Convolutional Neural Nets (Convnets)

Excellent reference material:

http://cs231n.stanford.edu/ - Stanford's Convnet class. Video of lectures available at https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv

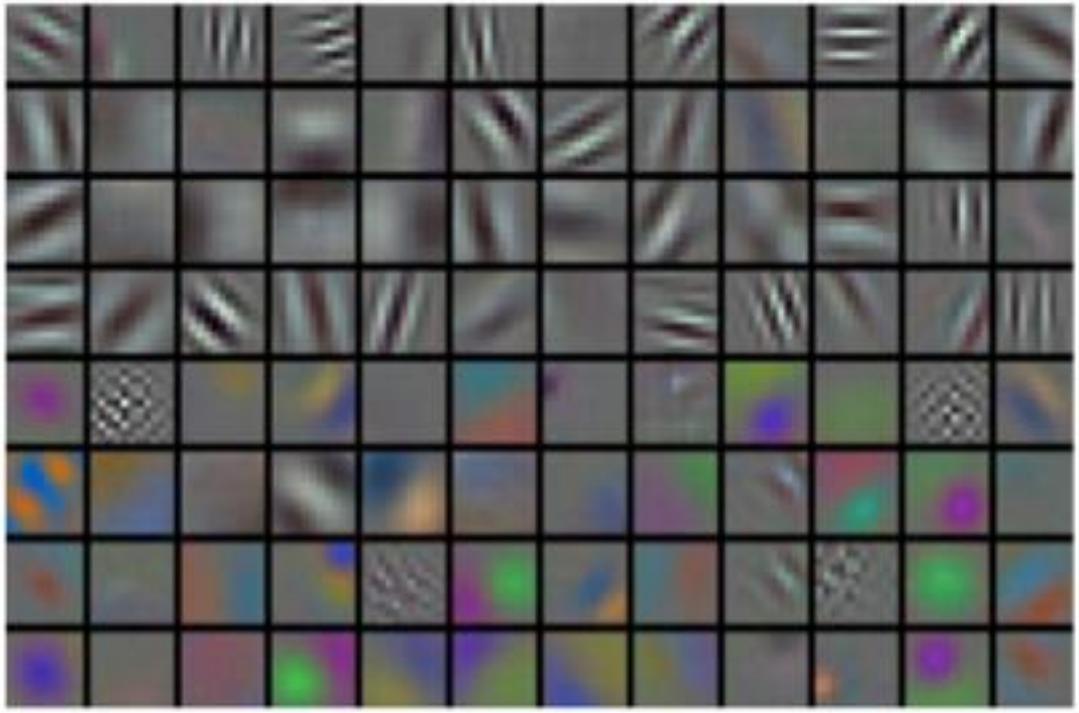
Motivation

- So far we've seen several "hand-crafted" features for describing local image structure.
- Examples
  - Derivative of Gaussian ..... edges
  - 2nd Derivative of Gaussian ..... edges
  - Laplacian of Gaussian ..... edges and blobs
  - Harris Corner R score ..... edges, corners, uniform regions

Motivation

Given a set of images with labels about what objects they contain, can we automatically learn features that are good for classifying those different objects?

Yes!

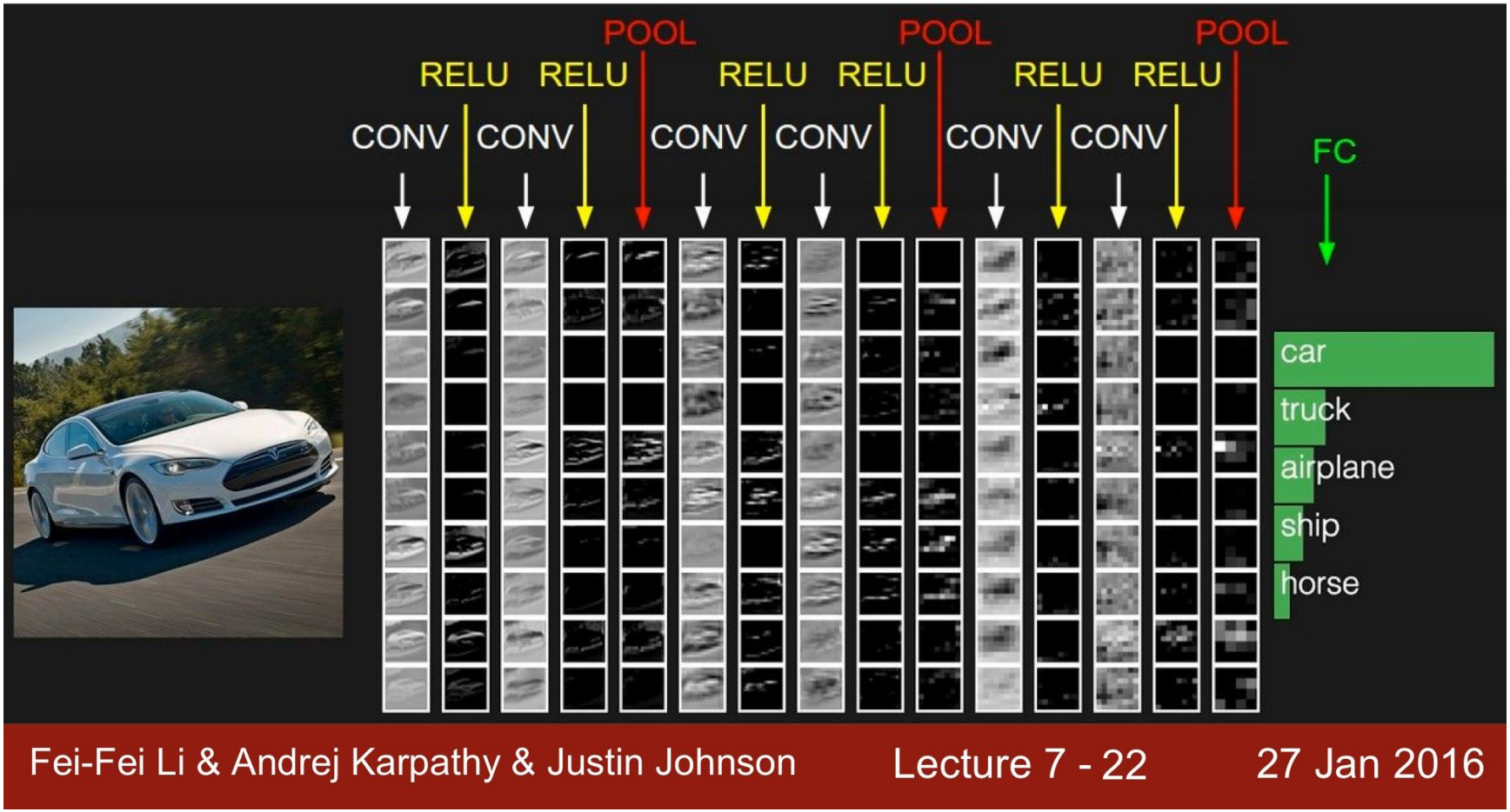


Some convolution filters learned by a neural network.

Convnets

- A very modular kind of classifier. Number of layers chained together are considered to be the “depth”. Result is a deep classifier, hence deep learning.

preview:



Demo running in browser, at http://cs231n.stanford.edu/

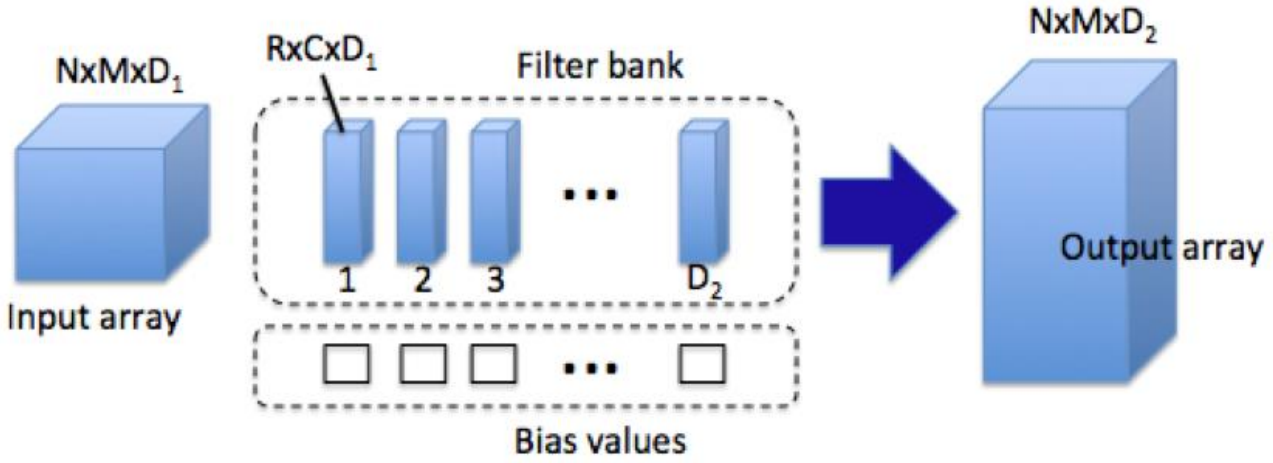
Convnet Overview

- Forming complex classifiers as long chains of simple computational building blocks.
- Building Blocks
  - Convolution (actually cross-correlation)
  - Nonlinear Activation (e.g. ReLU)
  - Pooling (e.g. max-pool)
  - Fully Connected Layers
  - Softmax
- Goal is to understand forward pass computation.
- Understanding backpropagation (how learning takes place) is outside the scope of this lecture.

Overview

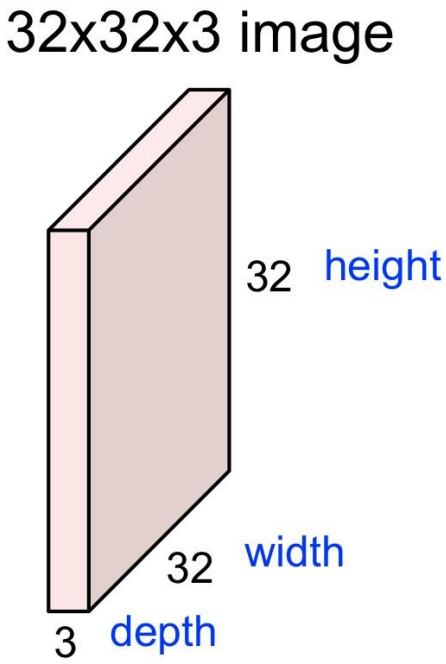
- Building Blocks
  - Convolution (actually cross-correlation)
  - Nonlinear Activation (e.g. ReLU)
  - Pooling (e.g. max-pool)
  - Fully Connected Layers
  - Softmax
- Goal is to understand forward pass computation

Convolution Layer



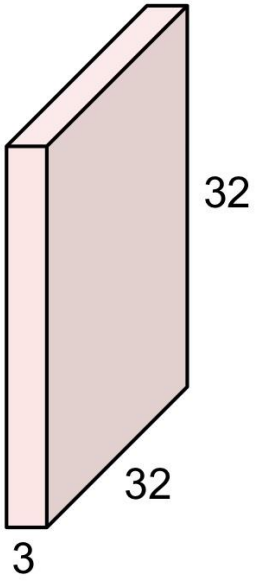
A convolution layer transforms an $N \times M \times D1$ input array into an $N \times M \times D2$ output array by convolving with a filter bank of $D2$ linear filters and adding bias values.

Convolution Layer

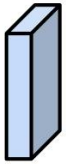


Convolution Layer

32x32x3 image



5x5x3 filter

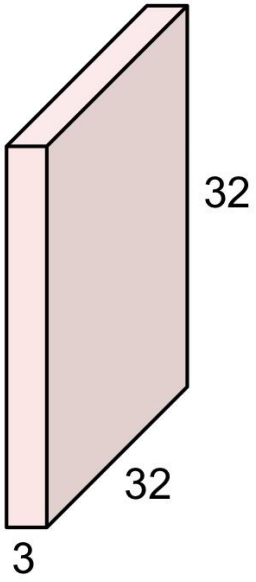


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

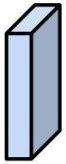
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

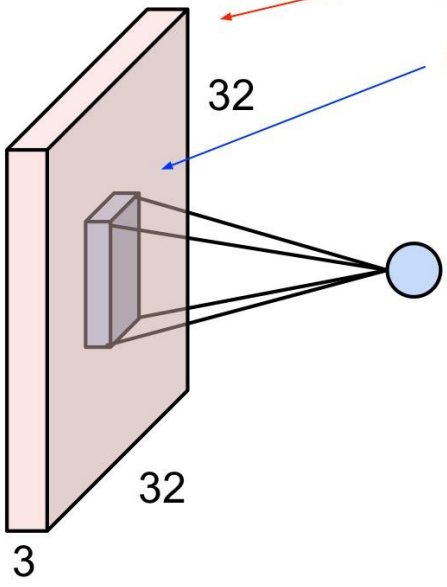


5x5x3 filter



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

Convolution Layer



32x32x3 image

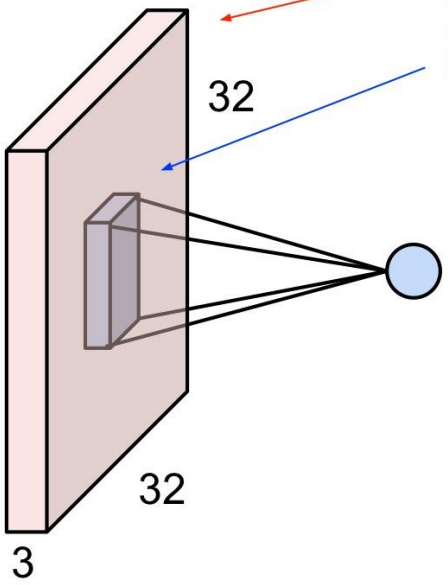
5x5x3 filter w

Scalar bias value b

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)

$$w^T x + b$$

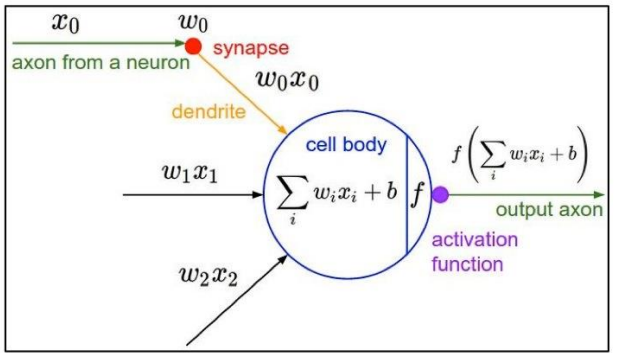
The brain/neuron view of CONV Layer



32x32x3 image

5x5x3 filter

Scalar bias value b

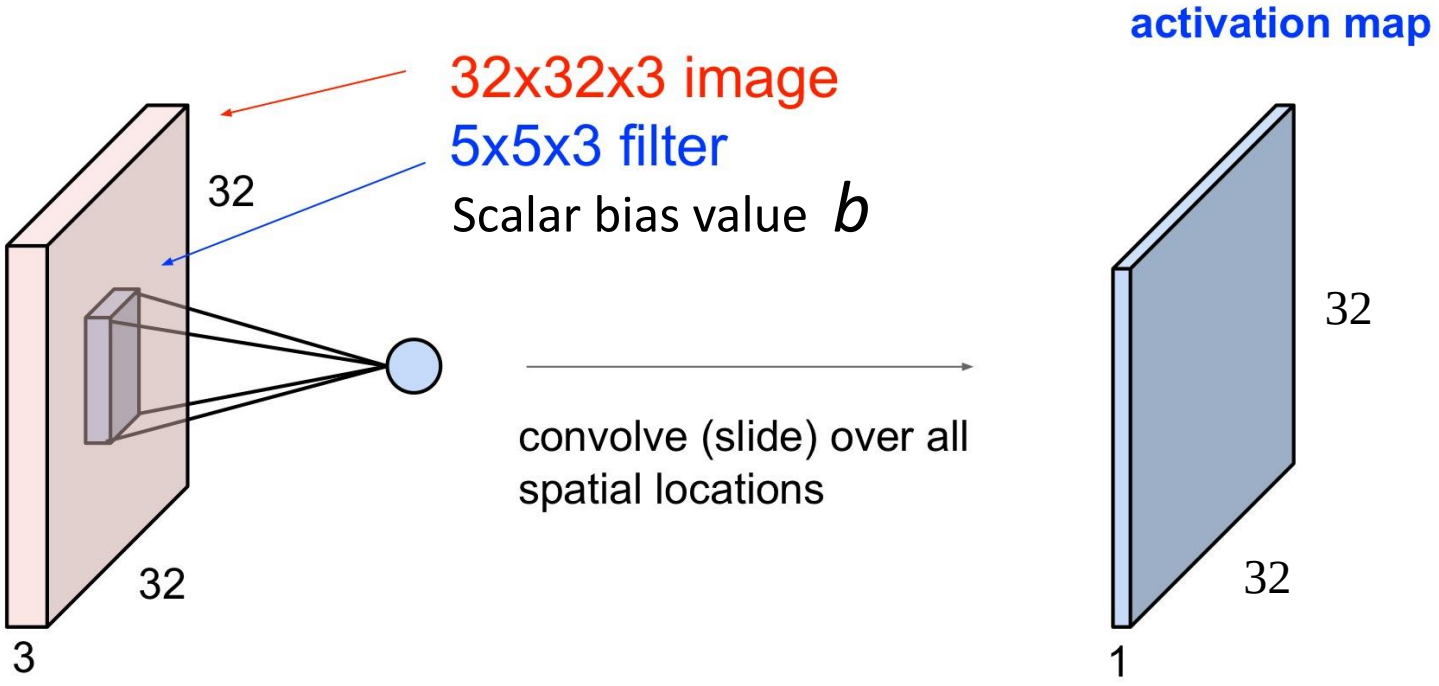


It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product) then add bias value b

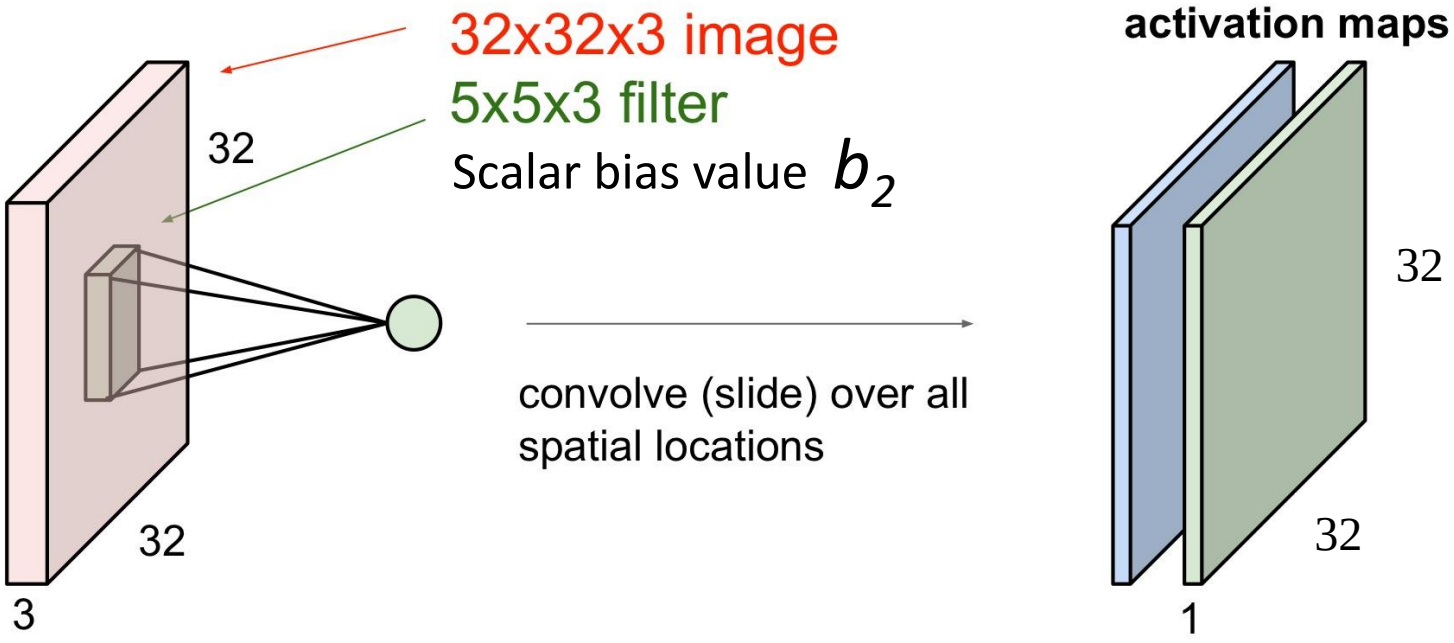
Activation function here is just identity function

Convolution Layer



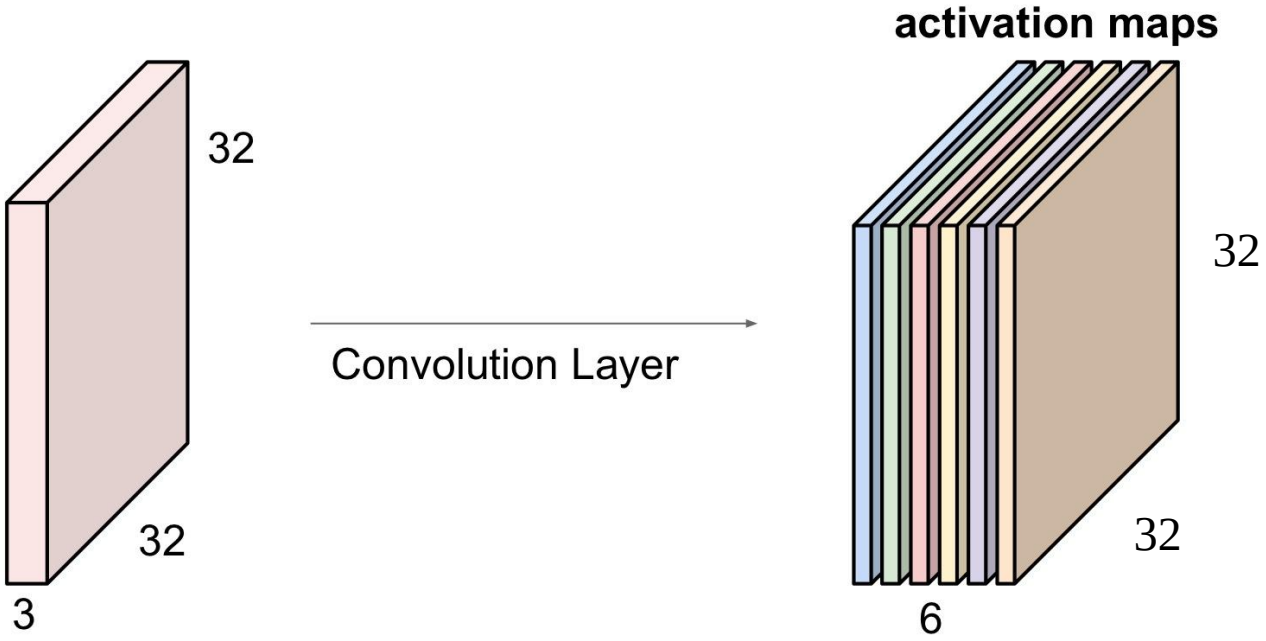
Convolution Layer

Consider a second, different filter



and 6 scalar bias values,

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 32x32x6

Equivalent to the idea of applying a convolution filter bank in order to generate a vector of feature values.

Overview

- Building Blocks
  - Convolution (actually cross-correlation)
  - Nonlinear Activation (e.g. ReLU)
  - Pooling (e.g. max-pool)
  - Fully Connected Layers
  - Softmax
- Goal is to understand forward pass computation

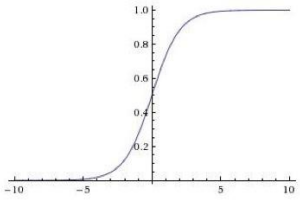
Motivation for Activation Functions

- Convolution is a linear operator
- Cascades of convolutions are also linear
- The complexity of a linear classifier's decision boundary is limited.
- Activation functions introduce nonlinearity into the system, so that it can have nonlinear decision boundaries!

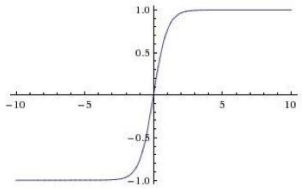
Activation Functions

Sigmoid

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

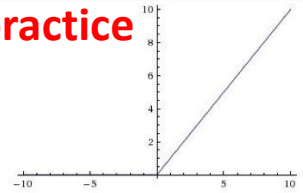


tanh tanh(x)

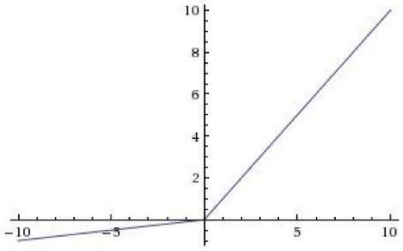


Most common in current practice

ReLU max(0,x)



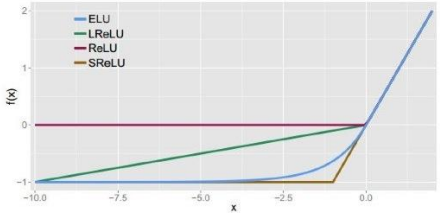
Leaky ReLU
max(0.1x, x)



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

ELU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$$



RELU clamps negative values in an array to 0. Nonnegative values remain the same.

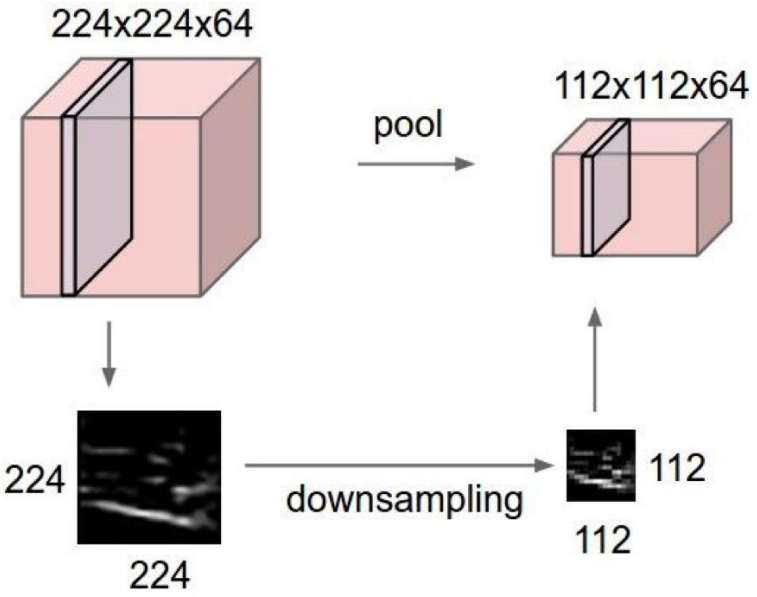
5	-1	→	5	0
-2	4		0	4

Overview

- Building Blocks
  - Convolution (actually cross-correlation)
  - Nonlinear Activation (e.g. ReLU)
  - Pooling (e.g. max-pool)
  - Fully Connected Layers
  - Softmax
- Goal is to understand forward pass computation

Pooling layer

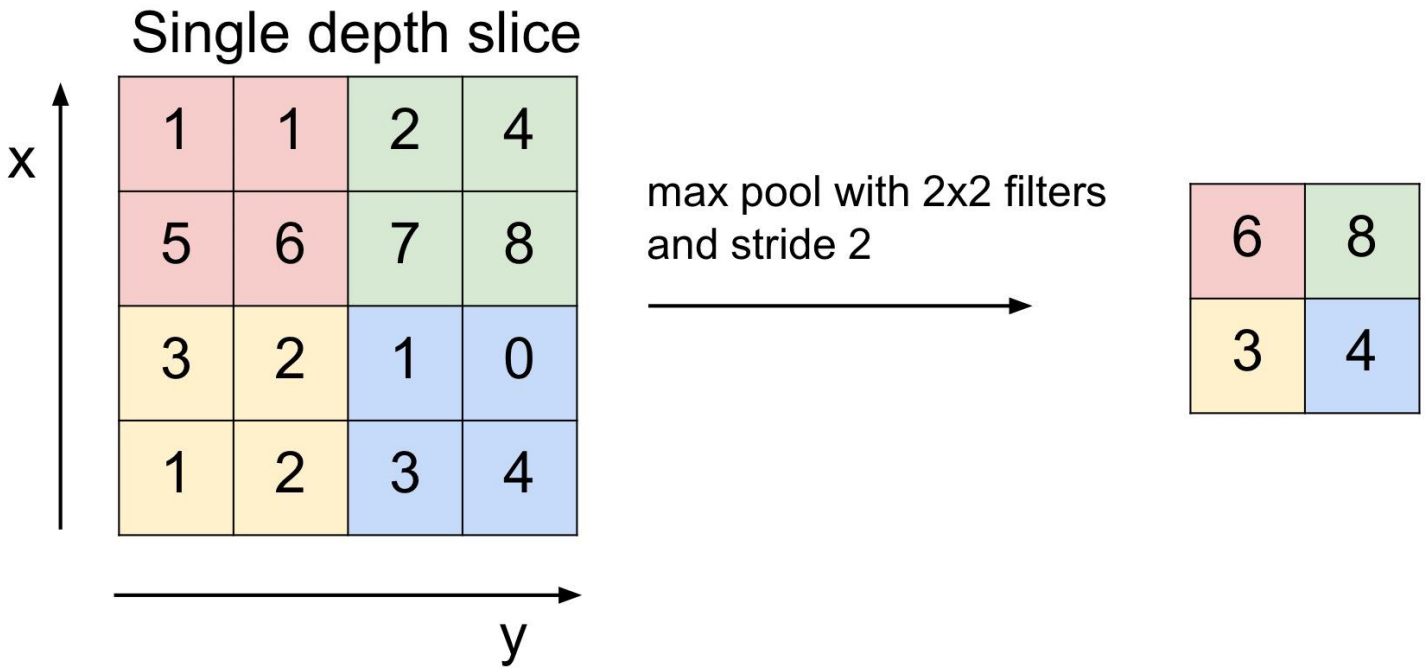
- makes the representations smaller and more manageable
- operates over each activation map independently:



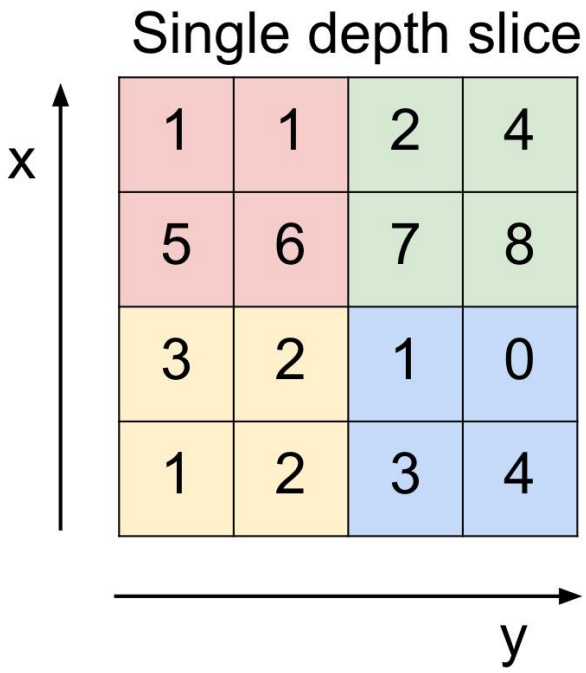
-- Also adds resilience to small geometric deformations.

-- A way to do spatial hierarchical processing (fine to coarse).

MAX POOLING


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

Mean Pooling



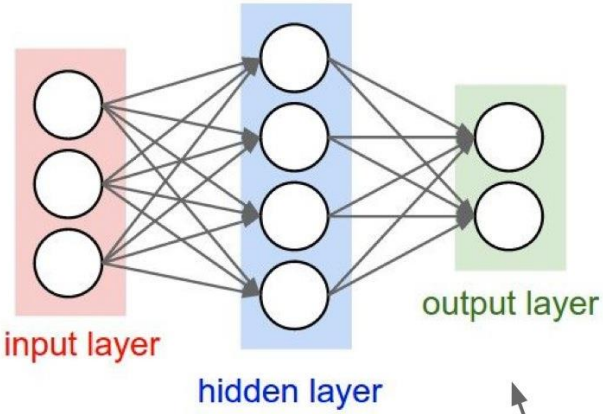
mean
~~max~~ pool with 2x2 filters
and stride 2

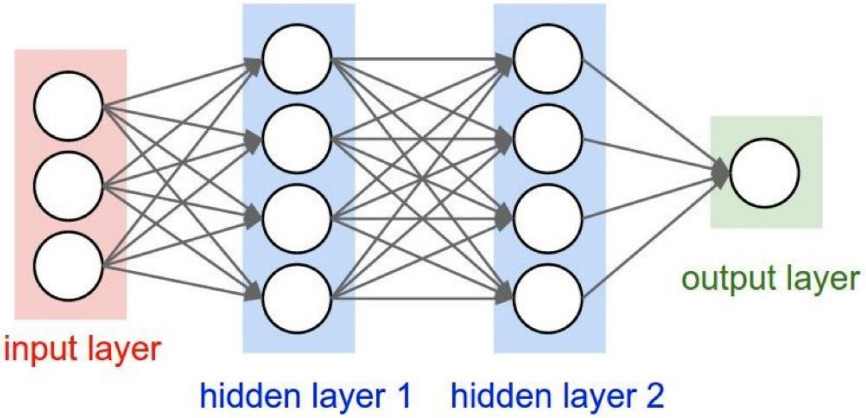
13/4	21/4
8/4	8/4

Overview

- Building Blocks
  - Convolution (actually cross-correlation)
  - Nonlinear Activation (e.g. ReLU)
  - Pooling (e.g. max-pool)
  - Fully Connected Layers
  - Softmax
- Goal is to understand forward pass computation

Neural Networks: Architectures



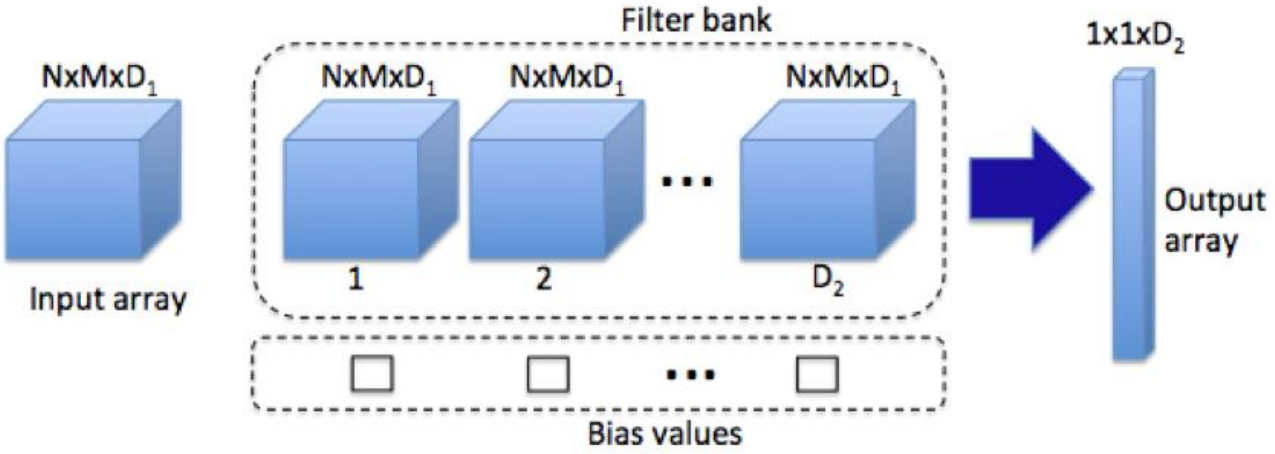


"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"3-layer Neural Net", or
"2-hidden-layer Neural Net"

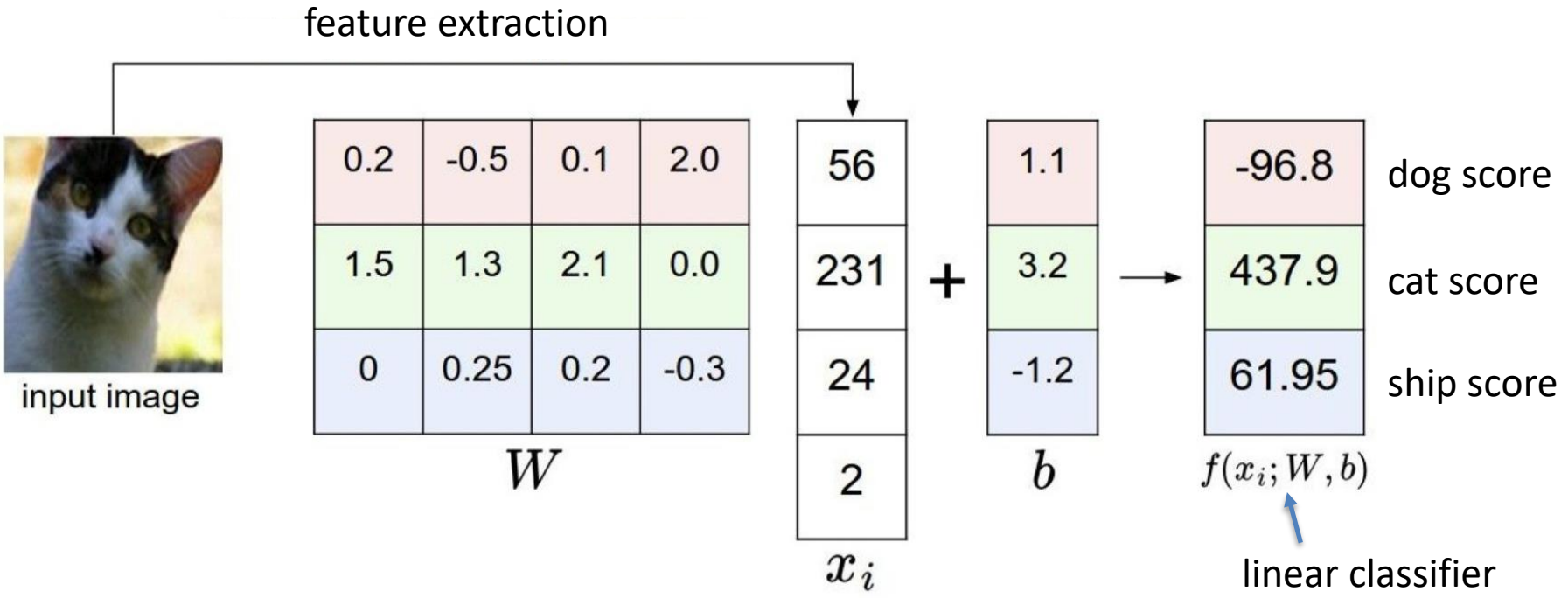
"Fully-connected" layers

Fully Connected Layer



I like to think of fully connected layers as a special case of convolution layers where the filter is the same size as the image, and applied in only one location (no sliding).

Classifying an image using four features into three classes...



Another way of visualizing fully connected layers.
Tie in with linear classifiers!!

Overview

- Building Blocks
  - Convolution (actually cross-correlation)
  - Nonlinear Activation (e.g. ReLU)
  - Pooling (e.g. max-pool)
  - Fully Connected Layers
  - Softmax
- Goal is to understand forward pass computation

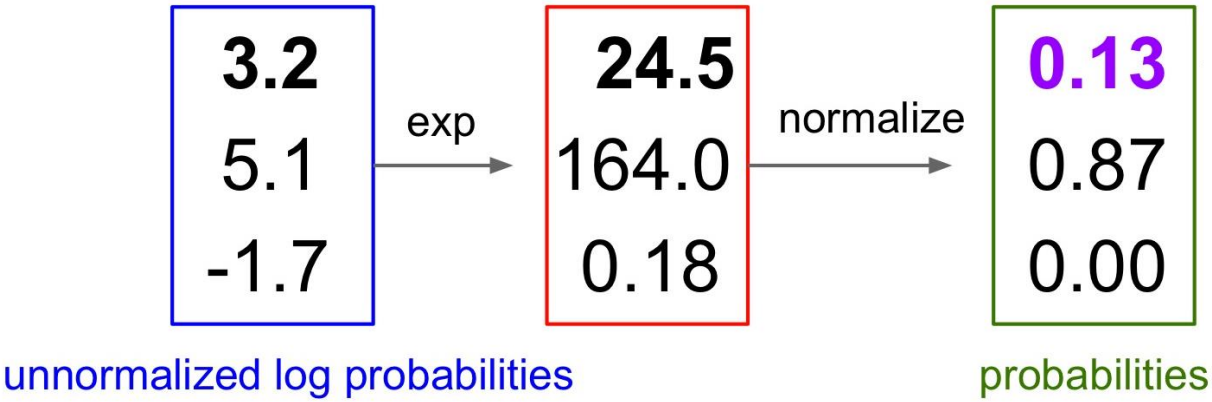
Softmax

- Turns an arbitrary set of numbers into “probabilities”.

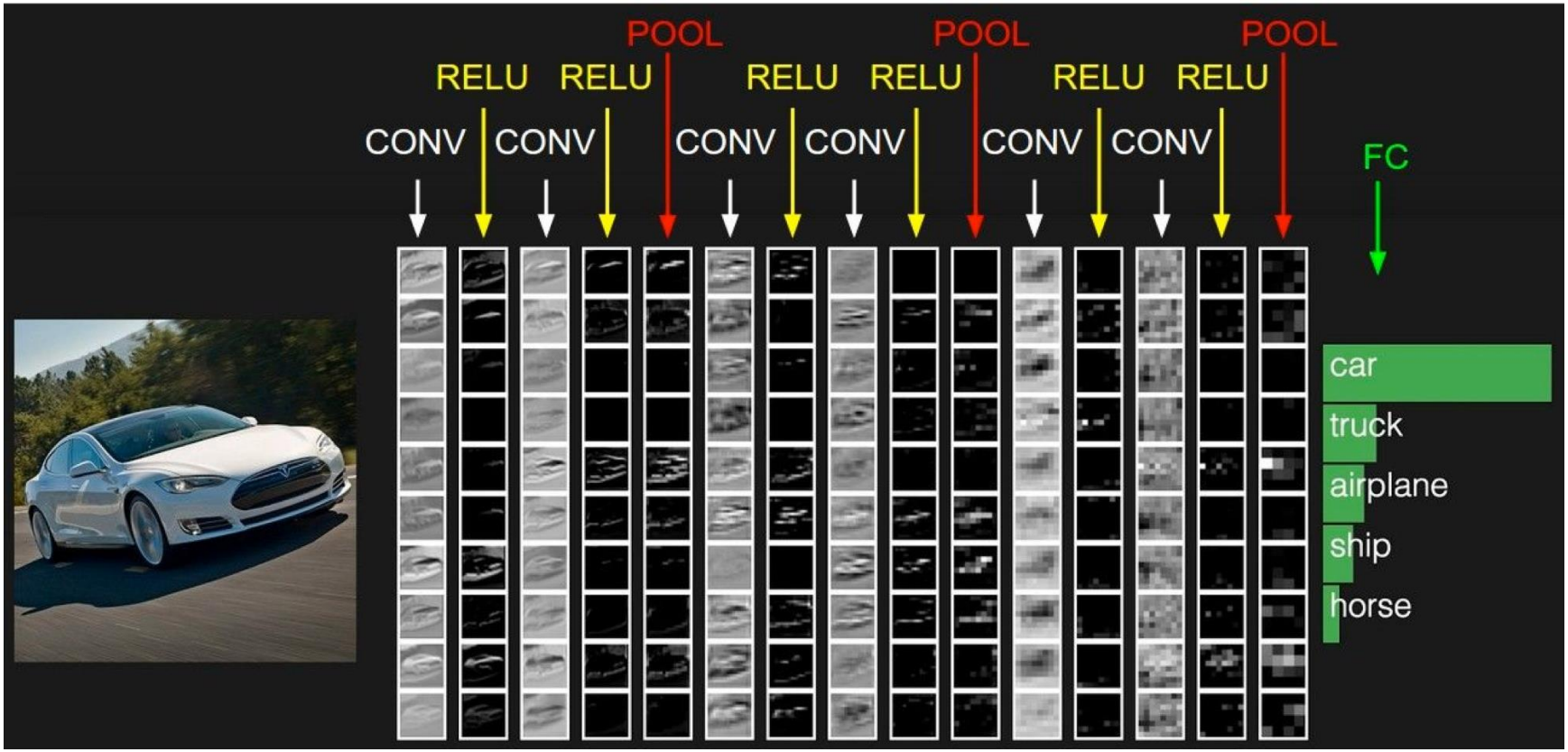
- $p_i = \exp(x_i) / \sum_{j=1 \text{ to } n} \exp(x_j)$

- For numerical stability, it is a good idea to first subtract max value from each x_i. i.e. $x_i' = x_i - \max(\{x_j, j=1,...n\})$.

Softmax


3.2
5.1
-1.7
exp
24.5
164.0
0.18
normalize
0.13
0.87
0.00
unnormalized log probabilities
probabilities

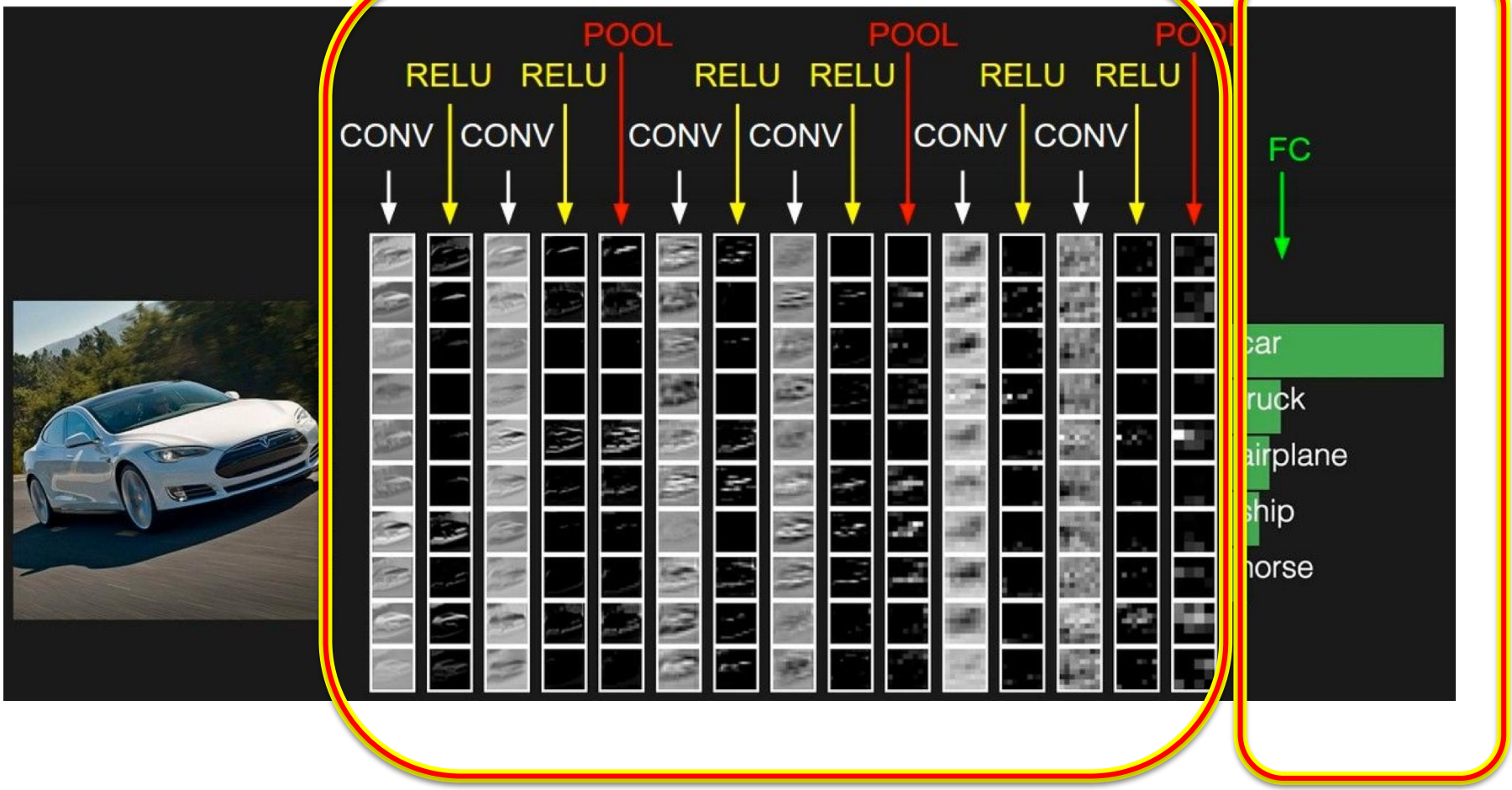
Putting it All Together



One Interpretation

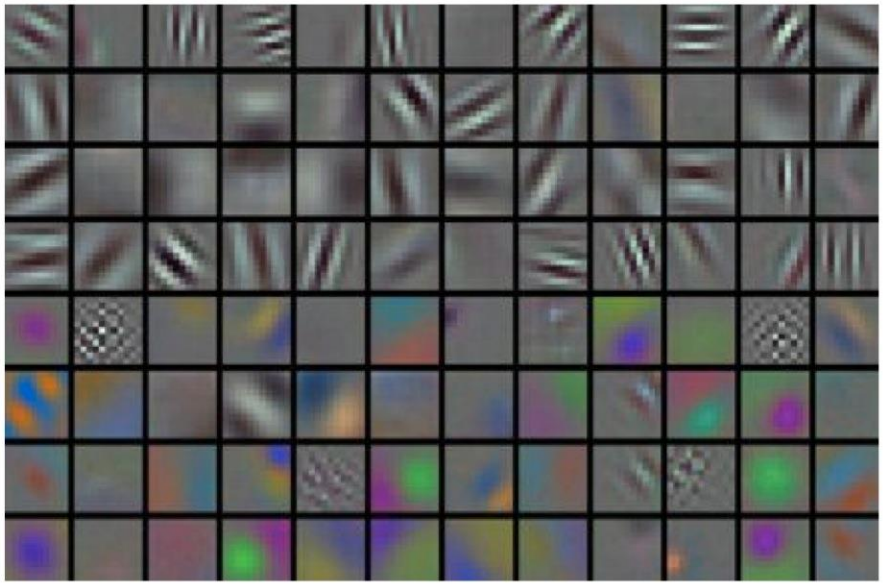
Extract features (form feature space the linear classifier operates on)

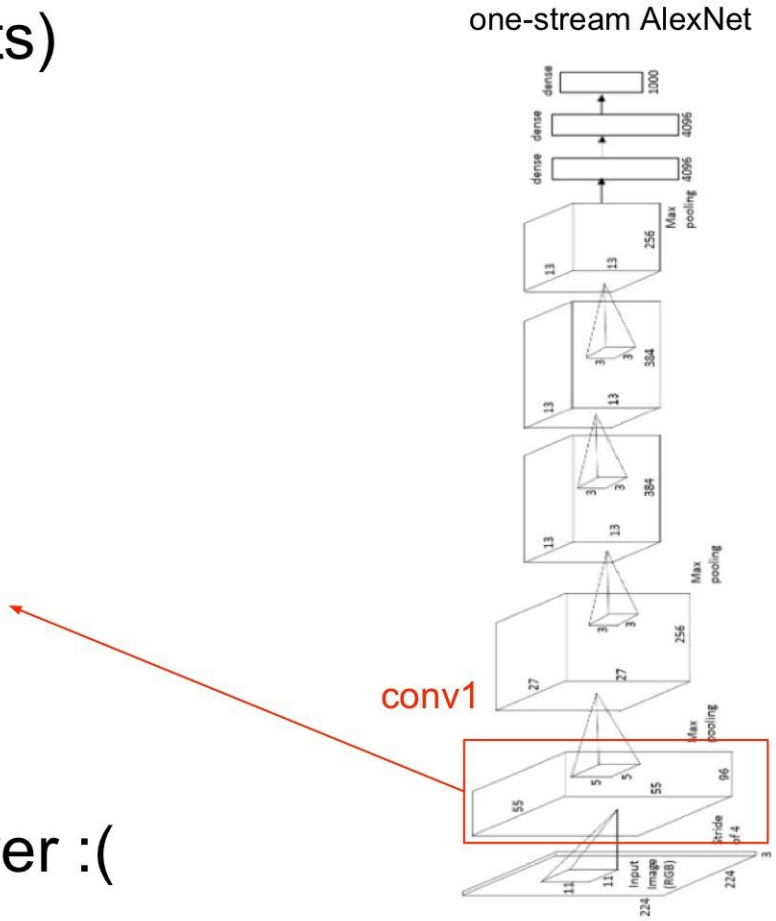
Apply multi-class linear classifier!



Learned Features

Visualize the filters/kernels (raw weights)





only interpretable on the first layer :(

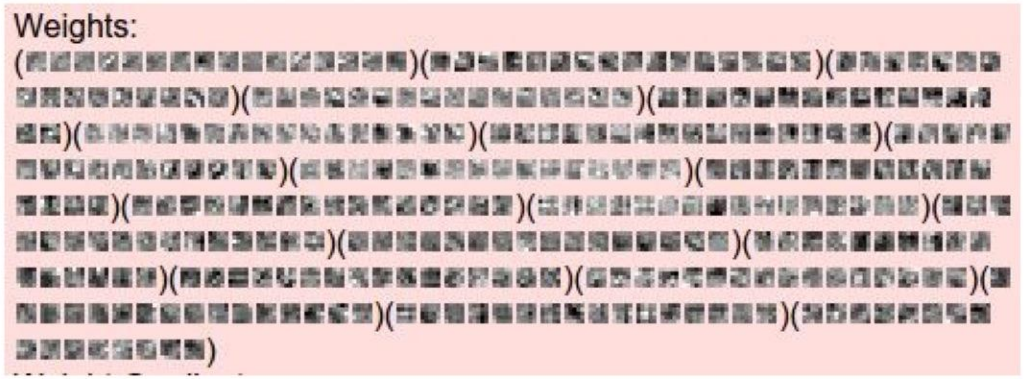
Notice that this is learning things that looks like derivative filters and color blob Gaussians.

Visualize the filters/kernels (raw weights)

you can still do it for higher layers, it's just not that interesting

(these are taken from ConvNetJS CIFAR-10 demo)

layer 1 weights



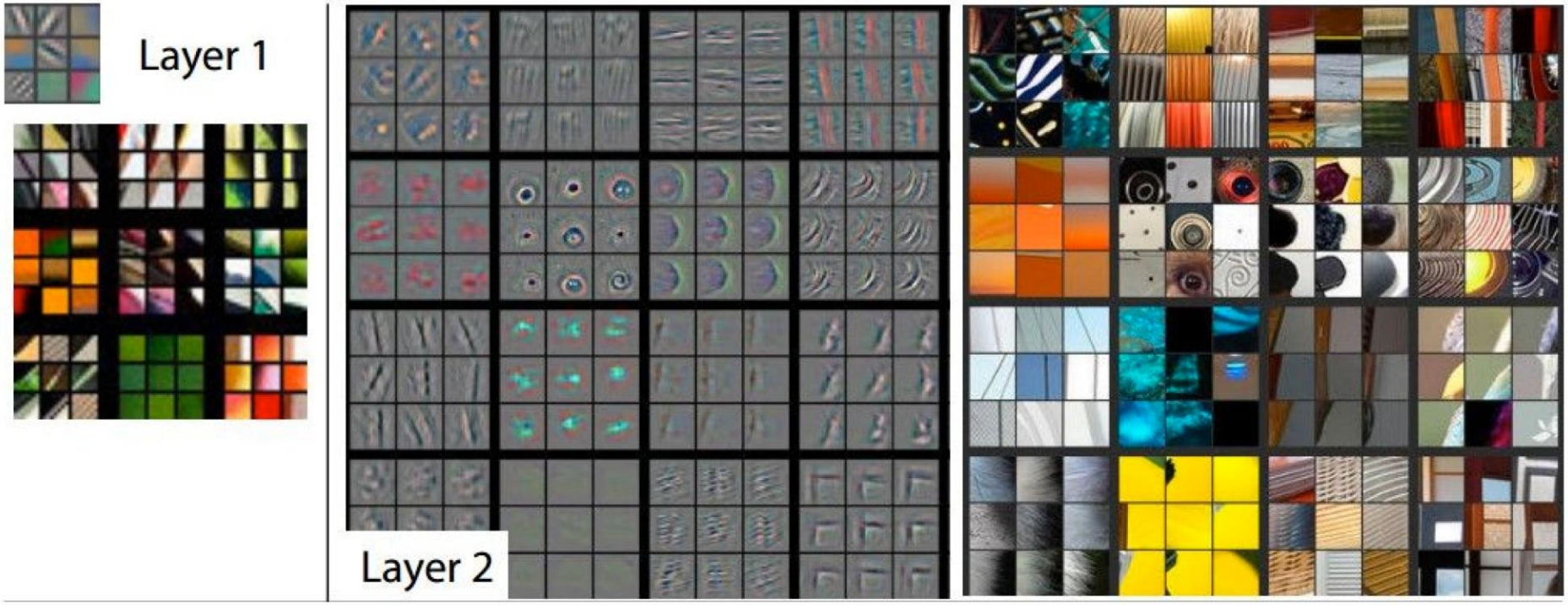
layer 2 weights

layer 3 weights

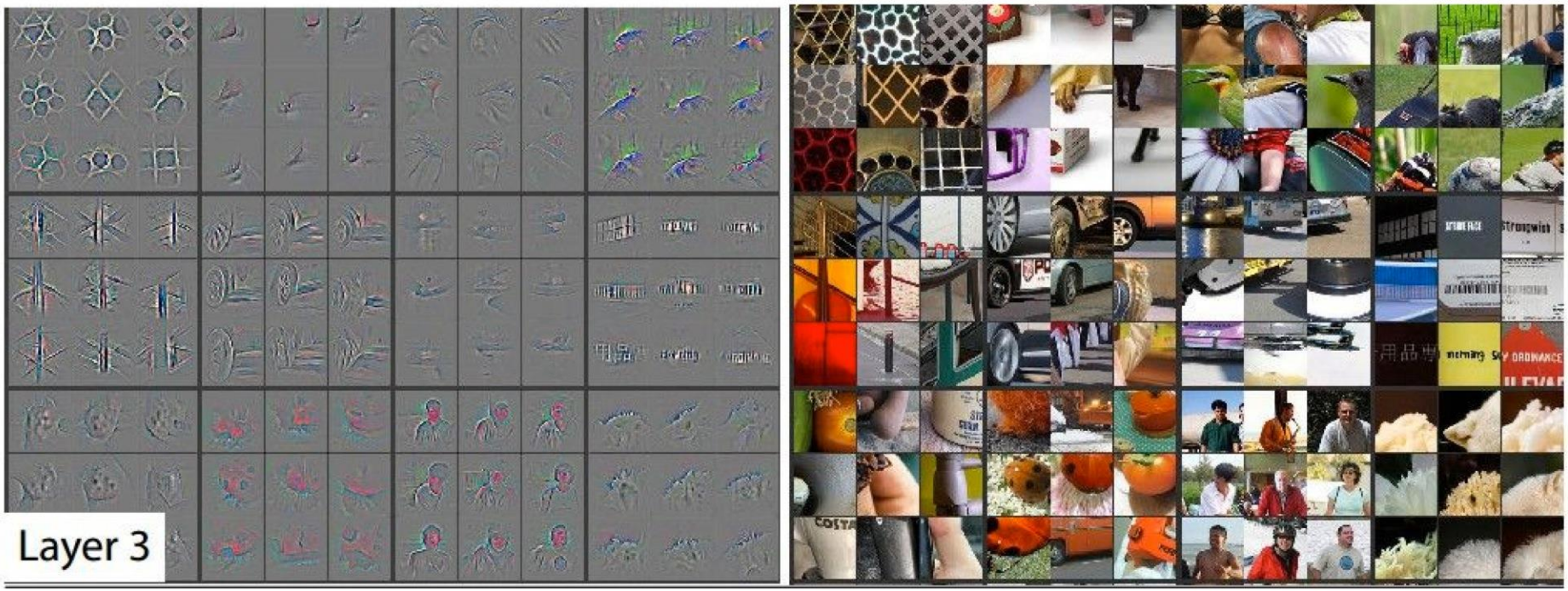
Weights in higher-level filters are rewarding or inhibiting lower-level features, not raw pixels directly.

Visualizing and Understanding Convolutional Networks
Zeiler & Fergus, 2013

Visualizing arbitrary neurons



Visualizing arbitrary neurons



Visualizing arbitrary neurons