CPSC 340: Machine Learning and Data Mining

Convolutional Neural Networks

Fall 2018

Admin

- Mike and I finish CNNs on Wednesday.

- After that, we will cover different topics:
  - Mike will do a demo of training CNNs with cloud/GPU resources.
  - I am planning to cover boosting (the other type of ensemble method).
    - The lecture will probably be 90 minutes (I won't be offended if you leave early, extra time won't be testable).

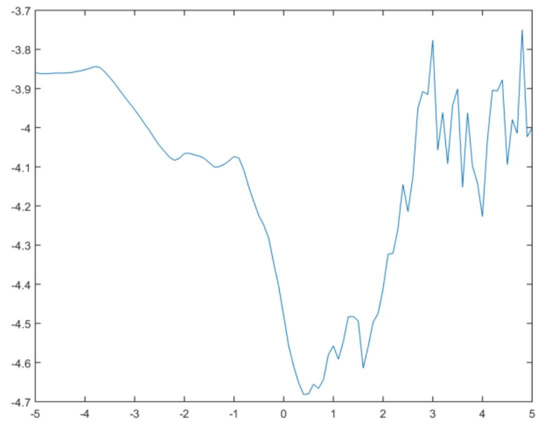
- Friday's lectures will also be different:
  - Mike will do a course review in his section.
  - Aline Tabet will give a guest lecture in this section ("ML Applications in Medicine").

- Final: Thursday December 13th at 8:30am in WOOD 2.
  - Similar style of questions to midterm.
  - 2 pages of notes.

- CPSC 532M students: course project due December 19 (details on Piazza).

Last Time: Convolutions

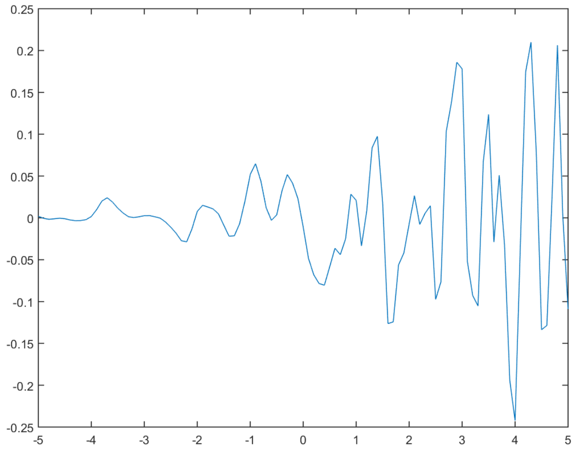
- Consider our original "signal":



- For each "time":
  - Compute dot-product of signal at surrounding times with a "filter".

$w = [-0.1416 \;\; -0.1781 \;\; -0.2746 \;\; 0.1640 \;\; 0.8607 \;\; 0.1640 \;\; -0.2746 \;\; -0.1781 \;\; -0.1416]$

- This gives a new "signal":
  - Measures a property of "neighbourhood".
  - This particular filter shows a local "how spiky" value.



1D Convolution

- 1D convolution example:
  - Signal:

$$x = [0 \;\; \boxed{1 \;\; 1 \;\; 2 \;\; 3 \;\; 5} \;\; 8 \;\; 13]$$

$$\underbrace{1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 8}_{n}$$

  - Filter:

$$w = [0 \;\; -1 \;\; 2 \;\; -1 \;\; 0]$$

$$w_{-2} \;\; w_{-1} \;\; w_0 \;\; w_1 \;\; w_2$$

  - Convolution:

$$z = [\;\; 0 \;\;]$$

$$\underbrace{1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 8}_{n}$$

Let's compute z_4:

$$x_{4-2:4+2} = [1 \;\; 1 \;\; 2 \;\; 3 \;\; 5]$$

Multiply element-wise

$$[0 \;\; -1 \;\; 4 \;\; -3 \;\; 0]$$

Add

$$z_4 = 0 - 1 + 4 - 3 + 0 = \underline{0}$$

1D Convolution

- 1D convolution example:
  - Signal:

$$x = \underbrace{\begin{array}{cccccccc} [0 & 1 & 1 & 2 & \boxed{3} & 5 & 8 & 13] \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}}_{n}$$

  - Filter:

$$w = \begin{array}{ccccc} [0 & -1 & 2 & -1 & 0] \\ w_{-2} & w_{-1} & w_0 & w_1 & w_2 \end{array}$$

  - Convolution:

$$z = \underbrace{\begin{array}{cccccccc} [\ & & & 0 & \boxed{-1} & & & \ ] \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}}_{n}$$

Let's compute z_5:

$$[1\ \ 2\ \ 3\ \ 5\ \ 8]$$

Multiply

$$[0\ \ -2\ \ 6\ \ -5\ \ 0]$$

Add

$$z_5 = -2 + 6 - 5 = -1$$

1D Convolution Examples

- Examples:
  - "Identity"

    $\hookrightarrow w = [0 \quad 1 \quad 0]$

  - "Translation"

    $\hookrightarrow w = [0 \quad 0 \quad 1]$

Let $x = [0 \quad 1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13]$

$z = [0 \quad 1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13]$

$0 \cdot x_0 + 1 \cdot x_1 + 0 \cdot x_2$ $\quad$ $0 \cdot x_1 + 1 \cdot x_2 + 0 \cdot x_3$

$z = [1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13 \quad ?]$

$0 \cdot x_0 + 0 \cdot x_1 + 1 \cdot x_2$

1D Convolution Examples

- Examples:
  - "Identity"
  - "Local Average"

Let $x = [0\ 1\ 1\ 2\ 3\ 5\ 8\ 13]$

$\hookrightarrow w = [0\ 1\ 0]$

$z = [\underbrace{0\ 1\ 1}_{\text{average}}\ \underbrace{2\ 3\ 5}_{\text{average}}\ 8\ 13]$

$\hookrightarrow w = [1/3\ 1/3\ 1/3]$

$z = [?\ 2/3\ 1\tfrac{1}{3}\ 2\ 3\tfrac{1}{3}\ 5\tfrac{1}{3}\ 8\tfrac{2}{3}\ ?]$

Boundary Issue

- What can we do about the "?" at the edges?

If $x = [0\ 1\ 1\ 2\ 3\ 5\ 8\ 13]$ and $w = [\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}]$ then $z = [?\ \frac{2}{3}\ 1\frac{1}{3}\ 2\ 3\frac{1}{3}\ 5\frac{1}{3}\ 8\frac{2}{3}\ ?]$

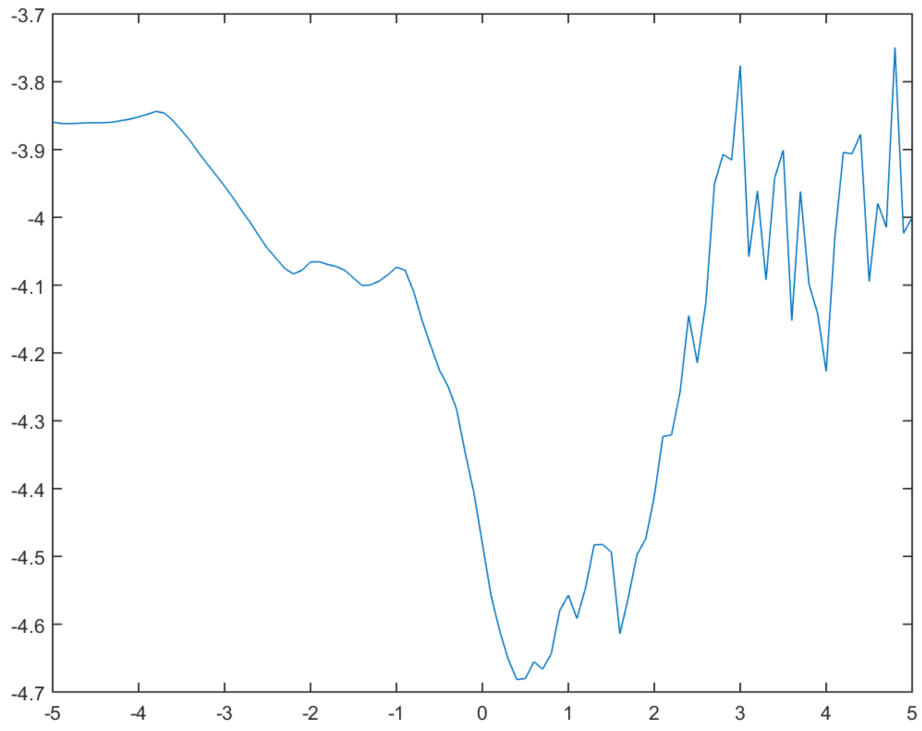
- Can assign values past the boundaries:
  - "Zero": $x = 0\ 0\ 0\ [0\ 1\ 1\ 2\ 3\ 5\ 8\ 13]\ 0\ 0\ 0$
  - "Replicate": $x = 0\ 0\ 0\ [0\ 1\ 1\ 2\ 3\ 5\ 8\ 13]\ 13\ 13\ 13$
  - "Mirror": $x = 2\ 1\ 1\ [0\ 1\ 1\ 2\ 3\ 5\ 8\ 13]\ 8\ 5\ 3$
- Or just ignore the "?" values and return a shorter vector:

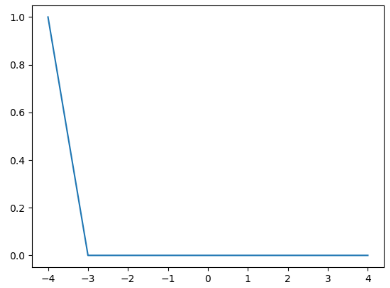
$$z = [\tfrac{2}{3}\ 1\tfrac{1}{3}\ 2\ 3\tfrac{1}{3}\ 5\tfrac{1}{3}\ 8\tfrac{2}{3}]$$

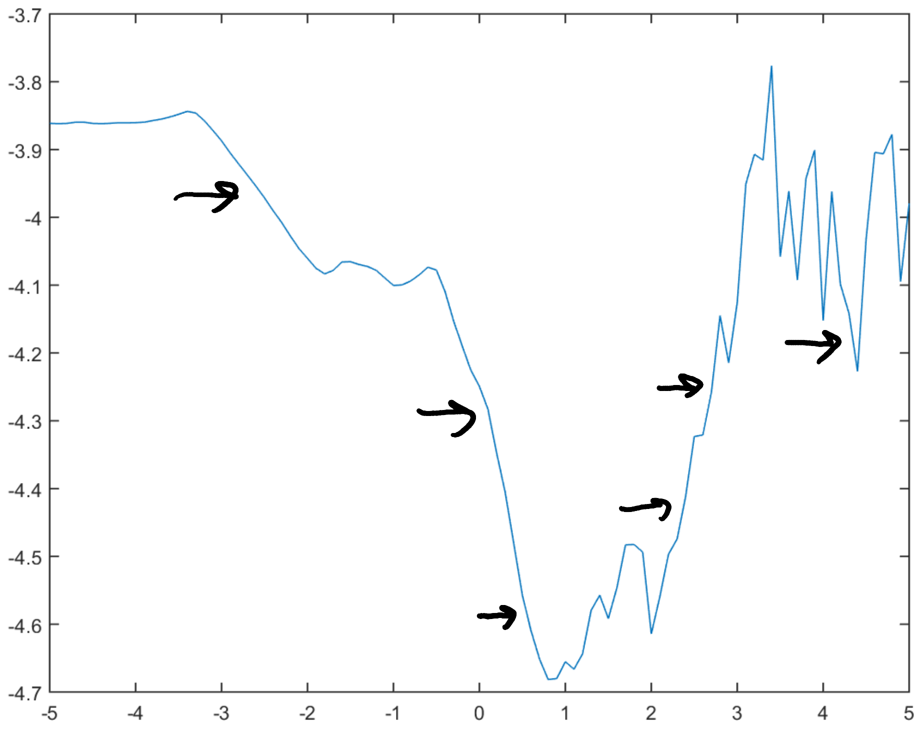
1D Convolution Examples

- Translation convolution shift signal:

$w = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$



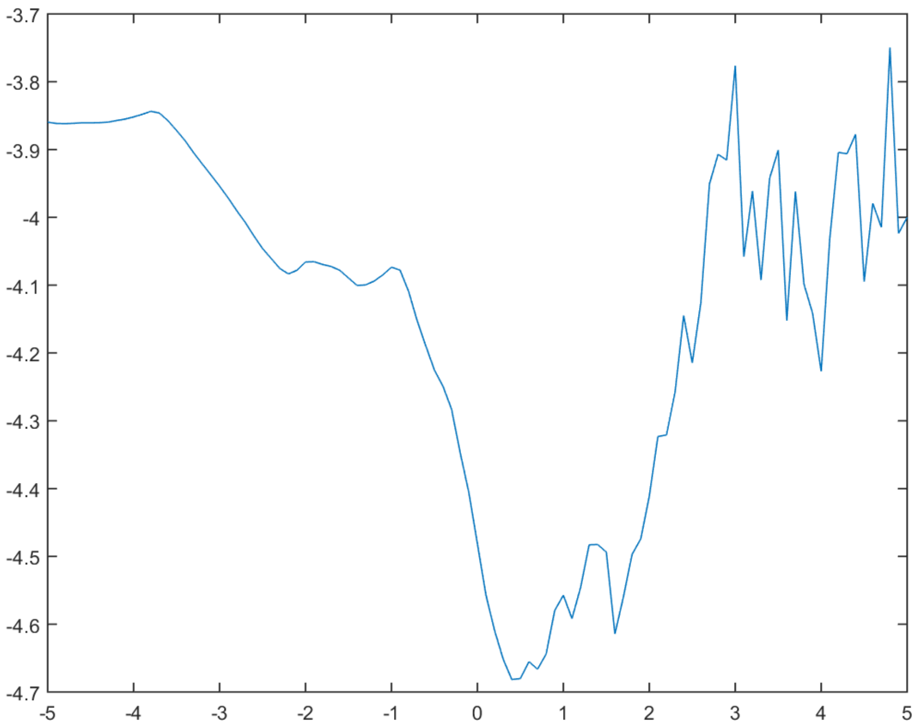


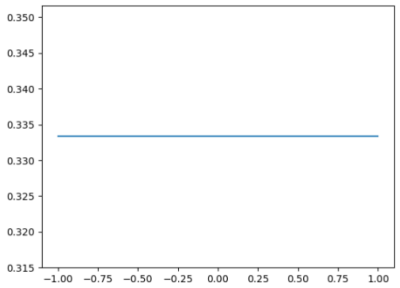


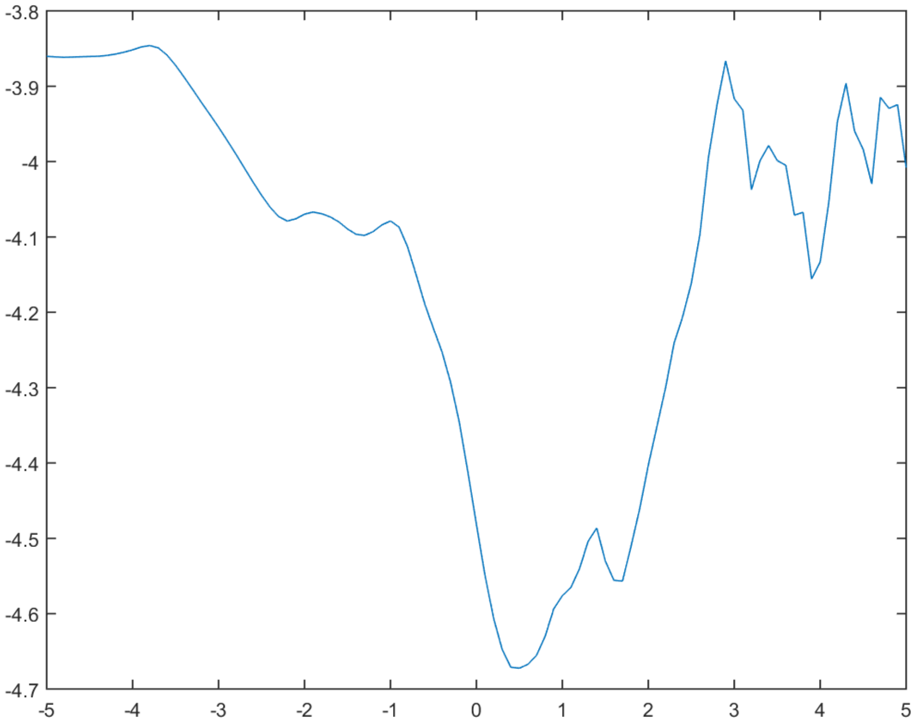
1D Convolution Examples

- Averaging convolution computes local mean:

$$w = \left[\tfrac{1}{3} \quad \tfrac{1}{3} \quad \tfrac{1}{3}\right]$$



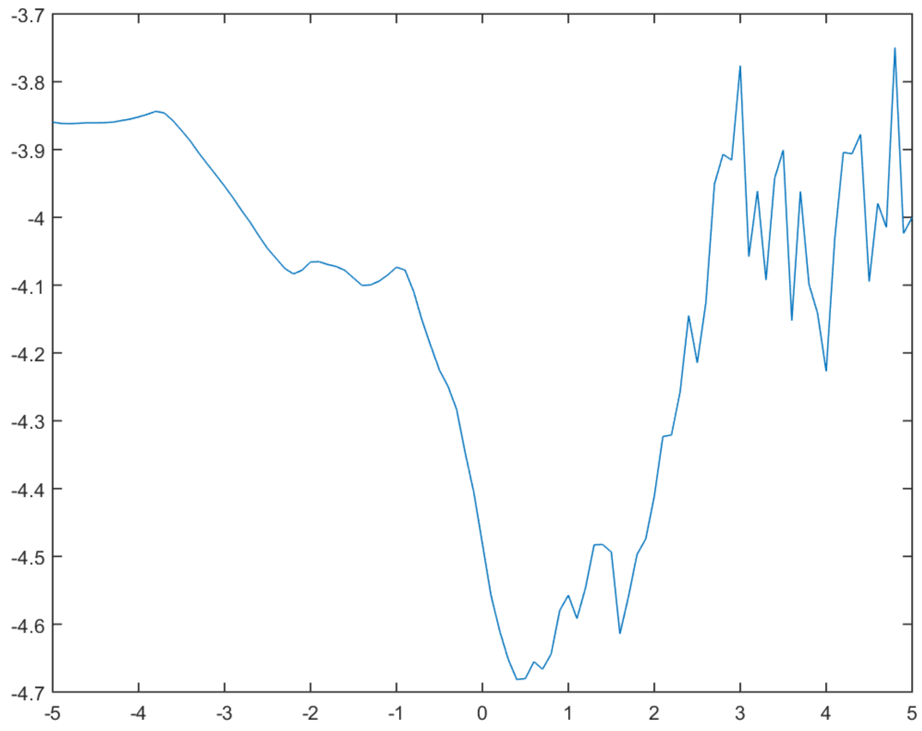


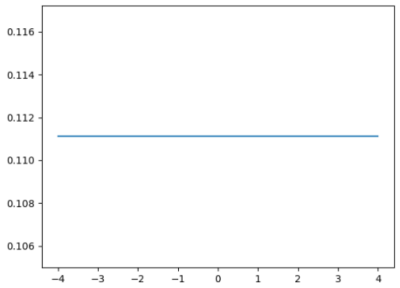


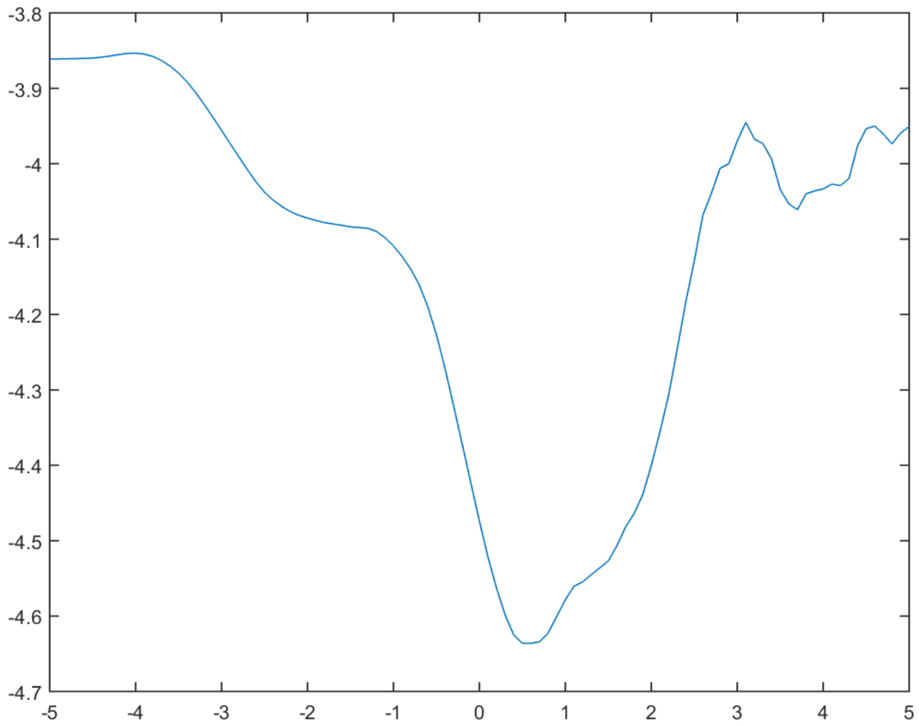
1D Convolution Examples

- Averaging over bigger window gives coarser view of signal:

$$w = [\tfrac{1}{9}\ \tfrac{1}{9}\ \tfrac{1}{9}\ \tfrac{1}{9}\ \tfrac{1}{9}\ \tfrac{1}{9}\ \tfrac{1}{9}\ \tfrac{1}{9}\ \tfrac{1}{9}]$$





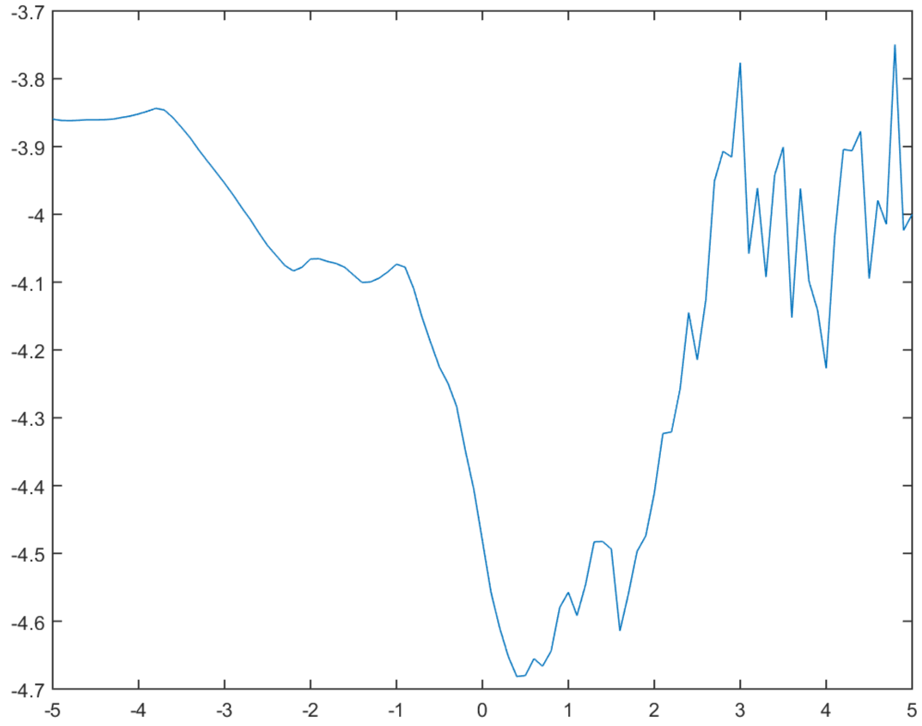


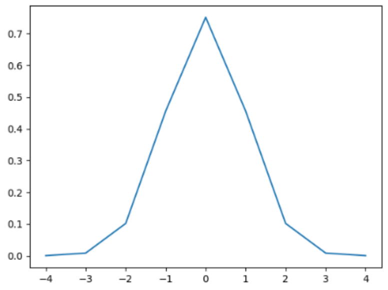
1D Convolution Examples

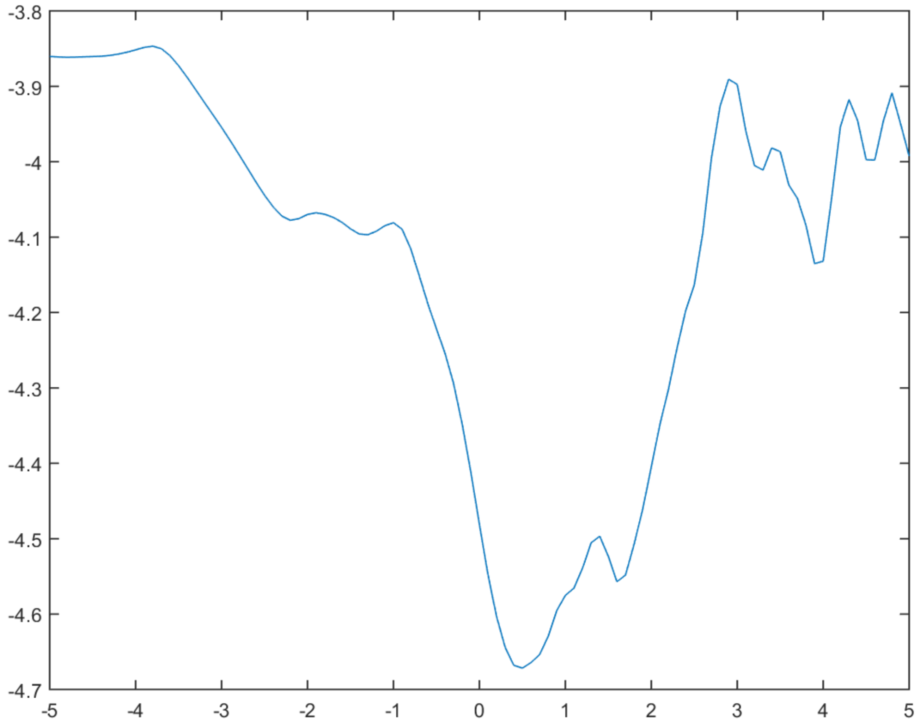
- Gaussian convolution blurs signal: $w_i \propto \exp\left(-\frac{i^2}{2\sigma^2}\right)$
  - Compared to averaging it's more smooth and maintains peaks better.

$W = [0.0001 \;\; 0.0044 \;\; 0.0540 \;\; 0.2420 \;\; 0.3989 \;\; 0.2420 \;\; 0.0540 \;\; 0.0044 \;\; 0.0001]$

$(\sigma = 1, \; m = 4)$



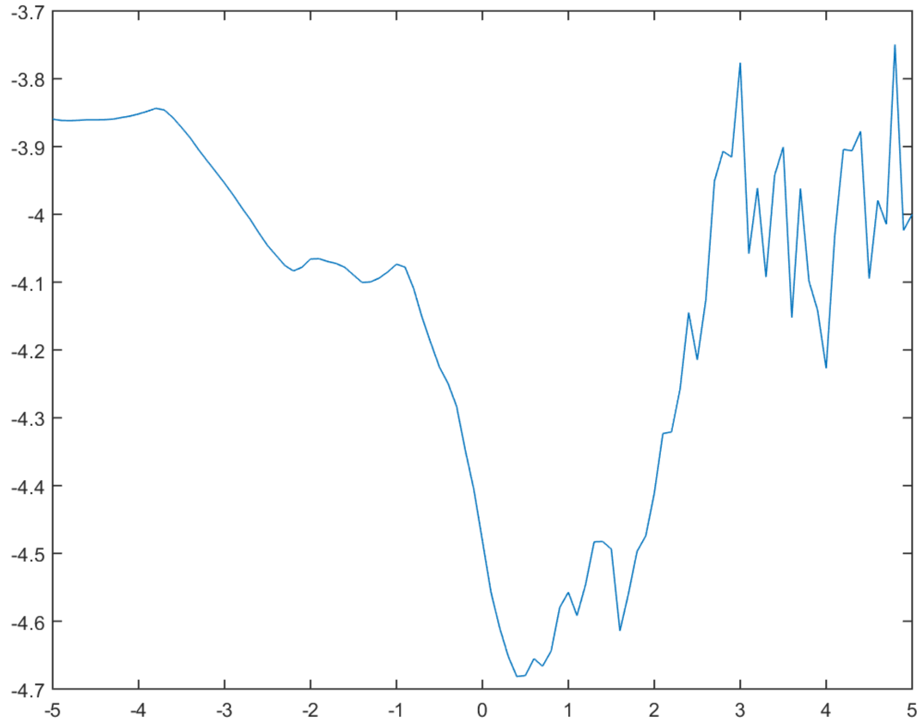


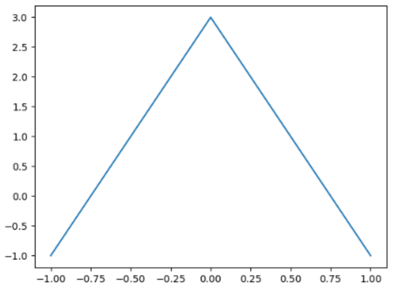


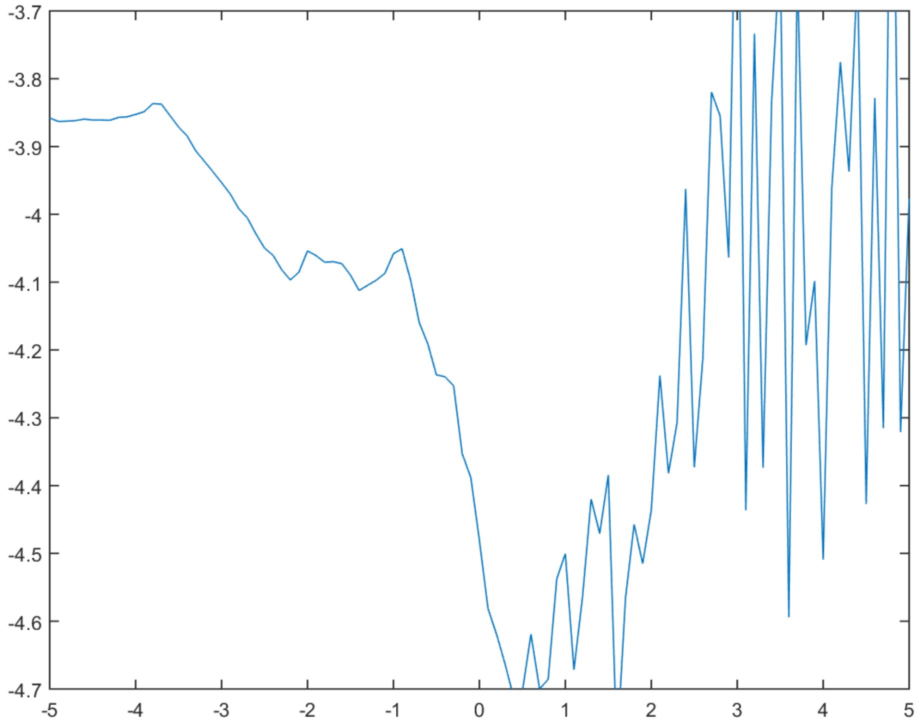
1D Convolution Examples

- Sharpen convolution enhances peaks.
  - An "average" that places negative weights on the surrounding pixels.

$w = [-1 \quad 3 \quad -1]$



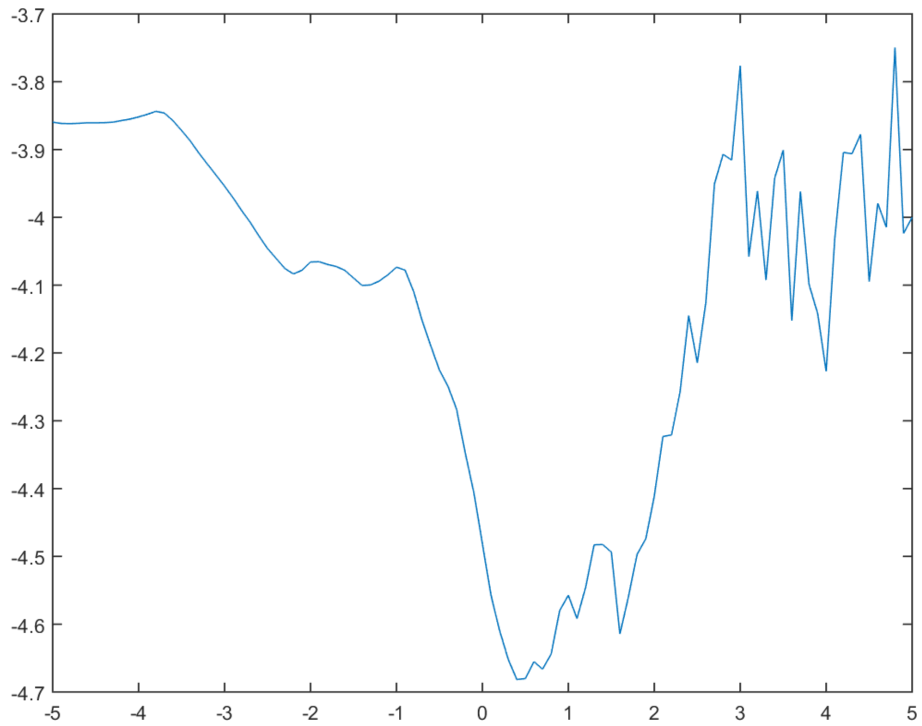


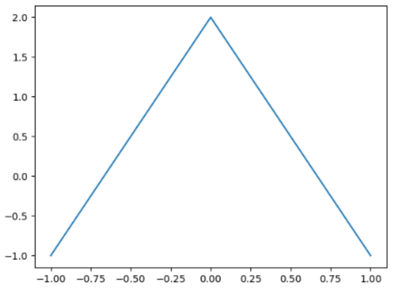


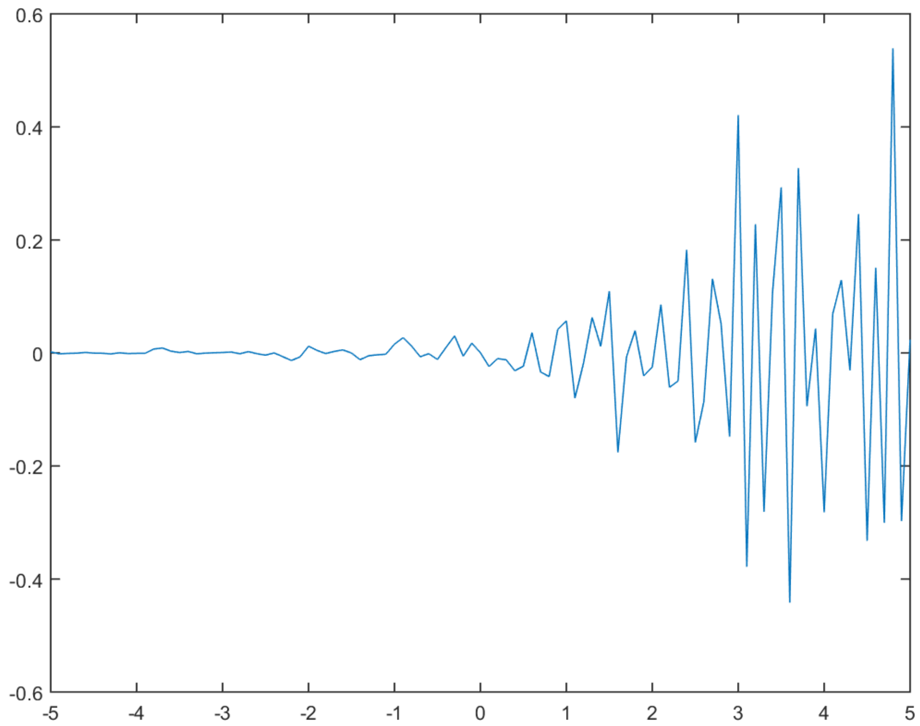
1D Convolution Examples

- Laplacian convolution approximates second derivative:
  - "Sum to zero" filters "respond" if input vector looks like the filter

$w = [-1 \quad 2 \quad -1]$

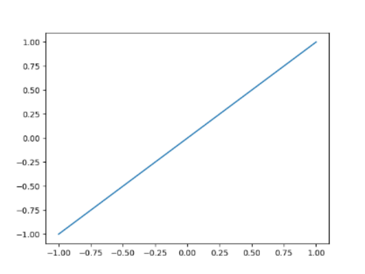




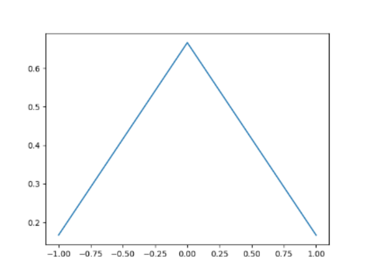


Digression: Derivatives and Integrals

- Numerical derivative approximations can be viewed as filters:
  - Centered difference: [-1, 0, 1] (derivativeCheck in findMin).



- Numerical integration approximations can be viewed as filters:
  - "Simpson's" rule: [1/6, 4/6, 1/6] (a bit like Gaussian filter).



- Derivative filters add to 0, integration filters add to 1,
  - For constant function, derivative should be 0 and average = constant.

1D Convolution Examples

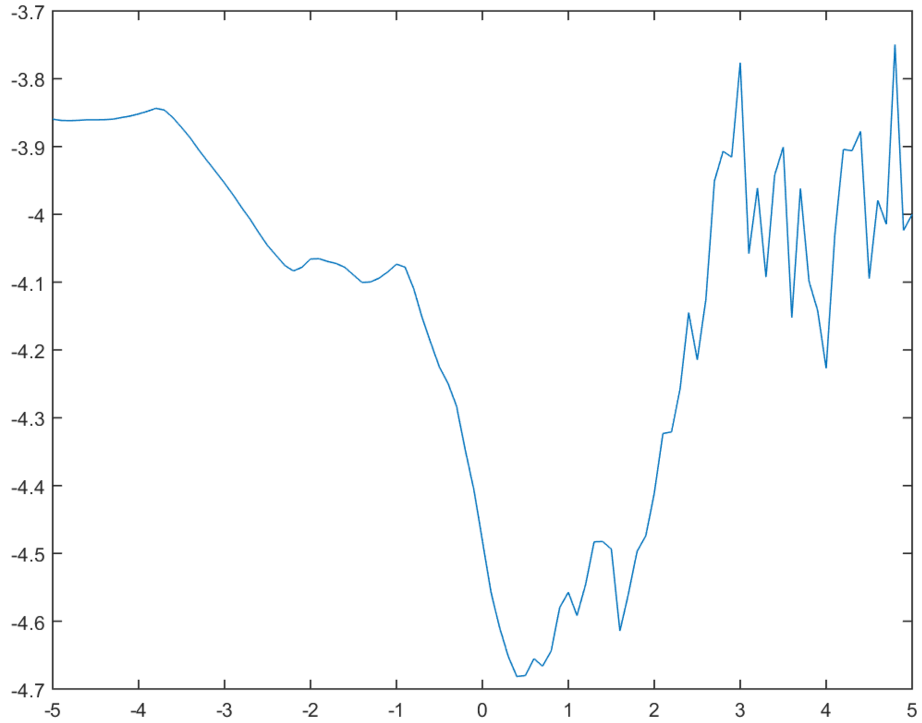
- Laplacian of Gaussian is a smoothed 2nd-derivative approximation:

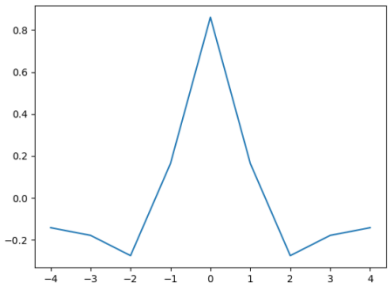
$$w_i = \left(1 - \frac{i^2}{2\sigma^2}\right)\exp\left(-\frac{i^2}{2\sigma^2}\right)$$

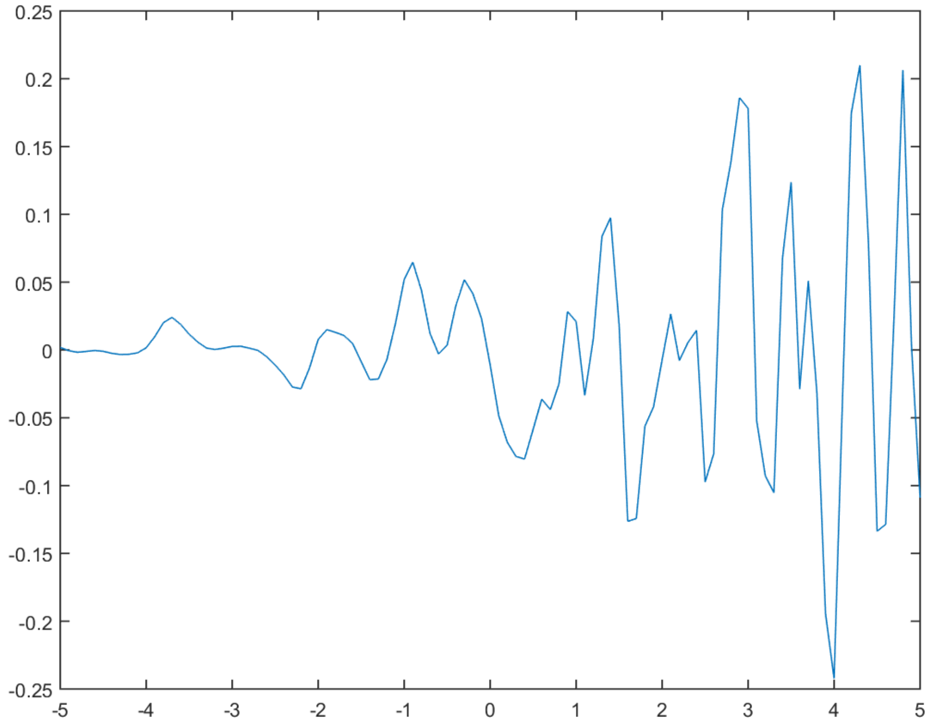
(then subtract mean)

$w = [-0.1416\ \ -0.1781\ \ -0.2746\ \ 0.1640\ \ 0.8607\ \ 0.1640\ \ -0.2746\ \ -0.1781\ \ -0.1416]$

$(\sigma^2 = 1,\ m = 4)$

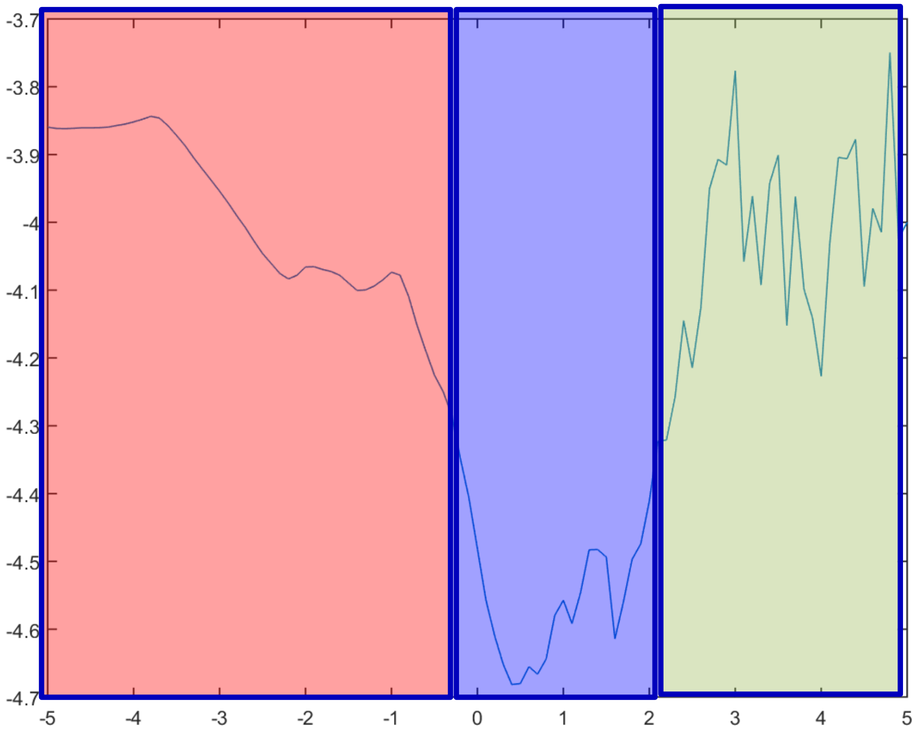




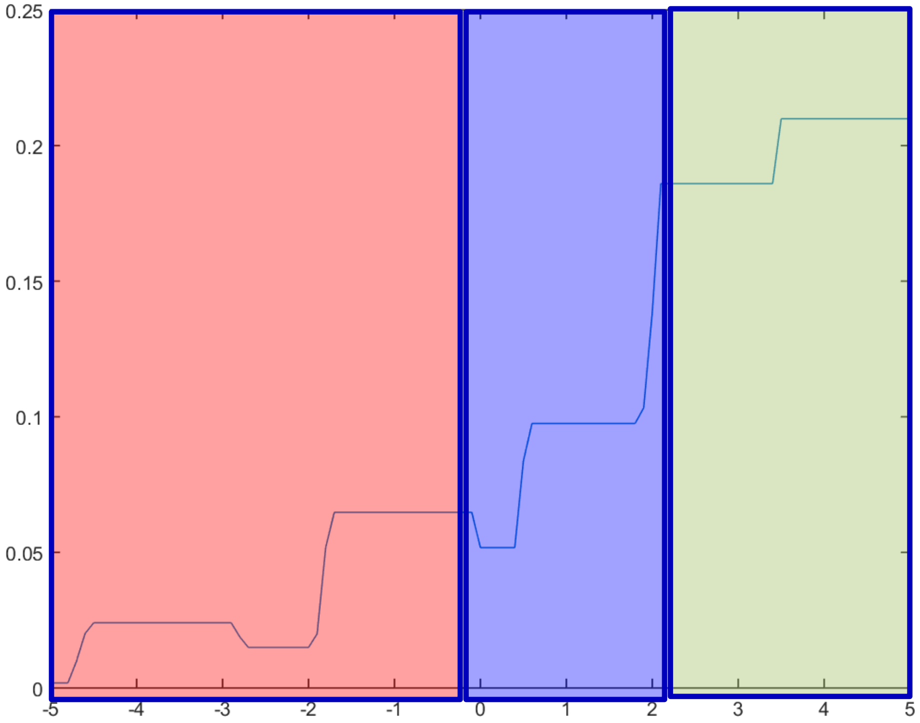


1D Convolution Examples

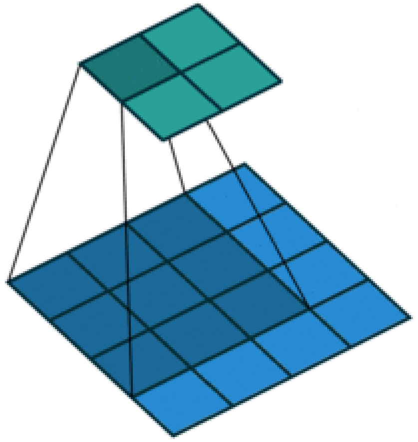
- We often use maximum over several convolutions as features:
  - Below is maximum of Laplacian of Gaussian at 'i' and its 16 KNNs.
  - We use different convolutions as our features (derivatives, integrals, etc.).



In this case, these 2 features are all we need.



Images and Higher-Order Convolution



- 2D convolution:
  - Signal 'x' is the pixel intensities in an 'n' by 'n' image.
  - Filter 'w' is the pixel intensities in a '2m+1' by '2m+1' image.
- The 2D convolution is given by:

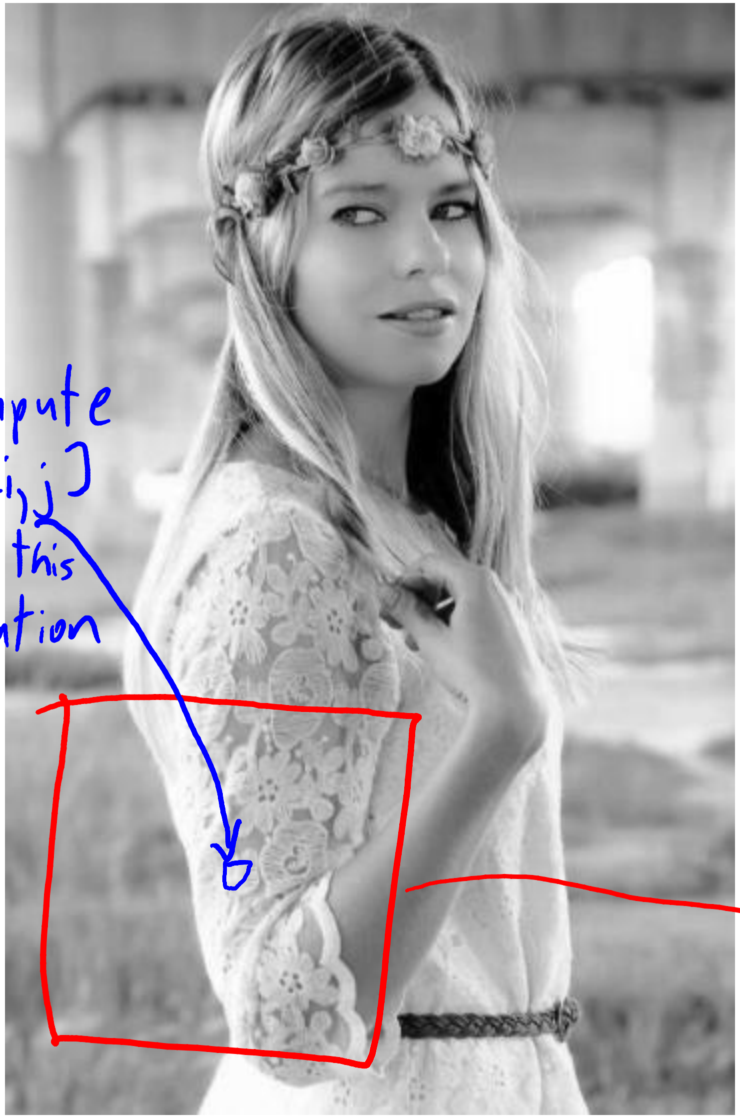
$$z[i_1, i_2] = \sum_{j_1=-m}^{m} \sum_{j_2=-m}^{m} w[j_1, j_2]\, x[i_1 + j_1, i_2 + j_2]$$

- 3D and higher-order convolutions are defined similarly.

$$z[i_1, i_2, i_3] = \sum_{j_1=-m}^{m} \sum_{j_2=-m}^{m} \sum_{j_3=-m}^{m} w[j_1, j_2, j_3]\, x[i_1 + j_1, i_2 + j_2, i_3 + j_3]$$

Image Convolution Examples

x

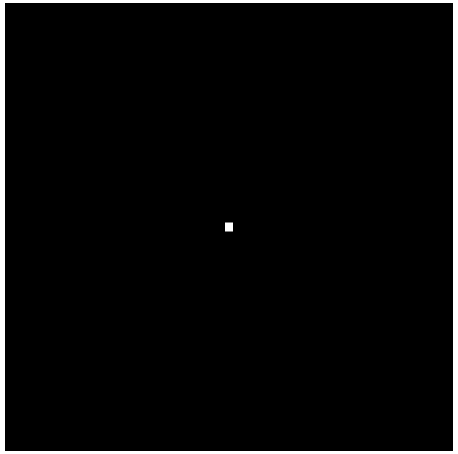


Compute
z[i,j]
for this
location

Identity convolution:
(zeroes with a '1' at $w_{0,0}$)

w

$*$



=

z

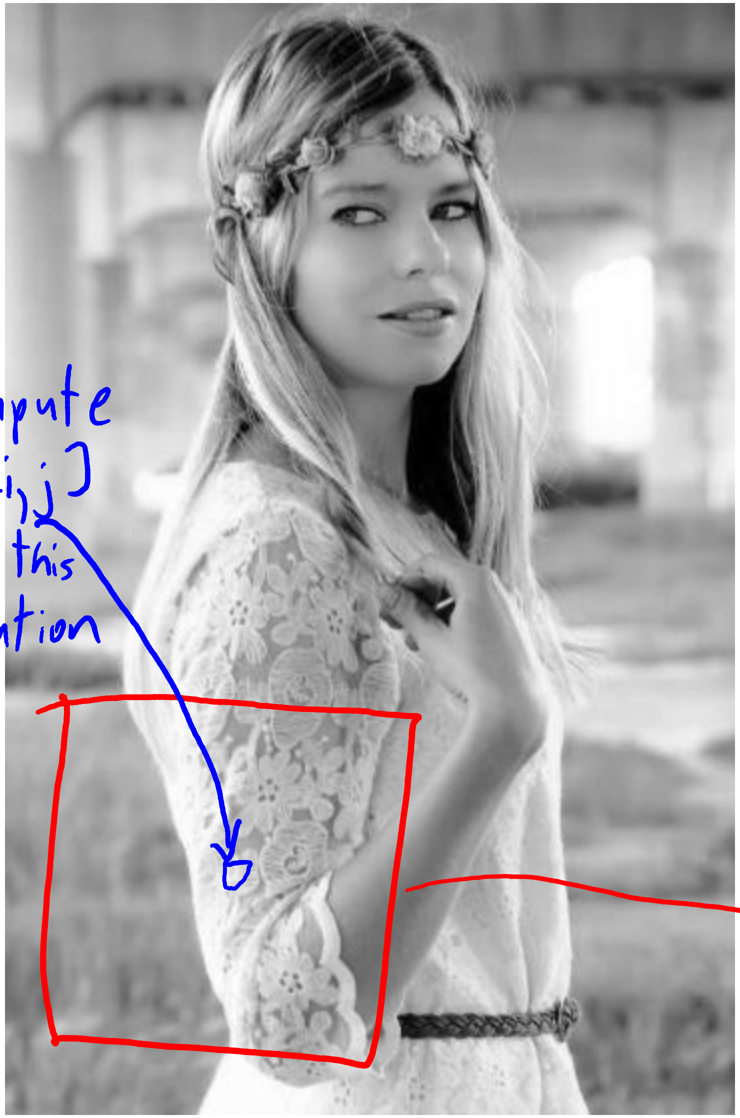
multiply element-wise
and add up result to get

z[i,j]

Image Convolution Examples

x

Compute z[i,j] for this location

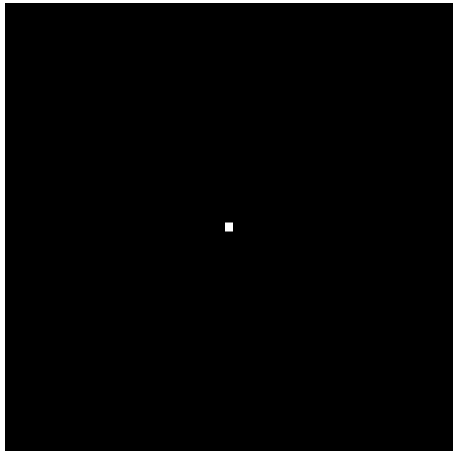


Identity convolution:

(zeroes with a '1' at $w_{0,0}$)

w

$*$



multiply element-wise and add up result to get

=

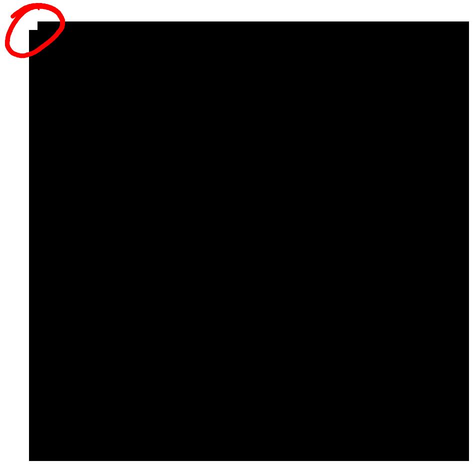
z

z[i,j]

Image Convolution Examples

Translation Convolution:

$*$



$=$

Boundary: "zero"

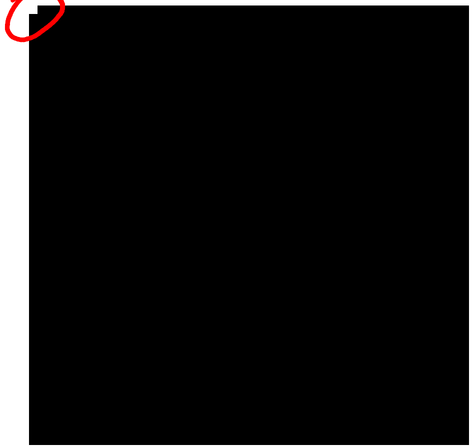
0

0

Image Convolution Examples

Translation Convolution:

$*$



$=$

Boundary: "replicate"

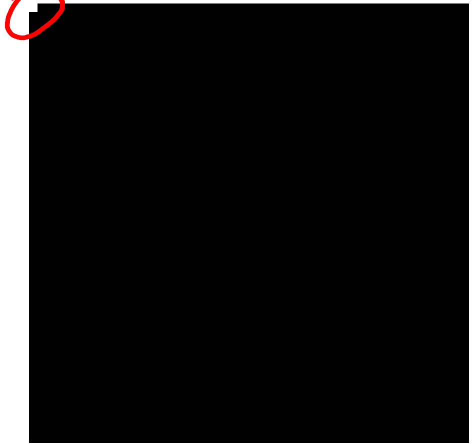
repeats

repeats

Image Convolution Examples

Translation Convolution:

$*$



$=$

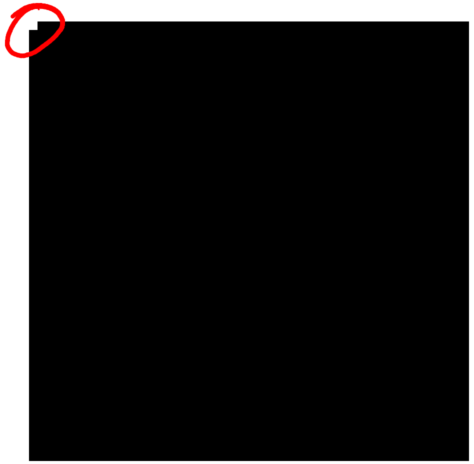
Boundary: "mirror"

flips

Image Convolution Examples

Translation Convolution:

*



=

Boundary: "ignore"

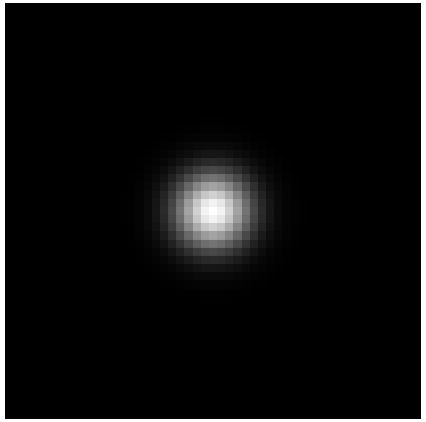
Image Convolution Examples

Average convolution:

$$* \frac{1}{51}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \ddots & 1 \\ 1 & 1 & 1 & & 1 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix} =$$

Image Convolution Examples

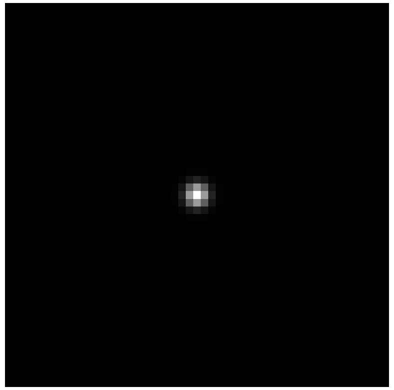
Gaussian Convolution:

$*$  $=$

<u>blurs</u> image to represent average (smoothing)

Image Convolution Examples

Gaussian Convolution:

$*$  $=$ 

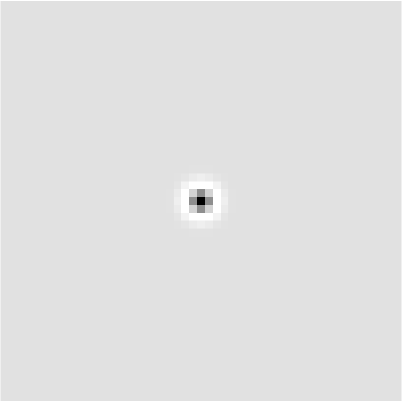
(smaller variance)

blurs image to represent average (smoothing)

Image Convolution Examples

Laplacian of Gaussian

$*$



$=$

"How much does it look like a black dot surrounded by white?"

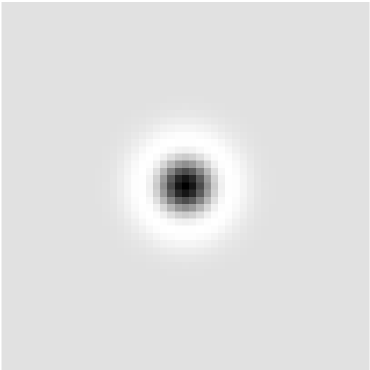
"signed" image (gray is 0)

Image Convolution Examples

Black/white as sides of edge

Laplacian of Gaussian

$*$



$=$

(larger variance)

Similar preprocessing may be done in basal ganglia and LGN.

Image Convolution Examples

"Emboss" filter:

$$* \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} =$$

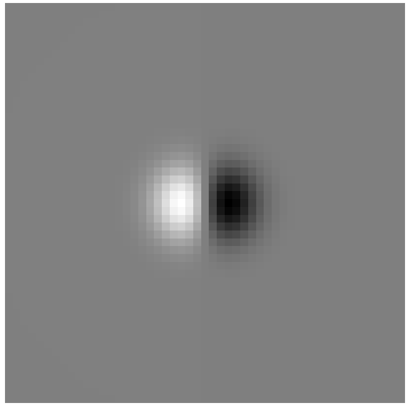
Many Photoshop effects are just convolutions.

http://setosa.io/ev/image-kernels

Image Convolution Examples

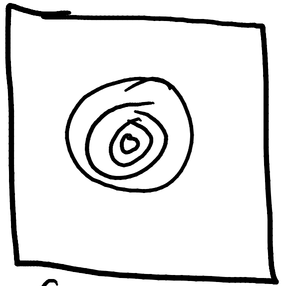
Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$



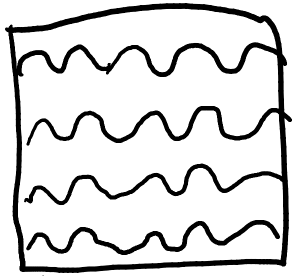
$=$

$\|$

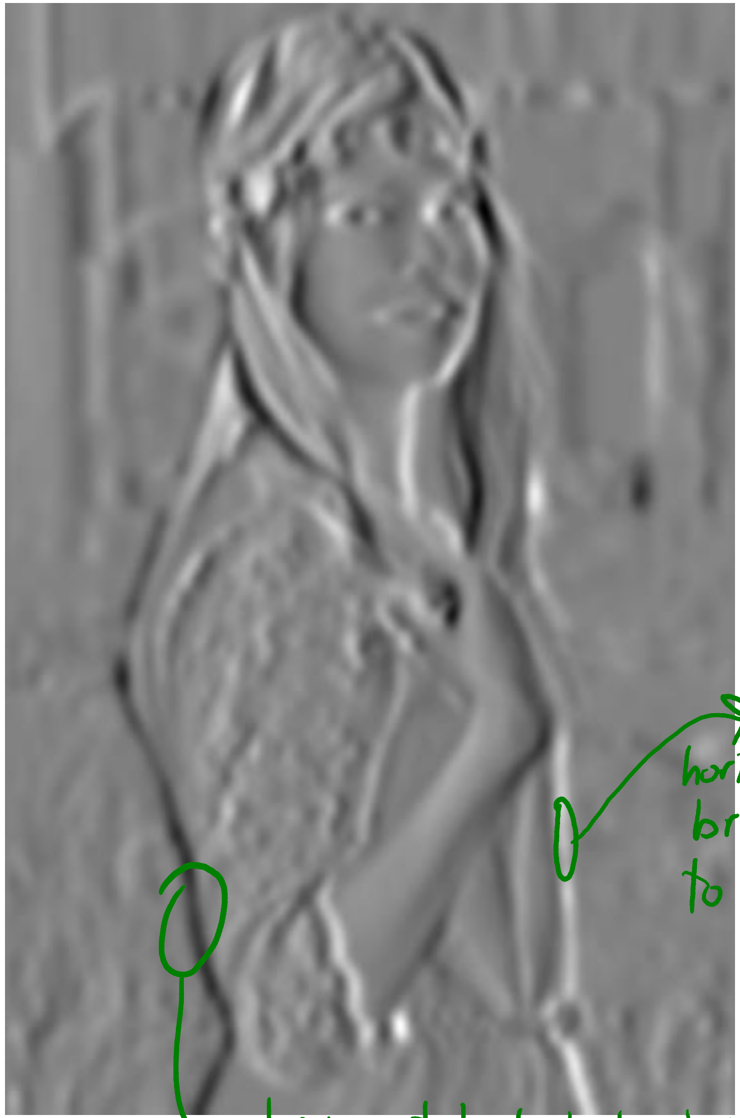


Gaussian

$.*$



Parallel Sine functions



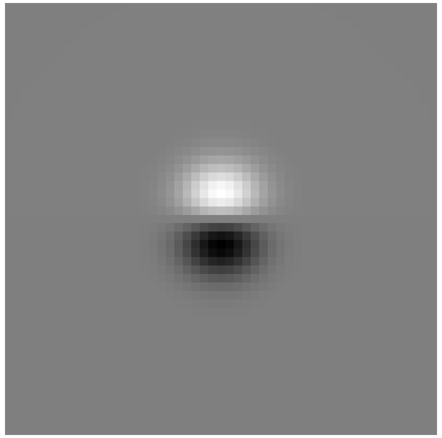
horizontal bright to dark

horizontal dark to bright

Image Convolution Examples

Gabor Filter
(Gaussian multiplied by
sine or cosine)

$*$



$=$

Different orientations of
the sine/cosine let us
detect changes with different
orientations.

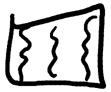
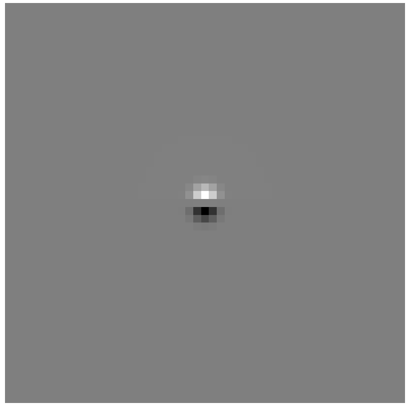


Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$



(smaller variance)

$=$

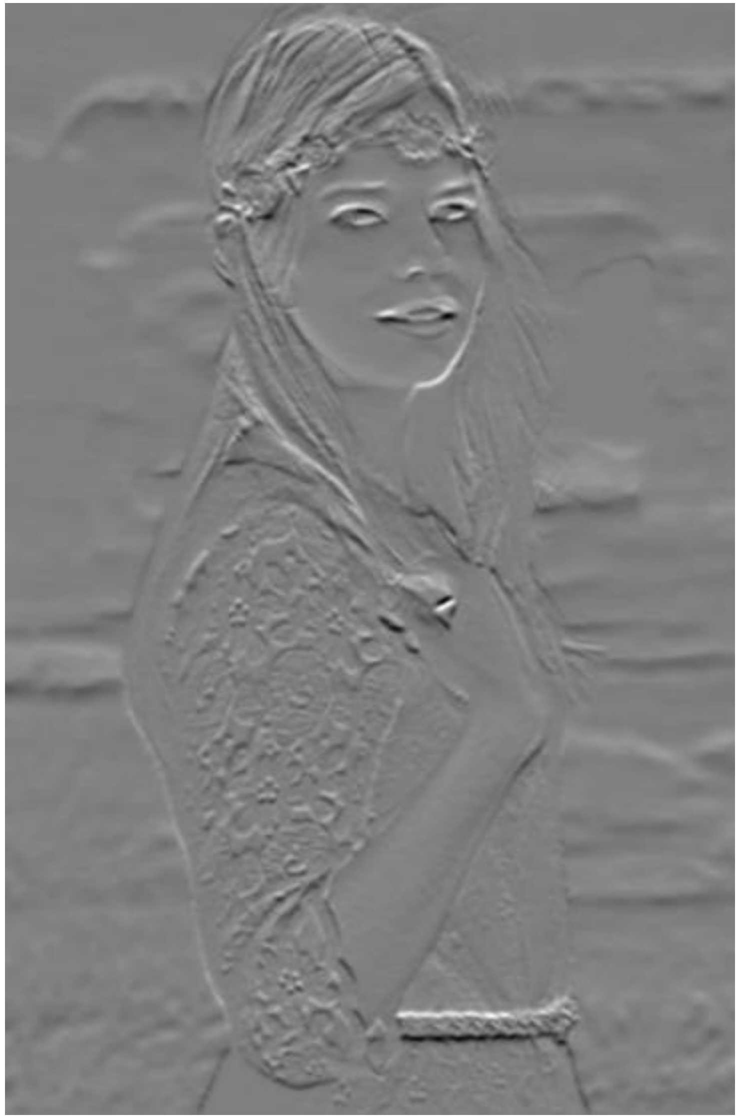
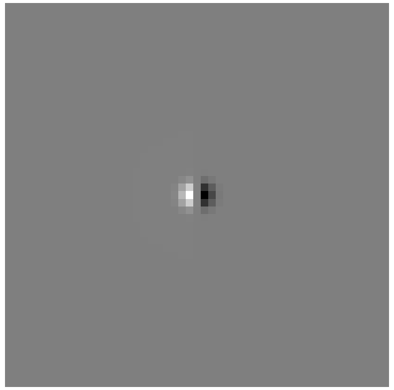


Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$



$=$

(smaller variance)

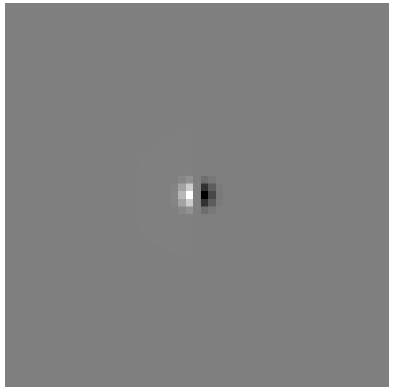
Vertical orientation

- Can obtain other orientations by rotating.
- May be similar to effect of V1 "simple cells."

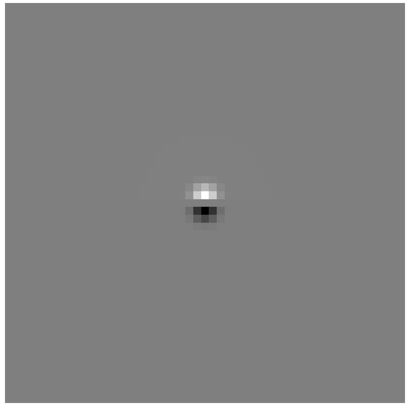
Image Convolution Examples

Max absolute value between horizontal and vertical Gabor:

*



*



maximum absolute value

"Horizontal/vertical edge detector"

3D Convolution

Represent as RGB

Can apply 3D convolutions

3D Convolution

Gaussian filter

3D Convolution

Gaussian filter
(higher variance on
green channel)

3D Convolution

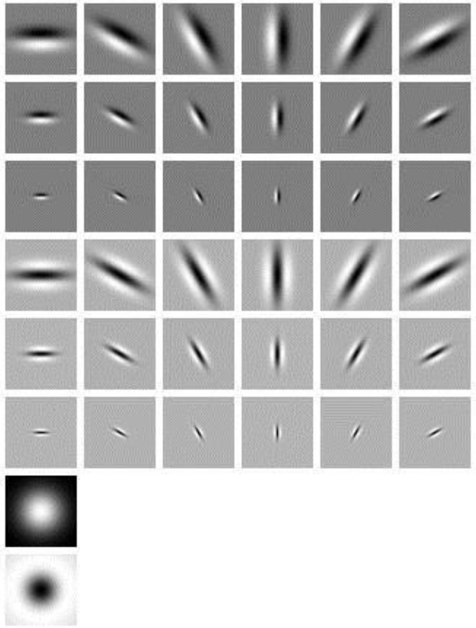
Sharpen the blue channel.

3D Convolution

Gabor filter on
each channel.

Filter Banks

- To characterize context, we used to use filter bank like "MR8":
  - 1 Gaussian filter, 1 Laplacian of Gaussian filter.
  - 6 max(Gabor) filters: 3 scales of sine/cosine (maxed over orientations).



- Convolutional neural networks are now replacing filter banks.

http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

(pause)

1D Convolution as Matrix Multiplication

- Each element of a convolution is an inner product:

$$z_i = \sum_{j=-m}^{m} w_j x_{i+j}$$

$$= w^T x_{(i-m:i+m)}$$

$$= \tilde{w}^T x \quad \text{where} \quad \tilde{w} = [0 \quad 0 \quad 0 \quad \overbrace{\text{——} \; w \; \text{——}}^{\text{positions } i-m \text{ through } i+m} \quad 0 \quad 0]$$

- So convolution is a matrix multiplication (I'm ignoring boundaries):

$$z = \tilde{W}x \quad \text{where} \quad \tilde{W} = \begin{bmatrix} \text{——} & w & \text{——} & 0 & 0 & 0 \\ 0 & \text{——} & w & \text{——} & 0 & 0 \\ 0 & 0 & \text{——} & w & \text{——} & 0 \\ 0 & 0 & 0 & \text{——} & w & \text{——} \end{bmatrix}$$

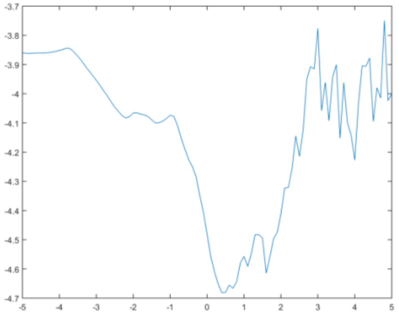
matrix can be very sparse and only has 2m+1 variables.

- The shorter 'w' is, the more sparse the matrix is.

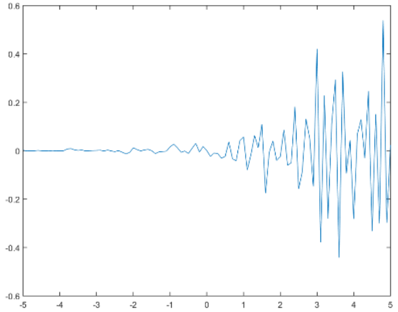
1D Convolution as Matrix Multiplication

- 1D convolution:
  - Takes signal ‘x’ and filter ‘w’ to produces vector ‘z’:

$$x \quad * \quad w \quad = \quad z$$



$$[-1 \quad 2 \quad -1]$$



  - Can be written as a matrix multiplication:

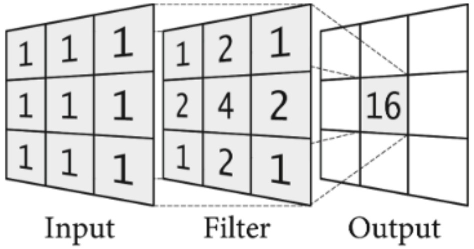
$$Wx = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & & & \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & -2 & 1 \end{bmatrix} x = z$$

2D Convolution as Matrix Multiplication

- 2D convolution:
  - Signal ‘x’, filter ‘w’, and output ‘z’ are now all images/matrices:

$$x \ast w = z$$

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$



  - Vectorized ‘z’ can be written as a matrix multiplication with vectorized ‘x’:

$$W = \left[\begin{array}{ccccccccc|ccccccccc|ccccccccc|ccc}
-2 & -1 & 0 & 0 & 0 & 0 & \cdots & & 0 & -1 & 0 & 1 & 0 & 0 & 0 & \cdots & & 0 & 0 & 1 & 2 & 0 & 0 & 0 & \cdots & & 0 & 0 & 0 & \cdots \\
0 & -2 & -1 & 0 & 0 & 0 & \cdots & & 0 & 0 & -1 & 0 & 1 & 0 & 0 & \cdots & & 0 & 0 & 0 & 1 & 2 & 0 & 0 & \cdots & & 0 & 0 & 0 & \cdots \\
 & & \vdots & & & \cdots & 2 & -1 & 0 & 0 & 0 & 0 & \cdots & & & -1 & 0 & 1 & 0 & 0 & 0 & \cdots & & 0 & 1 & 2 & 0 & 0 & \cdots \\
0 & 0 & 0 & & & & & & & & & & & & & & & & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & & 0 & 2 & -1 & 0 & 0 & 0 & 0 & \cdots & & 0 & -1 & 0 & 1 & 0 & 0 & 0 & \cdots & & 0 & 0 & 1 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & & 0 & 0 & 2 & -1 & 0 & 0 & 0 & \cdots & & 0 & 0 & -1 & 0 & 1 & 0 & 0 & \cdots & & 0 & 0 & 0 & 1 \\
 & & \vdots & & & & & & & & & & & \vdots & & & & & & & & & & \vdots & & & & & &
\end{array}\right]$$

Motivation for Convolutional Neural Networks

- Consider training neural networks on 256 by 256 images.
  - This is 256 by 256 by 3 ≈ 200,000 inputs.
- If first layer has k=10,000, then it has about 2 billion parameters.
  - We want to avoid this huge number (due to storage and overfitting).

- Key idea: make Wx_i act like several convolutions (to make it sparse):
  1. Each row of W only applies to part of x_i.
  2. Use the same parameters between rows.

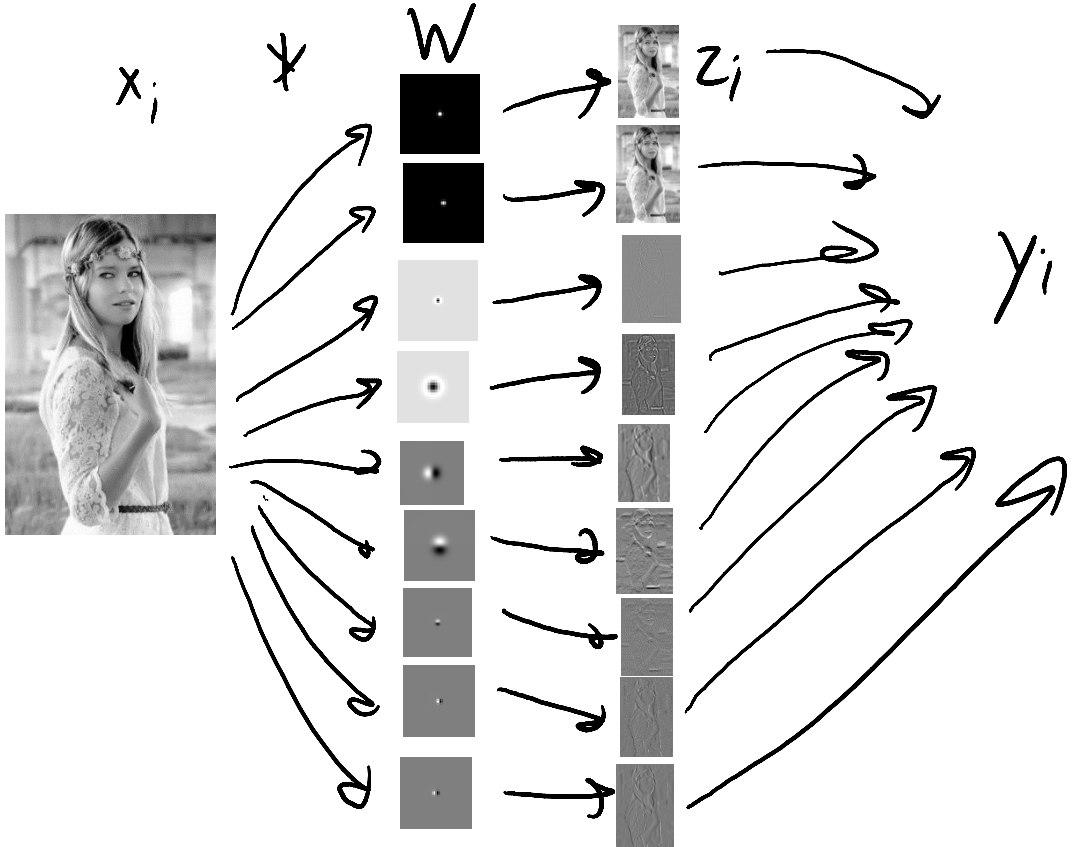
$$w_1 = [0 \;\; 0 \;\; 0 \;\text{——}\; w \;\text{——}\; 0 \;\; 0 \;\; 0]$$

$$w_2 = [0 \;\text{——}\; w \;\text{——}\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]$$

- Forces most weights to be zero, reduces number of parameters.

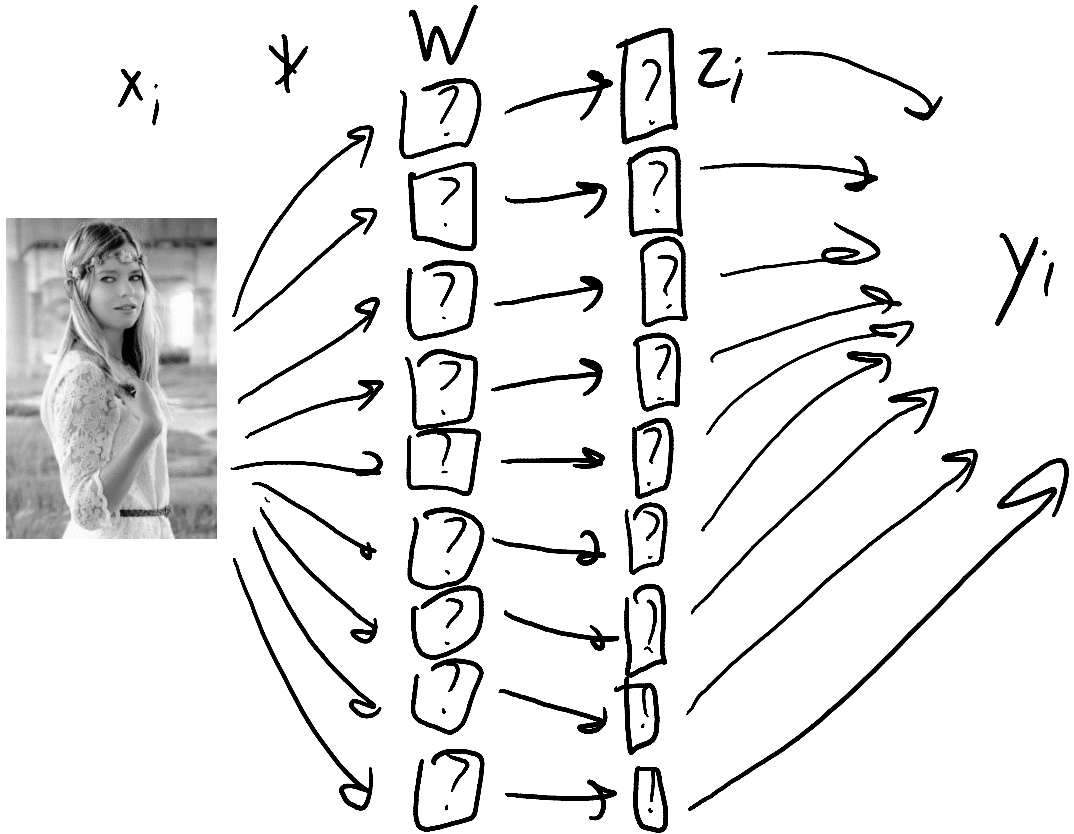
Motivation for Convolutional Neural Networks

- Classic vision methods uses fixed convolutions as features:
  - Usually have different types/variances/orientations.
  - Can do subsampling or take maxes across locations/orientations/scales.



Motivation for Convolutional Neural Networks

- Convolutional neural networks learn the features:
  - Learning ‘W’ and ‘v’ automatically chooses types/variances/orientations.
  - Don’t pick from fixed convolutions, but learn the elements of the filters.



Convolutional Neural Networks

- Convolutional Neural Networks classically have 3 layer "types":
  - Fully connected layer: usual neural network layer with unrestricted W.

$$W^{(m)} = \begin{bmatrix} \text{———} & w_1^{(m)} & \text{———} \\ \text{———} & w_2^{(m)} & \text{———} \\ & \vdots & \\ \text{———} & w_k^{(m)} & \text{———} \end{bmatrix}$$

Convolutional Neural Networks

- Convolutional Neural Networks classically have 3 layer “types”:
  - Fully connected layer: usual neural network layer with unrestricted W.
  - Convolutional layer: restrict W to act like several convolutions.

1D example

distance between centers of convolution is called "stride"

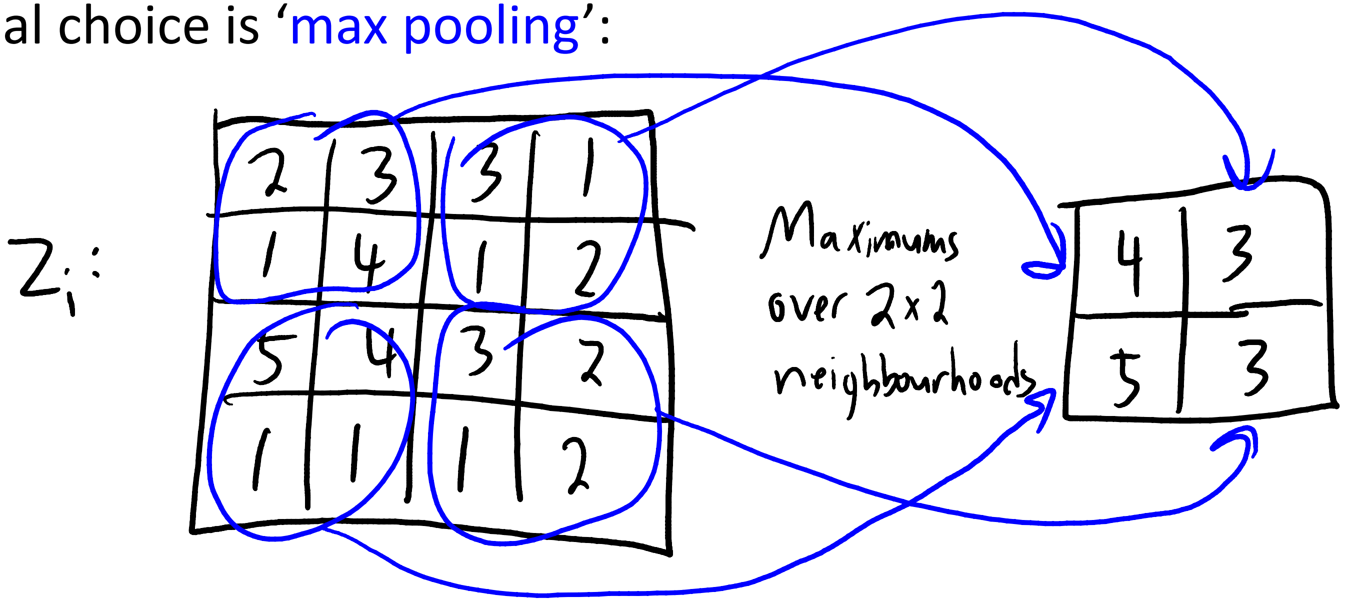
$$W^{(m)} = \begin{bmatrix} \cdots & w_1^{(m)} & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & w_1^{(m)} & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & w_1^{(m)} & \cdots \\ \cdots & w_2^{(m)} & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & w_2^{(m)} & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & w_2^{(m)} & \cdots \end{bmatrix}$$

Same $w_1^{(m)}$ used across multiple rows.

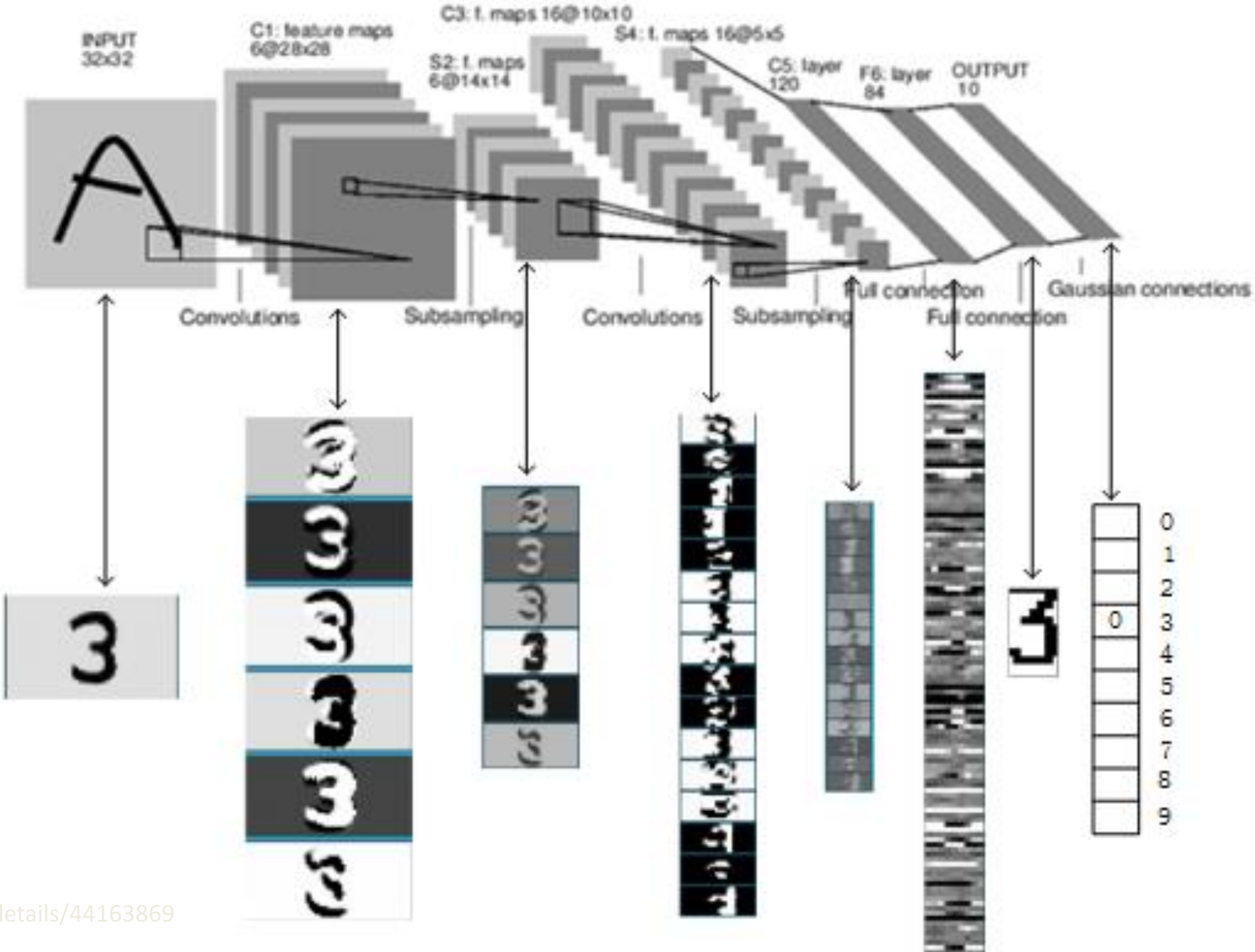
Sparse and small number of parameters

Convolutional Neural Networks

- Convolutional Neural Networks classically have 3 layer "types":
  - Fully connected layer: usual neural network layer with unrestricted W.
  - Convolutional layer: restrict W to act like several convolutions.
  - Pooling layer: combine results of convolutions.
    - Can add invariances or just make the number of parameters smaller.
    - Usual choice is 'max pooling':



LeNet for Optical Character Recognition



Summary

- Convolutions are flexible class of signal/image transformations.
  - Can approximate directional derivatives and integrals at different scales.
- Max(convolutions) can yield features that make classification easy.
- Convolutional neural networks:
  - Restrict $W^{(m)}$ matrices to represent sets of convolutions.
  - Often combined with max (pooling).

- Next time: modern convolutional neural networks and applications.
  - Image segmentation, depth estimation, image colorization, artistic style.

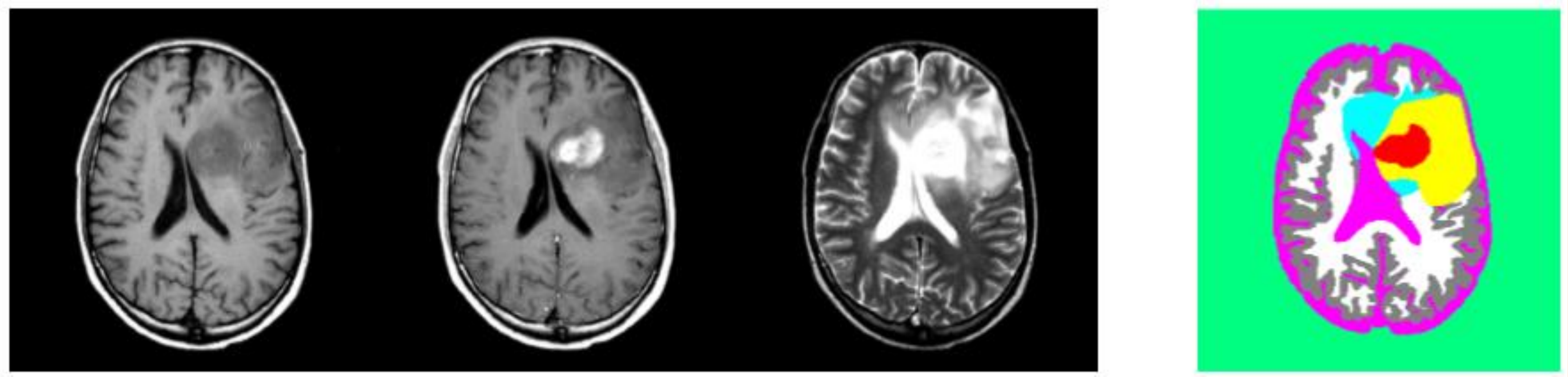
FFT implementation of convolution

- Convolutions can be implemented using fast Fourier transform:
  - Take FFT of image and filter, multiply elementwise, and take inverse FFT.

- It has faster asymptotic running time but there are some catches:
  - You need to be using periodic boundary conditions for the convolution.
  - Constants matter: it may not be faster in practice.
    - Especially compared to using GPUs to do the convolution in hardware.
  - The gains are largest for larger filters (compared to the image size).

Image Coordinates

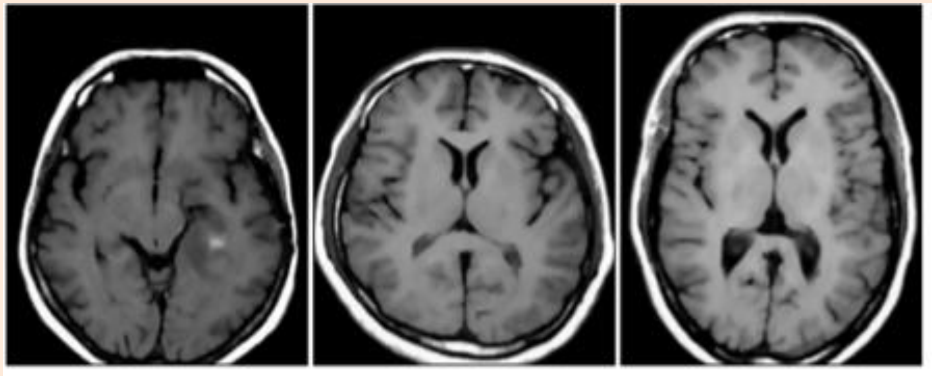
- Should we use the image coordinates?
  - E.g., the pixel is at location (124, 78) in the image.



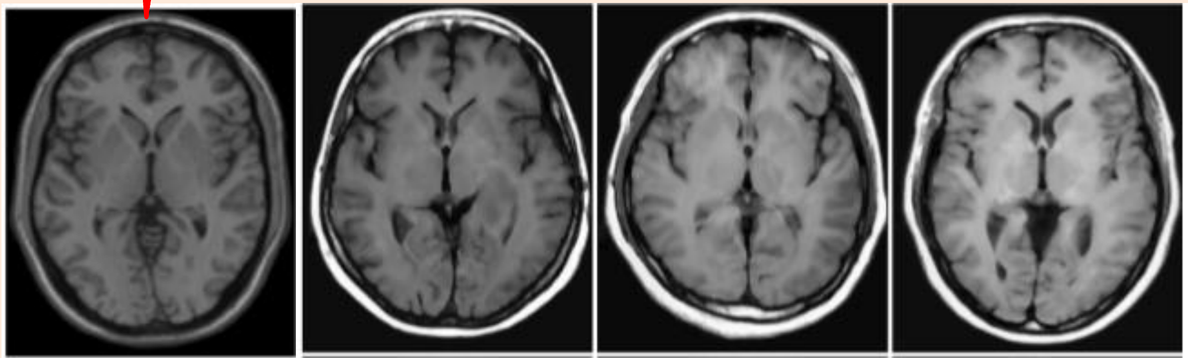
- Considerations:
  - Is the interpretation different in different areas of the image?
  - Are you using a linear model?
    - Would “distance to center” be more logical?
  - Do you have enough data to learn about all areas of the image?

Alignment-Based Features

- The position in the image is important in brain tumour application.
  - But we didn't have much data, so coordinates didn't make sense.
- We aligned the images with a "template image".



Template



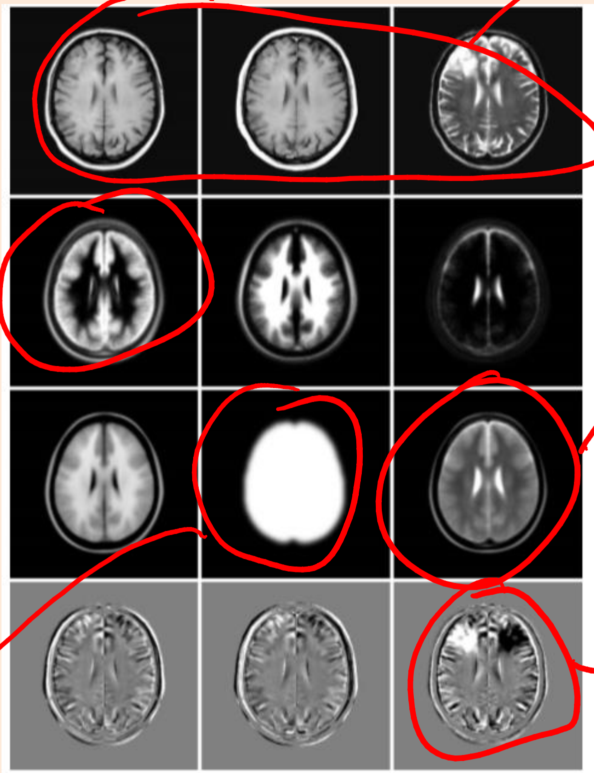
(Look different because we're showing middle slice and alignment is in 3D.)

Alignment-Based Features

- The position in the image is important in brain tumour application.
  - But we didn't have much data, so coordinates didn't make sense.
- We aligned the images with a "template image".
  - Allowed "alignment-based" features:

Original pixel values

Probability of gray matter at this pixel among tons of people aligned with template.



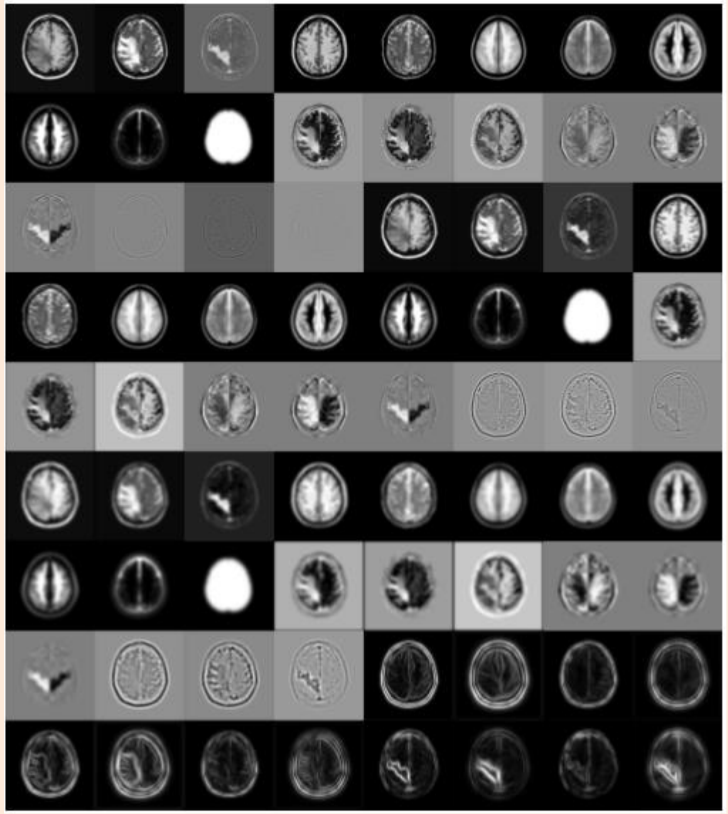
Actual pixel value of template image at this location.

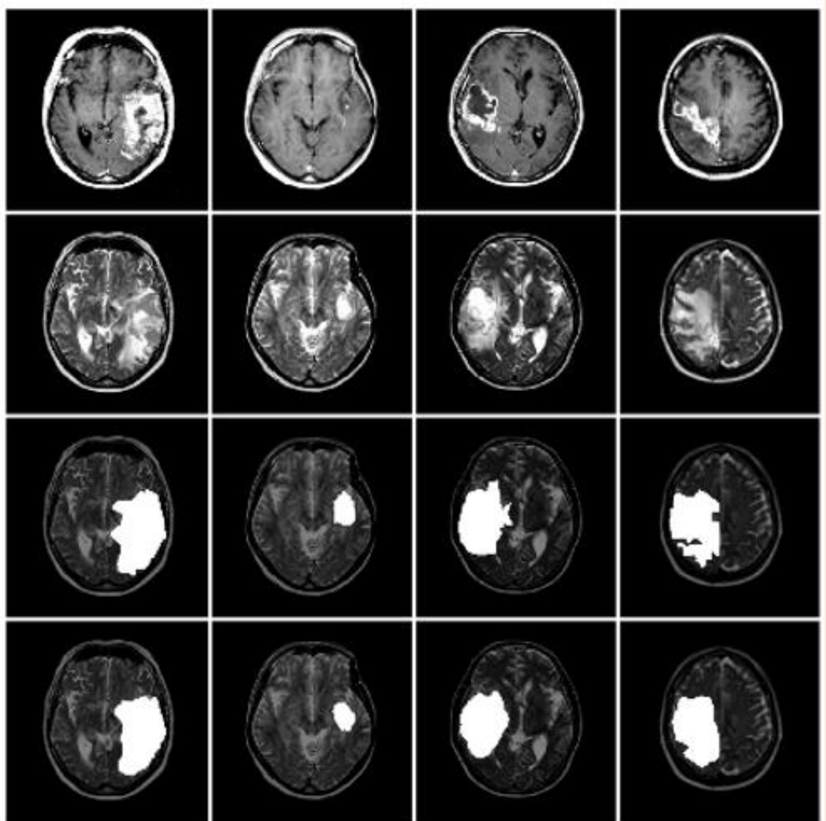
Probability of being brain pixel.

Left-right symmetry difference.

Motivation: Automatic Brain Tumor Segmentation

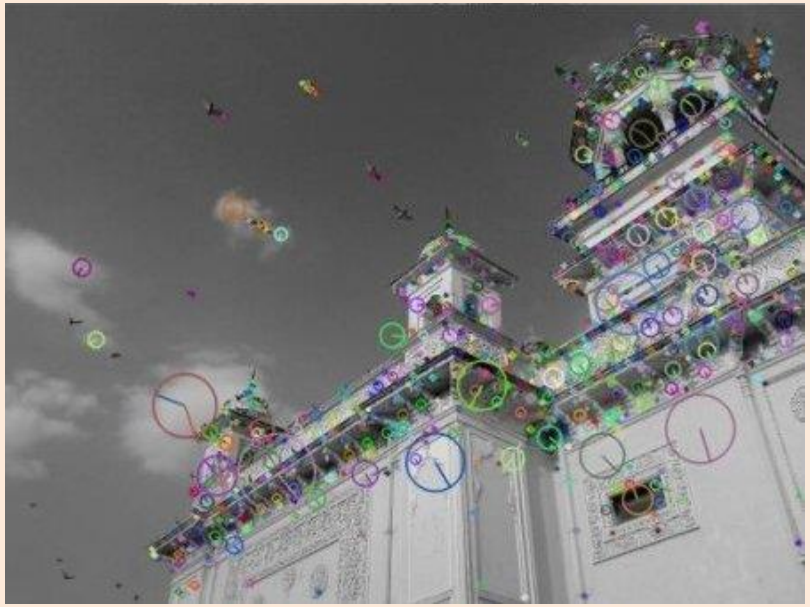
- Final features for brain tumour segmentation:
  - MR8 filter bank applied to original T1, T2, and T1 “contrast” – T1 “original”.
  - Gaussian convolution with 3 variances of alignment-based features.





SIFT Features

- Scale-invariant feature transform (SIFT):
  - Features used for object detection (“is particular object in the image”?)
  - Designed to detect unique visual features of objects at multiple scales.
  - Proven useful for a variety of object detection tasks.



http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html