CPSC 340: Machine Learning and Data Mining

Convolutional Neural Networks

Original version of these slides by Mark Schmidt, with modifications by Mike Gelbart.

Admin

- Assignment 6:
  - Due Friday.

- Final exam:
  - Saturday April 14, 3:30pm, SUB 2201.

Recap

- Last couple lectures: neural networks & deep learning
  - Simultaneously learn the basis and the linear/logistic regression weights
  - Alternate between matrix multiplication and element-wise nonlinearity
  - Very non convex, a huge bag of tricks out there to make them works
- Last lecture: convolutions
  - A way of thinking about a linear function operating on a vector
  - Can represent translation, averaging, approximate derivatives, and more

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.

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

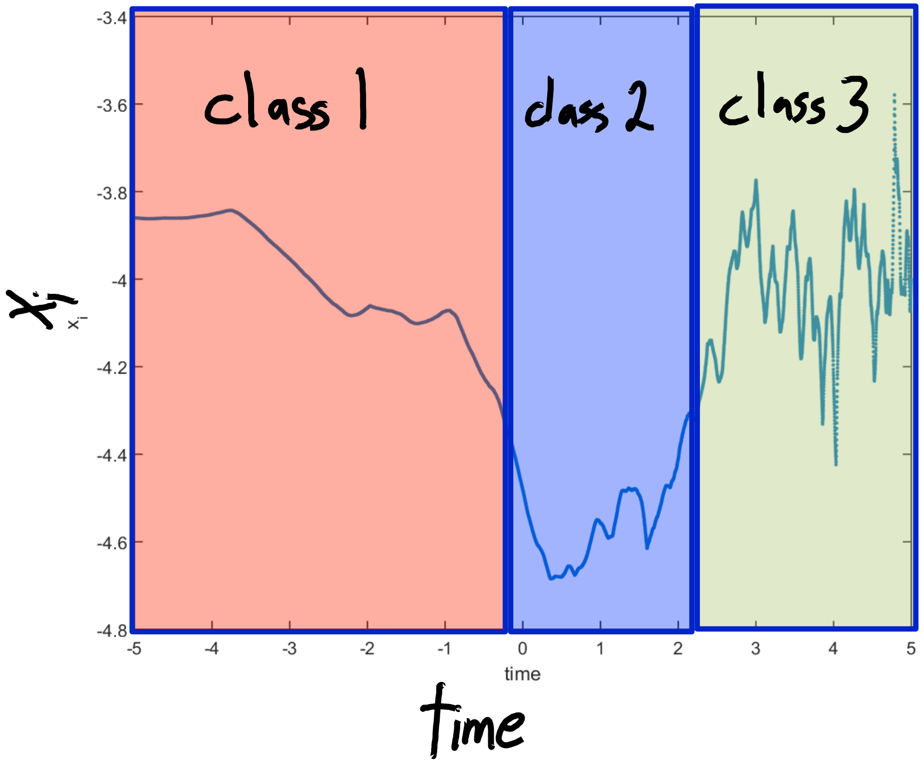
Jupyter notebook demo

Today: Convolutional Neural Networks

- We will solve some problems:
  - Flattening an image into a vector discards valuable spatial information
  - Using a fully connected networks leads to HUGE numbers of parameters
- By making some assumptions:
  - Low-level local features can help us understand images
  - We don’t need every pixel feeding into a unit at the next layer
  - We can represent these transformations with convolutions

Representing Neighbourhoods with Convolutions

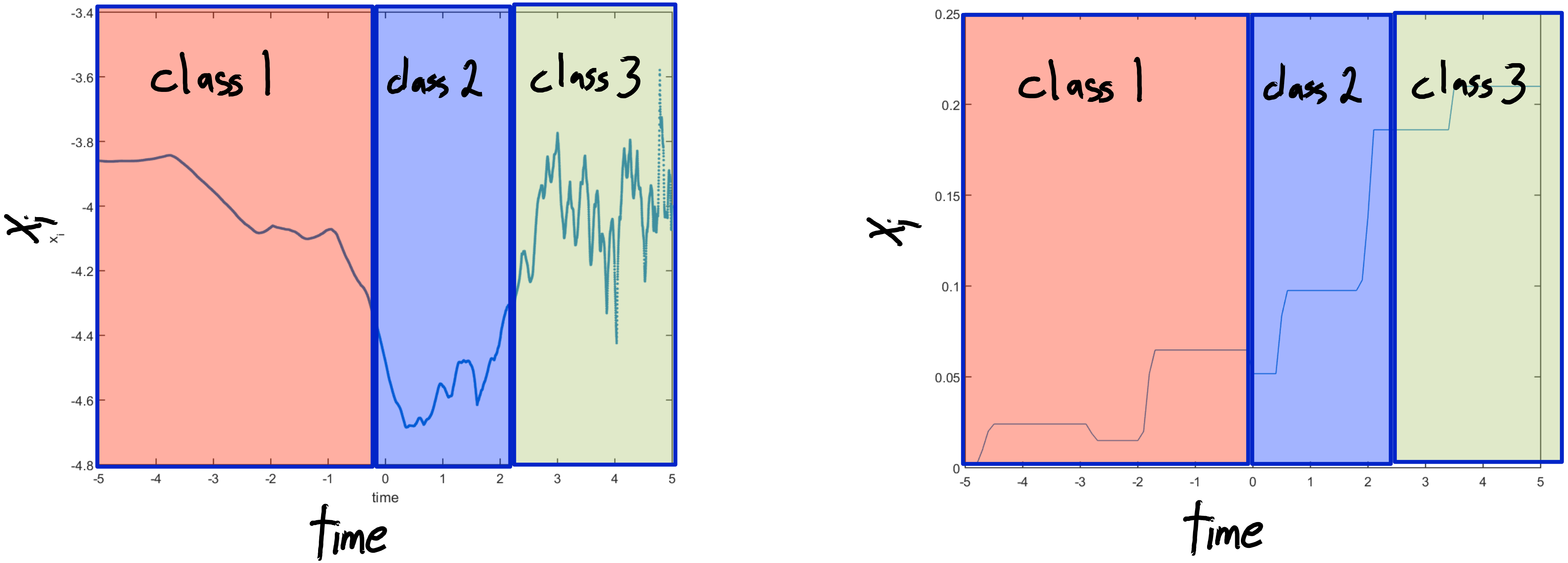
- Consider a 1D dataset:
  - Want to classify each time into y_i in {1,2,3}.
  - Example: speech data.



- 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

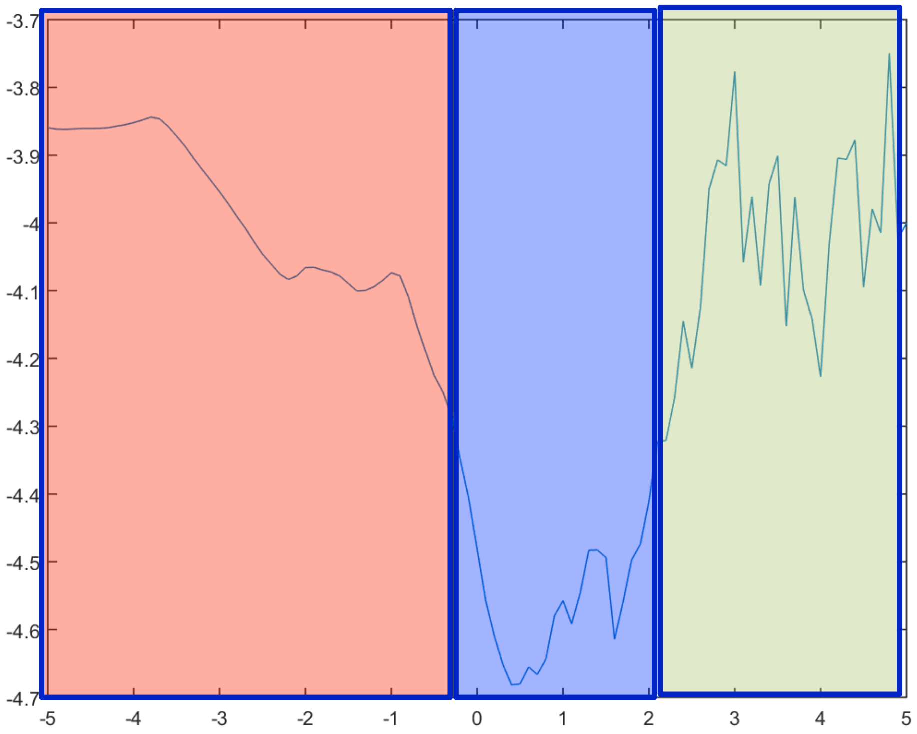
- Original features (left) and features from convolutions (right):

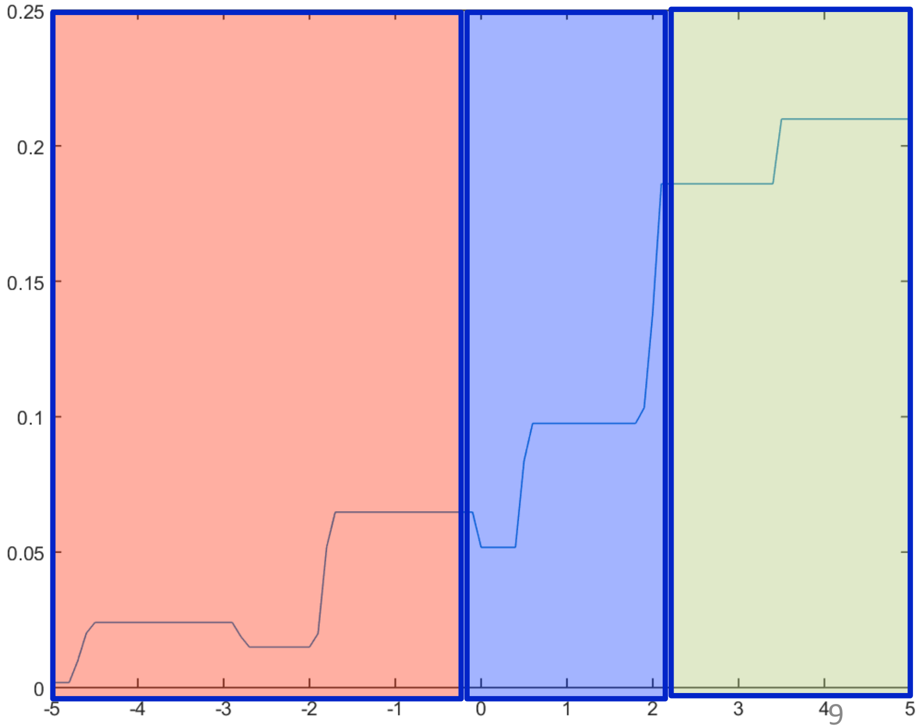


- Easy to distinguish the 3 classes with these 2 features.

1D Convolution Examples

- We often use maximum over several convolutions as features:
  - We could take maximum of Laplacian of Gaussian over x_i and neighbours.
  - We use different convolutions as our features (derivatives, integrals, etc.).





1D Convolution as Matrix Multiplication

- Each element of a convolution is an inner product:

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

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

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

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

$$z = \tilde{W}x \quad \text{where } \tilde{W} = \begin{bmatrix} \text{———} & w & \text{———} & 0 & 0 & 0 \\ 0 & \text{———} & w & \text{———} & 0 & 0 \\ 0 & 0 & \text{———} & w & \text{———} & 0 \\ 0 & 0 & 0 & \text{———} & w & \text{———} \end{bmatrix} \left.\vphantom{\begin{bmatrix} a \\ a \\ a \\ a \end{bmatrix}}\right\} \text{matrix can be very sparse and only has } 2m+1 \text{ variables}$$

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

Last Lectures: Deep Learning

Deep computer vision models are all convolutional neural networks:

- The $W^{(m)}$ are very sparse and have repeated parameters ("tied weights").
- Drastically reduces number of parameters (speeds training, reduces overfitting).

Motivation for Convolutional Neural Networks

- Consider training neural networks on 256 by 256 images.
  - This is 256 by 256 by 3 ≈ 200,000 inputs.
- If first layer has k=10,000, then it has about 2 billion parameters.
  - We want to avoid this huge number (due to storage/speed and overfitting).

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

$$w_1 = [0 \quad 0 \quad 0 \text{ --- } w \text{ --- } 0 \quad 0 \quad 0]$$

$$w_2 = [0 \text{ --- } w \text{ --- } 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

- Forces most weights to be zero, and others to be shared:
  - Reduces number of parameters.

Convolutional Neural Networks

- Convolutional Neural Networks classically have 3 layer "types":
  - Fully connected layer: usual neural network layer with unrestricted W.

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

Convolutional Neural Networks

- Convolutional Neural Networks classically have 3 layer “types”:
  - Fully connected layer: usual neural network layer with unrestricted W.
  - Convolutional layer: restrict W to results of several convolutions.

1D example

distance between centers of convolution is called “stride”

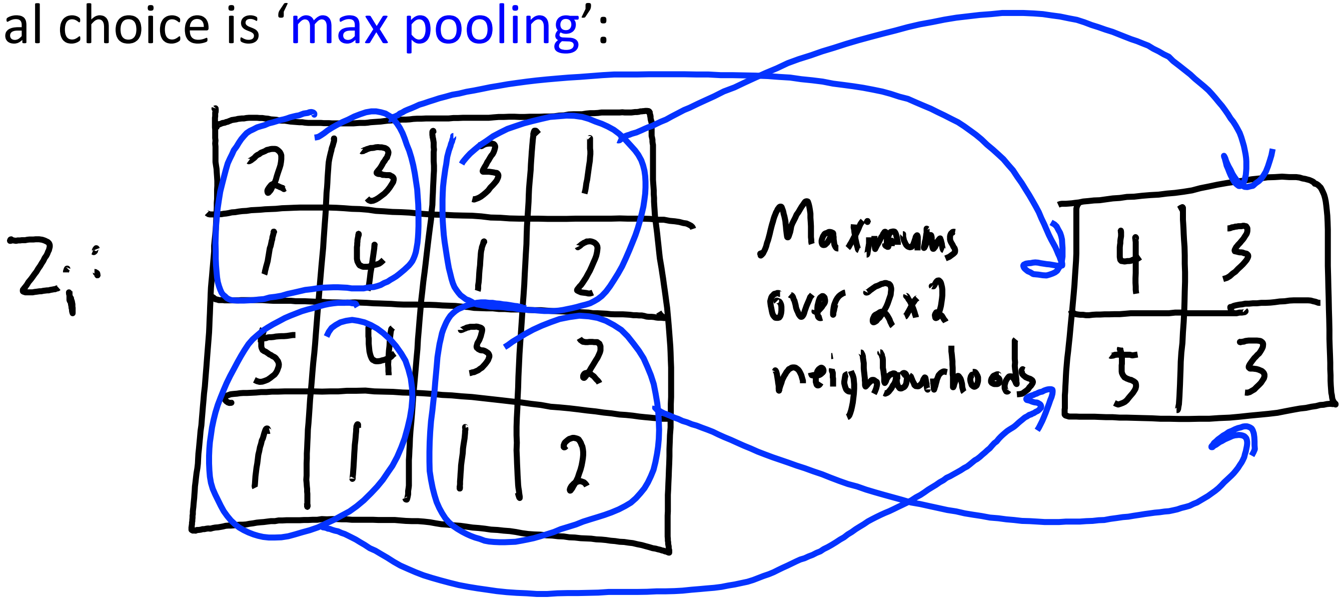
$$W^{(m)} = \begin{bmatrix} \text{——} & w_1^{(m)} & \text{——} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \text{——} & w_1^{(m)} & \text{——} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \text{——} & w_1^{(m)} & \text{——} \\ \text{——} & w_2^{(m)} & \text{——} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \text{——} & w_2^{(m)}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

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

Sparse and small number of parameters

Convolutional Neural Networks

- Convolutional Neural Networks classically have 3 layer "types":
  - Fully connected layer: usual neural network layer with unrestricted W.
  - Convolutional layer: restrict W to results of several convolutions.
  - Pooling layer: combine results of convolutions.
    - Can add invariances or just make the number of parameters smaller.
    - Usual choice is 'max pooling':



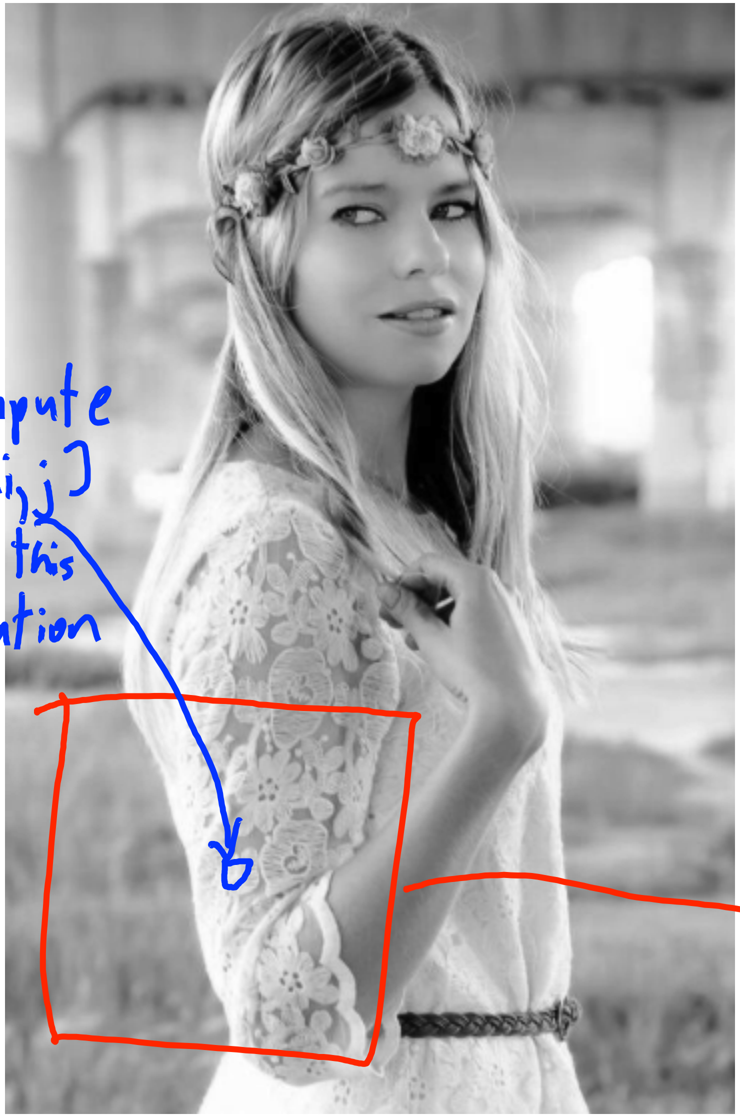
Back to Jupyter: counting parameters

Summary

- Convolutions are flexible class of signal/image transformations.
  - Can approximate derivatives and integrals at different scales.
- Max(convolutions) can yield features that make classification easy.
- Convolutional neural networks:
  - Restrict $W^{(m)}$ matrices to represent sets of convolutions.
  - Often combined with max (pooling).

Image Convolution Examples

x

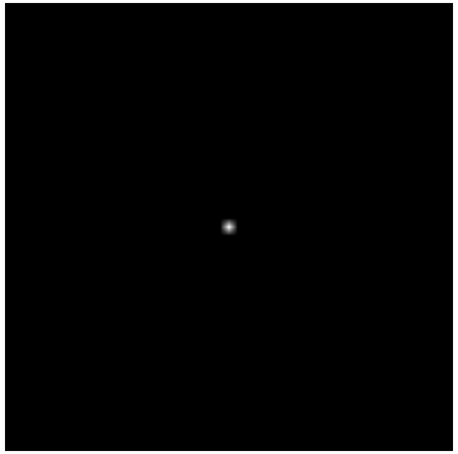


Compute z[i,j] for this location

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

w

$*$



=

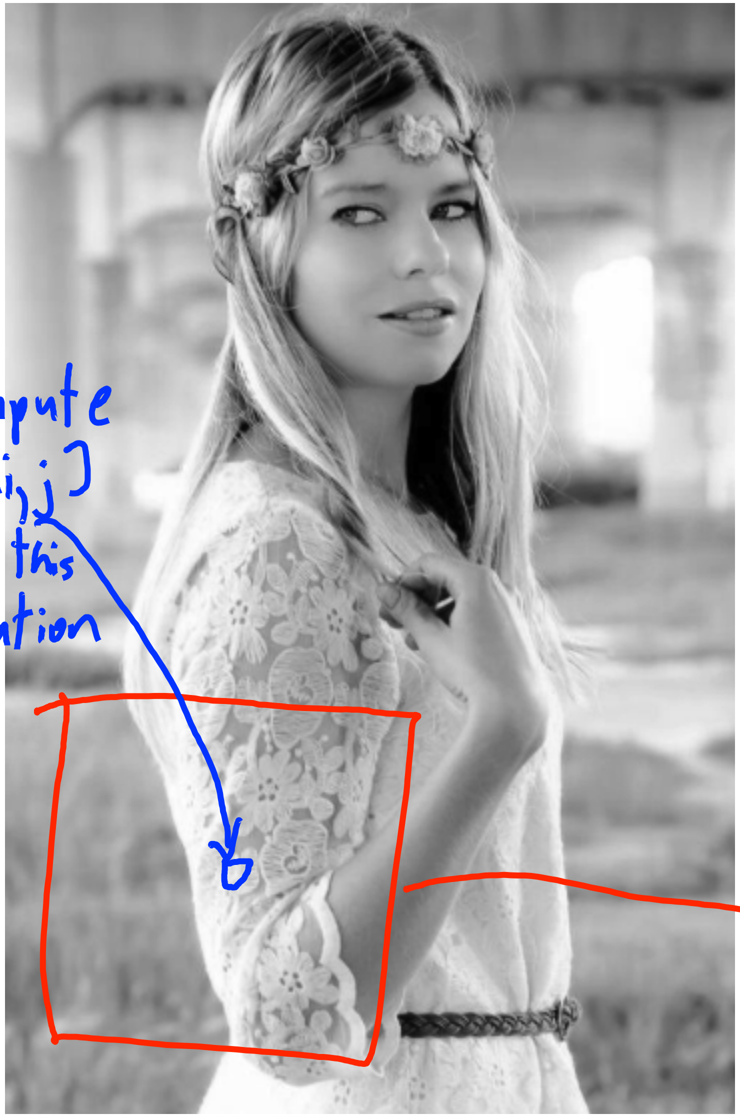
z

multiply element-wise and add up result to get

z[i,j]

Image Convolution Examples

x

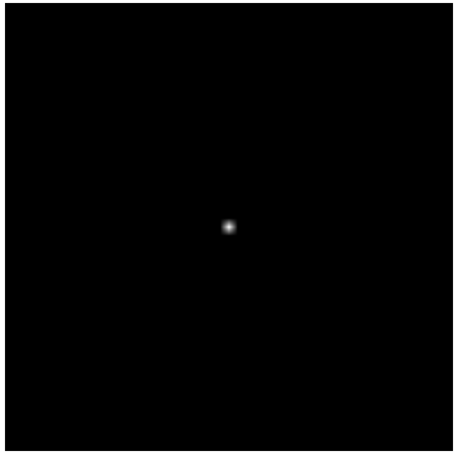


Compute $z[i,j]$ for this location

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

w

$*$



$=$

multiply element-wise and add up result to get

z

$z[i,j]$

Image Convolution Examples

Translation Convolution:

$*$

$=$

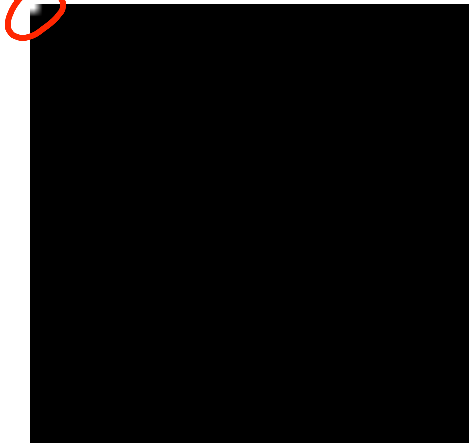
Boundary: "zero"

0

Image Convolution Examples

Translation Convolution:

*



Boundary: "replicate"

=

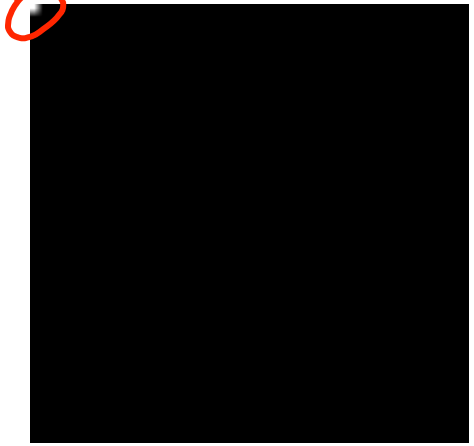
repeats

repeats

Image Convolution Examples

Translation Convolution:

$*$



$=$

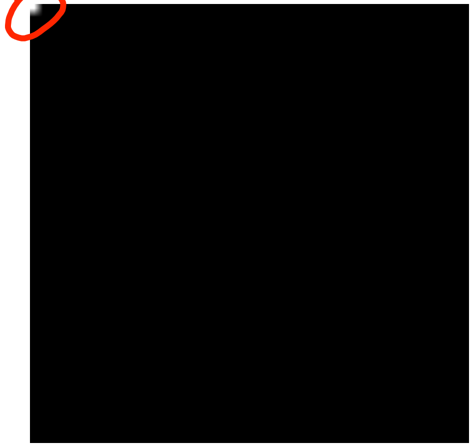
Boundary: "mirror"

flips

Image Convolution Examples

Translation Convolution:

$*$



$=$

Boundary: "ignore"

Image Convolution Examples

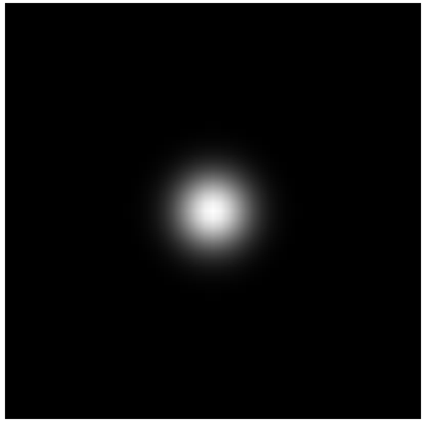
Average convolution:

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

Image Convolution Examples

Gaussian Convolution:

$*$



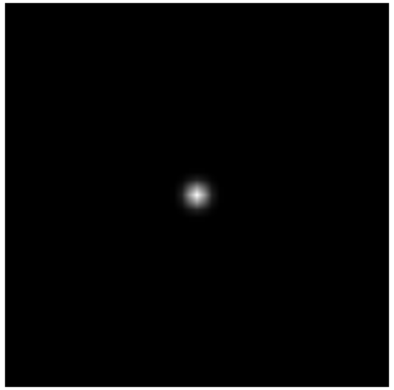
$=$

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

Image Convolution Examples

Gaussian Convolution:

$*$



$=$

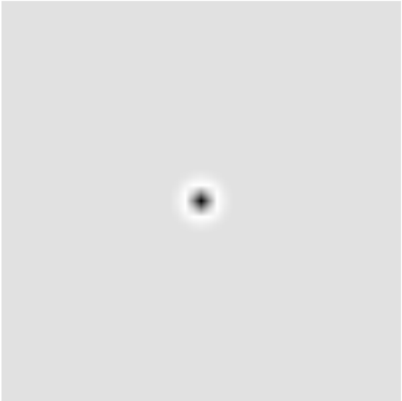
(smaller variance)

blurs image to represent average (smoothing)

Image Convolution Examples

Laplacian of Gaussian

$*$



$=$

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

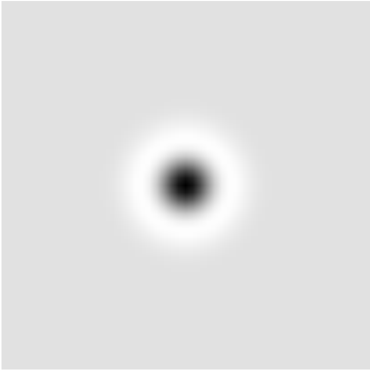
"signed" image (gray is 0)

Image Convolution Examples

Black/white as sides of edge

Laplacian of Gaussian

$*$



$=$

(larger variance)

Similar preprocessing may be done in basal ganglia and LGN.

Image Convolution Examples

"Emboss" filter:

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

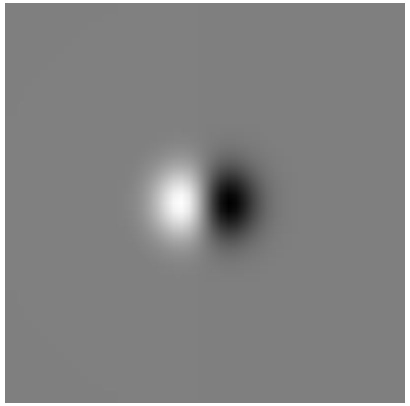
Many Photoshop effects are just convolutions.

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

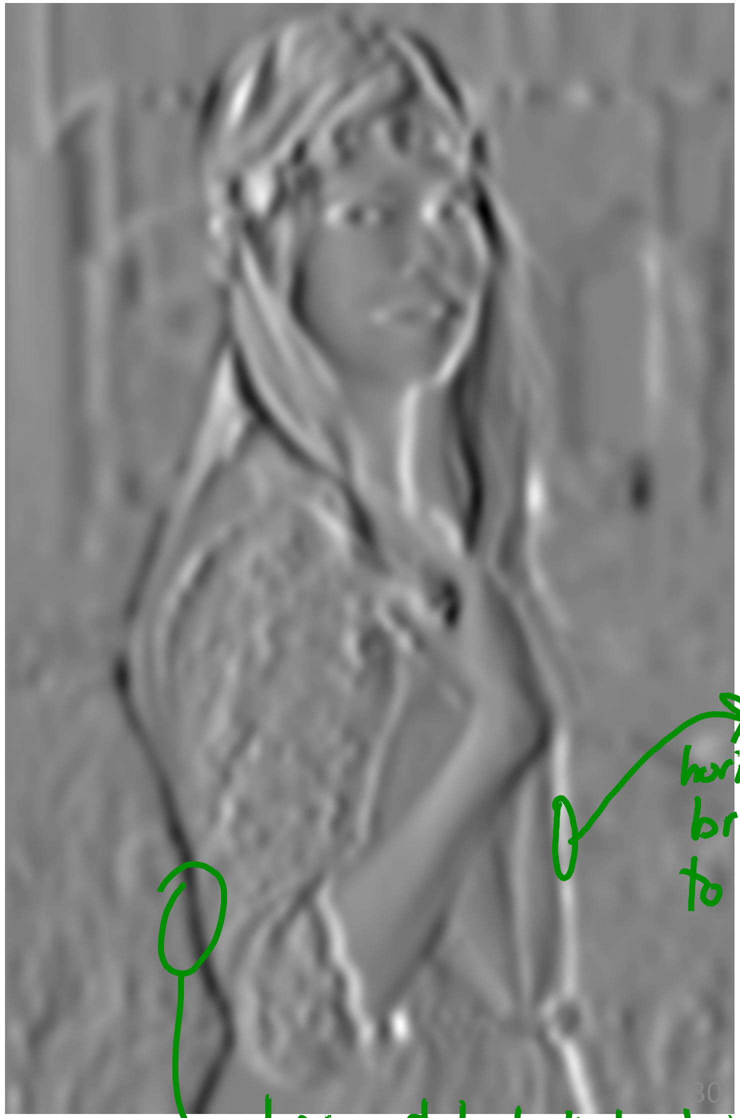
Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

*



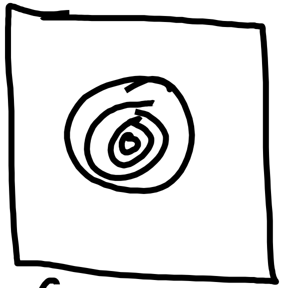
=



horizontal bright to dark

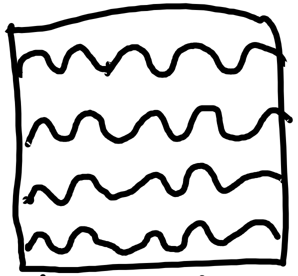
horizontal dark to bright

||



Gaussian

.*



Parallel Sine functions

Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$

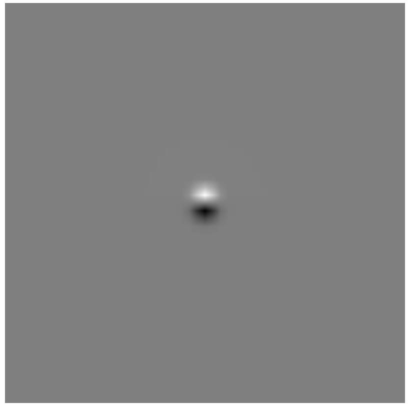
$=$

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

Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$



(smaller variance)

$=$

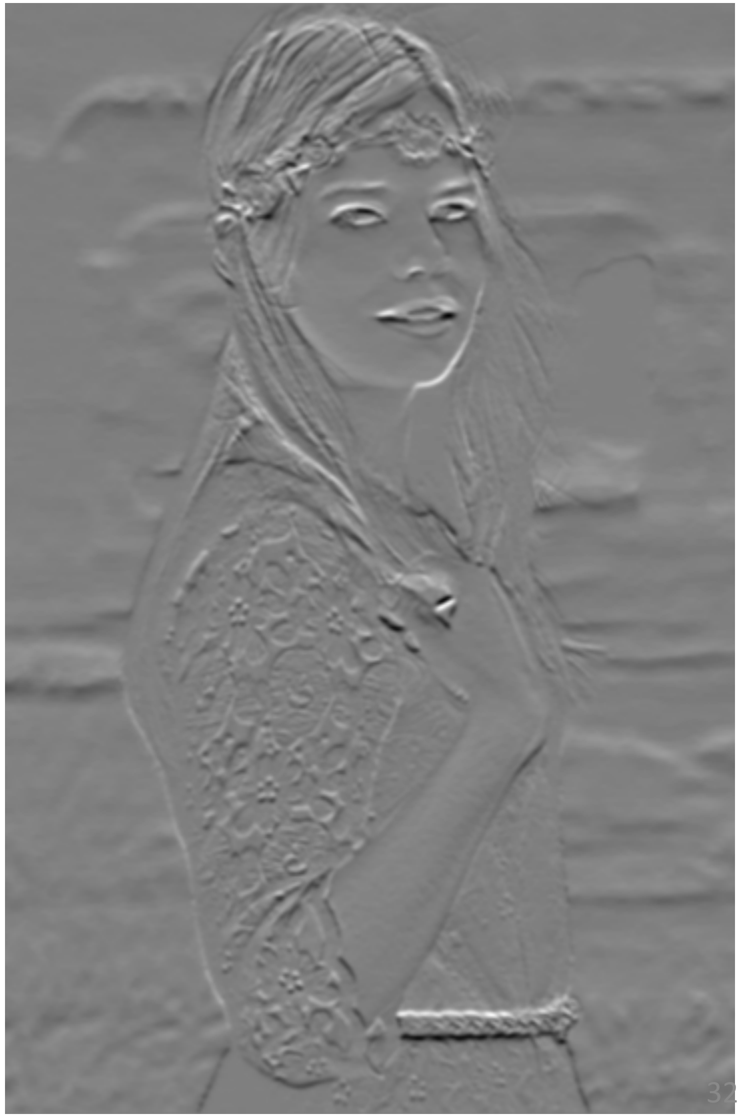
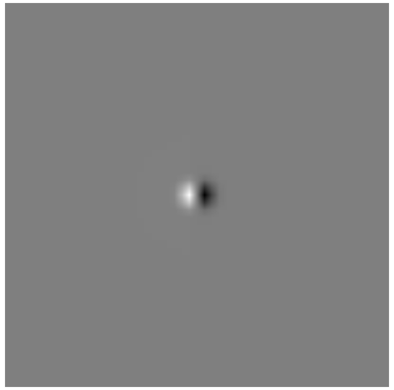


Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$



$=$

(smaller variance)

Vertical orientation

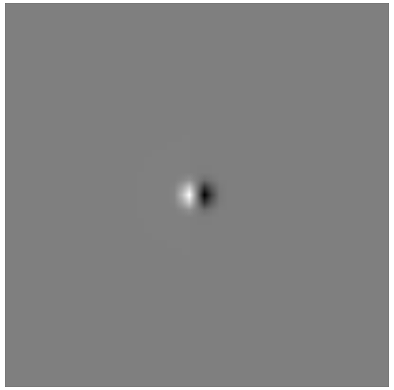
– Can obtain other orientations by rotating.

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

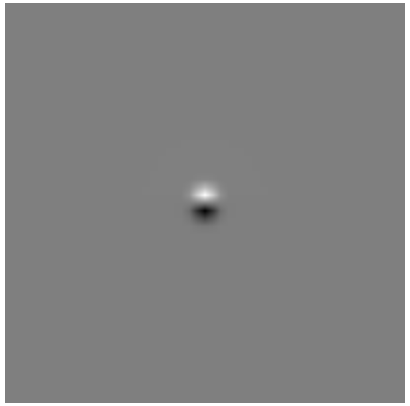
Image Convolution Examples

Max absolute value between horizontal and vertical Gabor:

$*$



$*$



maximum absolute value

"Horizontal/vertical edge detector"

3D Convolution

Represent as RGB

Can apply 3D convolutions

3D Convolution

Gaussian filter

3D Convolution

Gaussian filter
(higher variance on green channel)

3D Convolution

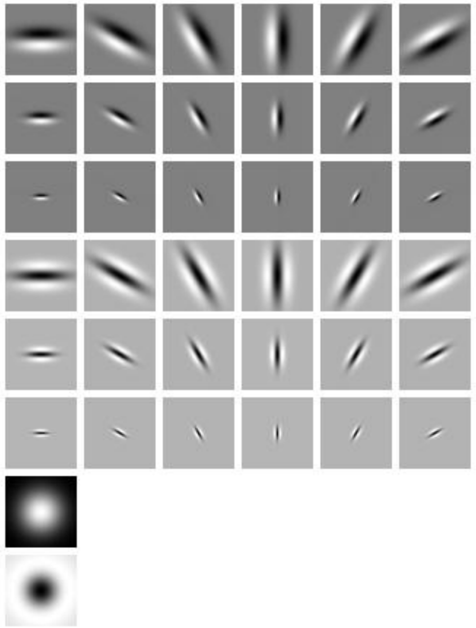
Sharpen the blue channel.

3D Convolution

Gabor filter on each channel.

Filter Banks

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

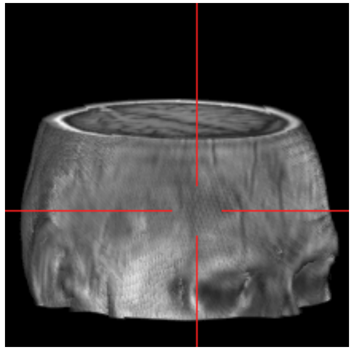


- Convolutional neural networks are now replacing filter banks.

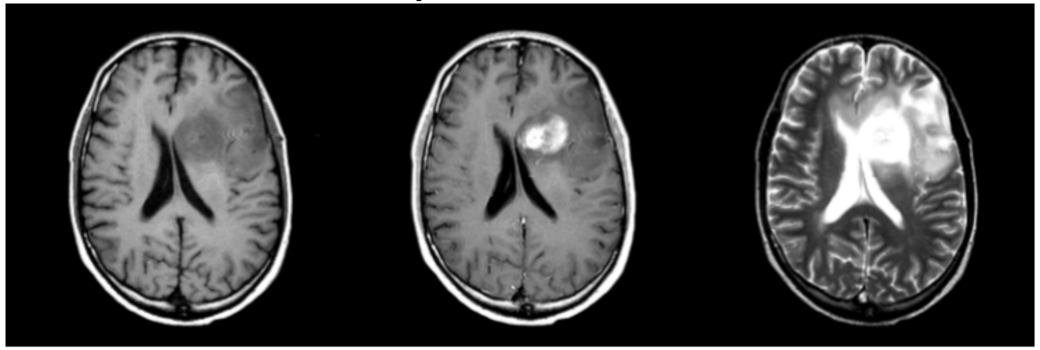
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Motivation: Automatic Brain Tumor Segmentation

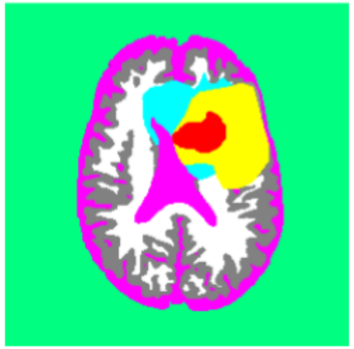
- Task: segmentation tumors and normal tissue in multi-modal MRI data.



Input:



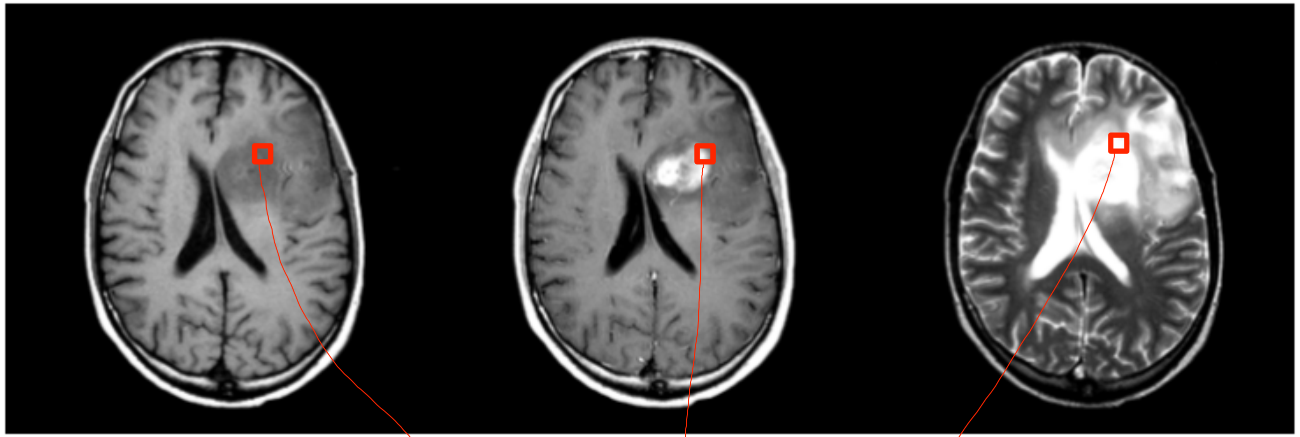
Output:



- 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:

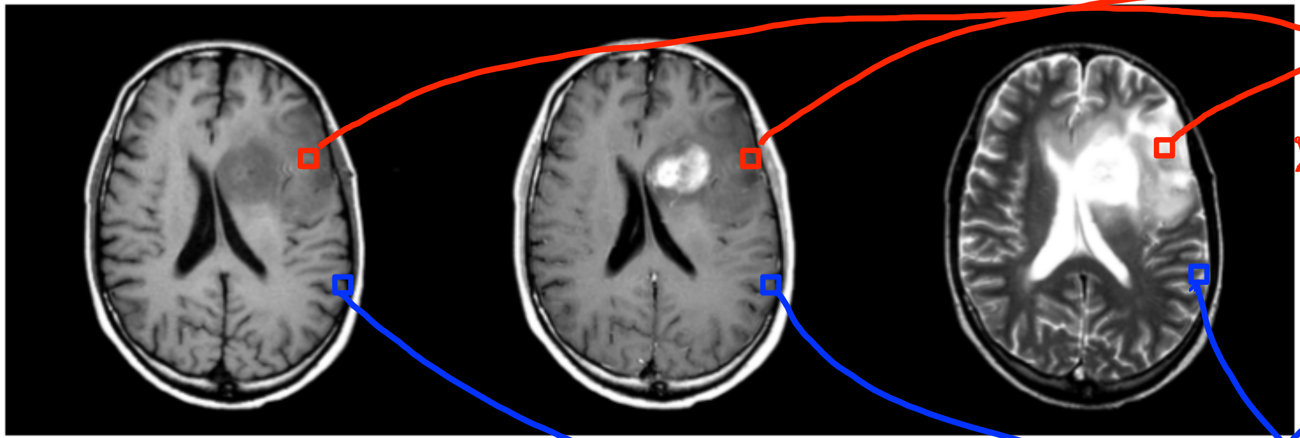


$x_i = (98, 187, 246)$ $y_i = \text{"tumour"}$

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

Need to Summarize Local Context

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



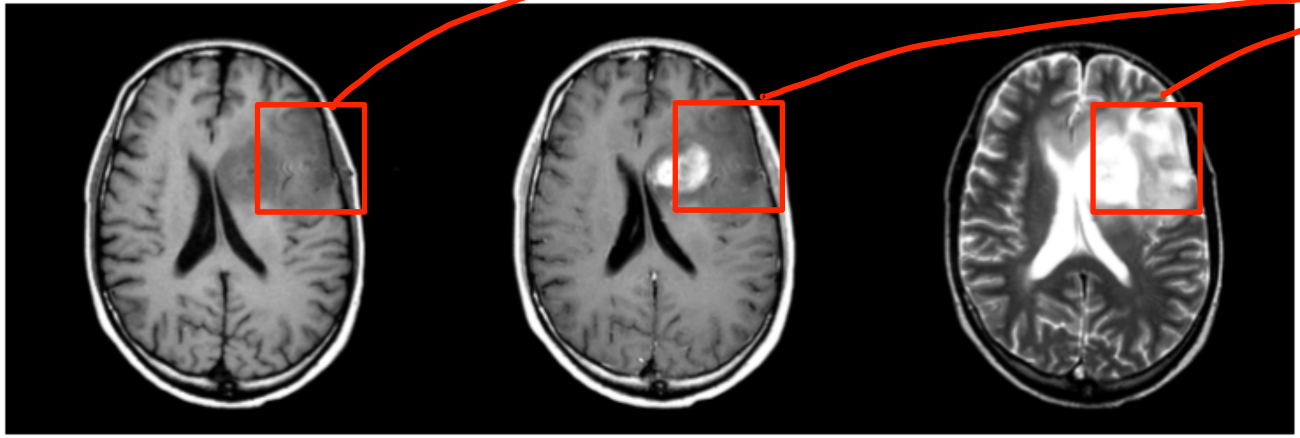
$x_i = (87, 54, 173)$ $y_i =$ "tumour"

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



$x_i = (\cdots , \cdots , \cdots)$

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

Number of parameters?

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

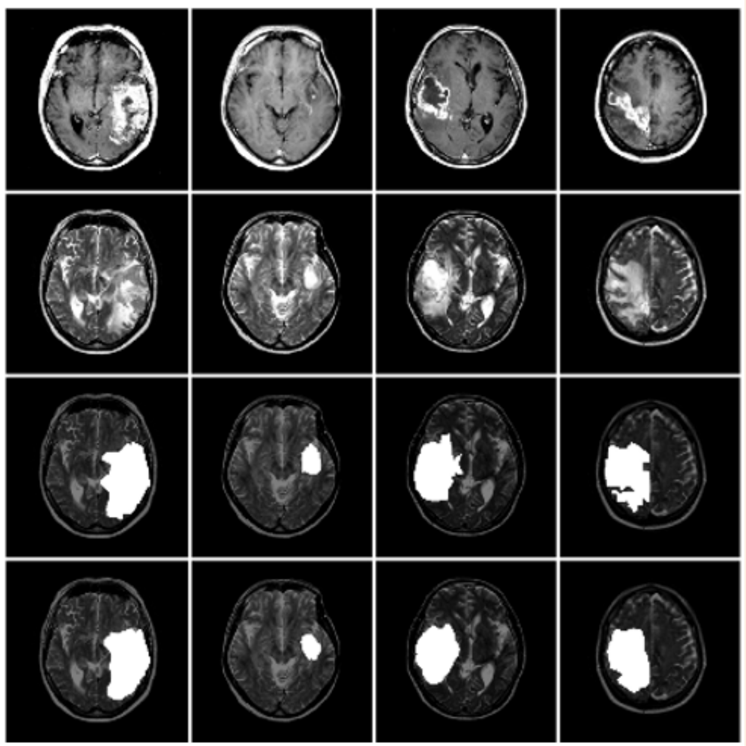
FFT implementation of convolution

- Convolutions can be implemented using fast Fourier transform:
  - Take FFT of image and filter, multiply elementwise, and take inverse FFT.

- It has faster asymptotic running time but there are some catches:
  - You need to be using periodic boundary conditions for the convolution.
  - Constants matter: it may not be faster in practice.
    - Especially compared to using GPUs to do the convolution in hardware.
  - The gains are largest for larger filters (compared to the image size).

Motivation: Automatic Brain Tumor Segmentation

- Brain tumour segmentation formulated as supervised learning:
  - Pixel-level classifier that predicts “tumour” or “non-tumour”.
  - Features: convolutions, expected values (in aligned template), and symmetry (all at multiple scales).



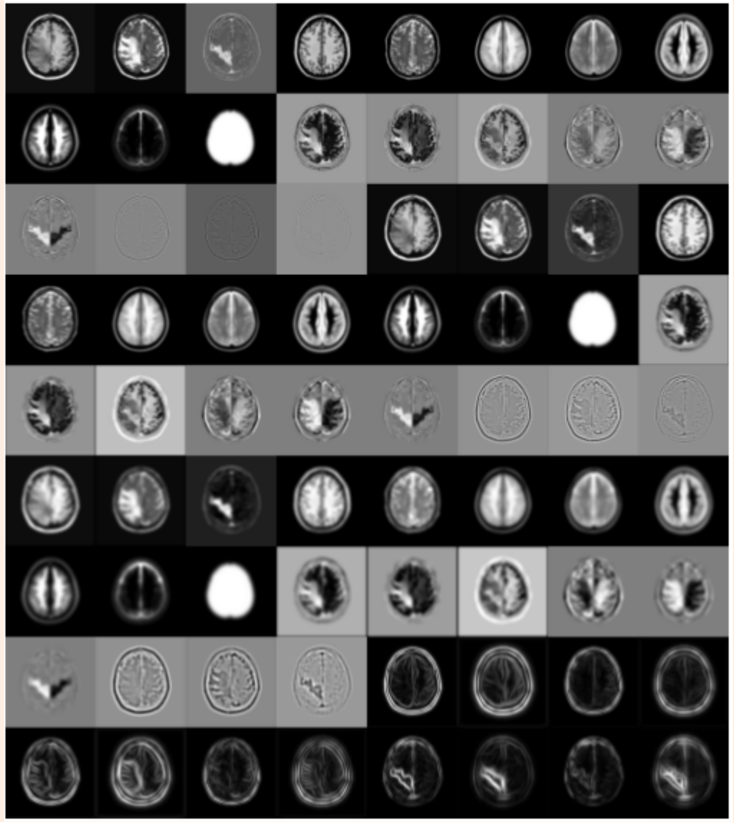
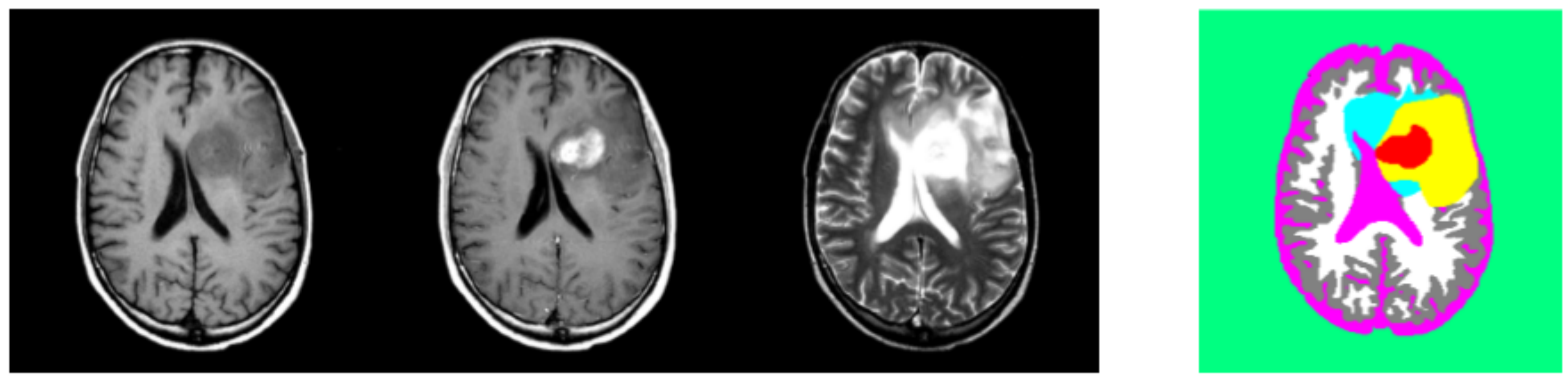


Image Coordinates

- 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?

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

LeNet for Optical Character Recognition