

CPSC 340: Machine Learning and Data Mining

Convolutions

Spring 2022 (2021W2)

What to Know About Tech Companies Using A.I. to Teach Their Own A.I.

As artificial intelligence developers run out of data to train their models, they are turning to “synthetic data” — data made by the A.I. itself.

Jeff Clune, a computer science professor at the University of British Columbia who previously worked as a researcher at OpenAI, said A.I. models could ultimately become more powerful than the human brain in some ways. But they will do so because they learned from the human brain.

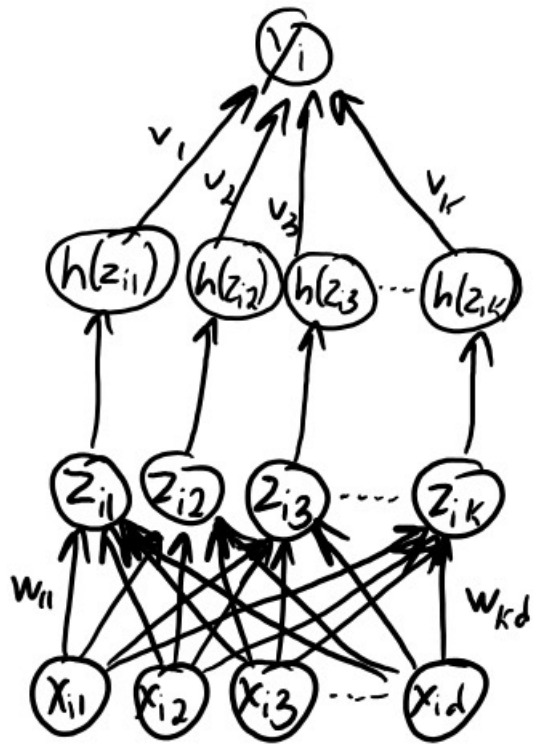
“To borrow from Newton: A.I. sees further by standing on the shoulders of giant human data sets,” he said.

Admin

- Course surveys
 - Faculty of Science requests we allocate class time to them
 - **<https://seoi.ubc.ca/surveys>**
- Reminder
 - We care deeply about your education, so we take them very seriously
 - You will be able to evaluate the class overall, and then each prof separately
 - As always, please remember we're real people, so both praise and constructive criticism feedback are great. Please avoid personal, hurtful, or unconstructive negative comments. Tone matters!

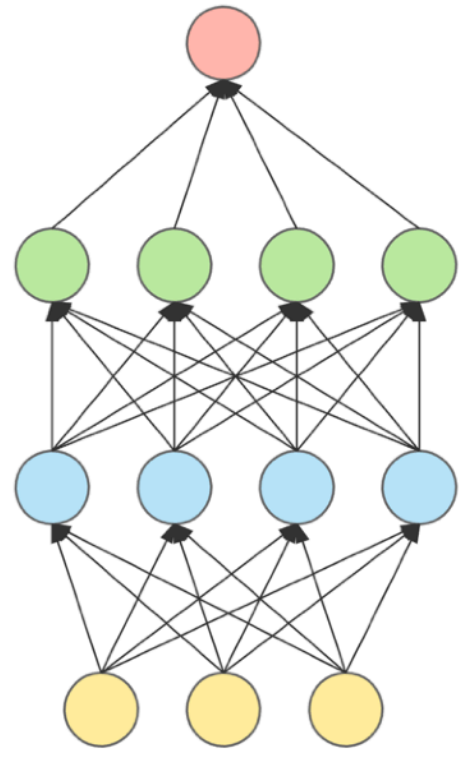
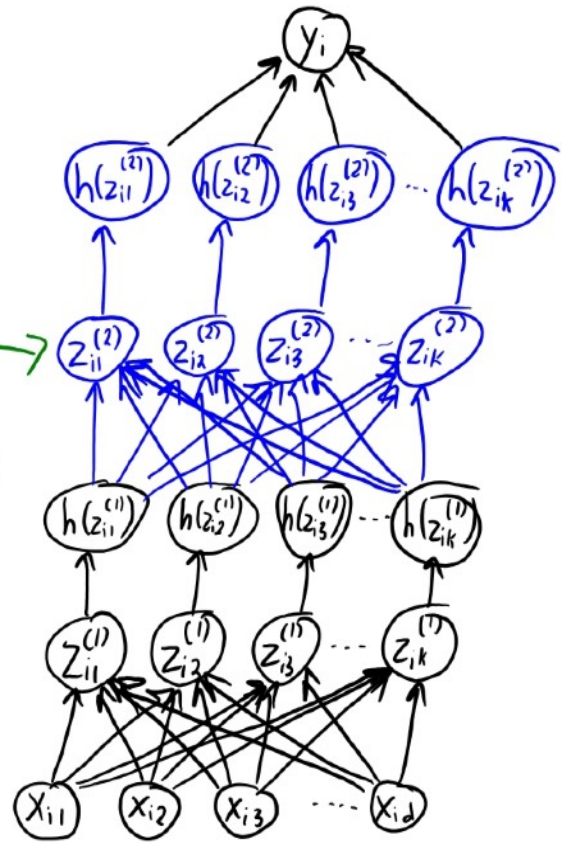
Deep Learning

Neural network:



Deep learning:

Second "layer" of latent features
You can add more "layers" to go "deeper"



output layer
hidden layer 2
hidden layer 1
input layer

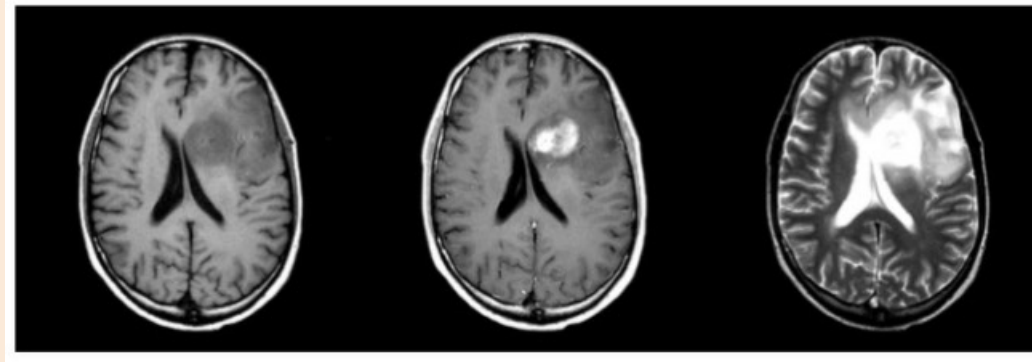
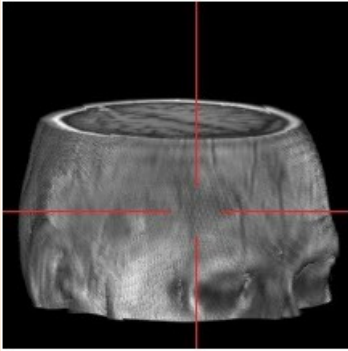
Convolutional Neural Networks

– arguably the most important idea in computer vision

Motivation: Automatic Brain Tumor Segmentation

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

Input:



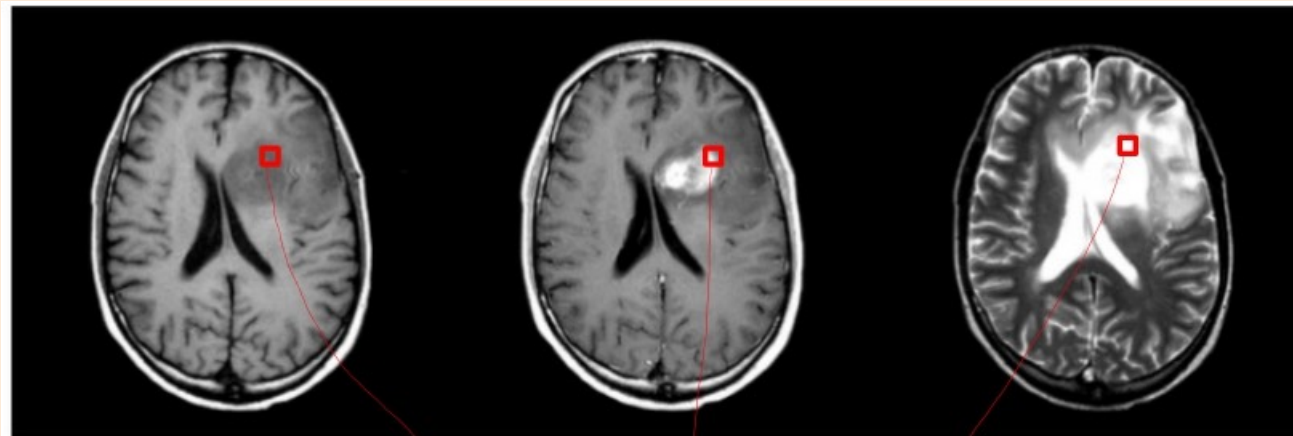
Output:



- 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.



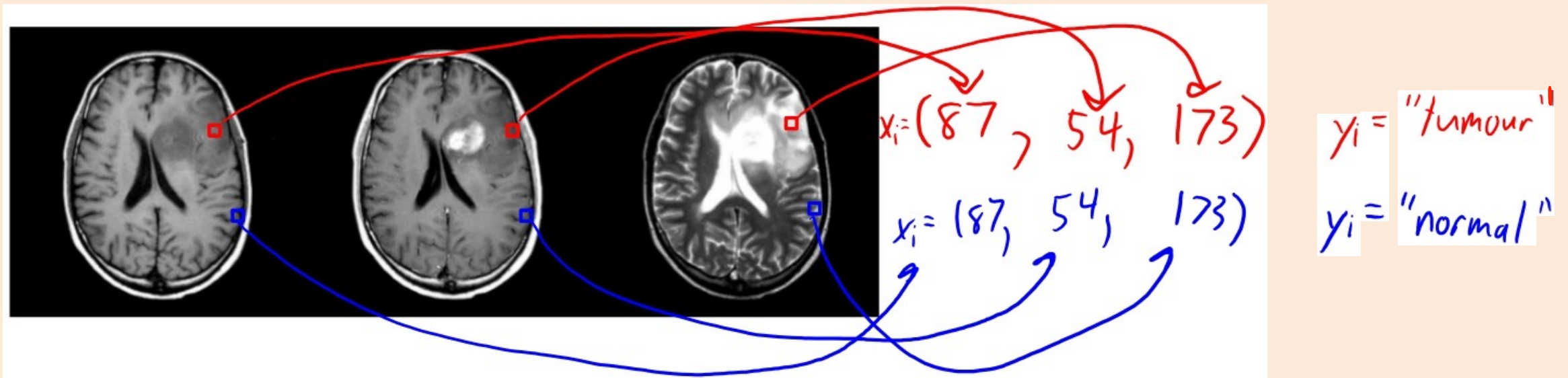
$$x_i = (98, 187, 246)$$

$$y_i = \text{"tumour"}$$

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

Need to Summarize Local Context

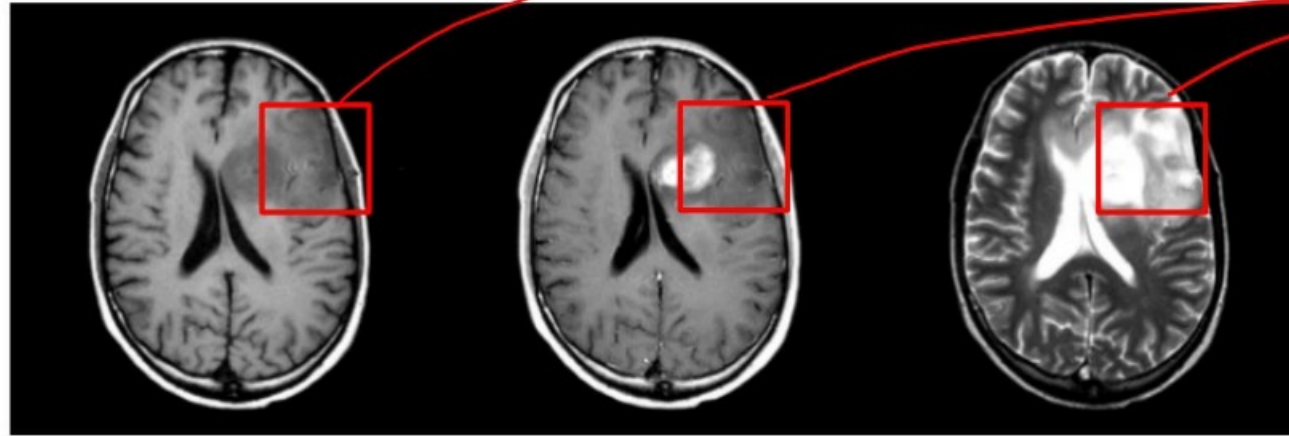
- The individual pixel intensity values are almost meaningless:
 - The same x_i could lead to different y_i .



- 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

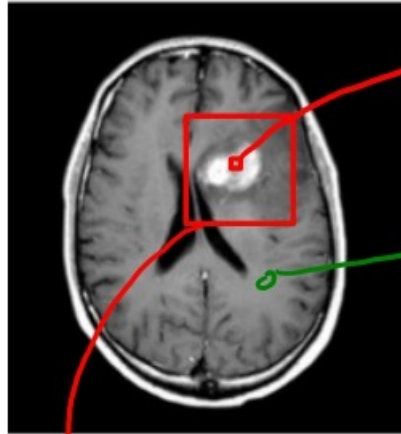
- We need to represent the “context” of the pixel (what is around it).



$$x_i = (\underbrace{\quad\quad\quad}_{\text{red wavy line}}, \underbrace{\quad\quad\quad}_{\text{red wavy line}}, \underbrace{\quad\quad\quad}_{\text{red wavy line}})$$

- 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.

Tumor detection: example feature: measuring “brightness” of a region

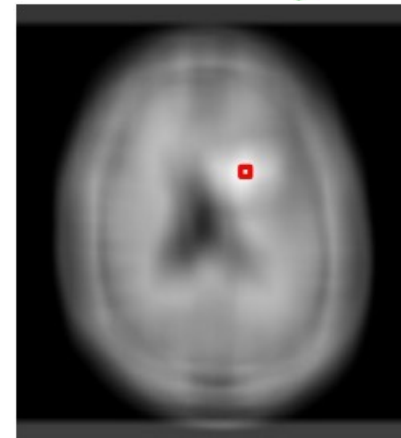


- This pixel is in a “bright” area of the image, which suggests a tumor
- But the actual numeric intensity value of the pixel is the same as in darker “gray matter” areas.
- I want a feature saying “this pixel is in a bright area of the image”.
- This will us help identify that it’s a tumor pixel.

- How to measure brightness in area? Easy way: take **average pixel intensity** in “neighborhood”.

$$z = \frac{1}{|nei|} \sum_{k \in nei} x_k \quad \left. \vphantom{\sum} \right\} \rightarrow \text{new feature is average value in neighbourhood.}$$

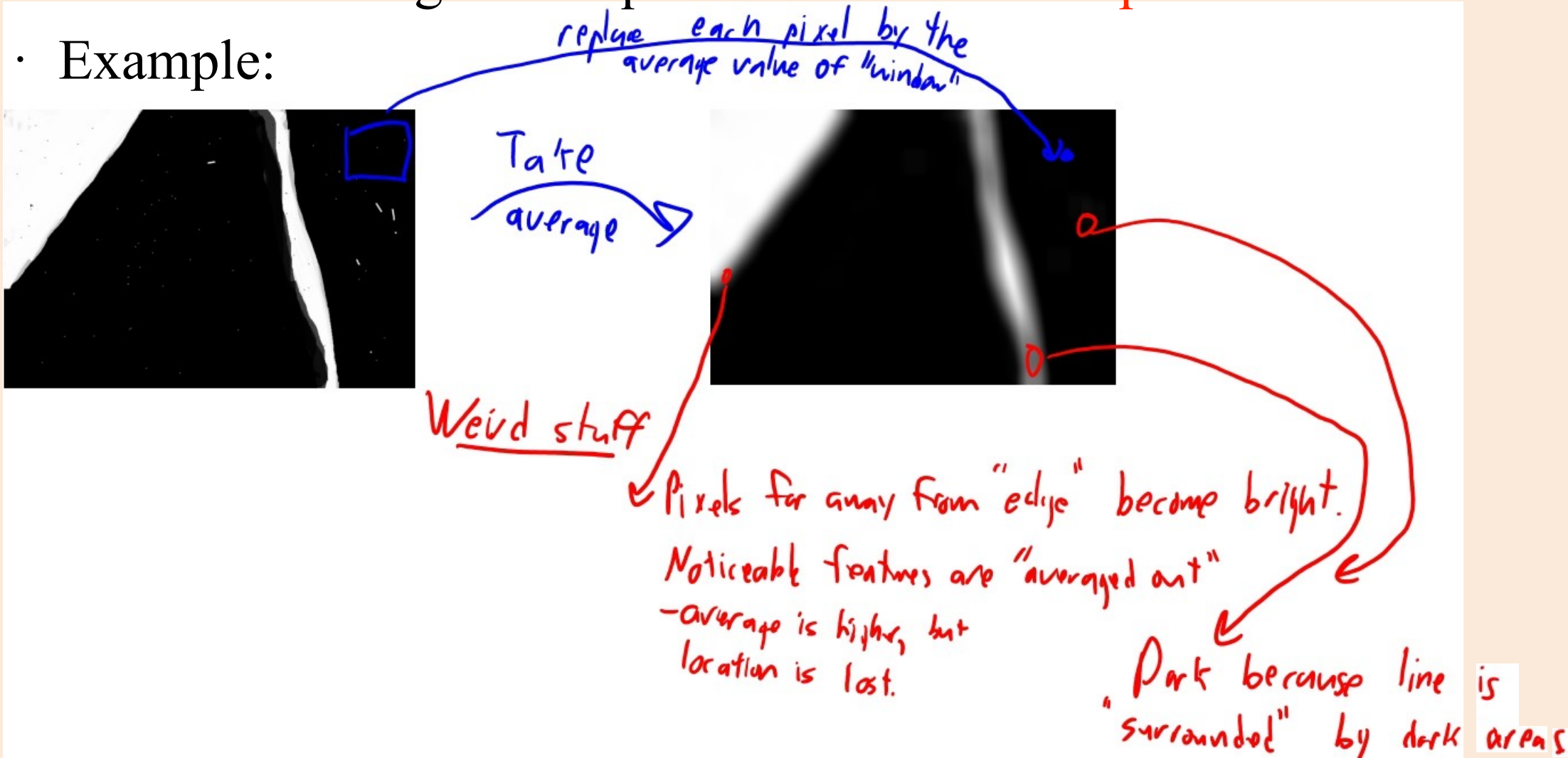
- Applying this “averaging” to **every pixel** gives a new image:
- We can use “**pixel value in new image**” as a new feature.
 - New feature helps identify if pixel is in a “bright” area.



The annoying thing about squares

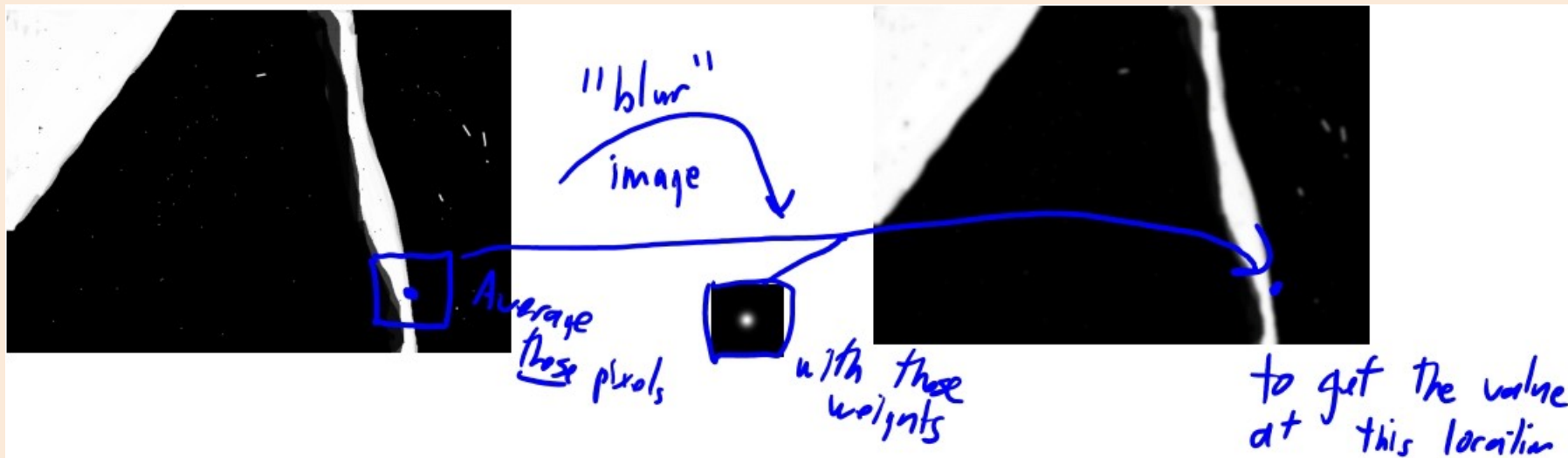
- “Take the average of a square window” **loses spatial information.**

• Example:



Fixing the “square” issues

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



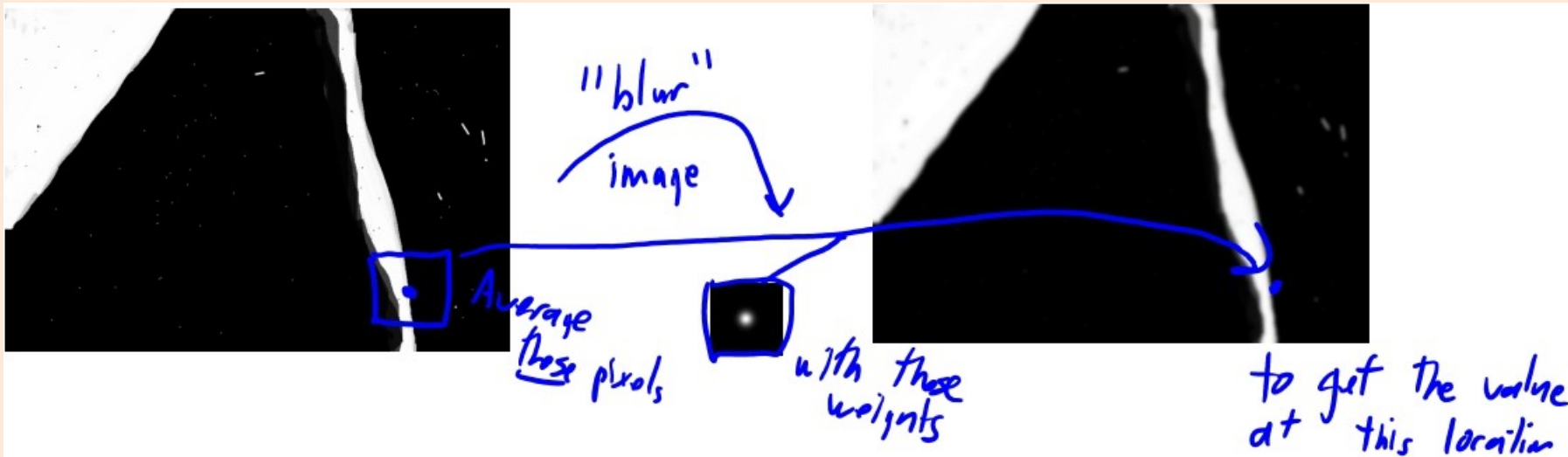
- How do you “blur”?
 - Take **weighted average of window**, putting more “weight” on “close” pixels:

$$z = \sum_{k \in \text{nei}} w_k x_k$$

w_k → weight on pixel x_k (averaging is special case where all pixels get equal weight)

Convolution

- Taking a “**weighted average of neighbours**” is called “**convolution**”.
 - Gives you a new (transformed) feature for each pixel



$$z = \sum_{k \in \text{nei}} w_k x_k$$

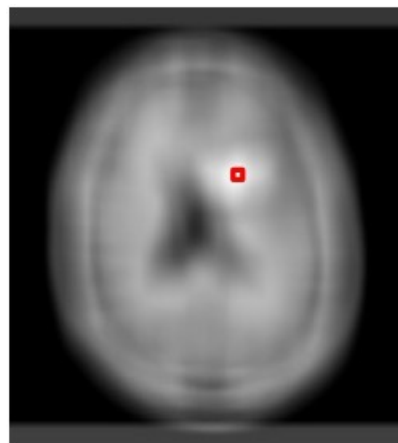
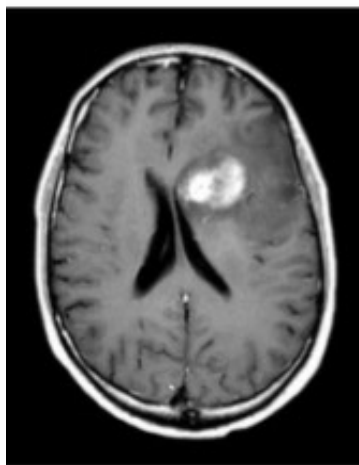
w_k → weight on pixel x_k (averaging is special case where all pixels get equal weight)

Convolution

- Taking a “**weighted average of neighbours**” is called “**convolution**”.
 - Gives you a new (transformed) feature for each pixel

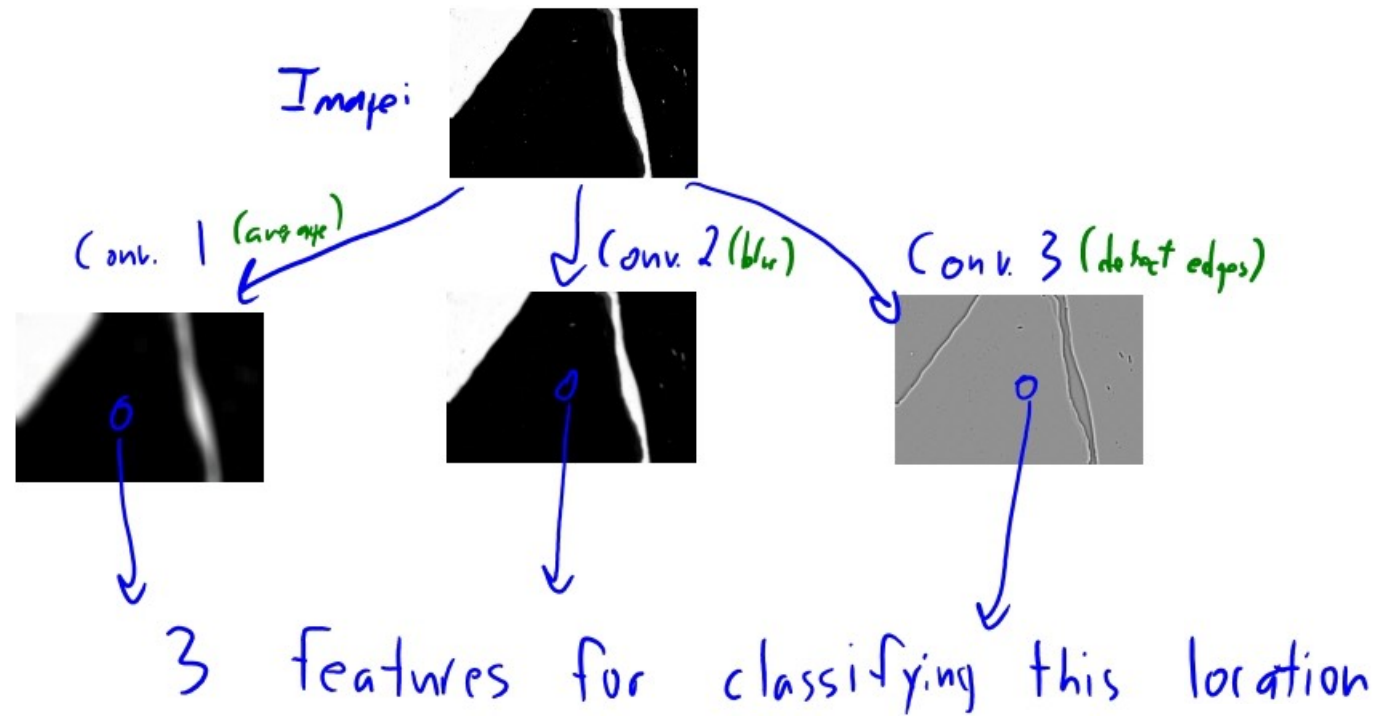
$$z = \sum_{k \in \text{neoi}} w_k x_k$$

w_k → weight on pixel *k*. (averaging is special case where all pixels get equal weight)



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 these different image values to give features** at each location.

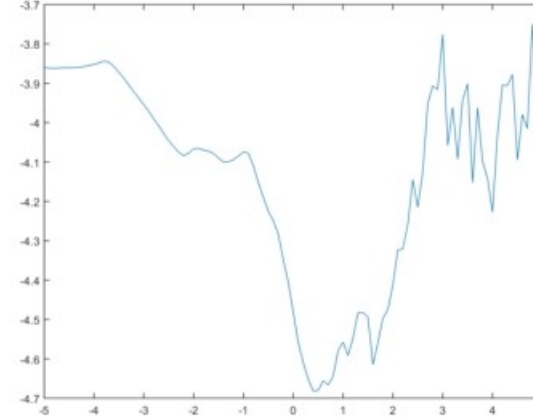


Convolutions: Big Picture

- What can features coming from convolutions represent?
 - Some filters give you an **average value of the neighbourhood**.
 - Some filters detect edges (directionally) (first derivative)
 - “Is there a change from dark to bright?”
 - “If so, from which direction in space?”
 - Some filters detect lines (“second derivative”)
 - “Is there a spike or is the change speeding up?”

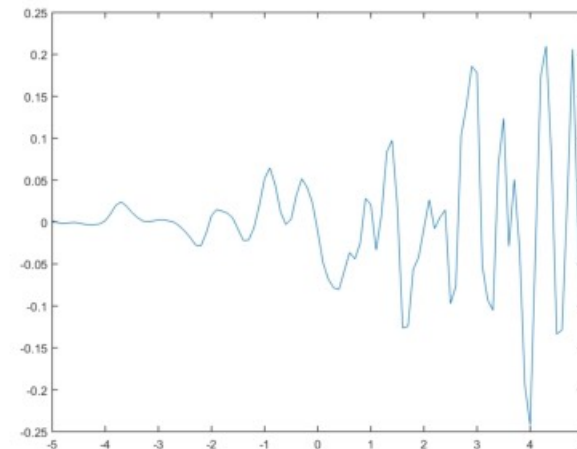
1D Convolution Example

- 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.



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

- 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 $i = 1, 2, \dots, n$

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

- Filter ‘w’ which is a vector of length ‘ $2m+1$ ’:

- Indexed by $i = -m, -m+1, \dots, -2, 0, 1, 2, \dots, m-1, m$

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

w_{-2} w_{-1} w_0 w_1 w_2

- 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':



- Filter 'w':



- Convolution 'z':



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”

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

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

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

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

- “Translation”

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

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

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

1D Convolution Examples

- Examples:

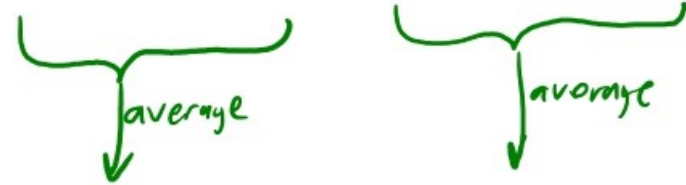
- “Identity”

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

- “Local Average”

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

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



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

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

- 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** :

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

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')

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

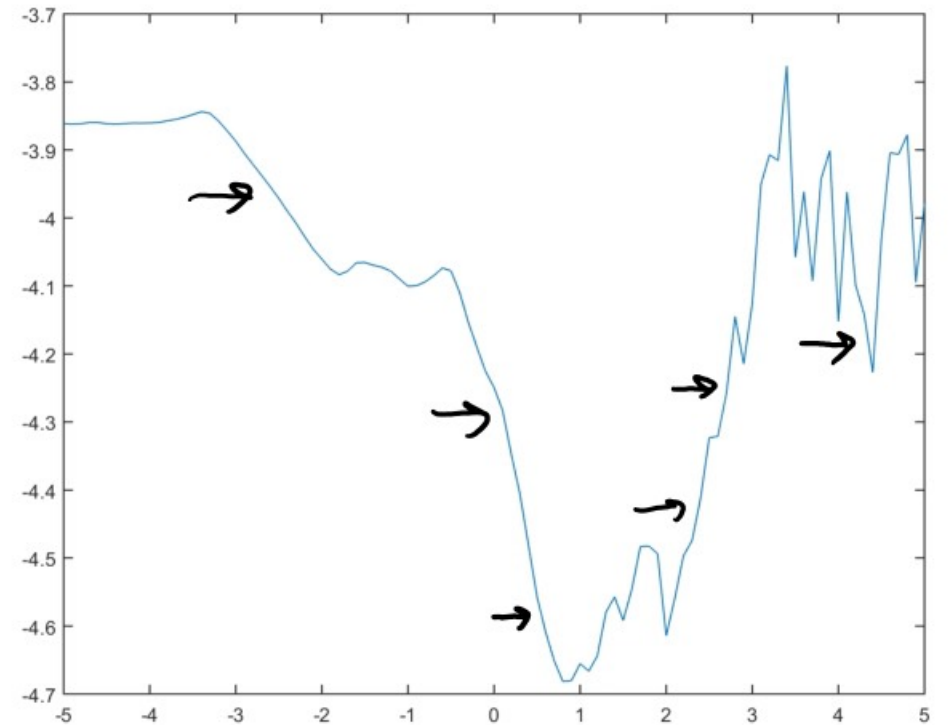
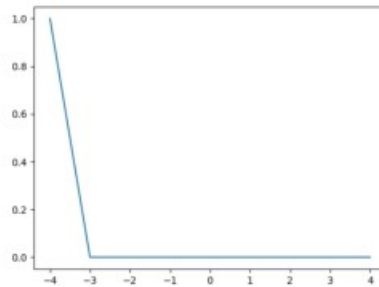
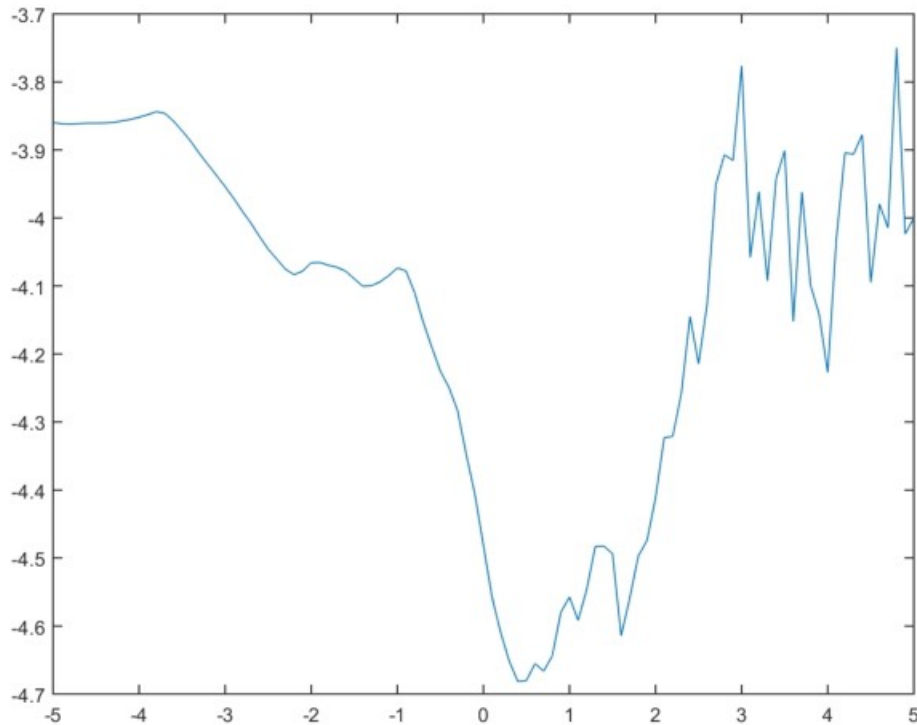
(assumes signal + filter are continuous)

- 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?”

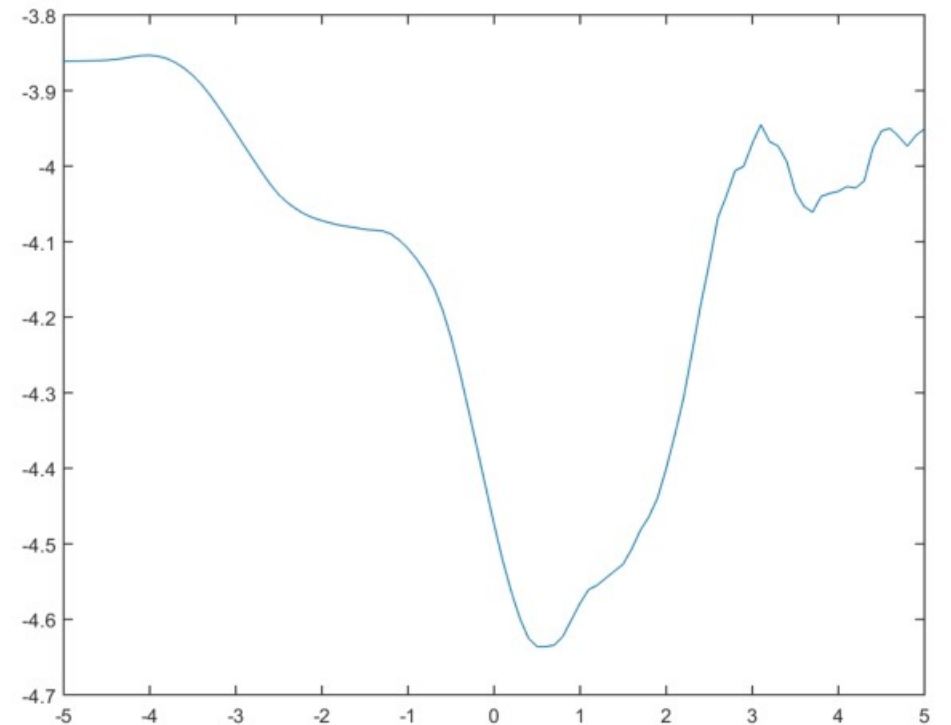
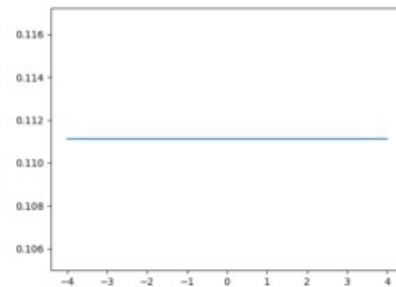
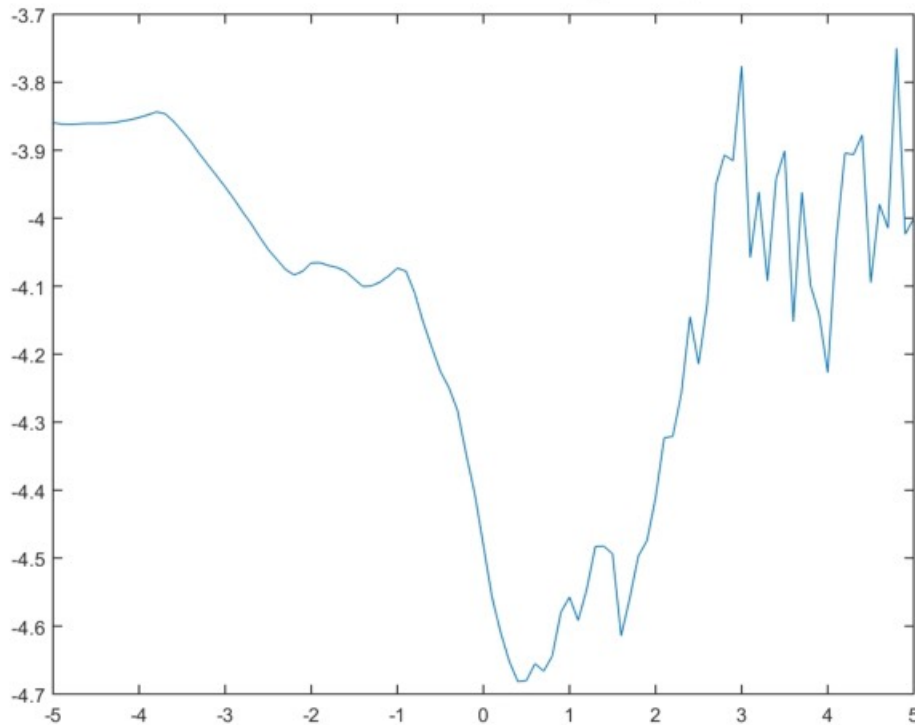
$$w = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$$



1D Convolution Examples

- **Averaging** convolution (“general value of signal in this region?”)
 - **Less sensitive to noise** (or spikes) than raw signal.

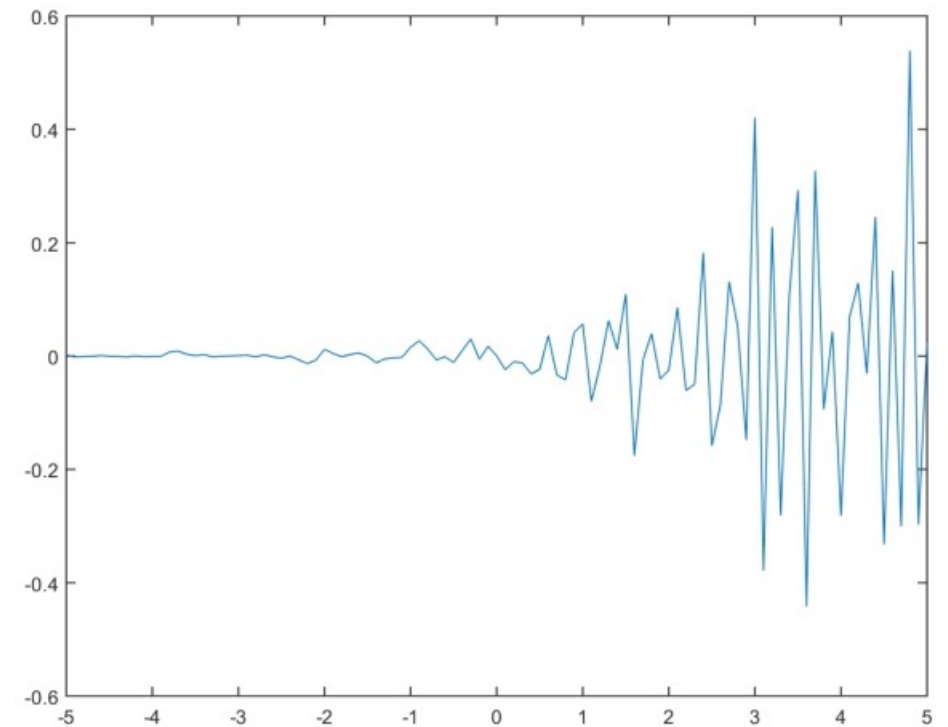
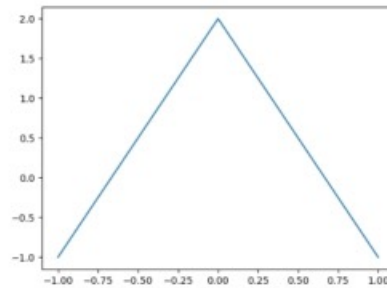
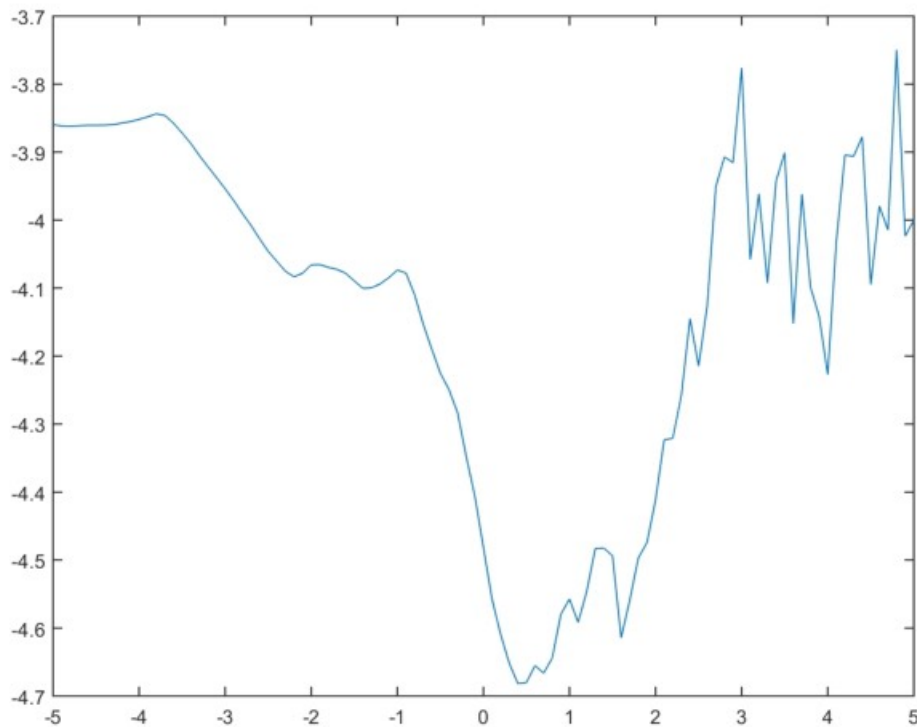
$$w = \left[\frac{1}{9} \quad \frac{1}{9} \quad \frac{1}{9} \quad \frac{1}{9} \quad \frac{1}{9} \quad \frac{1}{9} \quad \frac{1}{9} \quad \frac{1}{9} \quad \frac{1}{9} \right]$$



1D Convolution Examples

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

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

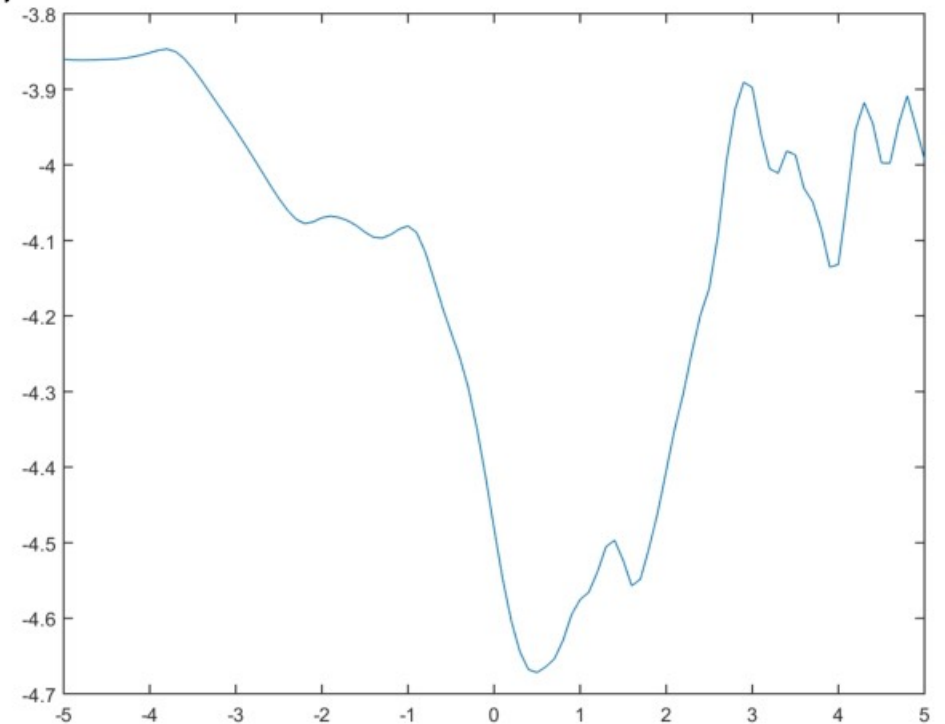
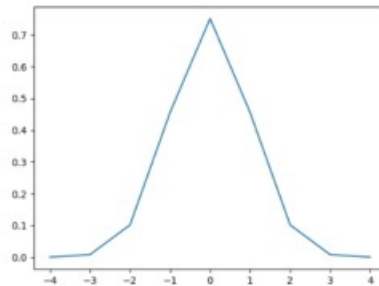
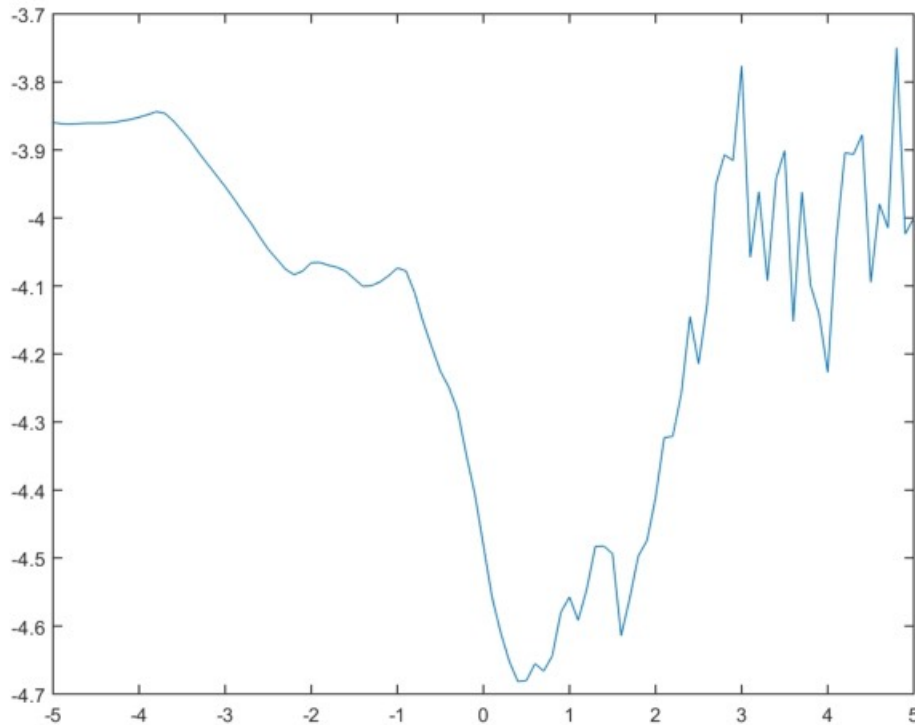


1D Convolution Examples

- **Gaussian** convolution (“blurring”): $w_i \propto \exp(-\frac{i^2}{2\sigma^2})$
 - Compared to averaging it’s more smooth and maintains peaks better.

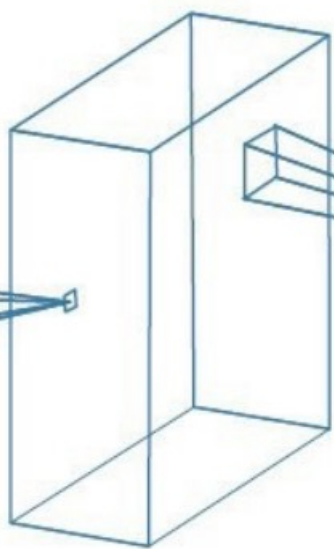
$$W = [0.0001 \quad 0.0644 \quad 0.0540 \quad 0.2420 \quad 0.3989 \quad 0.2420 \quad 0.0540 \quad 0.0644 \quad 0.0001]$$

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





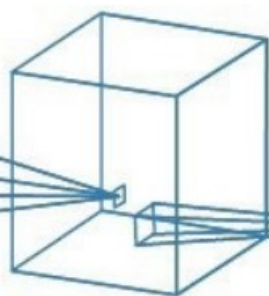
INPUT



CONVOLUTION + RELU



POOLING

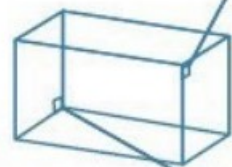


CONVOLUTION + RELU



POOLING

...



FLATTEN



FULLY CONNECTED



SOFTMAX



FEATURE LEARNING

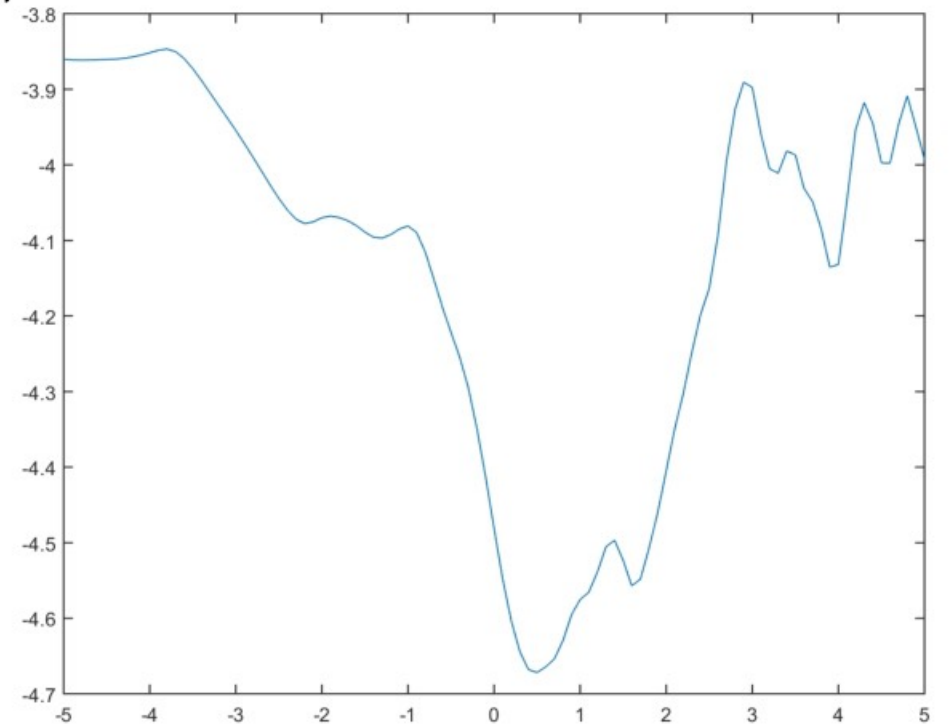
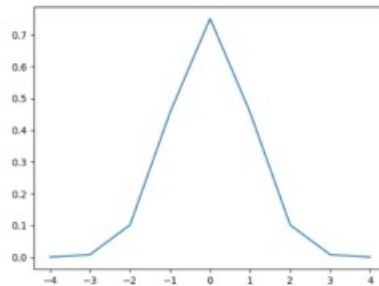
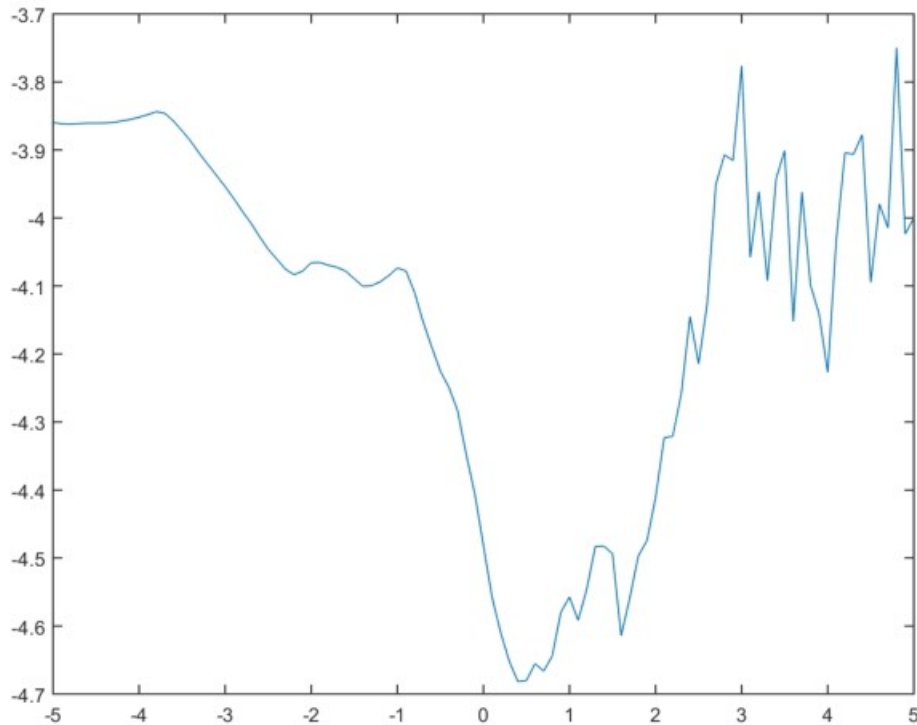
CLASSIFICATION

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.

$$W = [0.0001 \quad 0.0644 \quad 0.0540 \quad 0.2420 \quad 0.3989 \quad 0.2420 \quad 0.0540 \quad 0.0644 \quad 0.0001]$$

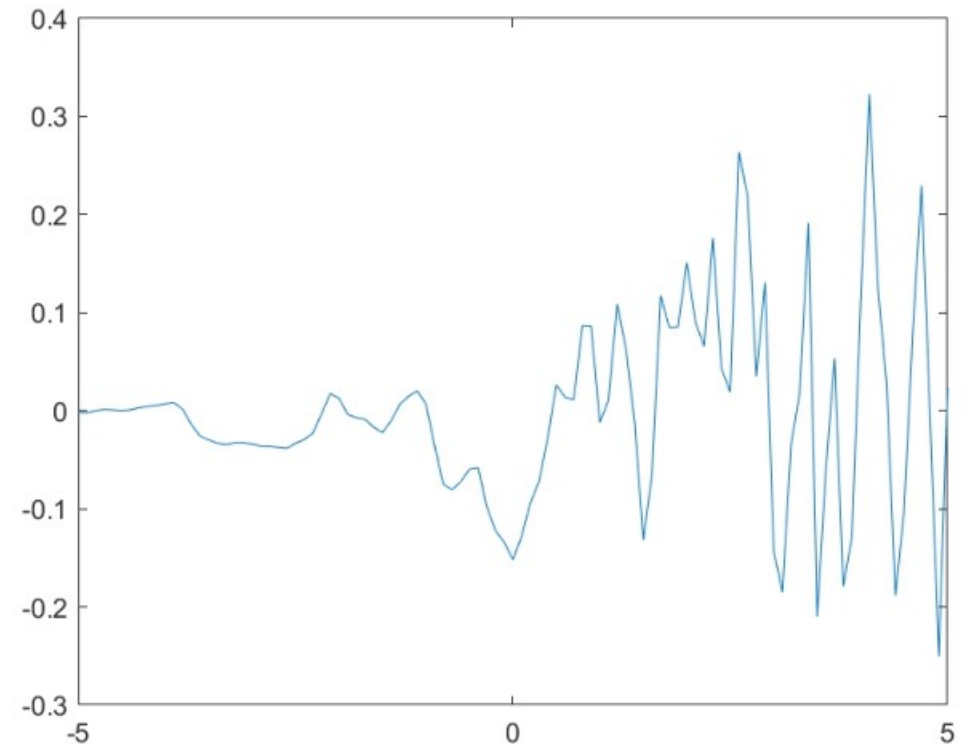
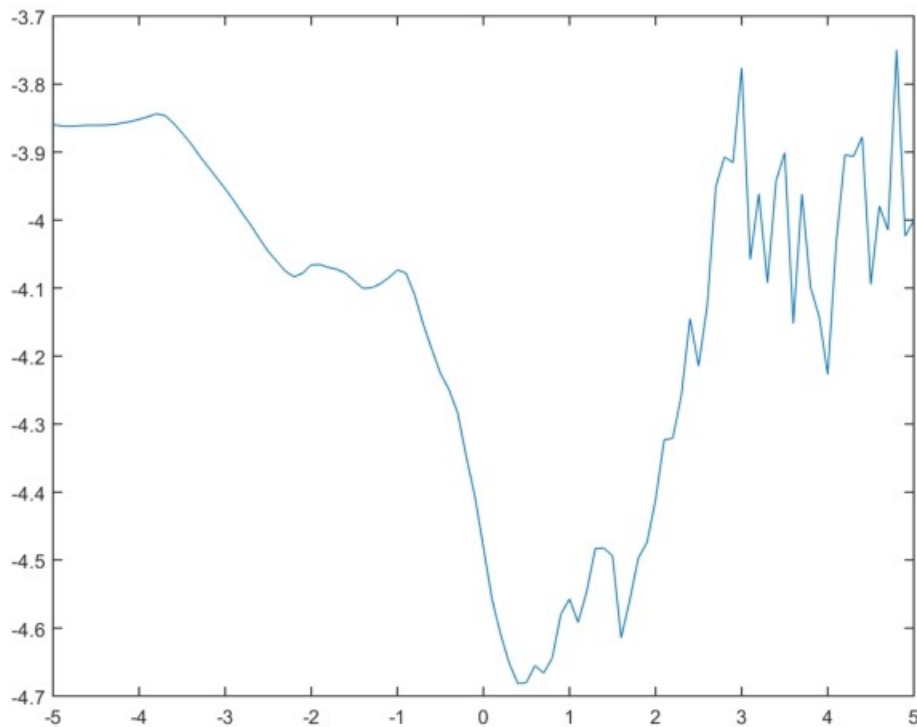
$(\sigma = 1, m = 4)$



1D Convolution Examples

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

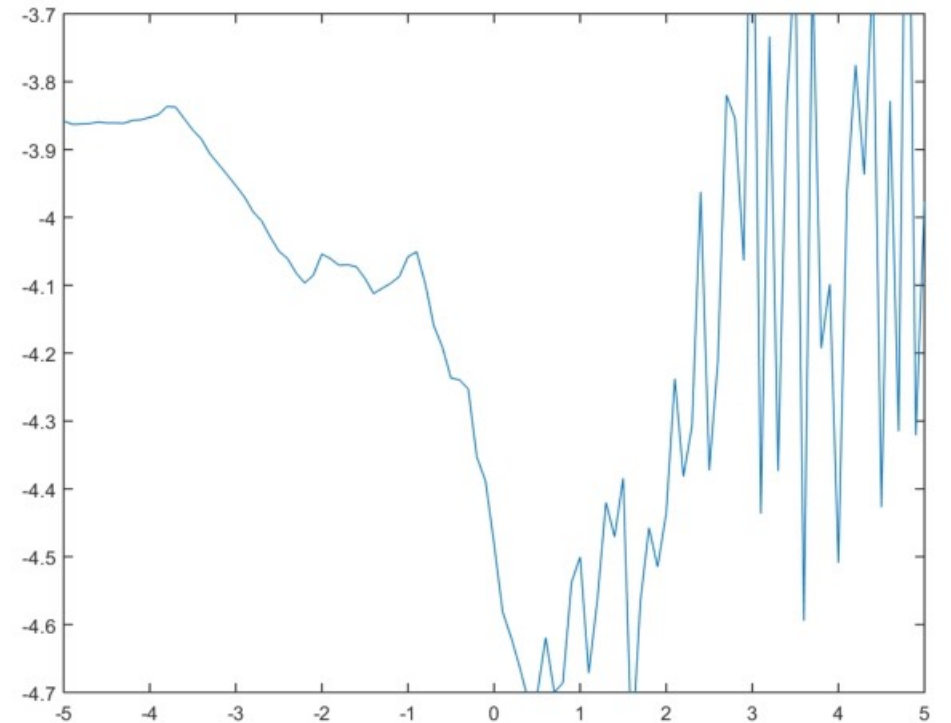
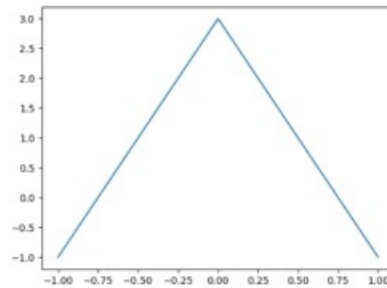
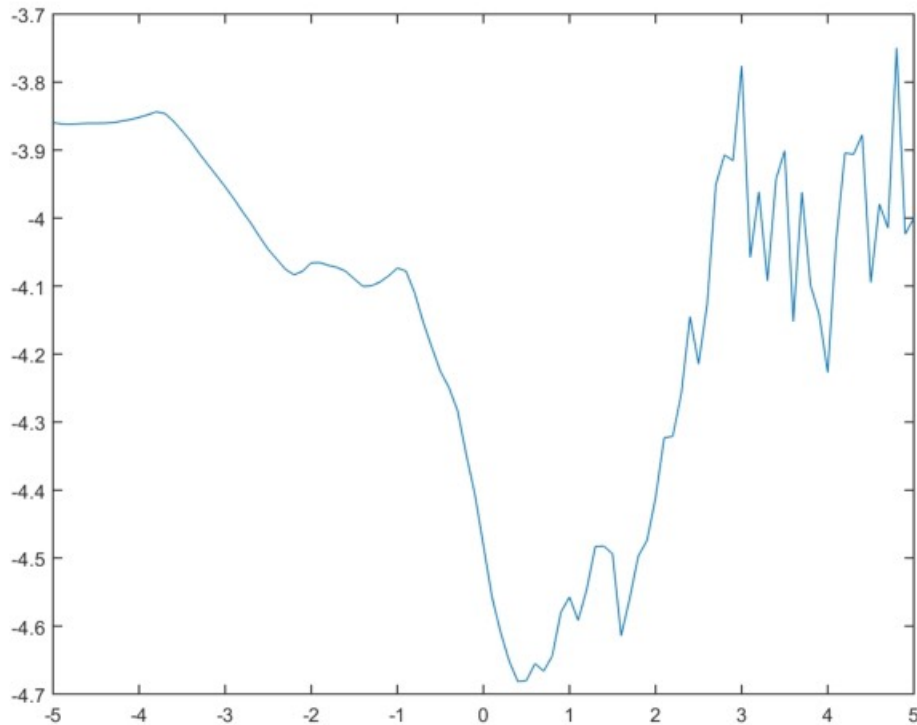
$$w = [-1 \quad 0 \quad 1]$$



1D Convolution Examples

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

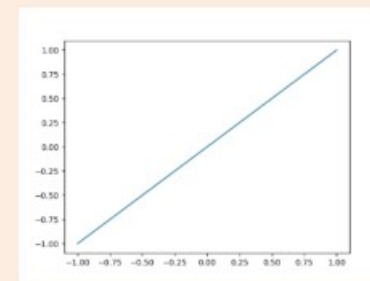
$$w = [-1 \quad 3 \quad -1]$$



Digression: Derivatives and Integrals

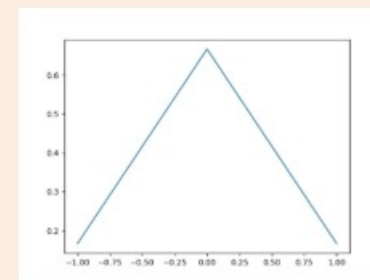
- Numerical derivative approximations can be viewed as filters:

- 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.

Laplacian of Gaussian Filter

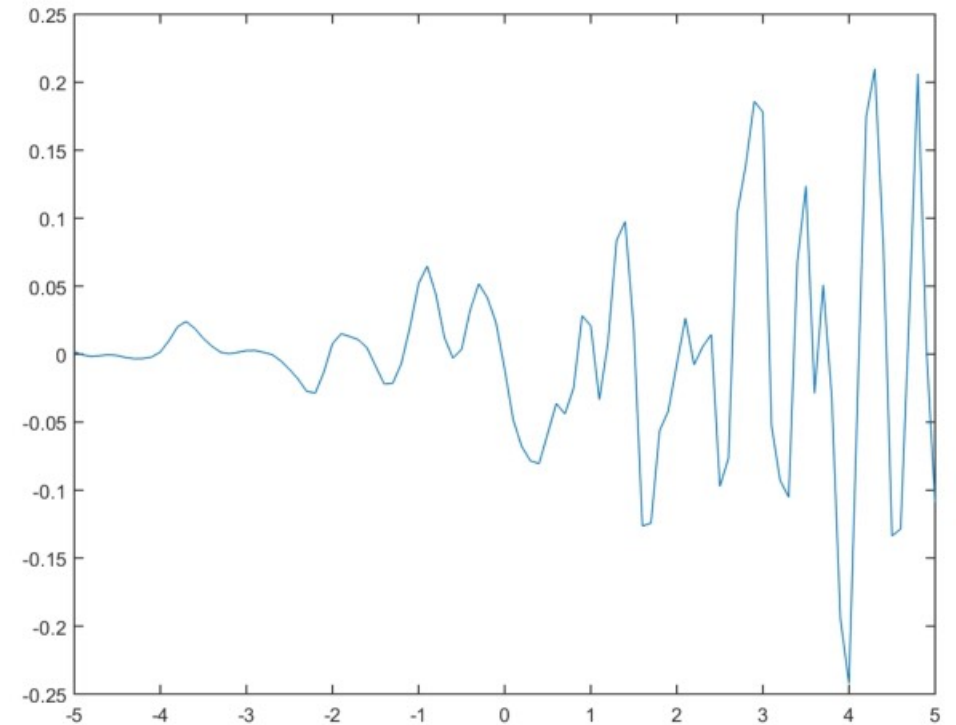
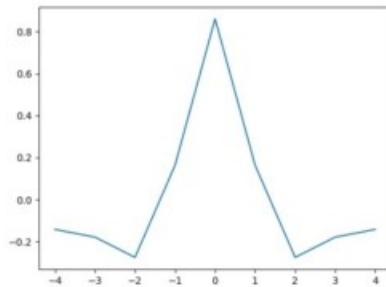
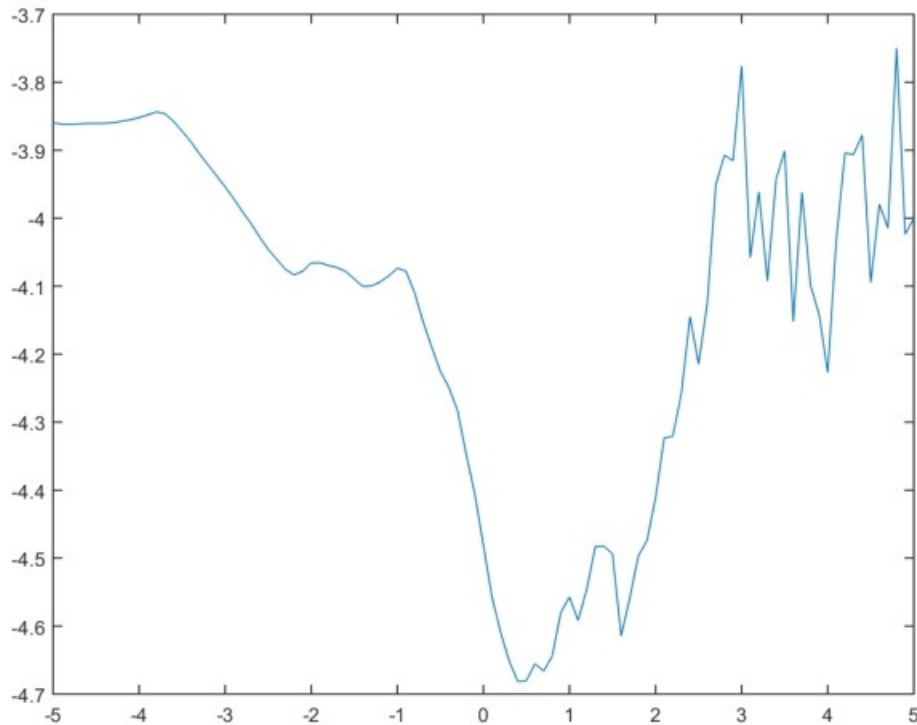
- Laplacian of Gaussian is a **smoothed 2nd-derivative** approximation:

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

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

(then subtract mean)

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



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.

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

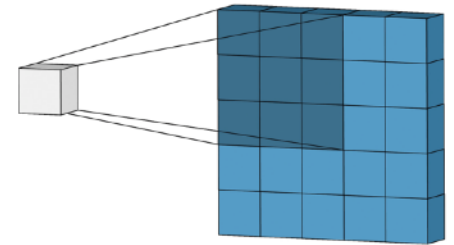
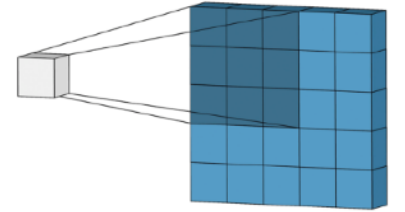
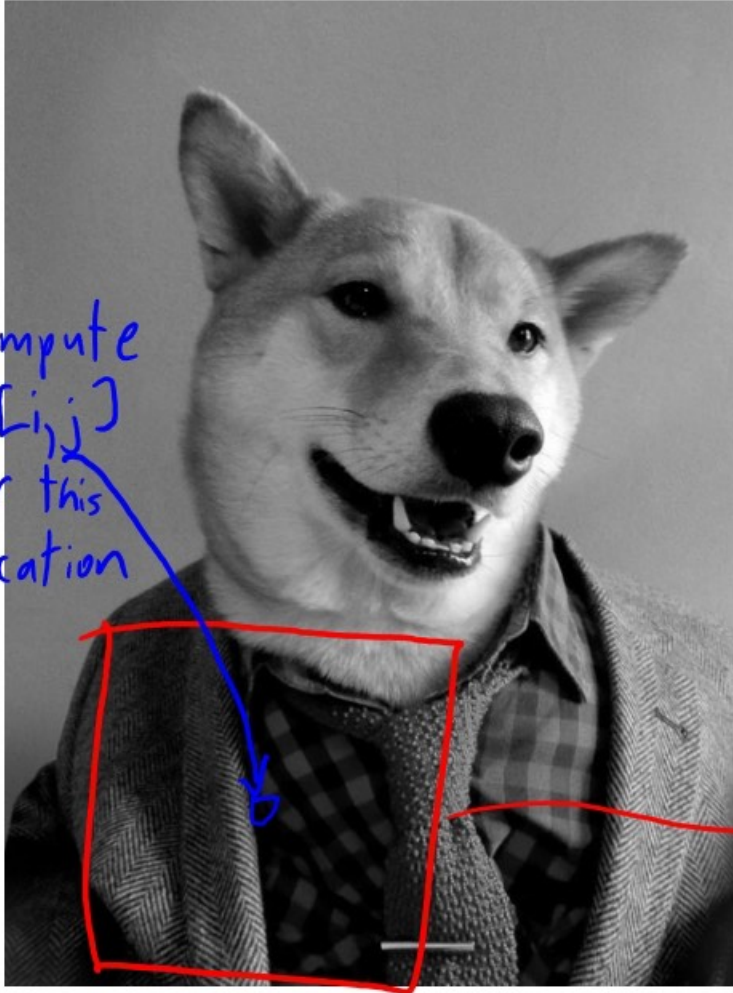


Image Convolution Examples z

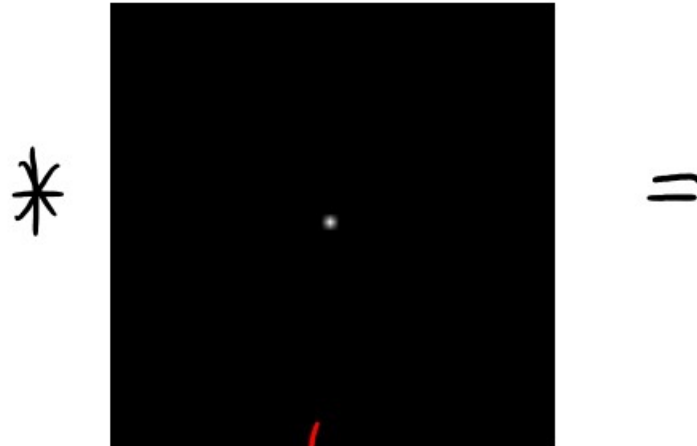
x



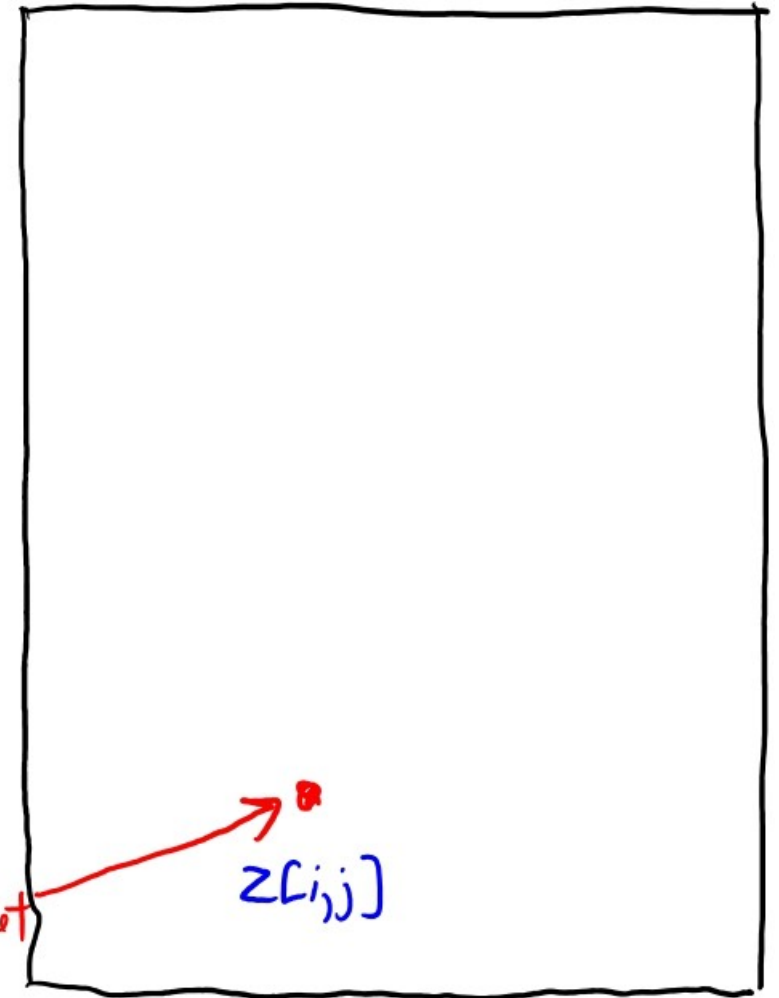
Compute $z[i,j]$ for this location

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

w



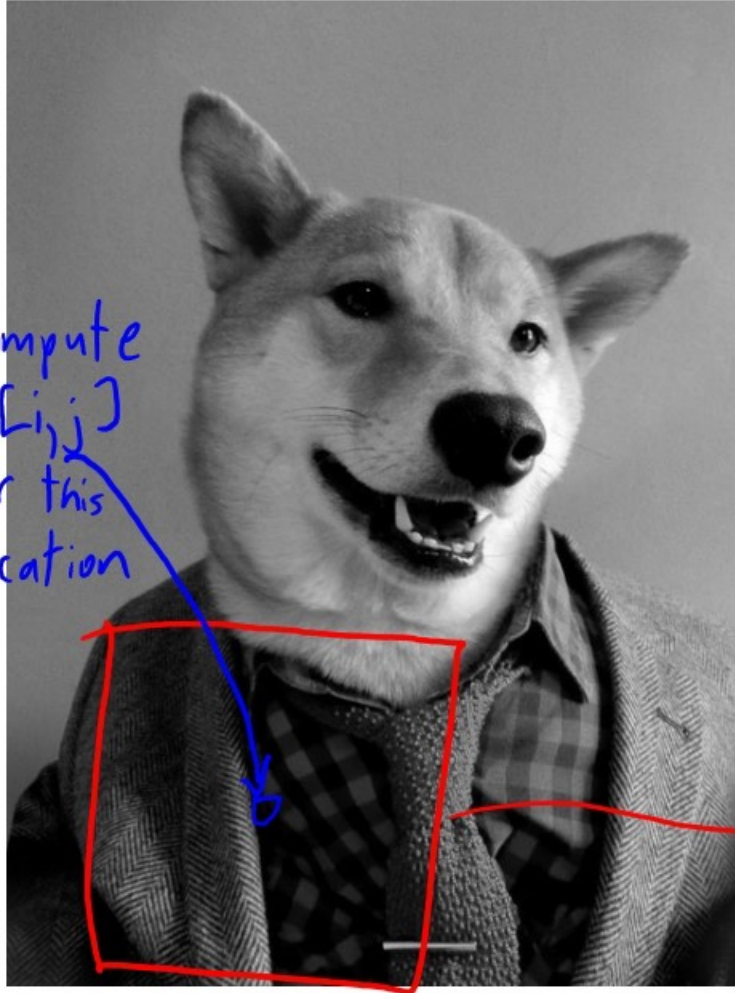
multiply element-wise
and add up result to get



$z[i,j]$

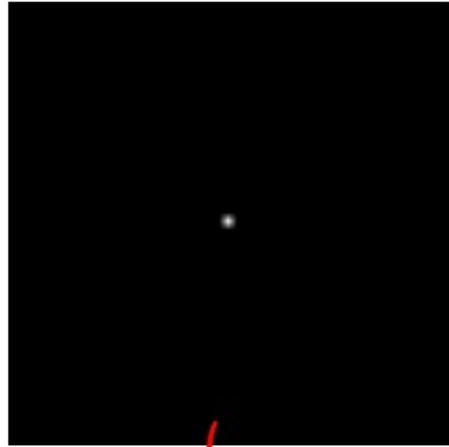
Image Convolution Examples z

x



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

w



$*$

$=$

multiply element-wise
and add up result to get

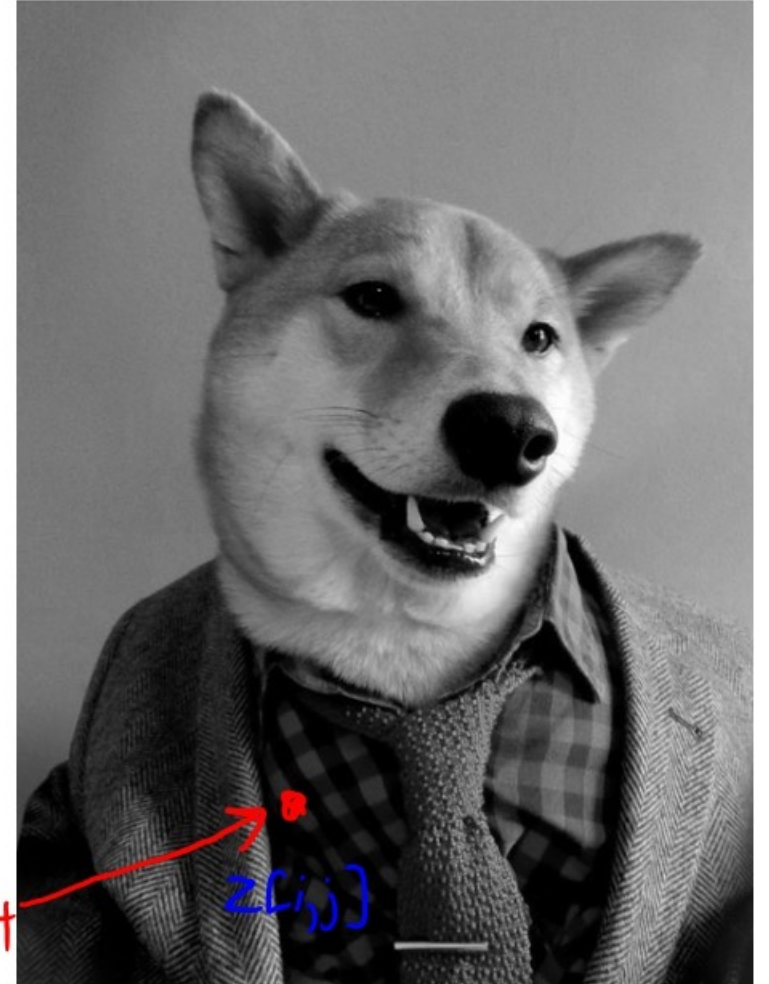


Image Convolution Examples



Translation Convolution:

$$* \begin{array}{|c|} \hline \text{0} \\ \hline \square \\ \hline \end{array} =$$

Boundary: "zero"



OK

Image Convolution Examples



Translation Convolution:

$$* \begin{array}{|c|} \hline \text{[Black Square]} \\ \hline \end{array} =$$

A diagram illustrating translation convolution. On the left, a black asterisk symbol is followed by a black square representing a kernel. The top-left corner of the square is circled in red. To the right of the square is an equals sign.

Boundary: "replicate"



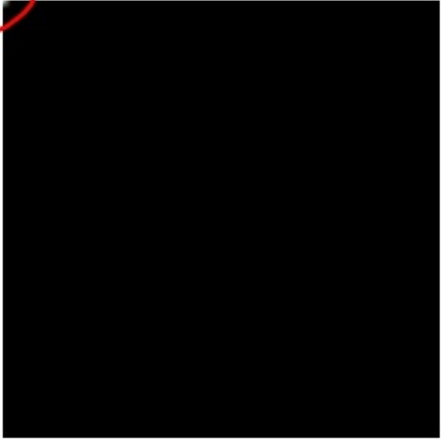

repents

repents

Image Convolution Examples



Translation Convolution:

$*$  $=$ 

Boundary: "mirror"

flips

The diagram illustrates a translation convolution operation. It shows an input image of a dog in a suit on the left, followed by a convolution kernel represented by a black square with a red circle at its top-left corner. An asterisk symbol indicates the convolution operation, followed by an equals sign and the resulting output image on the right. The output image is a mirrored version of the input image. Below the diagram, the text "Boundary: 'mirror'" and "flips" with green arrows pointing to the mirrored image, explains the boundary handling used.

Image Convolution Examples

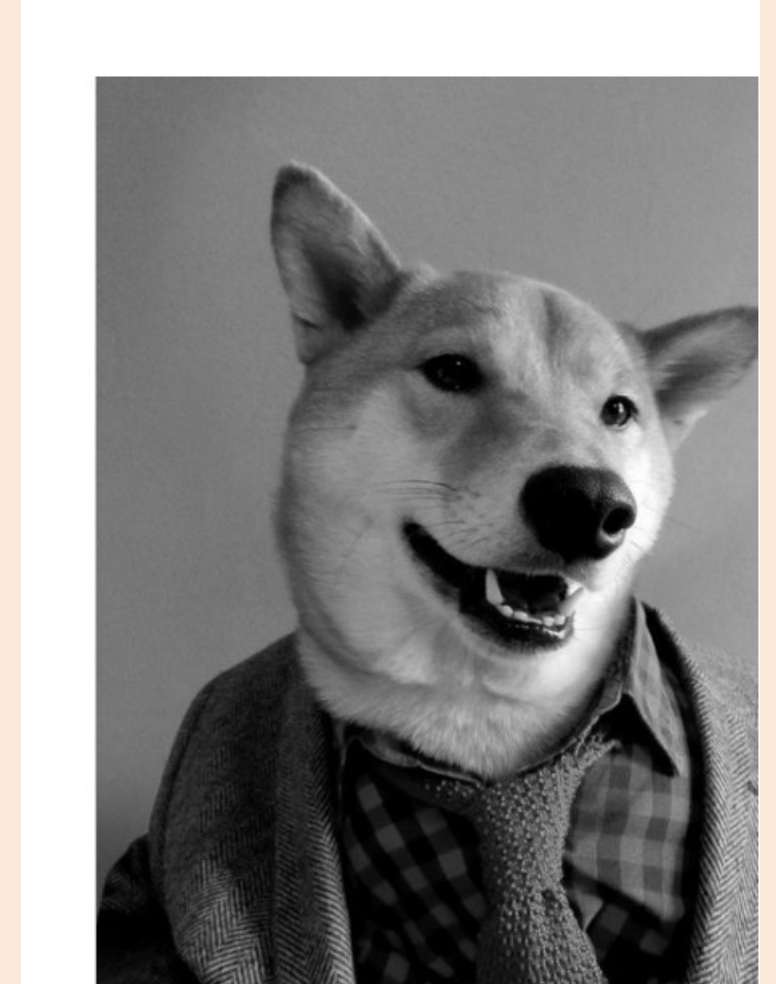
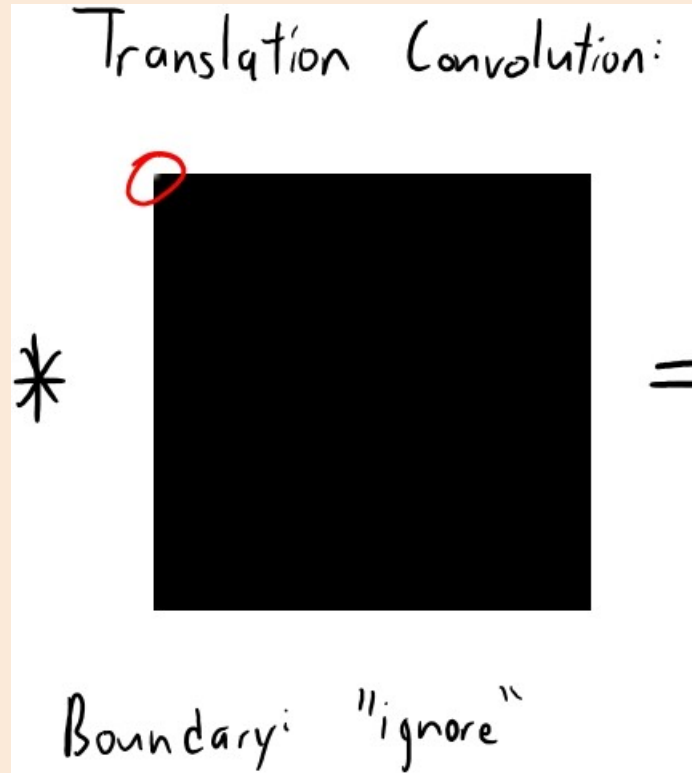


Image Convolution Examples



Average convolution:

$$* \frac{1}{51} \left[\begin{array}{c} | | | | | \\ | | | | | \\ | | | | | \\ | | | | | \\ | | | | | \end{array} \right] =$$

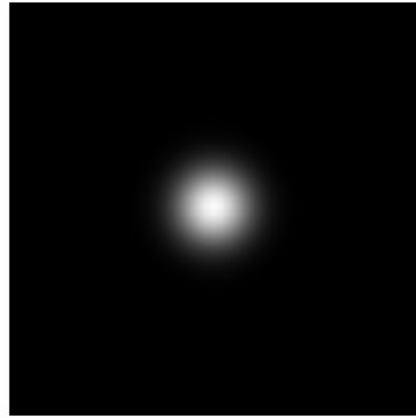


Image Convolution Examples



Gaussian Convolution:

*



=

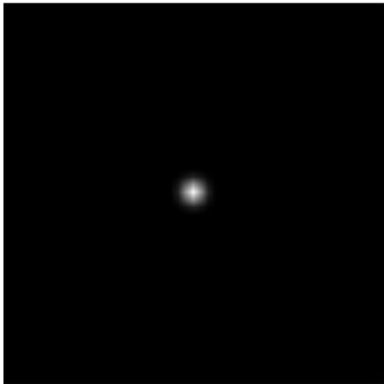
blurs image to represent
average
(smoothing)



Image Convolution Examples



Gaussian Convolution:

$*$  $=$

(smaller variance)


blurs image to represent
average
(smoothing)



Image Convolution Examples



Laplacian of Gaussian

$*$  $=$

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

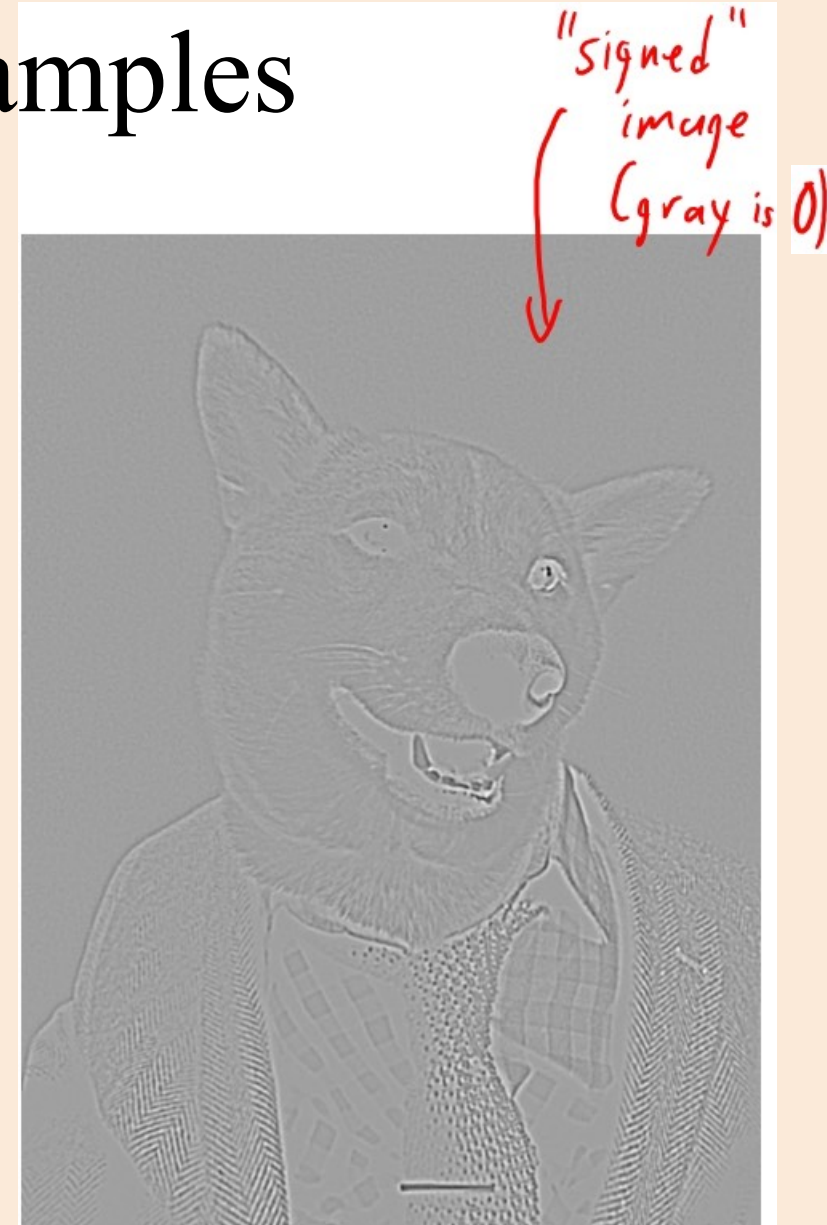


Image Convolution Examples



Laplacian of Gaussian

$$* \text{ [Gaussian Kernel] } =$$

(larger variance)

The equation shows a convolution operation between an asterisk and a Gaussian kernel (a dark spot in the center of a light gray square) followed by an equals sign. Below the kernel, the text "(larger variance)" is written in a handwritten style.

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.

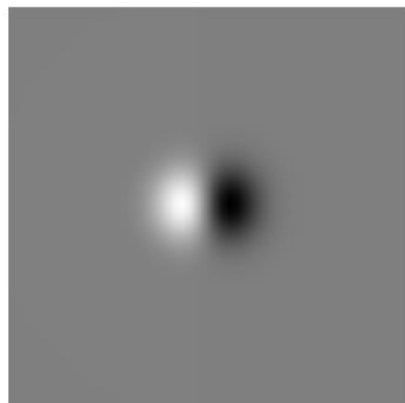


Image Convolution Examples



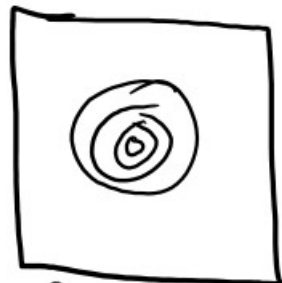
Gabor filter
(Gaussian multiplied by
sine or cosine)

*



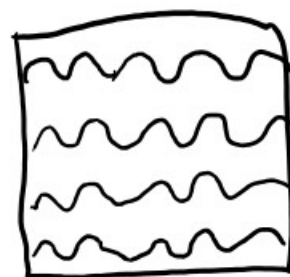
=

//



Gaussian

*



Parallel Sine functions

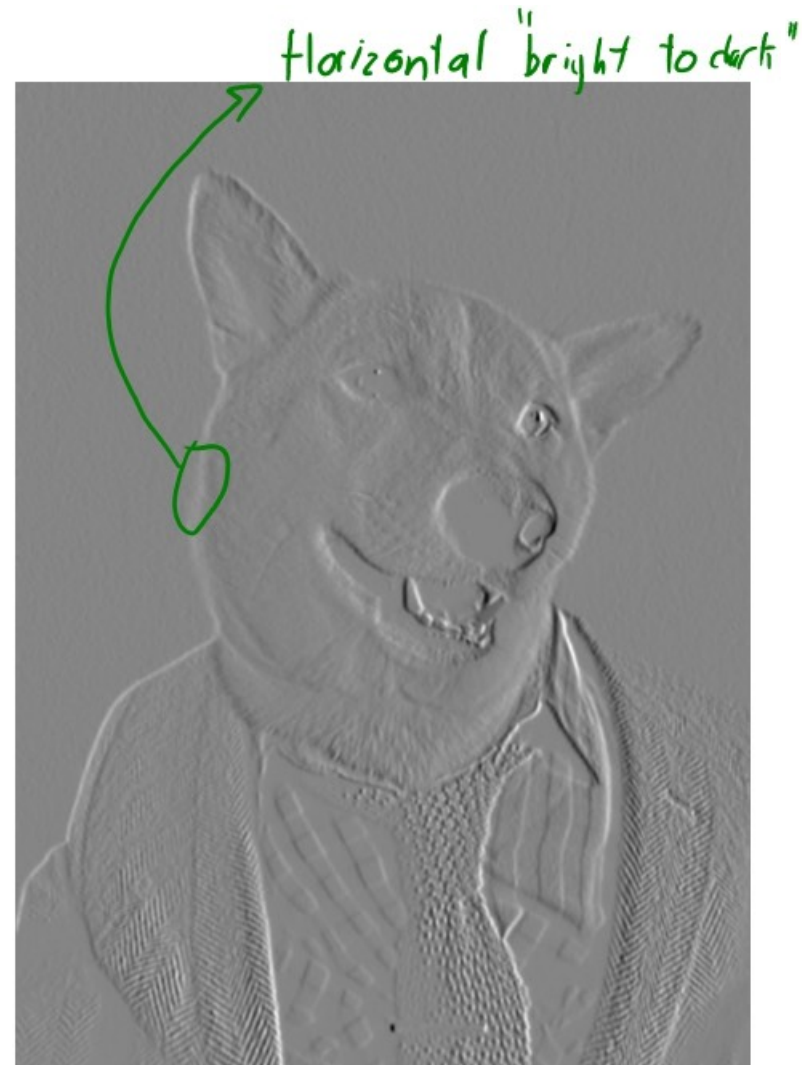
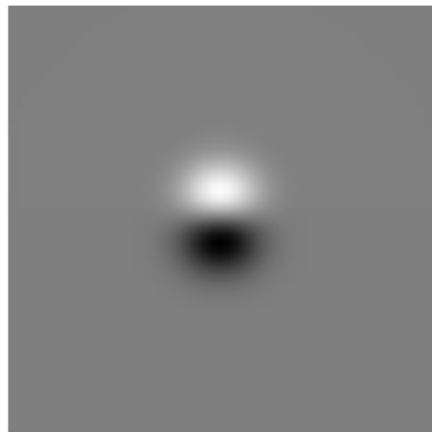


Image Convolution Examples

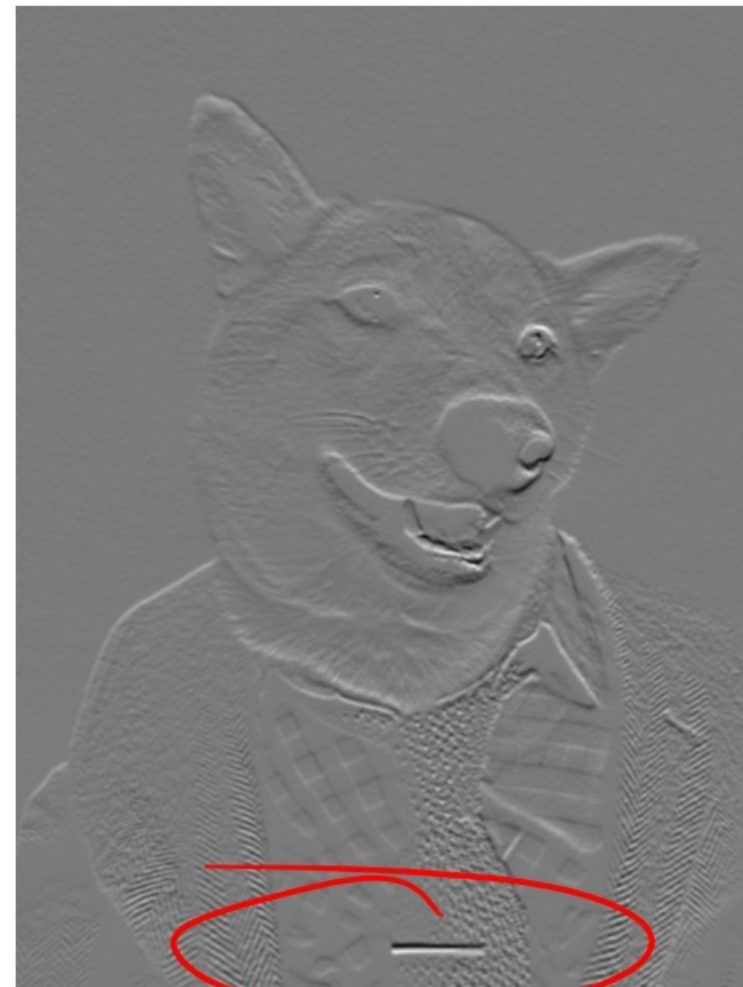


Gabor Filter
(Gaussian multiplied by
sine or cosine)

*



=



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



→ 2d derivatives have a direction.

We stopped here