CPSC 340: Machine Learning and Data Mining

Convolutional Neural Networks

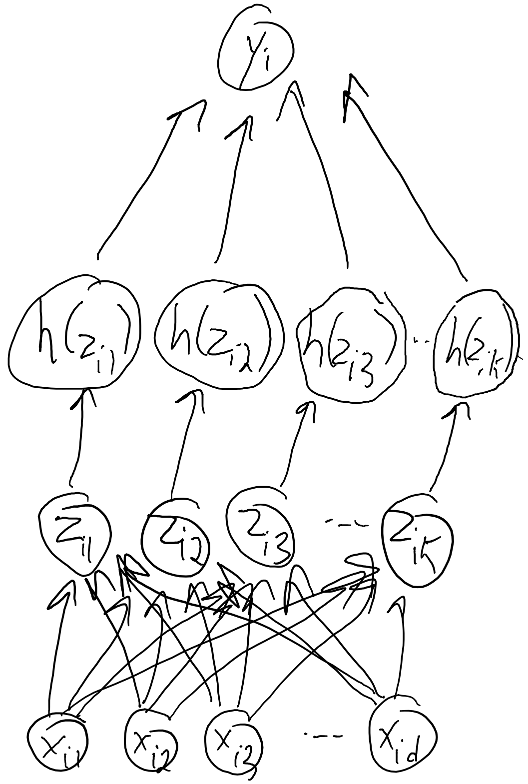
Fall 2015

Admin

- Office hours tomorrow will be in ICICS 146.
- Assignment 5:
  - Q1-2 on Piazza Saturday.
  - Full assignment coming tonight.
  - ‘Tutorial summary’ coming soon.

Last Time: Deep Learning

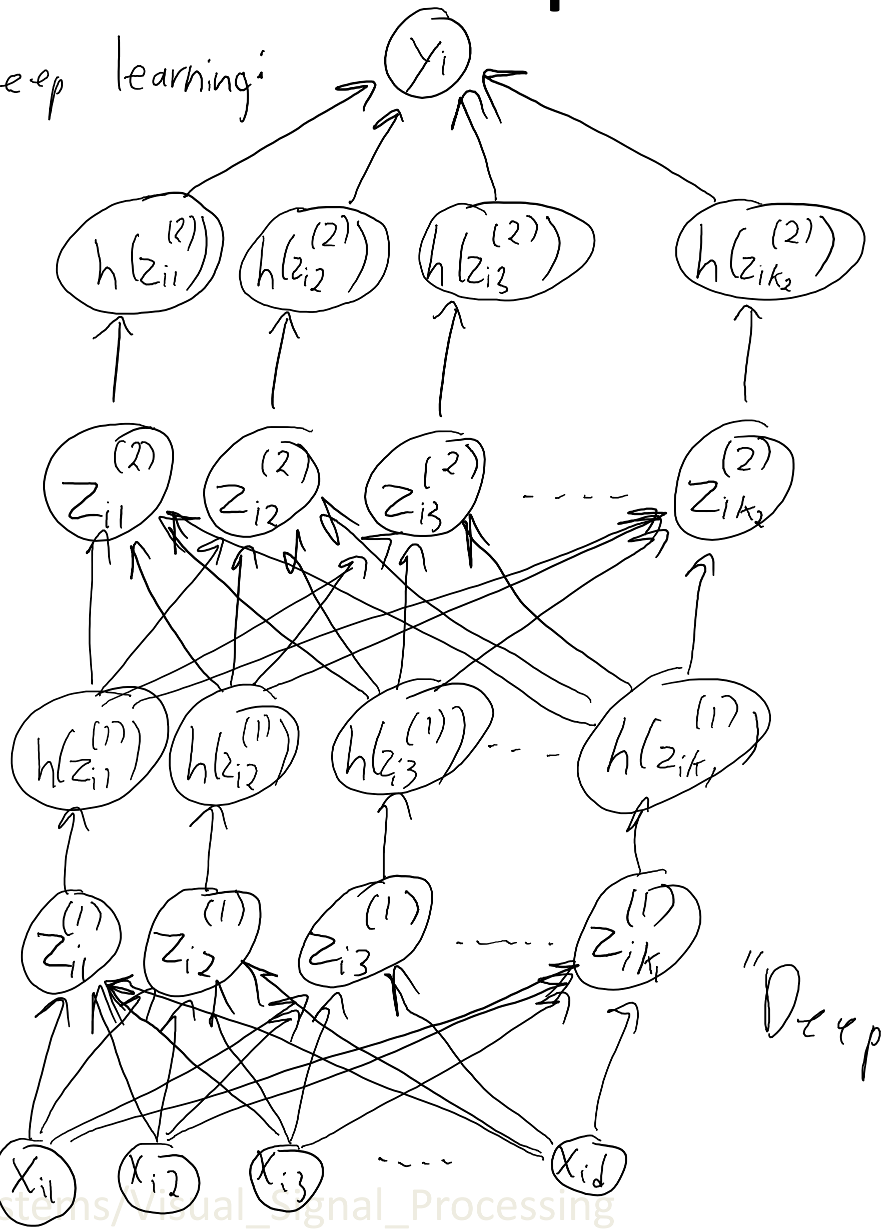
Neural networks:



Single-layer neural network:

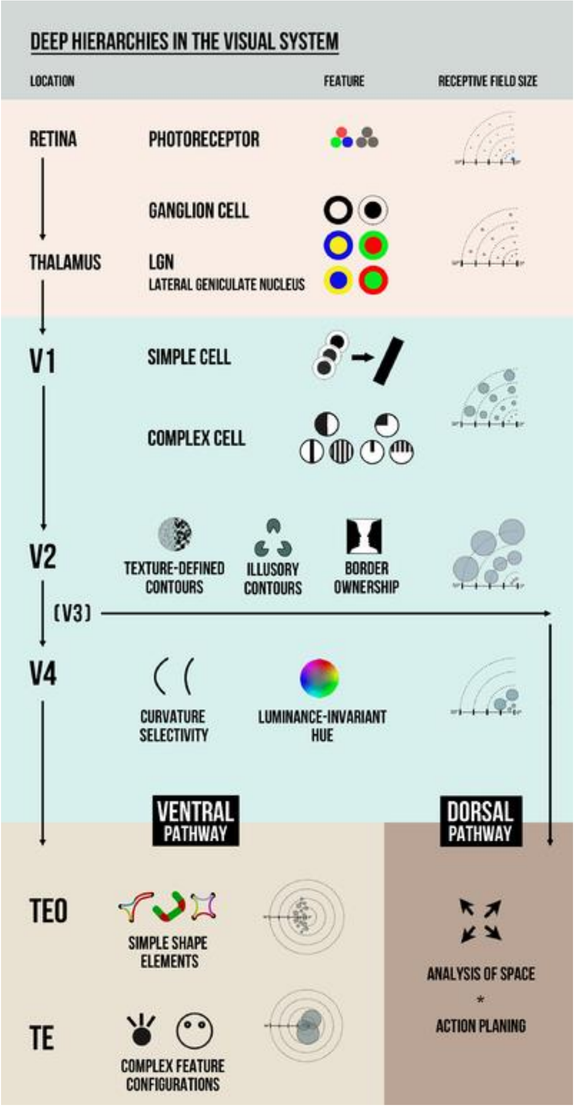
$$\hat{y}_i = w^T h(W x_i)$$

Deep learning:



"Deeper" neural network:

$$\hat{y}_i = w^T h(W_{(2)} h(W_{(1)} x_i))$$



Fitting Deep Neural Networks

- Deep neural network model:

$$\hat{y}_i = w^T h(W_{(3)} h(W_{(2)} h(W_{(1)} x_i)))$$

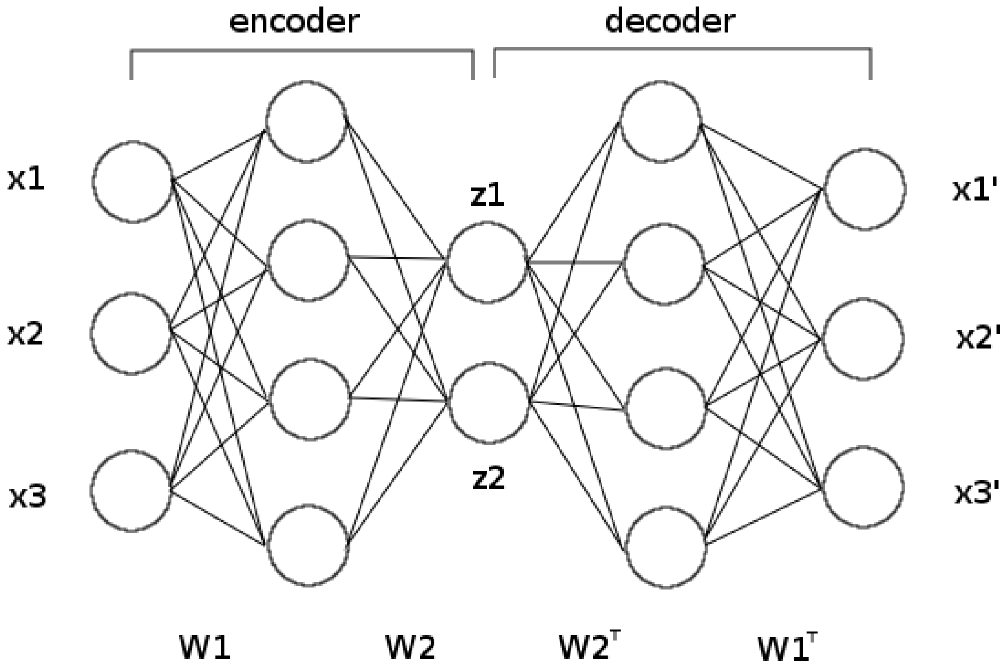
- Highly non-convex in the parameter $W_{(1)}$, $W_{(2)}$, $W_{(3)}$, w.
- We discussed a few tricks for training deep neural networks:
  - Replacing sigmoids with alternatives like logistic loss.
  - Careful selection of stochastic gradient step size (manual or automatic).
  - Momentum.
- Today:
  - Parameter initialization.
  - What happened to the fundamental trade-off?

Parameter Initialization

- Parameter initialization is crucial:
  - Can’t initialize weights in same layer to same value, or they will stay same.
  - Can’t initialize weights too large, it will take too long to learn.
- Random initialization:
  - Set bias variables to 0.
  - Uniformly sample from standard normal, divided by 10,000 (0.00001*randn).
  - Performing multiple initializations does not seem to be important.
- More recent:
  - Use different initialization in each layer.
  - Try to make variance the same across layers.
  - Use sample from standard normal, divide by sqrt(2*nInputs).
- Another strategy is to use a deep unsupervised model to initialize.

Autoencoders

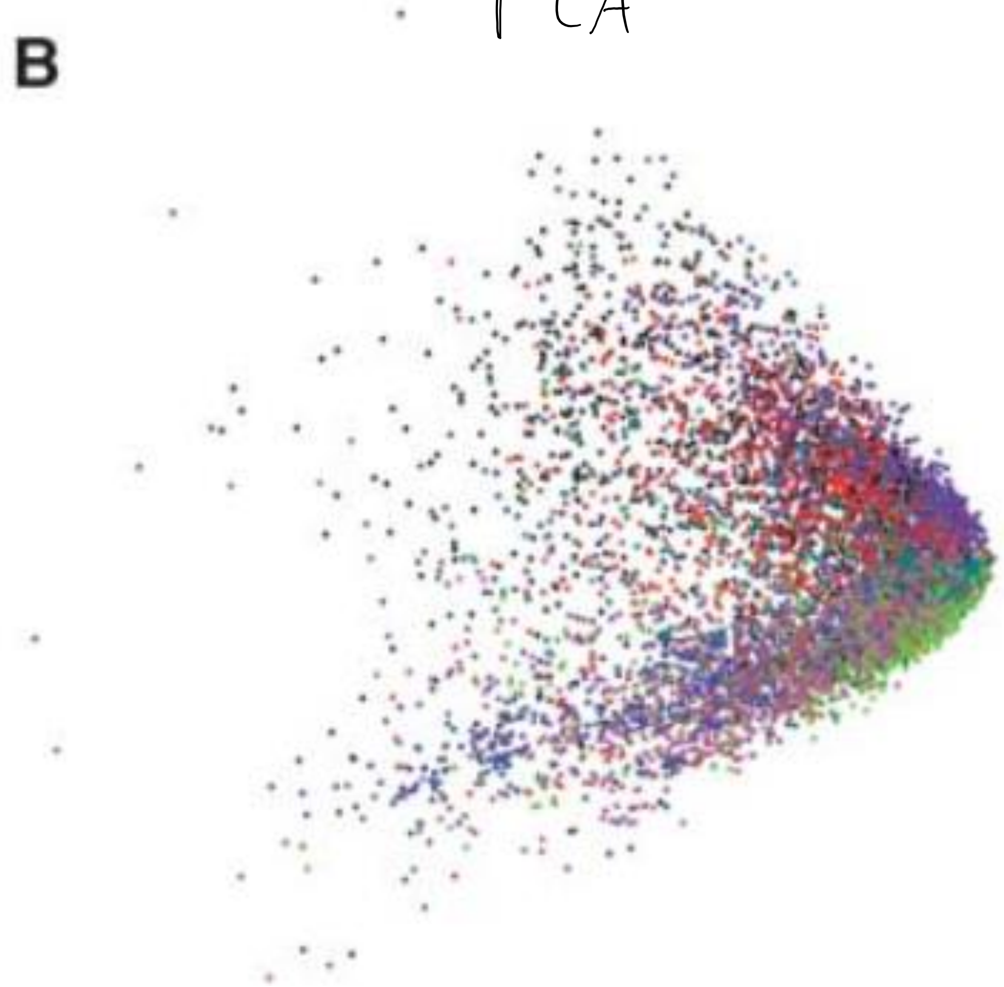
- Autoencoders are an unsupervised deep learning model:
  - Use the inputs as the output of the neural network.



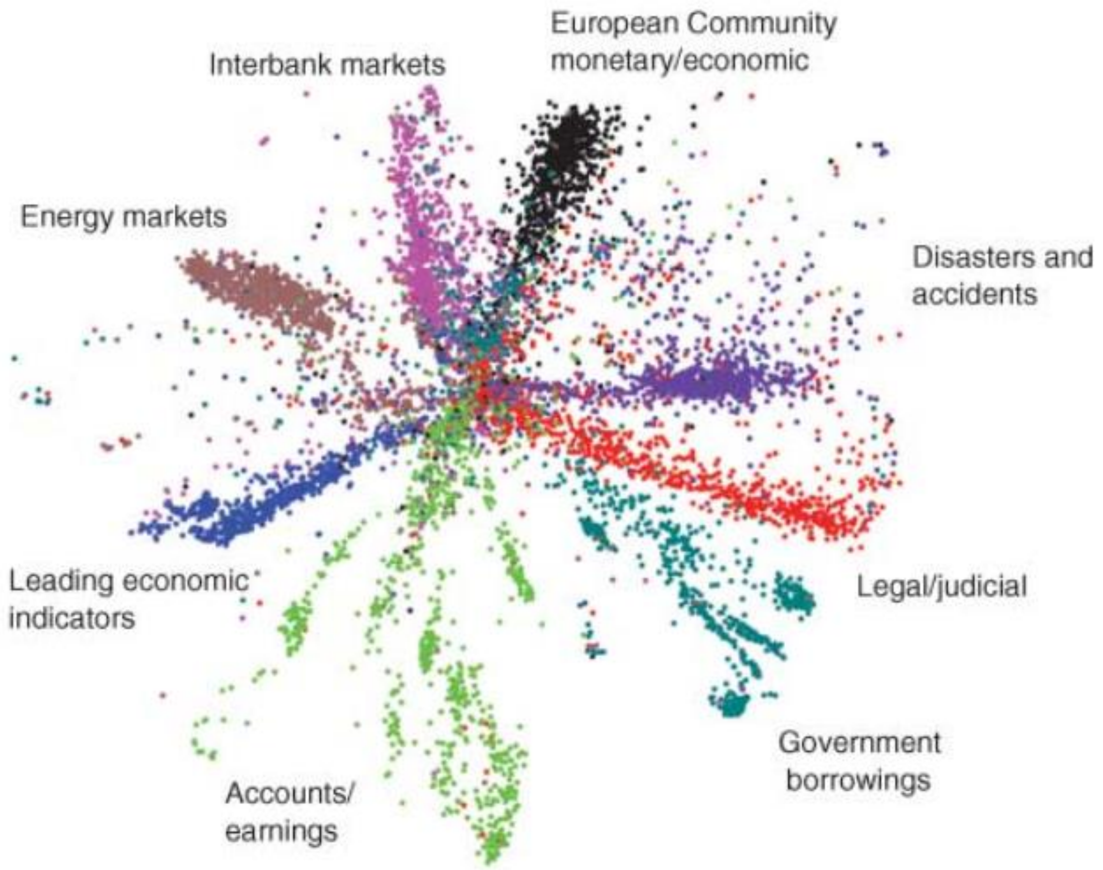
  - If middle layer has only 2 units, can use this for visualization.
  - Common to add noise to inputs ('denoising' autoencoder).

Autoencoders

PCA



Autoencoder



https://www.cs.toronto.edu/~hinton/science.pdf

Deep Learning and the Fundamental Trade-Off

- Neural networks are subject to the fundamental trade-off:
  – As we increase the depth, training error decreases.
  – As we increase the depth, training error no longer approximates test error.
- We want deep networks to model high non-linear data.
- But increasing the depth leads to overfitting.
- How could systems like GoogLeNet using ~30 layers?
  – Many forms of regularization and keeping model complexity under control.

Standard Regularization

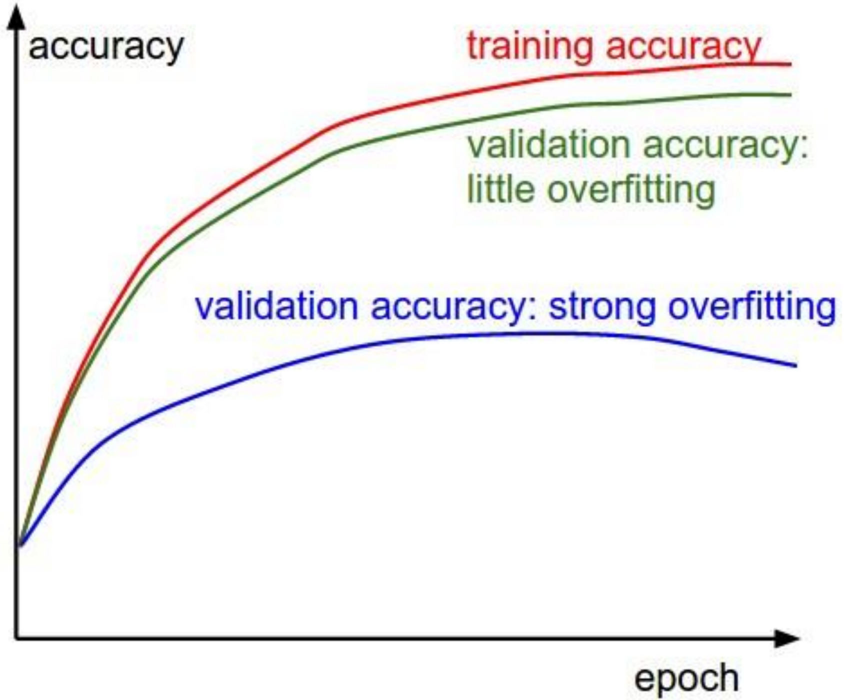
- We typically add our usual L2-regularizers:

$$\underset{w, W_{(1)}, W_{(2)}, W_{(3)}}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}\left(y_i - w^T h(W_{(3)} h(W_{(2)} h(W_{(1)} x_i)))\right)^2 + \frac{\lambda_4}{2}\|w\|^2 + \frac{\lambda_3}{2}\|W_{(3)}\|_F^2 + \frac{\lambda_2}{2}\|W_{(2)}\|_F^2 + \frac{\lambda_1}{2}\|W_{(1)}\|_F^2$$

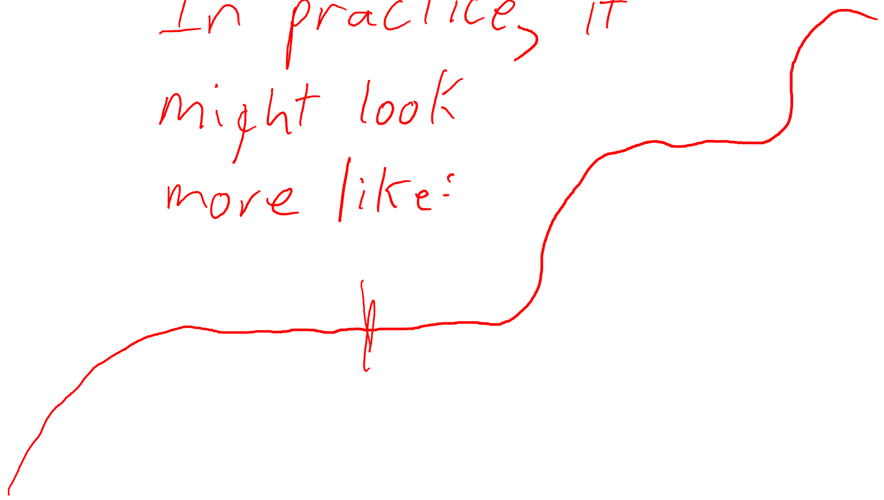
- Called 'weight decay' in neural network literature.
- Could also use L1-regularization.
- 'Hyper-parameter' optimization:
  - Try to optimize validation error in terms of λ_1, λ_2, λ_3, λ_4.
- Unlike linear models, this is rarely the only form of regularization.

Early Stopping

- Even with regularization, stochastic gradient may still overfit.
- Regularization by 'early stopping':
  - Monitor the validation error as we run stochastic gradient.
  - Stop the algorithm if validation error starts increasing.

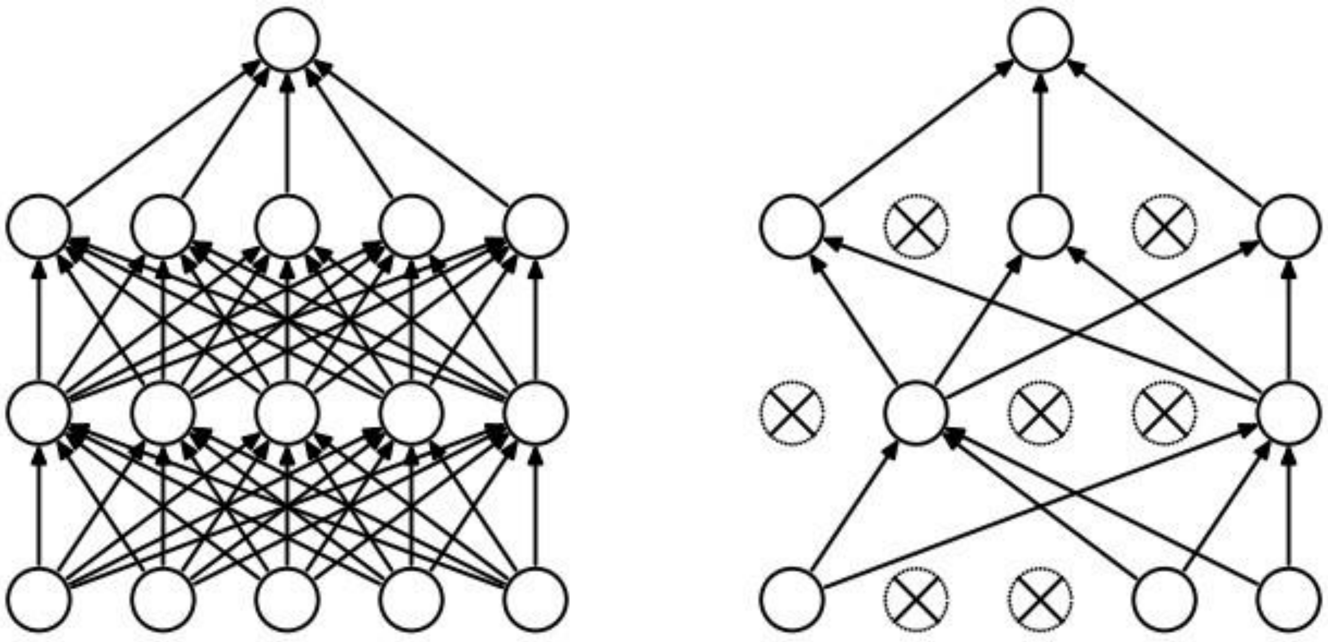


In practice, it might look more like:



Dropout

- Dropout is a third form of regularization:
  - On each iteration, randomly set some x_i and z_i to zero (often use 50%).



(a) Standard Neural Net (b) After applying dropout.

  - Encourages distributed representation rather than using specific z_i.

http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Convolutional Neural Networks

- Typically use multiple types of regularization:
  - L2-regularization.
  - Early stopping.
  - Dropout.
- Often, still not enough to get deep models working.
- Deep models most used are convolutional neural networks:
  - Place heavy restrictions on the elements of each $W_{(m)}$.
  - Sizes of $z_i^{(m)}$ and functions 'h' change at each level.

Discrete Convolution

- Given ‘n’ values ‘x’ with indices j=1,2,…,n.
- We define weights ‘w’ with indices j=-m,-m+1,…-2,0,1,2,…,m-1,m.
- The discrete convolution ‘*’ of ‘x’ with ‘g’ at ‘i’ given by

$$(x * w)[i] = \sum_{j=-m}^{m} x_{i+j} w_j$$

- This is an inner product between ‘w’ and part of ‘x’:

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

Discrete Convolution Example

- Given 'n' values 'x' with indices j=1,2,…,n.
- We define weights 'w' with indices j=-m,-m+1,…-2,0,1,2,…,m-1,m.
- The discrete convolution '*' of 'x' with 'g' at 'i' given by

$$(x * w)[i] = \sum_{j=-m}^{m} x_{i+j} w_j$$

- This is an inner product between 'w' and part of 'x':

$x = [1 \quad 2 \quad 3 \quad 5 \quad 7 \quad 11]$

- For example:

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

$\quad\;\; -1 \quad\; 0 \quad +1$

average of f(i), f(i-1), and f(i+1).

$(x * w)[2] = 1(\tfrac{1}{3}) + 2(\tfrac{1}{3}) + 3(\tfrac{1}{3})$

$= 2$

$(x * w)[5] = 5(\tfrac{1}{3}) + 7(\tfrac{1}{3}) + 11(\tfrac{1}{3})$

$= 7\tfrac{2}{3}$

Discrete Convolution Example

Let $x = [1 \quad 2 \quad 3 \quad 5 \quad 7 \quad 11]$

If $w = [0 \quad 1 \quad 0]$ then $(x * w)[i]$ returns x_i

Discrete Convolution Example

Let $x = [1 \quad 2 \quad \boxed{3 \quad \underset{\uparrow \text{position } 4}{5} \quad 7} \quad 11]$

If $w = [0 \quad 1 \quad 0]$ then $(x * w)[i]$ returns x_i

$$(x * w)[4] = [3 \quad 5 \quad 7]w$$

$$= 3(0) + 5(1) + 7(0)$$

$$= 5$$

$$= x_5$$

"Identity convolution"

Discrete Convolution Example

Let $x = [1 \quad 2 \quad 3 \quad 5 \quad 7 \quad 11]$

position 5

If $w = [1 \quad 0 \quad 0]$ then $(x * w)[i]$ returns x_{i-1}

$$(x * w)[5] = [5 \quad 7 \quad 11]w$$
$$= 5(1) + 7(0) + 11(0)$$
$$= 5$$

Discrete Convolution Example

Let $x = [1 \quad 2 \quad 3 \quad 5 \quad 7 \quad 11]$

position 5

If $w = [1 \quad 0 \quad 0]$ then $(x * w)[i]$ returns x_{i-1}

$$
\begin{aligned}
(x * w)[5] &= [5 \quad 7 \quad 11]\,w \\
&= 5(1) + 7(0) + 11(0) \\
&= 5
\end{aligned}
$$

If we apply it to each position, we get a translated signal:

$$
\begin{aligned}
x &= [1 \quad 2 \quad 3 \quad 5 \quad 7 \quad 11] \\
(x * w) &= [? \quad 1 \quad 2 \quad 3 \quad 5 \quad 7]
\end{aligned}
$$

Discrete Convolution Example

Let $x = [1 \quad 2 \quad 3 \quad 5 \quad 7 \quad 11]$

If $w = [\tfrac{1}{3} \quad \tfrac{1}{3} \quad \tfrac{1}{3}]$ then $(x * w)[i]$ returns <u>average</u> of x_{i-1}, x_i, and x_{i+1}.

$$(x * w)[5] = 5(\tfrac{1}{3}) + 7(\tfrac{1}{3}) + 11(\tfrac{1}{3})$$

$$= \frac{5+7+11}{3} = 7\tfrac{2}{3}$$

Applying it to all elements gives "local" average:

$$x = [1 \quad 2 \quad 3 \quad 5 \quad 7 \quad 11]$$

$$(x * w) = [? \quad 2 \quad 3\tfrac{1}{3} \quad 5 \quad 7\tfrac{2}{3} \quad ?]$$

Interpretation as Matrix Multiplication

- Convolution as inner product of with 'w' padded with zeros and 'x':

$$= \tilde{w}^T x$$

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

- Convolution for all 'i' is a matrix multiplication:

$$(x * w) = \tilde{W} x. \qquad \text{where } \tilde{W} = \begin{bmatrix} \text{———} w \text{———} & 0\,0\,0 \\ 0 \text{ ———} w \text{——} & 0\,0 \\ 0 \; 0 \text{ ———} w \text{——} & 0 \\ 0 \; 0 \; 0 \text{ ———} & w \text{——} \end{bmatrix}$$

- It is a special case of a latent-factor models (up to boundary issue).

Boundary Issue

- The boundary issue:

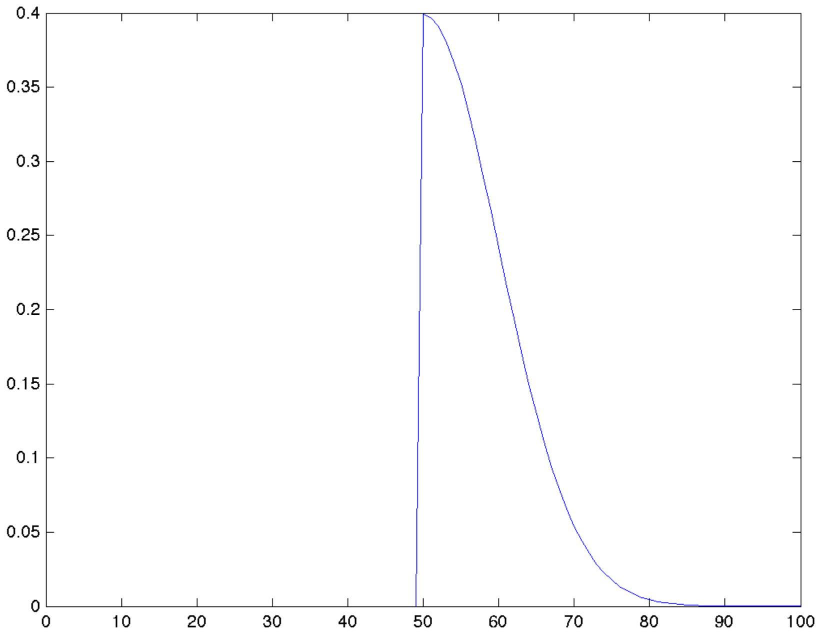
$$\tilde{w} = [\,0 \quad 0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad \overbrace{\text{———} \quad w \quad \text{———}}^{\text{positions } -m \text{ through } m}\,]$$

$$x = [\,- - - - - - - - - - - - - - - - - - -\,]???$$

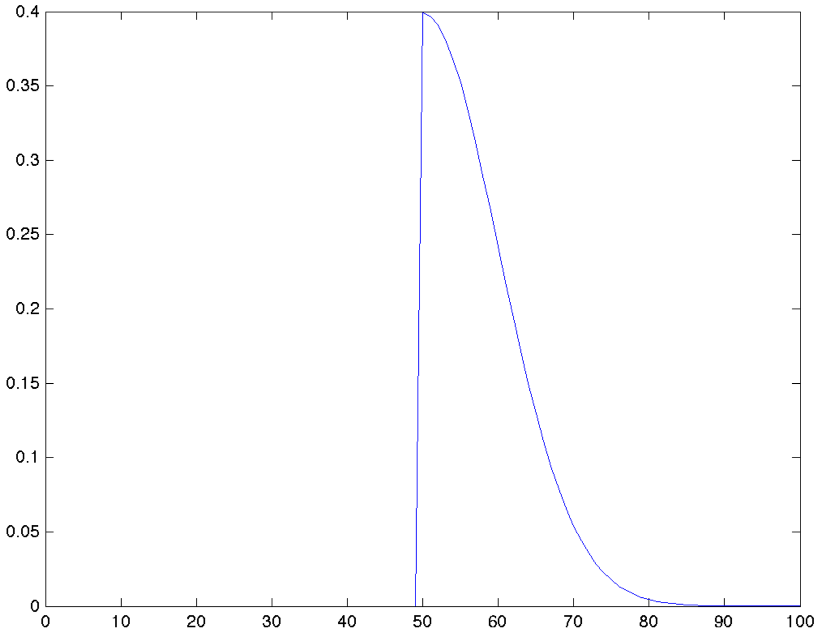
- Various ways to deal with the ‘?’ marks:
  - Don’t compute thse values.
  - Assume that they are 0.
  - ‘Replicate’ value at boundary.
  - ‘Mirror’ values at boundary.
    - $x_{n+1} = x_{n-1}$, $x_{n+2} = x_{n-2}$, etc.

Discrete Convolution Examples

- Convolutions can return the original signal:



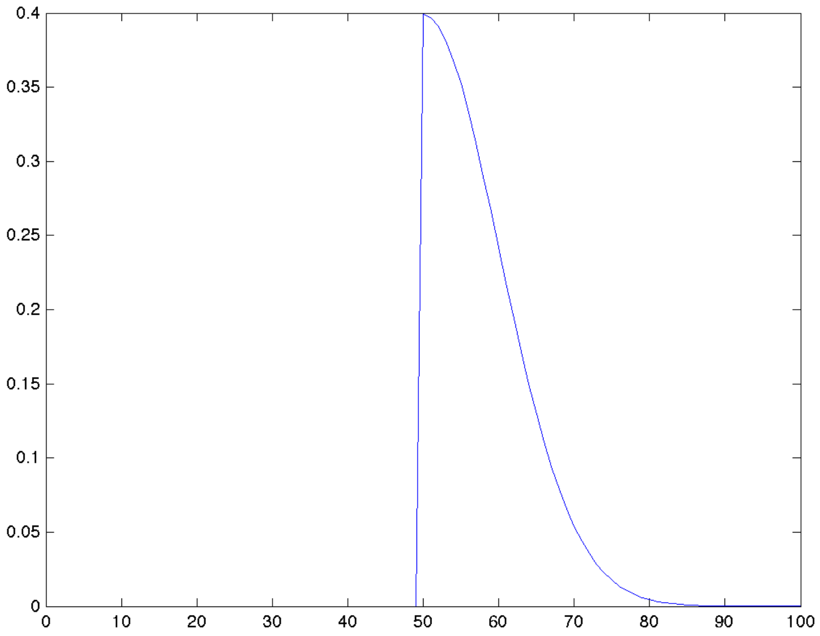
$$* \underbrace{[0 \; 0 \; 1 \; 0 \; 0]}_{w} =$$



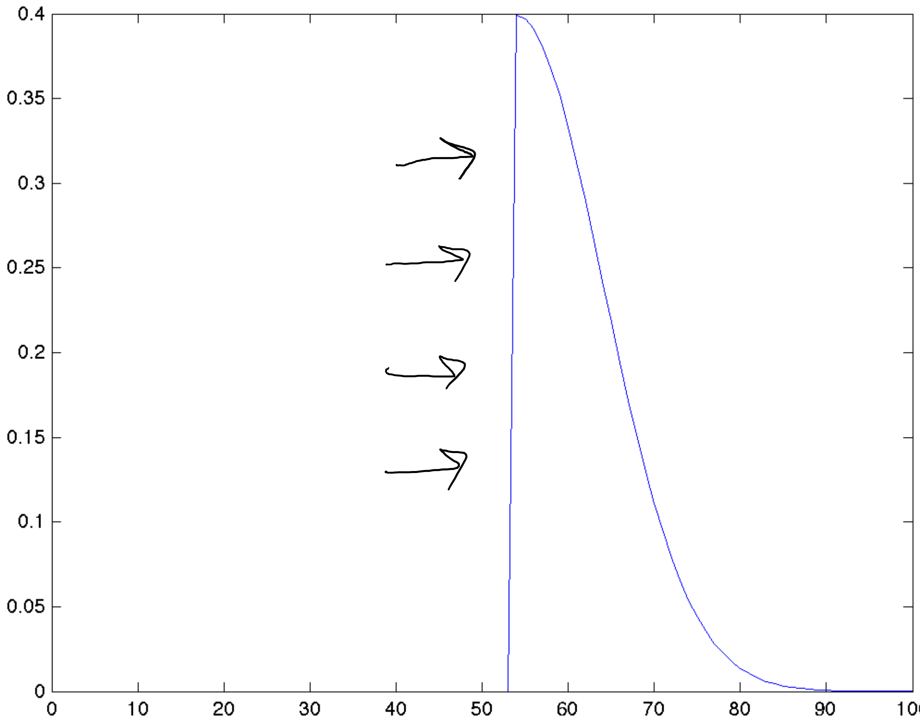
$$x = [0 \, 0 \, 0 \, 0 \, \ldots \ldots \, 0.4 \, \ldots \, 0.1 \, \ldots \ldots \, 0]$$

Discrete Convolution Examples

- Convolutions can translate the signal:

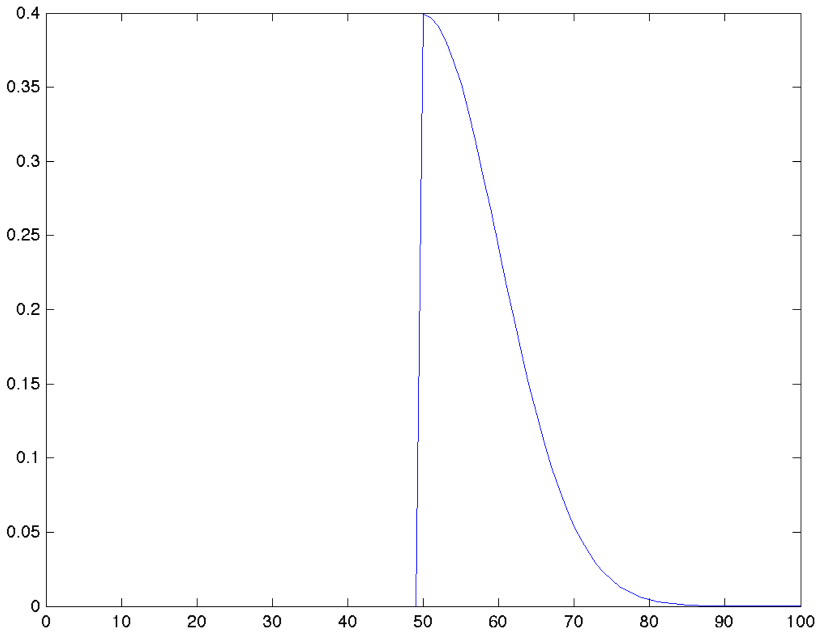


$*[1\ 0\ 0\ 0\ 0\ 0\ 0]$

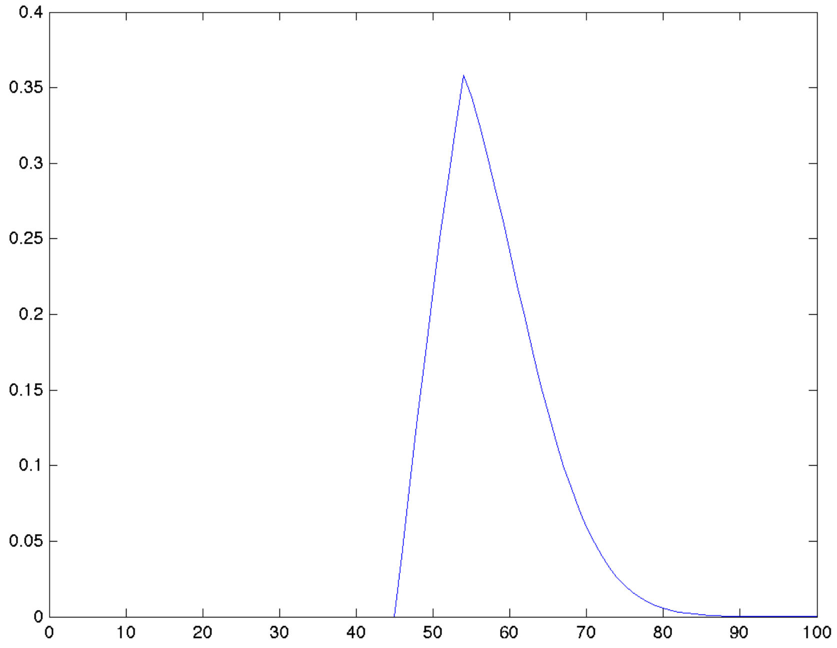


Discrete Convolution Examples

- Convolutions can locally average the signal:

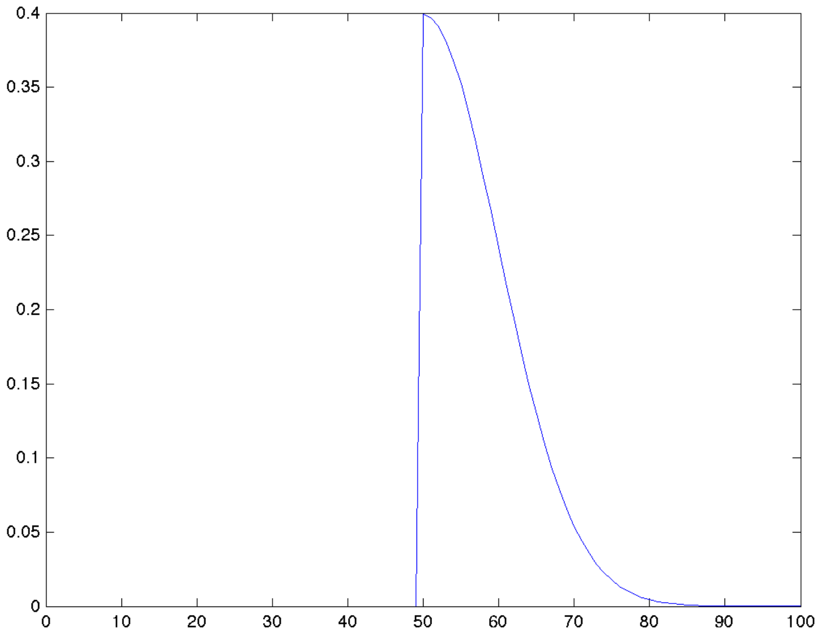


$$* \left[\tfrac{1}{9} \; \tfrac{1}{9} \; \tfrac{1}{9} \; \tfrac{1}{9} \; \tfrac{1}{9} \; \tfrac{1}{9} \; \tfrac{1}{9} \; \tfrac{1}{9} \; \tfrac{1}{9} \right] =$$

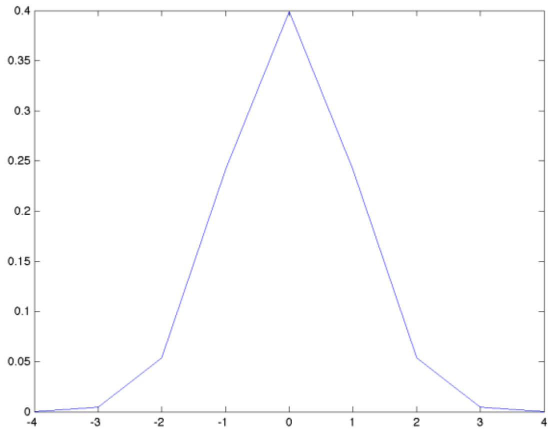


Discrete Convolution Examples

- Convolutions can smooth the signal:



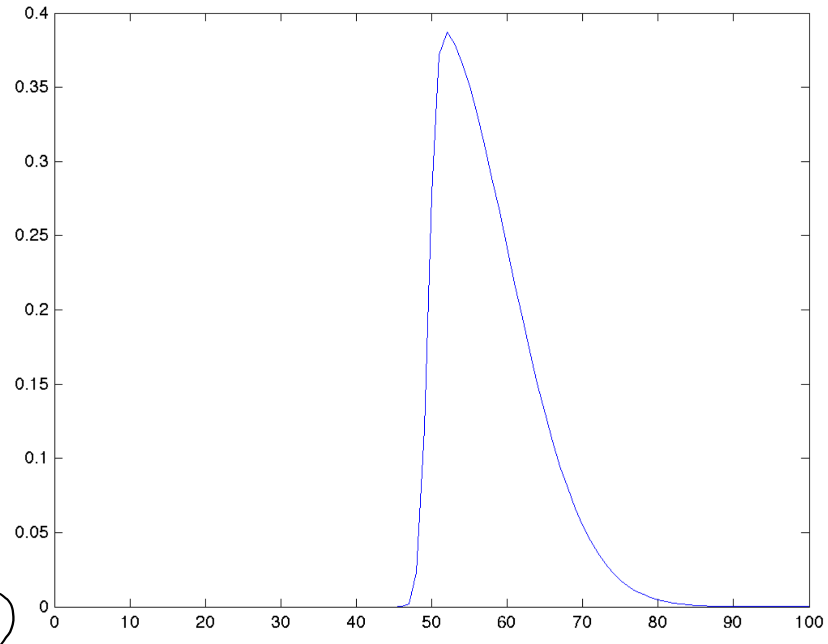
$*$



"Gaussian filter"

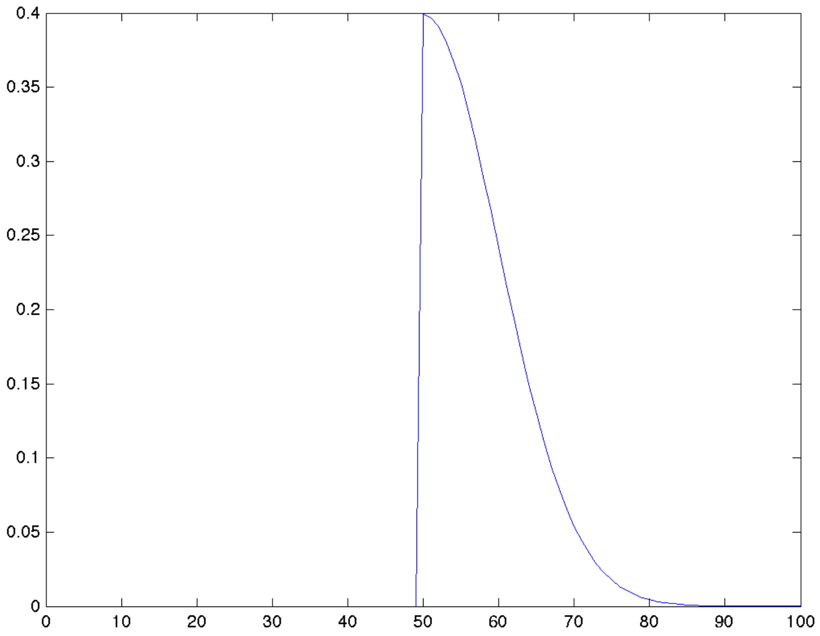
(PDF of Gaussian with mean 0)

$=$

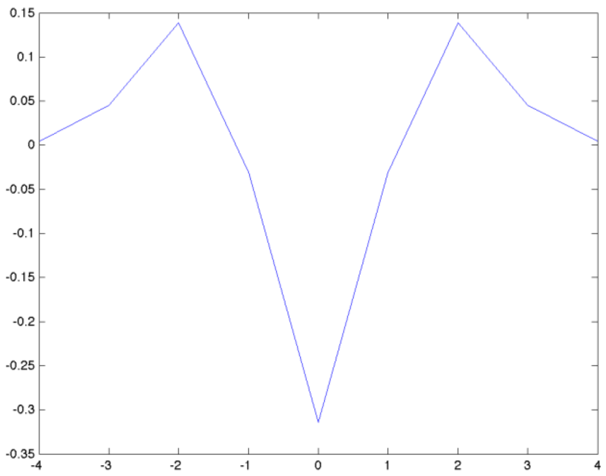


Discrete Convolution Examples

- Convolutions can detect edges in the signal:

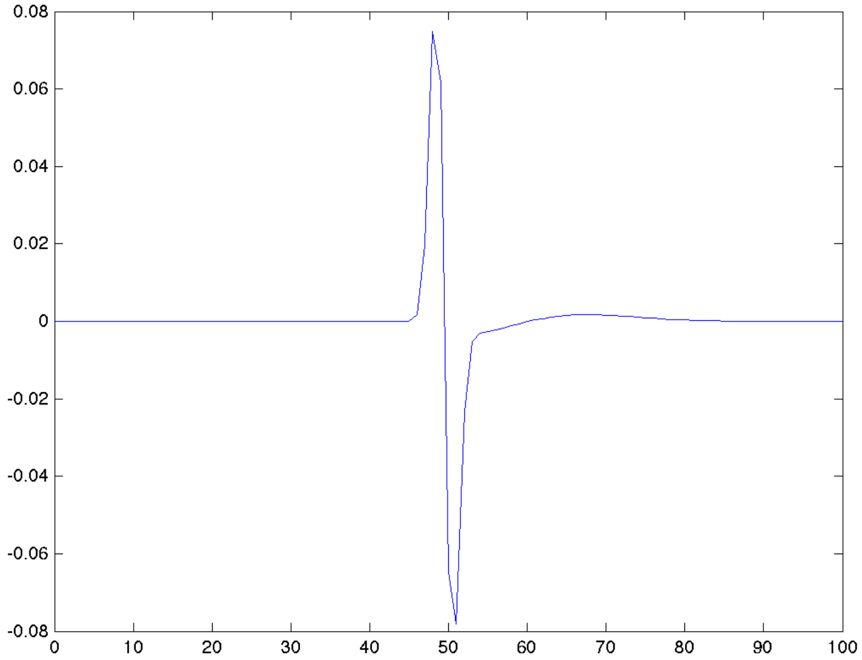


*



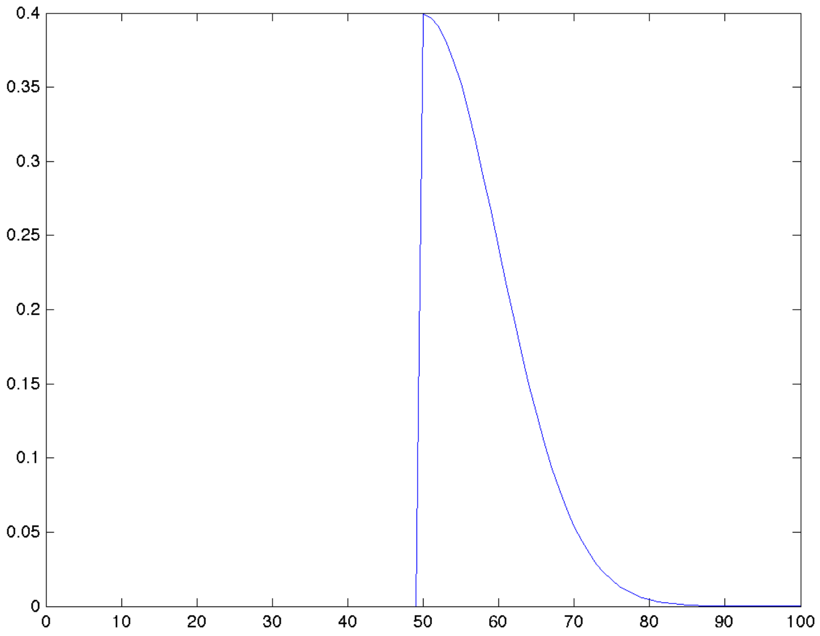
"Laplacian of Gaussian filter"
(approximation to second derivative of Gaussian)

=

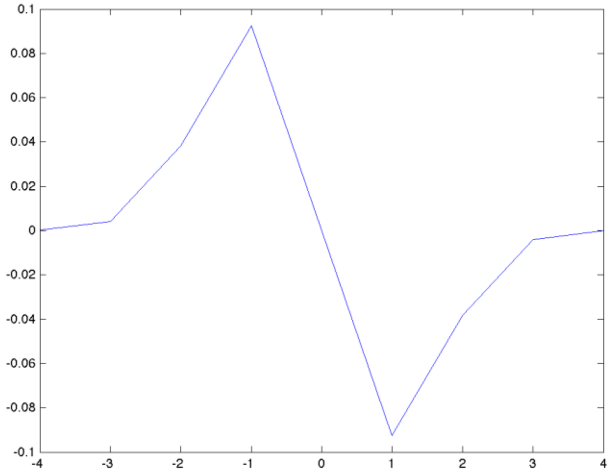


Discrete Convolution Examples

- Convolutions can detect oriented edges in the signal:



$*$



"Gabor-like filter"

Gaussian times cosine.

$=$

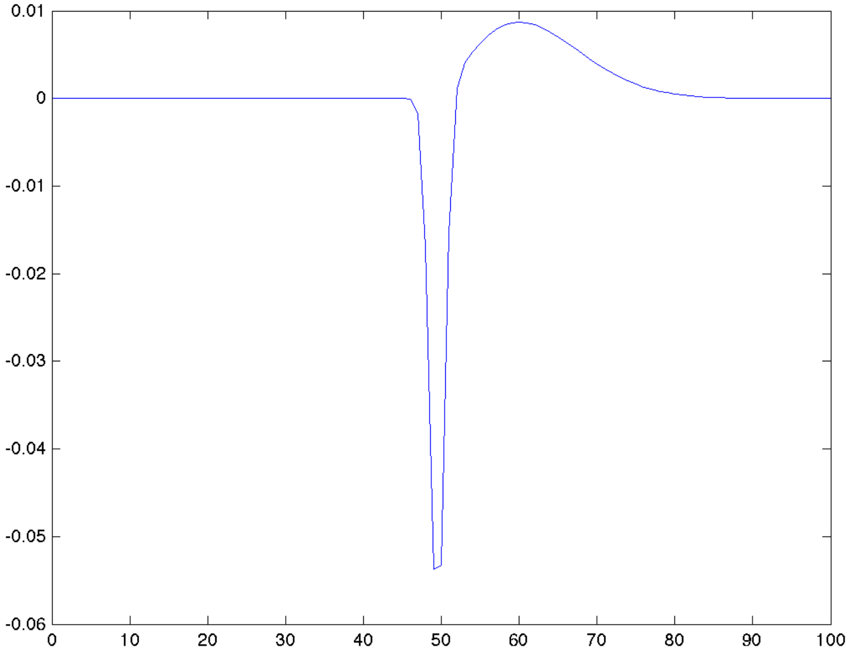


Image and Higher-Order Convolution

- Let ‘x’ be the pixel intensities in grayscale image of size ‘n’ by ‘n’.
  - Indexed 1 through n.
- Let ‘w’ be a smaller image of size ‘2m+1’ by ‘2m+1’.
  - Indexed –m through m.
- The two-dimensional convolution is given by:

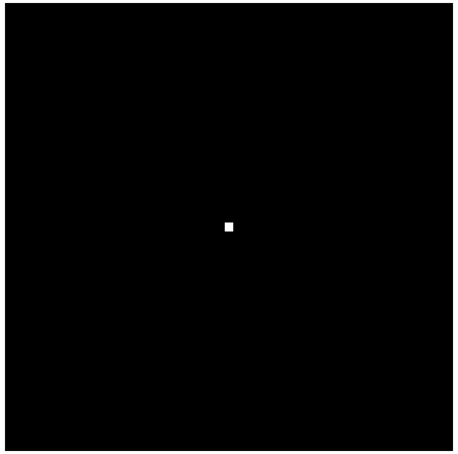
$$(x * w)(i_1, i_2) = \sum_{j_1=-m}^{m} \sum_{j_2=-m}^{m} x(i_1 + j_1, i_2 + j_2)\, w(j_1, j_2)$$

- Higher-order convolutions are defined similarly.

Image Convolution Examples

Identity convolution

*



=

multiply and add

Image Convolution Examples

Translation

*

=

Boundary: "use 0"

Image Convolution Examples

Translation

*

Boundary: "cycle"

=

Image Convolution Examples

Translation

$*$

$=$

Boundary: "replicate"

Image Convolution Examples

Translation

$*$

$=$

Boundary: "mirror"

Image Convolution Examples

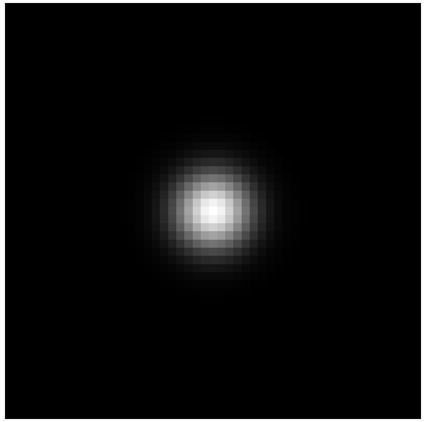
"Local average"

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

Image Convolution Examples

$*$

"Gaussian filter"

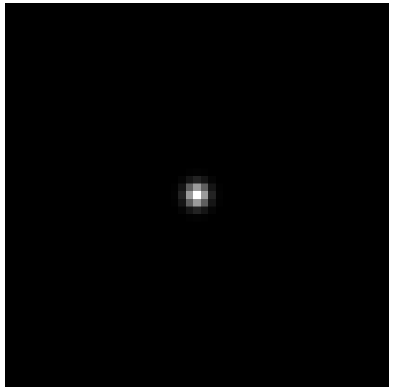


$=$

Image Convolution Examples

$*$

"Gaussian filter"



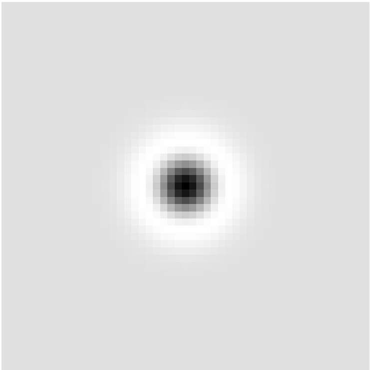
smaller variance

$=$

Image Convolution Examples

"Laplacian of Gaussian"

$*$



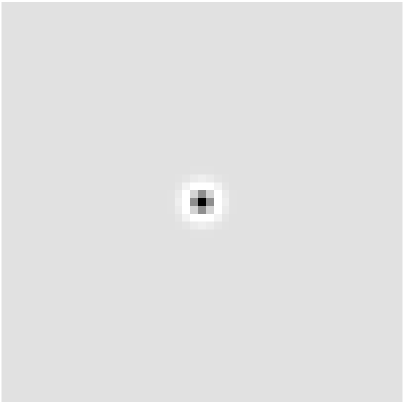
$=$

Similar preprocessing may be done in ganglia and LGN.

Image Convolution Examples

"Laplacian of Gaussian"

$*$



smaller variance

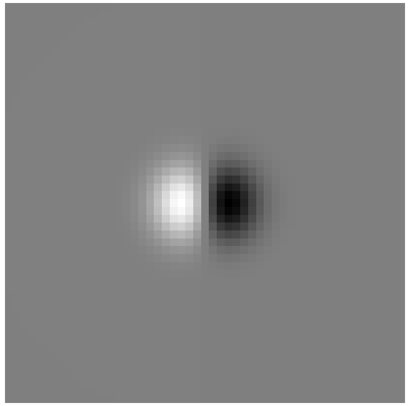
$=$

Similar preprocessing may be done in ganglia and LGN.

Image Convolution Examples

"Gabor-like filter"

$*$



$=$

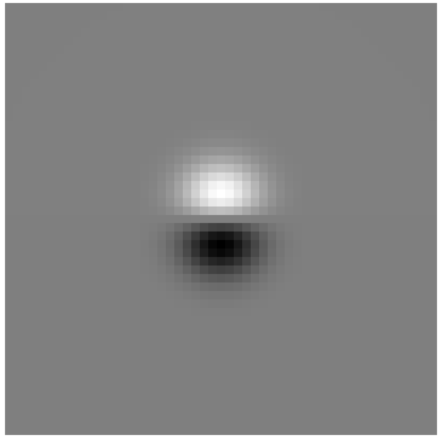
Horizontal oriented edges

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

Image Convolution Examples

"Gabor-like filter"

$*$



=

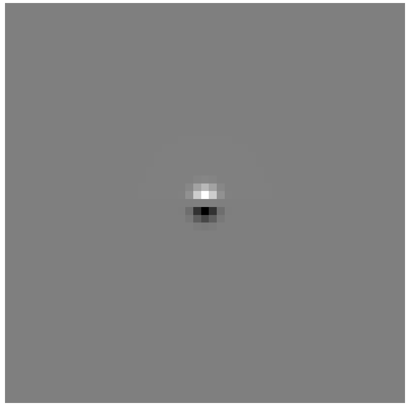
Vertical oriented edges

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

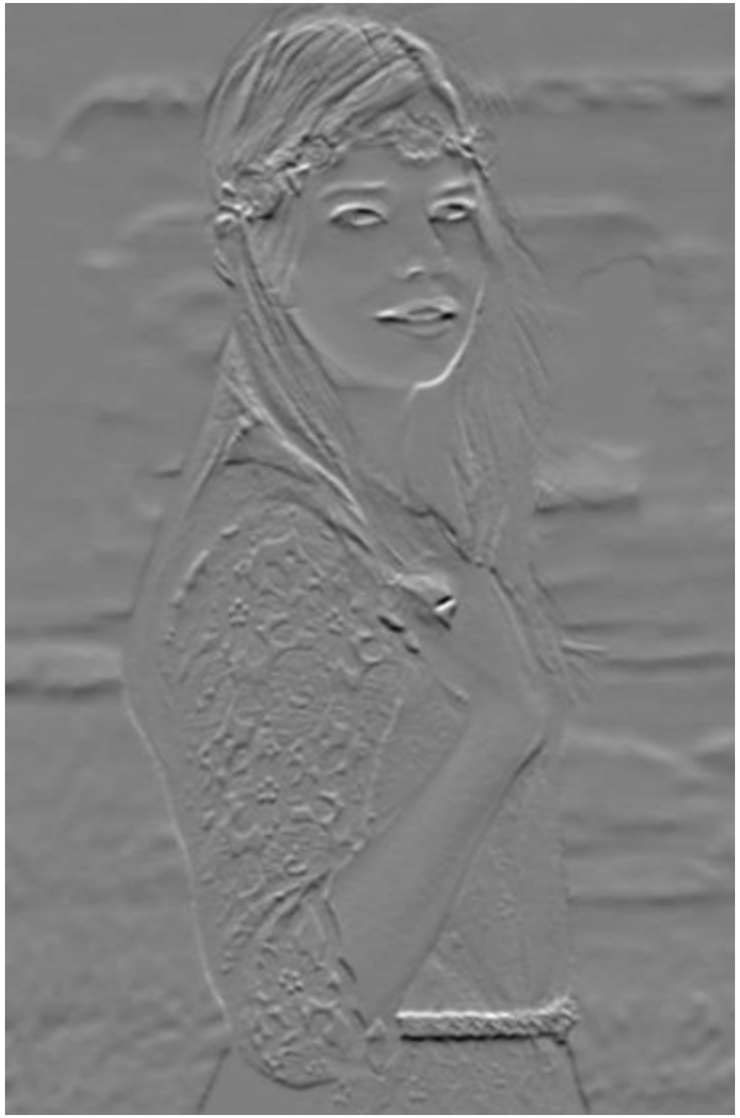
Image Convolution Examples

"Gabor-like filter"

$*$



$=$



smaller variance

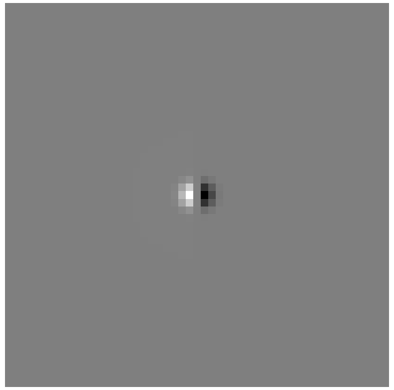
Vertical oriented edges

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

Image Convolution Examples

"Gabor-like filter"

$*$



$=$

smaller variance

Horizontal oriented edges

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

Motivation for Convolutional Neural Networks

- Consider training neural networks on 256 by 256 images.
- Each z_i in first layer has 65536 parameters (and 3x this for colour).
  - We want to avoid this huge number (avoid storage and overfitting).
- Key idea: treat Wx_i like a convolution (to make it smaller).
- Make it more like a normal image convolution:
  - Each row of W only applies to part of x_i.
  - Use the same parameters between rows.
- Same idea applies to speech, images, and maybe language.

$W_1 = [\,0 \;\; 0 \;\; 0 \;\text{——}\; w \;\text{——}\; 0 \; 0 \; 0\,]$

$W_2 = [\,0 \;\text{——}\; w \;\text{——}\; 0 \; 0 \; 0 \; 0 \; 0\,]$

Convolutional Neural Networks

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

distance between centers is called "stride"

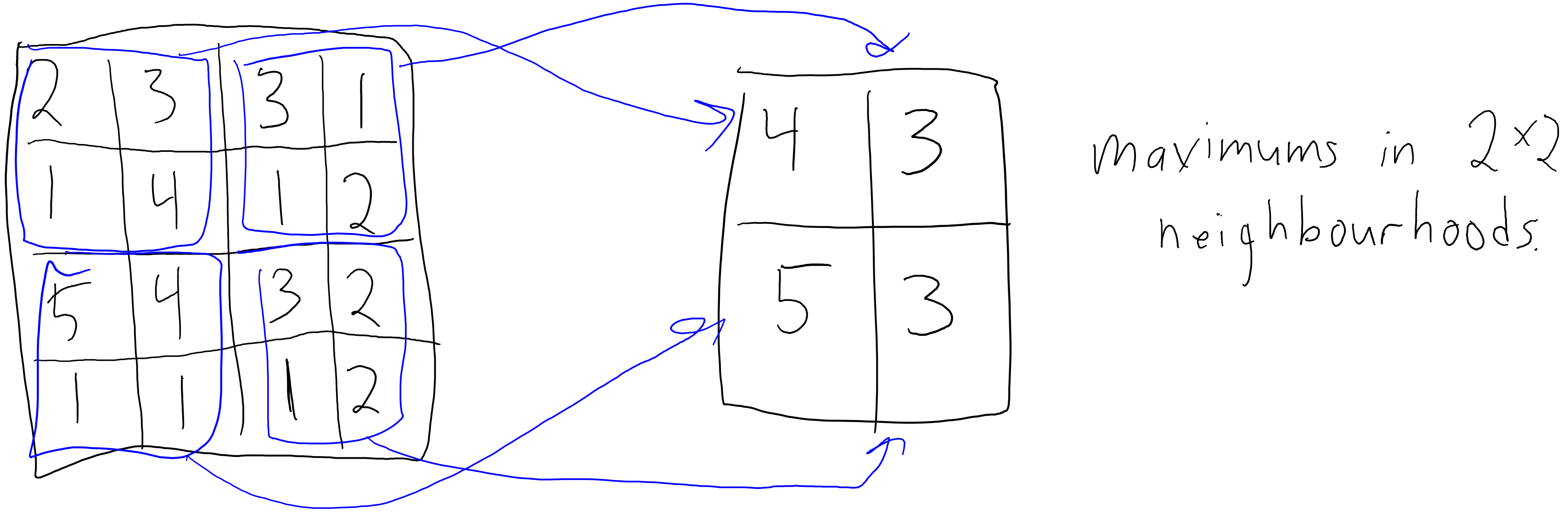
1D example:

$$W = \left[\begin{array}{ccccccccccc}
\text{---} & w_1 & \text{---} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \text{---} & w_1 & \text{---} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \text{---} & w_1 & \text{---} \\
\text{---} & w_2 & \text{---} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \text{---} & w_2 & \text{---} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \text{---} & w_2 & \text{---}
\end{array}\right]$$

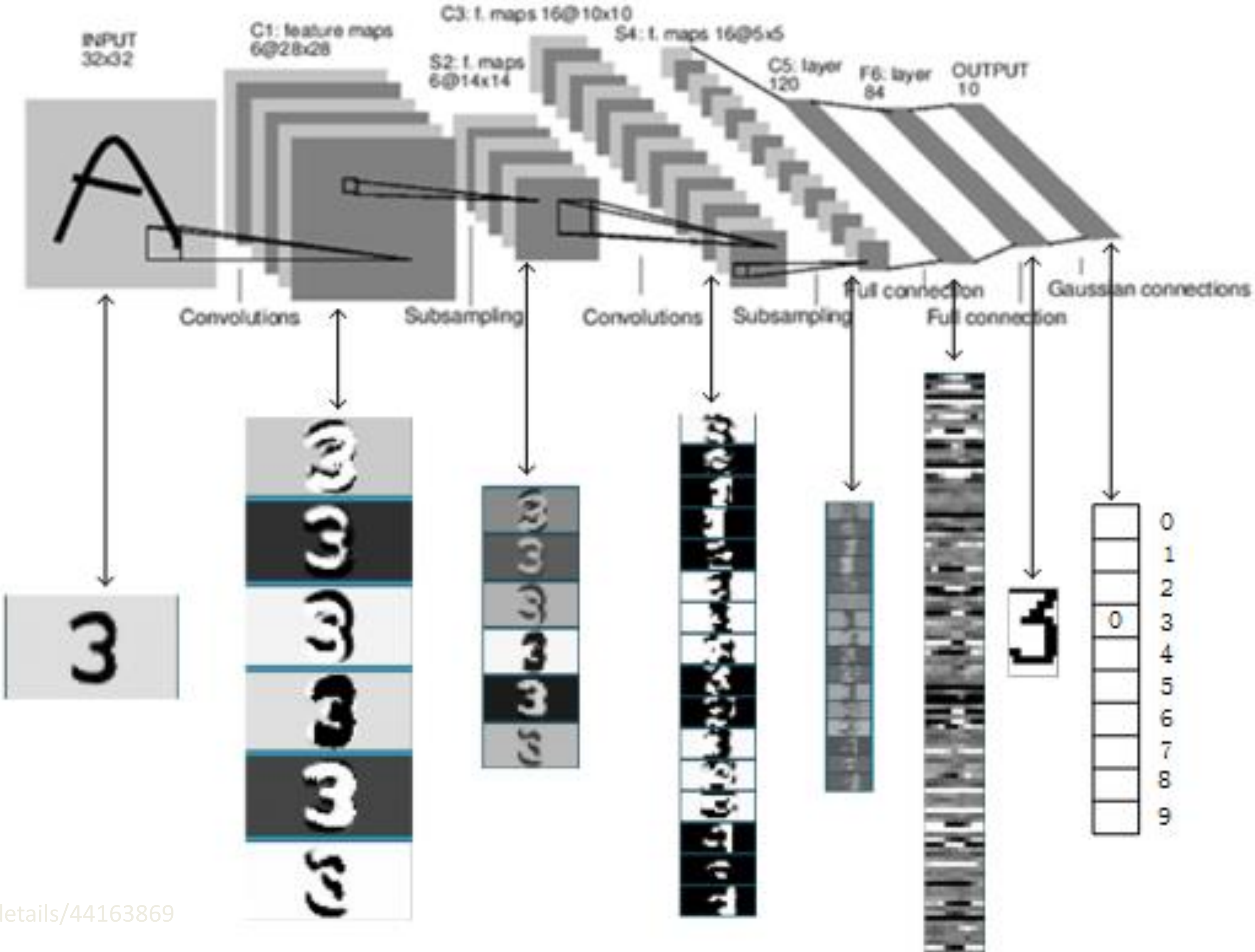
parameter w_1 is "shared" across locations

Convolutional Neural Networks

- Convolutional Neural Networks are neural with 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 to make results smaller.
    - Usual choice is 'max pooling':



LeNet for Optical Character Recognition



AlexNet (ImageNet Winner in 2011)

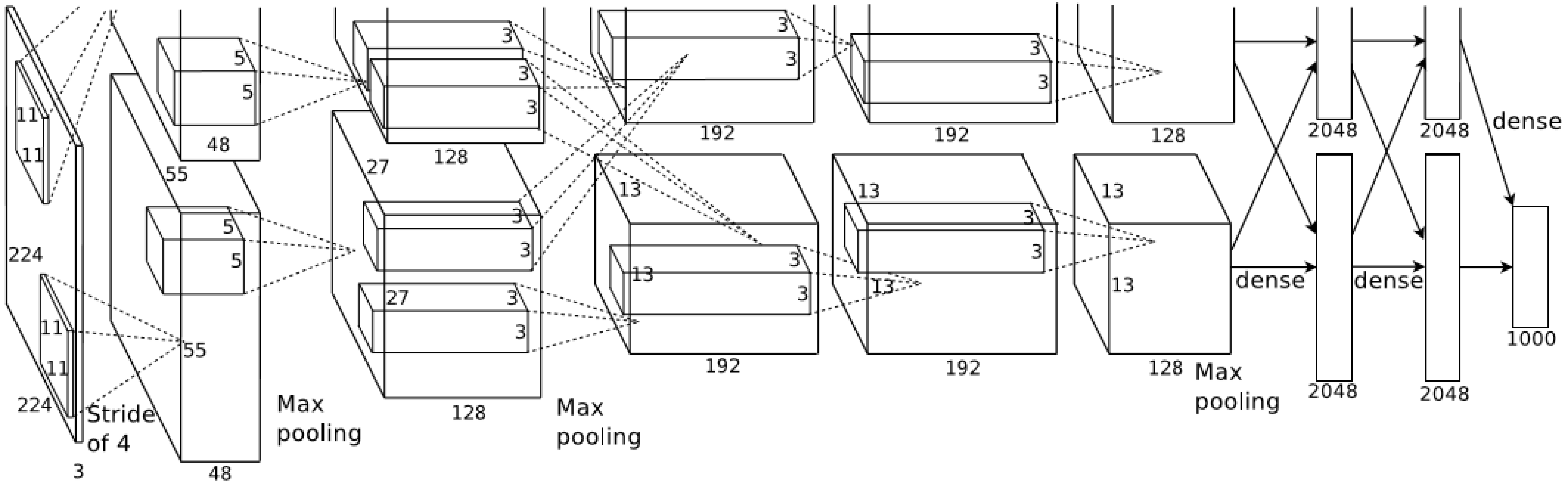


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

AlexNet (ImageNet Winne

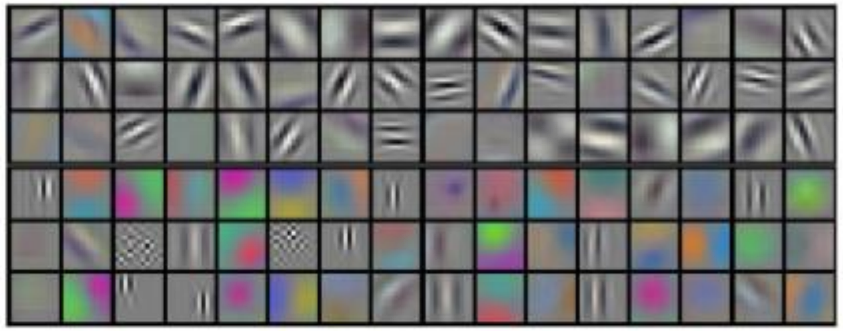


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The

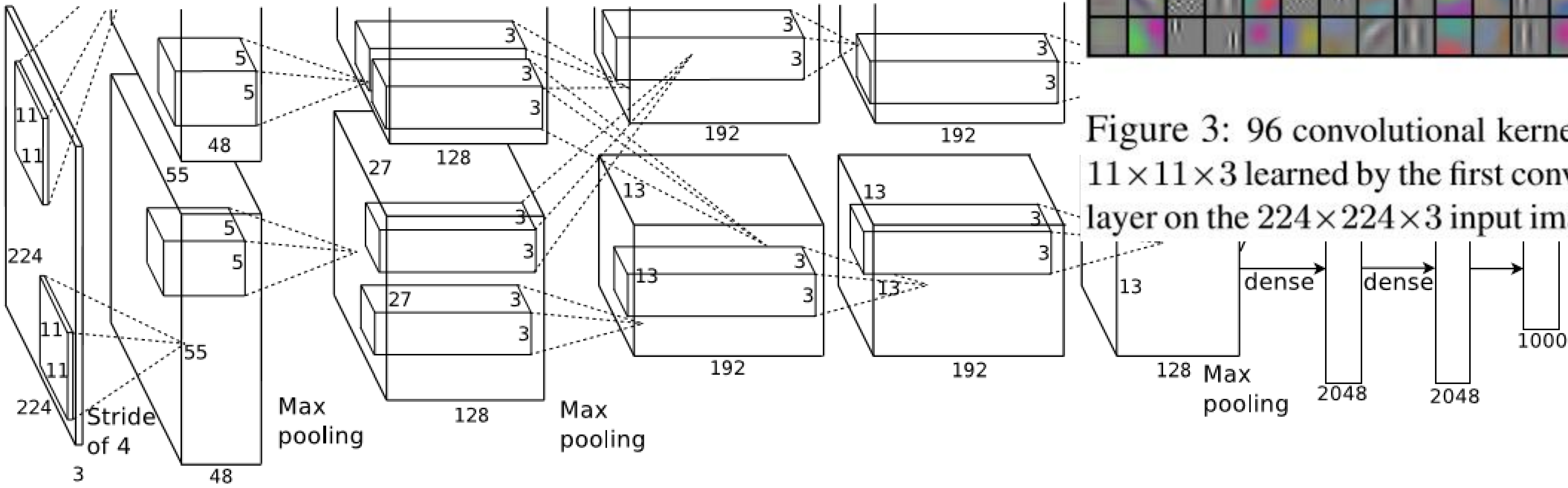
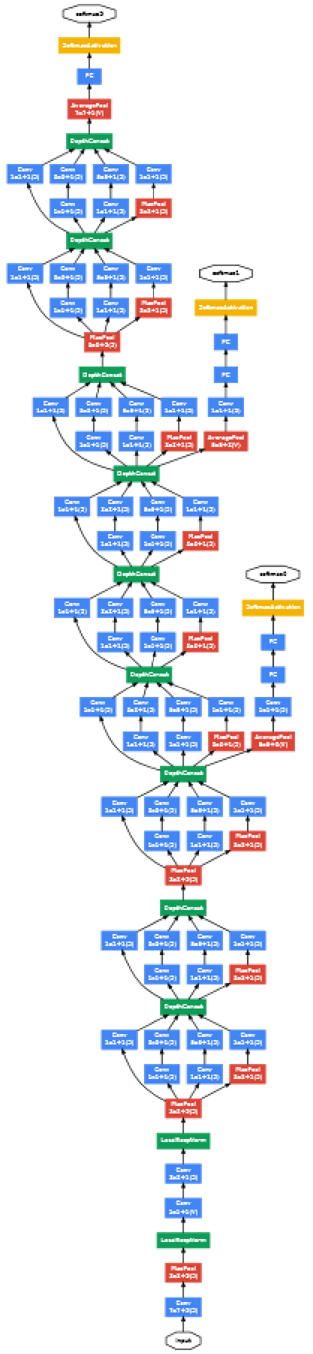


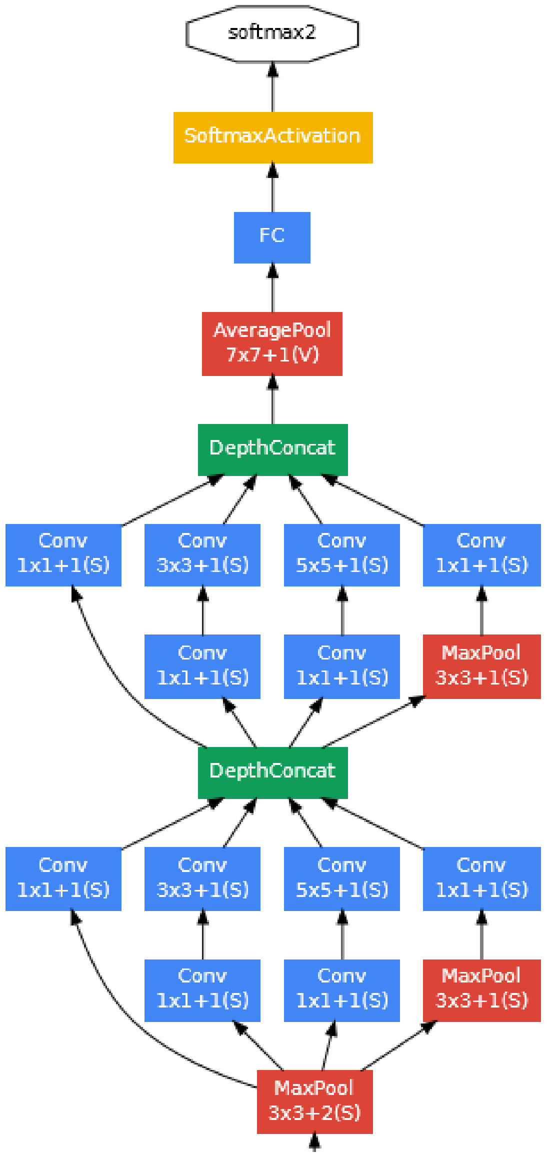
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

GoogLeNet (2014 Winner)



During training, use loss that depends on prediction at 3 depths.

At test time, only use deepest prediction.



http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Summary

- Parameter initialization is crucial to neural network performance.
- Autoencoders perform dimensionality reduction with neural nets.
- Regularization is crucial to neural net performance:
  - Usual L2, early stopping, dropout.
- Convolutions are flexible class of signal/image transformations.
- Convolutional neural networks are key in deep learning success.

- Next time: what if the output is not continuous/binary?