#DLUPC

Day 3 Lecture 1

Convolutional Neural Networks

Verónica Vilaplana
veronica.vilaplana@upc.edu

Associate Professor
Universitat Politecnica de Catalunya
Technical University of Catalonia

Index

- Motivation:
  - Local connectivity
  - Parameter sharing
  - Pooling and subsampling
- Layers
  - Convolutional
  - Pooling
  - Fully connected
  - Activation functions
  - Batch normalization
  - Upsampling

Motivation

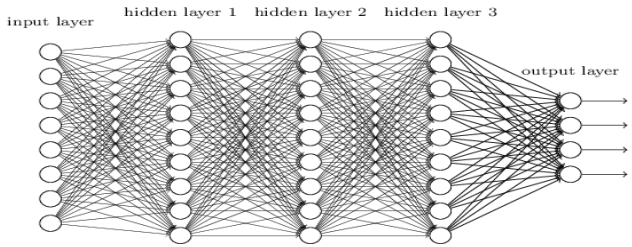
Neural networks for visual data

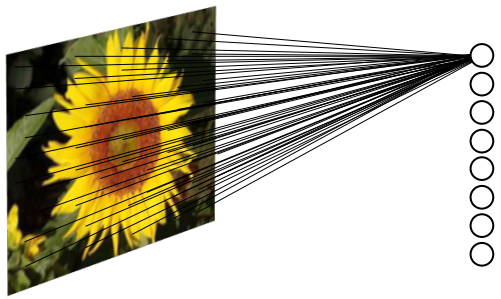
- **Example: Image recognition**
  - Given some input image, identify which object it contains

Caltech101 dataset

image size 150x112

sun flower





neurons connected to 16800 inputs

Neural networks for visual data

- We can design neural networks that are specifically adapted for such problems
  - must deal with very high-dimensional inputs
    - 150 x 112 pixels = 16800 inputs, or 3 x 16800 if RGB pixels
  - can exploit the 2D topology of pixels (or 3D for video data)
  - can build in invariance to certain variations we can expect
    - translations, illumination, etc.

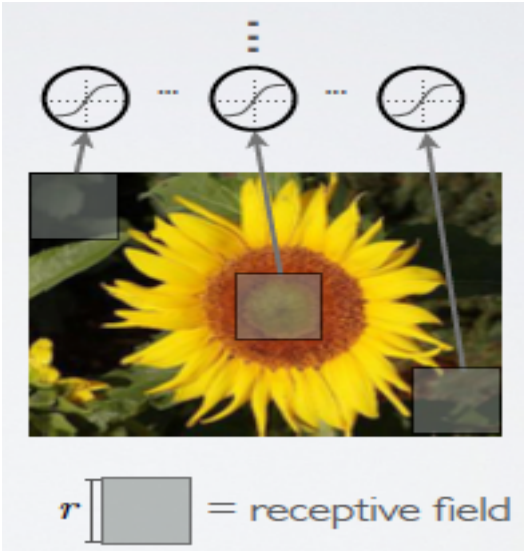
- **Convolutional networks are a specialized kind of neural network for processing data that has a known, grid-like topology. They leverage these ideas:**
  - local connectivity
  - parameter sharing
  - pooling / subsampling hidden units

S. Credit: H. Larochelle

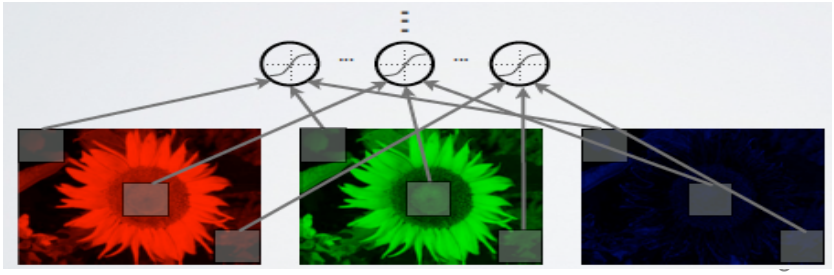
Convolutional neural networks

Local connectivity

- First idea: use a local connectivity of hidden units
  - each hidden unit is connected only to a subregion (patch) of the input image: **receptive field**
  - it is connected to all channels
    - 1 if greyscale image
    - 3 (R, G, B) for color image
    - …



- Solves the following problems:
  - fully connected hidden layer would have an **unmanageable** number of parameters
  - computing the linear activations of the hidden units would be very **expensive**

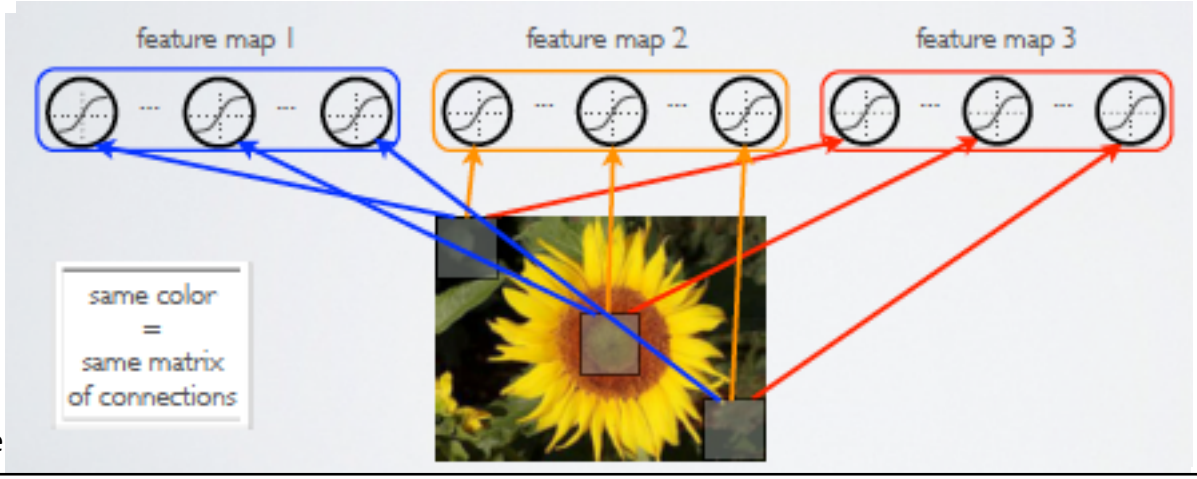


S. Credit: H. Larochelle

Convolutional neural networks

Parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same ‘‘feature map’’ share parameters
  - hidden units within a feature map cover different positions in the image
- Solves the following problems:
  - reduces even more the number of parameters
  - will extract the same features at every position (features are ‘‘equivariant’’)



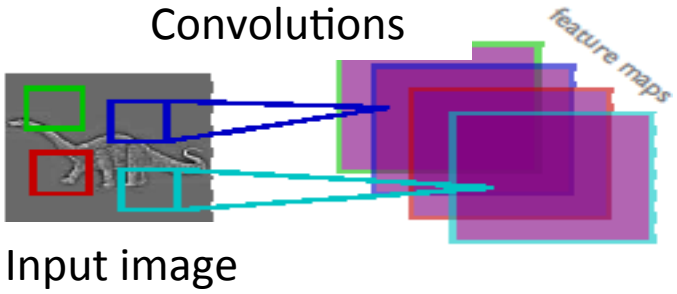
S. Credit: H. Larochelle

W_{ij} is the matrix connecting the ith input channel with the jth feature map

Convolutional neural networks

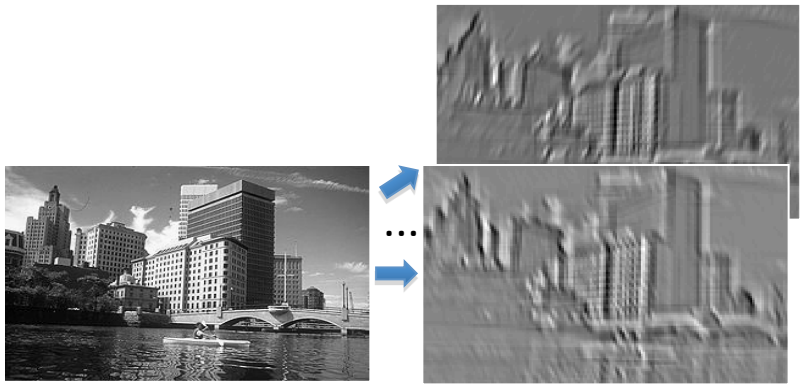
Parameter sharing

- Each feature map forms a 2D grid of features
  - can be computed with **a discrete 2D convolution** of a kernel matrix k_{ij}
    which is the hidden weights matrix W_{ij} with its rows and columns flipped



$$y_j = f(\sum_i k_{ij} * x_i)$$

x_i i-th channel of input, y_j hidden layer

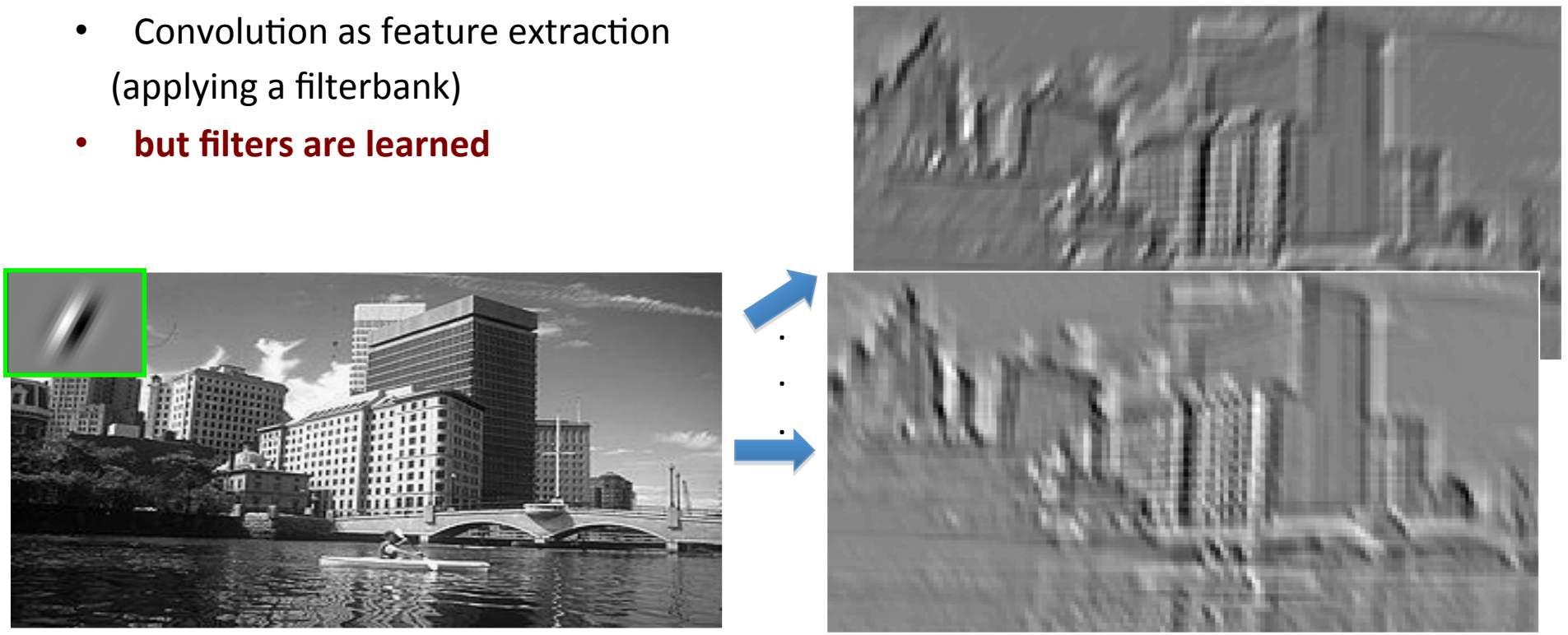


Input Feature Map

S. Credit: H. Larochelle

Convolutional neural networks

- Convolution as feature extraction (applying a filterbank)
- **but filters are learned**



Input

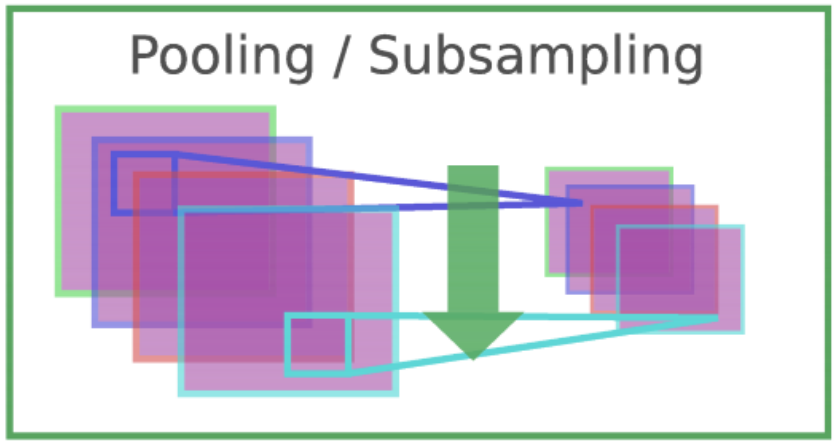
Feature Map

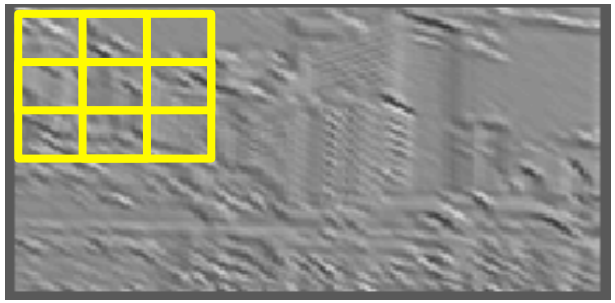
S. Credit: H. Larochelle

Convolutional neural networks

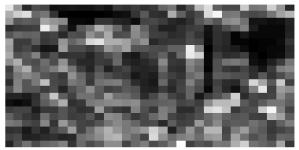
Pooling and subsampling

- Third idea: pool hidden units in same neighborhood
  - pooling is performed in non-overlapping neighborhoods (subsampling)
  - an alternative to ‘‘max’’ pooling is ‘‘average’’ pooling
  - pooling reduces dimensionality and provides invariance to small local changes





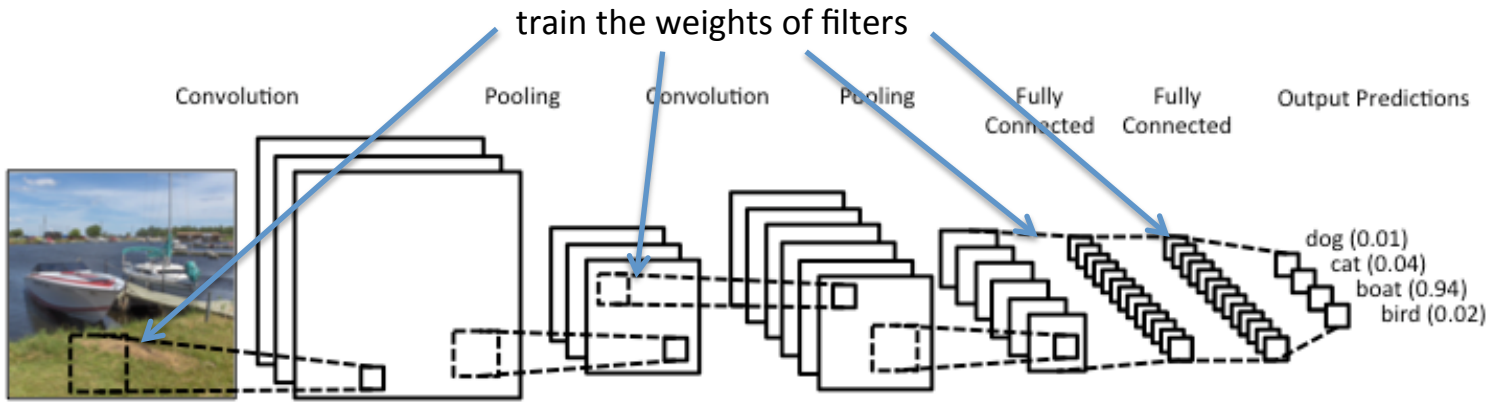
Max



S. Credit: H. Larochelle

Convolutional Neural Networks

- **Convolutional neural networks alternate between convolutional layers (followed by a nonlinearity) and pooling layers (basic architecture)**



- For recognition: output layer is a regular, fully connected layer with softmax non-linearity
  - output provides an estimate of the conditional probability of each class

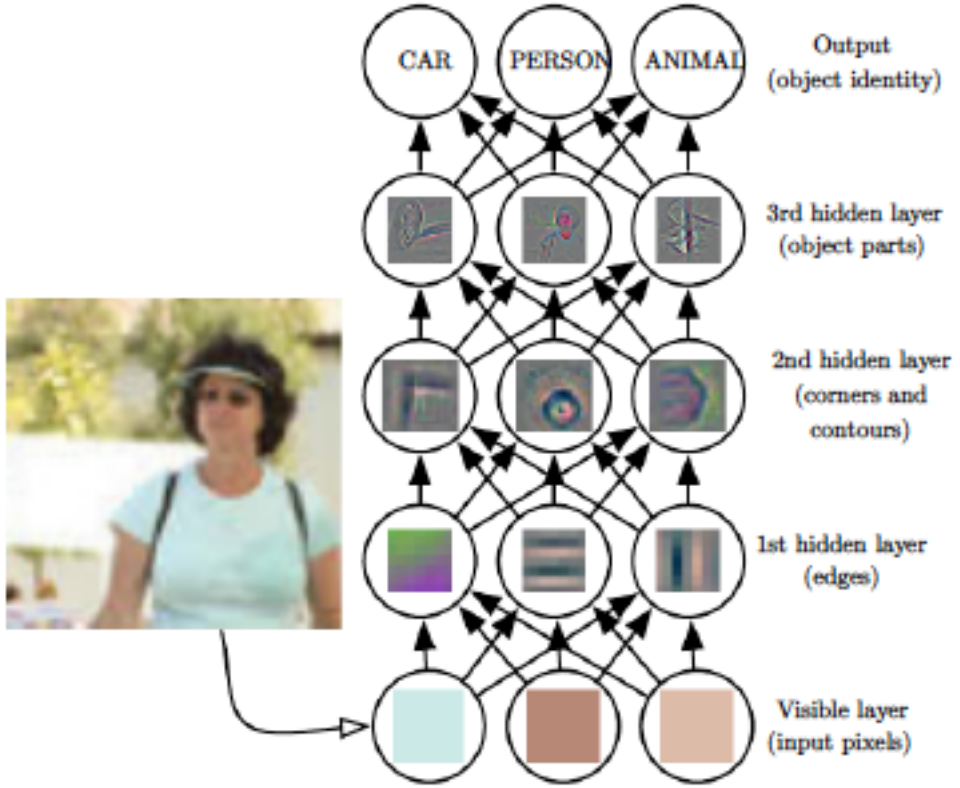
- The network is trained by stochastic gradient descent (& variants)
  - backpropagation is used similarly as in a fully connected network

S. Credit: H. Larochelle

Convolutional Neural Networks

CNN= learning hierarchical representations with increasing levels of abstraction

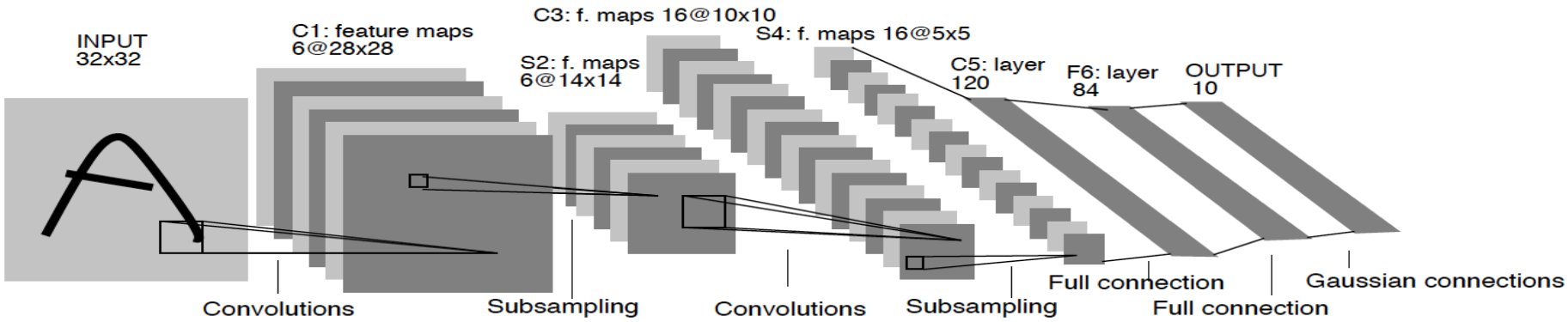
End-to-end training: joint optimization of features and classifier



F. Credit: DLBook

Example: LeNet-5

- LeCun et al., 1998



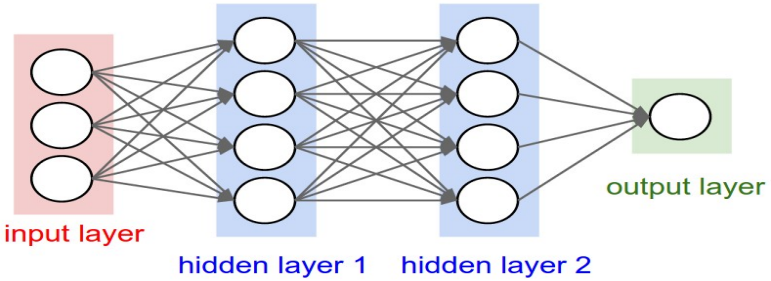
MNIST digit classification problem
handwritten digits
60,000 training examples
10,000 test samples
10 classes
28x28 grayscale images

Conv filters were 5x5, applied at stride 1
Sigmoid or tanh nonlinearity
Subsampling (average pooling) layers were 2x2 applied at stride 2
Fully connected layers at the end
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

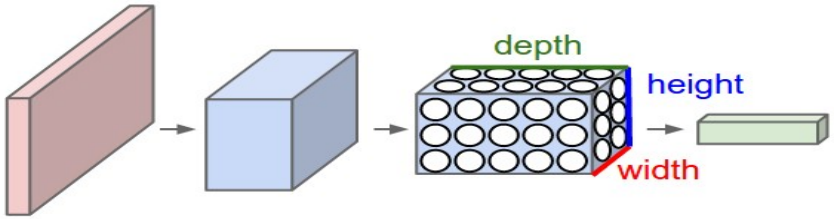
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

Layers

Convolutional Neural Networks



A regular 3-layer Neural Network



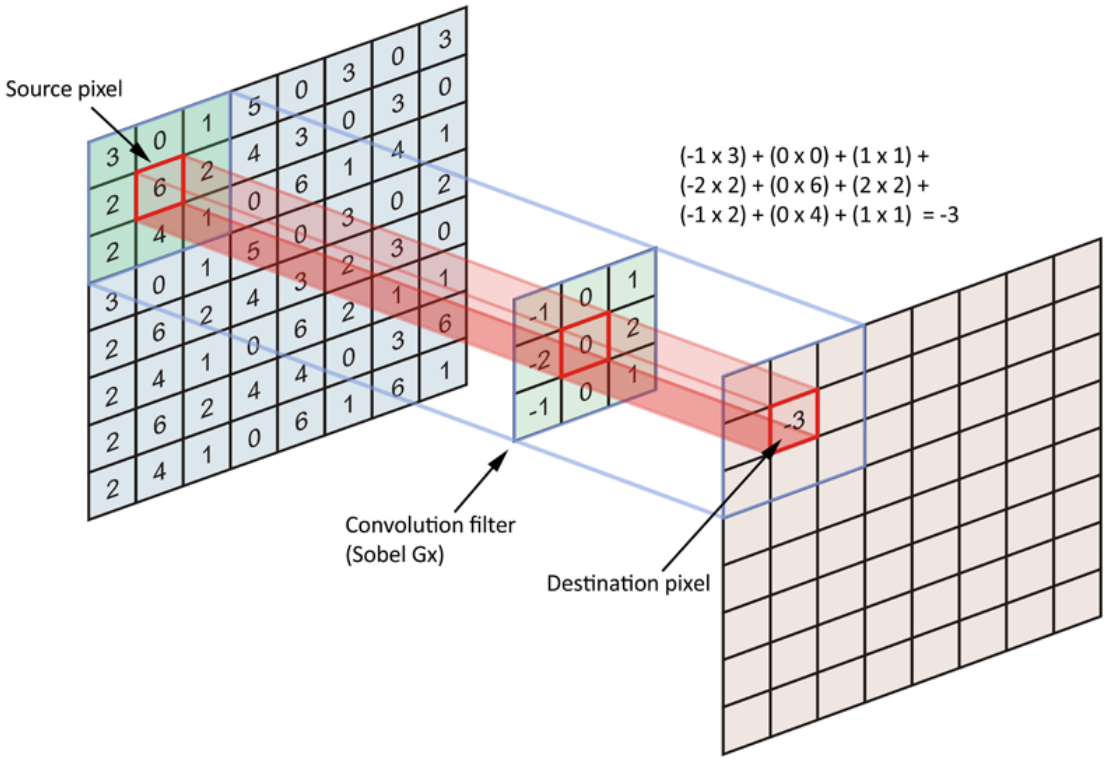
A ConvNet with 3 layers

- In ConvNets inputs are 'images' (architecture is constrained)
- A ConvNet arranges neurons in three dimensions (width, height, depth)
- Every layer transforms 3D input volume to a 3D output volume of neuron activations

S. Credit: Stanford cs231_2017

Convolutional layer

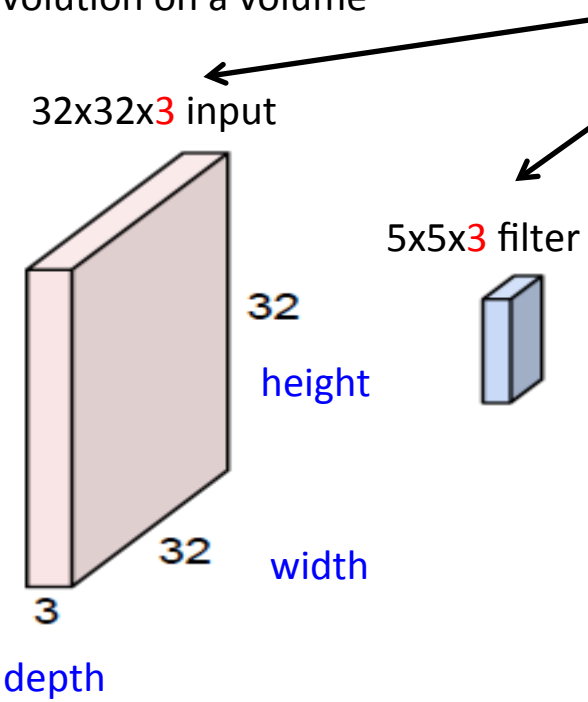
- Convolution on a 2D grid



[Image source]

Convolutional layer

- Convolution on a volume



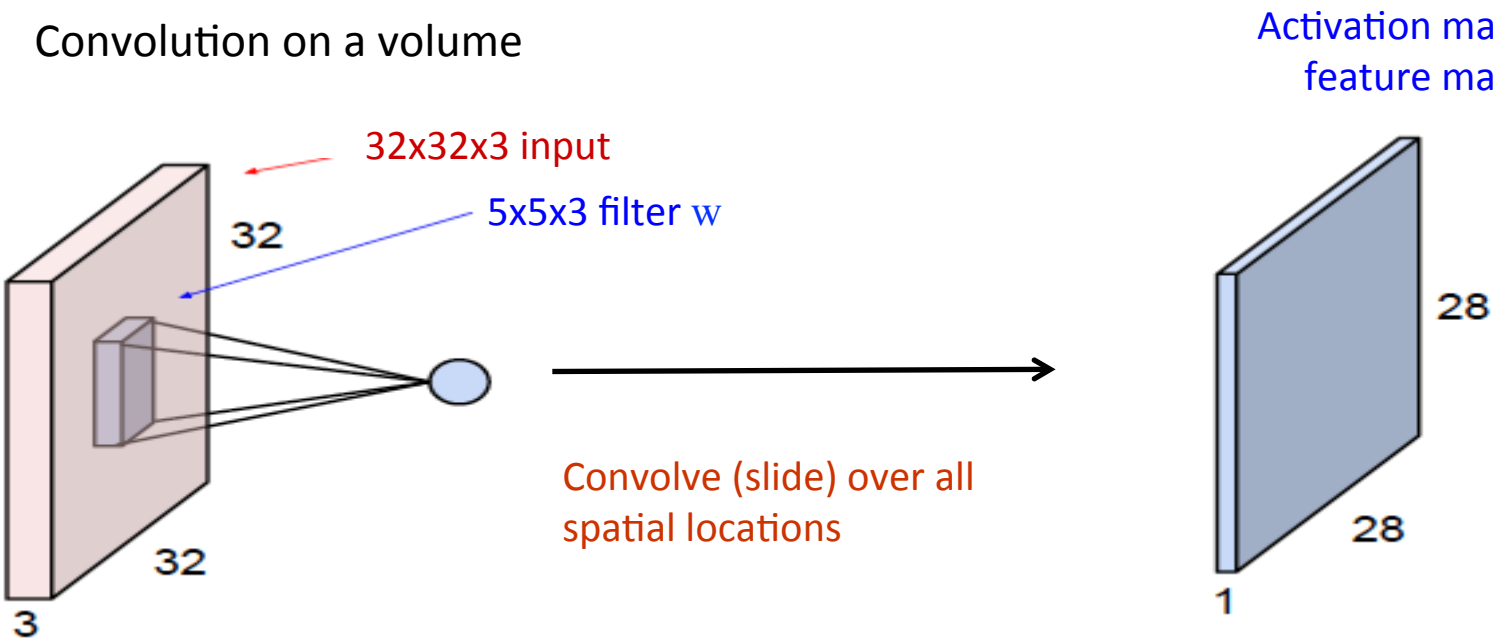
Filters always extend the full detph of the input volume

Convolve the filter with the input
i.e. slide over the input spatially, computing dot products

S. Credit: Stanford cs231_2017

Convolutional layer

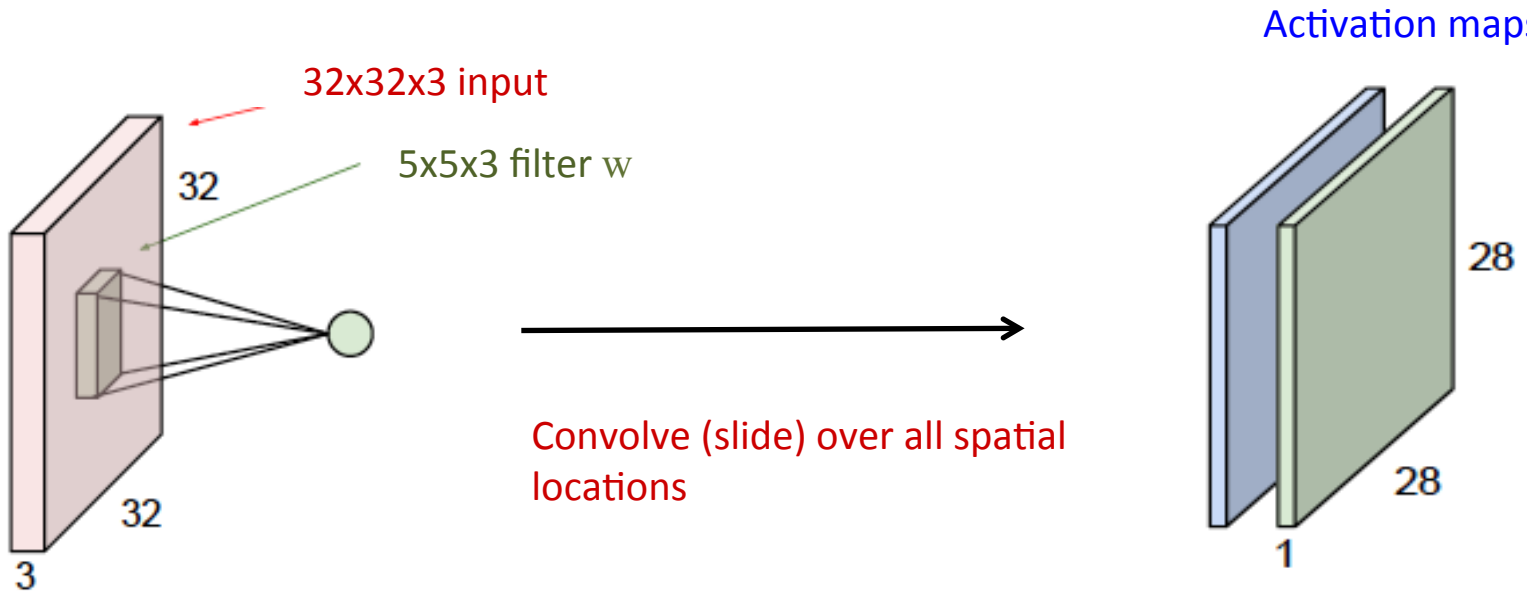
- Convolution on a volume



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the input: 5x5x3=75-dim dot product + bias $w^t x+b$

S. Credit: Stanford cs231_2017

Convolutional layer

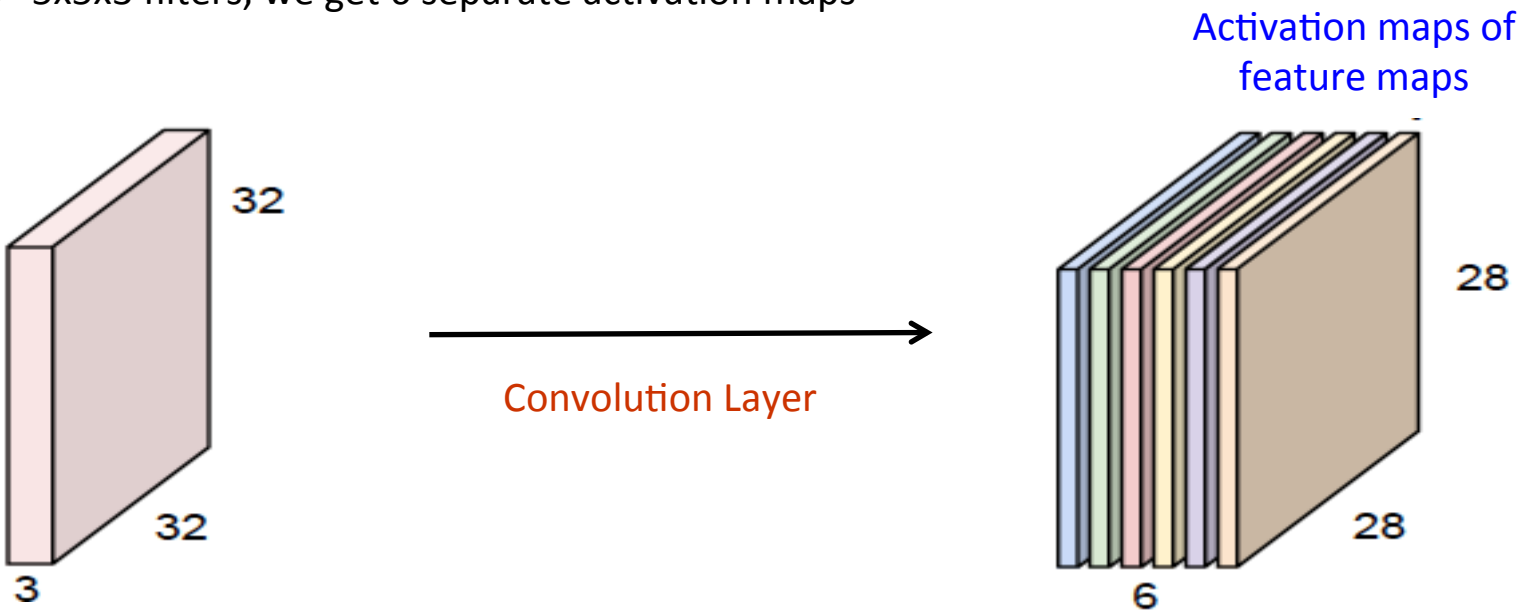


Consider a second, green filter

S. Credit: Stanford cs231_2017

Convolutional layer

If we have 6 5x5x3 filters, we get 6 separate activation maps



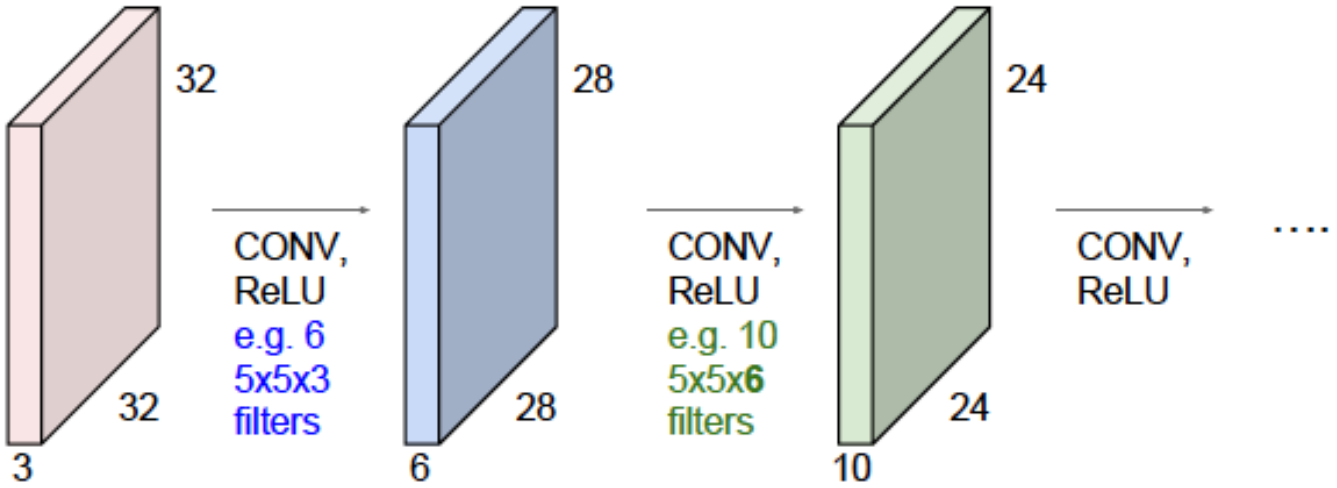
We stack the maps up to get a 'new volume' of size 28x28x6

S. Credit: Stanford cs231_2017

So applying a filterbank to an input (3D matrix) yields a cube-like output, a 3D matrix in which each slice is an output of convolution with one filter.

Convolutional layer

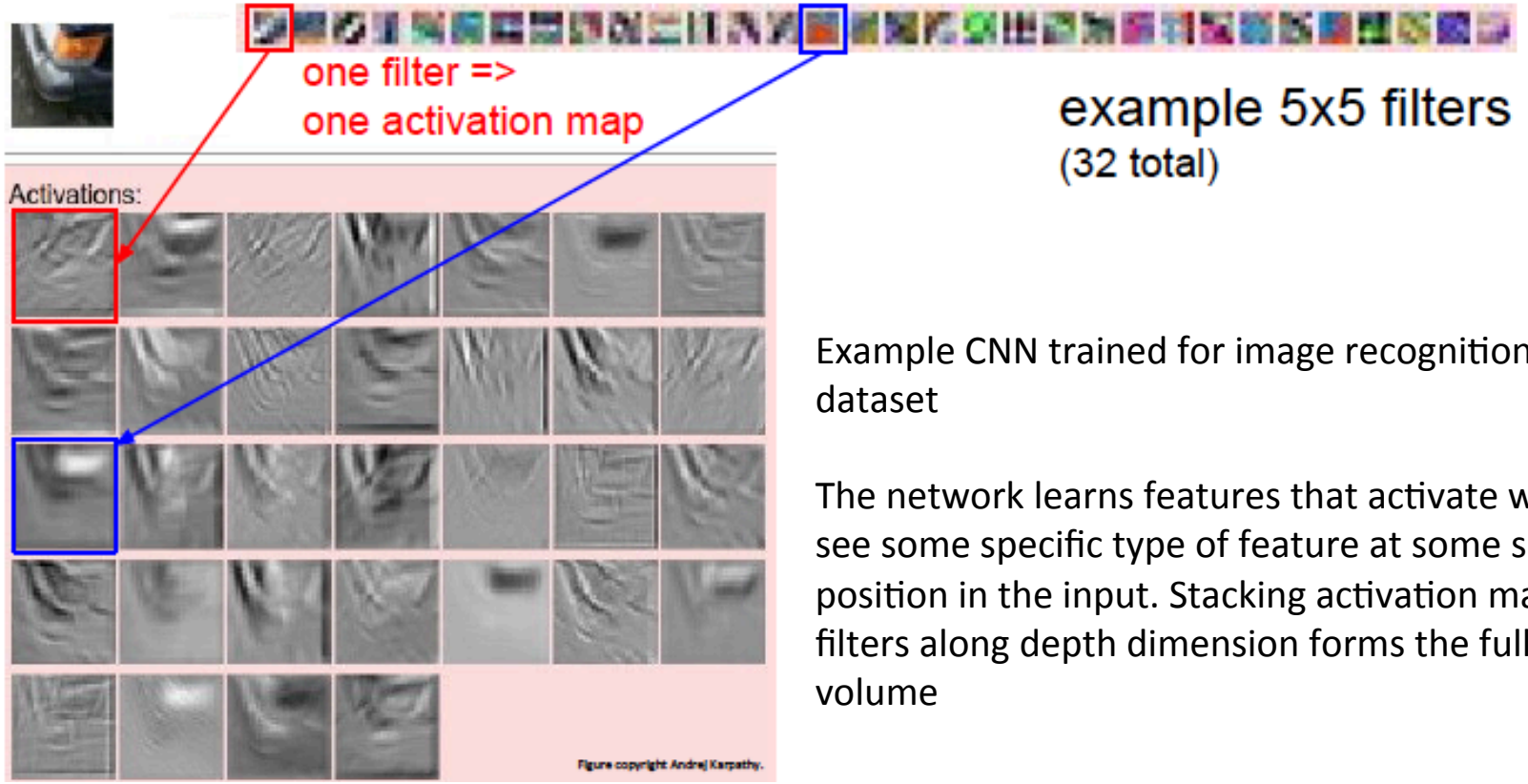
ConvNet is a sequence of Convolutional Layers, interspersed with activation functions and pooling layers (and a small number of fully connected layers)



We add more layers of filters. We apply filters (convolutions) to the output volume of the previous layer. The result of each convolution is a slice in the new volume.

S. Credit: Stanford cs231_2017

Example: filters and activation maps



Example CNN trained for image recognition on CIFAR dataset

The network learns features that activate when they see some specific type of feature at some spatial position in the input. Stacking activation maps for all filters along depth dimension forms the full output volume

http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

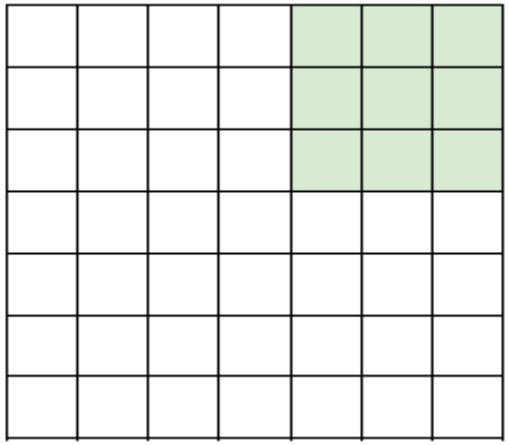
Convolution

Hyperparameters: filter spatial extent F, stride S, padding P

Stride is the number of pixels by which we slide the filter matrix over the input matrix.

Larger stride will produce smaller feature maps.

7

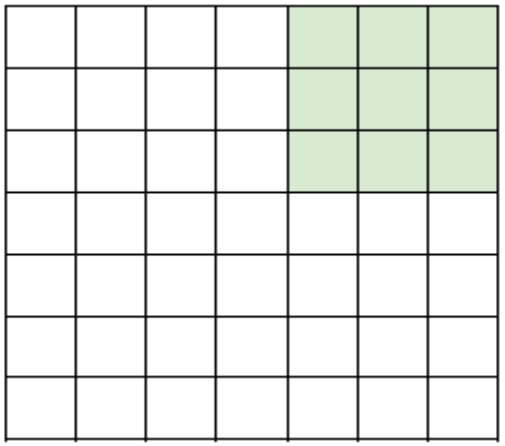


7

7x7 input (spatially)

assume 3x3 filter: 5x5 output (stride 1)

7



7

7x7 input (spatially)

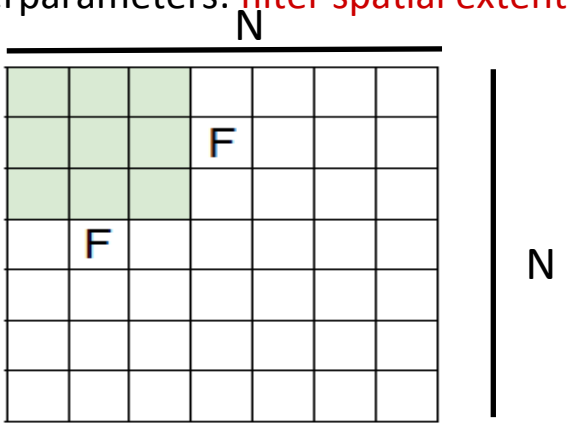
assume 3x3 filter

applied with stride 2: 3x3 output

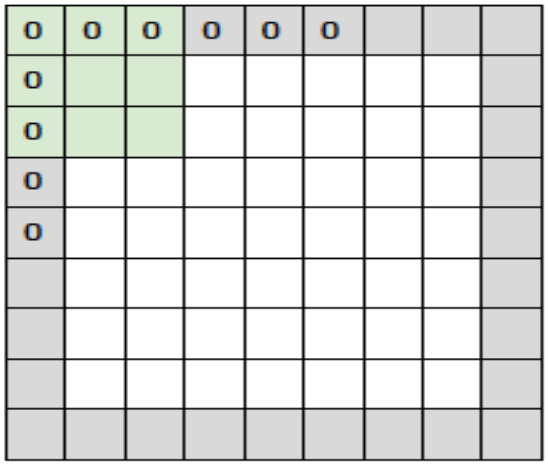
S. Credit: Stanford cs231_2017

Convolution

Hyperparameters: filter spatial extent F, stride S, padding P



Output size: (N-F)/ S + 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 not applied



common: zero-padding in the border

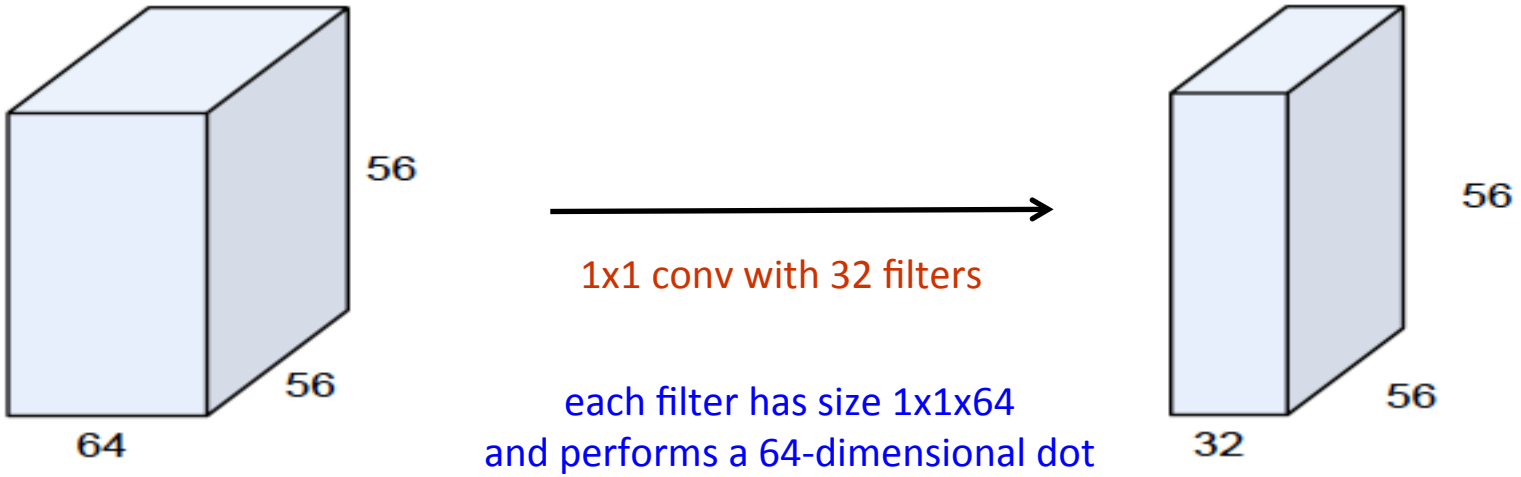
E.g. N=7, F=3 , stride 1, pad with 1 pixel border: output size 7x7

In general, common to see CONV layers with stride 1, filters FxF, zero-padding with P= (F-1)/2:
preserve size spatially
Output size: (N-F+2P) / S +1

S. Credit: Stanford cs231_2017

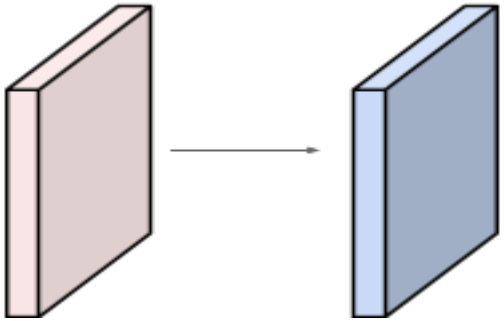
1x1 convolutions

1x1 convolution layers: used to reduce dimensionality (number of feature maps)



S. Credit: Stanford cs231_2017

Example: size, parameters



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

Output volume size:
(N + 2P – F) / S + 1
(32+2*2-5)/1+1= 32 spatially so 32x32x10

Number of parameters in this layer:
each filter has 5x5x3 + 1 =76 params (+1 for bias)
-> 76x10 = 760 parameters

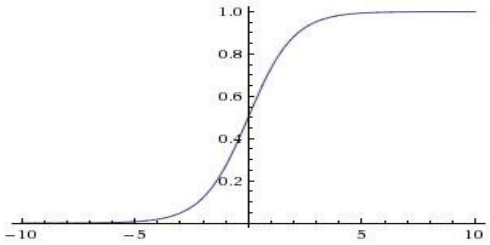
S. Credit: Stanford cs231_2017

Activation functions

- **Desirable properties:** mostly smooth, continuous, differentiable, fairly linear

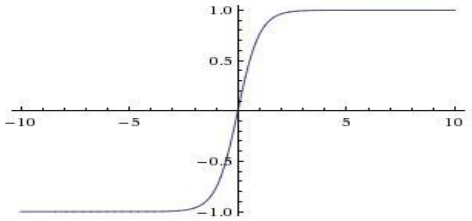
Sigmoid

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



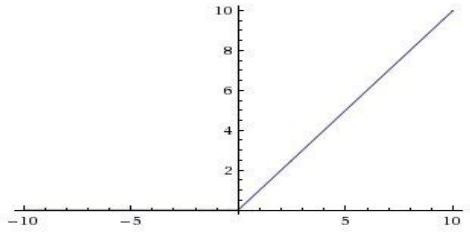
tanh

$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



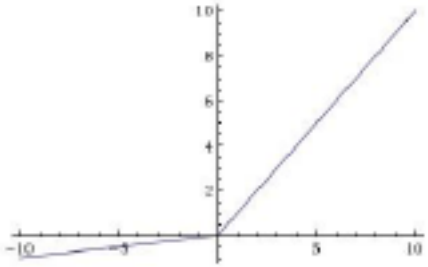
ReLu

$$\max(0, x)$$



Leaky ReLu

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

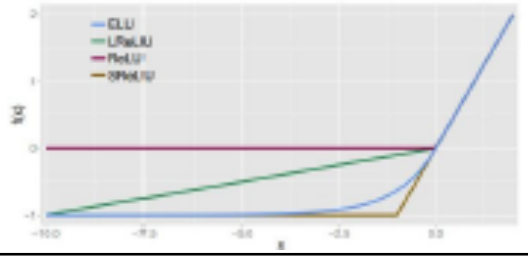


Maxout

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

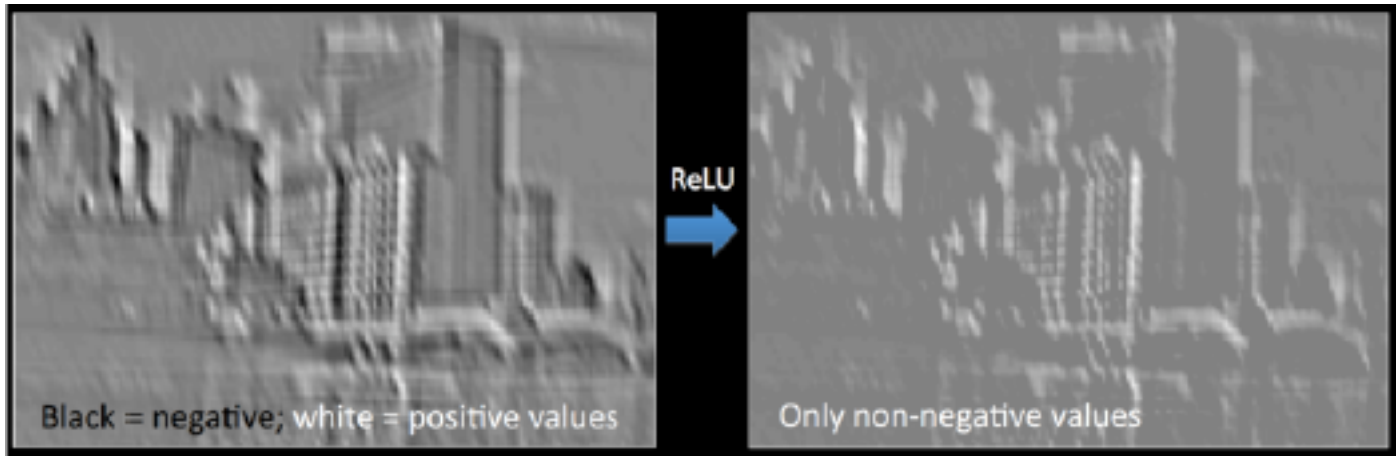
ELU

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



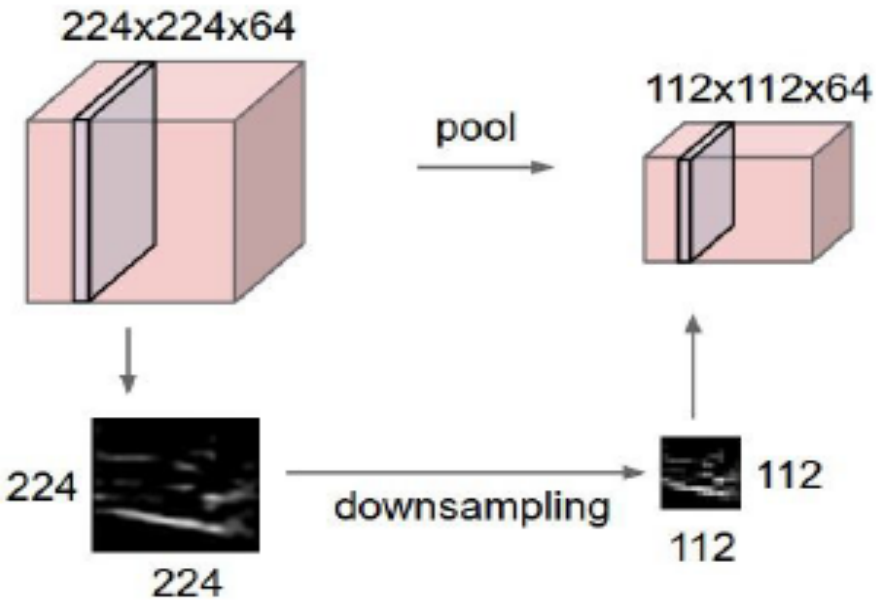
Activation functions

- Example



Pooling layer

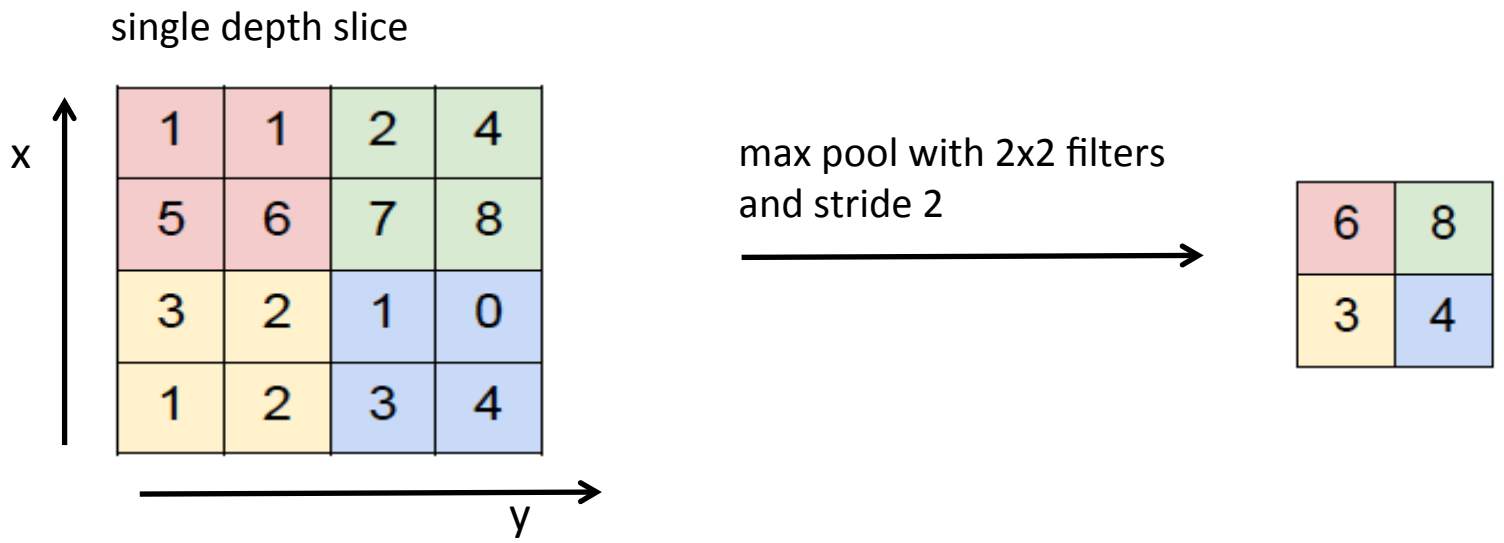
- Makes the representations smaller and more manageable for later layers
- Useful to get **invariance to small local changes**
- Operates over each activation map independently



S. Credit: Stanford cs231_2017

Pooling layer

- **Max pooling**

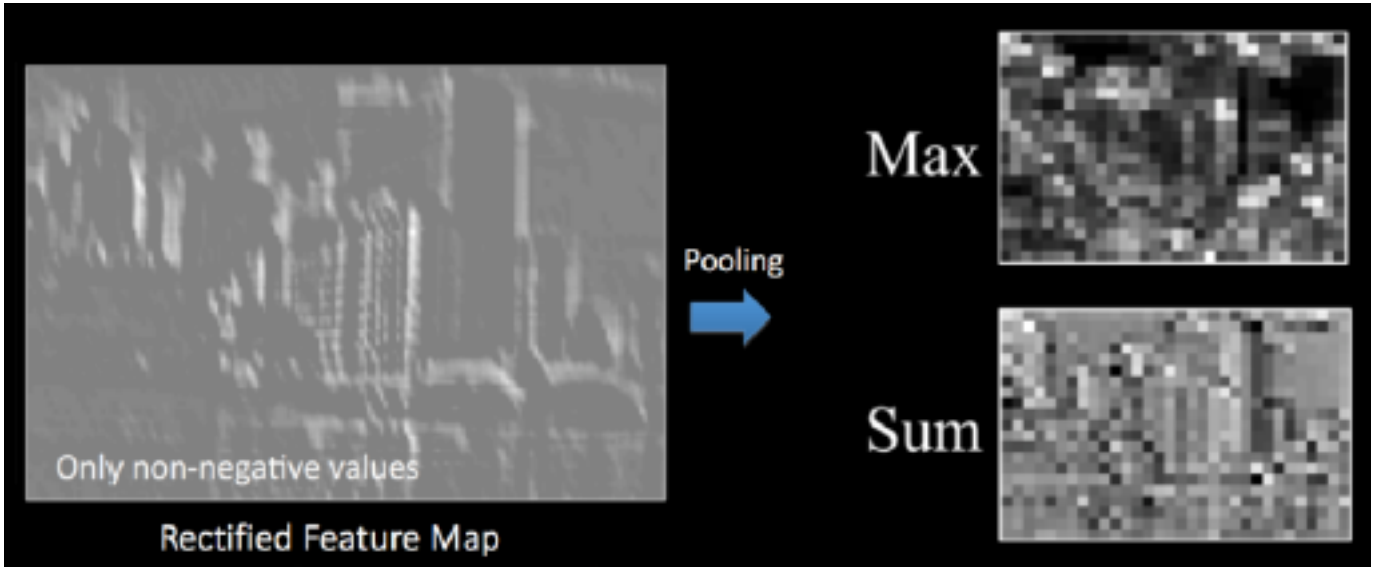


- Other pooling functions**: average pooling or L2-norm pooling**

S. Credit: Stanford cs231_2017

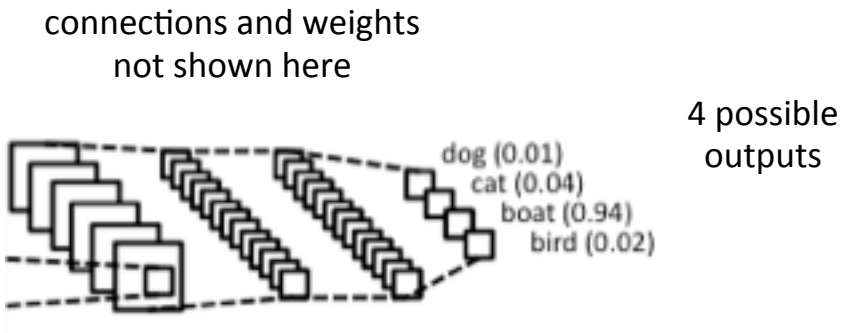
Pooling layer

- Example



Fully connected layer

- In the end it is common to add one or more fully (or densely) connected layers.
- Every neuron in the previous layer is connected to every neuron in the next layer (as in regular neural networks). Activation is computed as matrix multiplication plus bias
- At the output, softmax activation for classification



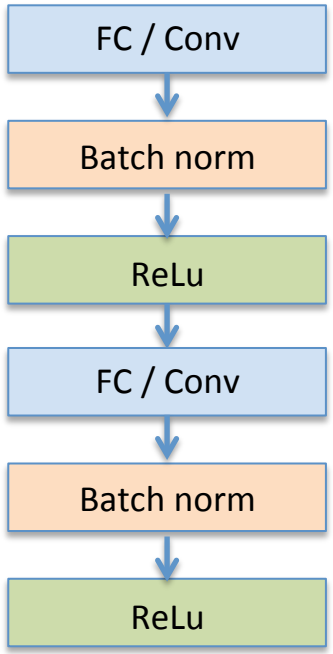
Batch normalization layer

- As learning progresses, the distribution of the layer inputs changes due to parameter updates (internal covariate shift)
- This can result in most inputs being in the non-linear regime of the activation function, slowing down learning
- Bach normalization is a technique to reduce this effect
  - Explicitly force the layer activations to have zero mean and unit variance w.r.t running batch estimates

$$\hat{x}^{(k)} = \frac{x^{(k)} - E(x^{(k)})}{\sqrt{\text{var}(x^{(k)})}}$$

  - Adds a learnable scale and bias term to allow the network to still use the nonlinearity

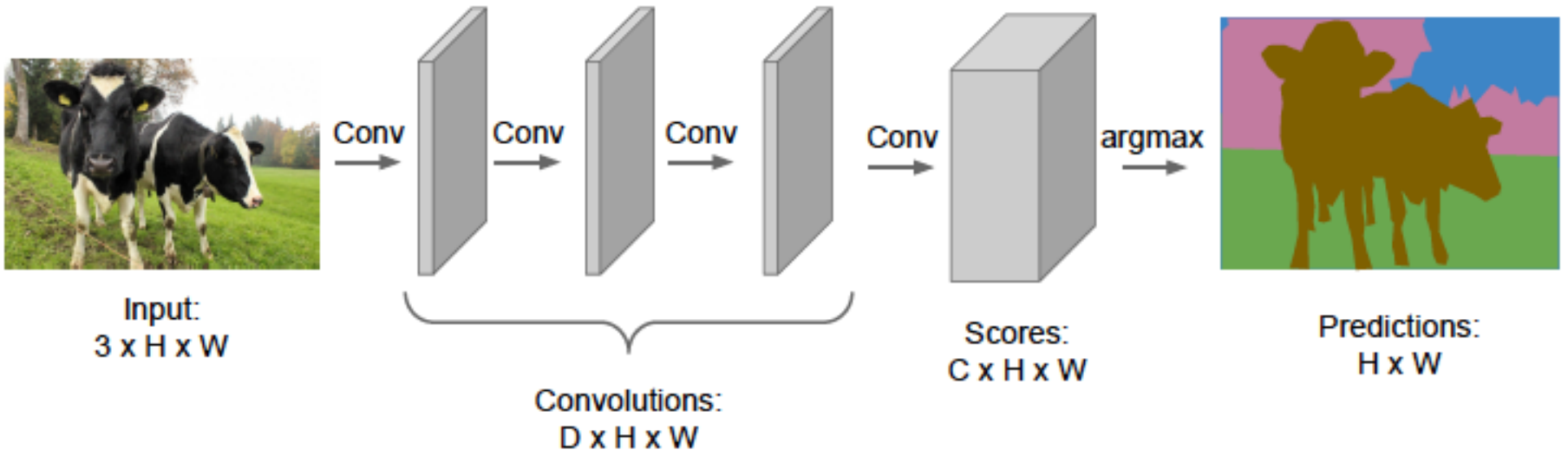
$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$$



Ioffe and Szegedy, 2015. "Batch normalization: accelerating deep network training by reducing internal covariate shift"

Upsampling layers: recovering spatial shape

- Motivation: semantic segmentation. Make predictions for all pixels at once

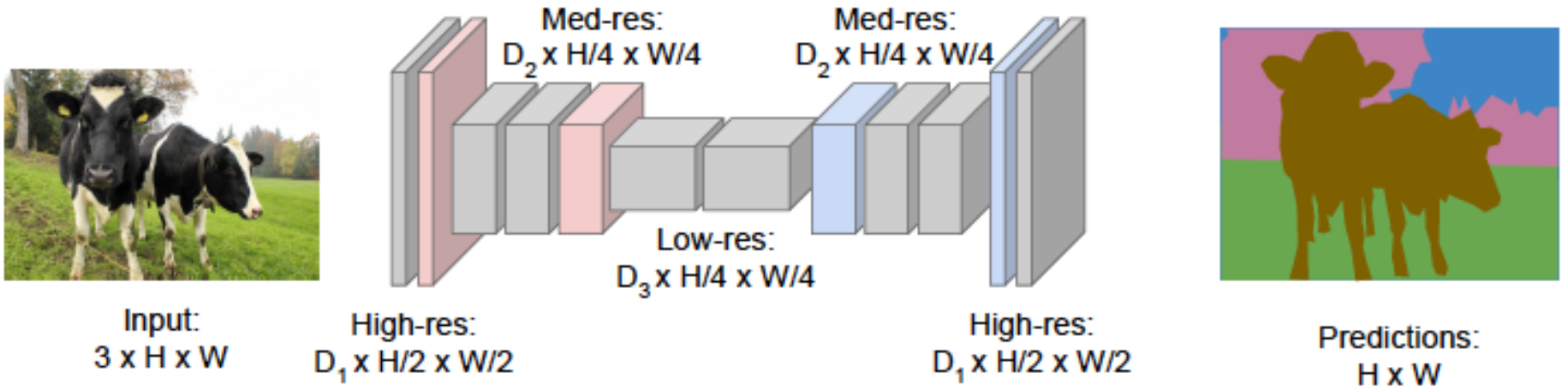


Problem: convolutions at original image resolution will be very expensive

S. Credit: Stanford cs231_2017

Upsampling layers: recovering spatial shape

- Motivation: semantic segmentation. Make predictions for all pixels at once



Design a network as a sequence of convolutional layers with downsampling **and upsampling layers**

More info:

Dumoulin et al, A guide to convolution arithmetic for deep learning, 2016

Shi, Is the deconvolution layer the same as a convolutional layer?, 2016

S. Credit: Stanford cs231_2017

Summary

- Convolutional neural networks are a specialized kind of neural network for processing data that has a known, grid-like topology

- CNNs leverage these ideas:
  - local connectivity
  - parameter sharing
  - pooling / subsampling hidden units

- Layers: convolutional, non-linear activation, pooling, batch normalization, upsampling