

# *Artificial Neural Networks in Artificial Intelligence*

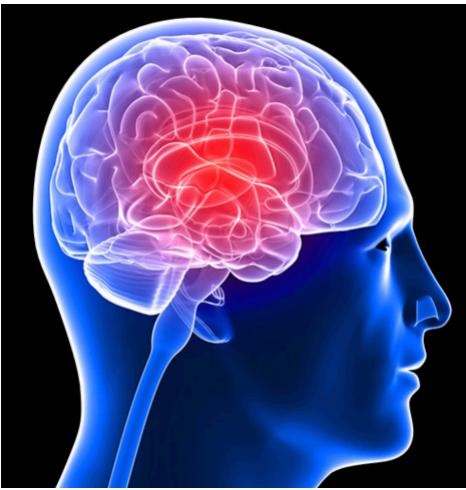
**Artificial Intelligence involves several different approaches, including**

- **Symbolic artificial intelligence** is the collection of all methods in artificial intelligence research that are based on high-level "symbolic" (human-readable) representations of problems, logic and search.
- **Bayesian decision networks** are graphical models that represent a set of variables and their dependencies ... ideal for taking an event that occurred and predicting the likelihood that any one of several possible causes was the contributing factor.
- **Evolutionary algorithms** use mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Candidate solutions play the role of individuals in a population, and the fitness function determines the quality of the solutions. Evolution of the candidate population then takes place after the repeated application of the mechanisms.

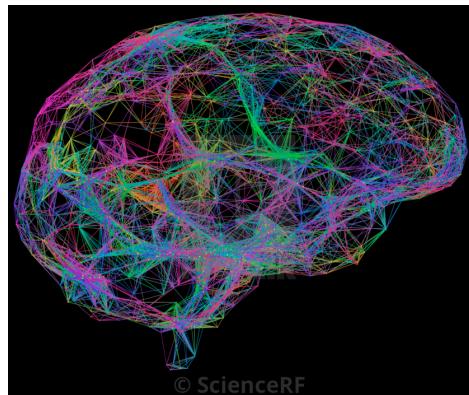
*Wikipedia*

**Here we illustrate**

- **Artificial Neural Networks**

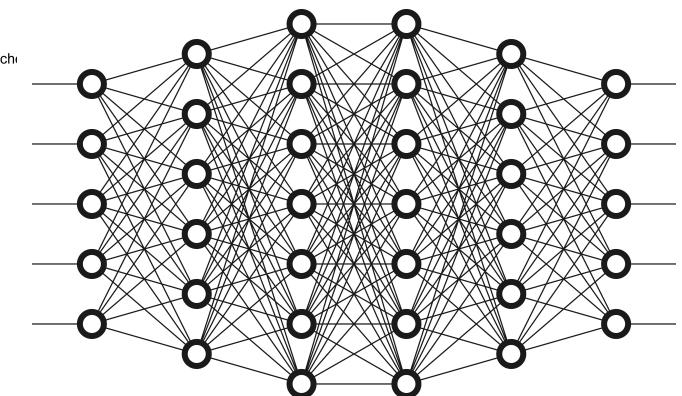
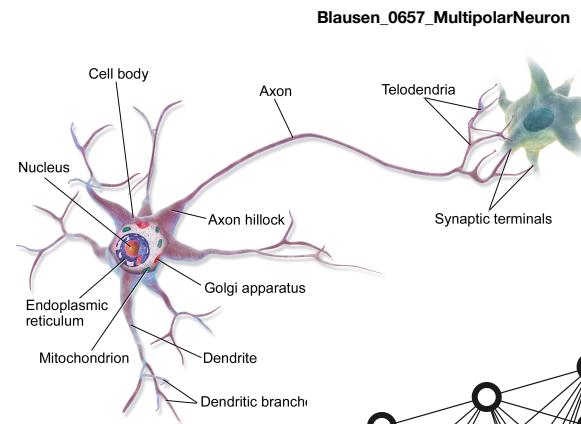


[www.braininjuryaustralia.org.au](http://www.braininjuryaustralia.org.au)



@ScienceRF

**Our brains sense and think using connected networks of cells called neurons**



neural\_network\_shutterstock\_all\_is\_magic.jpg

**These networks are the inspiration for computer simulations of “neural networks,” which can be trained to sense and solve complex problems**

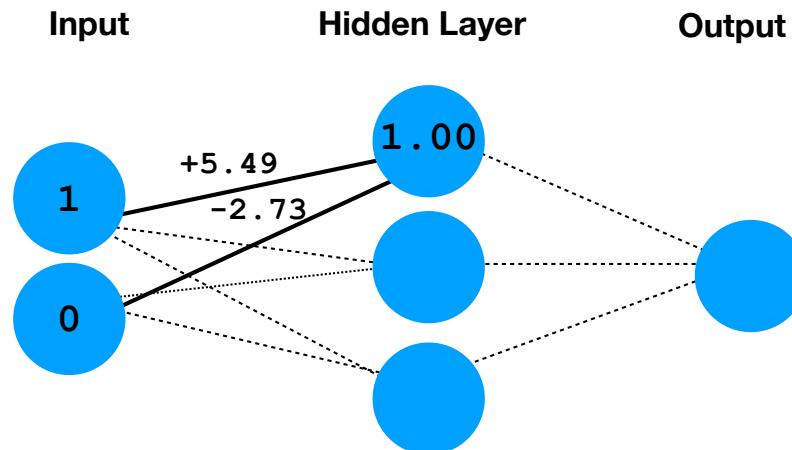
[github.com/RichardHerz](https://github.com/RichardHerz)

# Neural Network

Simulates XOR logic - exclusive or

Output is TRUE when one input is TRUE but not both

**2 inputs, 1 output,  
1 hidden layer  
with 3 neurons &  
9 synapses**



EXAMPLE for input of  
1  
0

These sums over all nodes in a layer are the product of matrix multiplication. Matrix multiplication is well suited to being accelerated in hardware Graphical Processing Units, since graphic transformations also involve matrix multiplication.

node value = sigmaFunc( sum of (node activation \* connection weight) )

where  $\text{sigmaFunc}(x) = \exp(x) / (1 + \exp(x))$  >> converts all input x values into range 0 to 1

INPUT > HIDDEN LAYER

$\text{sigmaFunc}(1 * 5.4868 + 0 * (-2.7276)) = 0.9959$  = hidden node 1 activation

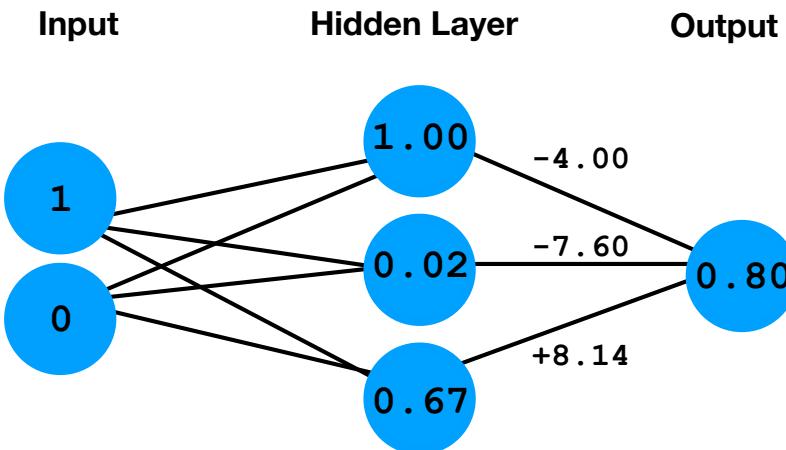
The connection - synapse - weights for a network are determined as the network is “trained” using combinations of known inputs and outputs

# Neural Network

Simulates XOR logic - exclusive or

Output is TRUE when one input is TRUE but not both

**2 inputs, 1 output,  
1 hidden layer  
with 3 neurons &  
9 synapses**



HIDDEN LAYER > OUTPUT

$$\text{sigmaFunc}( 0.9959 * (-4.0030) + 0.0153 * (-7.5988) + 0.6719 * 8.1402 ) = 0.7969 = \text{output node}$$

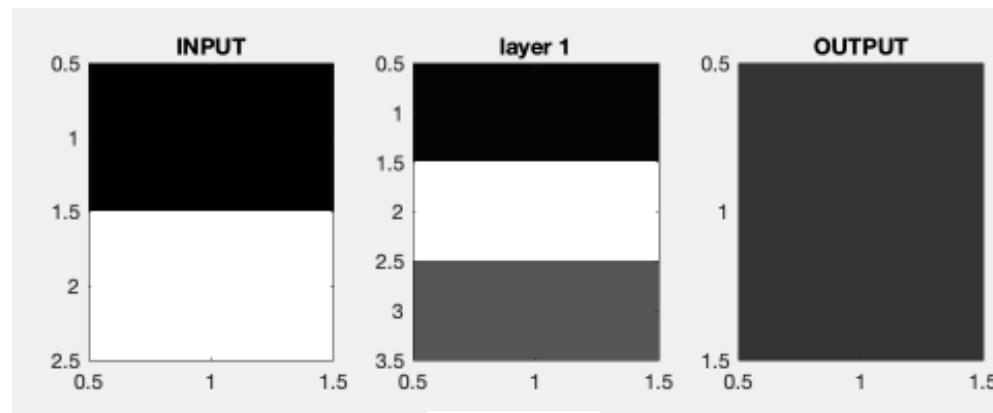
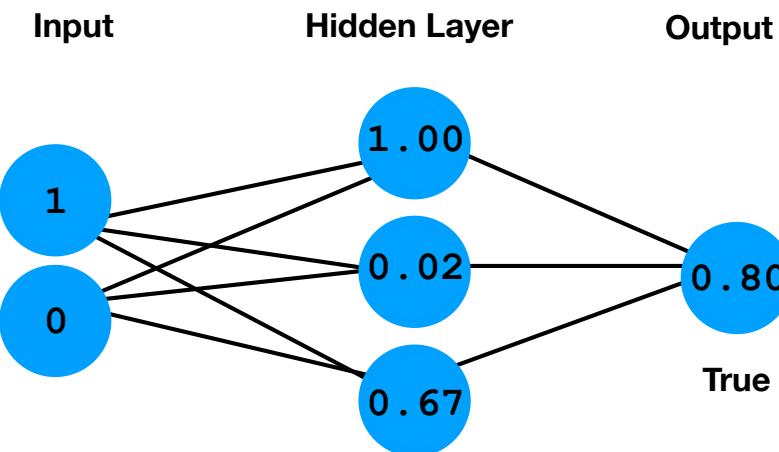
**Output is TRUE when one input is TRUE but not both**

# Neural Network

Simulates XOR logic - exclusive or

Output is TRUE when one input is TRUE but not both

**2 inputs, 1 output,  
1 hidden layer  
with 3 neurons &  
9 synapses**



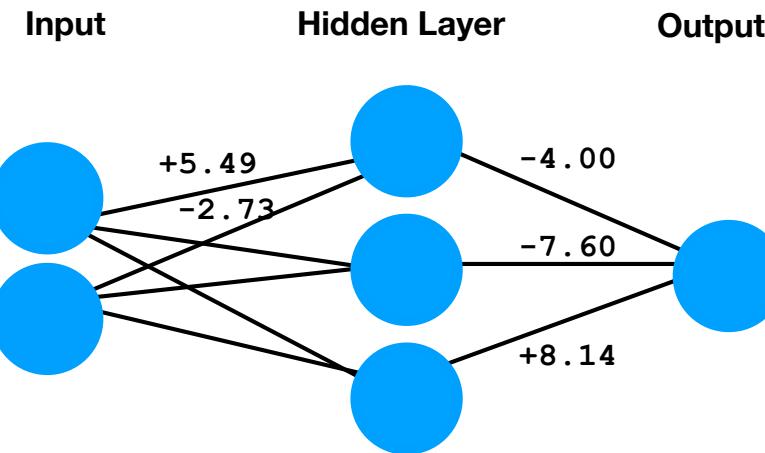
1	0.9959	
0	0.0153	0.7969
	0.6719	True

# Neural Network

Simulates XOR logic - exclusive or

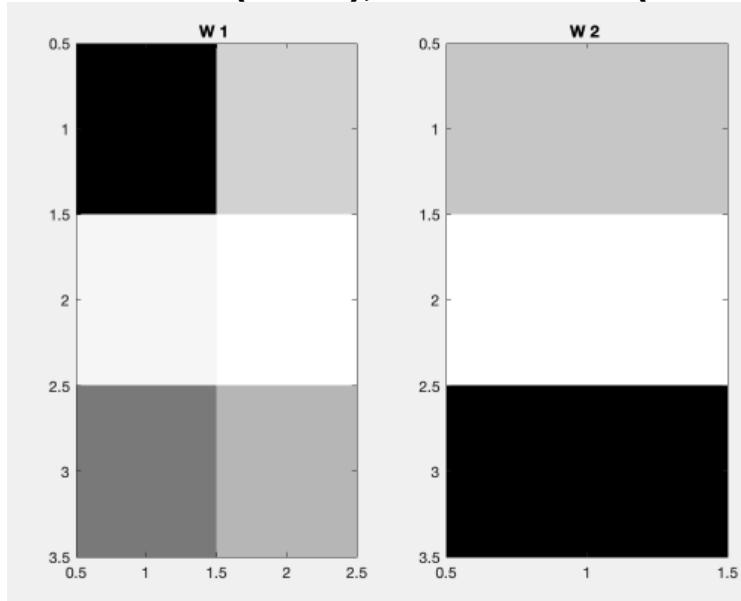
Output is TRUE when one input is TRUE but not both

**2 inputs, 1 output,  
1 hidden layer  
with 3 neurons &  
9 synapses**

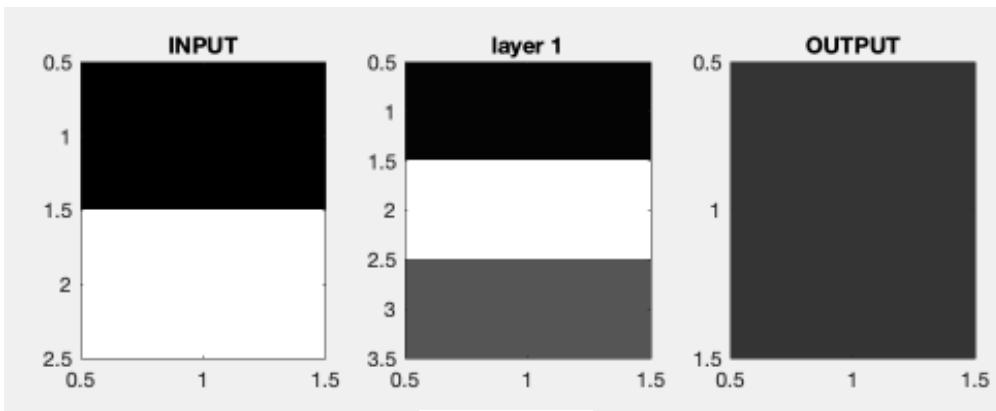


**Synapse weights to hidden  
layer and to output,  
min = -7.60 (white), max = +8.14 (black)**

5.4868	-2.7276
-4.1663	-4.4991
0.7169	-1.5748



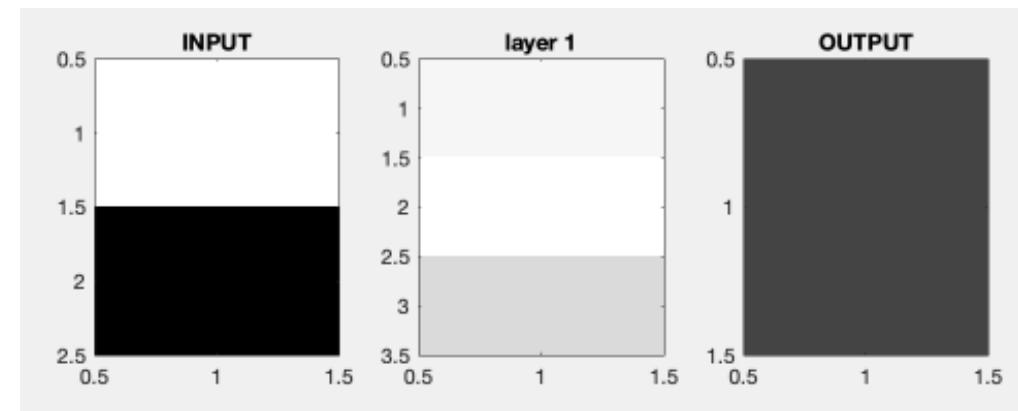
## Below are node values: input > hidden layer > output



1  
0

**0.9959**  
**0.0153**  
**0.6719**

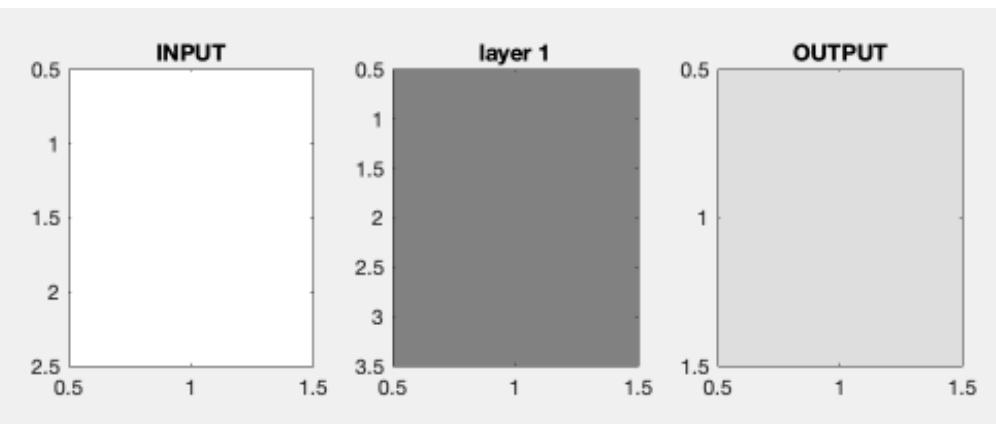
**0.7969**  
**True**



**0**  
**1**

**0.0614**  
**0.0110**  
**0.1715**

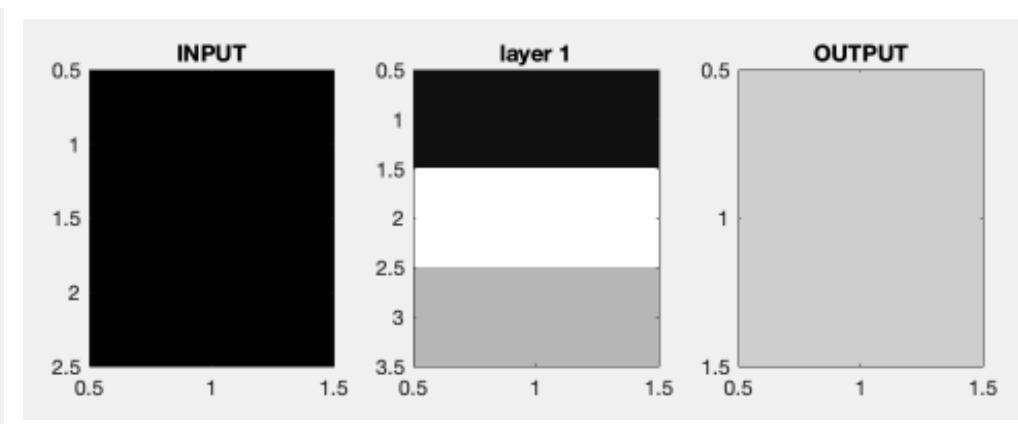
**0.7440**  
**True**



0  
0

**0.5000**  
**0.5000**  
**0.5000**

**0.1505**  
**False**



1  
1

**0.9404**  
**0.0002**  
**0.2978**

**0.2072**  
**False**

**Output is TRUE when one input is TRUE but not both**

## Neural Network

**4 inputs, 4 outputs**

**4 hidden layers, each  
with 60 neurons =  
240 neurons &  
11,280 synapses**

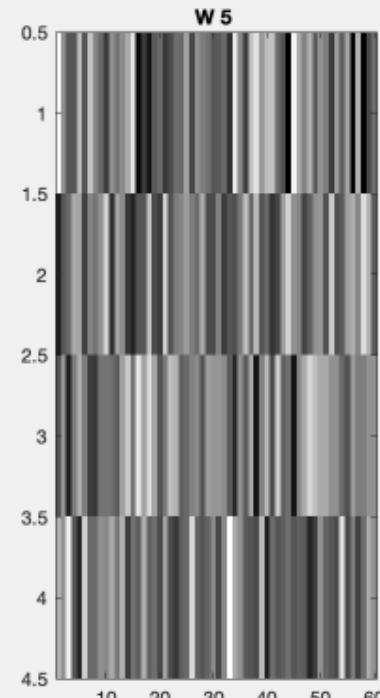
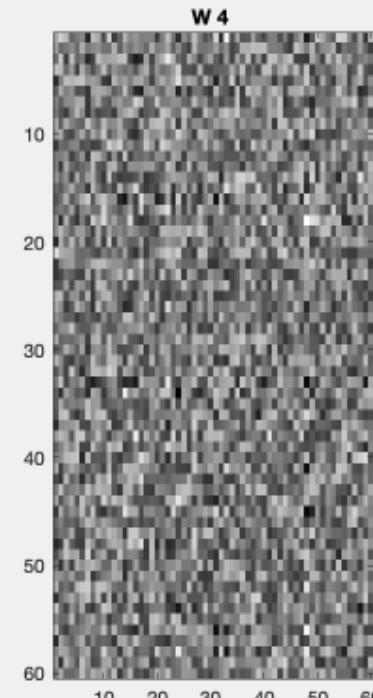
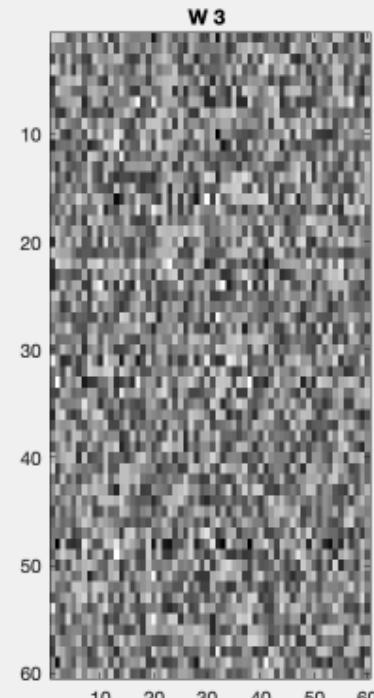
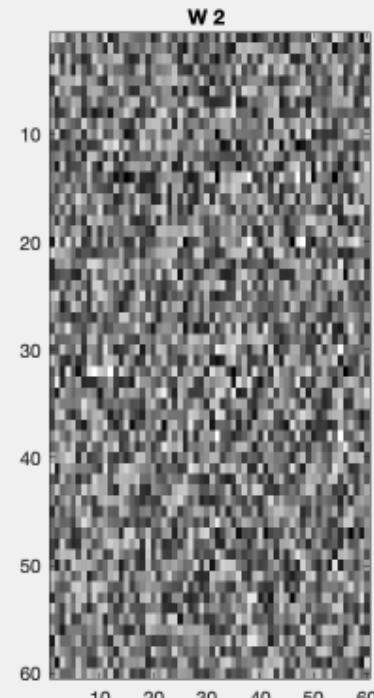
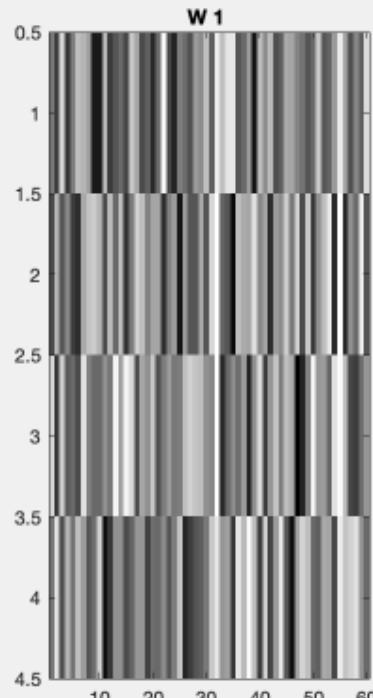
*A little more complex network which  
detects diagonal, horizontal and vertical  
inputs to a 2 x 2 “touch screen”*



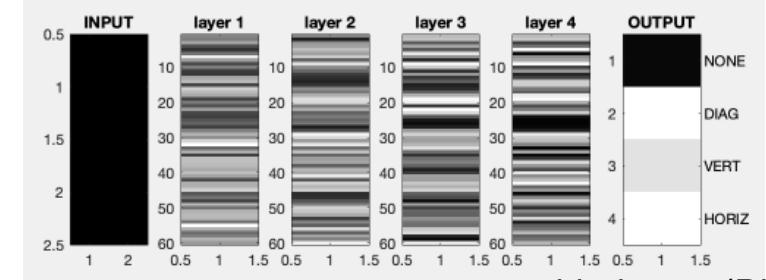
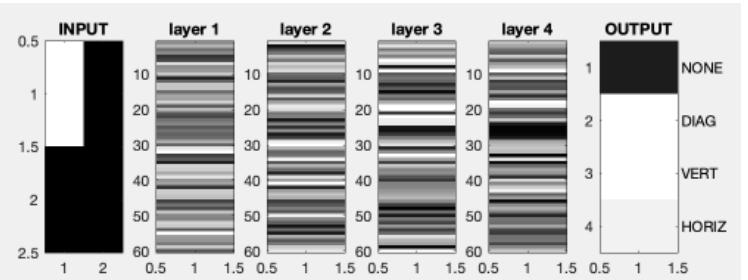
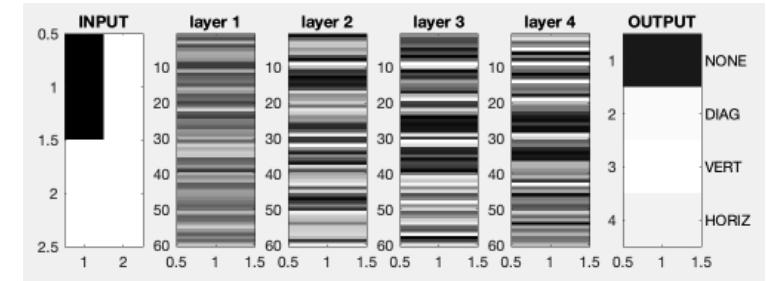
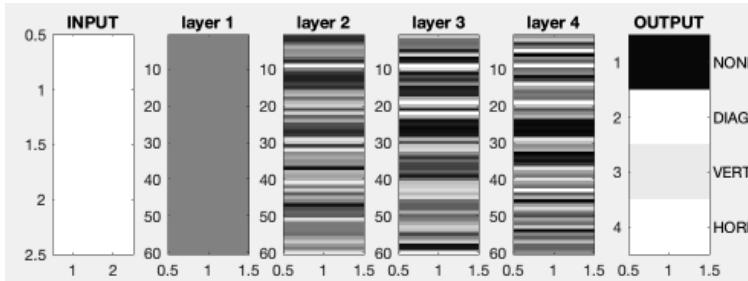
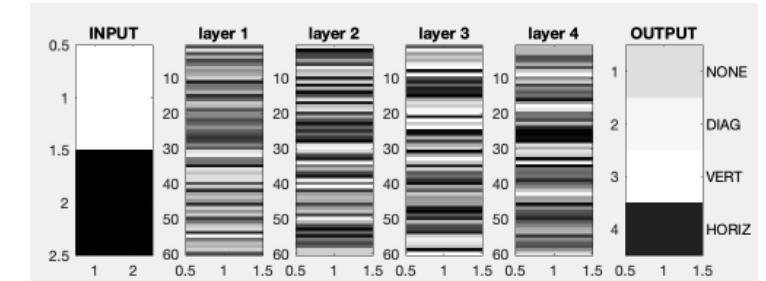
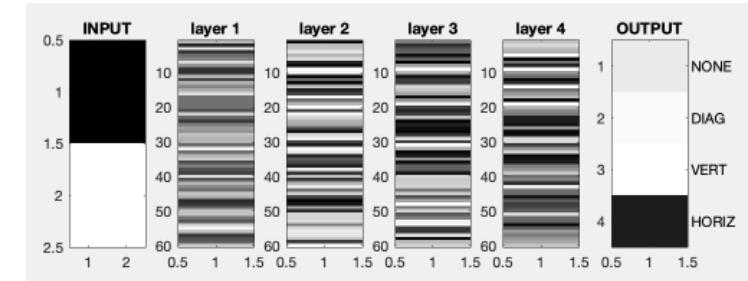
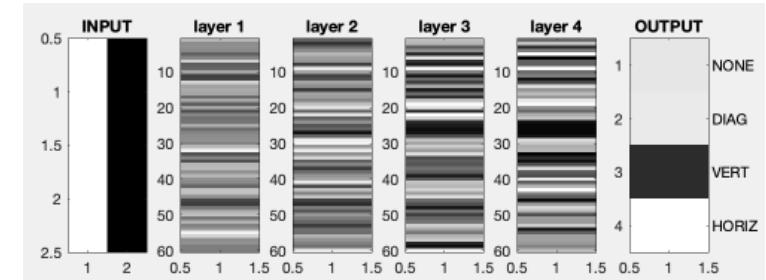
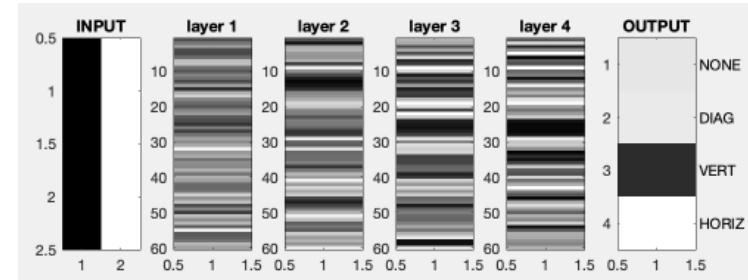
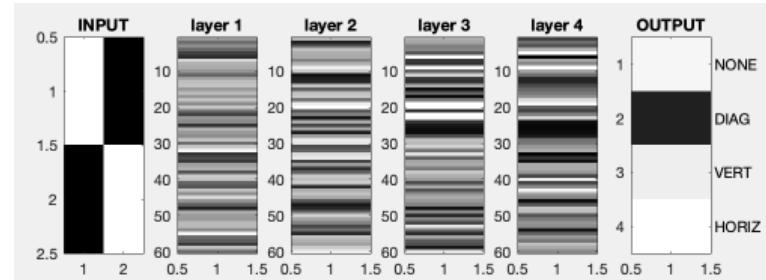
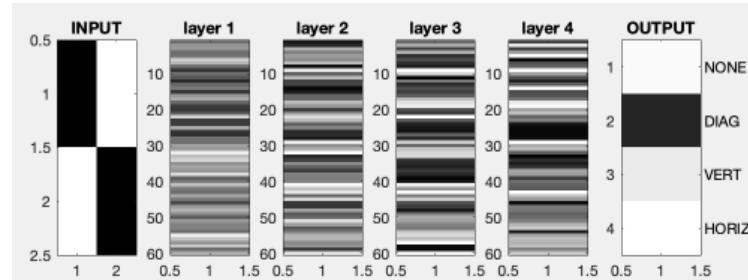
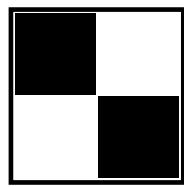
**Synapse weights to hidden**

**layers 1-4 and to output,**

**min = -1.23 (white), max = +1.25 (black)**

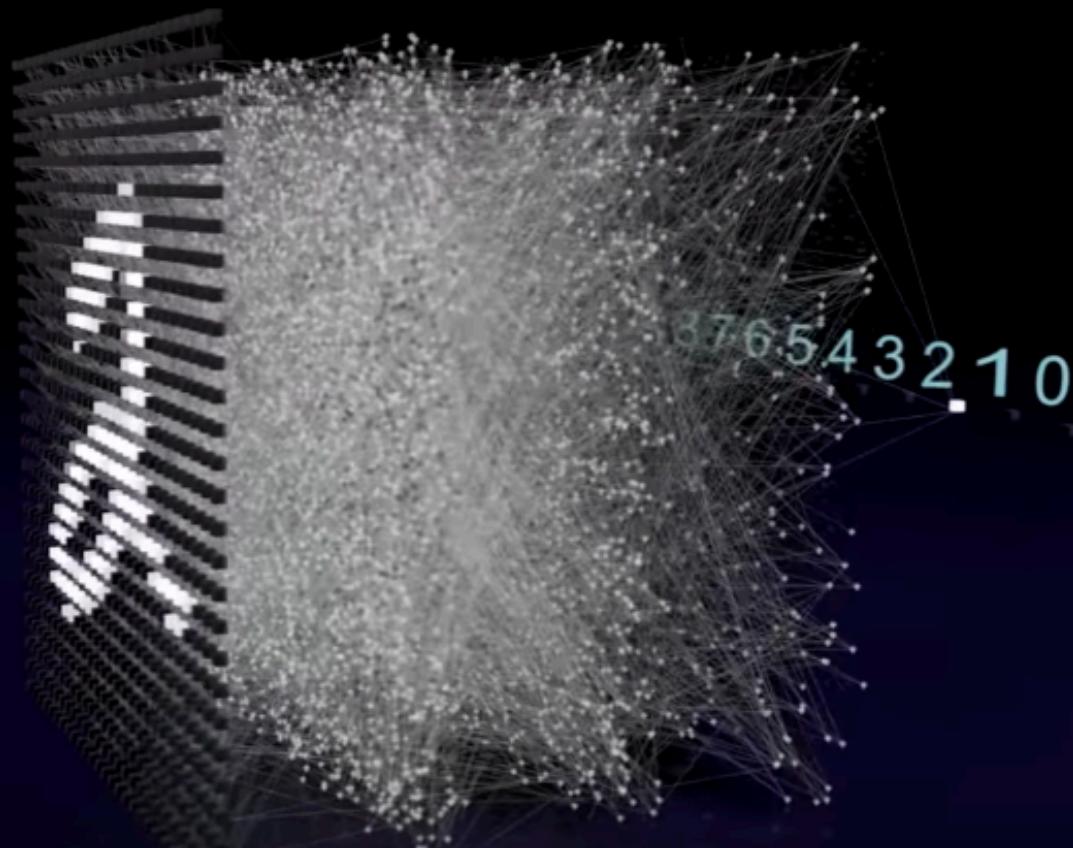


## Below are node activations: input > 4 hidden layers > output



**A neural network for a 28 x 28 “touch screen”  
note that only 2% of the 24+ million “synapses” are shown**

Type: ML Perceptron  
Data Set: MNIST  
Hidden Layers: 3  
Hidden Neurons: 10000  
Synapses: 24864180  
Synapses shown: 2%  
Learning: BP



Denis Dmitriev <https://youtu.be/3JQ3hYko51Y>

**A neural network represents a large number of coupled equations which, when given a set of input values, can produce a set of desired output values.**

**The more neurons and synapses - the more equations - and the greater complexity of inputs and outputs which can be "fit" by the system of equations.**

**"Deep learning" refers to solving complex problems using many hidden layers of neurons and, thus, many equations.**