Convolutional Neural Networks

Yuzhen Ye

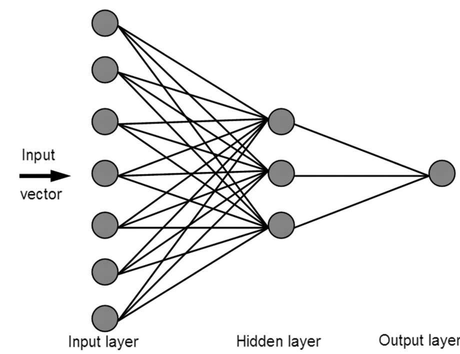
School of Informatics and Computing, Indiana University

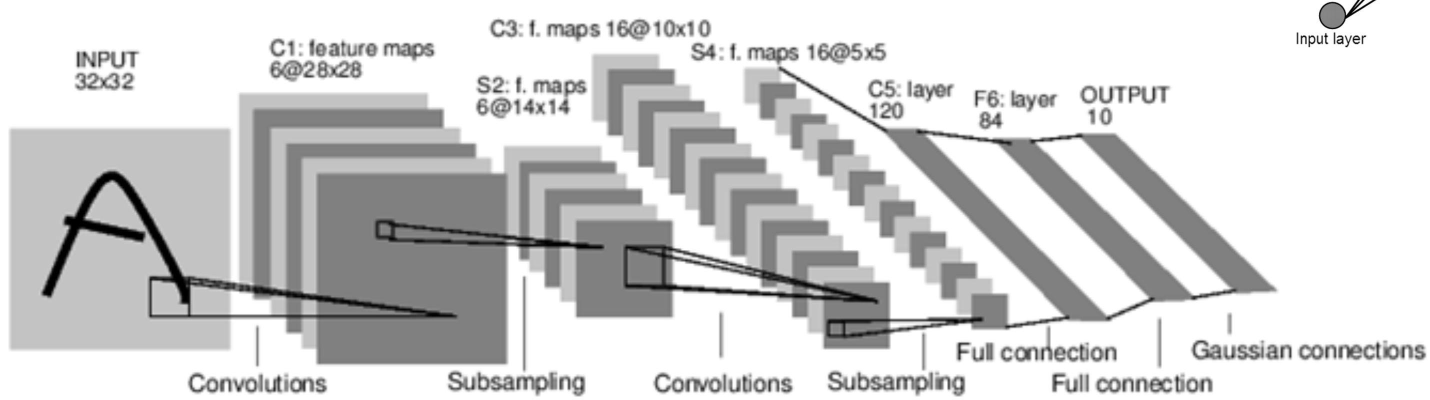
Contents

- CNN basics
- CNN for visual recognition (to explain the concept of convolution)
- CNN for bioinformatics

A Beginner's Guide To Understanding CNN

- Neural Networks with Convolution layers

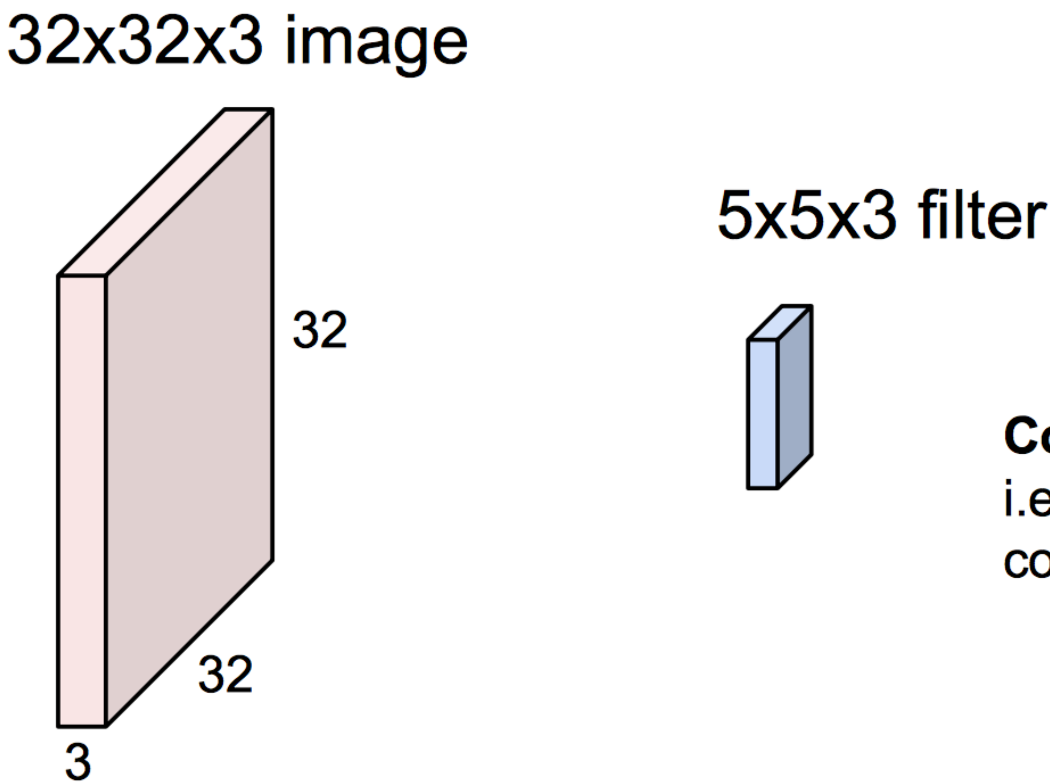




A Full Convolutional Neural Network (LeNet)

Ref 1: http://cs231n.stanford.edu
Ref 2: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Convolution Layer



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

Moving Average In 2D

$$f[x,y]$$

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

$$g[x,y]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Slide credit: Steve Seitz

Correlation filtering

Say the averaging window size is 2k+1 x 2k+1:

$$g(i,j) = \underbrace{\frac{1}{(2k+1)^2}}_{\textit{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} f(i+u, j+v)}_{\textit{Loop over all pixels in neighborhood around image pixel } f[i,j]}$$

Now generalize to allow different weights depending on neighboring pixel's relative position:

$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{h(u,v)}_{\textit{Non-uniform weights}} f(i+u, j+v)$$

Slide adapted from Kristen Grauman

Correlation filtering

$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h(u,v) f(i+u, j+v)$$

This is called cross-correlation, denoted $g = h \otimes f$

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter "kernel" or "mask" *h*[*u,v*] is the prescription for the weights in the linear combination.

Slide credit: Michael S. Ryoo

Averaging filter

- What values belong in the kernel *h* for the moving average example?

$f[x, y]$

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

$\otimes$

$h[u, v]$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & ? & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

"box filter"

$g[x, y]$

	0	10	20	30	30				

$$g = h \otimes f$$

Slide credit: Michael S. Ryoo

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

gh-frequency components from the image ("low-pas

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

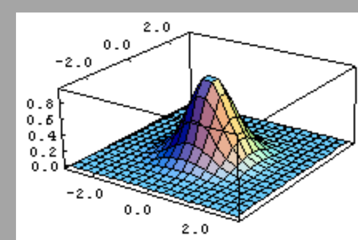
$f[x,y]$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$h[u,v]$

This kernel is an approximation of a 2d Gaussian function:

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



Slide credit: Steve Seitz

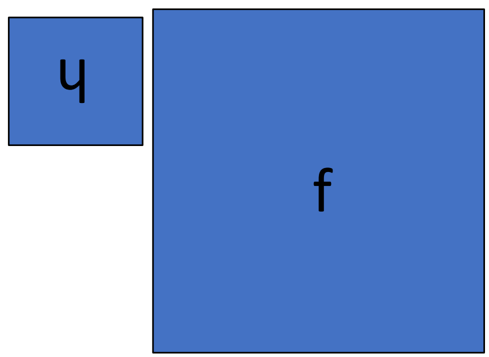
Convolution

- Convolution is a simple mathematical operation which is fundamental to many common image processing operators.
- Convolution is performed by sliding the kernel over the image, generally starting at the top left corner, so as to move the kernel through all the positions where the kernel fits entirely within the boundaries of the image.
- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h(u,v) f(i-u, j-v)$$

$$g = h * f$$

Notation for convolution operator



Slide credit: Michael S. Ryoo

Convolution vs. correlation

Convolution

$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h(u,v) f(i-u, j-v)$$

$$g = h * f$$

Cross-correlation

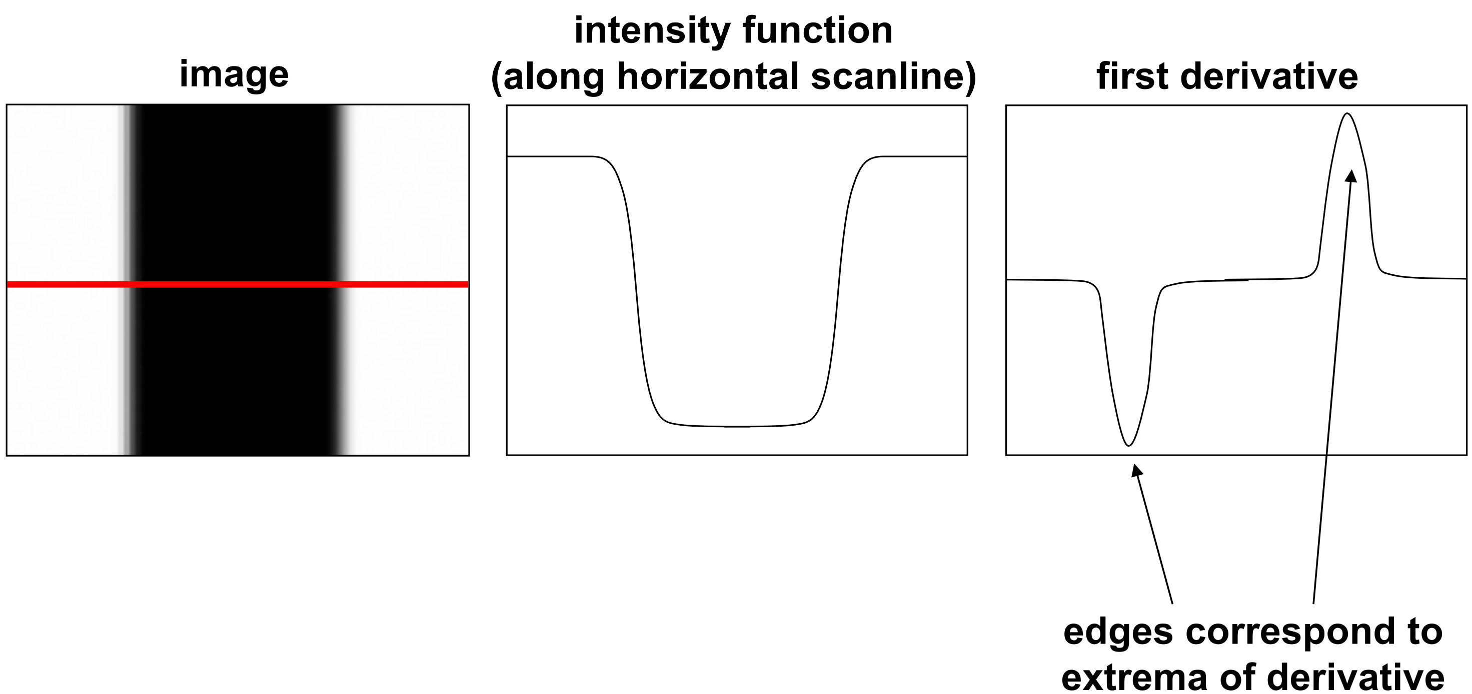
$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h(u,v) f(i+u, j+v)$$

$$g = h \otimes f$$

Slide adapted from Kristen Grauman

Derivatives and edges

An edge is a place of rapid change in the image intensity function.



Slide credit: Svetlana Lazebnik

Derivatives with convolution

For 2D function, f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

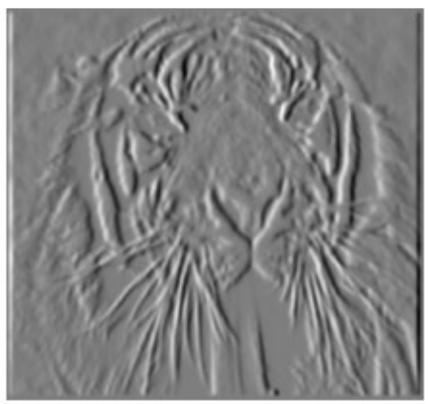
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

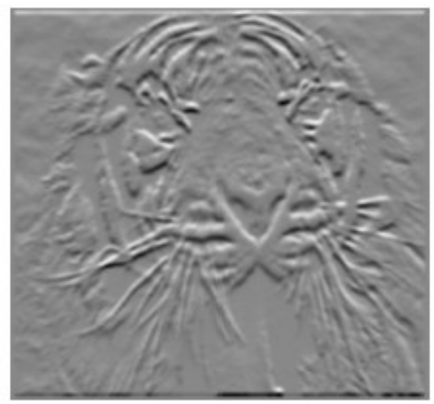
To implement above as convolution, what would be the associated filter?

Slide credit: Kristen Grauman

Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$





$$\frac{\partial f(x,y)}{\partial y}$$

-1	1

-1
1

?

or

1
-1

Which shows changes with respect to x?

(showing filters for correlation)

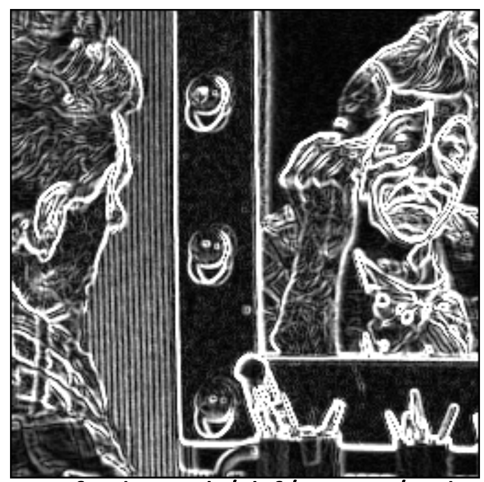
Filters as feature (edge) detectors

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \; ; \; M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \; ; \; M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \; ; \; M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

`sobel filter` →



Slide credit: Kristen Grauman

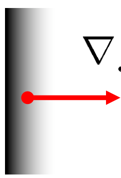
http://homepages.inf.ed.ac.uk/rbf/HIPR2/index.htm

Image gradient

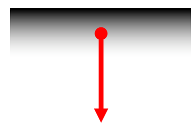
The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

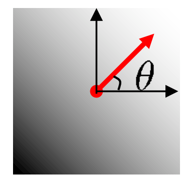
The gradient points in the direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The **gradient direction** (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

The **edge strength** is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

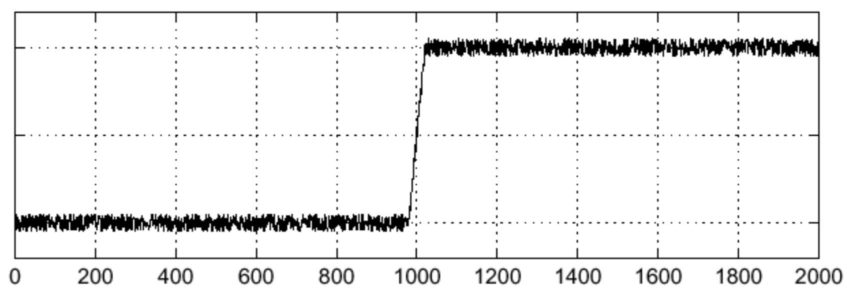
Slide credit: Steve Seitz

Effects of noise

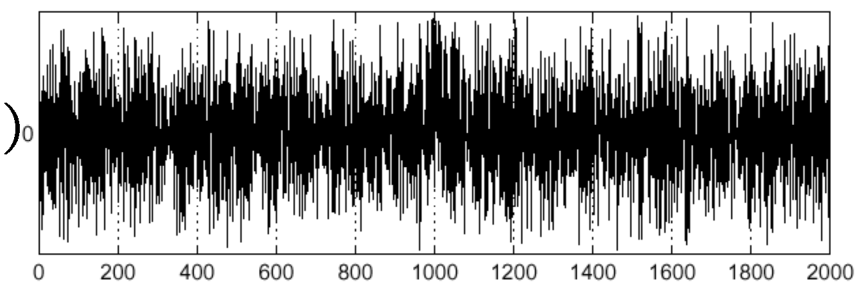
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$



$\frac{d}{dx}f(x)$



Where is the edge?

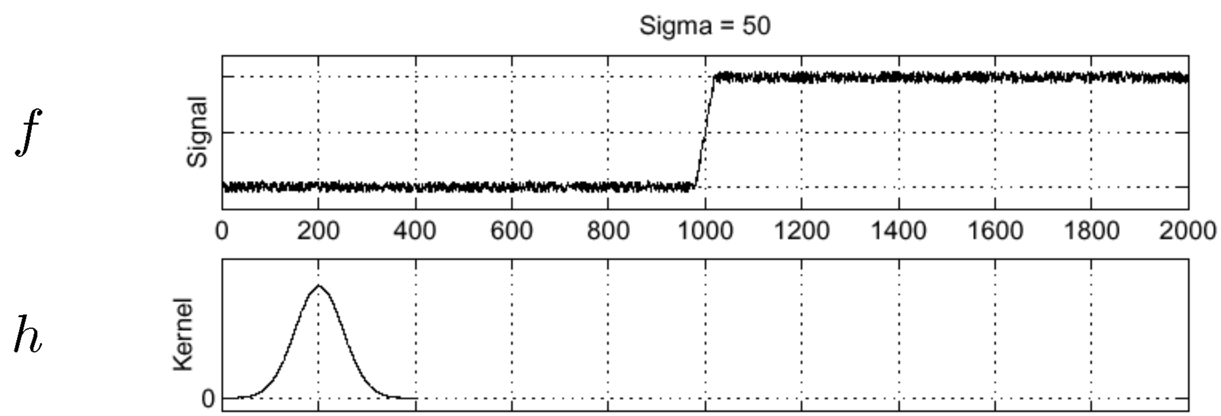
Slide credit: Steve Seitz

Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Slide credit: Michael S. Ryoo

Solution: smooth first



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

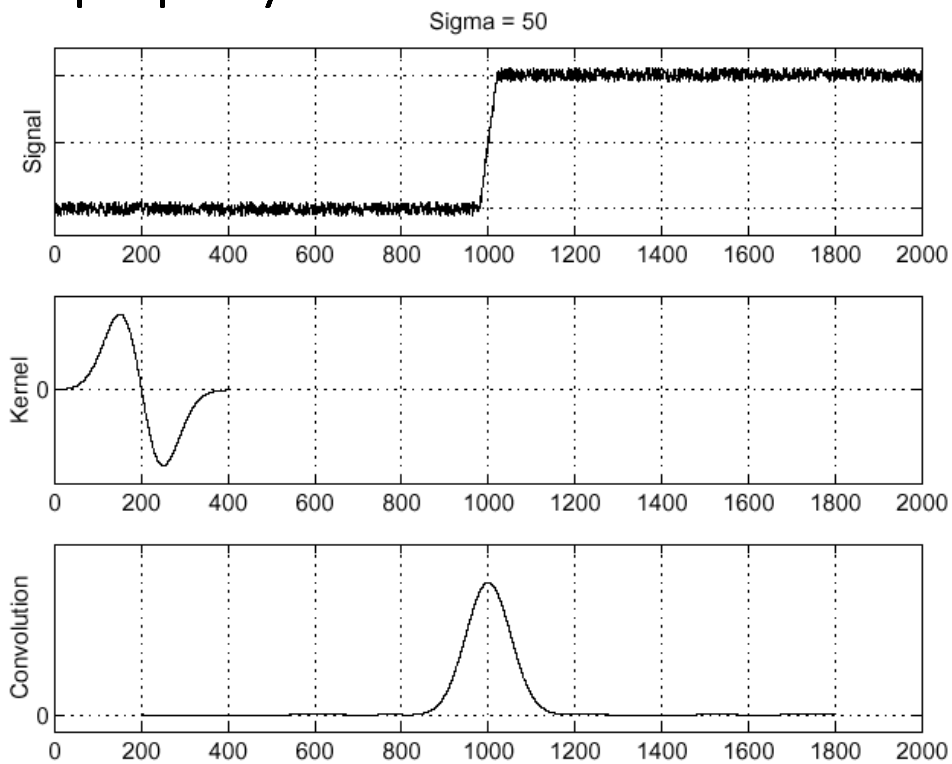
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



Slide credit: Steve Seitz

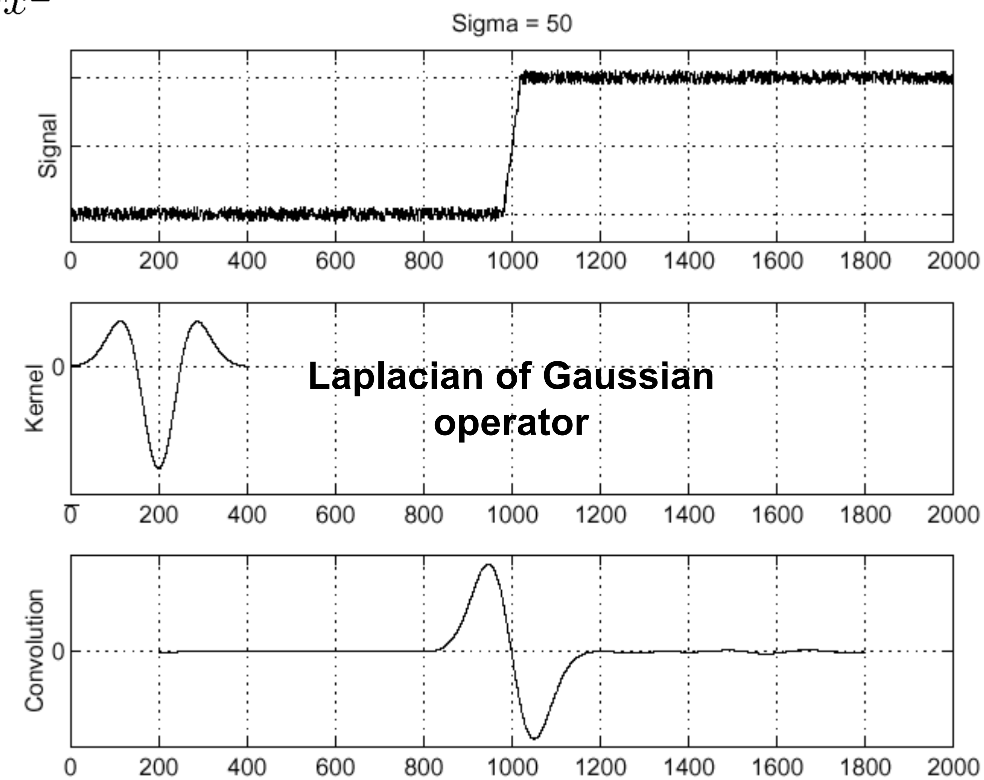
Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2}h$

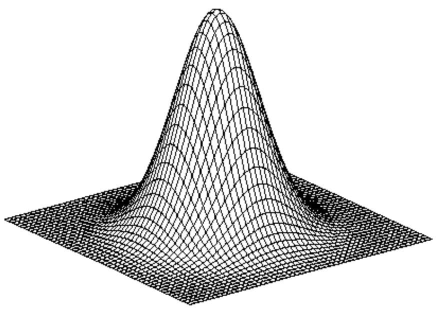
$(\frac{\partial^2}{\partial x^2}h) \star f$



Where is the edge? Zero-crossings of bottom graph

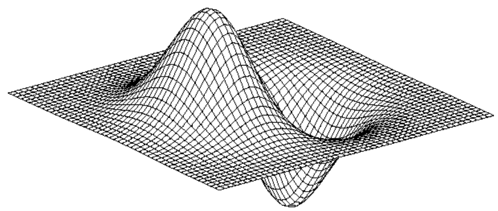
Slide credit: Steve Seitz

2D edge detection filters



Gaussian

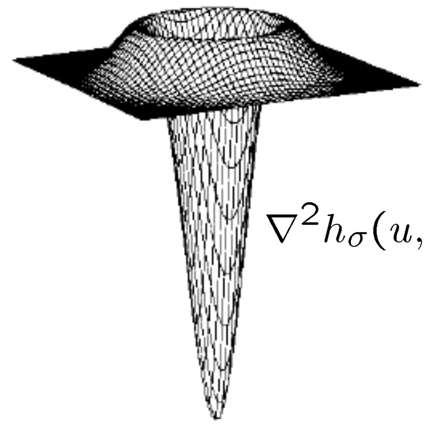
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Two commonly used discrete approximations to the Laplacian filter

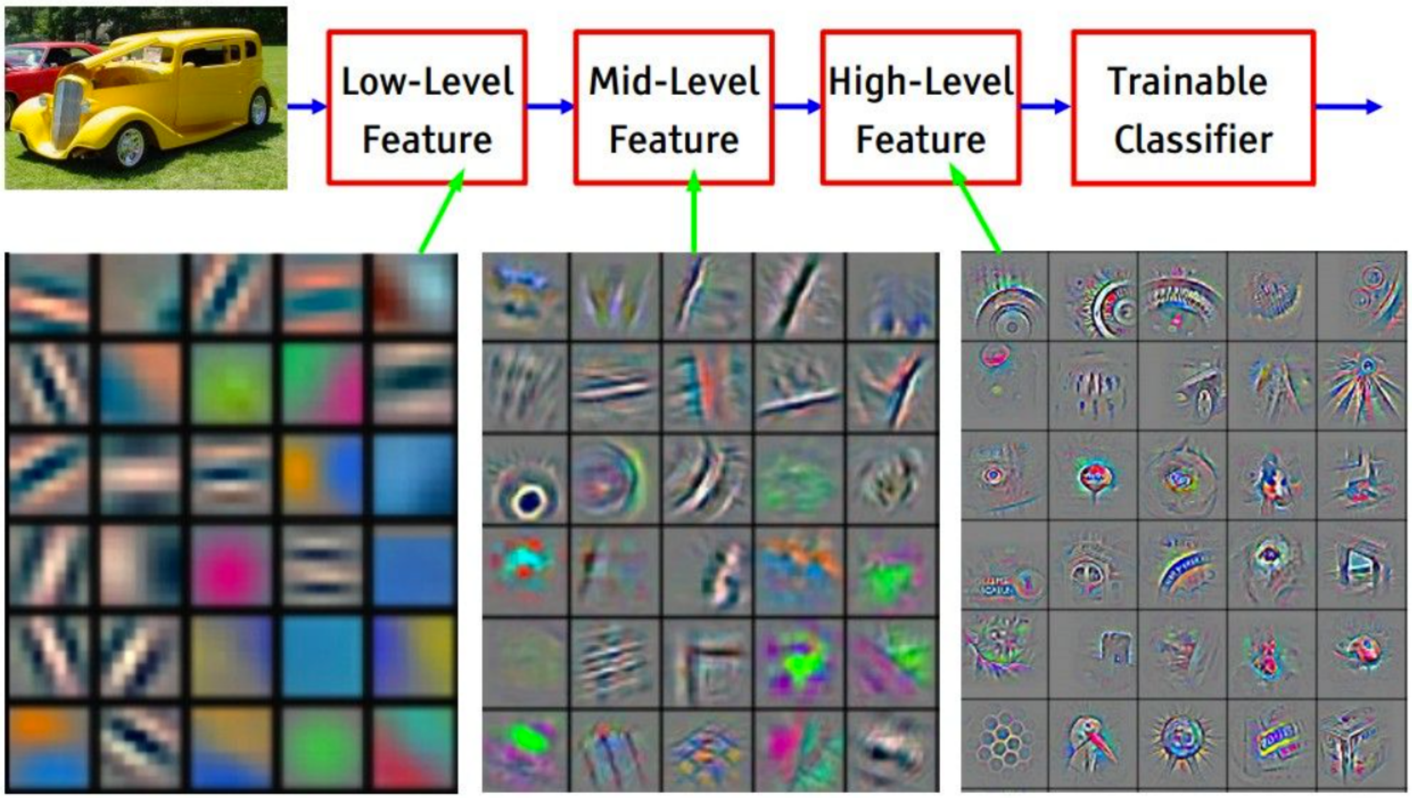
- ∇^2 is the Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Slide credit: Steve Seitz

Preview

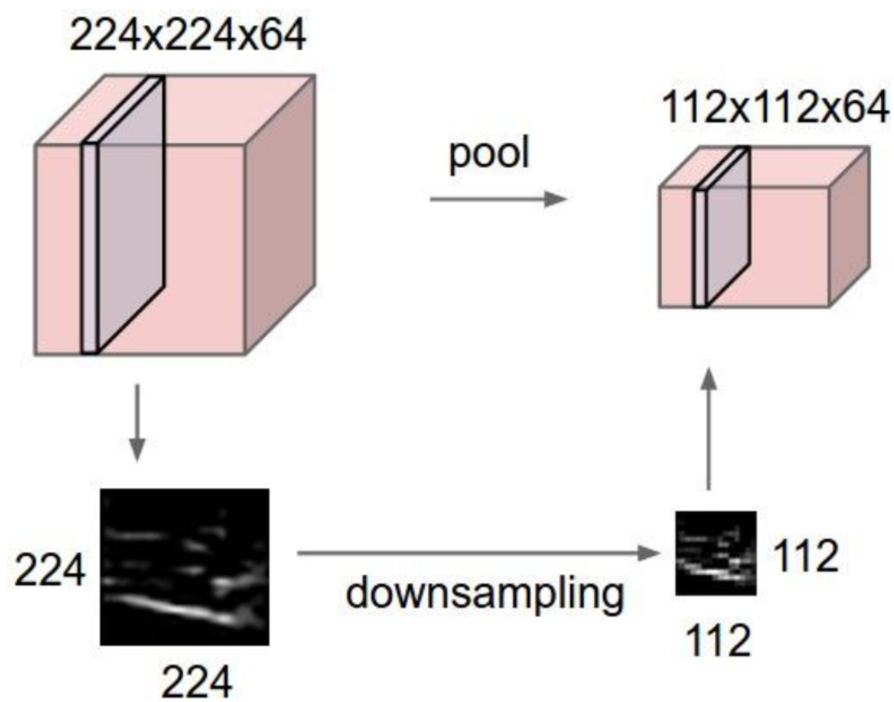
[From recent Yann LeCun slides]



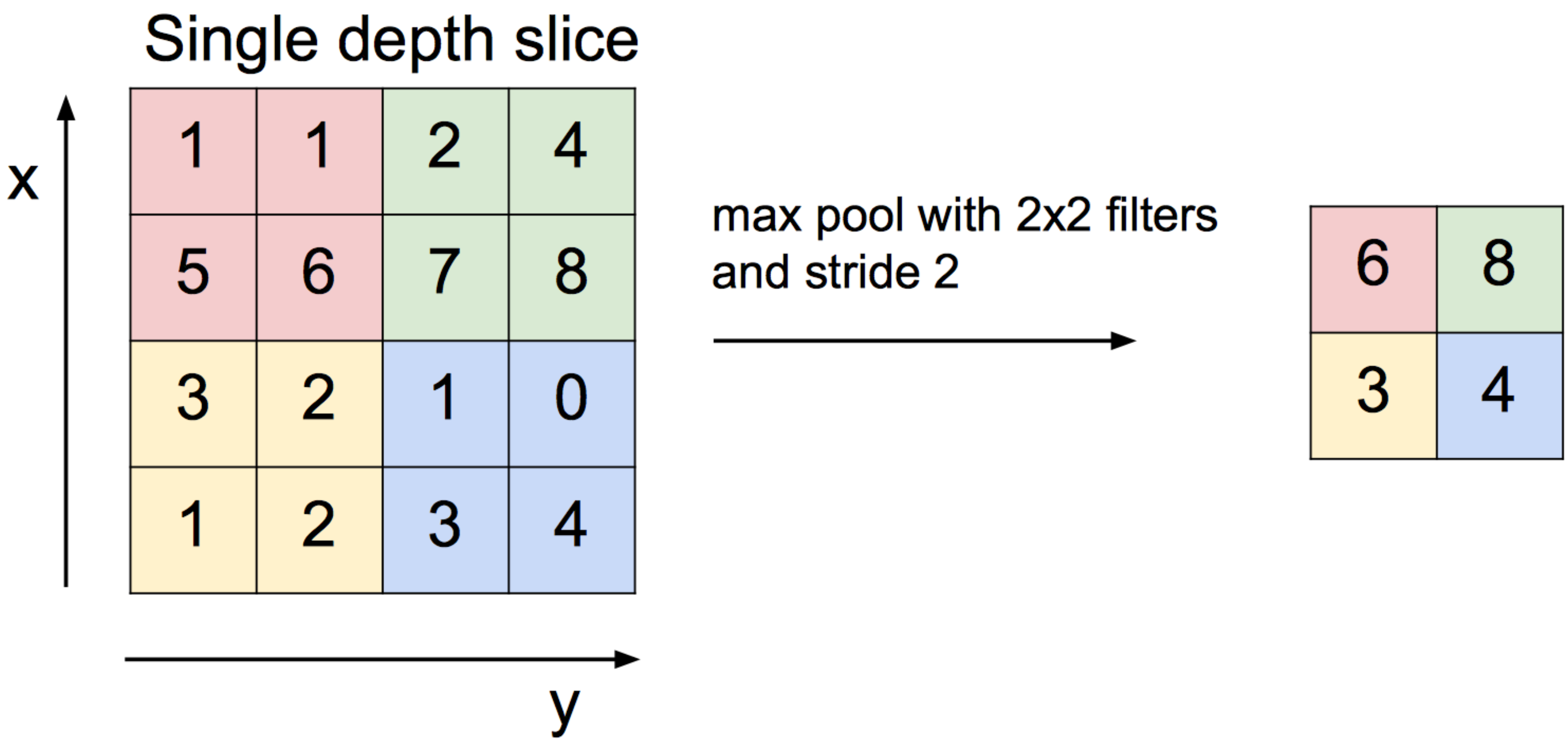
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Pooling layer

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

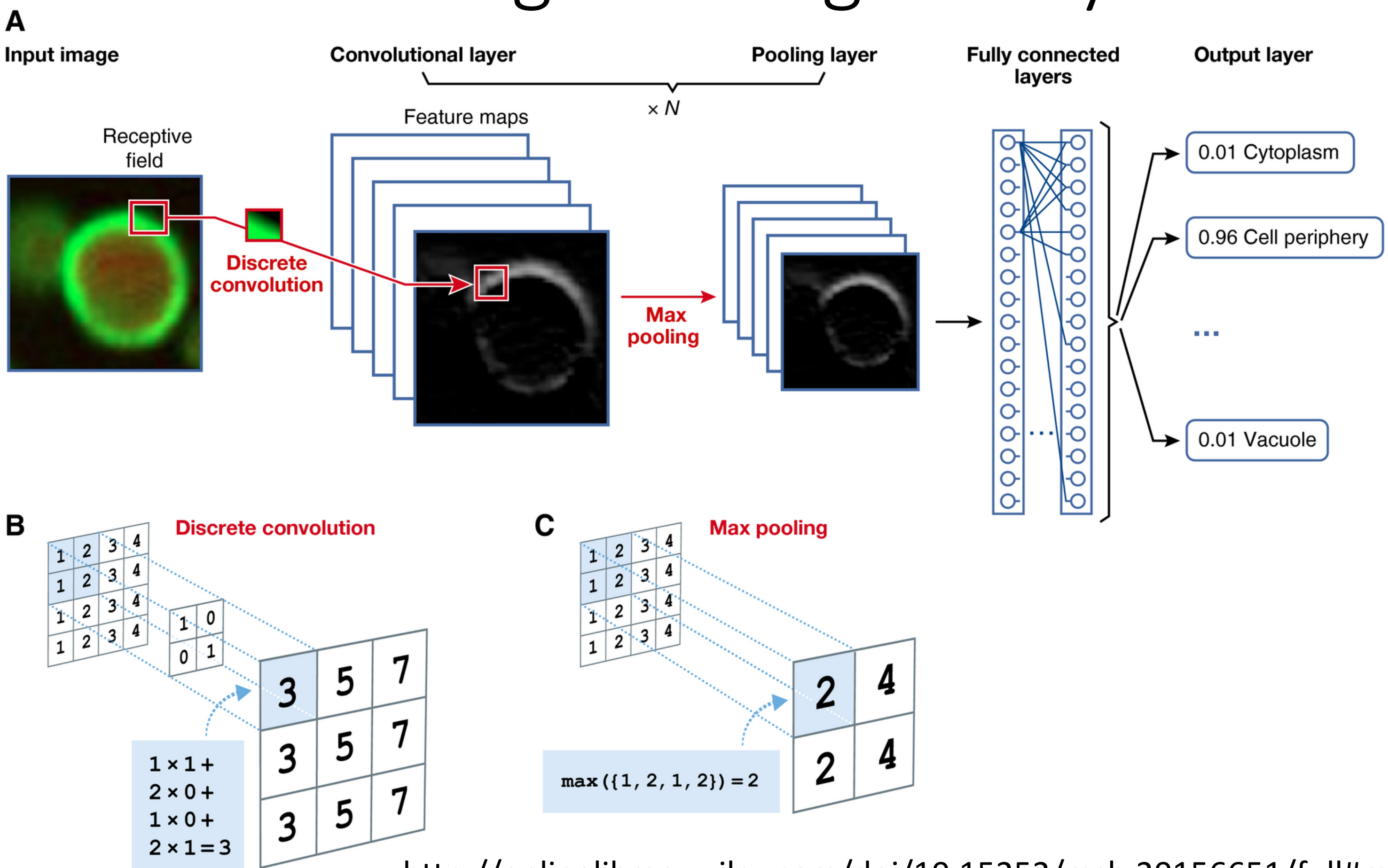


MAX POOLING



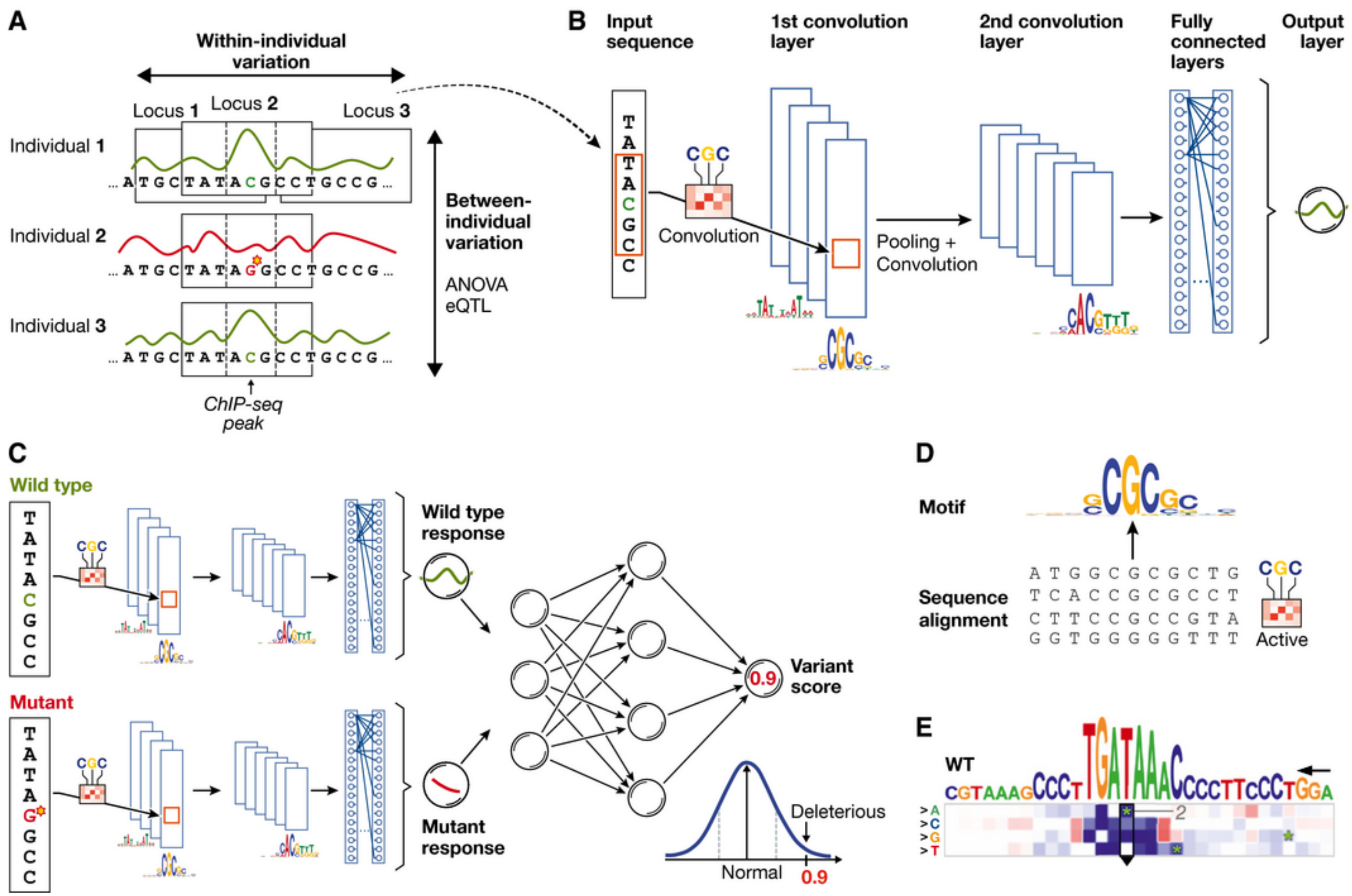
Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

CNN for biological image analysis



http://onlinelibrary.wiley.com/doi/10.15252/msb.20156651/full#msb156651-fig-0002

CNN for predicting molecular traits



Molecular Systems Biology
Volume 12, Issue 7, 29 JUL 2016 DOI: 10.15252/msb.20156651
http://onlinelibrary.wiley.com/doi/10.15252/msb.20156651/full#msb156651-fig-0002

Input data: one-dimensional genomic sequences with one channel per nucleotide

[Visual recognition: 2D-image with three color channels]