CPSC 340:
Machine Learning and Data Mining

Convolutional Neural Networks

Quick aside: neural networks for classification

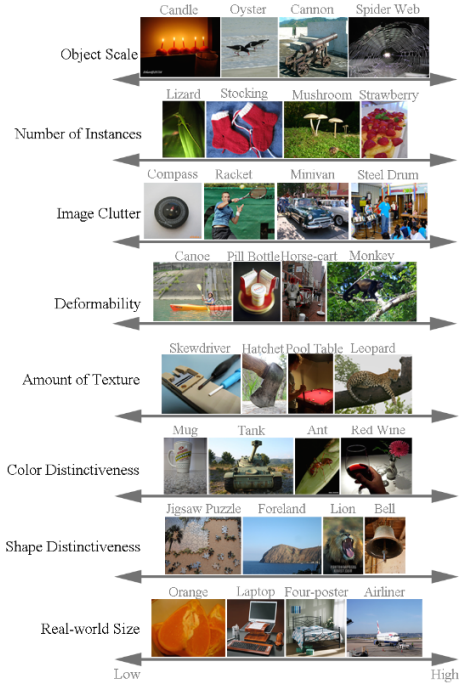
- What if we want to do *k*-class classification with neural networks?
  - we use a softmax layer at the end
  - this means the output layer has *k* units (a vector of length *k*)
  - we can use the same loss function as for softmax logistic regression
  - other loss functions also exist, like the cross-entropy loss

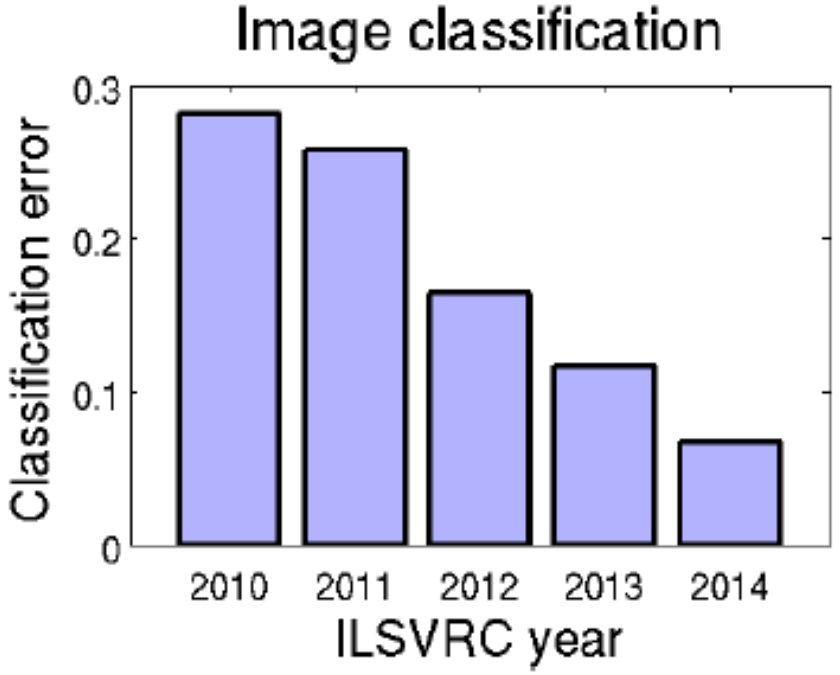
Last Lectures: Deep Learning

- We've been discussing neural network / deep learning models:

$$y_i = w^T h(W^{(m)} h(W^{(m-1)} h(\cdots W^{(2)} h(W^{(1)} x_i)) \cdots))$$

- Last time we discussed unprecedented vision/speech performance.





https://arxiv.org/pdf/1409.0575v3.pdf

Last Lectures: Deep Learning

- We've been discussing neural network / deep learning models:

$$y_i = w^T h(W^{(m)} h(W^{(m-1)} h(\cdots W^{(2)} h(W^{(1)} x_i)) \cdots))$$

- Last time we discussed heuristics to make it work:
  - Parameter initialization and data transformations.
  - Setting the step size(s) in stochastic gradient.
  - Alternative non-linear functions like ReLU.
  - Different forms of regularization:
    - L2-regularization, early stopping, dropout.
- These are often still not enough to get deep models working.

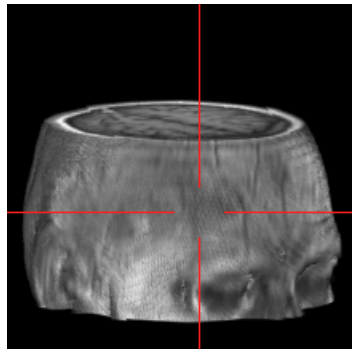
Convolutional Neural Networks

- NNs typically use multiple types of regularization:
  - L2-regularization.
  - Early stopping.
  - Dropout.
- Often, still not enough to get deep models working.

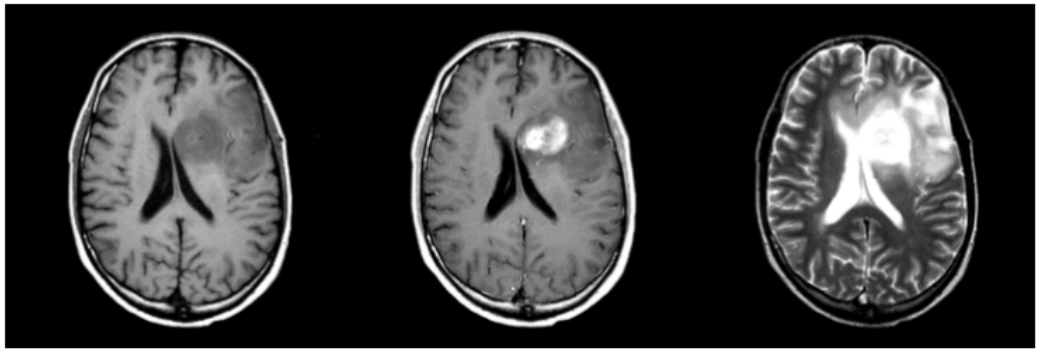
- Deep computer vision models are all convolutional neural nets:
  - The $W^{(m)}$ are very sparse and have repeated parameters ("tied weights").
  - Drastically reduces number of parameters (speeds up training).

Motivation: Automatic Brain Tumor Segmentation

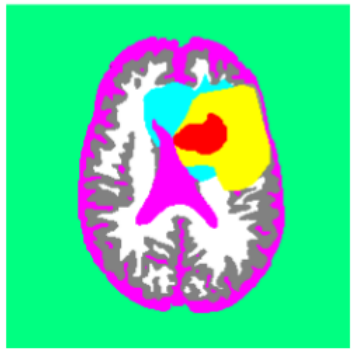
- Task: segmentation tumors and normal tissue in multi-modal MRI data.



Input:



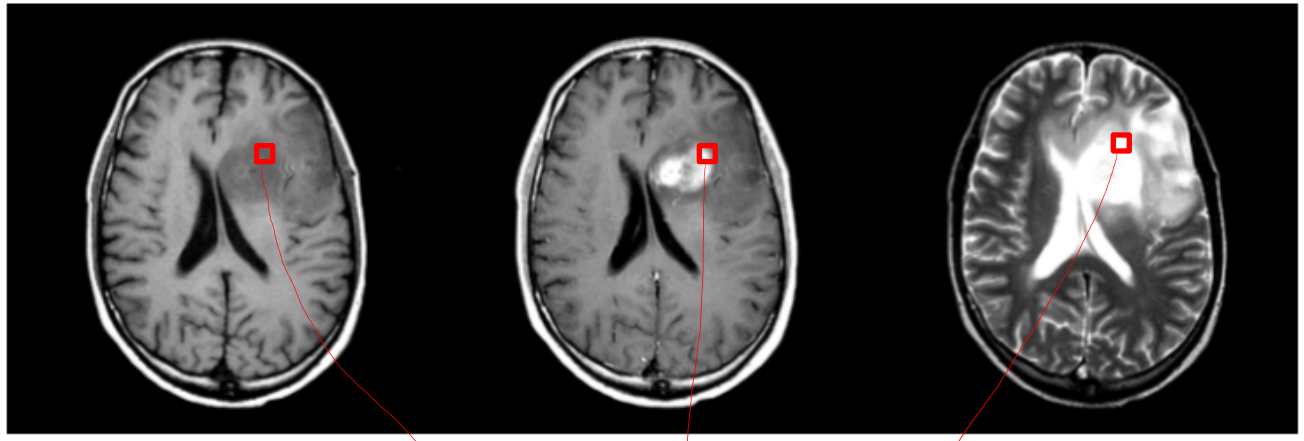
Output:



- 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:

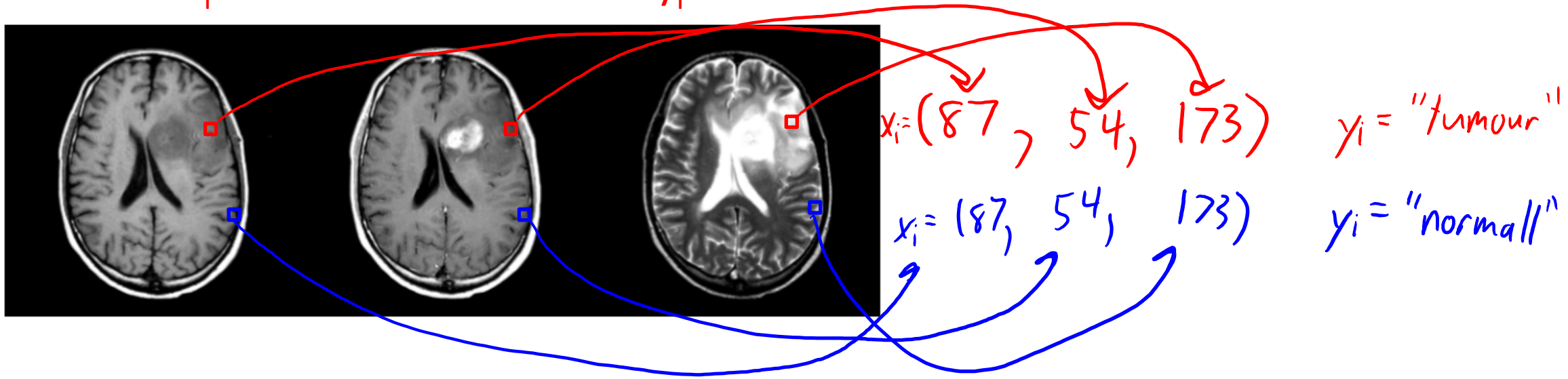


$x_i = (98, 187, 246)$ $y_i = \text{"tumour"}$

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

Need for Context

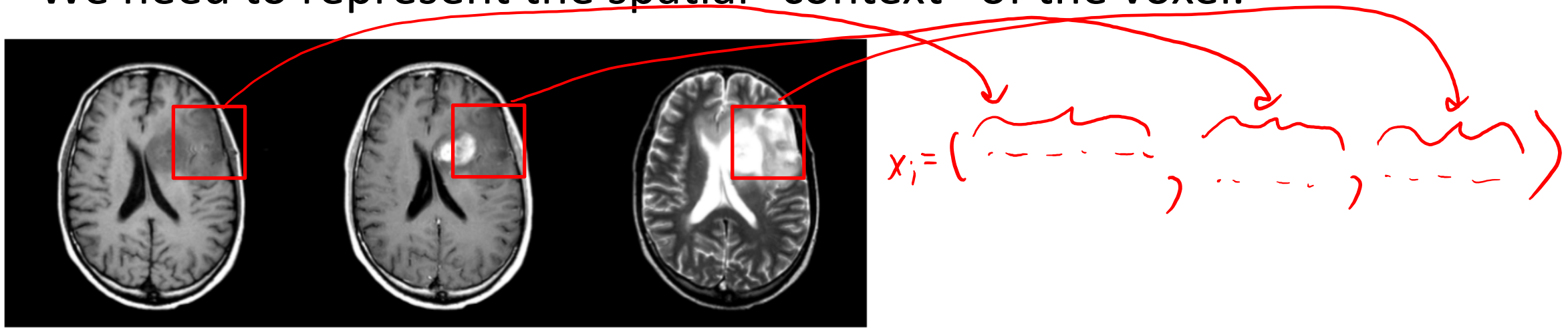
- The individual voxel values are almost meaningless:
  - This x_i could lead to different y_i.



- Intensities not standardized.
- Non-trivial overlap in signal for different tissue types.
- "Partial volume" effects at boundaries of tissue types.

Need for Context

- We need to represent the spatial "context" of the voxel.



  - Include all the values of neighbouring voxels?
    - Using all voxels requires lots of data to find patterns.
  - Measure summary statistics (mean, variance, etc.) of the neighbourhood?
    - Loses spatial information present in voxels.
  - Standard approach is uses convolutions to represent neighbourhood.

1D Convolution

- 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

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

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

- 1D convolution output:
  - New vector ‘z’ of length ‘n’ with elements:

$$z_i = w_{-m} x_{i-m} + w_{-m+1} x_{i-m+1} + \cdots + w_{m+1} x_{i+m+1} + w_m x_{i+m}$$

  - You can think of this as centering w at z_i and taking a dot product.

1D Convolution Examples

- Element z_i of 1D convolution is given by:

$$z_i = w_{-m} x_{i-m} + w_{-m+1} x_{i-m+1} + \cdots + w_{m+1} x_{i+m+1} + w_m x_{i+m}$$

- Examples:

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

  - "Identity"

$w = [0\ 1\ 0]$

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

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

  - "Translation"

$w = [0\ 0\ 1]$

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

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

1D Convolution Examples

- Element z_i of 1D convolution is given by:

$$z_i = w_{-m} x_{i-m} + w_{-m+1} x_{i-m+1} + \cdots + w_{m+1} x_{i+m+1} + w_m x_{i+m}$$

- Examples:
  - "Identity"

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

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

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

  - "Average"

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

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

Boundary Issue

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

If x=[0 1 1 2 3 5 8 13] and w=[⅓ ⅓ ⅓] then z=[? 2⅓ 1⅓ 2 3⅓ 5⅓ 8⅔ ?]

- 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=[2⅓ 1⅓ 2 3⅓ 5⅓ 8⅔]

1D Convolution in Matrix Notation

- Each element of a convolution is an inner product:

$$z_i = w_{-m} x_{i-m} + w_{-m+1} x_{i-m+1} + \cdots + w_{m+1} x_{i+m+1} + w_m x_{i+m}$$

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

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

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

- So convolution is a matrix multiplication:

$$z = \tilde{W} x \quad \text{where} \quad \tilde{W} = \begin{bmatrix} \text{—}\ w\ \text{—} & 0 & 0 & 0 \\ 0 & \text{—}\ w\ \text{—} & 0 & 0 \\ 0 & 0 & \text{—}\ w\ \text{—} & 0 \\ 0 & 0 & 0 & \text{—}\ w\ \text{—} \end{bmatrix}$$

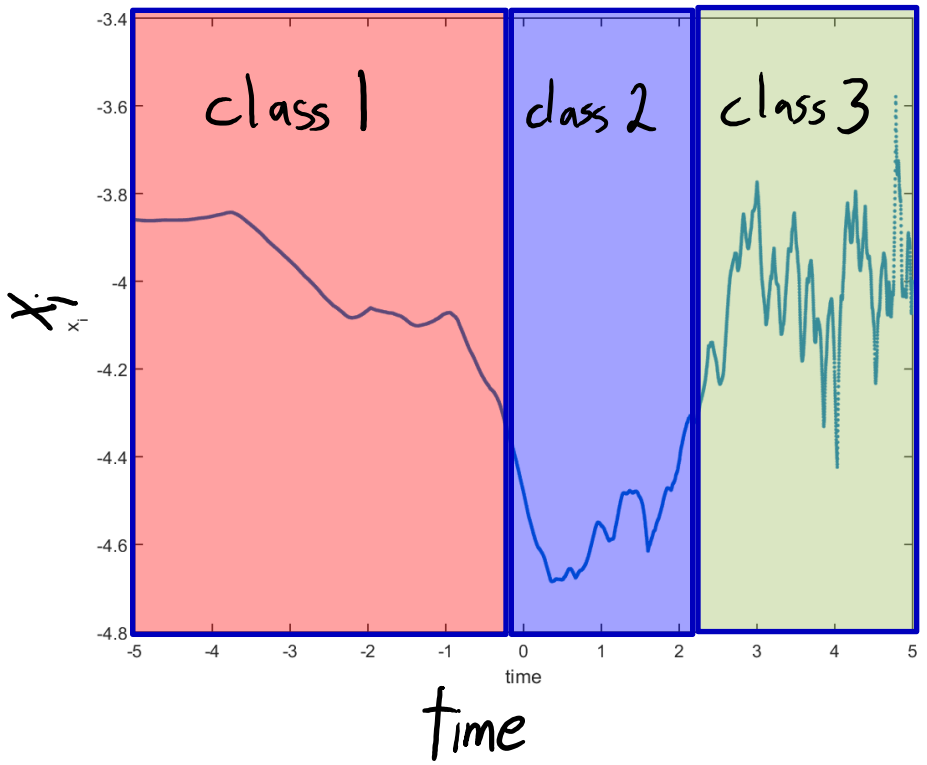
matrix can be very sparse and only has $2m+1$ variables.

1D Convolution in Matrix Notation

- Ask yourself, for element `i` of the output vector, what elements of the input vector does it depend on?
- When writing this as matrix multiplication, we have a `0` in that column if the output doesn’t depend on that input.
- The smaller the filters, the more sparse the matrix is.

Why is this useful?

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

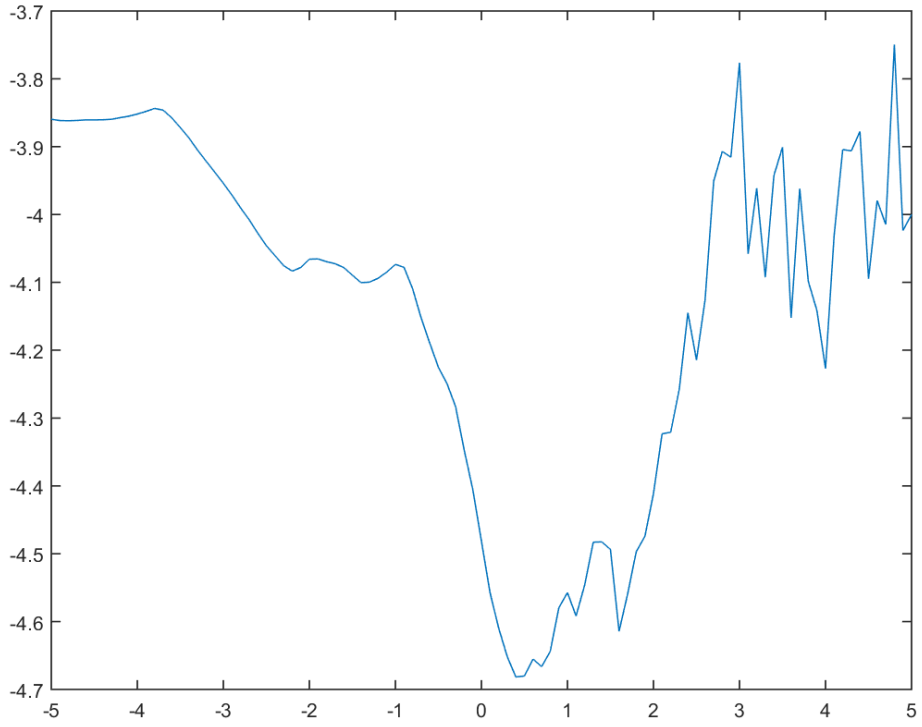


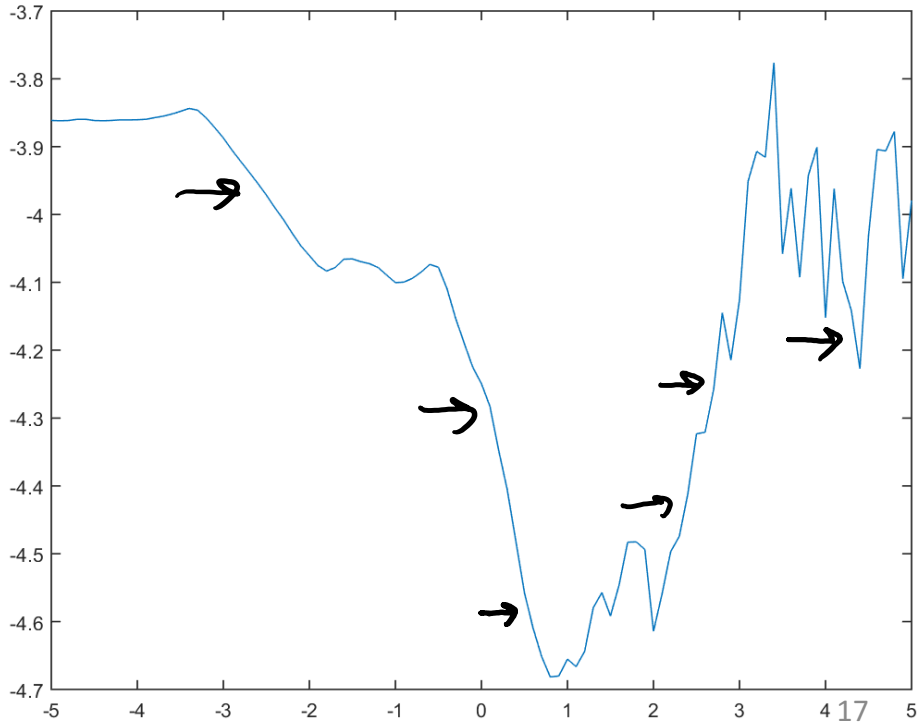
- Easy to distinguish class 2 from the other classes (x_i are smaller).
- Harder to distinguish between class 1 and class 3 (similar x_i range).
  - But convolutions can represent that class 3 is more "spiky".

1D Convolution Examples

- Translation convolution shift signal:

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

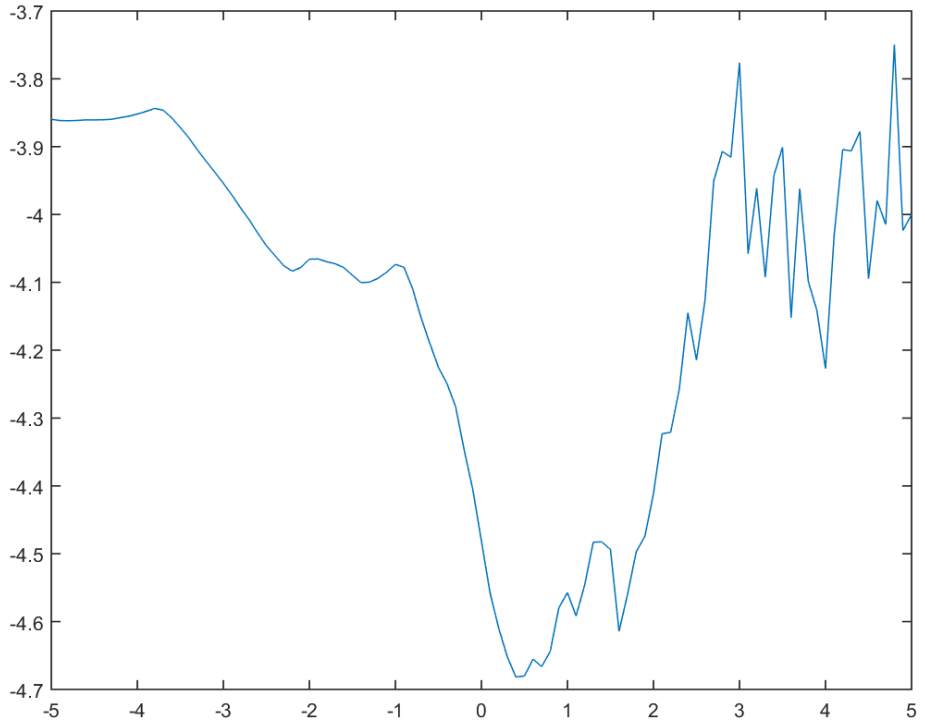


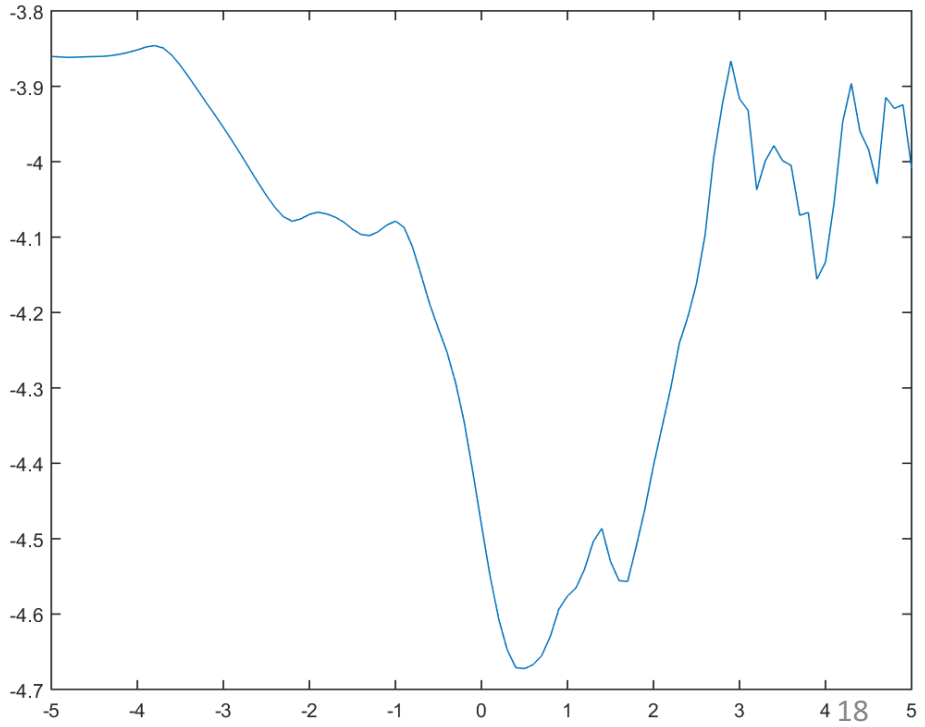


1D Convolution Examples

- Averaging convolution computes local mean:

$$w = [\,1/3 \quad 1/3 \quad 1/3\,]$$

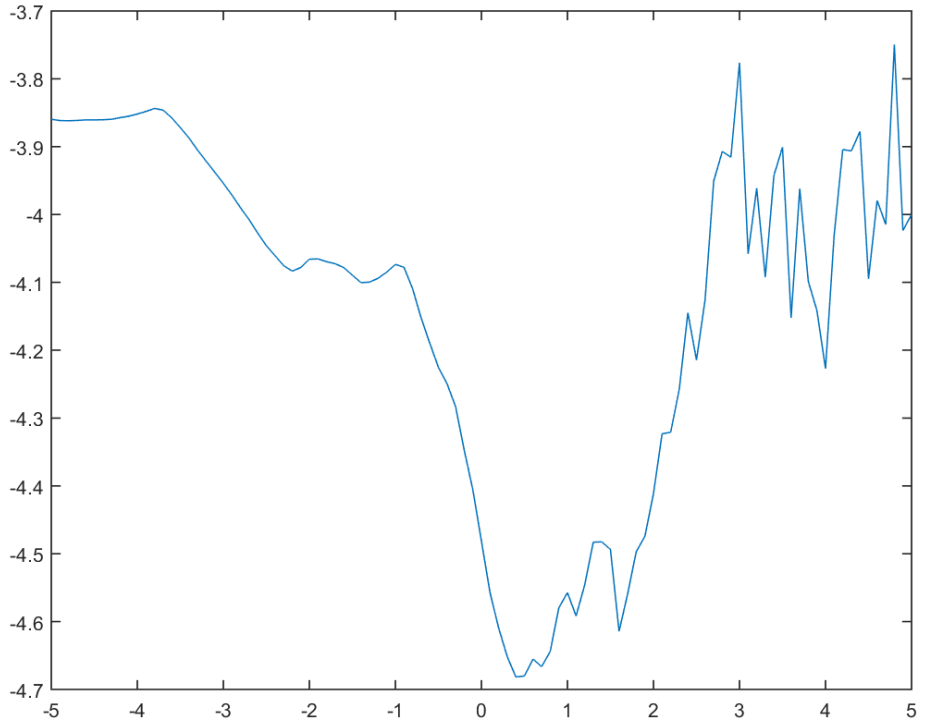


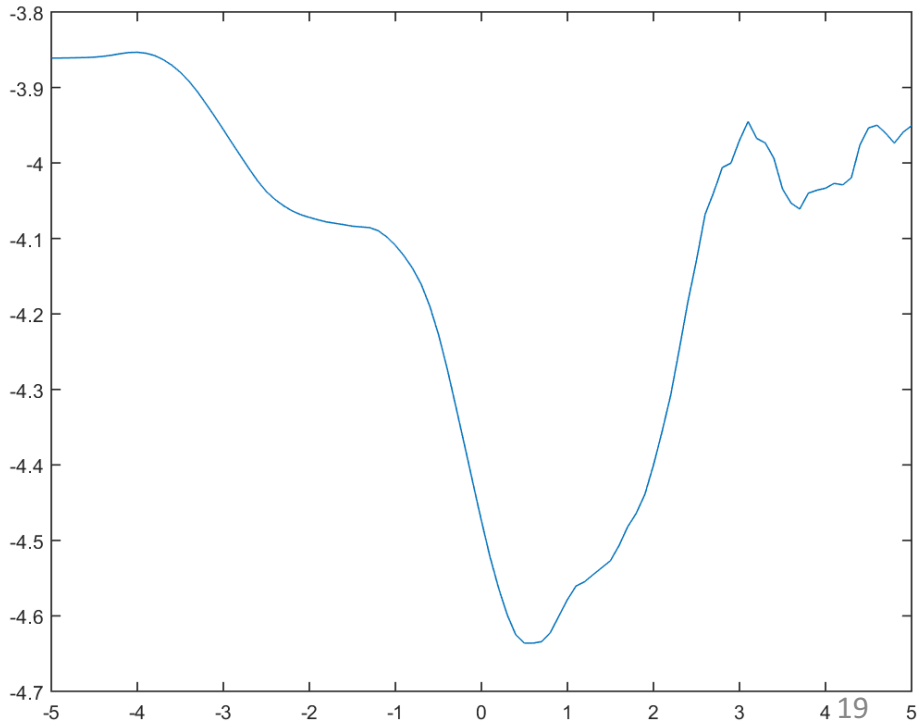


1D Convolution Examples

- Averaging over bigger window gives coarser view of signal:

$$w = [1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9]$$



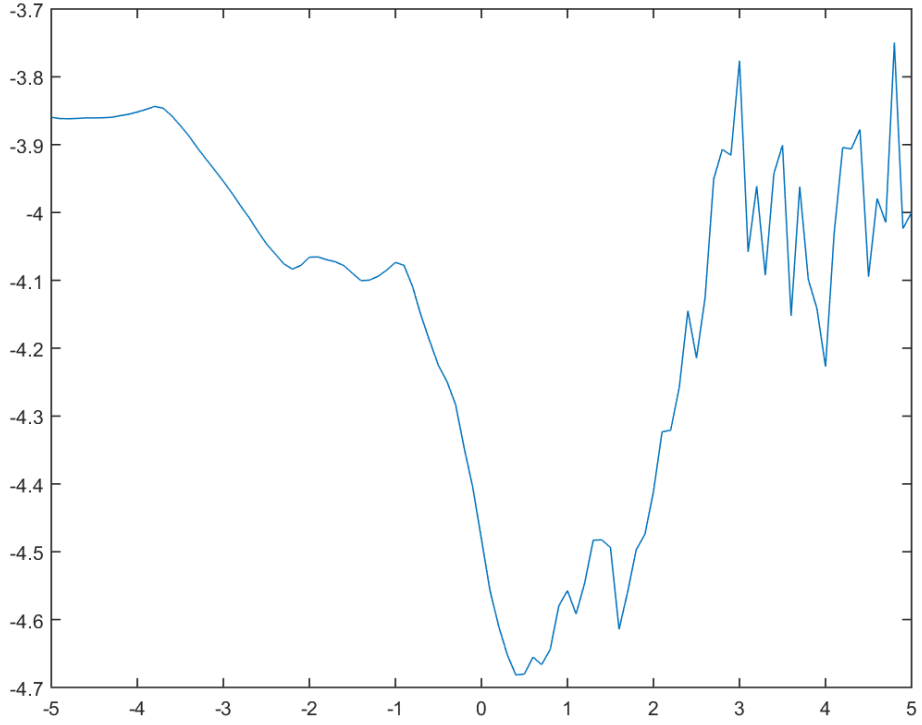


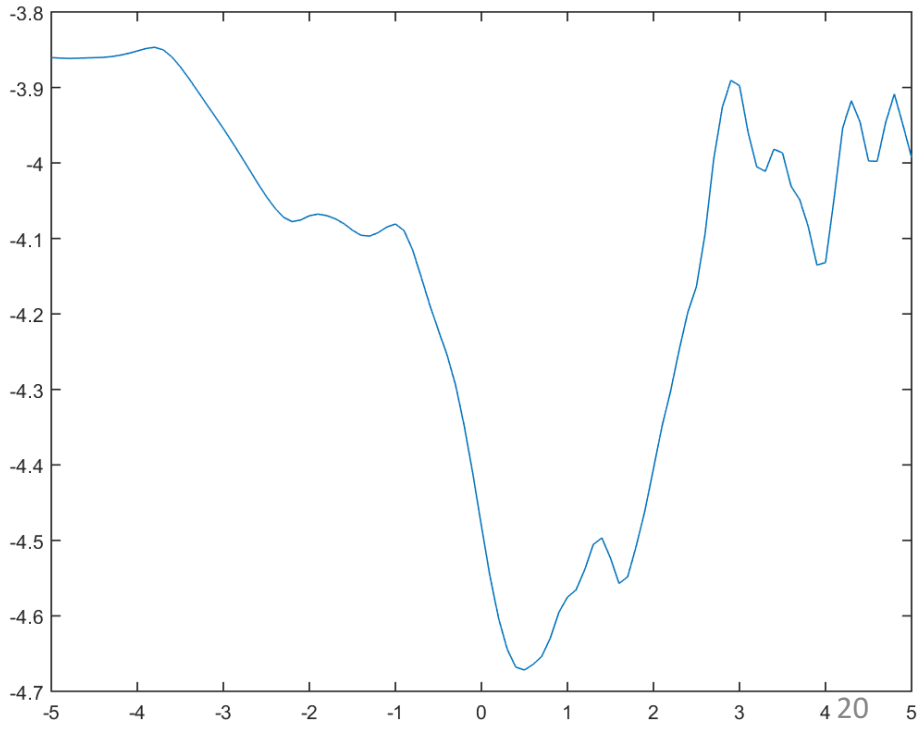
1D Convolution Examples

- 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.

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

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

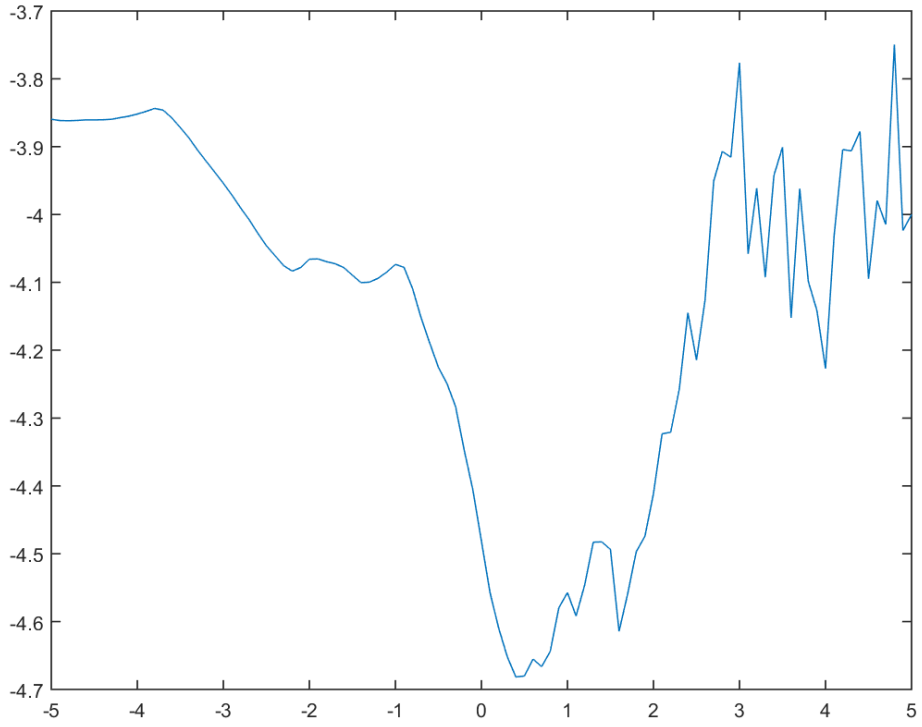


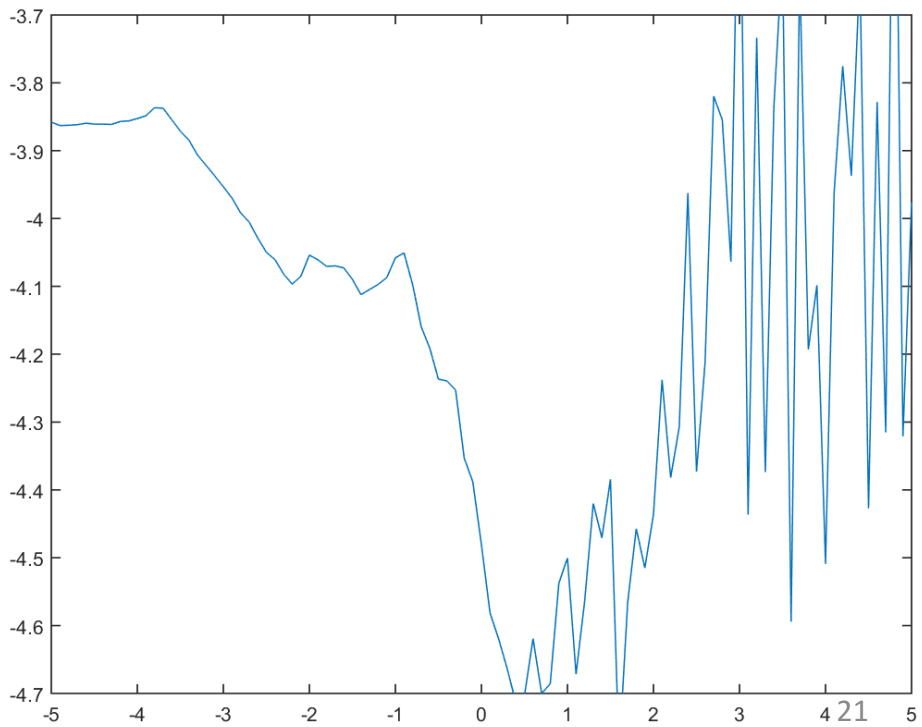


1D Convolution Examples

- Sharpen convolution enhances peaks.

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

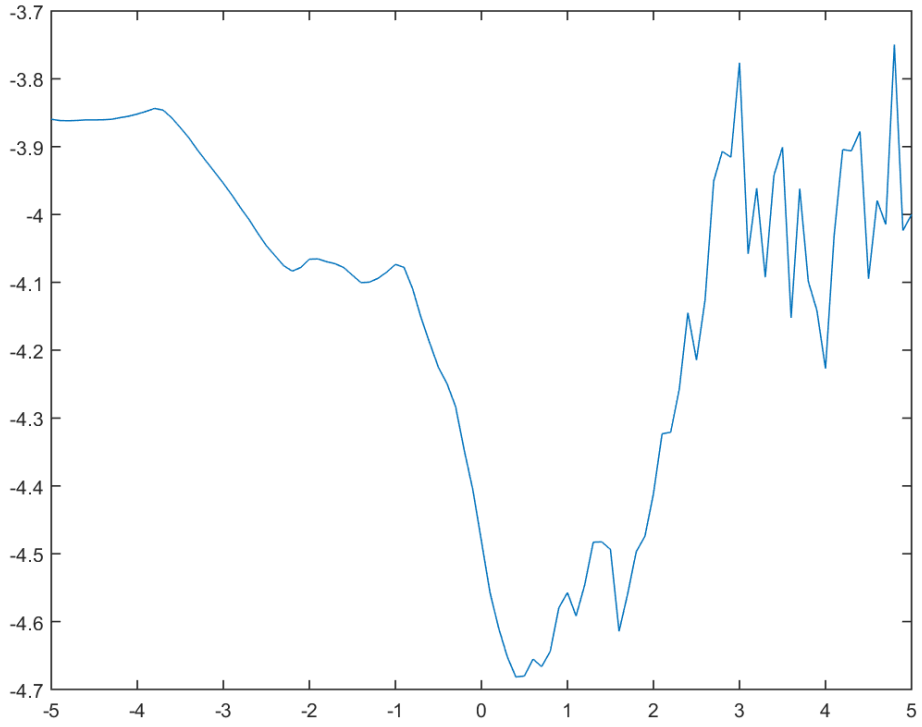


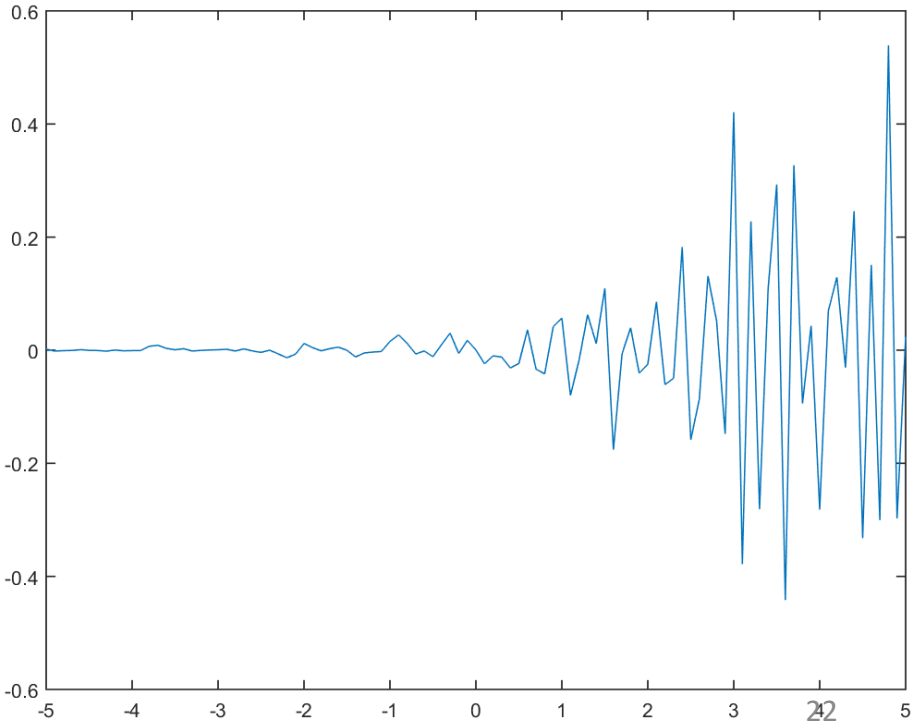


1D Convolution Examples

- Laplacian convolution approximates second derivative:

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

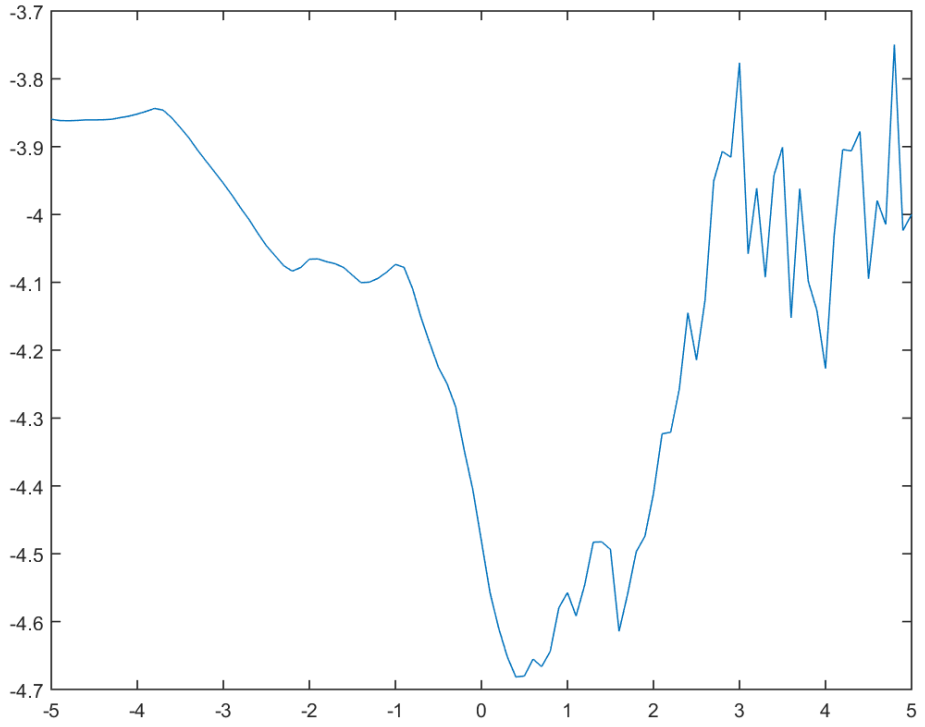


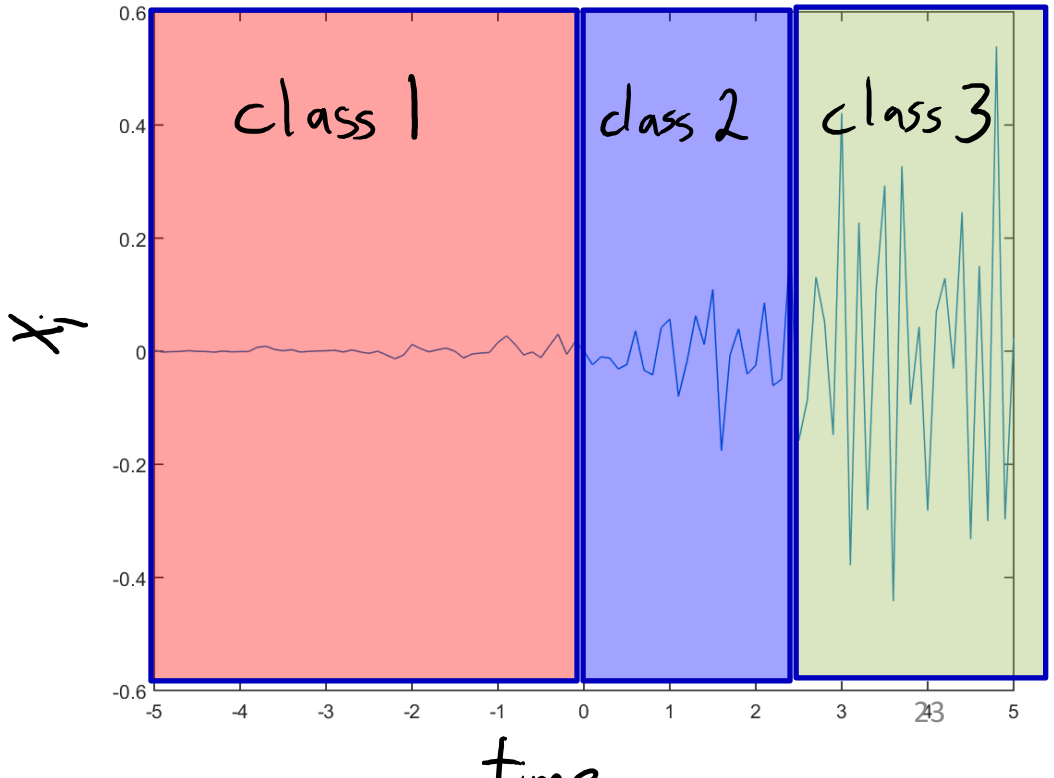


1D Convolution Examples

- Laplacian convolution approximates derivative:

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





Quick aside: finite differences

- If you've taken CPSC 303 you can now interpret all the finite difference formulas as convolutions with various filters
  - Forward difference [1, -1]
  - Centred difference [1, 0, -1]
  - Centred 2nd derivative [-1, 2, 1]
- And likewise for integrals:
  - Trapezoid [1/2, 1/2]
  - Simpson [1/6, 4/6, 1/6]
- It's not a coincidence that the difference formulas add to 0 and the integration formulas add to 1
  - They need to give the correct answer on a constant function.

Another way to think about convolutions

- A convolution “responds” when the input vector looks like the filter
- This is because it’s a dot product, which is maximizes when the two vectors are parallel
- Thus, the filter [-1, 1] gives a big output when things *change* from one element to the next
  - And the sign reflects the direction of change
  - So it makes sense that this approximates a derivative
  - OTOH the response is 0 if the input signal is flat (constant)
- This intuition will be useful when we talk about convnets shortly…

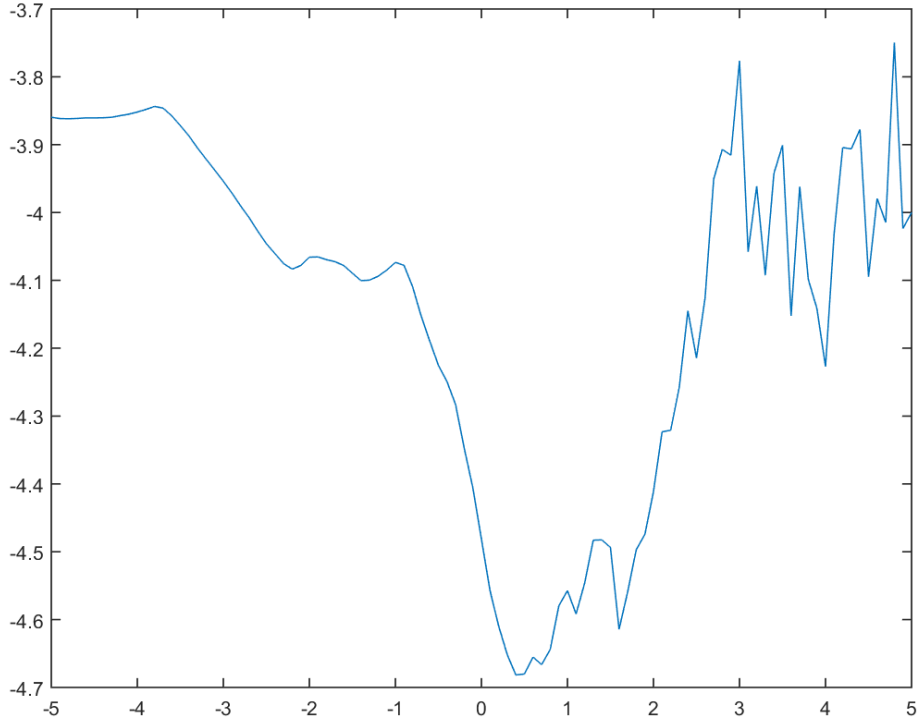
1D Convolution Examples

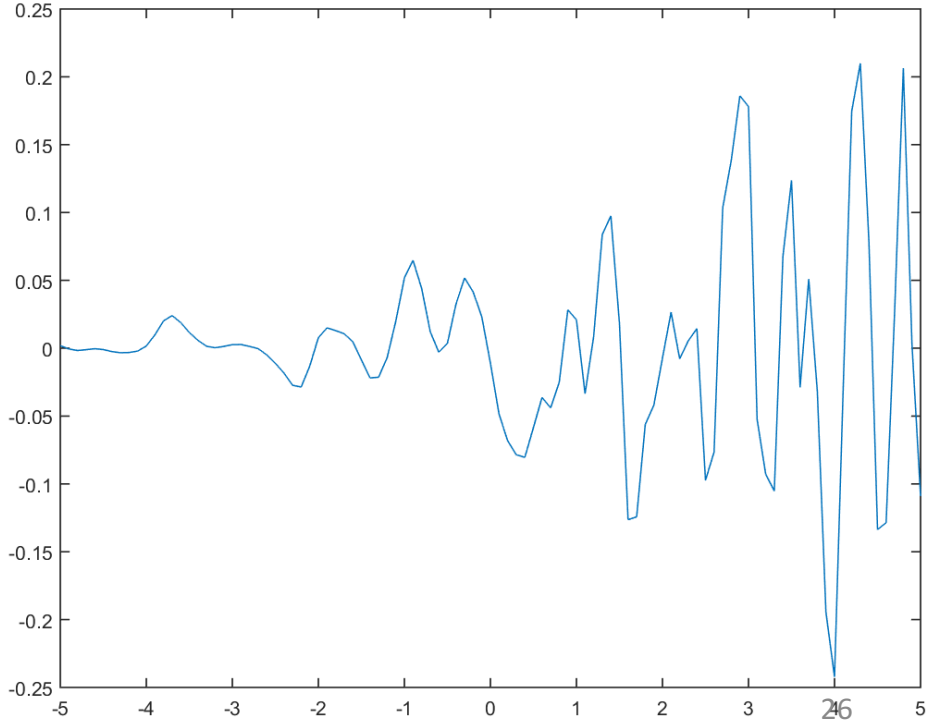
- Laplacian of Gaussian approximates derivative after blurring:

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

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

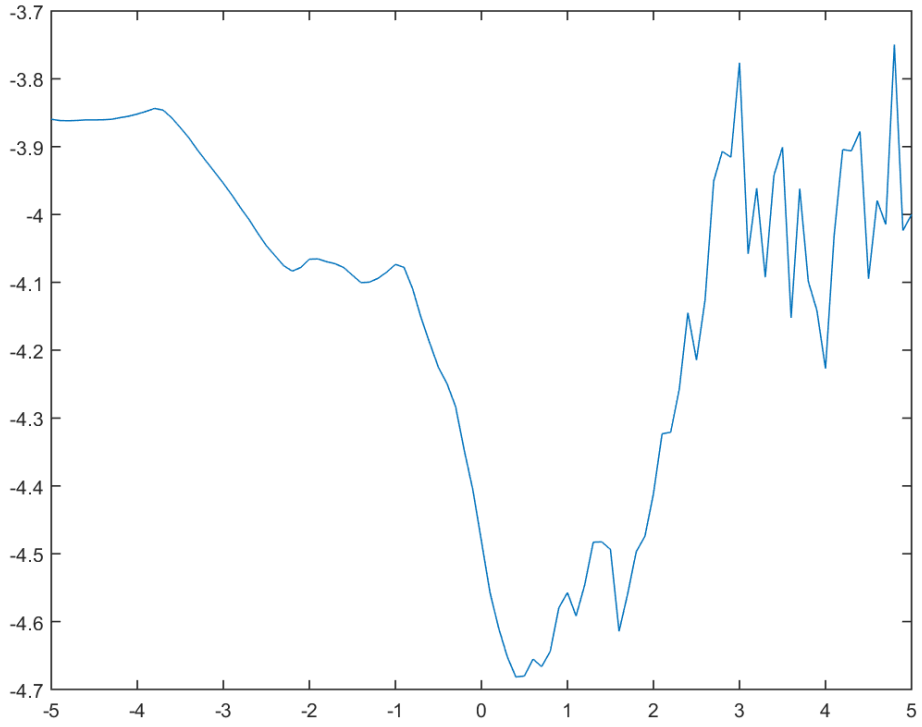
$$(\sigma^2 = 1, \ m = 4)$$

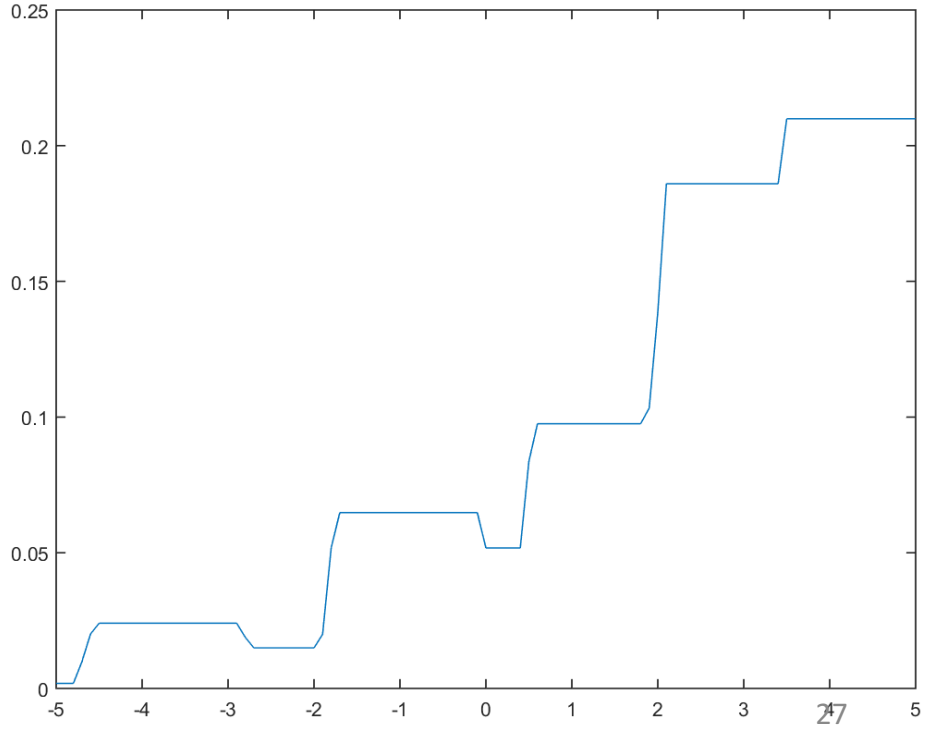




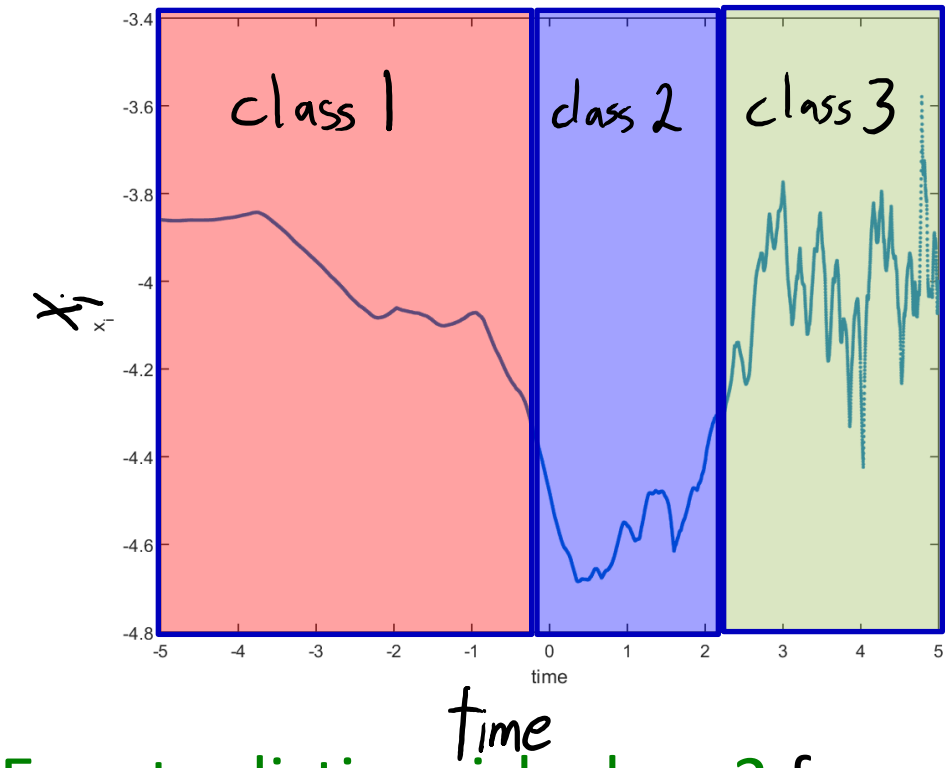
1D Convolution Examples

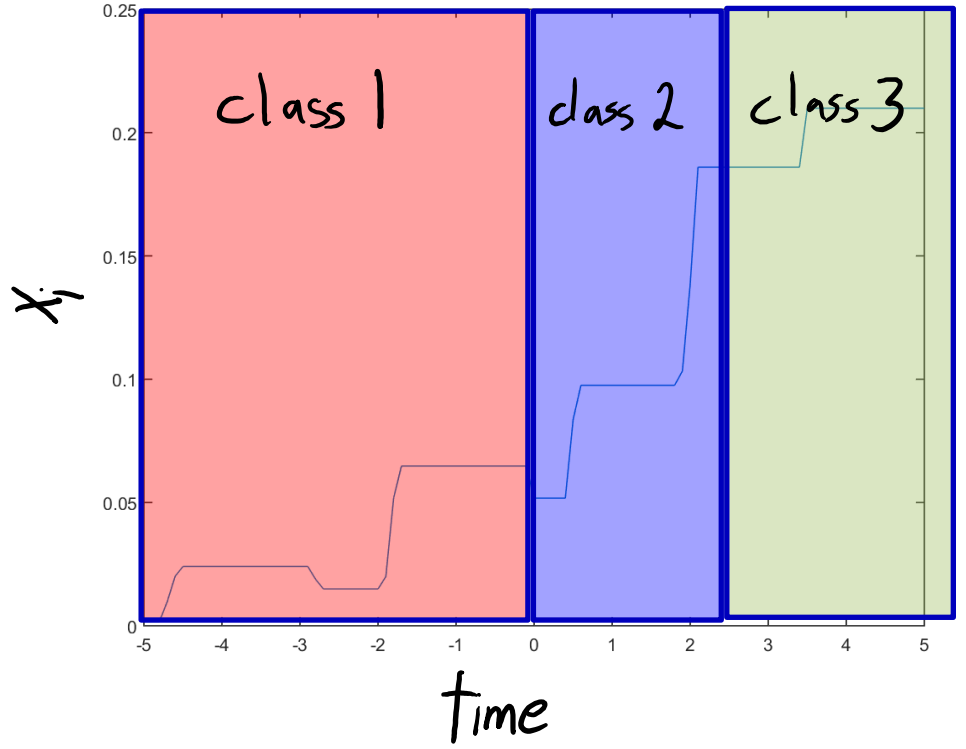
- We often use maximum over several convolutions as features:
  - We could take maximum of Laplacian of Gaussian over x_i and its 16 neighbours.
  - We can use a bunch of different convolutions as a collection of features





Why is this useful?





- Easy to distinguish class 2 from with original signal.
- Easy to distinguish class 1 from 3 with max(Laplacian(Gaussian)).
  - Convolutions and max(convolutions) are very useful for sequence data.
    - For sound data two related techniques are Fourier transforms and spectrograms.

Images and Higher-Order Convolution

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

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

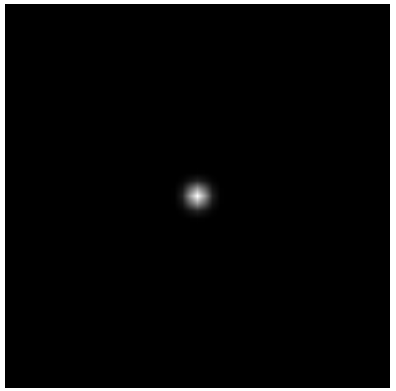
- 3D and higher-order convolutions are defined similarly.

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

Image Convolution Examples

Gaussian Convolution:

$*$



(smaller variance)

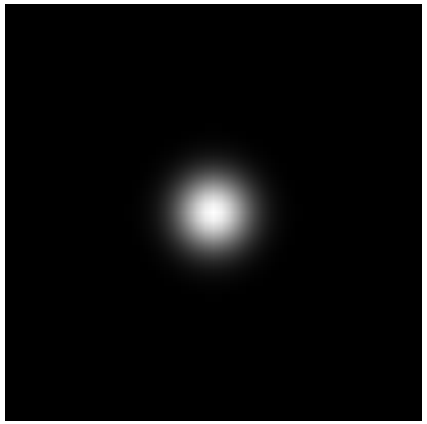
$=$

(smooths image)

Image Convolution Examples

Gaussian Convolution:

*



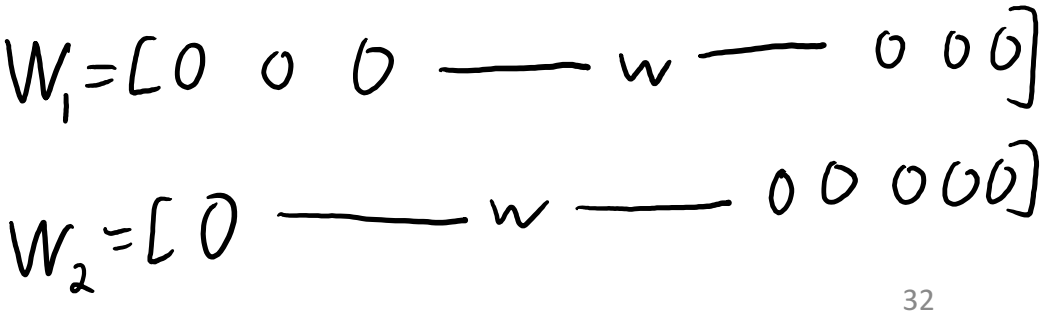
=

(smooths image)

Motivation for Convolutional Neural Networks

- Consider training neural networks on 256 by 256 images.
  - This is 256x256x3 ≈ 200,000 inputs
- Imagine the first layer size is 10,000
- The first layer has about 2 billion parameters
  - We want to avoid this huge number (due to storage and overfitting).

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

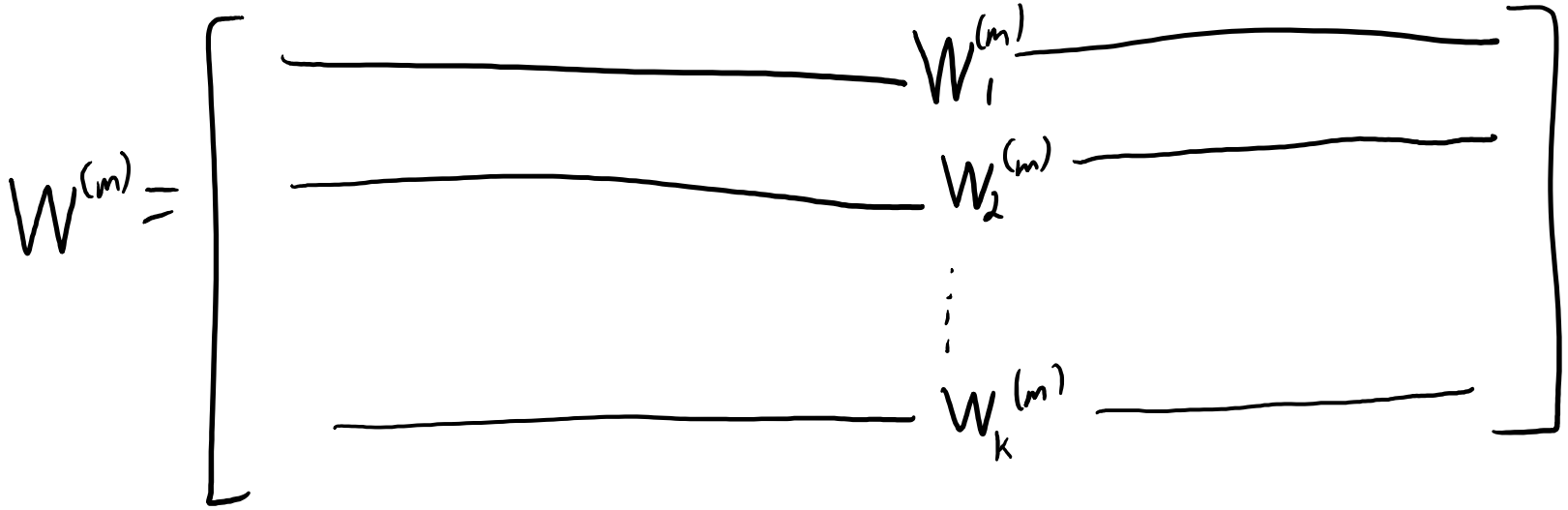


(intermission)

- That’s it for convolutions
  - There are many more bonus slides on 2-D convolutions
- We’ll now talk about Convolutional Neural Networks (CNNs)

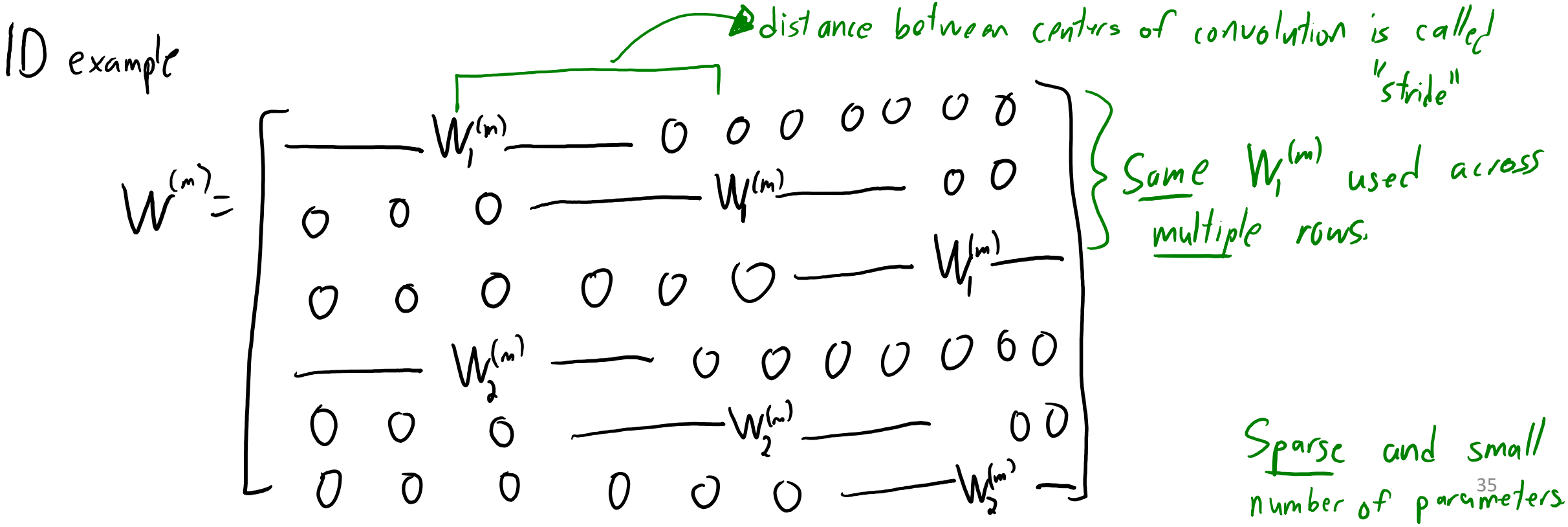
Convolutional Neural Networks

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



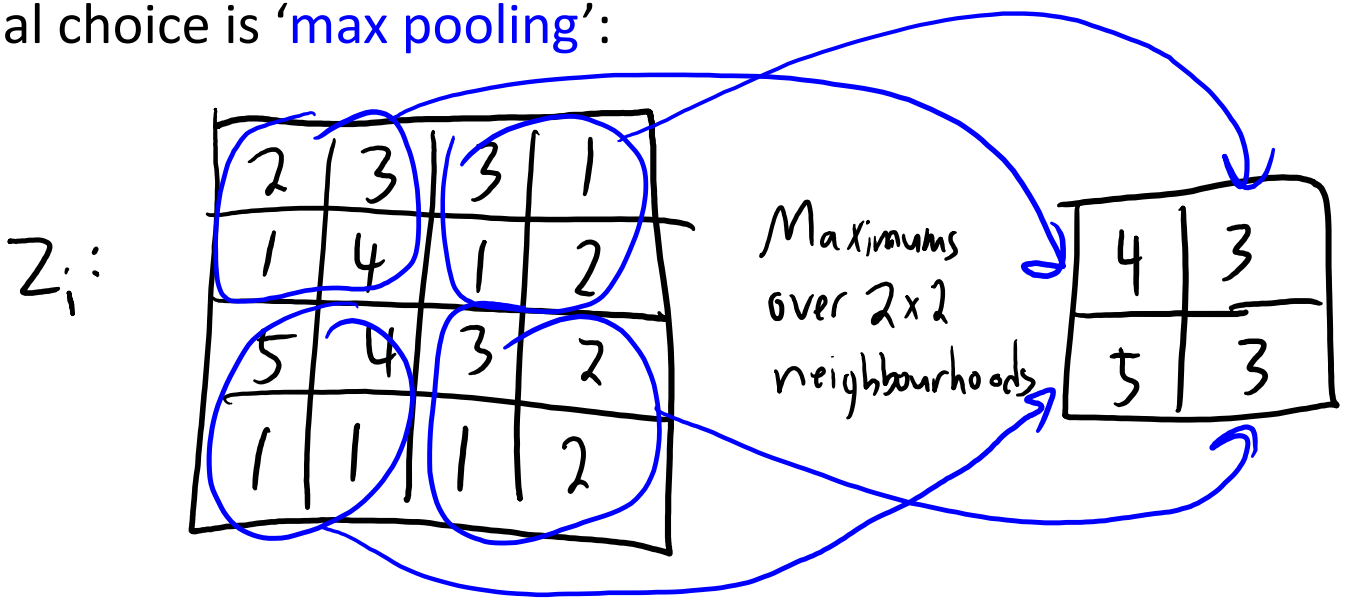
Convolutional Neural Networks

- 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.



Convolutional Neural Networks

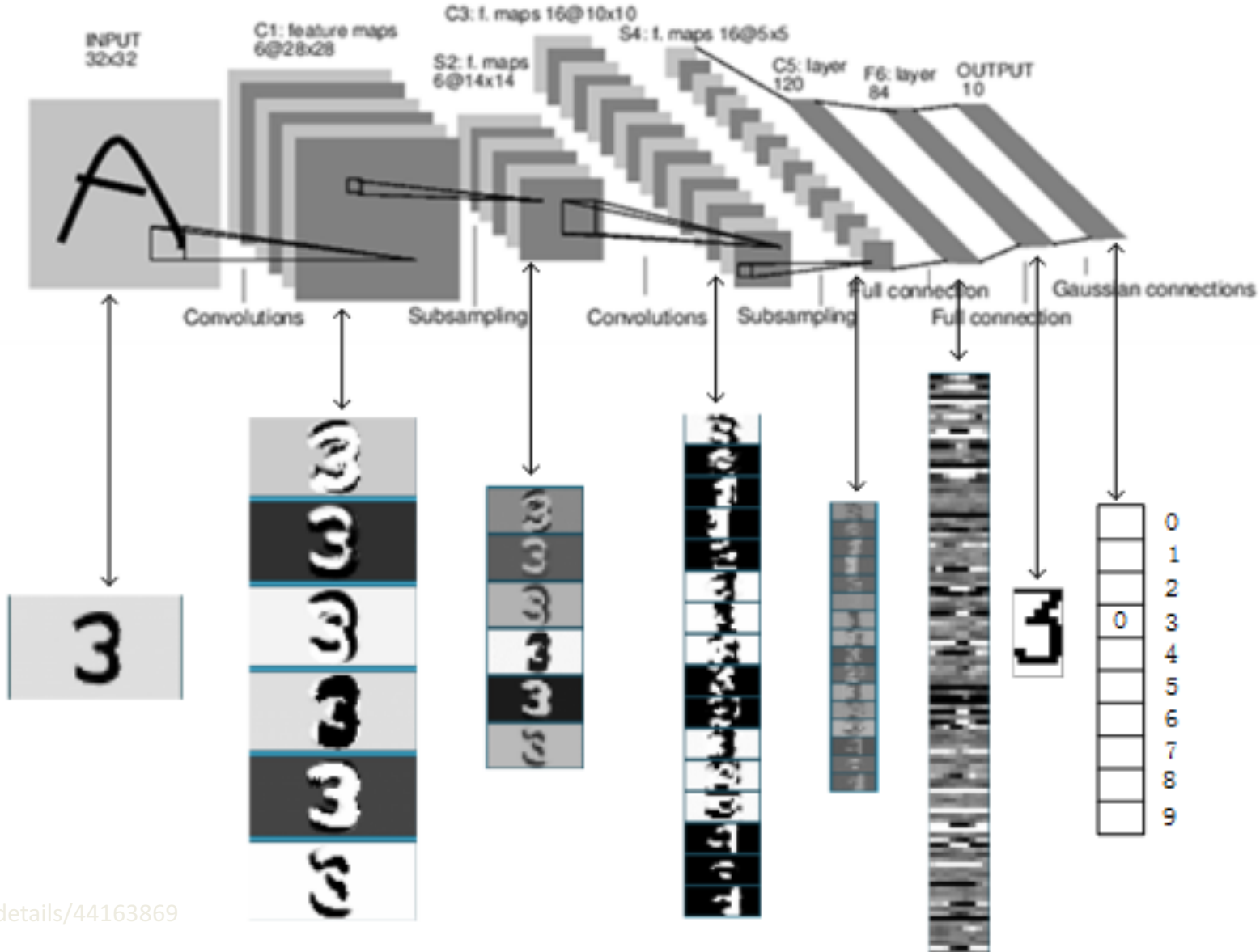
- Convolutional Neural Networks classically have 3 layer "types":
  - Fully connected layer: usual neural network layer with unrestricted W.
  - Convolutional layer: restrict W to results of several convolutions.
  - Pooling layer: downsamples result of convolution.
    - Can add invariances or just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



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.

LeNet for Optical Character Recognition



Summary

- Convolutions are flexible class of signal/image transformations.
- 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).

- Next time: modern convolutional neural networks and applications.
  - Image segmentation, depth estimation, image colorization, artistic style.

(bonus) FFT implementation of convolution

- There is another implementation of convolutions that uses the Fast Fourier Transform (FFT)
  - You FFT the image and filter, multiply elementwise, and then inverse FFT
- It has faster asymptotic running time but there are some catches
  - You need to be using periodic boundary conditions for the convolution
  - Asymptotically faster doesn't always mean faster -- it depends on the constants!
  - The gains are largest for larger filters (compared to the image size).