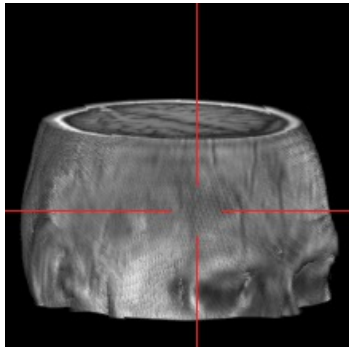
CPSC 340: Machine Learning and Data Mining

Convolutions

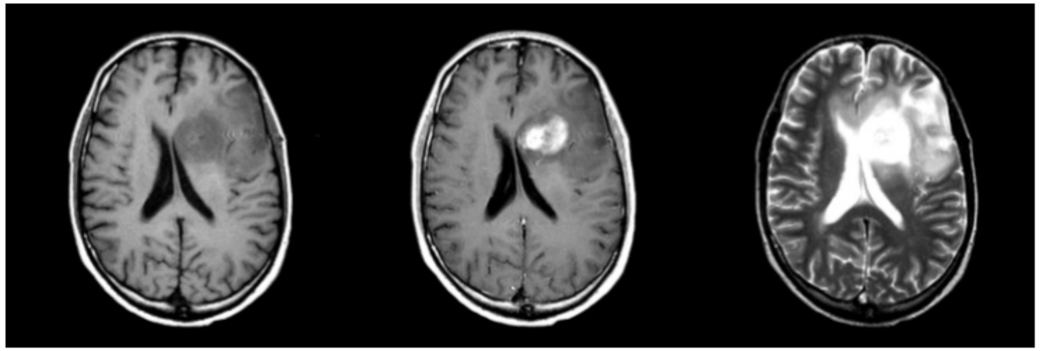
Spring 2022 (2021W2)

Motivation: Automatic Brain Tumor Segmentation

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



Input:

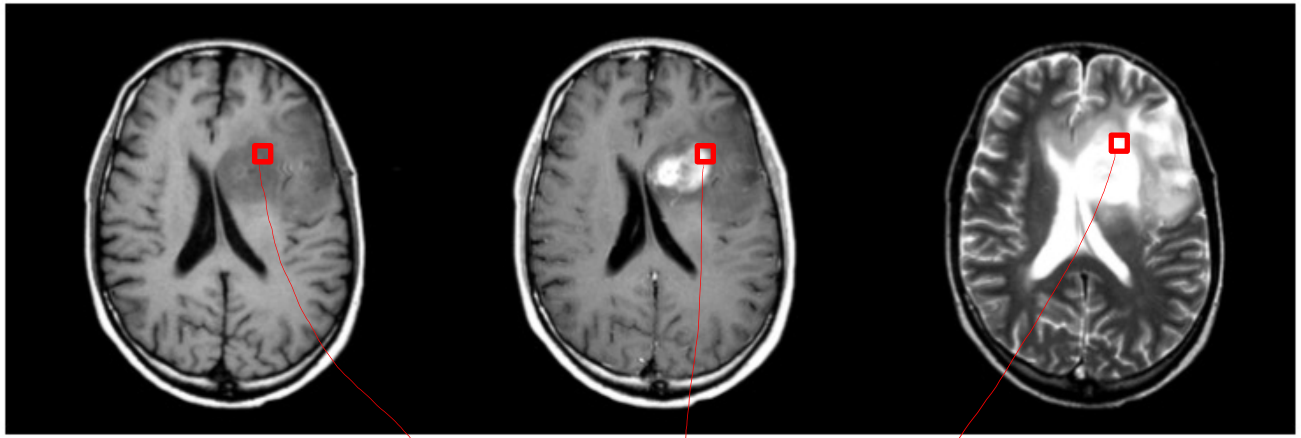


Output:

- 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:
  - Standard representation of image: each pixel gets "intensity" between 0 and 255.

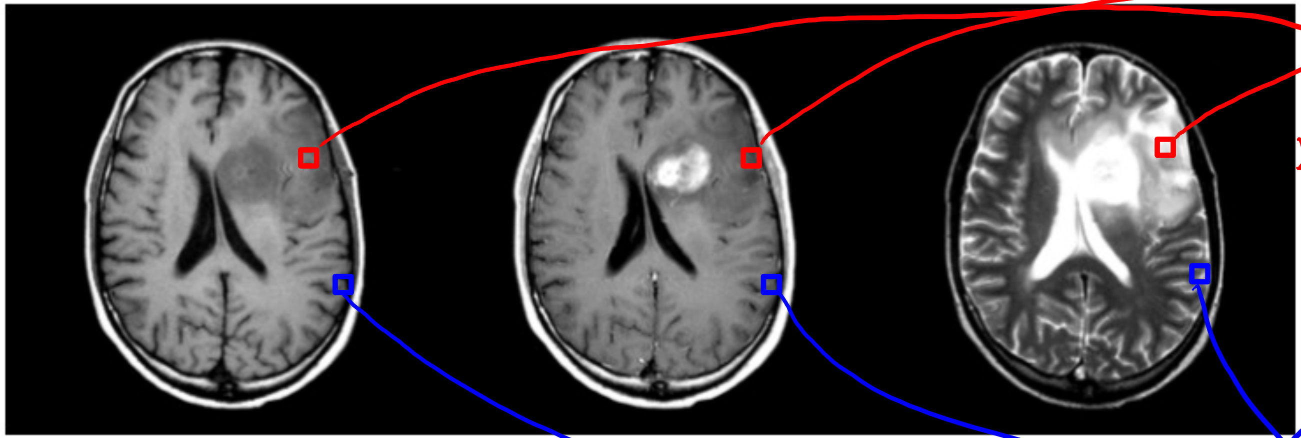


$x_i = (98, 187, 246)$ $y_i = \text{"tumour"}$

- 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 pixel intensity values are almost meaningless:
  - The same x_i could lead to different y_i.



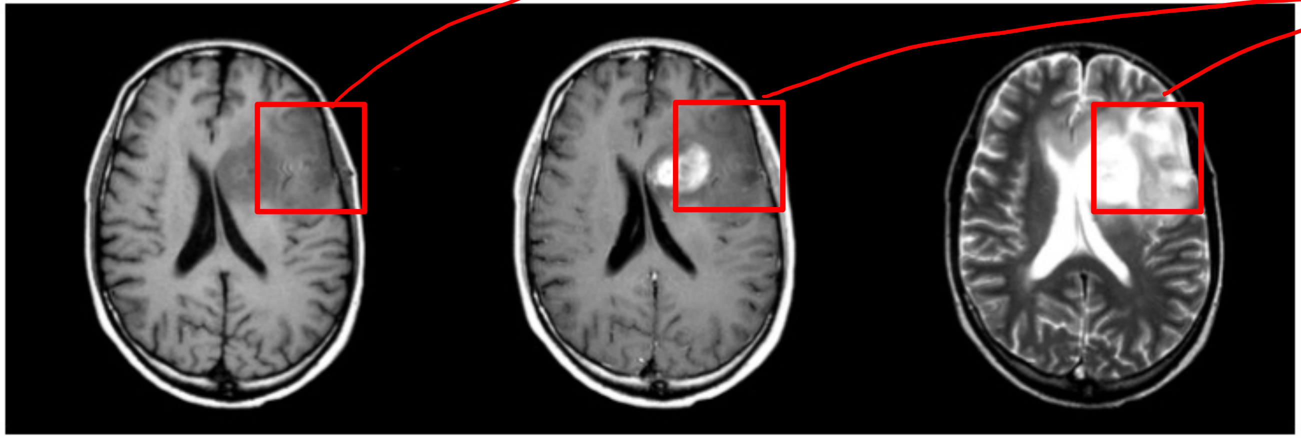
$x_i = (87, 54, 173)$ $y_i =$ "tumour"

$x_i = (87, 54, 173)$ $y_i =$ "normal"

- 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

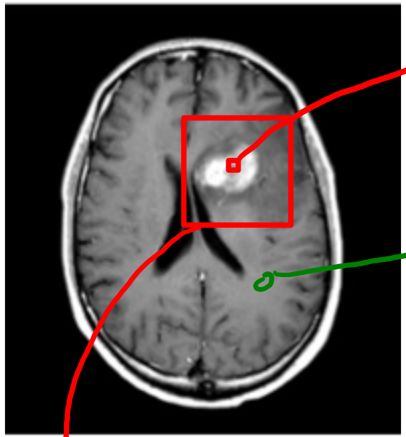
- We need to represent the "context" of the pixel (what is around it).



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

  - Include all the values of neighbouring pixels as extra features?
    - Run into coupon collection problems: 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.

Example: Measuring "brightness" of an Area



- This pixel is in a "bright" area of the image, which reflects "bleeding" of tumour.
- But the actual numeric intensity value of the pixel is the same as in darker "gray matter" areas.

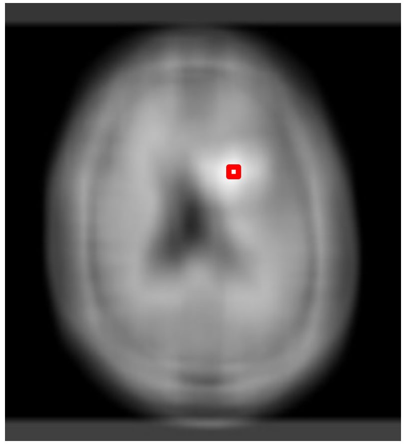
- I want a feature saying "this pixel is in a bright area of the image".
  - This will us help identify that it's a tumour pixel.

- How to measure brightness in area? Easy way: take average pixel intensity in "neighbourhood".

$$z = \frac{1}{|nei|} \sum_{k \in nei} x_k$$

new feature is average value in neighbourhood.

- Applying this "averaging" to every pixel gives a new image:

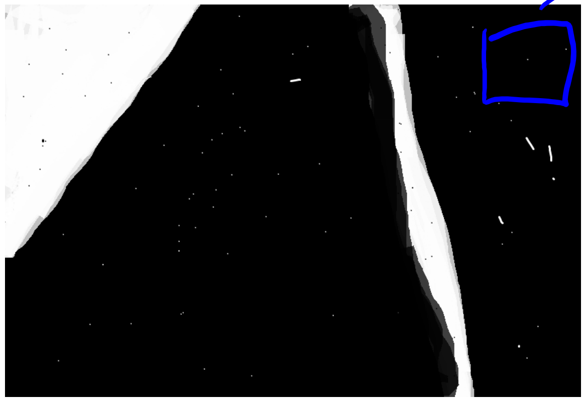


- We can use "pixel value in new image" as a new feature.
  - New feature helps identify if pixel is in a "bright" area.

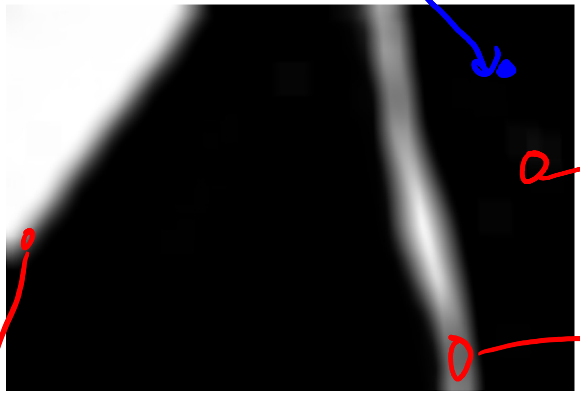
The annoying thing about squares

- “Take the average of a square window” loses spatial information.
- Example:

replace each pixel by the average value of "window"



Take average



Weird stuff

Pixels far away from "edge" become bright.

Noticeable features are "averaged out"

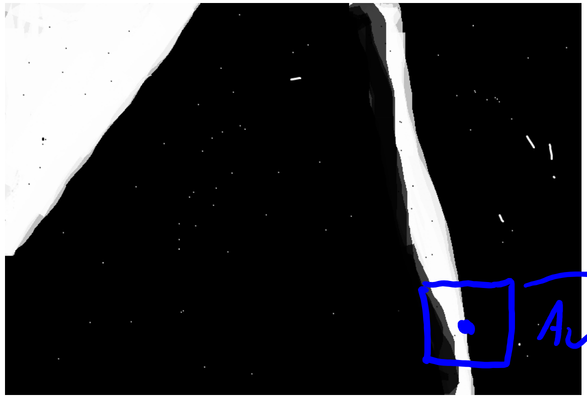
-average is higher, but location is lost.

"Dark because line is "surrounded" by dark areas

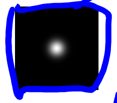
Fixing the "square" issues

- Consider instead "blurring" the image.
  - Gets rid of "local" noise, but better preserves spatial information.

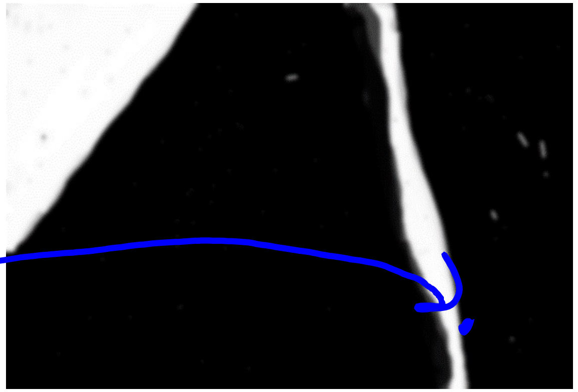
"blur" image



Average those pixels



with these weights



to get the value at this location

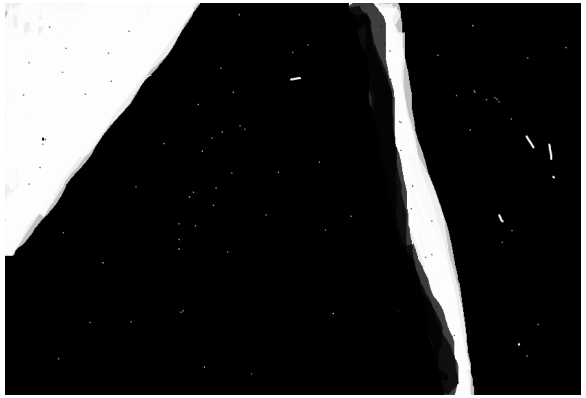
- How do you "blur"?
  - Take weighted average of window, putting more "weight" on "close" pixels:

$$z = \sum w_k x_k$$

kernel ↳ weight on pixel 'k' (averaging is special case where all pixels get equal weight)

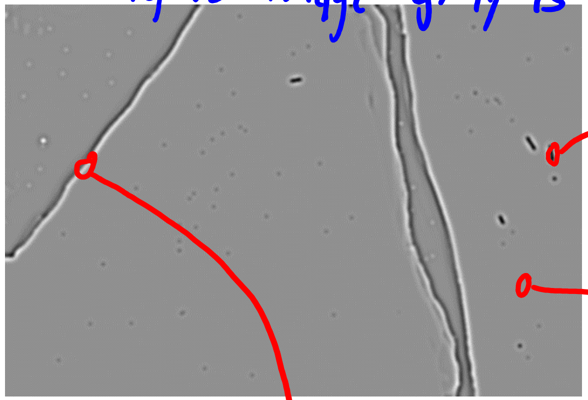
Fixing the “square” issues

- Another neat thing we can do: use negative weights.
  - These features can describe “differences” across space.



"Average" with positive and negative weights

"Signed" image: gray is 0.



"My neighbours are darker than me"

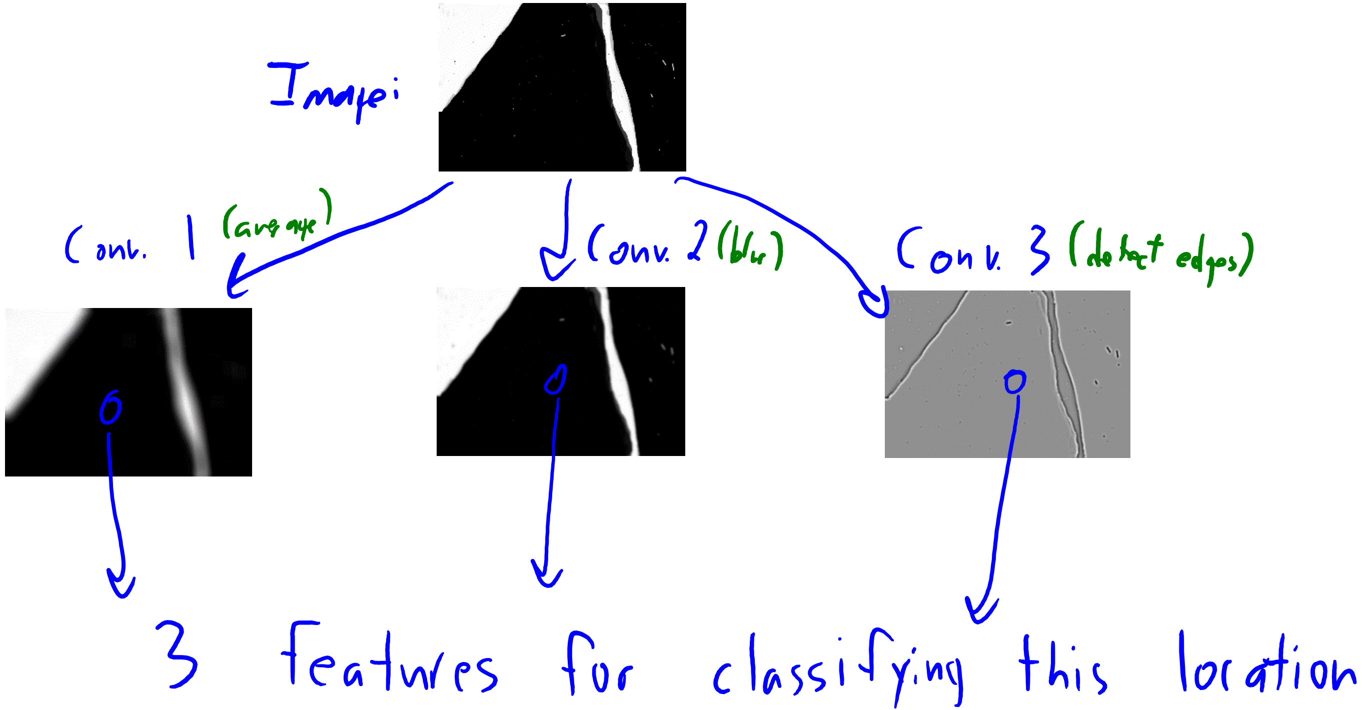
"My neighbours are the same as me"

Useful feature: "My neighbours are brighter than me"

- Taking a “weighted average of neighbours” is called “convolution”.
  - Gives you something like the “words” that make up image regions.

Convolution: Big Picture

- How do you use convolution to get features?
  - Apply several different convolutions to your image.
  - Each convolution gives a different "image" value at each location.
  - Use theses different image values to give features at each location.

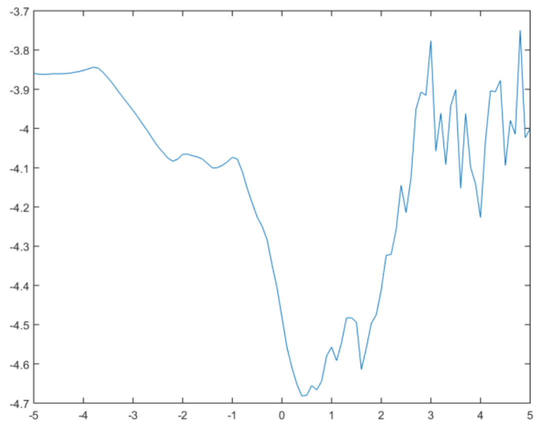


Convolutions: Big Picture

- What can features coming from convolutions represent?
  - Some filters give you an average value of the neighbourhood.
  - Some filters approximate the "first derivative" in the neighbourhood.
    - "Is there a change from dark to bright?"
    - "If so, from which direction in space?"
  - Some filters approximate the "second derivative" in the neighbourhood.
    - "Is there a spike or is the change speeding up?"
- Hope: we can characterize "what happens in a neighbourhood", with just a few numbers.

1D Convolution Example

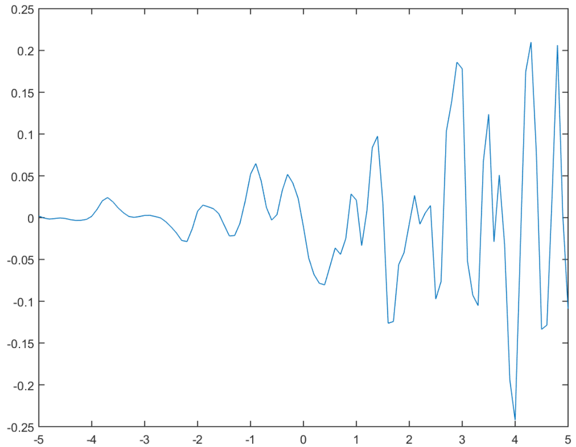
- Consider a 1D "signal" (maybe from sound):
  - We'll come back to images later.



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

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

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



1D Convolution (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 position 'i', and taking a dot product of 'w' with that "part" x_i.

1D Convolution

- 1D convolution example:
  - Signal ‘x’:

0	1	1	2	3	5	8	13

  - Filter ‘w’:

0	-1	2	-1	0

  - Convolution ‘z’:

1D Convolution

- 1D convolution example:
  - Signal ‘x’:

0	1	1	2	3	5	8	13

  - Filter ‘w’:

0	-1	2	-1	0

  - Convolution ‘z’:

		-1					

take dot-product $(0\cdot 0 + 1\cdot(-1) + 1\cdot 2 + 2\cdot(-1) + 3\cdot 0)$

1D Convolution

- 1D convolution example:
  - Signal ‘x’:

0	1	1	2	3	5	8	13

  - Filter ‘w’:

0	-1	2	-1	0

  - Convolution ‘z’:

		-1	0				

1D Convolution

- 1D convolution example:
  - Signal 'x':

0	**1**	**1**	**2**	**3**	**5**	**8**	**13**

  - Filter 'w':

0	**-1**	**2**	**-1**	**0**

  - Convolution 'z':

		-1	**0**	**-1**			

1D Convolution

- 1D convolution example:
  - Signal ‘x’:

0	**1**	**1**	**2**	**3**	**5**	**8**	**13**

  - Filter ‘w’:

0	**-1**	**2**	**-1**	**0**

  - Convolution ‘z’:

		-1	**0**	**-1**	**-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"

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

  - "Local Average"

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

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

average

average

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

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:

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

bonus!

Formal Convolution Definition

- We've defined the convolution as:

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

- In other classes you may see it defined as:

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

(reverses 'w')

$$z_i = \int_{-\infty}^{\infty} w_j x_{i-j} \, dj$$

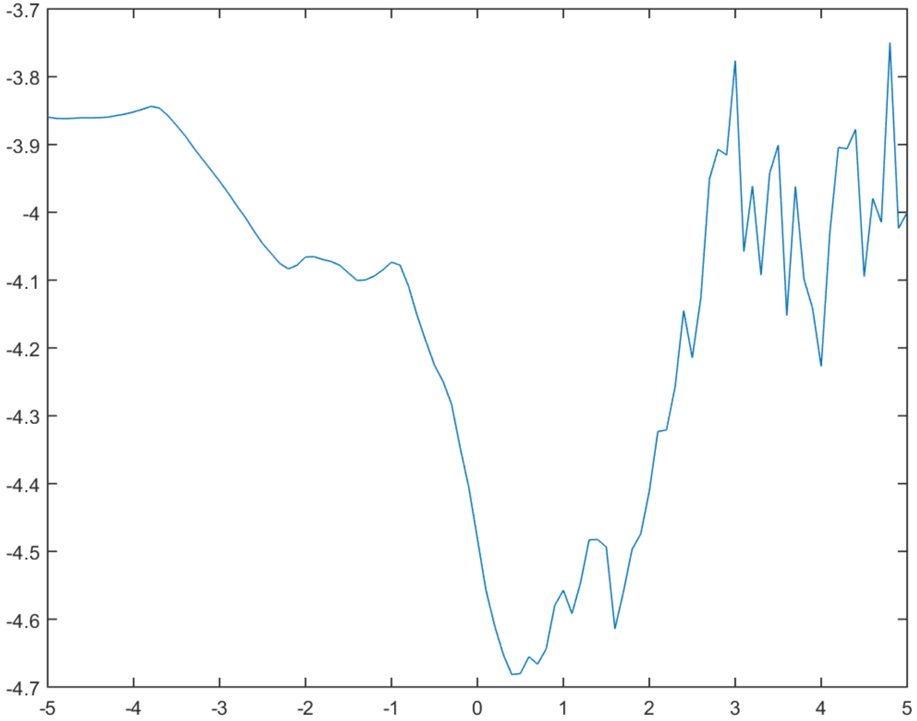
(assumes signal + filter are continuous)

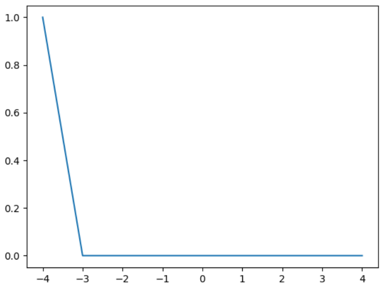
- For simplicity we're skipping the "reverse" step, and assuming 'w' and 'x' are sampled at discrete points (not functions).
- But keep this mind if you read about convolutions elsewhere.

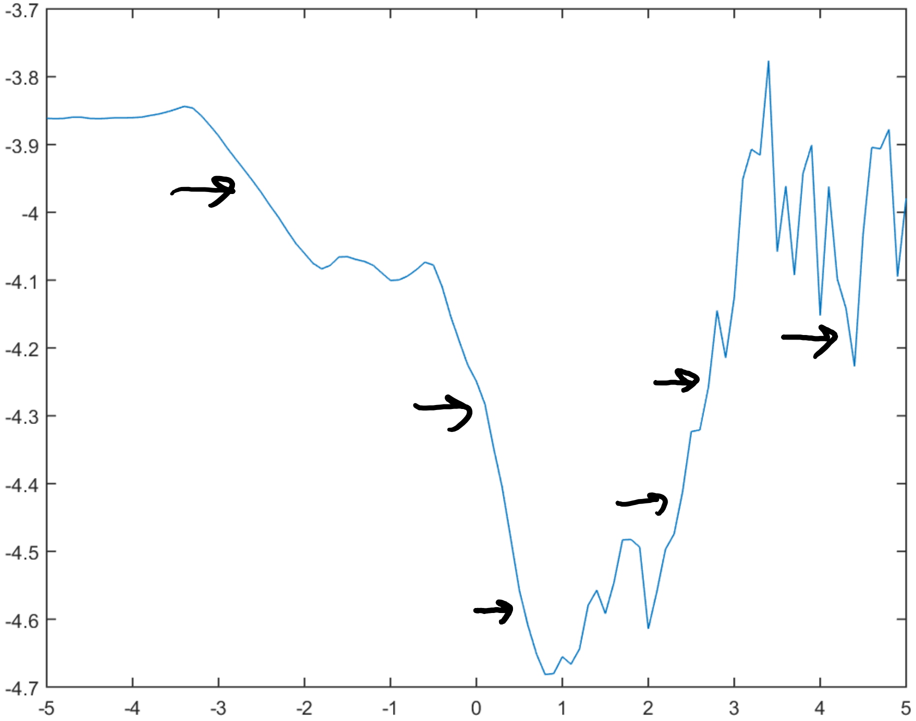
1D Convolution Examples

- Translation convolution shift signal:
  - “What is my neighbour’s value?”

w=[1 0 0 0 0 0 0 0 0]



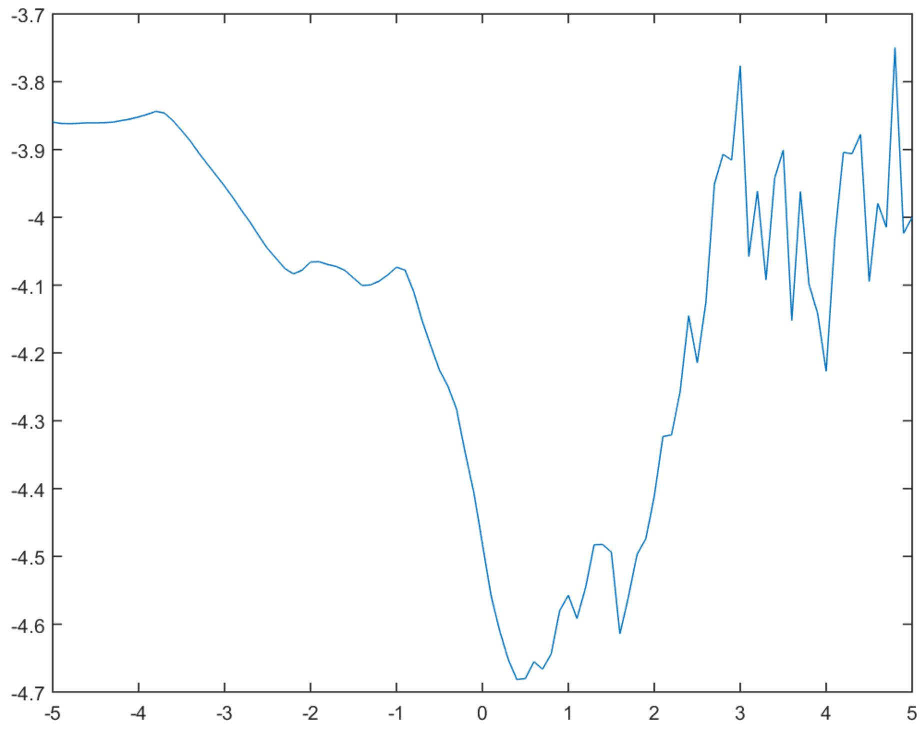


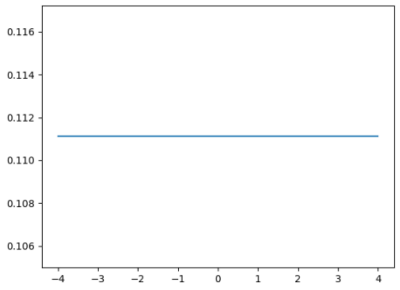


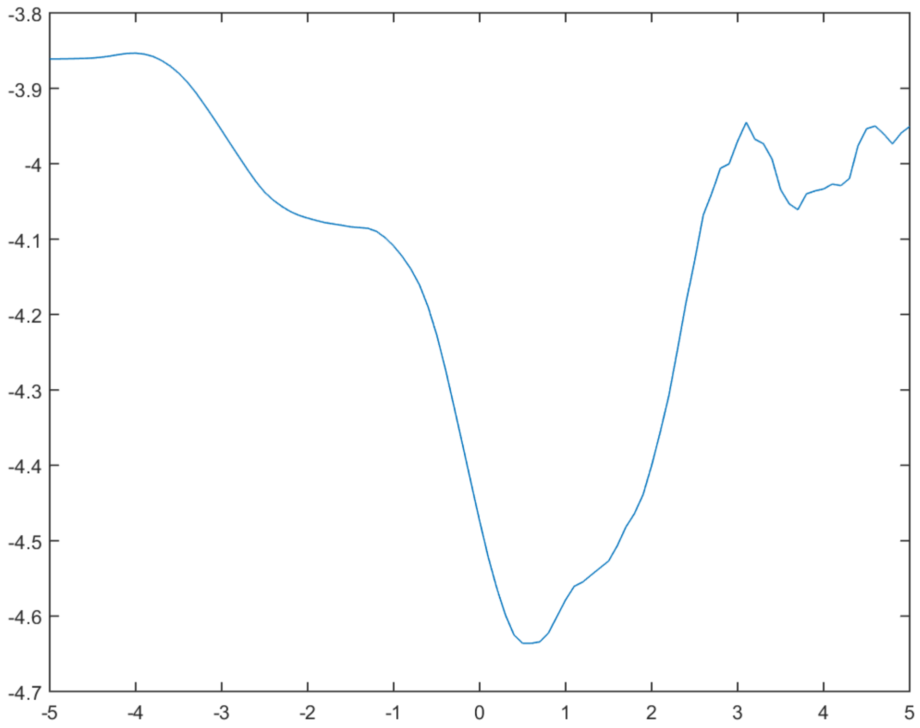
1D Convolution Examples

- Averaging convolution (“is signal generally high in this region?”)
  - Less sensitive to noise (or spikes) than raw signal.

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





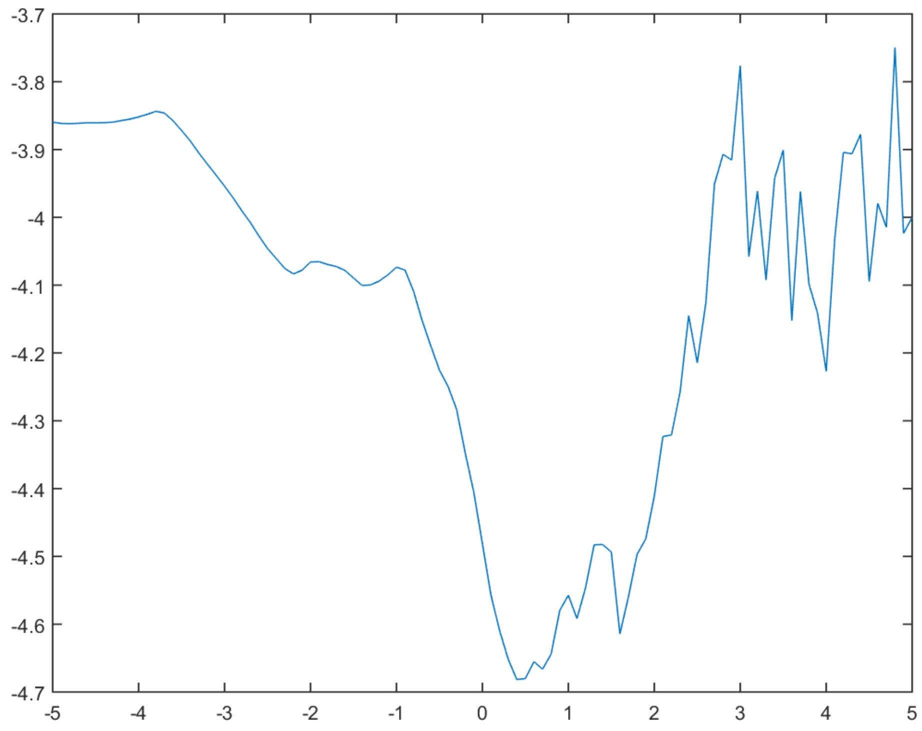


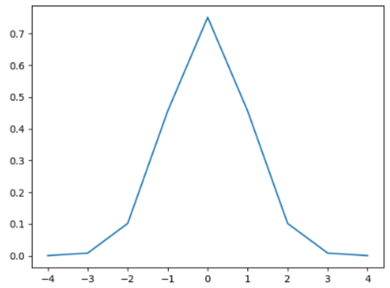
1D Convolution Examples

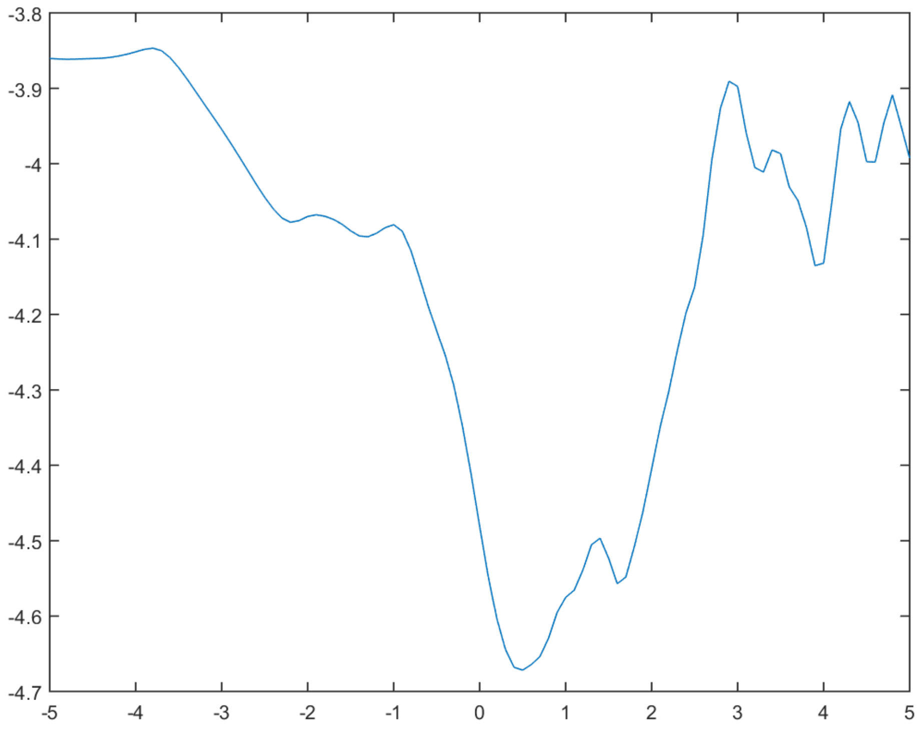
- Gaussian convolution ("blurring"): $w_i \propto \exp\left(-\frac{i^2}{2\sigma^2}\right)$
  – Compared to averaging it's more smooth and maintains peaks better.

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

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



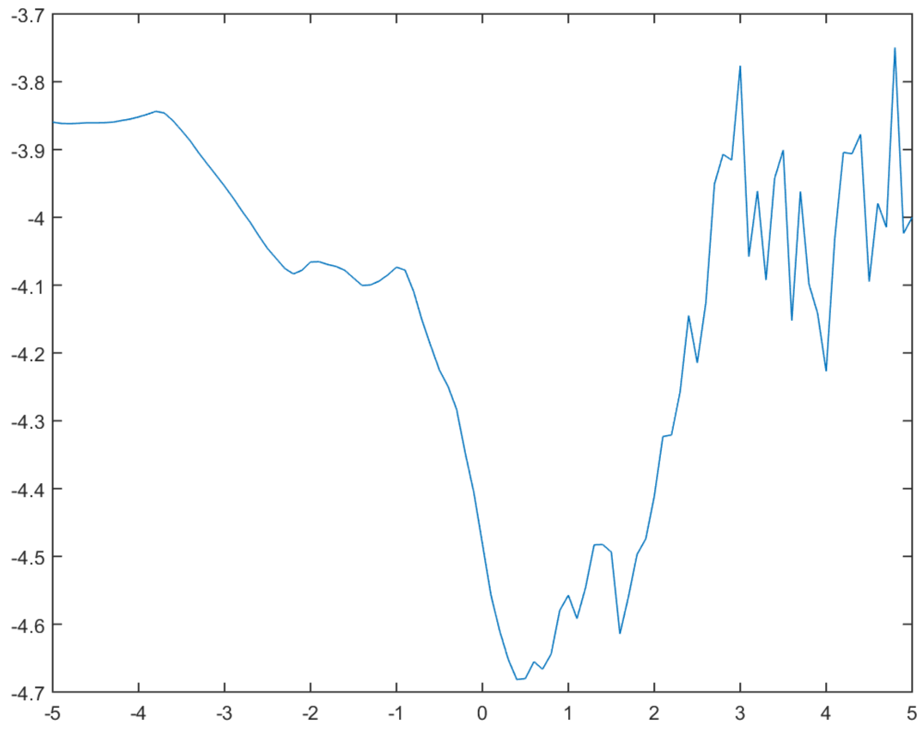


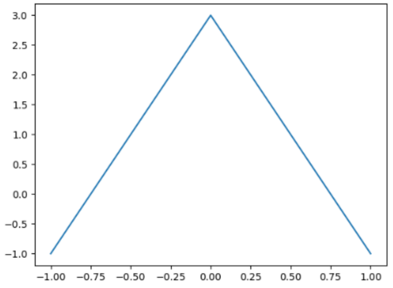


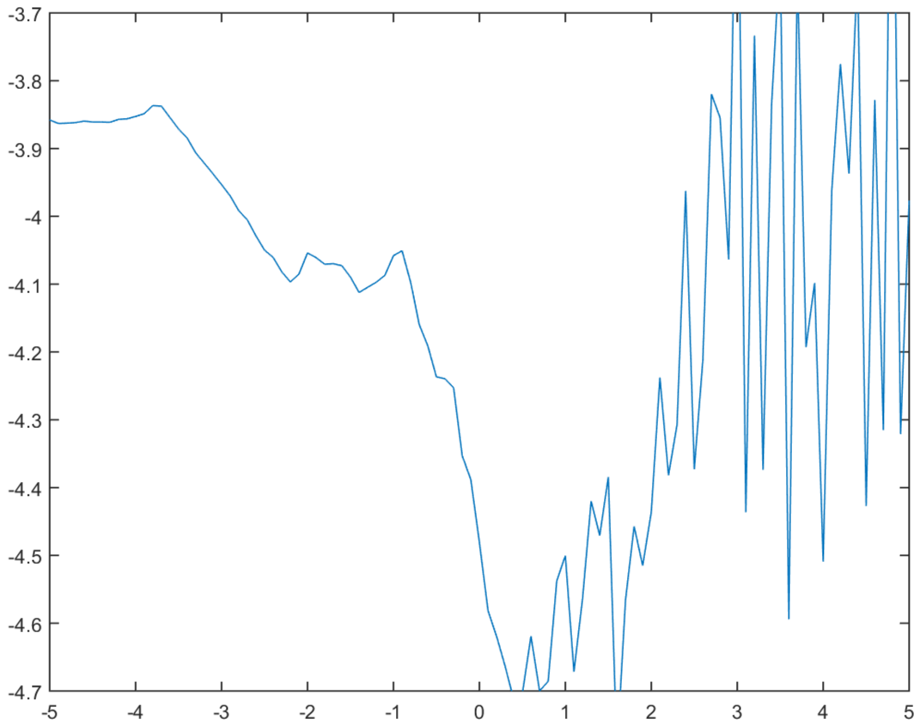
1D Convolution Examples

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

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



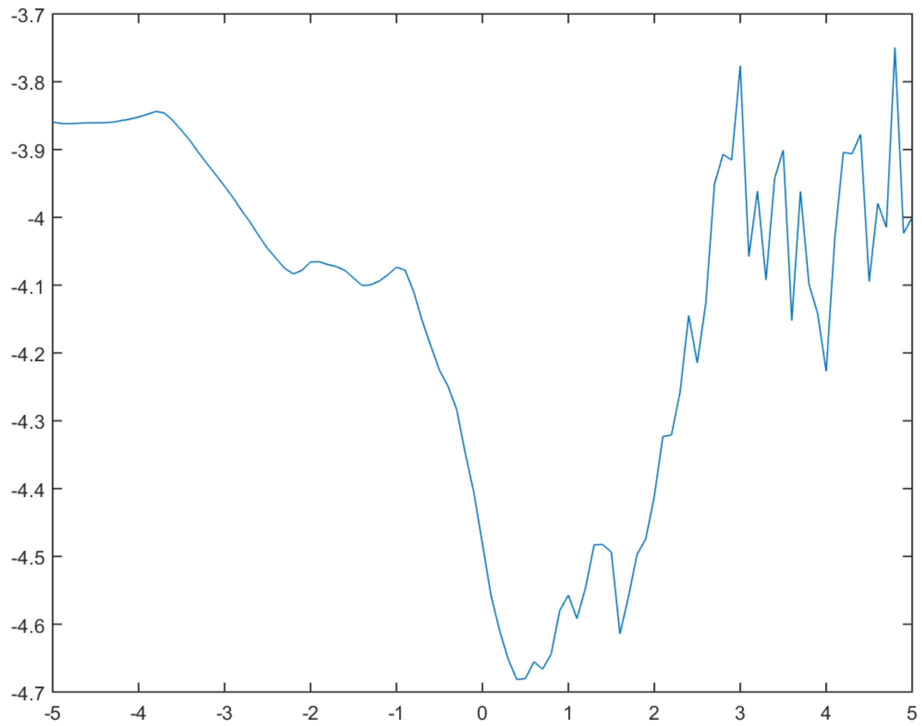


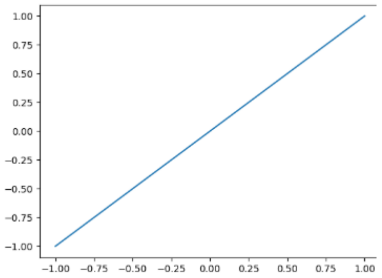


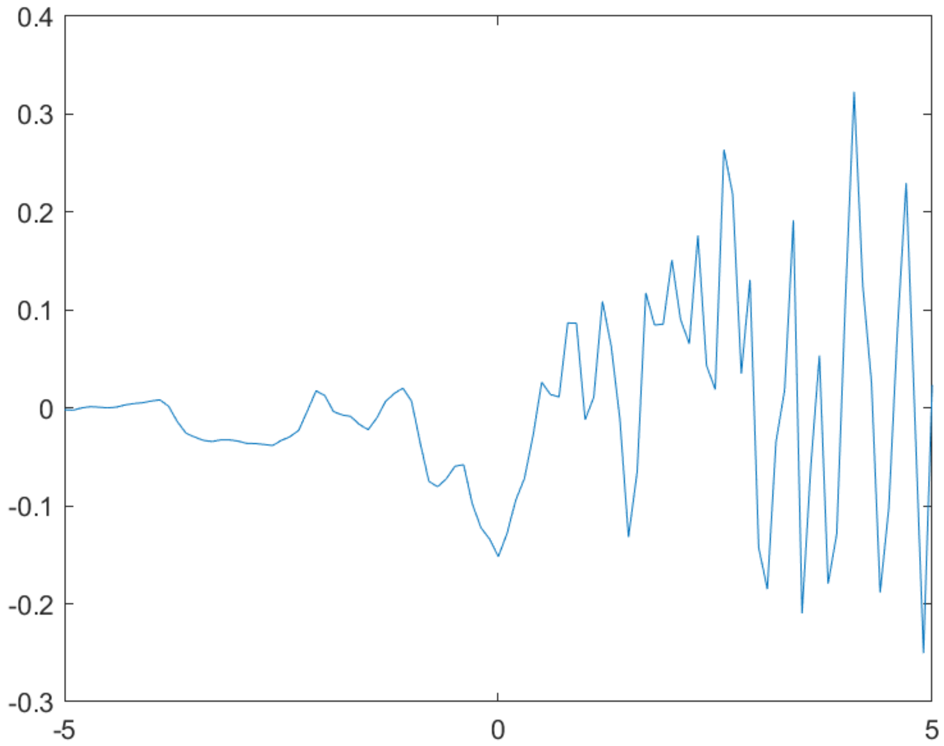
1D Convolution Examples

- Centered difference convolution approximates first derivative:
  - Positive means change from low to high (negative means high to low).

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



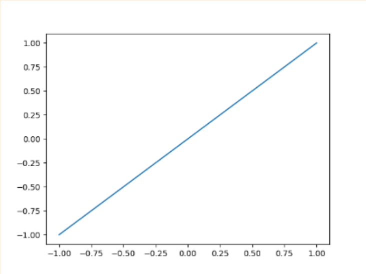




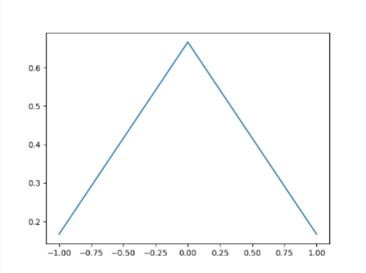
bonus!

Digression: Derivatives and Integrals

- Numerical derivative approximations can be viewed as filters:
  - Centered difference: [-1, 0, 1]
    (like check_correctness in the homework code)



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

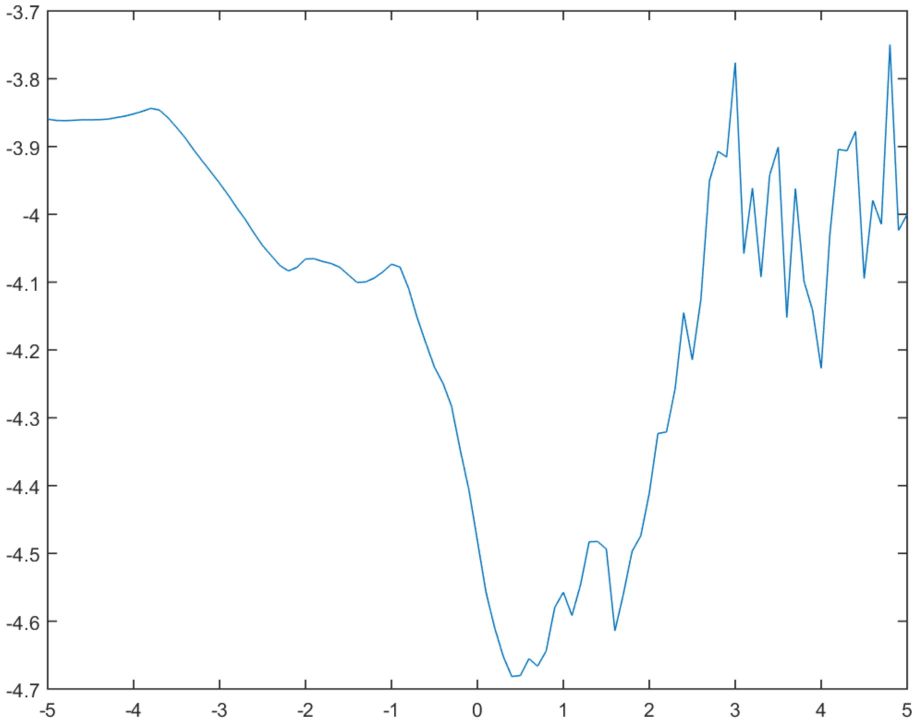


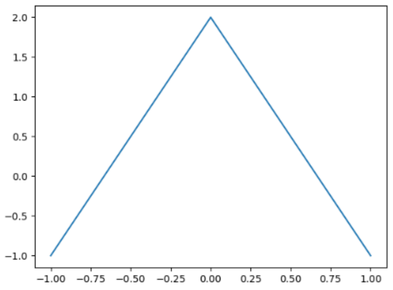
- Derivative filters add to 0, integration filters add to 1
  - For constant function, derivative should be 0 and average = constant.

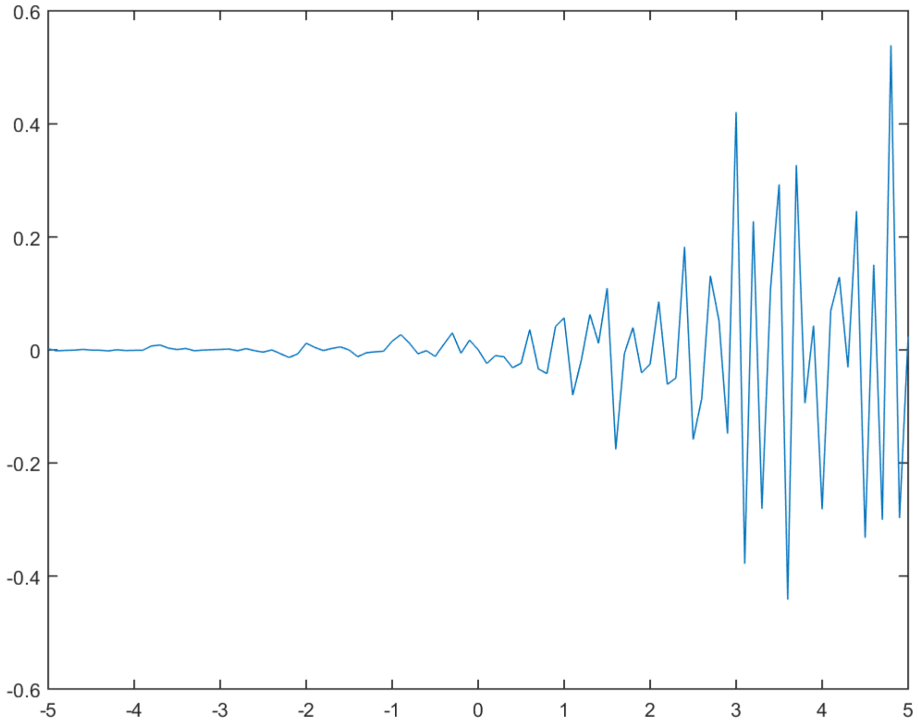
1D Convolution Examples

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

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







Laplacian of Gaussian Filter

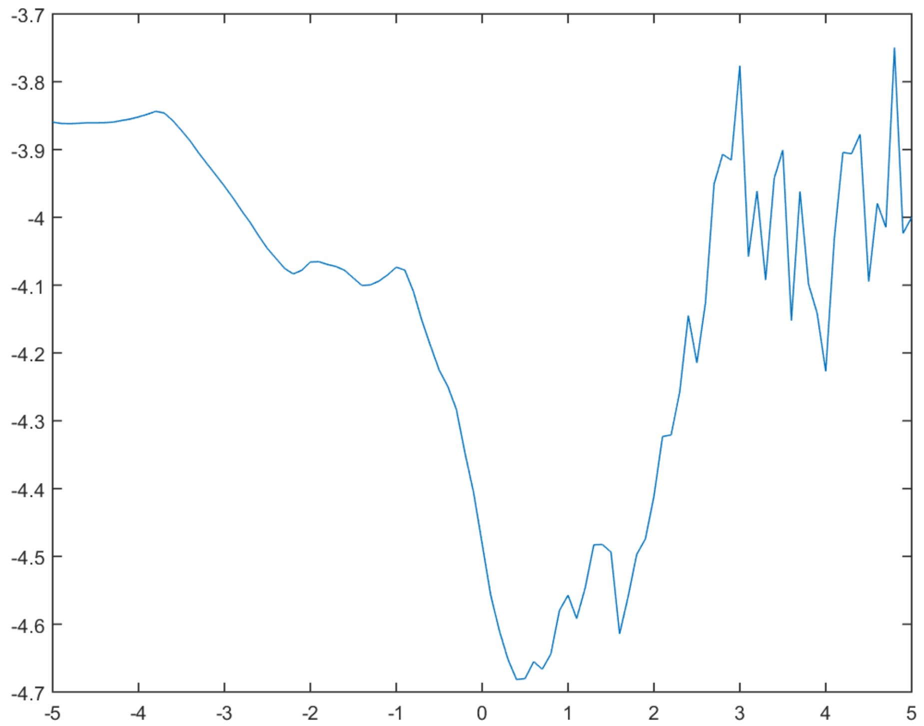
- Laplacian of Gaussian is a smoothed 2^{nd}-derivative approximation:

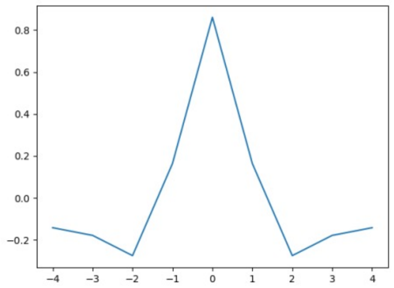
$$w_i = \left(1 - \frac{i^2}{2\sigma^2}\right)\exp\left(-\frac{i^2}{2\sigma^2}\right)$$

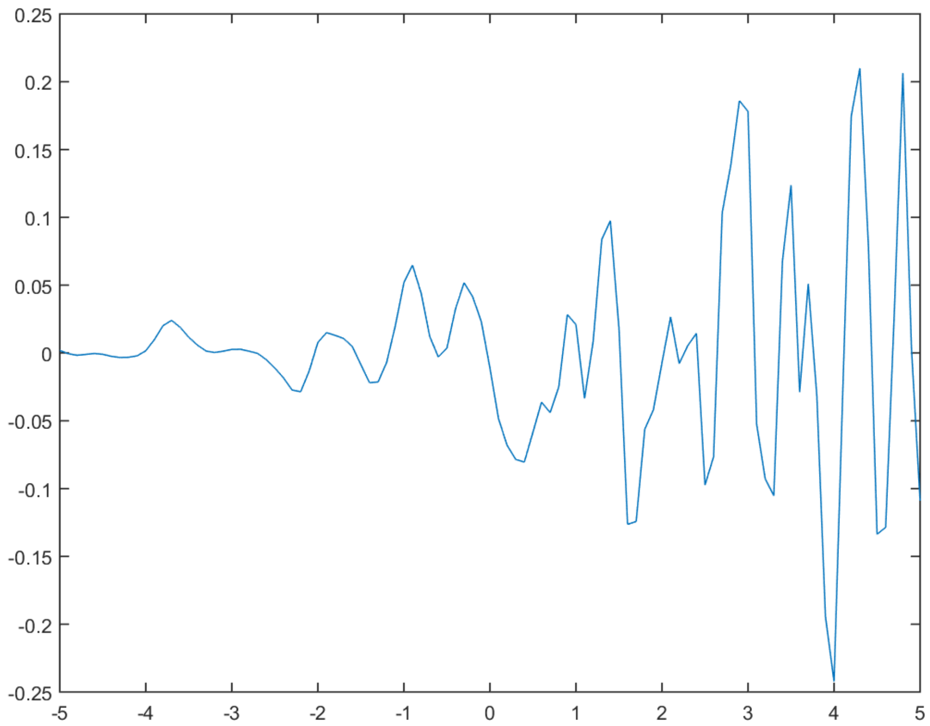
(then subtract mean)

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

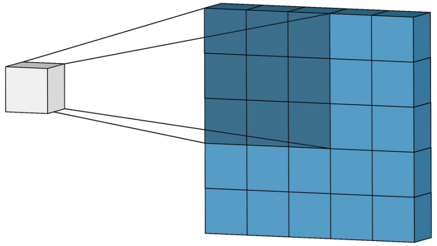
$(\sigma^2 = 1,\ m = 4)$







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.

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

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

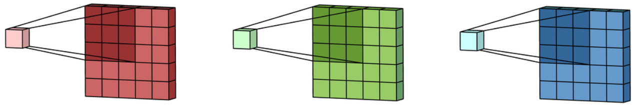
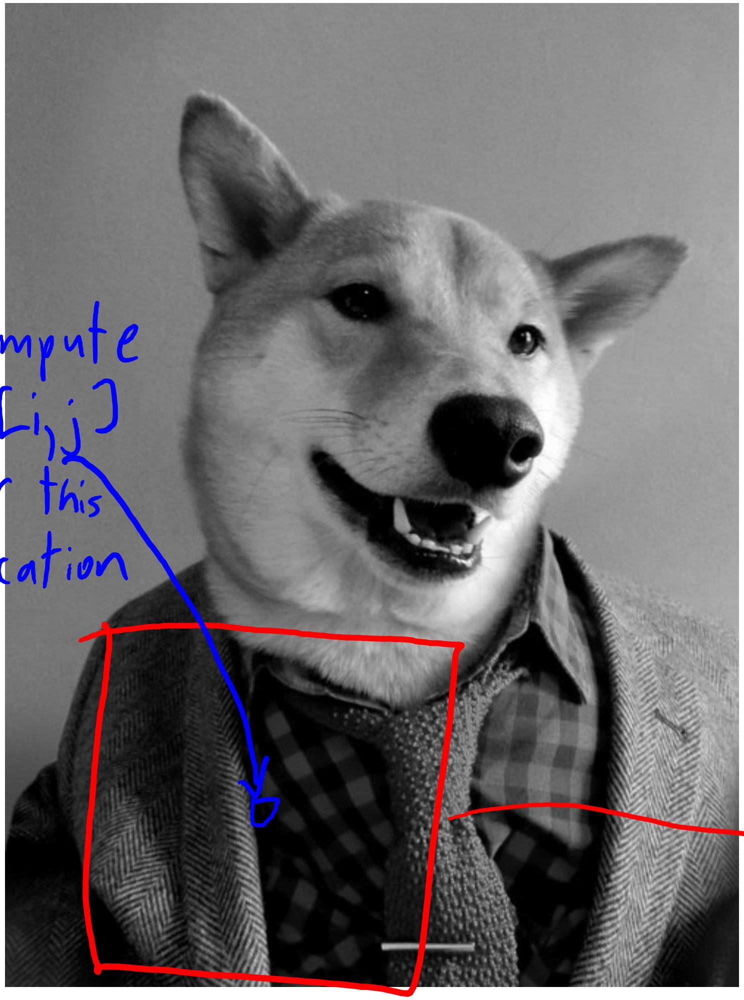


Image Convolution Examples

x

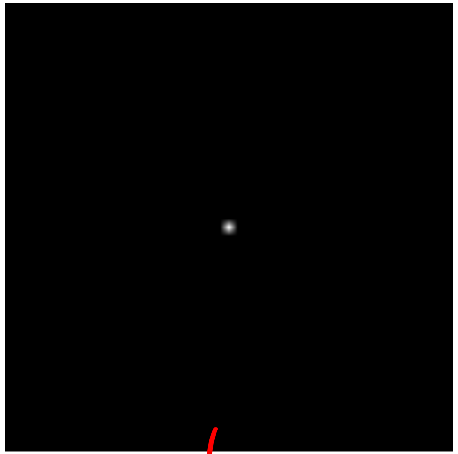


Compute z[i,j] for this location

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

w

$*$



$=$

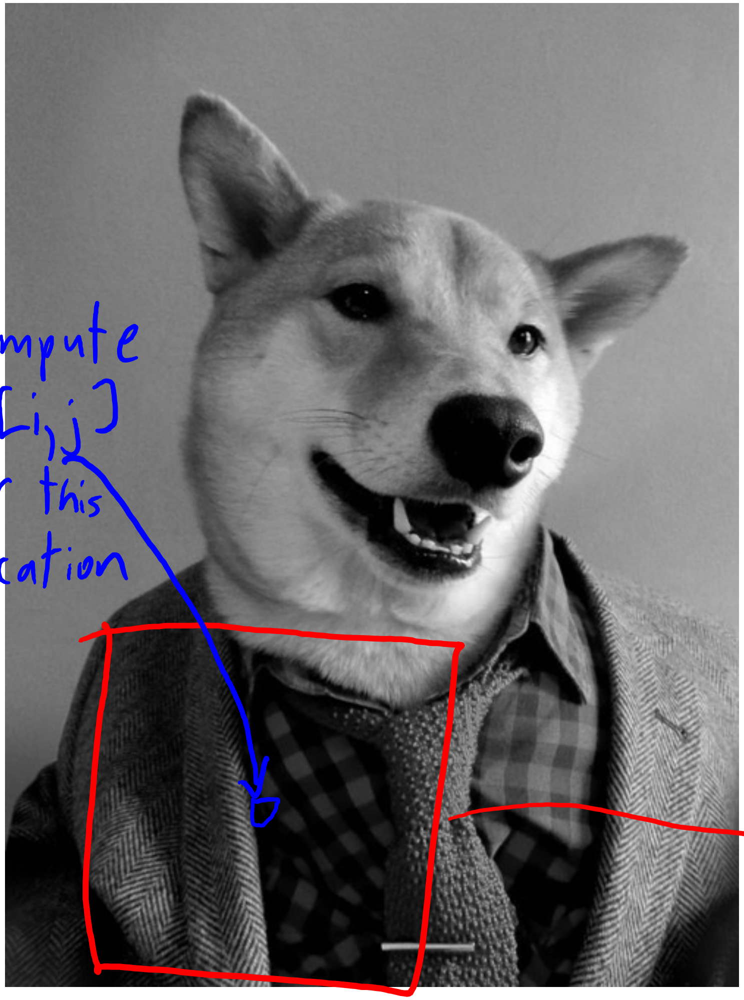
z

multiply element-wise and add up result to get

z[i,j]

Image Convolution Examples

x



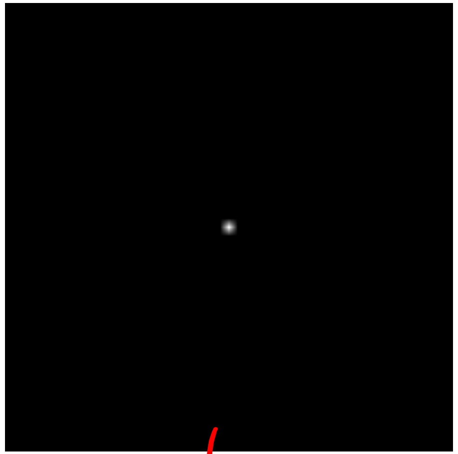
Compute z[i,j] for this location

Identity convolution:

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

$*$

w



$=$

multiply element-wise and add up result to get

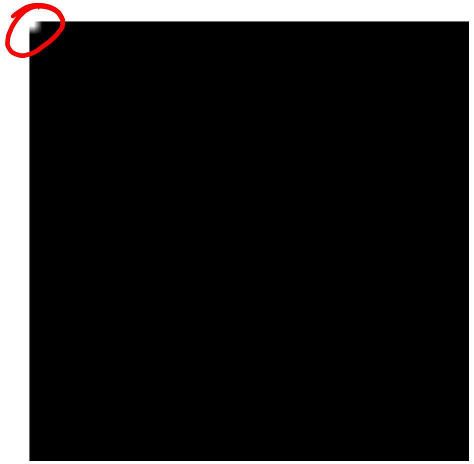
z

z[i,j]

Image Convolution Examples

Translation Convolution:

$*$



$=$

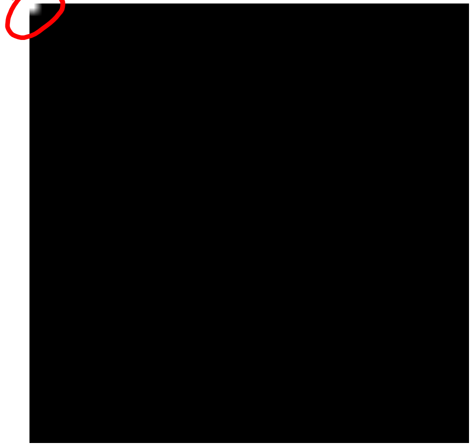
0

Boundary: "zero"

Image Convolution Examples

Translation Convolution:

$*$



Boundary: "replicate"

$=$

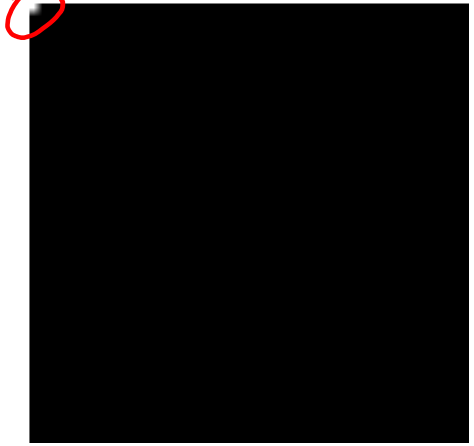
repeats

repeats

Image Convolution Examples

Translation Convolution:

*



=

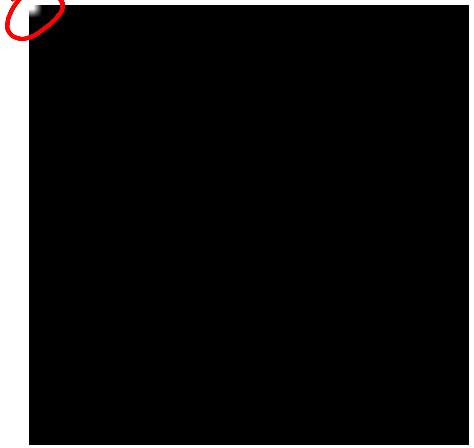
Boundary: "mirror"

flips

Image Convolution Examples

Translation Convolution:

$*$



$=$

Boundary: "ignore"

Image Convolution Examples

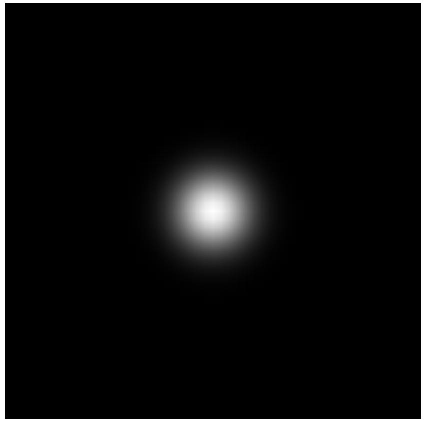
Average convolution:

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

Image Convolution Examples

$*$

Gaussian Convolution:



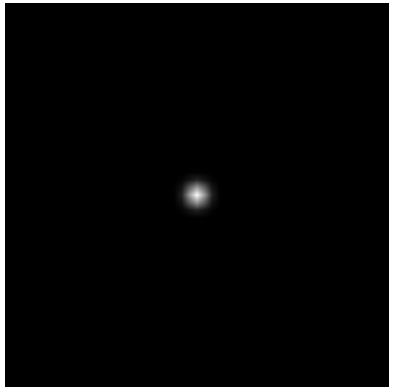
$=$

blurs image to represent average (smoothing)

Image Convolution Examples

Gaussian Convolution:

$*$



(smaller variance)

$=$

blurs image to represent average (smoothing)

Image Convolution Examples

"signed" image (gray is 0)

Laplacian of Gaussian

$*$

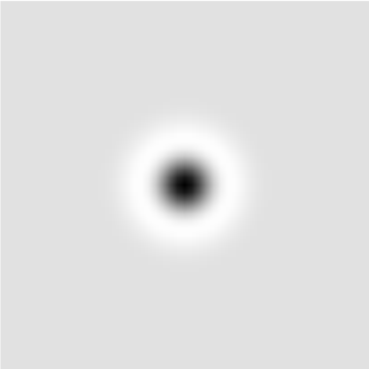
$=$

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

Image Convolution Examples

Laplacian of Gaussian

$*$



$=$

(larger variance)

Black/white as sides of edge

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.

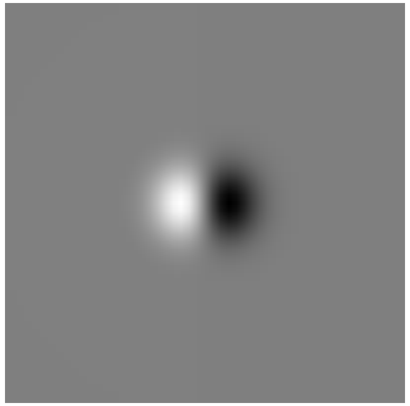
http://setosa.io/ev/image-kernels

Image Convolution Examples

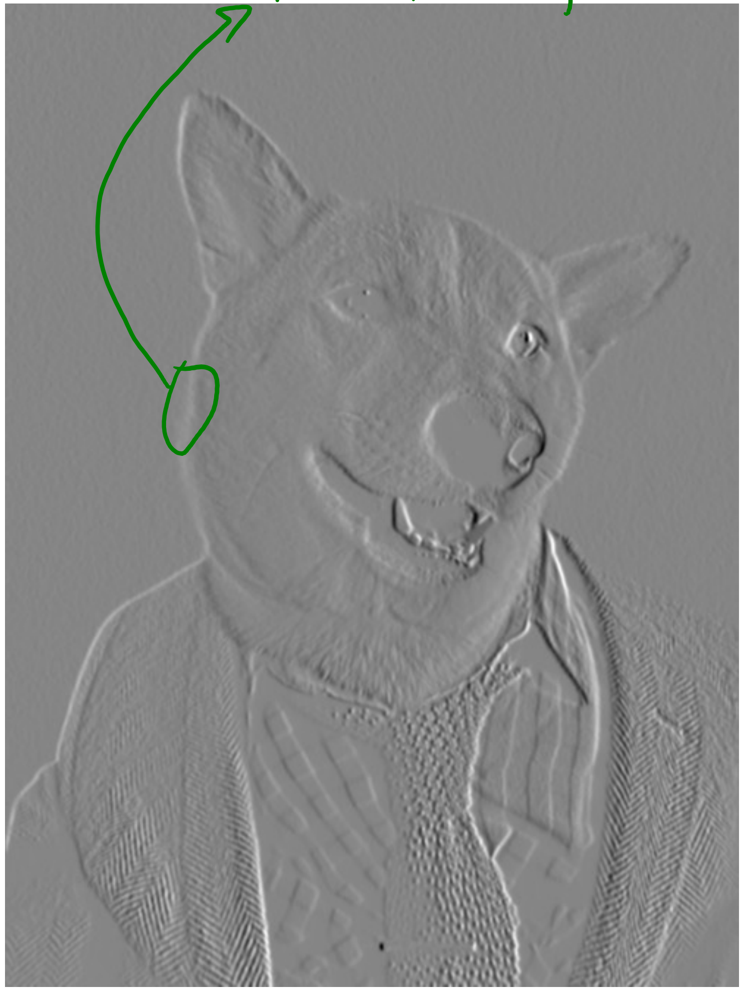
Gabor Filter
(Gaussian multiplied by sine or cosine)

Horizontal "bright to dark"

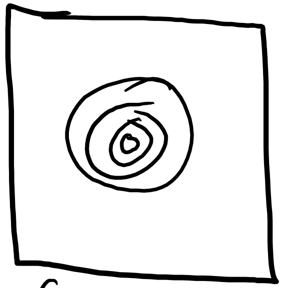
$*$



$=$

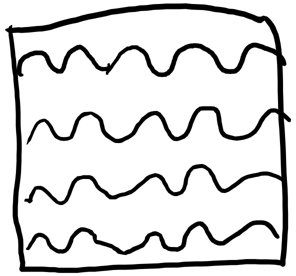


$||$



Gaussian

$.*$



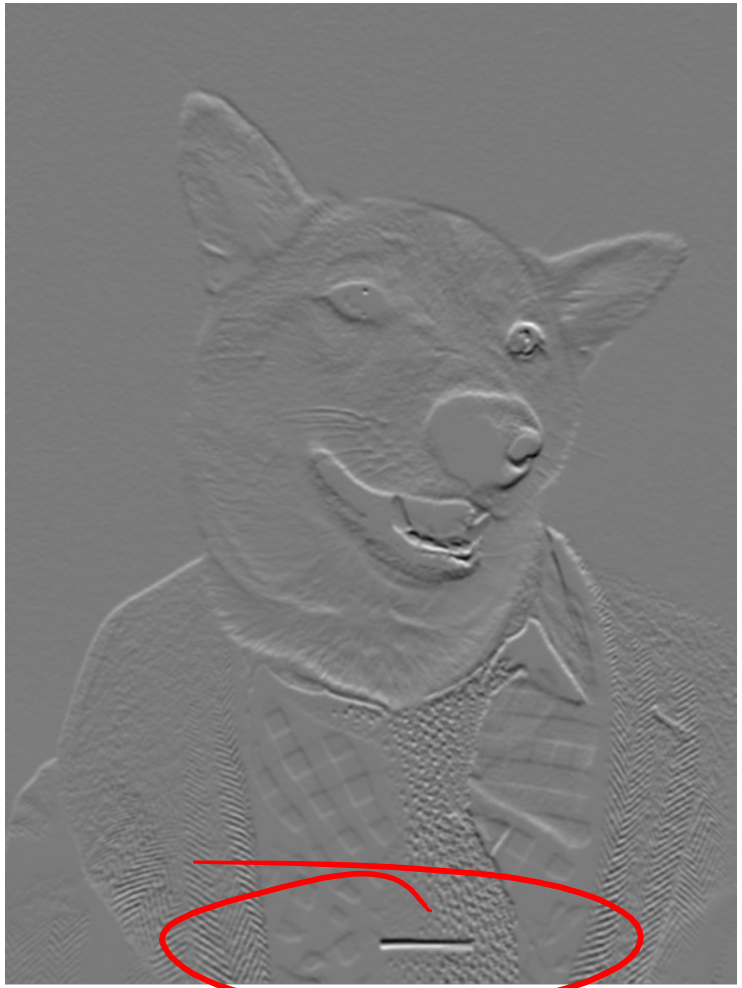
Parallel Sine functions

Image Convolution Examples

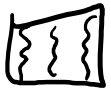
Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$

$=$



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

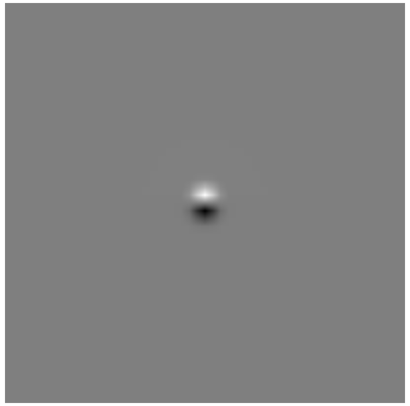


2d derivatives have a direction.

Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

*



(smaller variance)

=

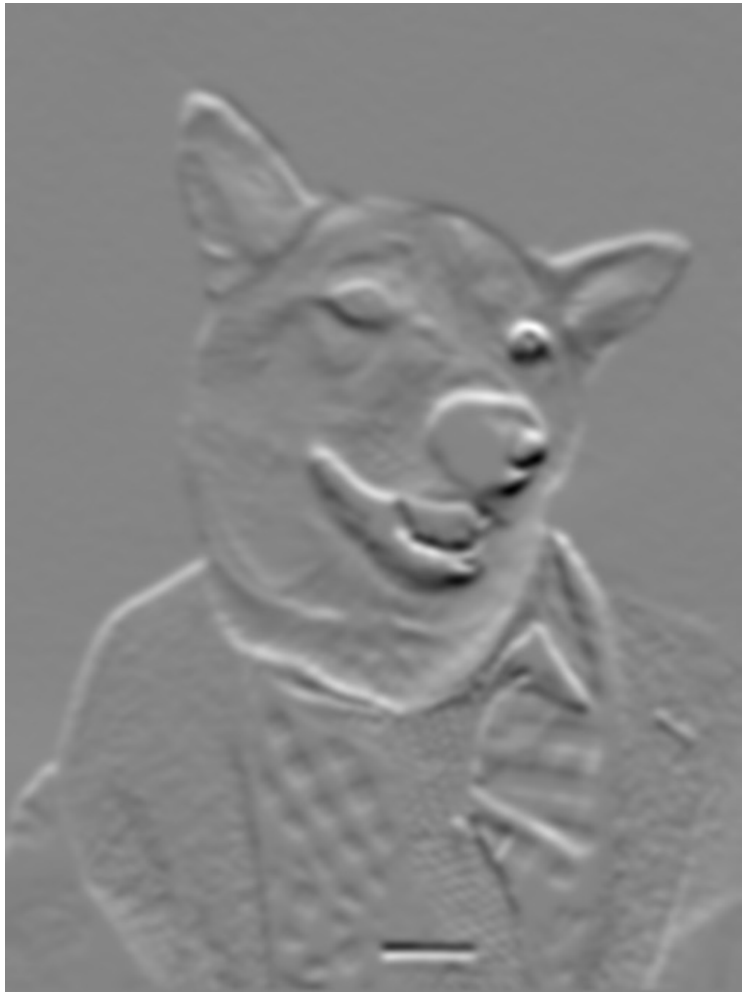
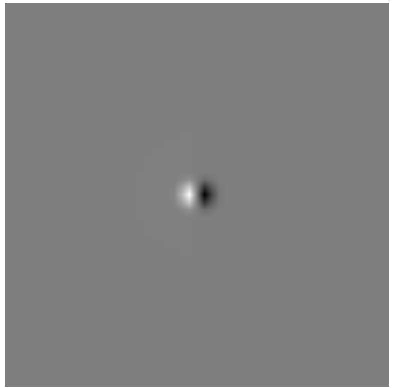


Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$



$=$

(smaller variance)

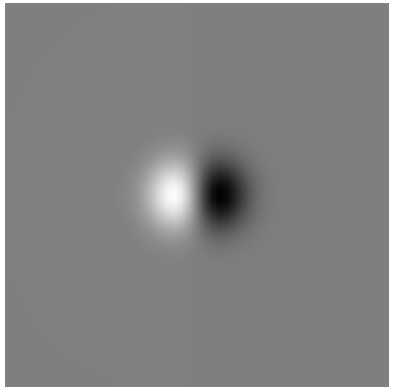
Vertical orientation

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

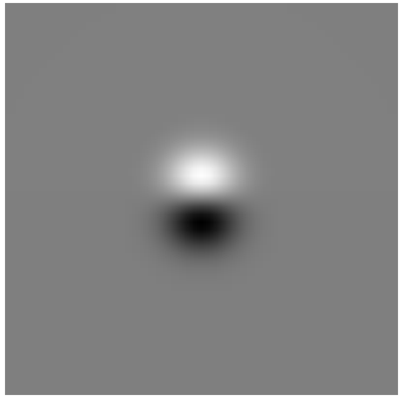
Image Convolution Examples

Max absolute value between horizontal and vertical Gabor:

*



*



maximum absolute value

"Horizontal/vertical edge detector"

3D Convolution

Represent as RGB

Can apply 3D convolutions

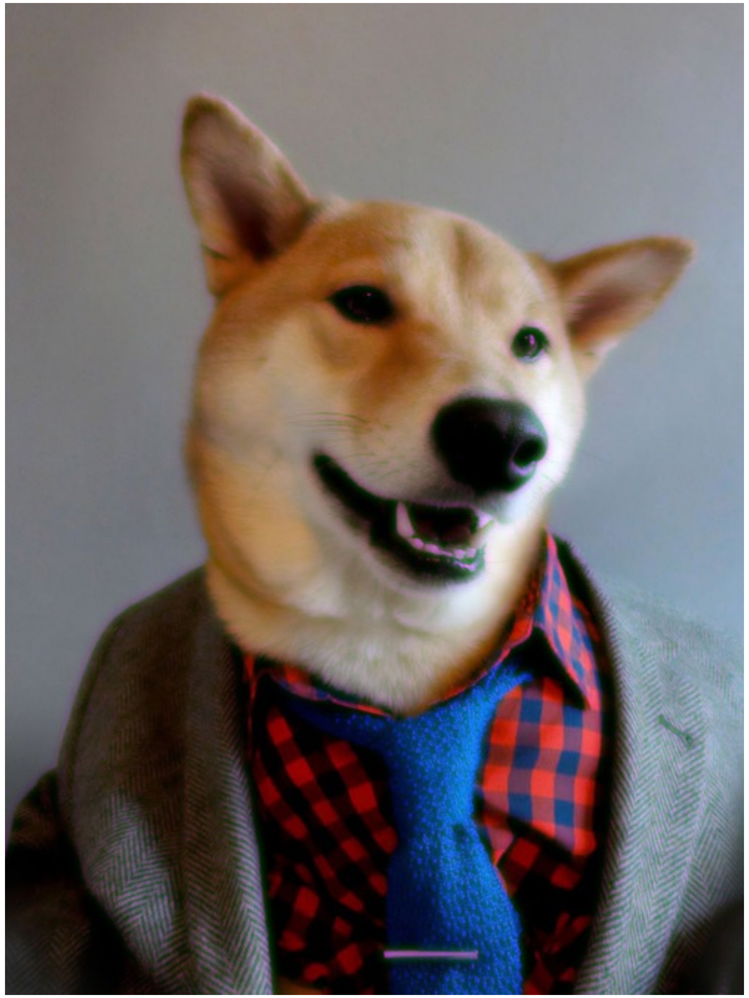
3D Convolution

Gaussian filter

3D Convolution

Gaussian filter

(higher variance on green channel)



3D Convolution

Sharpen the blue channel.

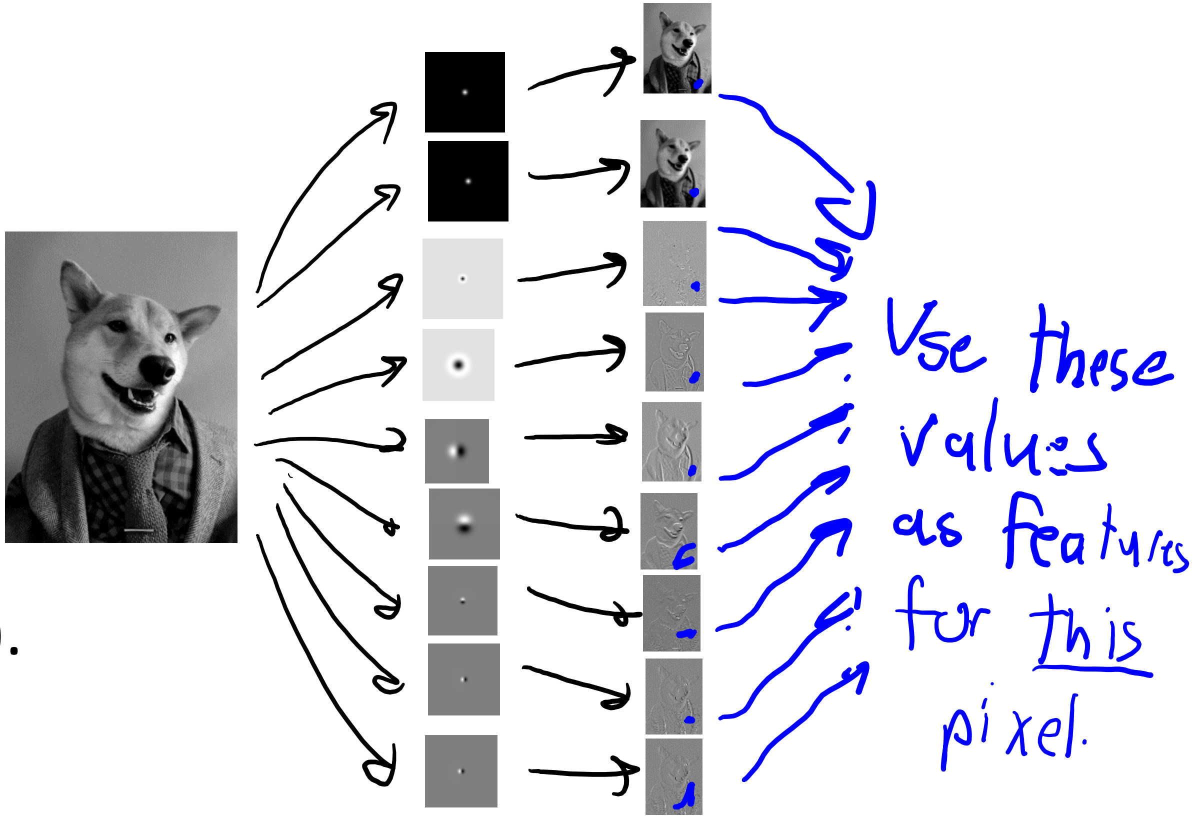
3D Convolution

Gabor filter on
each channel.

Convolutions as Features

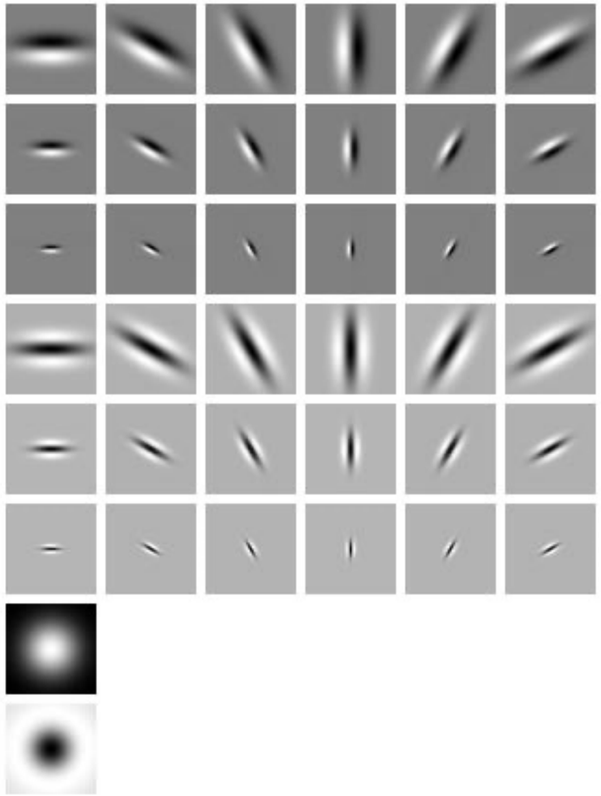
- Classic vision methods use convolutions as features:
  - Usually have different types/variances/orientations.
  - Can take maxes across locations/orientations/scales.

- Notable convolutions:
  - Gaussian (blurring/averaging).
  - Laplace of Gaussian (second-derivative).
  - Gabor filters (directional first- or higher-derivative).



Filter Banks

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



- Convolutional neural networks (next time!) are replacing filter banks.

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

Summary

- Convolutions are flexible class of signal/image transformations.
  - Can approximate directional derivatives and integrals at different scales.
  - Max(convolutions) can yield features invariant to some transformations.
- Filter banks:
  - Make features for a vision problem by taking a bunch of convolutions.

- Next time:
  - Combining this with deep learning.

Global and Local Features for Domain Adaptation

bonus!

- Suppose you want to solve a classification task,
  where you have very little labeled data from your domain.
- But you have access to a huge dataset with the same labels,
  from a different domain.
- Example:
  - You want to label POS tags in medical articles, and pay a few $$$ to label some.
  - You have access the thousands of examples of Wall Street Journal POS labels.
- Domain adaptation: using data from different domain to help.

bonus!

Global and Local Features for Domain Adaptation

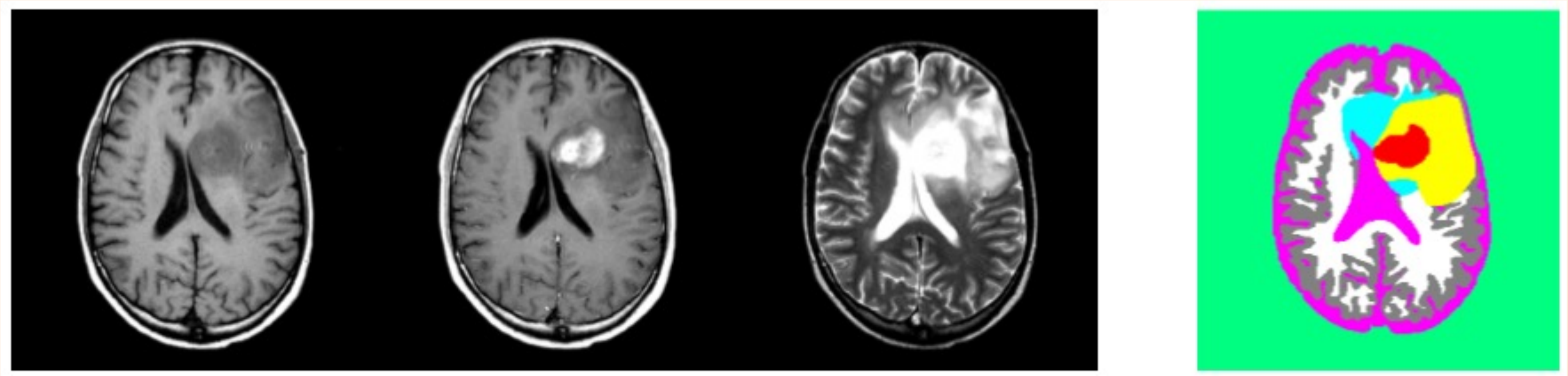
- "Frustratingly easy domain adaptation":
  - Use "global" features across the domains, and "local" features for each domain.
  - "Global" features let you learn patterns that occur across domains.
    - Leads to sensible predictions for new domains without any data.
  - "Local" features let you learn patterns specific to each domain.
    - Improves accuracy on particular domains where you have more data.
  - For linear classifiers this would look like:

$$\hat{y}_i = \text{sign}(\underbrace{w_g^T x_{ig}}_{\text{features used across domains}} + \underbrace{w_d^T x_{id}}_{\text{features/weights specific to domain}})$$

bonus!

Image Coordinates

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

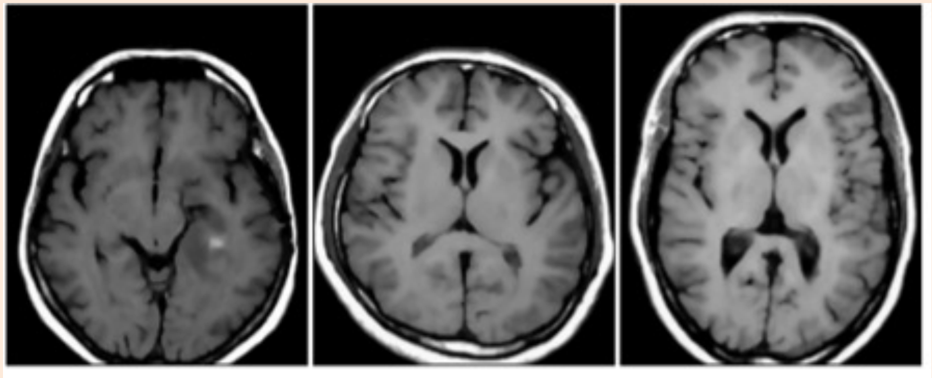


- Considerations:
  - Is the interpretation different in different areas of the image?
  - Are you using a linear model?
    - Would "distance to center" be more logical?
  - Do you have enough data to learn about all areas of the image?

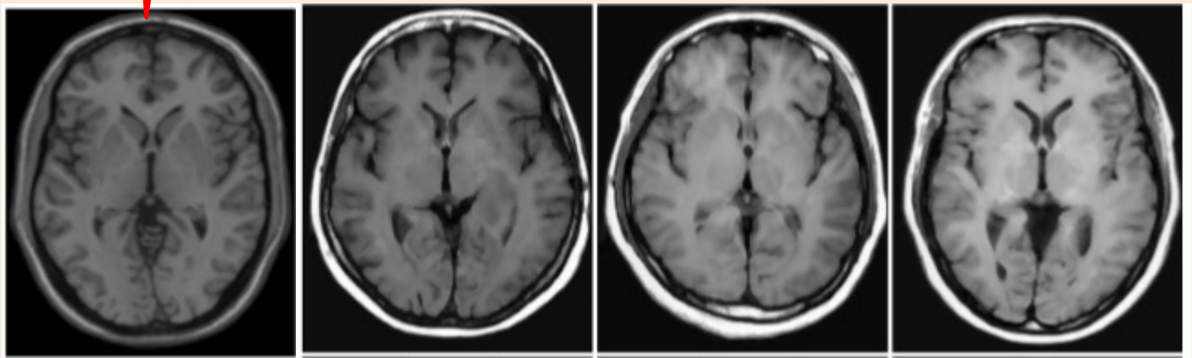
bonus!

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.)

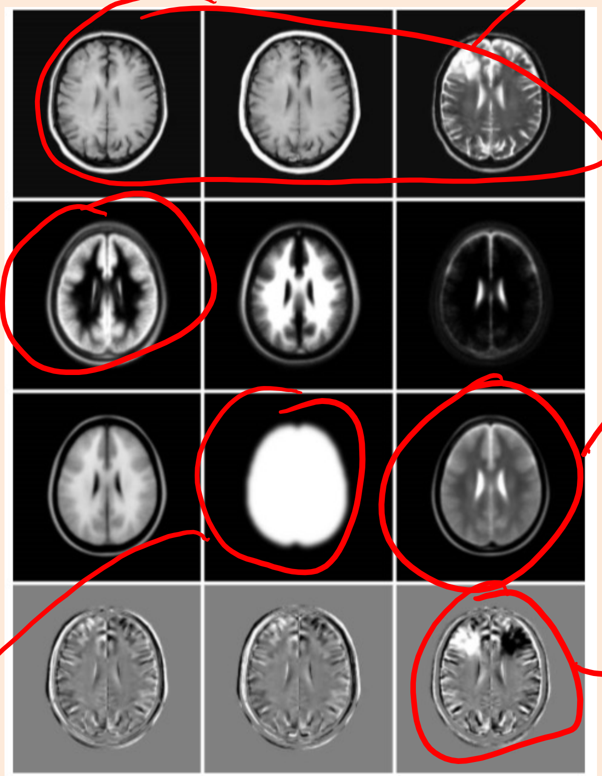
bonus!

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.

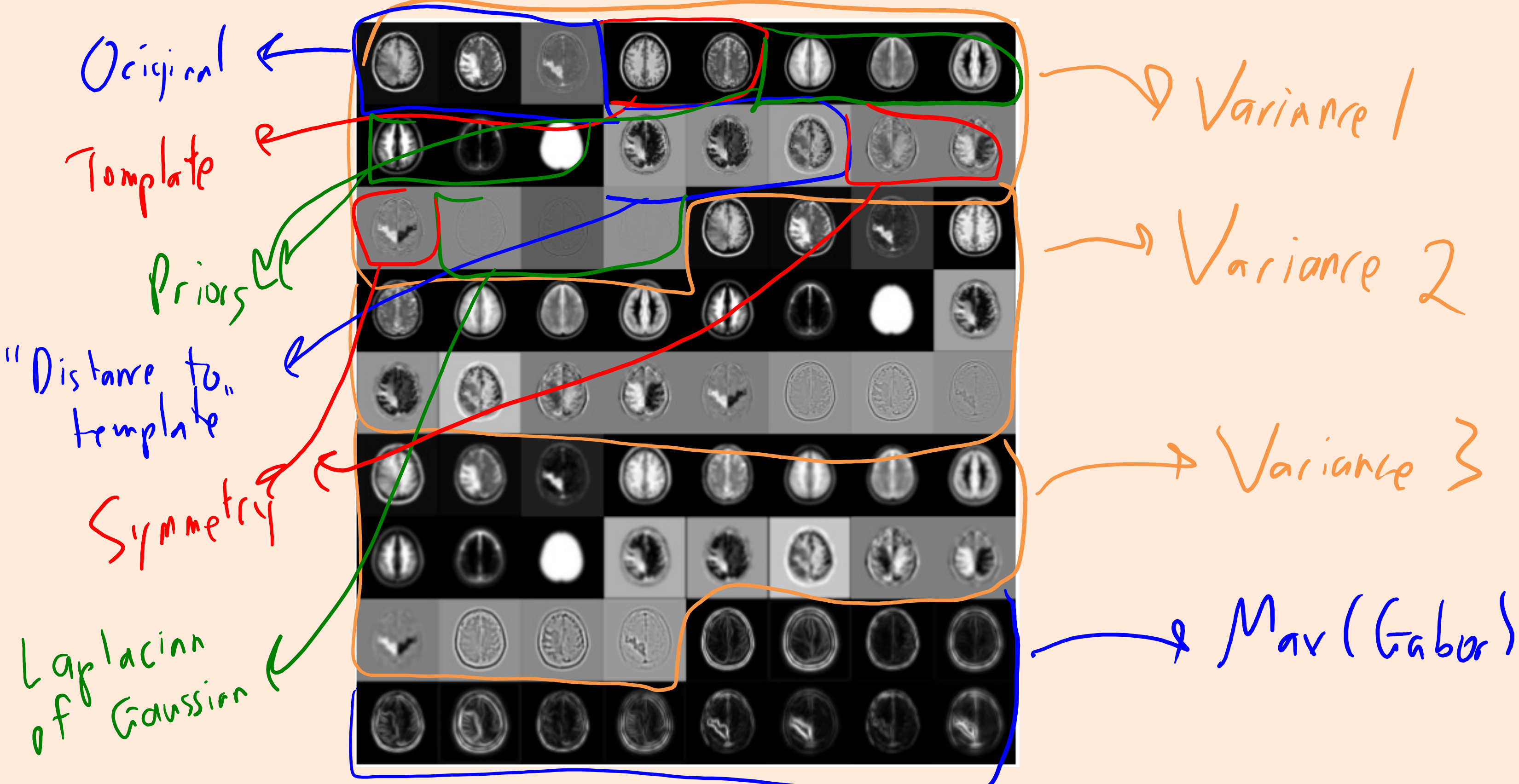
Probability of being brain pixel.

Left-right symmetry difference.

bonus!

Motivation: Automatic Brain Tumor Segmentation

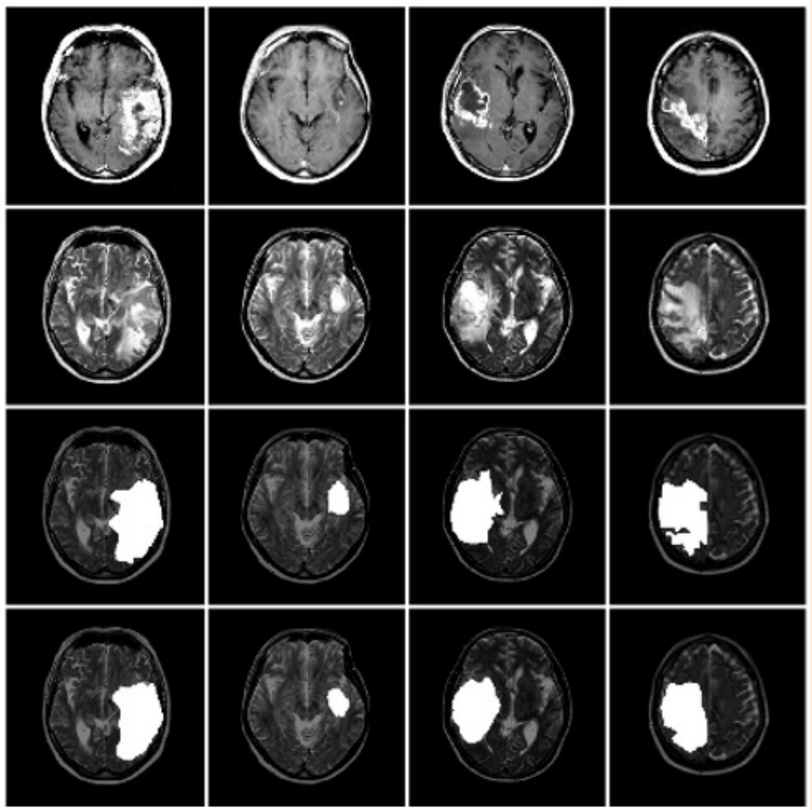
- Final features for brain tumour segmentation:
  - Gaussian convolution of original/template/priors/symmetry, Laplacian of Gaussian on original.
    - All with 3 variances.
    - Max(Gabor) with sine and cosine on orginal (3 variances).



bonus!

Motivation: Automatic Brain Tumour Segmentation

- Logistic regression and SVMs among best methods.
  - When using these 72 features from last slide.
  - If you used all features I came up with, it overfit.



- Possible solutions to overfitting:
  - Forward selection was too slow.
    - Just one image gives 8 million training examples.
  - I did manual feature selection ("guess and check").
  - L2-regularization with all features also worked.
    - But this is slow at test time.
    - L1-regularization gives best of regularization and feature selection.

bonus!

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).

bonus!

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