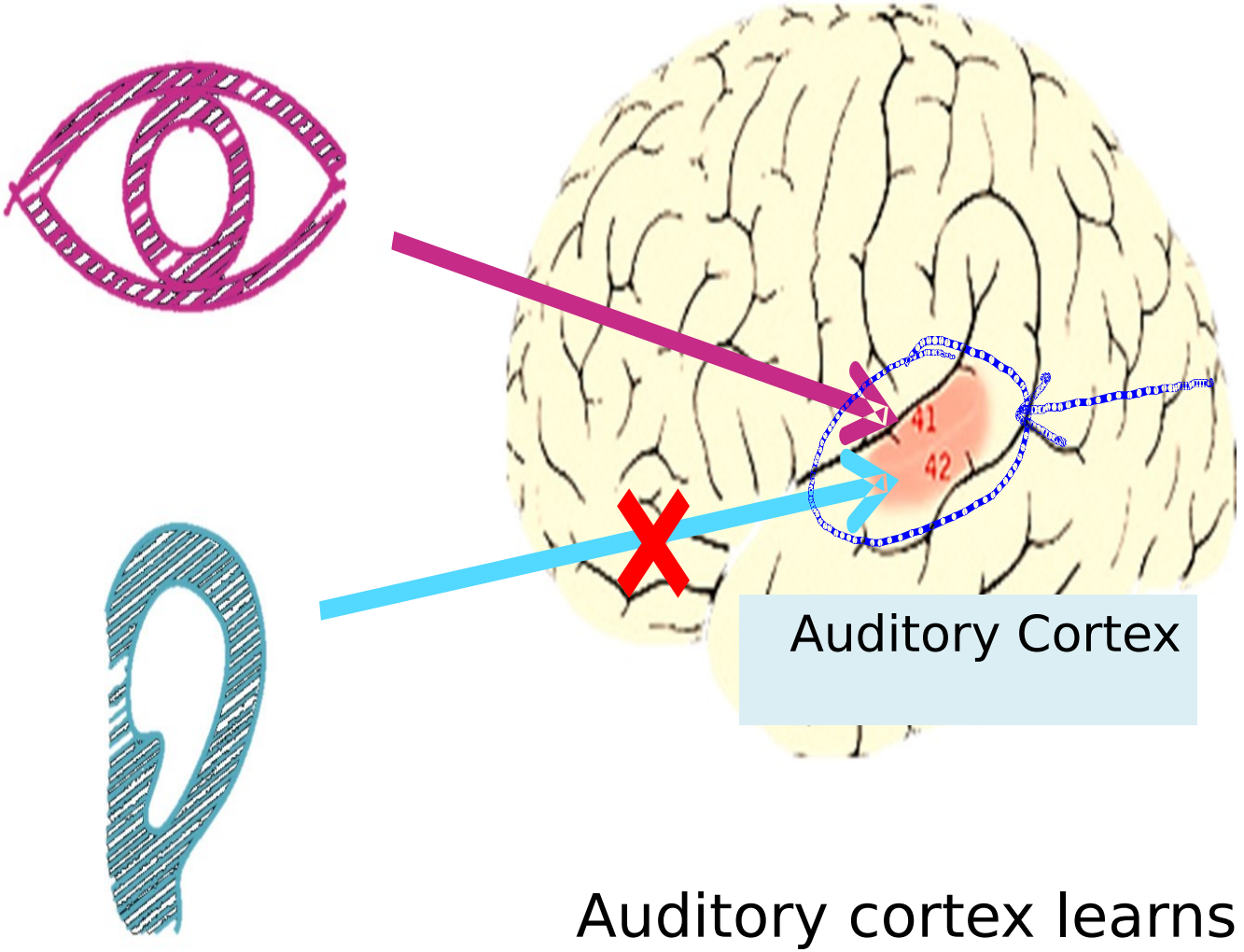
Deep Learning Internals-Part 1

By Mohit Kumar

Neural Networks

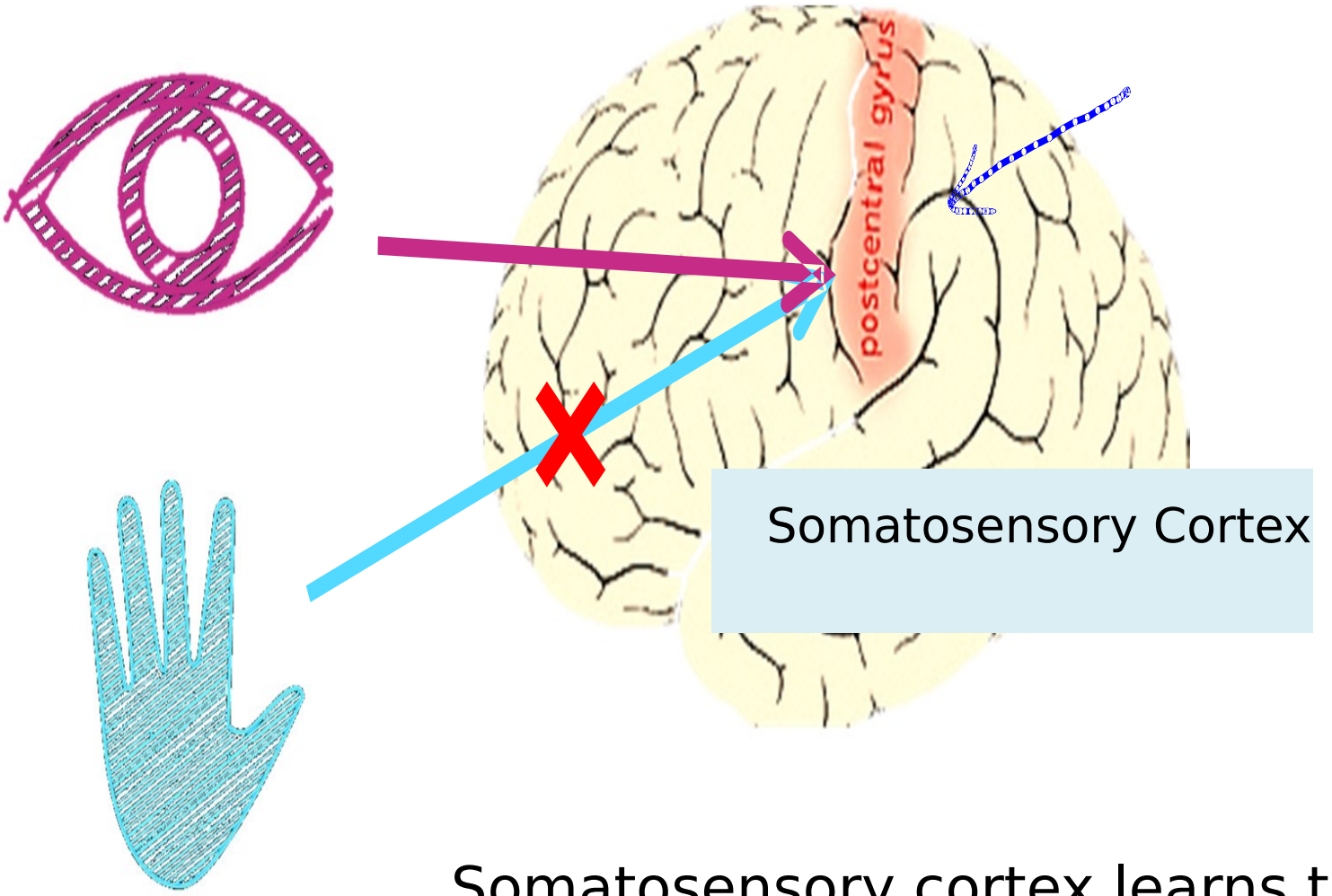
- Origins: Algorithms that try to mimic the brain.
- Was very widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications

The “one learning algorithm” hypothesis



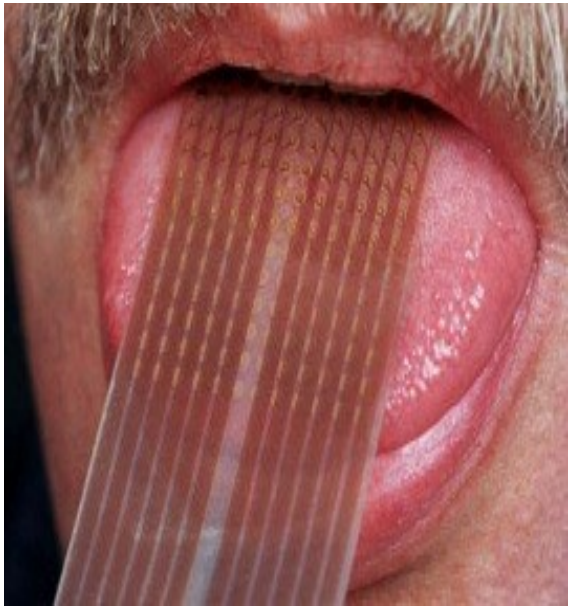
Auditory cortex learns to see

The “one learning algorithm” hypothesis



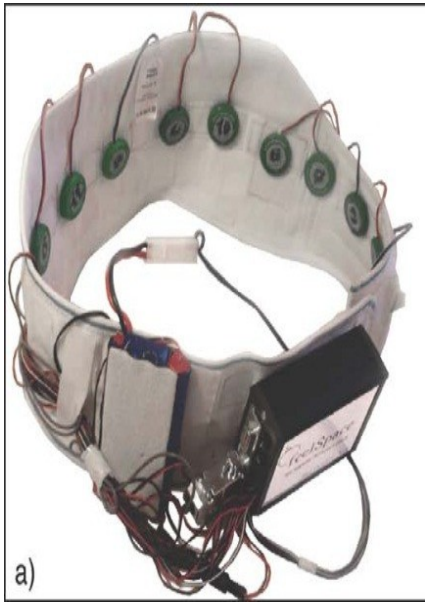
Somatosensory cortex learns to see

Sensor representations in the brain

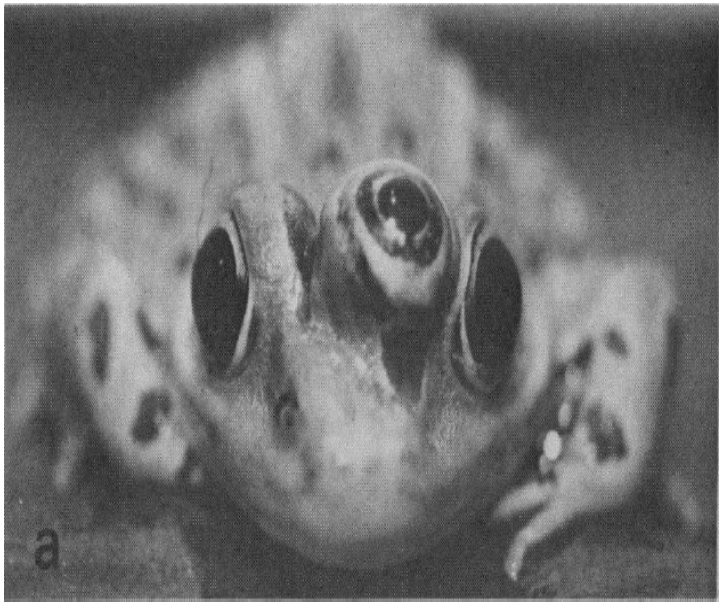


Seeing with your tongue

Human echolocation (sonar)

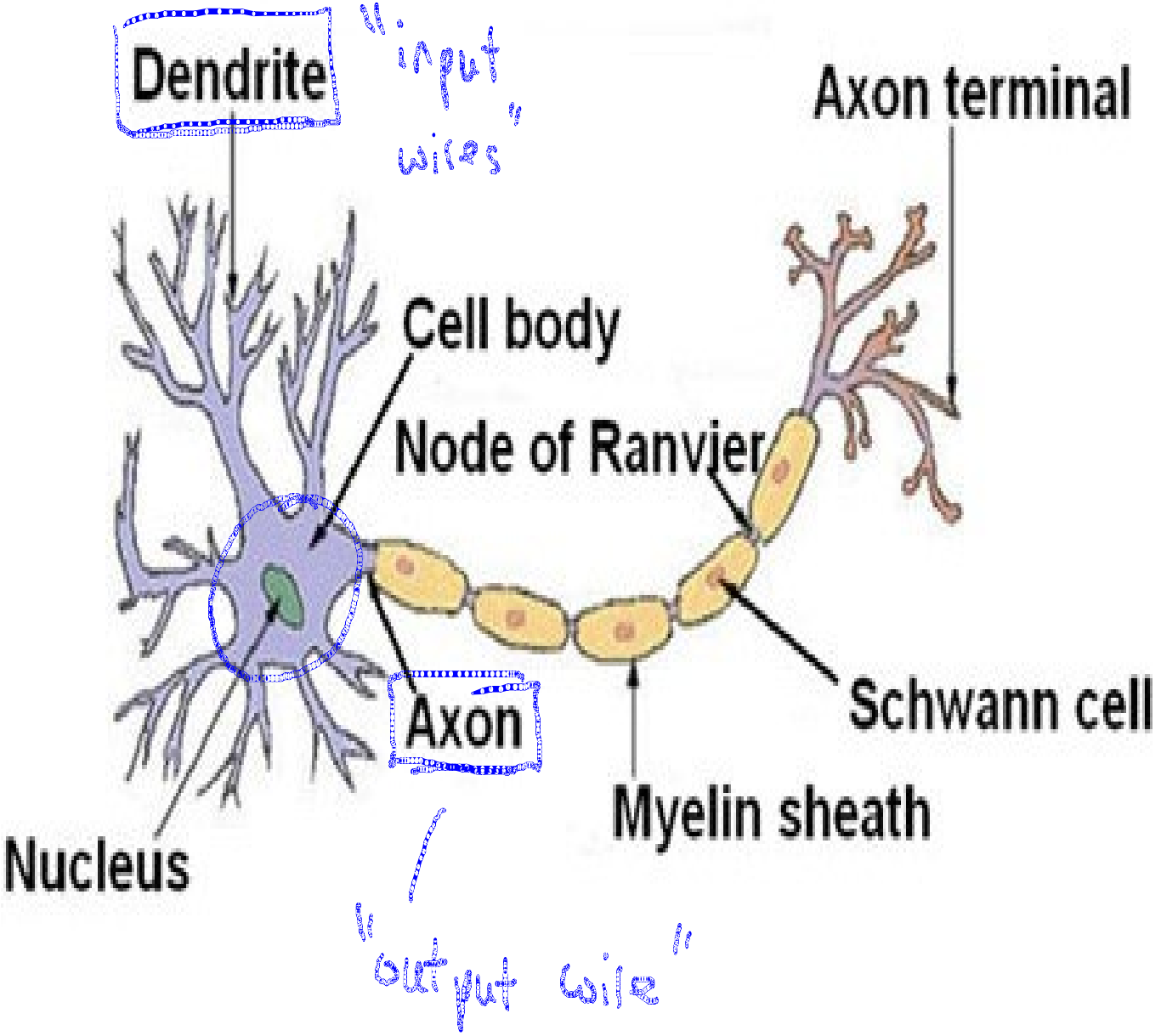


Haptic belt: Direction sense

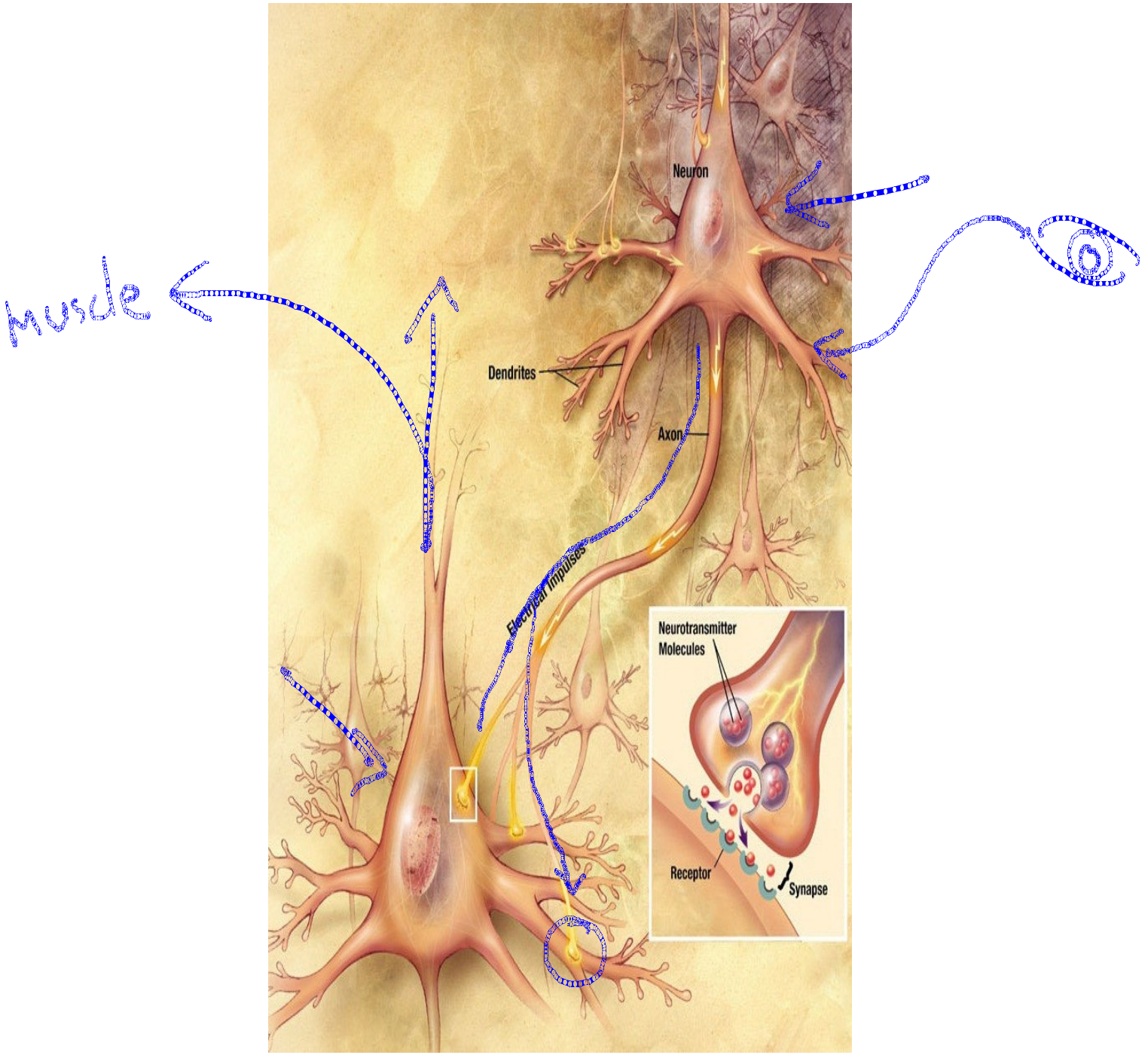


Implanting a 3rd eye

Neuron in the brain



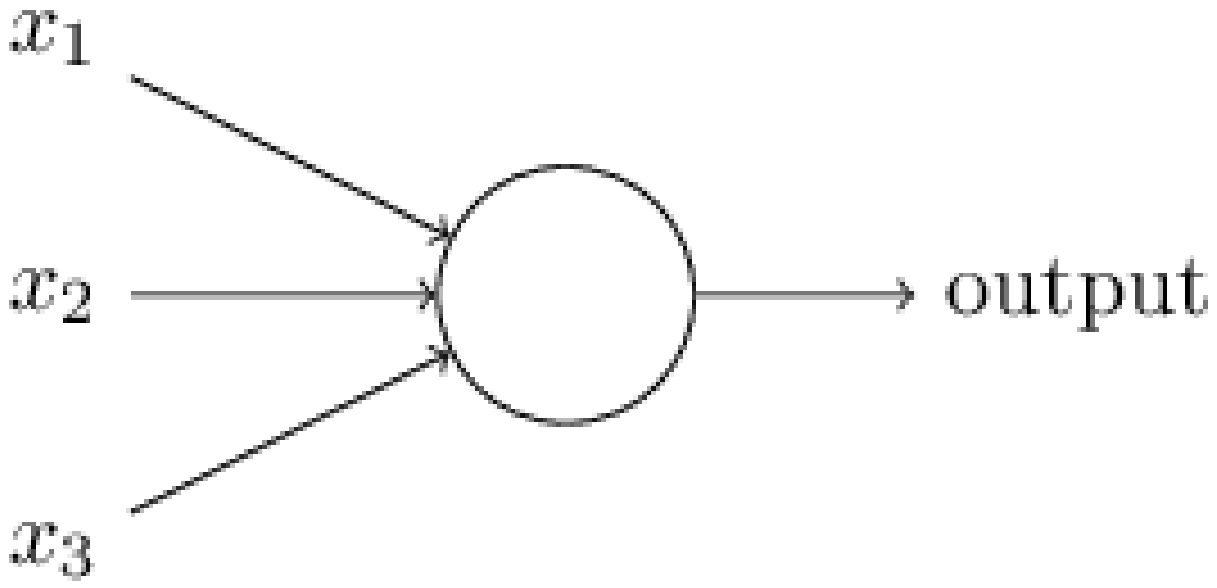
Neurons in the brain



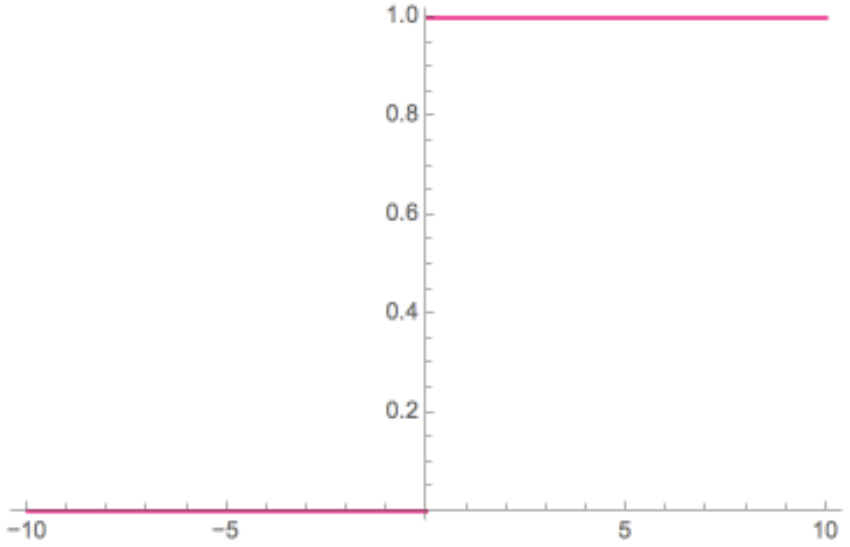
Artificial Neurons:Perceptrons

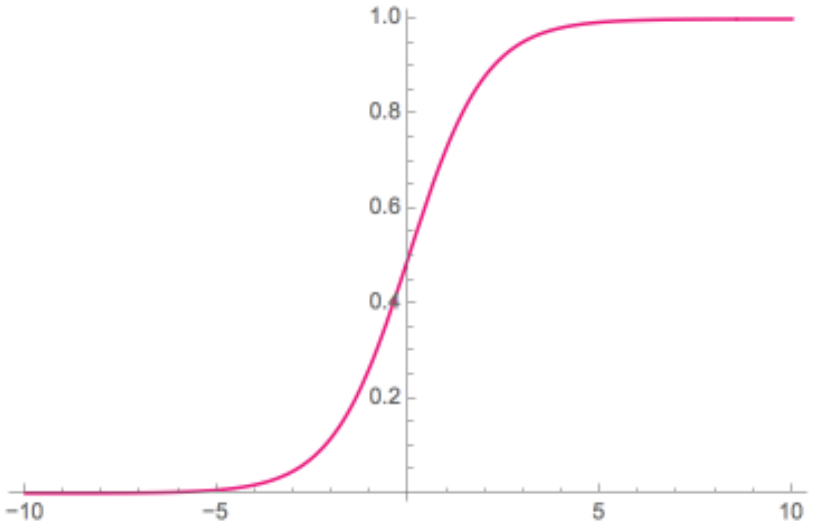
- Suppose the weekend is coming up, and you've heard that there's going to be a food festival in your city.
- You might make your decision by weighing up three factors:
  - Is the weather good?
  - Does your boyfriend or girlfriend want to accompany you?
  - Is the festival near public transit? (You don't own a car).

Artificial Neurons:Perceptrons

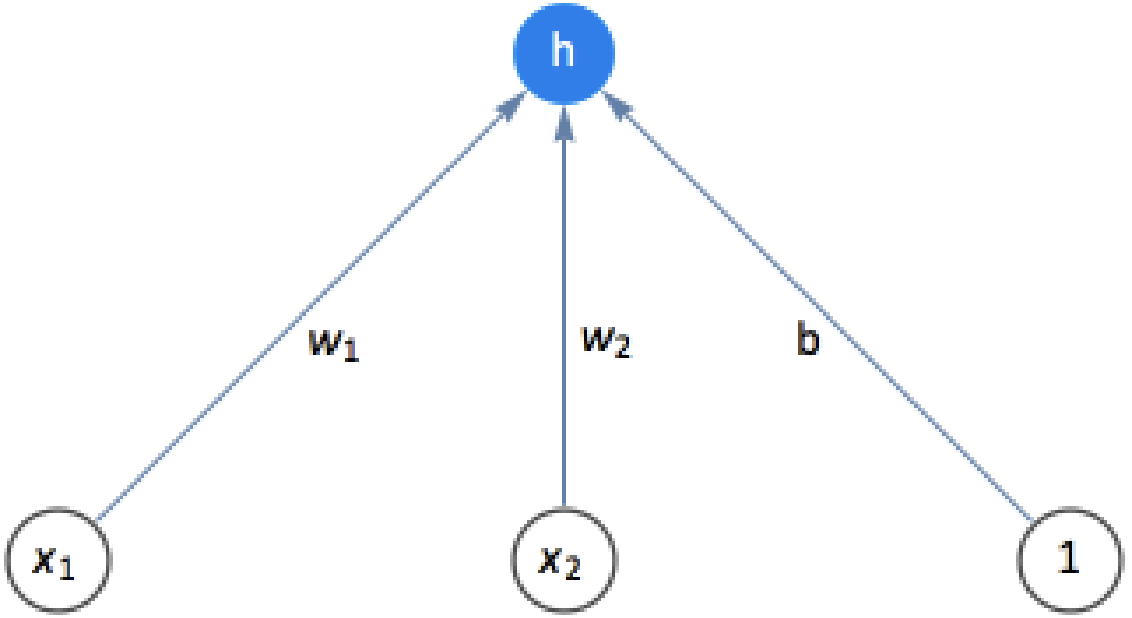


Artificial Neurons:Sigmoid Neuron



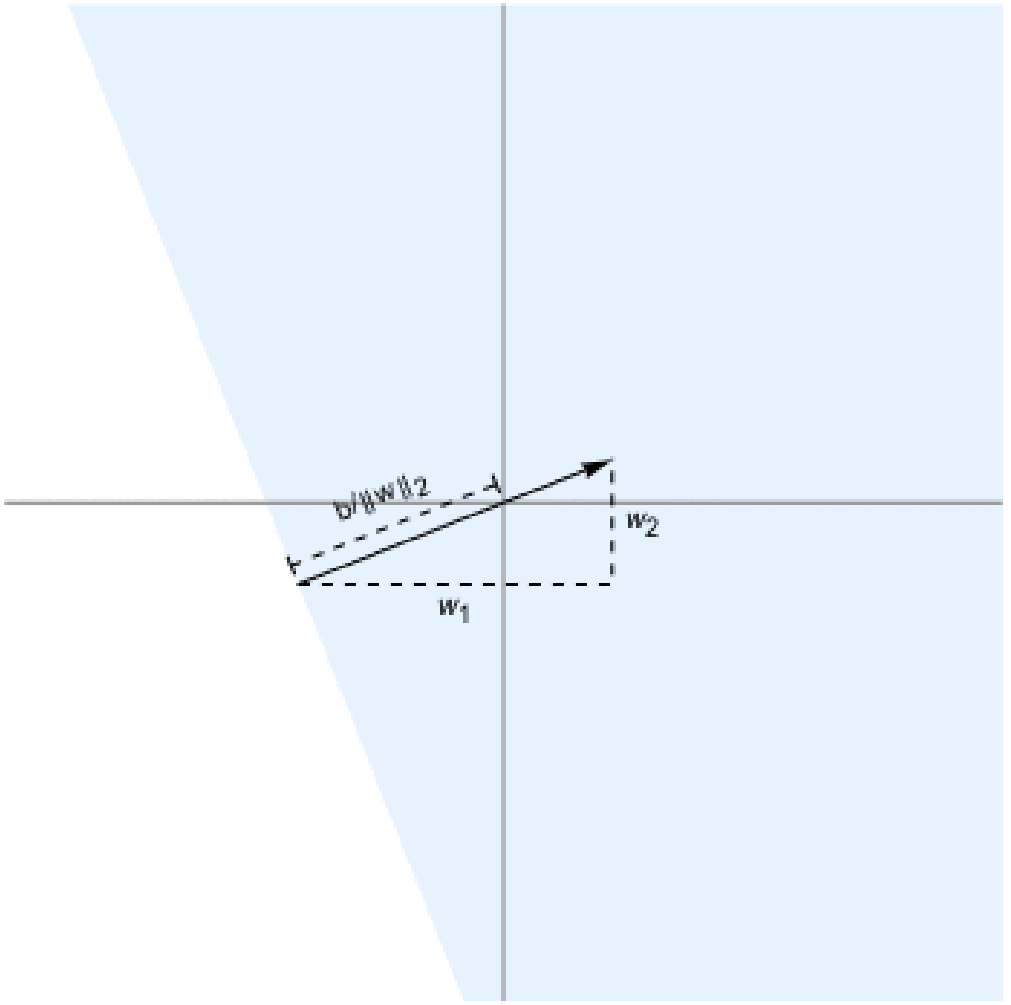


Artificial Neurons:Sigmoid Neuron

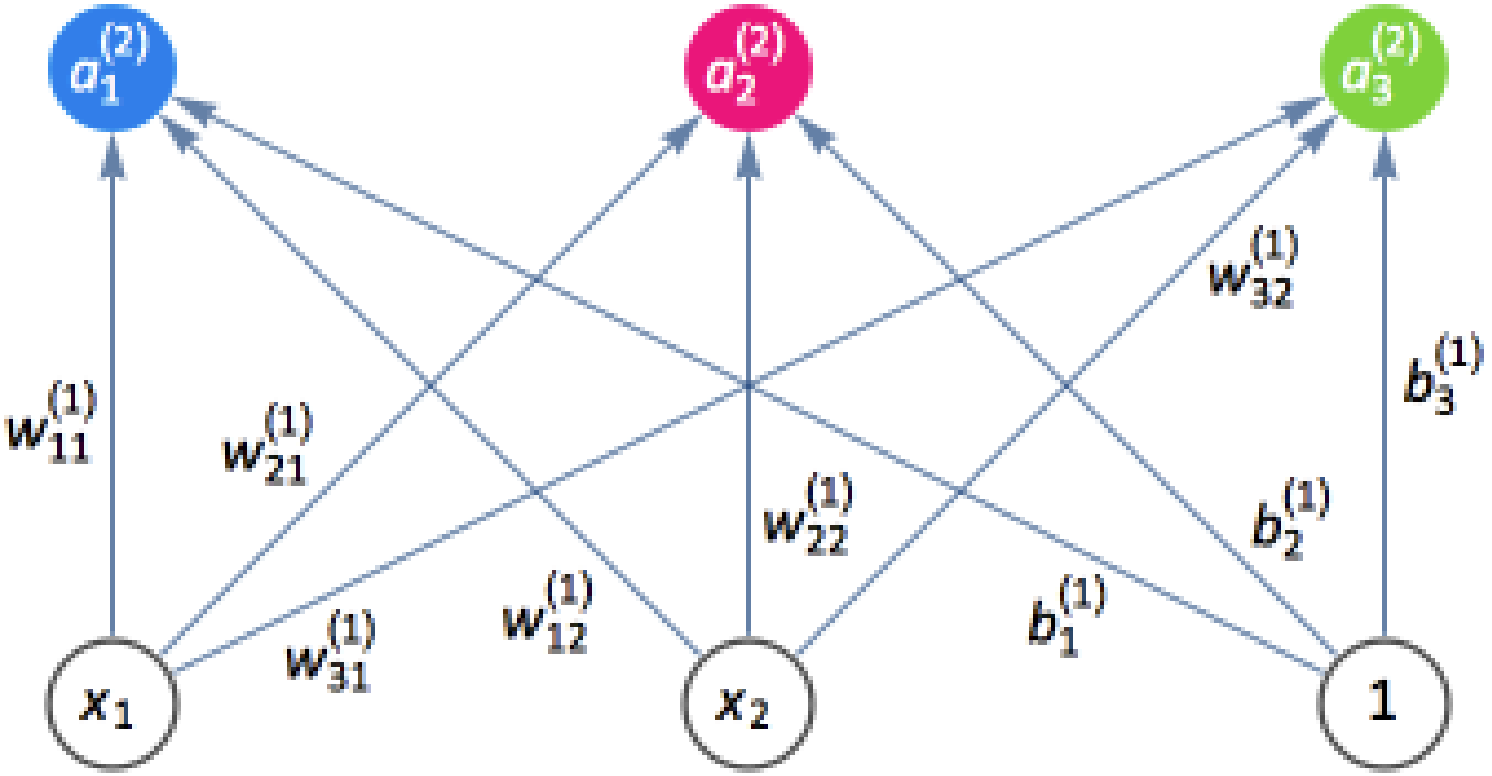


$$h = f\left(w_1 x_1 + w_2 x_2 + b\right)$$

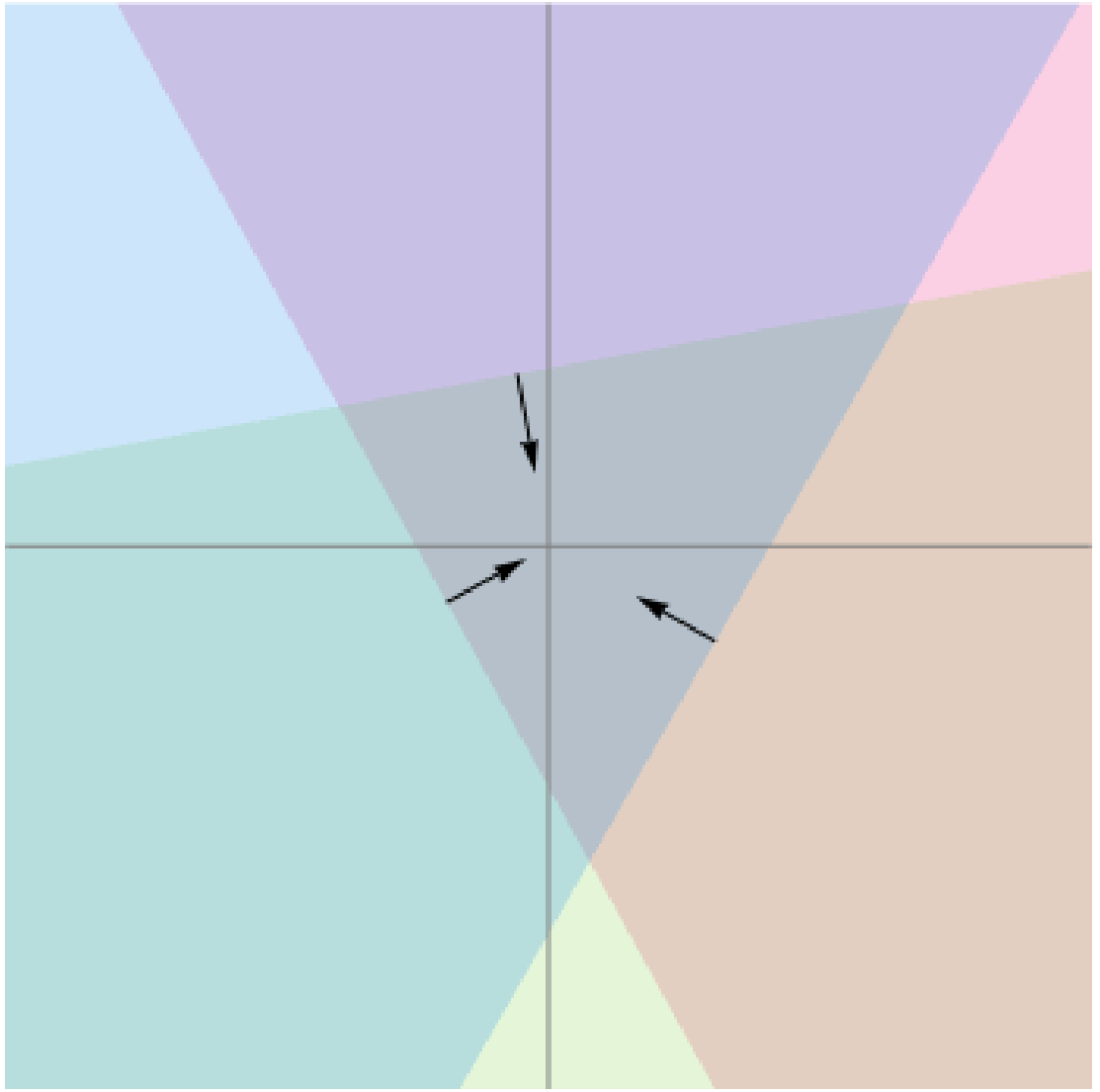
Artificial Neurons:Sigmoid Neuron



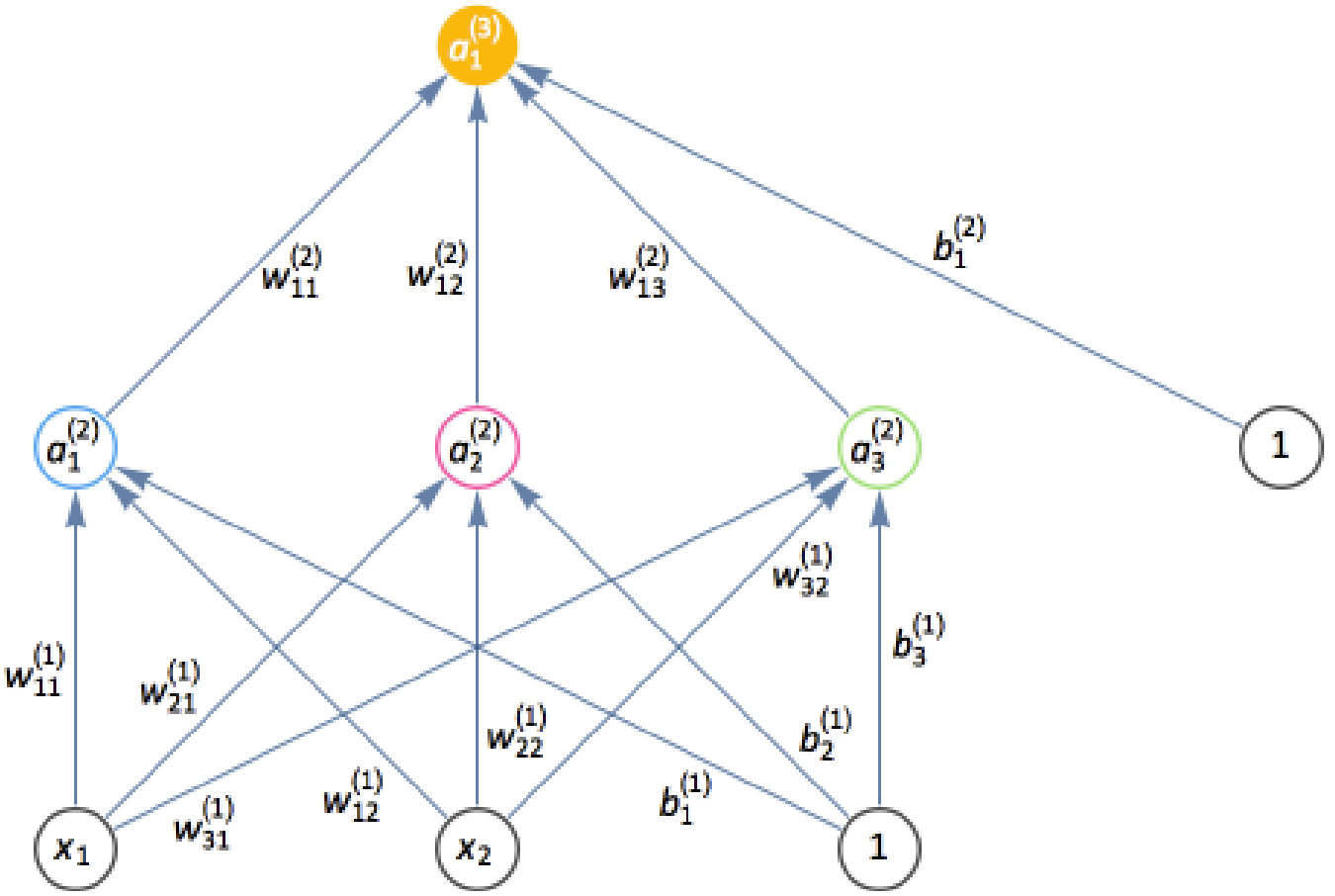
Deep Neural Network



Deep Neural Network

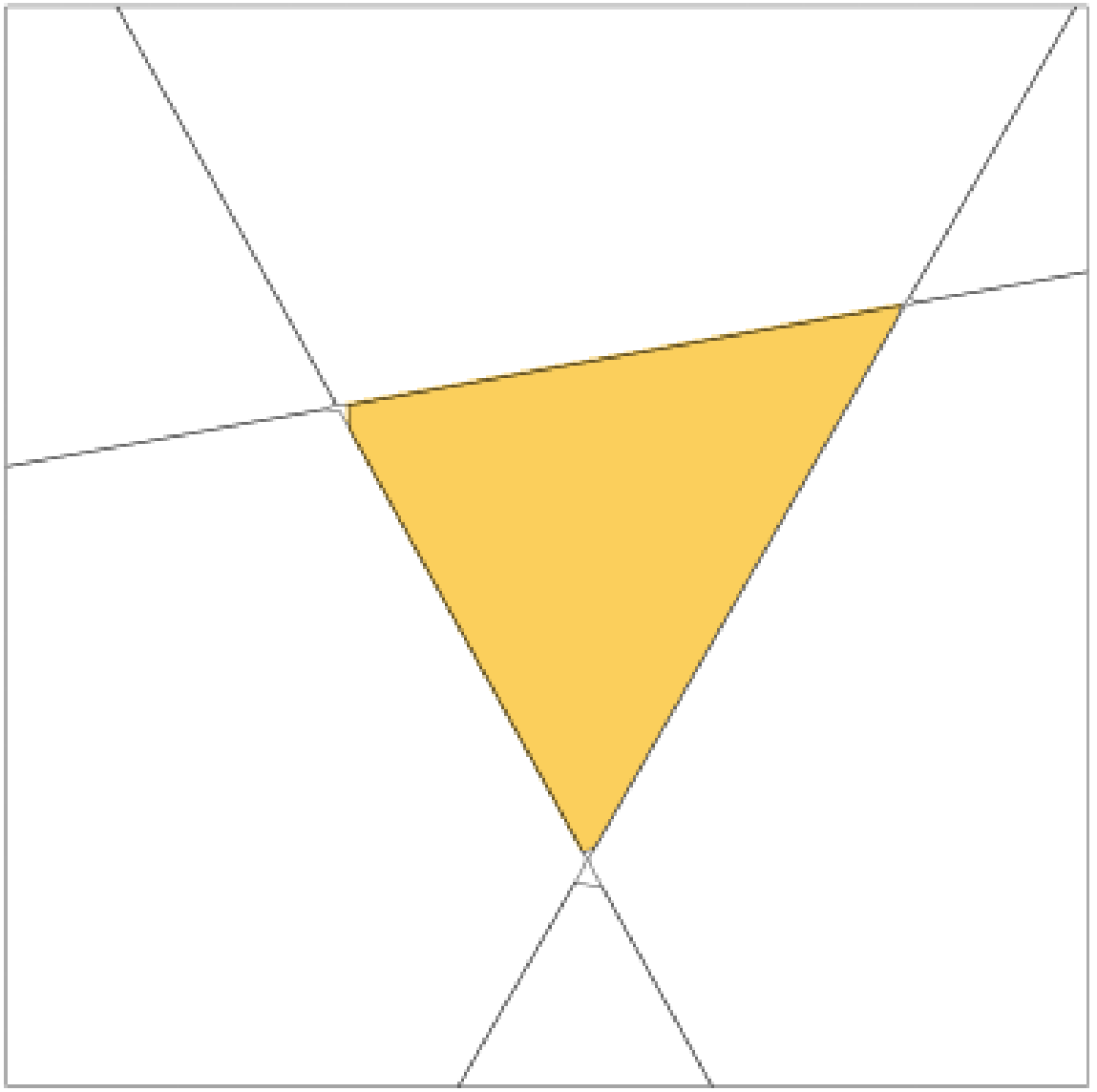


Deep Neural Network



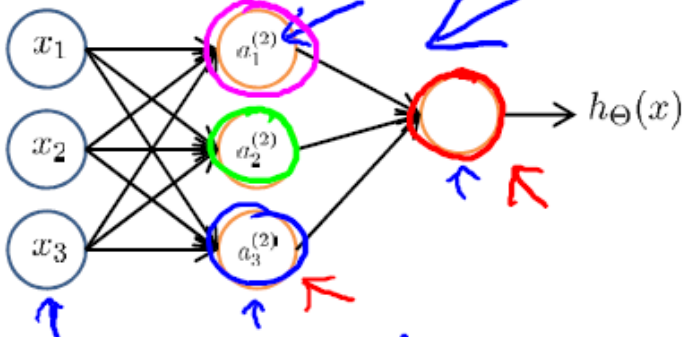
$$a_1^{(3)} = f\left(w_{11}^{(2)} a_1^{(2)} + w_{12}^{(2)} a_2^{(2)} + w_{13}^{(2)} a_3^{(2)} + b_1^{(2)}\right)$$

Deep Neural Network



Deep Neural Network

Neural Network



→ $a_i^{(j)}$ = "activation" of unit i in layer j

→ $\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

$\Theta^{(1)} \in \mathbb{R}^{3\times 4}$

$h_\Theta(x)$

→ $a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$

→ $a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$

→ $a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$

$\Theta^{(2)}$

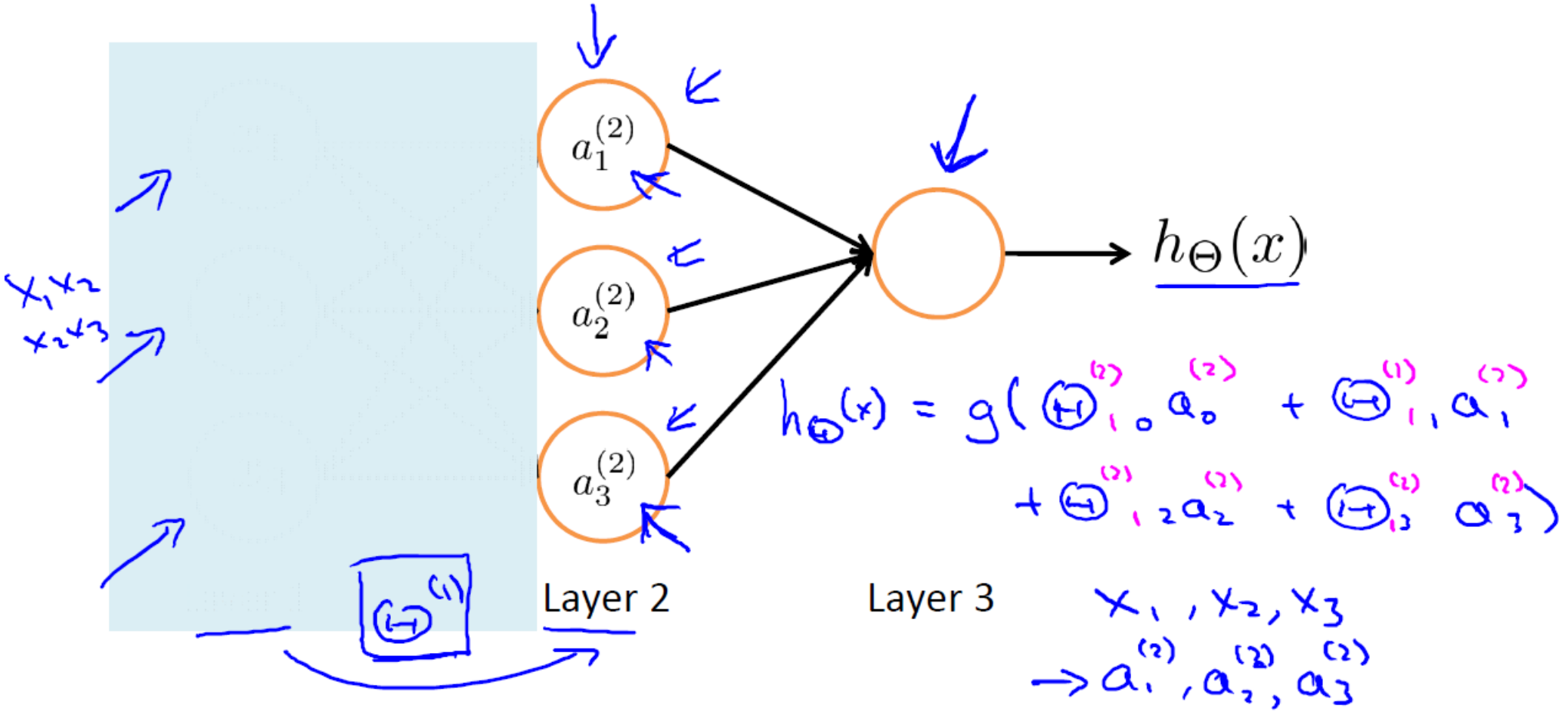
→ $h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$

→ If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.

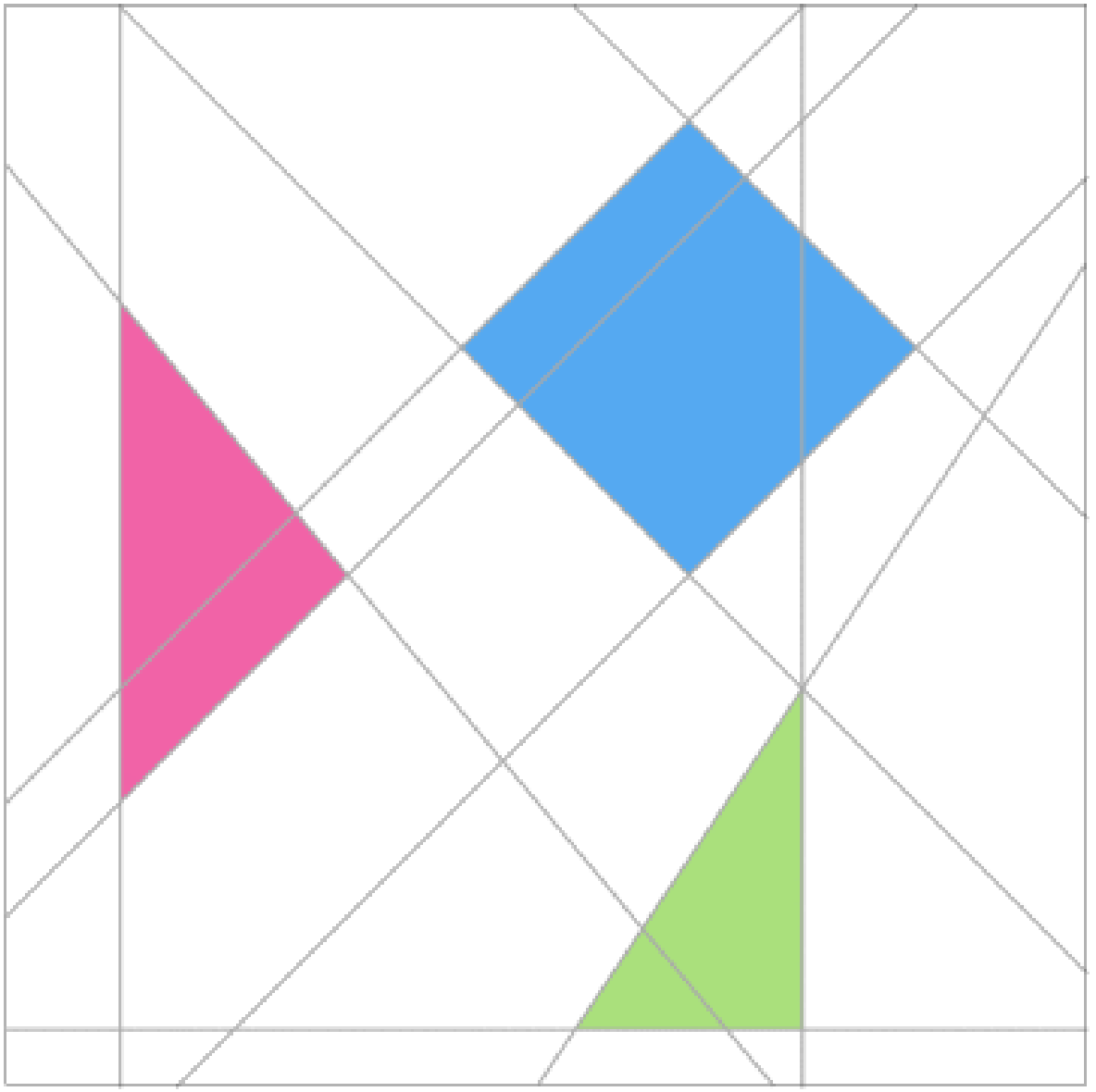
$s_{j+1} \times (s_j + 1)$

Deep Neural Network

Neural Network learning its own features



Deep Neural Network

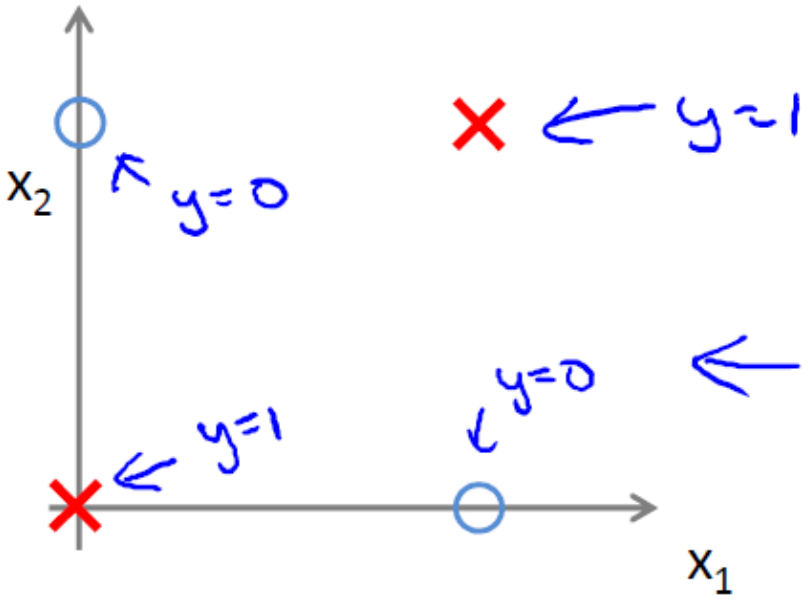


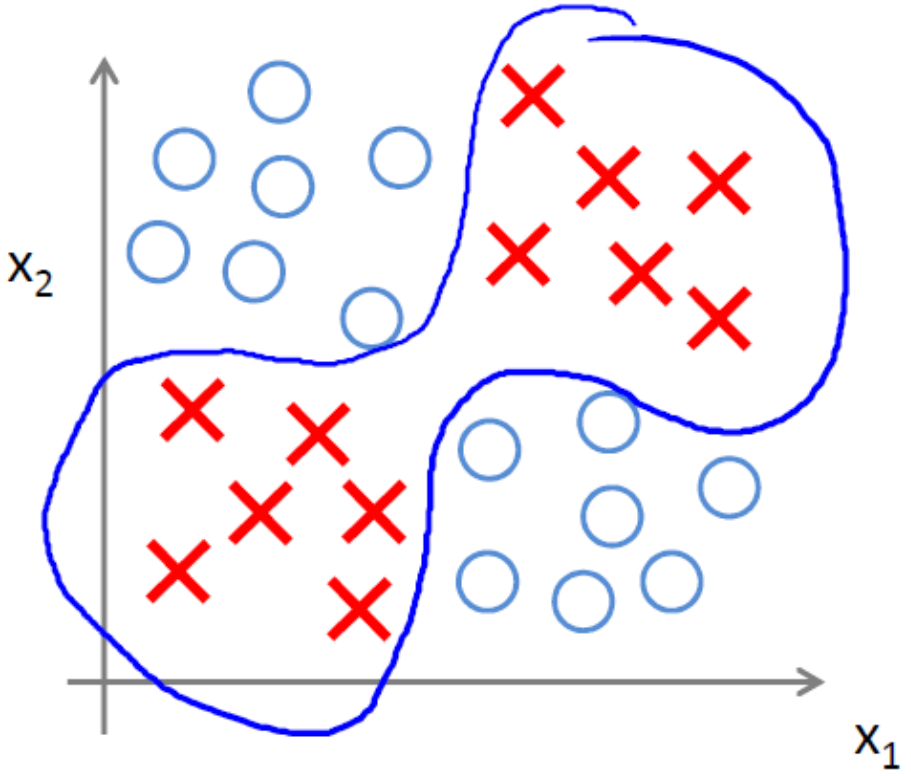
Deep Neural Network

Deep Neural Network

Non-linear classification example: XOR/XNOR

x_1, x_2 are binary (0 or 1).





$y = x_1 \text{ XOR } x_2$

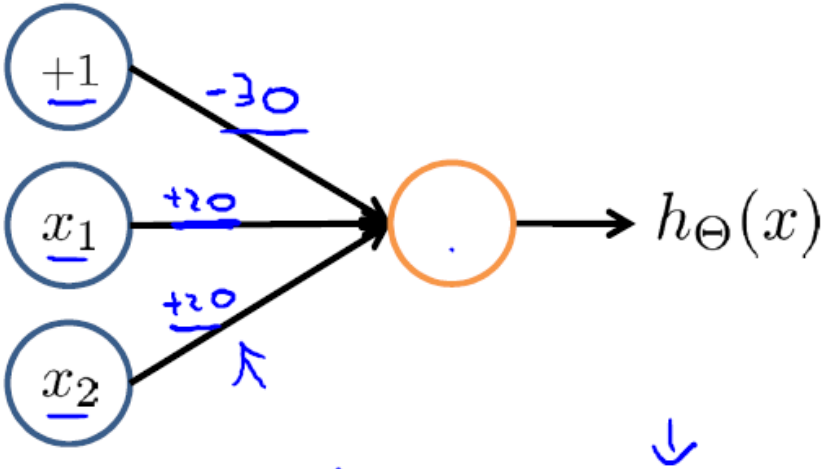
$x_1 \text{ XNOR } x_2$

$\text{NOT } (x_1 \text{ XOR } x_2)$

Deep Neural Network

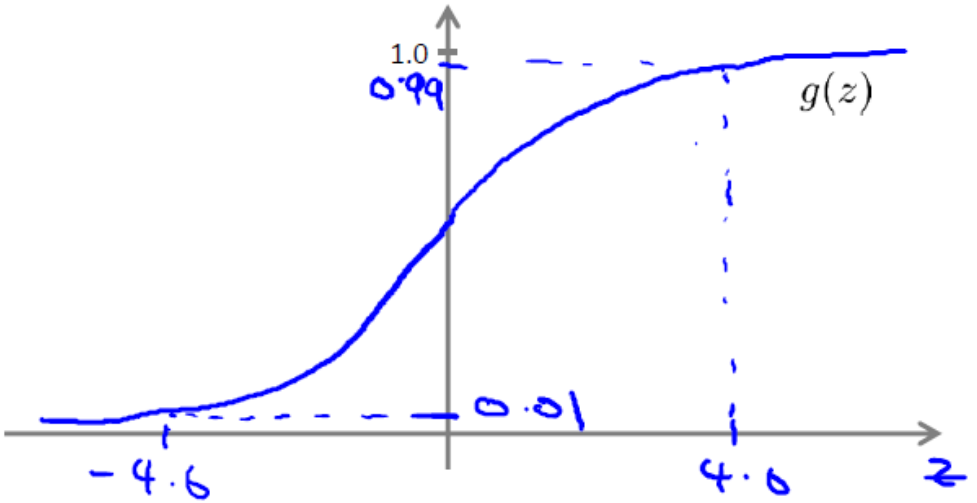
Simple example: AND

- $x_1, x_2 \in \{0, 1\}$
- $y = x_1$ AND x_2



$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$

$\Theta^{(1)}_{10}$ $\Theta^{(1)}_{11}$ $\Theta^{(1)}_{12}$

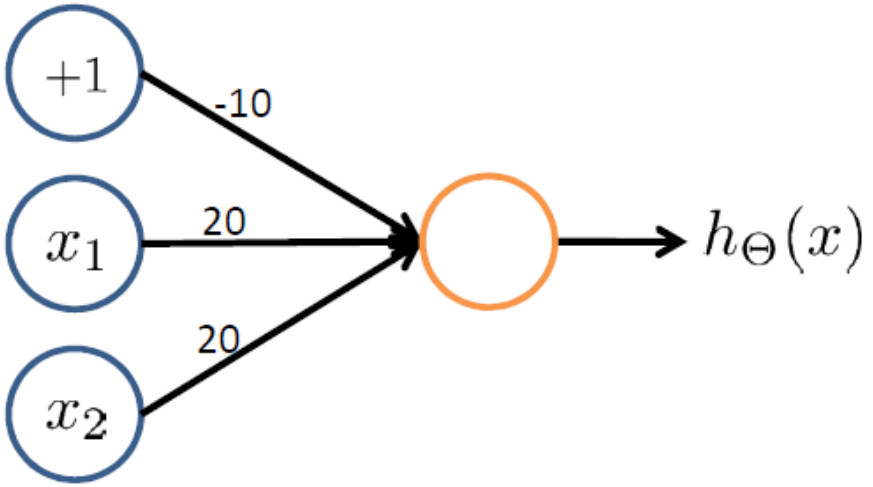


x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

$h_\Theta(x) \approx x_1$ AND x_2

Deep Neural Network

Example: OR function



$g(-10 + 20x_1 + 20x_2)$

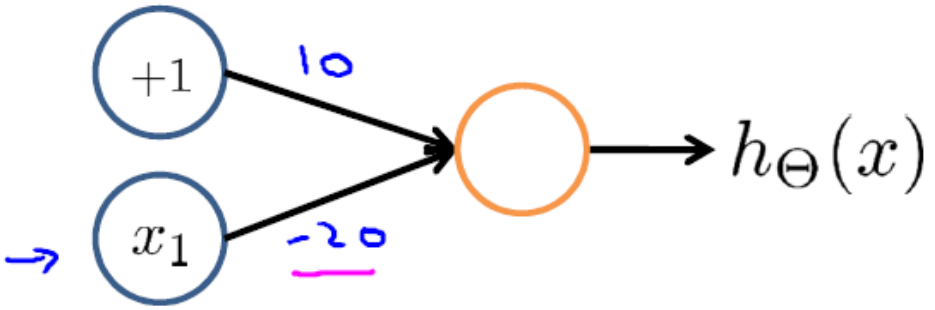
x_1	x_2	$h_\Theta(x)$
0	0	$g(-10) \approx 0$
0	1	$g(10) \approx 1$
1	0	≈ 1
1	1	≈ 1

Deep Neural Network

→ x_1 AND x_2 → x_1 OR x_2 {0,1}.

Negation:

NOT x_1



x_1	$h_\Theta(x)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

$h_\Theta(x) = g(10 - 20x_1)$

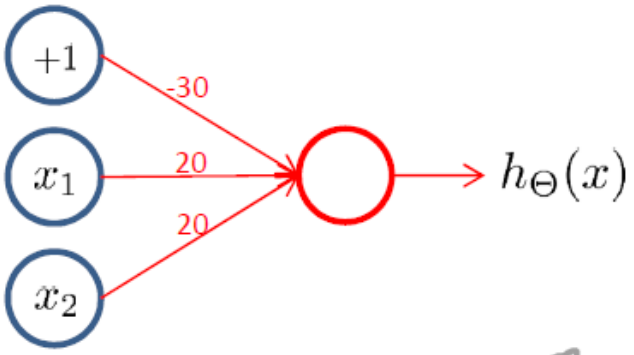
→ (NOT x_1) AND (NOT x_2)

(=1 if and only if

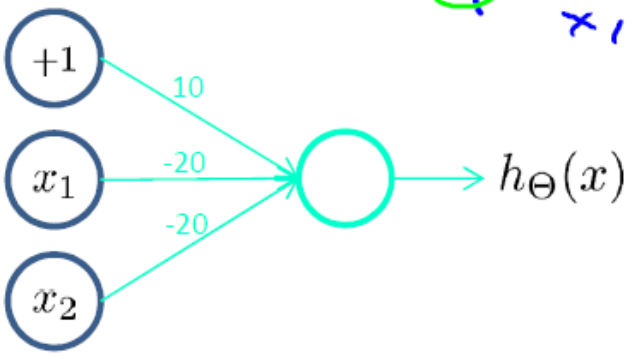
→ $x_1 = x_2 = 0$

Deep Neural Network

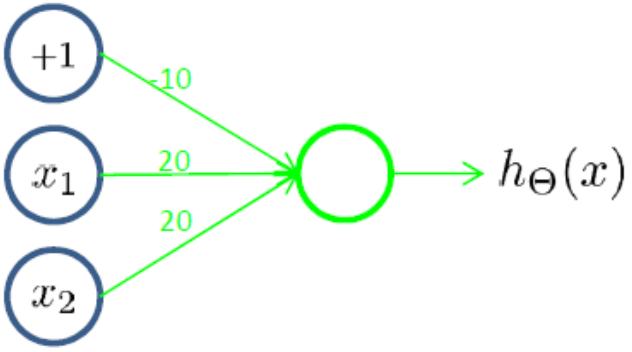
Putting it together: x_1 XNOR x_2



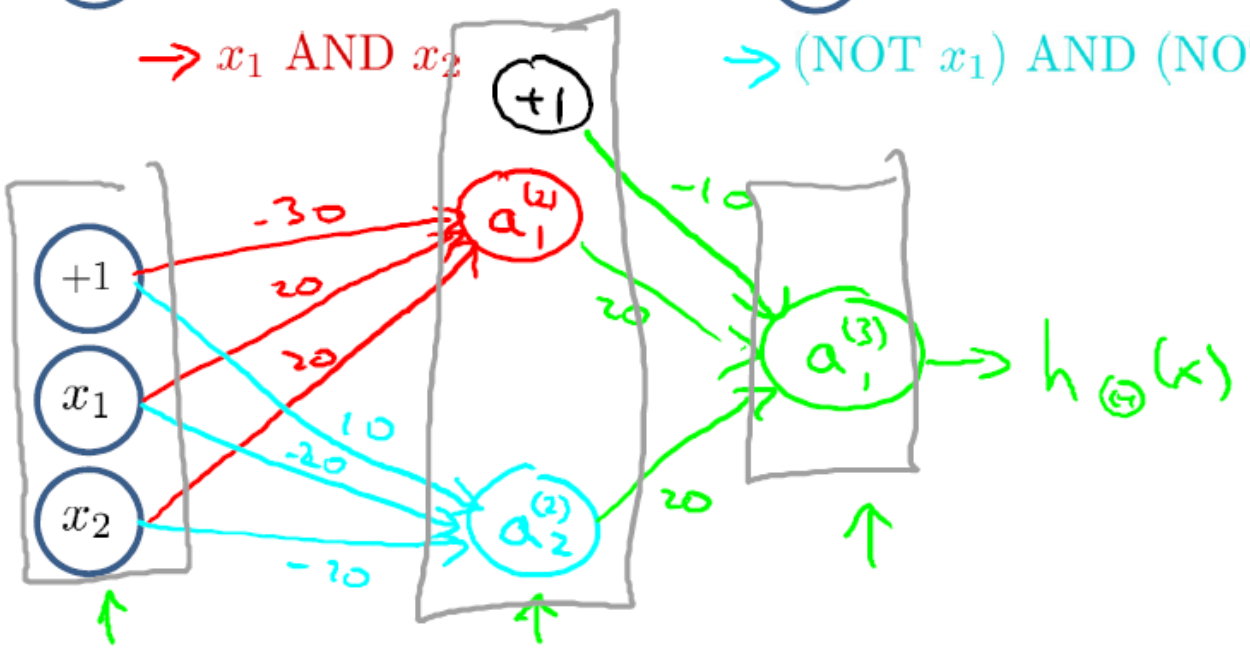
→ x_1 AND x_2



→ (NOT x_1) AND (NOT x_2)

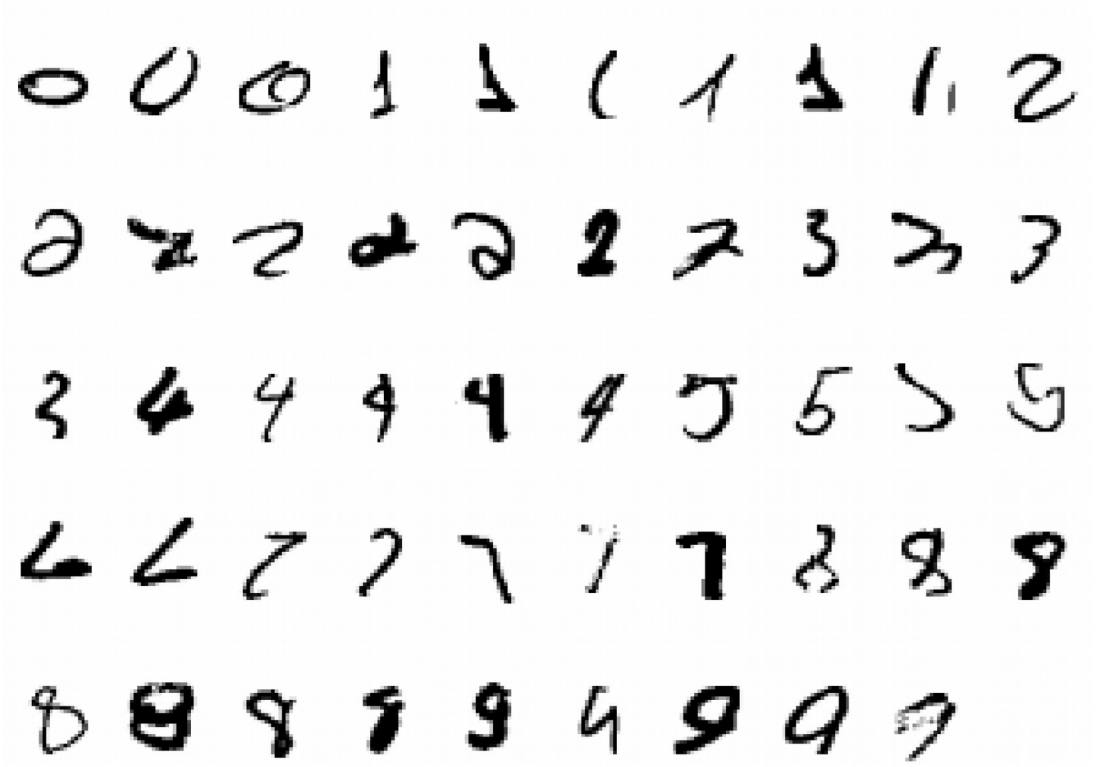


→ x_1 OR x_2



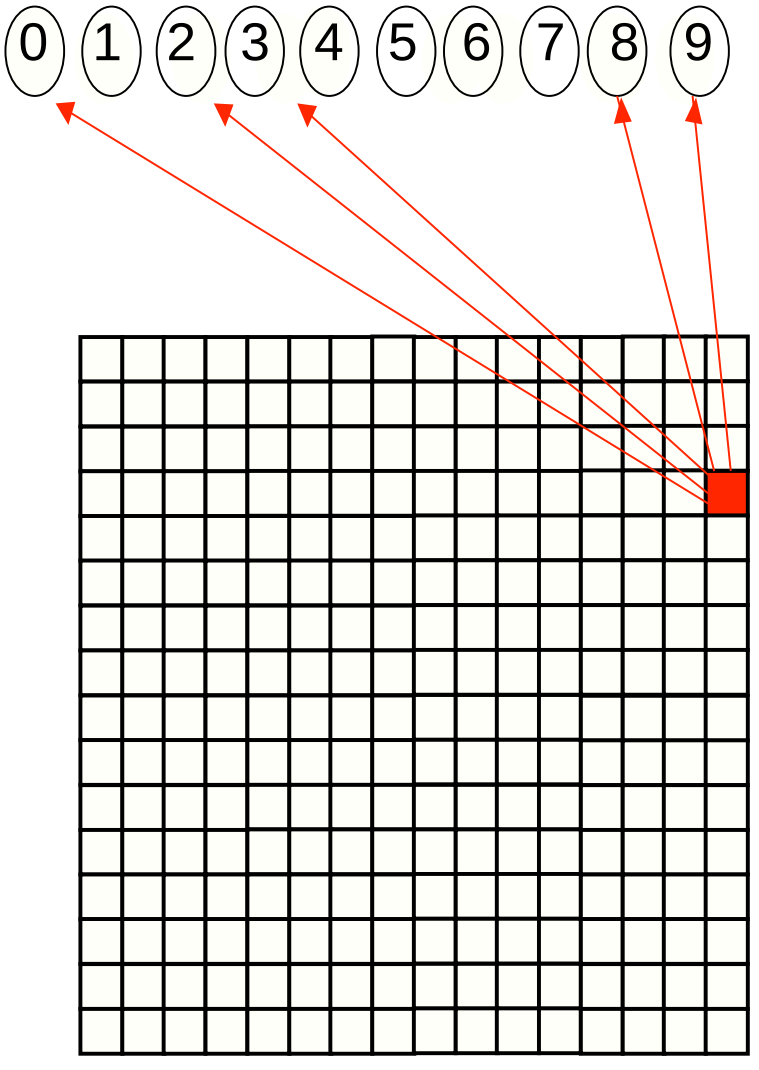
x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

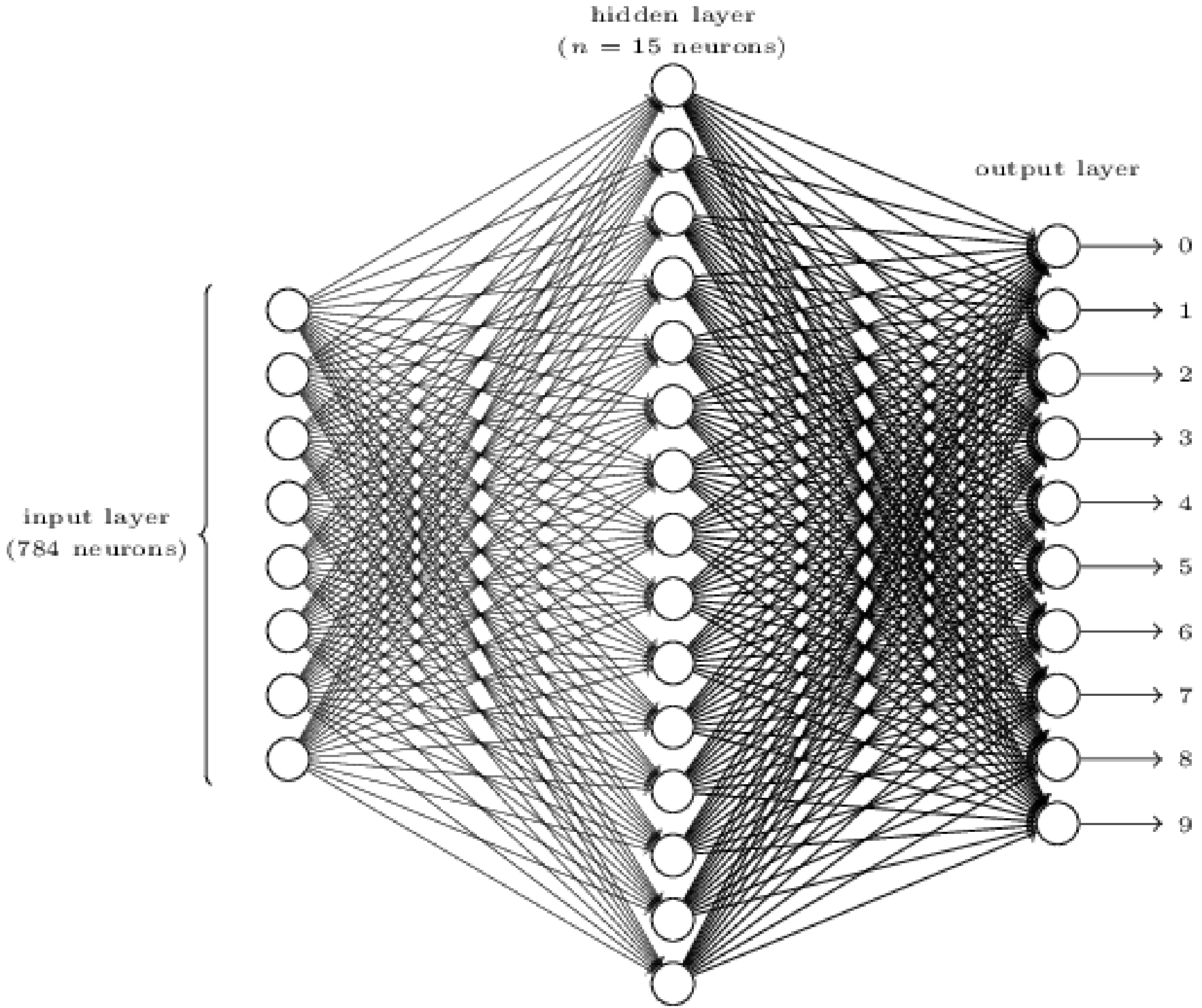
It is very hard to say what makes a 2



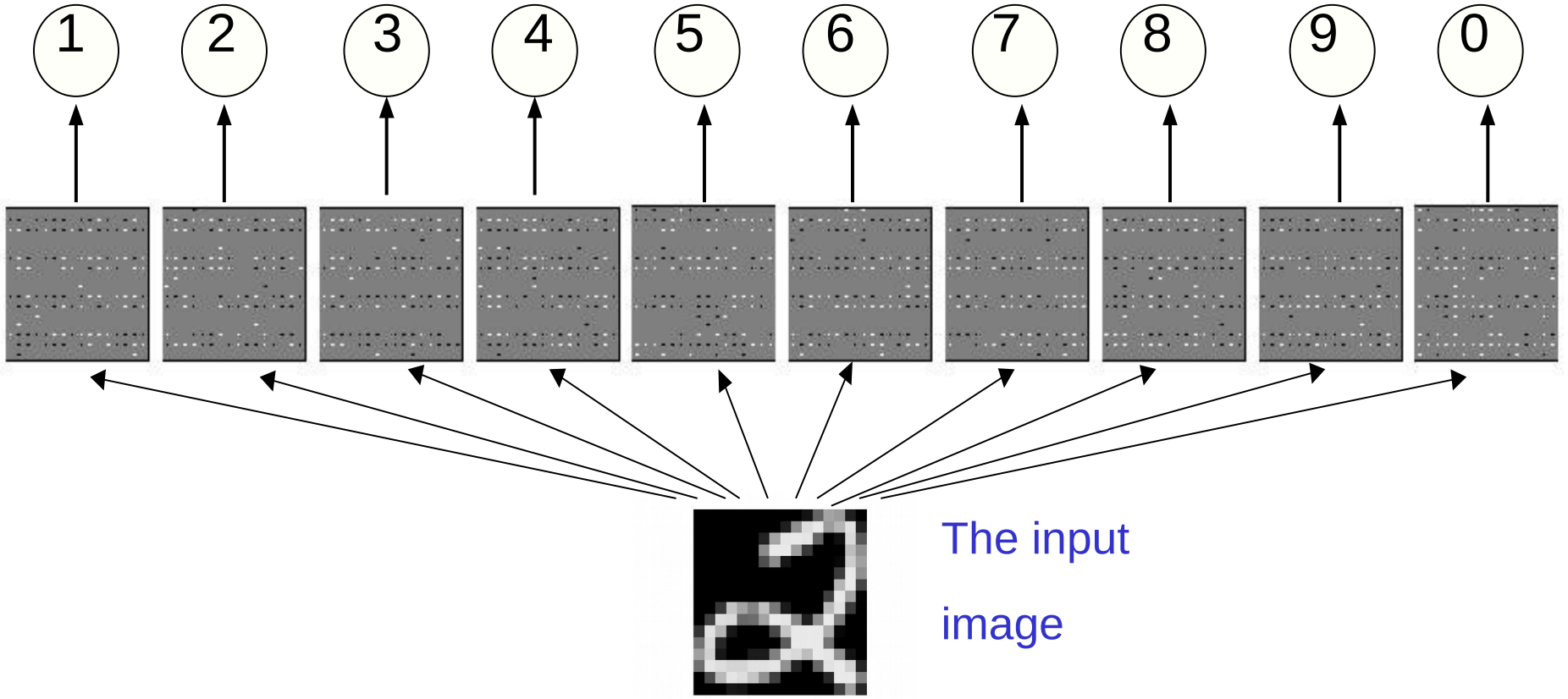
A very simple way to recognize handwritten shapes

- ? Consider a neural network with two layers of neurons.
  - ? neurons in the top layer represent known shapes.
  - ? neurons in the bottom layer represent pixel intensities.
- ? A pixel gets to vote if it has ink on it.
  - ? Each inked pixel can vote for several different shapes.
- ? The shape that gets the most votes wins.




hidden layer
(n = 15 neurons)
output layer
input layer
(784 neurons)
0
1
2
3
4
5
6
7
8
9

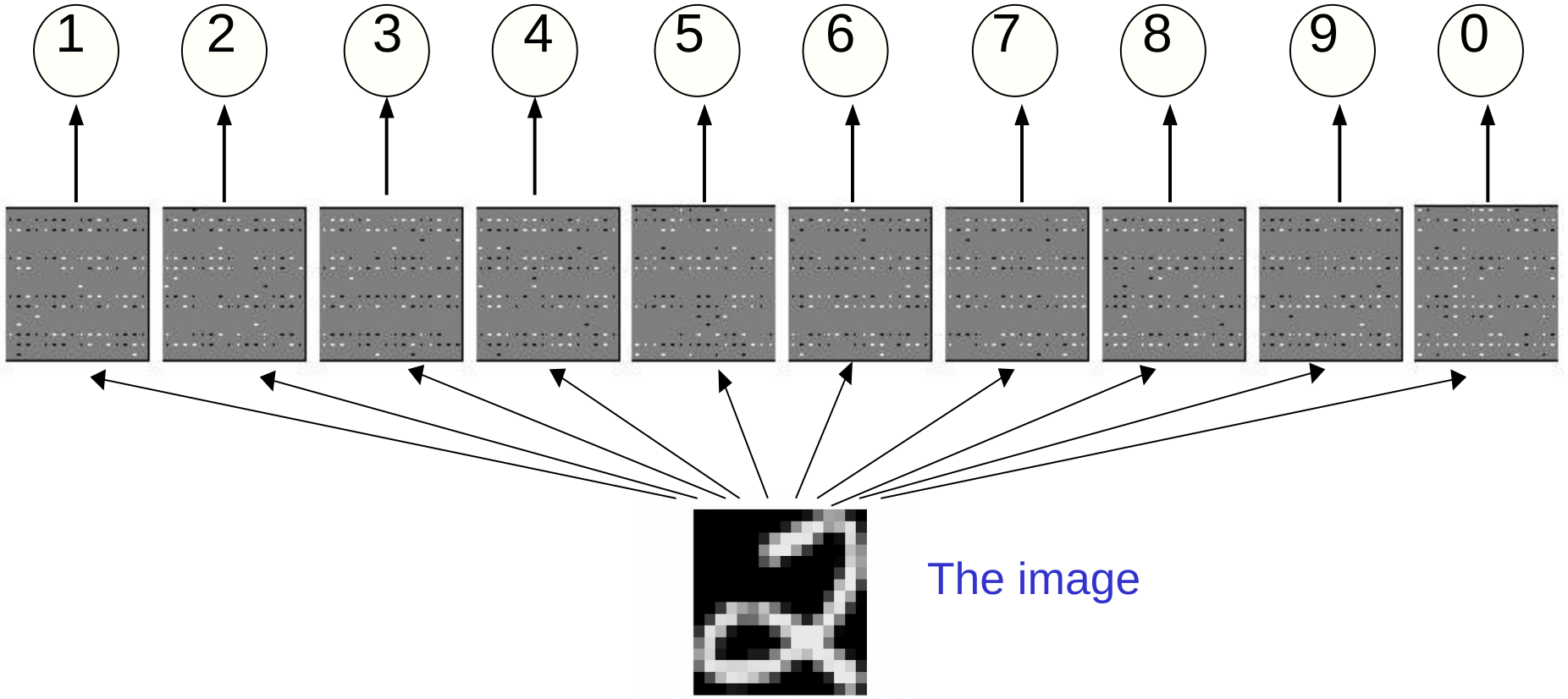
How to display the weights



Give each output unit its own “map” of the input image and display the weight coming from each pixel in the location of that pixel in the map.

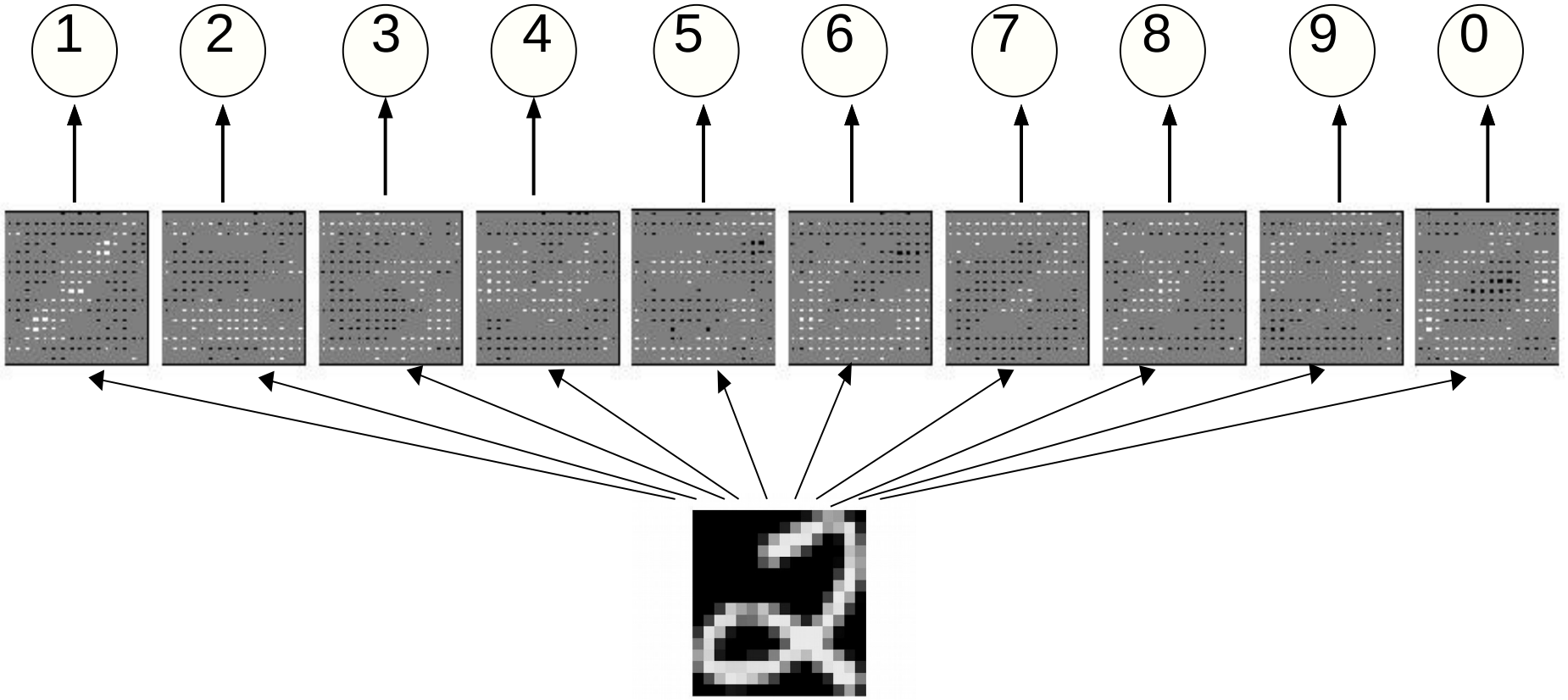
Use a black or white blob with the area representing the magnitude of the weight and the color representing the sign.

How to learn the weights

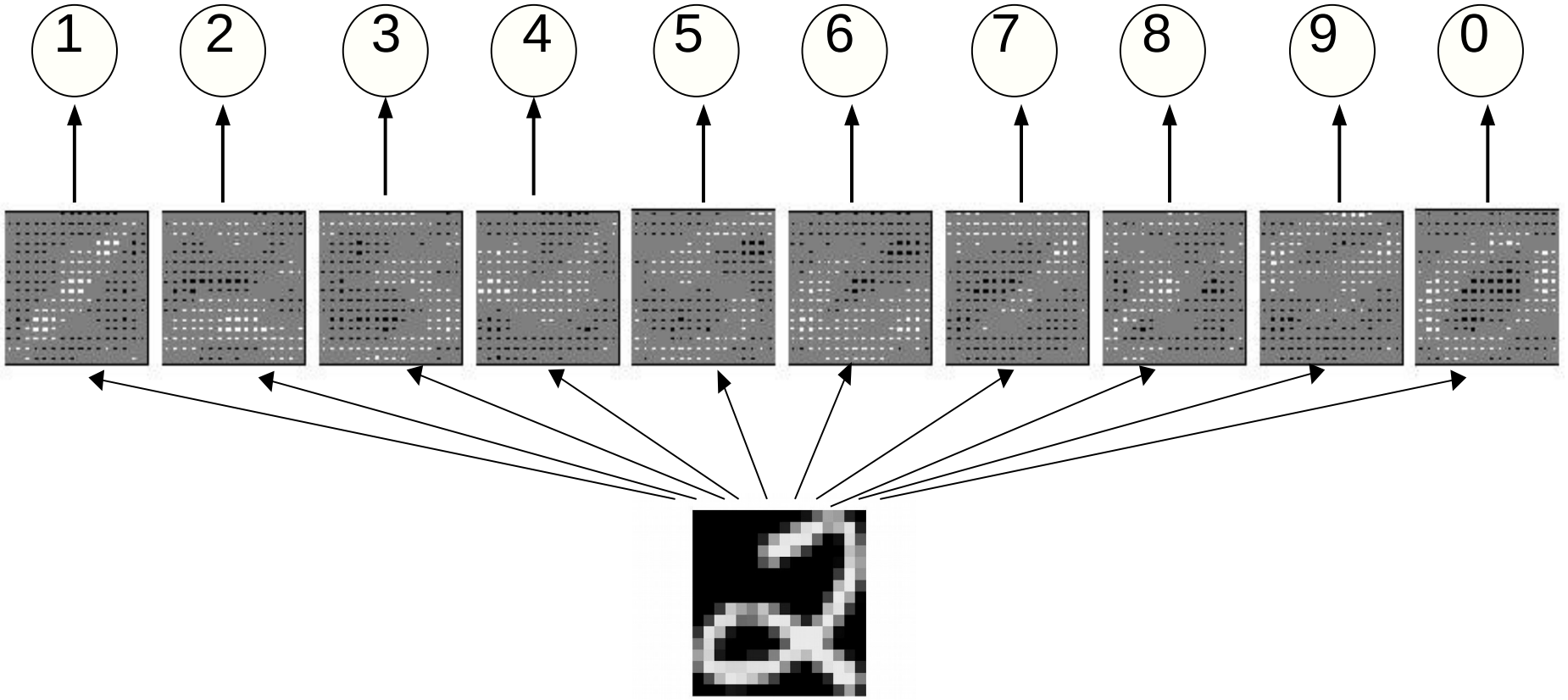


Show the network an image and increment the weights from active pixels to the correct class.

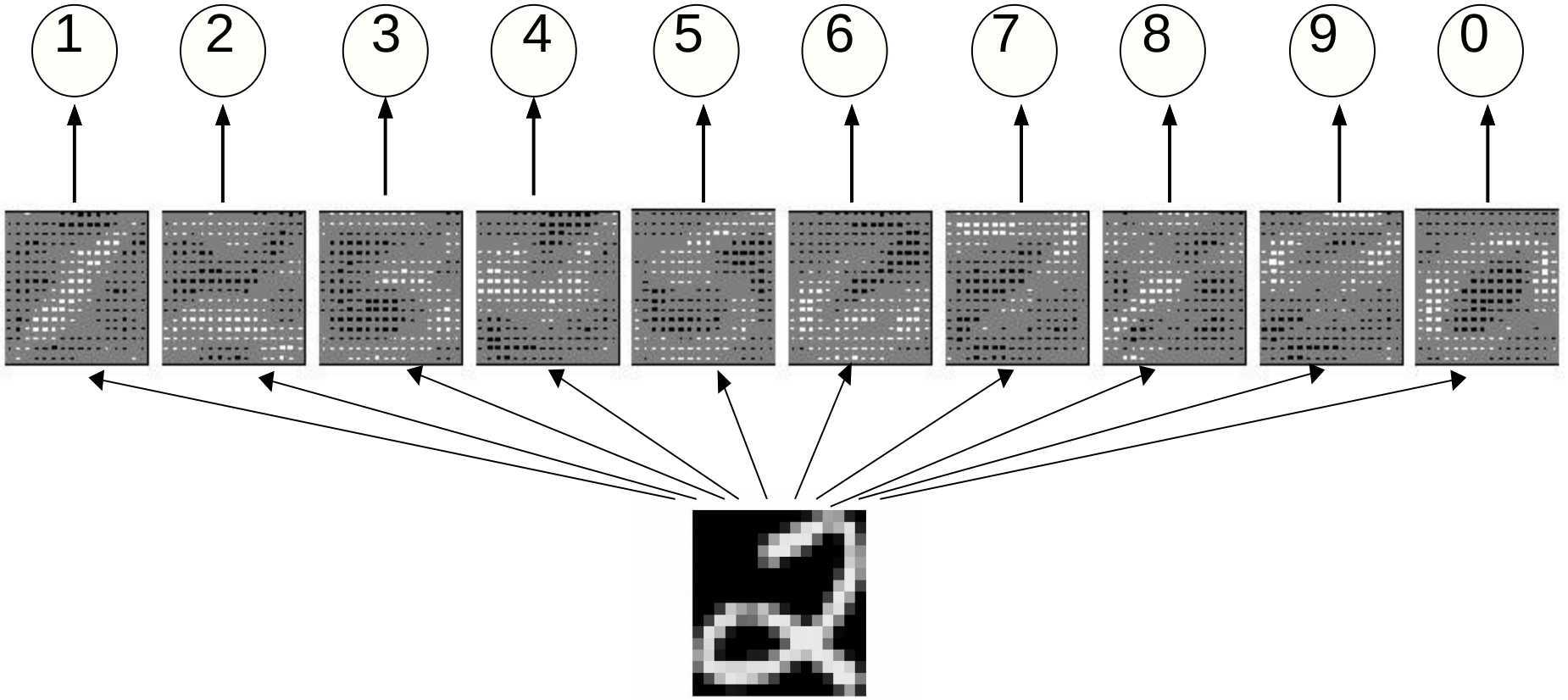
Then decrement the weights from active pixels to whatever class the network guesses.



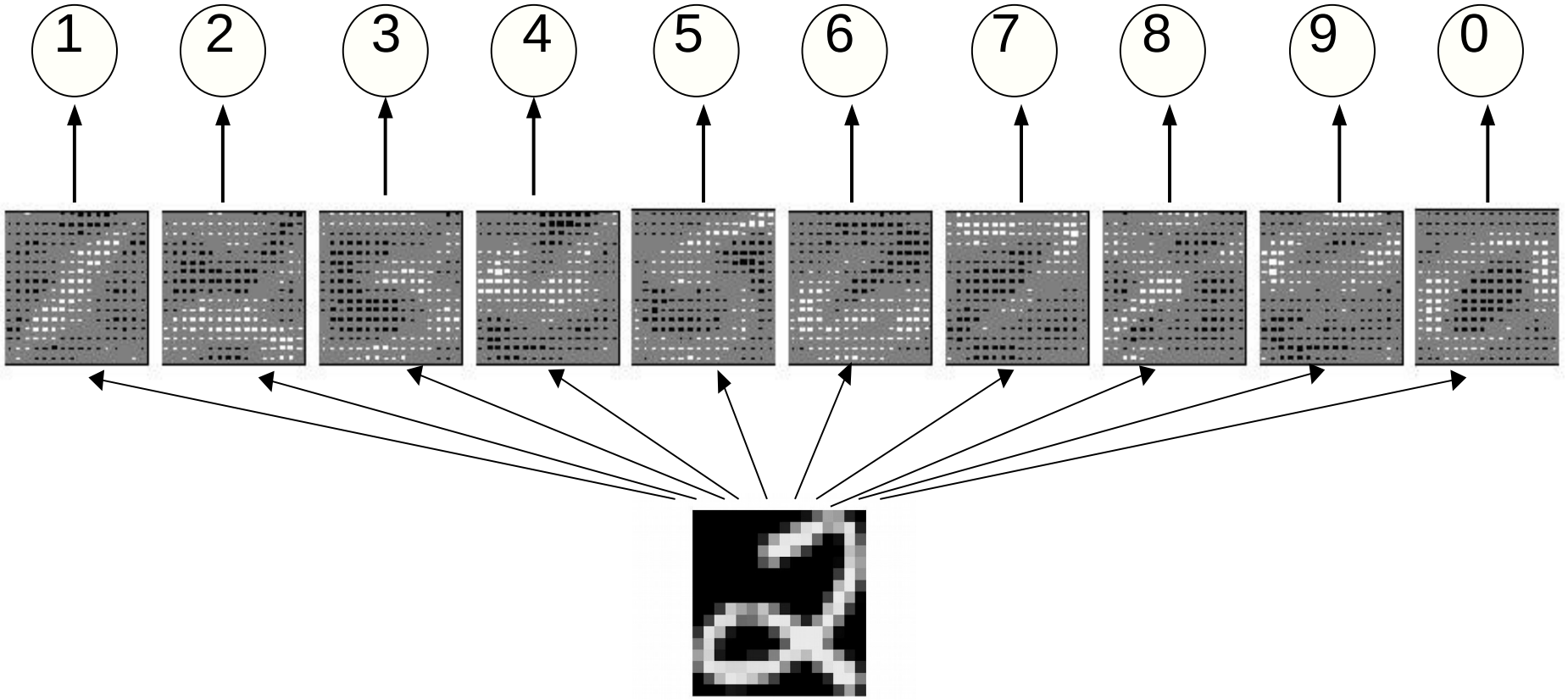
The image



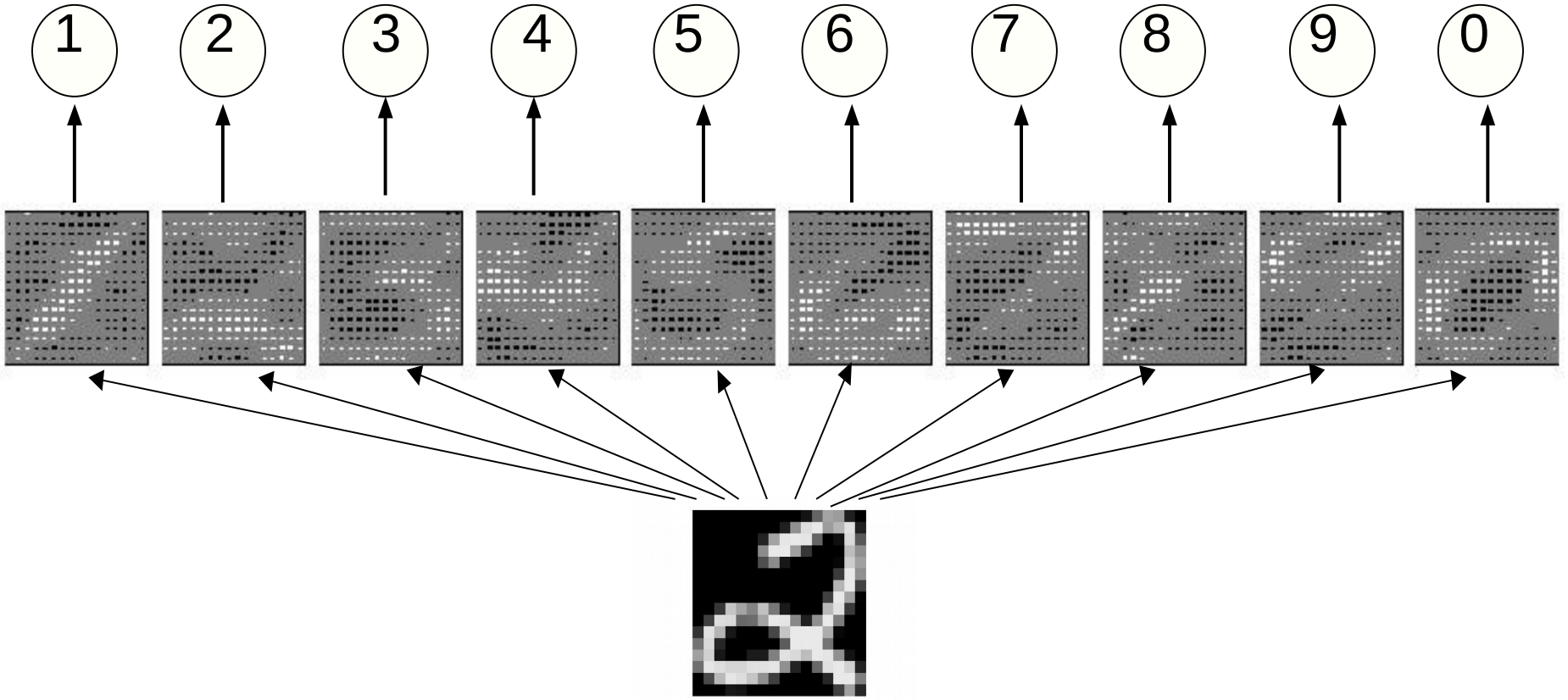
The image



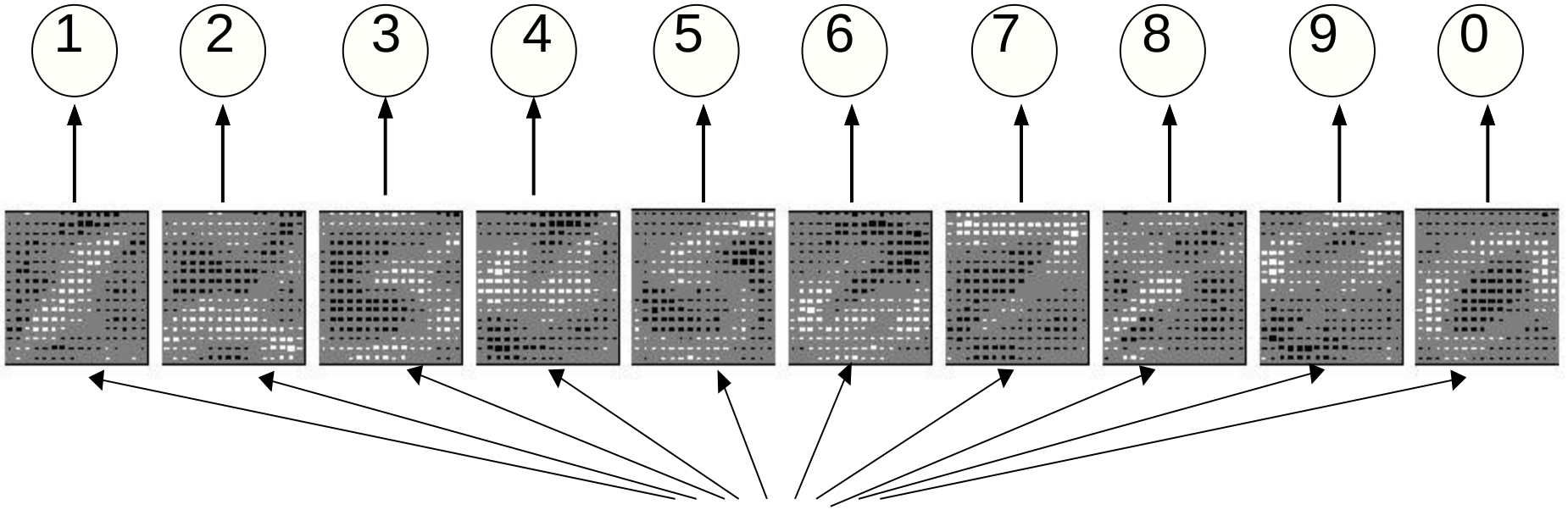
The image



The image

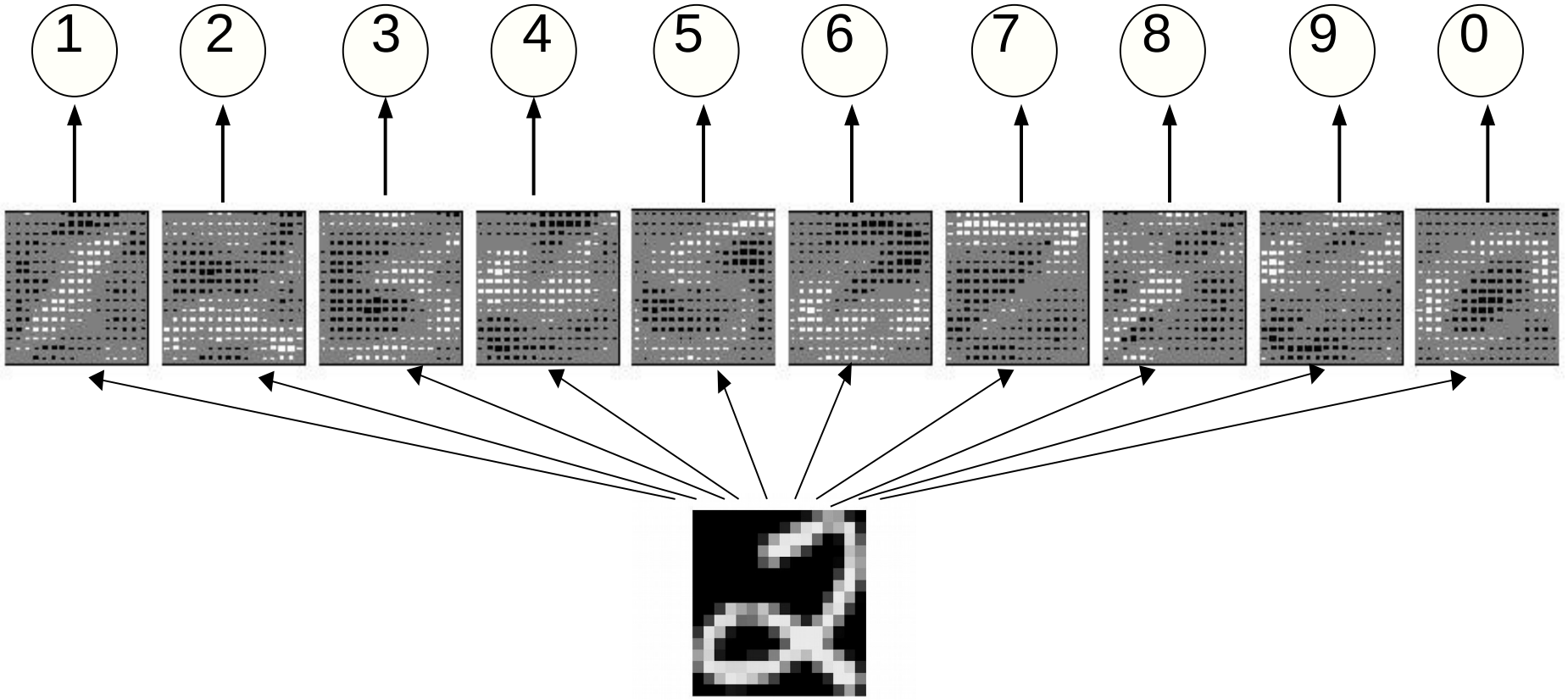


The image

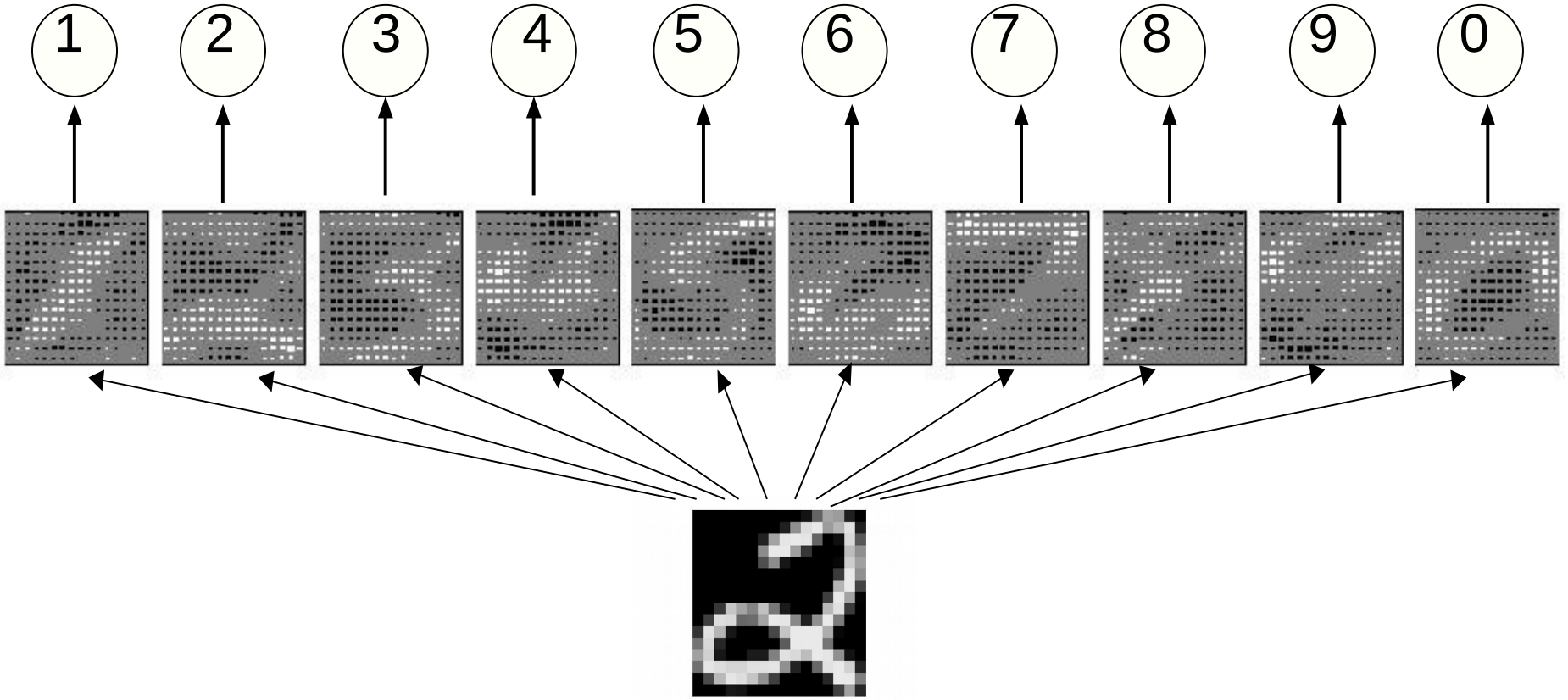


The image

The learned weights



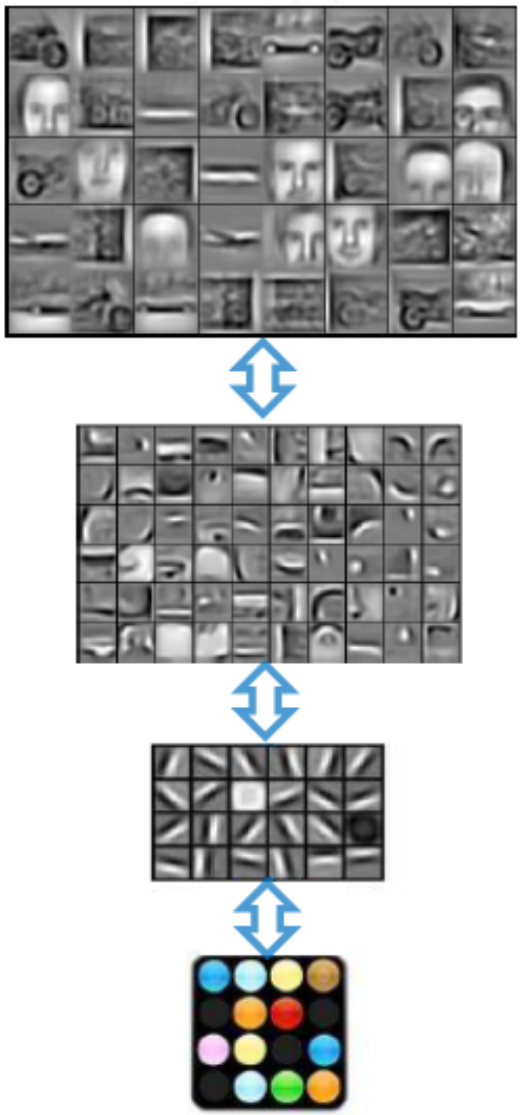
The image

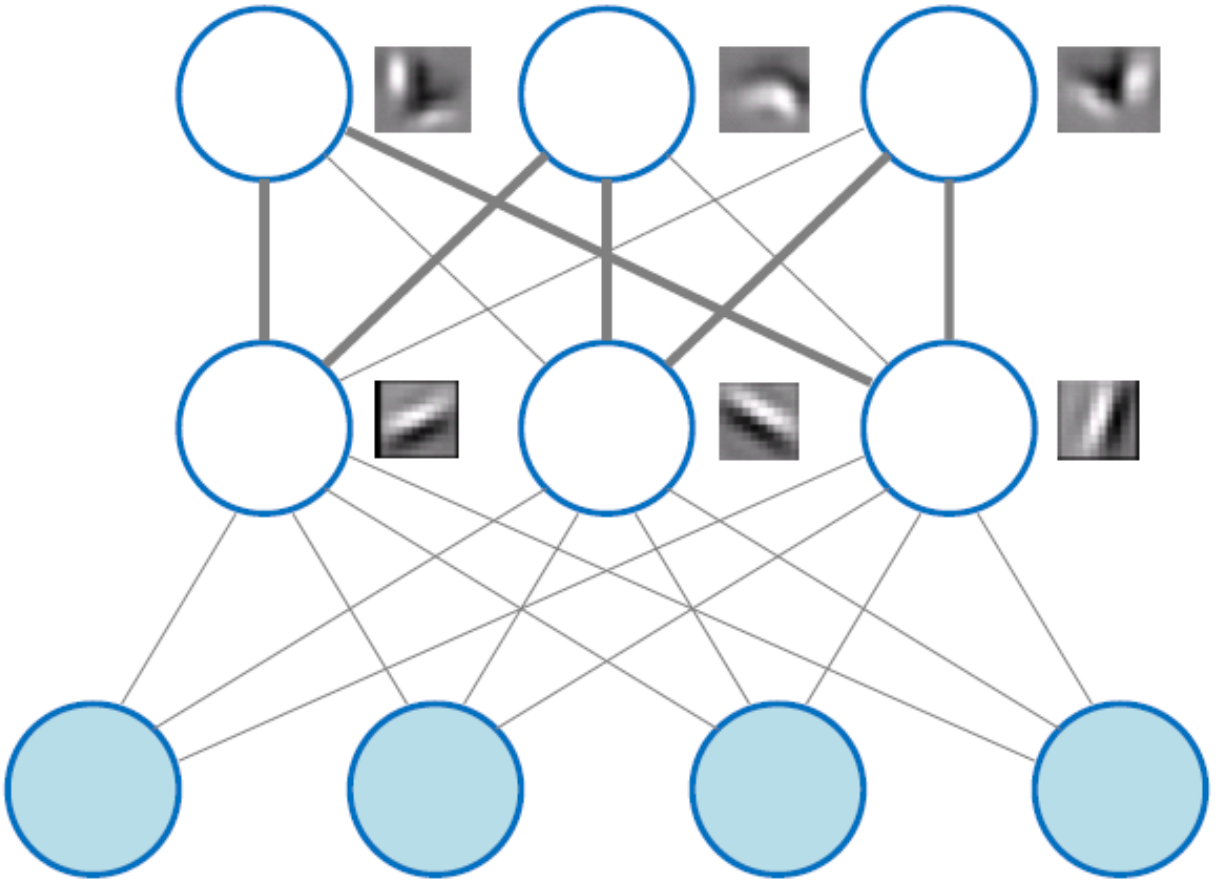


The image

Learning Feature Hierarchy

- Deep Learning
  - Deep architectures can be representationally efficient.
  - Natural progression from low level to high level structures.
  - Can share the lower-level representations for multiple tasks.





Higher layer: DBNs
(Combinations
of edges)

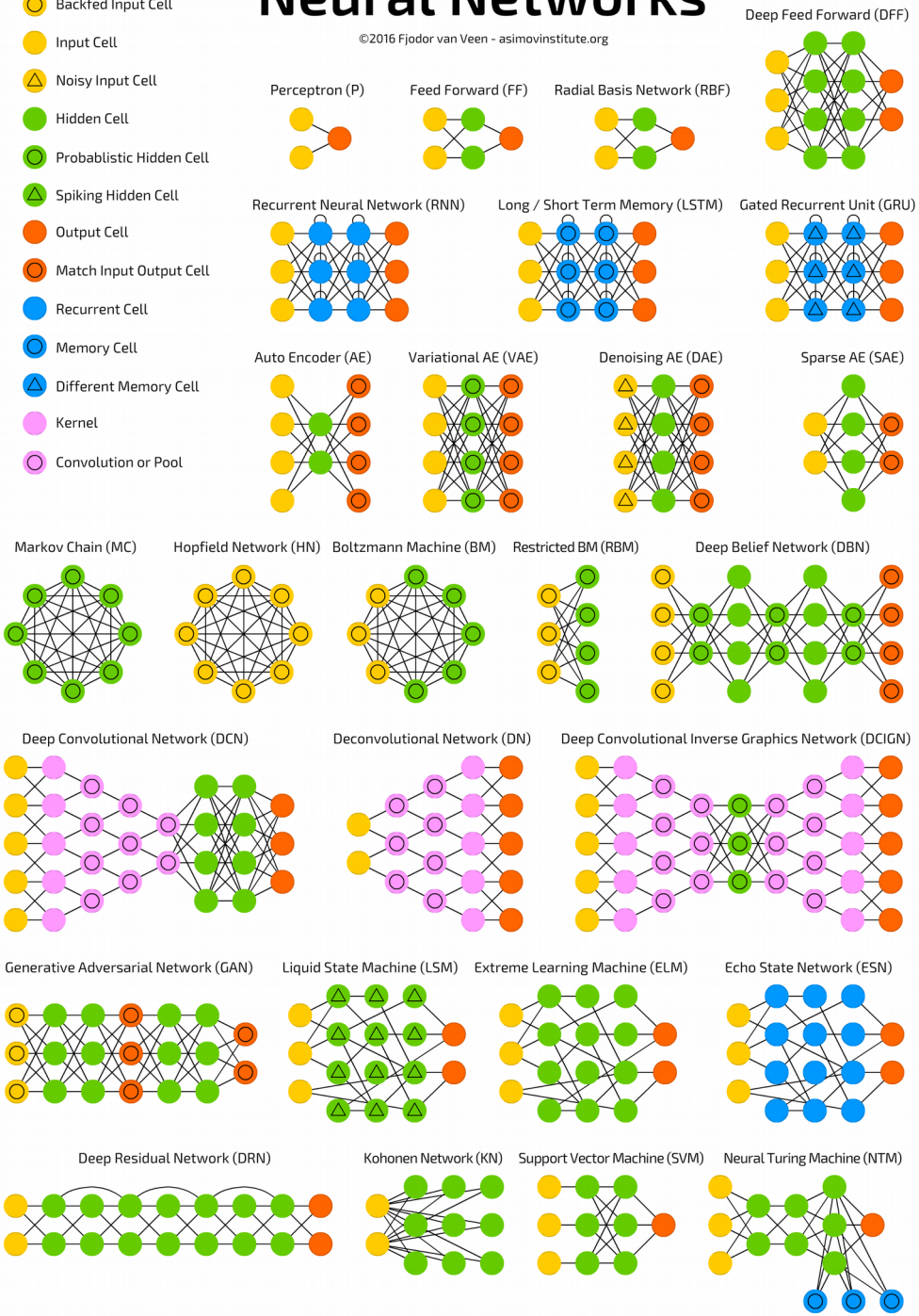
First layer: RBMs
(edges)

Input image patch
(pixels)

A mostly complete chart of

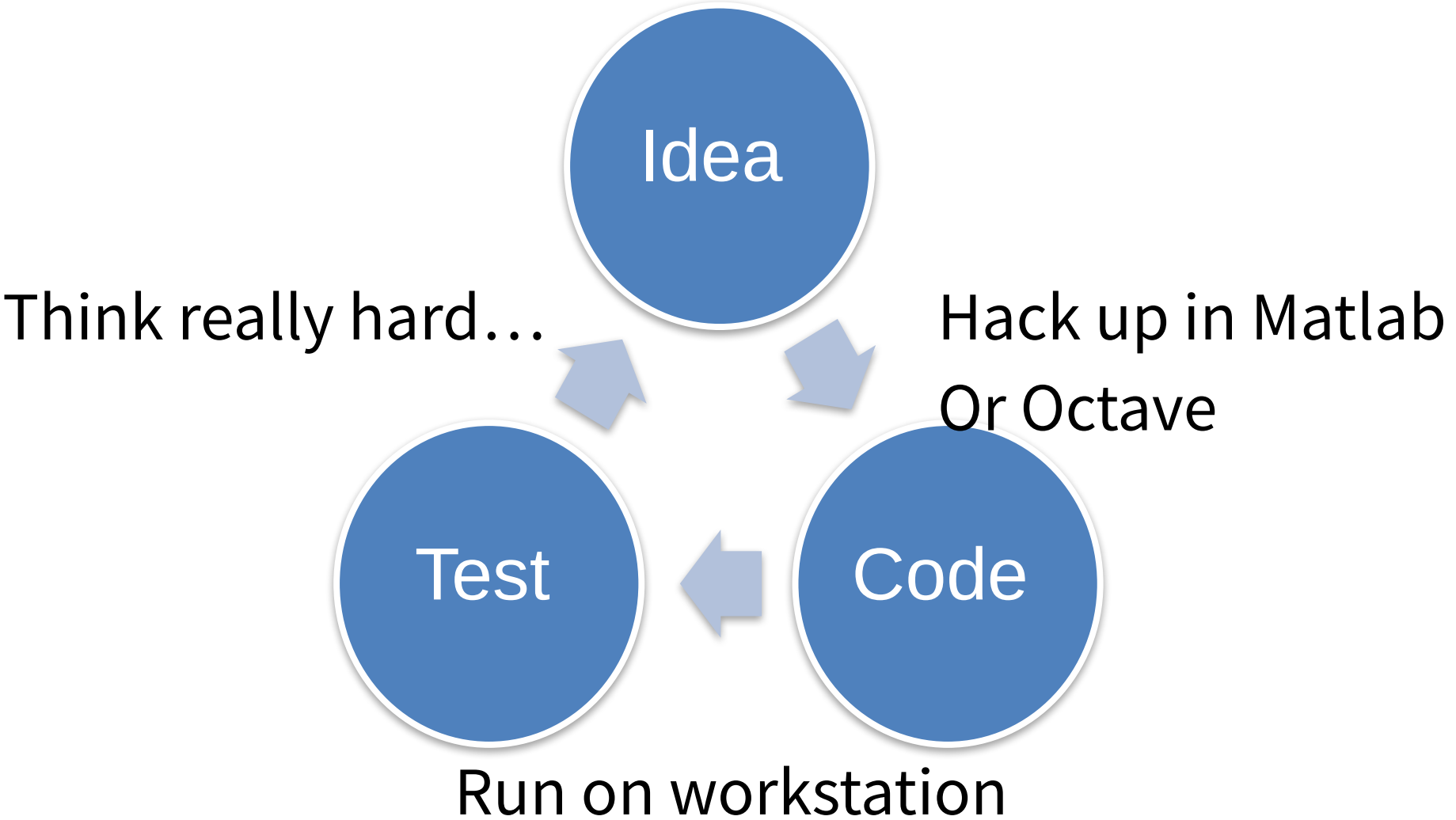
Neural Networks

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool



AI in practice

- Enormous amounts of research time spent inventing new features.

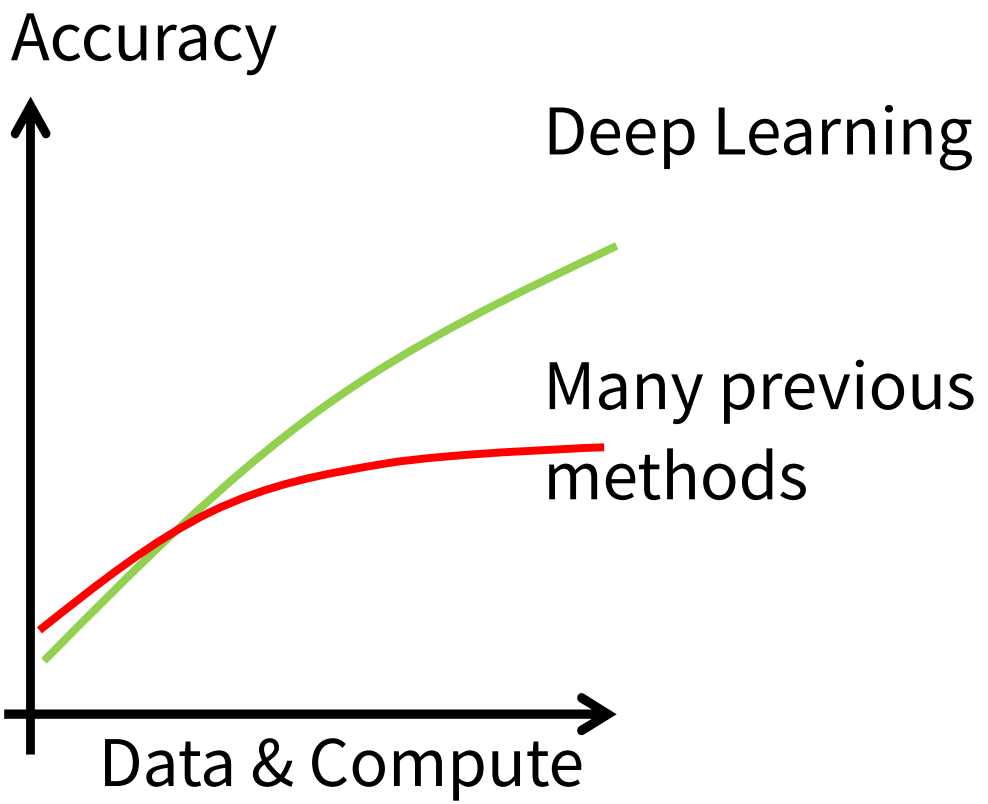


Why Deep Learning?

1. Scale Matters
   - Bigger models usually win

2. Data Matters
   - More data means less cleverness necessary

3. Productivity Matters
   - Teams with better tools can try out more ideas

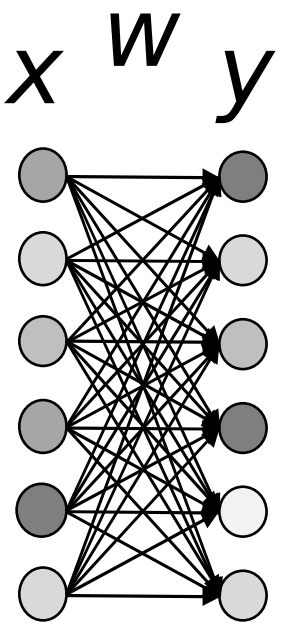


Scaling up

- Make progress on AI by focusing on systems
  - Make models bigger
  - Tackle more data
  - Reduce research cycle time
    - Accelerate large-scale experiments

Training Deep Neural Networks

$$y_j = f\left(\sum_i w_{ij}\, x_i\right)$$



- Computation dominated by dot products
- Multiple inputs, multiple outputs, batch means GEMM
  - Compute bound
- Convolutional layers even more compute bound

Computational Characteristics

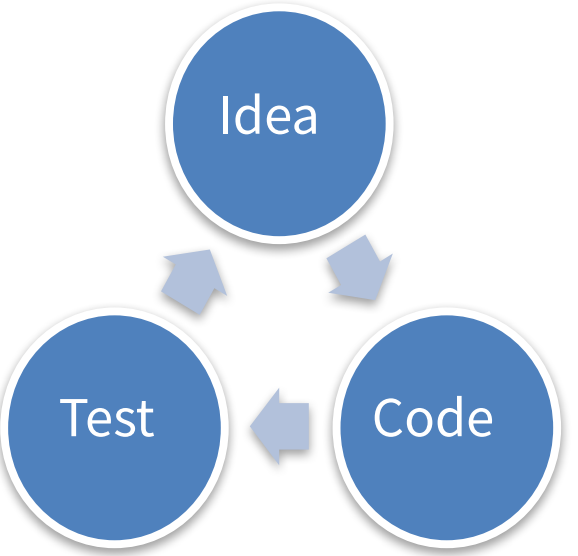
- High arithmetic intensity
  - Arithmetic operations / byte of data
  - O(Exaflops) / O(Terabytes) : 10^6
    - In contrast, some other ML training jobs are O(Petaflops)/O(Petabytes) = 10^0

- Medium size datasets
  - Generally fit on 1 node
  - HDFS, fault tolerance, disk I/O not bottlenecks

Training 1 model: ~20 Exaflops

Deep Neural Network training is HPC

- Turnaround time is key
- Use most efficient hardware
  - Parallel, heterogeneous computing
  - Fast interconnect (PCIe, Infiniband)
- Push strong scalability
  - Models and data have to be of commensurate size



Infrastructure

- Software: CUDA, MPI, Majel (internal library)
- Hardware:

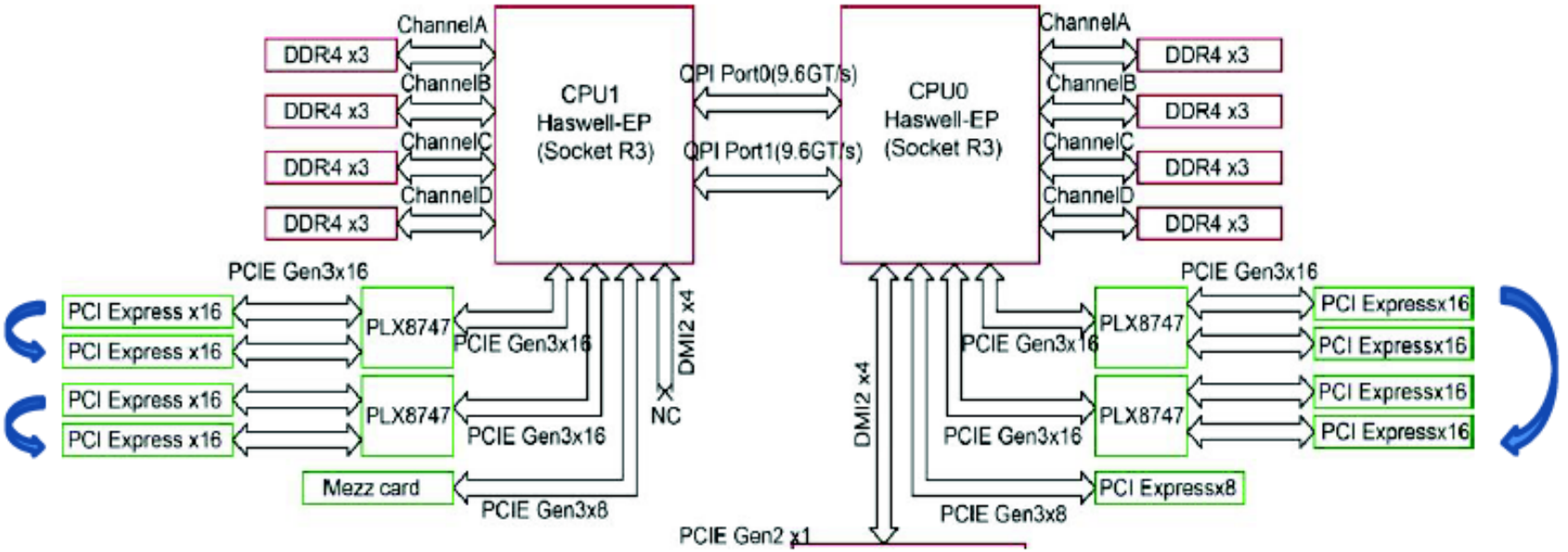
NVIDIA GeForce GTX Titan X

Titan X x8

Mellanox Interconnect

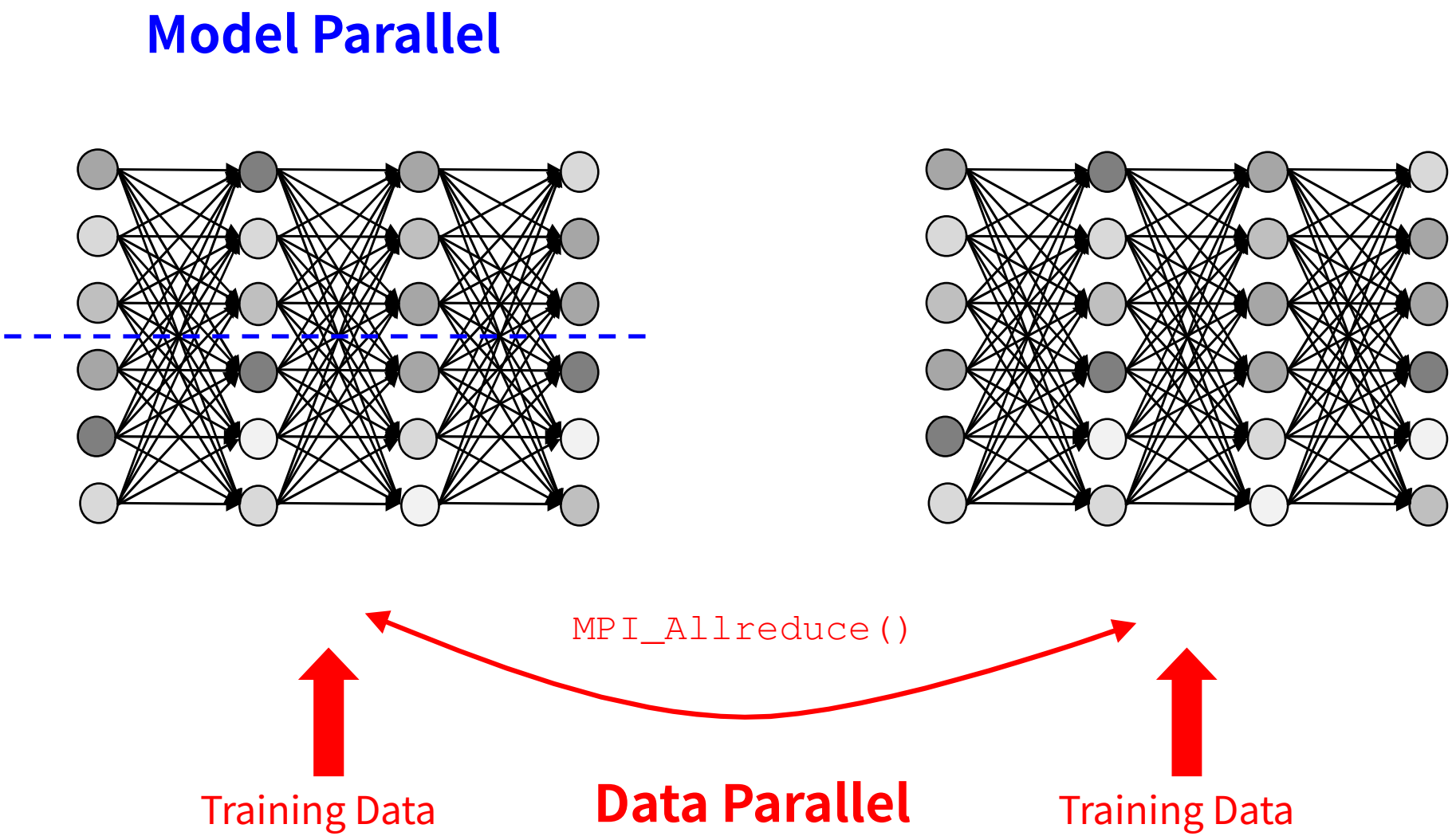
http://www.tyan.com

Node Architecture



- All pairs of GPUs communicate simultaneously over PCIe Gen 3 x16
- Groups of 4 GPUs form Peer to Peer domain
- Avoid moving data to CPUs or across QPI

Parallelism



Determinism

- Determinism very important
- So much randomness,
  hard to tell if you have a bug
- Networks train despite bugs,
  although accuracy impaired
- Reproducibility is important
  - For the usual scientific reasons
  - Progress not possible without reproducibility

Conclusions

- Computationally dense processors (like GPUs) required
- Programmability
  - We don't know the algorithms of the future
- Lower precision
  - But not too low
  - Interesting algorithm/dataset engineering here
- We need better support for multi-GPU
  - E.g. Atomics between GPUs, collectives
  - Looking forward to NVLink

Conclusions

- Deep Learning is solving many hard problems
- Training deep neural networks is an HPC problem
- Scaling brings AI progress!