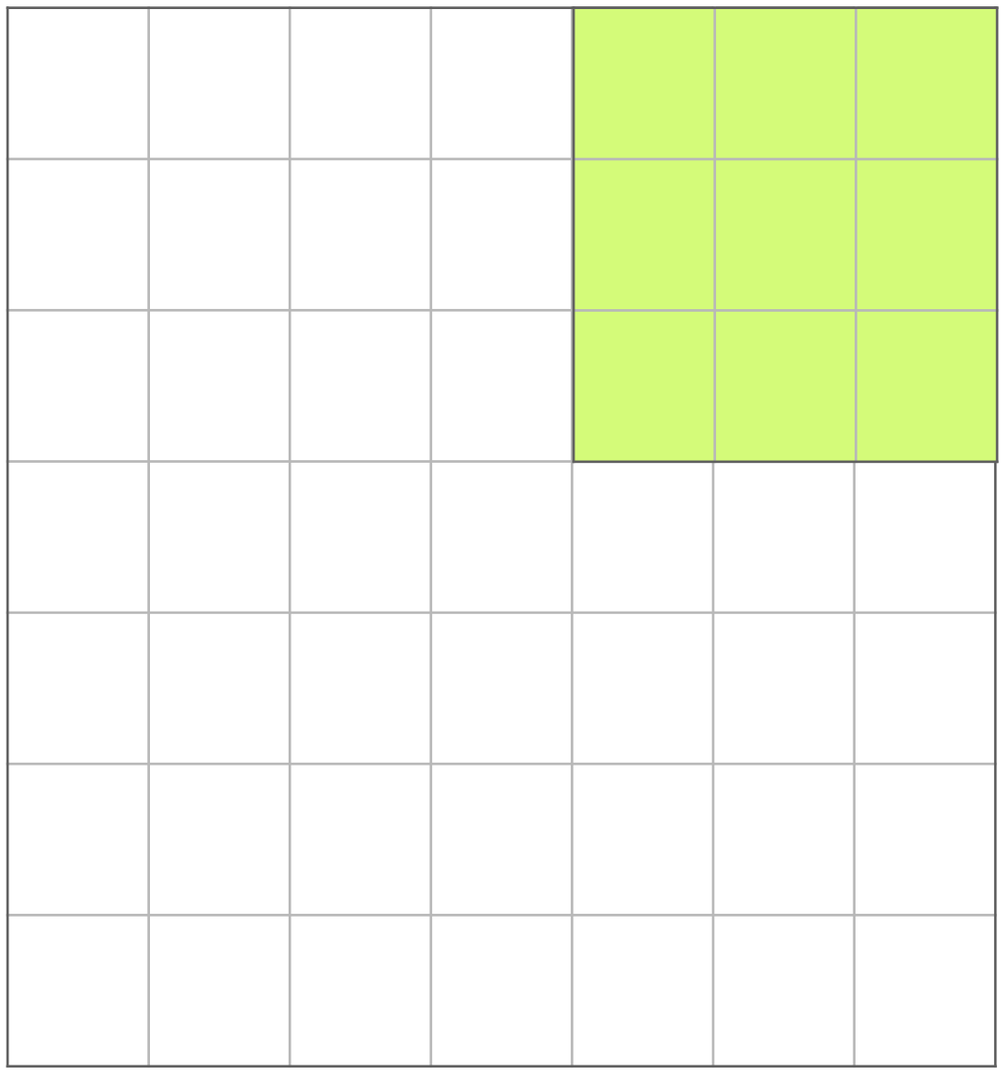
7x7 input (spatially)

Assume 3x3 filter

applied with **stride 2**

Convolutional layer

- A closer look at spatial dimensions:

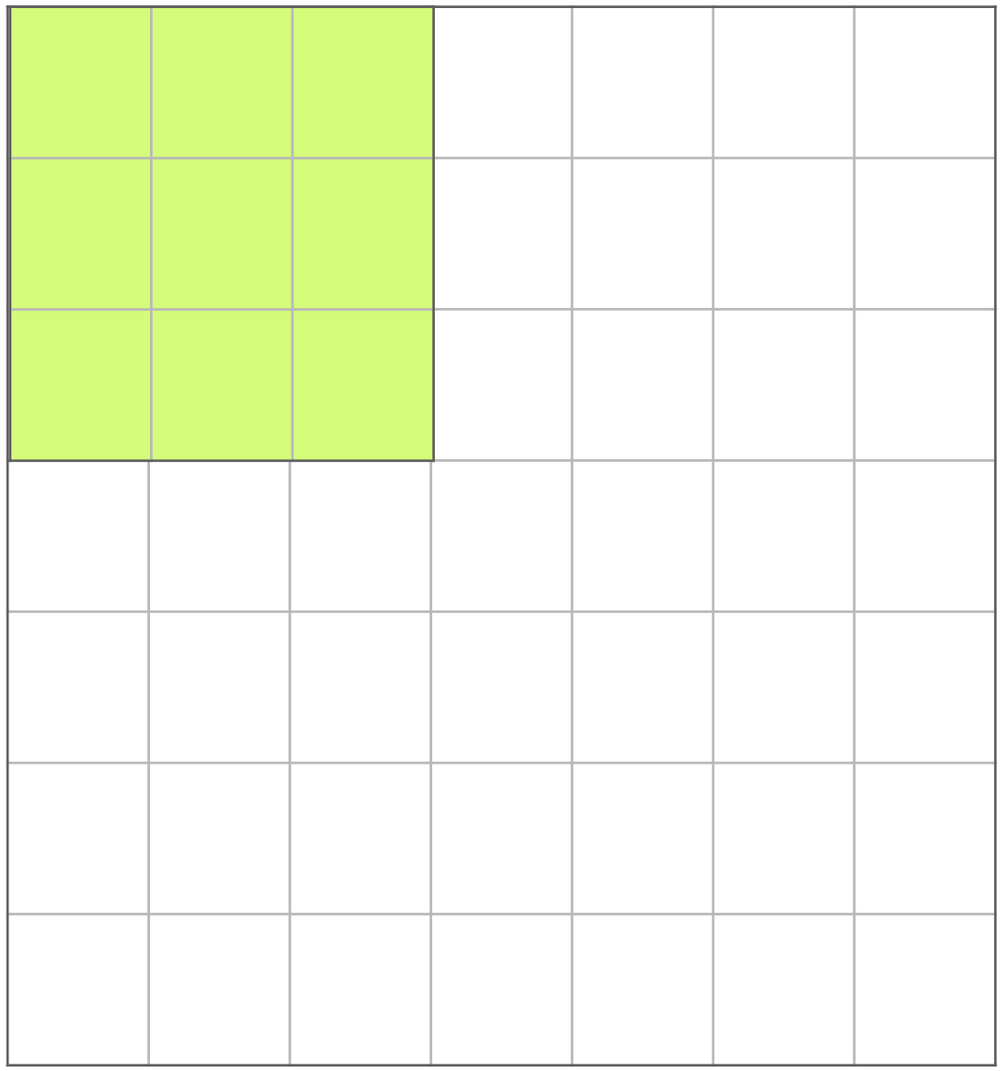


7x7 input (spatially)
Assume 3x3 filter
applied with **stride 2**

→ **3x3 output**

Convolutional layer

- A closer look at spatial dimensions:



7x7 input (spatially)

Assume 3x3 filter

applied with **stride 3?**

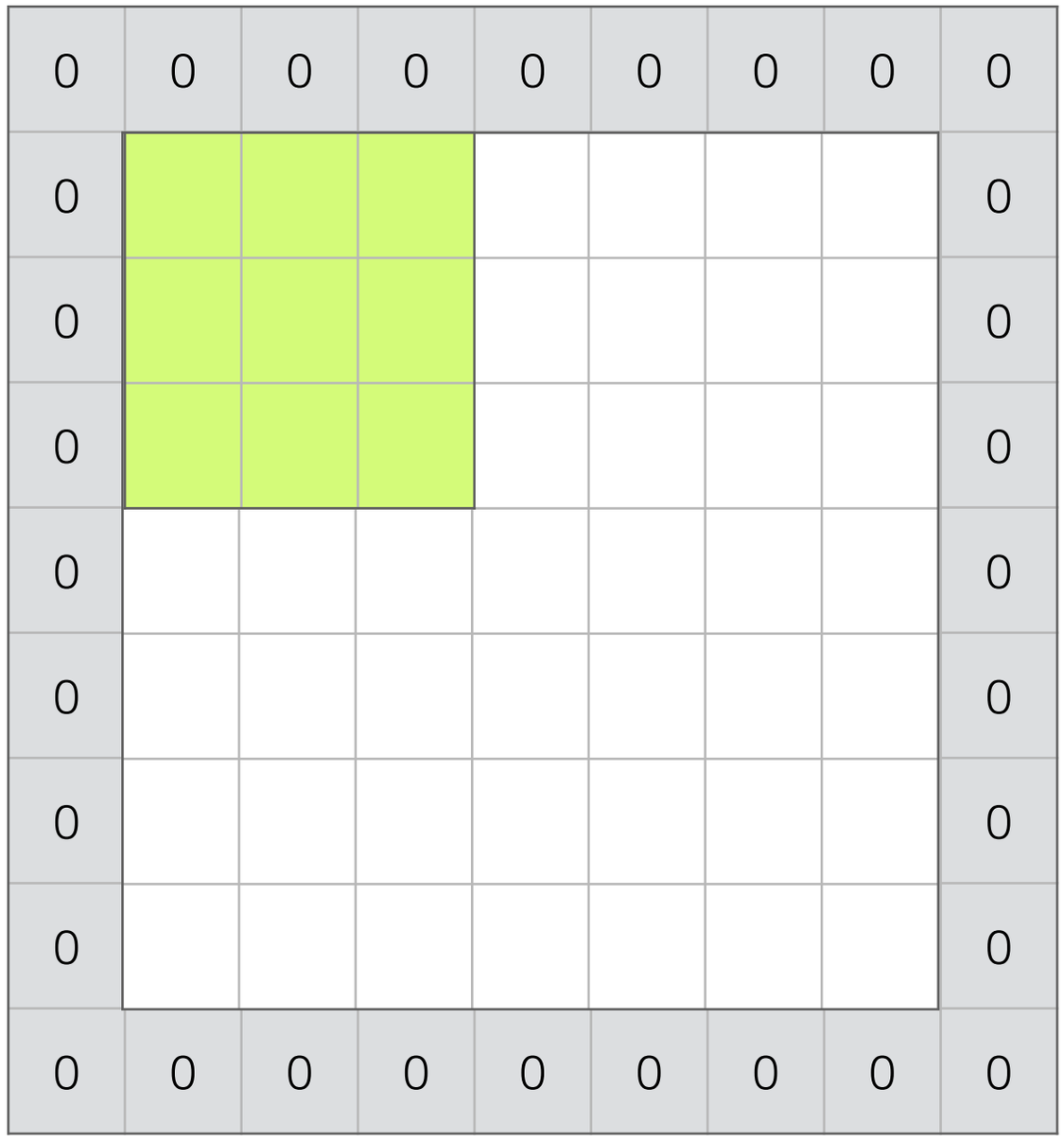
It doesn't fit!

Cannot apply 3x3 filter

on 7x7 input with stride 3

Convolutional layer

- Padding: it's common to zero pad the border



7x7 input (spatially)

Assume 3x3 filter applied with **stride 3**

Pad with 1 pixel border; what is the output?

→ **3x3 output**

In general is common to see CNNs with stride 1, filters of size FxF, and zero-padding with (F-1)/2 (this will preserve size spatially)

Conv layer: spatial arrangement

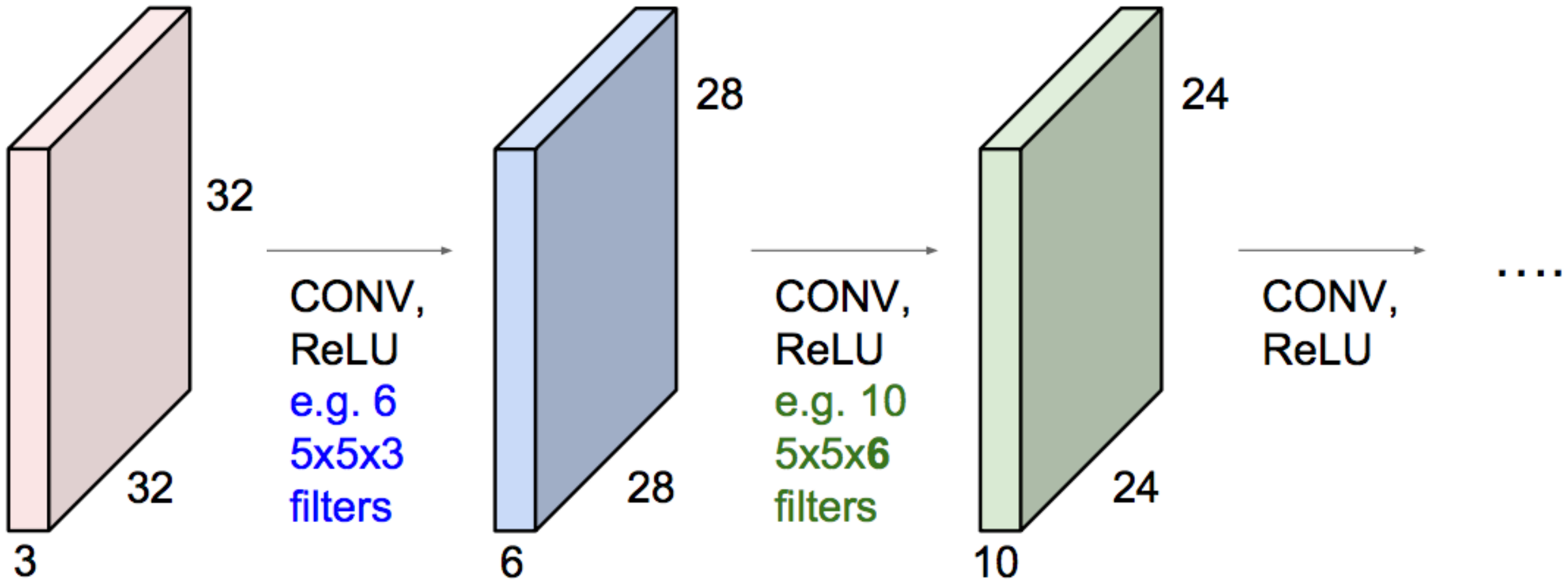
- Three hyper-parameters control the size of the output volume: the *depth*, *stride* and *zero-padding*
- Output volume size: *(W-F+2*P)/S+1*
  - *W*: input volume size
  - *F*: filter size (commonly referred as to *receptive field*)
  - *S*: the stride with which filter(s) are applied
  - *P*: the amount of zero padding used on the border

Previous example:

7x7 input, 3x3 filter, stride 1 and pad 0 => 5x5 output,
*i.e. (7-3+2*0)/1+1=5*

Convolutional Neural Networks

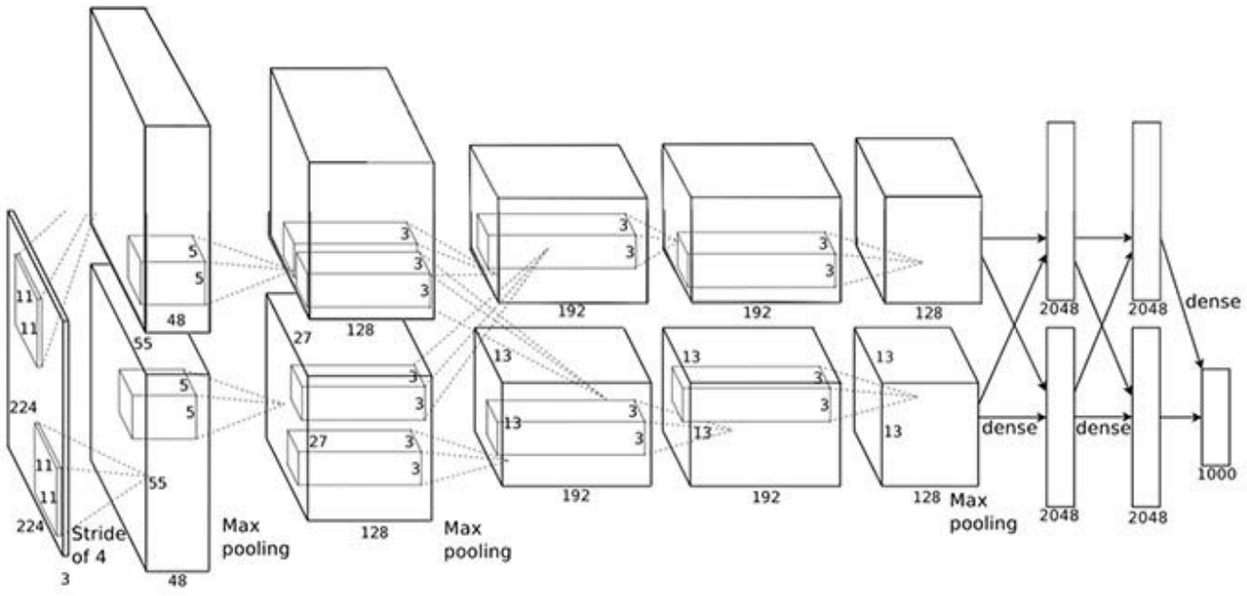
- Remember: 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially (i.e. 32, 28, 24, …)

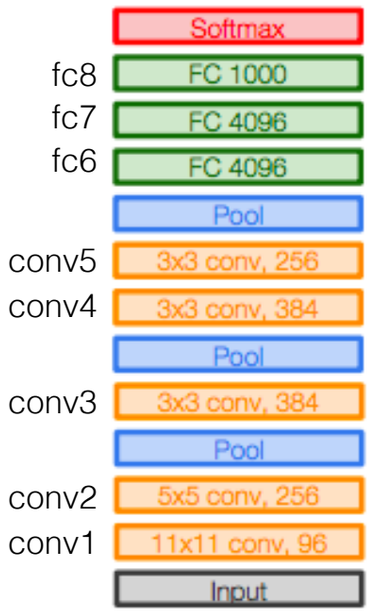


Shrinking too fast is bad. It doesn't work well.

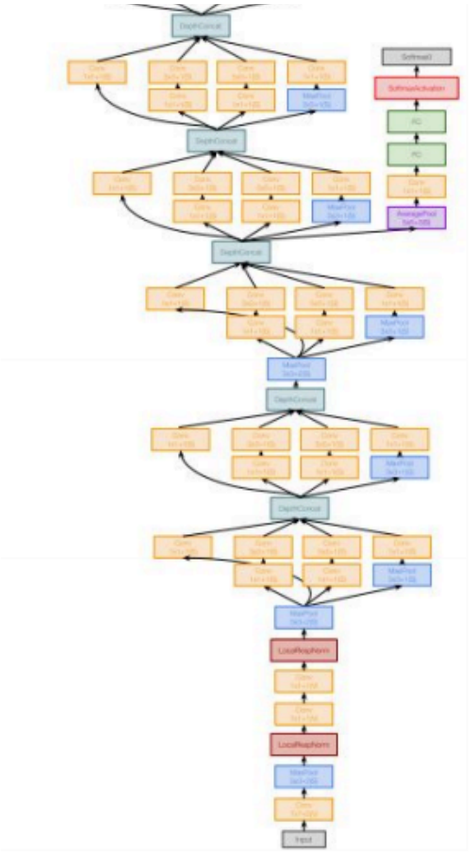
Coming up

- **Next lectures:**
  - Labs on ANN/MLP and ConvNets
  - ConvNets: a bit more on CNNs, architectures, learning





AlexNet



GoogLeNet