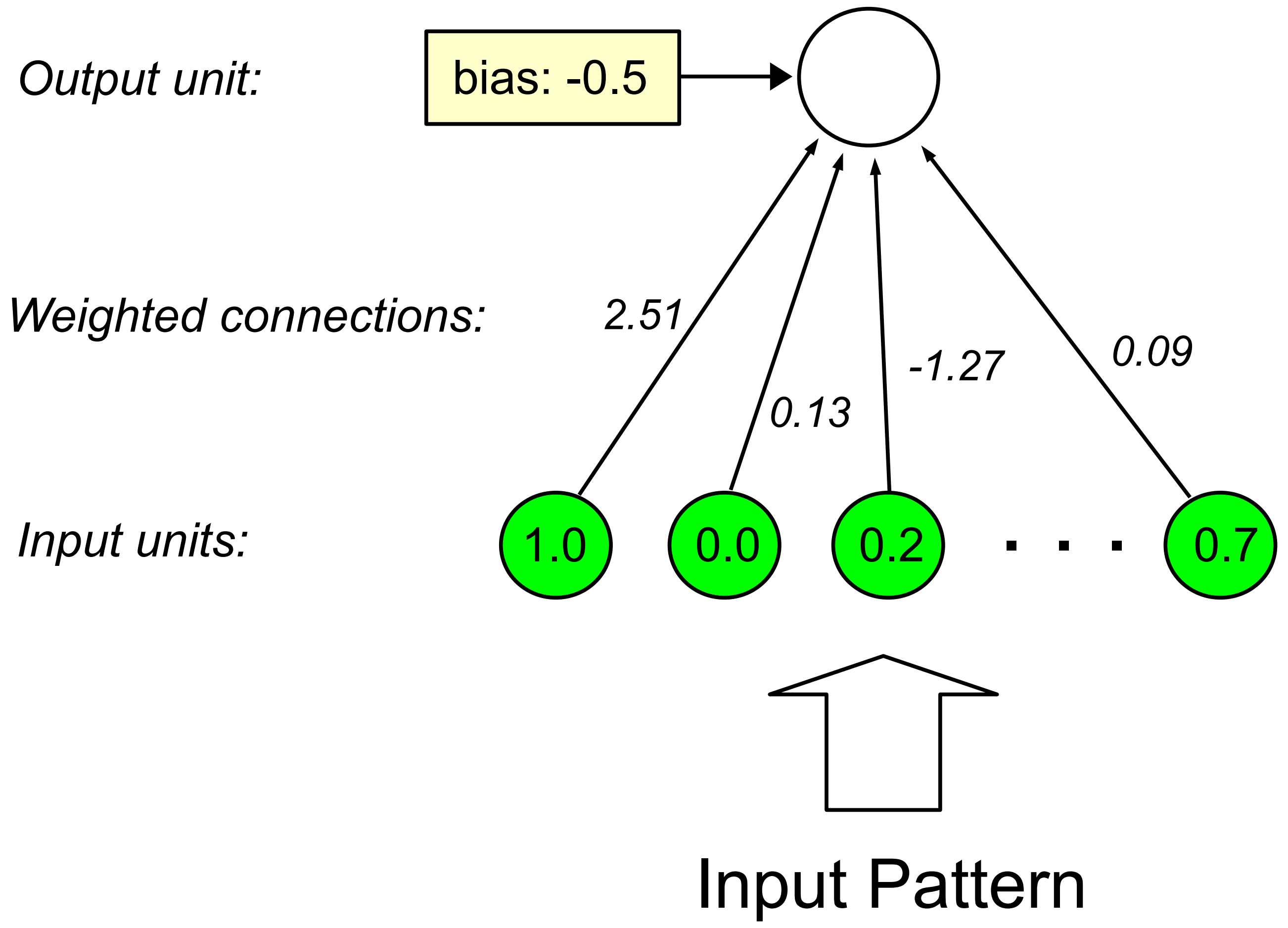
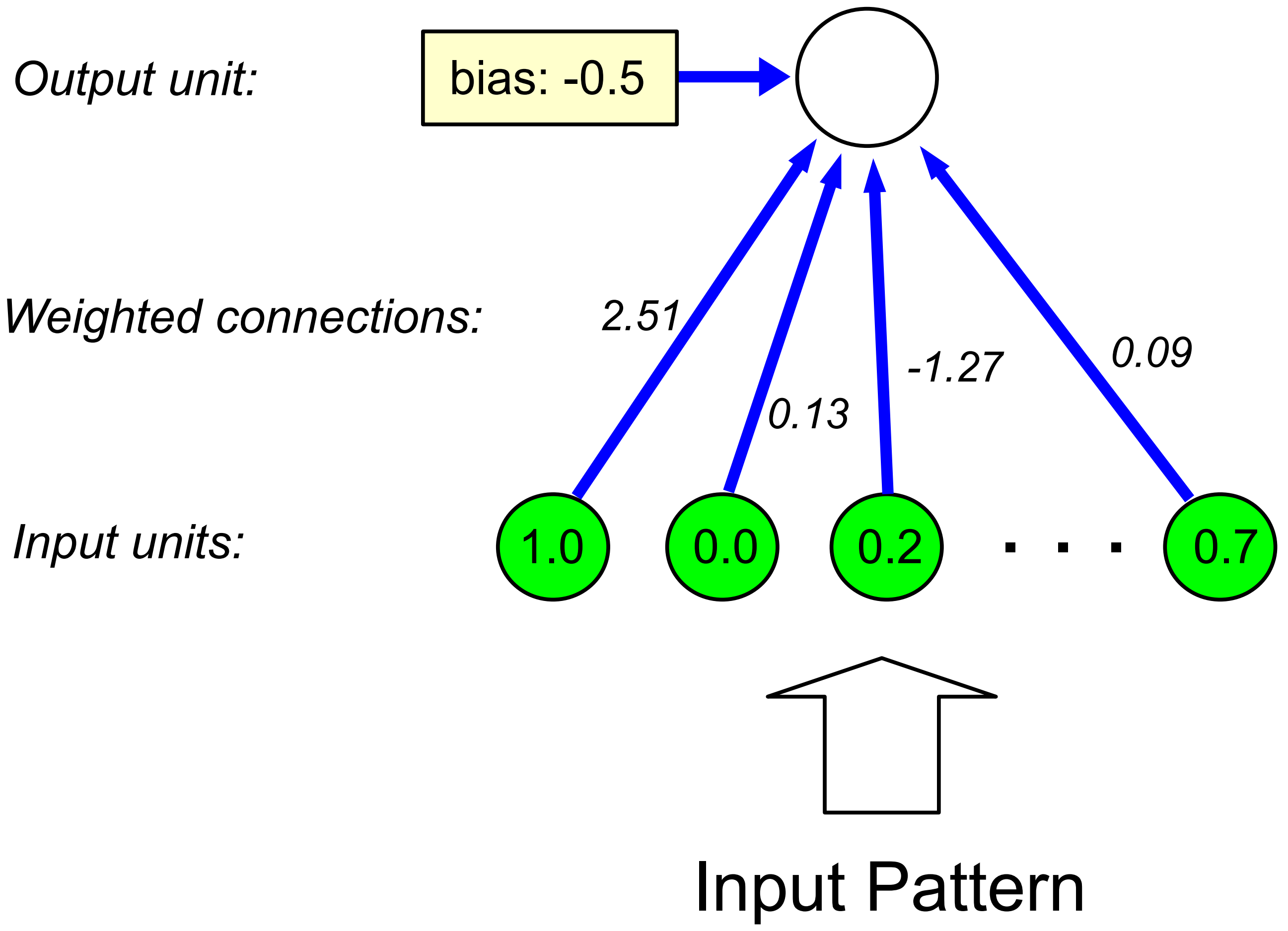
Artificial Neurons: Continuous Version



Artificial Neurons: Continuous Version

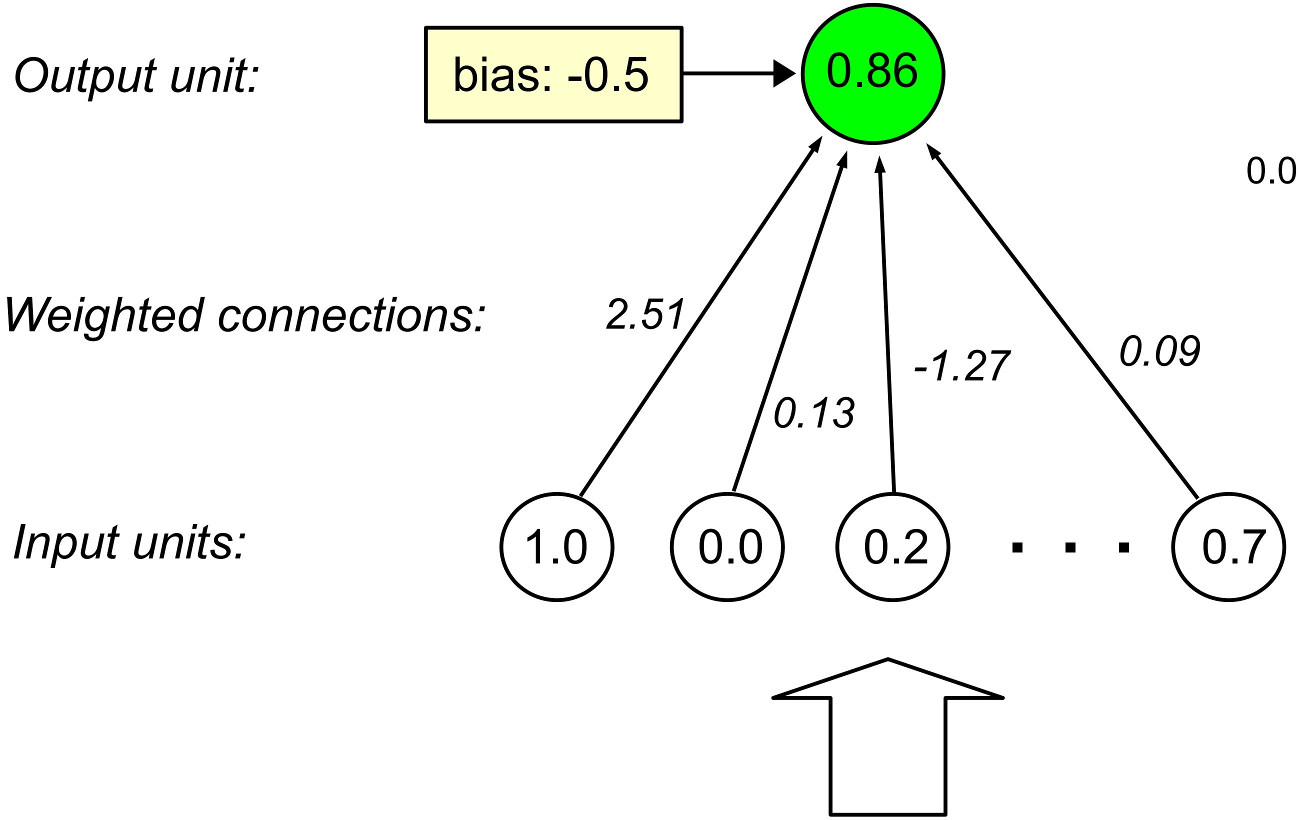
1.0 × 2.51 + **0.0** × 0.13 + **0.2** × -1.27 + . . . + **0.7** × 0.09 **+ -0.5** = 1.82



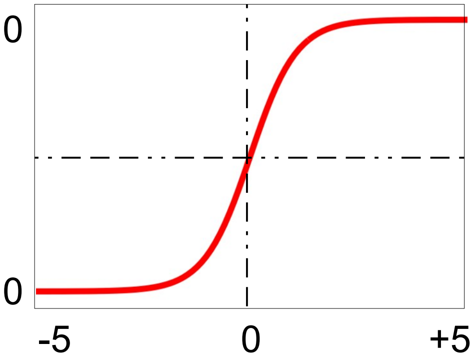
Artificial Neurons: Continuous Version

1.0 × 2.51 + **0.0** × 0.13 + **0.2** × -1.27 + . . . + **0.7** × 0.09 **+ -0.5** = 1.82

$\sigma(1.82) = 0.86$



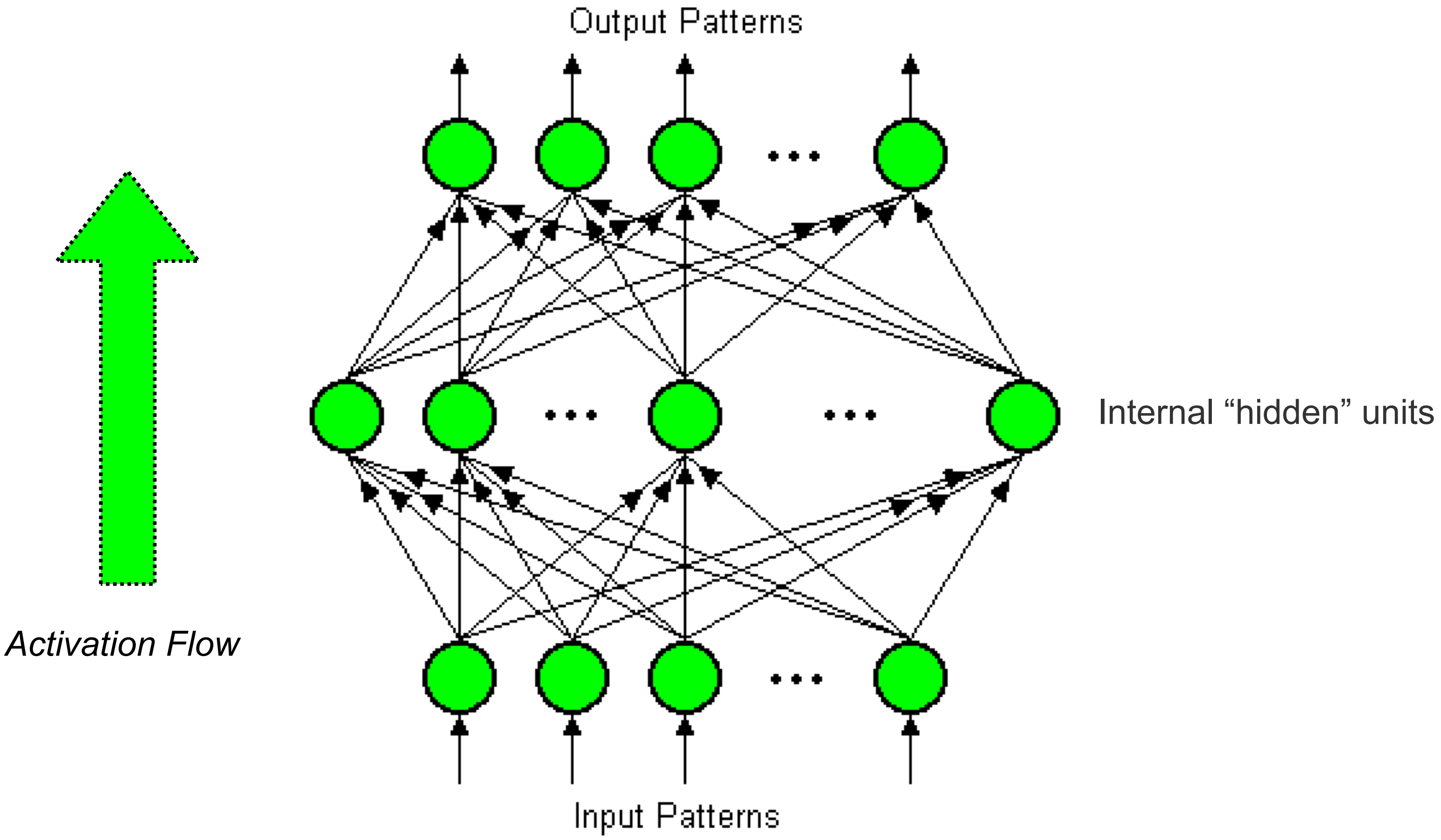
Input Pattern



$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Pattern Associator Networks

- Units are arranged into successive layers
- Feed-forward connections only
- Layer activations represent stimulus/response associations



Output Patterns

Internal “hidden” units

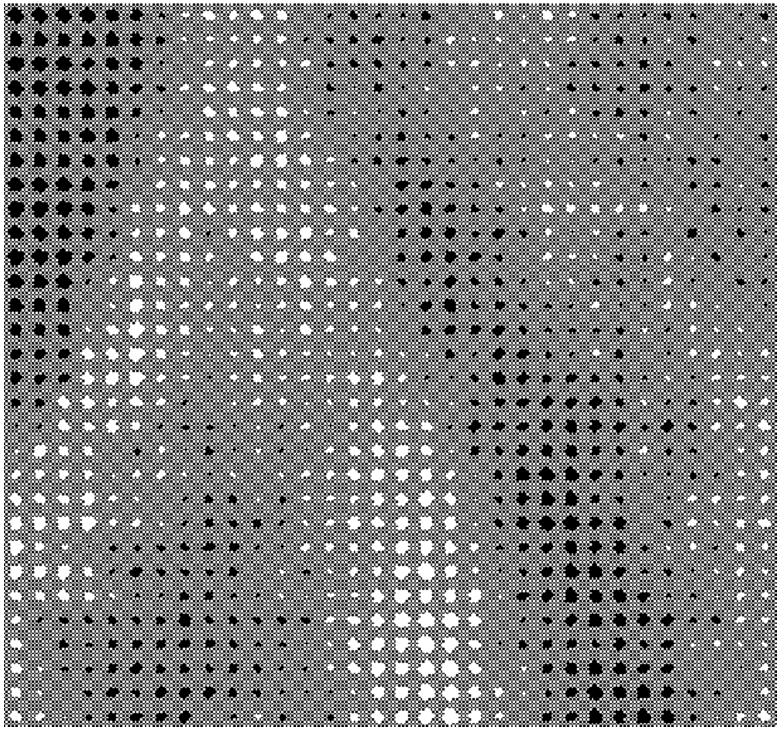
Activation Flow

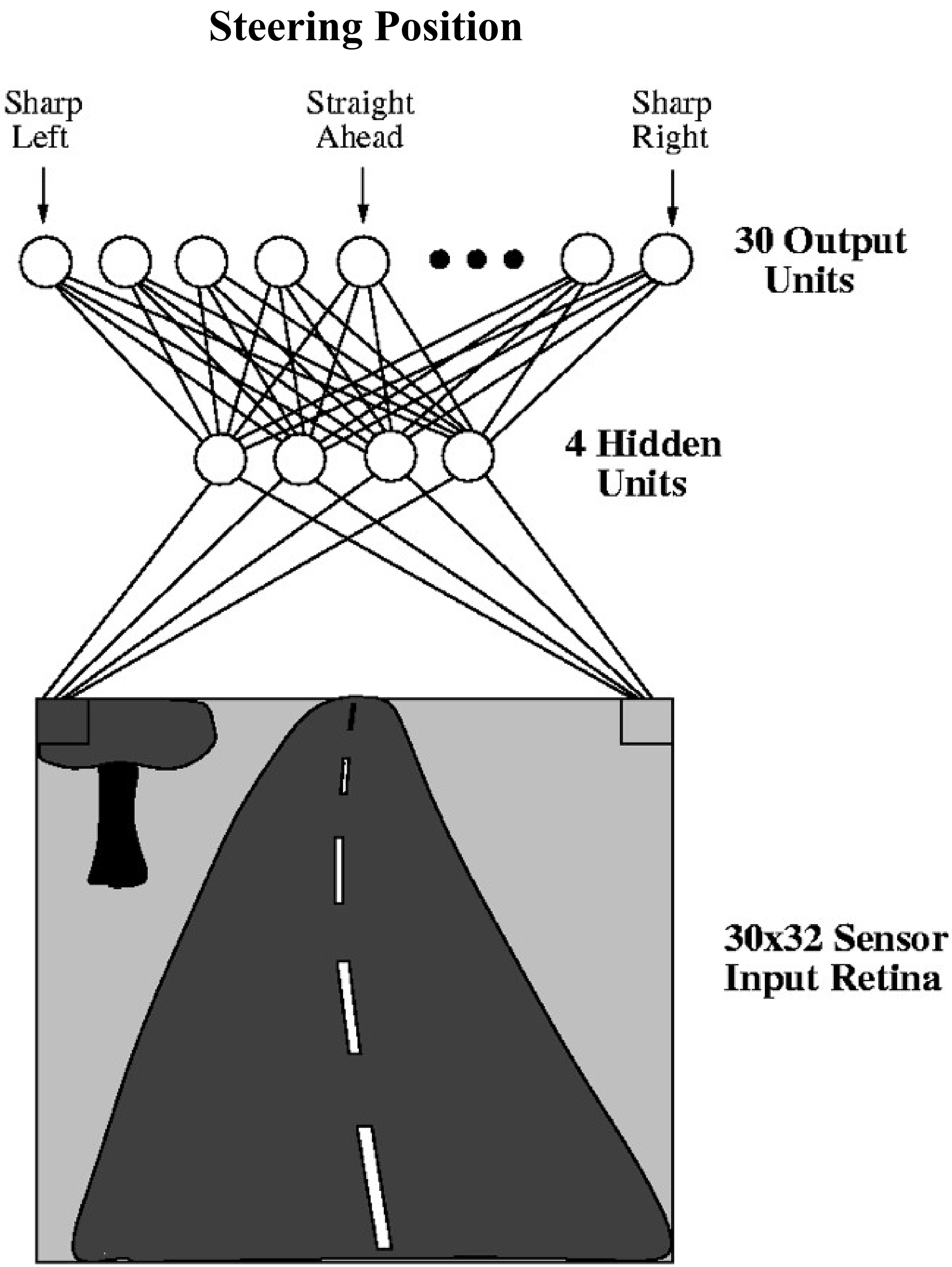
Input Patterns

ALVINN Project

Work by Dean Pomerleau, Carnegie Mellon University (1990s)

- Autonomous vehicle controlled by a neural network
- Input: image of road, Output: steering wheel position
- Neural network trained on human driver data
- In 1995, steered a car semi-autonomously from coast to coast (all but 50 of 2,850 miles)





Steering Position

Sharp Left

Straight Ahead

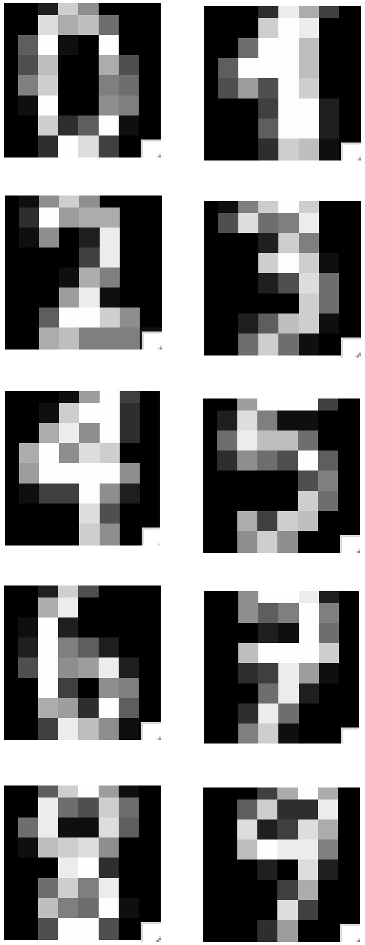
Sharp Right

30 Output Units

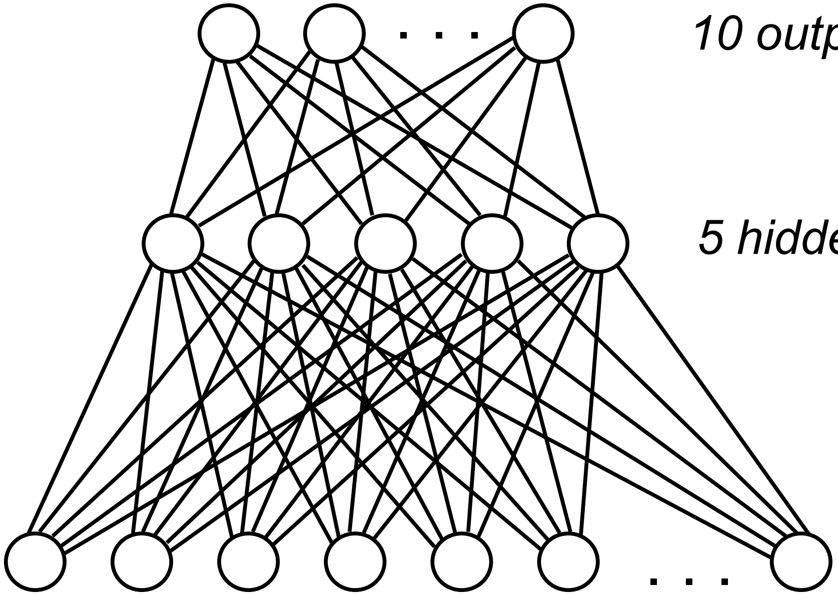
4 Hidden Units

30x32 Sensor Input Retina

Recognizing Handwritten Digits



0 1 9

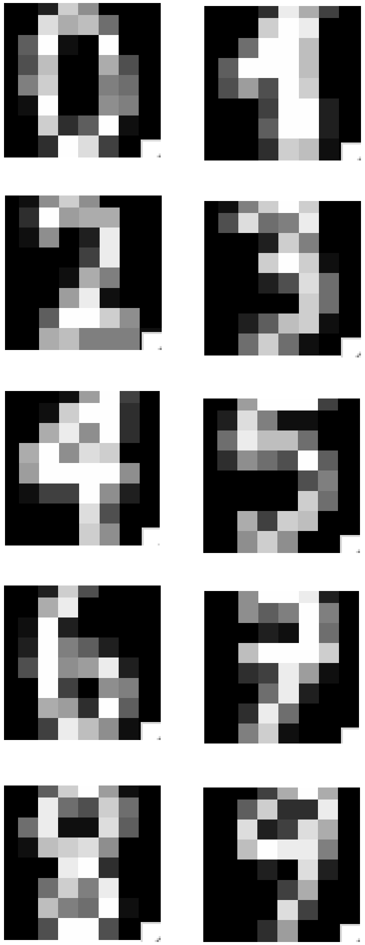


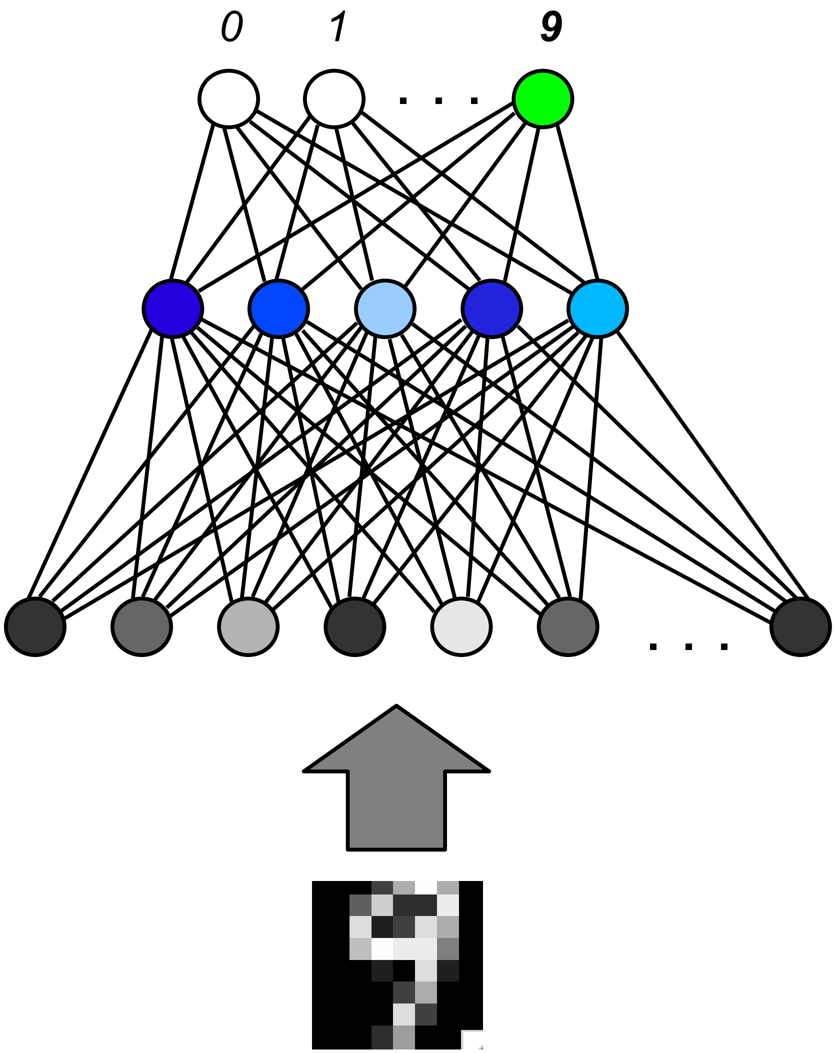
10 output units

5 hidden units

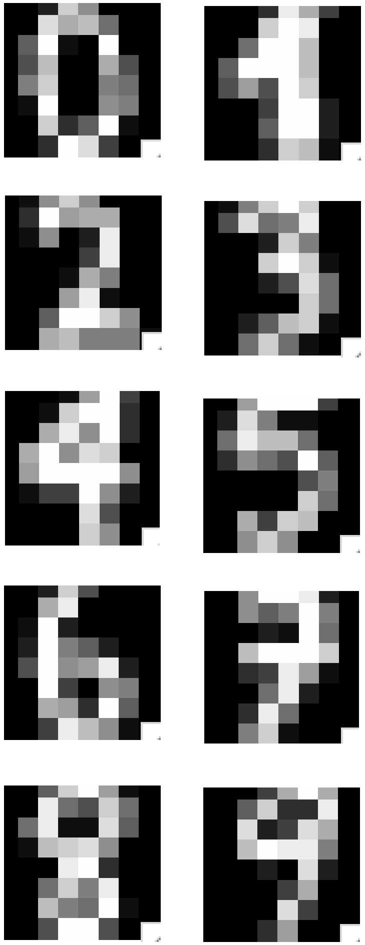
8 x 8 “retina” (64 input units)

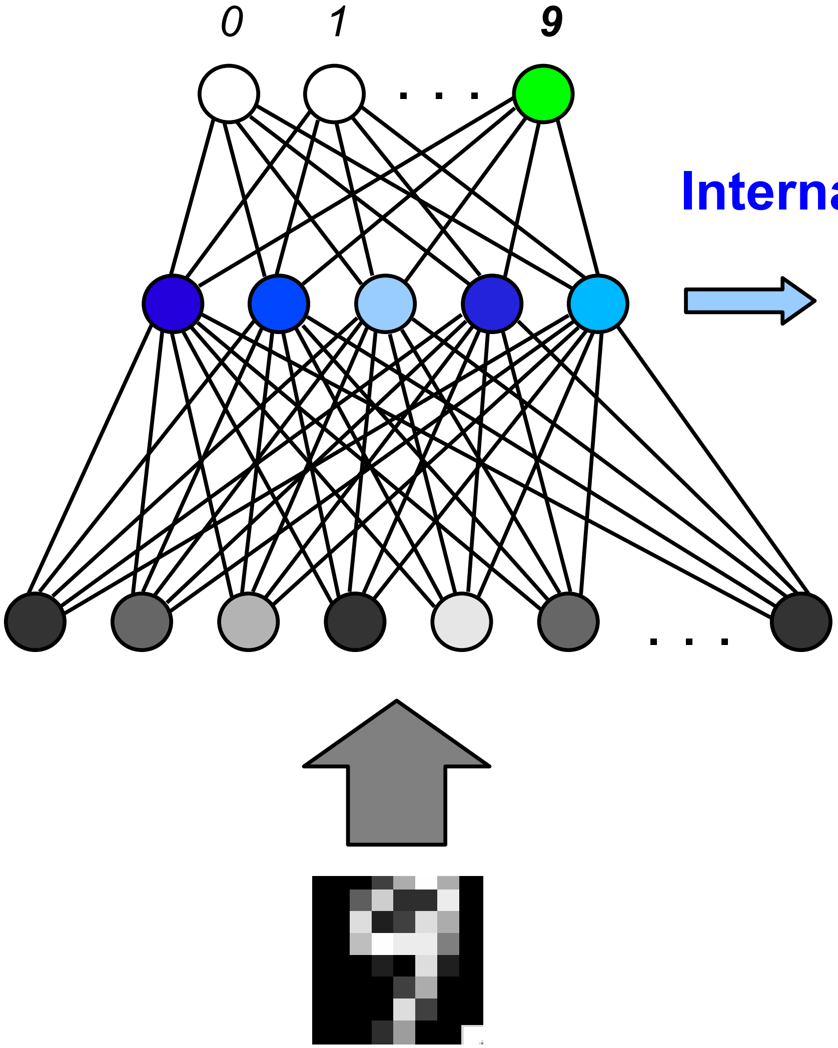
Recognizing Handwritten Digits

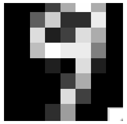




Recognizing Handwritten Digits



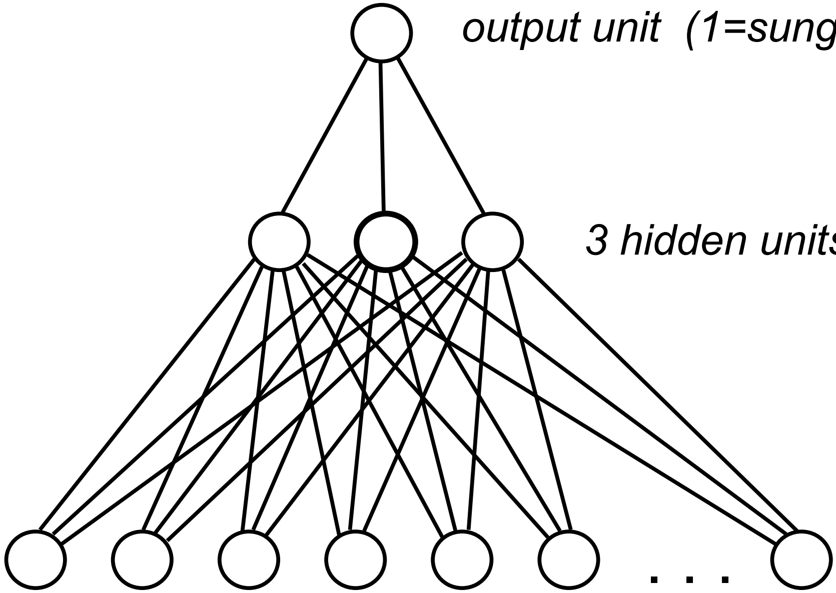


Internal representation of 

[0.54, 0.16, 0.77, 0.29, 0.85]

Handwritten Digits Demo

Recognizing Sunglasses

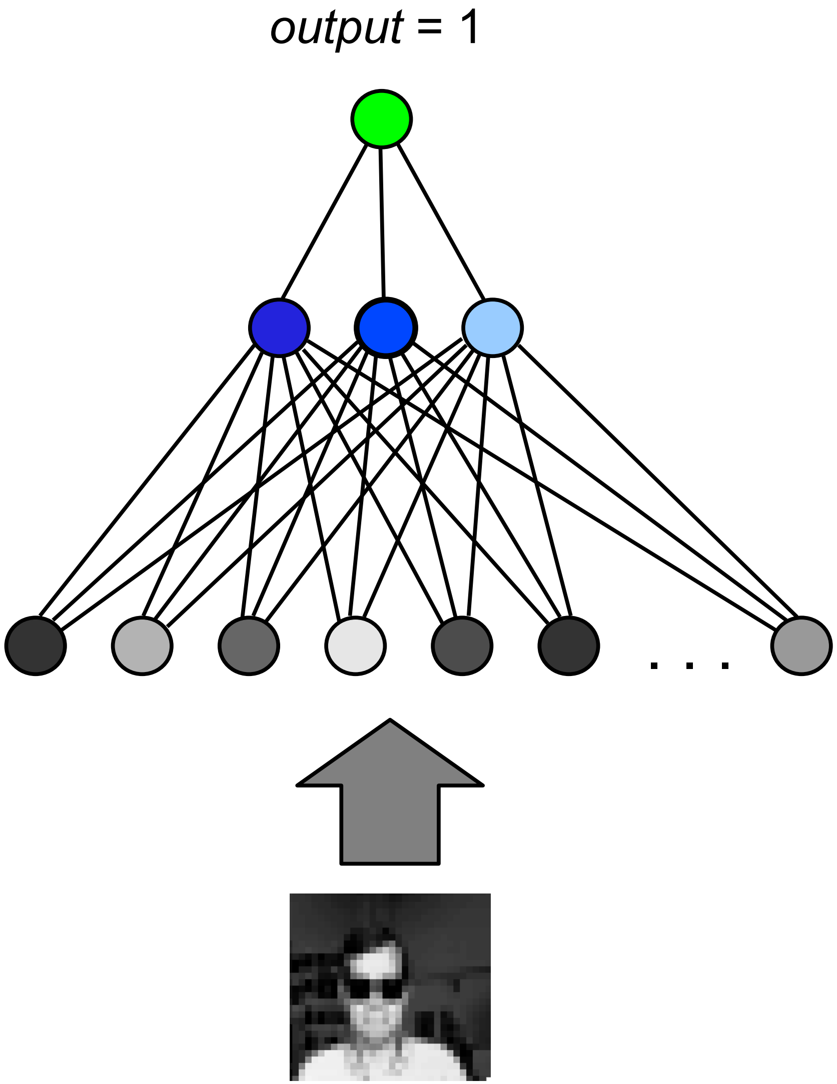


output unit (1=sunglasses, 0=no sunglasses)

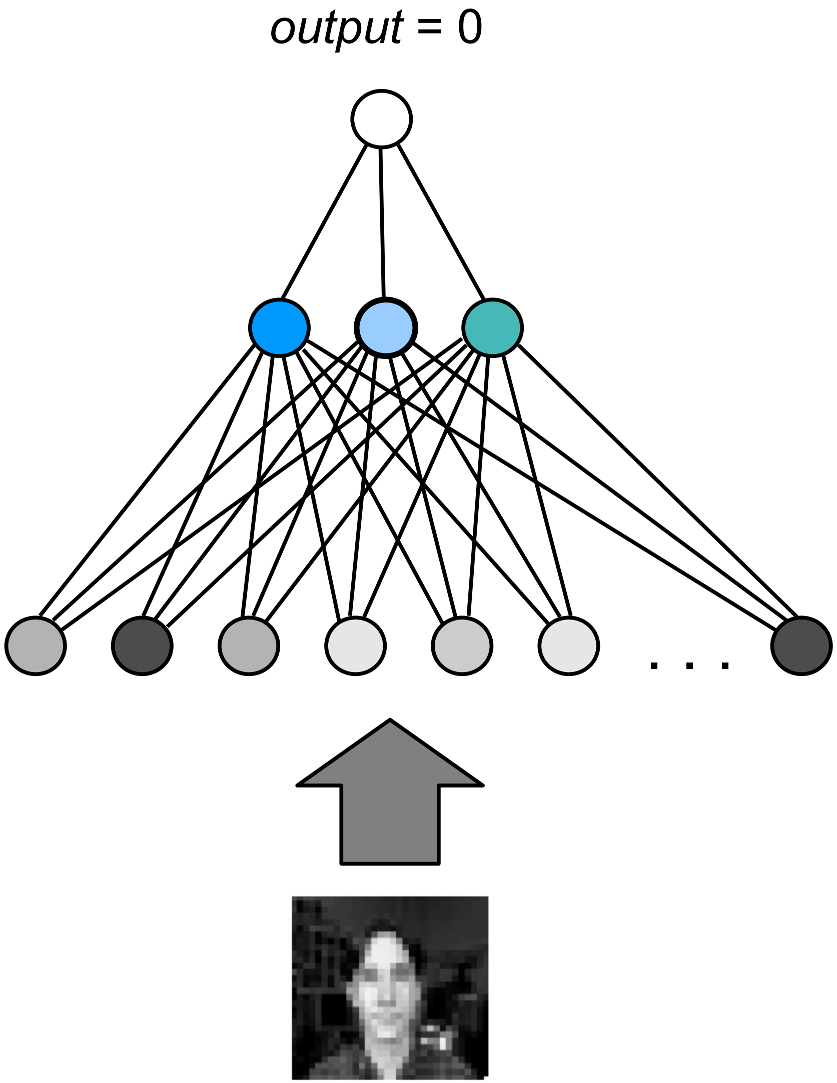
3 hidden units

30 x 32 “retina” (960 input units)

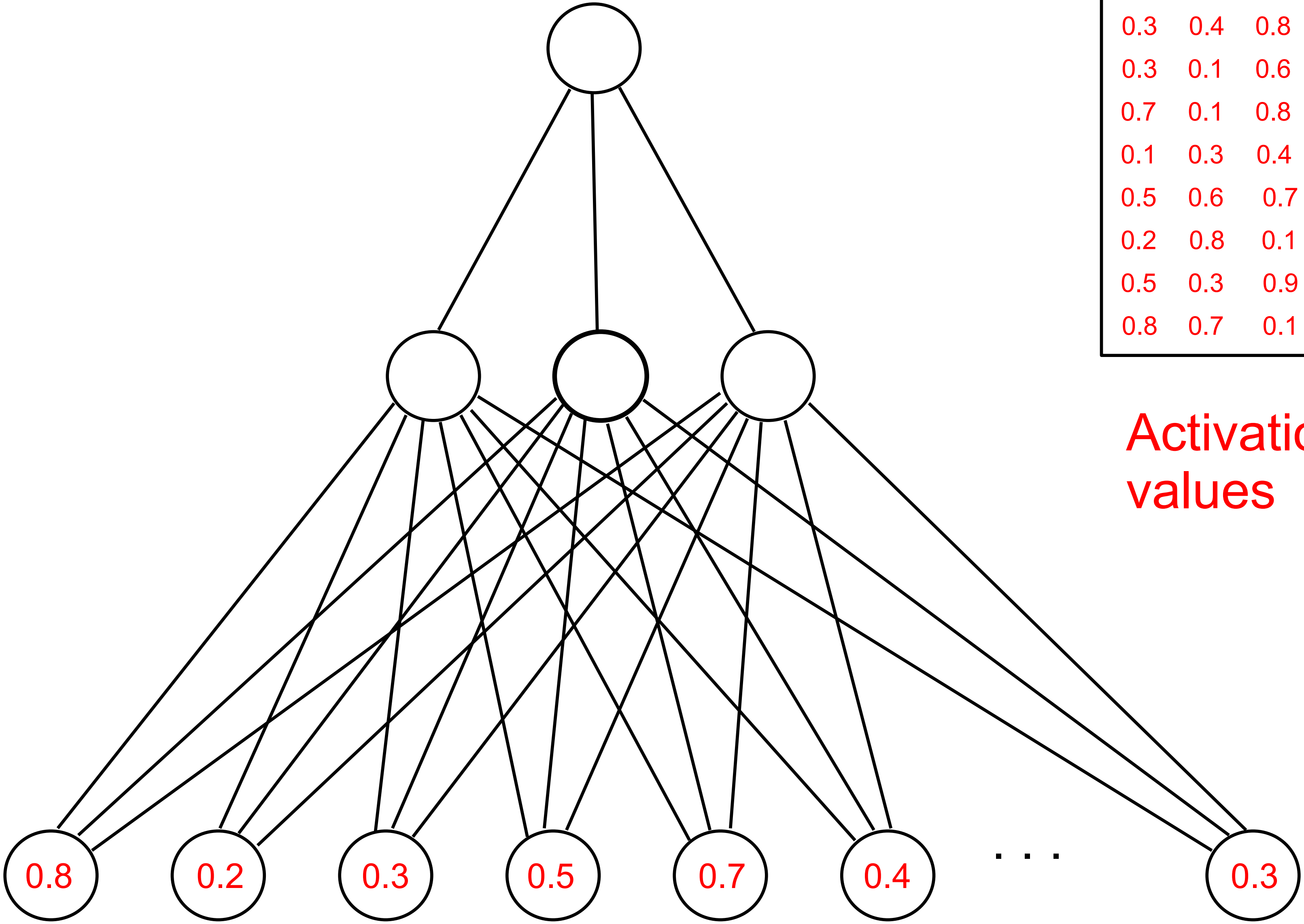
Recognizing Sunglasses



Recognizing Sunglasses



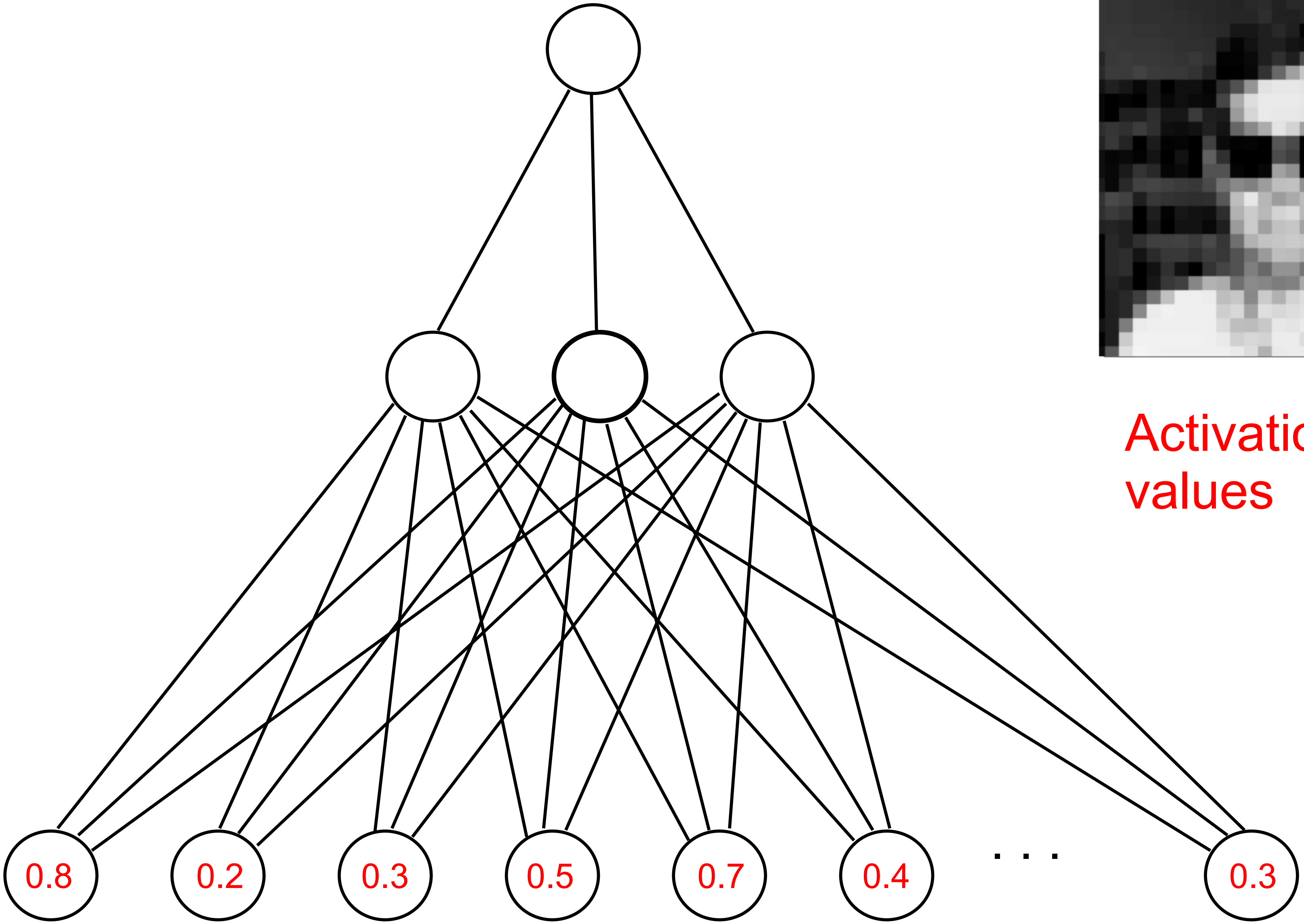
Recognizing Sunglasses

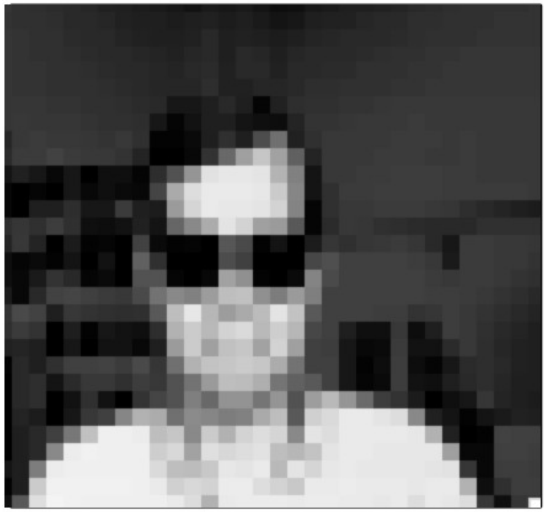


0.8	0.2	0.3	0.5	0.7	0.4
0.3	0.4	0.8	0.3	0.5	0.0
0.3	0.1	0.6	0.5	0.0	0.8
0.7	0.1	0.8	0.2	0.4	0.1
0.1	0.3	0.4	0.0	0.9	0.6
0.5	0.6	0.7	0.5	0.8	0.3
0.2	0.8	0.1	0.8	0.7	0.9
0.5	0.3	0.9	0.5	0.2	0.4
0.8	0.7	0.1	0.9	0.1	0.2

Activation values

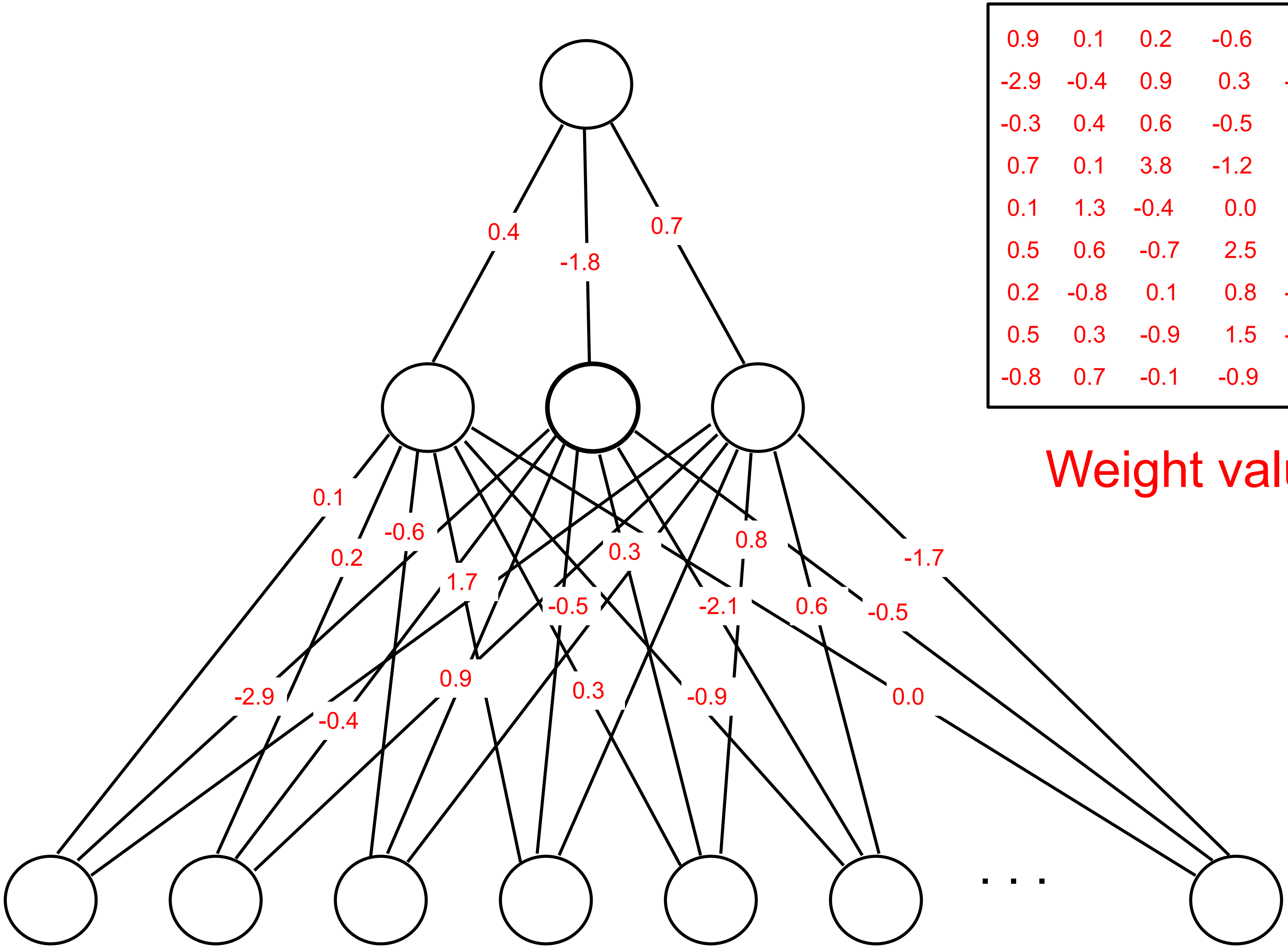
Recognizing Sunglasses





Activation values

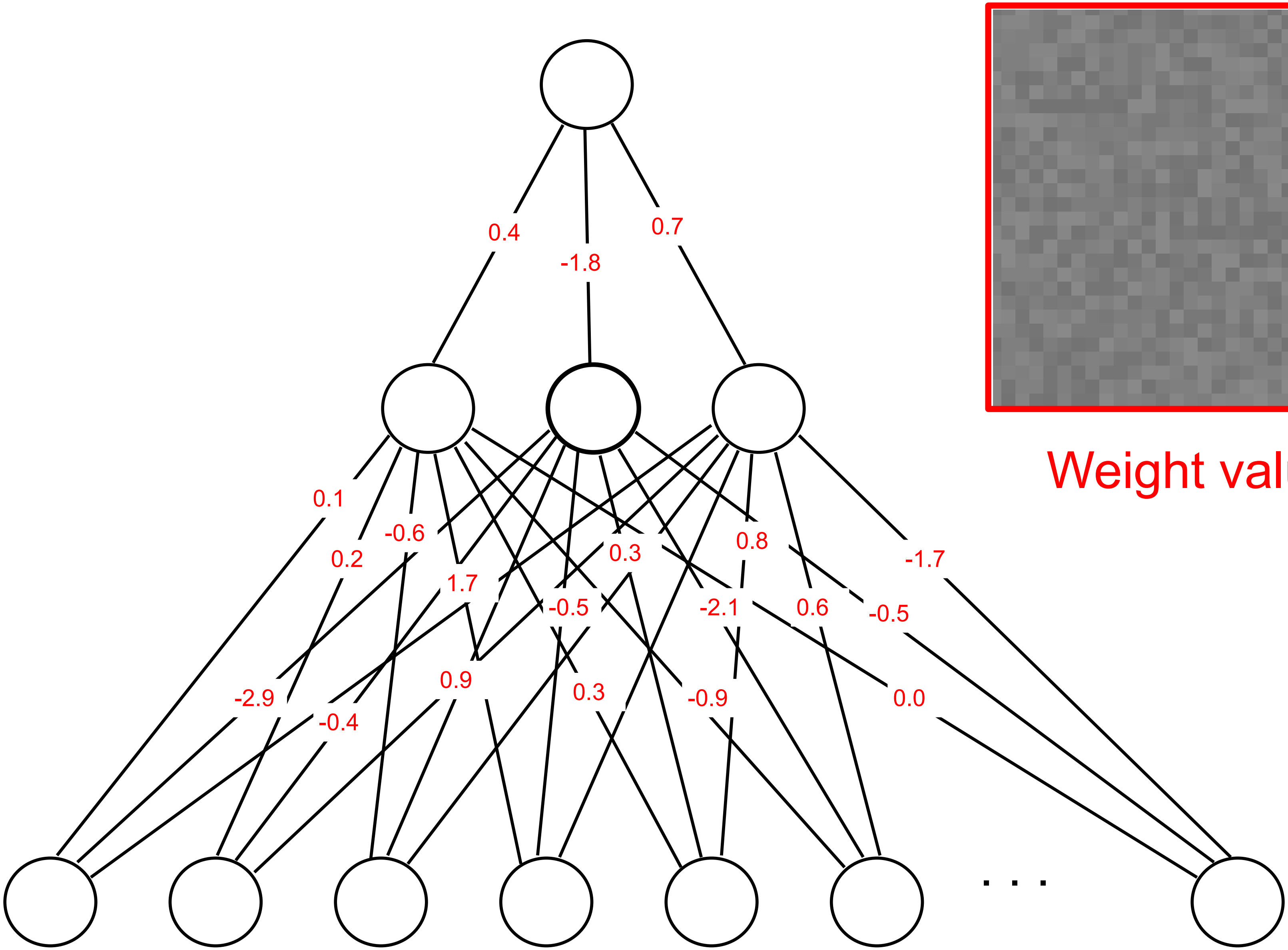
Recognizing Sunglasses

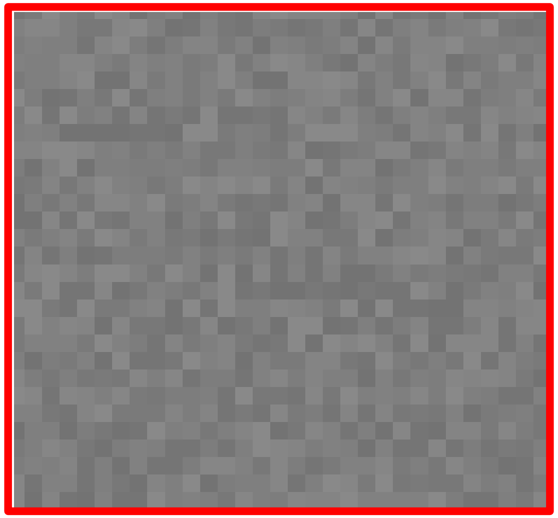


0.9	0.1	0.2	-0.6	1.7	-0.5
-2.9	-0.4	0.9	0.3	-0.5	0.0
-0.3	0.4	0.6	-0.5	2.0	0.8
0.7	0.1	3.8	-1.2	1.4	0.1
0.1	1.3	-0.4	0.0	0.9	0.6
0.5	0.6	-0.7	2.5	1.8	0.3
0.2	-0.8	0.1	0.8	-0.7	1.9
0.5	0.3	-0.9	1.5	-0.2	0.4
-0.8	0.7	-0.1	-0.9	1.1	3.2

Weight values

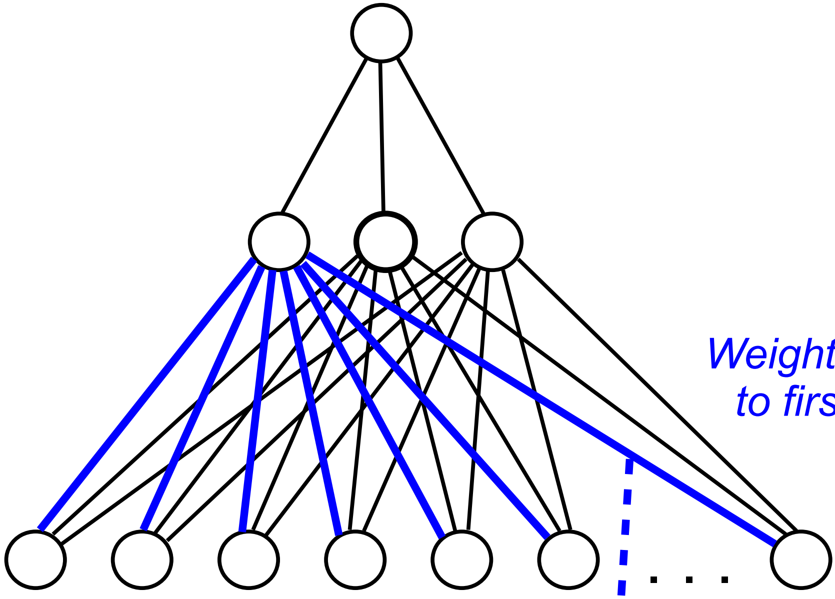
Recognizing Sunglasses



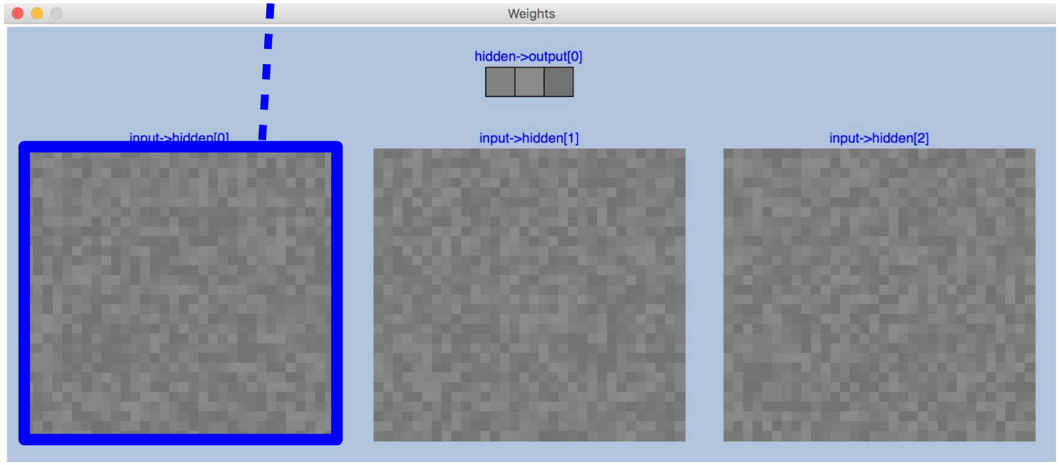


Weight values

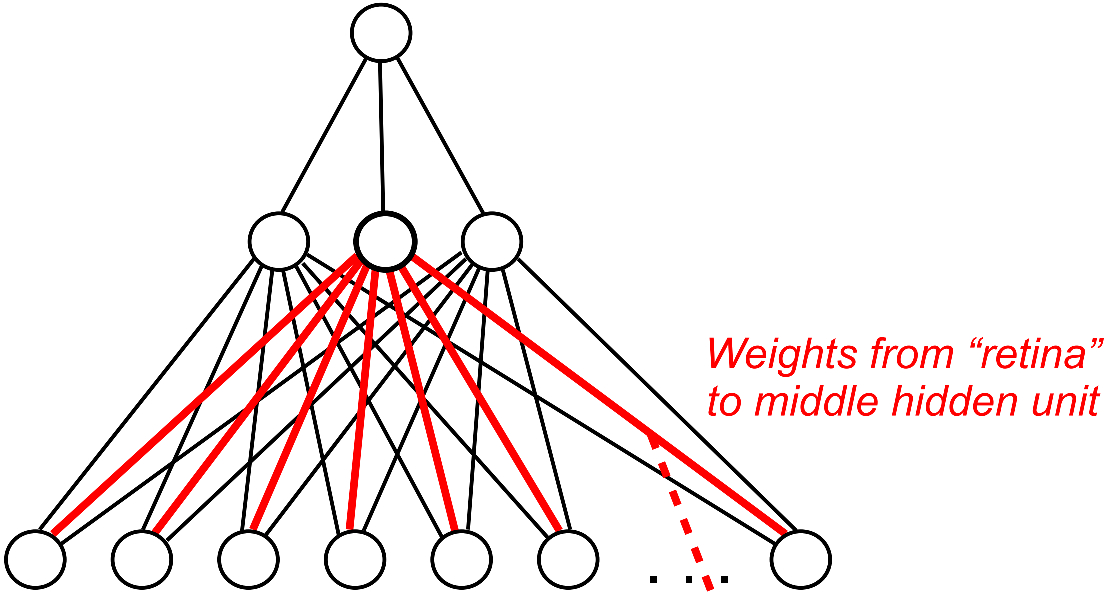
Recognizing Sunglasses



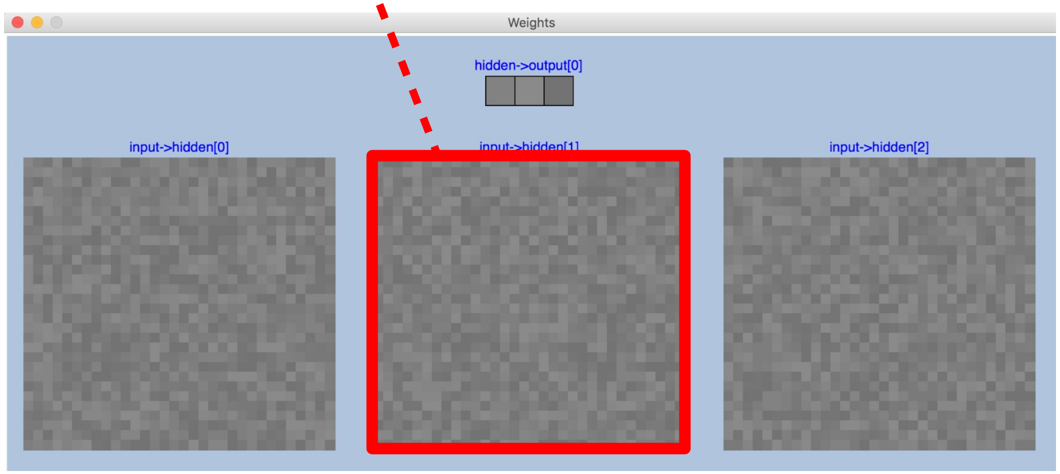
*Weights from “retina”
to first hidden unit*



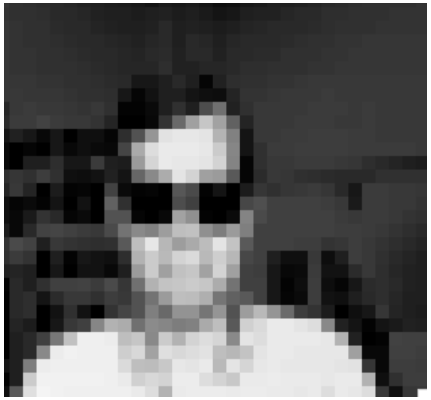
Recognizing Sunglasses

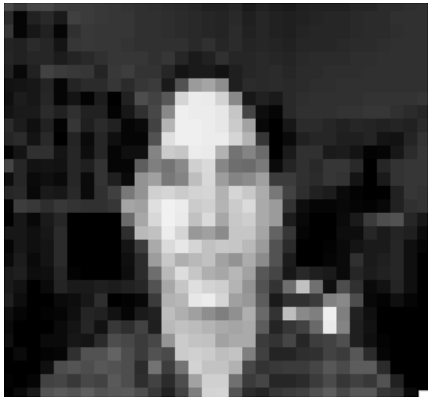


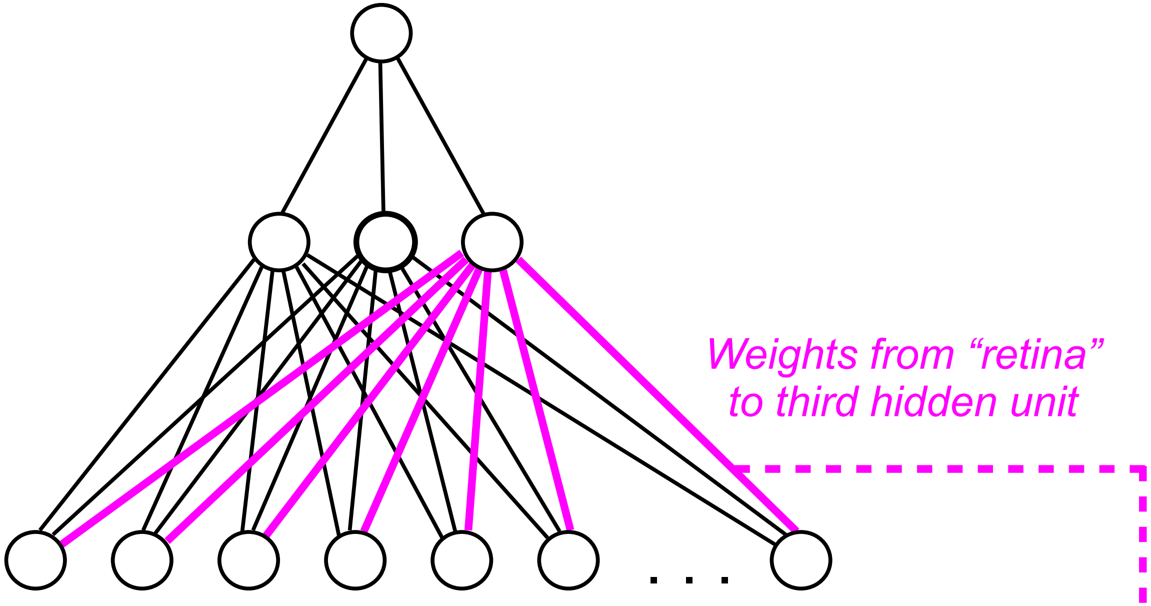
Weights from “retina” to middle hidden unit



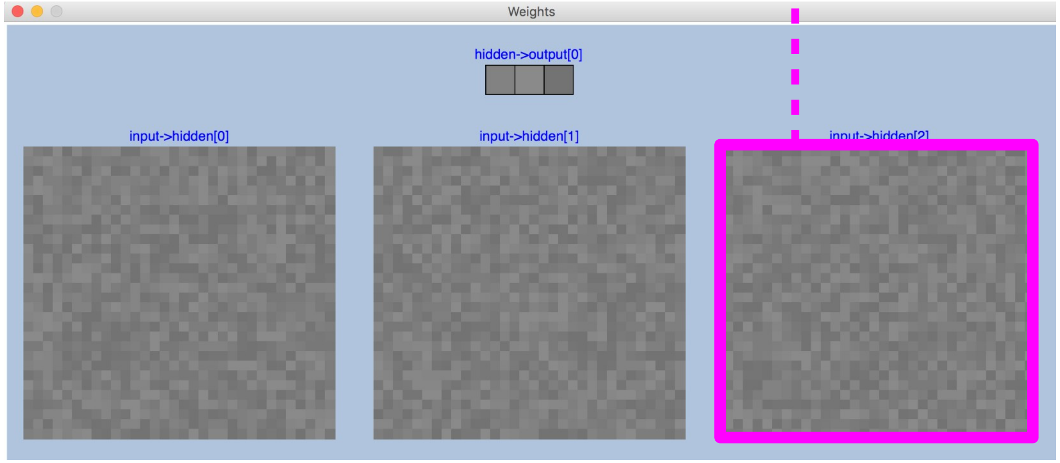
Recognizing Sunglasses



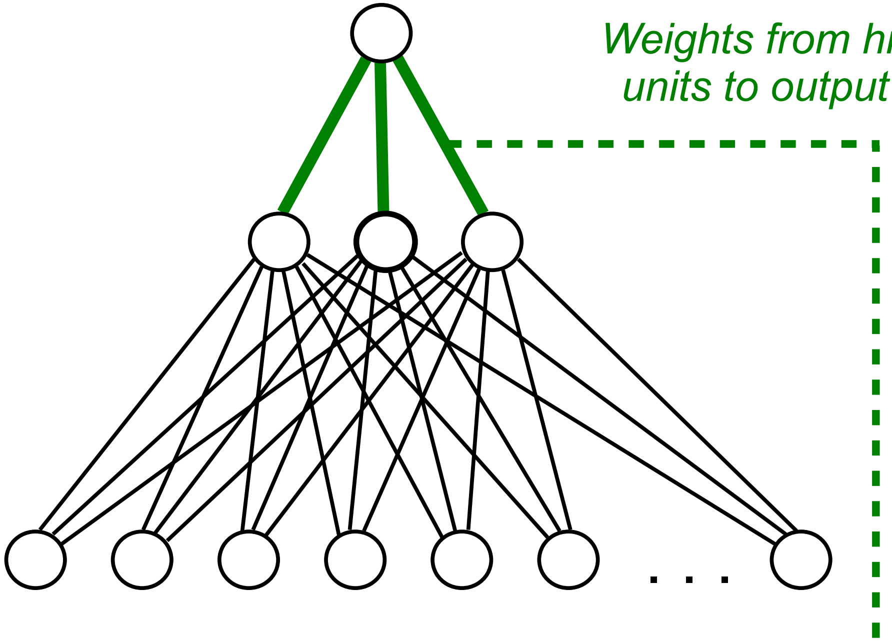




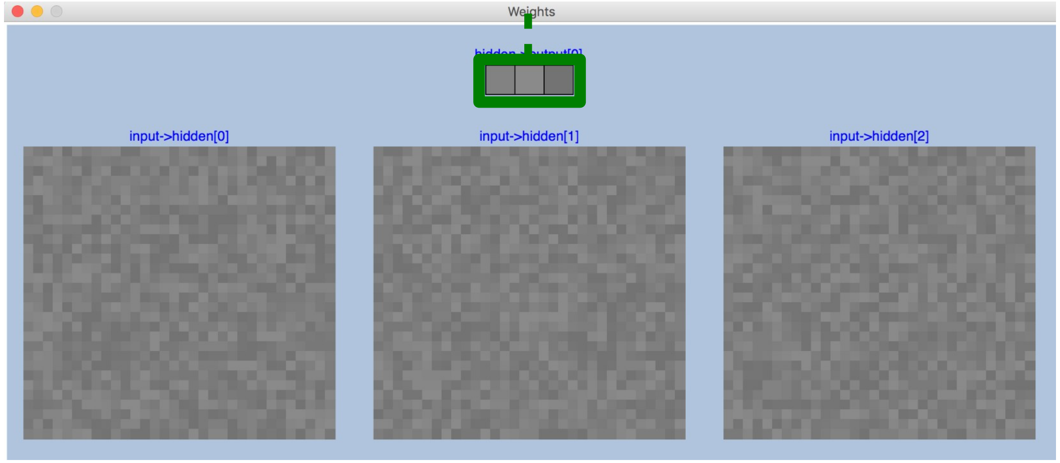
*Weights from “retina”
to third hidden unit*



Recognizing Sunglasses

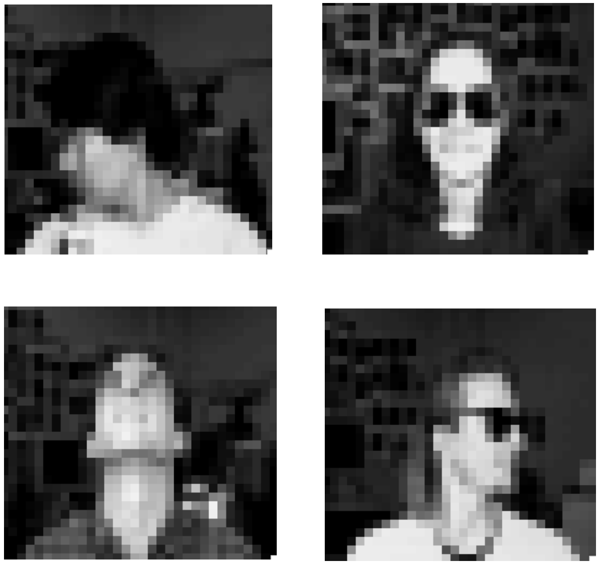


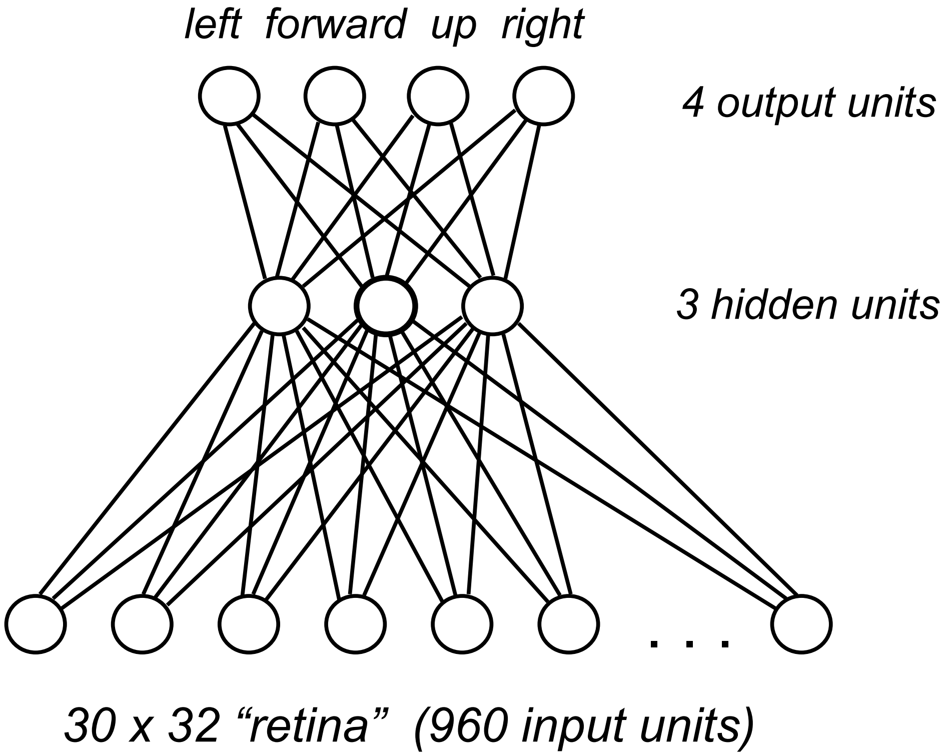
Weights from hidden units to output unit



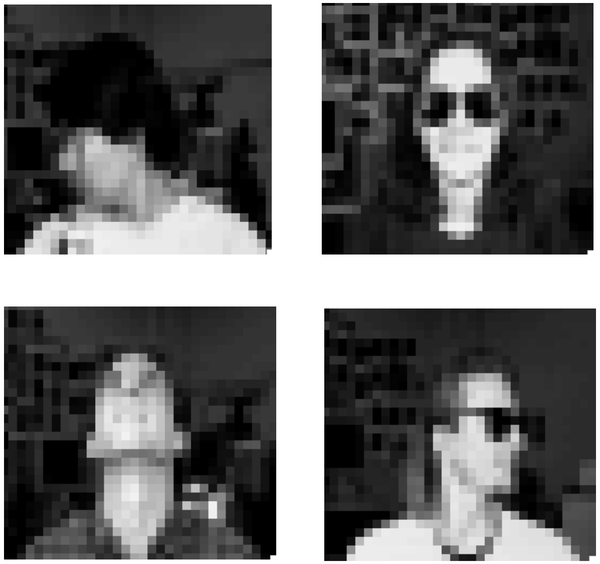
Sunglasses Recognizer Demo

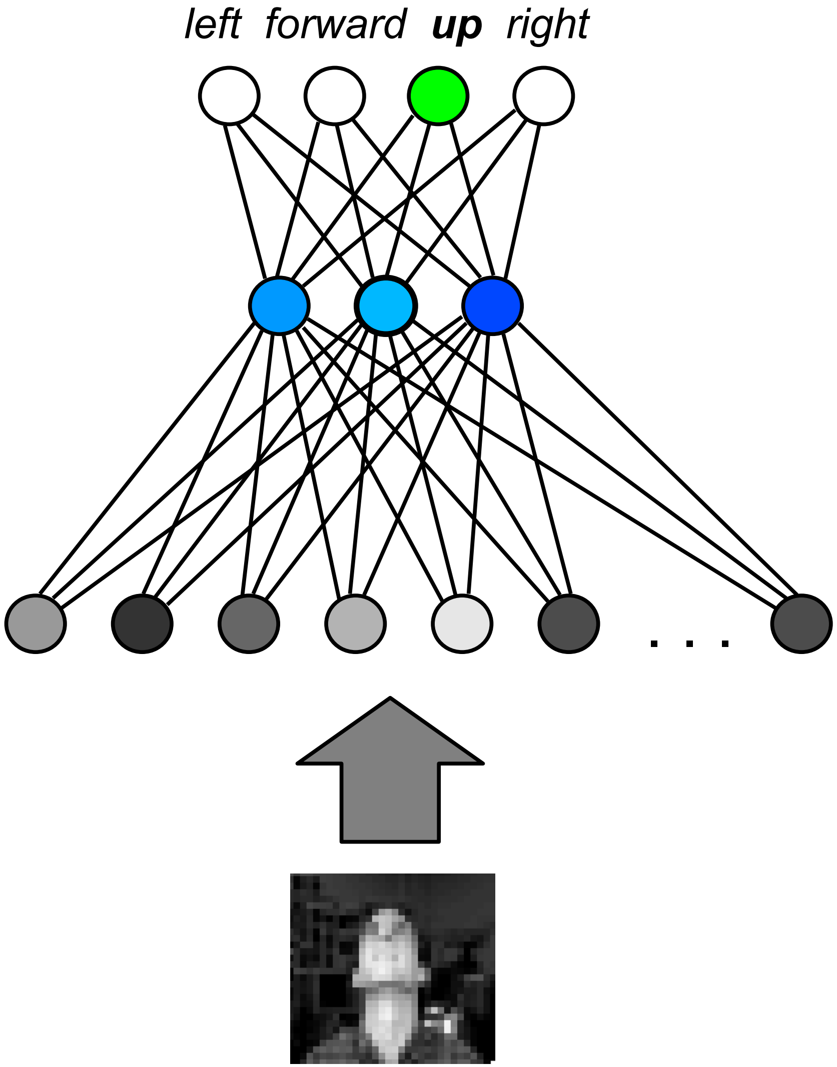
Recognizing Poses



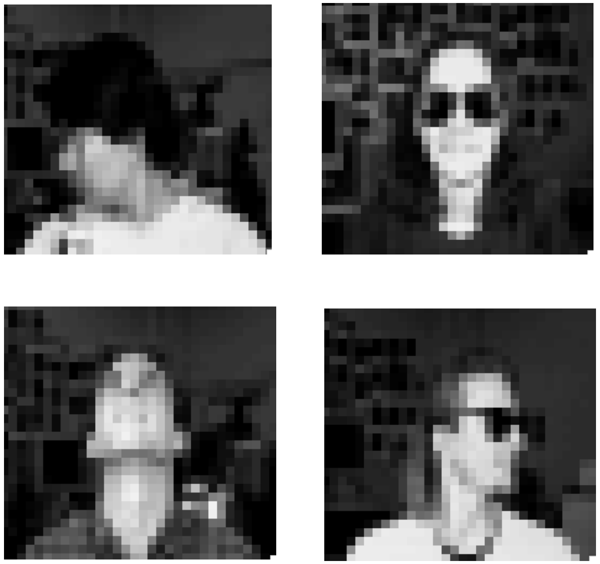


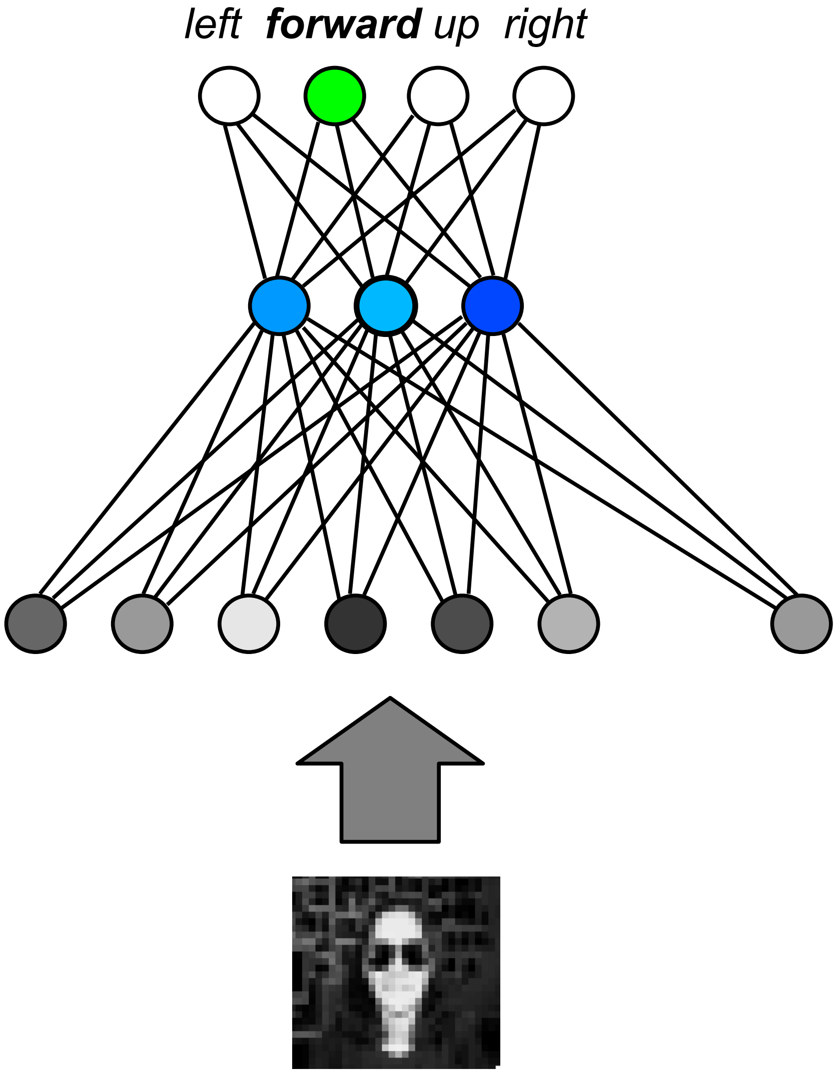
Recognizing Poses





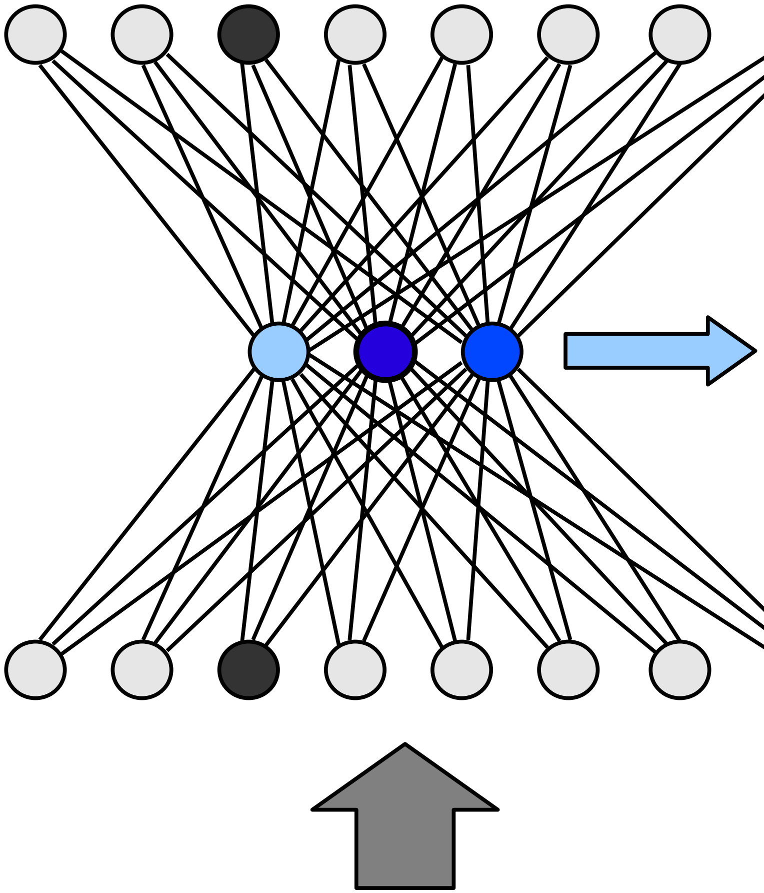
Recognizing Poses





Pose Recognizer Demo

Auto-Associator Networks



Target = 0 0 1 0 0 0 0 0

8 output units

3 hidden units

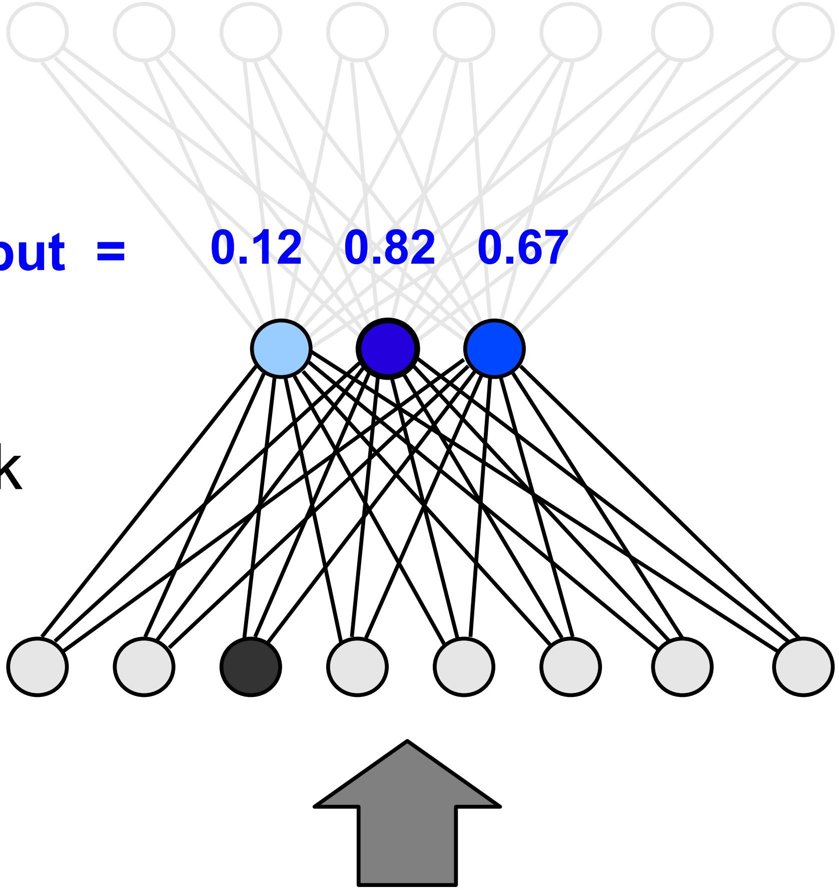
8 input units

Internal representation

0.12 0.82 0.67

Input = 0 0 1 0 0 0 0 0

Auto-Associator Networks

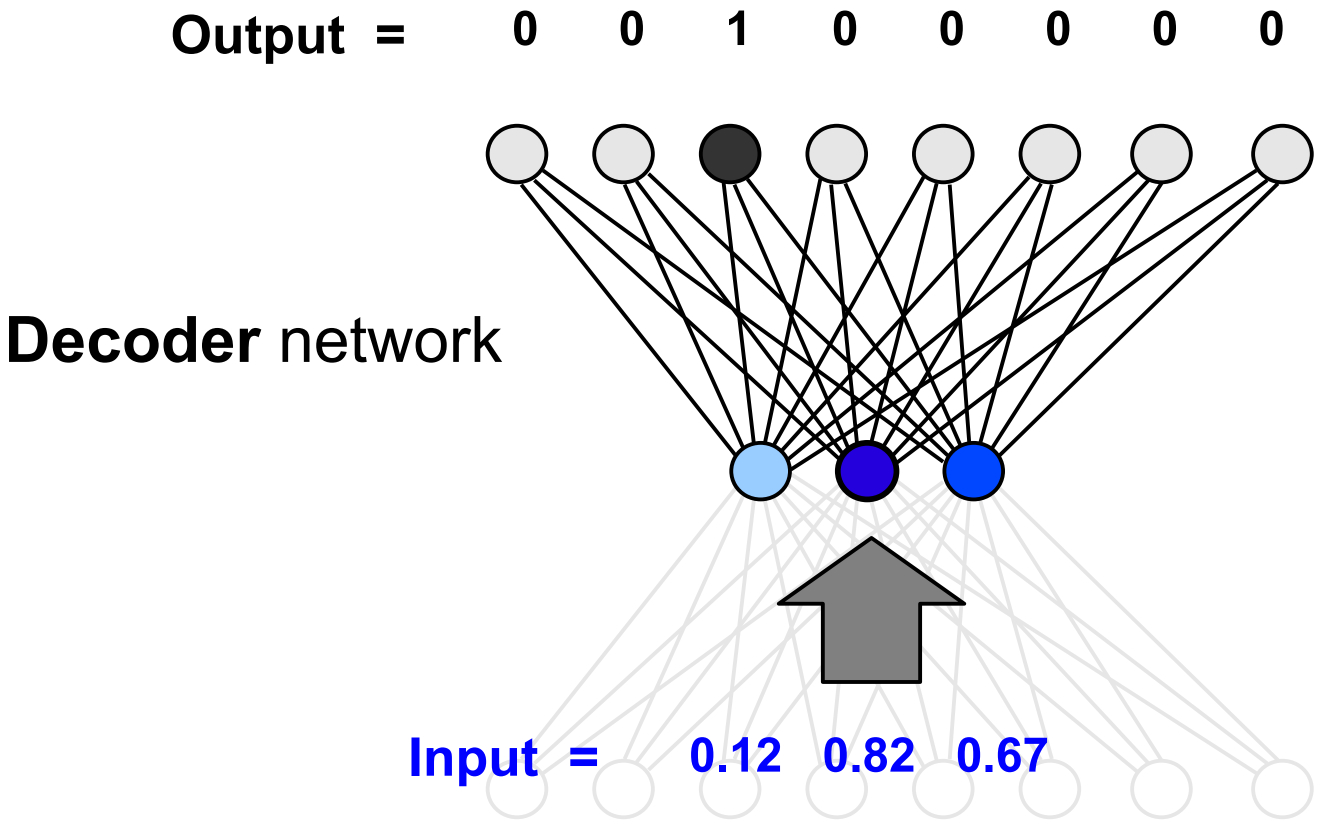


Output = 0.12 0.82 0.67

Encoder network

Input = 0 0 1 0 0 0 0 0

Auto-Associator Networks



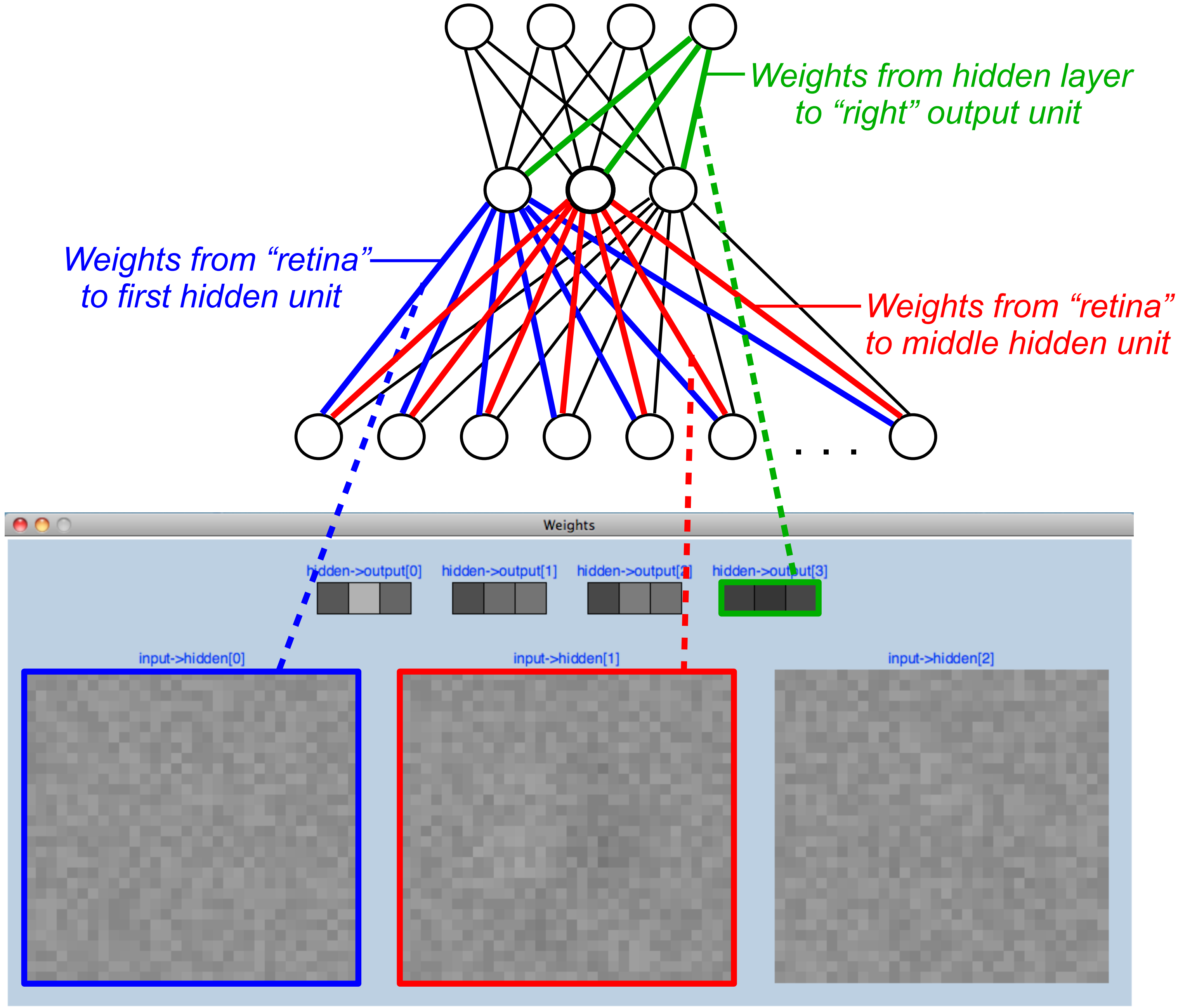
Output = 0 0 1 0 0 0 0 0

Decoder network

Input = 0.12 0.82 0.67

Auto-Association Demos

The Knowledge is in the Connection Weights



Weights from hidden layer to “right” output unit

Weights from “retina” to first hidden unit

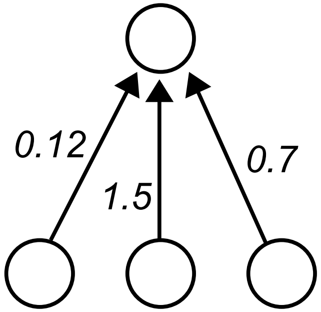
Weights from “retina” to middle hidden unit

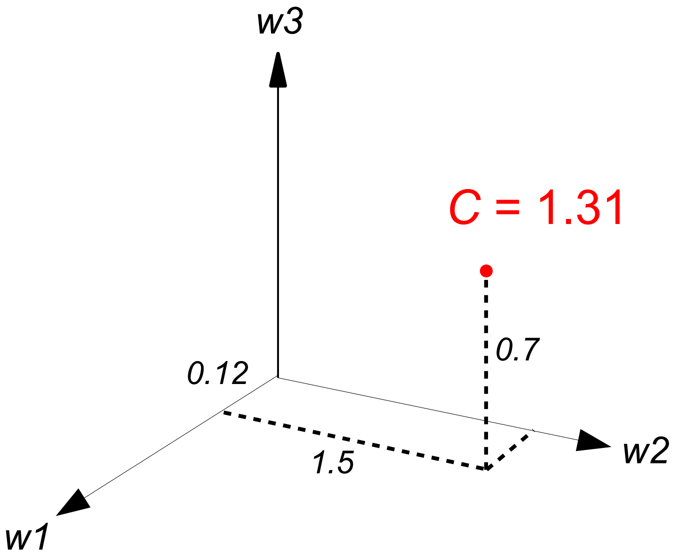
Neural Network Learning

- Connection weights determine network behavior
- Behavior could be “good” or “bad”
- **Cost function** quantifies this measure (also called **error**)

$$C = \tfrac{1}{2}(target_1 - output_1)^2 + \tfrac{1}{2}(target_2 - output_2)^2 + \ldots$$

Input	Target	Output
0 0 0	0	0.80
0 0 1	0	0.77
0 1 0	0	0.82
0 1 1	1	0.60
1 0 0	0	0.53
1 0 1	1	0.59
1 1 0	1	0.73
1 1 1	1	0.81

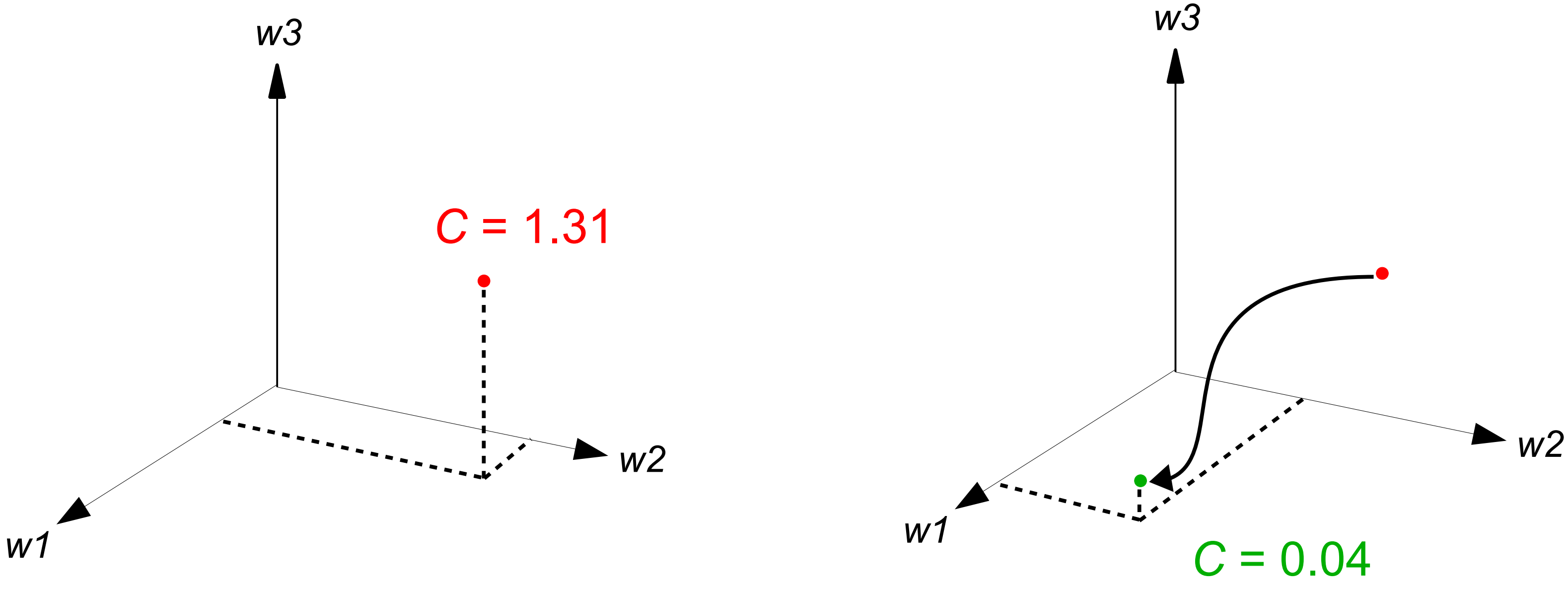




“Weight space”

Neural Network Learning

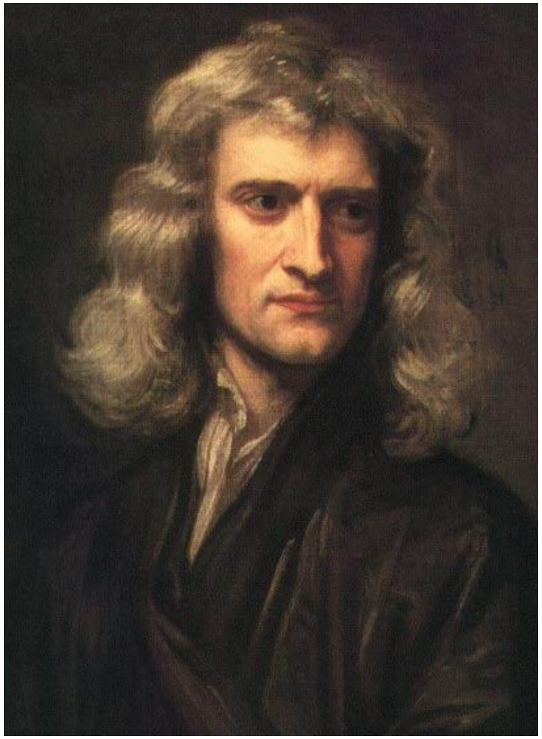
- How to change the weights so that *C* goes down?
- **Backpropagation learning algorithm** modifies the weights
- On each time step, the overall cost/error of the weights moves “downhill” in the direction of the **gradient**



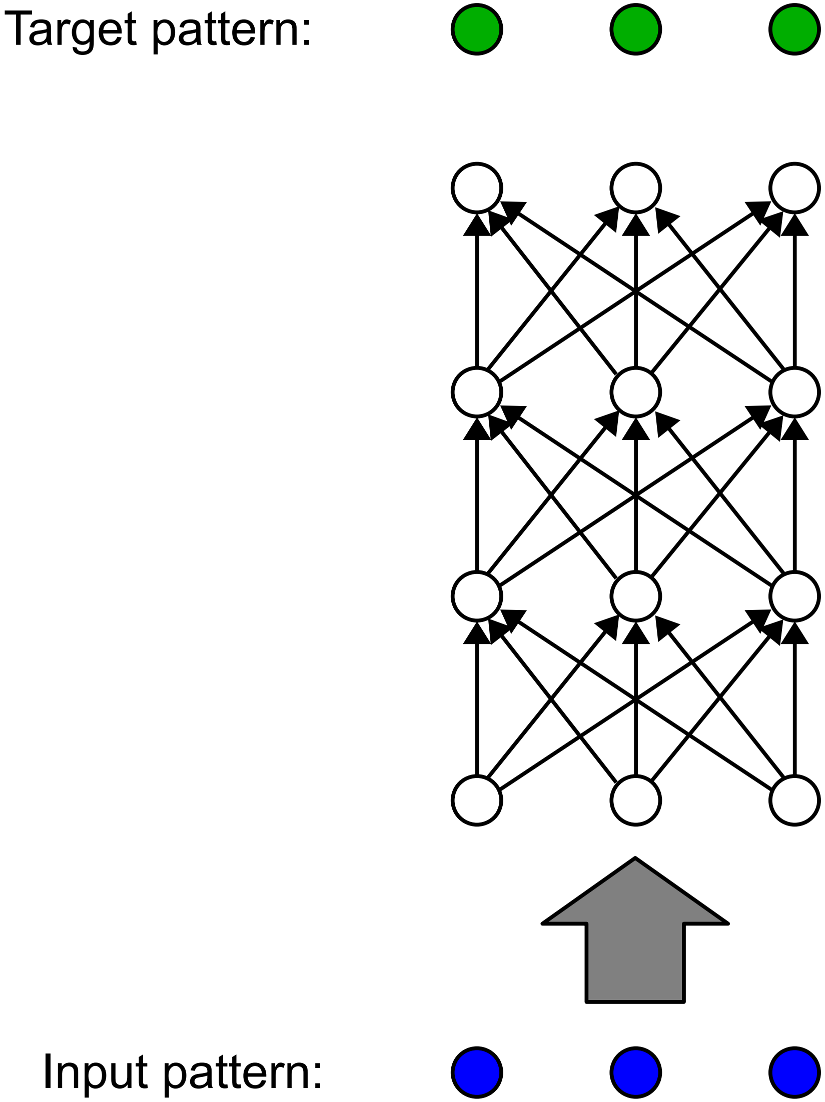
Neural Network Learning

- Thank you, Newton and Leibniz!

$$\Delta w_i = -\eta \;\partial C / \partial w_i$$



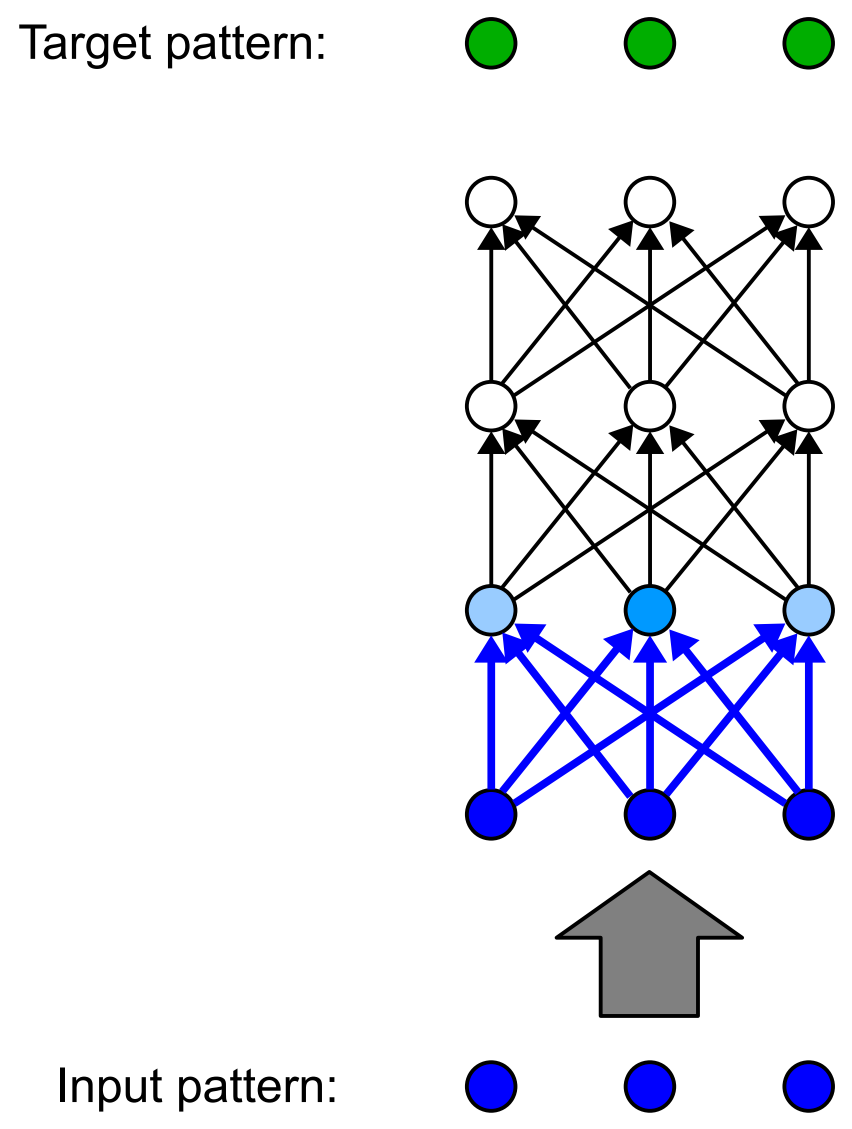
Backpropagation Algorithm



Target pattern:

Input pattern:

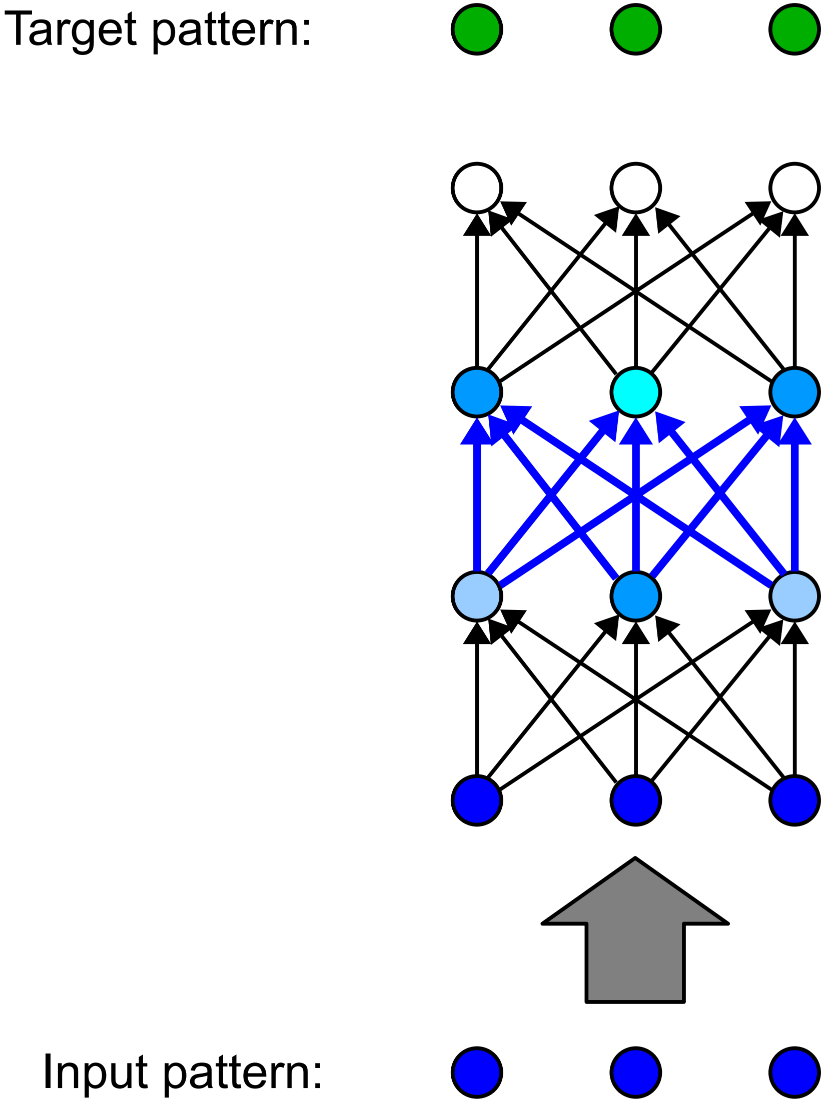
Backpropagation Algorithm



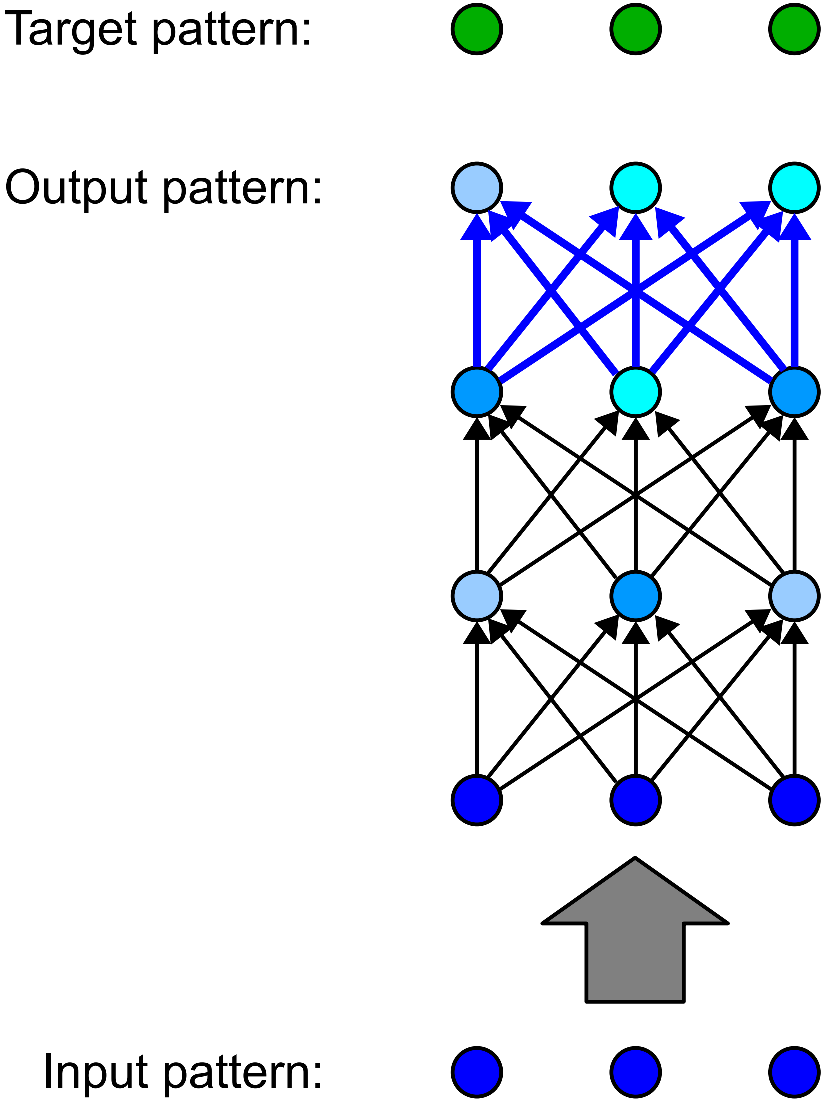
Target pattern:

Input pattern:

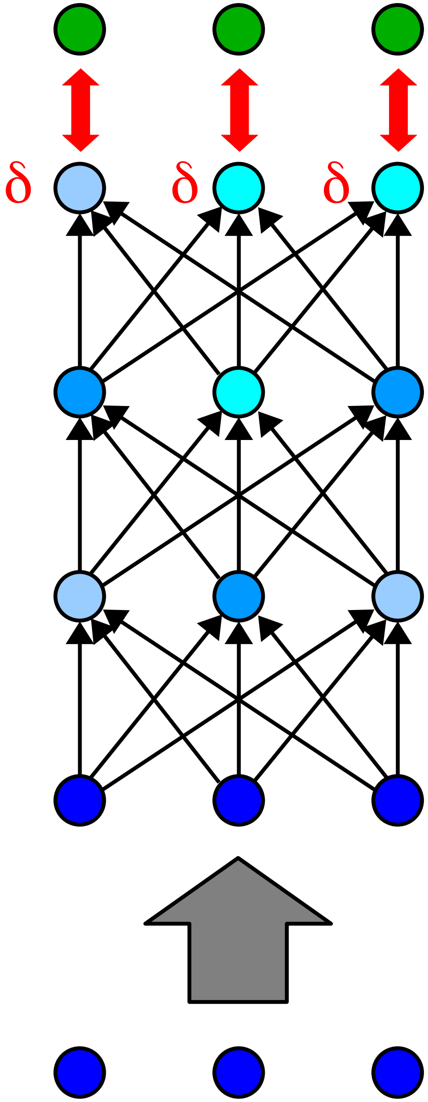
Backpropagation Algorithm



Backpropagation Algorithm



Backpropagation Algorithm



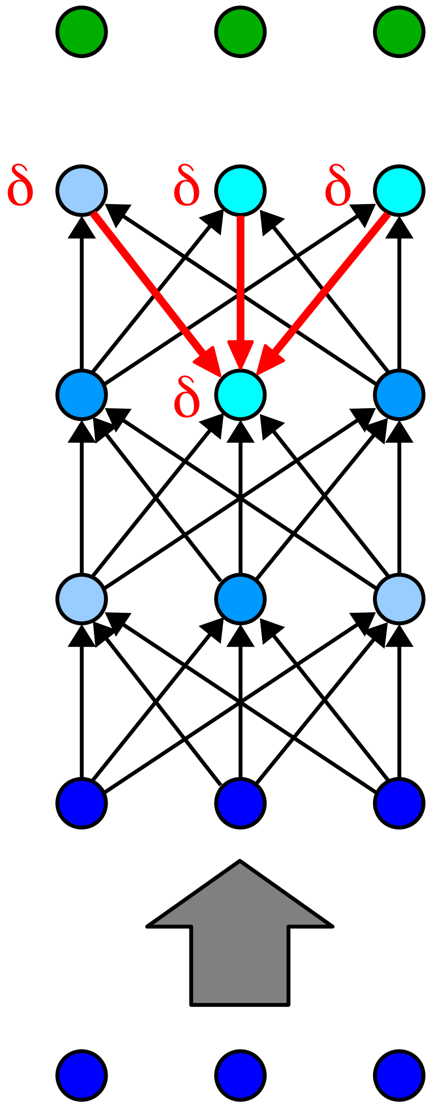
Target pattern:

Compute the δ value for each unit

Output pattern:

Input pattern:

Backpropagation Algorithm

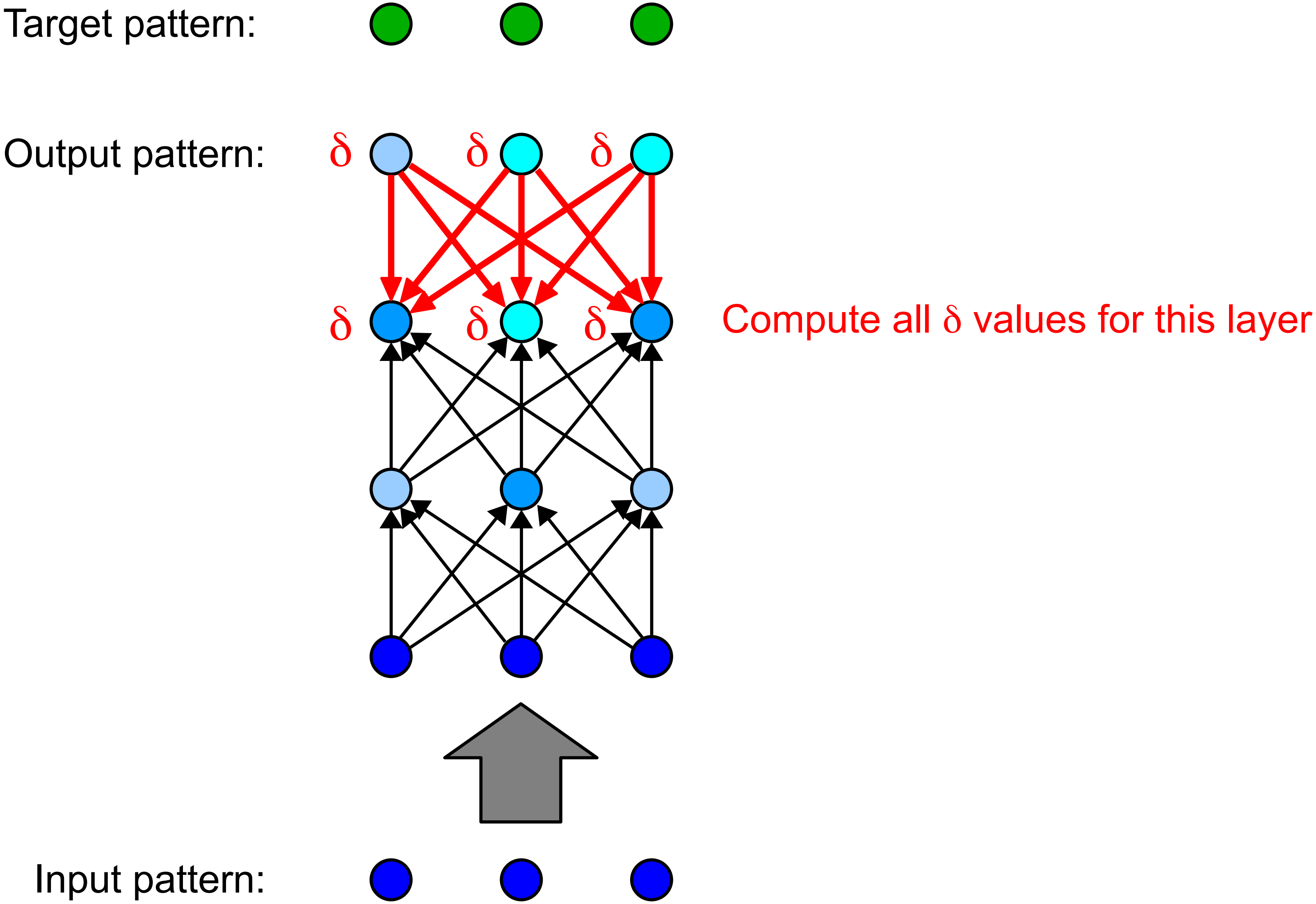


Target pattern:

Output pattern:

Input pattern:

Backpropagation Algorithm



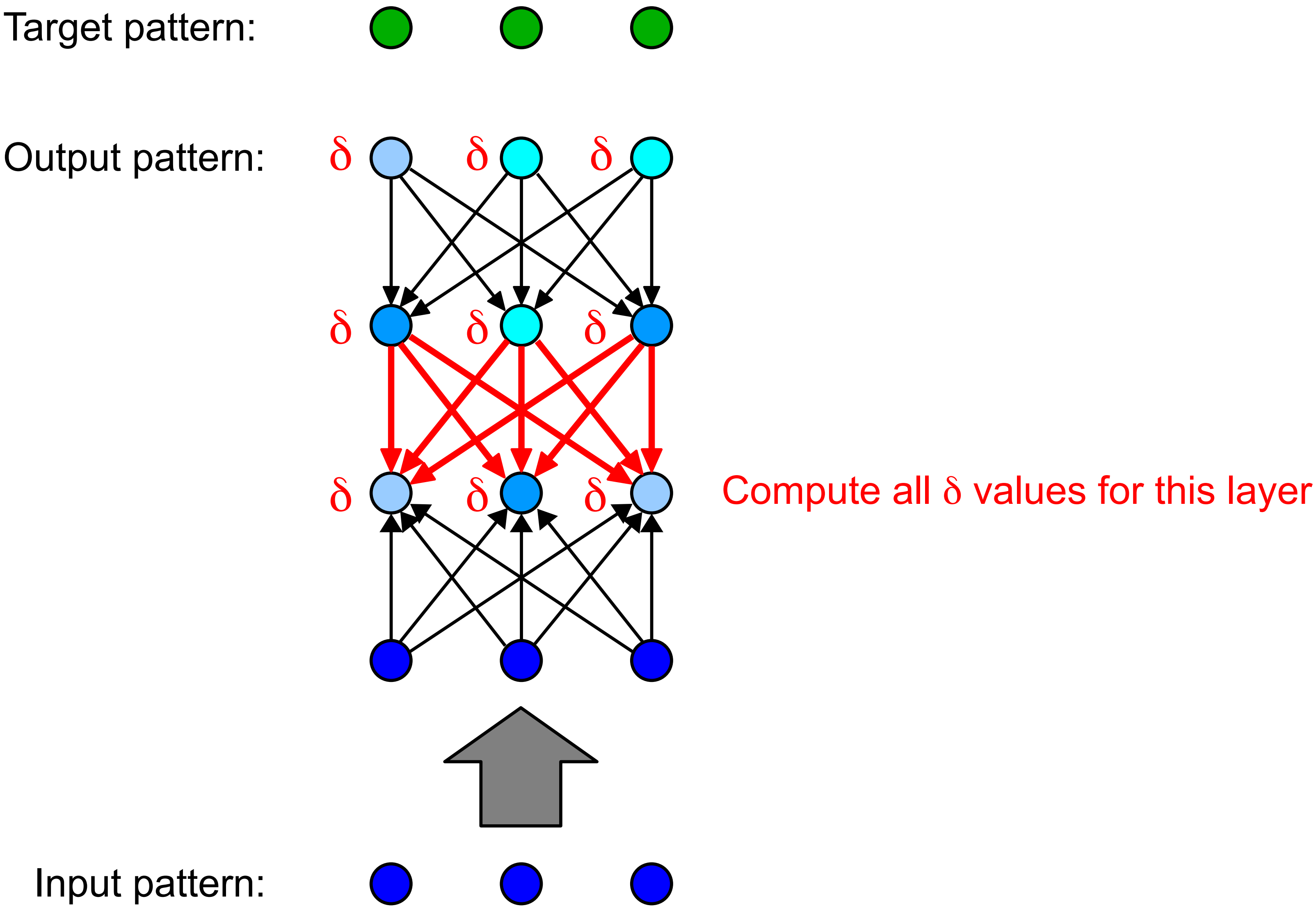
Target pattern:

Output pattern:

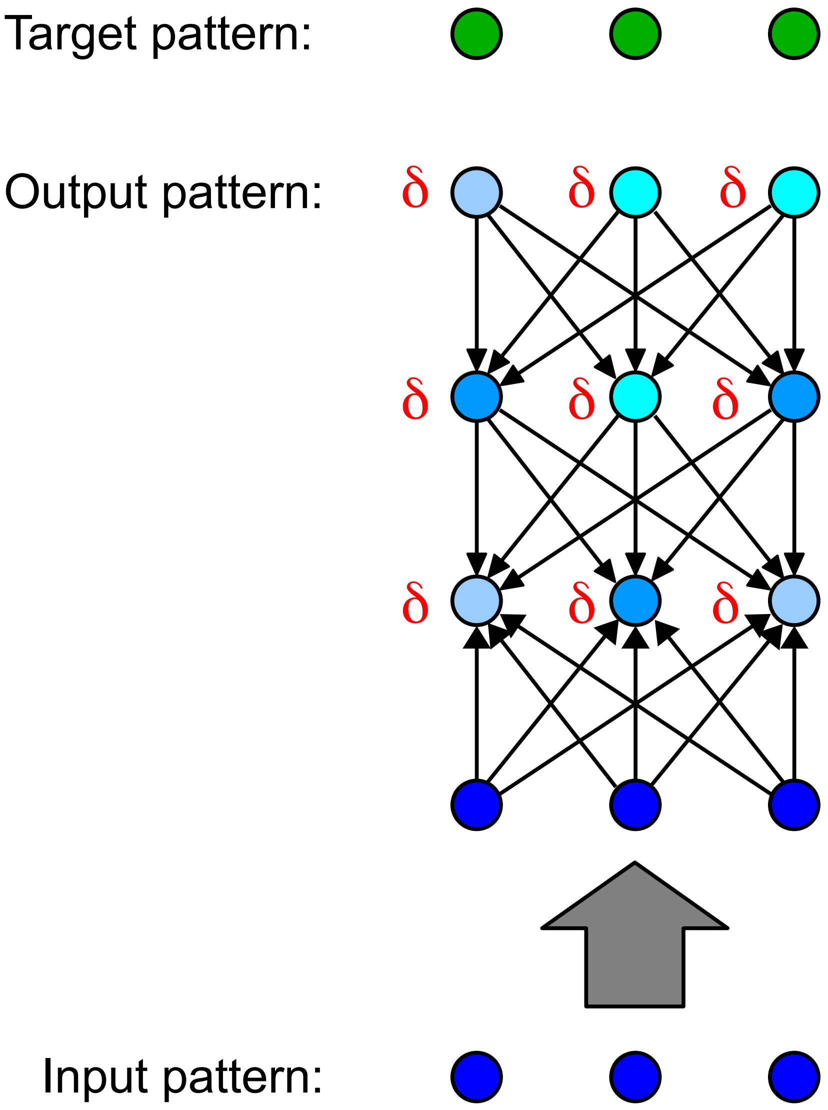
Compute all δ values for this layer

Input pattern:

Backpropagation Algorithm

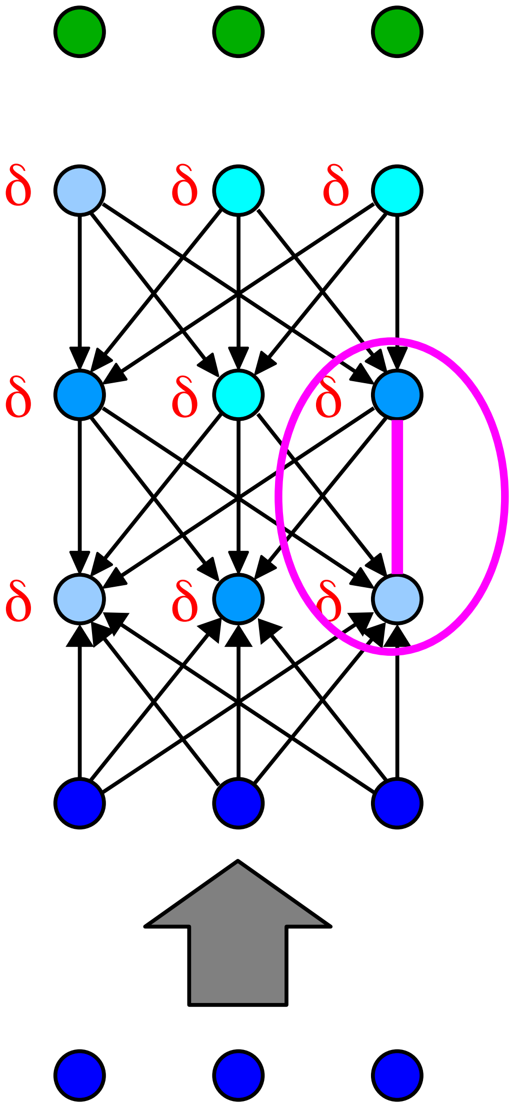


Backpropagation Algorithm



And so on …
for each hidden layer

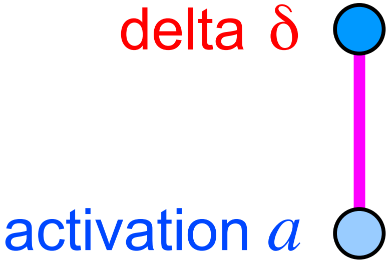
Backpropagation Algorithm



Target pattern:

Output pattern:

Input pattern:

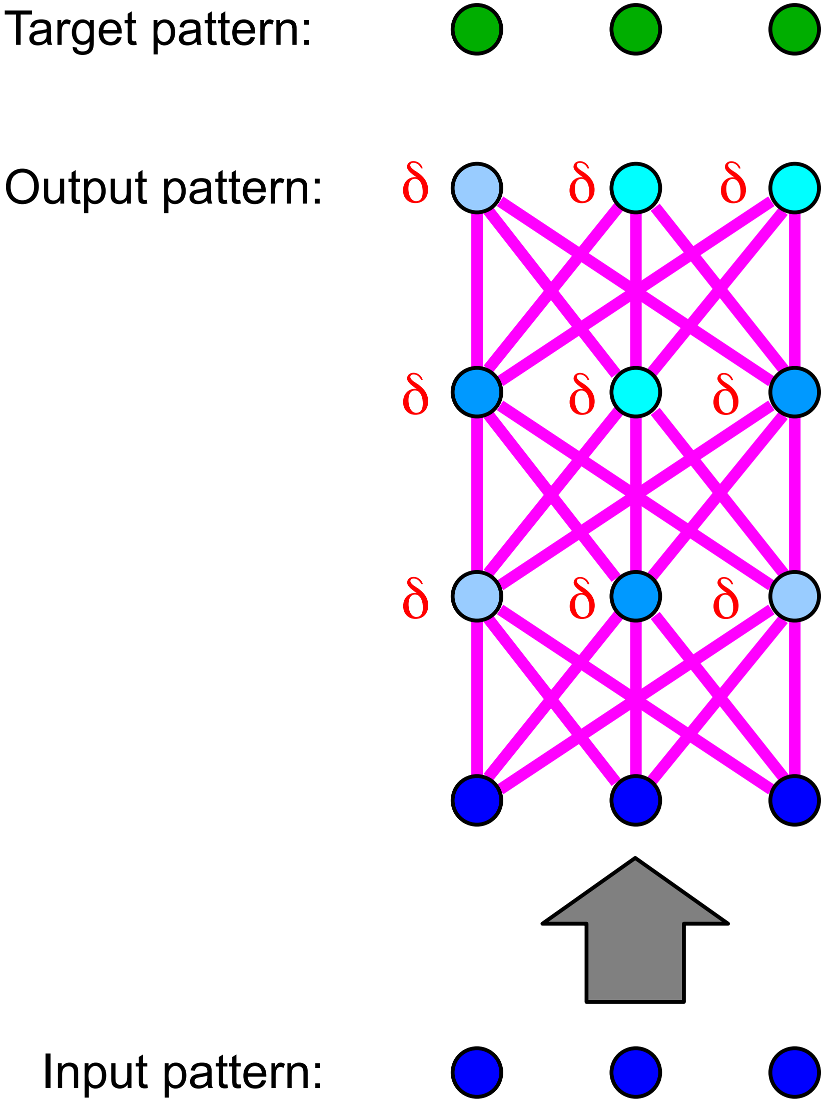


Amount to change weight:

$$\boxed{\Delta w = -\eta \times \delta \times a}$$

Learning rate

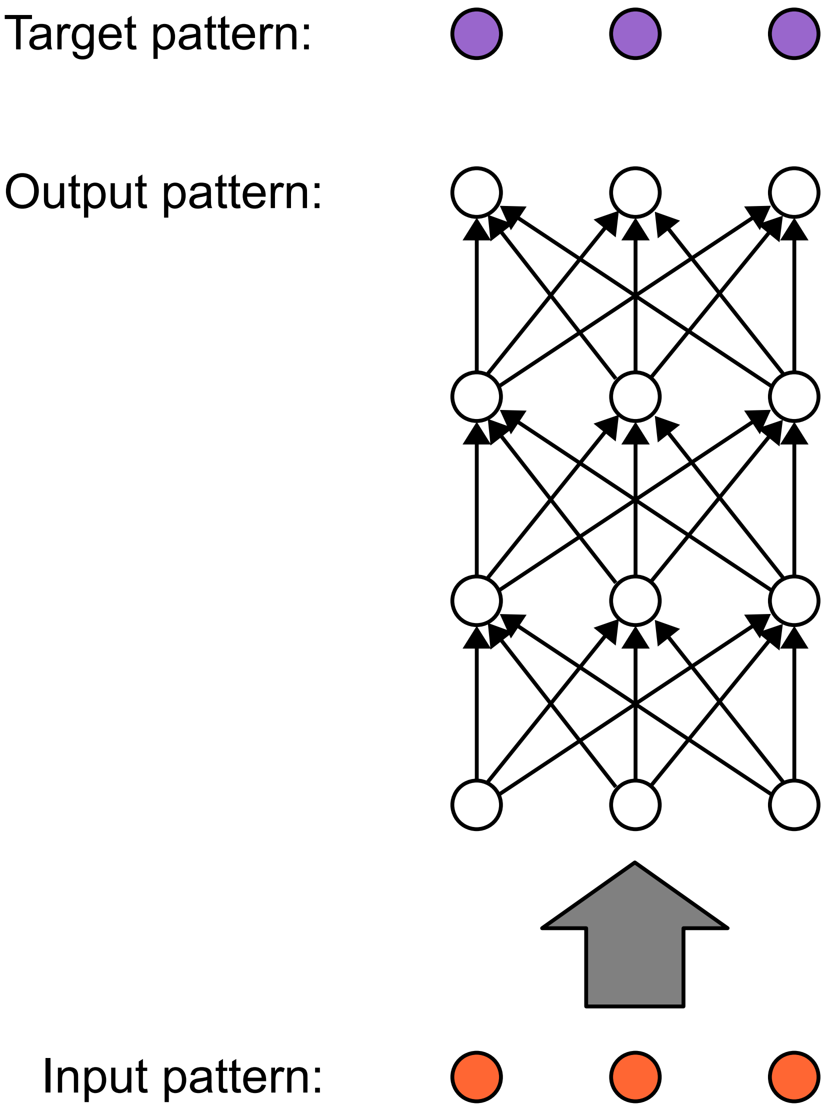
Backpropagation Algorithm



Update ALL weights in the same way

$$\Delta w = -\eta \times \delta \times a$$

Backpropagation Algorithm



Choose another input and target pattern and continue