CPSC 340: Machine Learning and Data Mining

Convolutions

Fall 2019

Last Time: “Global” and “Local” Features

A4Q3: dimensions of 'W' updated

- Consider the following weird feature transformation for identifying important e-mails:

“CPSS”	“340”
1	0
1	0
1	1
0	0
1	1

⇒

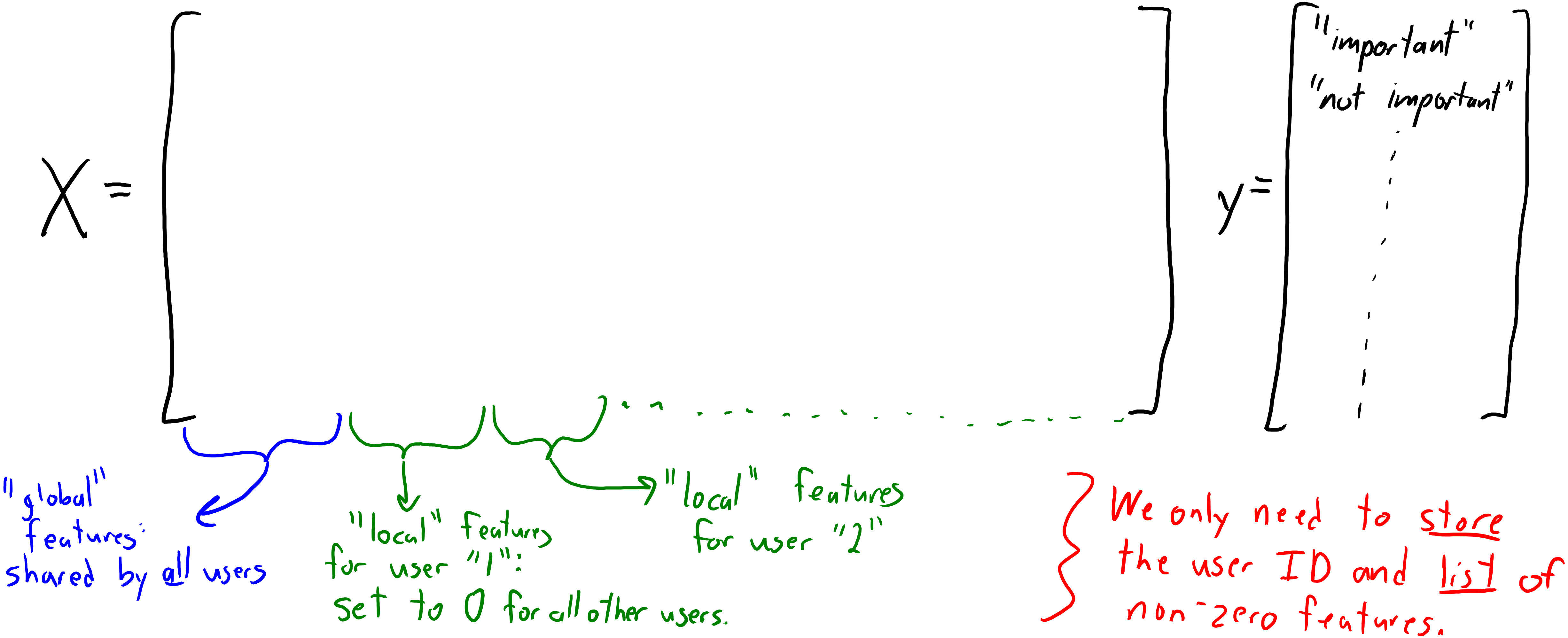
“CPSC” (any user)	“340” (any user)	“CPSC” (user?)	“340” (user?)
1	0	User 1	<no “340”>
1	0	User 1	<no “340”>
1	1	User 2	User 2
0	0	<no “CPSC”>	<no “340”>
1	1	User 3	User 3

- The categorical (user?) features get expanded out into ‘k’ binary features.
  - Where ‘k’ is the number of users.
  - All those features are set to 0 if the word was not used.

- “Any user” (“global”) features increase/decrease importance of word for every user.
- “User” (“local”) features increase/decrease importance of word for specific users.
  - Lets us learn more about users where we have a lot of data

The Big Global/Local Feature Table for E-mails

- Each row is one e-mail (there are lots of rows):


X =
y =
"important"
"not important"
"global" features: shared by all users
"local" features for user "1": set to 0 for all other users.
"local" features for user "2"
We only need to store the user ID and list of non-zero features.

Predicting Importance of E-mail For New User

- Consider a new user:
  - We start out with no information about them.
  - So we use global features to predict what is important to a generic user.

$$\hat{y}_i = \text{sign}(\underbrace{w_g^T x_{ig}}) \rightarrow \text{features/weights } \underline{\text{shared}} \text{ across users.}$$

  - Local features are initialized to zero.
- With more data, update global features and user's local features:
  - Local features make prediction *personalized*.

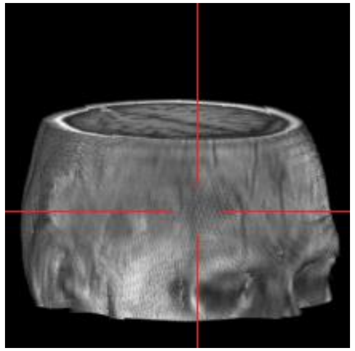
$$\hat{y}_i = \text{sign}(w_g^T x_{ig} + \underbrace{w_u^T x_{iu}}) \rightarrow \text{features/weights } \underline{\text{specific}} \text{ to user.}$$

  - What is important to *this* user?
- G-mail system: classification with logistic regression.
  - Trained with a variant of stochastic gradient (later).

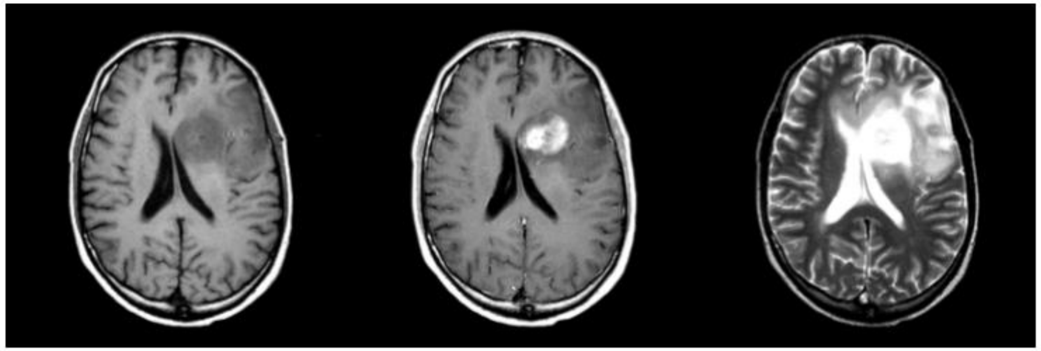
(pause)

Motivation: Automatic Brain Tumor Segmentation

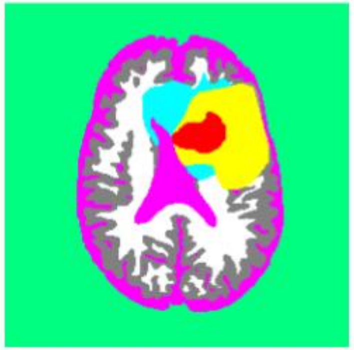
- Task: segmentation tumors and normal tissue in multi-modal MRI data.



Input:



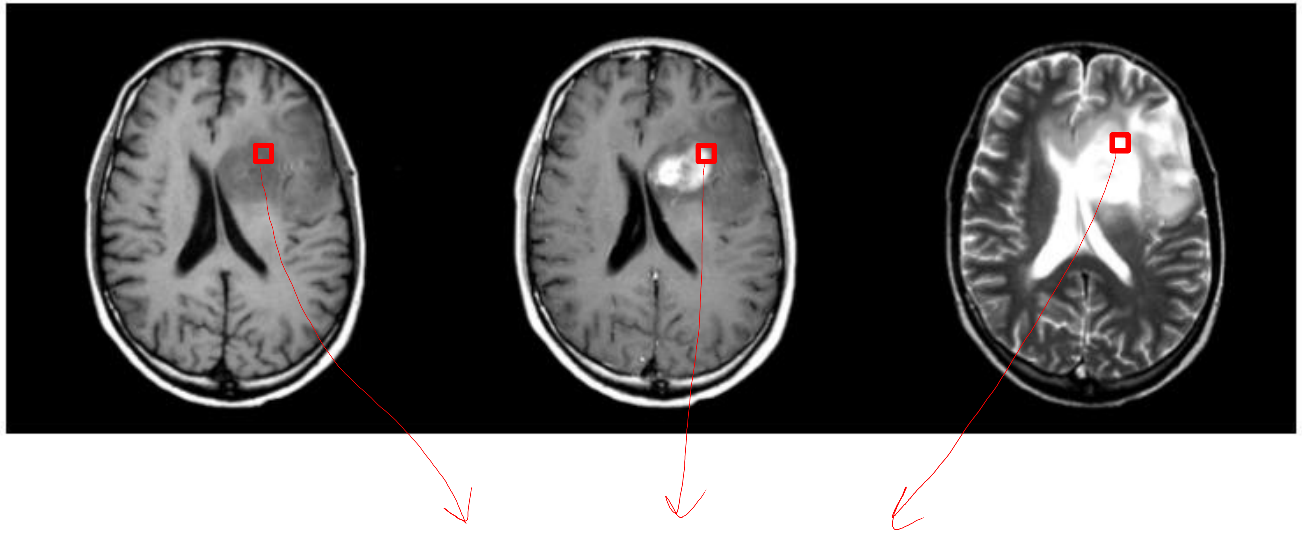
Output:



- 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:

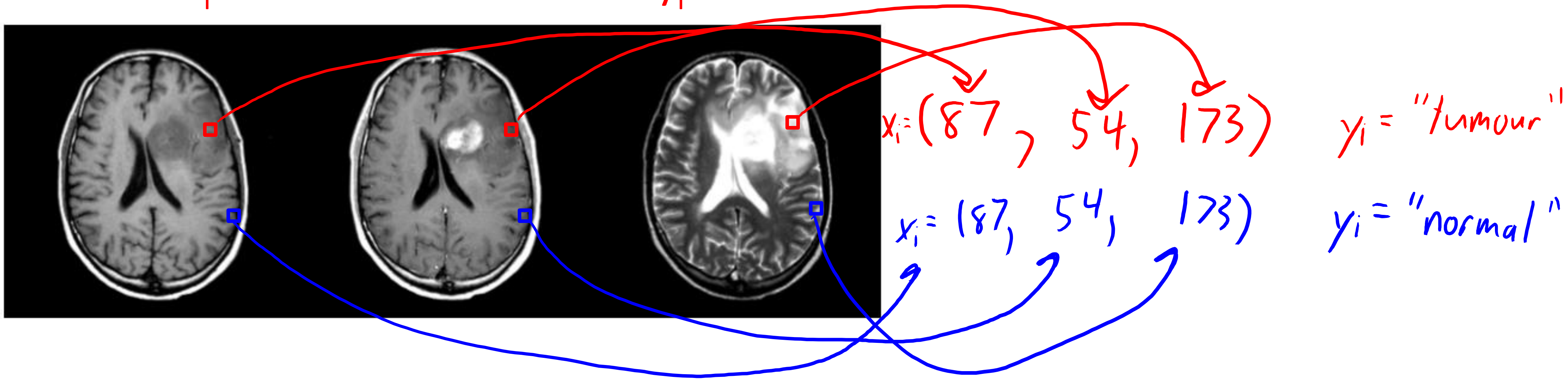


$x_i = (98, 187, 246)$ $y_i =$ "tumour"

- 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.



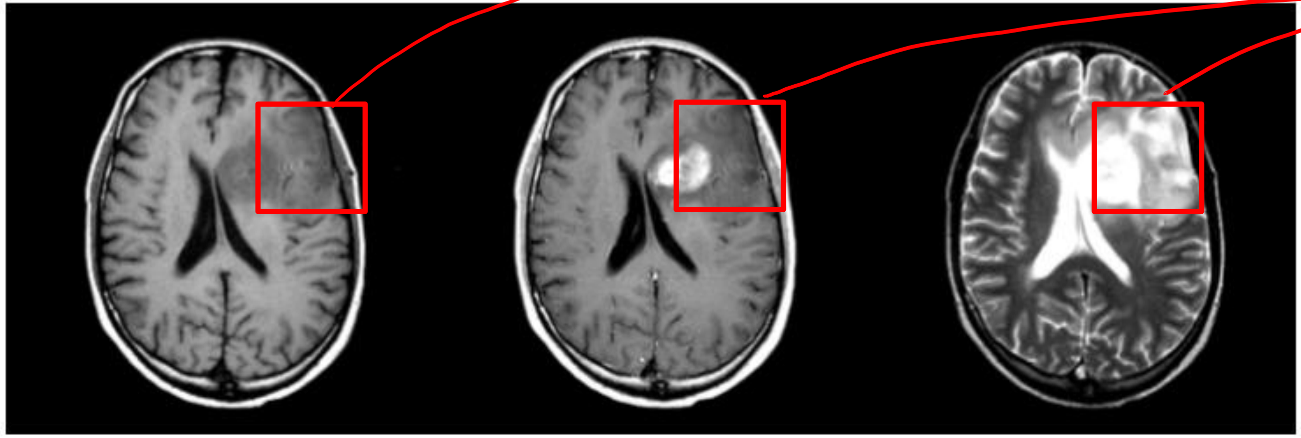
$x_i = (87, 54, 173)$ $y_i =$ "tumour"

$x_i = (87, 54, 173)$ $y_i =$ "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 spatial "context" of the voxel.

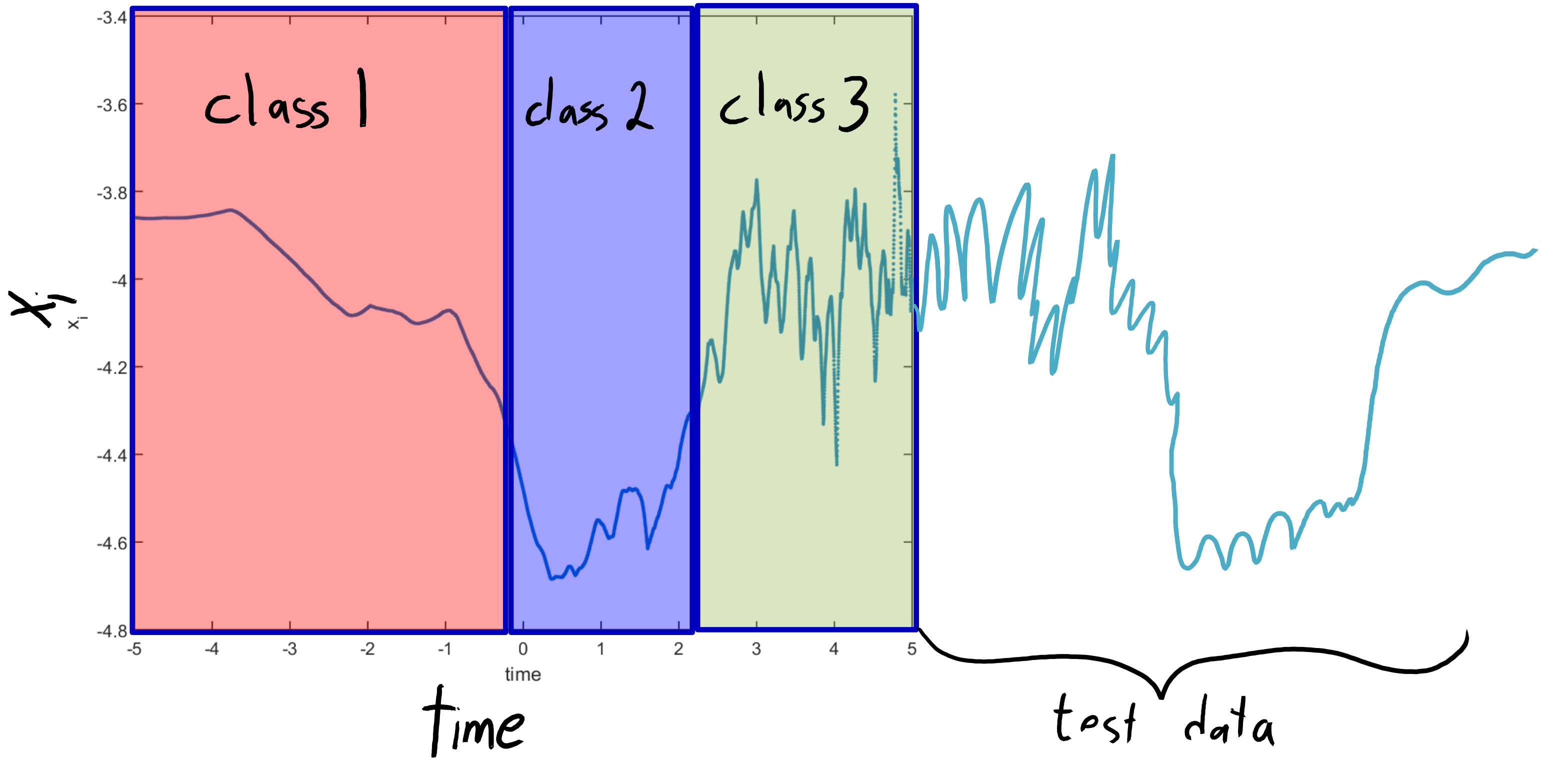


$x_i = (\overbrace{\cdots\cdots}, \overbrace{\cdots\cdots}, \overbrace{\cdots\cdots})$

  - Include all the values of neighbouring voxels as extra features?
    - 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

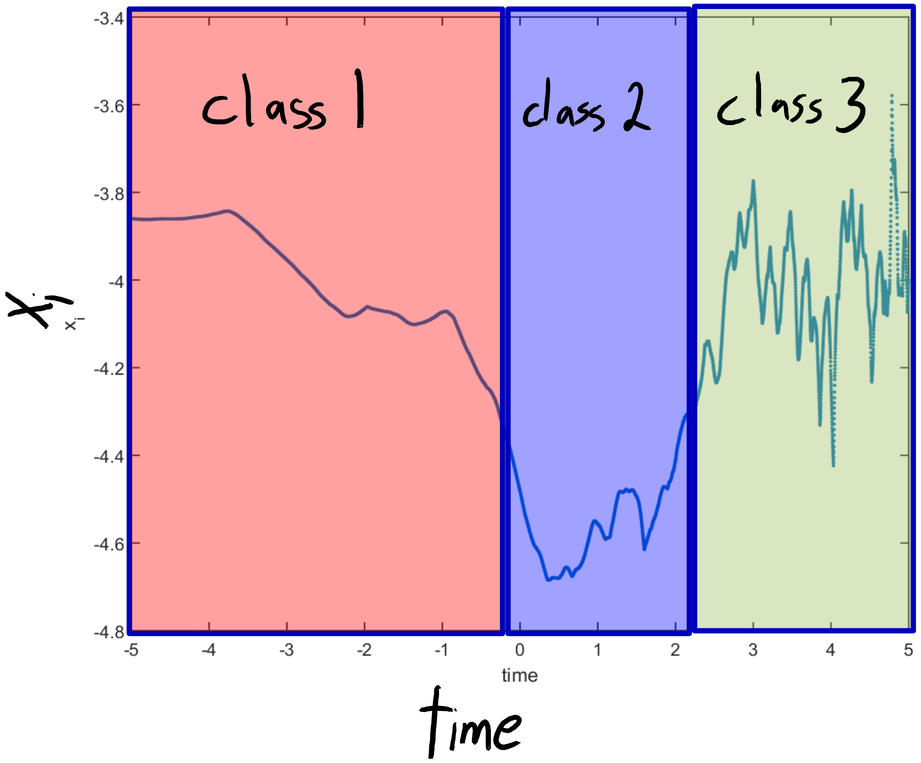
- Consider a 1D dataset:
  - Want to classify each time into y_i in {1,2,3}.
  - Example: speech data.

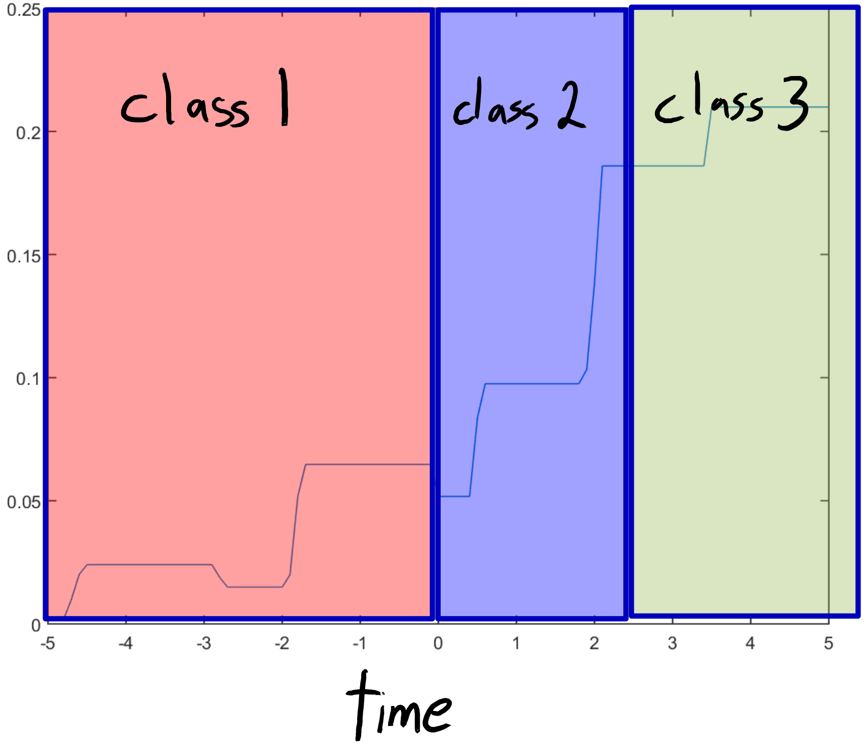


- Easy to distinguish class 2 from the other classes (x_i are smaller).
- Harder to distinguish between class 1 and class 3 (similar x_i range).
  - But convolutions can represent that class 3 is in "spiky" region.

Representing Neighbourhoods with Convolutions

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

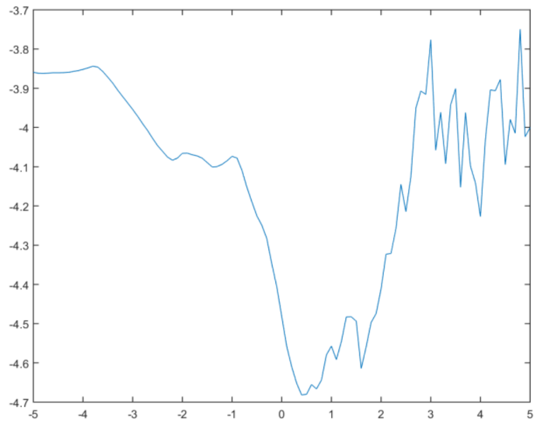




- Easy to distinguish the 3 classes with these 2 features.

1D Convolution Example

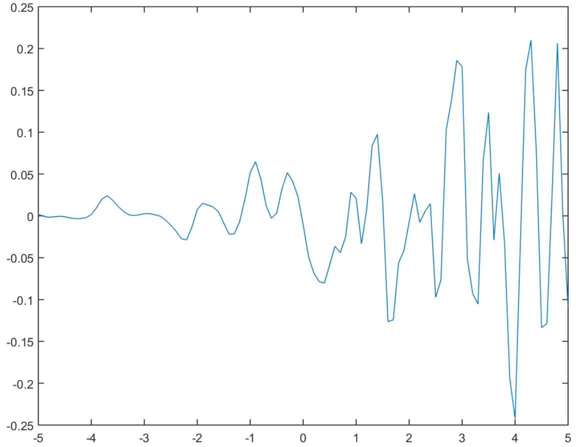
- Consider our original “signal”:



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

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

- 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,…,n.
  - Filter ‘w’ which is a vector of length ‘2m+1’:
    - Indexed by i=-m,-m+1,…-2,0,1,2,…,m-1,m

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

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

- 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:

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

  - Filter:

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

  - Convolution:

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

Let's compute z_4:

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

Dot-product: $w^T x_{i-m:i+m} = 0$

1D Convolution

- 1D convolution example:
  - Signal:

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

  - Filter:

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

  - Convolution:

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

Let's compute z_5:

$$\underbrace{[1\ \ 2\ \ 3\ \ 5\ \ 8]}$$

Dot-product: $w^T x_{i-m:i+m} = -1$

1D Convolution Examples

- Examples:
  - “Identity”

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

↳ $w = [0 \quad 1 \quad 0]$

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

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

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

  - “Translation”

↳ $w = [0 \quad 0 \quad 1]$

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

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

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

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

Boundary Issue

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

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

- 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:

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

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')

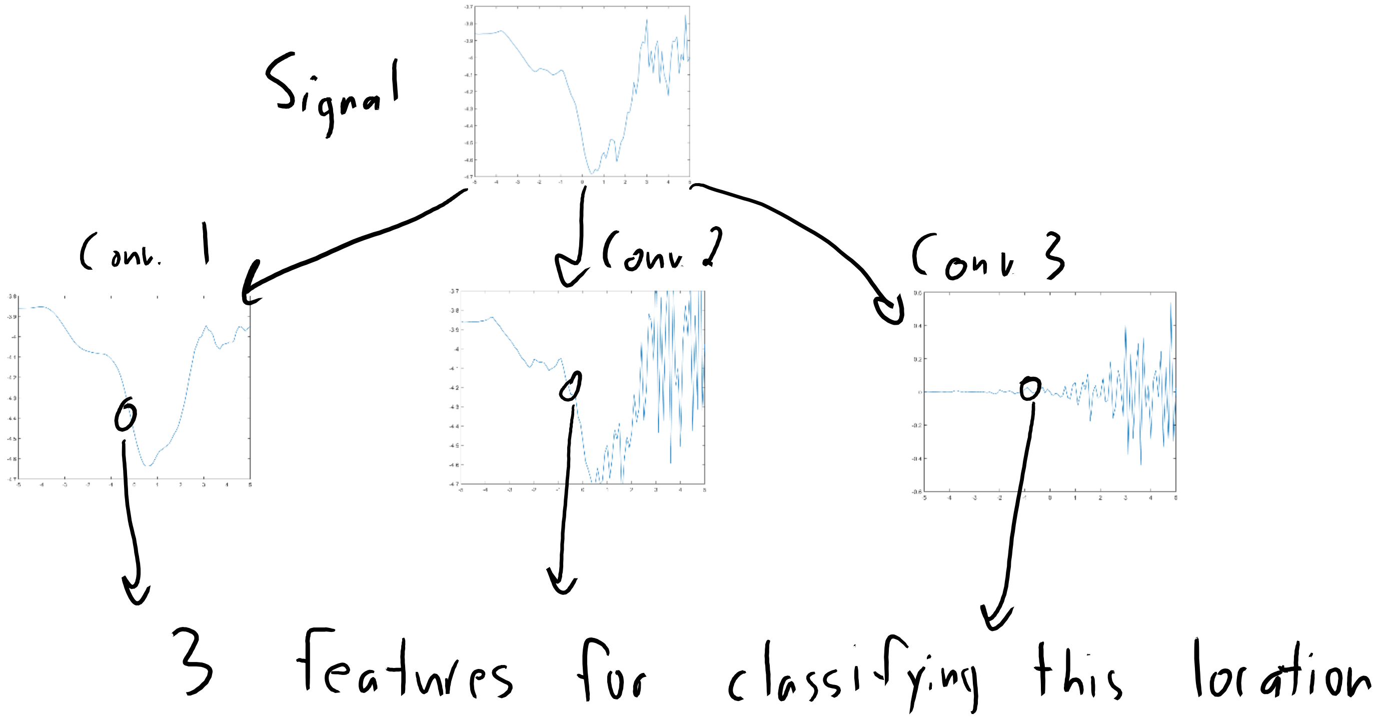
$$z_i = \int_{-\infty}^{\infty} w_j x_{i-j} \, dj$$

(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.

Convolutions: Big Picture

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



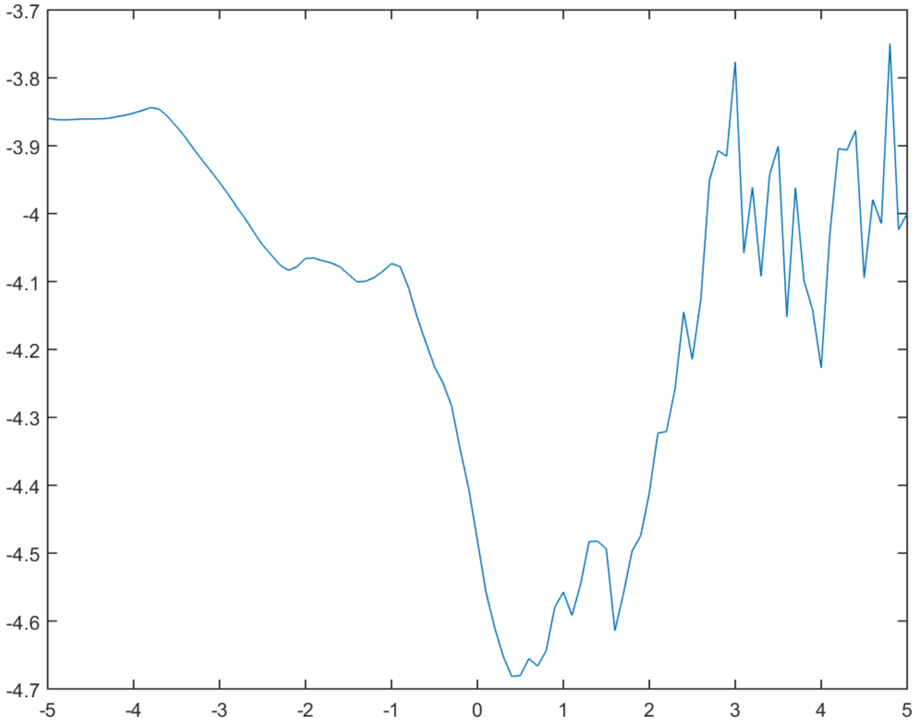
Convolutions: Big Picture

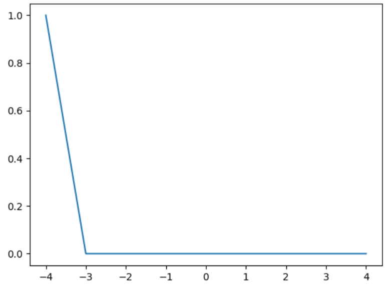
- What can features coming from convolutions represent?
  - Some filters give you an average value of the neighbourhood.
  - Some filters approximate the “first derivative” in the neighbourhood.
    - “Is there a change from low to high (or dark to bright)?”
  - Some filters approximate the “second derivative” in the neighbourhood.
    - “Is there a spike or is the signal speeding up?”
- Hope: we can characterize “what happens in a neighbourhood”, with just a few numbers.

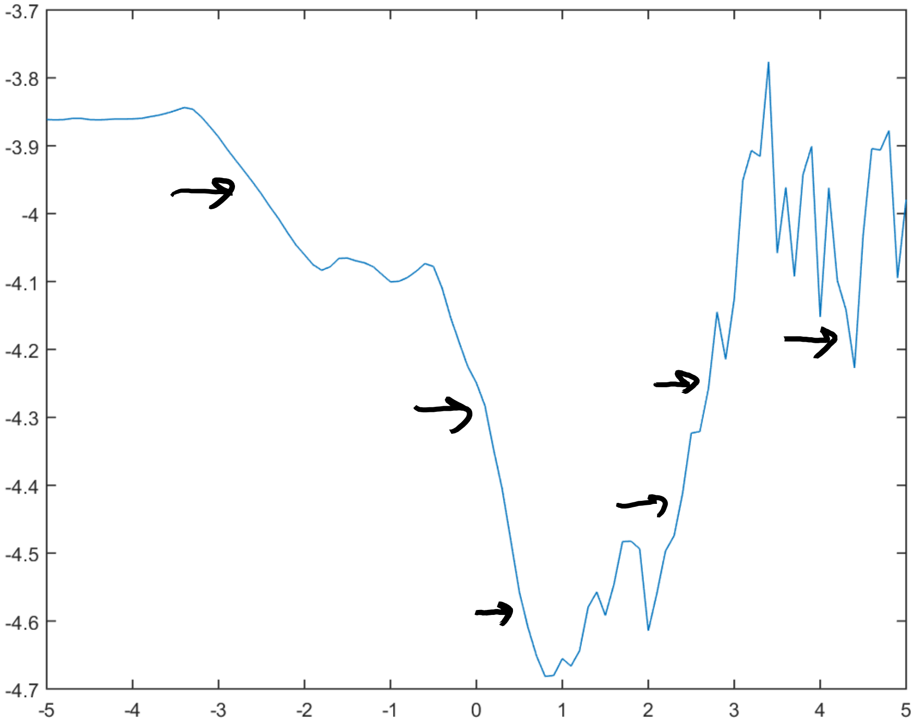
1D Convolution Examples

- Translation convolution shift signal:
  - "What is my neighbour's value?"

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



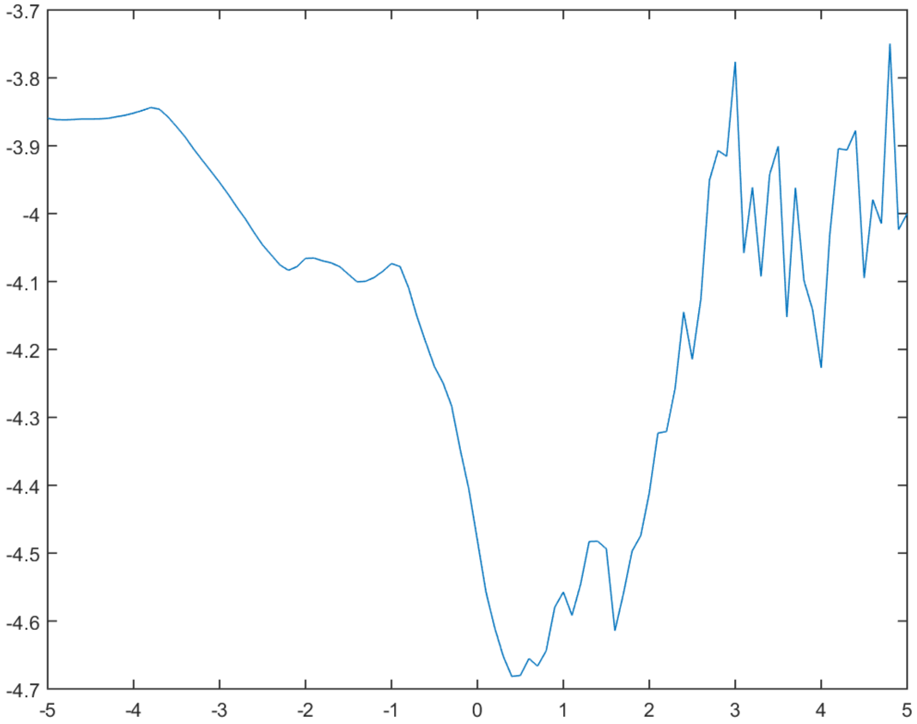


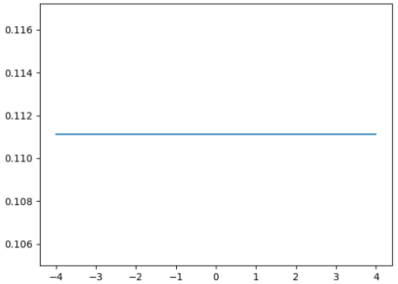


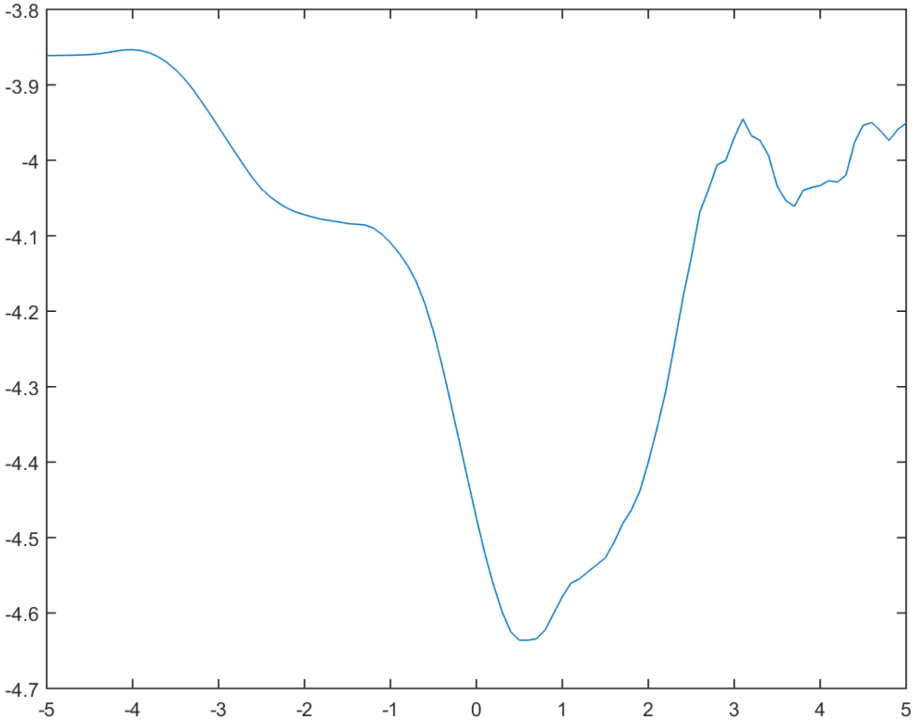
1D Convolution Examples

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

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





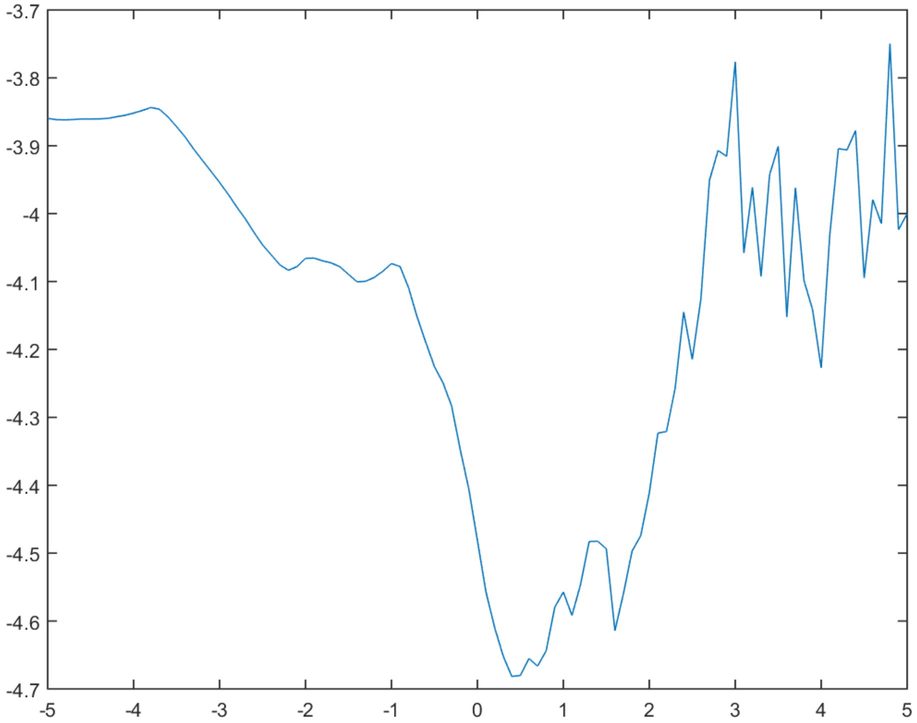


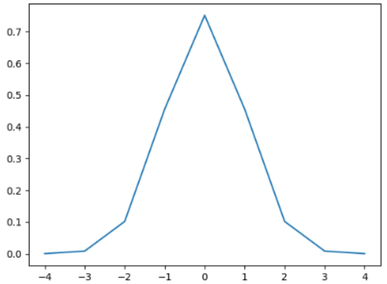
1D Convolution Examples

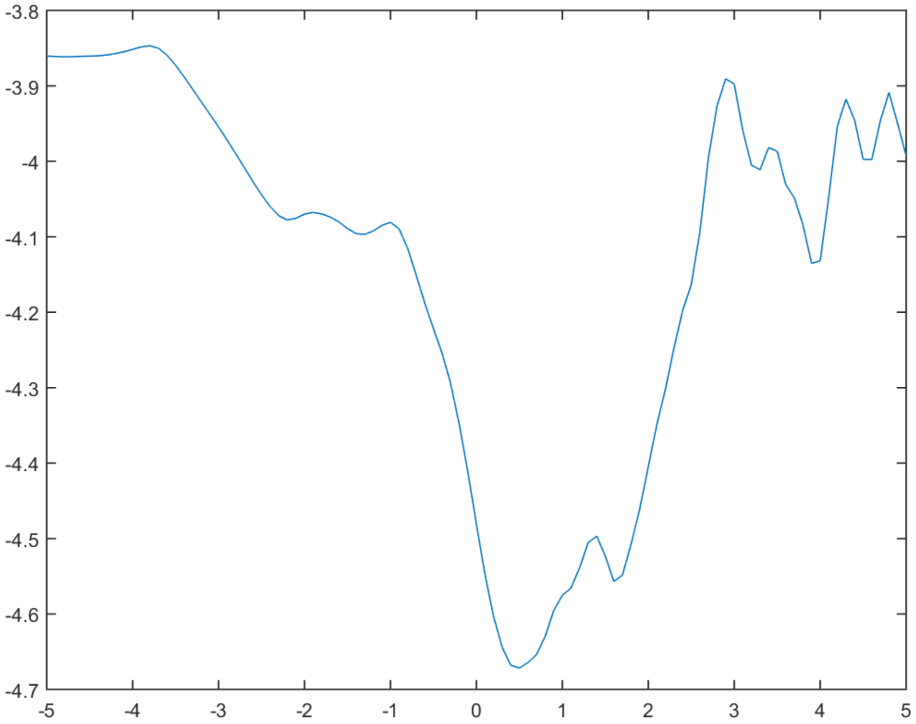
- Gaussian convolution "blurs" signal: $w_i \propto \exp\left(-\frac{i^2}{2\sigma^2}\right)$
  - Compared to averaging it's more smooth and maintains peaks better.

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

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



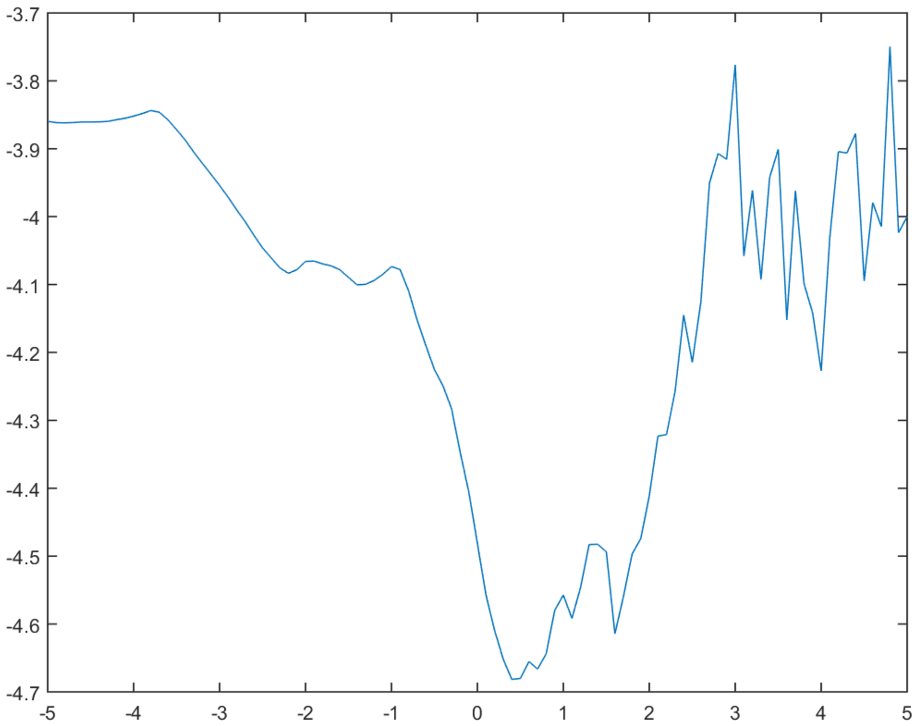


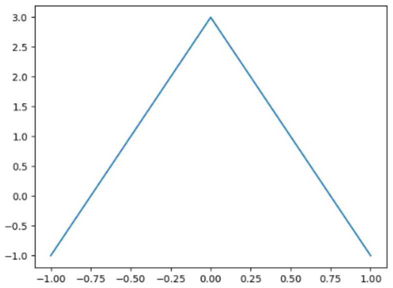


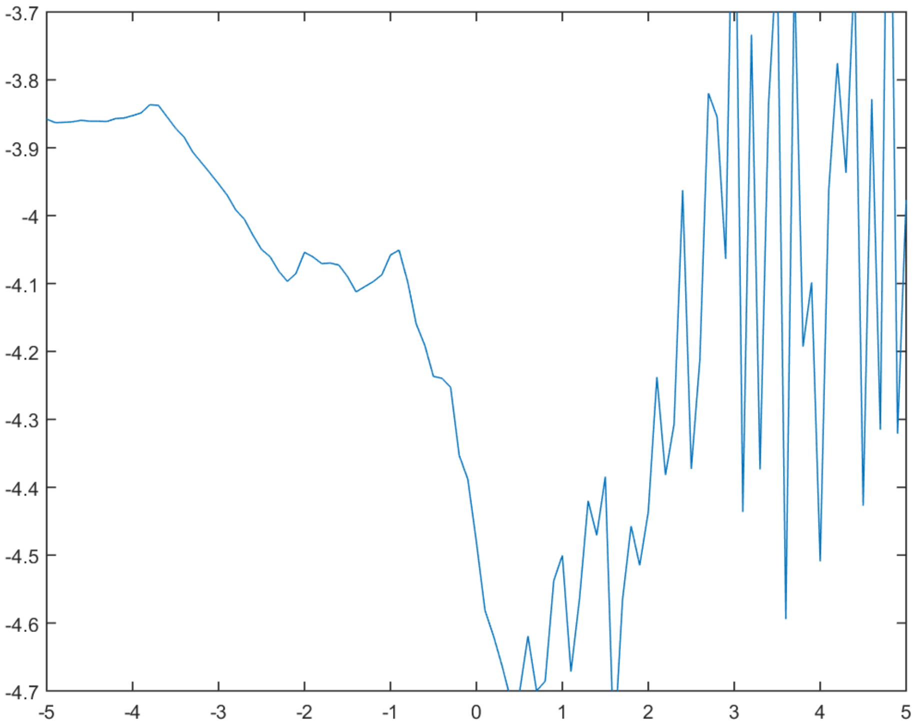
1D Convolution Examples

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

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



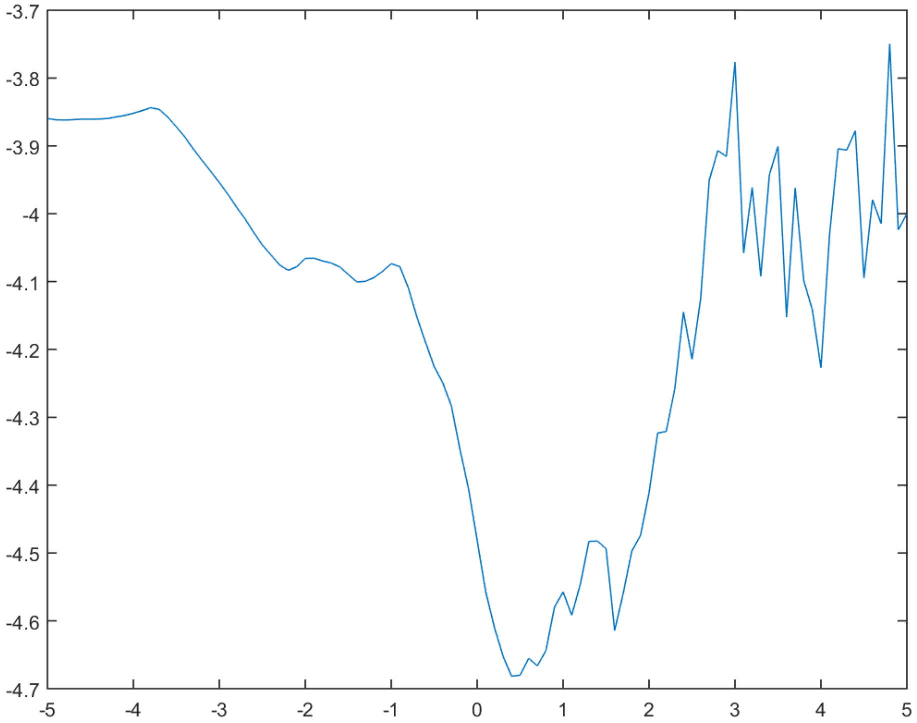


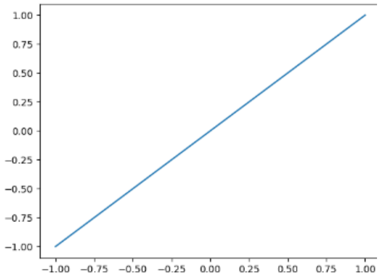


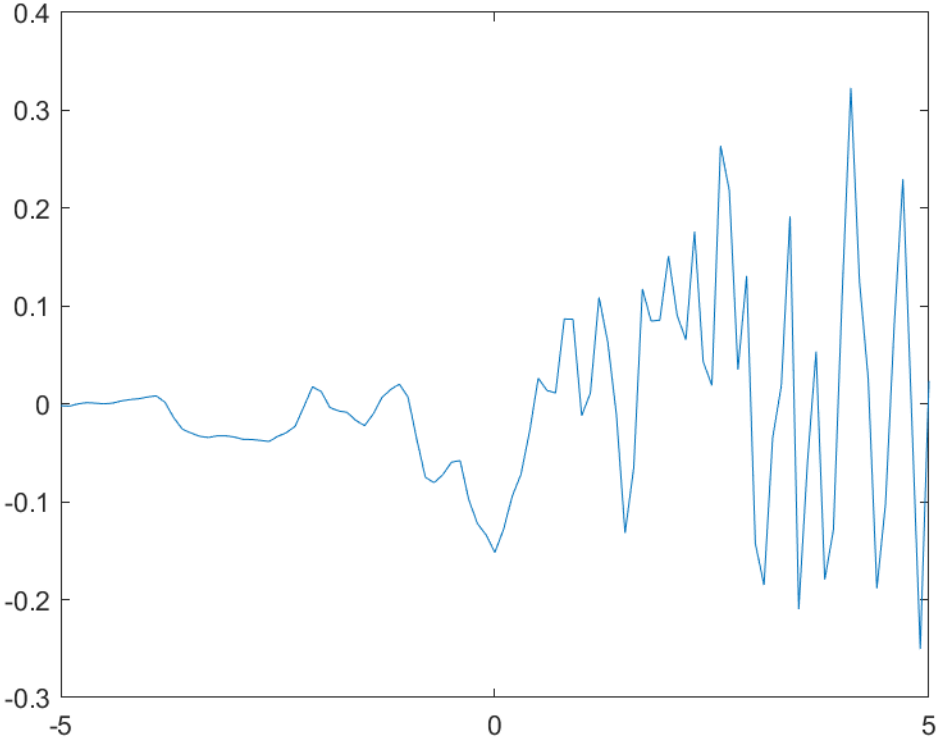
1D Convolution Examples

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

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

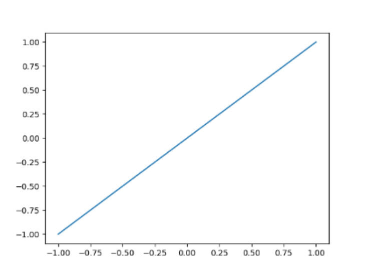




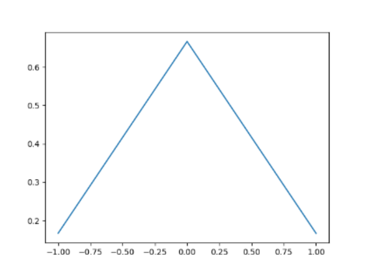


Digression: Derivatives and Integrals

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



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

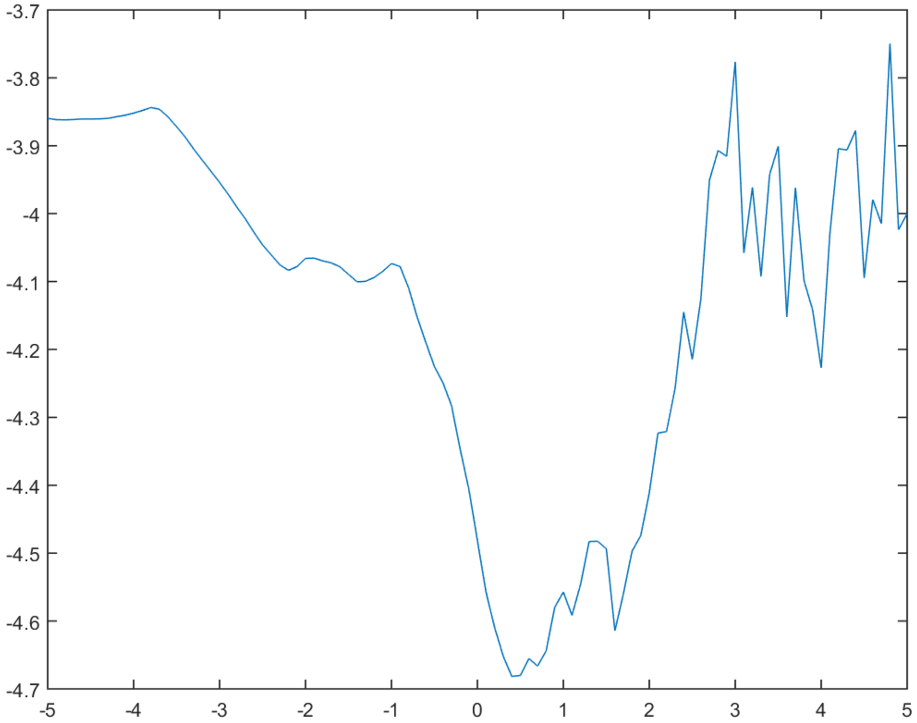


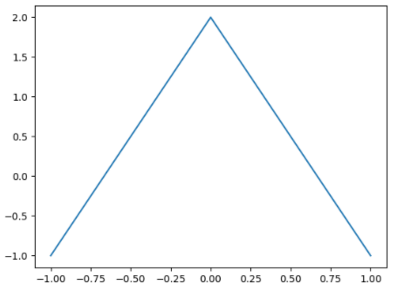
- Derivative filters add to 0, integration filters add to 1,
  - For constant function, derivative should be 0 and average = constant.

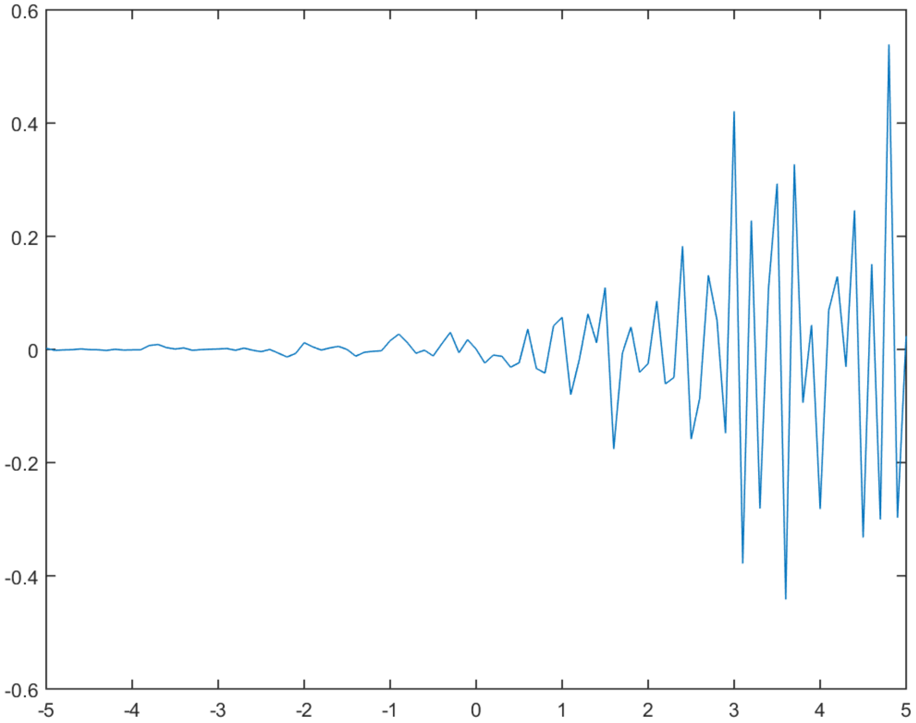
1D Convolution Examples

- Laplacian 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

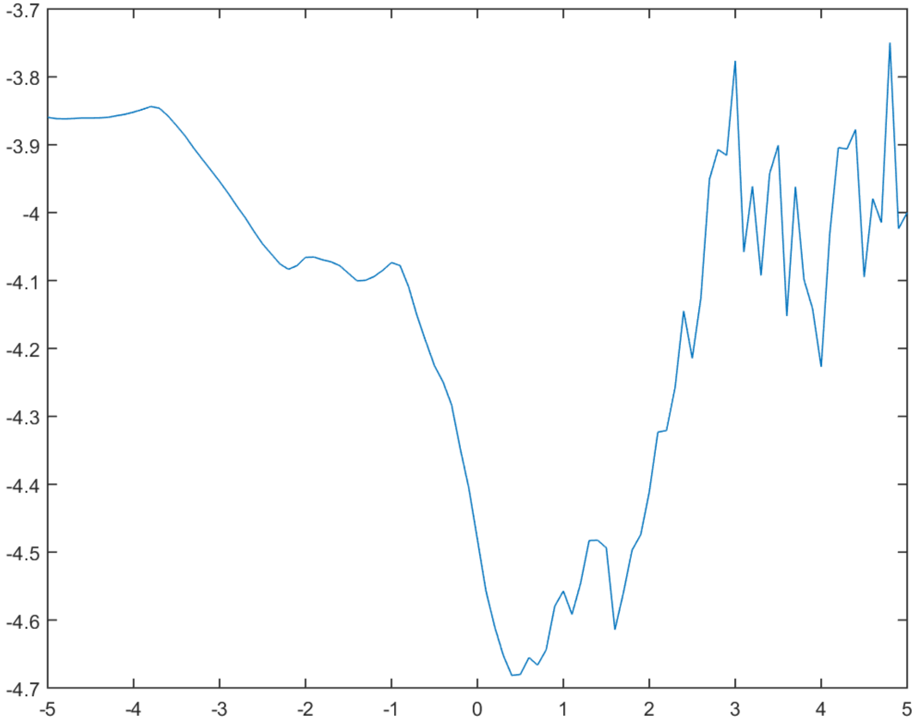
- Laplacian of Gaussian is a smoothed 2nd-derivative approximation:

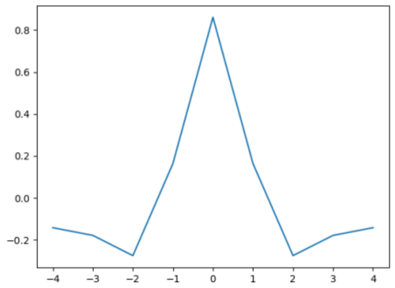
$$w_i = \left(1 - \frac{i^2}{2\sigma^2}\right)\exp\left(-\frac{i^2}{2\sigma^2}\right)$$

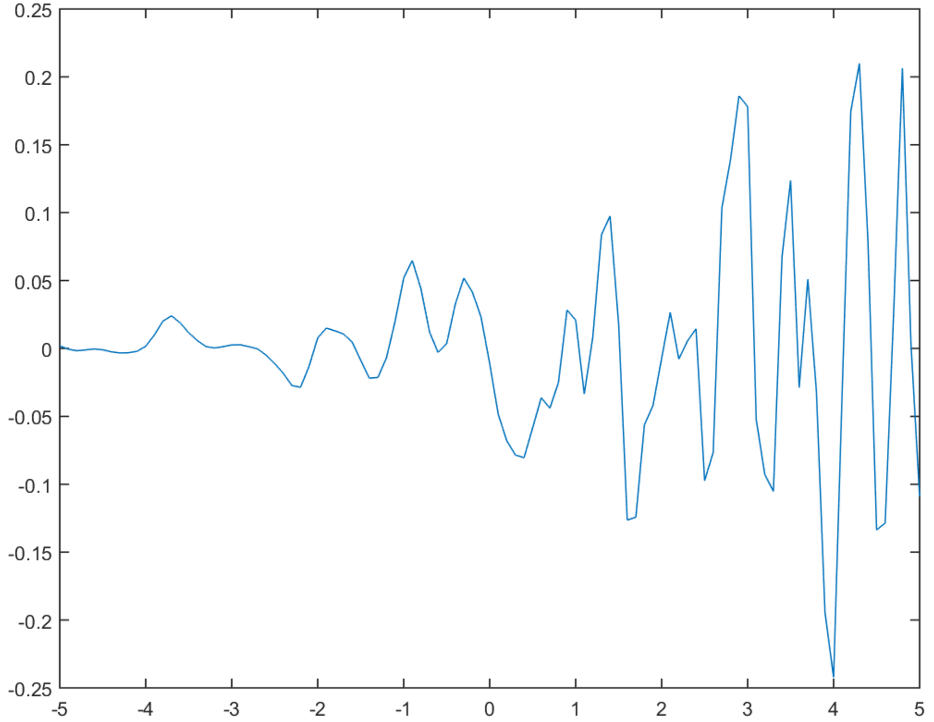
(then subtract mean)

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

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

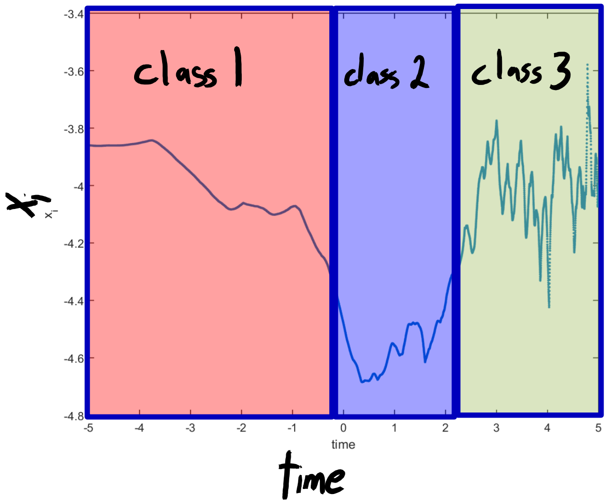


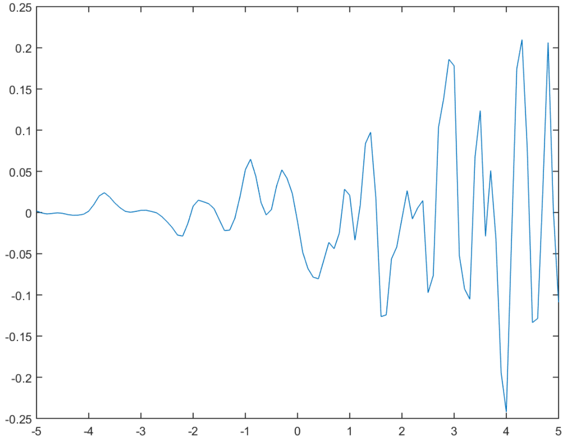




Taking Maximums of Convolutions

- Remember our motivation example:

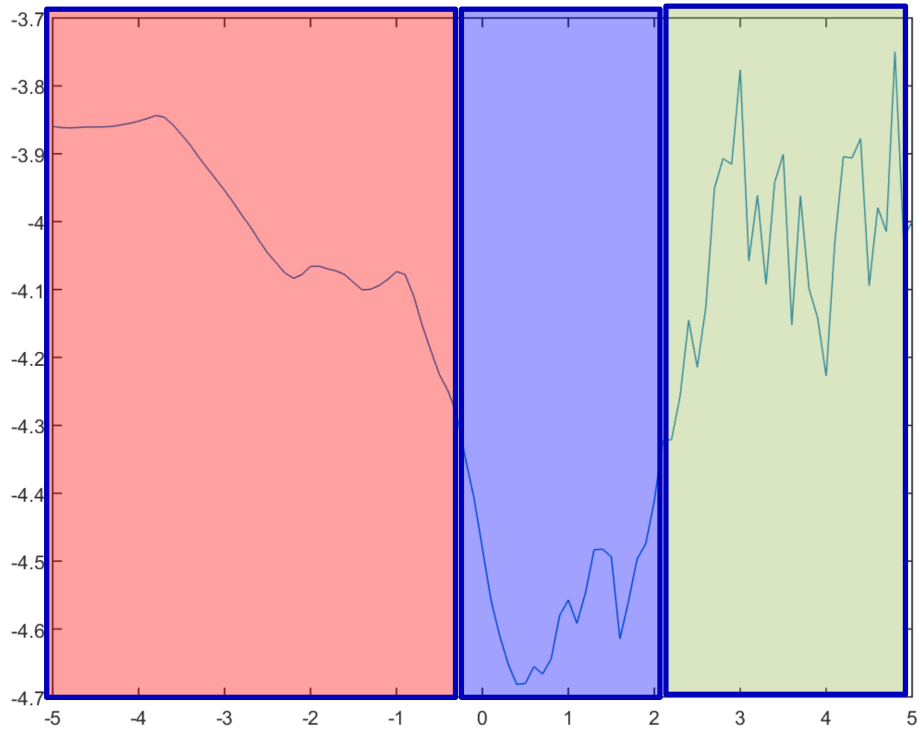


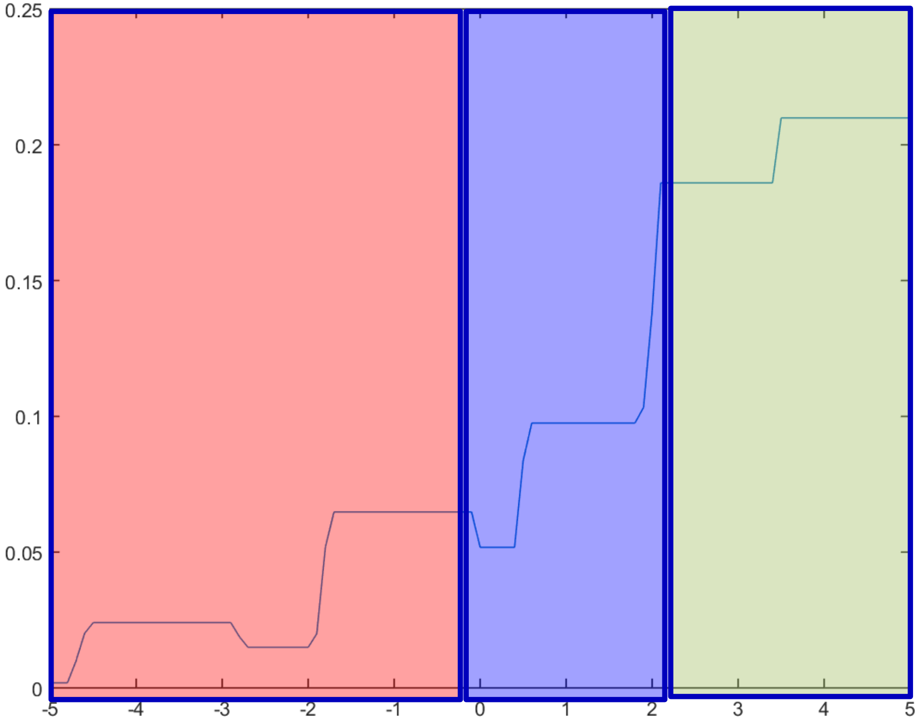


- Laplacian of Gaussian filters looks useful:
  - Class 1 and 3 usually often have different values.
    - Close to zero for class 1, often far from zero for class 3.
    - But class 3 values are still sometimes close to 0.
- What we take maximum absolute value over 16 adjacent times?

Taking Maximums of Convolutions

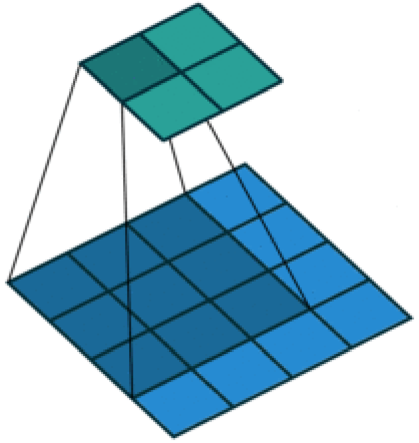
- We often use maximum over several convolutions as features:
  - On right is the maximum(abs(Laplacian of Gaussian)) at ‘i’ and its 16 KNNs.
  - We can solve the problem with just the 2 features below at each location.





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://github.com/vdumoulin/conv_arithmetic

Convolutions as Features

- Classic vision methods uses 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).

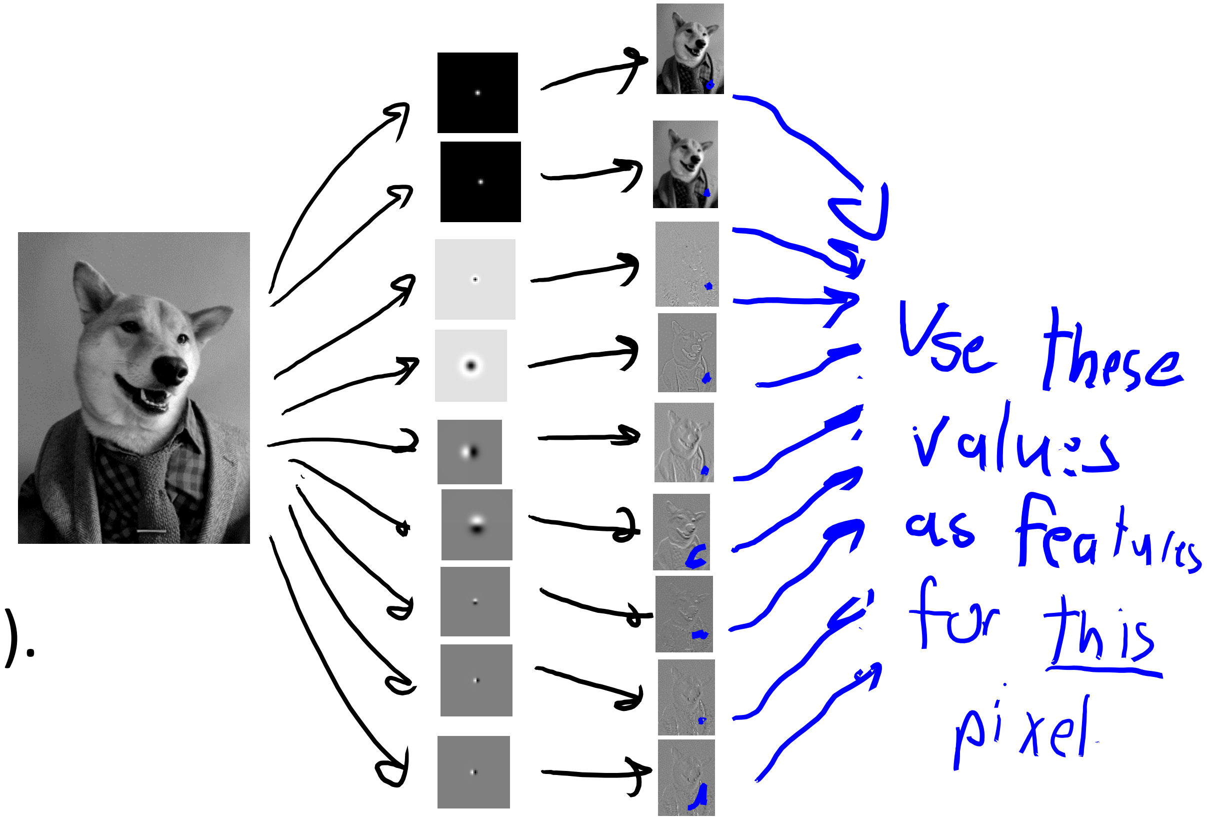


Image Convolution Examples

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

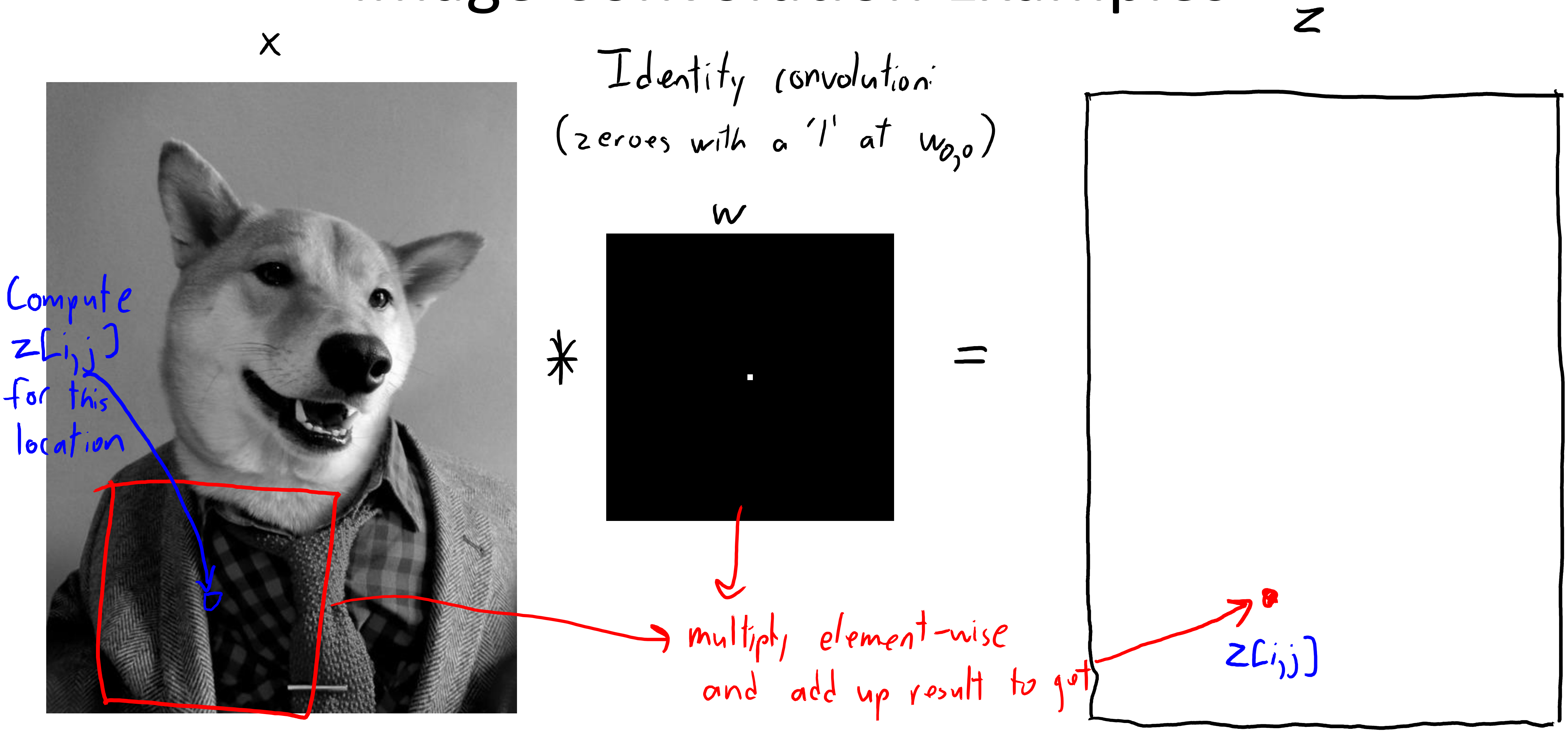
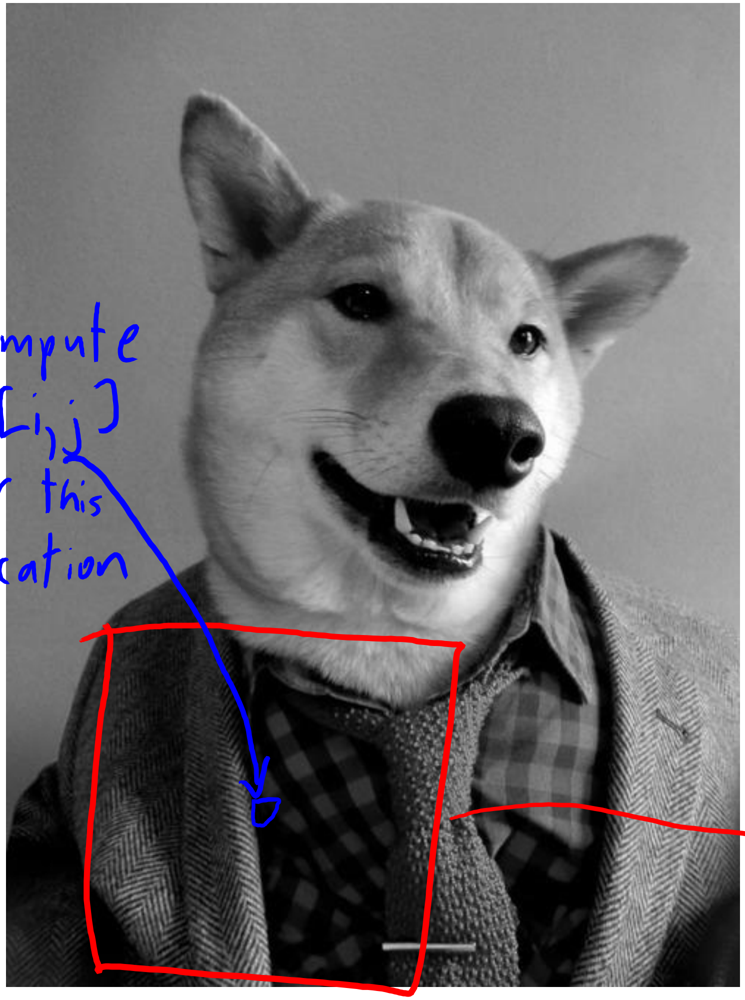

x
w
*
=
z
Compute z[i,j] for this location
multiply element-wise and add up result to get
z[i,j]

Image Convolution Examples

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

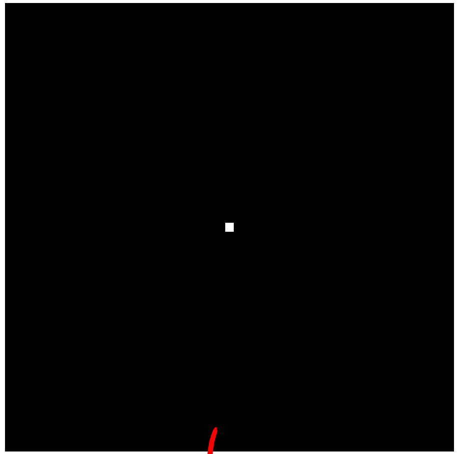
x



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

$*$

w



$=$

z

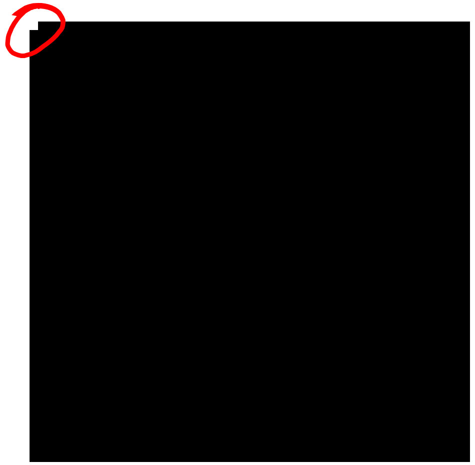
$z[i,j]$

multiply element-wise and add up result to get

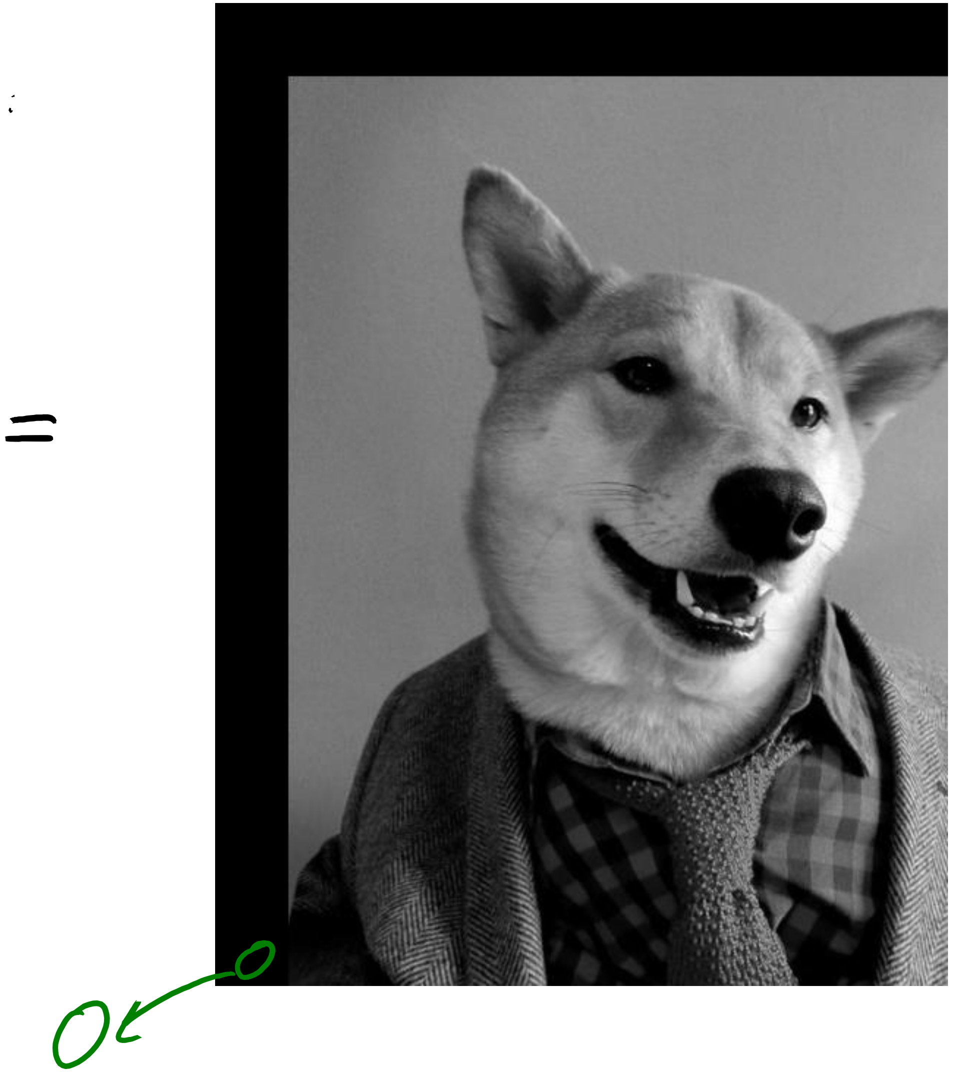
Image Convolution Examples

Translation Convolution:

*



=

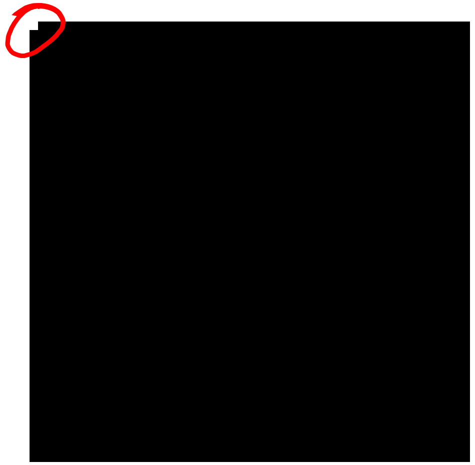


Boundary: "zero"

Image Convolution Examples

Translation Convolution:

$*$



$=$

Boundary: "replicate"

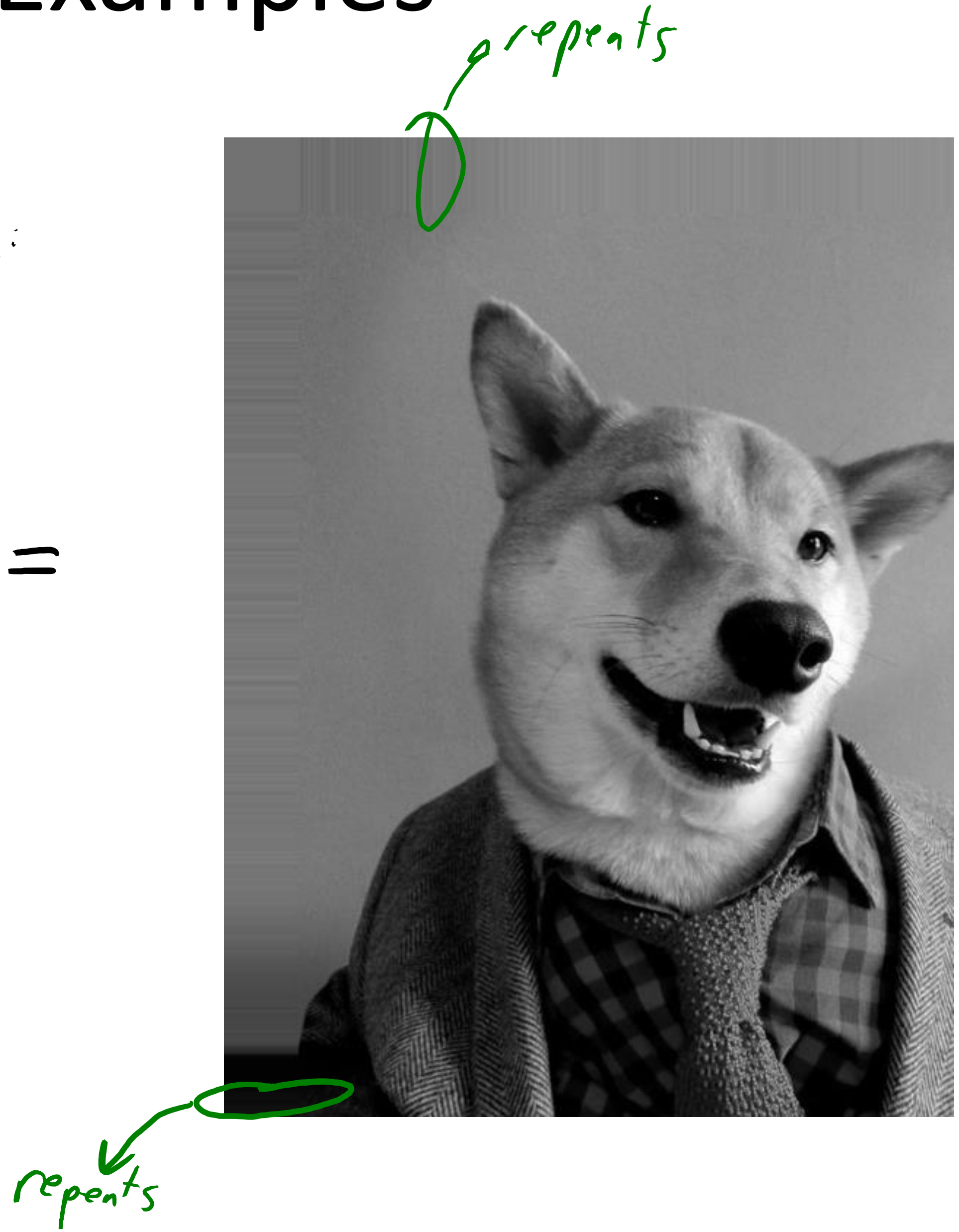
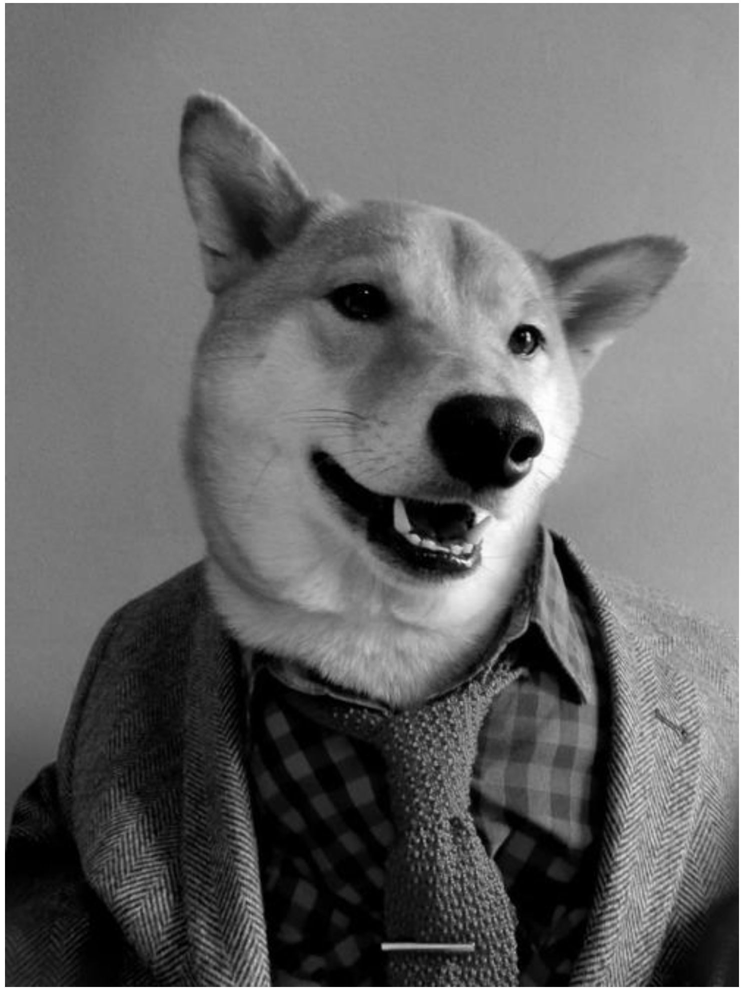
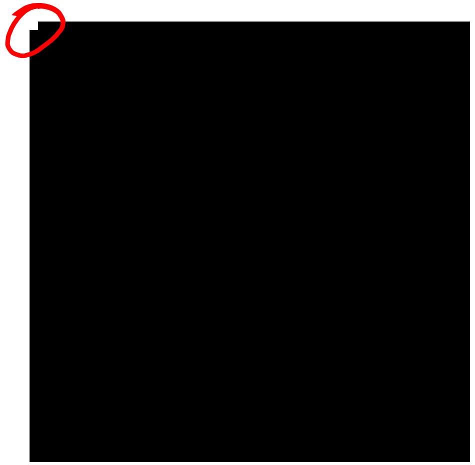


Image Convolution Examples



Translation Convolution:

$*$



$=$

Boundary: "mirror"

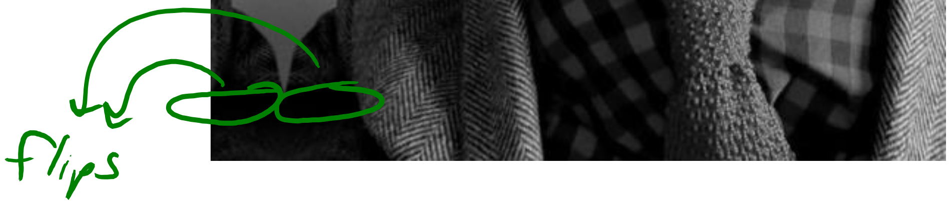
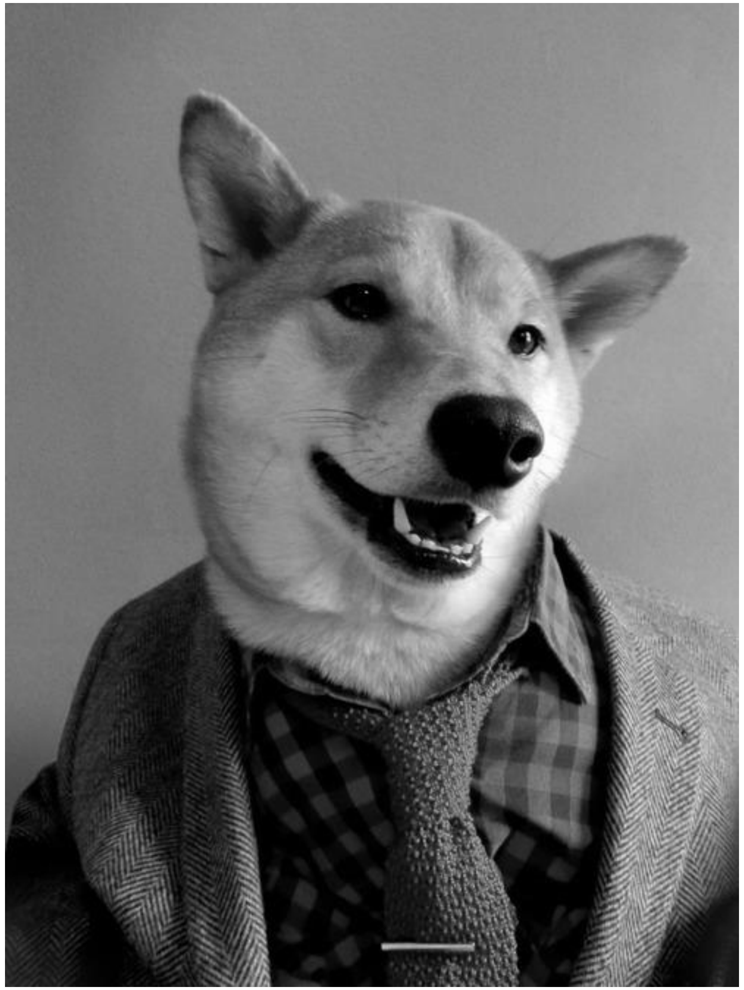
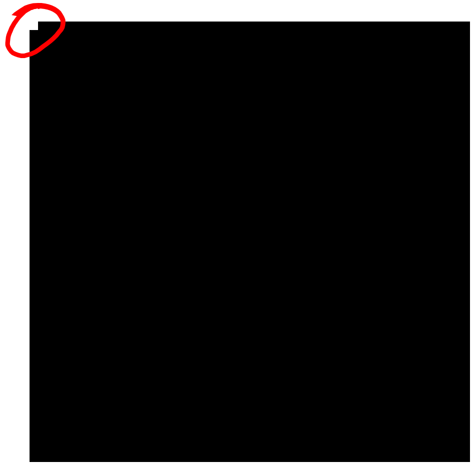


Image Convolution Examples



Translation Convolution:

$*$



$=$

Boundary: "ignore"

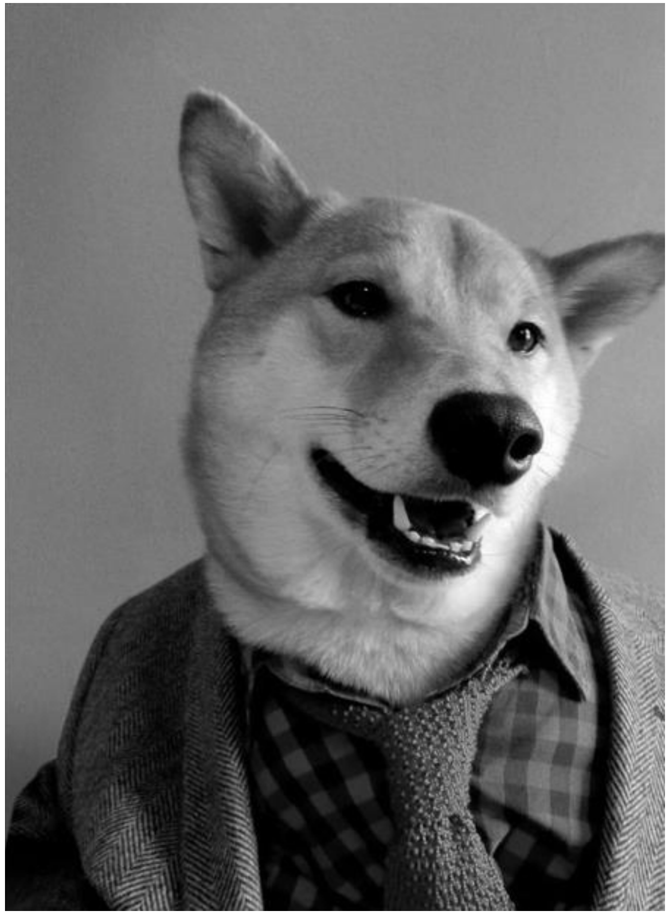
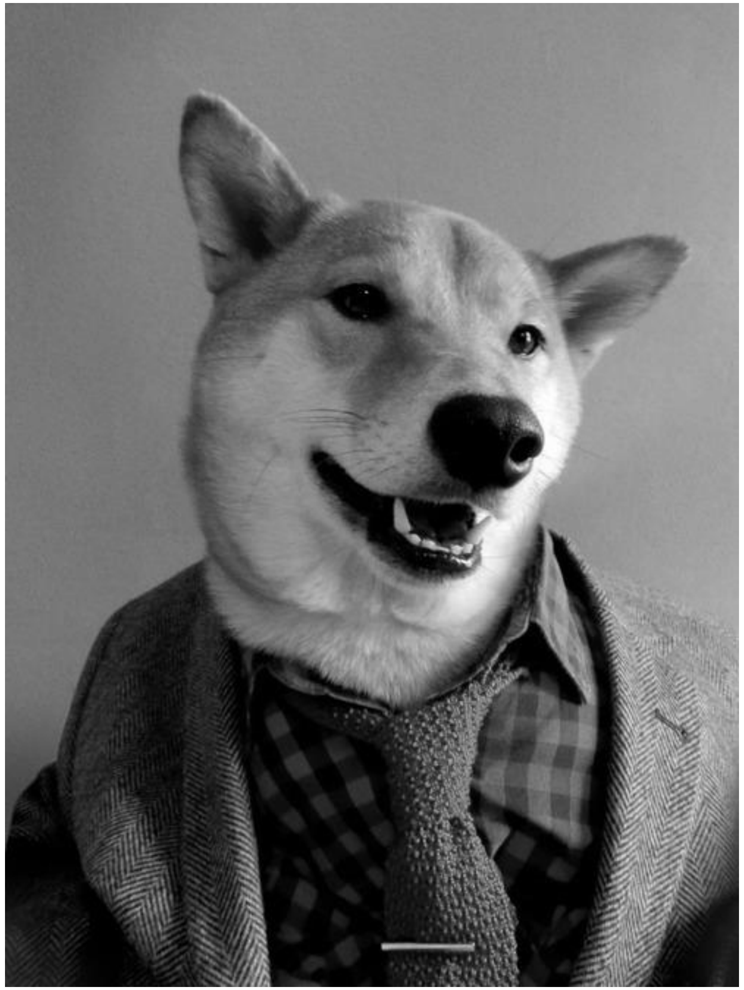


Image Convolution Examples



Average convolution:

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

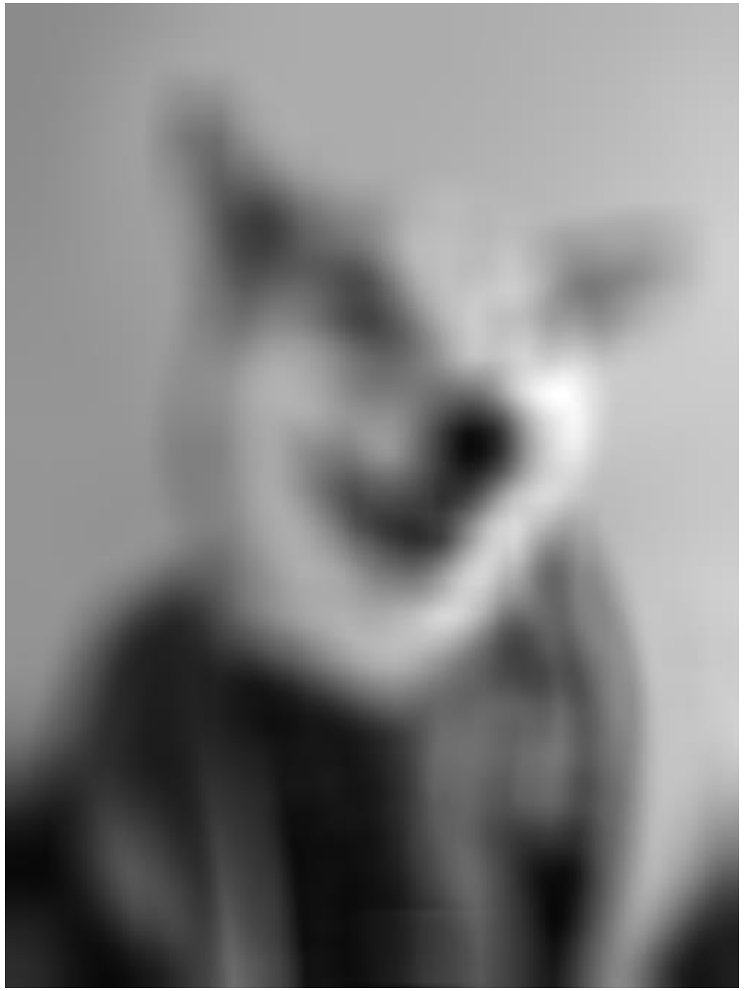
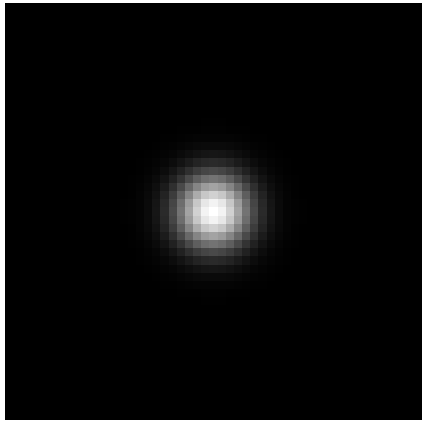


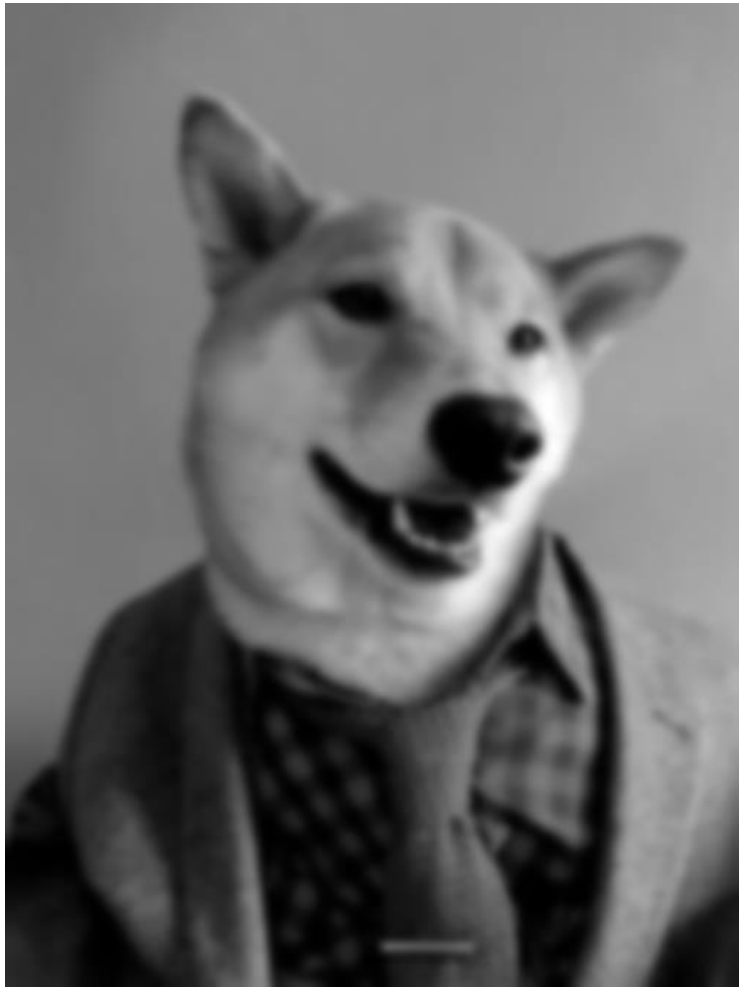
Image Convolution Examples

Gaussian Convolution:

$*$

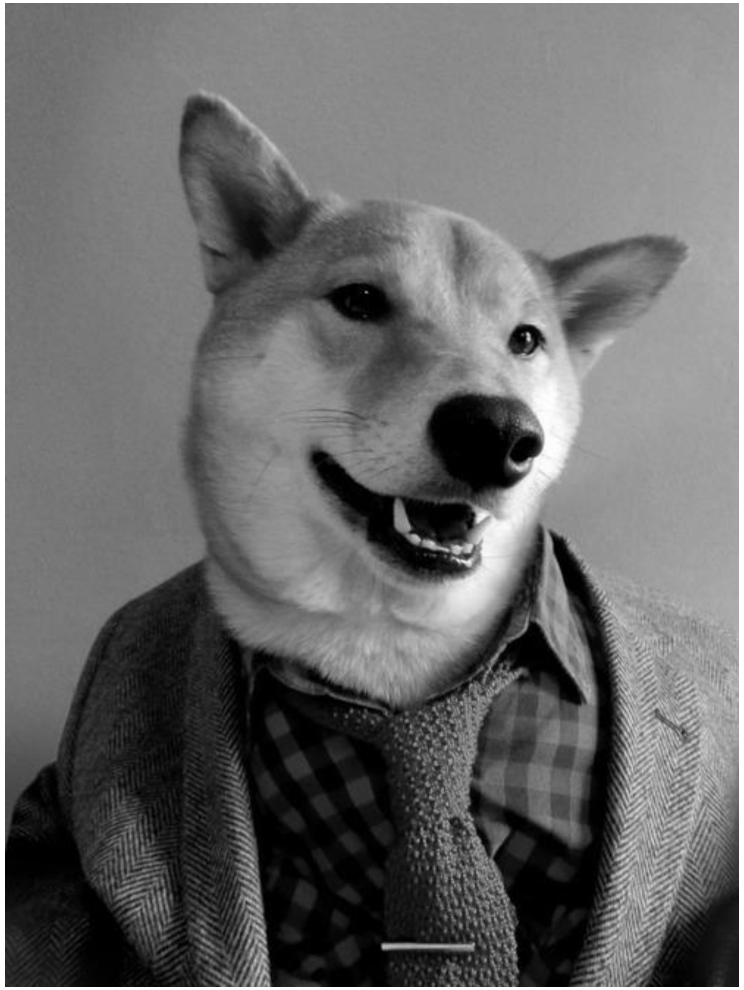


$=$



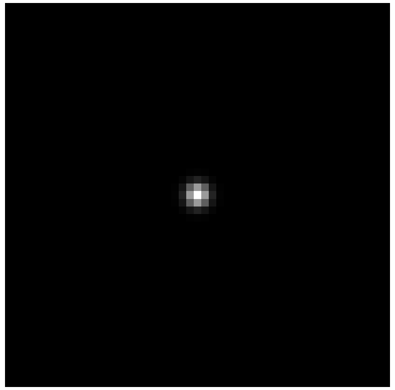
<u>blurs</u> image to represent average (smoothing)

Image Convolution Examples



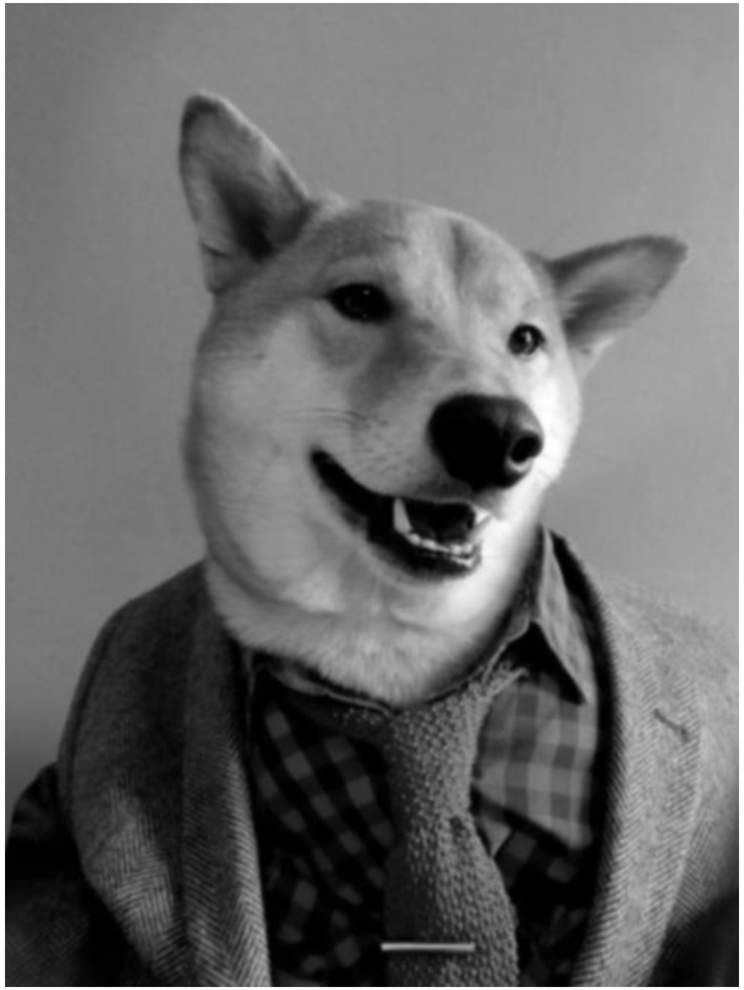
Gaussian Convolution:

*



(smaller variance)

=

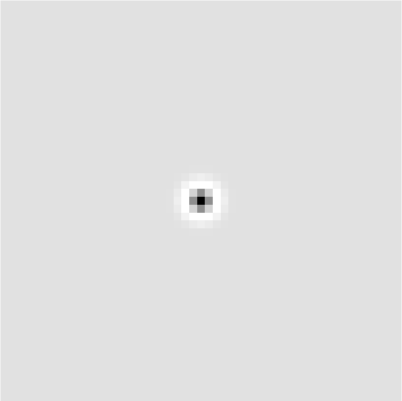


blurs image to represent average (smoothing)

Image Convolution Examples

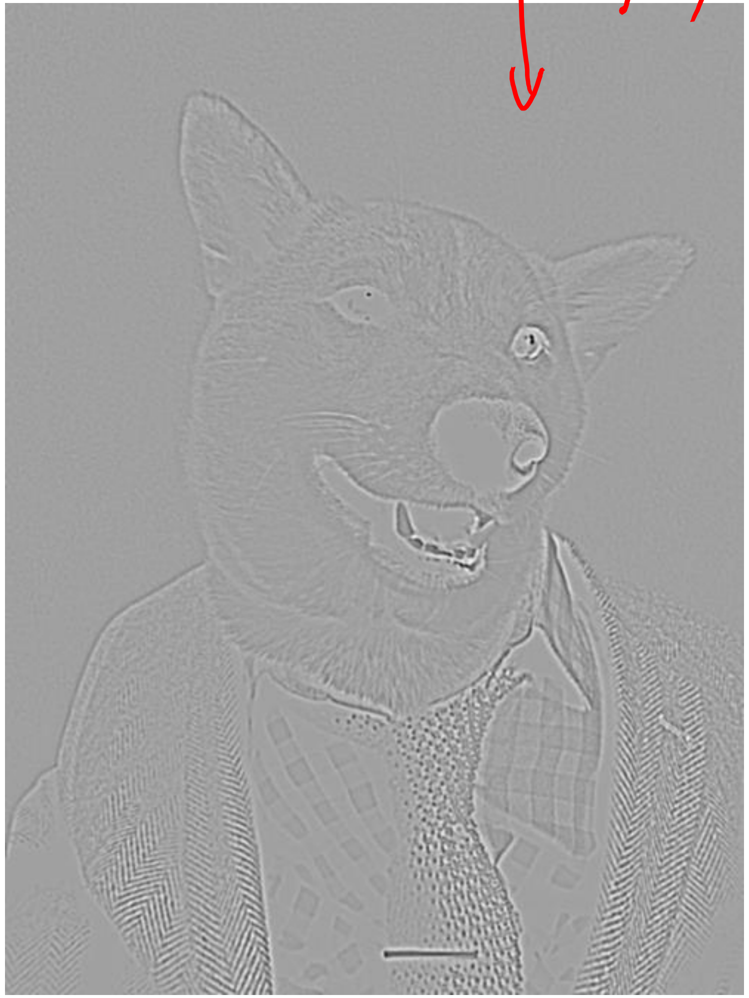
Laplacian of Gaussian

$*$



$=$

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

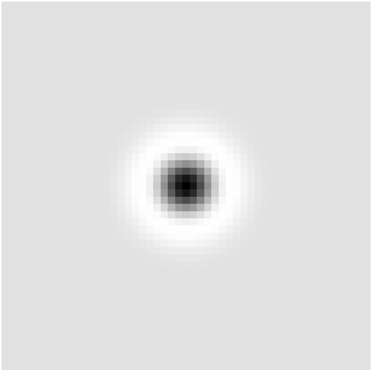


"signed" image (gray is 0)

Image Convolution Examples

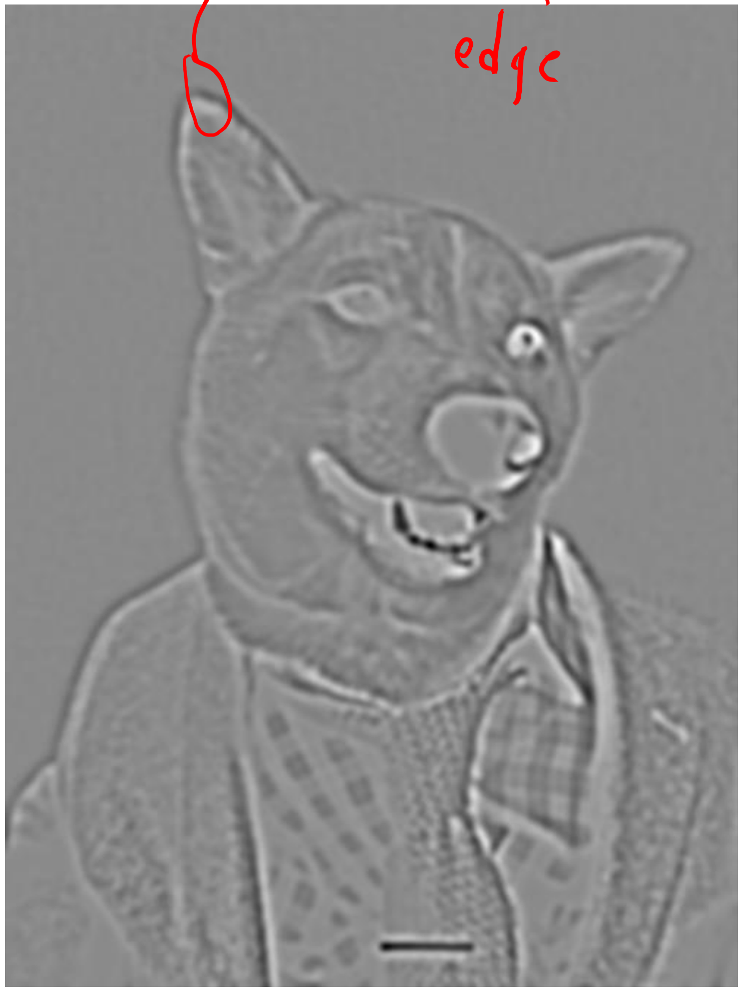
Laplacian of Gaussian

$*$



$=$

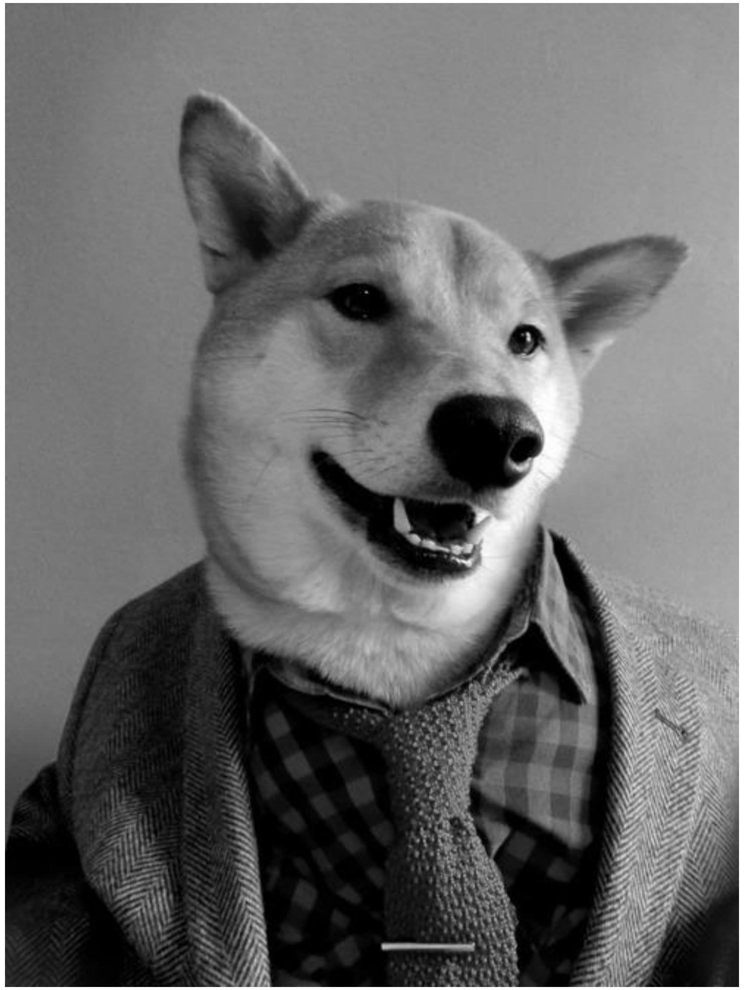
(larger variance)



Black/white as sides of edge

Similar preprocessing may be done in basal ganglia and LGN.

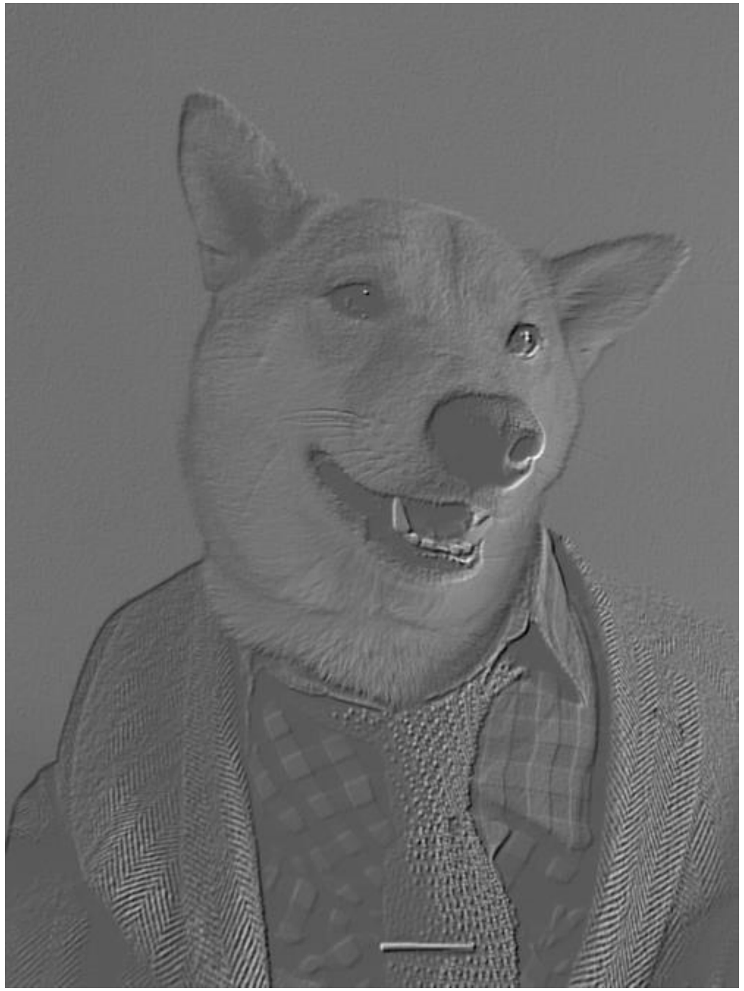
Image Convolution Examples



"Emboss" filter:

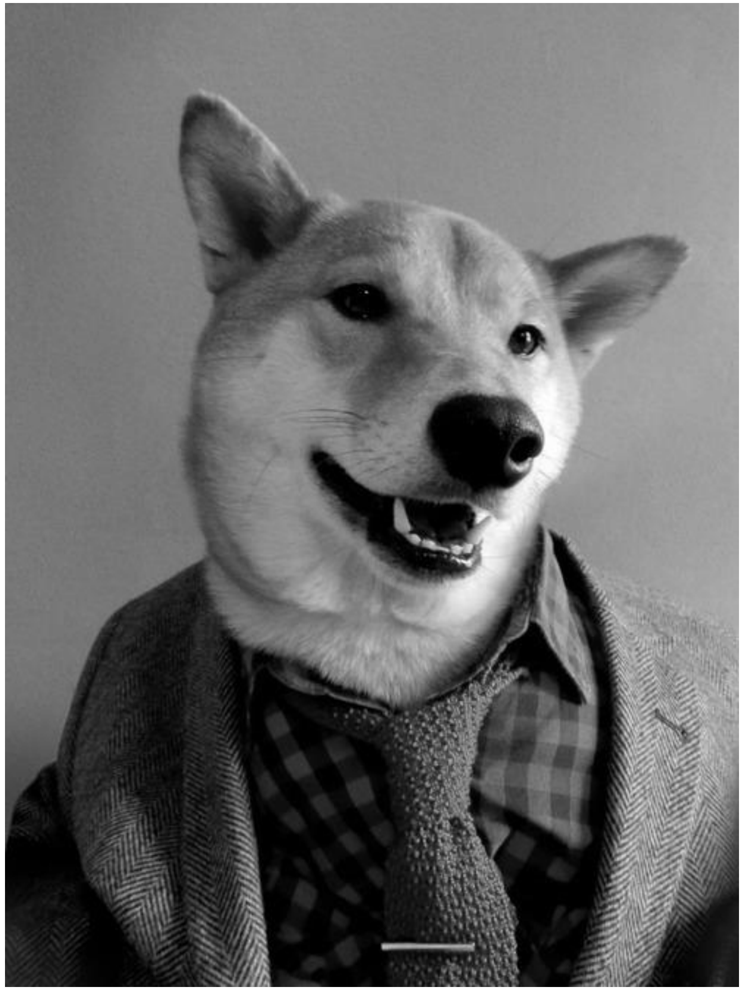
$$* \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} =$$

Many Photoshop effects are just convolutions.



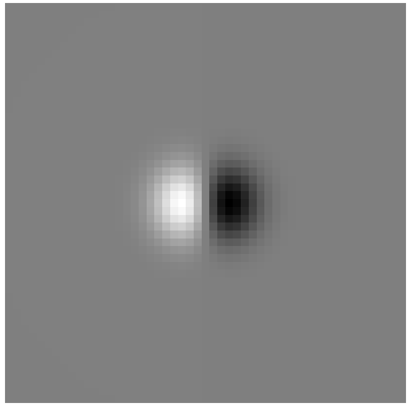
http://setosa.io/ev/image-kernels

Image Convolution Examples

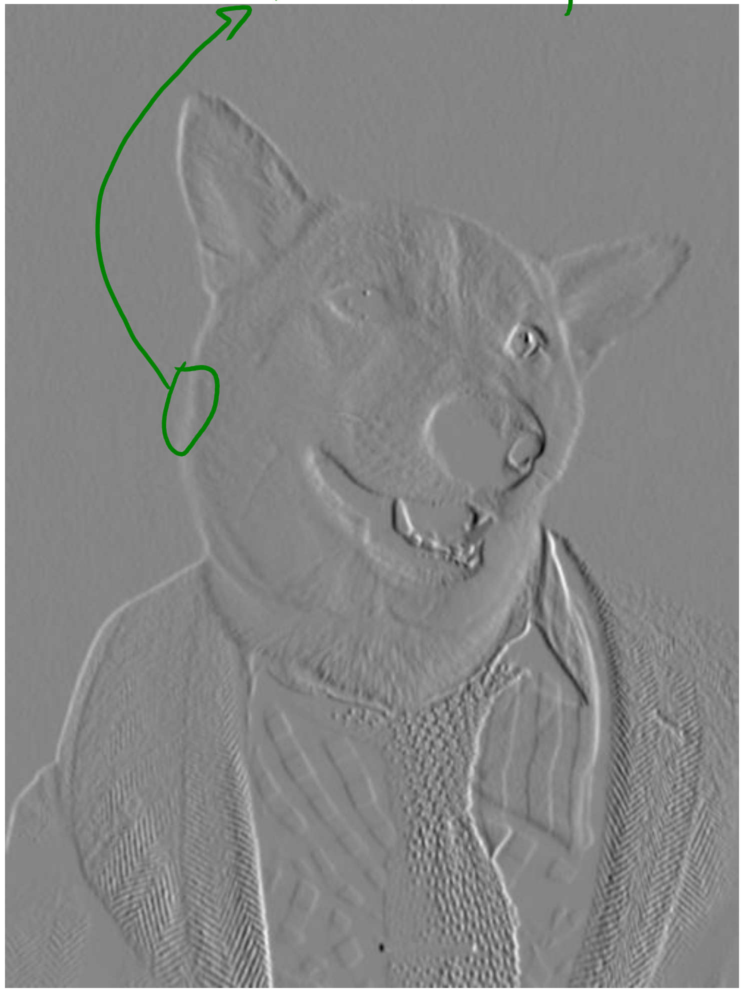


*

Gabor Filter
(Gaussian multiplied by
sine or cosine)

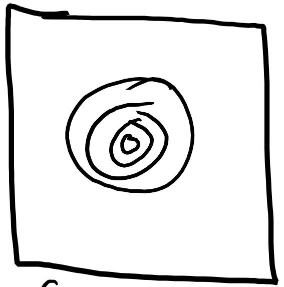


=



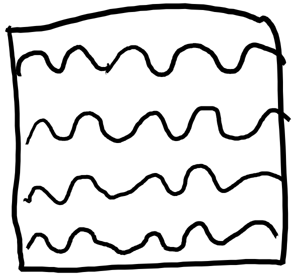
Horizontal "bright to dark"

||



Gaussian

.*

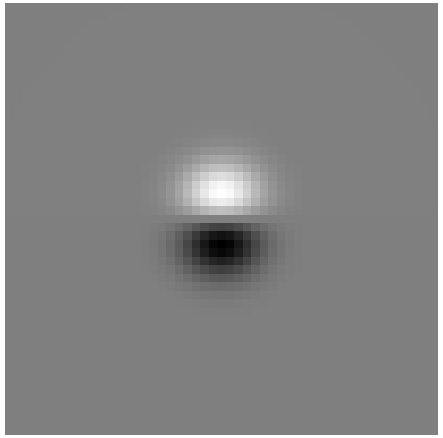


Parallel Sine functions

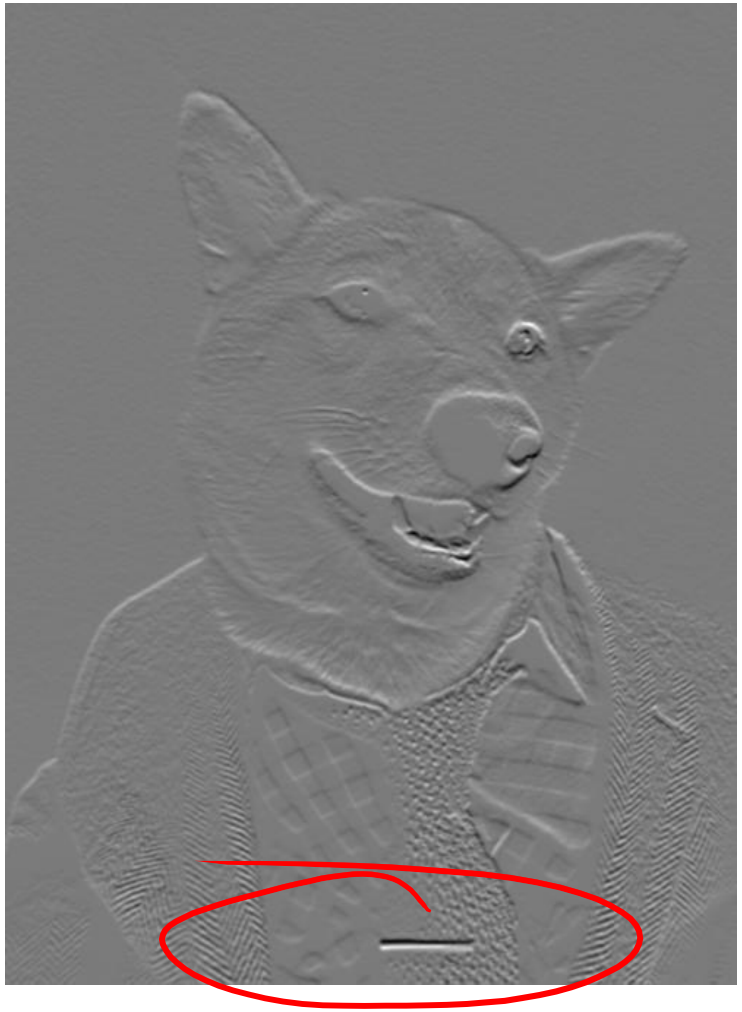
Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

*



=



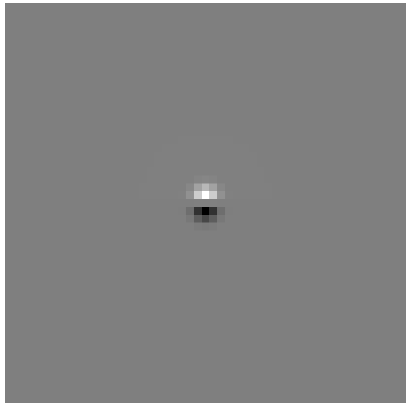
Different orientations of the sine/cosine let us detect changes with different orientations.

2d derivatives have a direction.

Image Convolution Examples

Gabor Filter
(Gaussian multiplied by
sine or cosine)

*



(smaller variance)

=

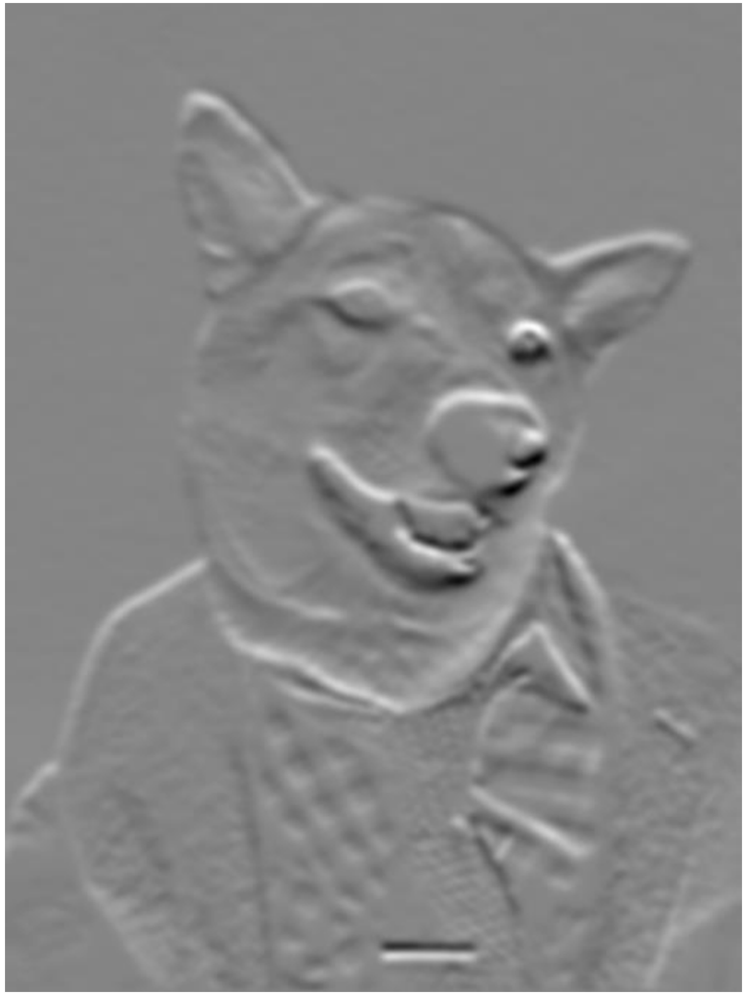
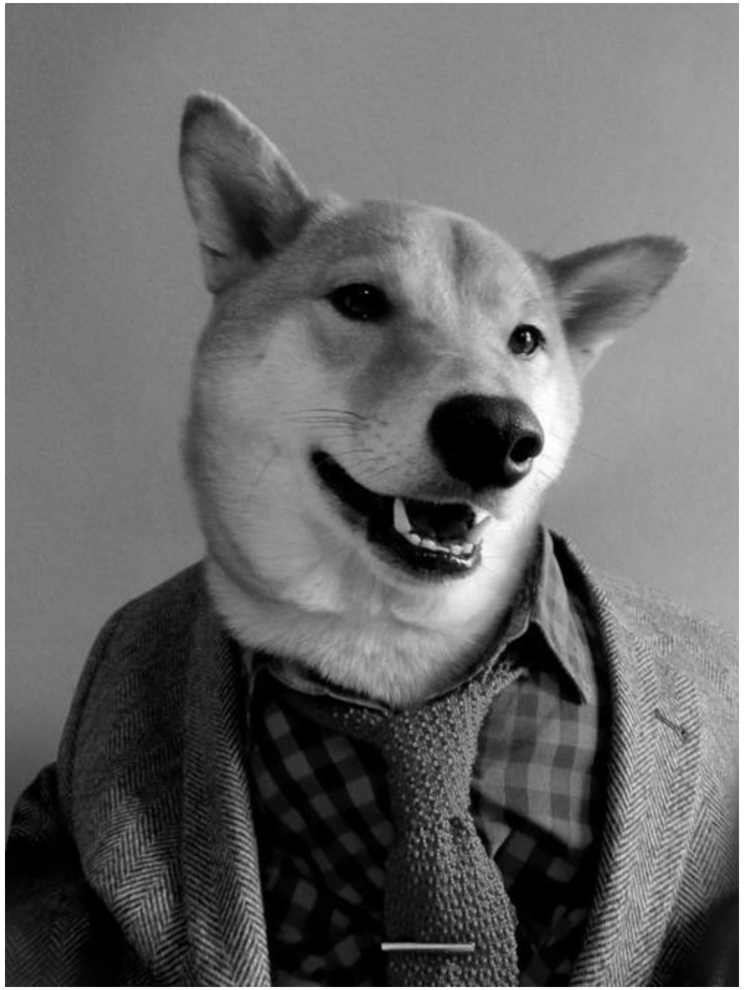
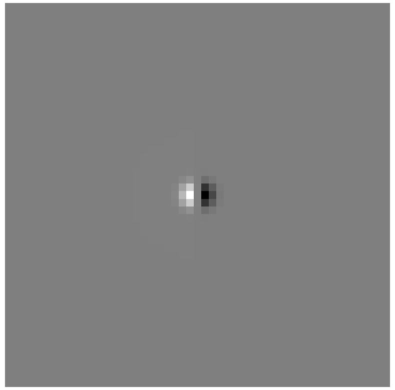


Image Convolution Examples

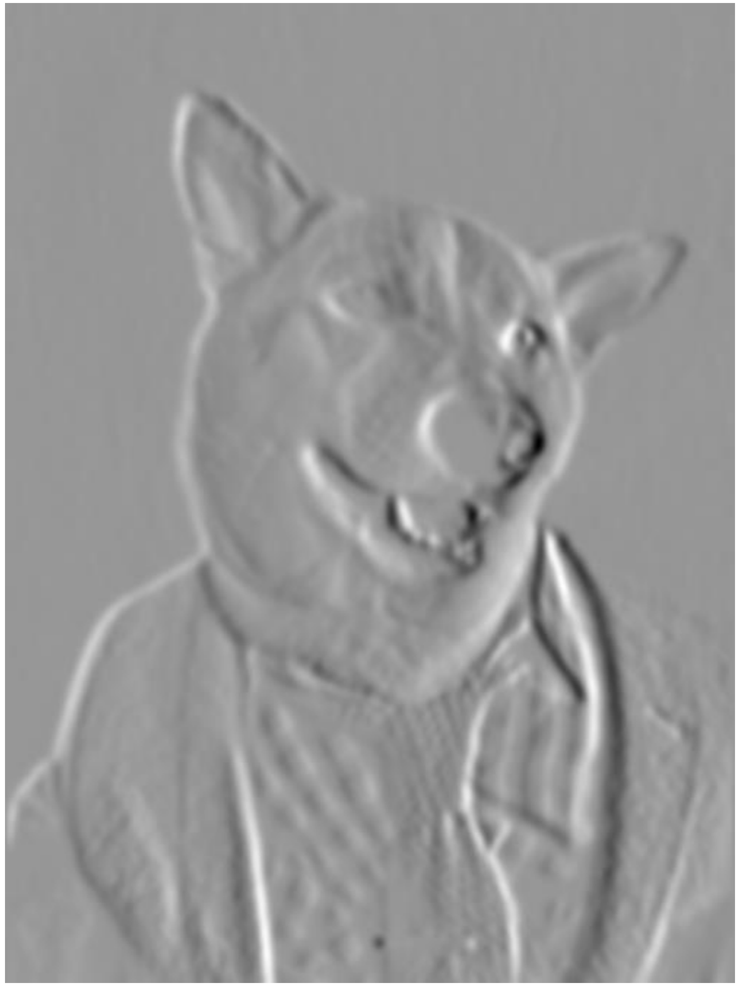
Gabor Filter
(Gaussian multiplied by sine or cosine)



*



=



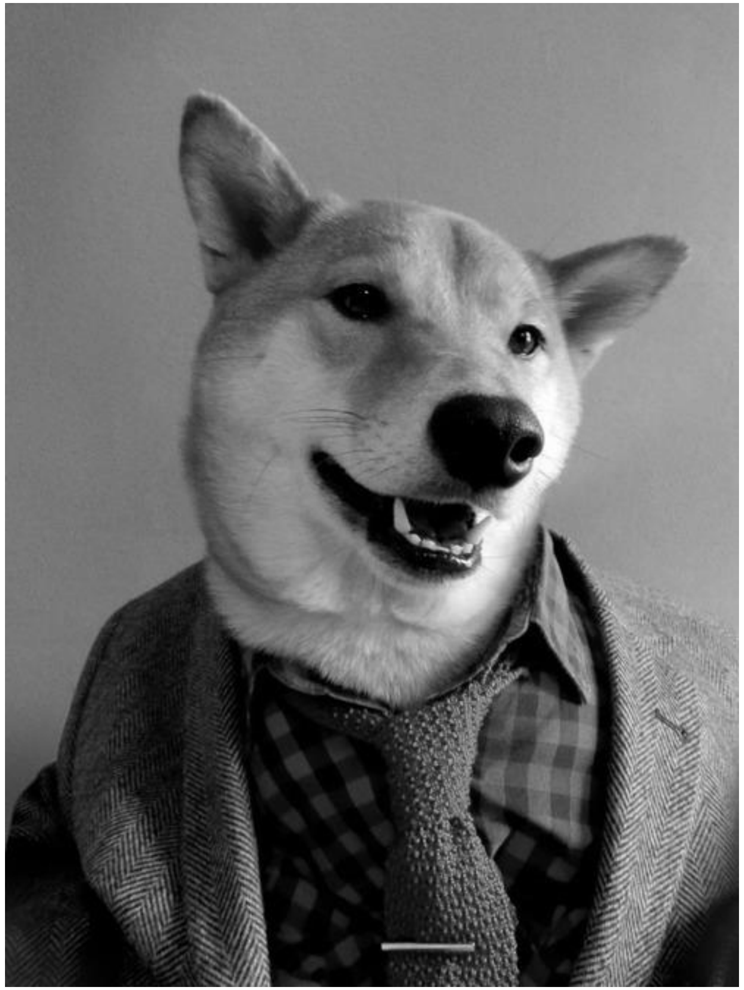
(smaller variance)

Vertical orientation

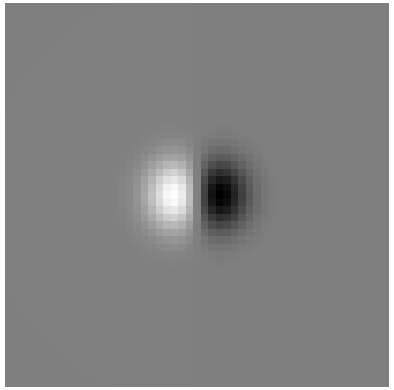
- Can obtain other orientations by rotating.
- May be similar to effect of V1 "simple cells."

Image Convolution Examples

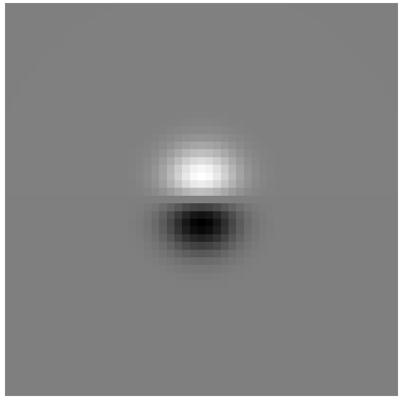
Max absolute value between horizontal and vertical Gabor:



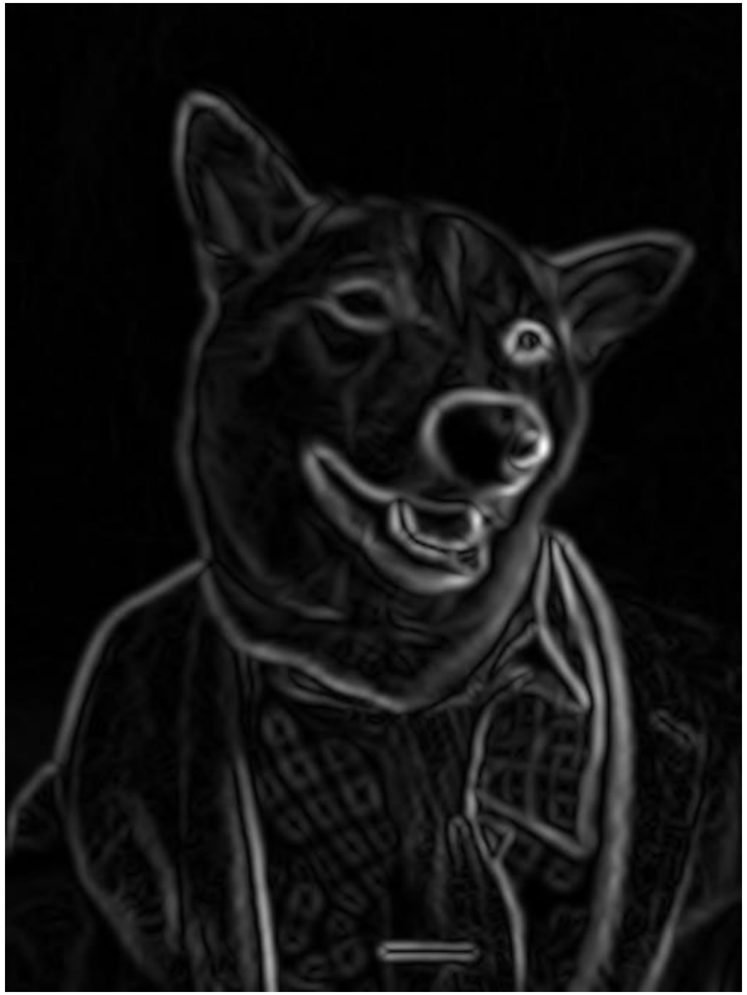
*



*



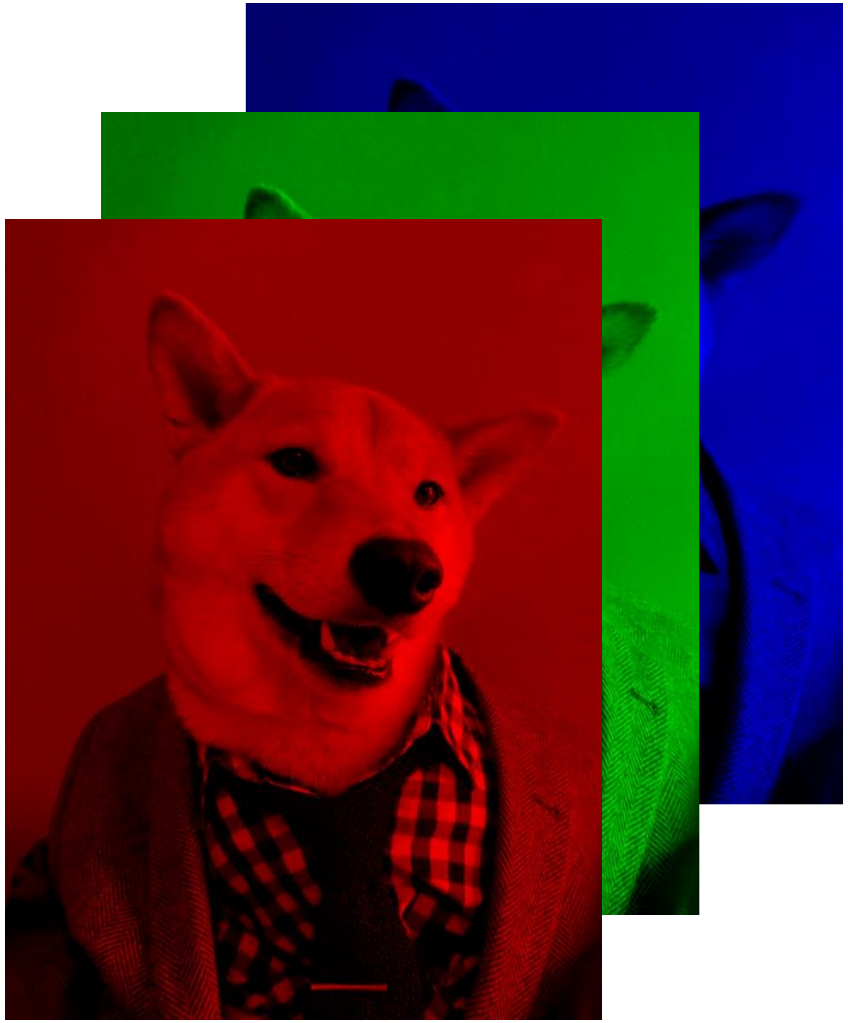
maximum absolute value



"Horizontal/vertical edge detector"

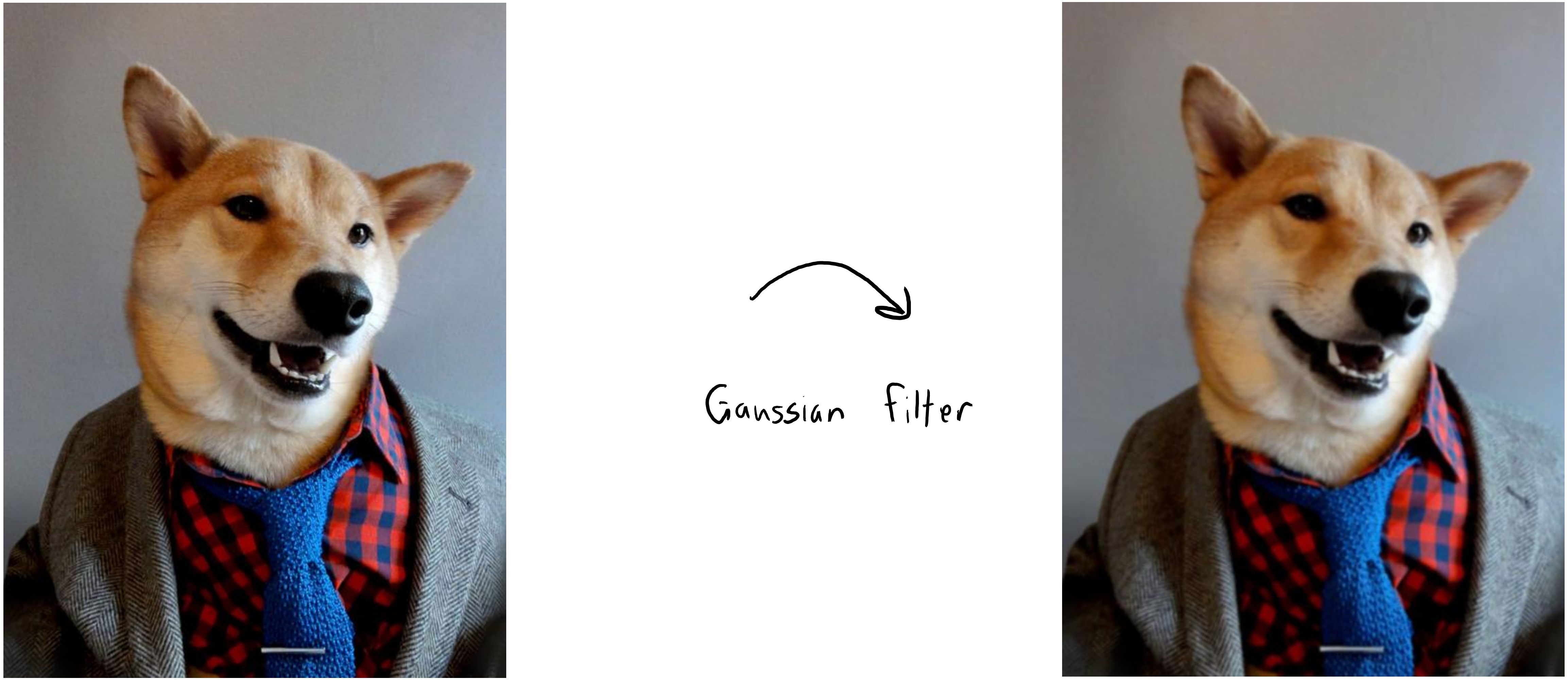
3D Convolution

Represent as RGB



Can apply 3D convolutions

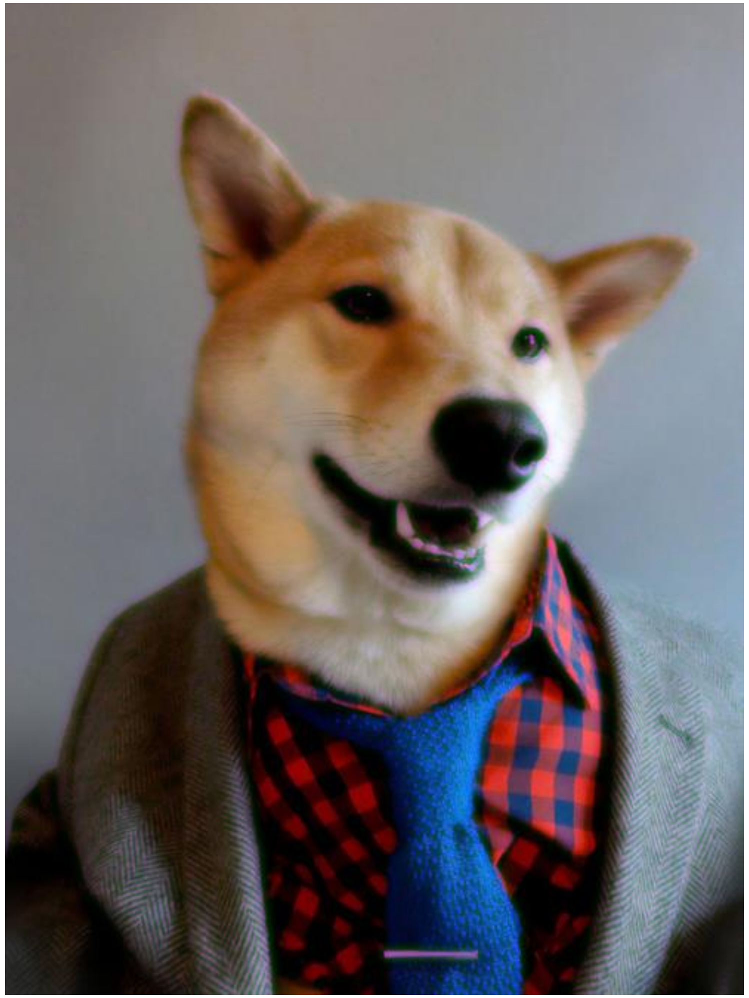
3D Convolution



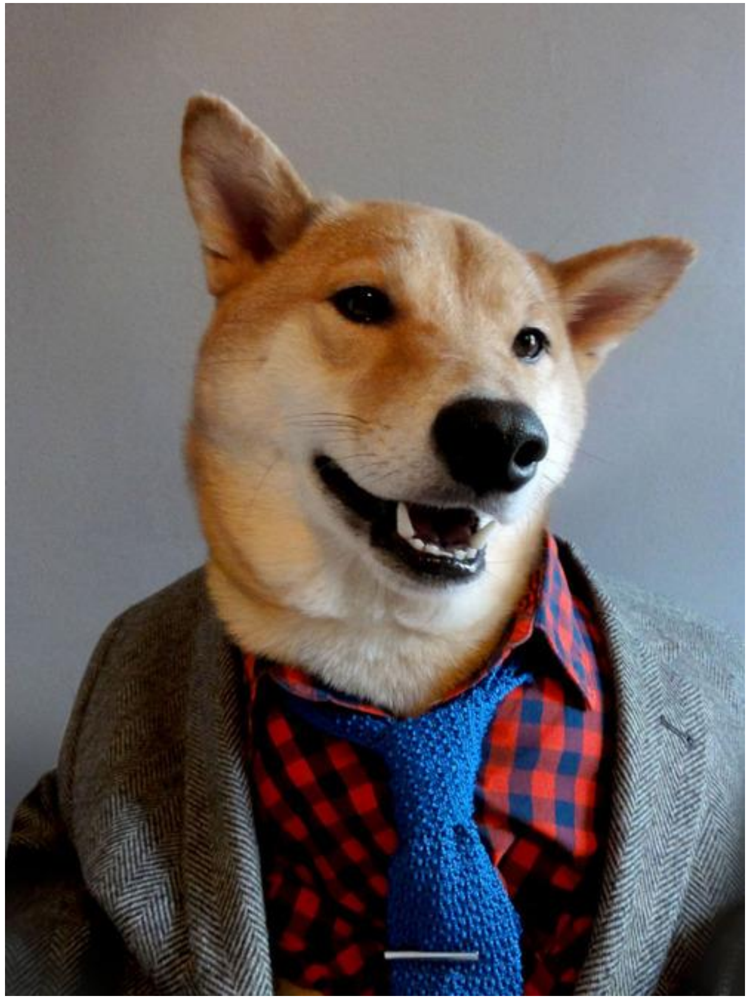
3D Convolution

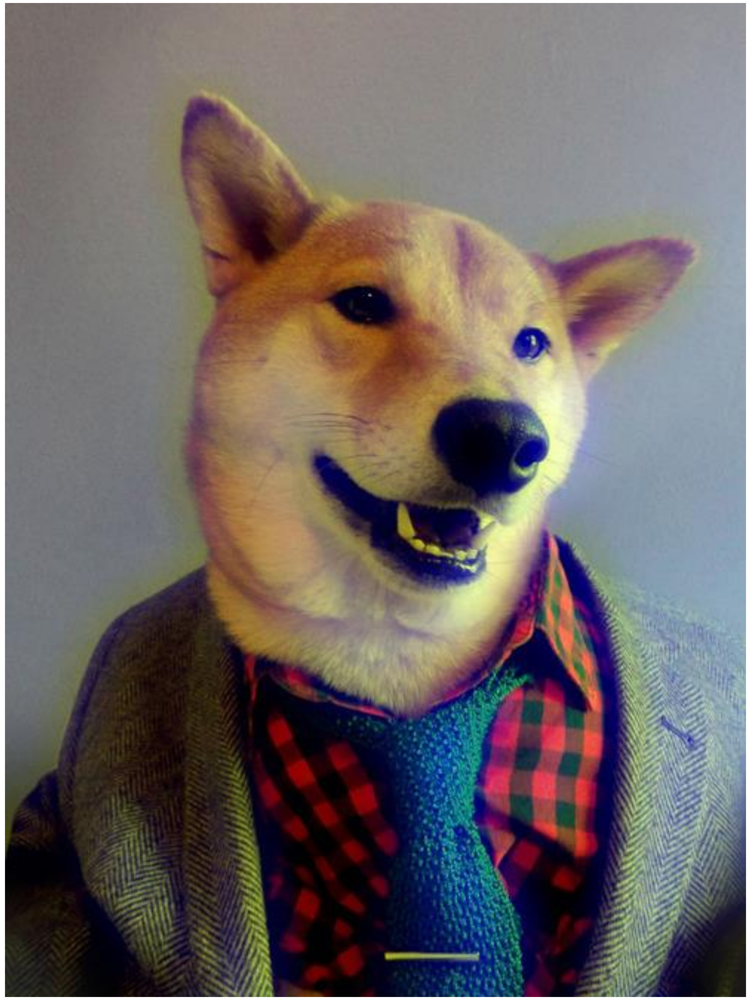
Gaussian filter

(higher variance on green channel)



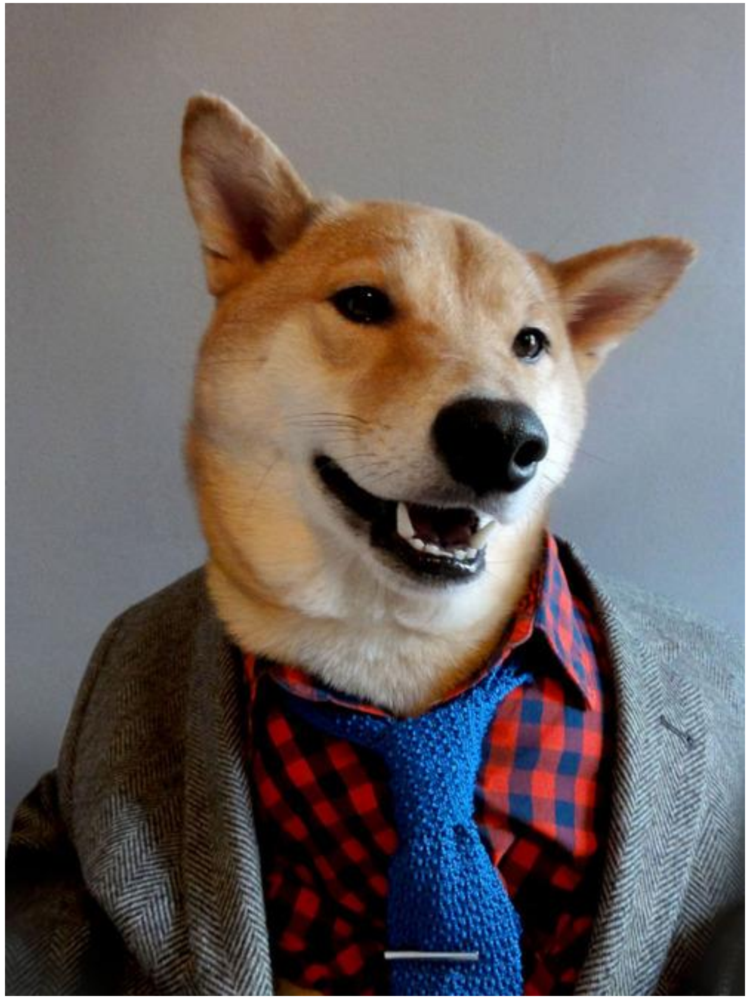
3D Convolution



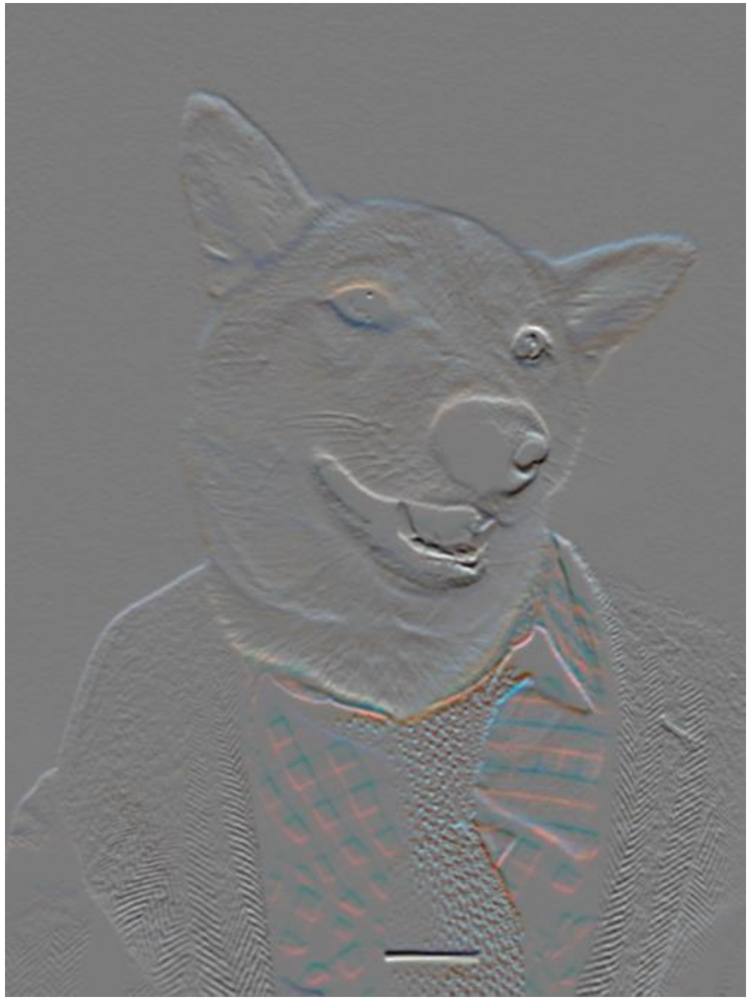


Sharpen the blue channel.

3D Convolution

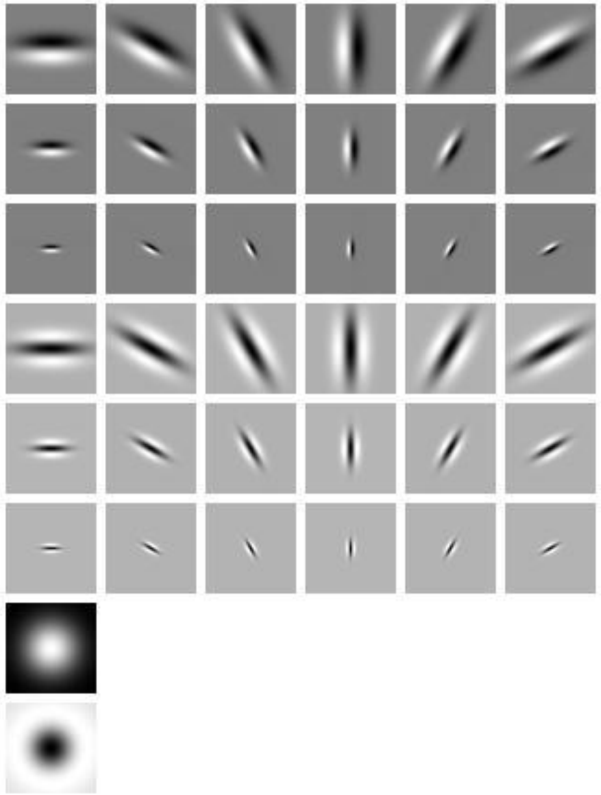


Gabor filter on each channel.



Filter Banks

- To characterize context, we used to use filter 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 (Part 5) 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 that make classification easy.
- Filter banks:
  - Make features for a vision problem by takin a bunch of convolutions.

- Next time:
  - A trick that lets you find gold and use the polynomial basis with d > 1.

Global and Local Features for Domain Adaptation

- 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.

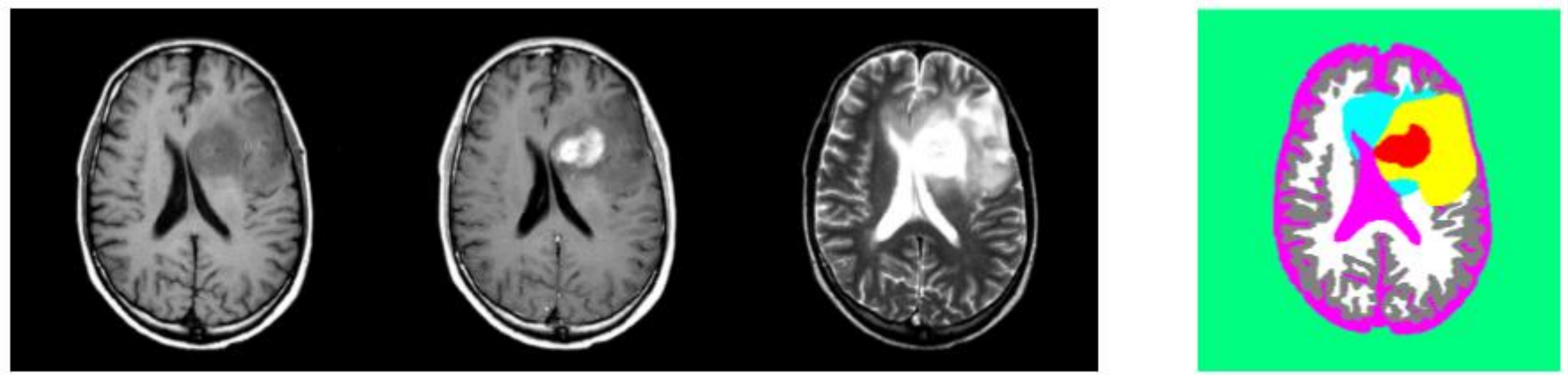
Global and Local Features for Domain Adaptation

- "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:

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

Image Coordinates

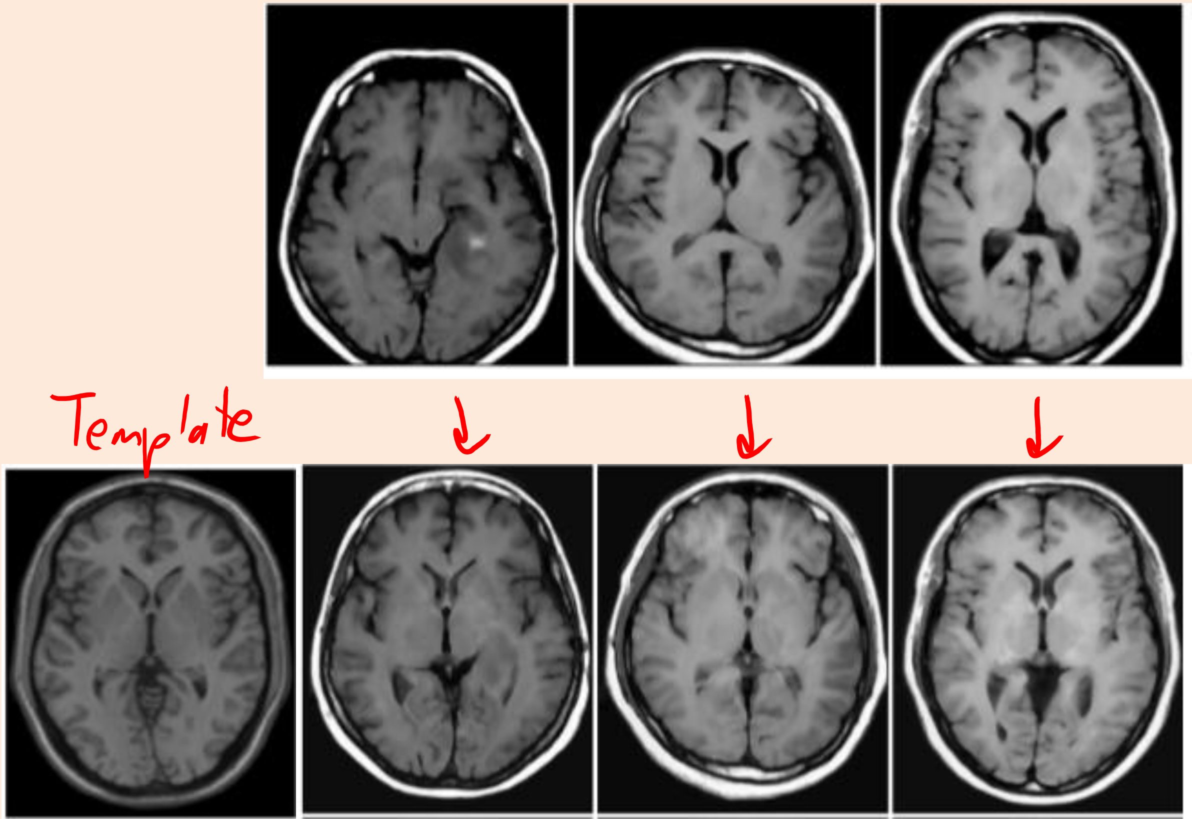
- Should we use the image coordinates?
  - E.g., the pixel is at location (124, 78) in the image.



- 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

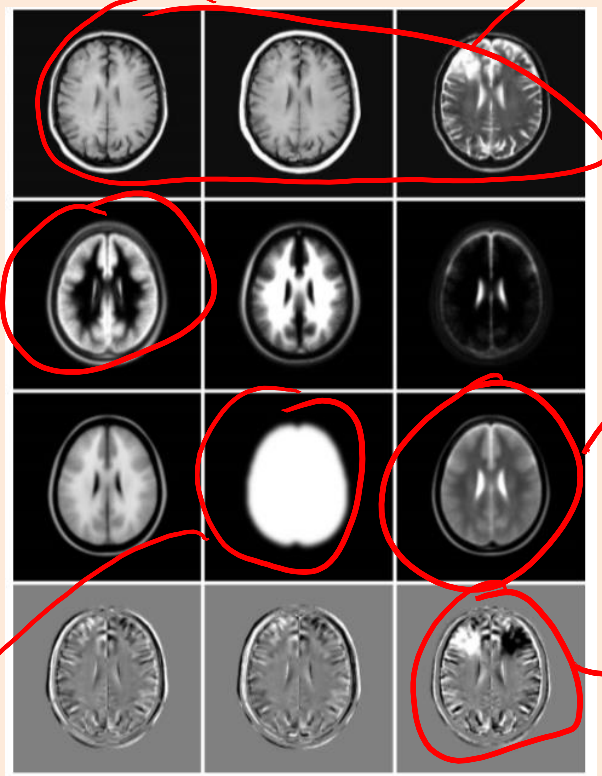
- The position in the image is important in brain tumour application.
  - But we didn't have much data, so coordinates didn't make sense.
- We aligned the images with a "template image".



(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.

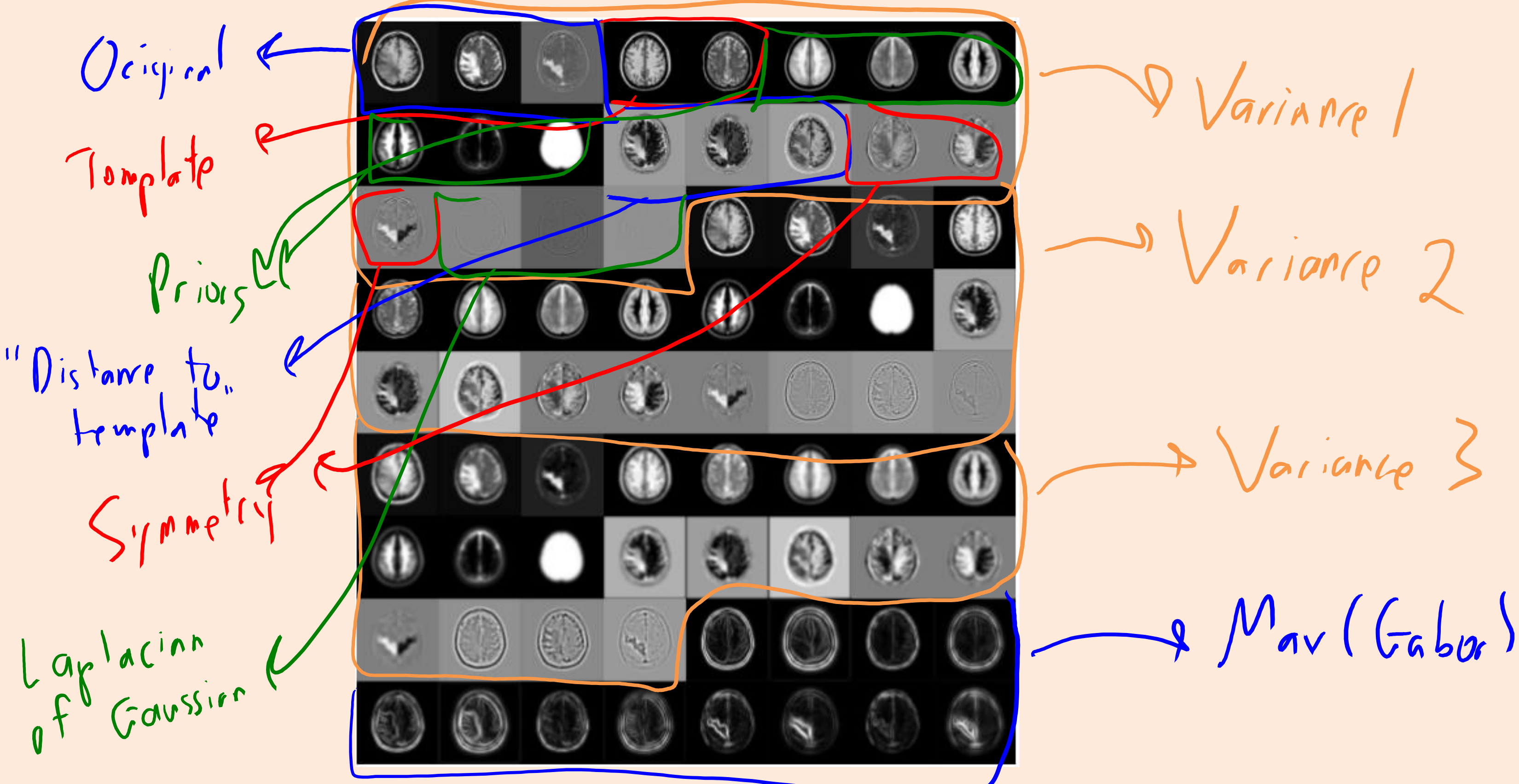
Actual pixel value of template image at this location.

Probability of being brain pixel.

Left-right symmetry difference.

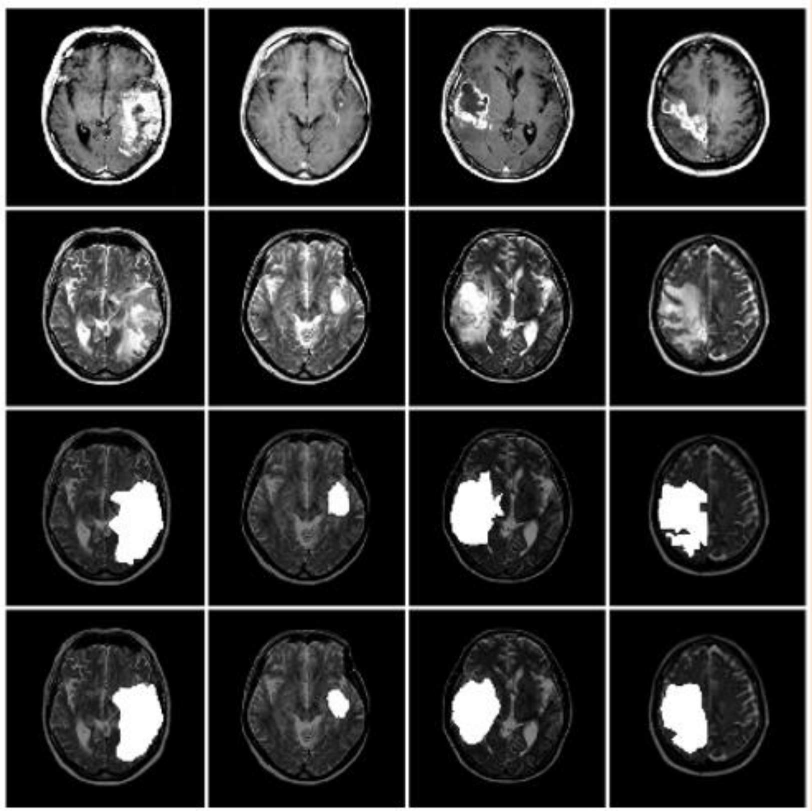
Motivation: Automatic Brain Tumor Segmentation

- 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).



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.

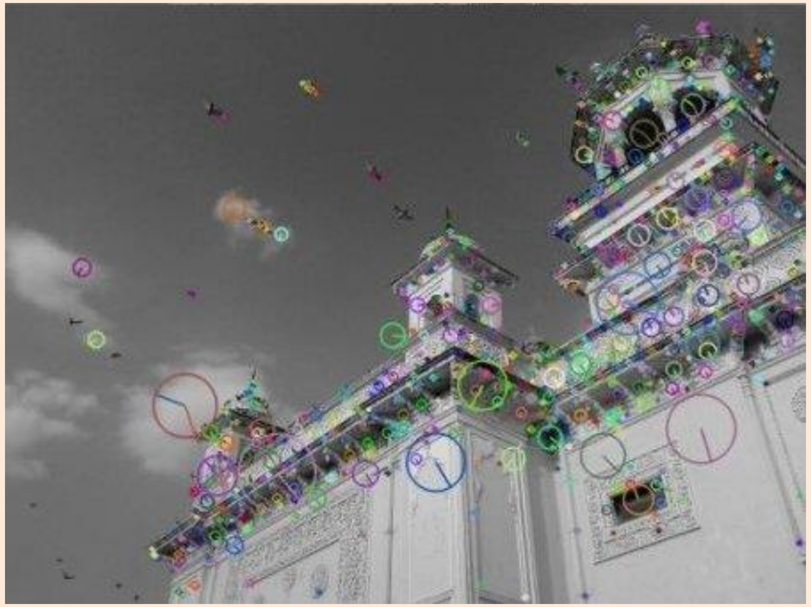
FFT implementation of convolution

- 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).

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