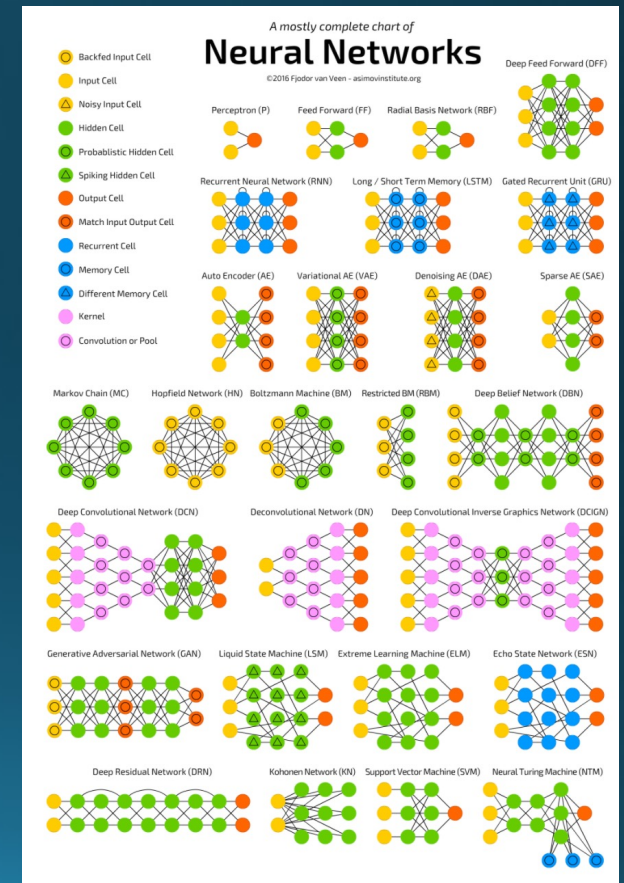


CSCI 4360/6360 Data Science II

Convolutional Neural Networks

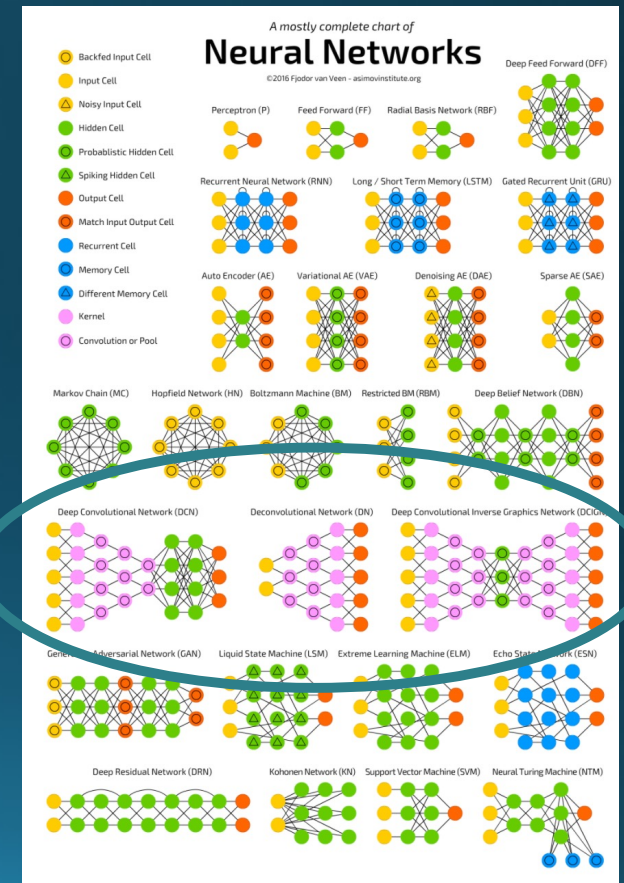
The Neural Network Zoo

- <http://www.asimovinstitute.org/neural-network-zoo/>



The Neural Network Zoo

- <http://www.asimovinstitute.org/neural-network-zoo/>



Convolution

- Basically a fancy way of saying “multiplication”
- Originally devised to make non-differentiable signals differentiable
- KDE is related to convolution
- For an input function f and convolutional filter g :

$$f \circledast g$$

scipy.signal.convolve

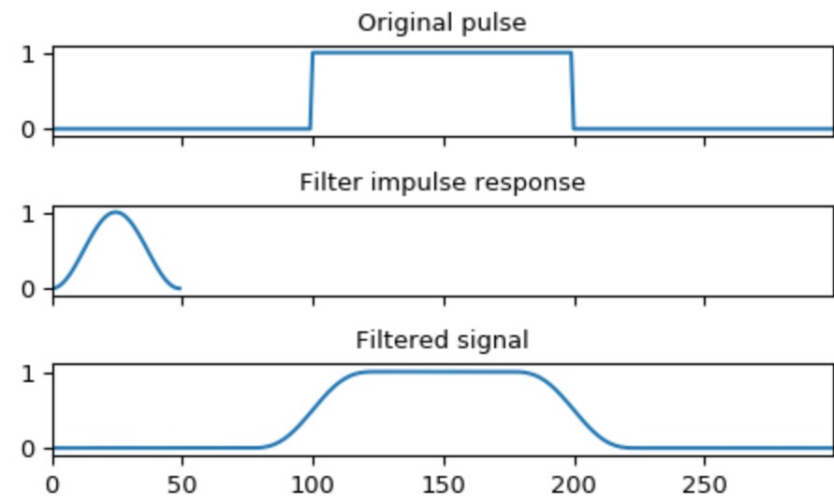
```
scipy.signal.convolve(in1, in2, mode='full', method='auto')
```

Convolve two N-dimensional arrays.

Convolve *in1* and *in2*, with the output size determined by the *mode* argument.

Parameters:

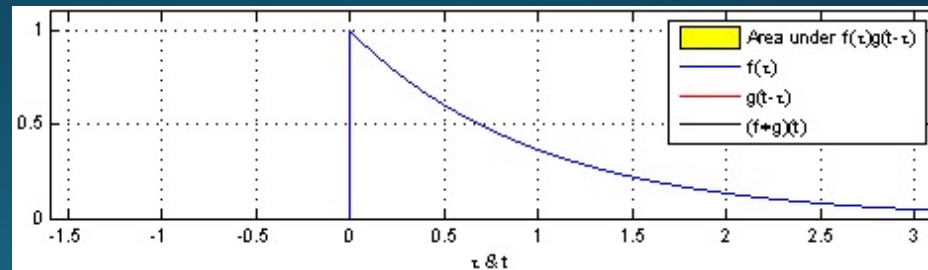
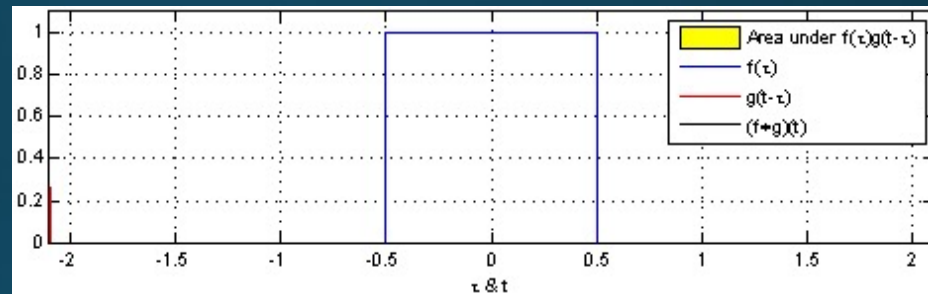
- in1* : *array_like*
First input.
- in2* : *array_like*
Second input. Should have the same number of dimensions as *in1*.
- mode* : *str* {'full', 'valid', 'same'}, optional
A string indicating the size of the output:



Convolution

- Can be viewed as an *integral transform*
 - One of the signals is shifted

$$\begin{aligned}(f \circledast g)(t) &= \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau \\ &= \int_{-\infty}^{\infty} f(t - \tau)g(\tau)d\tau\end{aligned}$$



Convolution in 2D

- 2D convolutions are critical in computer vision
- Basic idea is still the same
 - Choose a kernel
 - Run kernel over image
 - Build a representation of the convolved image (likely an intermediate representation)
- Lots of applications

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0



Image

4		

Convolved
Feature

Convolution in 2D

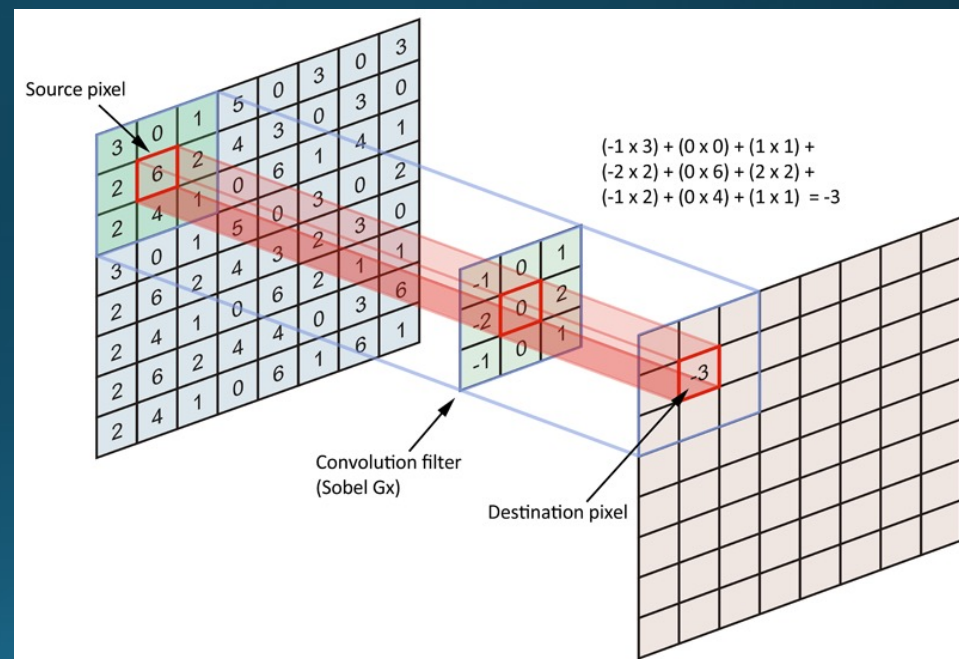
- Specific kernels can highlight different image features

Input image	Convolution Kernel	Feature map
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

- This kernel is an **edge detector** (others can be smoothers, sharpeners, etc)

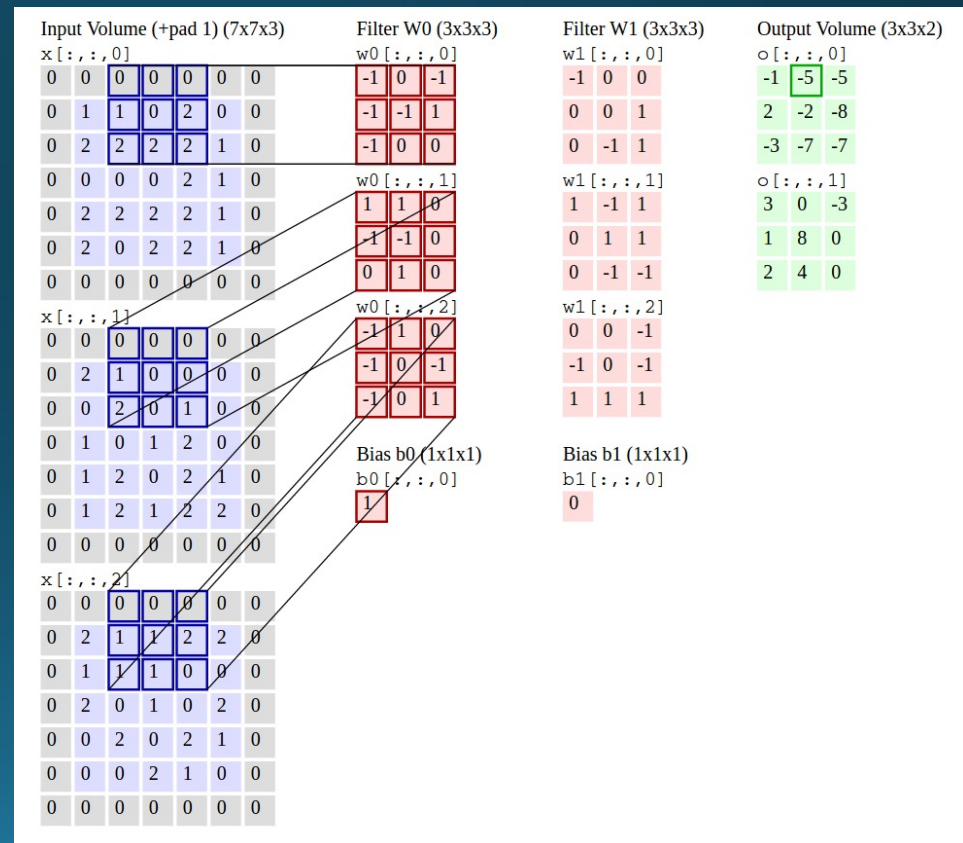
Convolution in 2D

- Works basically the same as 1D
- Filter / kernel computes a dot product with underlying pixels
- Generates an output
- Shift kernel and repeat



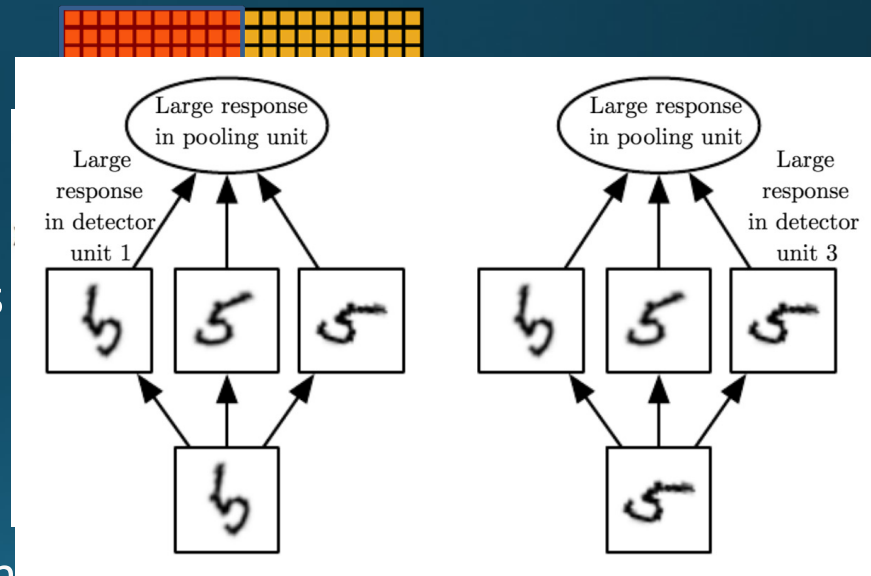
Convolution in 2D

- **Stride** dictates how far the kernel moves after each convolution
- **Padding** is used to help with edge cases
- Pictured: stride of 2, padding of 1



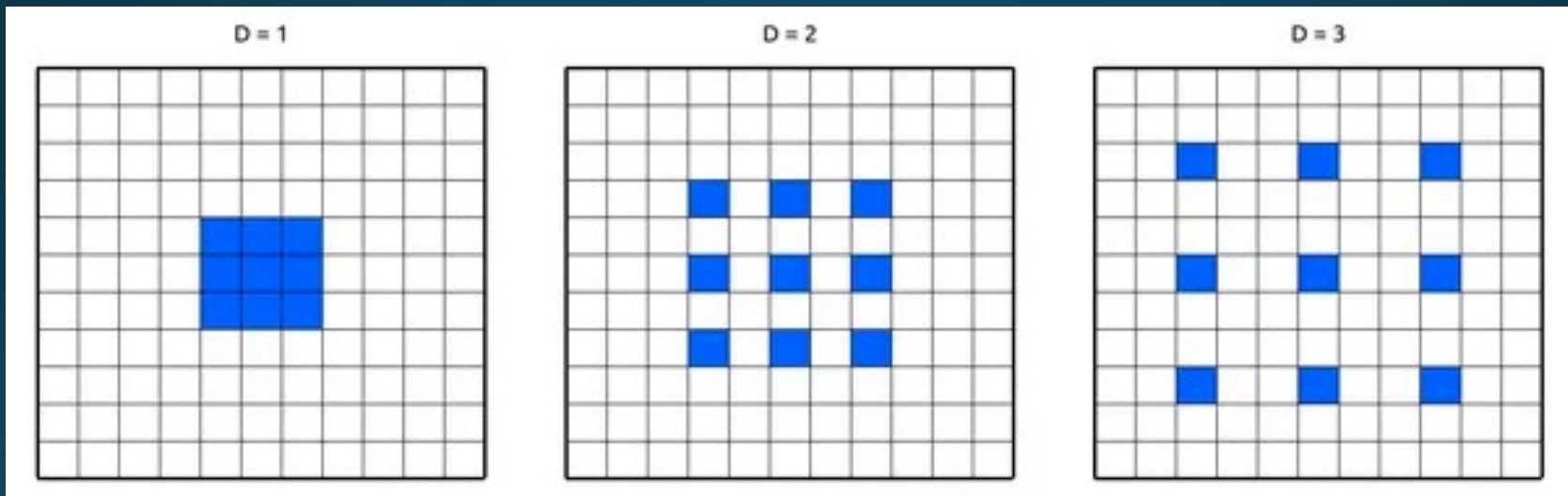
Pooling

- Repeated convolutions can generate large intermediate feature maps
- “Pooling” is used to reduce dimensionality of feature maps while maintaining most informative features
- Mean-pooling, **max-pooling**
- Functions as a regularizer (or an infinitely-strong prior)



Filters

- Different filter topologies

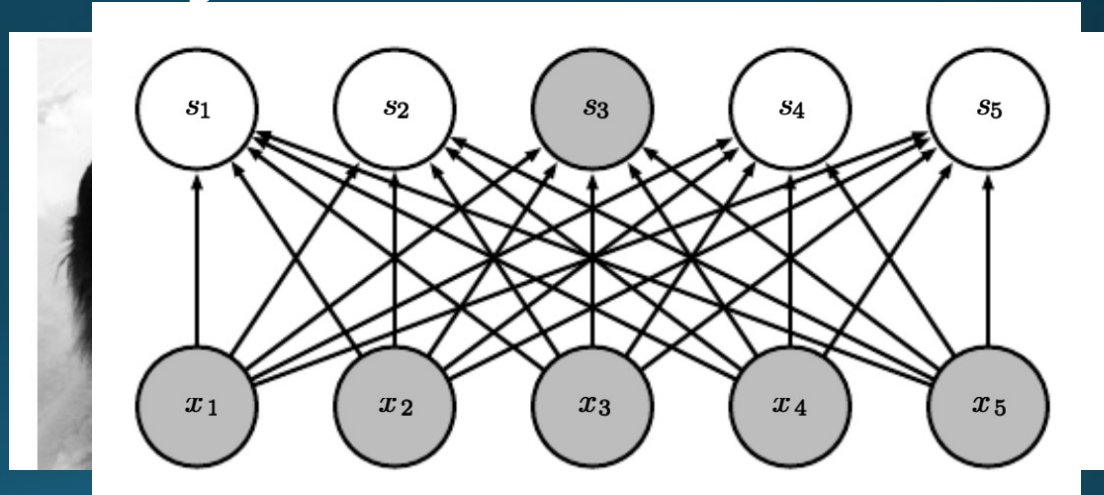


- Captures long-range pixel dependencies
- *Very* computationally expensive to implement

Convolution

- Key point: **parameter sharing**

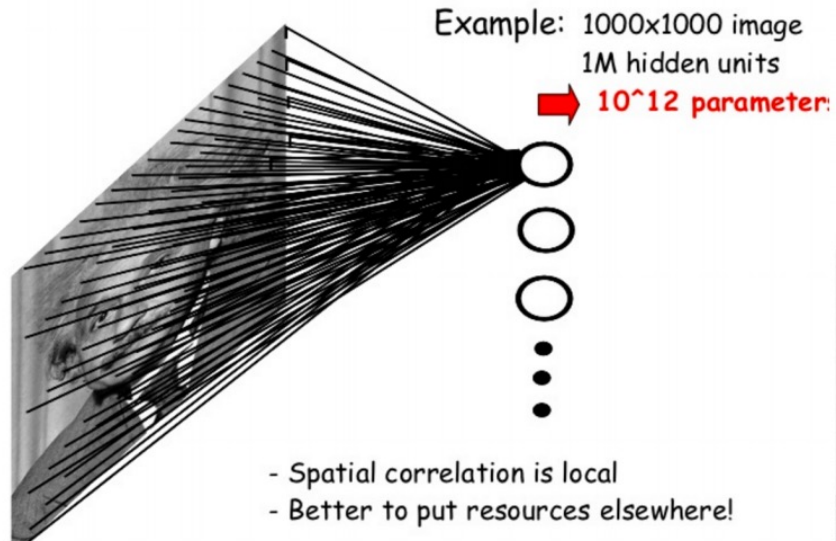
- Images are sparse
 - Pixel dependencies don't span arbitrarily large distances
 - Important effects are local



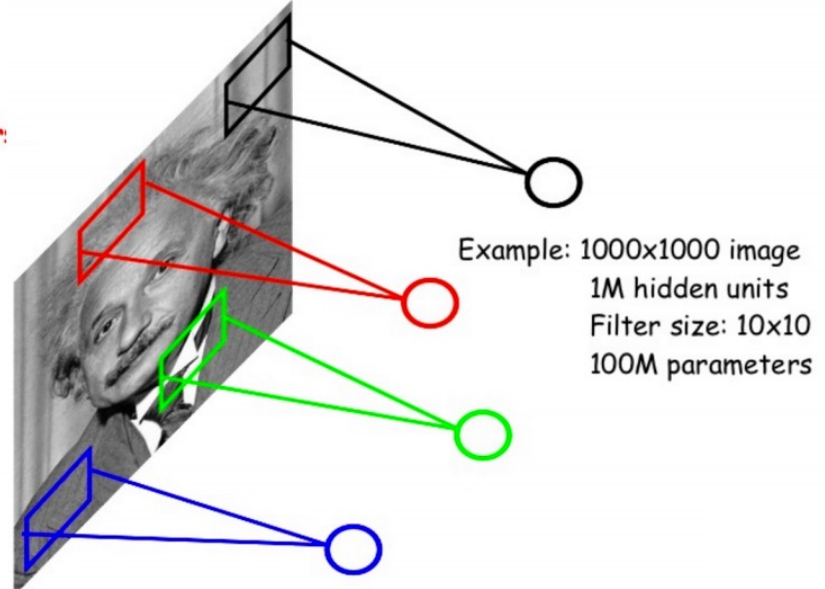
- Instead of a fully-connected network...
- ...we have one that is more sparsely-connected

Parameter Sharing

FULLY CONNECTED NEURAL NET

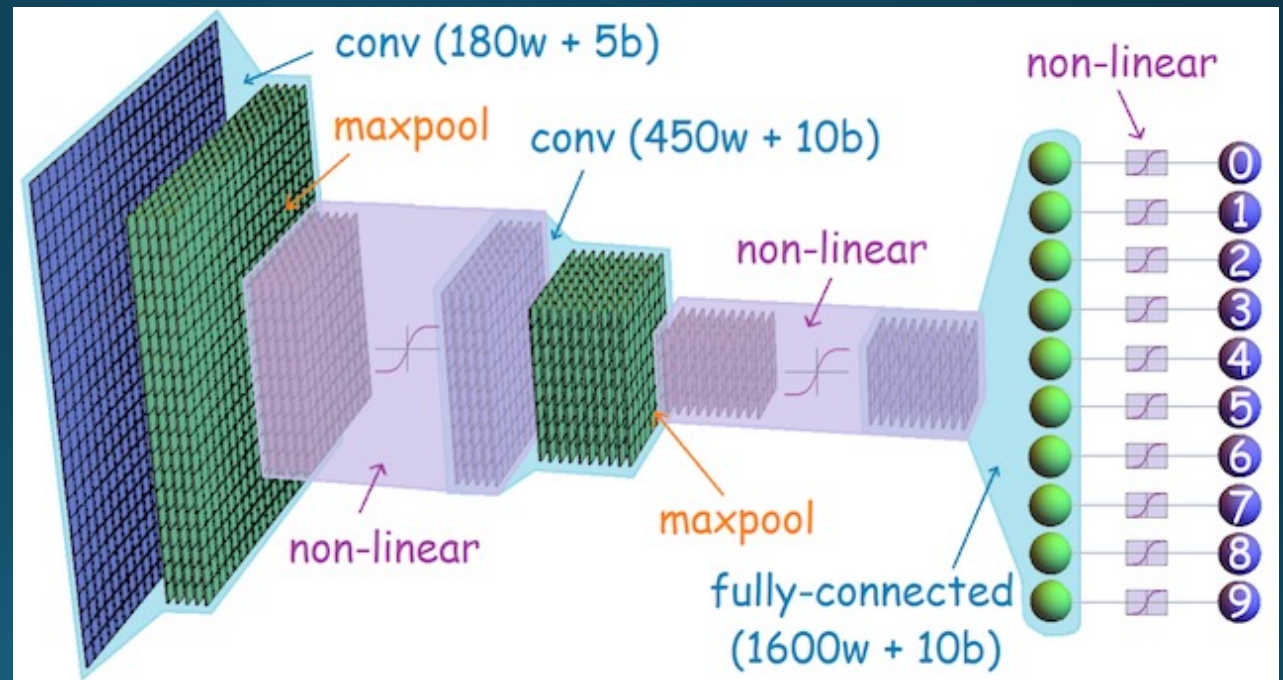


LOCALLY CONNECTED NEURAL NET



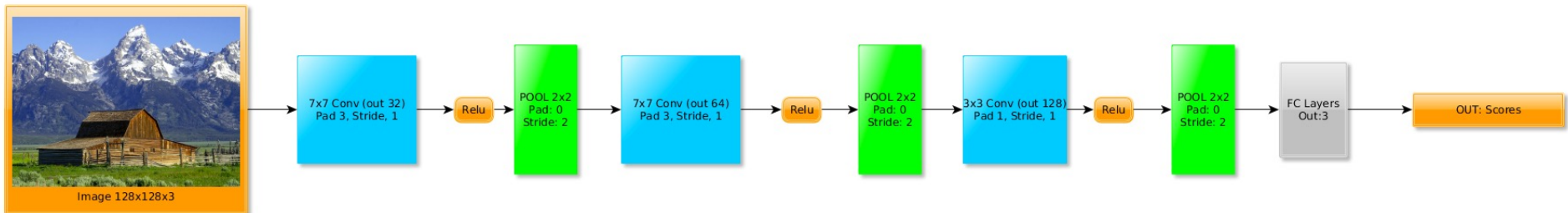
CNNs in Practice

- Stacked
 - Convolutions
 - Pools
 - Activations
- Fully-connected classification layer



CNNs in Practice

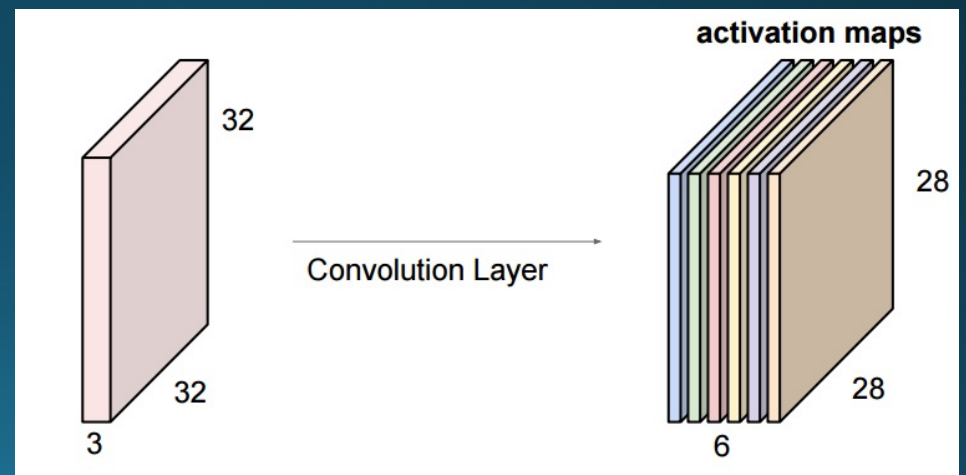
- Pattern can be repeated several times



- Still “deep”, but convolutions are **the most important part**

CNNs in Practice

- Filters are the things that “search” for something in particular in an image
- To search for many different things, have many different filters

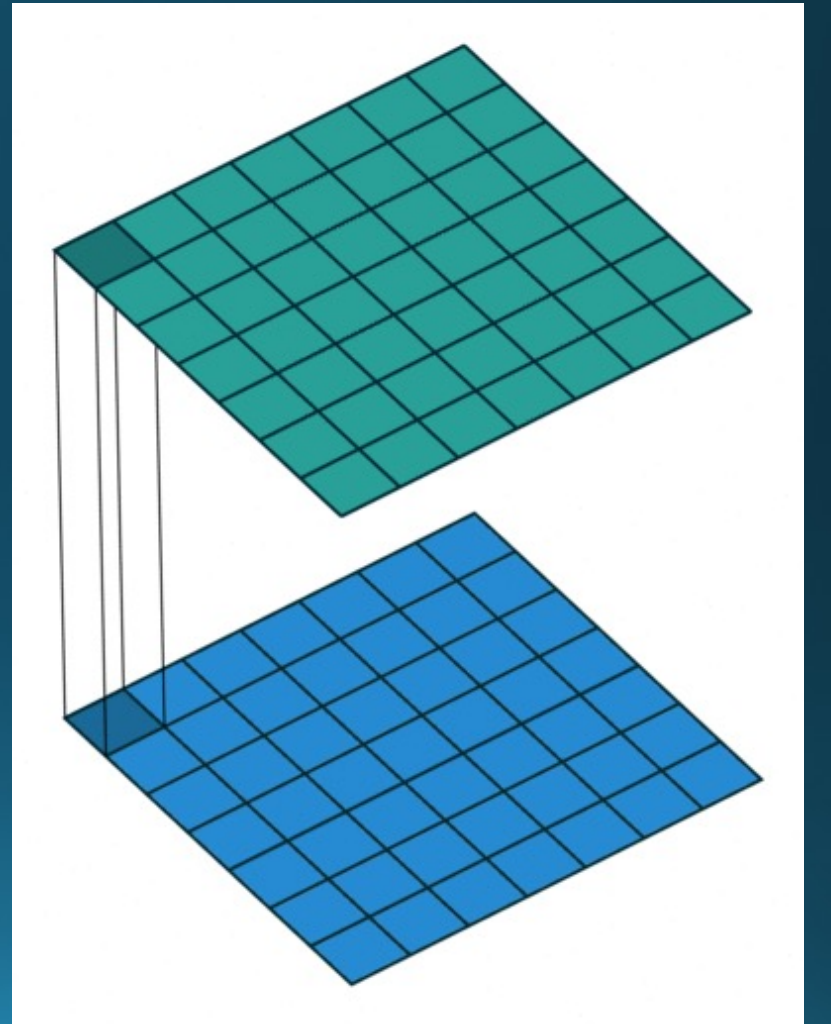


CNNs in Practice

- Hyperparameters relevant to CNNs:
 - Kernel size
 - Usually small
 - Stride
 - Usually 1 (larger for pooling layers)
 - Zero padding depth
 - Enough to permit convolutional output size to be the same as input size
 - Number of convolutional filters
 - Number of “patterns” for the network to search for

CNNs in Practice

- 1x1 convolutions are a special case
- Convolve the **feature maps**, rather than the **pixel maps**
- Function as a dimensionality reduction step (like pooling)
 - Can also be used in pooling



CNN Applications: Object Localization

- Two discrete steps:
 - Localizing a bounding box (*regression*)
 - Identifying the object (*classification*)
- Generate “region proposals”
- Classification accuracy



Classification head”

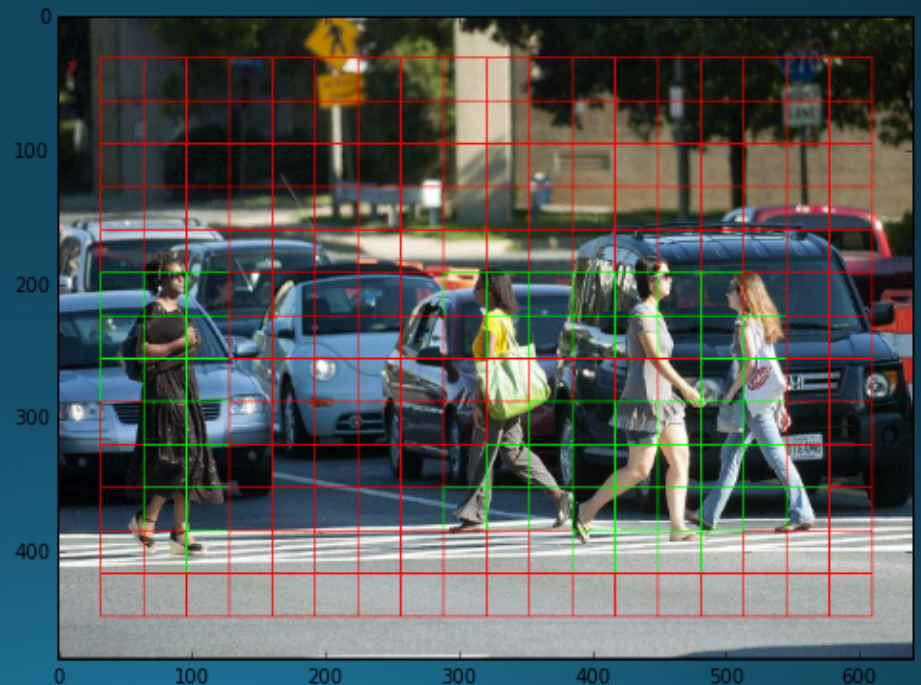
The best result now is Faster RCNN with a resnet 101 layer.

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	0.2 seconds
(Speedup)	1x	25x	250x
mAP (VOC 2007)	66.0	66.9	66.9



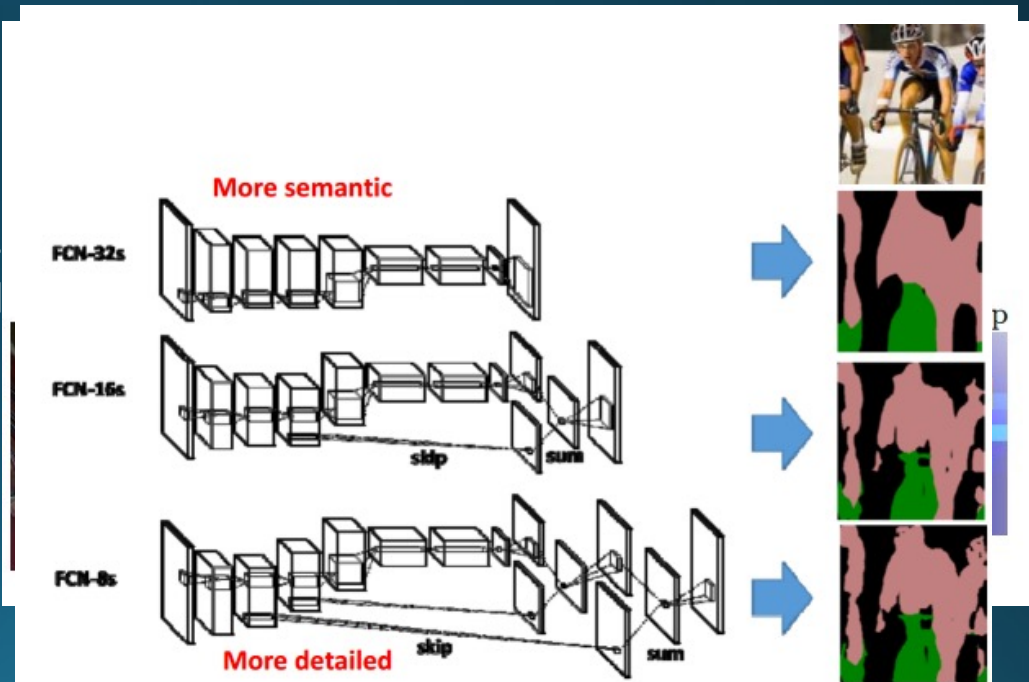
CNN Applications: Single-shot Detection

- Combines region-proposal (regression) and object detection (classification) into a single step
- Use deep-level feature maps to predict class scores and bounding boxes
- Families of Single-shot detectors:
 - YOLO (single activation map for both class and region)
 - SSD (different activations)
 - R-FCN (like Faster R-CNN)



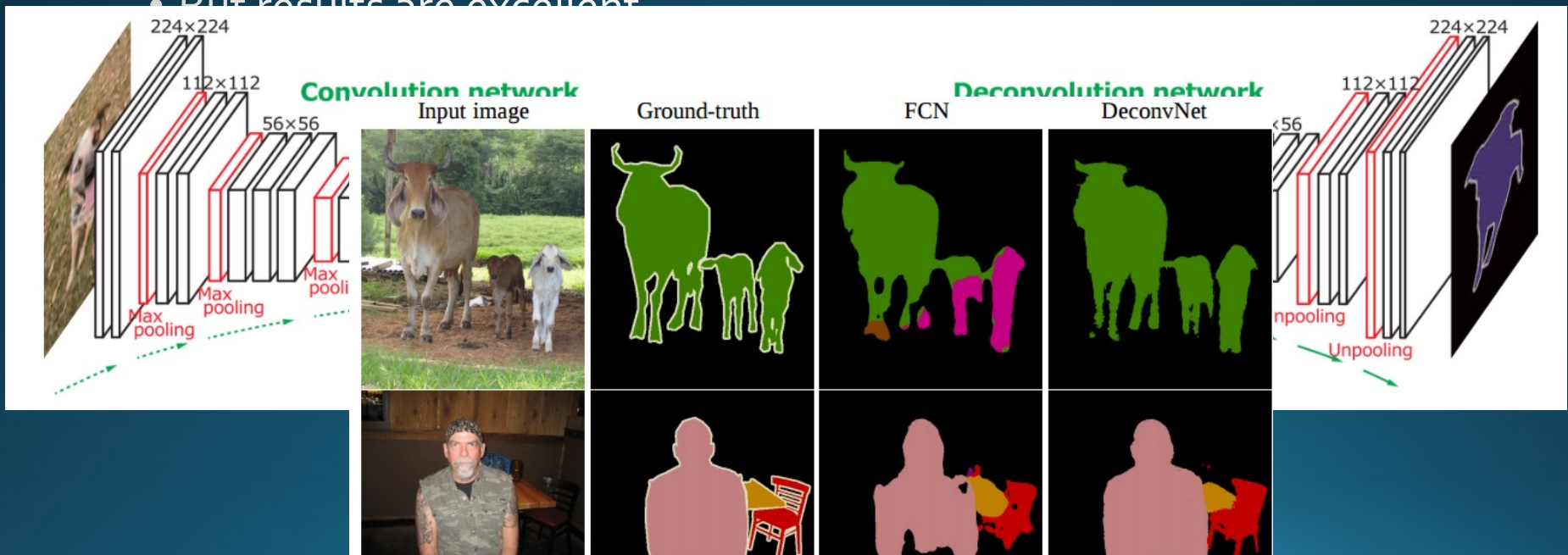
CNN Applications: Object Segmentation

- Create a map of the detected object areas
- “Fully-convolutional” networks
 - Substitute fully-connected layer at end for another convolutional layer
 - Activations show object
- Resolution is lost in upsampling step
 - Skip-connections to bring in some of the “lost” resolution
- *EXTREME* Segmentation
 - Replace upsampling with a complete deconvolution stack



CNN Applications: Object Segmentation

- "DeconvNet": *Super-expensive* to train
- But results are excellent



Conclusions

- CNNs are mostly “convolutions inside a deep network”
 - Main operator (i.e. **most important**) is the convolution
 - Exploits image sparsity: important features are **local**
- A couple new[ish] tricks include
 - Automatically learning the filters as part of the training process
 - Using pooling
 - 1x1 convolutions
- Applications include
 - Object detection (is there an object)
 - Object localization and segmentation (where is the object)
 - Object classification (what is the object)
 - Zero- and single-shot detectors

References

- The Neural Network Zoo
 - <http://www.asimovinstitute.org/neural-network-zoo/>
- Deep Learning Book, Chapter 9: “Convolutional Networks”
 - <http://www.deeplearningbook.org/contents/convnets.html>
- Convolution Arithmetic code (for generating awesome gifs)
 - https://github.com/vdumoulin/conv_arithmetic
- 1x1 Convolutions
 - <https://iamaaditya.github.io/2016/03/one-by-one-convolution/>
- AI Gitbook
 - <https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/>