School of Computer Science
Carnegie Mellon

10-601B Introduction to Machine Learning

Deep Learning (Part II)

Readings:
Nielsen (online book)
Neural Networks and Deep Learning

Matt Gormley
Lecture 17
October 26, 2016

Reminders

- Homework 5
  - due Wed., Nov. 2nd
- Homework 6
  - (not out yet)
  - implement a Conv Net!

Outline

- **Deep Neural Networks (DNNs)**
  - Three ideas for training a DNN
  - Experiments: MNIST digit classification
  - Autoencoders
  - Pretraining
- **Convolutional Neural Networks (CNNs)**
  - Convolutional layers
  - Pooling layers
  - Image recognition
- **Recurrent Neural Networks (RNNs)**
  - Bidirectional RNNs
  - Deep Bidirectional RNNs
  - Deep Bidirectional LSTMs
  - Connection to forward-backward algorithm

Part I

Part II

CONVOLUTIONAL NEURAL NETS

Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Convolution

0	0	0
0	1	1
0	1	0

Convolved Image

1	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	0	0	0	0

Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Convolution

0	0	0
0	1	1
0	1	0

Convolved Image

3	2	2	3	1
2	0	2	1	0
2	2	1	0	0
3	1	0	0	0
1	0	0	0	0

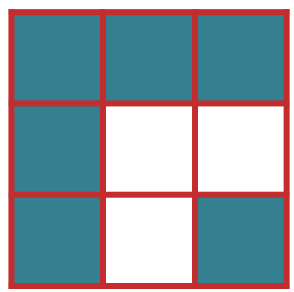
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Convolution



Convolved Image

3	2	2	3	1
2	0	2	1	0
2	2	1	0	0
3	1	0	0	0
1	0	0	0	0

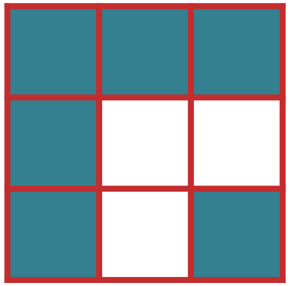
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Convolution



Convolved Image

3	2	2	3	1
2	0	2	1	0
2	2	1	0	0
3	1	0	0	0
1	0	0	0	0

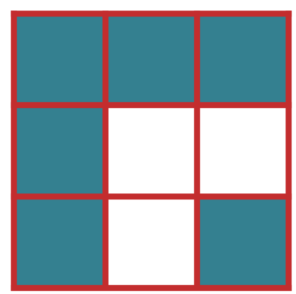
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Convolution

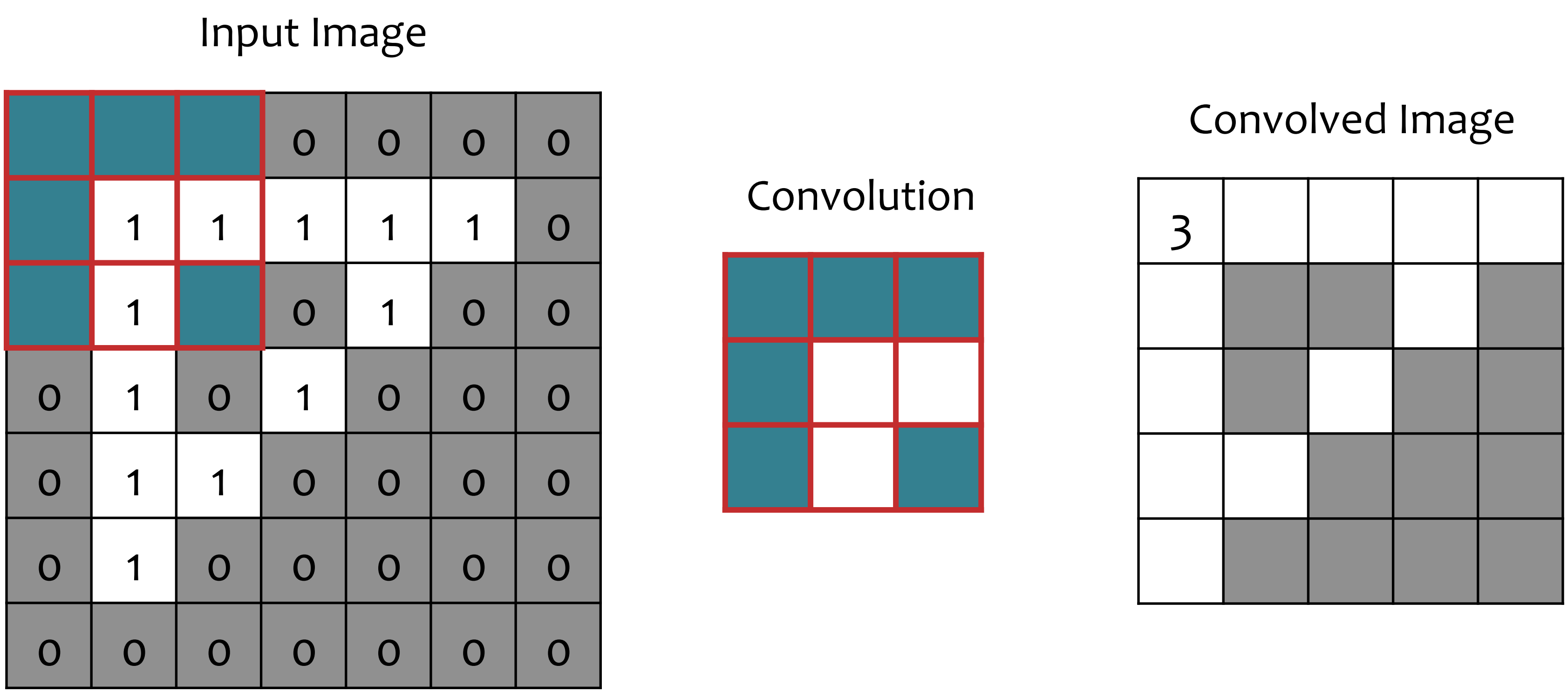


Convolved Image

3	2	2	3	1
2	0	2	1	0
2	2	1	0	0
3	1	0	0	0
1	0	0	0	0

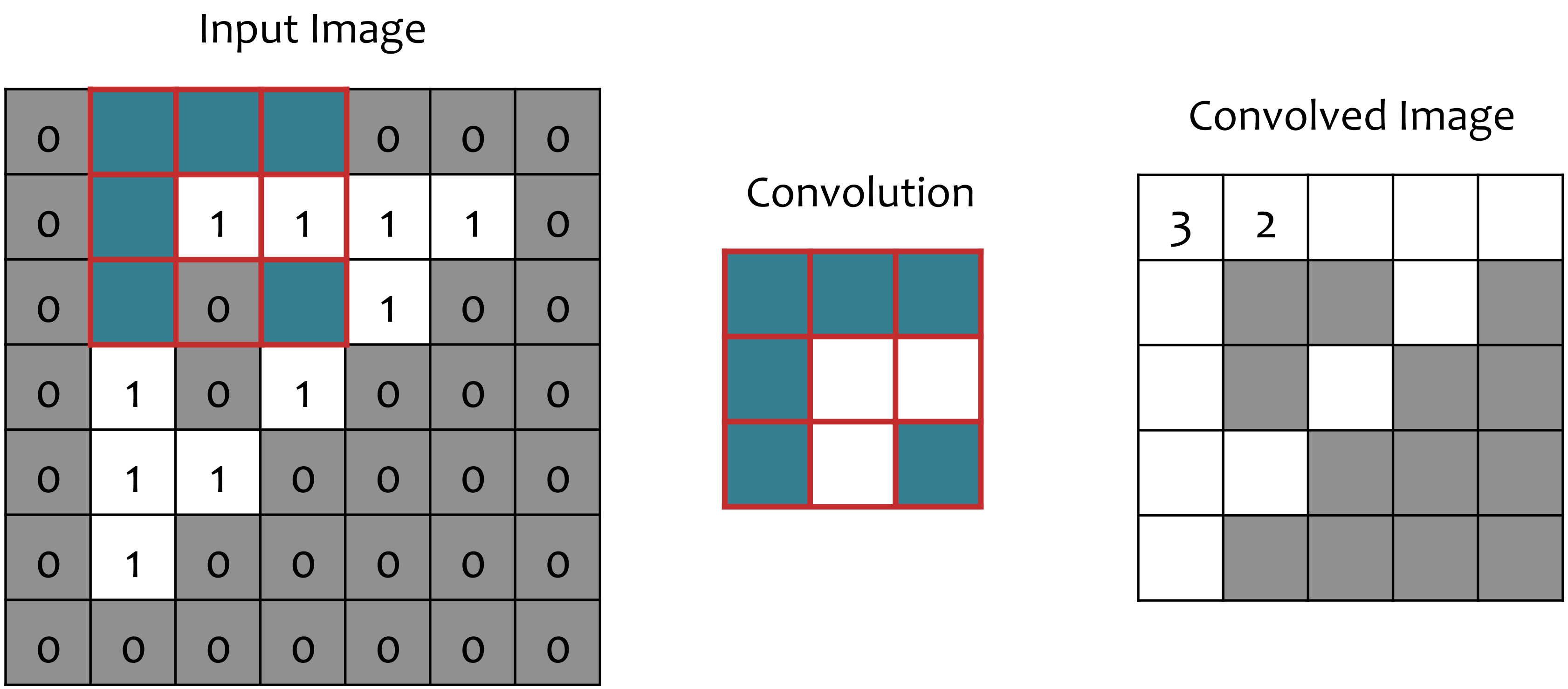
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



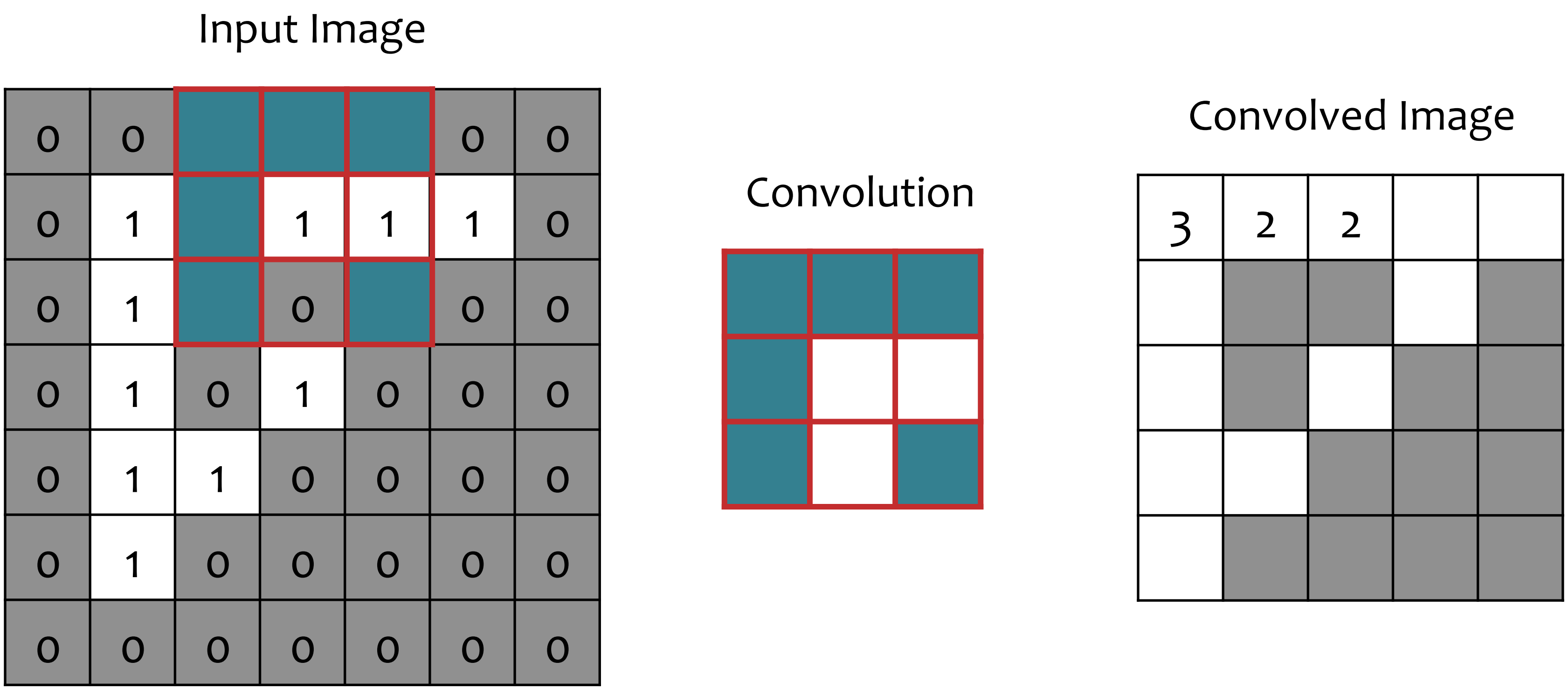
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



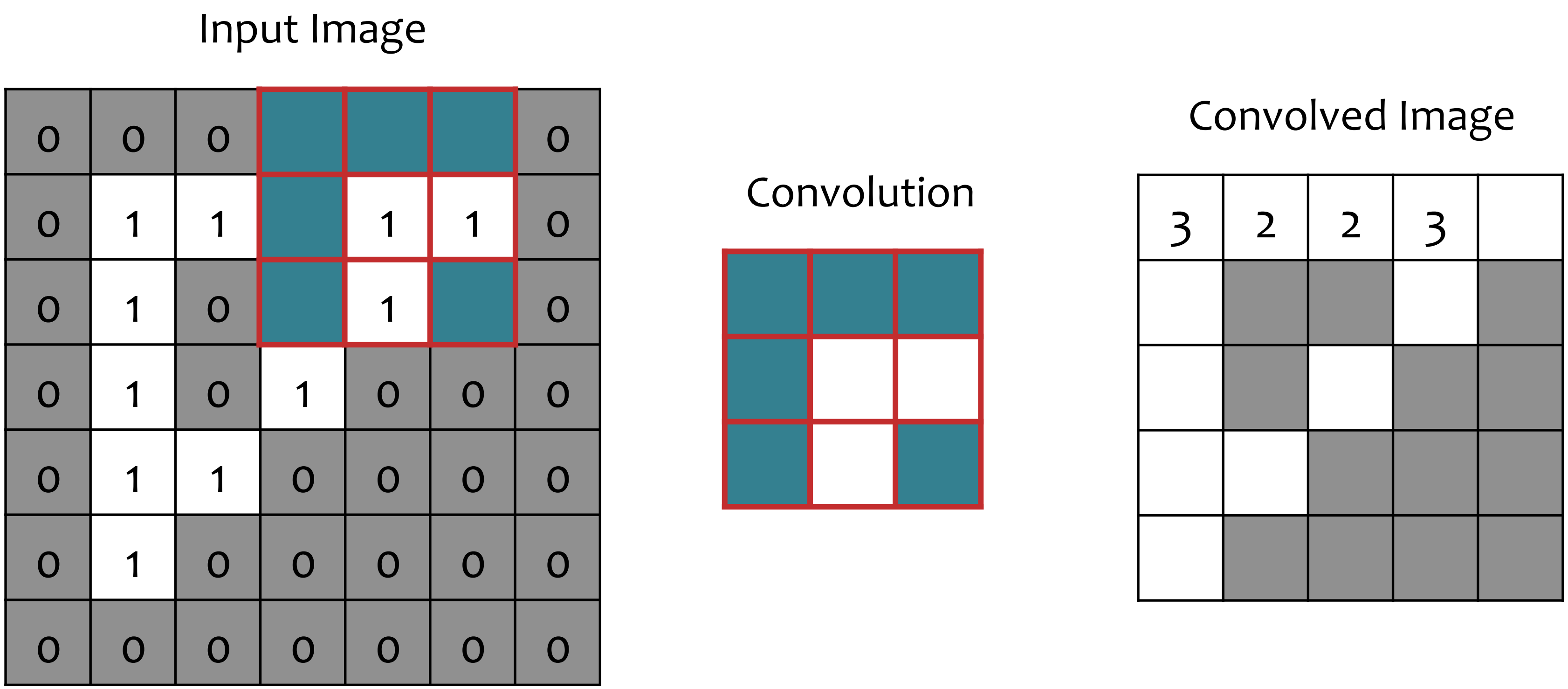
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



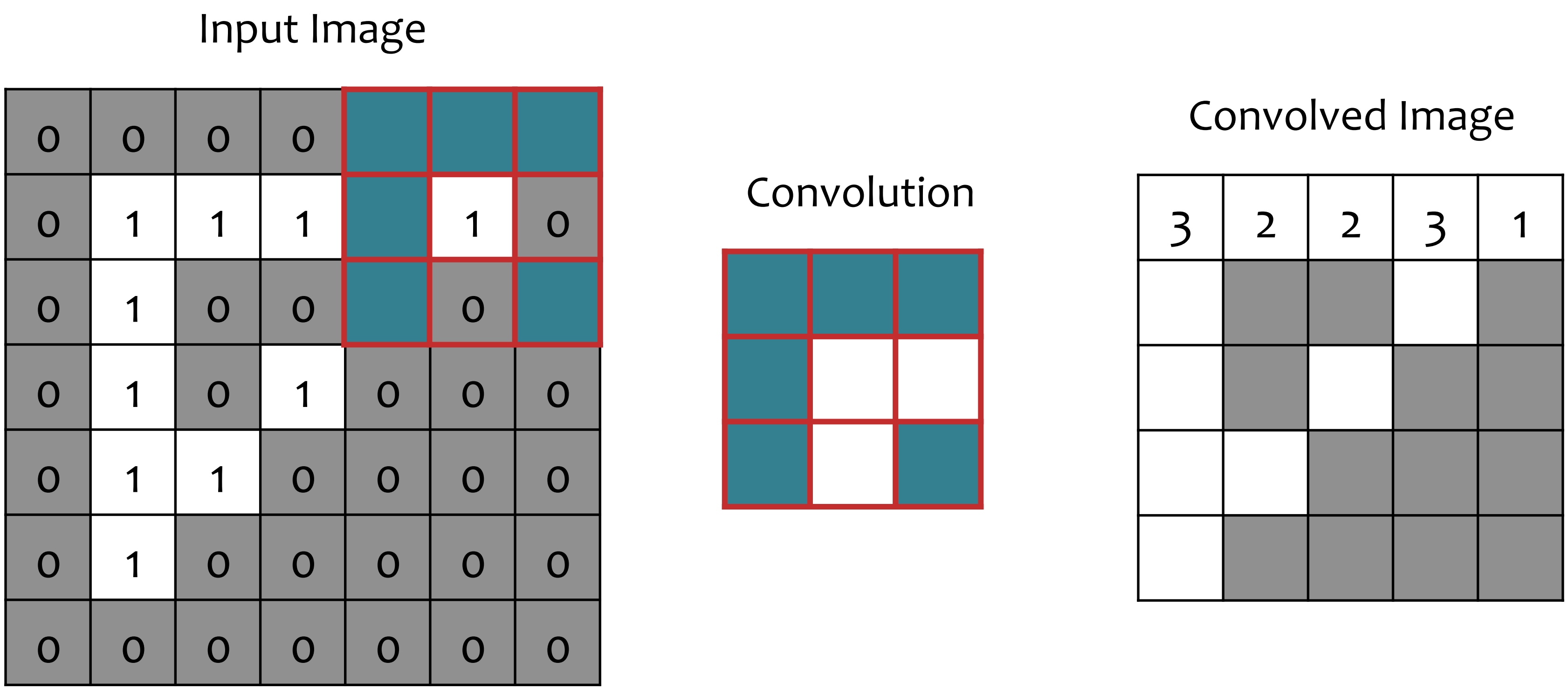
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



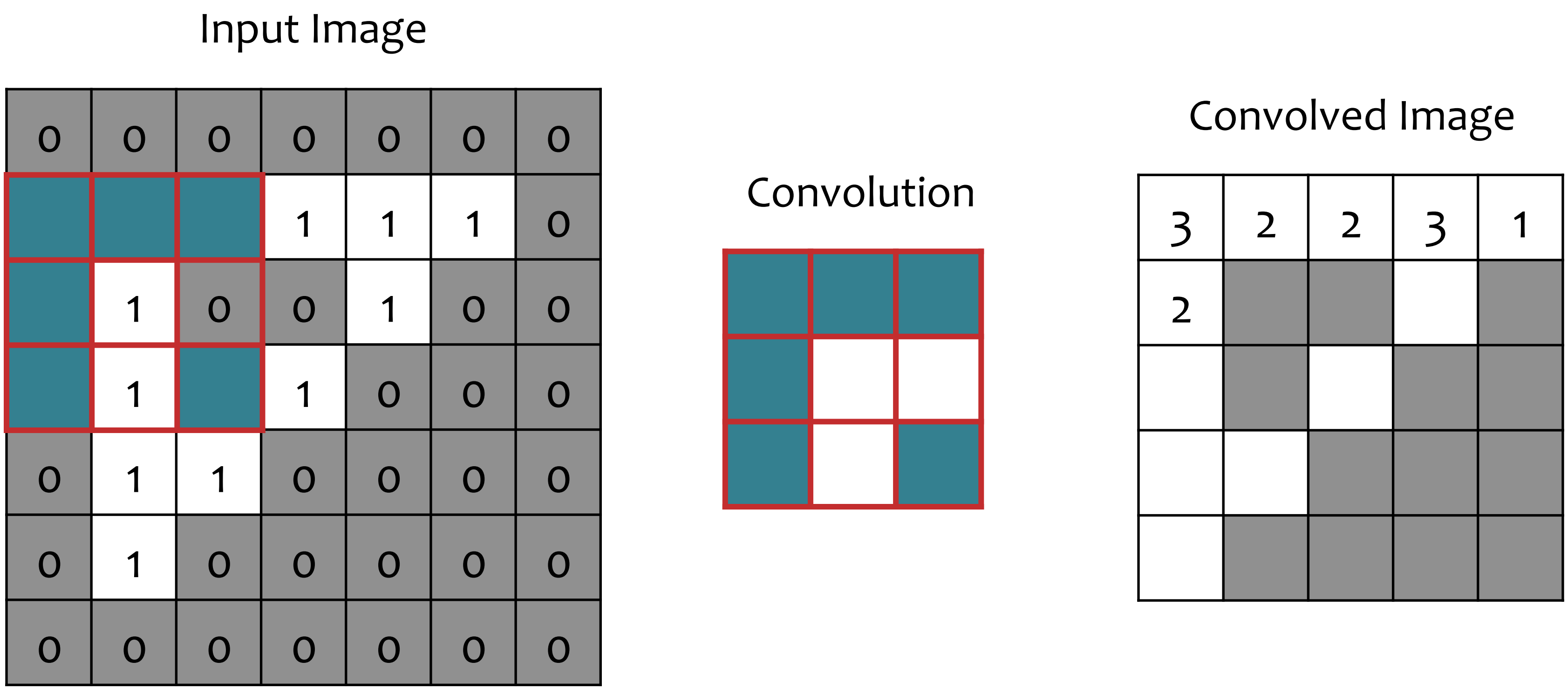
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



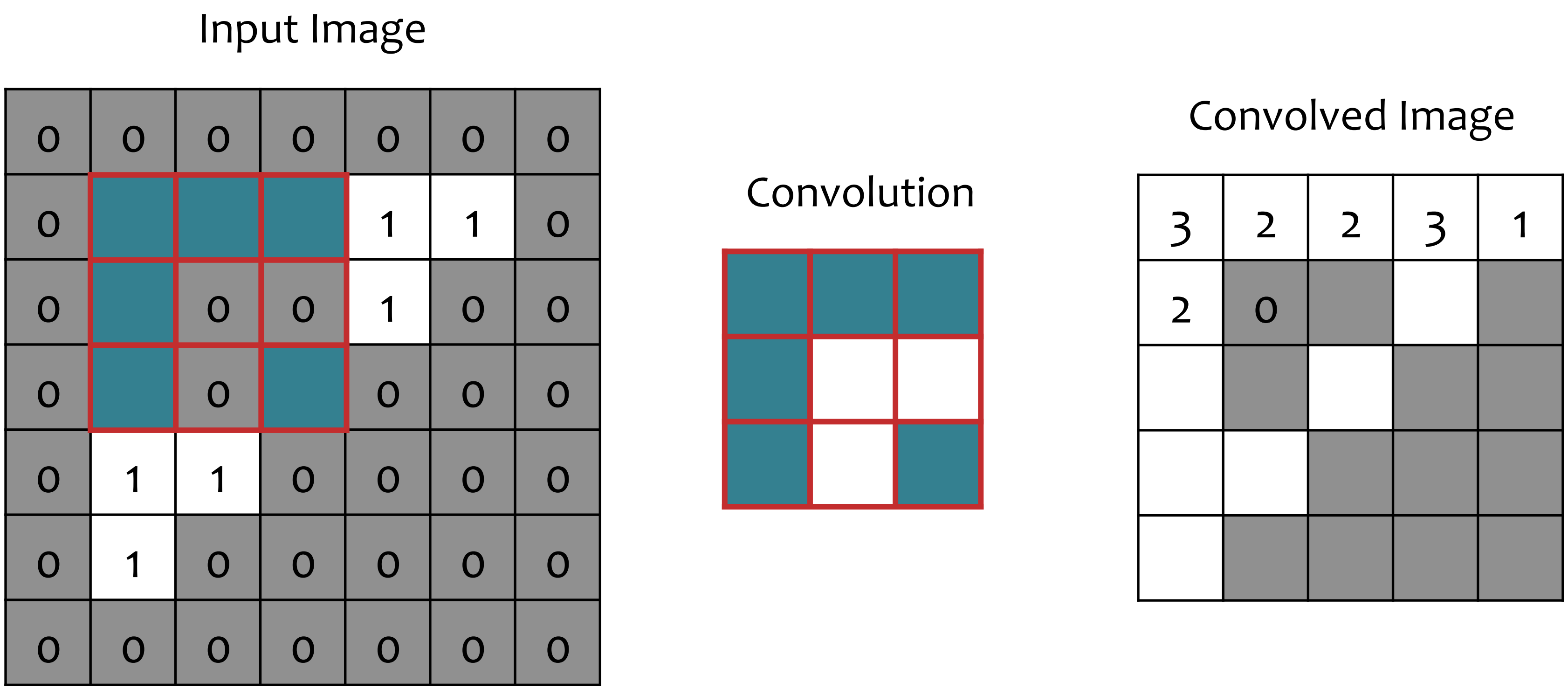
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.



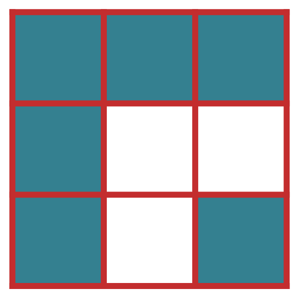
Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Convolution



Convolved Image

3	2	2	3	1
2	0	2	1	0
2	2	1	0	0
3	1	0	0	0
1	0	0	0	0

Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Identity Convolution

0	0	0
0	1	0
0	0	0

Convolved Image

1	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	0	0	0	0

Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Blurring Convolution

.1	.1	.1
.1	.2	.1
.1	.1	.1

Convolved Image

.4	.5	.5	.5	.4
.4	.2	.3	.6	.3
.5	.4	.4	.2	.1
.5	.6	.2	.1	0
.4	.3	.1	0	0

Background: Image Processing

A **convolution matrix** is used in image processing for tasks such as edge detection, blurring, sharpening, etc.

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Blurring Convolution

.1	.1	.1
.1	.2	.1
.1	.1	.1

Convolved Image

.4	.5	.5	.5	.4
.4	.2	.3	.6	.3
.5	.4	.4	.2	.1
.5	.6	.2	.1	0
.4	.3	.1	0	0

Convolutional Neural Network (CNN)

CNN key idea:
Treat convolution matrix as parameters and learn them!

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Learned Convolution

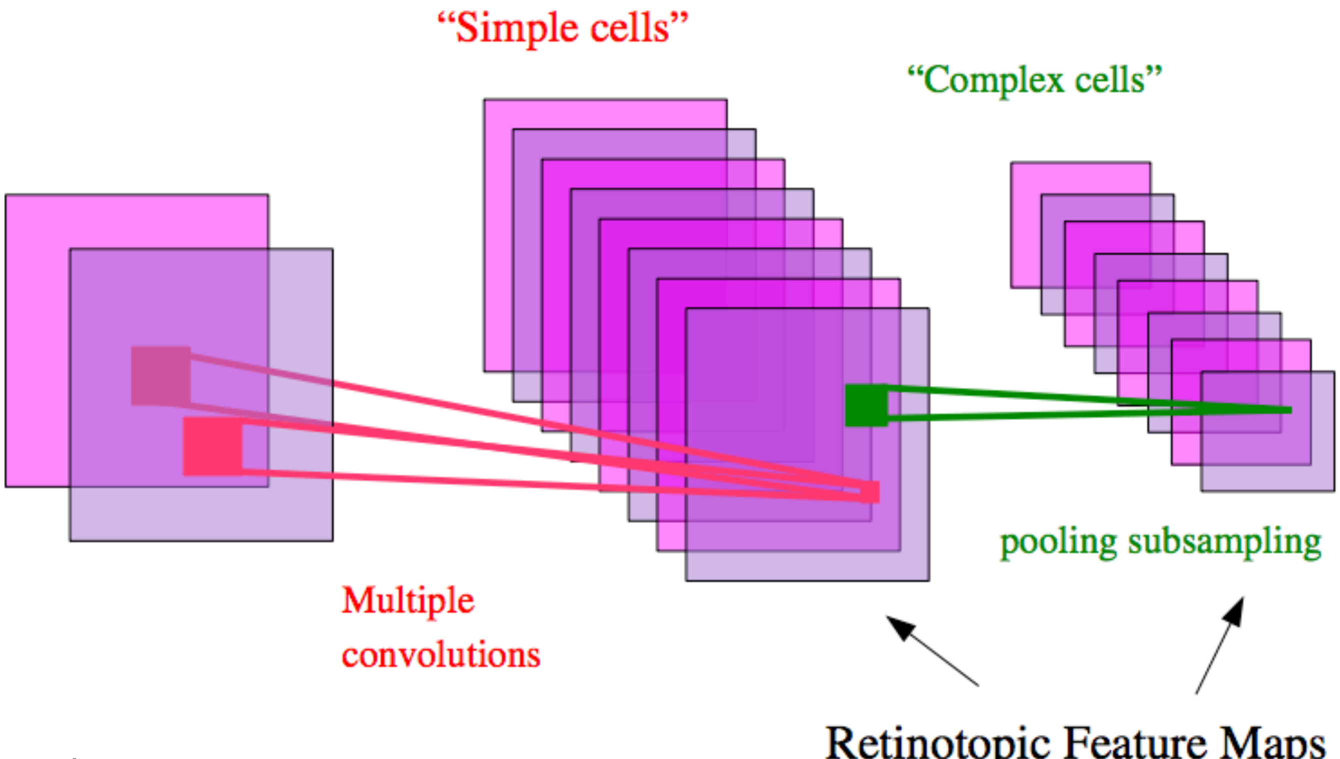
θ_{11}	θ_{12}	θ_{13}
θ_{21}	θ_{22}	θ_{23}
θ_{31}	θ_{32}	θ_{33}

Convolved Image

.4	.5	.5	.5	.4
.4	.2	.3	.6	.3
.5	.4	.4	.2	.1
.5	.6	.2	.1	0
.4	.3	.1	0	0

Model of vision in animals

- **[Hubel & Wiesel 1962]:**
  - simple cells detect local features
  - complex cells "pool" the outputs of simple cells within a retinotopic neighborhood.



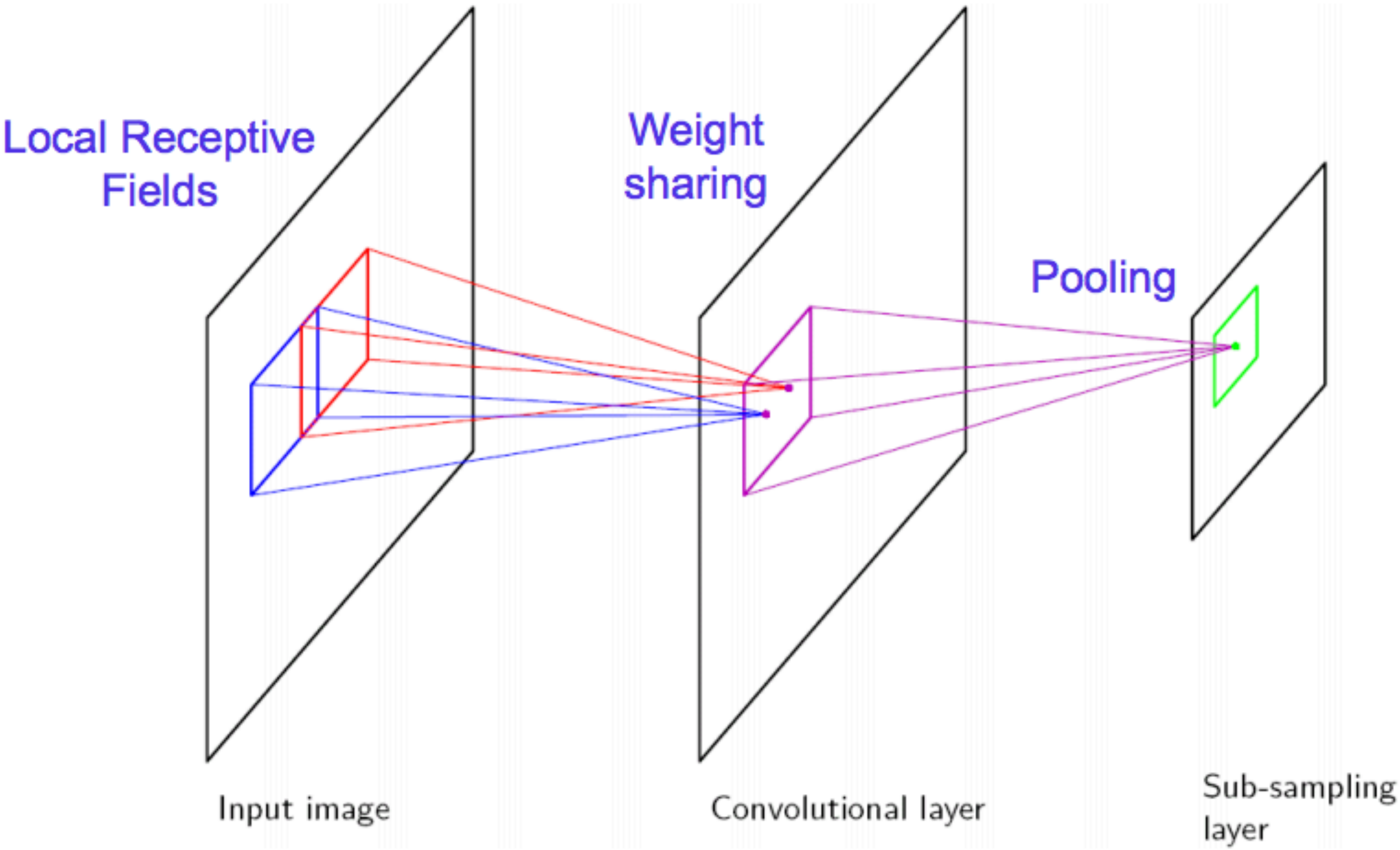
Slide from William Cohen

Huber & Wiesel Video

https://www.youtube.com/watch?v=8VdFf3egwfg

Vision with ANNs

(LeCun et al., 1989)



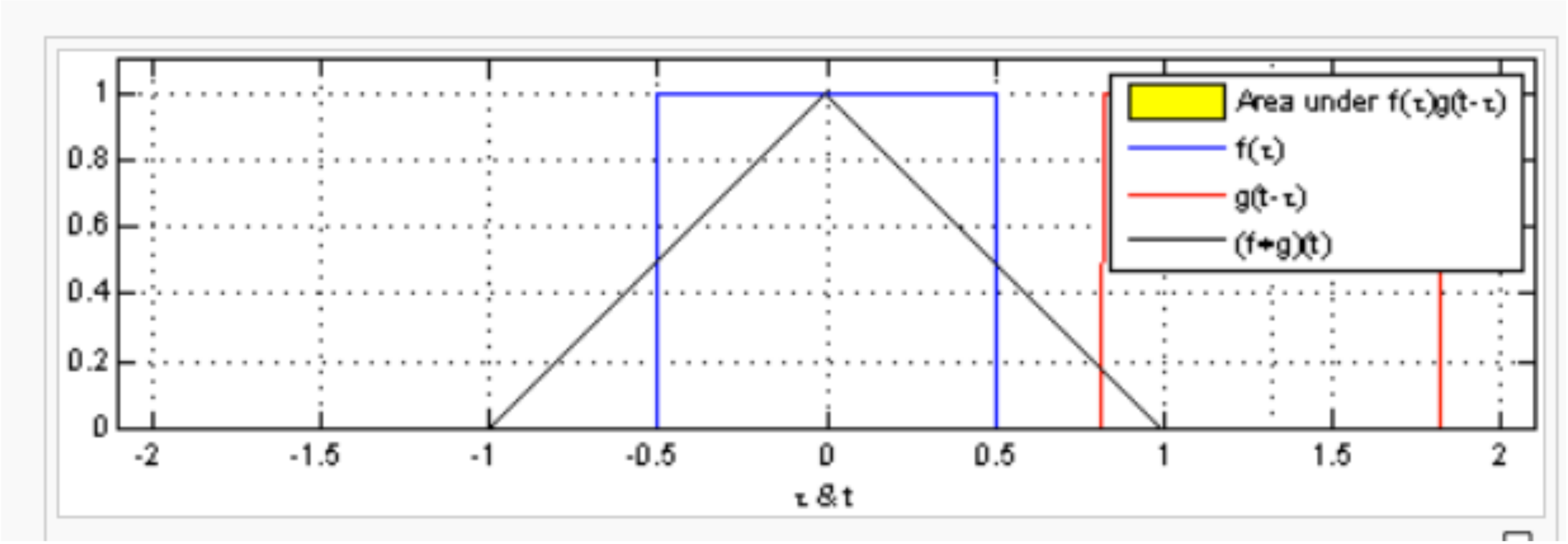
Slide from William Cohen

What's a convolution?

https://en.wikipedia.org/wiki/Convolution

1-D

$$
\begin{aligned}
(f * g)(t) &\stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau \\
&= \int_{-\infty}^{\infty} f(t-\tau)\, g(\tau)\, d\tau.
\end{aligned}
$$



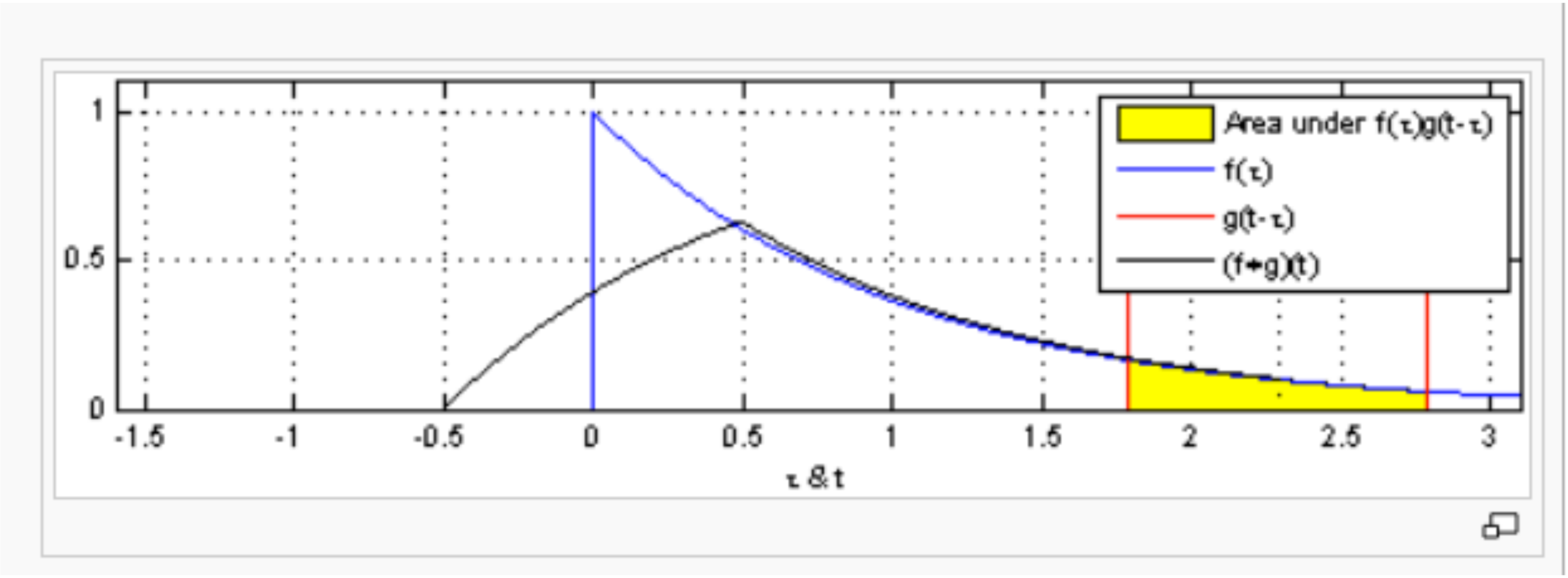
Slide from William Cohen

What's a convolution?

https://en.wikipedia.org/wiki/Convolution

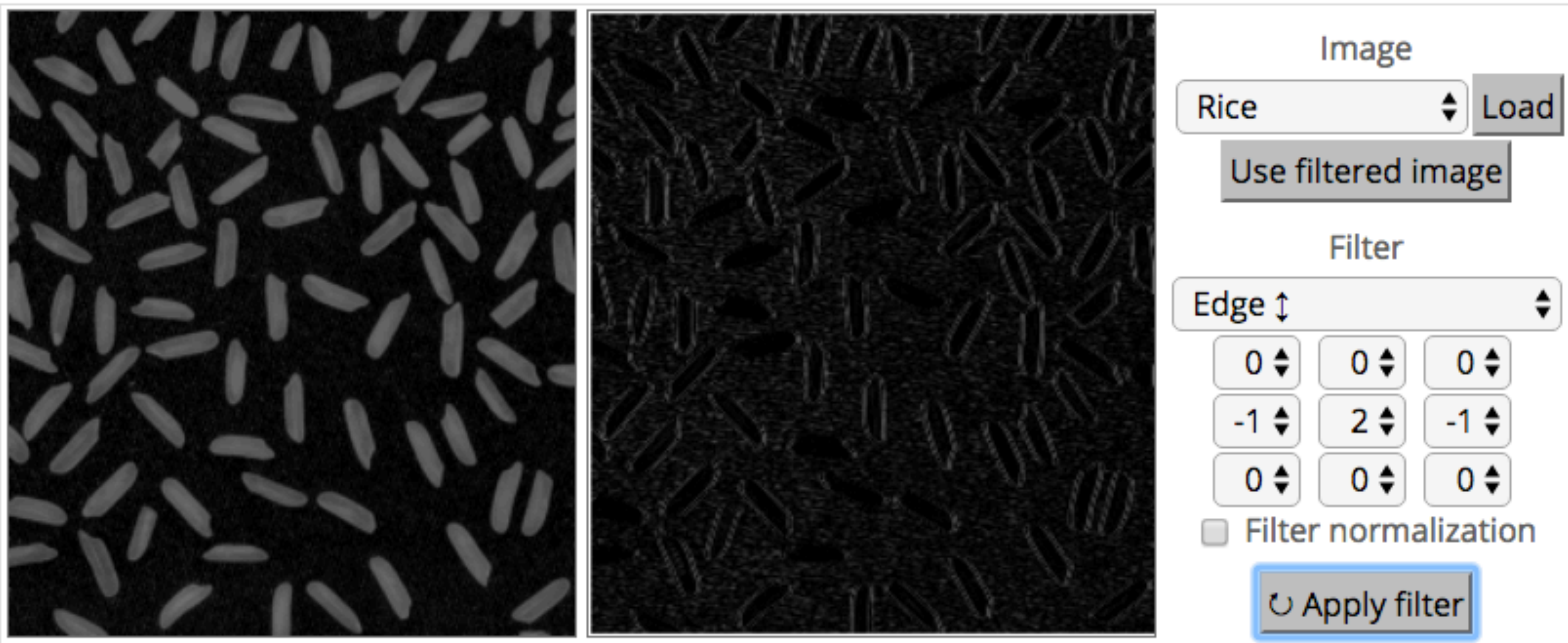
1-D

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$
$$= \int_{-\infty}^{\infty} f(t-\tau)\, g(\tau)\, d\tau.$$



What's a convolution?

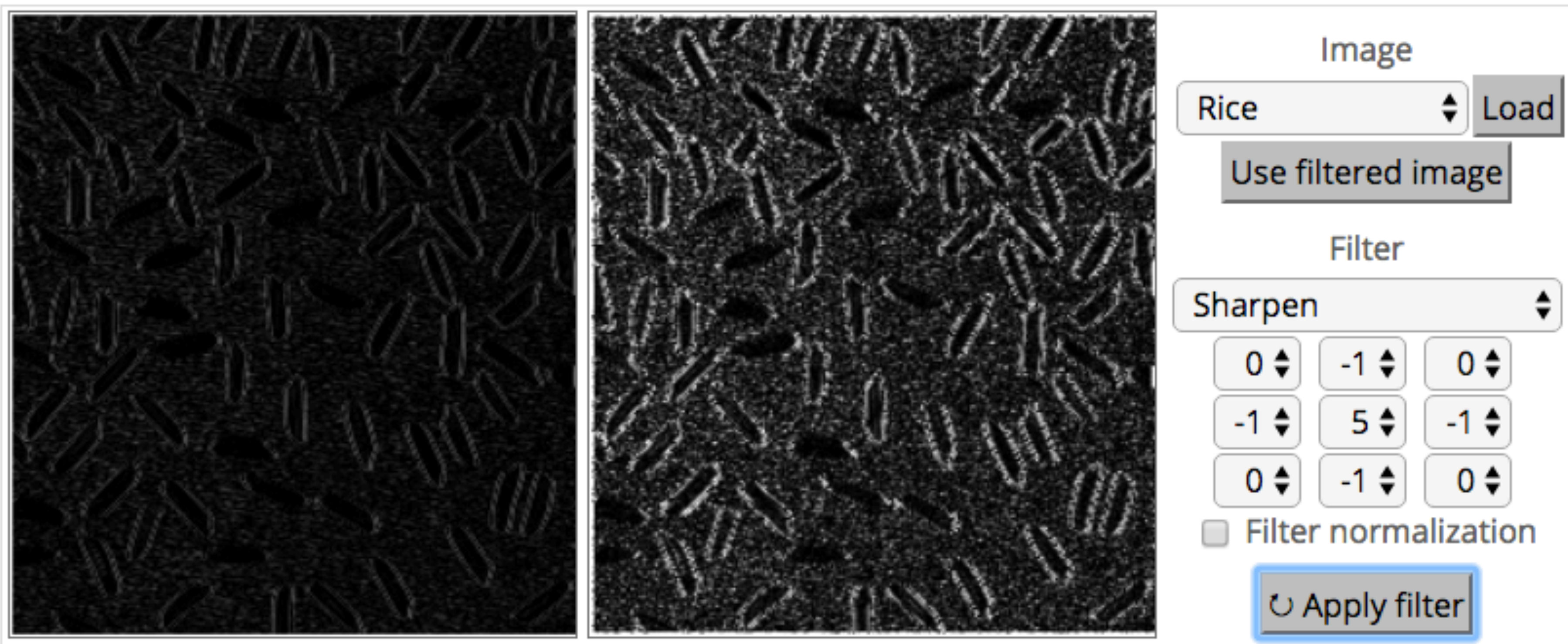
http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Slide from William Cohen

What’s a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Slide from William Cohen

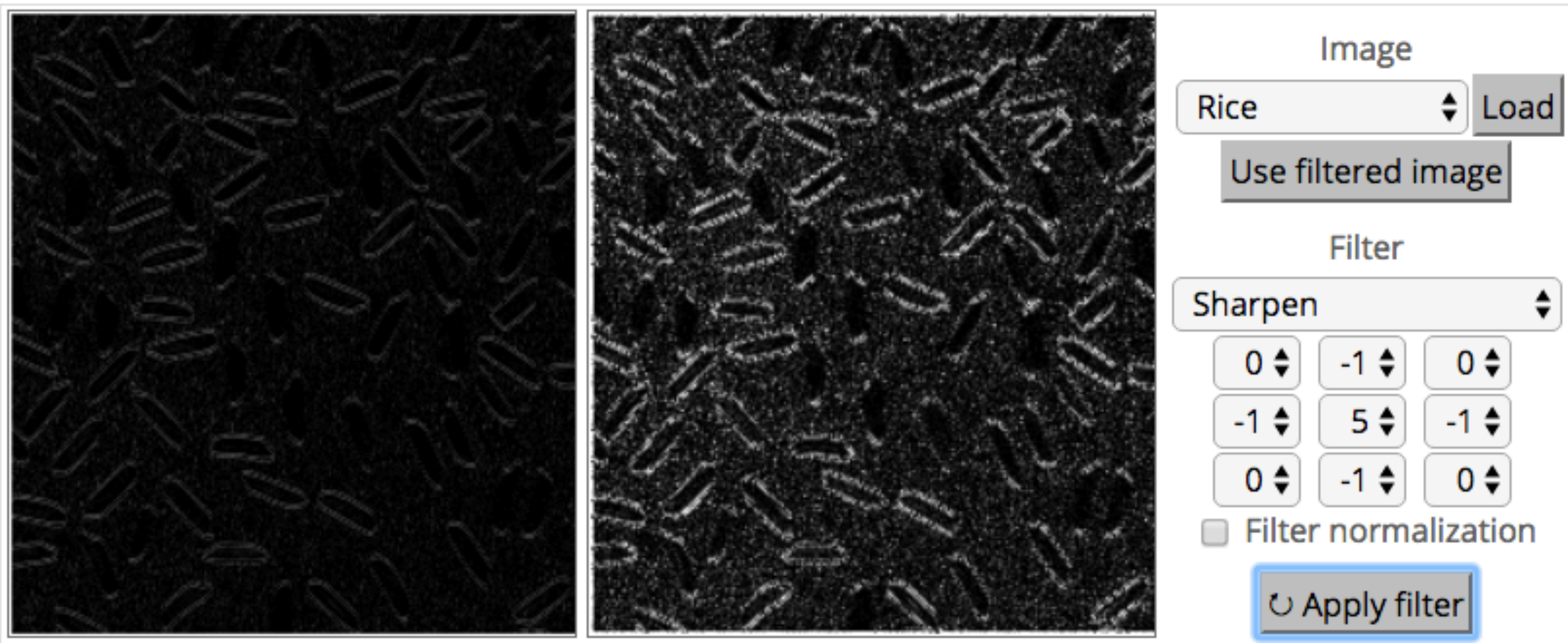
What's a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo

Slide from William Cohen

What's a convolution?

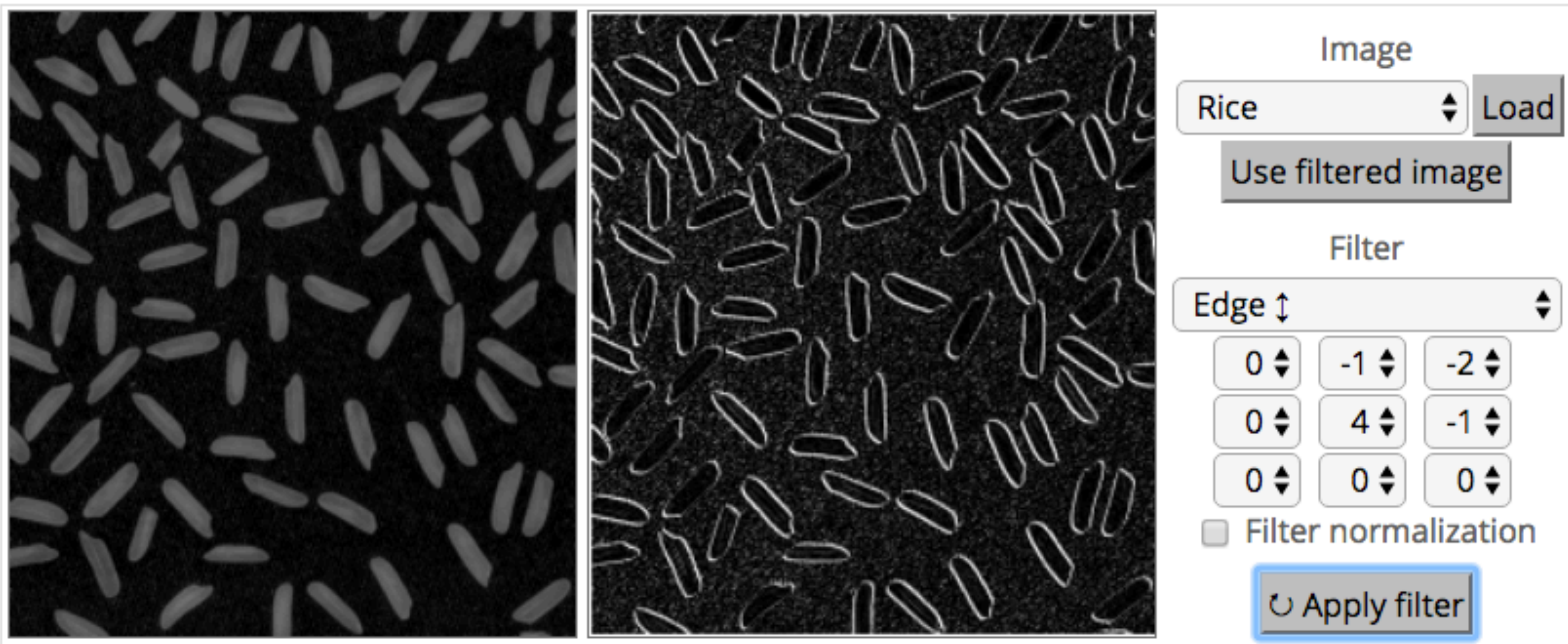
http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Slide from William Cohen

What’s a convolution?

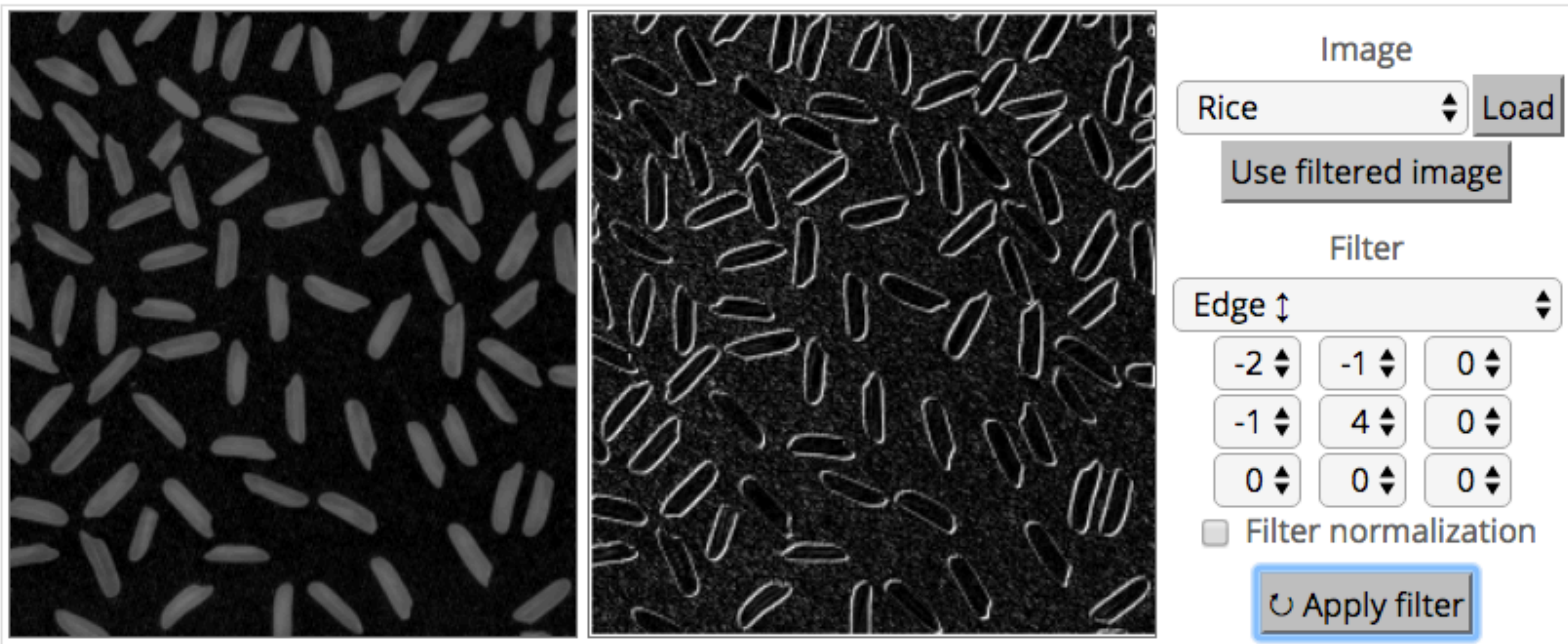
http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Slide from William Cohen

What’s a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Slide from William Cohen

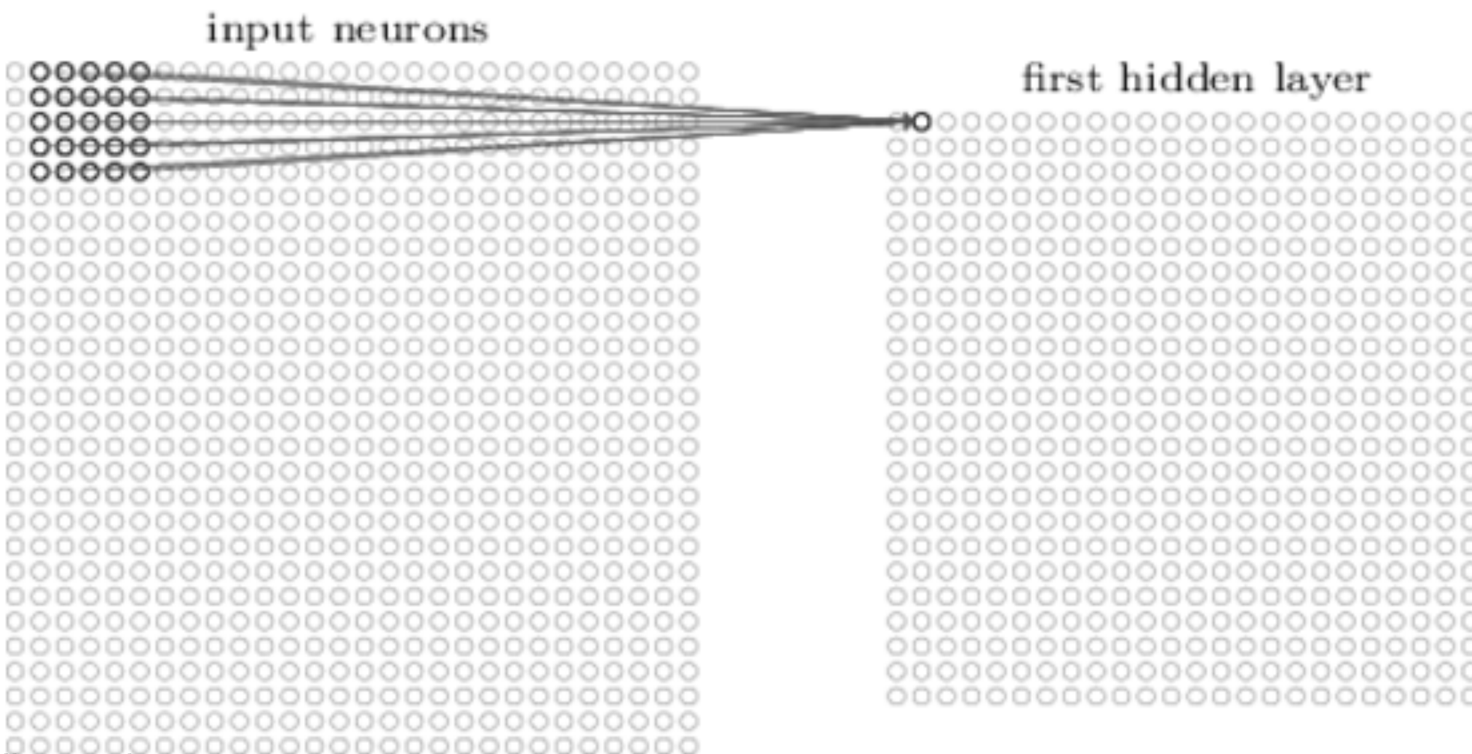
What's a convolution?

- Basic idea:
  - Pick a 3x3 matrix F of weights
  - Slide this over an image and compute the "inner product" (similarity) of F and the corresponding field of the image, and replace the pixel in the center of the field with the output of the inner product operation
- Key point:
  - Different convolutions extract different types of low-level "features" from an image
  - All that we need to vary to generate these different features is the weights of F

Slide from William Cohen

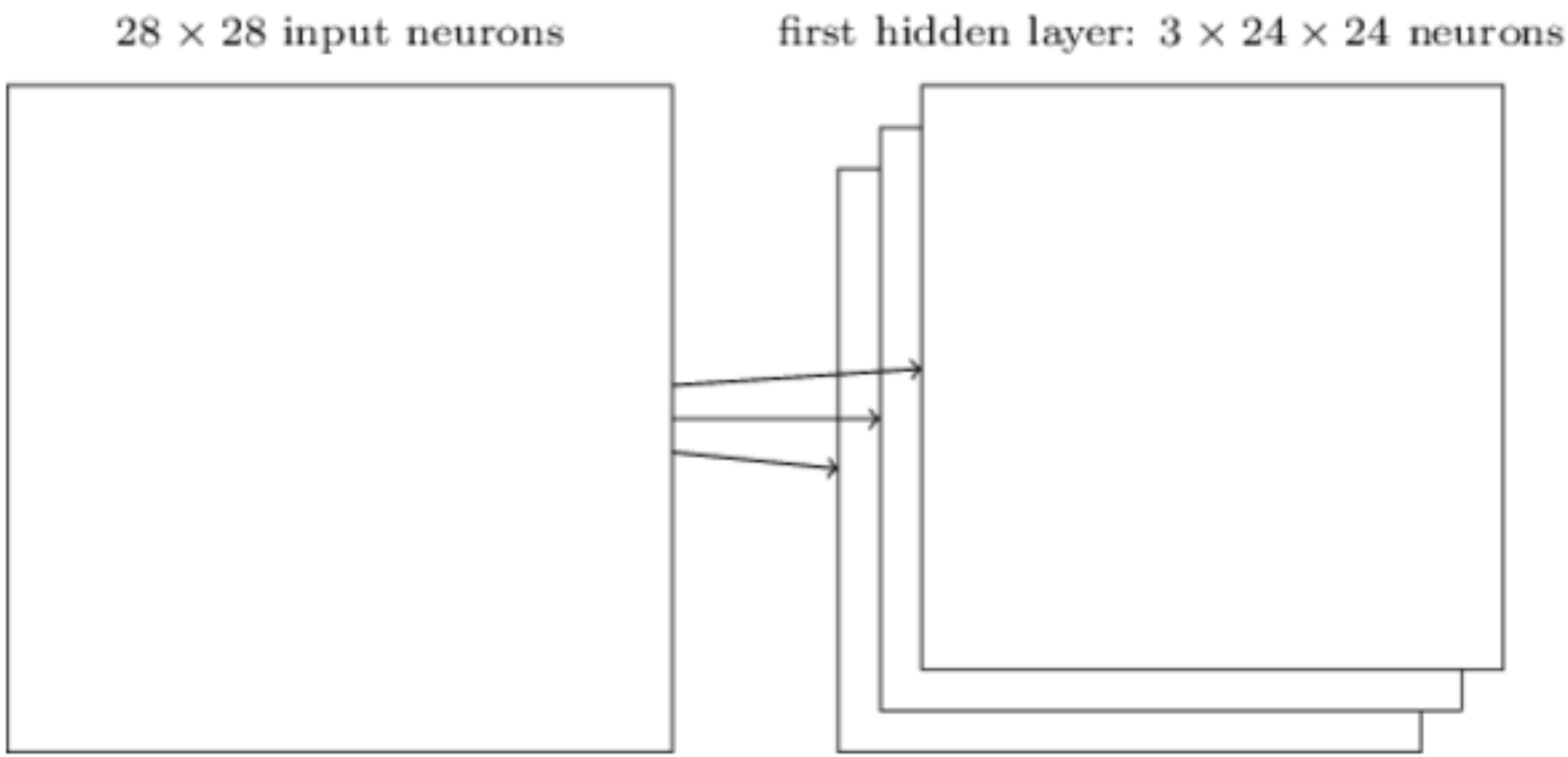
How do we convolve an image with an ANN?

Note that the parameters in the matrix defining the convolution are **tied** across all places that it is used



Slide from William Cohen

How do we do many convolutions of an image with an ANN?



Slide from William Cohen

Convolutional Neural Network (CNN)

Typical layers include:

- Convolutional layer
- Max-pooling layer
- Fully connected layer
- (Nonlinear layer)
- Softmax

These can be arranged into arbitrarily deep topologies

PROC. OF THE IEEE, NOVEMBER 1998 7

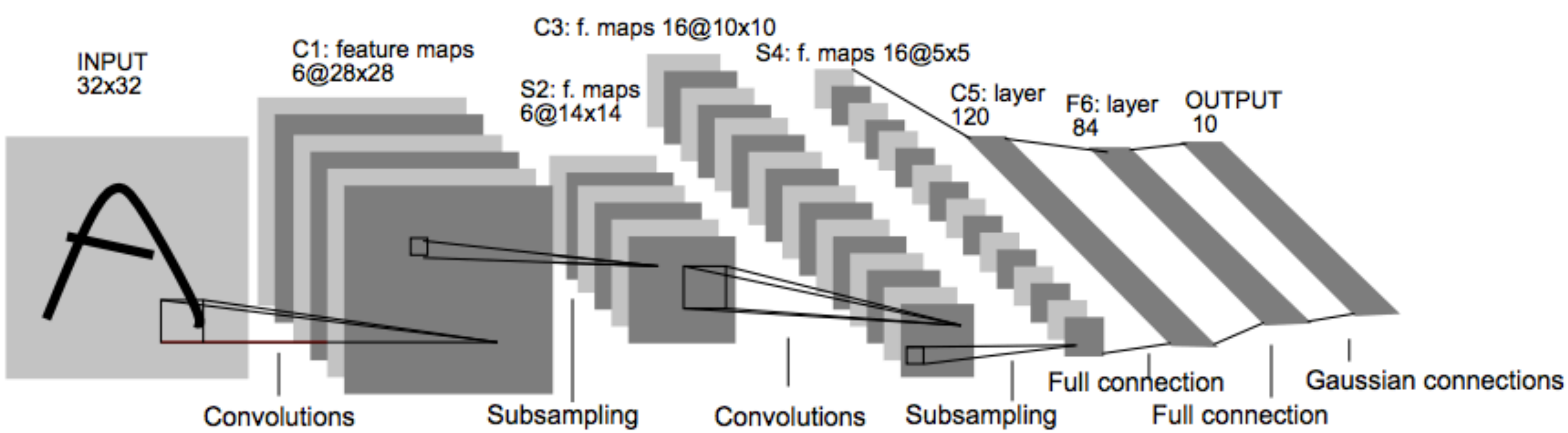
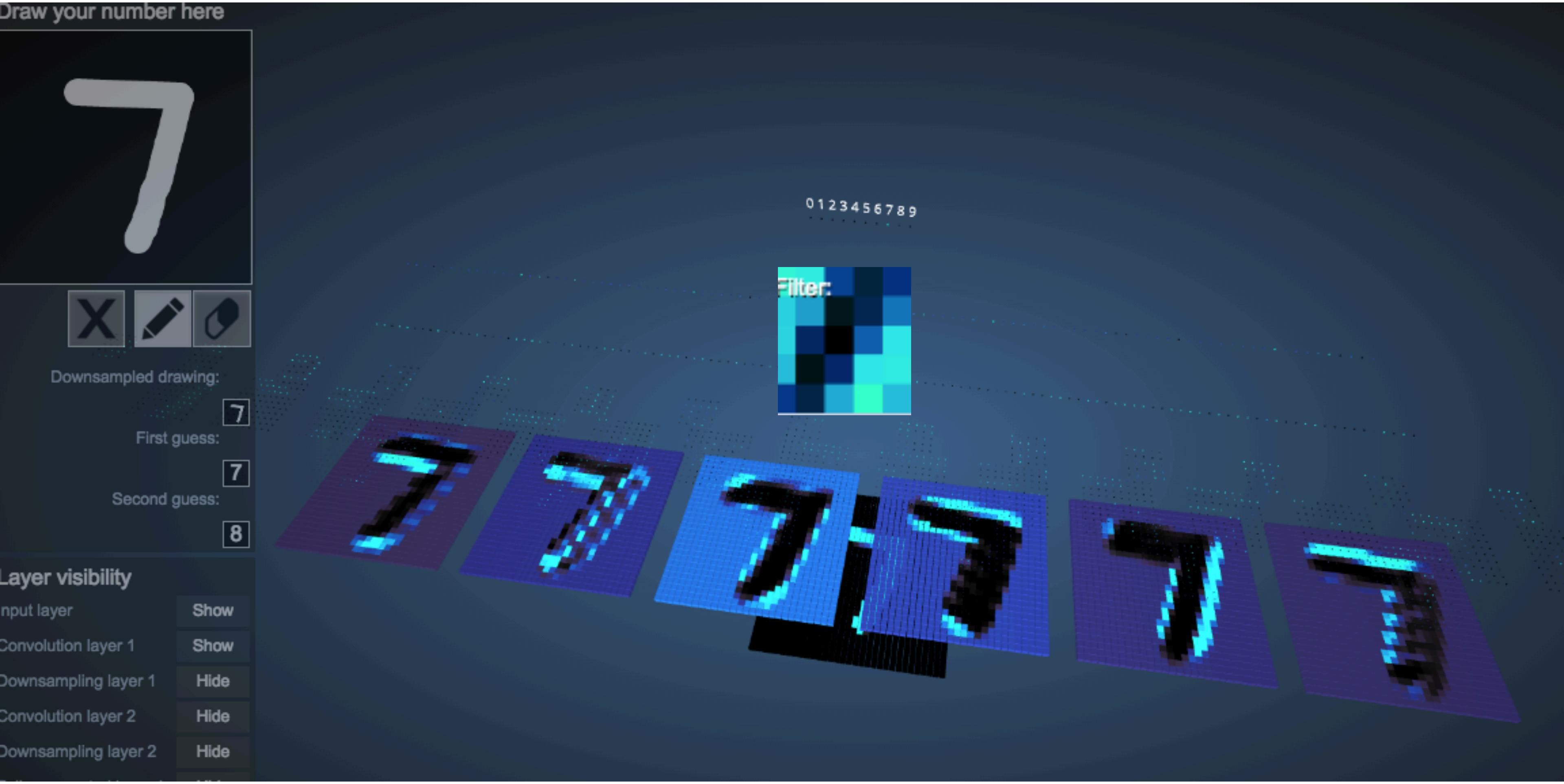


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Example: 6 convolutions of a digit

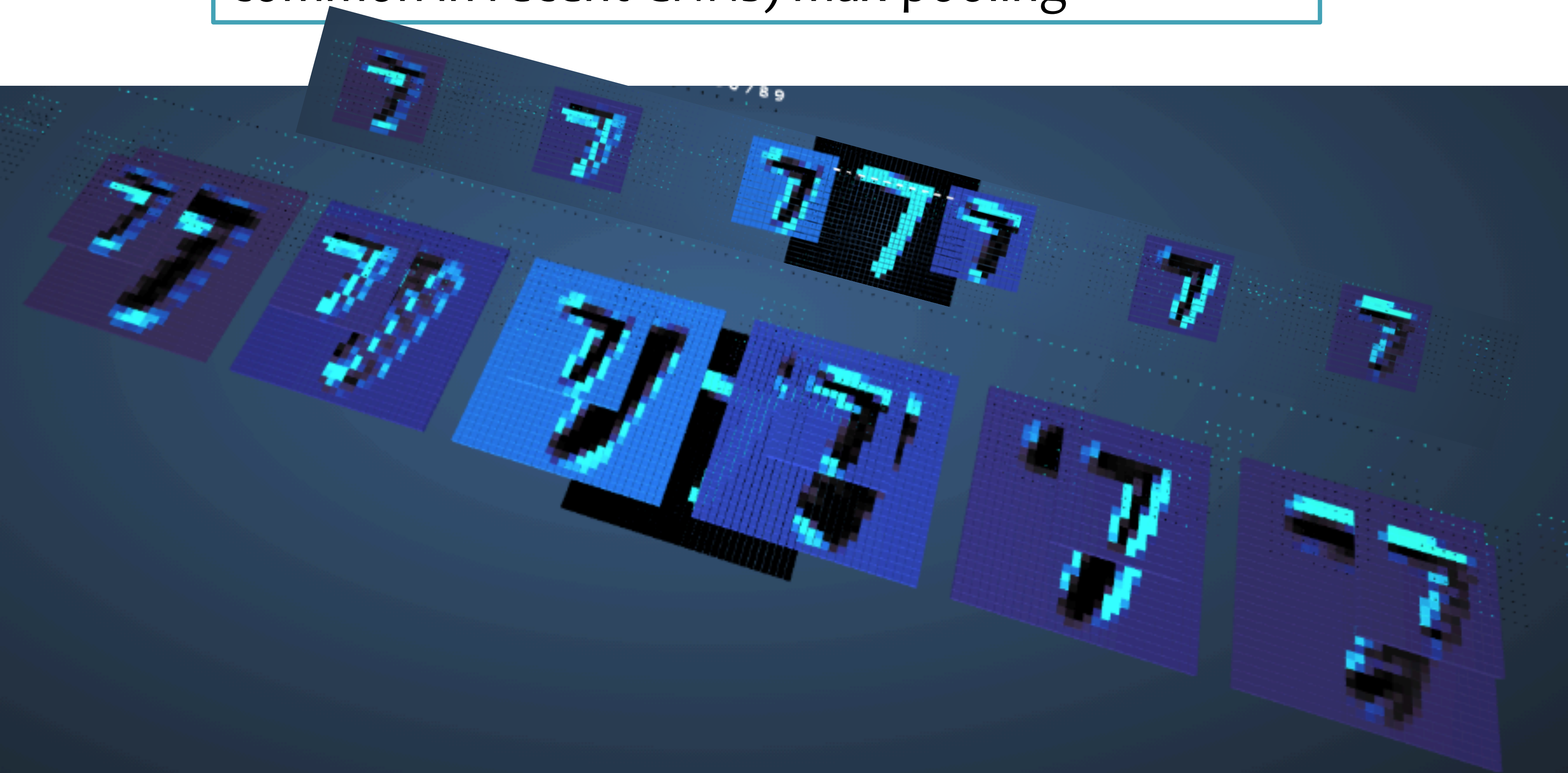
http://scs.ryerson.ca/~aharley/vis/conv/



Slide from William Cohen

CNNs typically alternate convolutions, non-linearity, and then downsampling

Downsampling is usually averaging or (more common in recent CNNs) max-pooling



Slide from William Cohen

Convolution of a Color Image

- Color images consist of 3 floats per pixel for RGB (red, green blue) color values
- Convolution must also be 3-dimensional

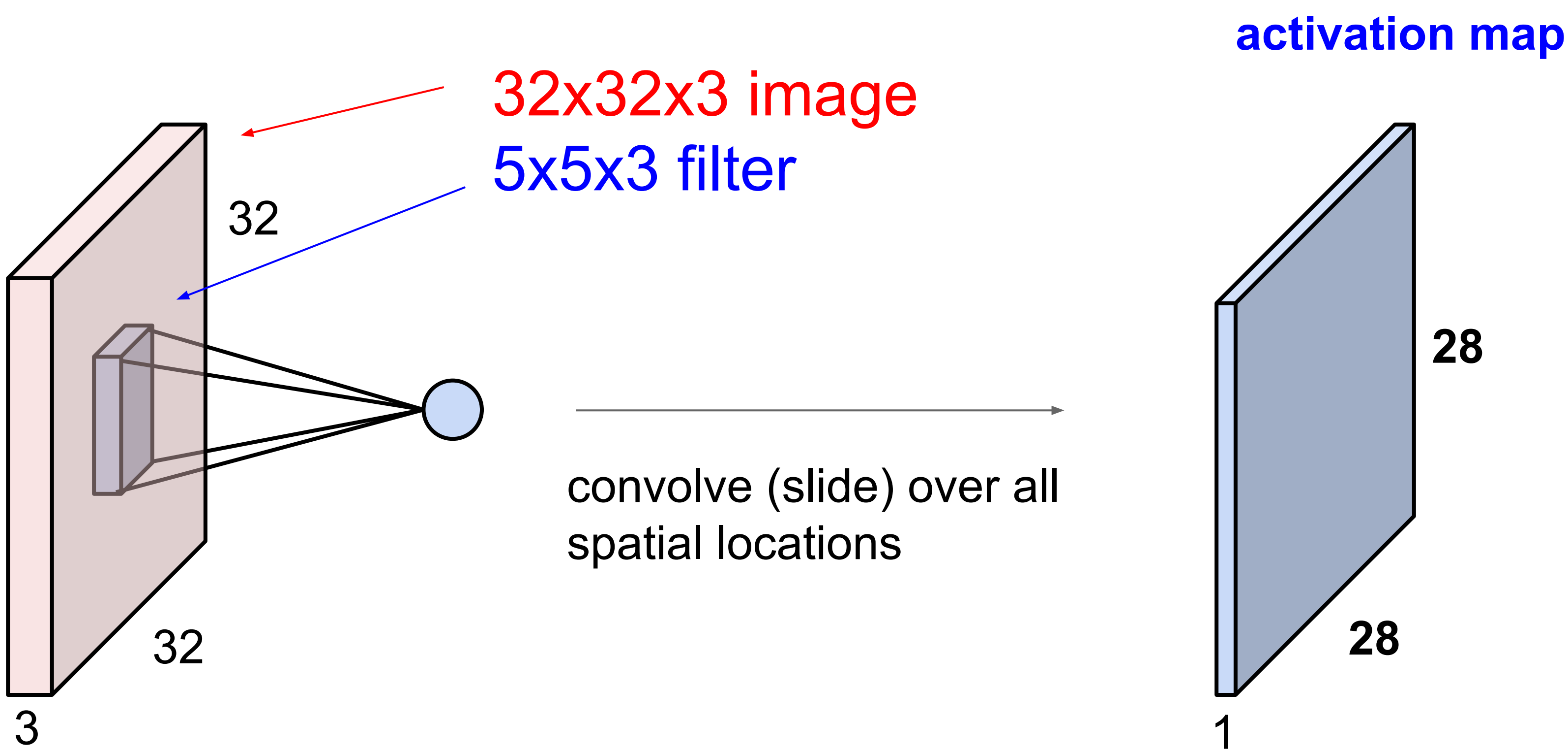


Figure from Fei-Fei Li & Andrej Karpathy & Justin Johnson (CS231N)

Convolution of a Color Image

Animation of 3D convolution

http://cs231n.github.io/convolutional-networks/

Figure from Fei-Fei Li & Andrej Karpathy & Justin Johnson (CS231N)

Max-pooling

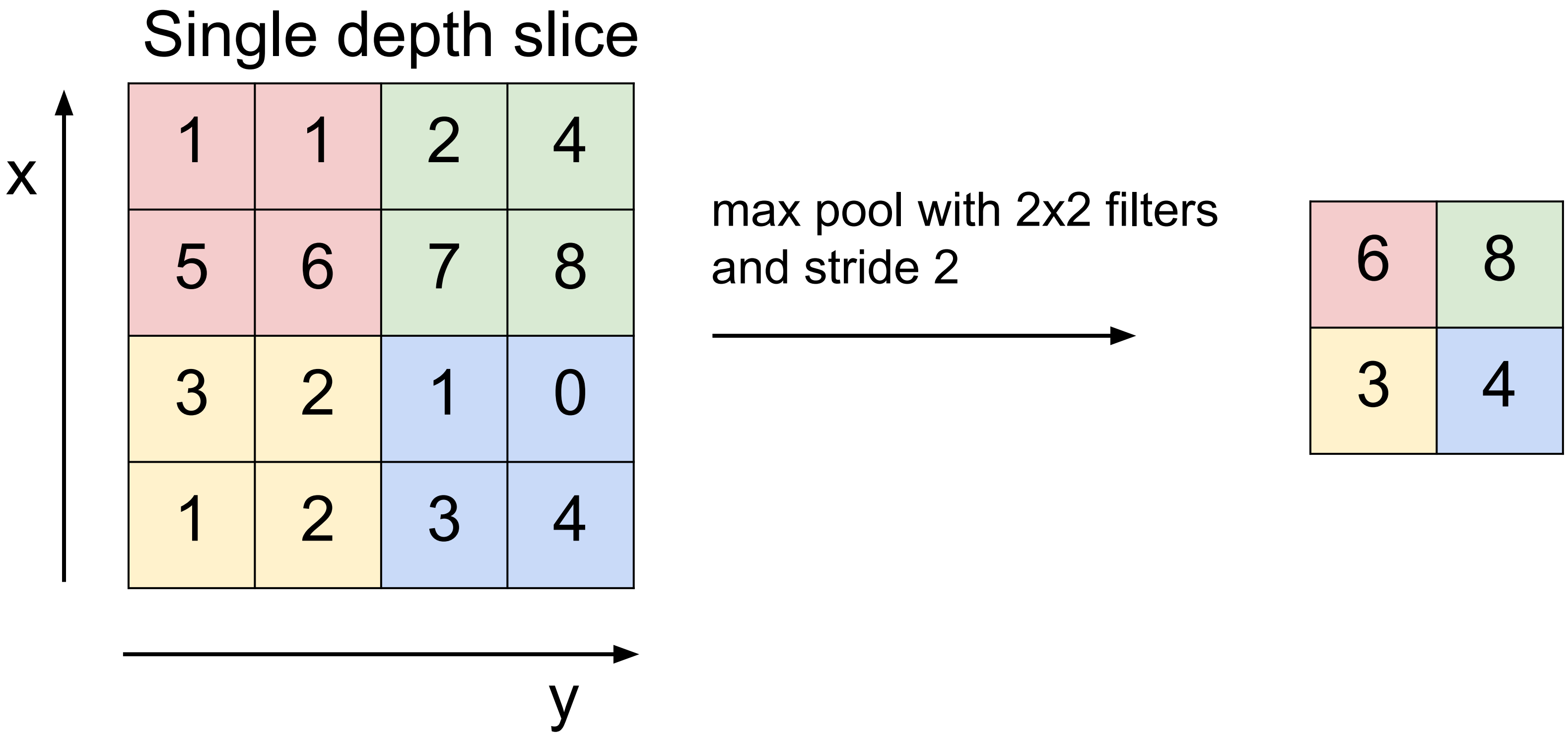


Figure from Fei-Fei Li & Andrej Karpathy & Justin Johnson (CS231N)

Max-pooling

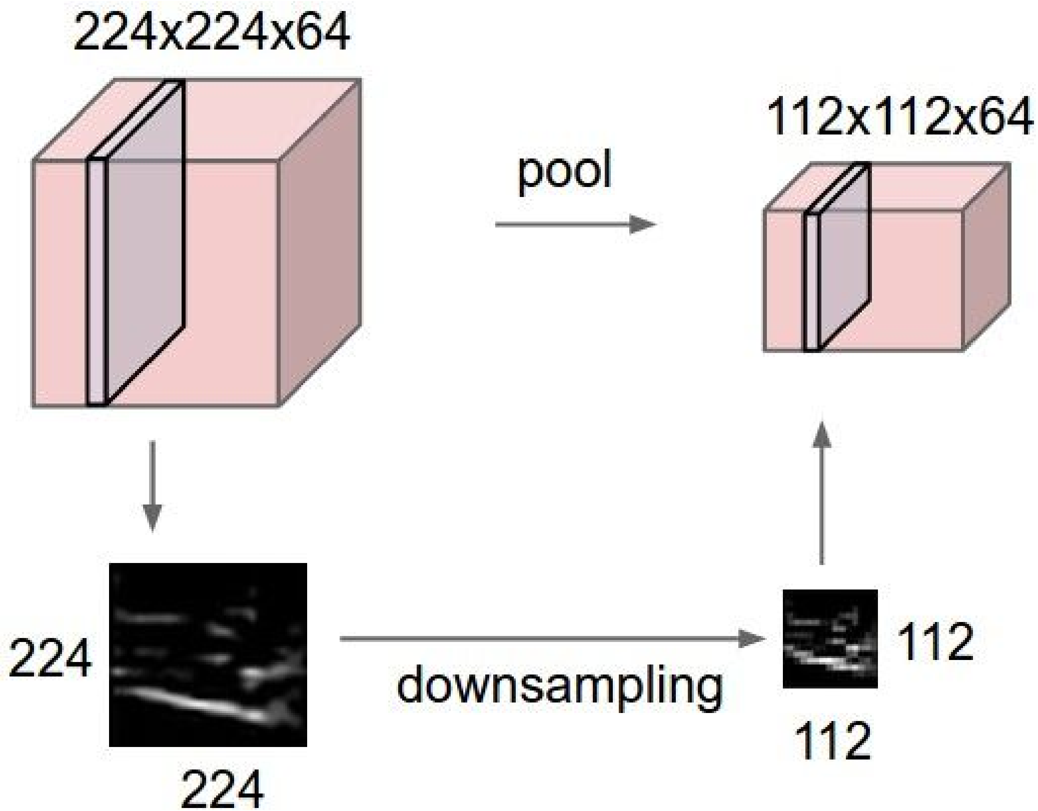


Figure from Fei-Fei Li & Andrej Karpathy & Justin Johnson (CS231N)

Why do max-pooling?

- Saves space
- Reduces overfitting?
- Because I'm going to add *more* convolutions after it!
  - Allows the short-range convolutions to extend over larger subfields of the images
    - So we can spot larger objects
    - Eg, a long horizontal line, or a corner, or …

PROC. OF THE IEEE, NOVEMBER 1998 7

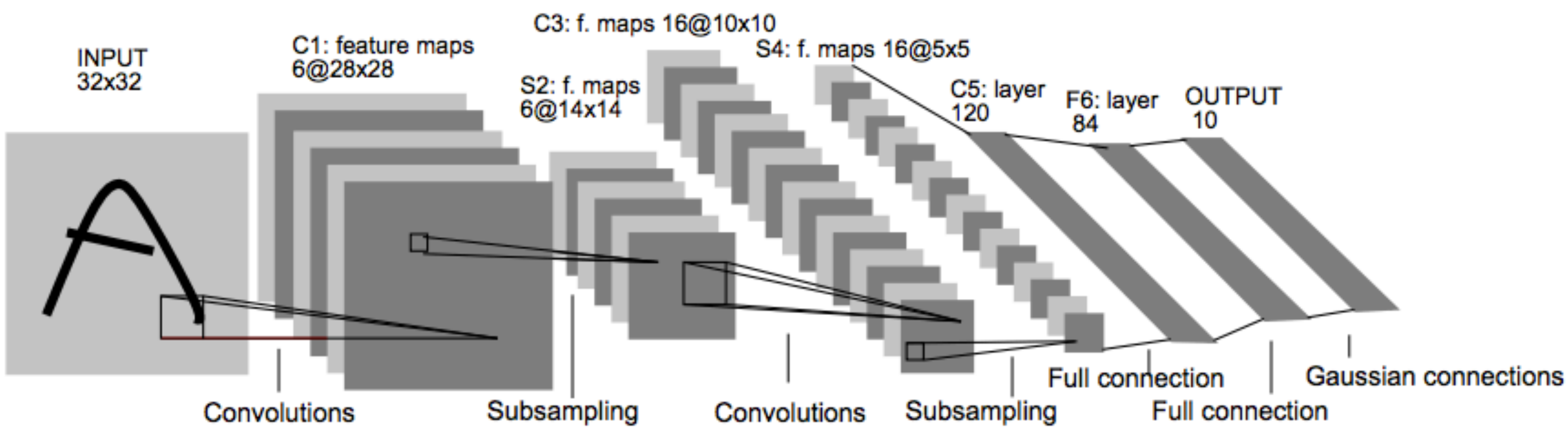
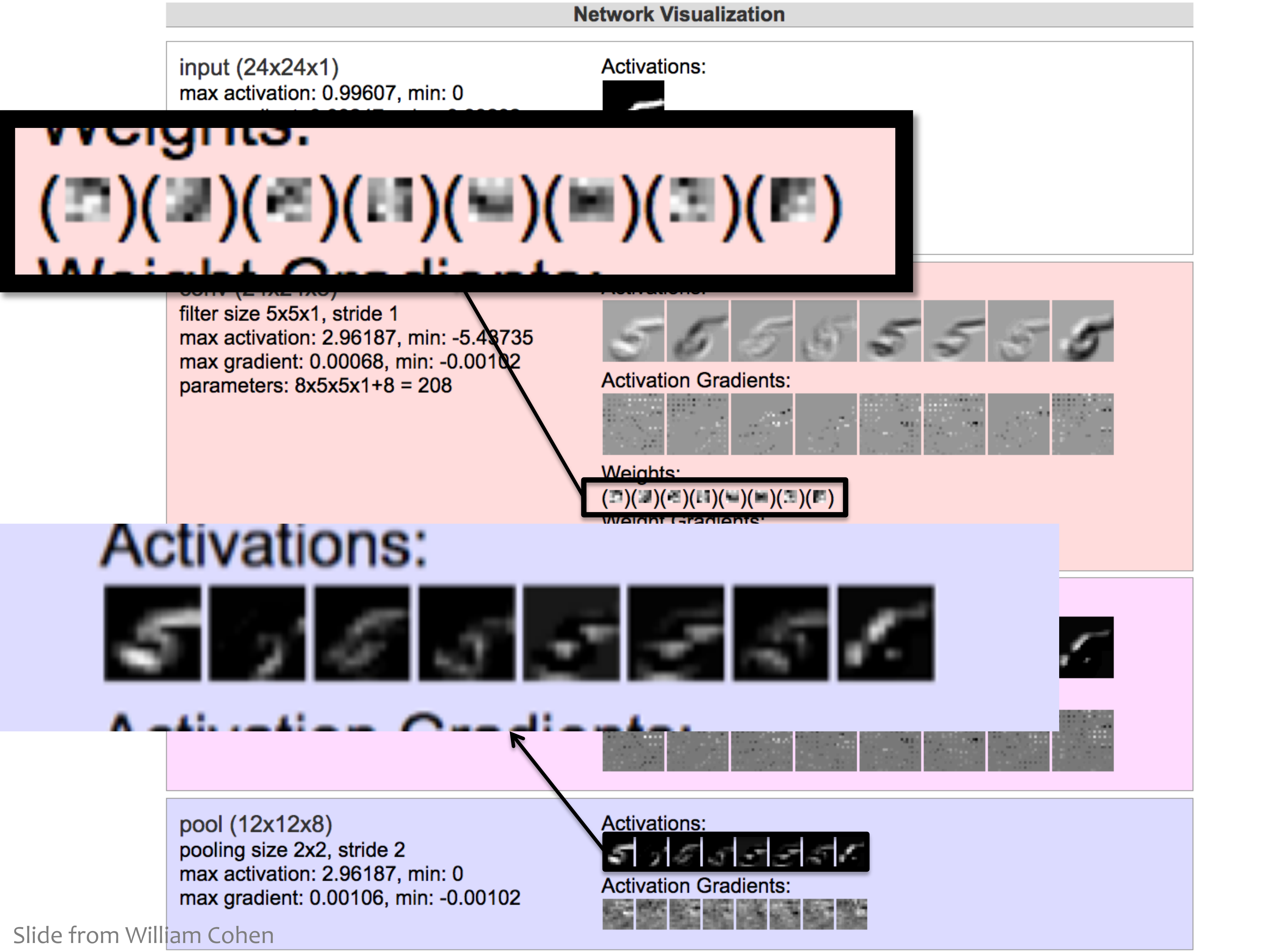


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Another CNN visualization

https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

Slide from William Cohen



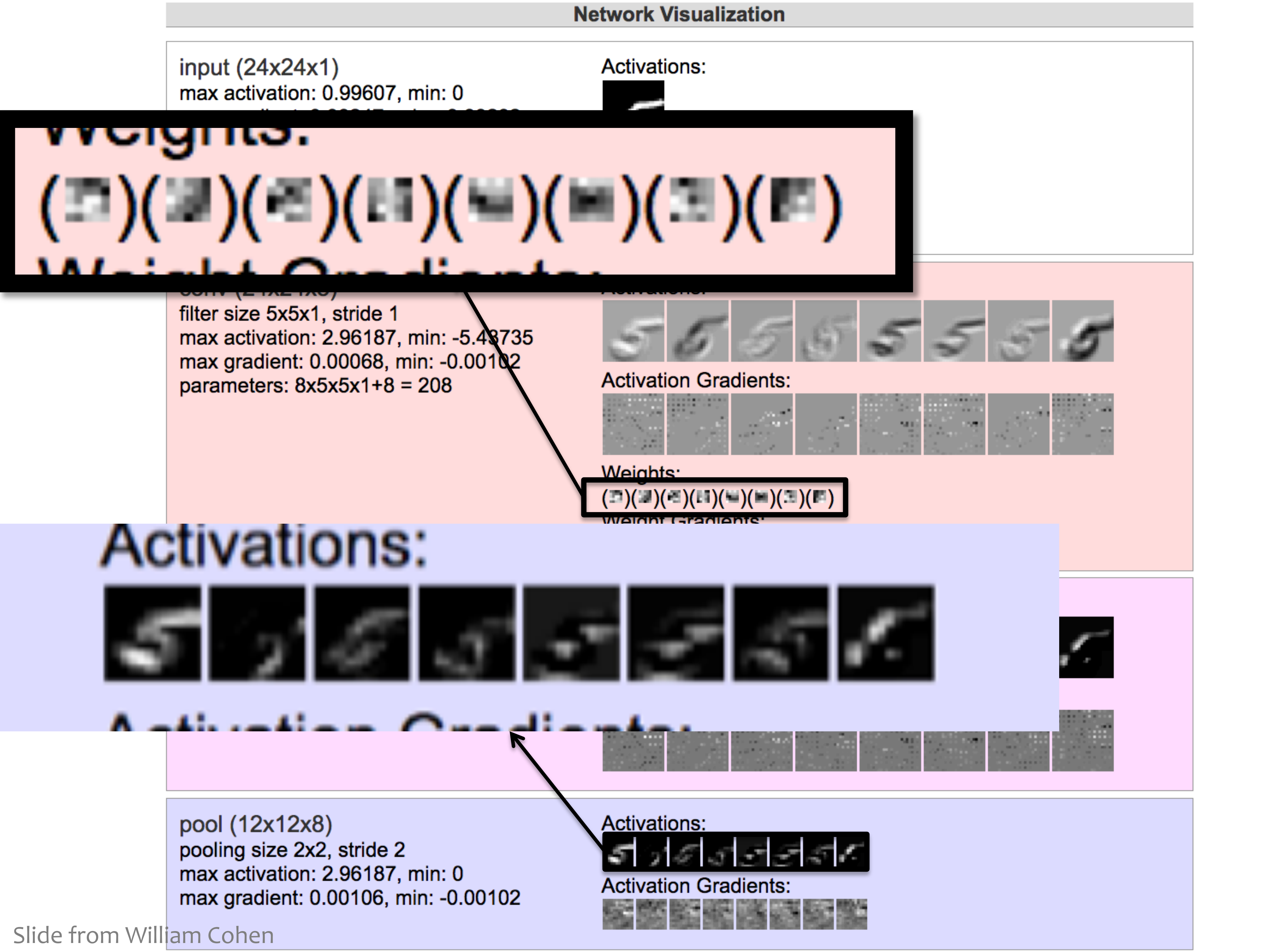
Slide from William Cohen

Why do max-pooling?

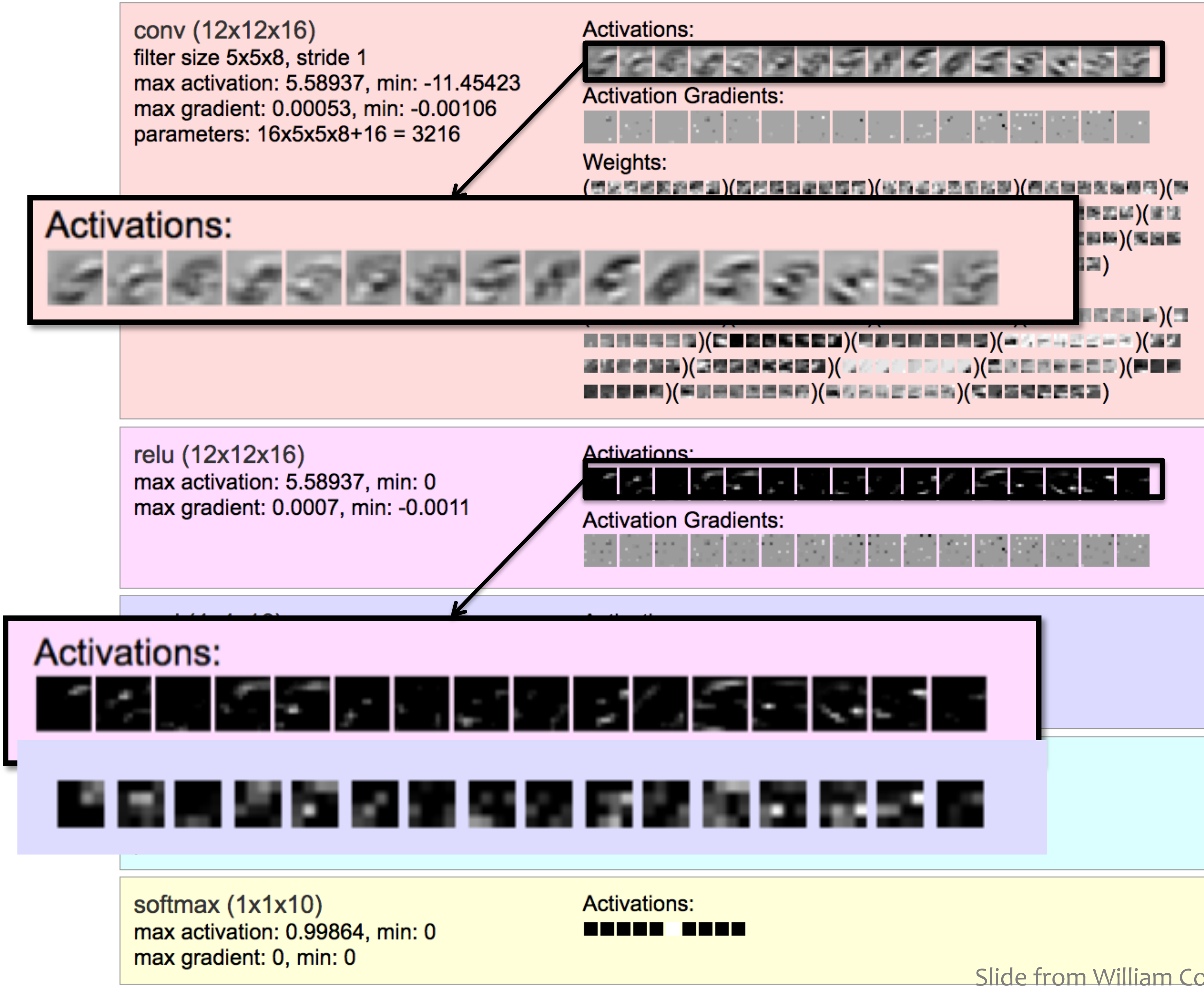
- Saves space
- Reduces overfitting?
- Because I'm going to add *more* convolutions after it!
  – Allows the short-range convolutions to extend over larger subfields of the images
    - So we can spot larger objects
    - Eg, a long horizontal line, or a corner, or …

- At some point the feature maps start to get very sparse and blobby – they are indicators of some semantic property, not a recognizable transformation of the image
- Then just use them as features in a "normal" ANN

Slide from William Cohen



Slide from William Cohen



Slide from William Cohen

Why do max-pooling?

- Saves space
- Reduces overfitting?
- Because I'm going to add *more* convolutions after it!
  - Allows the short-range convolutions to extend over larger subfields of the images
    - So we can spot larger objects
    - Eg, a long horizontal line, or a corner, or …

PROC. OF THE IEEE, NOVEMBER 1998 7

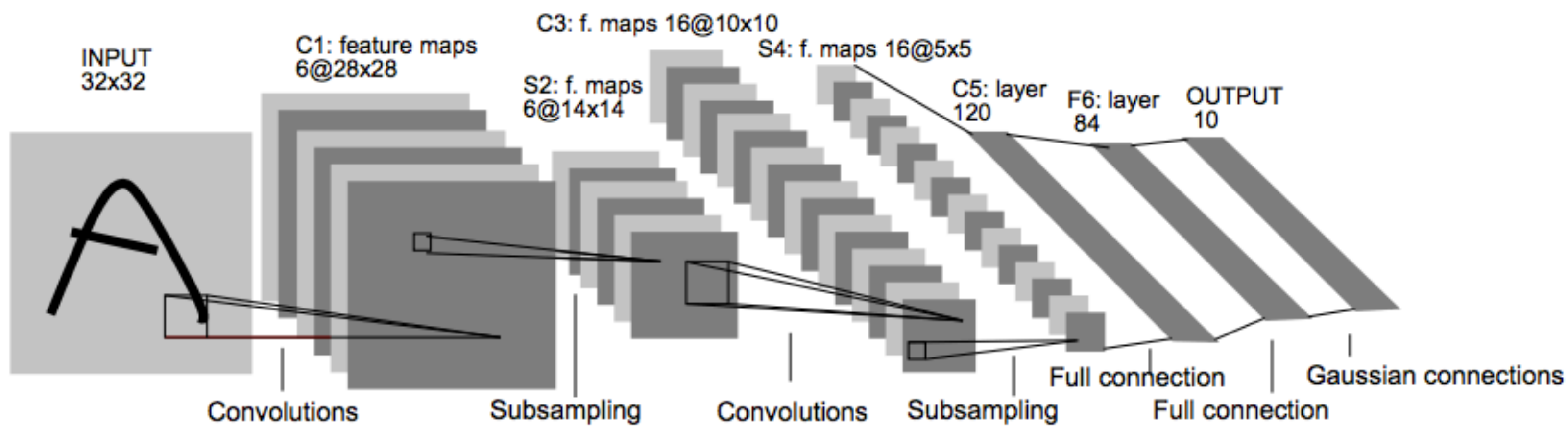
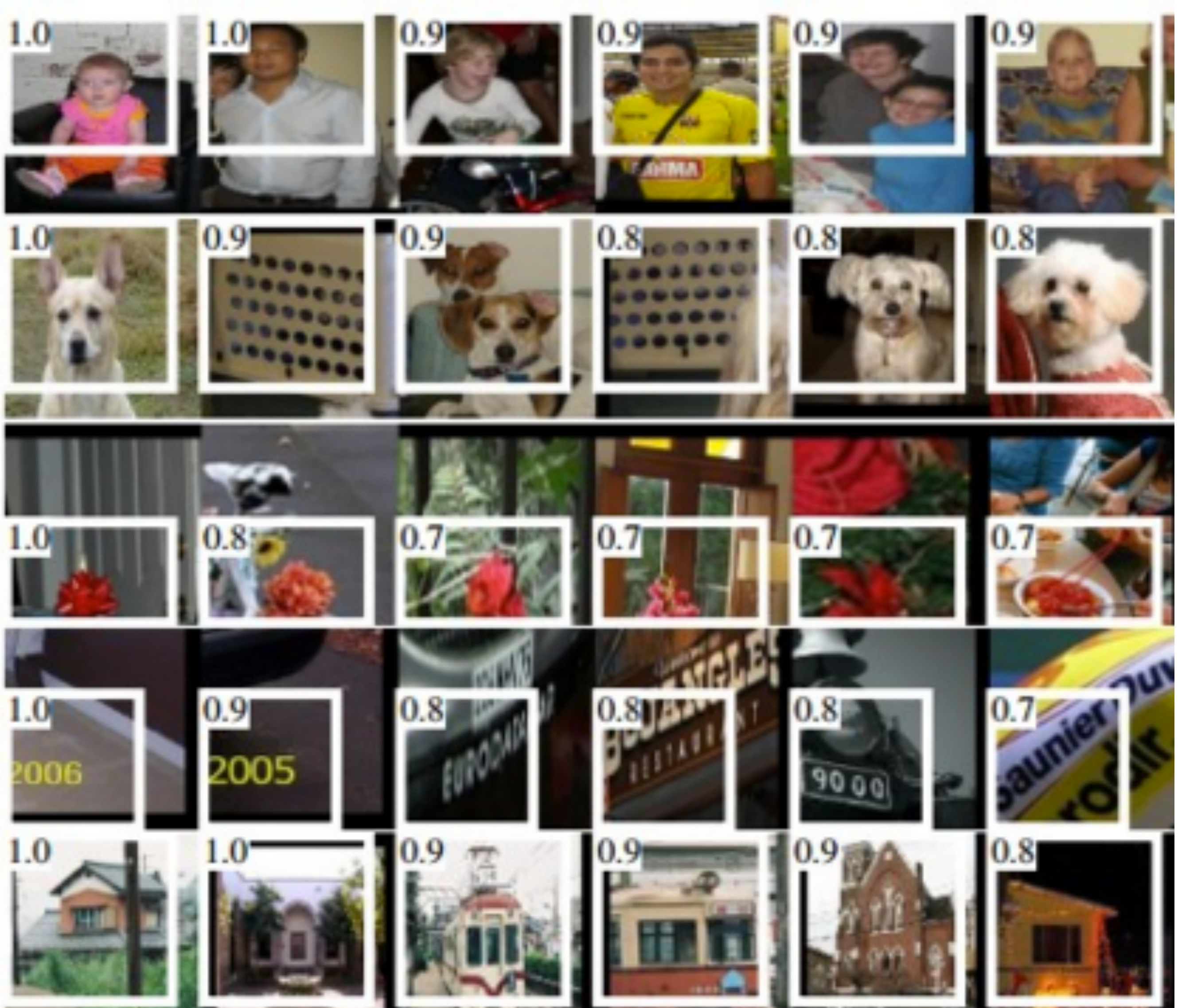


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Slide from William Cohen

Alternating convolution and downsampling



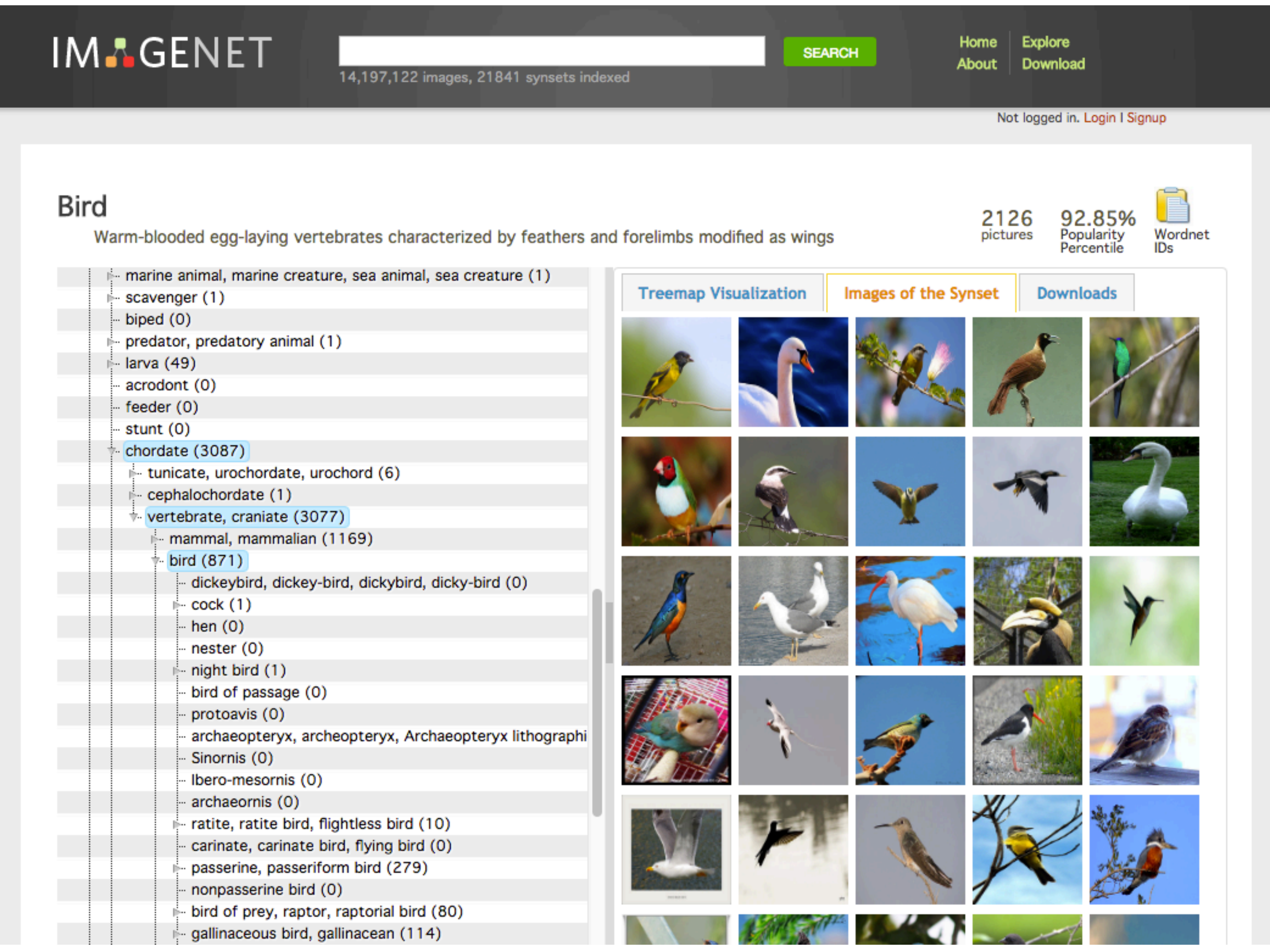
5 layers up

The subfield in a large dataset that gives the strongest output for a neuron

Slide from William Cohen

Example: Image Classification

- ImageNet LSVRC-2011 contest:
  - **Dataset:** 1.2 million labeled images, 1000 classes
  - **Task**: Given a new image, label it with the correct class
  - **Multiclass** classification problem
- Examples from http://image-net.org/


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
Explore
About
Download
Not logged in. Login I Signup
Bird
Warm-blooded egg-laying vertebrates characterized by feathers and forelimbs modified as wings
2126 pictures
92.85% Popularity Percentile
Wordnet IDs
marine animal, marine creature, sea animal, sea creature (1)
scavenger (1)
biped (0)
predator, predatory animal (1)
larva (49)
acrodont (0)
feeder (0)
stunt (0)
chordate (3087)
tunicate, urochordate, urochord (6)
cephalochordate (1)
vertebrate, craniate (3077)
mammal, mammalian (1169)
bird (871)
dickeybird, dickey-bird, dickybird, dicky-bird (0)
cock (1)
hen (0)
nester (0)
night bird (1)
bird of passage (0)
protoavis (0)
archaeopteryx, archeopteryx, Archaeopteryx lithographi
Sinornis (0)
Ibero-mesornis (0)
archaeornis (0)
ratite, ratite bird, flightless bird (10)
carinate, carinate bird, flying bird (0)
passerine, passeriform bird (279)
nonpasserine bird (0)
bird of prey, raptor, raptorial bird (80)
gallinaceous bird, gallinacean (114)
Treemap Visualization
Images of the Synset
Downloads

IMAGENET

SEARCH

14,197,122 images, 21841 synsets indexed

Home | Explore
About | Download

Not logged in. Login | Signup

German iris, Iris kochii

Iris of northern Italy having deep blue-purple flowers; similar to but smaller than Iris germanica

469 pictures | 49.6% Popularity Percentile | Wordnet IDs

- halophyte (0)
- succulent (39)
- cultivar (0)
- cultivated plant (0)
- weed (54)
- evergreen, evergreen plant (0)
- deciduous plant (0)
- vine (272)
- creeper (0)
- woody plant, ligneous plant (1868)
- geophyte (0)
- desert plant, xerophyte, xerophytic plant, xerophile, xerophilc
- mesophyte, mesophytic plant (0)
- aquatic plant, water plant, hydrophyte, hydrophytic plant (11
- tuberous plant (0)
- bulbous plant (179)
  - iridaceous plant (27)
    - iris, flag, fleur-de-lis, sword lily (19)
      - bearded iris (4)
        - Florentine iris, orris, Iris germanica florentina, Iris
        - German iris, Iris germanica (0)
        - German iris, Iris kochii (0)
        - Dalmatian iris, Iris pallida (0)
      - beardless iris (4)
      - bulbous iris (0)
      - dwarf iris, Iris cristata (0)
      - stinking iris, gladdon, gladdon iris, stinking gladwyn,
      - Persian iris, Iris persica (0)
      - yellow iris, yellow flag, yellow water flag, Iris pseuda
      - dwarf iris, vernal iris, Iris verna (0)
      - blue flag, Iris versicolor (0)

Treemap Visualization | Images of the Synset | Downloads

IMAGENET

SEARCH

14,197,122 images, 21841 synsets indexed

Home | Explore
About | Download

Not logged in. Login | Signup

Court, courtyard

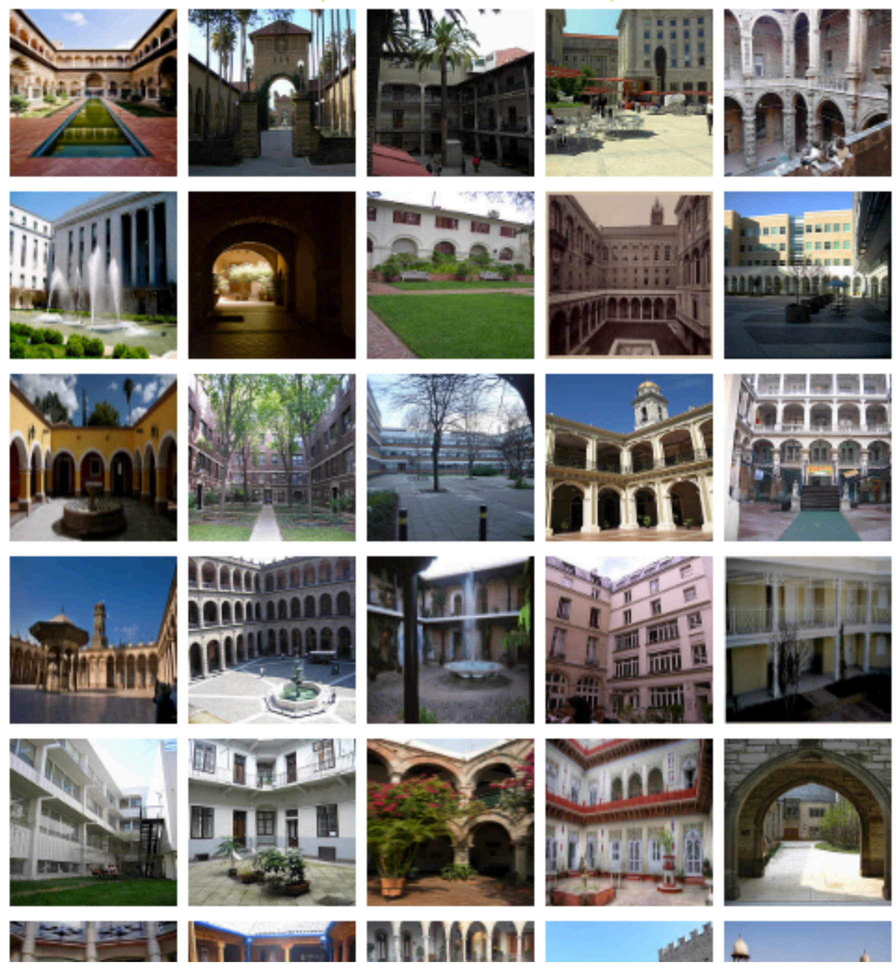
An area wholly or partly surrounded by walls or buildings; "the house was built around an inner court"

165 pictures | 92.61% Popularity Percentile | Wordnet IDs

Numbers in brackets: (the number of synsets in the subtree).

- ImageNet 2011 Fall Release (32326)
  - plant, flora, plant life (4486)
  - geological formation, formation (175)
  - natural object (1112)
  - sport, athletics (176)
  - artifact, artefact (10504)
    - instrumentality, instrumentation (5494)
    - structure, construction (1405)
      - airdock, hangar, repair shed (0)
      - altar (1)
      - arcade, colonnade (1)
      - arch (31)
      - area (344)
        - aisle (0)
        - auditorium (1)
        - baggage claim (0)
        - box (1)
        - breakfast area, breakfast nook (0)
        - bullpen (0)
        - chancel, sanctuary, bema (0)
        - choir (0)
        - corner, nook (2)
        - court, courtyard (6)
          - atrium (0)
          - bailey (0)
          - cloister (0)
          - food court (0)
          - forecourt (0)
          - parvis (0)

Treemap Visualization | Images of the Synset | Downloads



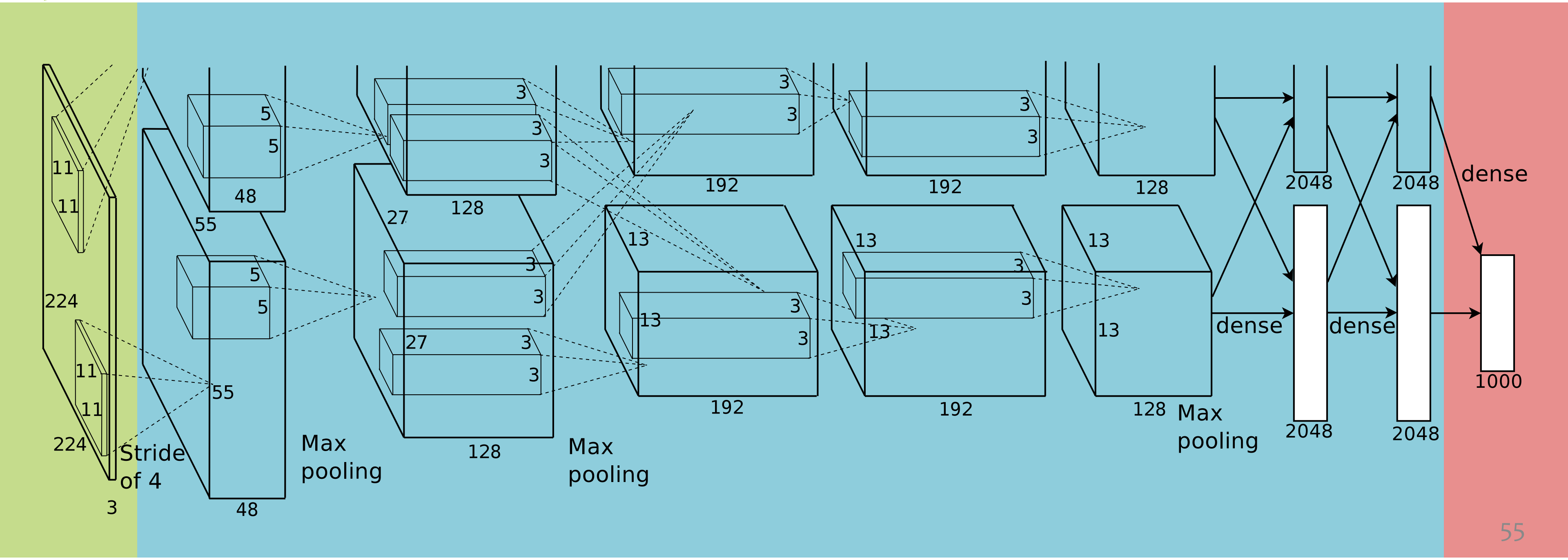
Example: Image Classification

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2012)
15.3% error on ImageNet LSVRC-2012 contest

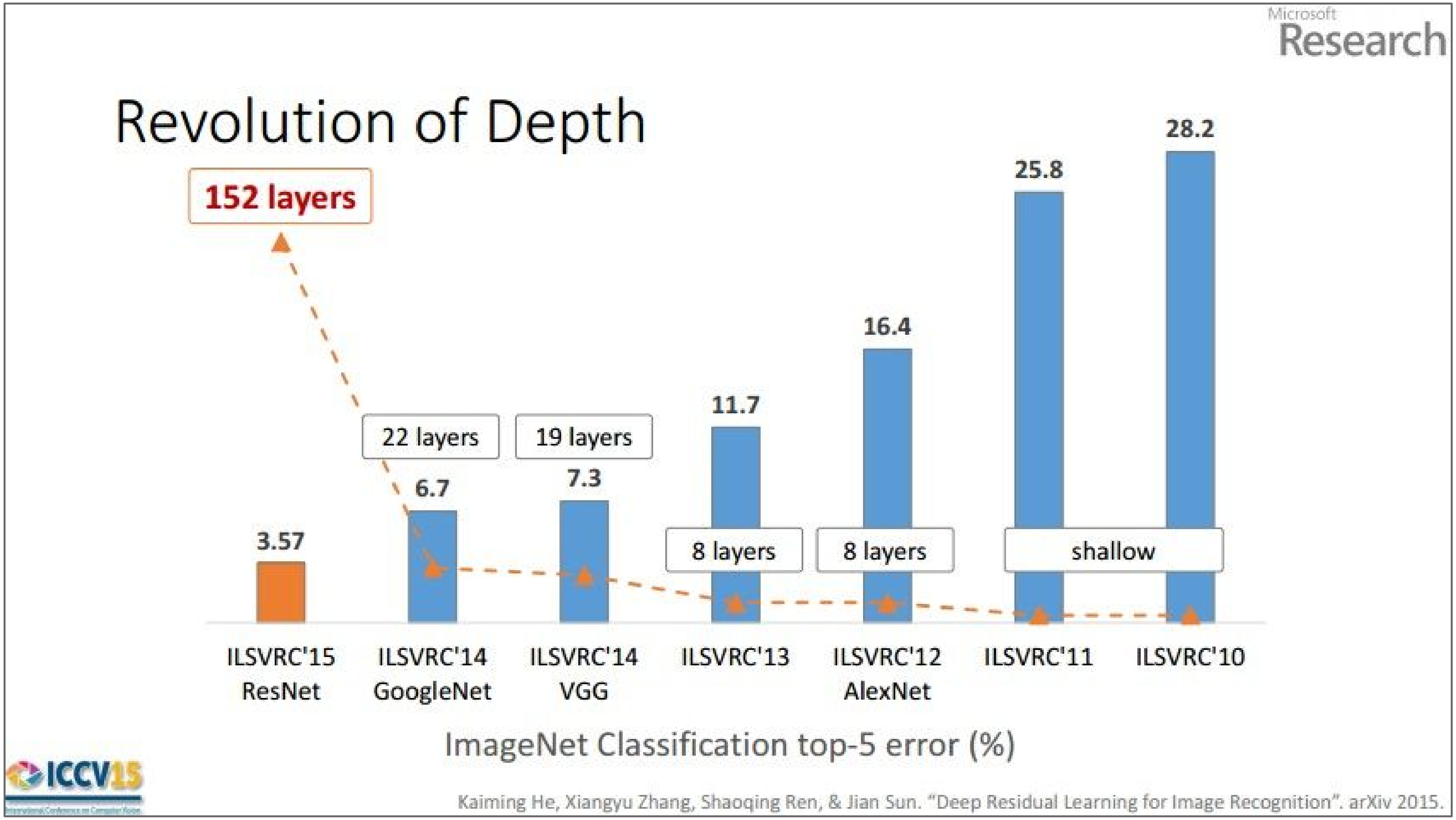
Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



CNNs for Image Recognition

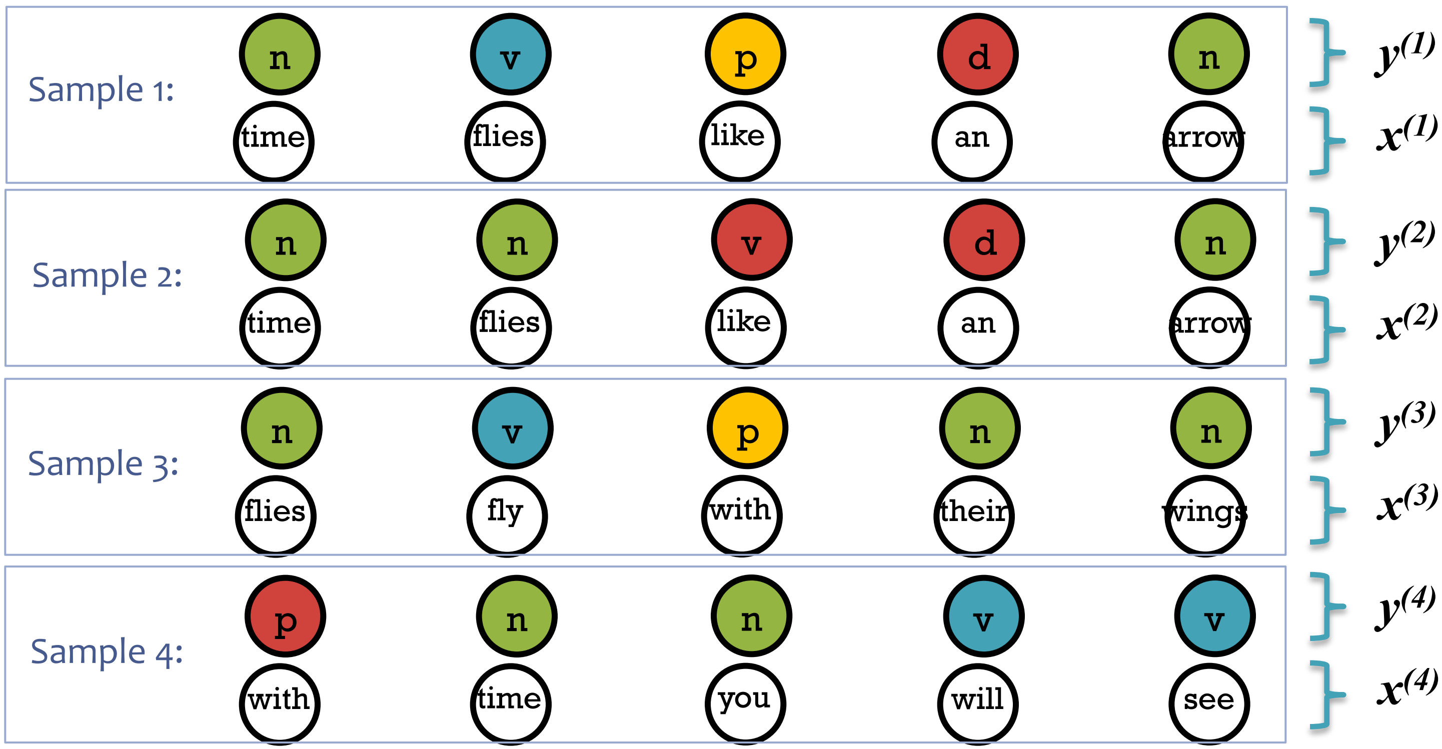


Slide from Kaiming He

RECURRENT NEURAL NETWORKS

Dataset for Supervised Part-of-Speech (POS) Tagging

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$



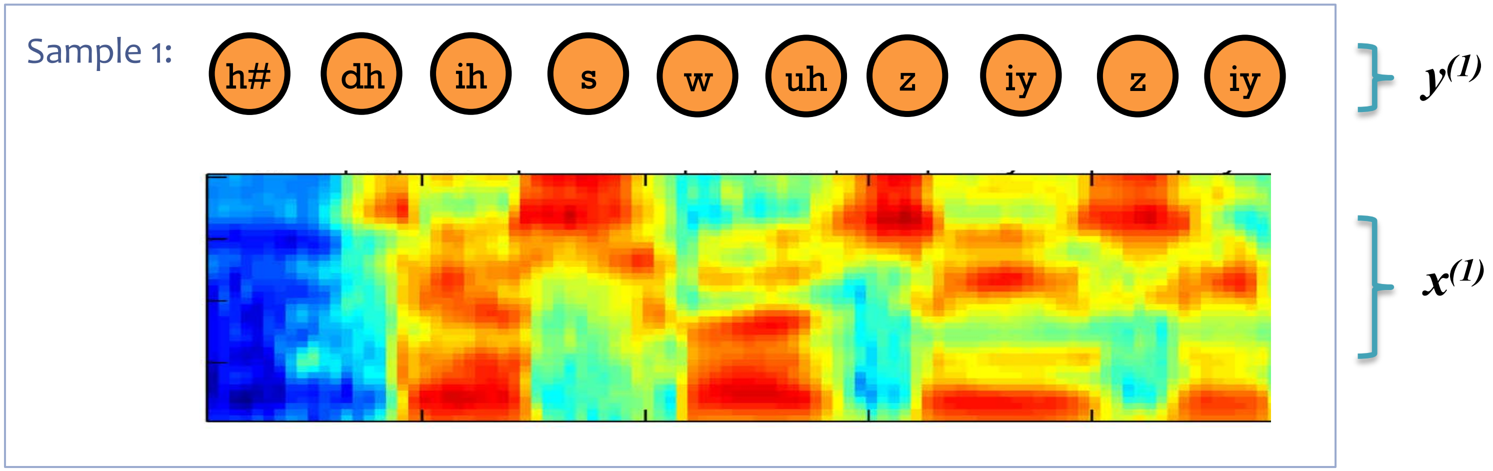
Dataset for Supervised Handwriting Recognition

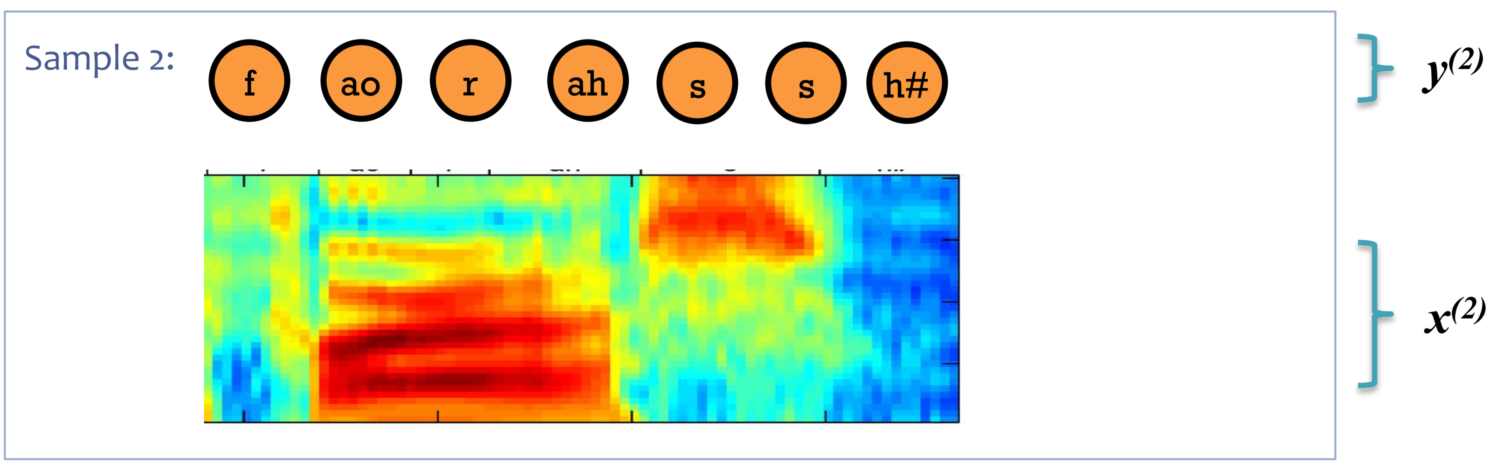
Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$

Figures from (Chatzis & Demiris, 2013)

Dataset for Supervised Phoneme (Speech) Recognition

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$

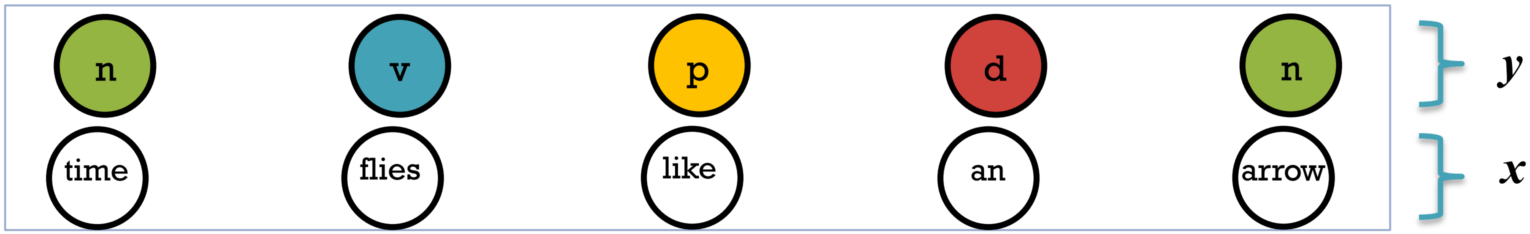




Figures from (Jansen & Niyogi, 2013)

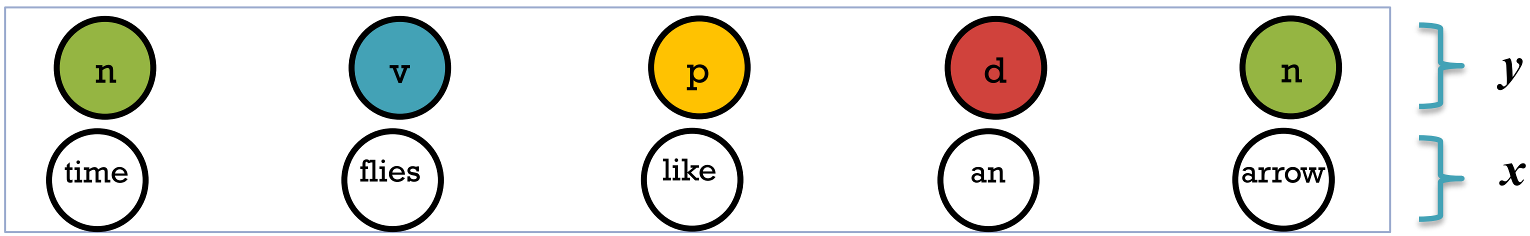
Time Series Data

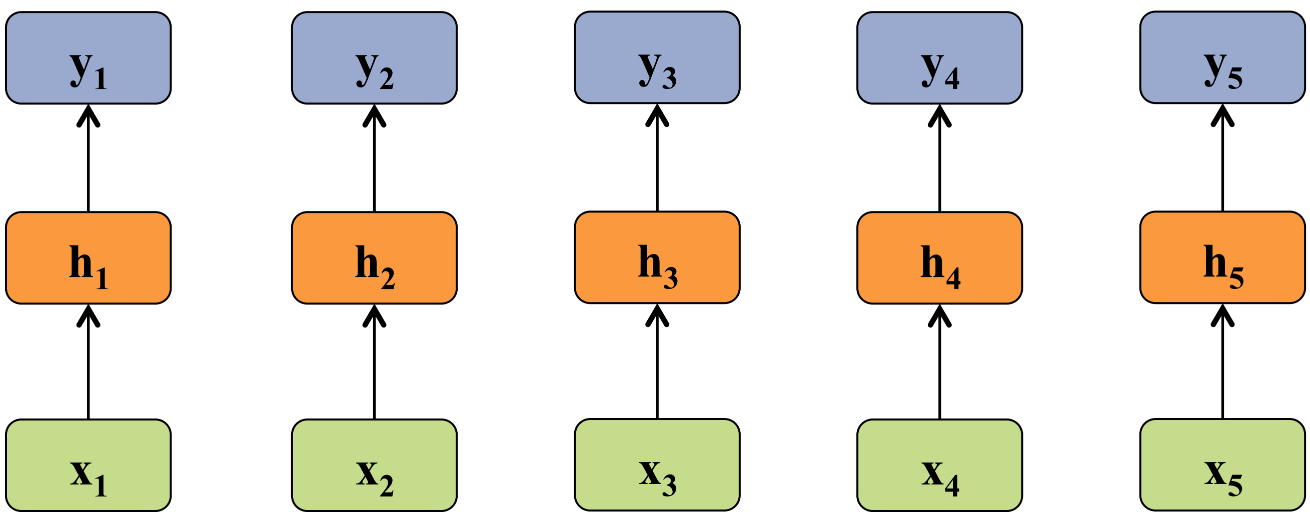
Question 1: How could we apply the neural networks we've seen so far (which expect **fixed size input/output**) to a prediction task with **variable length input/output**?



Time Series Data

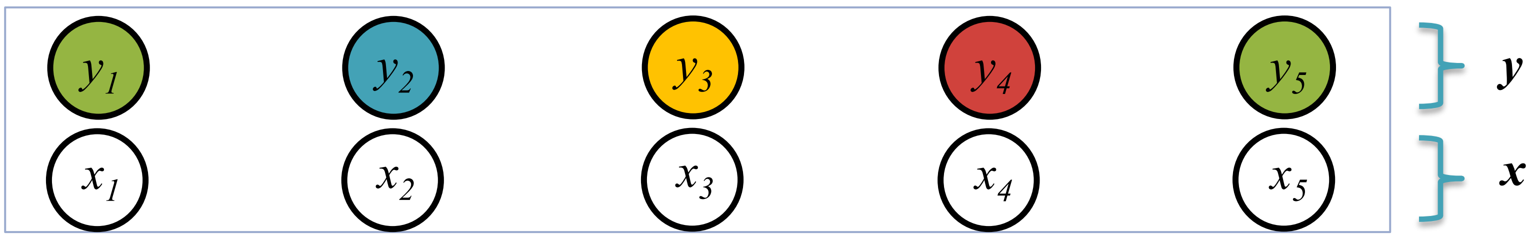
Question 1: How could we apply the neural networks we've seen so far (which expect **fixed size input/output**) to a prediction task with **variable length input/output**?





Time Series Data

Question 2: How could we incorporate context (e.g. words to the left/right, or tags to the left/right) into our solution?



Multiple Choice:

Working left-to-right, use features of…

	x_{i-1}	x_i	x_{i+1}	y_{i-1}	y_i	y_{i+1}
A	✓					
B				✓		
C	✓			✓		
D	✓			✓	✓	✓
E	✓	✓		✓	✓	✓
F	✓	✓	✓	✓		
G	✓	✓	✓	✓	✓	
H	✓	✓	✓	✓	✓	✓

Recurrent Neural Networks (RNNs)

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\mathbf{h} = (h_1, h_2, \ldots, h_T), h_i \in \mathcal{R}^J$

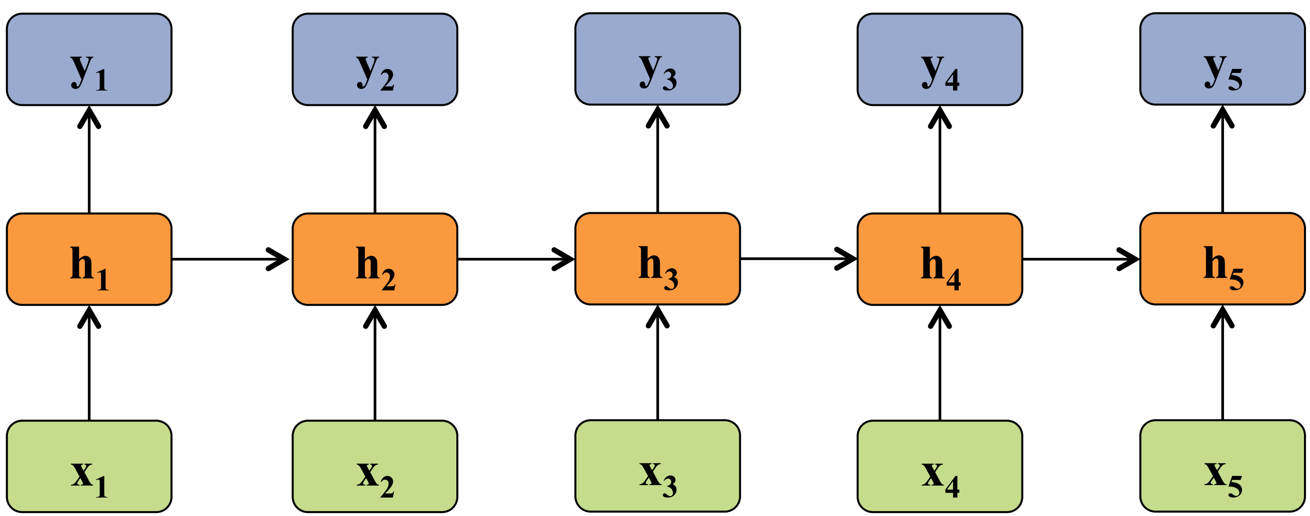
outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

nonlinearity: $\mathcal{H}$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$

$$y_t = W_{hy}h_t + b_y$$



Recurrent Neural Networks (RNNs)

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\mathbf{h} = (h_1, h_2, \ldots, h_T), h_i \in \mathcal{R}^J$

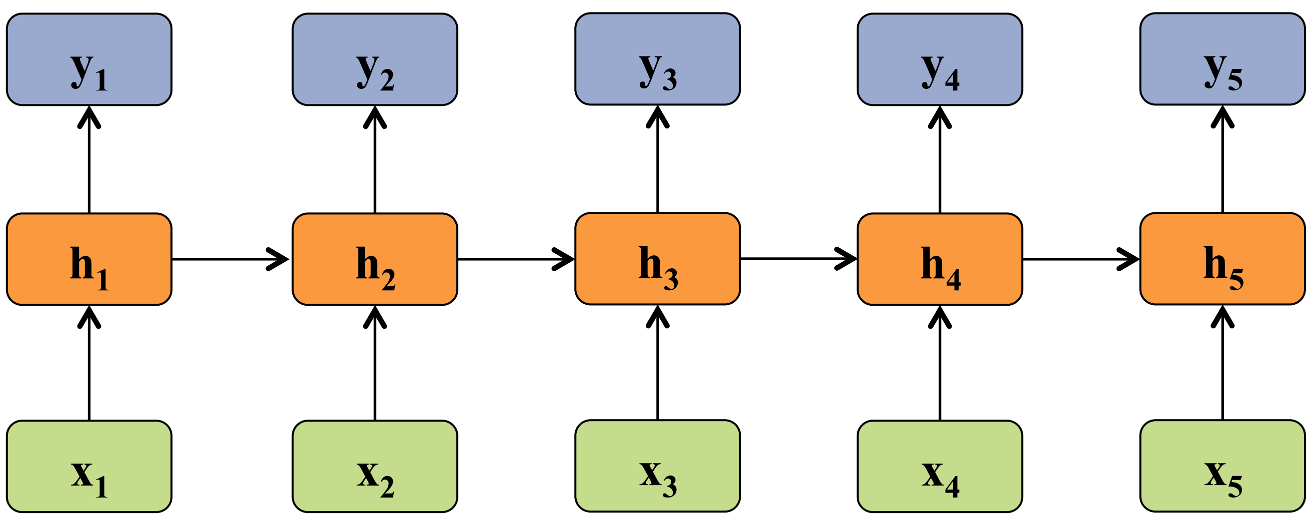
outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

nonlinearity: $\mathcal{H}$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$

$$y_t = W_{hy}h_t + b_y$$



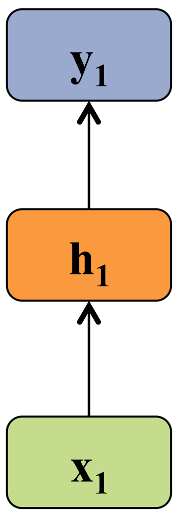
Recurrent Neural Networks (RNNs)

$$\begin{aligned}
\text{inputs: } & \mathbf{x} = (x_1, x_2, \dots, x_T), x_i \in \mathcal{R}^I \\
\text{hidden units: } & \mathbf{h} = (h_1, h_2, \dots, h_T), h_i \in \mathcal{R}^J \\
\text{outputs: } & \mathbf{y} = (y_1, y_2, \dots, y_T), y_i \in \mathcal{R}^K \\
\text{nonlinearity: } & \mathcal{H}
\end{aligned}$$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$

$$y_t = W_{hy}h_t + b_y$$



- If $T=1$, then we have a standard feed-forward **neural net with one hidden layer**
- All of the deep nets from last lecture required **fixed size inputs/outputs**

Recurrent Neural Networks (RNNs)

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\mathbf{h} = (h_1, h_2, \ldots, h_T), h_i \in \mathcal{R}^J$

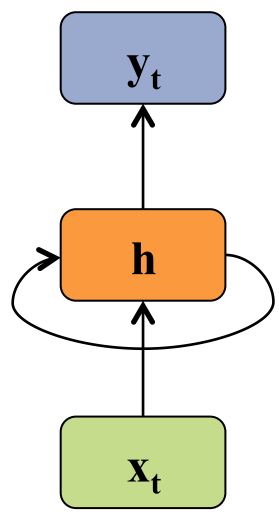
outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

nonlinearity: $\mathcal{H}$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh} x_t + W_{hh} h_{t-1} + b_h\right)$$

$$y_t = W_{hy} h_t + b_y$$



Recurrent Neural Networks (RNNs)

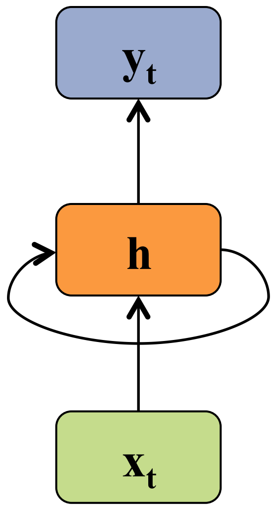
$$\begin{aligned} \text{inputs: } & \mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I \\ \text{hidden units: } & \mathbf{h} = (h_1, h_2, \ldots, h_T), h_i \in \mathcal{R}^J \\ \text{outputs: } & \mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K \\ \text{nonlinearity: } & \mathcal{H} \end{aligned}$$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh} x_t + W_{hh} h_{t-1} + b_h\right)$$

$$y_t = W_{hy} h_t + b_y$$

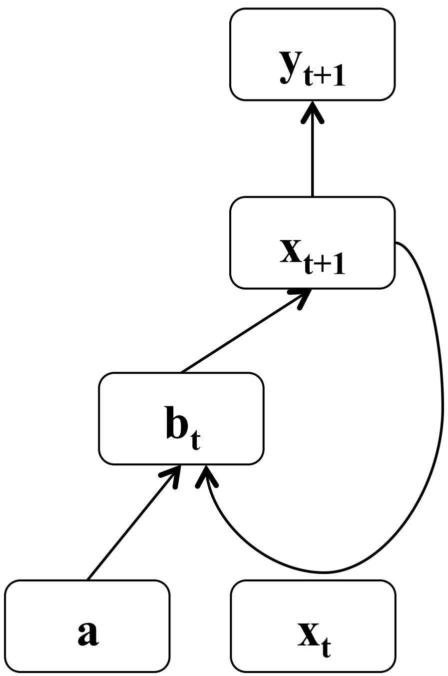
- By unrolling the RNN through time, we can **share parameters** and accommodate **arbitrary length** input/output pairs
- Applications: **time-series data** such as sentences, speech, stock-market, signal data, etc.



Background: Backprop through time

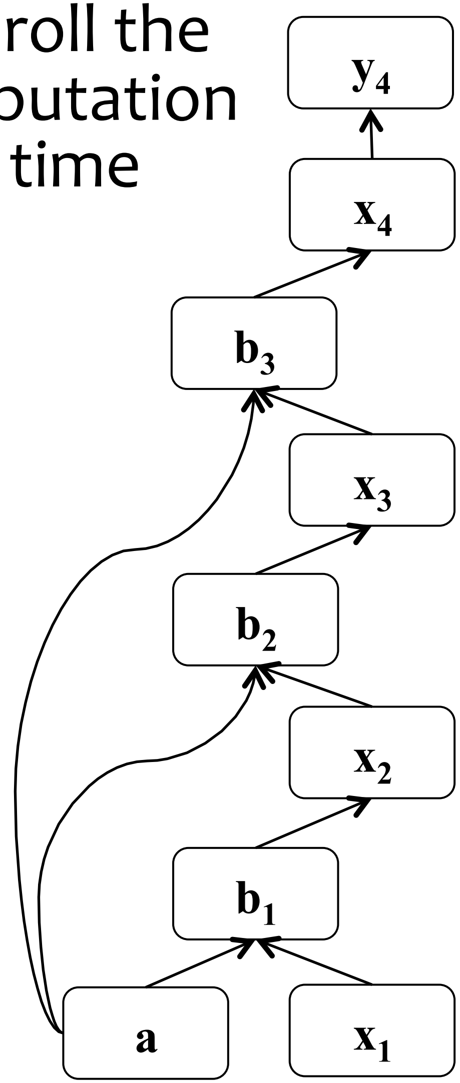
(Robinson & Fallside, 1987)
(Werbos, 1988)
(Mozer, 1995)

Recurrent neural network:



BPTT:

1. Unroll the computation over time



2. Run backprop through the resulting feed-forward network

Bidirectional RNN

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$

outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

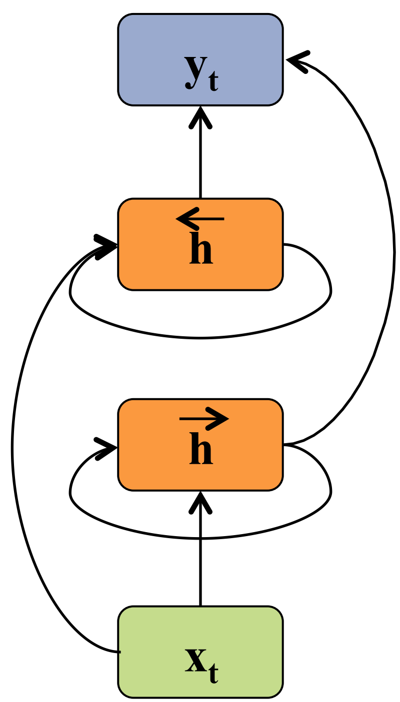
nonlinearity: $\mathcal{H}$

Recursive Definition:

$$\overrightarrow{h}_t = \mathcal{H}\left(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right)$$

$$\overleftarrow{h}_t = \mathcal{H}\left(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right)$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y$$



Bidirectional RNN

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$

outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

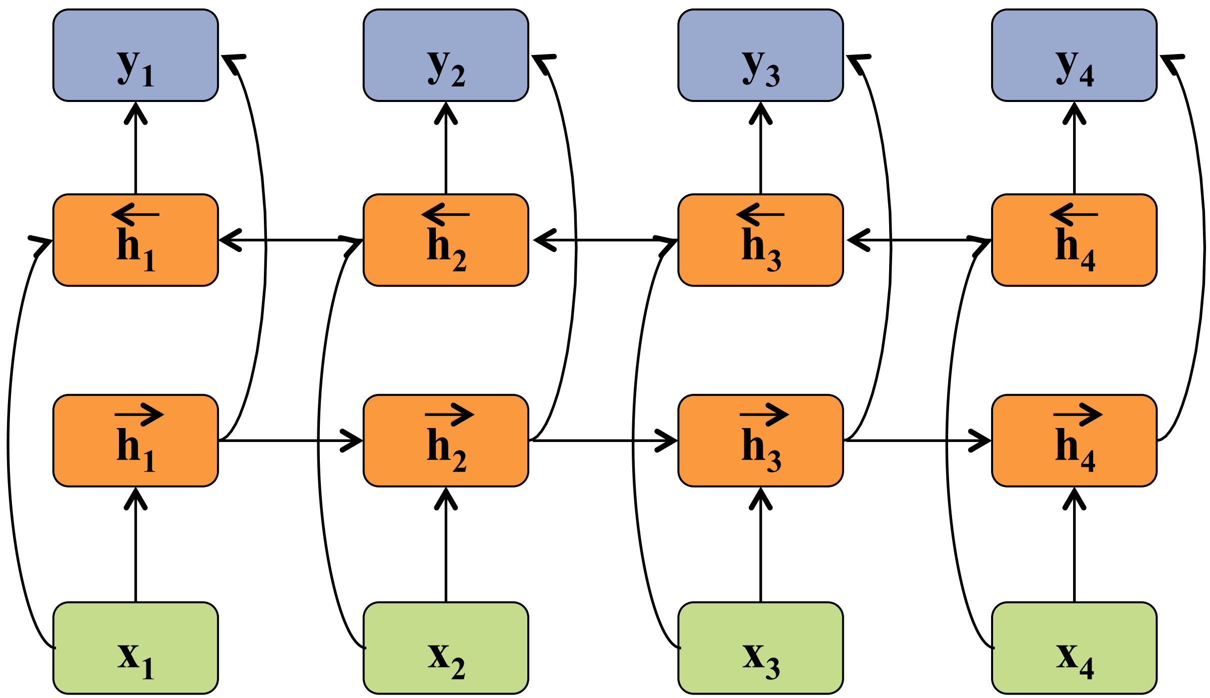
nonlinearity: $\mathcal{H}$

Recursive Definition:

$$\overrightarrow{h}_t = \mathcal{H}\left(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right)$$

$$\overleftarrow{h}_t = \mathcal{H}\left(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right)$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y$$



Bidirectional RNN

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$

outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

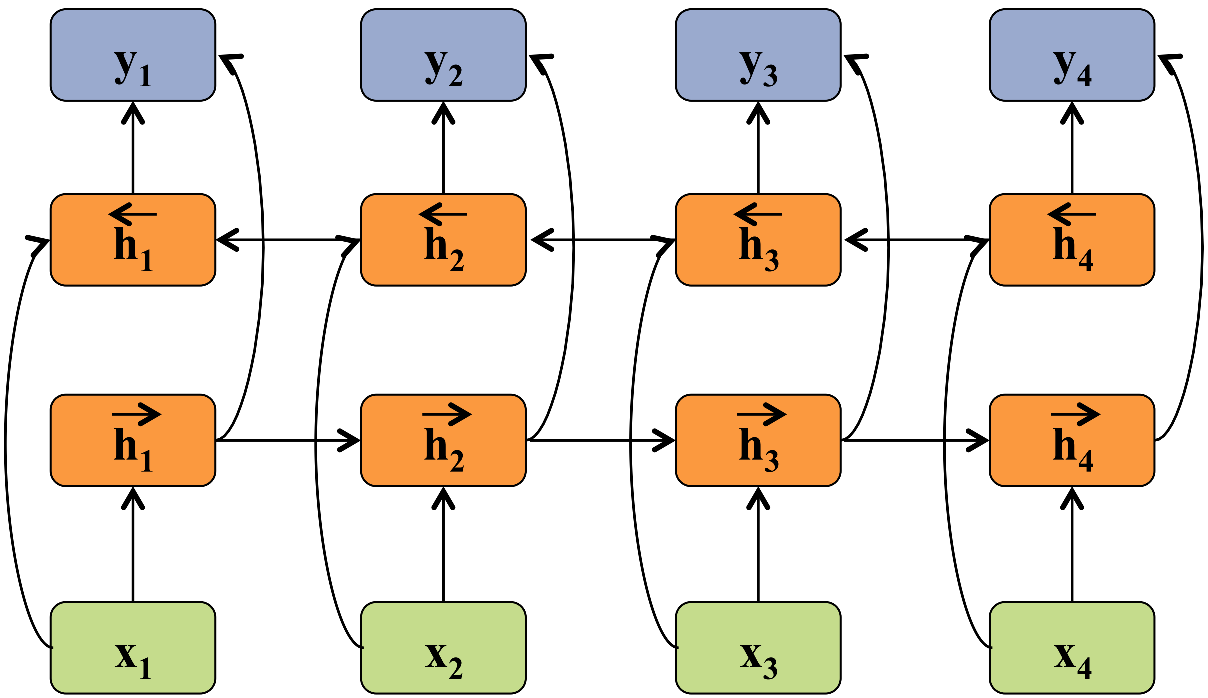
nonlinearity: $\mathcal{H}$

Recursive Definition:

$$\overrightarrow{h}_t = \mathcal{H}\left(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right)$$

$$\overleftarrow{h}_t = \mathcal{H}\left(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right)$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y$$



Bidirectional RNN

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$

outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

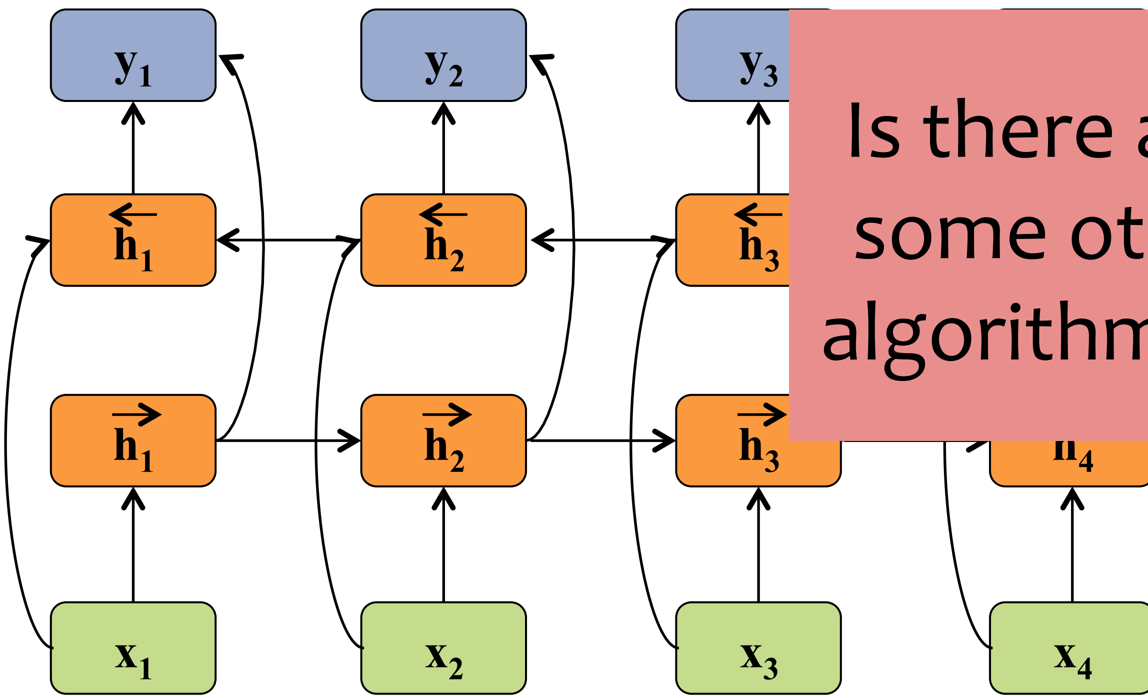
nonlinearity: $\mathcal{H}$

Recursive Definition:

$$\overrightarrow{h}_t = \mathcal{H}\left(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right)$$

$$\overleftarrow{h}_t = \mathcal{H}\left(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right)$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y$$



Is there an analogy to some other recursive algorithm(s) we know?

Deep RNNs

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

nonlinearity: $\mathcal{H}$

Recursive Definition:

$$h_t^n = \mathcal{H}\left(W_{h^{n-1}h^n} h_t^{n-1} + W_{h^n h^n} h_{t-1}^n + b_h^n\right)$$

$$y_t = W_{h^N y} h_t^N + b_y$$

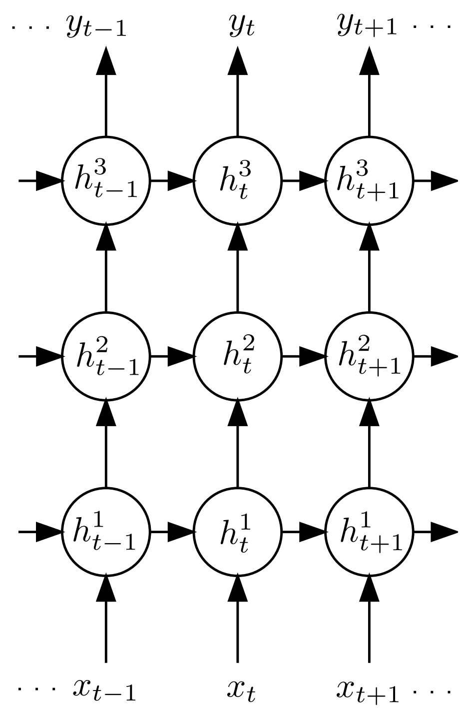


Figure from (Graves et al., 2013)

Deep Bidirectional RNNs

$$\text{inputs: } \mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$$
$$\text{outputs: } \mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$$
$$\text{nonlinearity: } \mathcal{H}$$

- Notice that the upper level hidden units have input from **two previous layers** (i.e. wider input)
- Likewise for the output layer
- What analogy can we draw to DNNs, DBNs, DBMs?

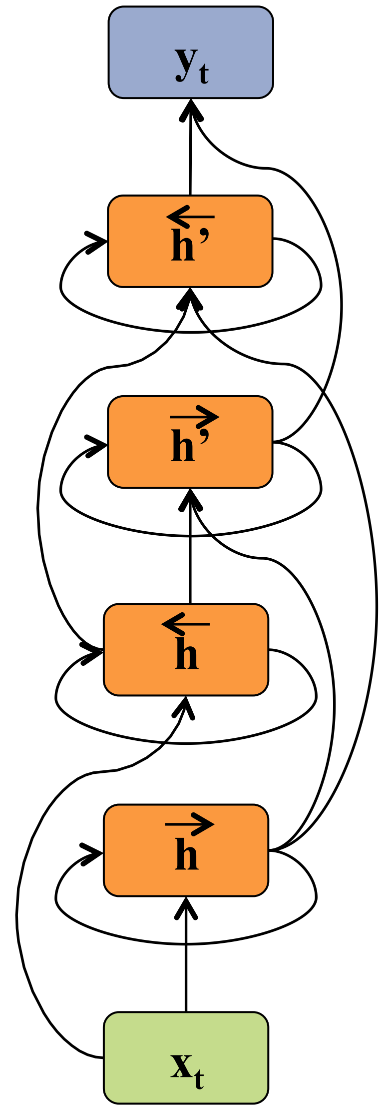
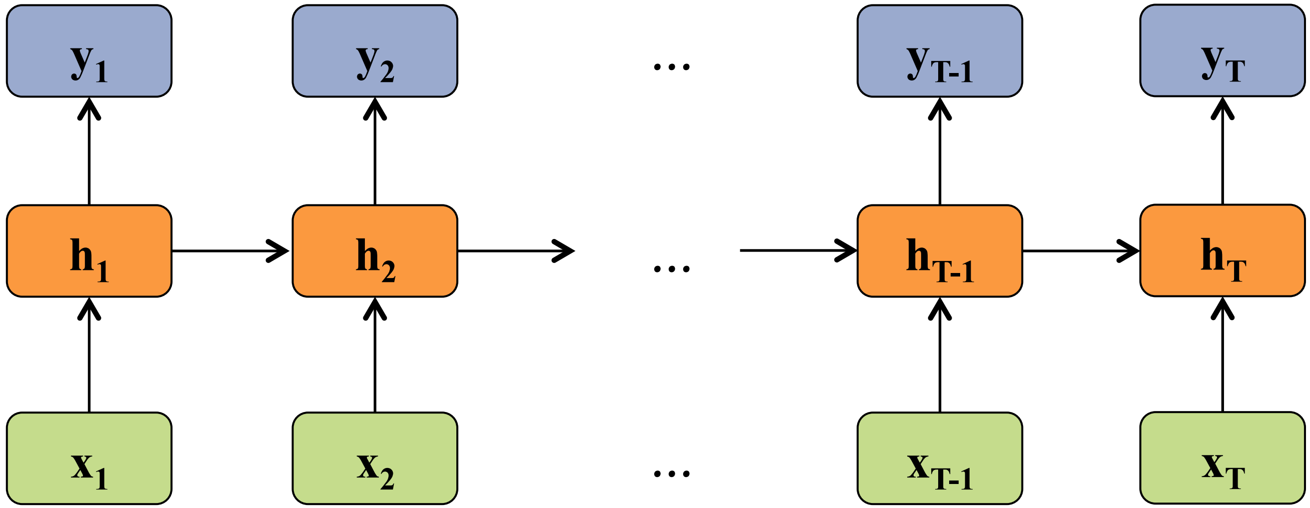


Figure from (Graves et al., 2013)

Long Short-Term Memory (LSTM)

Motivation:

- Standard RNNs have trouble learning long distance dependencies
- LSTMs combat this issue



Long Short-Term Memory (LSTM)

Motivation:

- Vanishing gradient problem for Standard RNNs
- Figure shows sensitivity (darker = more sensitive) to the input at time t=1

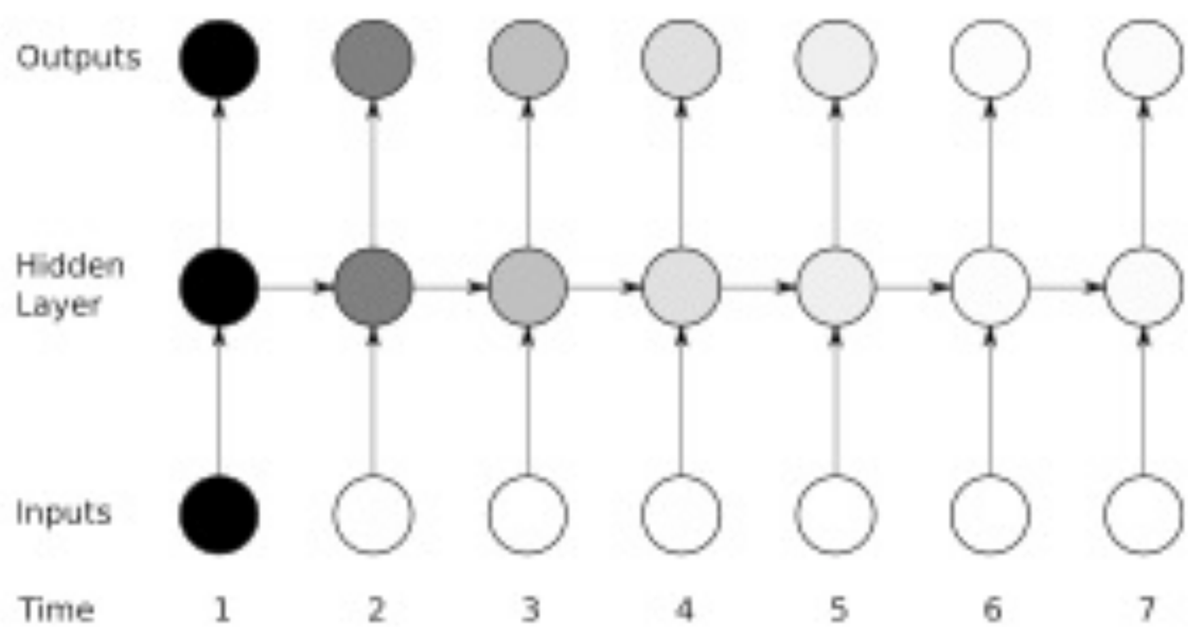


Figure from (Graves, 2012)

Long Short-Term Memory (LSTM)

Motivation:

- LSTM units have a rich internal structure
- The various "gates" determine the propagation of information and can choose to "remember" or "forget" information

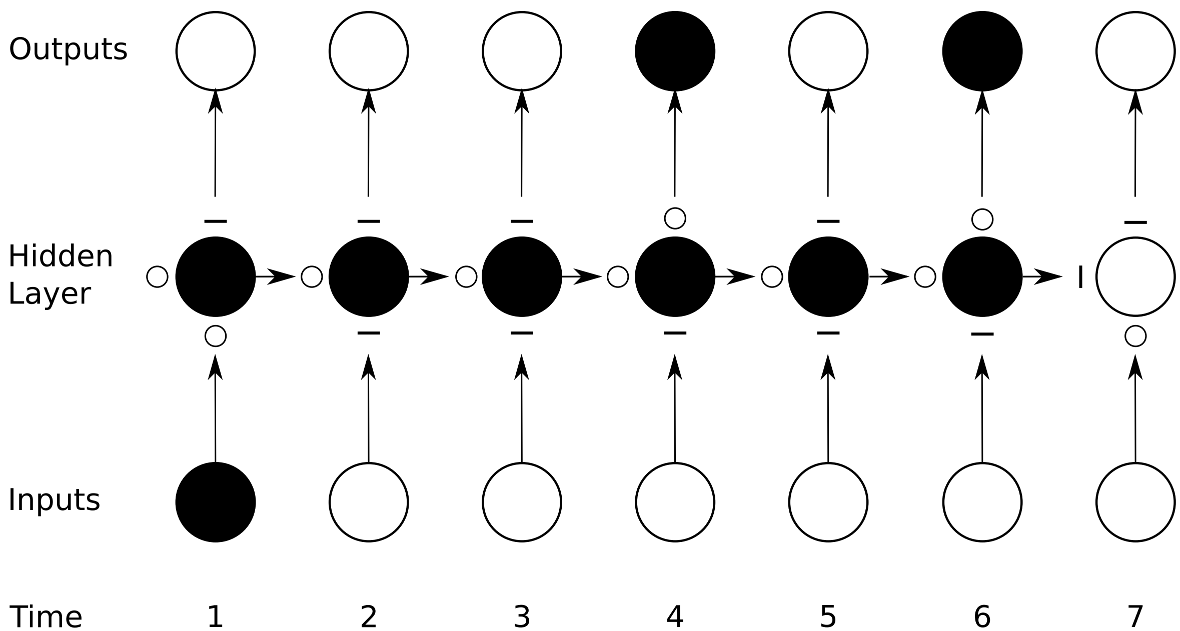
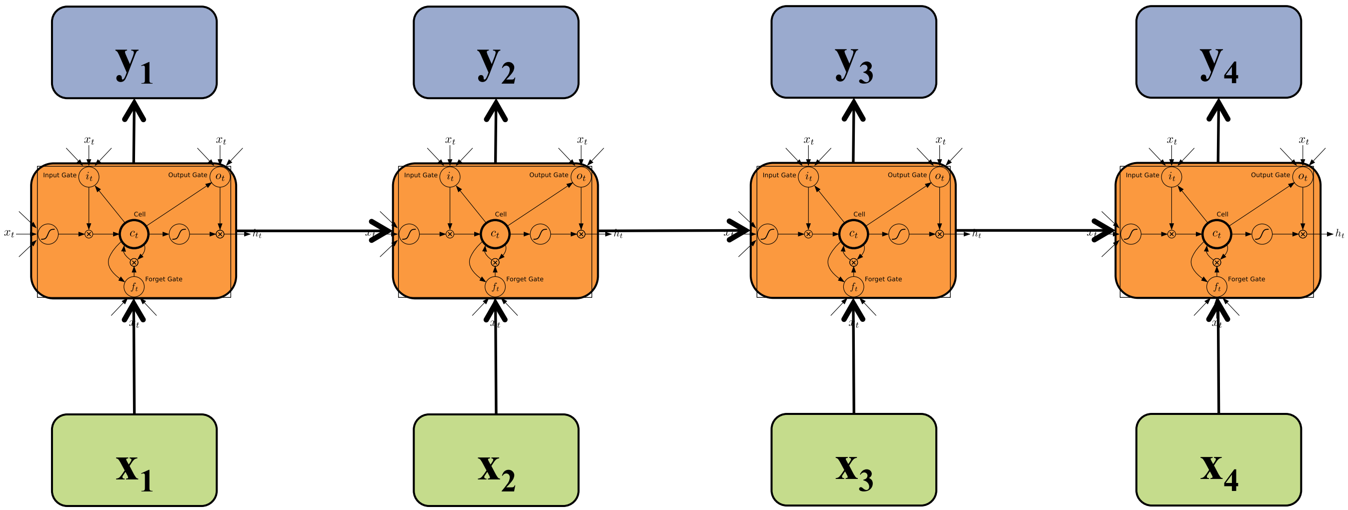


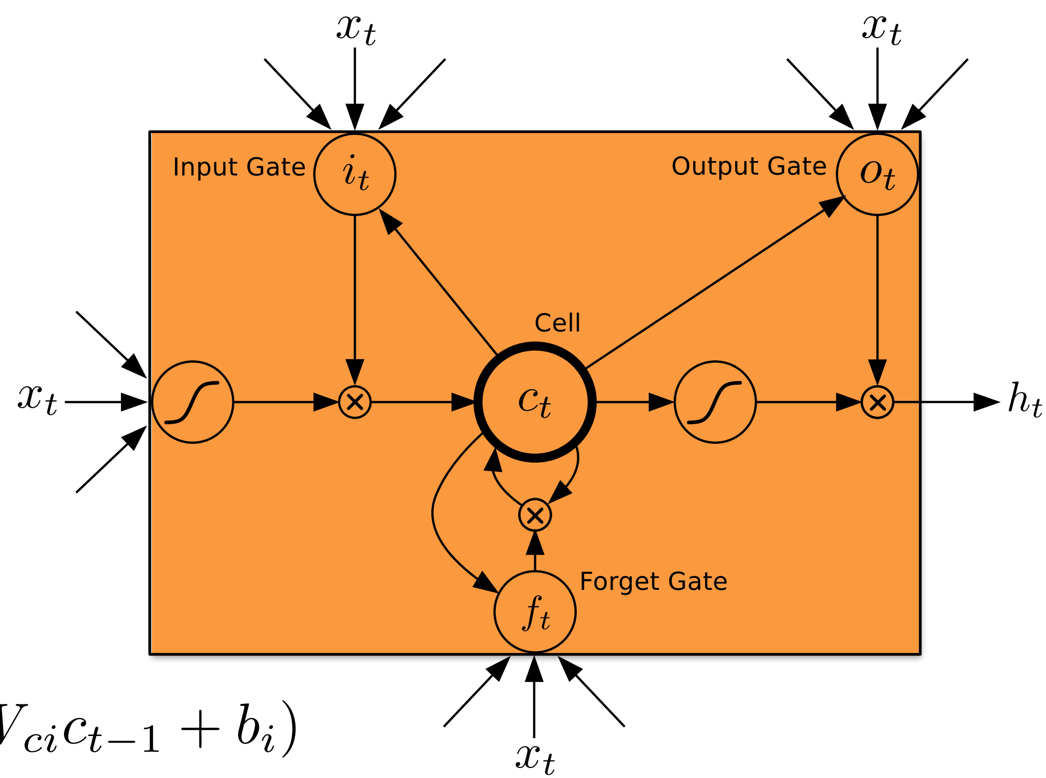
Figure from (Graves, 2012)

Long Short-Term Memory (LSTM)



Long Short-Term Memory (LSTM)

- **Input gate:** masks out the standard RNN inputs
- **Forget gate:** masks out the previous cell
- **Cell:** stores the input/ forget mixture
- **Output gate:** masks out the values of the next hidden



$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right)$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)$$

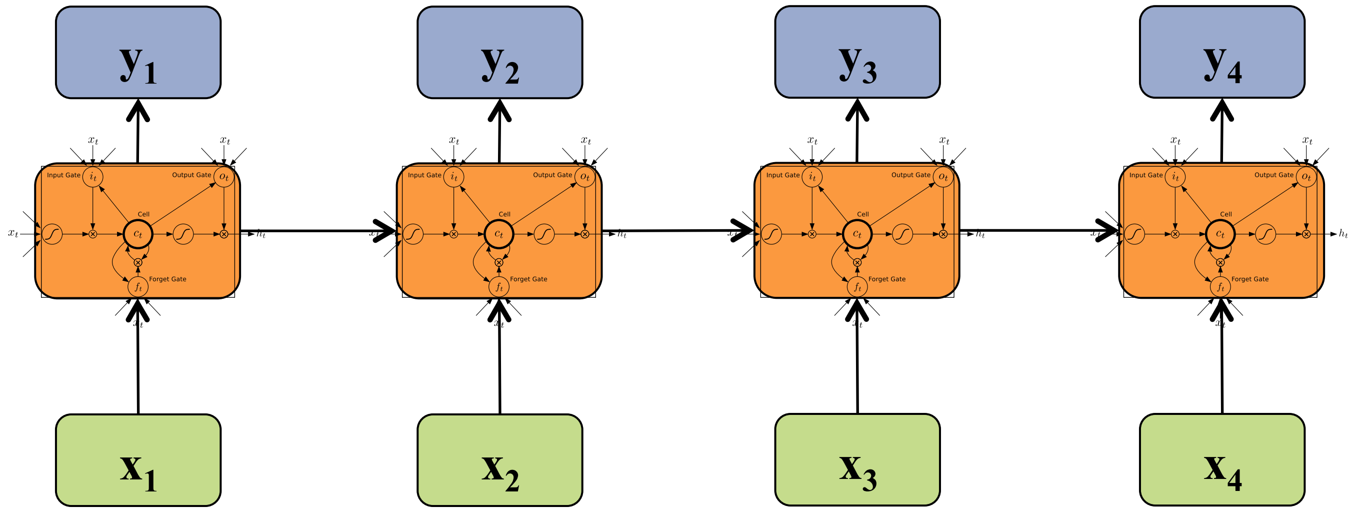
$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right)$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right)$$

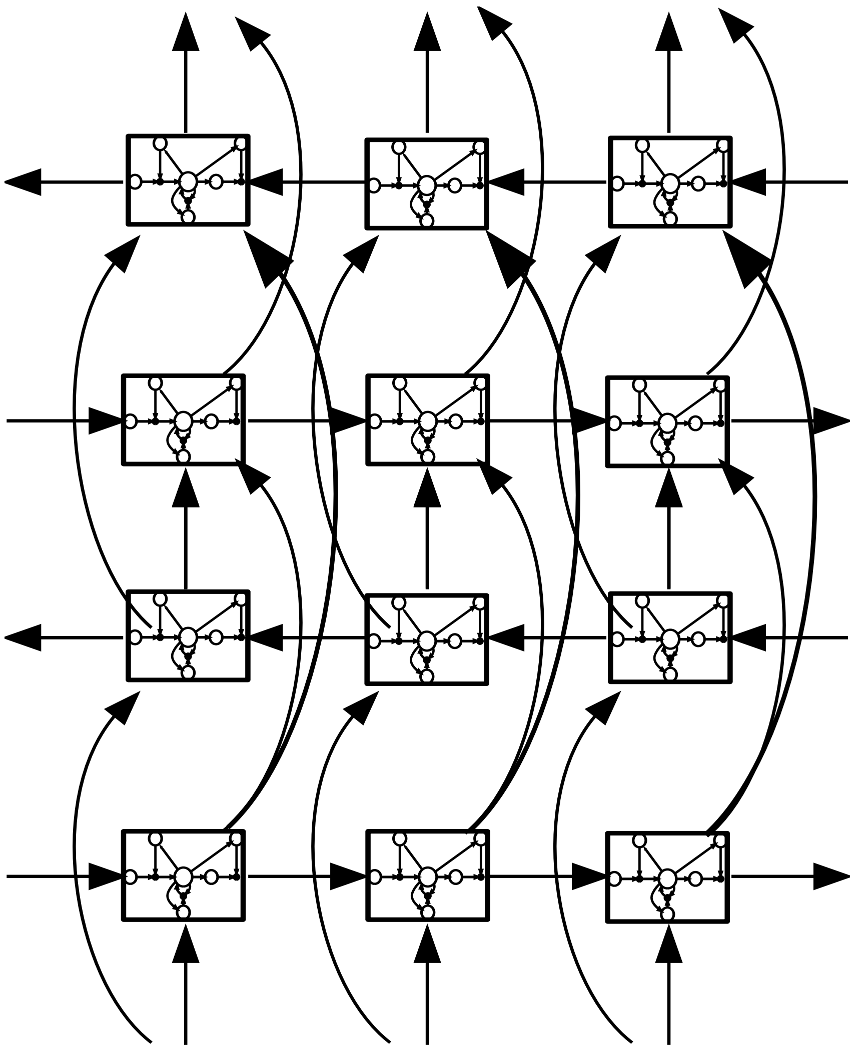
$$h_t = o_t \tanh(c_t)$$

Figure from (Graves et al., 2013)

Long Short-Term Memory (LSTM)



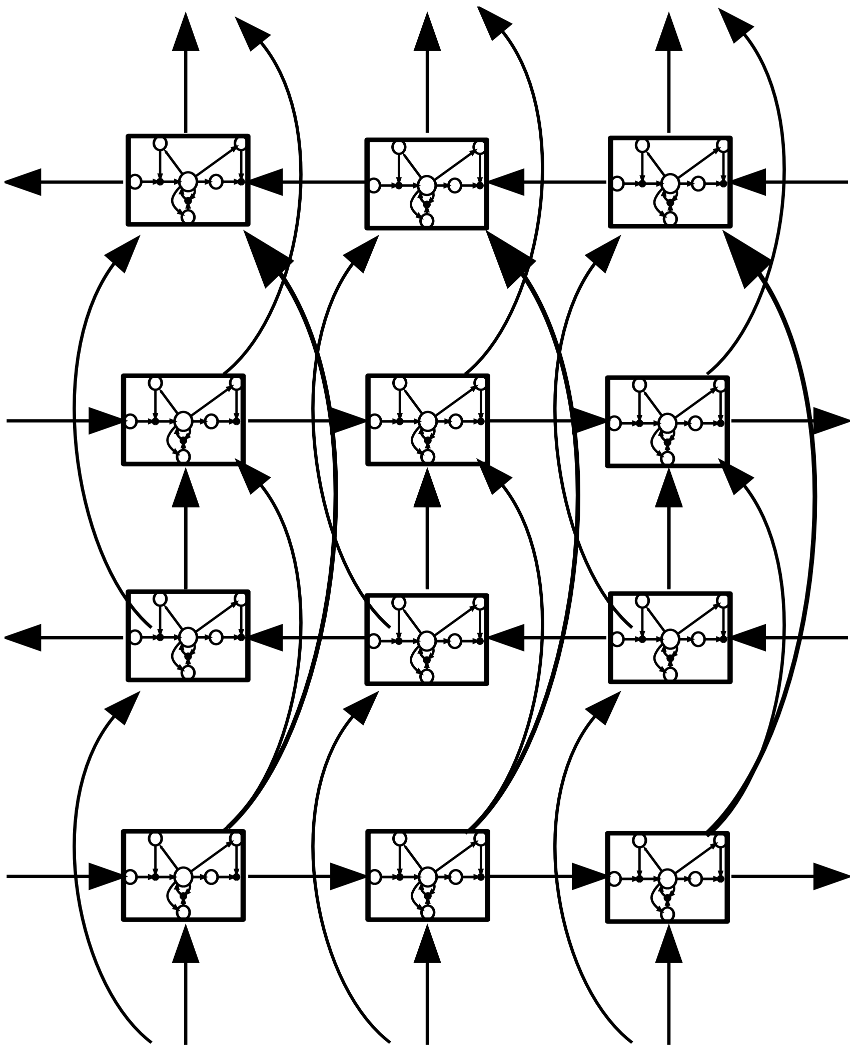
Deep Bidirectional LSTM (DBLSTM)



- Figure: input/output layers not shown
- **Same general topology** as a Deep Bidirectional RNN, but with **LSTM units** in the hidden layers
- No additional **representational power** over DBRNN, but **easier to learn** in practice

Figure from (Graves et al., 2013)

Deep Bidirectional LSTM (DBLSTM)



How important is this particular architecture?

Jozefowicz et al. (2015) **evaluated 10,000 different LSTM-like architectures** and found several variants that worked just as well on several tasks.

Figure from (Graves et al., 2013)

Summary

- **CNNs**
  - Are used for all aspects of **computer vision,** and have won numerous pattern recognition competitions
  - Able learn **interpretable features** at different levels of abstraction
  - Typically, consist of convolution layers, pooling layers, nonlinearities, and fully connected layers
- **RNNs**
  - Applicable to tasks such as **sequence labeling**, speech recognition, machine translation, etc.
  - Able to **learn context features** for time series data
  - Vanishing gradients are still a problem – but **LSTM units** can help

Tutorials

- LSTMs
  - Christopher Olah’s blog
  - http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Convolutional Neural Networks
  - Andrej Karpathy, CS231n Notes
  - http://cs231n.github.io/convolutional-networks/