CPSC 340: Machine Learning and Data Mining

Convolutional Neural Networks

Fall 2017

Admin

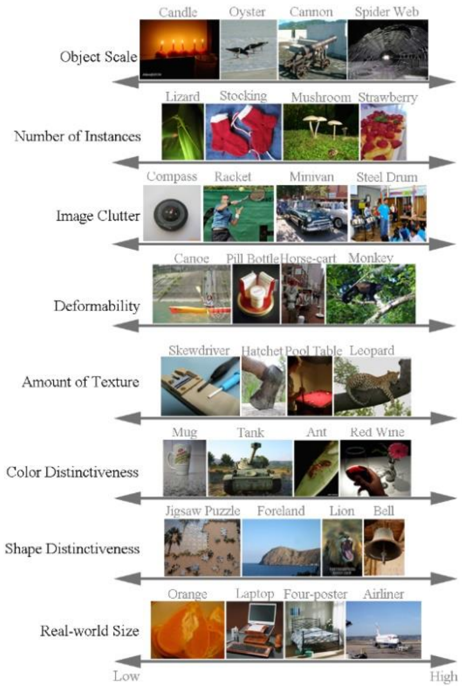
- Assignment 5:
  - Due tonight, 1 late day for Wednesday, 2 for Friday.
- Final:
  - Next Tuesday, details and previous exams posted on Piazza.
- Extra office hours:
  - 3:00-?:?? Thursday in ICICS 146 (with me).
  - Monday we'll have office hours at 11-12 (1 TA) and 1-2 (2 TAs).
  - Tuesday we'll have office hours from 12-2 (1 TA).

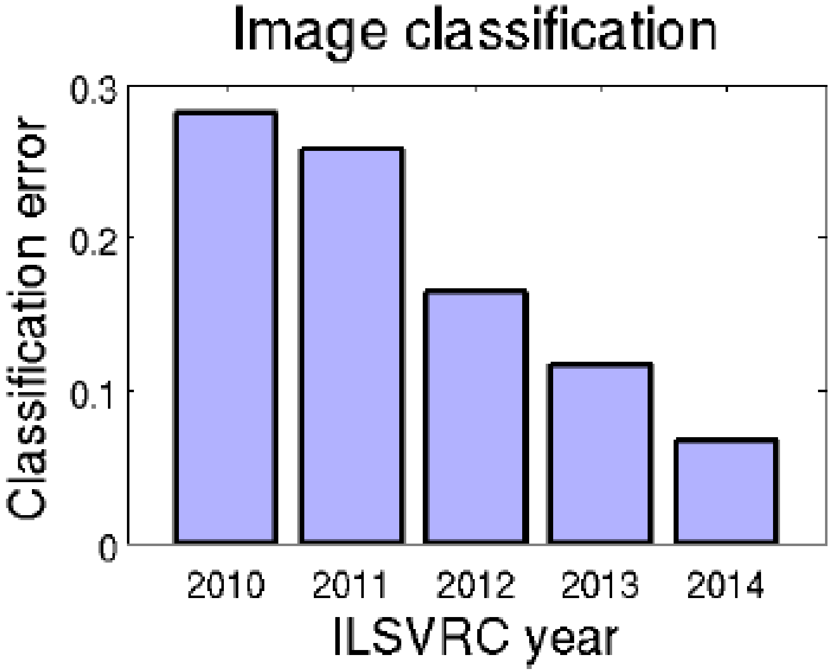
Last Lectures: Deep Learning

- We've been discussing neural network / deep learning models:

$$\hat{y}_i = v^T h(W^{(m)} h(W^{(m-1)} h(\cdots\cdots W^{(2)} h(W^{(1)} x_i))\cdots))$$

- We discussed unprecedented vision/speech performance.





https://arxiv.org/pdf/1409.0575v3.pdf

Last Lectures: Deep Learning

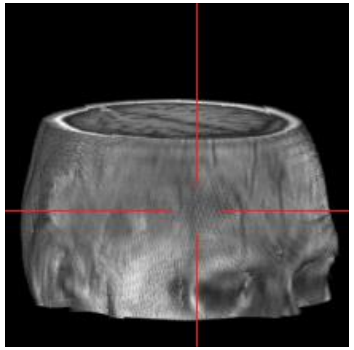
- Last time we discussed heuristics to make it work:
  - Parameter initialization and data transformations.
  - Setting the step size(s) in stochastic gradient.
  - Alternative non-linear functions like ReLU.
  - Different forms of regularization:
    - L2-regularization, early stopping, dropout.

- These are often still not enough to get deep models working.

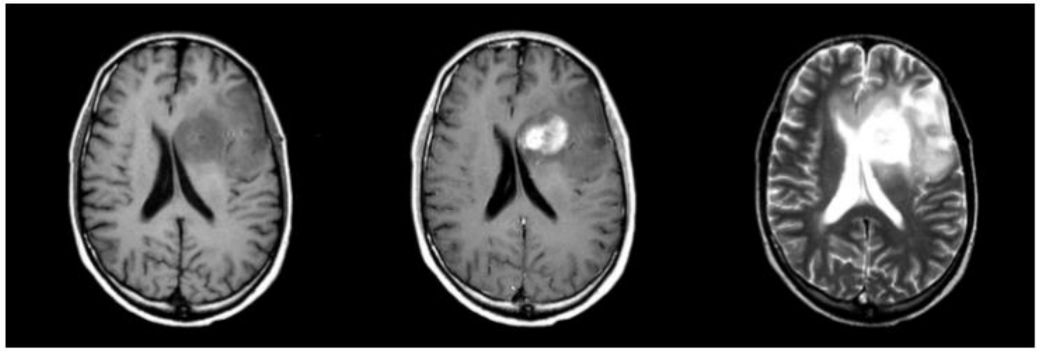
- Deep computer vision models are all convolutional neural networks:
  - The $W^{(m)}$ are very sparse and have repeated parameters ("tied weights").
  - Drastically reduces number of parameters (speeds training, reduces overfitting).

Motivation: Automatic Brain Tumor Segmentation

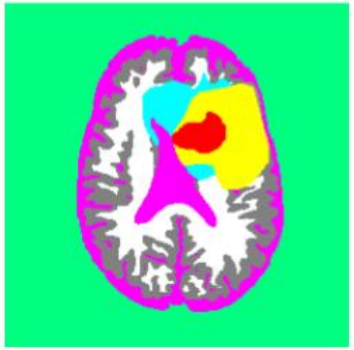
- Task: segmentation tumors and normal tissue in multi-modal MRI data.



Input:



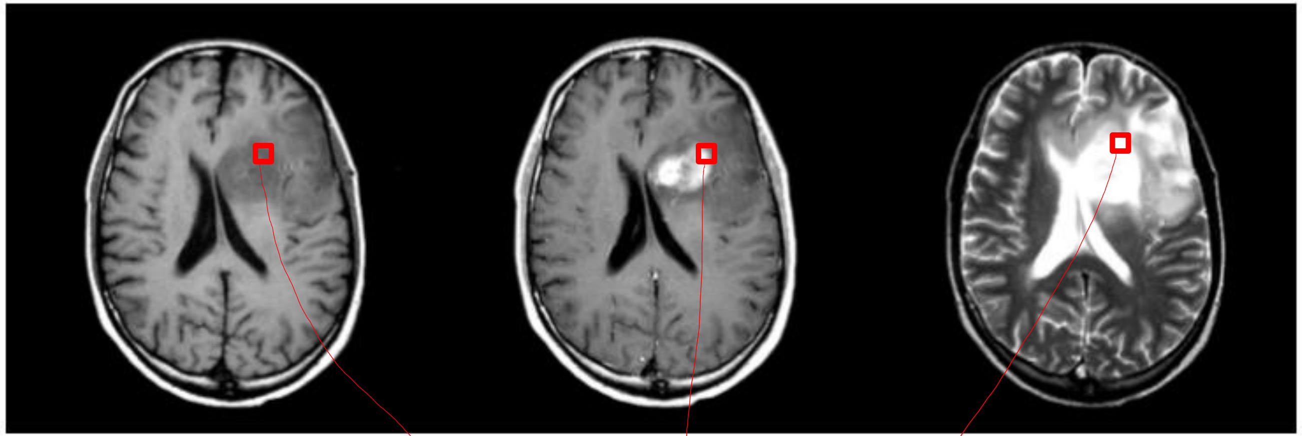
Output:



- Applications:
  - Radiation therapy target planning, quantifying treatment responses.
  - Mining growth patterns, image-guided surgery.
- Challenges:
  - Variety of tumor appearances, similarity to normal tissue.
  - “You are never going to solve this problem.”

Naïve Voxel-Level Classifier

- We could treat classifying a voxel as supervised learning:

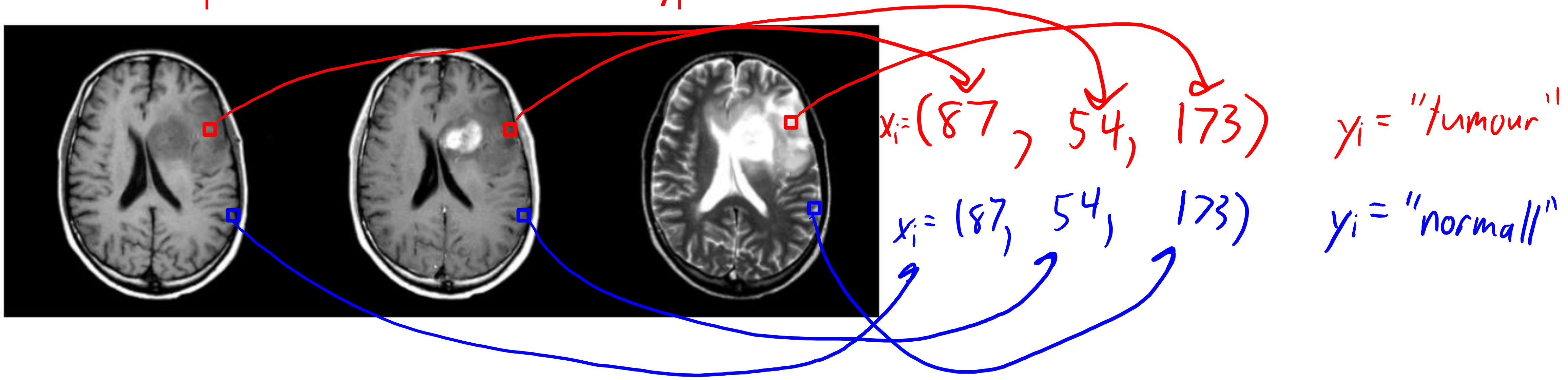


$x_i = (98, 187, 246)$ $\quad$ $y_i =$ "tumour"

- We can formulate predicting y_i given x_i as supervised learning.
- But it doesn't work at all with these features.

Need to Summarize Local Context

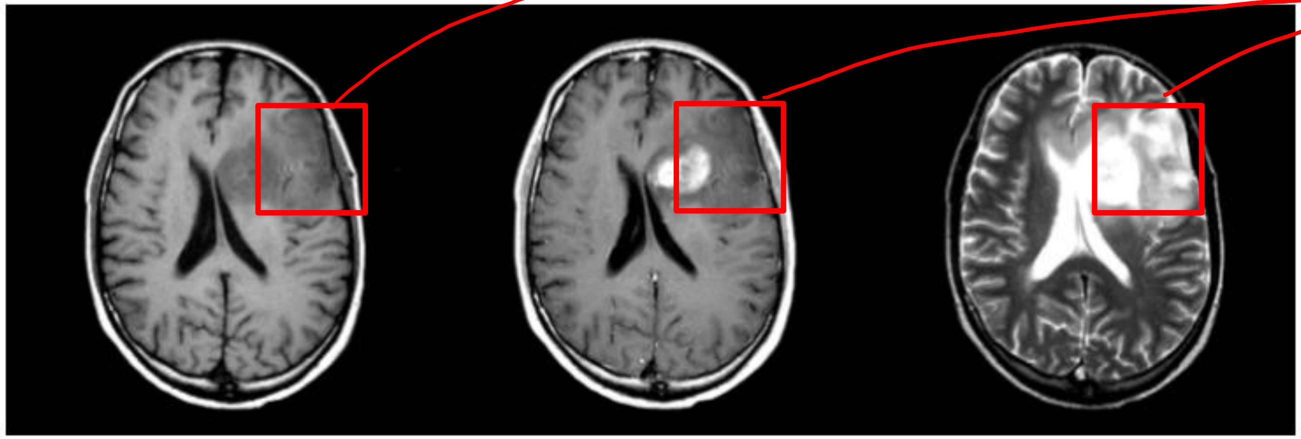
- The individual voxel values are almost meaningless:
  - This x_i could lead to different y_i.



- Intensities not standardized.
- Non-trivial overlap in signal for different tissue types.
- "Partial volume" effects at boundaries of tissue types.

Need to Summarize Local Context

- We need to represent the spatial "context" of the voxel.

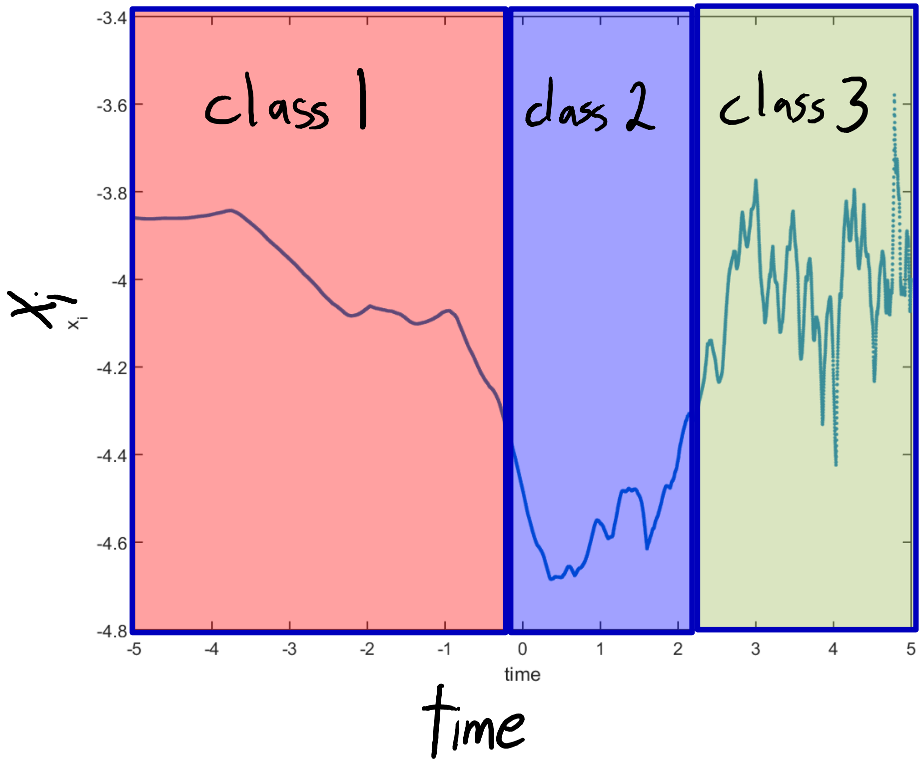


$x_i = (\overbrace{\ldots\ldots}, \overbrace{\ldots\ldots}, \overbrace{\ldots\ldots})$

  – Include all the values of neighbouring voxels?
    - Variation on coupon collection problem: requires lots of data to find patterns.
  – Measure neighbourhood summary statistics (mean, variance, histogram)?
    - Variation on bag of words problem: loses spatial information present in voxels.
  – Standard approach uses convolutions to represent neighbourhood.

Representing Neighbourhoods with Convolutions

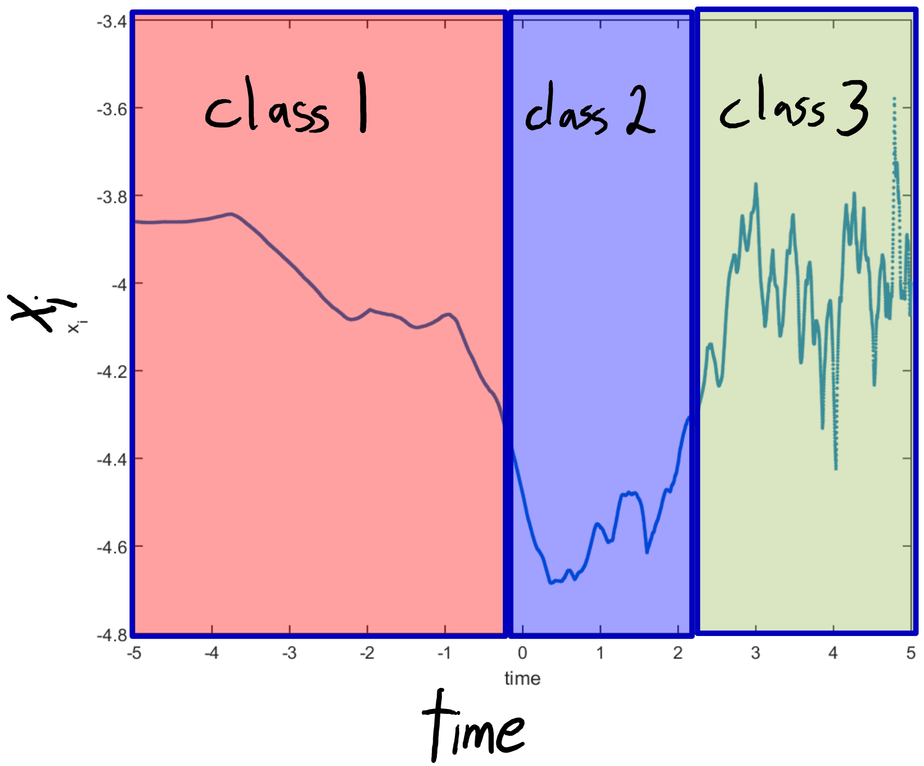
- Consider a 1D dataset:
  - Want to classify each time into y_i in {1,2,3}.
  - Example: speech data.

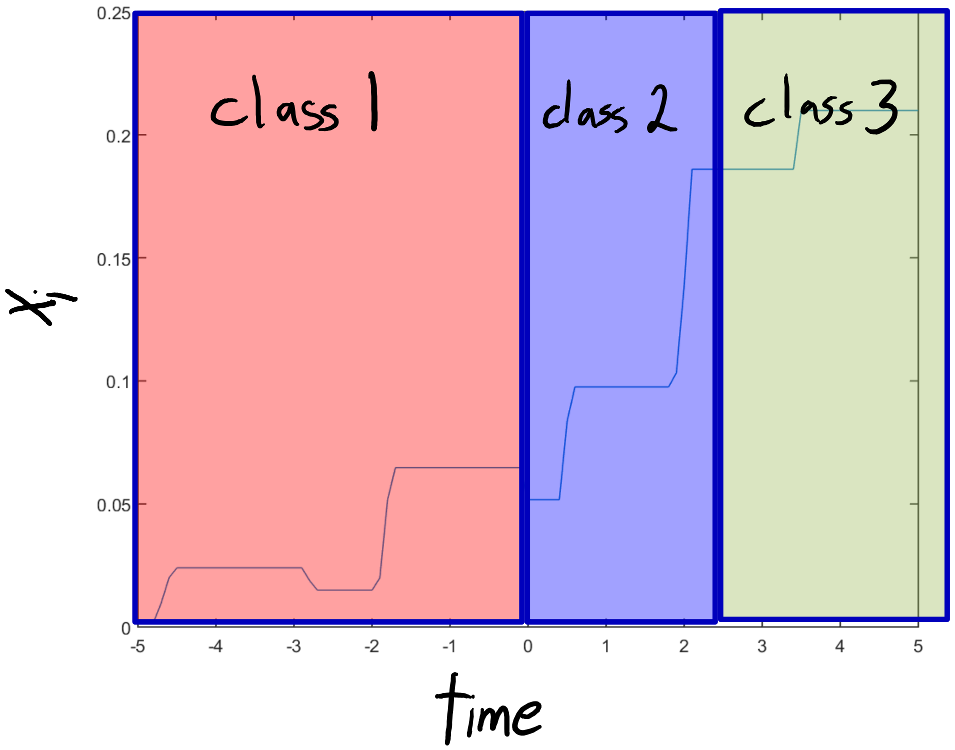


- Easy to distinguish class 2 from the other classes (x_i are smaller).
- Harder to distinguish between class 1 and class 3 (similar x_i range).
  - But convolutions can represent that class 3 is in "spiky" region.

Representing Neighbourhoods with Convolutions

- Original features (left) and features from convolutions (right):





- Easy to distinguish the 3 classes with these 2 features.

1D Convolution (notation is specific to this lecture)

- 1D convolution input:
  - Signal 'x' which is a vector length 'n'.
    - Indexed by i=1,2,…,n.
  - Filter 'w' which is a vector of length '2m+1':
    - Indexed by i=-m,-m+1,…-2,0,1,2,…,m-1,m

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

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

- Output is a vector of length 'n' with elements:

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

  - You can think of this as centering w at z_i and taking a dot product.

1D Convolution

- 1D convolution example:
  - Signal:

$$x = [\underbrace{\overset{1}{0}\ \overset{2}{1}\ \overset{3}{1}\ \overset{4}{2}\ \overset{5}{3}\ \overset{6}{5}\ \overset{7}{8}\ \overset{8}{13}}_{n}]$$

  - Filter:

$$w = [\underbrace{\underset{w_{-2}}{0}\ \underset{w_{-1}}{-1}\ \underset{w_0}{2}\ \underset{w_1}{-1}\ \underset{w_2}{0}}]$$

  - Convolution:

$$z = [\underbrace{\overset{1}{}\ \overset{2}{}\ \overset{3}{}\ \overset{4}{0}\ \overset{5}{}\ \overset{6}{}\ \overset{7}{}\ \overset{8}{}}_{n}]$$

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: $x = [0 \;\; 1 \;\; 1 \;\; 2 \;\; \boxed{3} \;\; 5 \;\; 8 \;\; 13]$
    $\underbrace{1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 8}_{n}$
  - Filter: $w = [0 \;\; -1 \;\; 2 \;\; -1 \;\; 0]$
    $w_{-2} \;\; w_{-1} \;\; w_0 \;\; w_1 \;\; w_2$
  - Convolution:
    $z = [\;\; 0 \;\; \boxed{-1} \;\;]$
    $\underbrace{1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 8}_{n}$

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

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

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

$\hookrightarrow w = [\tfrac{1}{3}\ \ \tfrac{1}{3}\ \ \tfrac{1}{3}]$

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

Boundary Issue

- What can we 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 $z = [?\ \frac{2}{3}\ 1\frac{1}{3}\ 2\ 3\frac{1}{3}\ 5\frac{1}{3}\ 8\frac{2}{3}\ ?]$

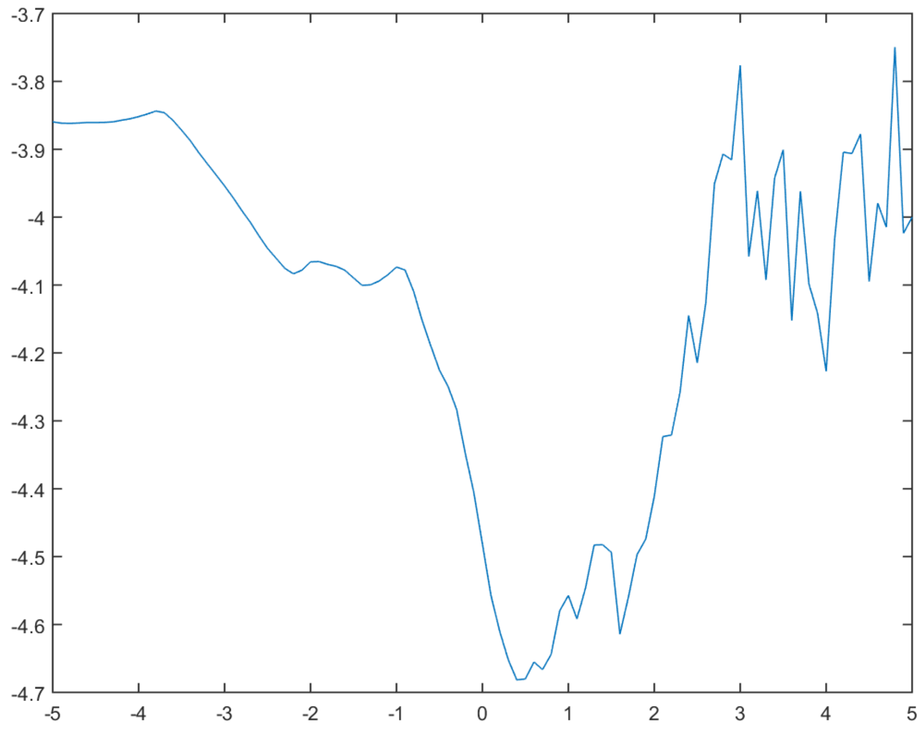
- Can assign values past the boundaries:
  - "Zero": $x = 0\ 0\ 0\ [0\ 1\ 1\ 2\ 3\ 5\ 8\ 13]\ 0\ 0\ 0$
  - "Replicate": $x = 0\ 0\ 0\ [0\ 1\ 1\ 2\ 3\ 5\ 8\ 13]\ 13\ 13\ 13$
  - "Mirror": $x = 2\ 1\ 1\ [0\ 1\ 1\ 2\ 3\ 5\ 8\ 13]\ 8\ 5\ 3$
- Or just ignore the "?" values and return a shorter vector:

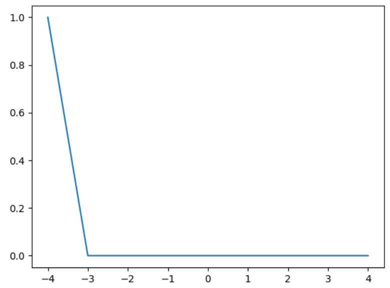
$$z = [\tfrac{2}{3}\ 1\tfrac{1}{3}\ 2\ 3\tfrac{1}{3}\ 5\tfrac{1}{3}\ 8\tfrac{2}{3}]$$

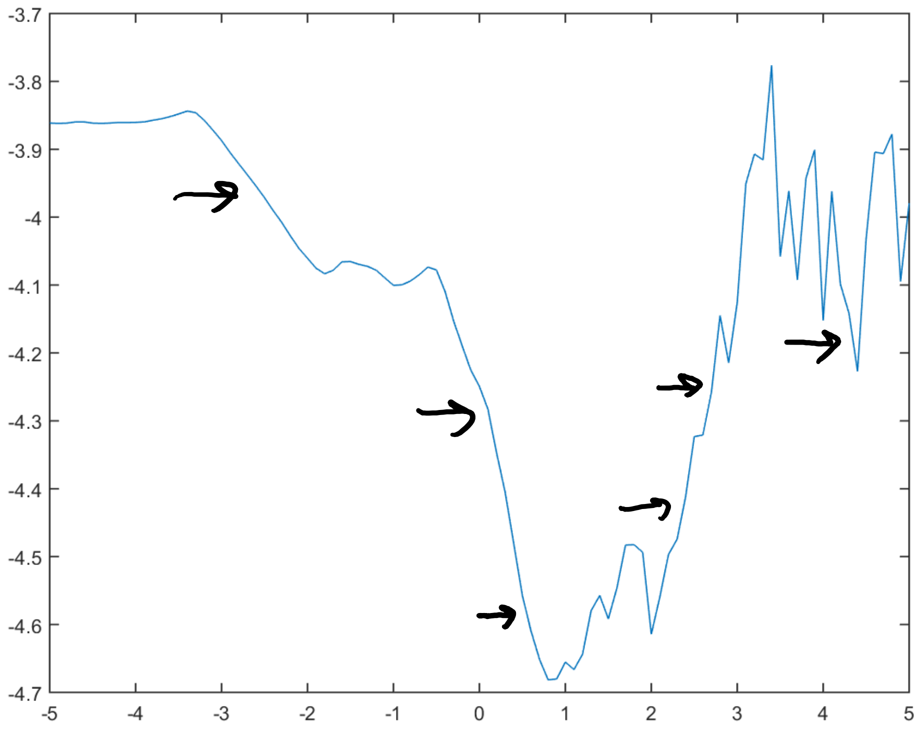
1D Convolution Examples

- Translation convolution shift signal:

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



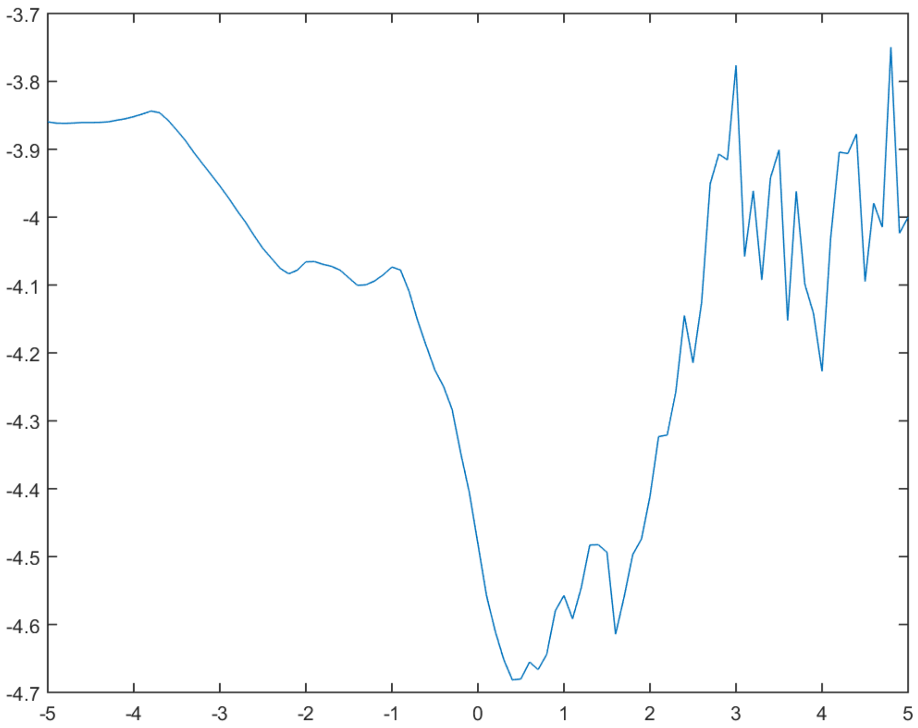


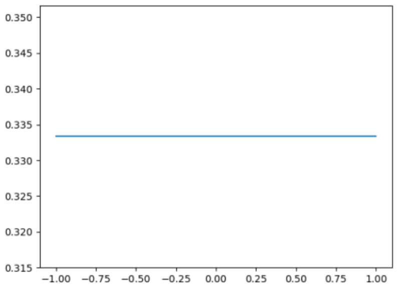


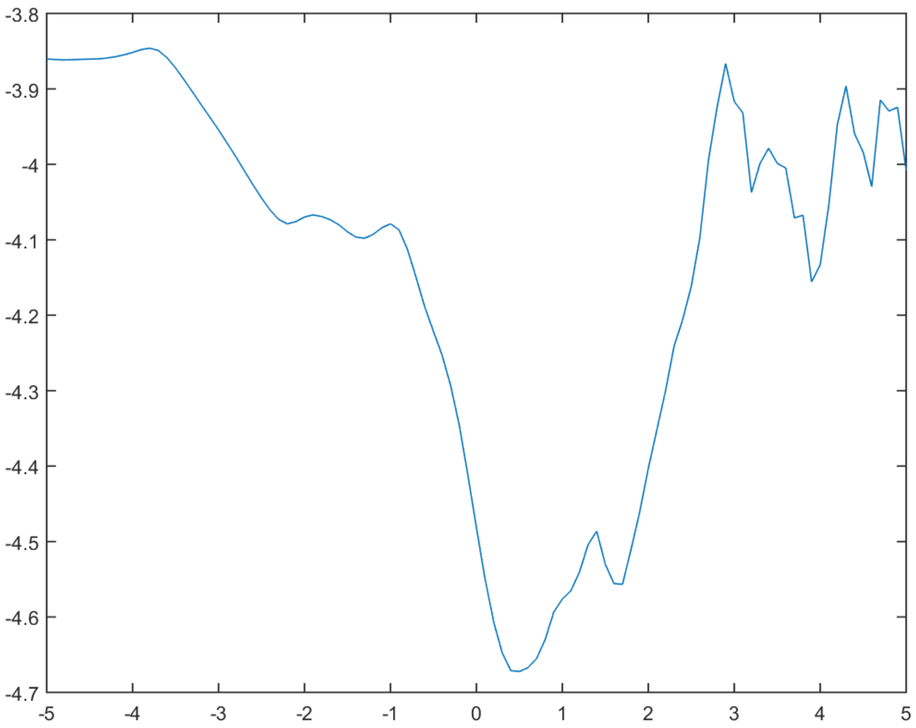
1D Convolution Examples

- Averaging convolution computes local mean:

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



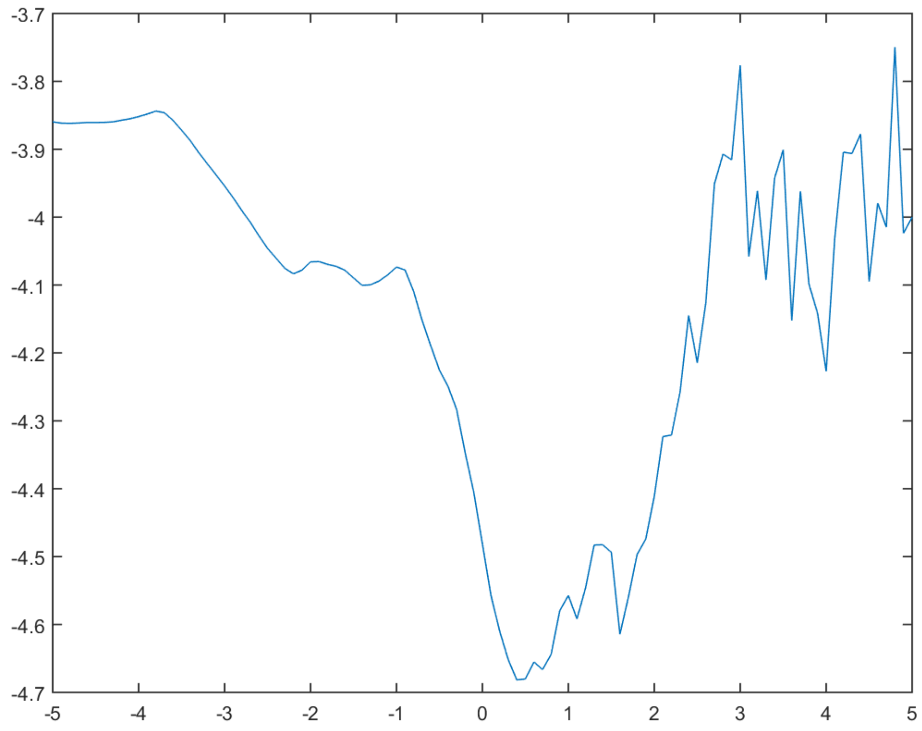


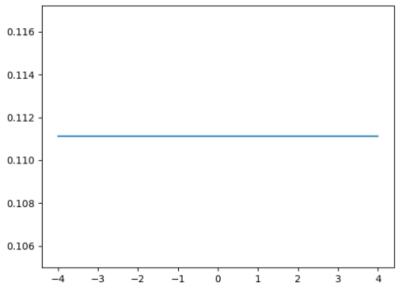


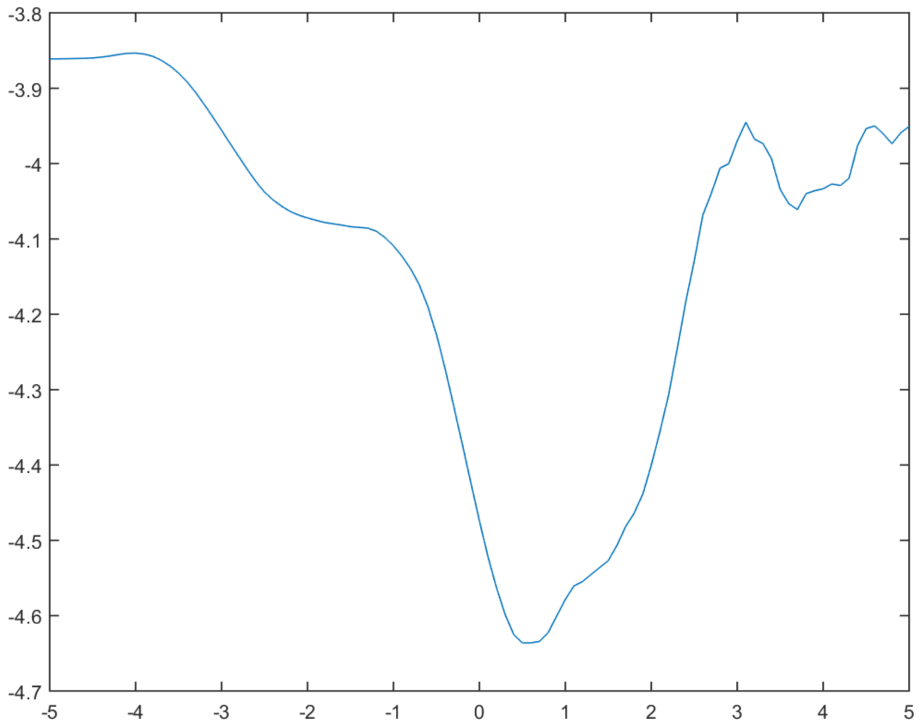
1D Convolution Examples

- Averaging over bigger window gives coarser view of signal:

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





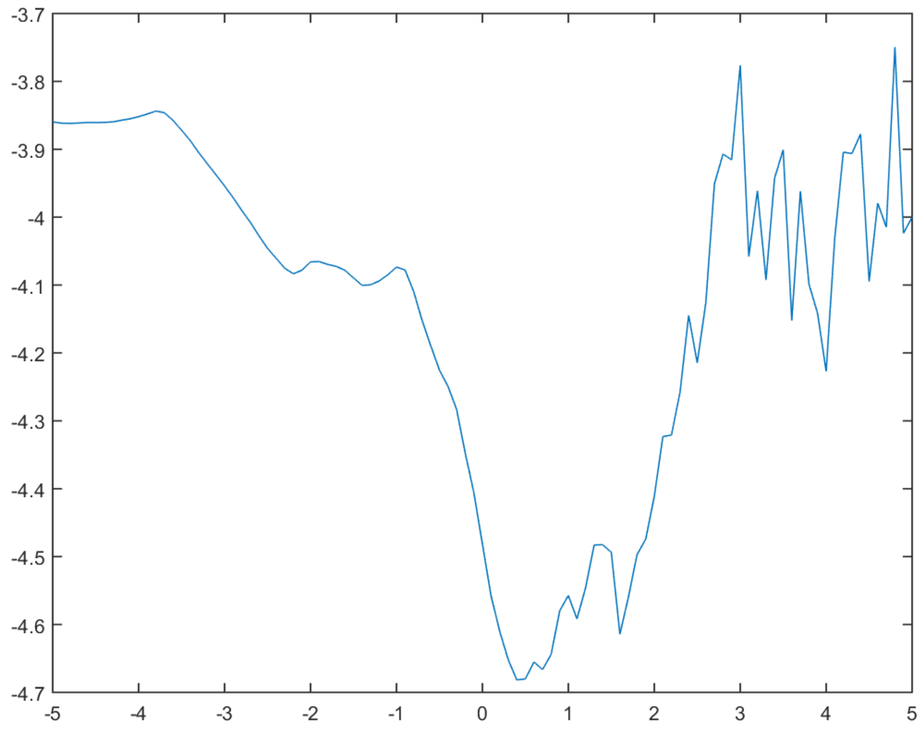


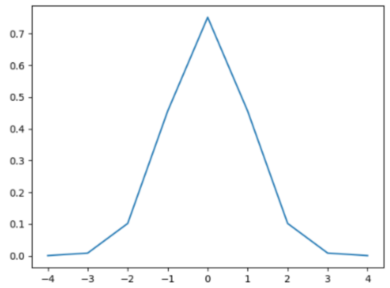
1D Convolution Examples

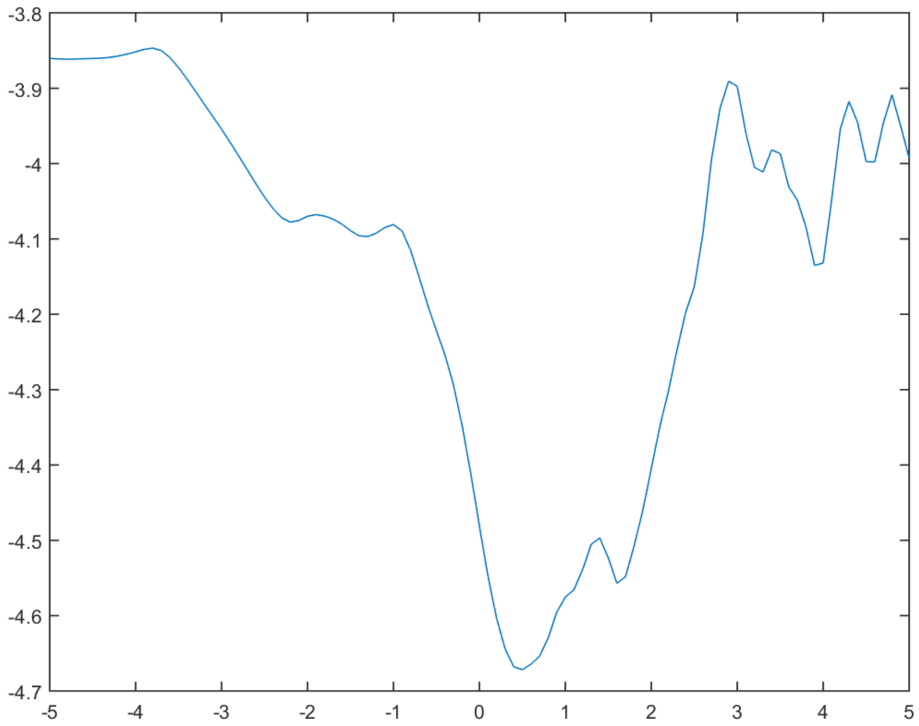
- Gaussian convolution blurs signal: $w_i \propto \exp\left(-\frac{i^2}{2\sigma^2}\right)$
  - Compared to averaging it's more smooth and maintains peaks better.

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

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



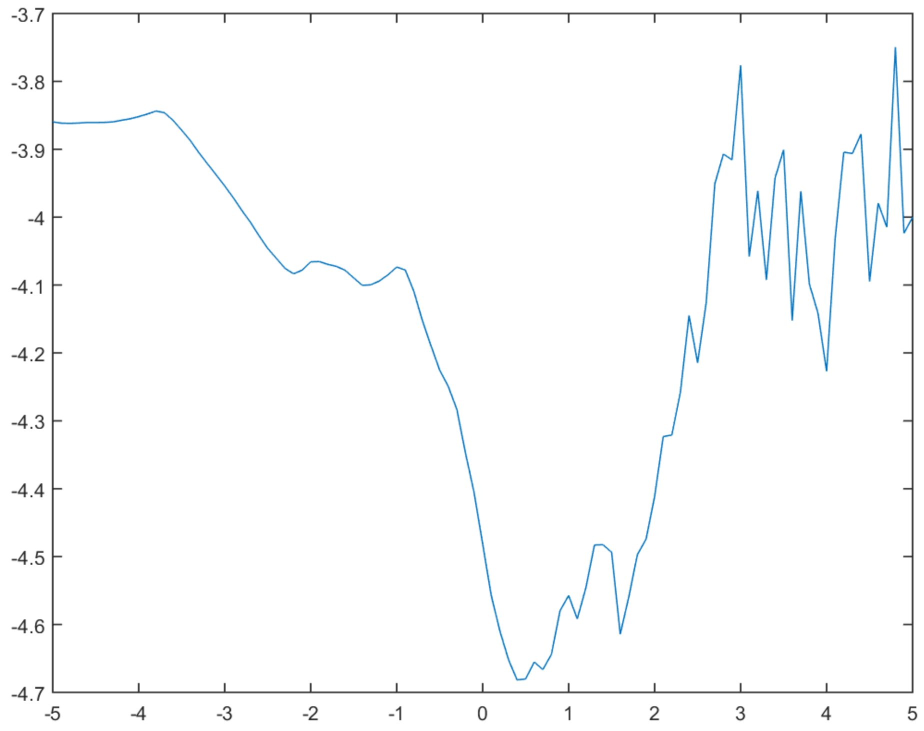


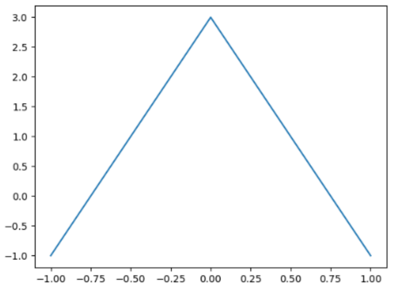


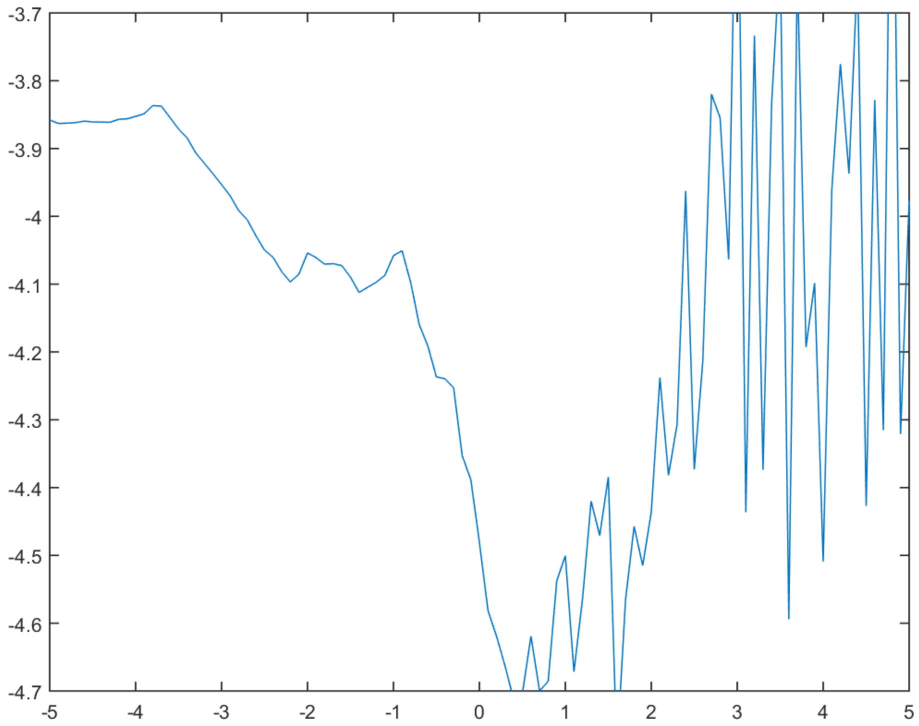
1D Convolution Examples

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

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



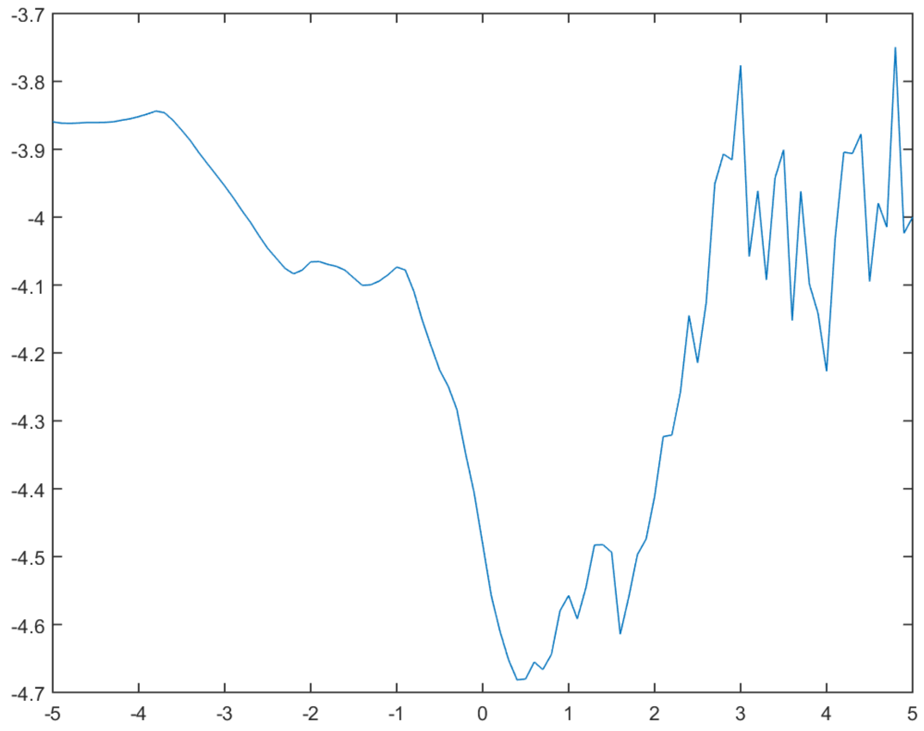


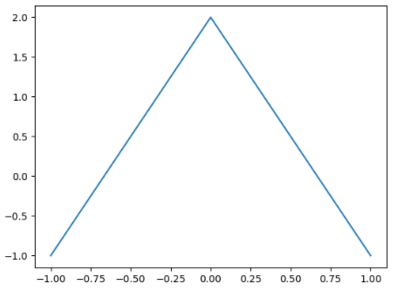


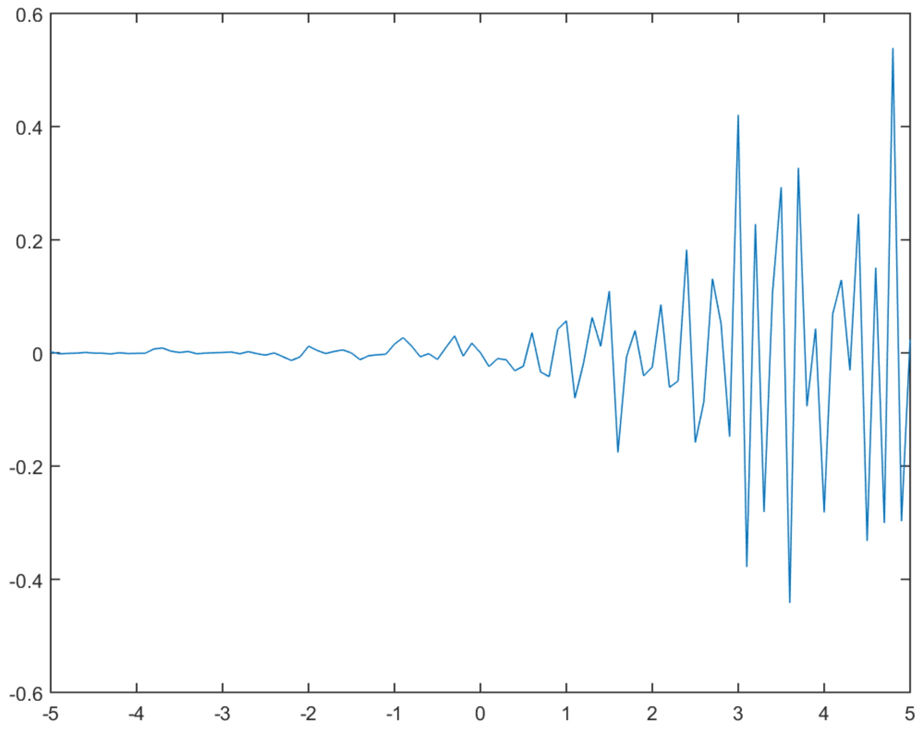
1D Convolution Examples

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

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

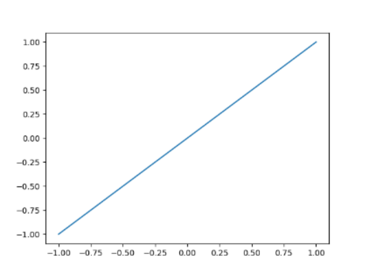




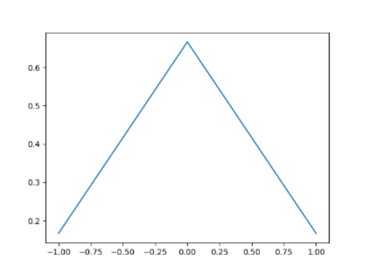


Digression: Derivatives and Integrals

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



- 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

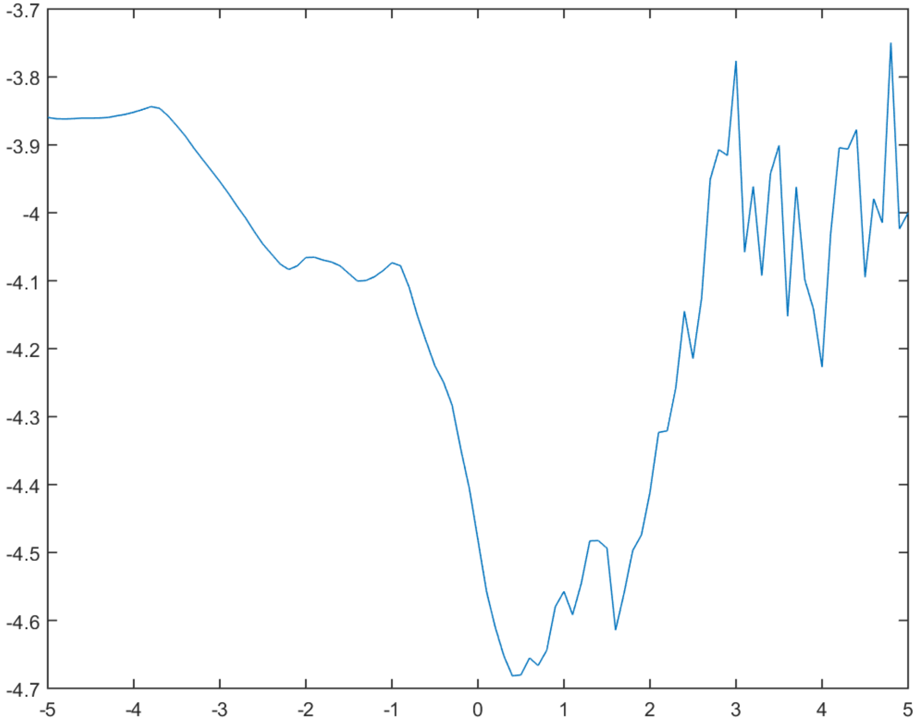
- Laplacian of Gaussian is a smoothed 2^{nd}-derivative approximation:

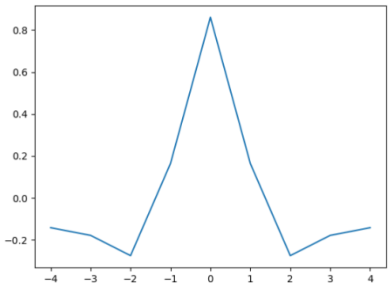
$$w_i = \left(1 - \frac{i^2}{2\sigma^2}\right)\exp\left(-\frac{i^2}{2\sigma^2}\right)$$

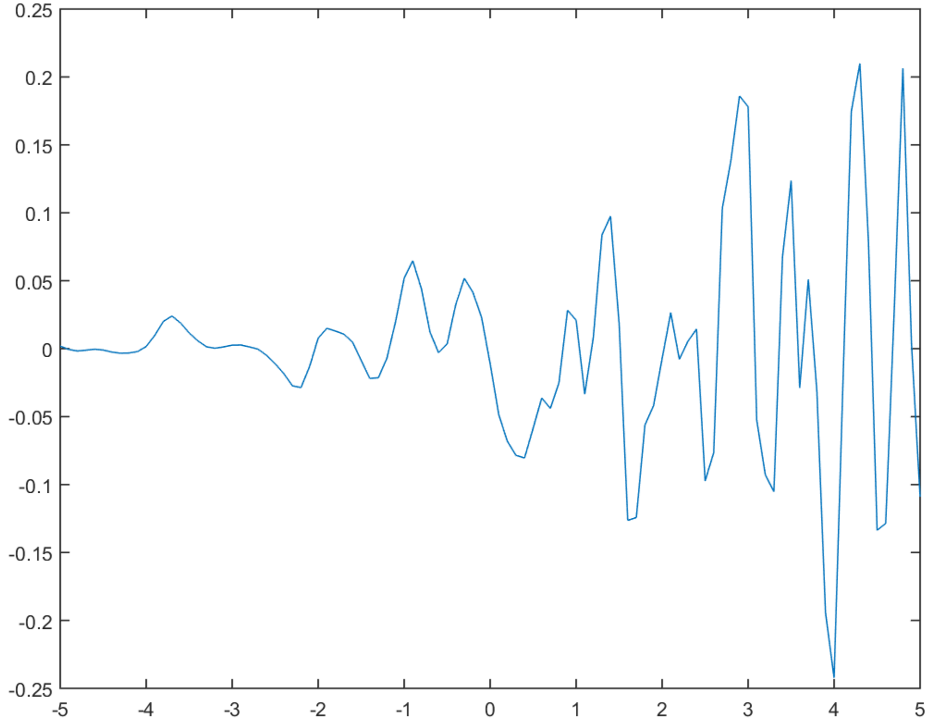
(then subtract mean)

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

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

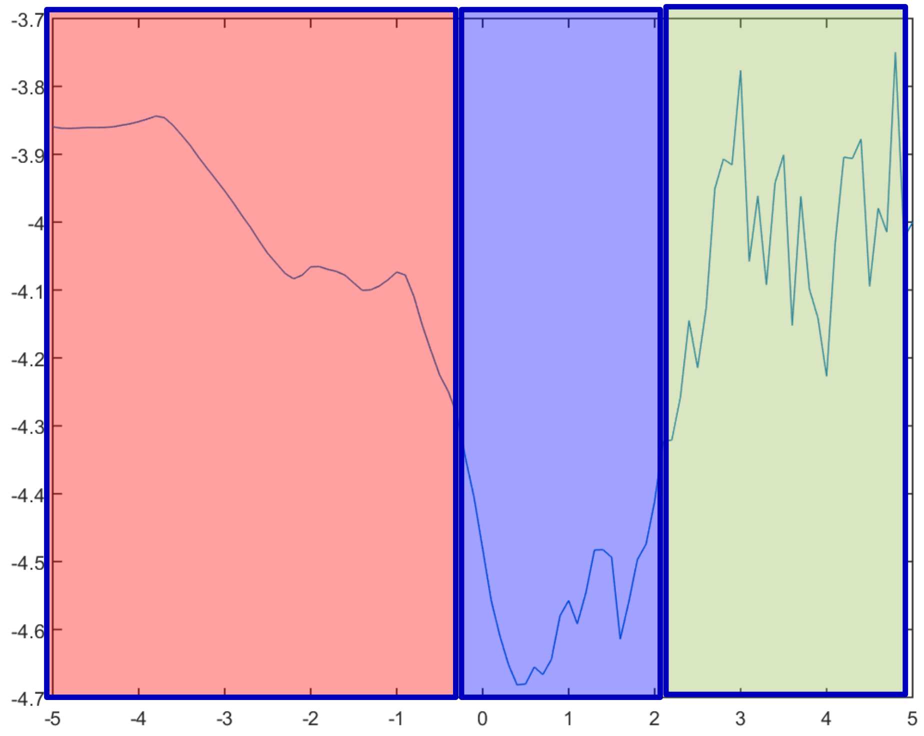




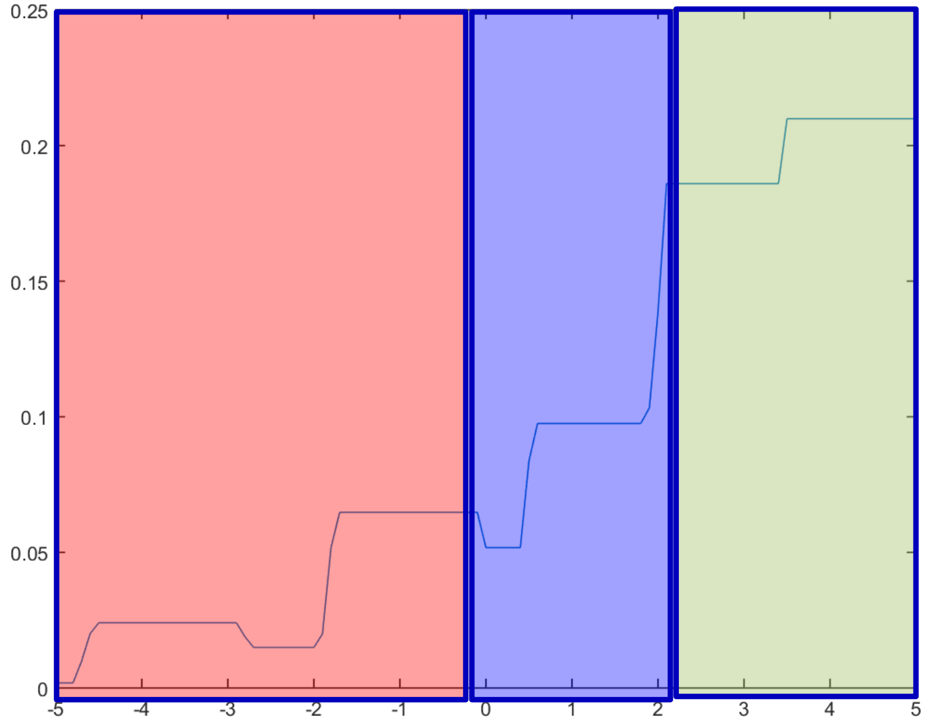


1D Convolution Examples

- We often use maximum over several convolutions as features:
  - We could take maximum of Laplacian of Gaussian over x_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:

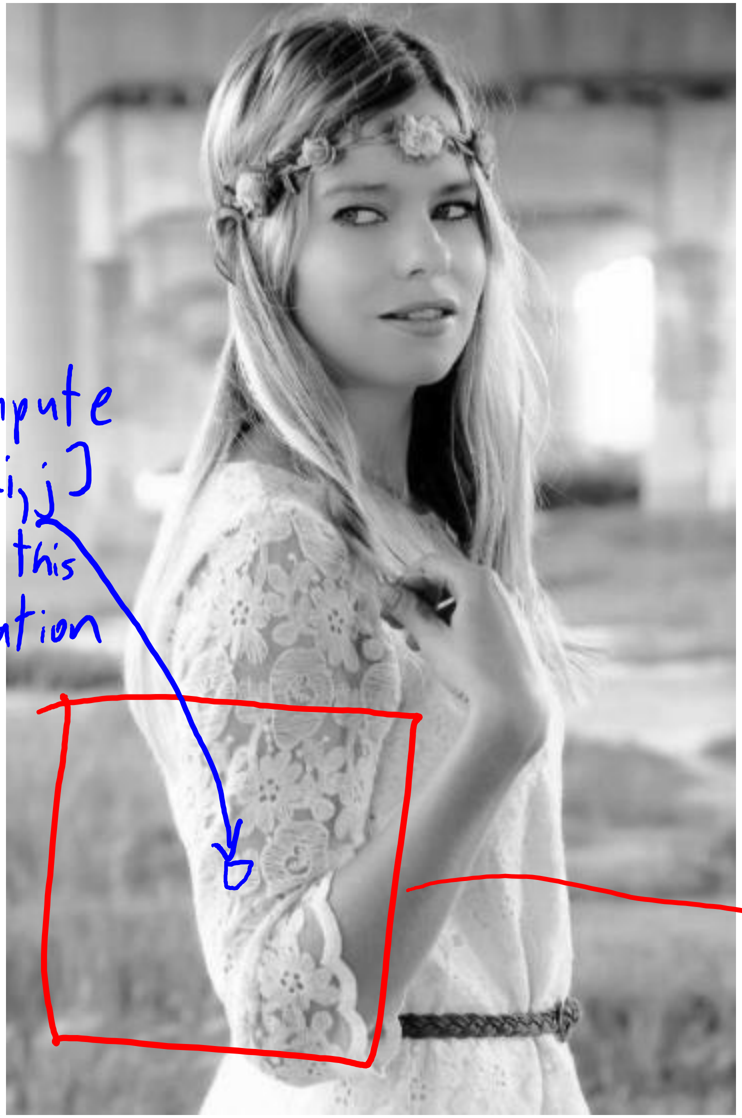
$$z[i_1, i_2] = \sum_{j_1=-m}^{m} \sum_{j_2=-m}^{m} w[j_1, j_2]\, x[i_1 + j_1, i_2 + j_2]$$

- 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



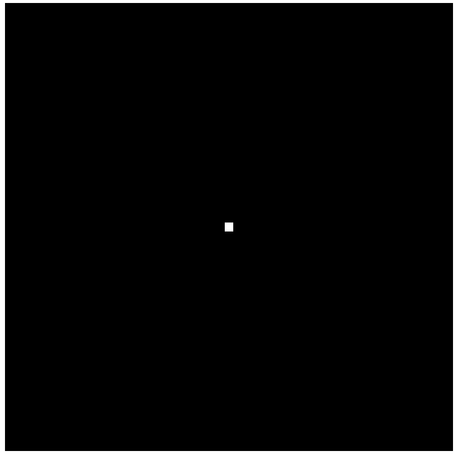
Compute z[i,j] for this location

Identity convolution:

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

w

$*$



=

z

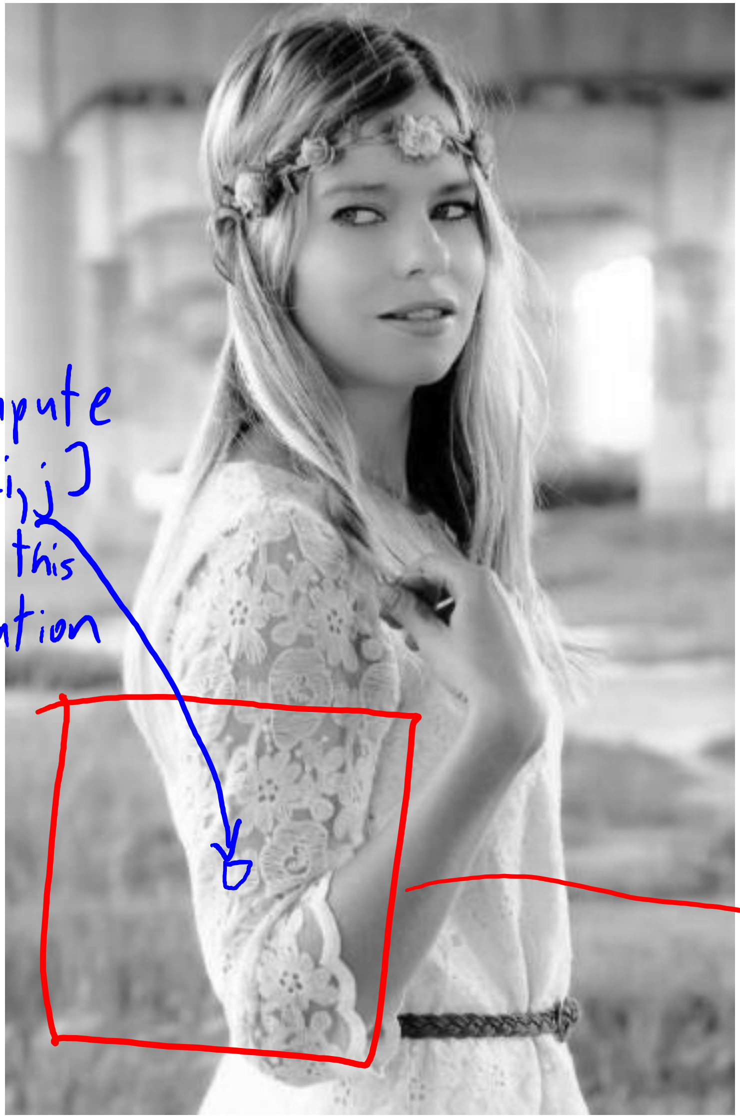
multiply element-wise and add up result to get

z[i,j]

Image Convolution Examples

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

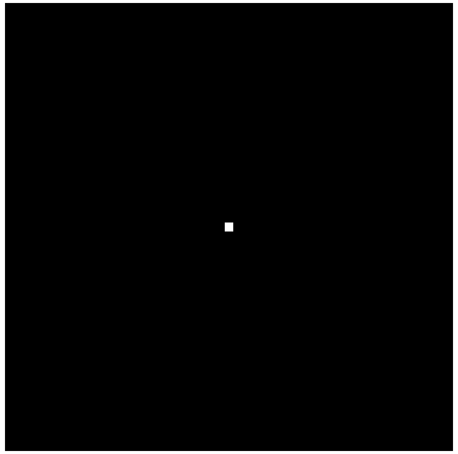
x



Compute
$z[i,j]$
for this
location

$*$

w



=

z

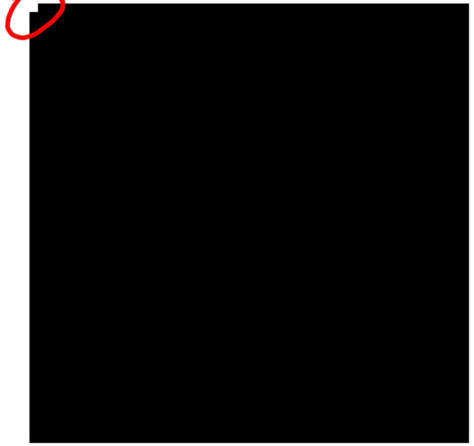
multiply element-wise
and add up result to get

$z[i,j]$

Image Convolution Examples

Translation Convolution:

*



Boundary: "zero"

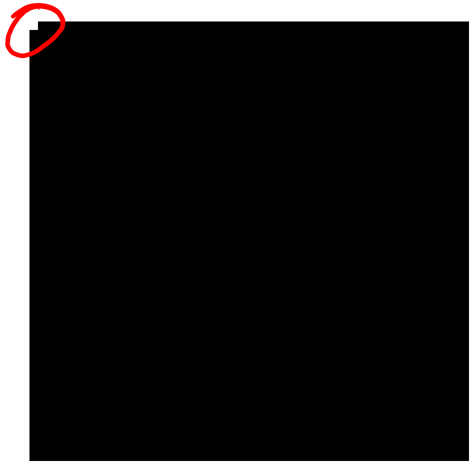
=

0

Image Convolution Examples

Translation Convolution:

*



=

Boundary: "replicate"

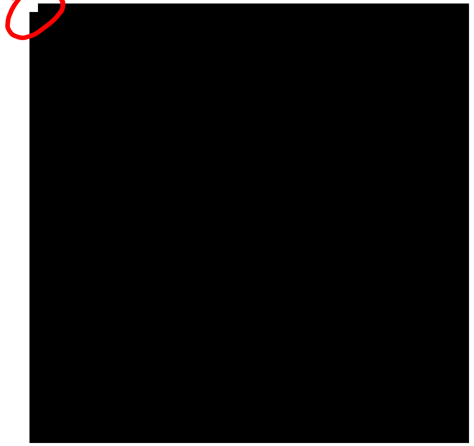
repeats

repeats

Image Convolution Examples

Translation Convolution:

*



=

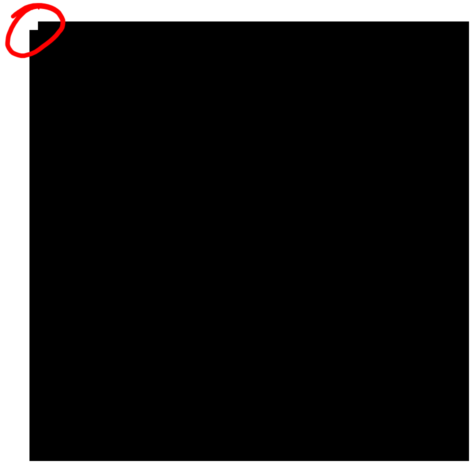
Boundary: "mirror"

flips

Image Convolution Examples

Translation Convolution:

$*$



$=$

Boundary: "ignore"

Image Convolution Examples

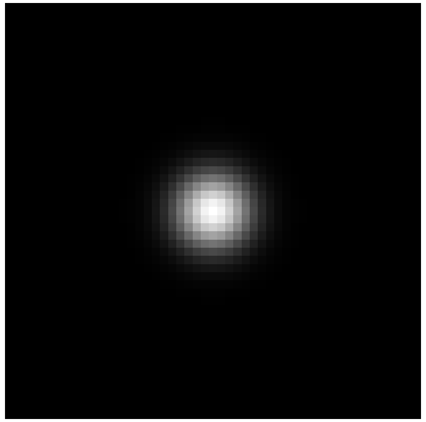
Average convolution:

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

Image Convolution Examples

Gaussian Convolution:

$*$



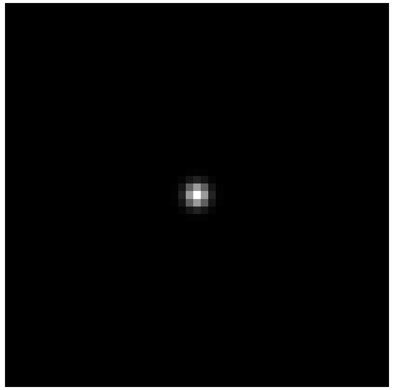
$=$

blurs image to represent average (smoothing)

Image Convolution Examples

Gaussian Convolution:

$*$



(smaller variance)

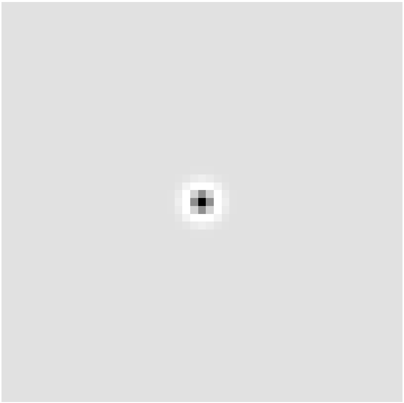
$=$

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

Image Convolution Examples

Laplacian of Gaussian

$*$



$=$

"signed" image (gray is 0)

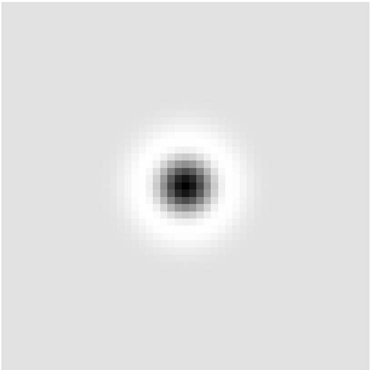
"How much does it look like a black dot surrounded by white?"

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:

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

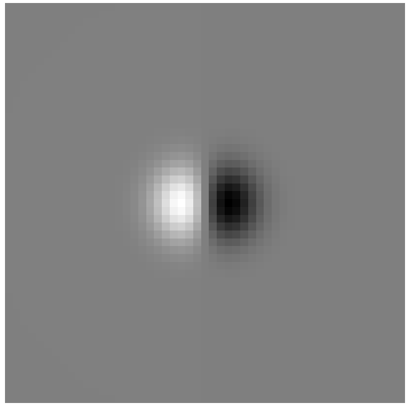
Many Photoshop effects are just convolutions.

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

Image Convolution Examples

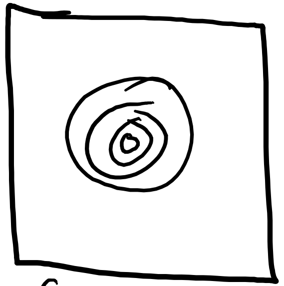
Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$



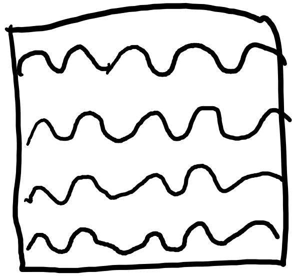
$=$

||

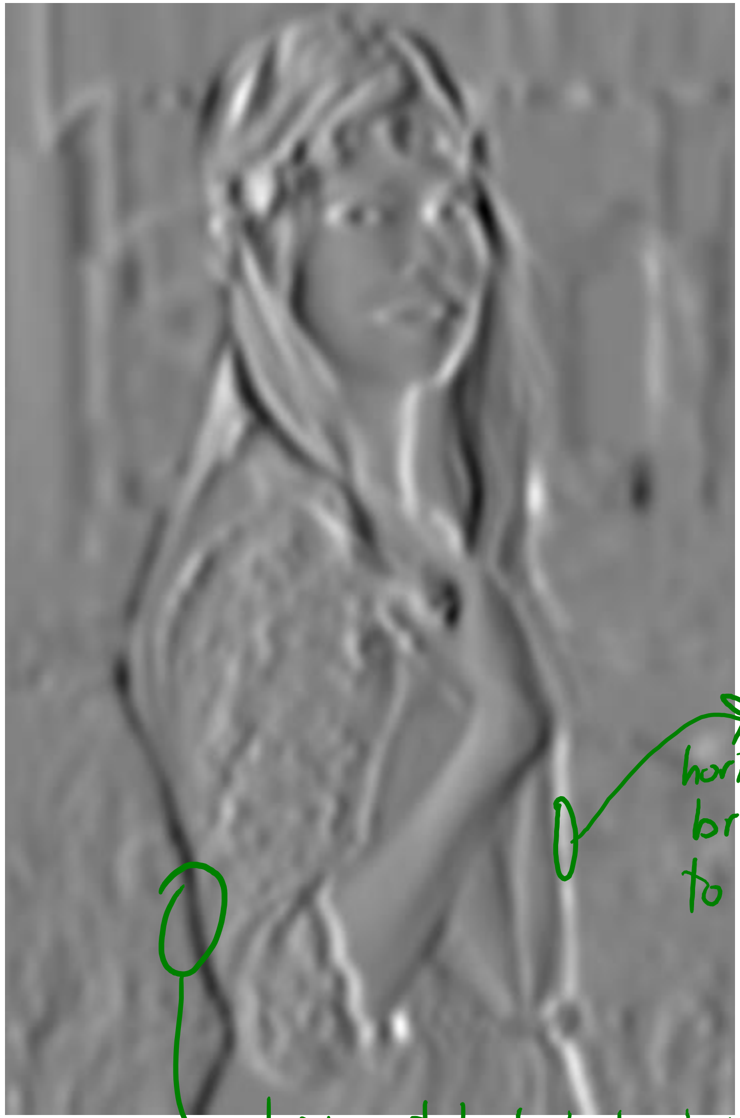


Gaussian

$.*$



Parallel Sine functions



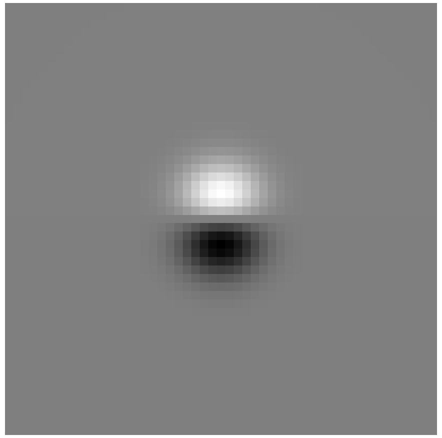
horizontal bright to dark

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.

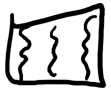
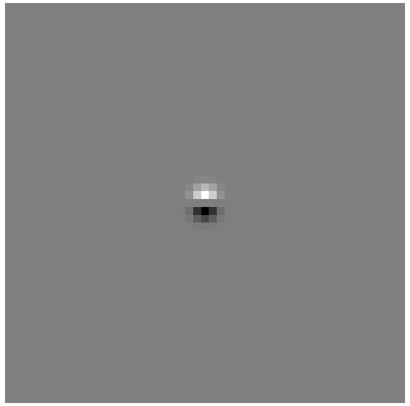


Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$



(smaller variance)

$=$

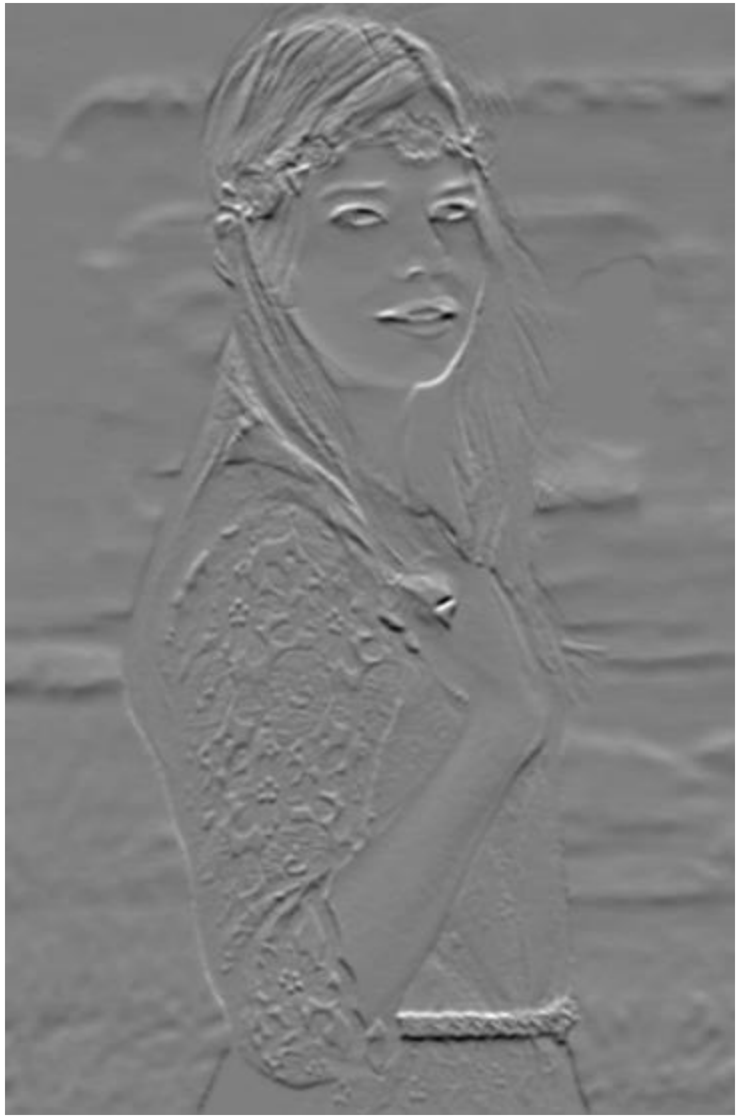
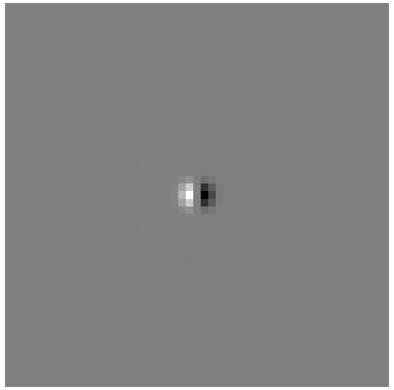


Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$  =

(smaller variance)

Vertical orientation

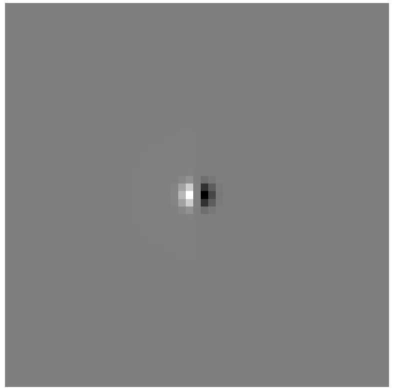
– Can obtain other orientations by rotating.

– May be similar to effect of V1 "simple cells."

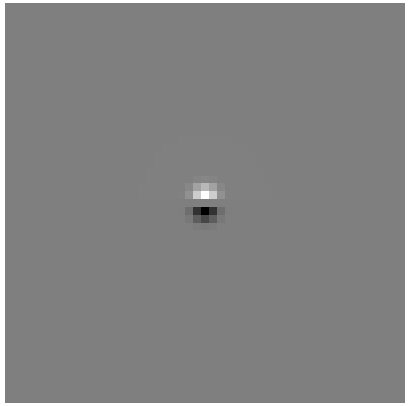
Image Convolution Examples

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

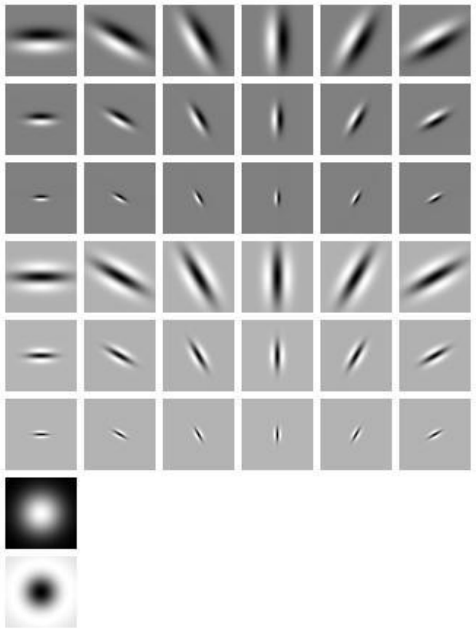
Sharpen the blue channel.

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

1D Convolution as Matrix Multiplication

- Each element of a convolution is an inner product:

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

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

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

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

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

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

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

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 convolutions (to make it smaller):
  1. Each row of W only applies to part of x_i.
  2. Use the same parameters between rows.

$$w_1 = [0 \quad 0 \quad 0 \; — \; w \; — \; 0 \quad 0 \quad 0]$$

$$w_2 = [0 \; — \; w \; — \; 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

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

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 results of several convolutions.

1D example

distance between centers of convolution is called "stride"

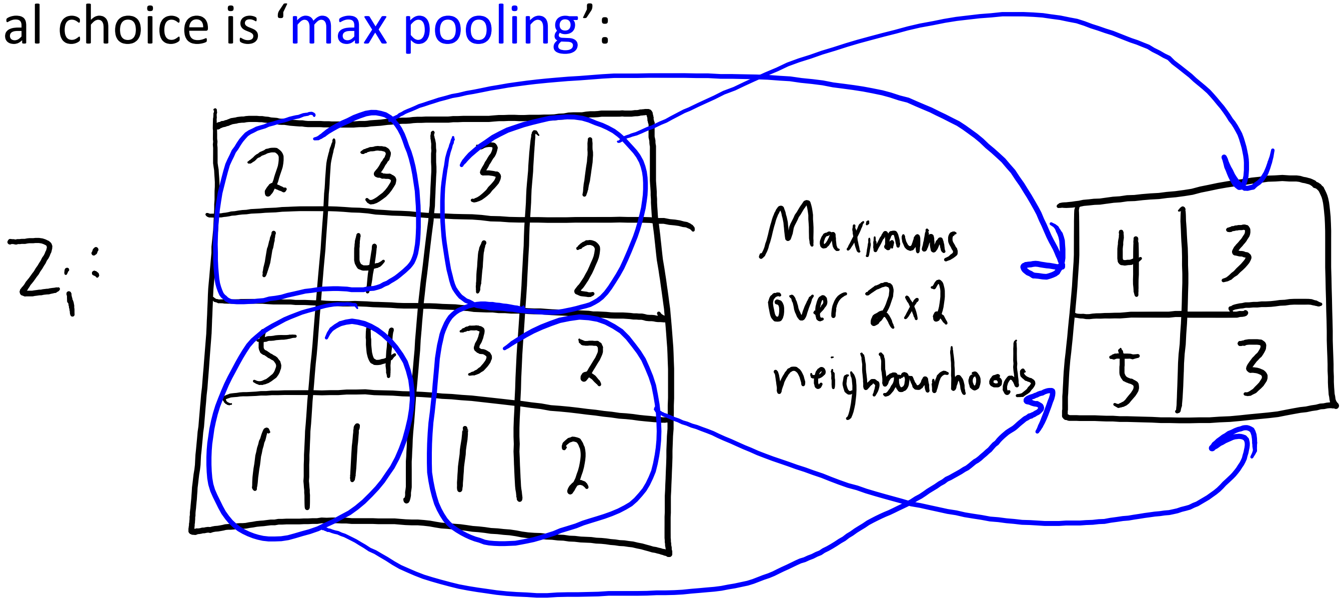
$$W^{(m)} = \begin{bmatrix} \text{———} & w_1^{(m)} & \text{———} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \text{———} & w_1^{(m)} & \text{———} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \text{———} & w_1^{(m)} & \text{———} \\ \text{———} & w_2^{(m)} & \text{———} & 0 & 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & \text{———} & w_2^{(m)} & \text{———} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \text{———} & w_2^{(m)} & \text{———} \end{bmatrix}$$

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

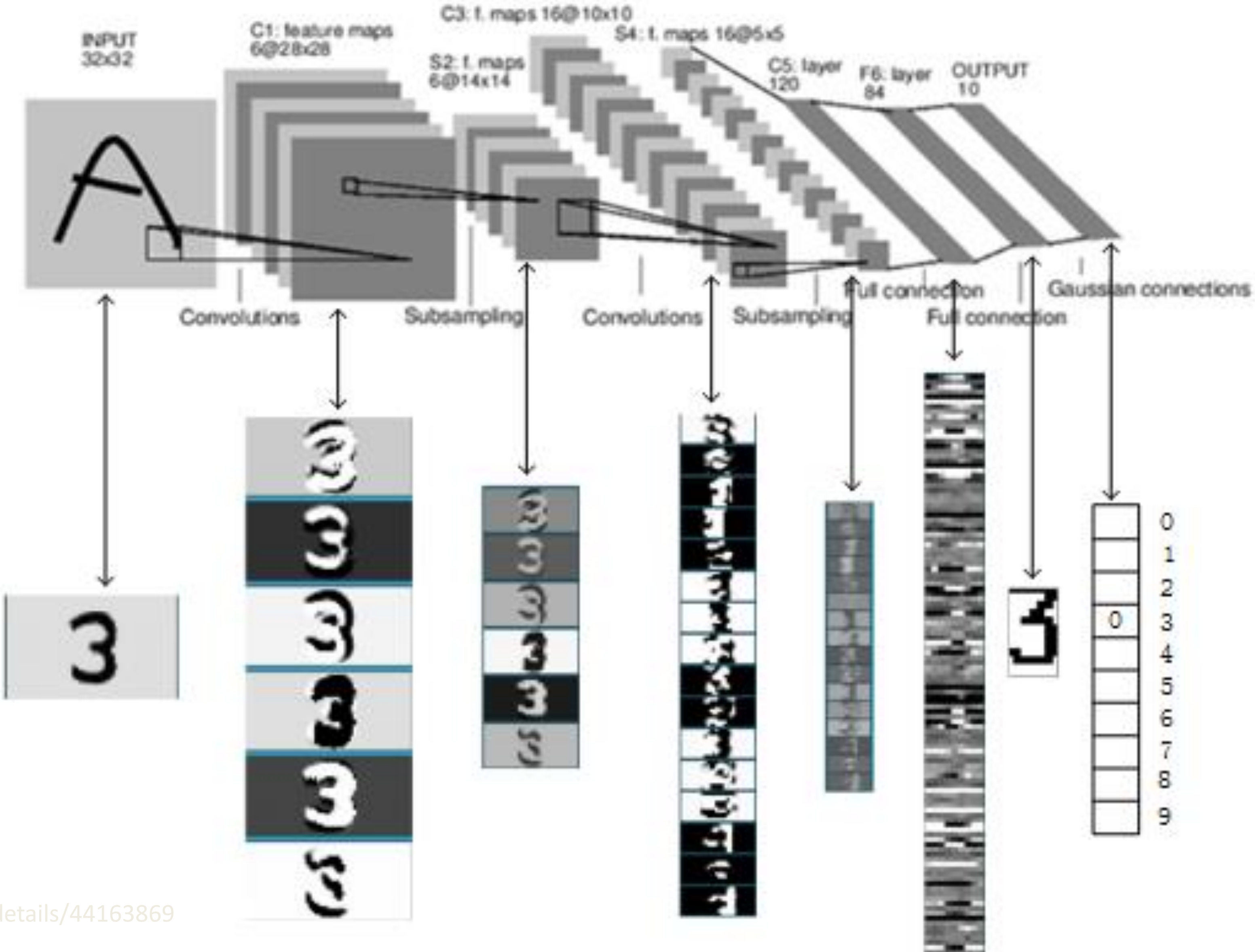
Sparse and small number of parameters

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 results of 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 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.

Number of parameters?

- Example with 1 conv/pool layer and 2 fully connected layers:
  - you start with a 28x28x3 RGB image
  - 32 filters each of size 5x5x3
  - 2x2 max pooling
  - fully connected layer with 128 hidden units
  - fully connected layer going to 10 output units for 10-class classification
- How many parameters does this model have?
  - the first convolutional layer has 5x5x3x32 (+32 bias).
  - this results in images of size 24x24 (this depends on how you handle convolutions at boundaries).
  - After 2x2 max pooling they are 12x12.
  - When we flatten this representation, we get 12x12x32 activations. This gives us 12x12x32x128 (+128 bias).
  - Finally we have a dense layer with 128x10 (+10 bias) parameters.
  - The grand total is 5x5x32x3 + 12x12x32x128 + 128x10 + 32 + 128 + 10 = 2400 + 589824 + 1280 + 170 = 593674.
- Most of the parameters come from the dense layer in this case (non-sparse).
- This kind of calculation is tedious but it’s a good way to understand the details.

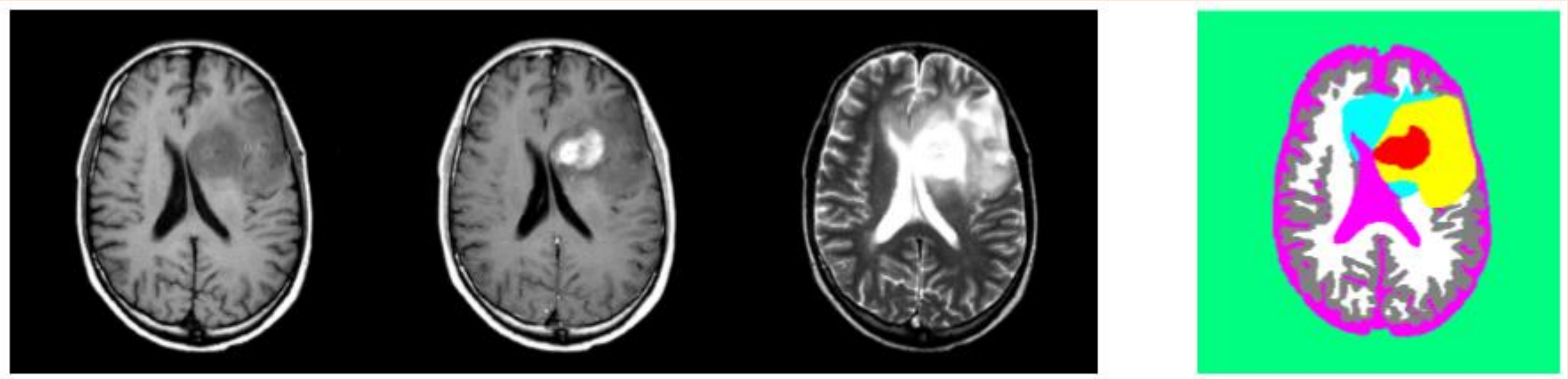
FFT implementation of convolution

- 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

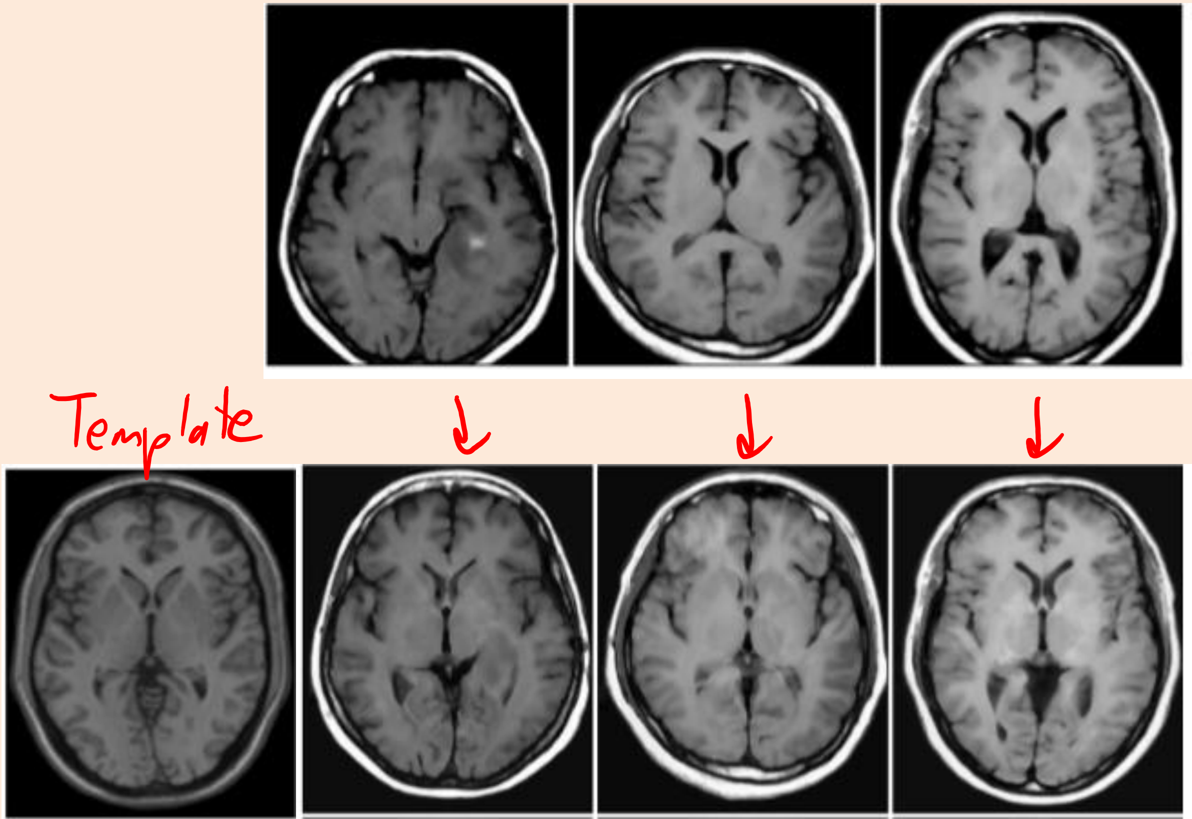
- Should we use the image coordinates?
  - E.g., the pixel is at location (124, 78) in the image.



- 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".

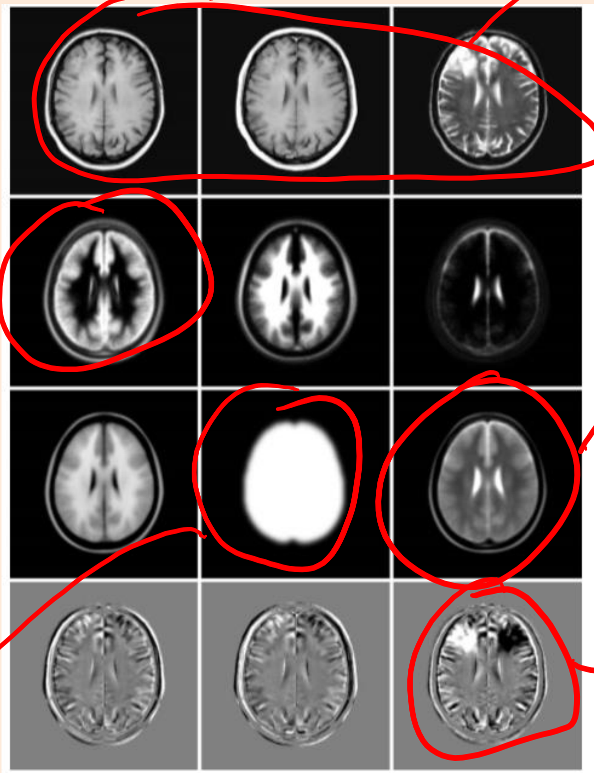


(Look different because we're showing middle slice and alignment is in 3D.)

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.

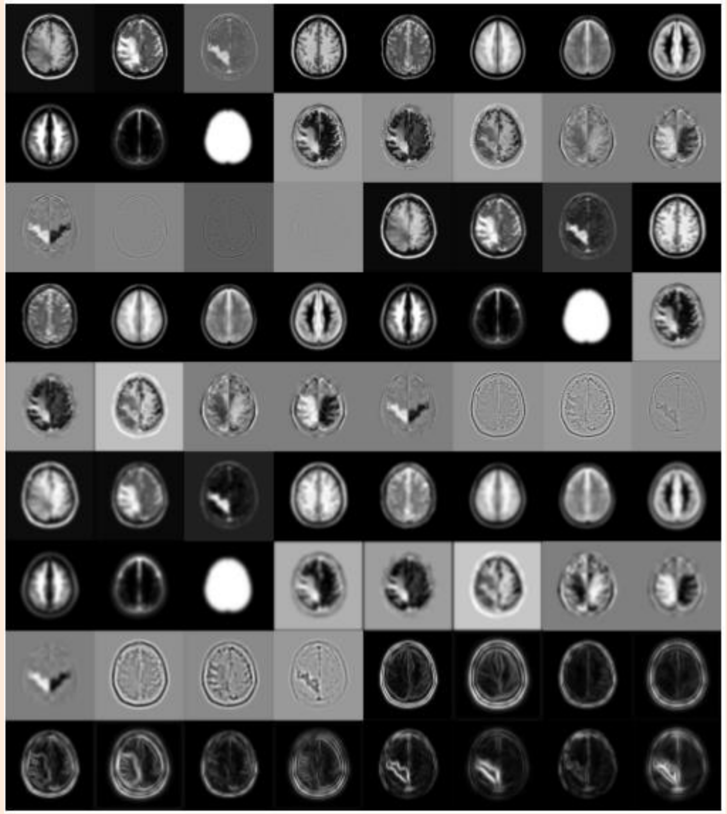
Actual pixel value of template image at this location.

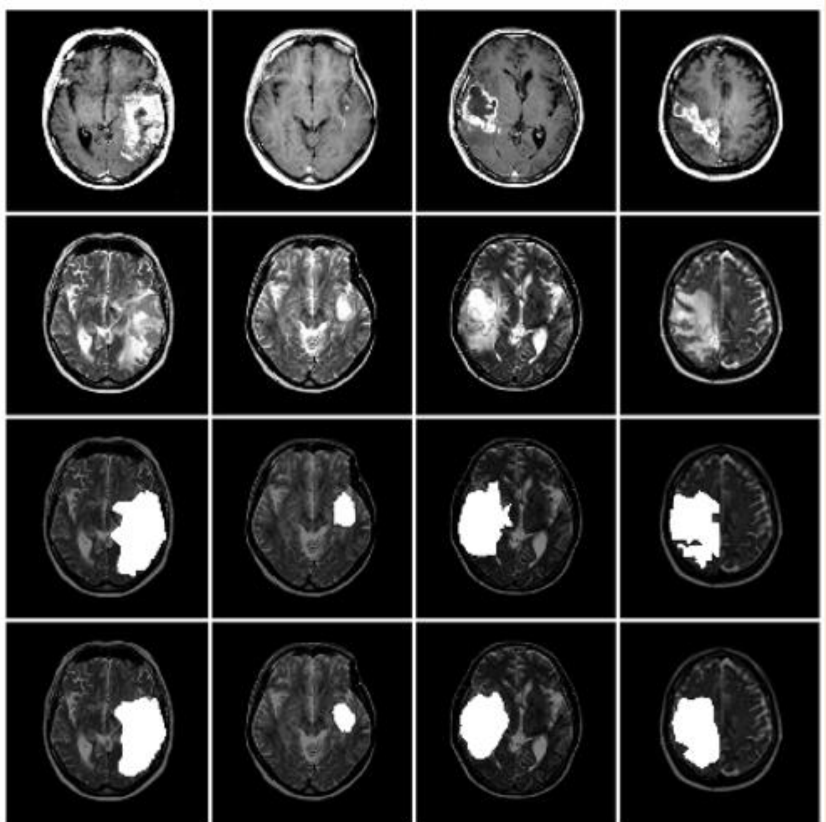
Left-right symmetry difference.

Probability of being brain pixel.

Motivation: Automatic Brain Tumor Segmentation

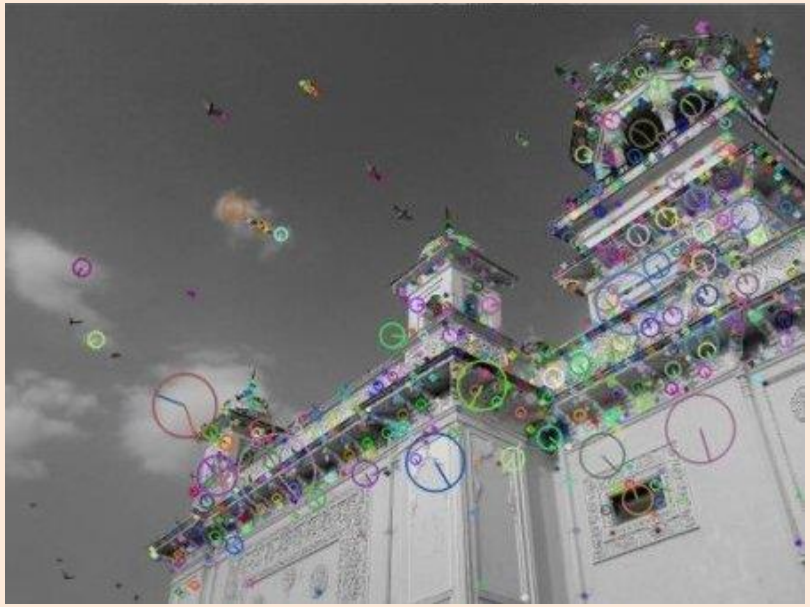
- Final features for brain tumour segmentation:
  - MR8 filter bank applied to original T1, T2, and T1 “contrast” – T1 “original”.
  - Gaussian convolution with 3 variances of alignment-based features.





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