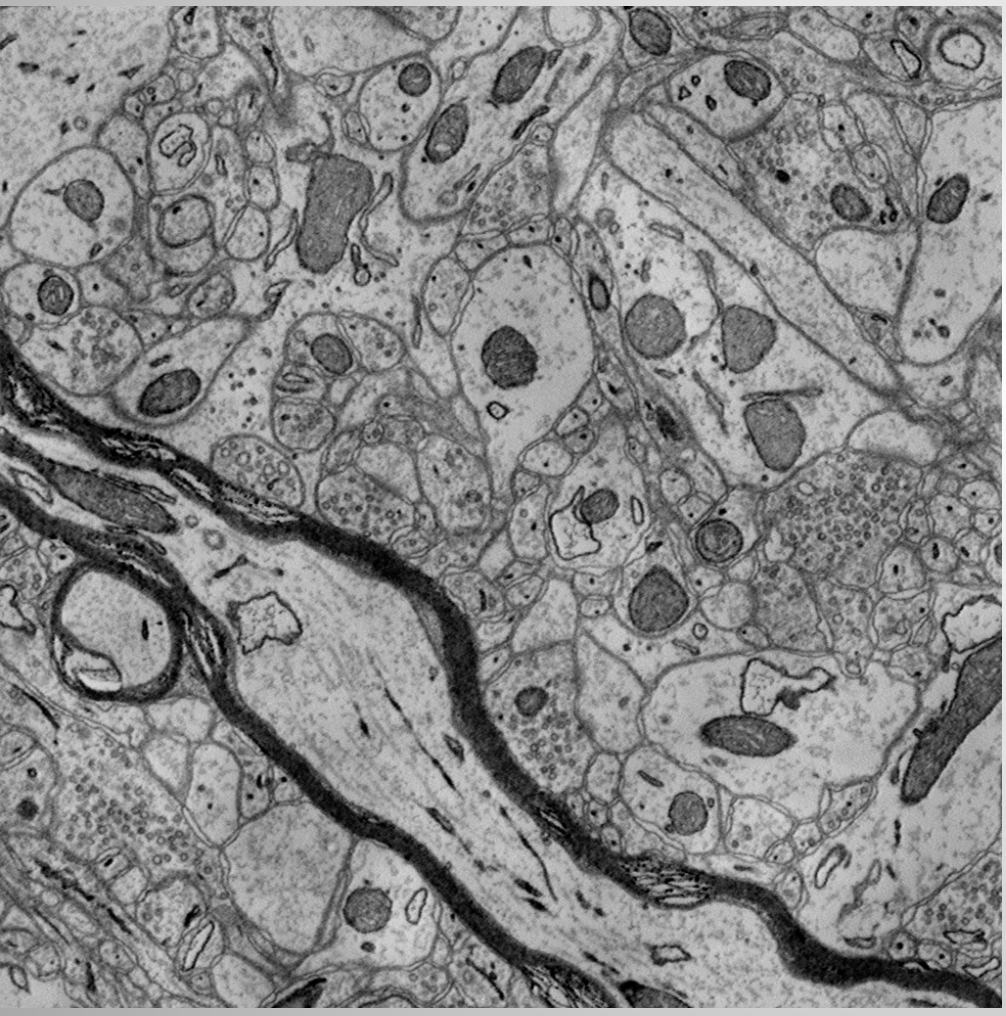
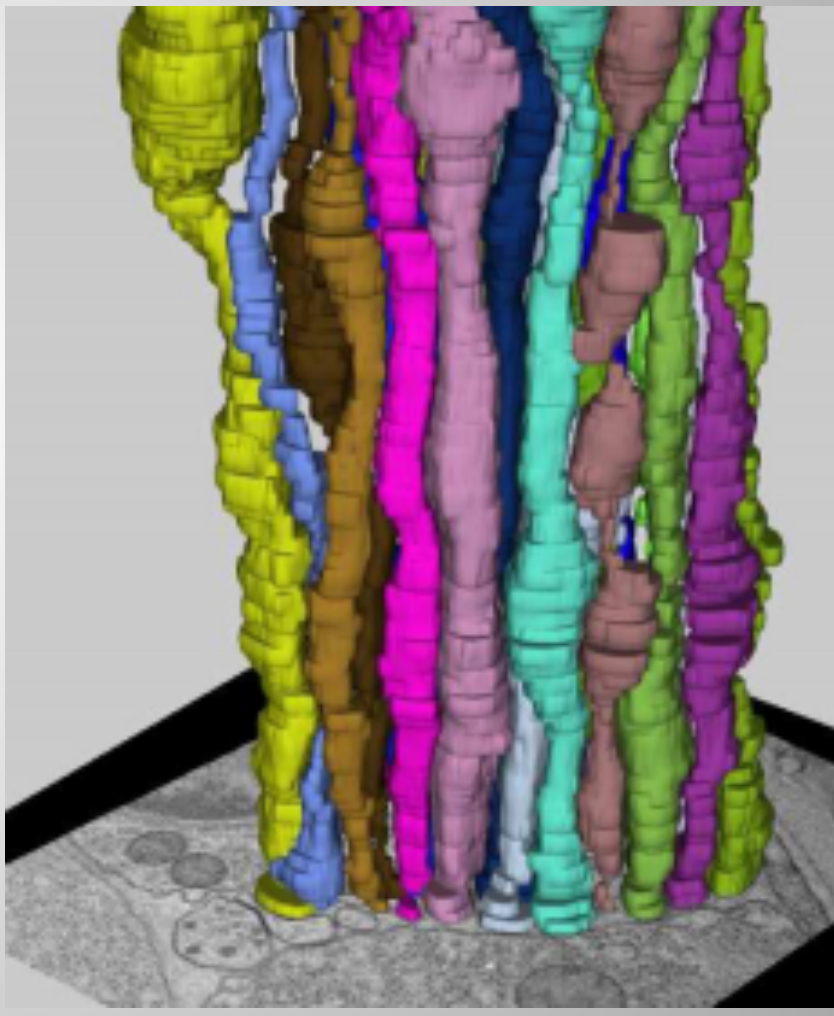
Fully Convolutional Neural Networks in Julia



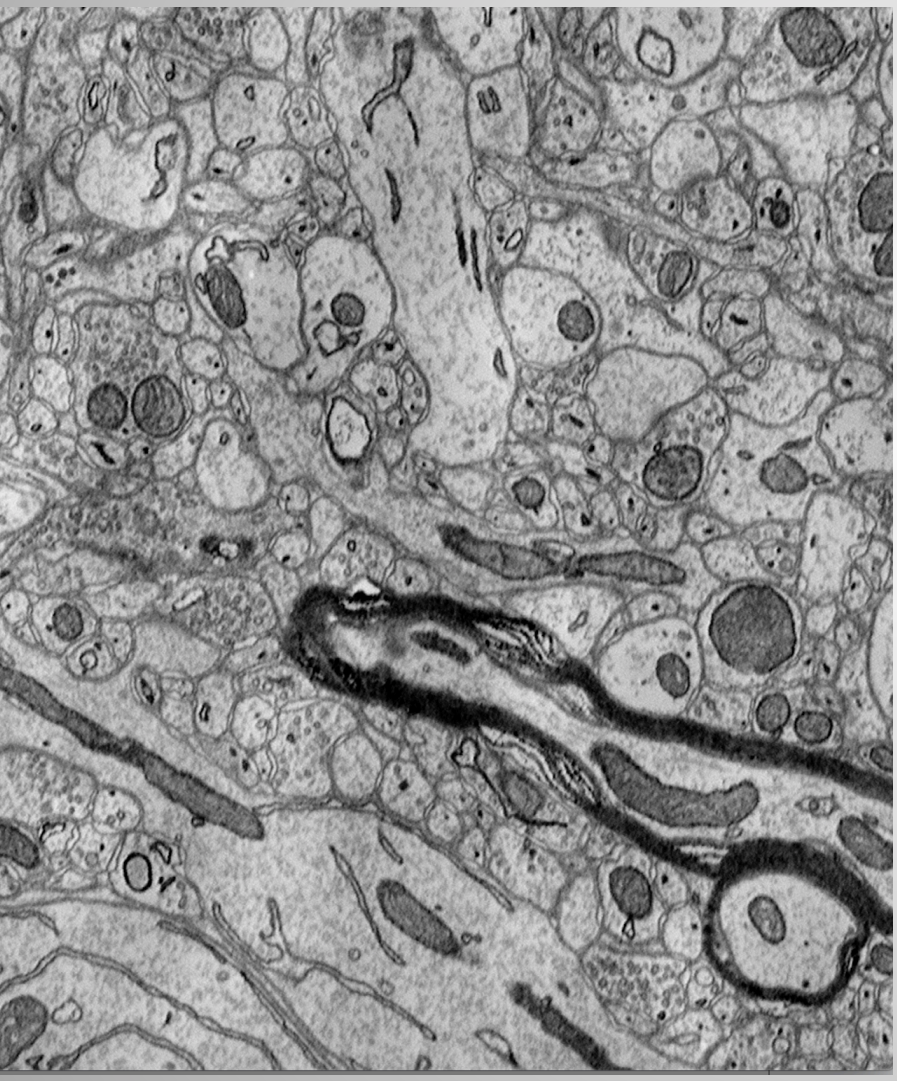
->

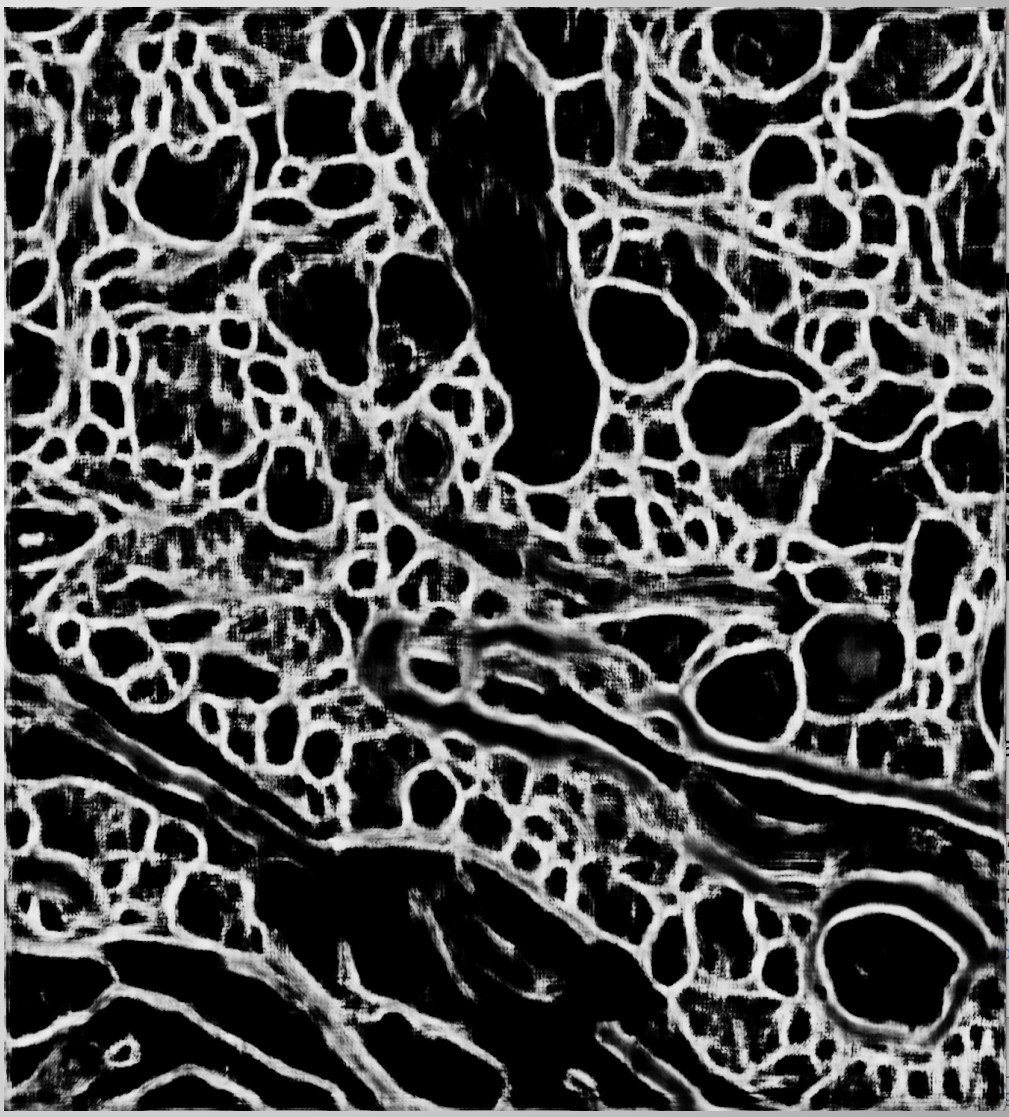


Agenda

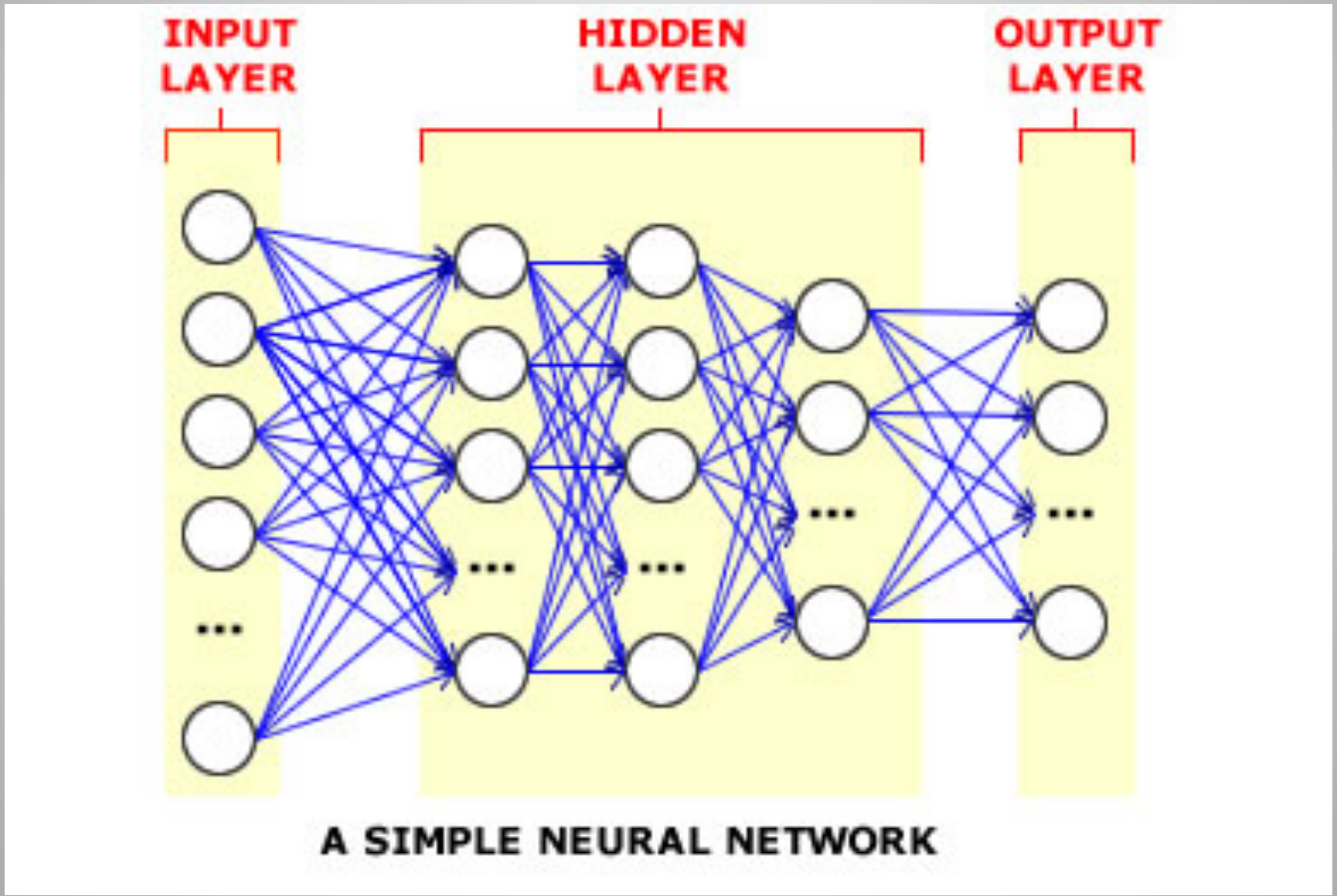
- Probability Map Generation
- Convolutional Neural Networks (CNN)
- Sliding Window
- Fully Convolutional Network (FC): conv + pool
- FullMocha/Caffe Implementation
- Evaluation

Probability map

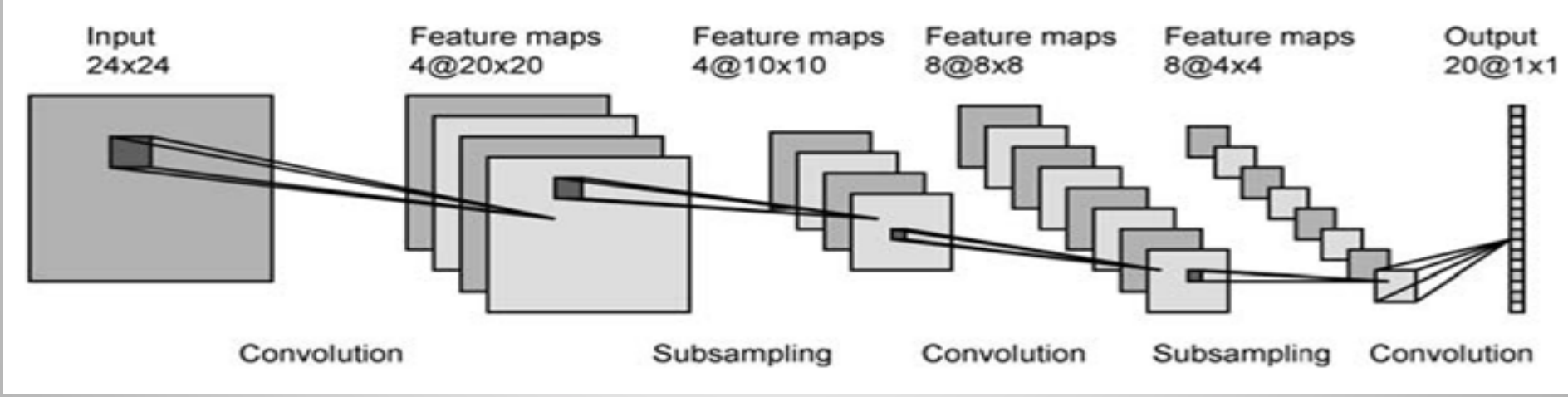


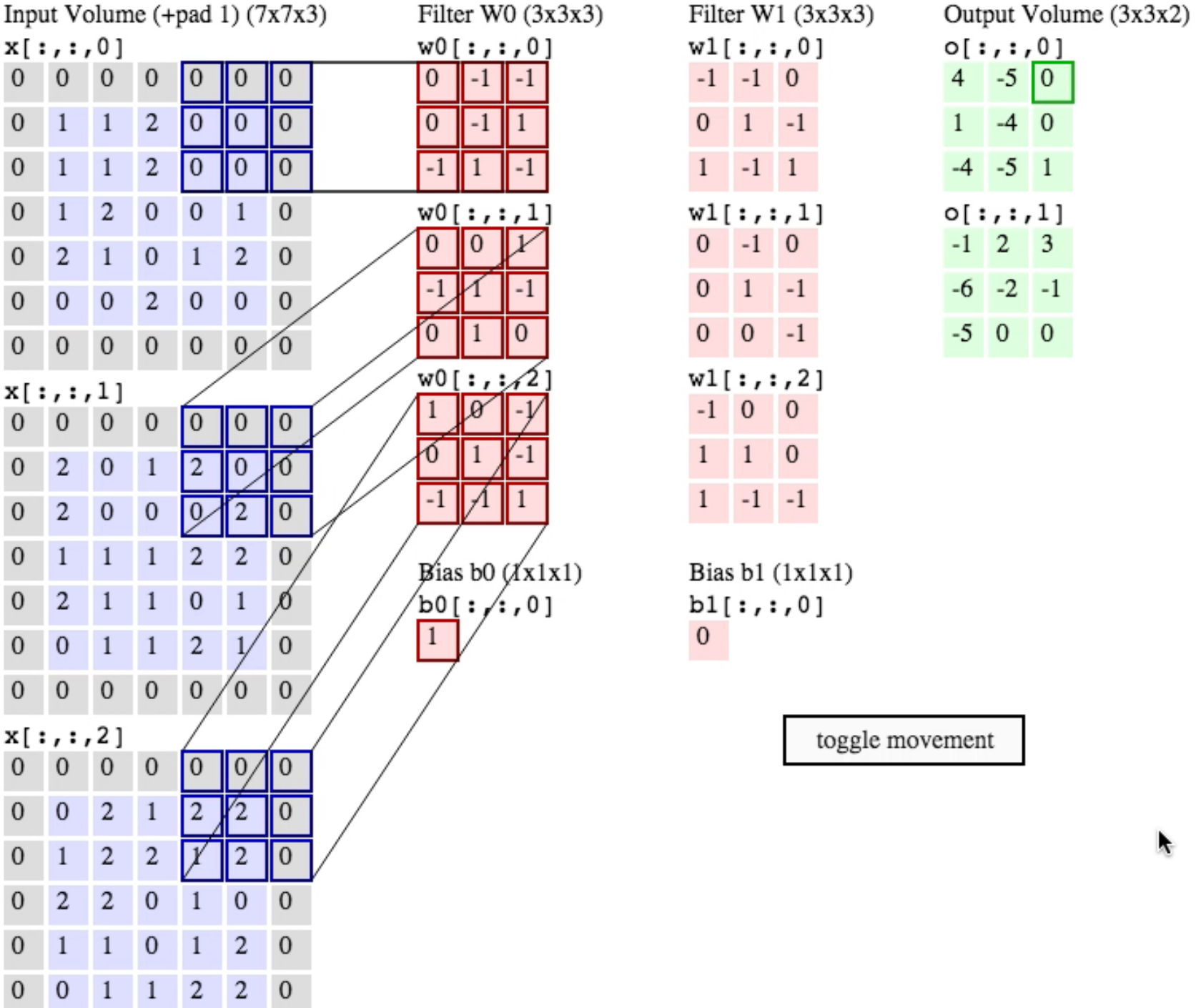


Convolutional Neural Networks



Convolutional Neural Networks




Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 1 2 0 0 0
0 1 1 2 0 0 0
0 1 2 0 0 1 0
0 2 1 0 1 2 0
0 0 0 2 0 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 2 0 1 2 0 0
0 2 0 0 0 2 0
0 1 1 1 2 2 0
0 2 1 1 0 1 0
0 0 1 1 2 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 2 1 2 2 0
0 1 2 2 1 2 0
0 2 2 0 1 0 0
0 1 1 0 1 2 0
0 0 1 1 2 2 0
Filter W0 (3x3x3)
w0[:,:,0]
0 -1 -1
0 -1 1
-1 1 -1
w0[:,:,1]
0 0 1
-1 1 -1
0 1 0
w0[:,:,2]
1 0 -1
0 1 -1
-1 -1 1
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 -1 0
0 1 -1
1 -1 1
w1[:,:,1]
0 -1 0
0 1 -1
0 0 -1
w1[:,:,2]
-1 0 0
1 1 0
1 -1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
4 -5 0
1 -4 0
-4 -5 1
o[:,:,1]
-1 2 3
-6 -2 -1
-5 0 0
toggle movement

Sliding Window

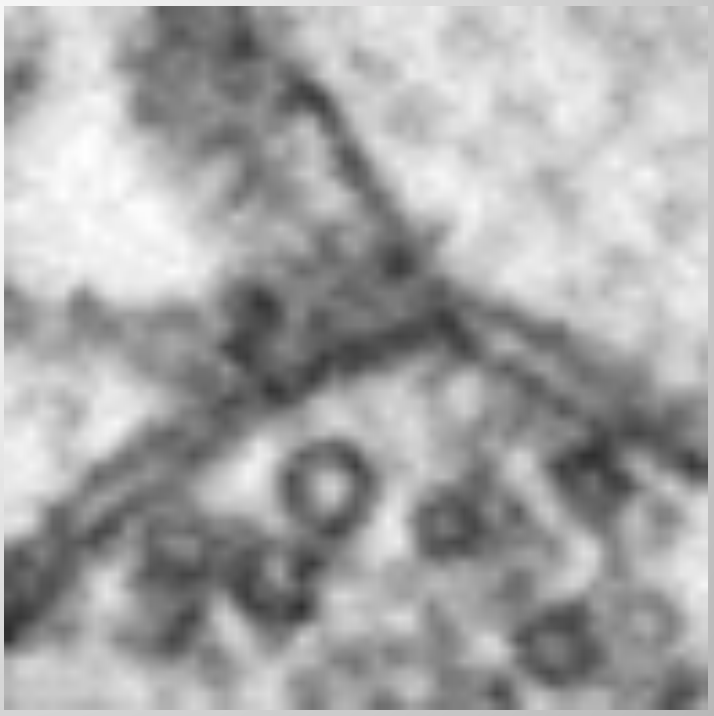
- A window 49x49 pixels

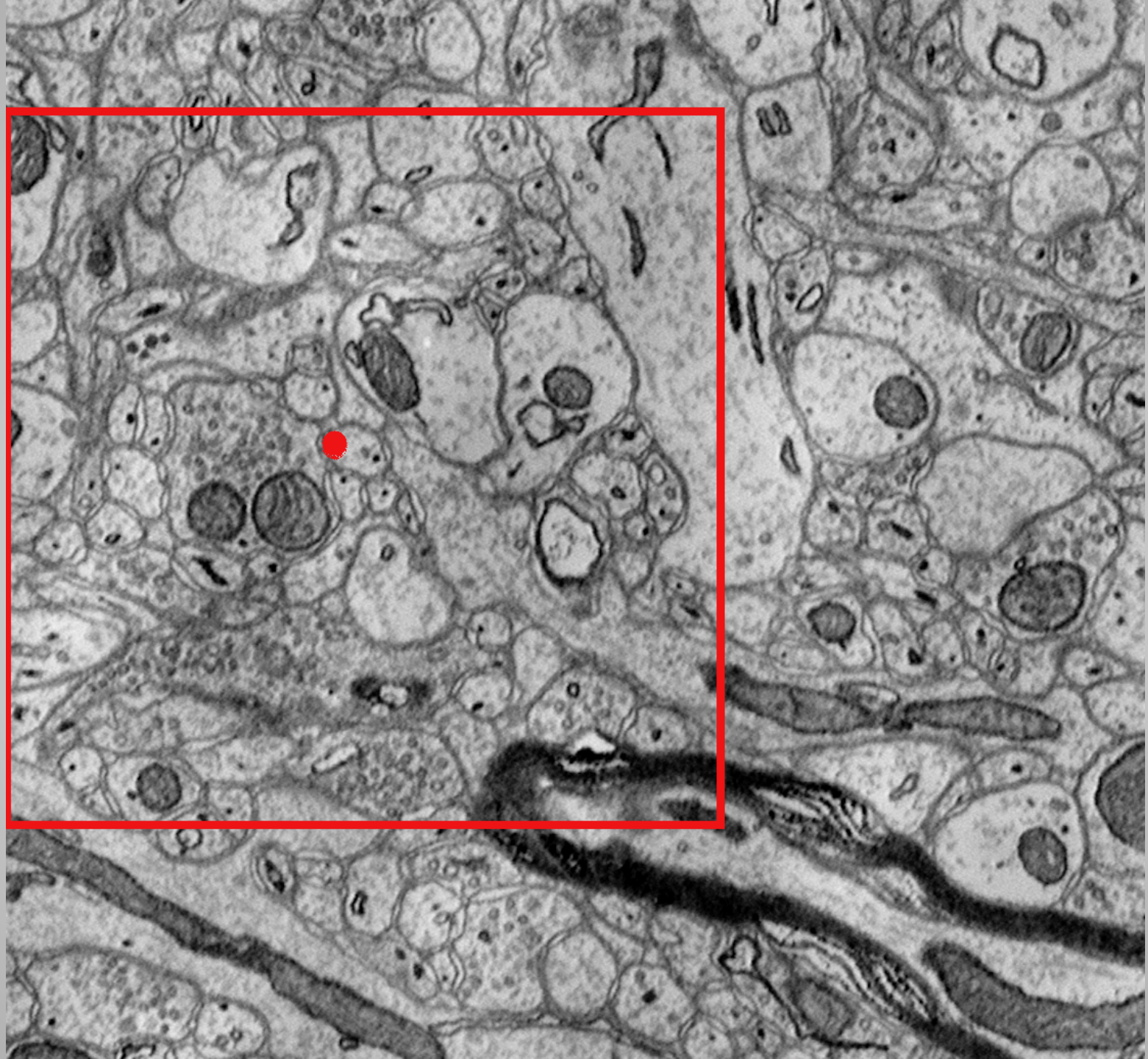
membrane

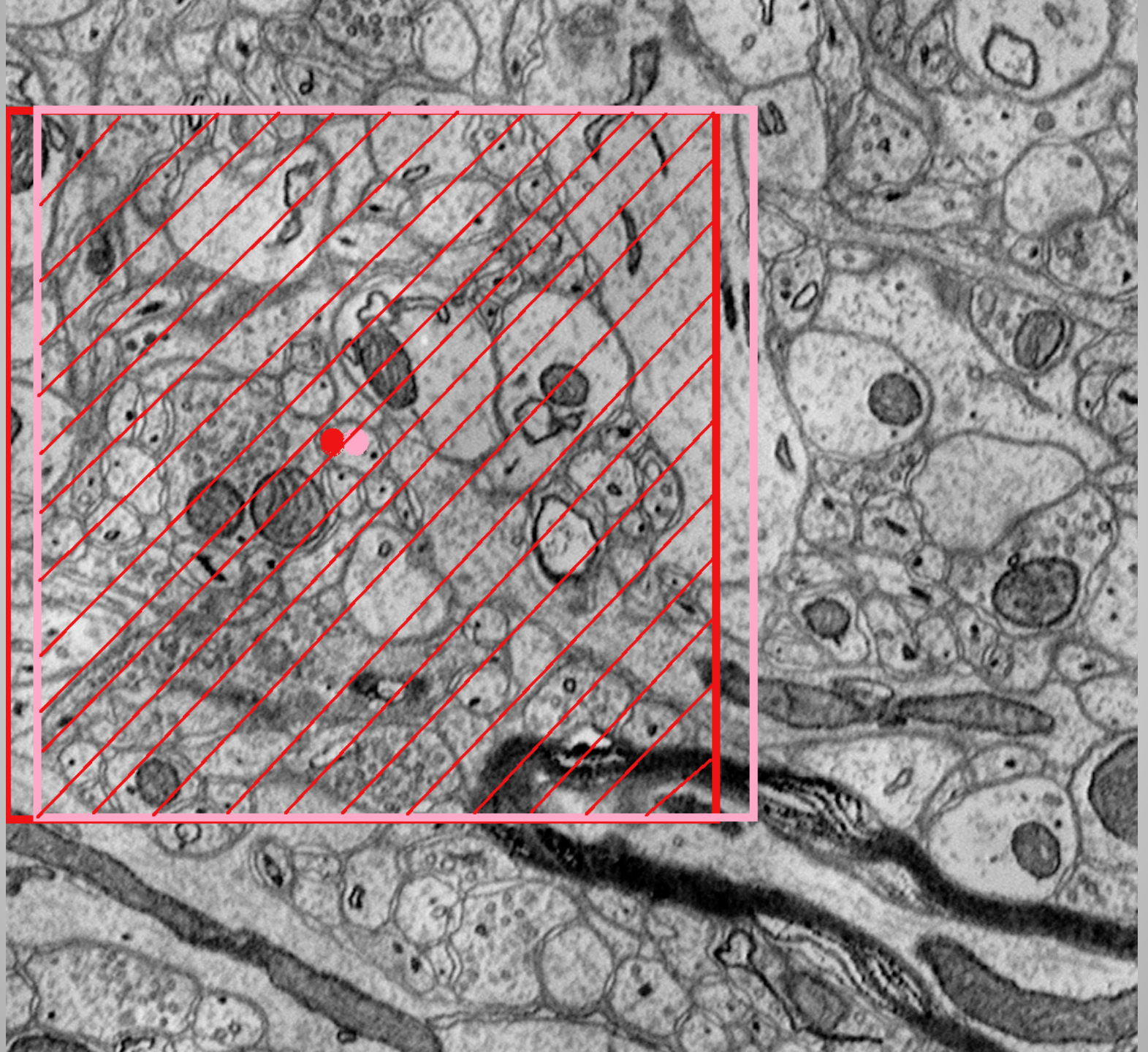
0:

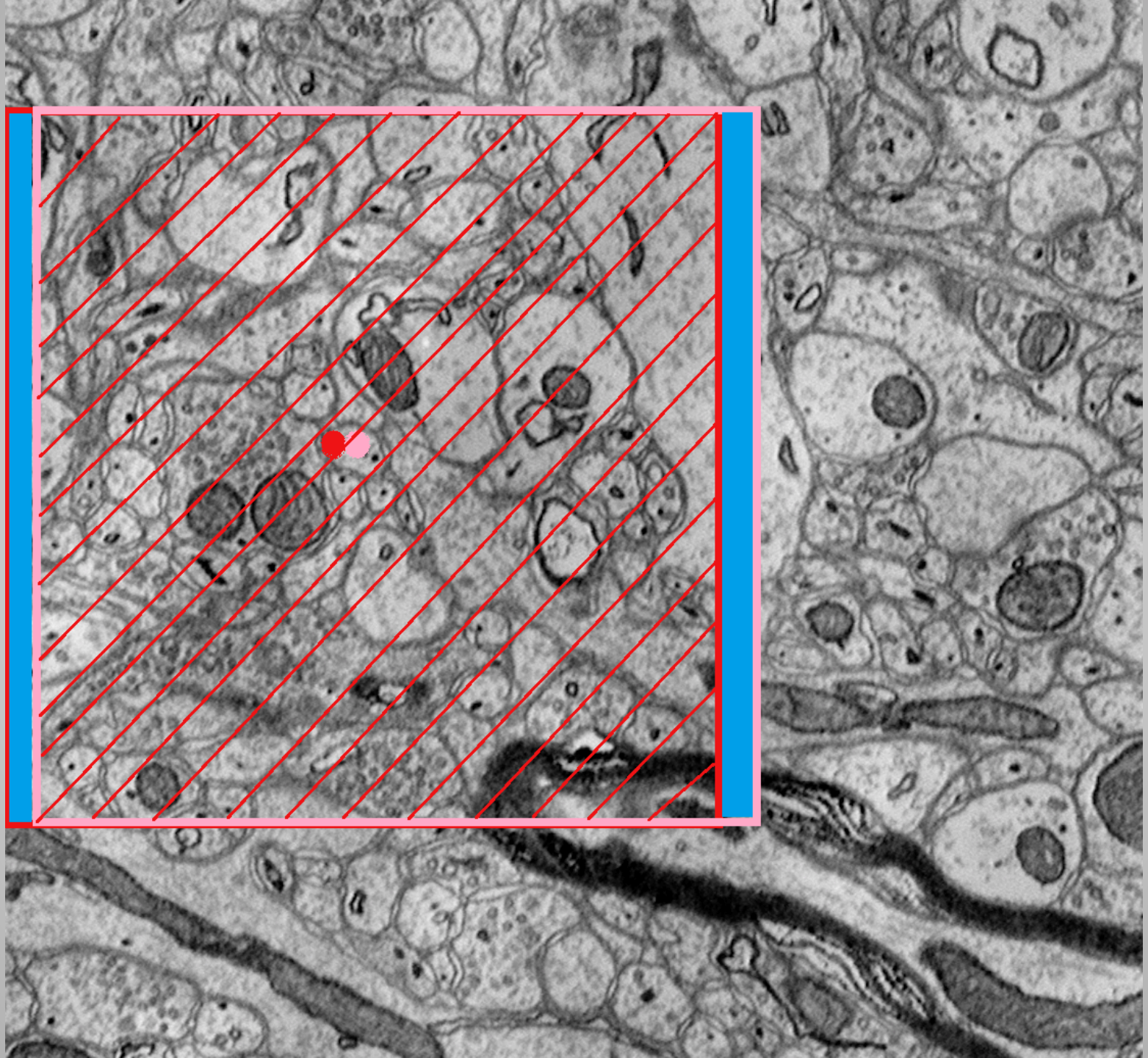
non-membrane

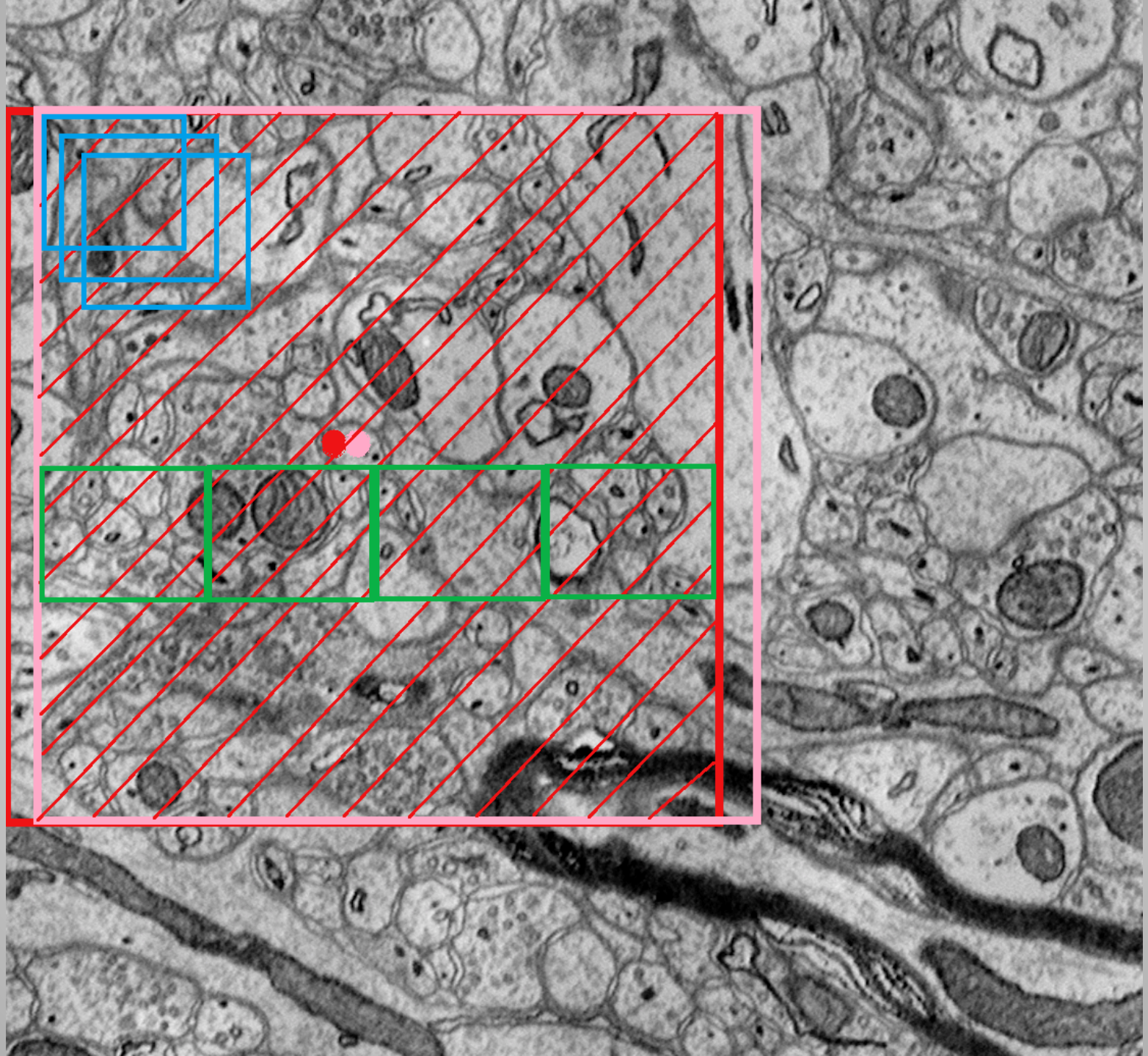
1:











Theoretical Speedup

$$C_l = s^2 \cdot |P_{l-1}| \cdot |P_l| \cdot w_l^2 \cdot k_l^2 \cdot 2$$

$$C_l = s_{x,l} \cdot s_{y,l} \cdot |P_{l-1}| \cdot |P_l| \cdot F_l \cdot k_l^2 \cdot 2$$

Layer (l)	s	s_{l-1}	$\lvert\mathbf{P}_{l-1}\rvert$	$\lvert\mathbf{P}_l\rvert$	w_l	k_l	F_l	$\text{FLOPS}_l^{\text{patch}}[\cdot 10^9]$	$\text{FLOPS}_l^{\text{image}}[\cdot 10^9]$	speedup
1	512	559	1	48	92	4	1	3408	0.5	7114.8
3	512	279	48	48	42	5	4	53271	35.9	1485.1
5	512	139	48	48	18	4	16	6262	22.8	274.7
7	512	69	48	48	6	4	64	695	22.5	30.9
Total								63636	81.6	779.8

Mocha.jl

- Caffe – library from Berkley (C++/CUDA)
- Mocha.jl – Caffe inspired – takes Caffe-trained models (classifiers) as input

- Modified library (extension) that takes *same weigth* and *same network* and runs fully convolutional approach.

Implementation Speedup

- 100x1024 px, 49x49 window, 7 layers

Python/Caffe, GPU	180s	1	speedup
Python/Caffe, CPU	954s	5.3	0.19x
Julia/Mocha, CPU	1526s	8.5	0.12x
Fully Conv Julia/CPU	47s	0.26	3.8x
Native C++ Fully Conv	3.8s	0.002	47x
Julia/CPU C++ backend	~3.8s ??		

>>

FullMocha.jl

- Remaining work to do: implement other network types (fully connected, dropout, ReLU etc)
- Open source in December/January

Network Structure

Layer (l)	Type	Maps ($\|\mathbf{P}_l\|$) and neurons	Kernel ($k_l \times k_l$)
0	input	1 map of 95x95 neurons	
1	convolutional	48 maps of 92x92 neurons	4x4
2	max pooling	48 maps of 46x46 neurons	2x2
3	convolutional	48 maps of 42x42 neurons	5x5
4	max pooling	48 maps of 21x21 neurons	2x2
5	convolutional	48 maps of 18x18 neurons	4x4
6	max pooling	48 maps of 9x9 neurons	2x2
7	convolutional	48 maps of 6x6 neurons	4x4
8	max pooling	48 maps of 3x3 neurons	2x2
9	fully connected	200 neurons	1x1
10	fully connected	2 neurons	1x1