CPSC 340: Machine Learning and Data Mining

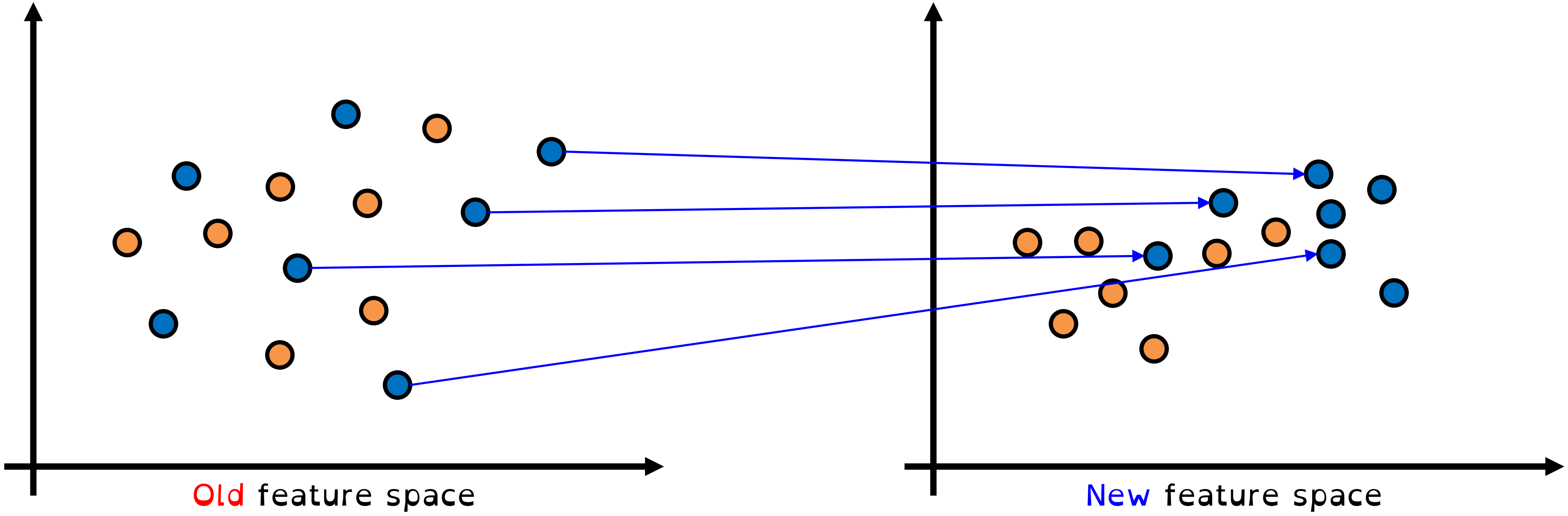
Text and Image Data

Summer 2021

Admin

- Assignment 4 due Monday.
- Assignment 5 out. Due next Friday.
- Midterm grading is almost done.
  - Expect ~80% average
- Final exam is Wednesday, June 23
  - Similar format
  - Will be challenging!

Last Time: Feature Engineering



- Hand-crafted transformation of feature space

In This Lecture

1. Representing Text Data (20 minutes)
2. Global vs. Local Features (10 minutes)
3. Representing Image Data (20 minutes)

Coming Up Next

REPRESENTING TEXT DATA

Text Example 1: Language Identification

- Consider data that doesn't look like this:

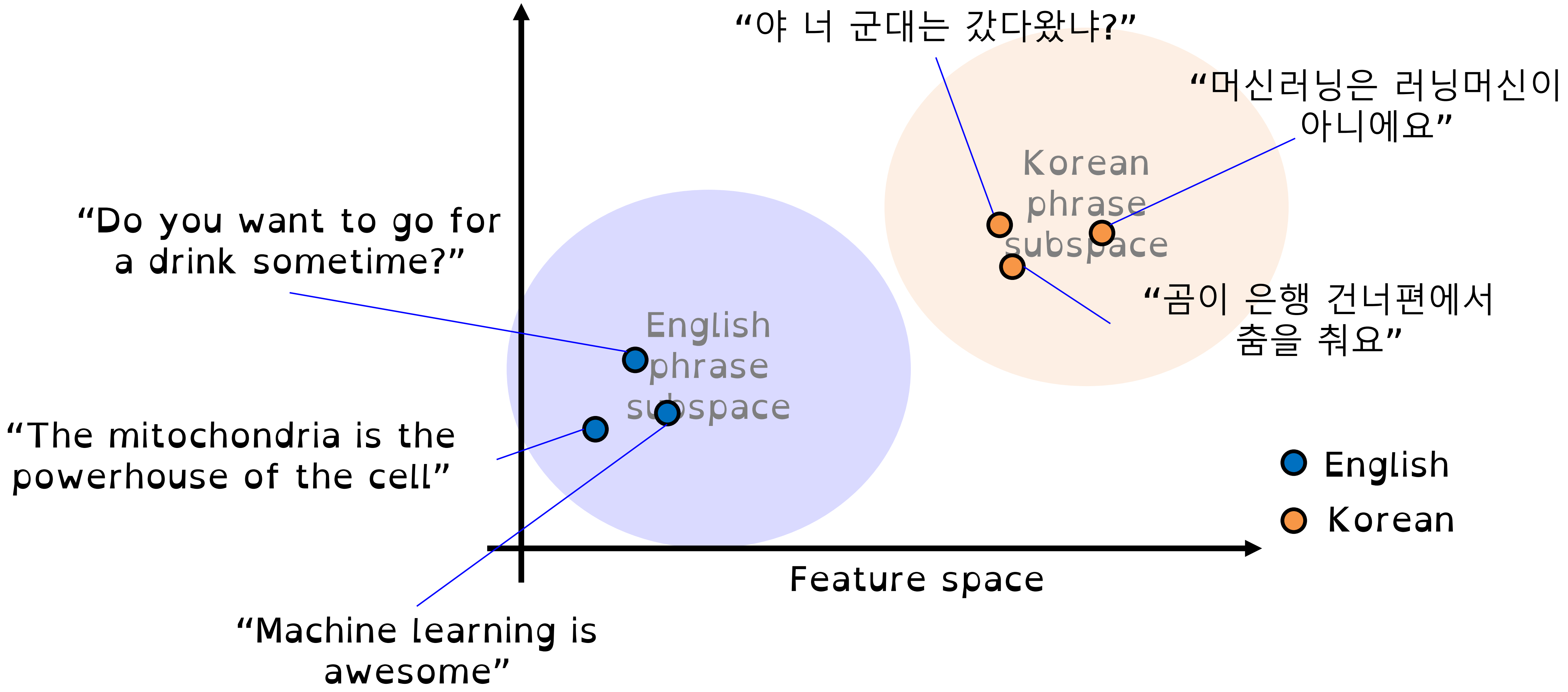
$$X = \begin{bmatrix} 0.5377 & 0.3188 & 3.5784 \\ 1.8339 & -1.3077 & 2.7694 \\ -2.2588 & -0.4336 & -1.3499 \\ 0.8622 & 0.3426 & 3.0349 \end{bmatrix}, \quad y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix},$$

- But instead looks like this:

$$X = \begin{bmatrix} \text{Do you want to go for a drink sometime?} \\ \text{J'achète du pain tous les jours.} \\ \text{Fais ce que tu veux.} \\ \text{There are inner products between sentences?} \end{bmatrix}, y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix}.$$

Q: How should we represent sentences using features?

Can We Build a Feature Space Like This?



“야 너 군대는 갔다왔냐?”

“머신러닝은 러닝머신이 아니에요”

Korean phrase subspace

“곰이 은행 건너편에서 춤을 춰요”

“Do you want to go for a drink sometime?”

English phrase subspace

“The mitochondria is the powerhouse of the cell”

“Machine learning is awesome”

Feature space

● English

● Korean

A (Bad) Universal Representation

- Treat character in position ‘j’ of the sentence as a categorical feature.
  - “fais ce que tu veux” => x_i = [f a i s ‘’ c e ‘’ q u e ‘’ t u ‘’ v e u x .]

    102 97 105

- “Pad” end of the sentence up to maximum #characters:

    0

  - “fais ce que tu veux” => x_i = [f a i s ‘’ c e ‘’ q u e ‘’ t u ‘’ v e u x . γ γ γ γ γ γ γ γ …]

- Advantage:
  - – No information is lost, KNN can eventually solve the problem.
- Disadvantage: throws out everything we know about language.
  - – We don’t have positional invariance:
    - Needs to learn that “veux” starting from any position indicates “French”.
    - Here, sentences are made of characters not words.
  - – High overfitting risk, you will need a lot of examples for this easy task.

Recall: Bag-of-Words

- Bag-of-words: represent sentences/documents by word counts:

The **International Conference on Machine Learning** (ICML) is the leading international academic conference in machine learning

ICML	International	Conference	Machine	Learning	Leading	Academic
1	2	2	2	2	1	1

- Bag of words loses a ton of information/meaning:
  - E.g. order of words, relative positions
  - But it easily solves language identification problem

Universal Representation vs. Bag of Words

- Why is bag of words better than “array of characters” here?
  - It needs less data because it captures invariances for the task:
    - Most features give strong indication of one language or the other.
    - It doesn’t matter *where* the French words appear.
  - It overfits less because it throws away irrelevant information.
    - Exact sequence of words isn’t particularly relevant here.

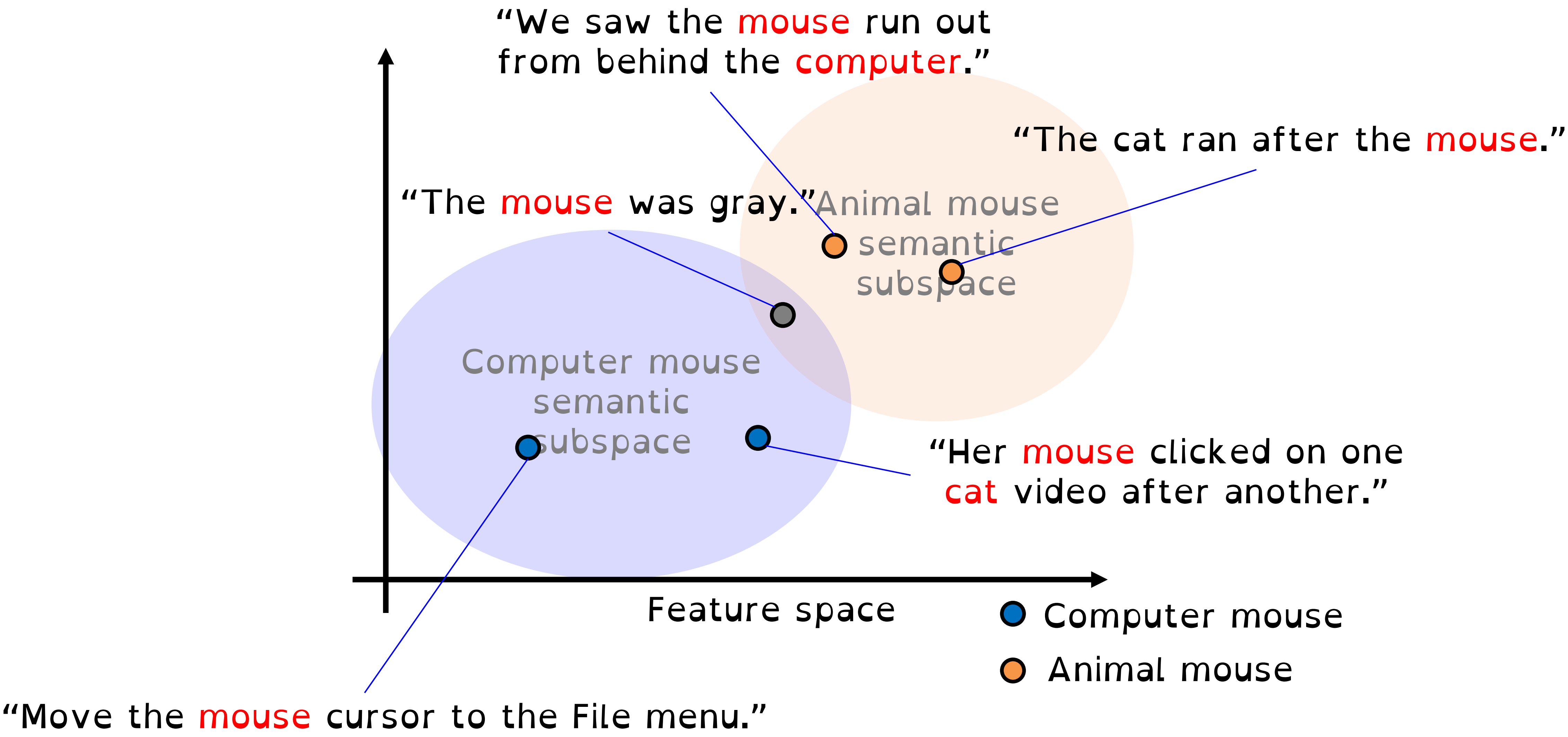
Text Example 2: Word Sense Disambiguation

- Consider the following two sentences:
  - “The cat ran after the mouse.”
  - “Move the mouse cursor to the File menu.”

- Word sense disambiguation (WSD): classify “meaning” of a word:
  - A surprisingly difficult task.

- You can do ok with bag of words, but it will have problems:
  - “Her mouse clicked on one cat video after another.”
  - “We saw the mouse run out from behind the computer.”
  - “The mouse was gray.” (ambiguous without more context)

Can We Build a Feature Space Like This?



“We saw the mouse run out from behind the computer.”

“The cat ran after the mouse.”

“The mouse was gray.”

“Her mouse clicked on one cat video after another.”

“Move the mouse cursor to the File menu.”

Animal mouse semantic subspace
Computer mouse semantic subspace
Feature space
● Computer mouse
● Animal mouse

Bigrams and Trigrams

- A bigram is an ordered set of two words:
  - Like “computer mouse” or “mouse ran”.
- A trigram is an ordered set of three words:
  - Like “cat and mouse” or “clicked mouse on”.

- These give more context/meaning than bag of words:
  - Includes neighbouring words as well as order of words.
  - Trigrams are widely-used for various language tasks.

- General case is called n-gram.
  - Unfortunately, coupon collecting becomes a problem with larger ‘n’.

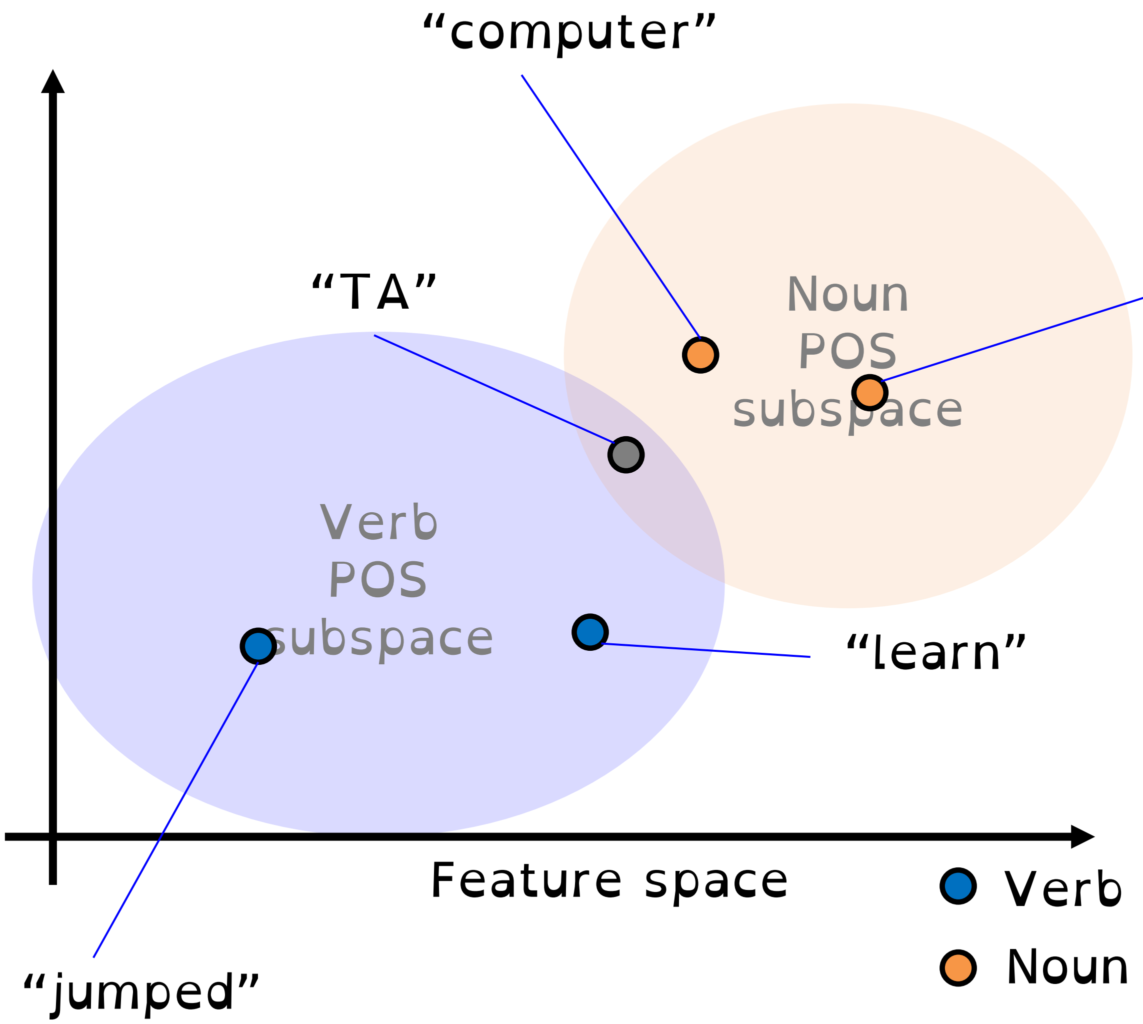
Text Example 3: Part of Speech (POS) Tagging

- Consider problem of finding the verb in a sentence:
  - “The 340 students jumped at the chance to hear about POS features.”

- Part of speech (POS) tagging is the problem of labeling all words.
  - >40 common syntactic POS tags.
  - Current systems have ~97% accuracy on standard (“clean”) test sets.
  - You can achieve this by applying a “word-level” classifier to each word.
    - That independently classifies each word with one of the 40 tags.

- What features of a word should we use for POS tagging?

Can We Build a Feature Space Like This?



POS Features

- Regularized multi-class logistic regression with these features gives ~97% accuracy:
  - Categorical features whose domain is all words (“lexical” features):
    - The word (e.g., “jumped” is usually a verb).
    - The previous word (e.g., “he” hit vs. “a” hit).
    - The previous previous word.
    - The next word.
    - The next next word.
  - Categorical features whose domain is combinations of letters (“stem” features):
    - Prefix of length 1 (“what letter does the word start with?”)
    - Prefix of length 2.
    - Prefix of length 3.
    - Prefix of length 4 (“does it start with JUMP?”)
    - Suffix of length 1.
    - Suffix of length 2.
    - Suffix of length 3 (“does it end in ING?”)
    - Suffix of length 4.
  - Binary features (“shape” features):
    - Does word contain a number?
    - Does word contain a capital?
    - Does word contain a hyphen?

Ordinal Features

- Categorical features with an ordering are called ordinal features.

Rating
Bad
Very Good
Good
Good
Very Bad
Good
Medium

→

Rating
2
5
4
4
1
4
3

- If using decision trees, makes sense to replace with numbers.
  - Captures ordering between the ratings.
  - A rule like (rating ≥ 3) means (rating ≥ Good), which make sense.

Ordinal Features

- With linear models, “convert to number” assumes ratings are equally spaced.
  - “Bad” and “Medium” distance is similar to “Good” and “Very Good” distance.
- One alternative that preserves ordering with binary features:

Rating
Bad
Very Good
Good
Good
Very Bad
Good
Medium

→

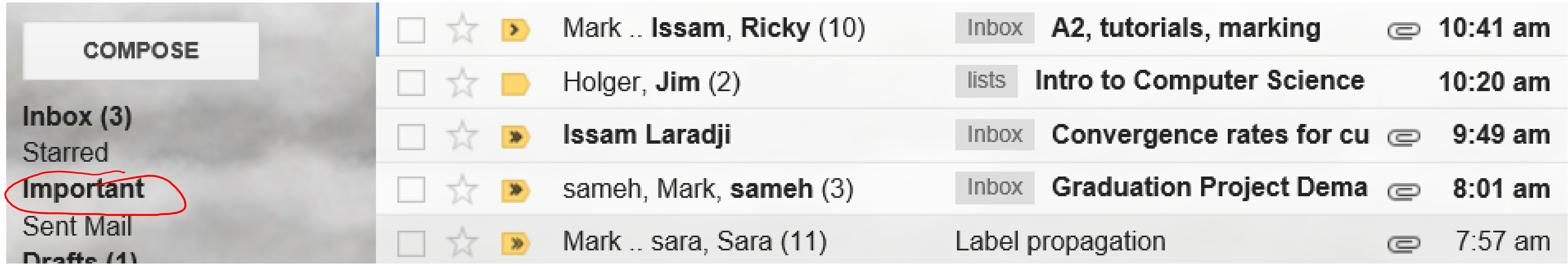
≥ Bad	≥ Medium	≥ Good	Very Good
1	0	0	0
1	1	1	1
1	1	1	0
1	1	1	0
0	0	0	0
1	1	1	0
1	1	0	0

- Regression weight w_{medium} represents:
  - “How much medium changes prediction over bad”.

Coming Up Next

GLOBAL AND LOCAL FEATURES

Motivation: “Personalized” Important Emails



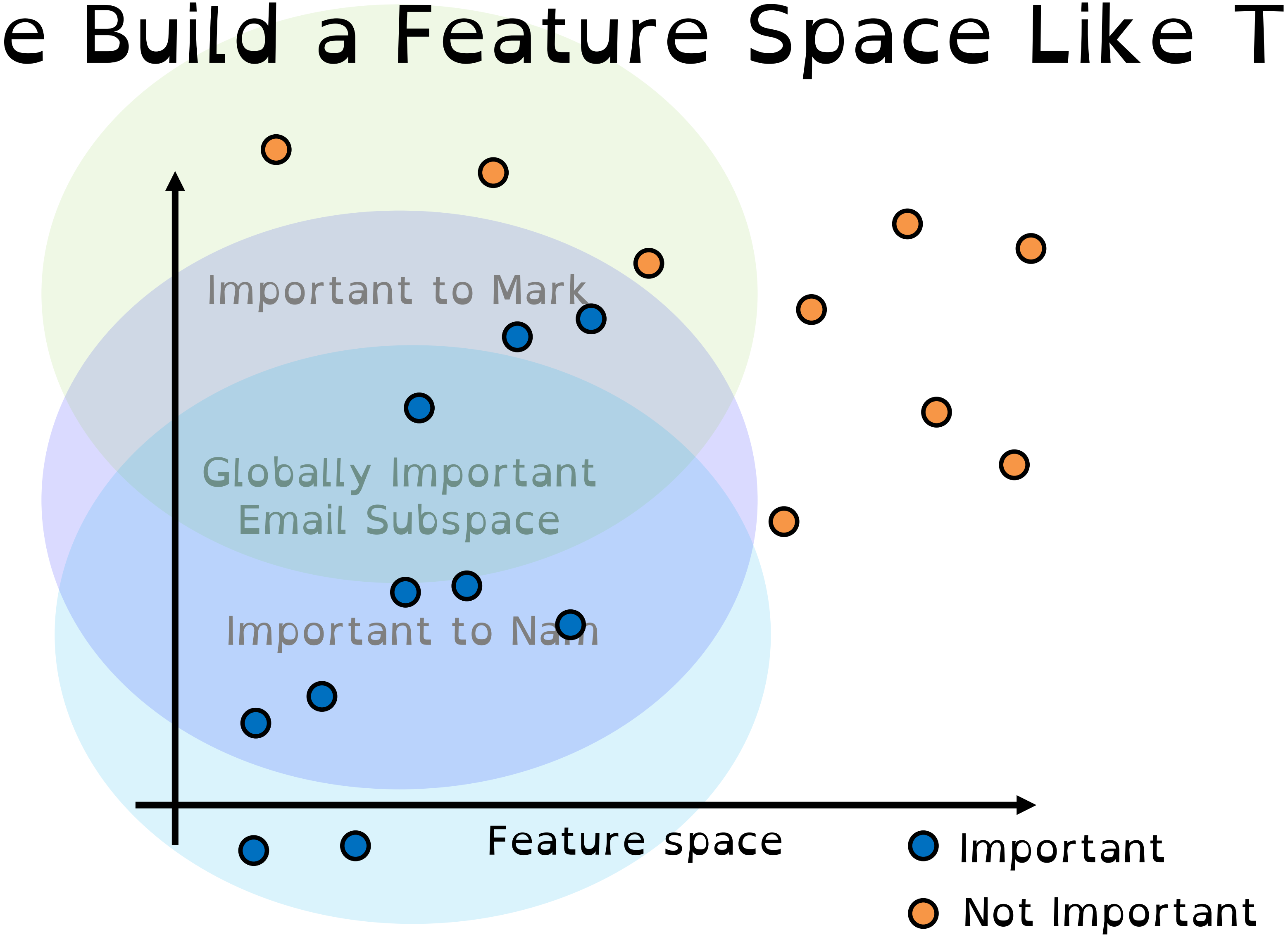
Mark .. **Issam**, **Ricky** (10)	Inbox	**A2, tutorials, marking**	**10:41 am**
Holger, **Jim** (2)	lists	**Intro to Computer Science**	**10:20 am**
Issam Laradji	Inbox	**Convergence rates for cu**	**9:49 am**
sameh, Mark, **sameh** (3)	Inbox	**Graduation Project Dema**	**8:01 am**
Mark .. sara, Sara (11)		Label propagation	7:57 am

- Features: bad of words, trigrams, regular expressions, and so on.

- There might be some “globally” important messages:
  - “This is your mother, something terrible happened, give me a call ASAP.”

- But your “important” message may be unimportant to others.
  - Similar for spam: “spam” for one user could be “not spam” for another.

Can We Build a Feature Space Like This?



“Global” and “Local” Features

- Consider the following weird feature transformation:

“340”
1
1
1
0
1

⇒

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

did “340” appear in this e-mail?

if “340” appeared in this e-mail,
who was it addressed to?

- First feature will increase/decrease importance of “340” for every user (including new users).
- Second (categorical feature) increases/decreases importance of “340” for specific users.
  - Lets us learn more about specific users where we have a lot of data

“Global” and “Local” Features

- Recall we usually represent categorical features using “1 of k” binaries:

“340”
1
1
1
0
1

⇒

“340” (any user)	“340” (user = Nam)	“340” (user = Mark)
1	1	0
1	1	0
1	0	1
0	0	0
1	0	0

Contribute to “globally important” score

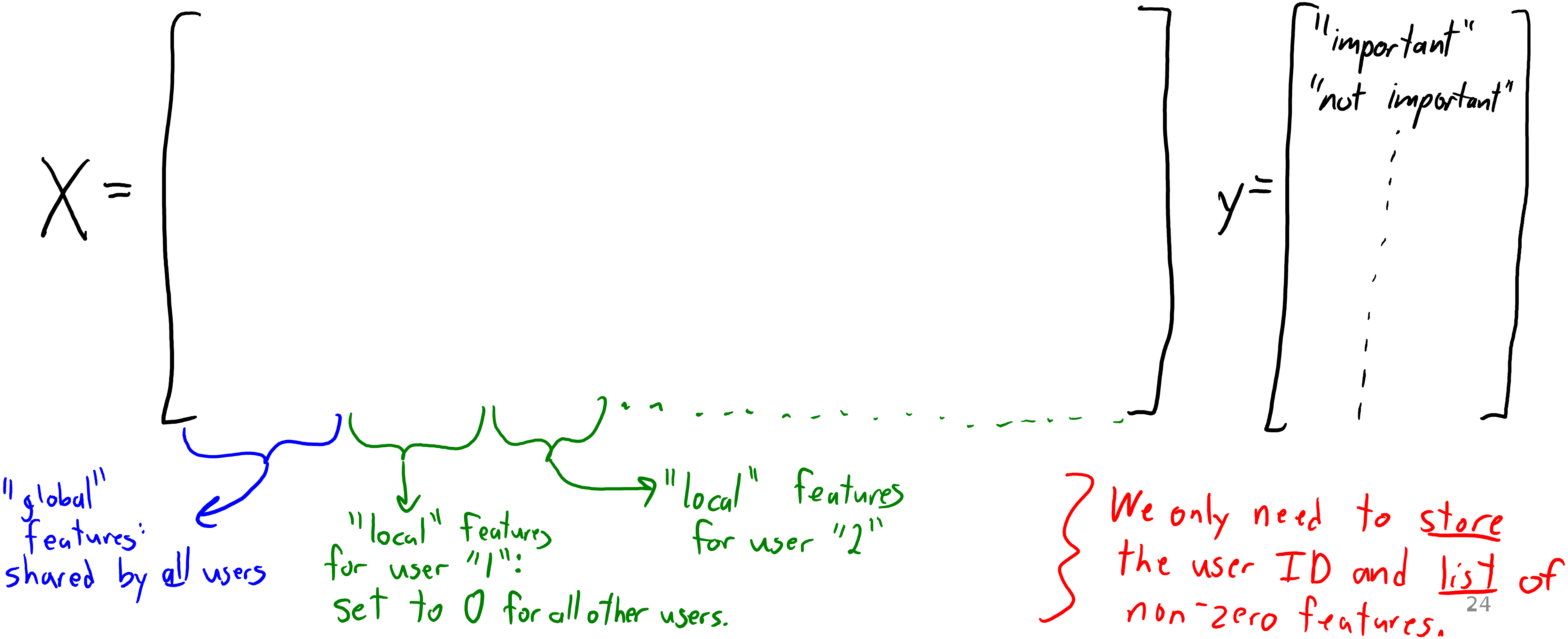
Contribute to “important to Nam” score

Contribute to “important to Mark” score

- First feature “moves the line up” for all users.
- Second feature “moves the line up” when the email is to user 1.
- Third feature “moves the line up” when the email is to user 2.

The Big Global/Local Feature Table for E-mails

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



"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 Email 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.

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

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

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

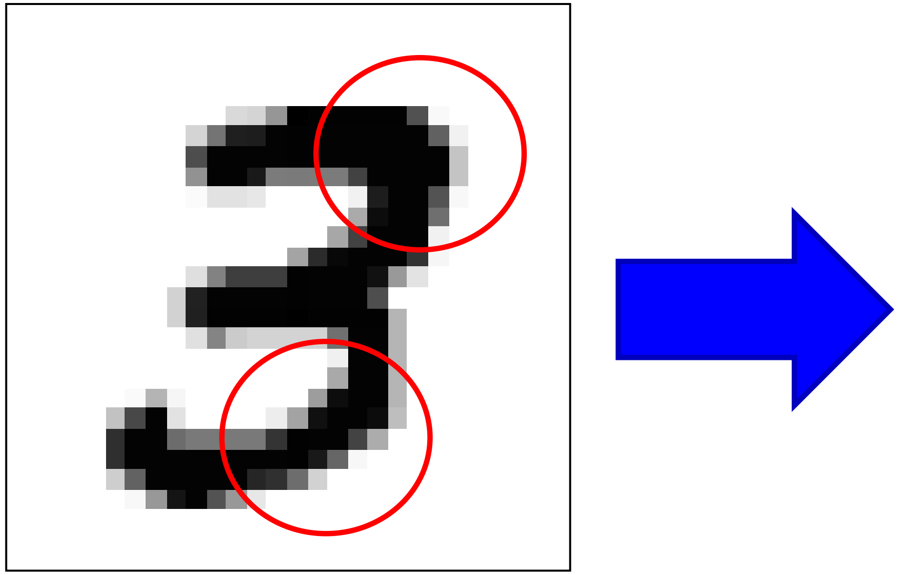
  - What is important to *this* user?
- Gmail system: classification with logistic regression.
  - Trained with a variant of stochastic gradient (later).

Coming Up Next

REPRESENTING IMAGE DATA

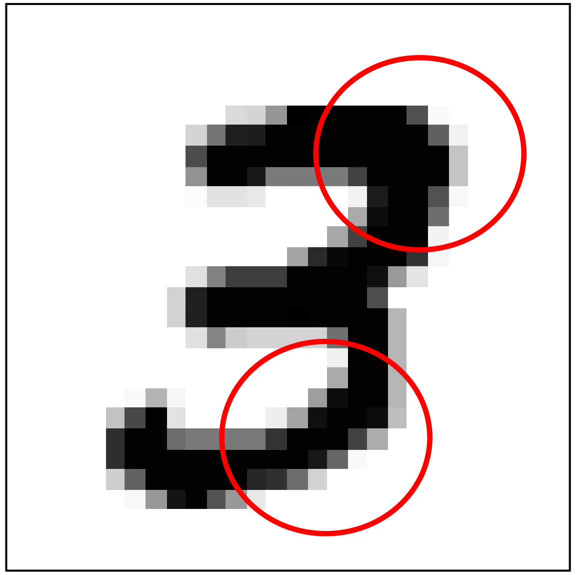
“Signal” View of Images

- Assumption: image is a 2-dimensional signal
  - “signal” := as opposed to noise, temporal/spatial information that’s relevant for prediction
  - But signals are optimized for viewing, not analyzing!



Q: Does this “sea of numbers” capture the fact that “3” is made of two round shapes?

What Makes a Model “Smart”?



- If your model is “smart”, then it should extract relevant information from the raw signal!
  - “Learning the features”
  - Models need to be complex to do this

Q: What if I want to use simple models?

“Convolution”

- Idea: images contain certain “patterns”
  - Images are linear combinations of these patterns (bases)

- “Convolution” := detect patterns and their activations across raw format of data
  - e.g. detect certain frequencies in given waveform
  - e.g. detect round shapes in handwriting images

- Idea: use these activations as features
  - Often expressive enough for simpler models to perform well.
  - If I have regularization, irrelevant features will be suppressed

Coming Up Next

1D CONVOLUTION

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

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

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

- 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’:

Convolution: sliding window with dot-products

1D Convolution

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

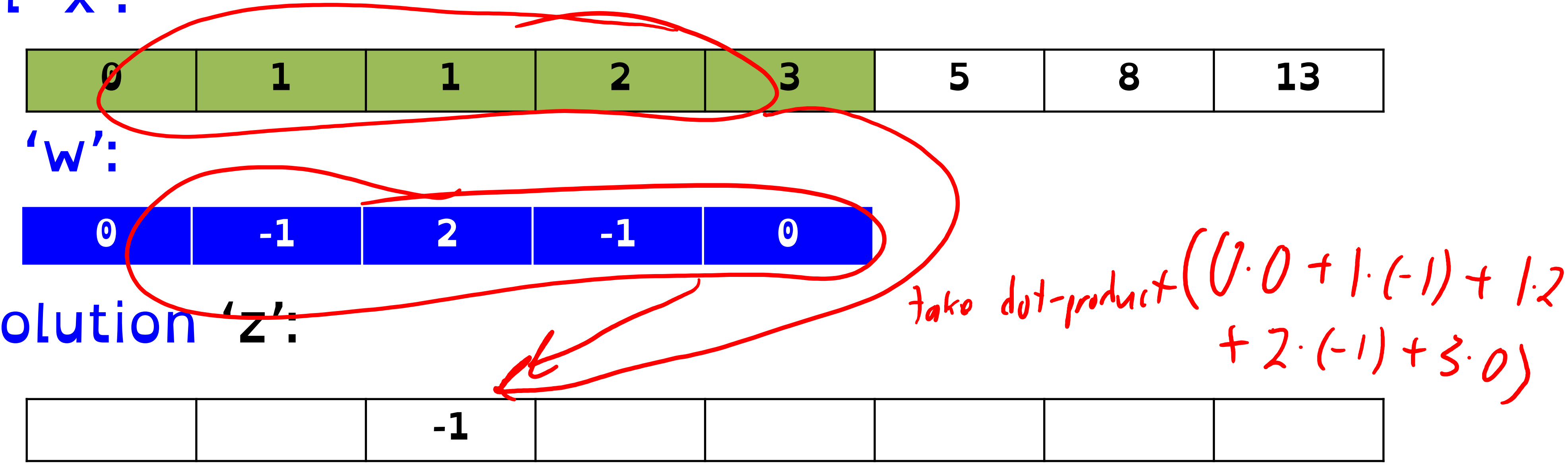
0	1	1	2	3	5	8	13

  - 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

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

- Examples:
  - “Identity”

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

    $0 \cdot x_0 + 1 \cdot x_1 + 0 \cdot x_2 \qquad 0 \cdot x_1 + 1 \cdot x_2 + 0 \cdot x_3$
  - “Translation”

    ↳ $w = [0 \quad 0 \quad 1]$ $\qquad z = [1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13 \quad ?]$

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

1D Convolution Examples

- Examples:
  - "Identity"

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

  - "Local Average"

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

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

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

Boundary Issue

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

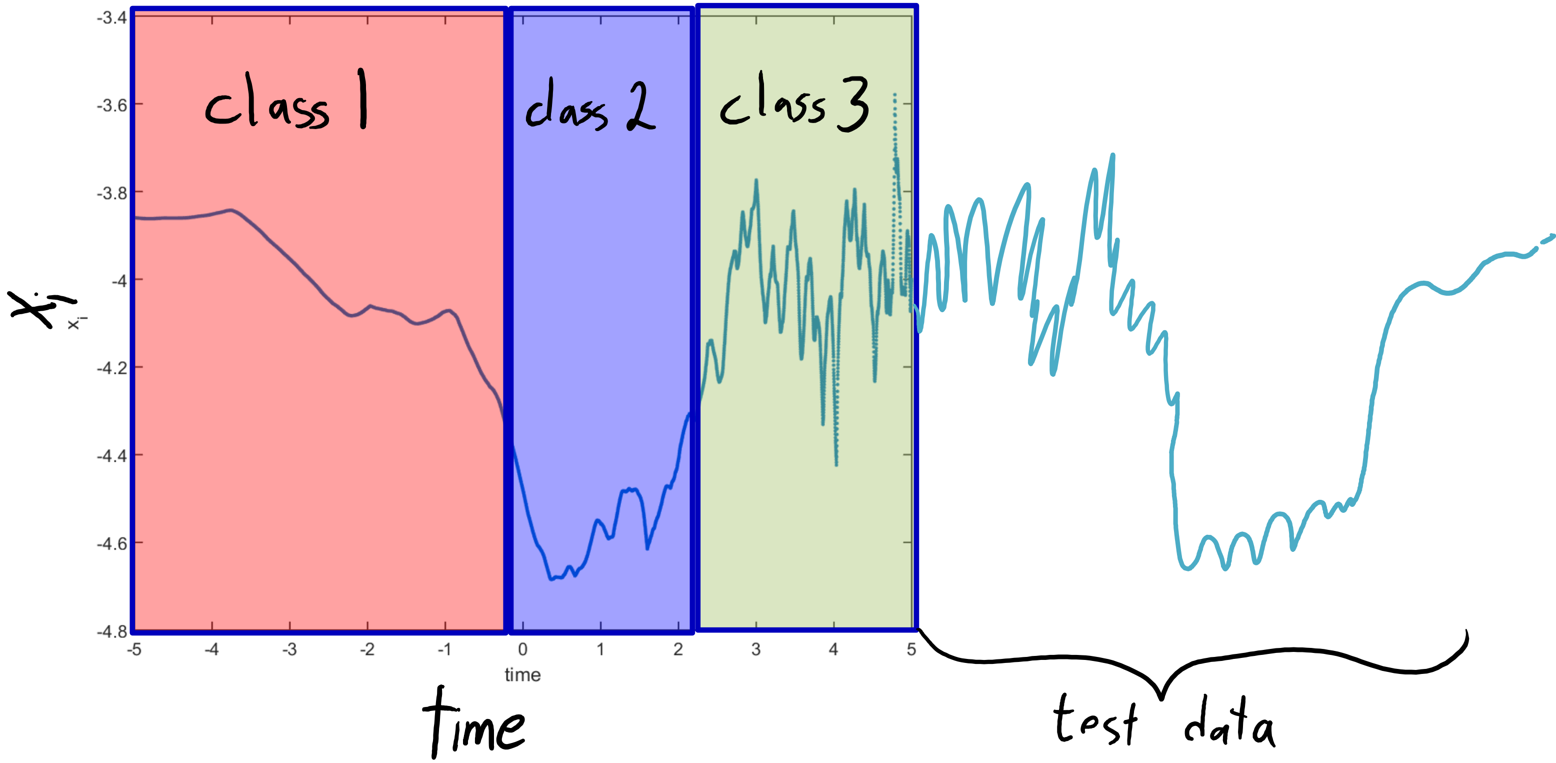
If $x = [0\ 1\ 1\ 2\ 3\ 5\ 8\ 13]$ and $w = [\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}]$ then $z = [?\ \frac{2}{3}\ 1\frac{1}{3}\ 2\ 3\frac{1}{3}\ 5\frac{1}{3}\ 8\frac{2}{3}\ ?]$

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

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

Representing Neighbourhoods with Convolutions

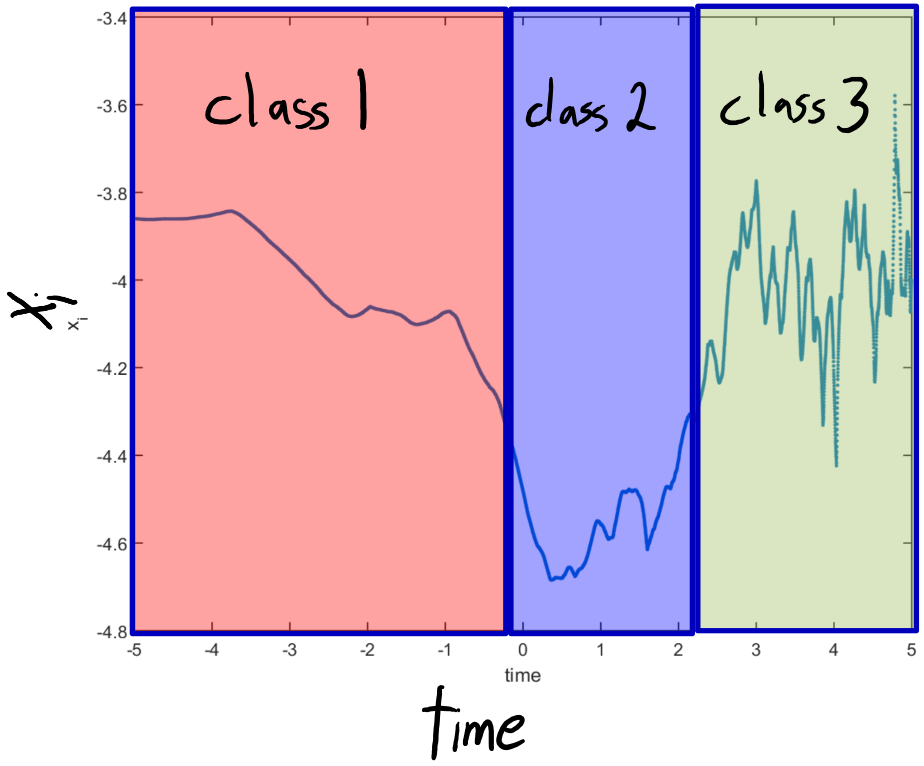
- Consider a 1D dataset:
  - Want to classify each time into y_i in {1,2,3}.
  - Example: speech data.

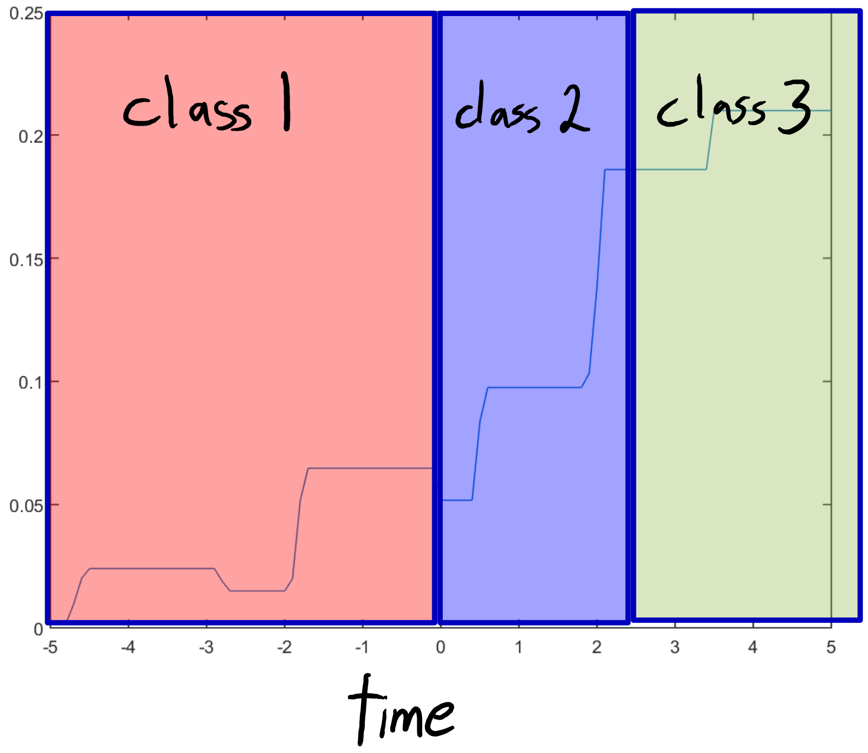


- Easy to distinguish class 2 from the other classes (x_i are smaller).
- Harder to distinguish between class 1 and class 3 (similar x_i range).
  - But convolutions can represent that class 3 is in “spiky” region.

Representing Neighbourhoods with Convolutions

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



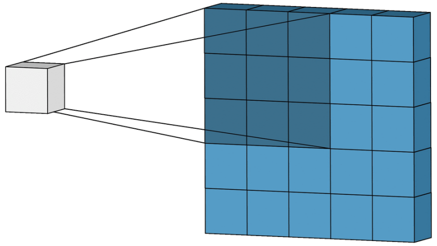


- Easy to distinguish the 3 classes with these 2 features.

Coming Up Next

2D CONVOLUTION

Images and Convolution

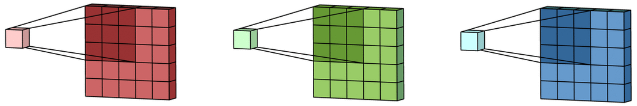


- 2D convolution:
  - Signal ‘x’ is the pixel intensities in an ‘n’ by ‘n’ image.
  - Filter ‘w’ is the pixel intensities in a ‘2m+1’ by ‘2m+1’ ima
- The 2D convolution is given by:

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

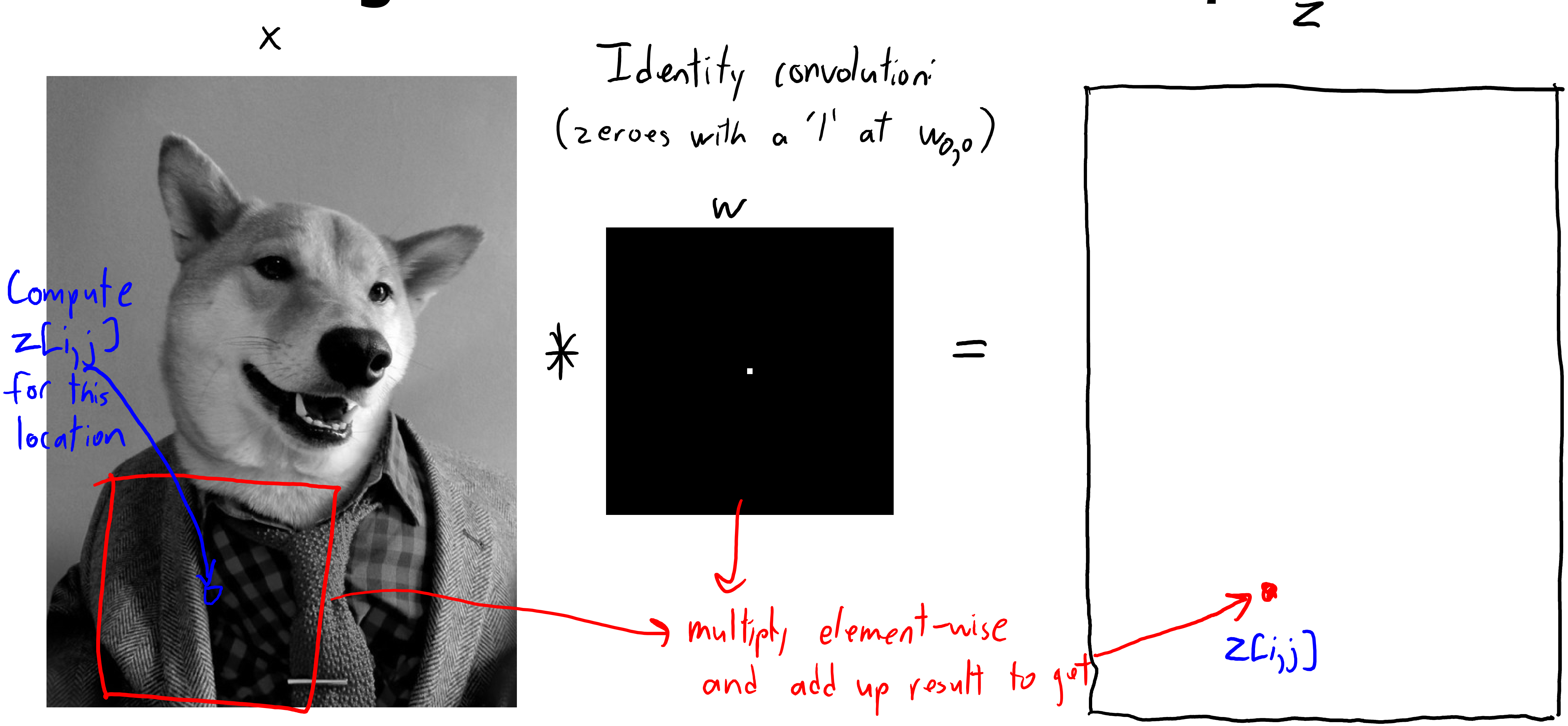
- 3D and higher-order convolutions are defined similarly.

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



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

Image Convolution Examples



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

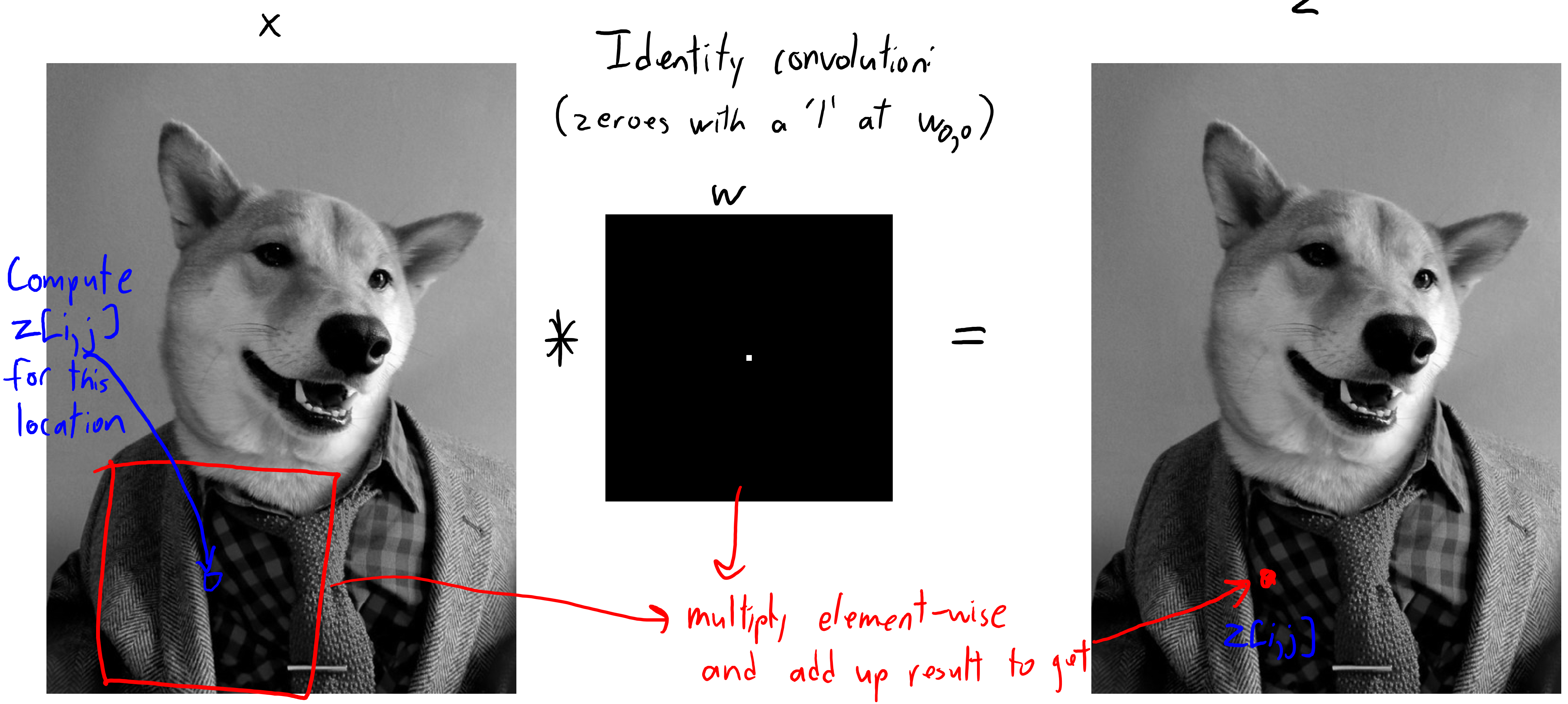
Compute z[i,j] for this location

multiply element-wise and add up result to get

z[i,j]

Image Convolution Examples

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

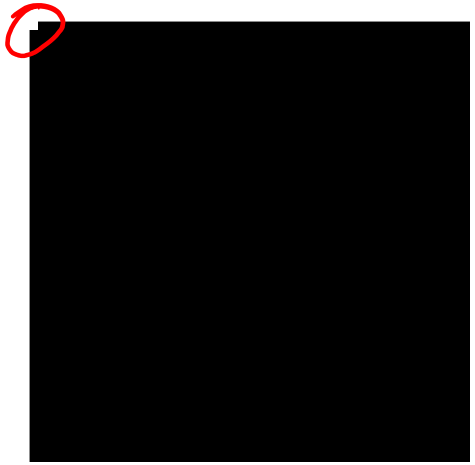


multiply element-wise and add up result to get

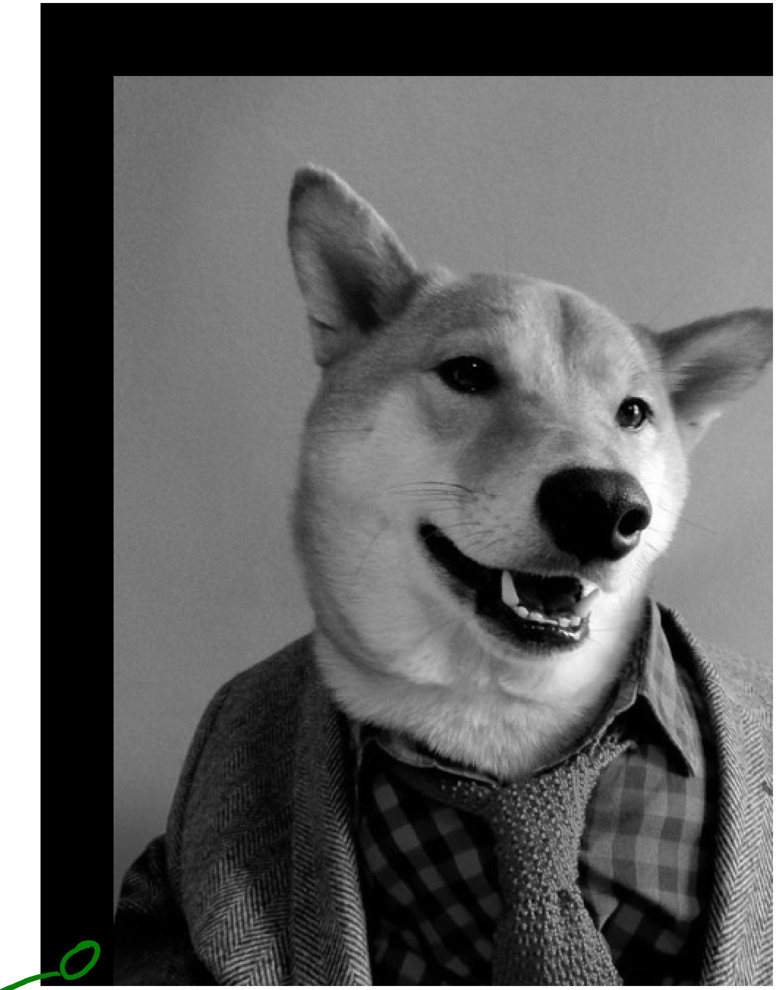
Image Convolution Examples

Translation Convolution:

*



=



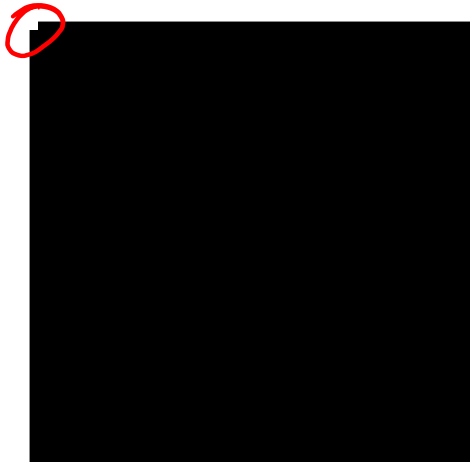
Boundary: "zero"

0

Image Convolution Examples

$*$

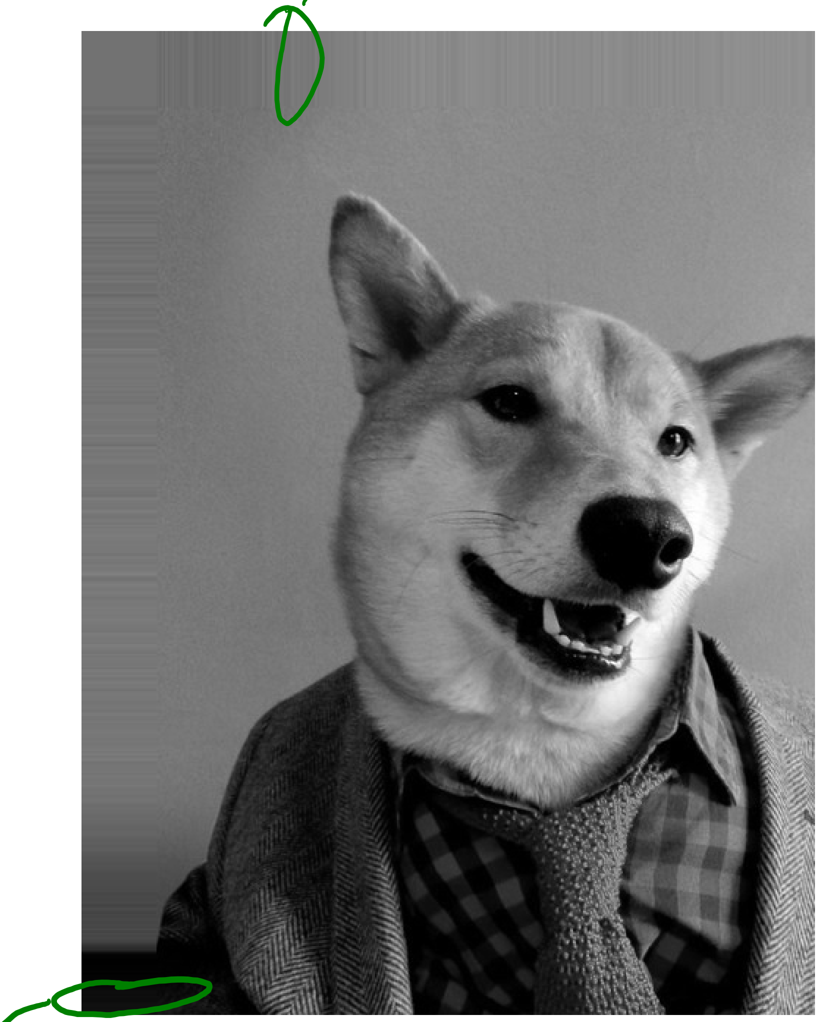
Translation Convolution:



Boundary: "replicate"

$=$

repeats

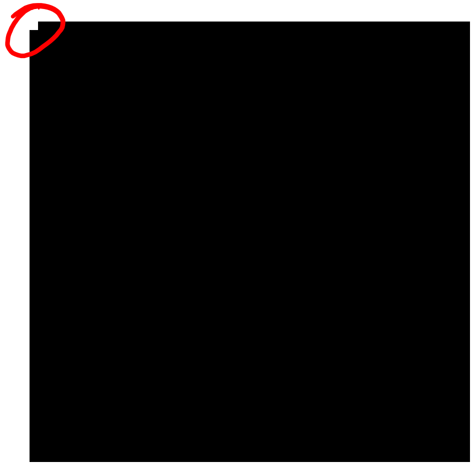


repeats

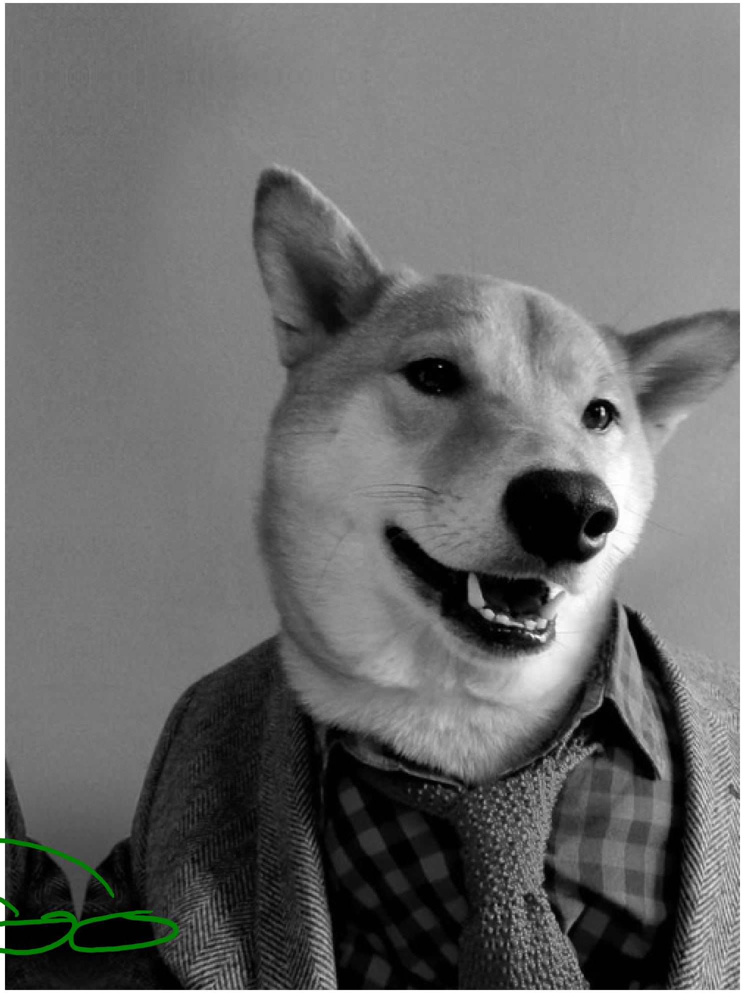
Image Convolution Examples

Translation Convolution:

$*$



$=$



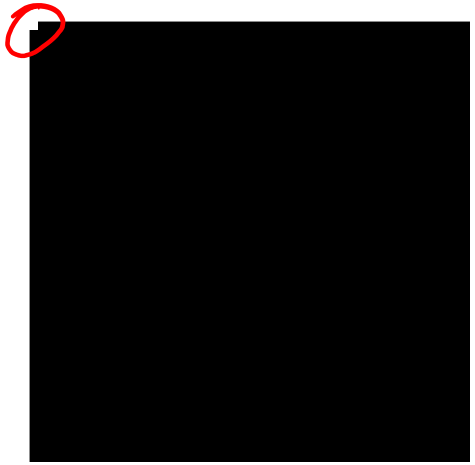
Boundary: "mirror"

flips

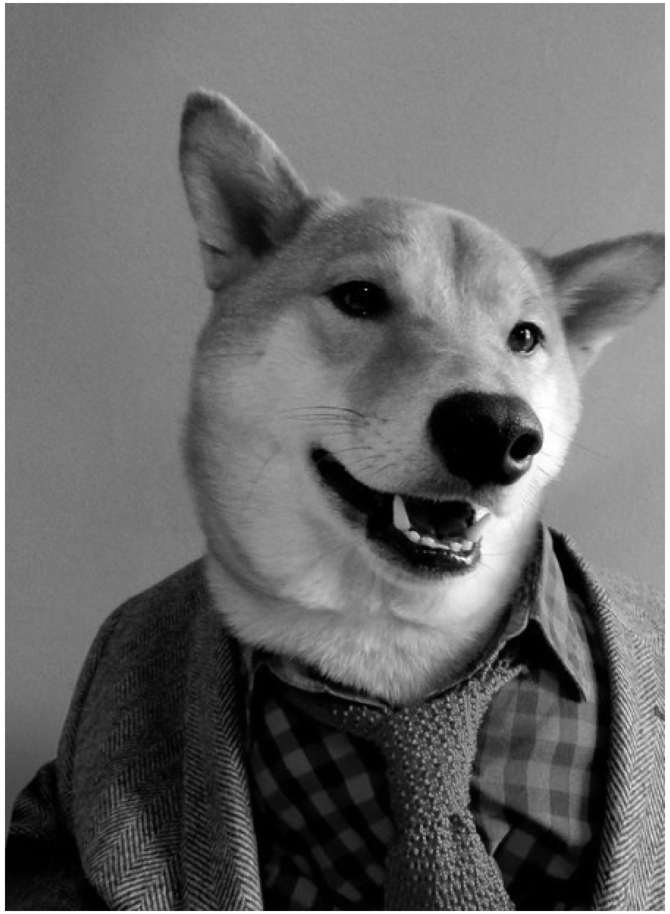
Image Convolution Examples

Translation Convolution:

$*$

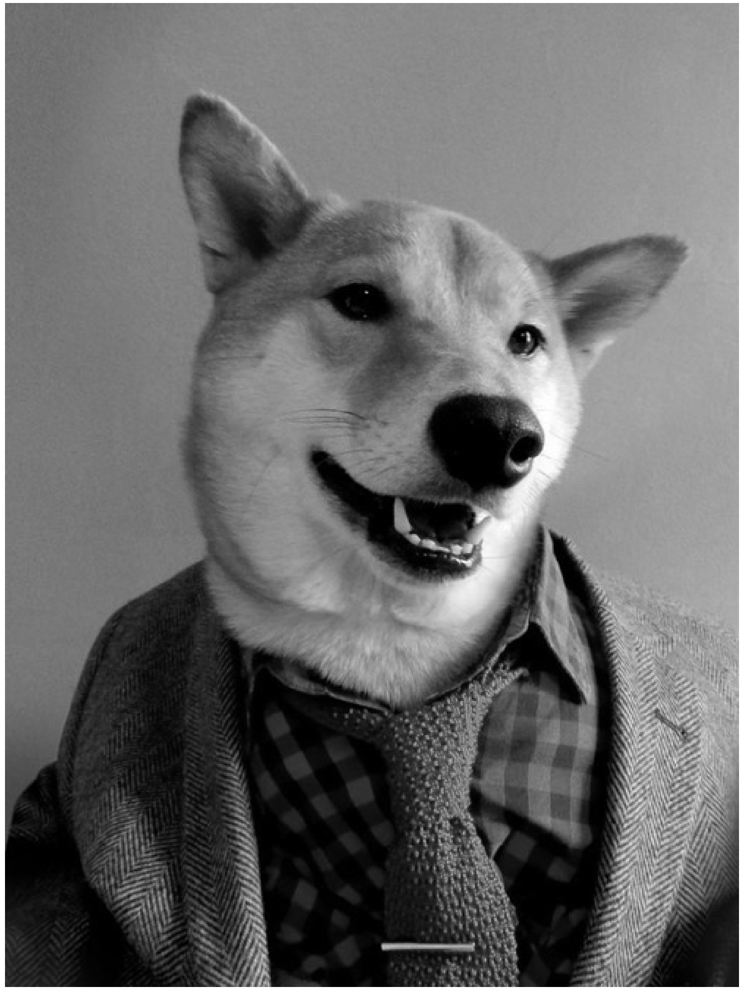


$=$



Boundary: "ignore"

Image Convolution Examples



Average convolution:

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

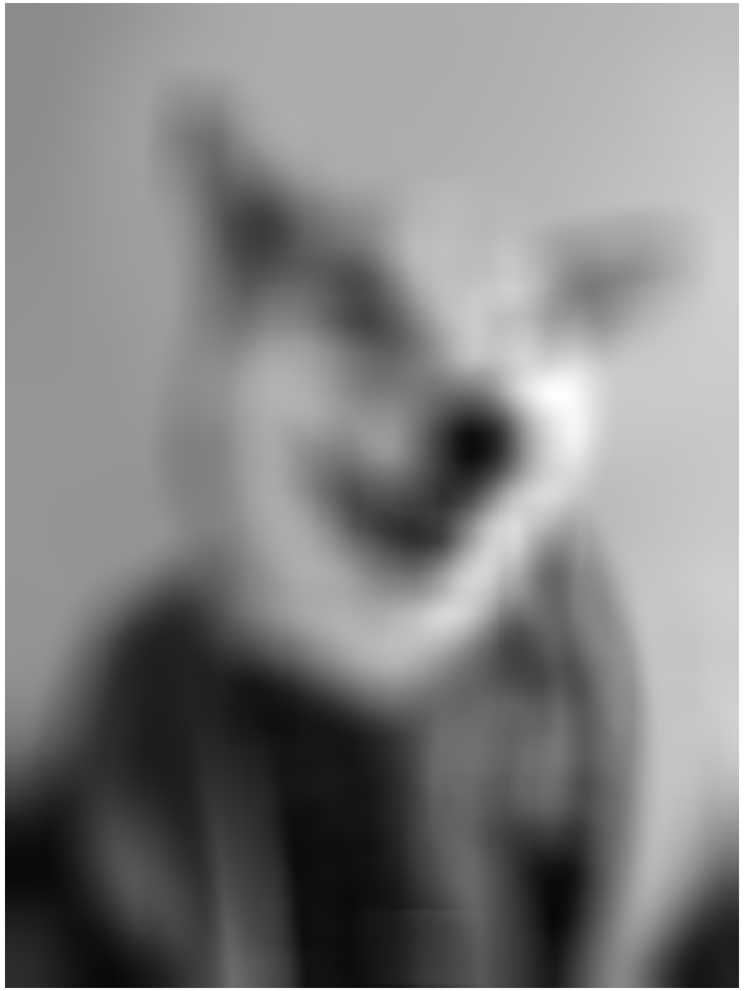
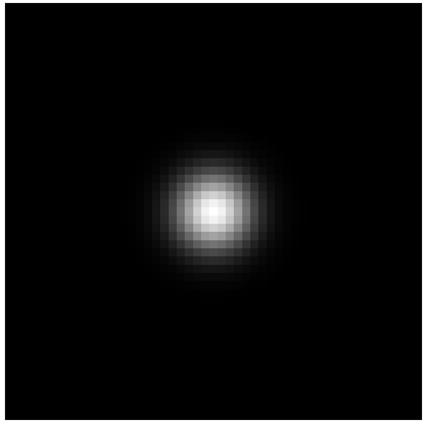


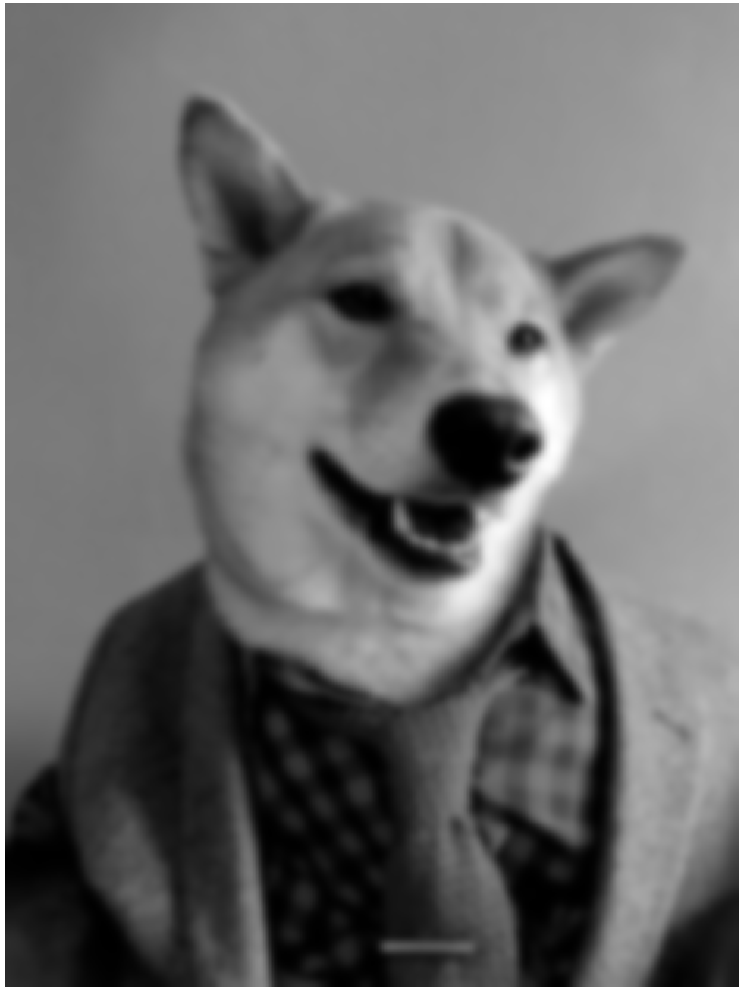
Image Convolution Examples

Gaussian Convolution:

$*$



=

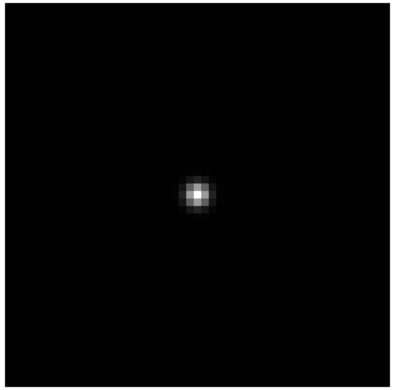


blurs image to represent average (smoothing)

Image Convolution Examples

$*$

Gaussian Convolution:



(smaller variance)

blurs image to represent average (smoothing)

$=$

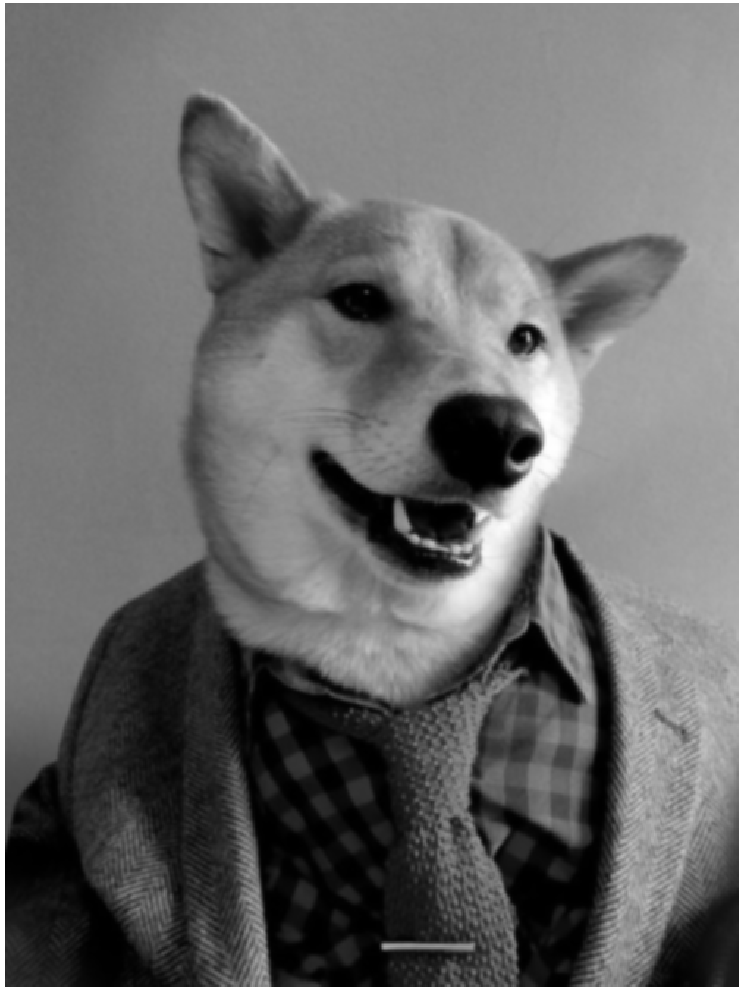
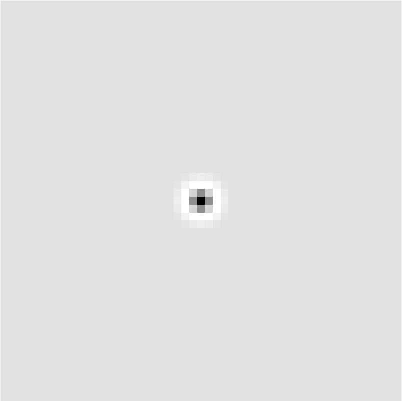


Image Convolution Examples

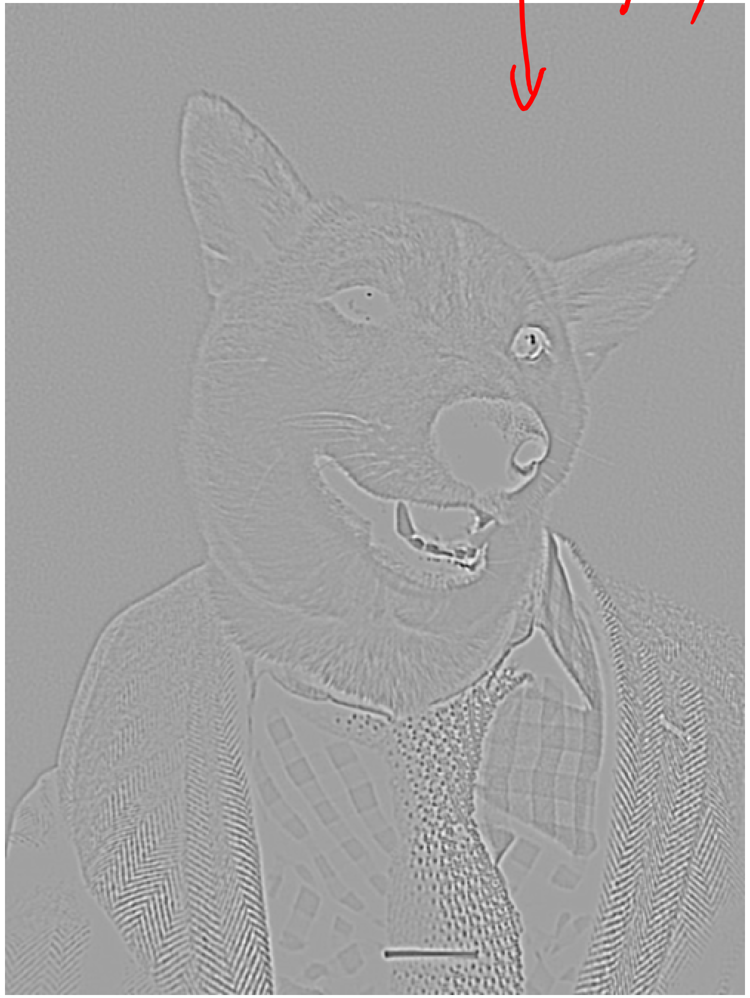
Laplacian of Gaussian

*



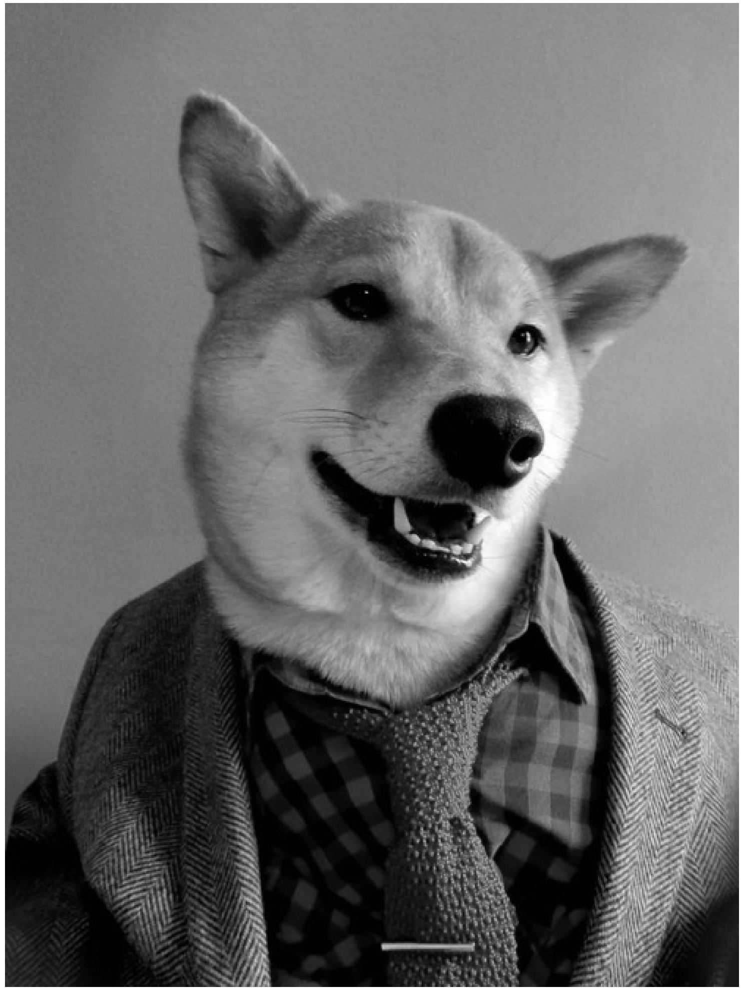
=

"signed" image (gray is 0)



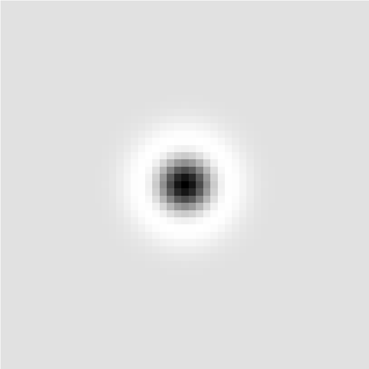
"How much does it look like a black dot surrounded by white?"

Image Convolution Examples



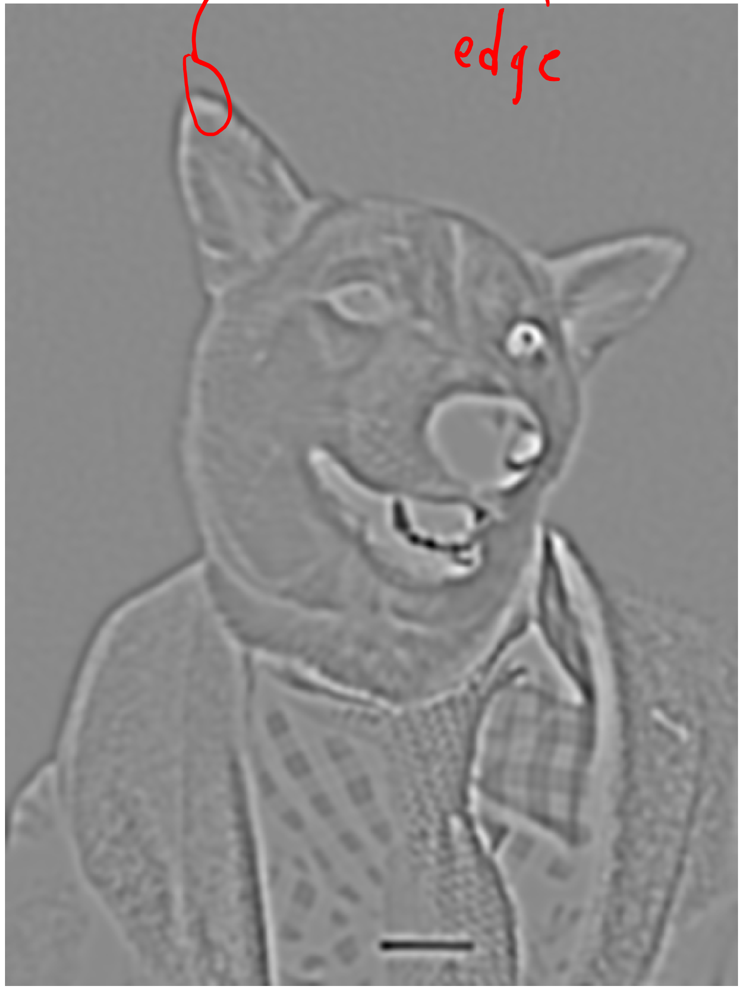
Laplacian of Gaussian

*



=

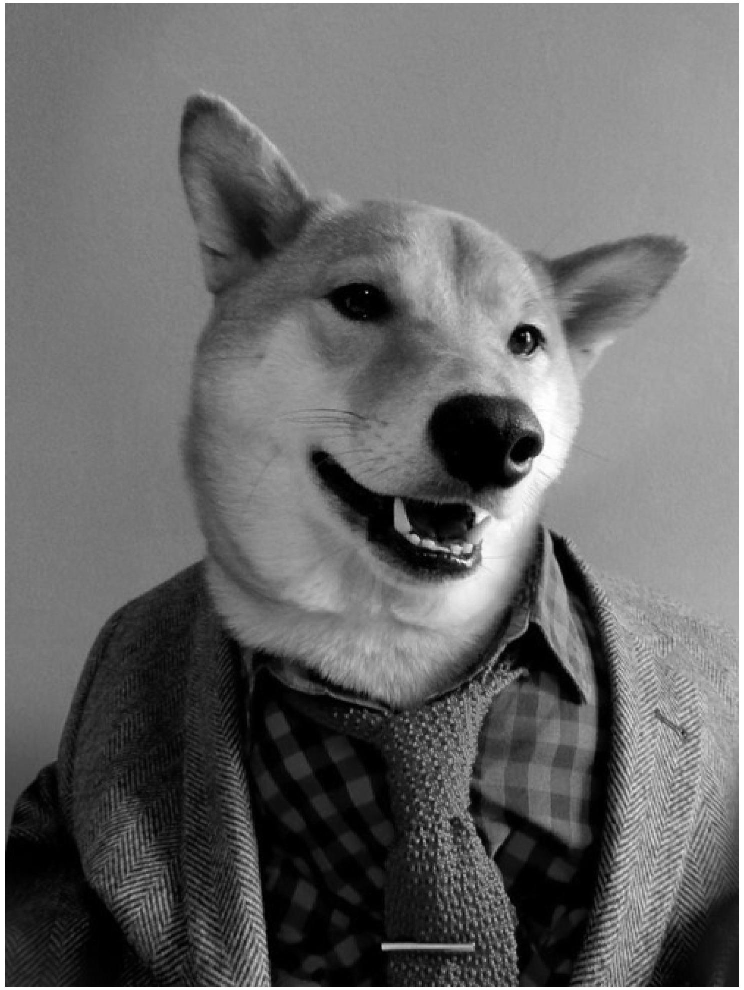
(larger variance)



Black/white as sides of edge

Similar preprocessing may be done in basal ganglia and LGN.

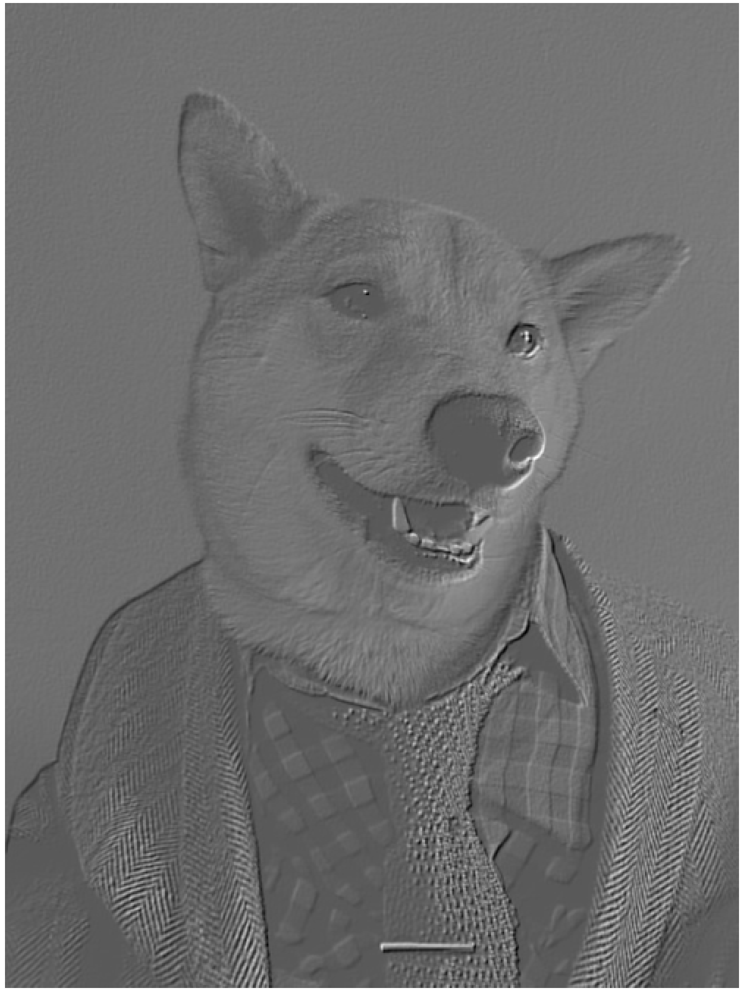
Image Convolution Examples



"Emboss" filter:

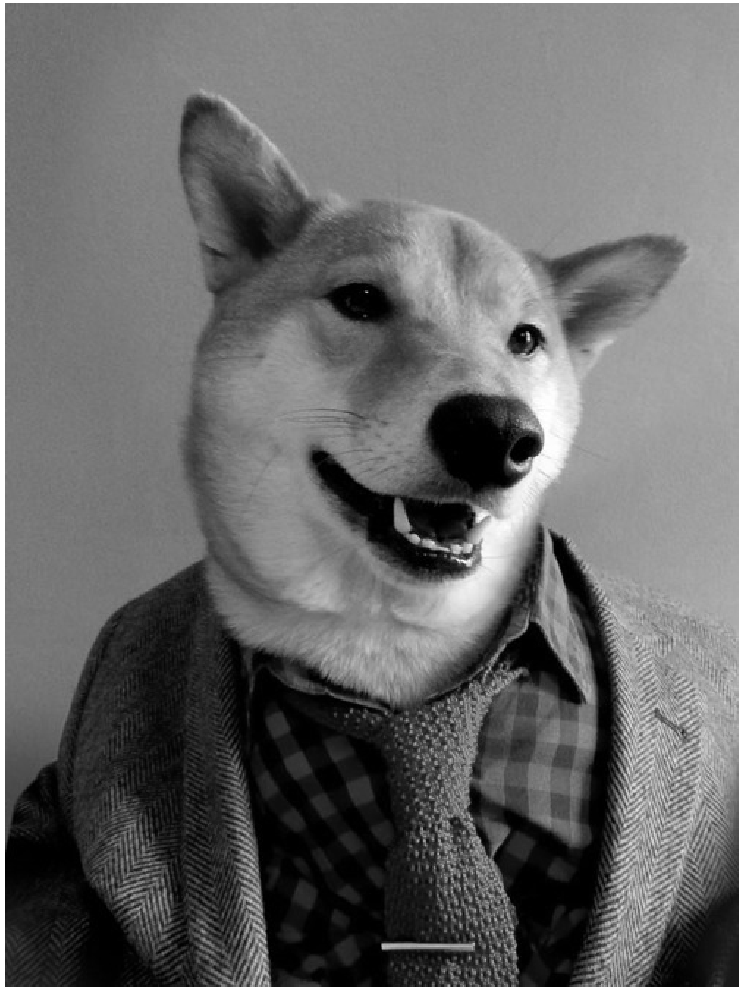
$$* \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} =$$

Many Photoshop effects are just convolutions.



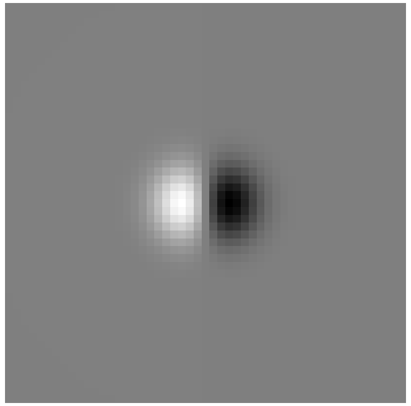
http://setosa.io/ev/image-kernels

Image Convolution Examples



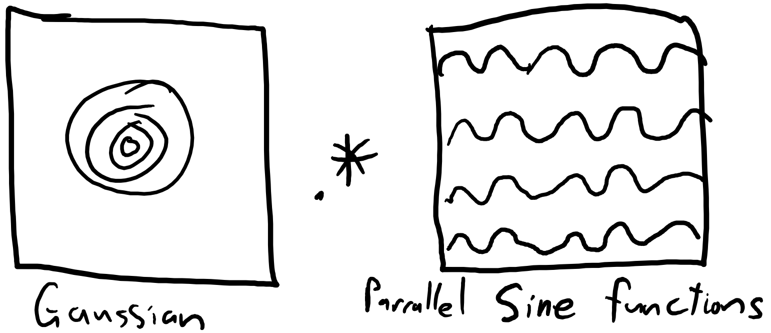
Gabor Filter
(Gaussian multiplied by sine or cosine)

*



=

||



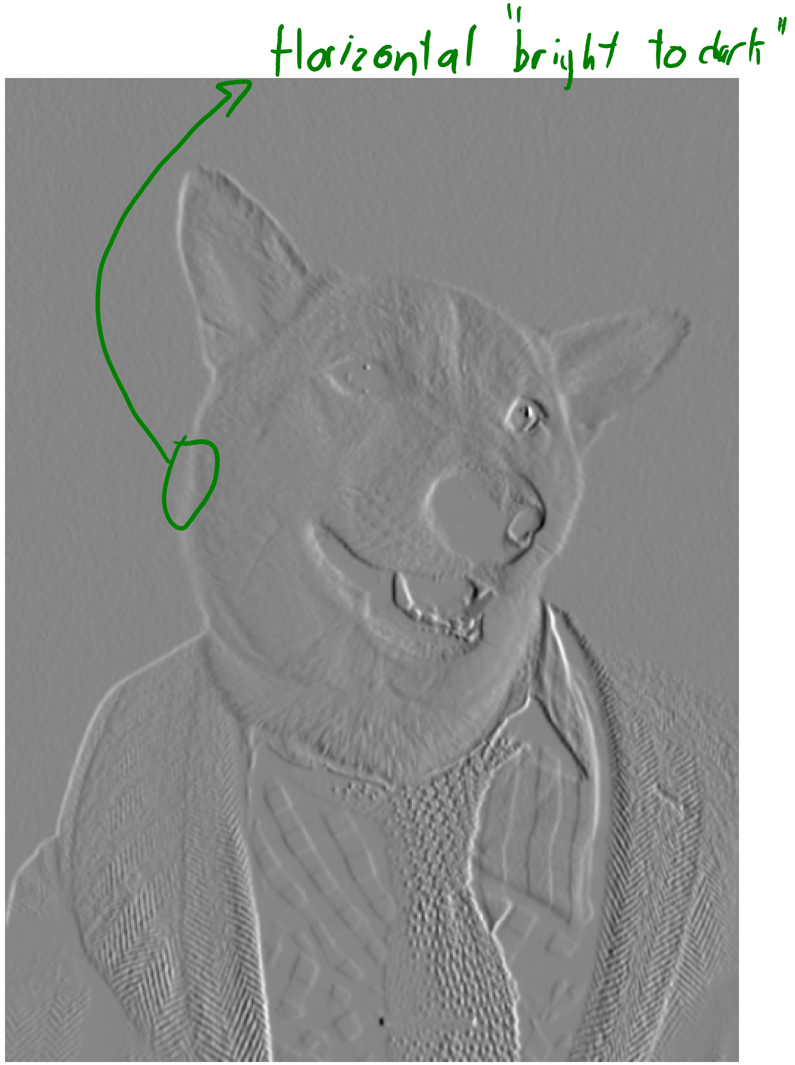
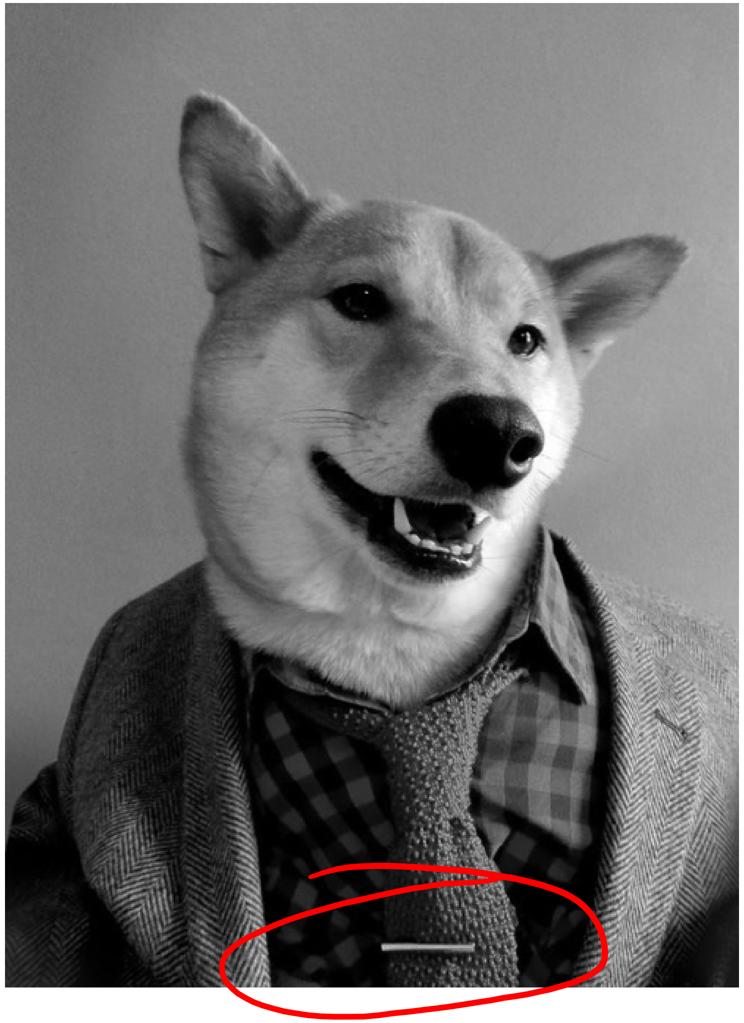
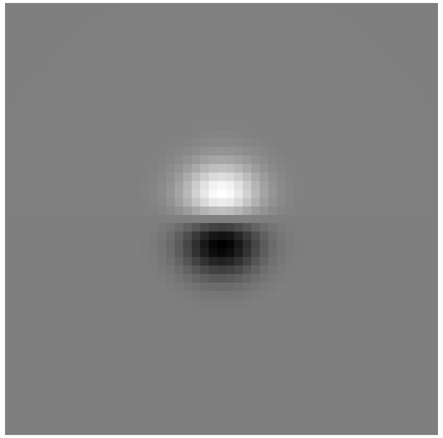


Image Convolution Examples

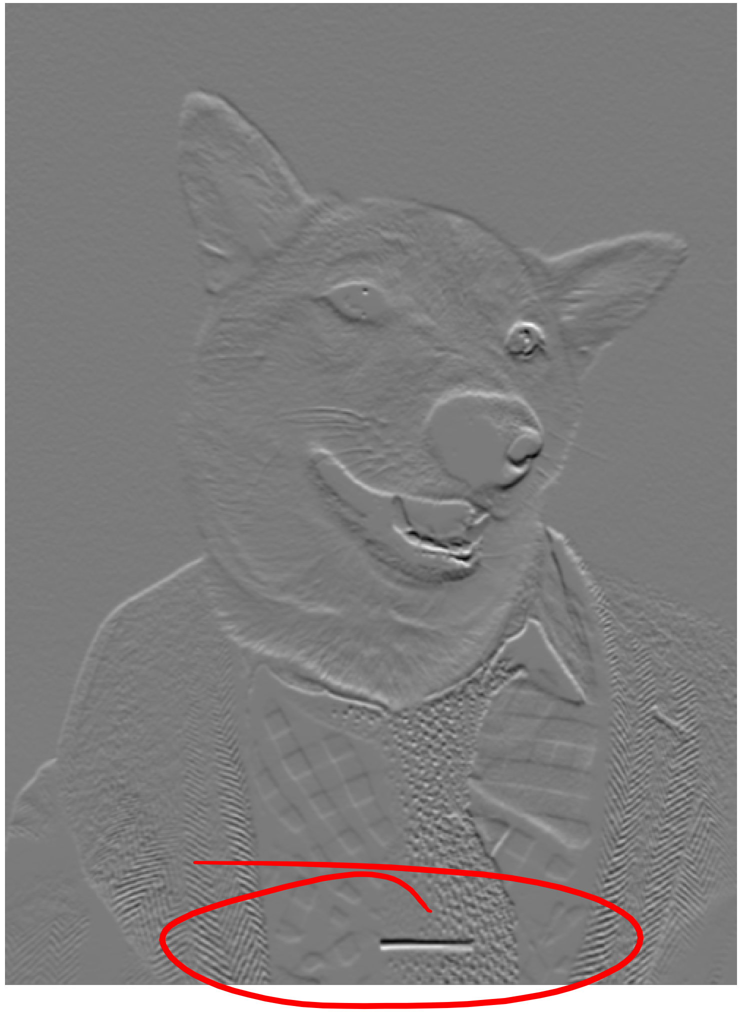


Gabor Filter
(Gaussian multiplied by sine or cosine)

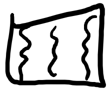
*



=

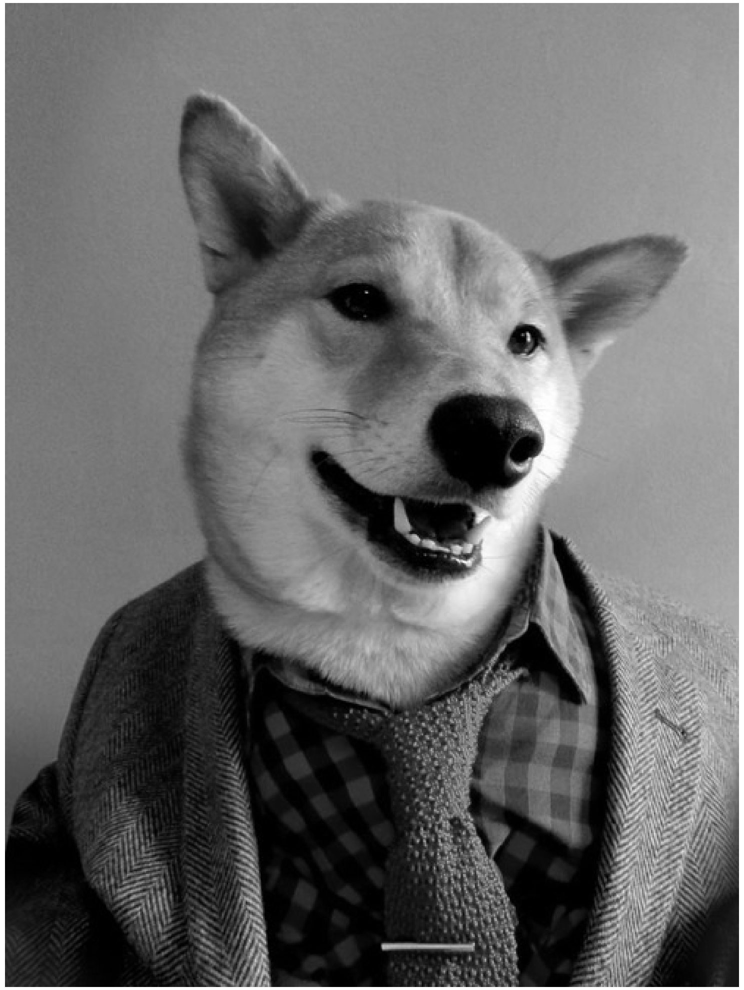


Different orientations of the sine/cosine let us detect changes with different orientations.



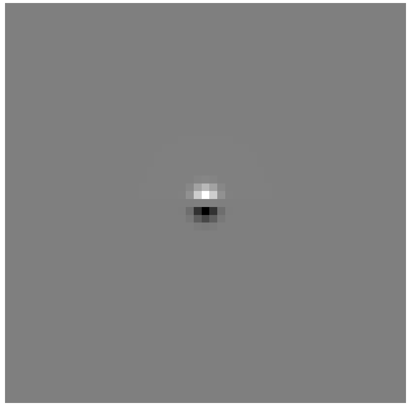
2d derivatives have a direction.

Image Convolution Examples



Gabor Filter
(Gaussian multiplied by sine or cosine)

*



(smaller variance)

=

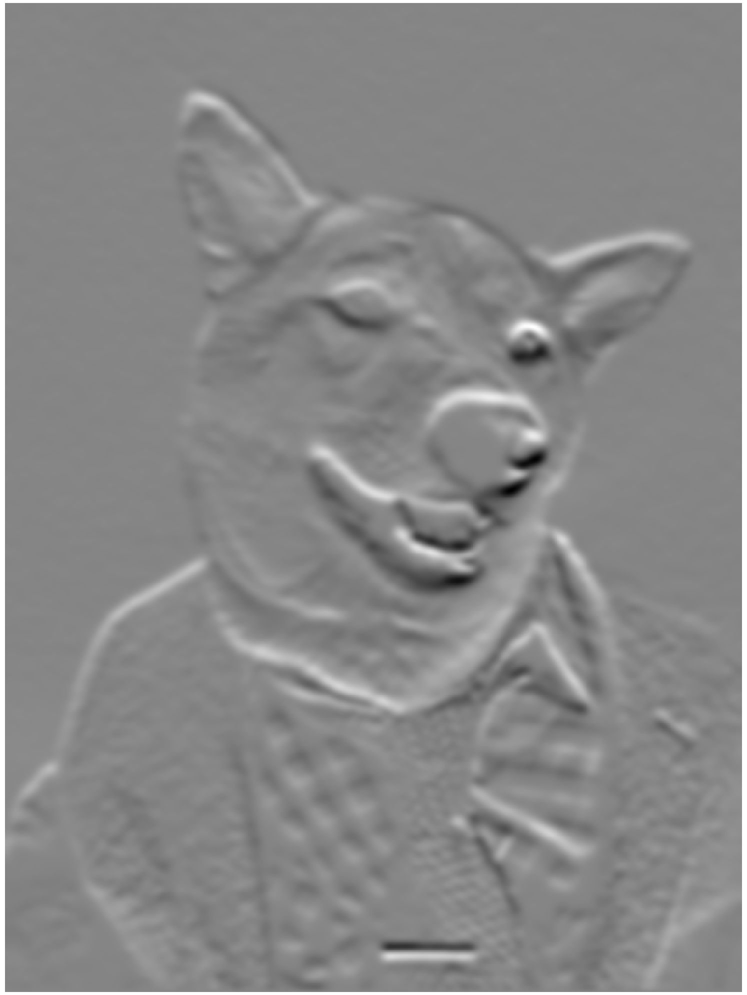
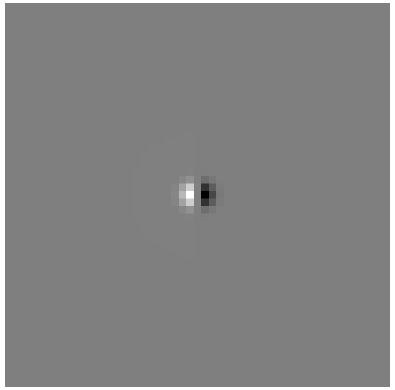


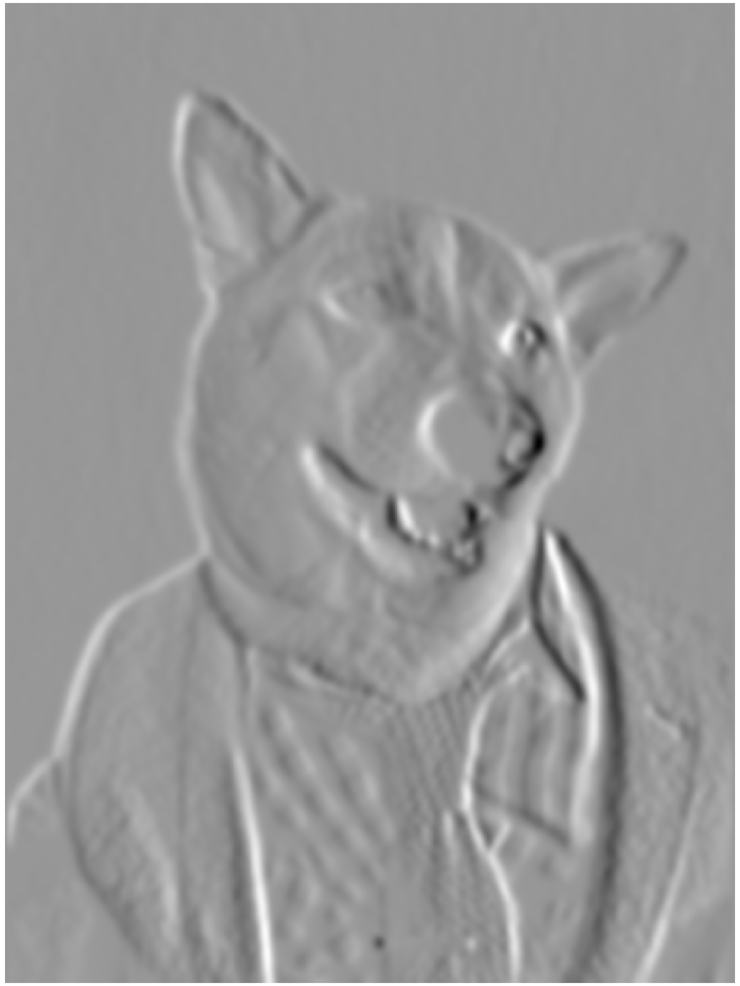
Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$



$=$



(smaller variance)

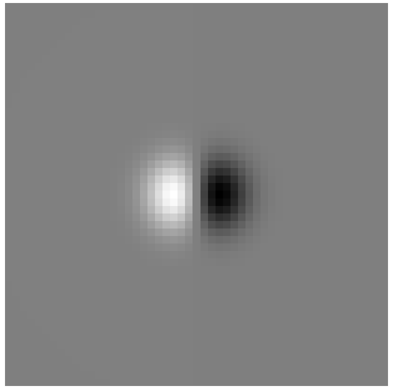
Vertical orientation

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

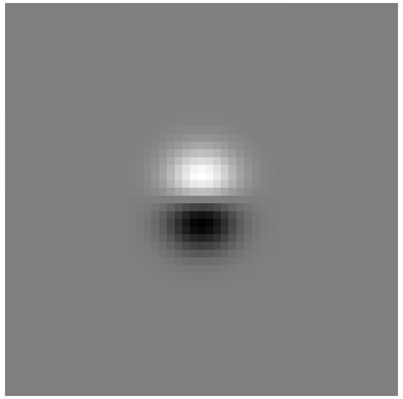
Image Convolution Examples

Max absolute value between horizontal and vertical Gabor:

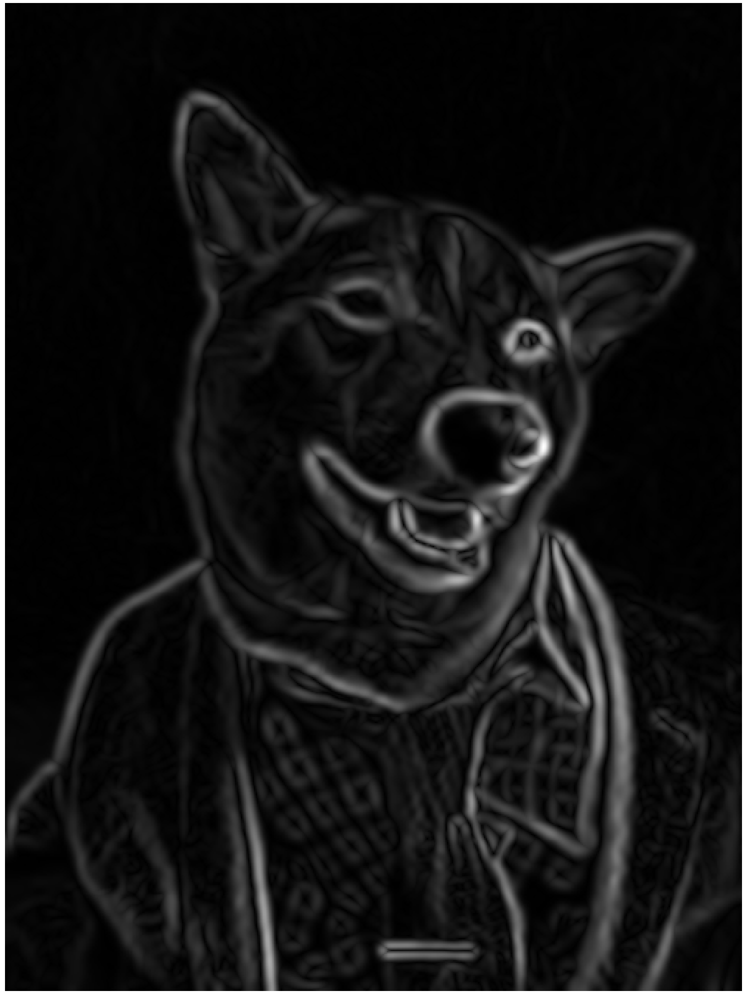
*



*



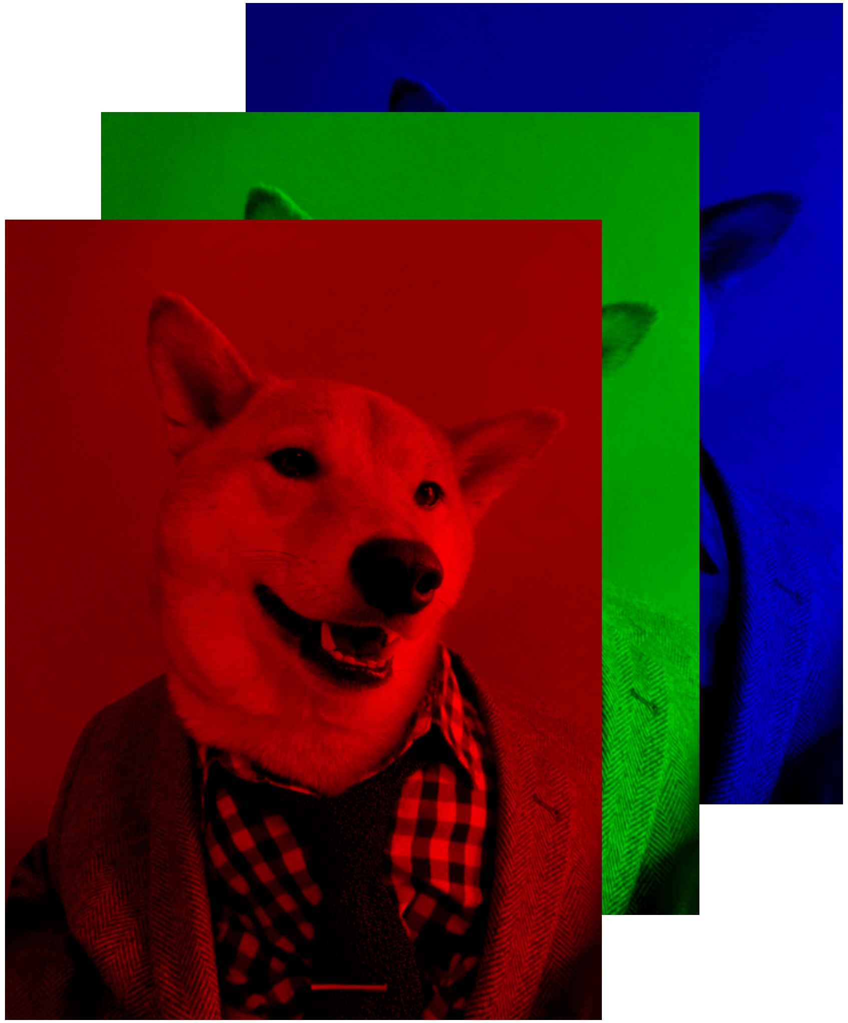
maximum absolute value



"Horizontal/vertical edge detector"

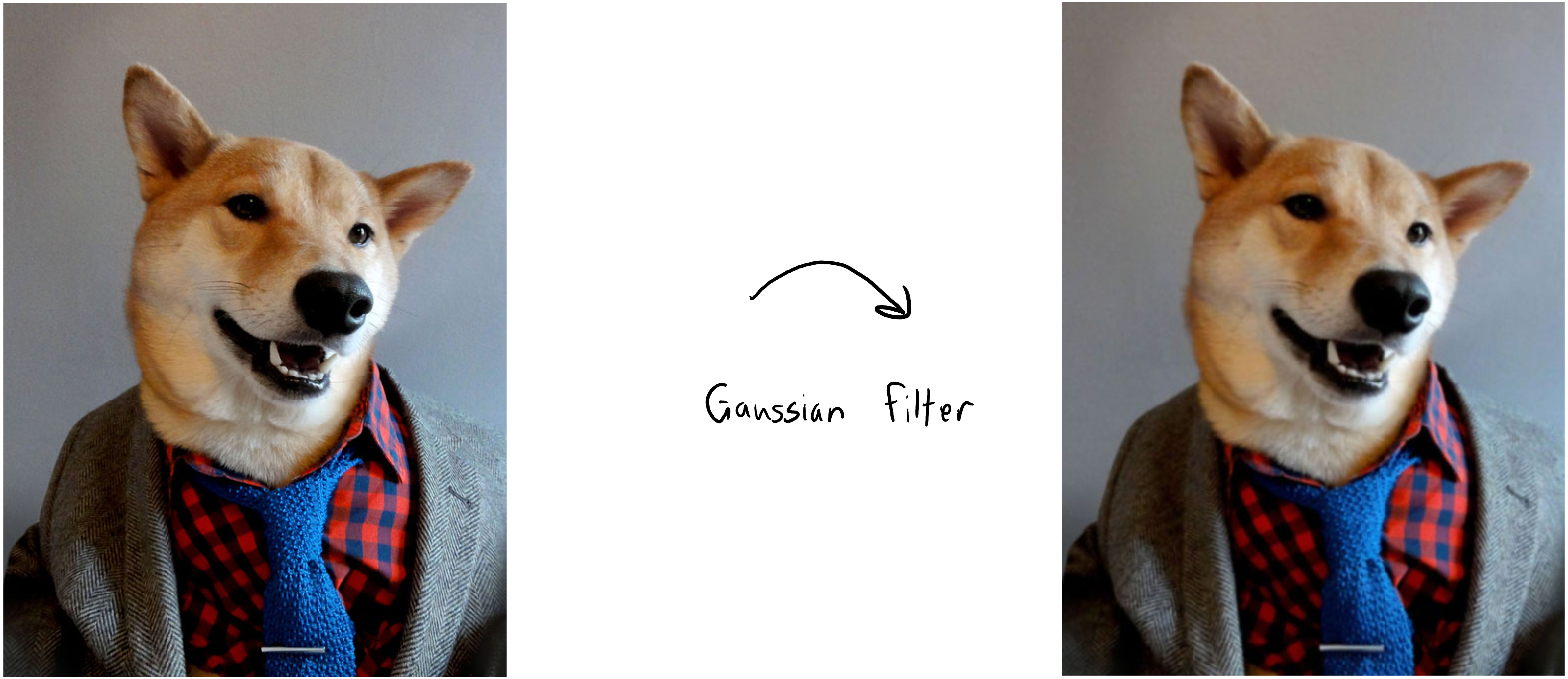
3D Convolution

Represent as RGB



Can apply 3D convolutions

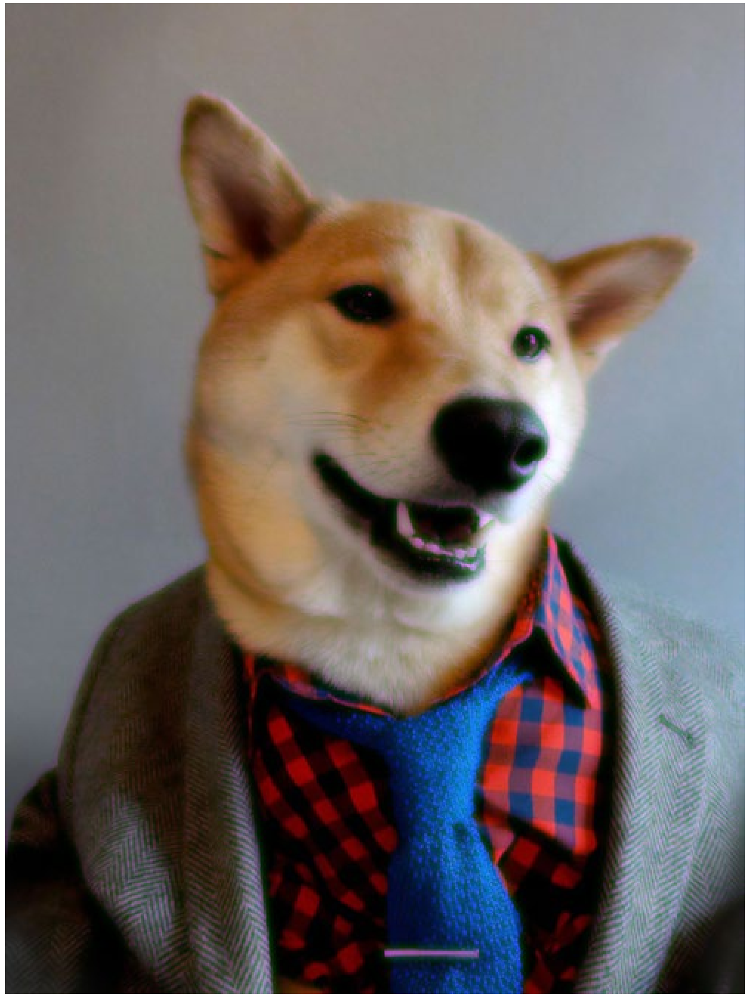
3D Convolution



3D Convolution

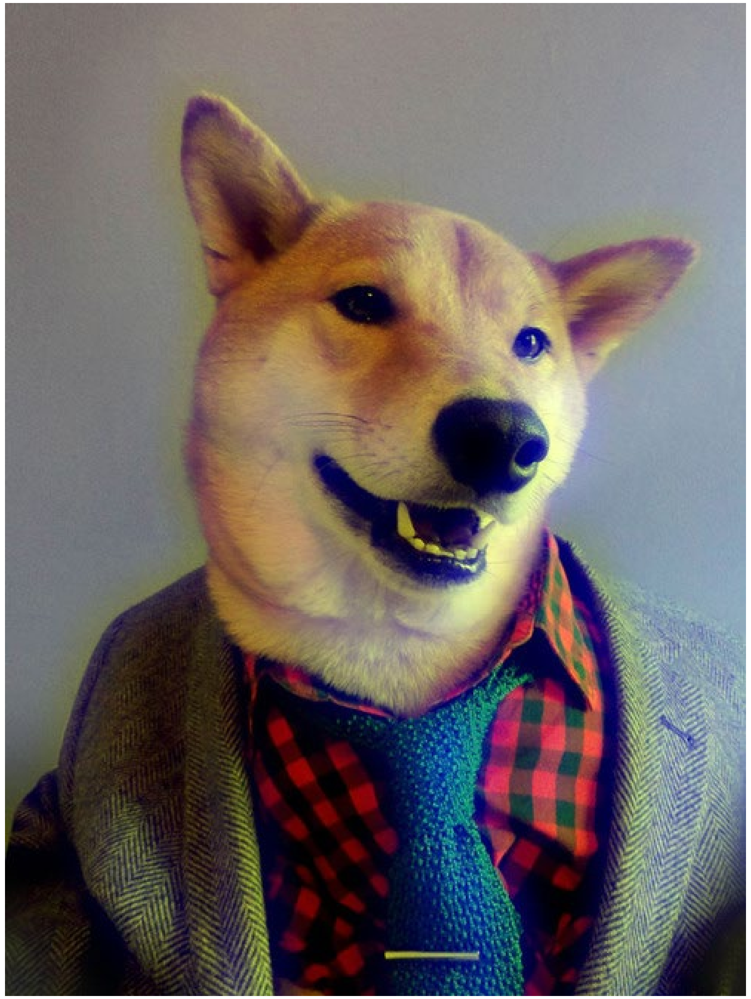
Gaussian filter

(higher variance on green channel)



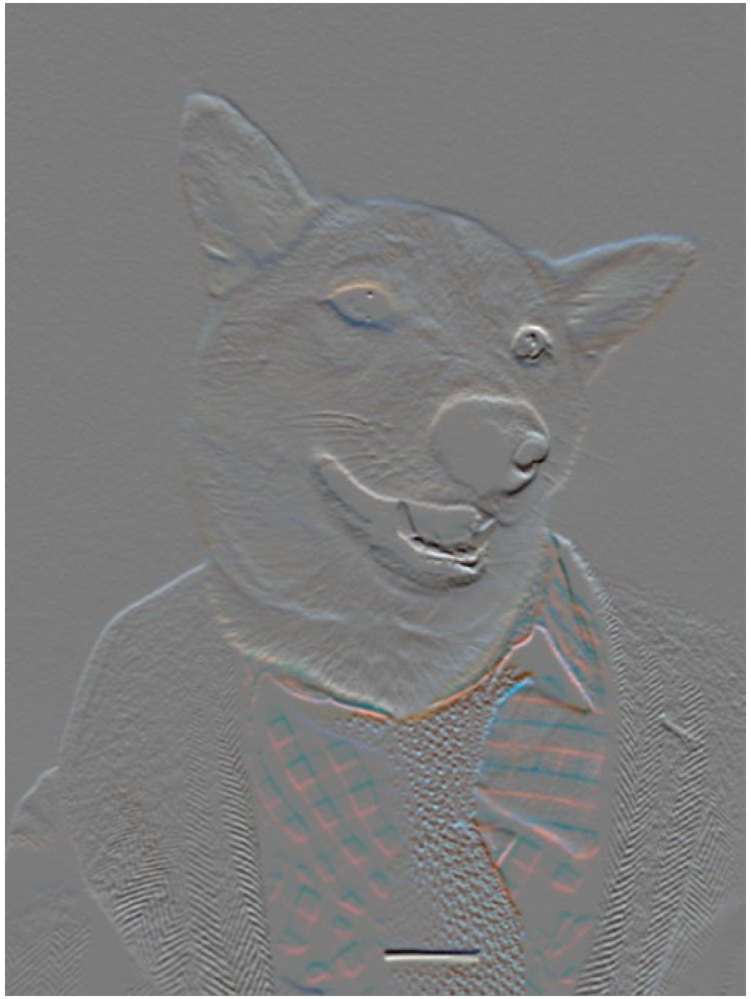
3D Convolution

Sharpen the blue channel.



3D Convolution

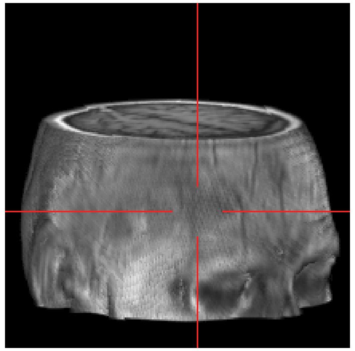
Gabor filter on each channel.

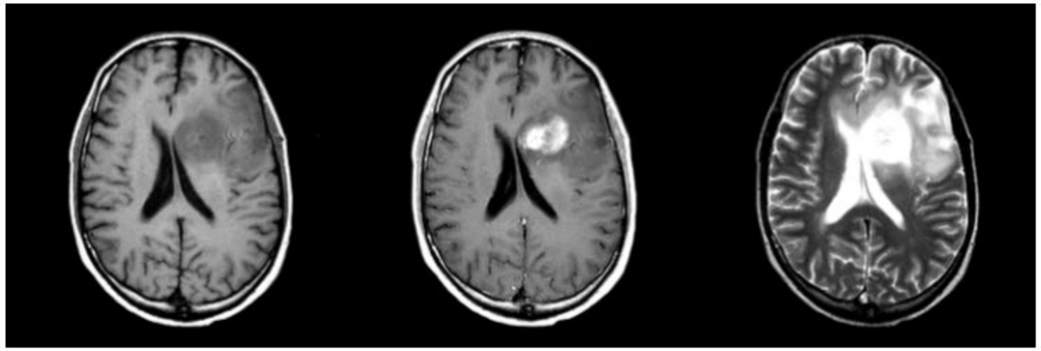


Motivation: Automatic Brain Tumor Segmentation

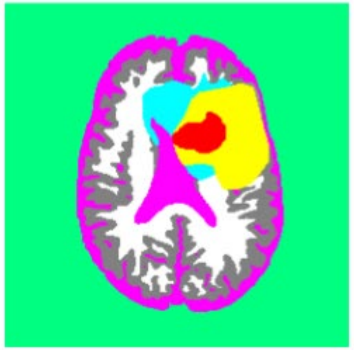
- Task: labeling tumors and normal tissue in multi-modal MRI data.

Input:





Output:

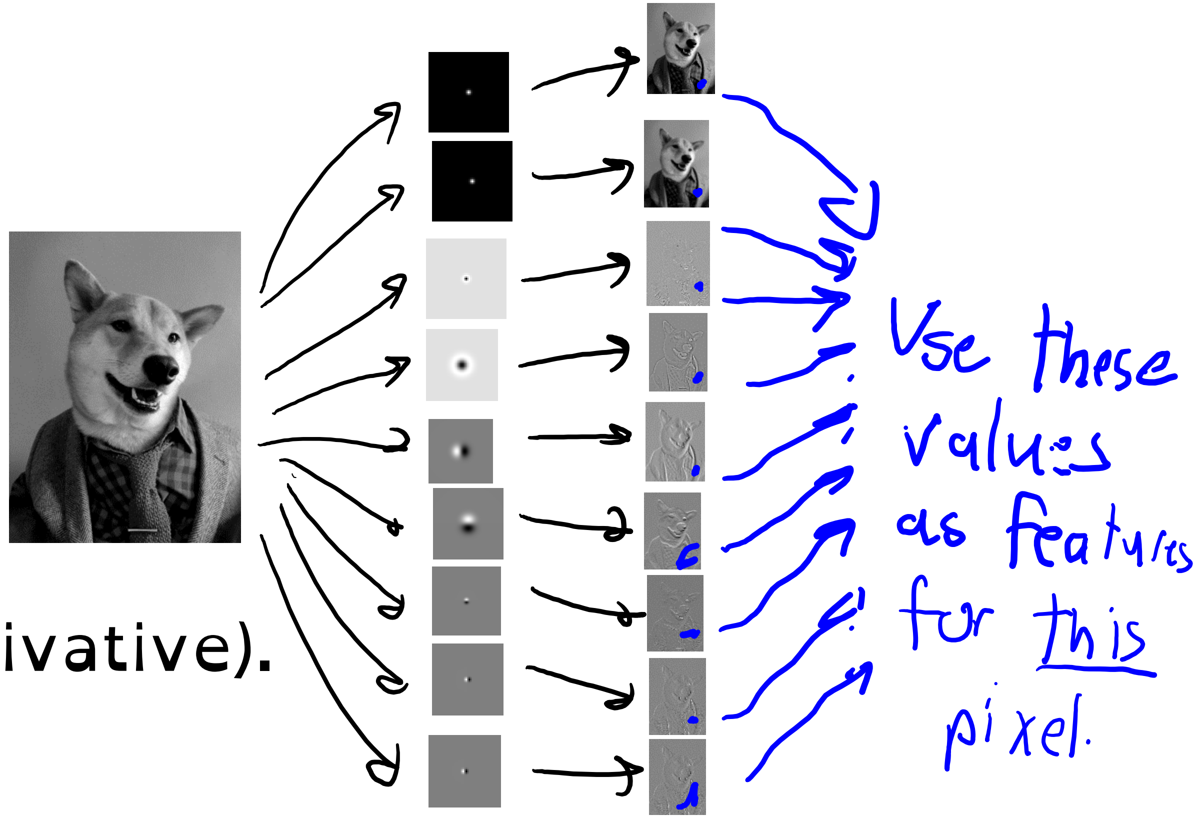


- Applications:
  - Radiation therapy target planning, quantifying treatment responses.
  - Mining growth patterns, image-guided surgery.
- Challenges:
  - Variety of tumor appearances, similarity to normal tissue.
  - “You are never going to solve this problem.”

Convolutions as Features

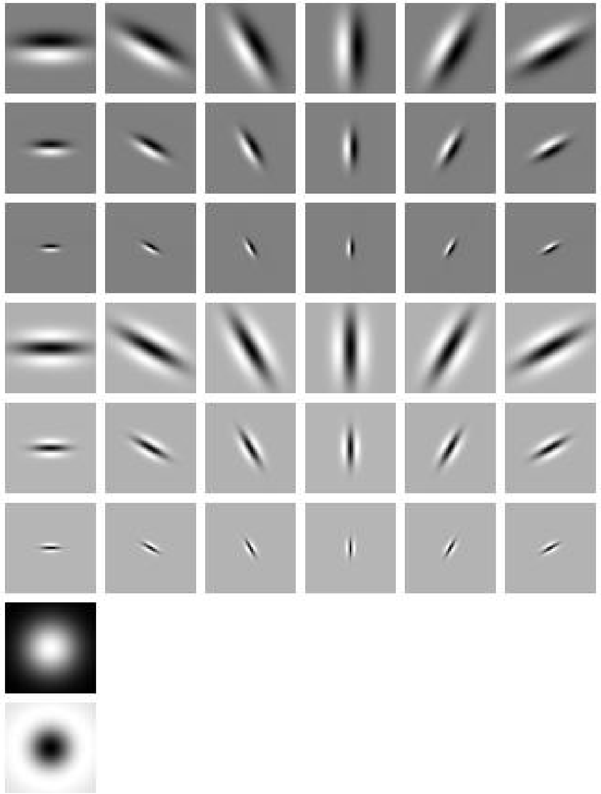
- Classic vision methods use convolutions as features:
  - Usually have different types/variances/orientations.
  - Can take maxes across locations/orientations/scales.

- 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 (Part 5) are replacing filter banks.

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

Summary

- Bag of words: not a good representation in general.
  - But good features if word order isn't needed to solve problem.
- Text features (beyond bag of words): trigrams, lexical, stem, shape.
  - Try to capture important invariances in text data.
- Global vs. local features allow "personalized" predictions.
- Convolutions are flexible class of signal/image transformations.

- Next time: Kernels.

Review Questions

- Q1: What does it mean that bag-of-words can encourage invariance?

- Q2: If we use linear models, how does the number of local features affect the two parts of the fundamental trade-off?

- Q3: How does convolution help represent the "neighbourhood" of a pixel?

- Q4: How can "smoothing" filters help with noise in image data?

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.

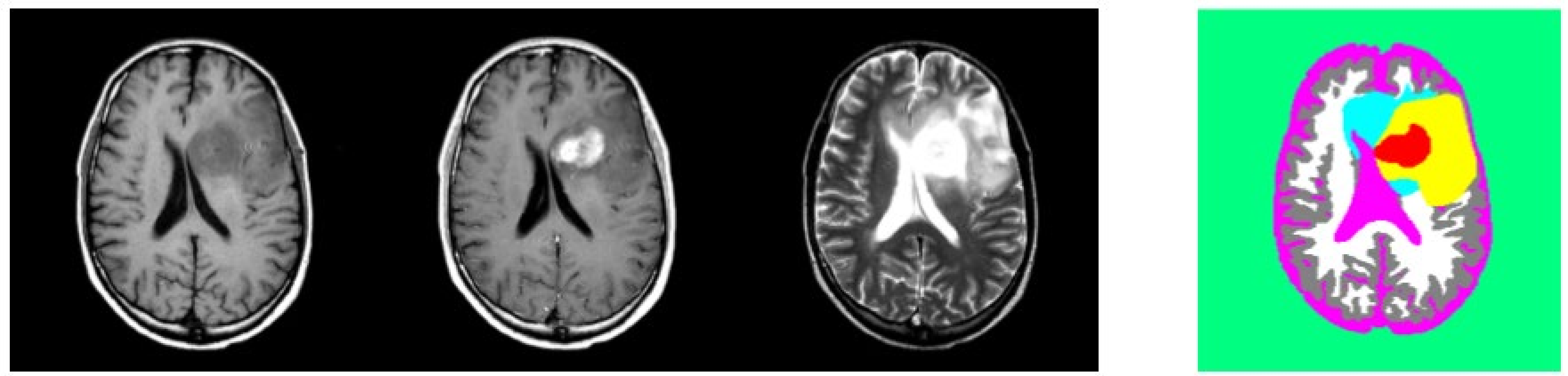
Global and Local Features for Domain Adaptation

- "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

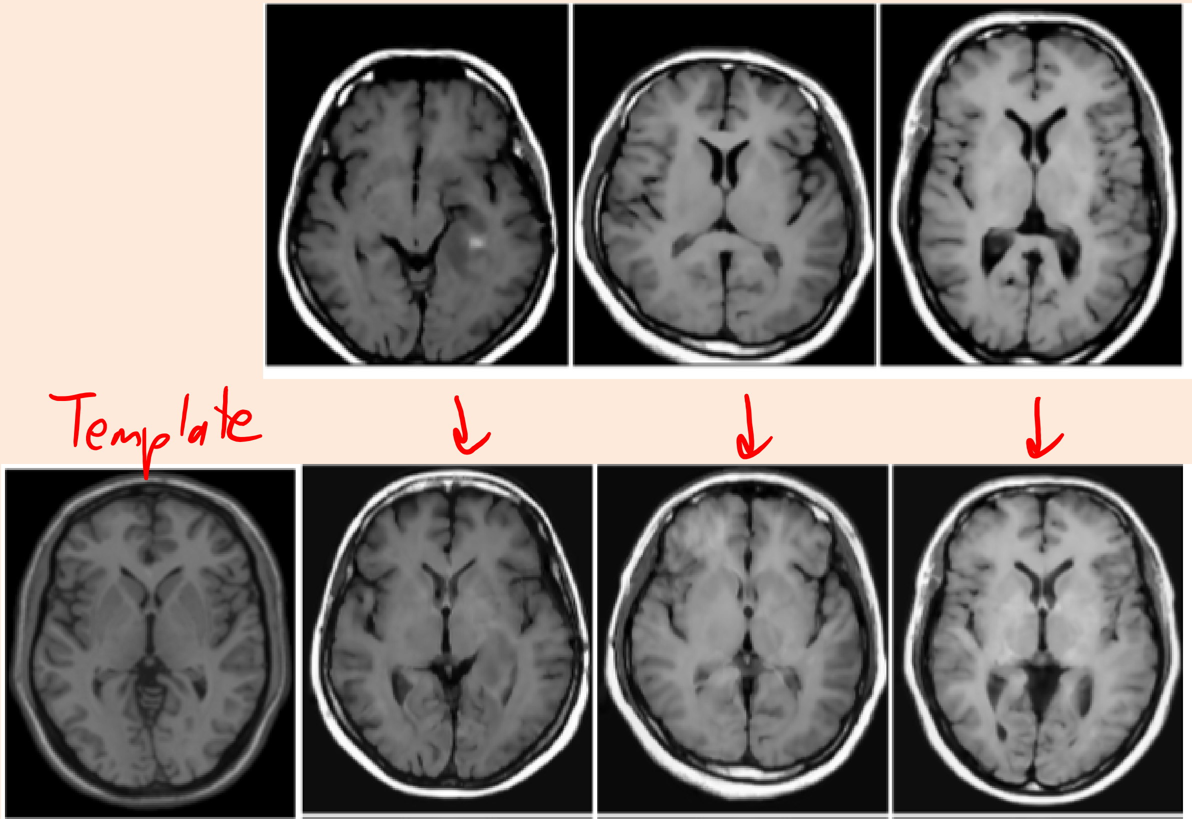
- Should we use the image coordinates?
  - E.g., the pixel is at location (124, 78) in the image.



- 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

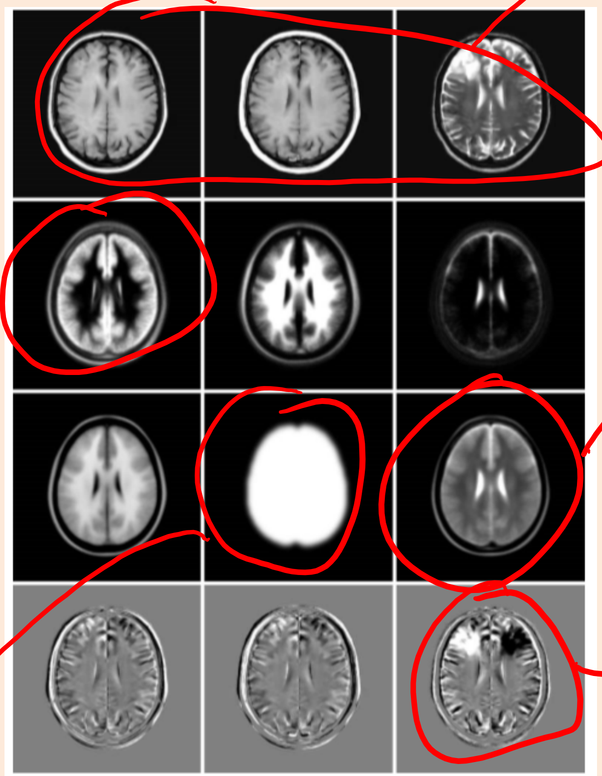
- The position in the image is important in brain tumour application.
  - But we didn't have much data, so coordinates didn't make sense.
- We aligned the images with a "template image".



(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.

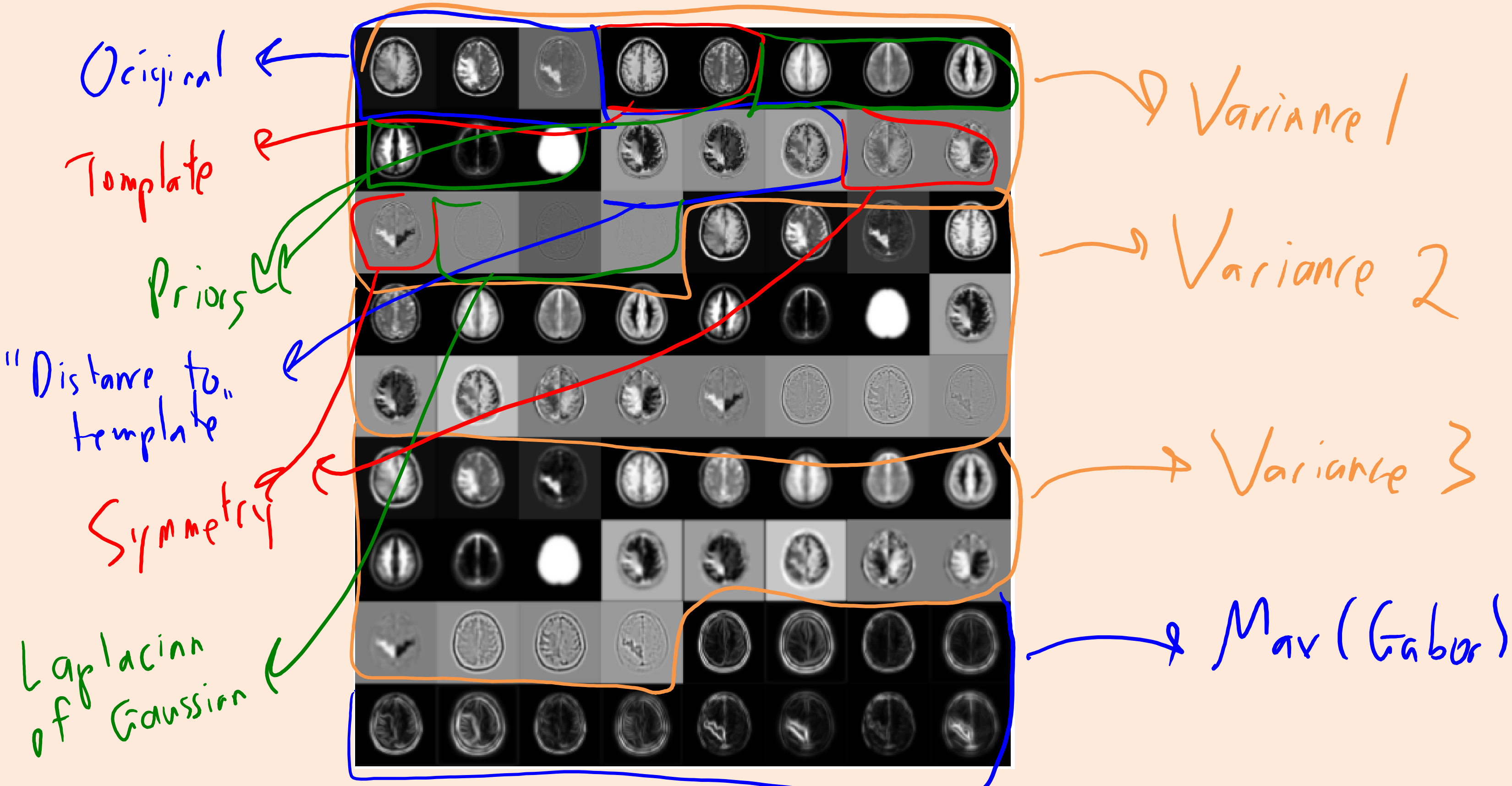
Actual pixel value of template image at this location.

Left-right symmetry difference.

Probability of being brain pixel.

Motivation: Automatic Brain Tumor Segmentation

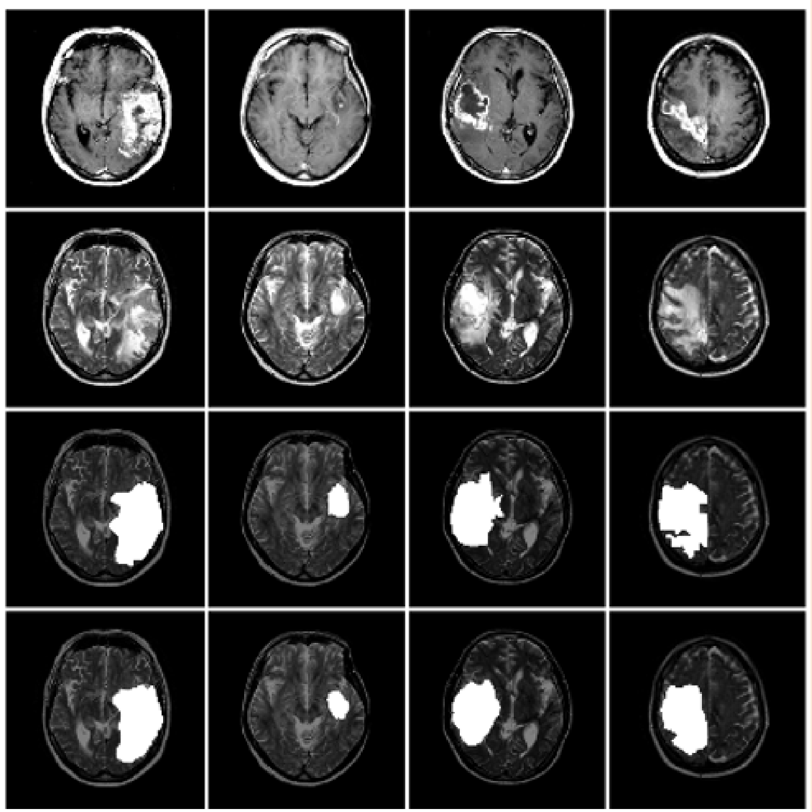
- Final features for brain tumour segmentation:
  - Gaussian convolution of original/template/priors/symmetry, Laplacian of Gaussian on original.
    - All with 3 variances.
    - Max(Gabor) with sine and cosine on orginal (3 variances).



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.



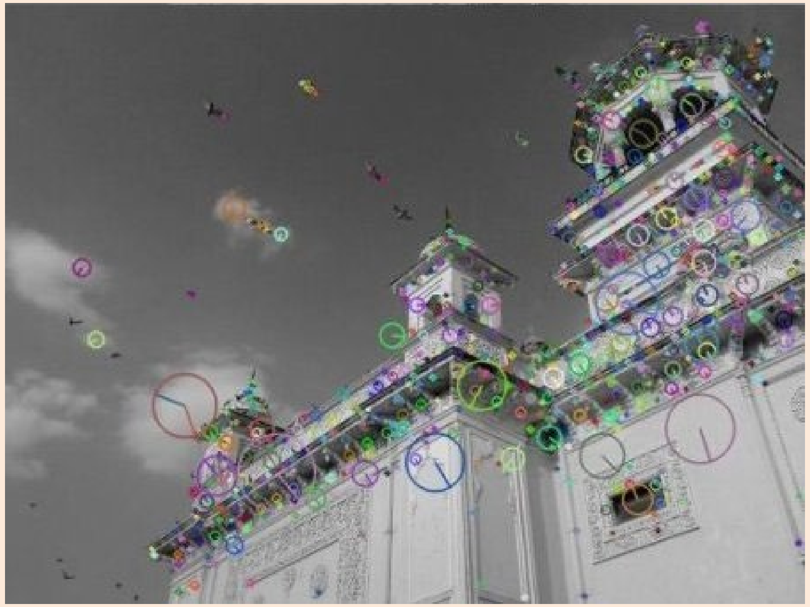
FFT implementation of convolution

- 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