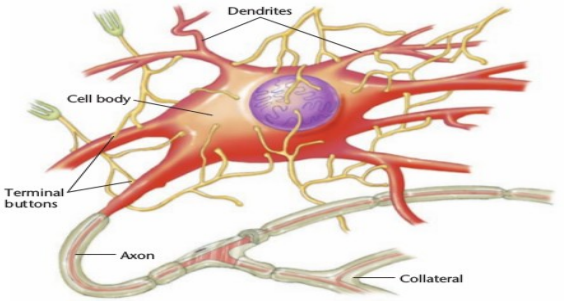
Mathematical models of the brain

Mathematical models of the brain

- McCulloch & Pitts neuron
- Rosenblatt’s perceptron
- Multi-layer perceptron
- Hebbian learning
- Hodgkin-Huxley model
- Recurrent neural networks
- Hopfield network

McCulloch and Pitts neurons

- McCulloch and Pitts (1943) assumptions:
  - They are binary devices (Vi = [0,1])
  - Each neuron has a fixed threshold, theta
  - The neuron receives inputs from excitatory synapses, all having identical weights.
  - Inhibitory inputs have an absolute veto power over any excitatory inputs.
  - At each time step the neurons are simultaneously (synchronously) updated by summing the weighted excitatory inputs and setting the output (Vi) to 1 iff the sum is greater than or equal to the threshold AND if the neuron receives no inhibitory input.



McCulloch and Pitts neurons

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. MCCULLOCH and WALTER H. PITTS

Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

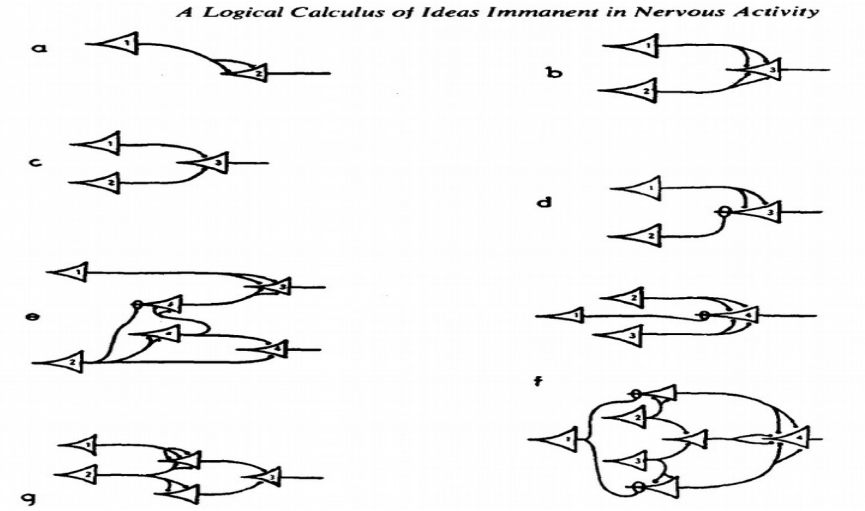


Figure 1a $N_2(t) \cdot \equiv \cdot N_1(t-1)$

Figure 1b $N_3(t) \cdot \equiv \cdot N_1(t-1) \vee N_2(t-1)$

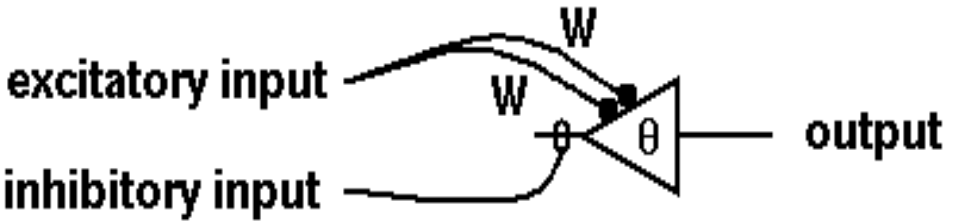
Figure 1c $N_3(t) \cdot \equiv \cdot N_1(t-1) \cdot N_2(t-1)$

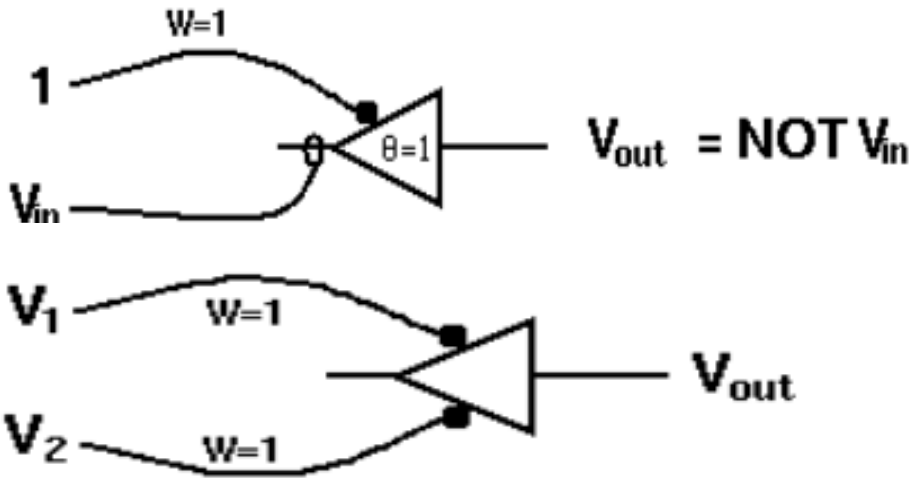
Figure 1d $N_3(t) \cdot \equiv \cdot N_1(t-1) \cdot \sim N_2(t-1)$

Figure 1e $N_3(t) : \equiv : N_1(t-1) \cdot \vee \cdot N_2(t-3) \cdot \sim N_2(t-2)$
$N_4(t) \cdot \equiv \cdot N_2(t-2) \cdot N_2(t-1)$

Figure 1f $N_4(t) : \equiv : \sim N_1(t-1) \cdot N_2(t-1) \vee N_3(t-1) \cdot \vee \cdot N_1(t-1) \cdot N_2(t-1) \cdot N_3(t-1)$
$N_4(t) : \equiv : \sim N_1(t-2) \cdot N_2(t-2) \vee N_3(t-2) \cdot \vee \cdot N_1(t-2) \cdot N_2(t-2) \cdot N_3(t-2)$

McCulloch and Pitts neurons





Quiz - NAND gate

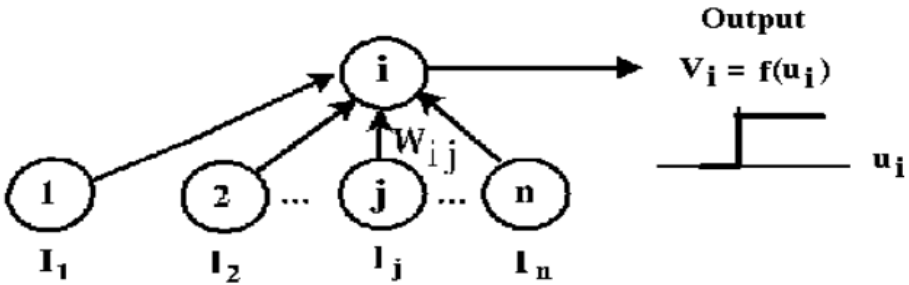
TRUTH TABLE

INPUTS			OUTPUT
W	X	Y	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

http://ecee.colorado.edu/~ecen4831/lectures/NNet2.html

Rosenblatt's simple perceptron

- The weights and thresholds were not all identical.
- Weights can be positive or negative.
- There is no absolute inhibitory synapse.
- Although the neurons were still two-state, the output function f(u) goes from [-1,1], not [0,1].
- Most importantly, there was a learning rule.



$$V_i = f(u_i) = \begin{cases} 0 & : \quad u_i < 0 \\ 1 & : \quad u_i \geq 0 \end{cases}$$

$$u_i = \sum_j W_{ij} I_j + \theta_i$$

http://ecee.colorado.edu/~ecen4831/lectures/NNet2.html

Learning with the perceptron

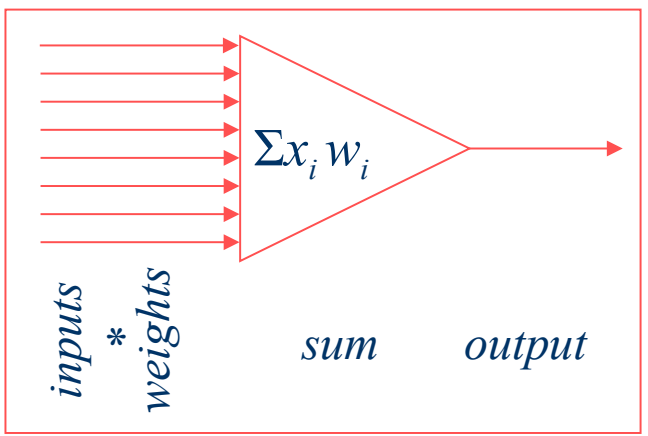
- $T = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_n, y_n)\}$ is a training set of n pairs of input $\mathbf{x}_i$ and desired output y_i
- To learn the correct weights **w**:
  - Initialize **w** randomly
  - For each sample j do:
    - Calculate the actual output $y'_j = \mathbf{w}\mathbf{x}_j$
    - Adapt the weights $\mathbf{w}_k' = \mathbf{w}_k + \alpha(y_j - y_j')\mathbf{x}_{jk}$ for each $\mathbf{w}_k$
  - Repeat until the error is sufficiently small

The Perceptron

Frank Rosenblatt (1962). *Principles of Neurodynamics*, Spartan, New York

Subsequent progress was inspired by the invention of *learning rules* inspired by ideas from neuroscience…

Rosenblatt's *Perceptron* could automatically learn to categorise or classify input vectors into types.



It obeyed the following rule:

If the sum of the weighted inputs exceeds a threshold, output 1, else output -1.

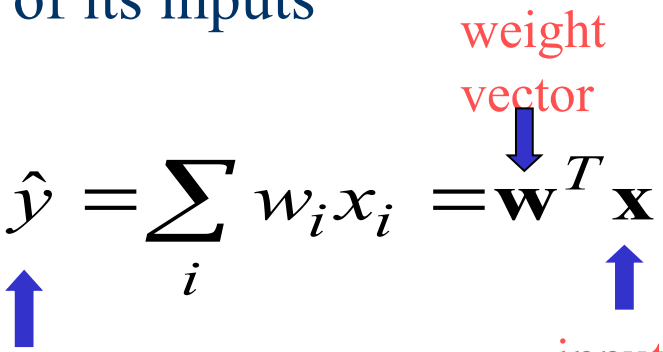
$$1 \text{ if } \Sigma\ input_i * weight_i > threshold$$

$$-1 \text{ if } \Sigma\ input_i * weight_i < threshold$$

https://www.cs.tau.ac.il/~nin/Courses/NC05/SingLayerPerc.ppt

Linear neurons

- The neuron has a real-valued output which is a weighted sum of its inputs



- The aim of learning is to minimize the discrepancy between the desired output and the actual output
  - How de we measure the discrepancies?
  - Do we update the weights after every training case?
  - Why don't we solve it analytically?

https://www.cs.tau.ac.il/~nin/Courses/NC05/SingLayerPerc.ppt

A motivating example

- Each day you get lunch at the cafeteria.
  - Your diet consists of fish, chips, and beer.
  - You get several portions of each
- The cashier only tells you the total price of the meal
  - After several days, you should be able to figure out the price of each portion.
- Each meal price gives a linear constraint on the prices of the portions:

$$price = x_{fish} w_{fish} + x_{chips} w_{chips} + x_{beer} w_{beer}$$

https://www.cs.tau.ac.il/~nin/Courses/NC05/SingLayerPerc.ppt

Two ways to solve the equations

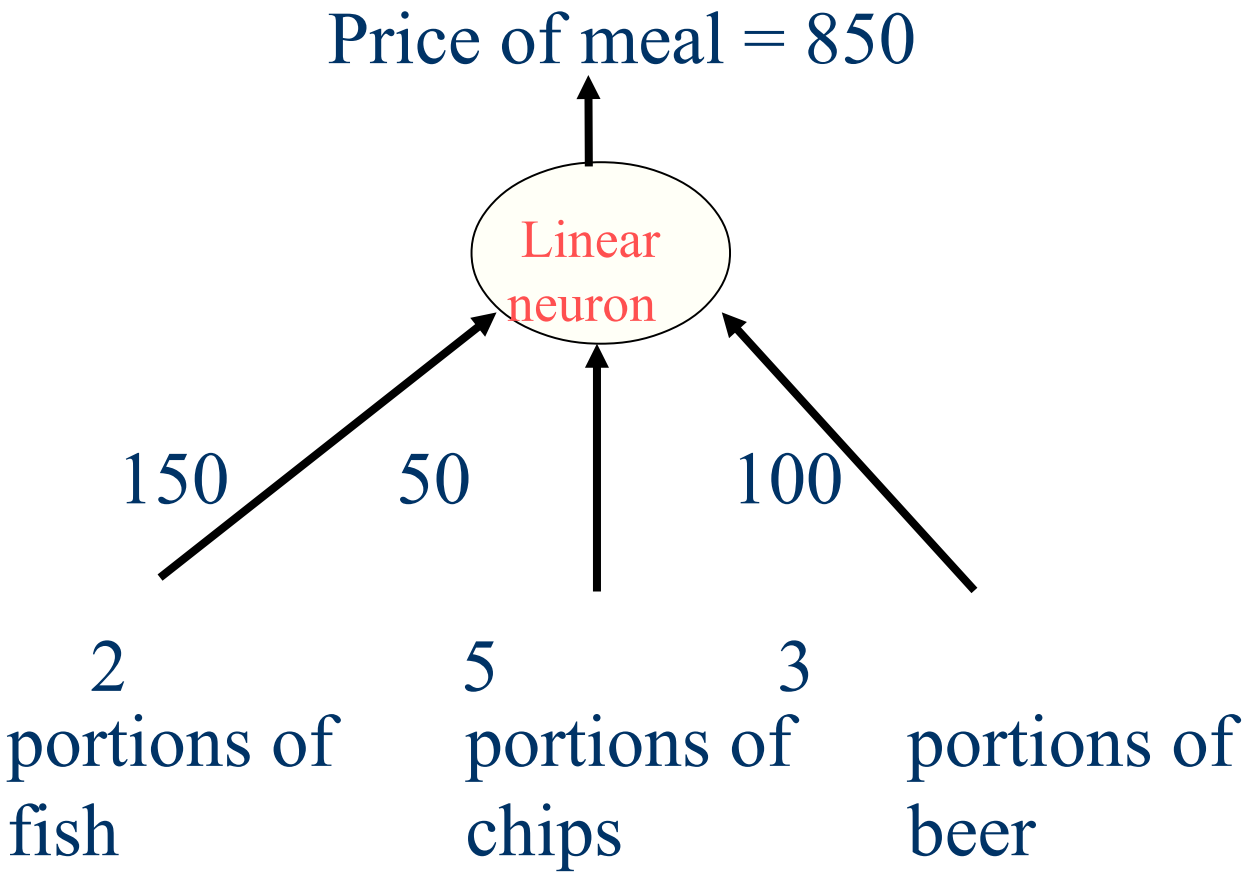
- The obvious approach is just to solve a set of simultaneous linear equations, one per meal.
- But we want a method that could be implemented in a neural network.
- The prices of the portions are like the weights in of a linear neuron.

$$\mathbf{w} = (w_{fish}, w_{chips}, w_{beer})$$

- We will start with guesses for the weights and then adjust the guesses to give a better fit to the prices given by the cashier.

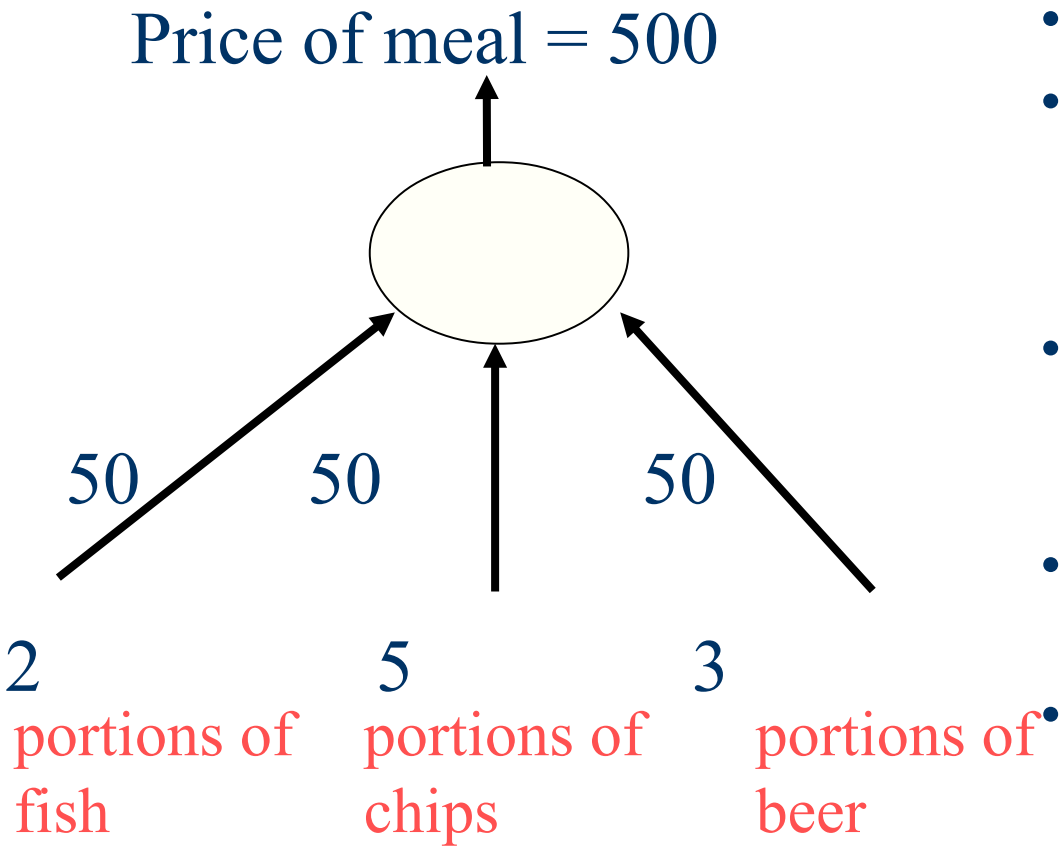
https://www.cs.tau.ac.il/~nin/Courses/NC05/SingLayerPerc.ppt

The cashier's brain



https://www.cs.tau.ac.il/~nin/Courses/NC05/SingLayerPerc.ppt

A model of the cashier's brain with arbitrary initial weights



- Residual error = 350
- The learning rule is:

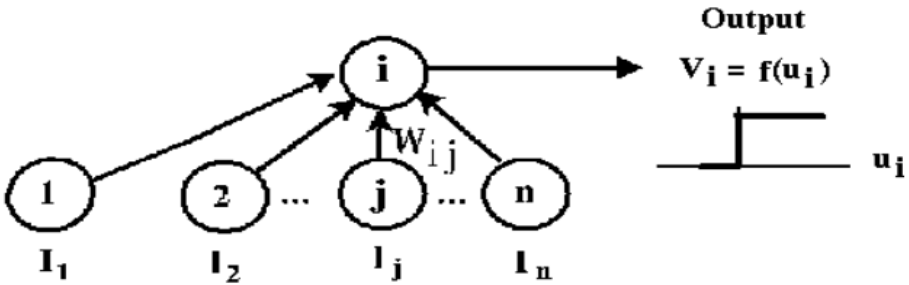
$$\Delta w_i = \varepsilon \, x_i \, (y - \hat{y})$$

- With a learning rate of ε 1/35, the weight changes are +20, +50, +30
- This gives new weights of 70, 100, 80
- Notice that the weight for chips got worse!

https://www.cs.tau.ac.il/~nin/Courses/NC05/SingLayerPerc.ppt

Rosenblatt's simple perceptron

- The weights and thresholds were not all identical.
- Weights can be positive or negative.
- There is no absolute inhibitory synapse.
- Although the neurons were still two-state, the output function f(u) goes from [-1,1], not [0,1].
- Most importantly, there was a learning rule.



$$V_i = f(u_i) = \begin{cases} 0 & : \quad u_i < 0 \\ 1 & : \quad u_i \geq 0 \end{cases}$$

$$u_i = \sum_j W_{ij} I_j + \theta_i$$

http://ecee.colorado.edu/~ecen4831/lectures/NNet2.html

Learning with the perceptron

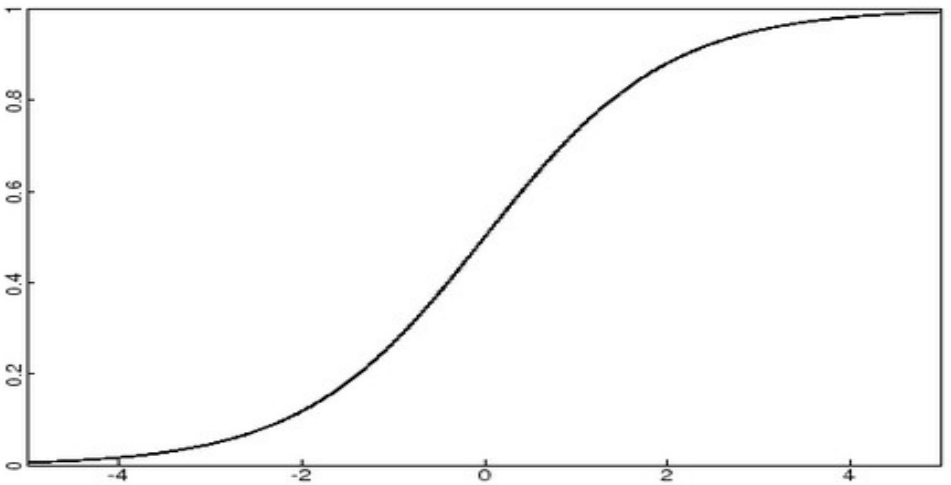
- $T = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_n, y_n)\}$ is a training set of n pairs of input $\mathbf{x}_i$ and desired output y_i
- To learn the correct weights **w**:
  - Initialize **w** randomly
  - For each sample j do:
    - Calculate the actual output $y'_j = \mathbf{w}\mathbf{x}_j$
    - Adapt the weights $\mathbf{w}_k' = \mathbf{w}_k + \alpha(y_j - y_j')\mathbf{x}_{jk}$ for each $\mathbf{w}_k$
  - Repeat until the error is sufficiently small

Other considerations

- Bias term: w_0
- Adding nonlinearity:
  - Logistic function
  - Hyperbolic tangent

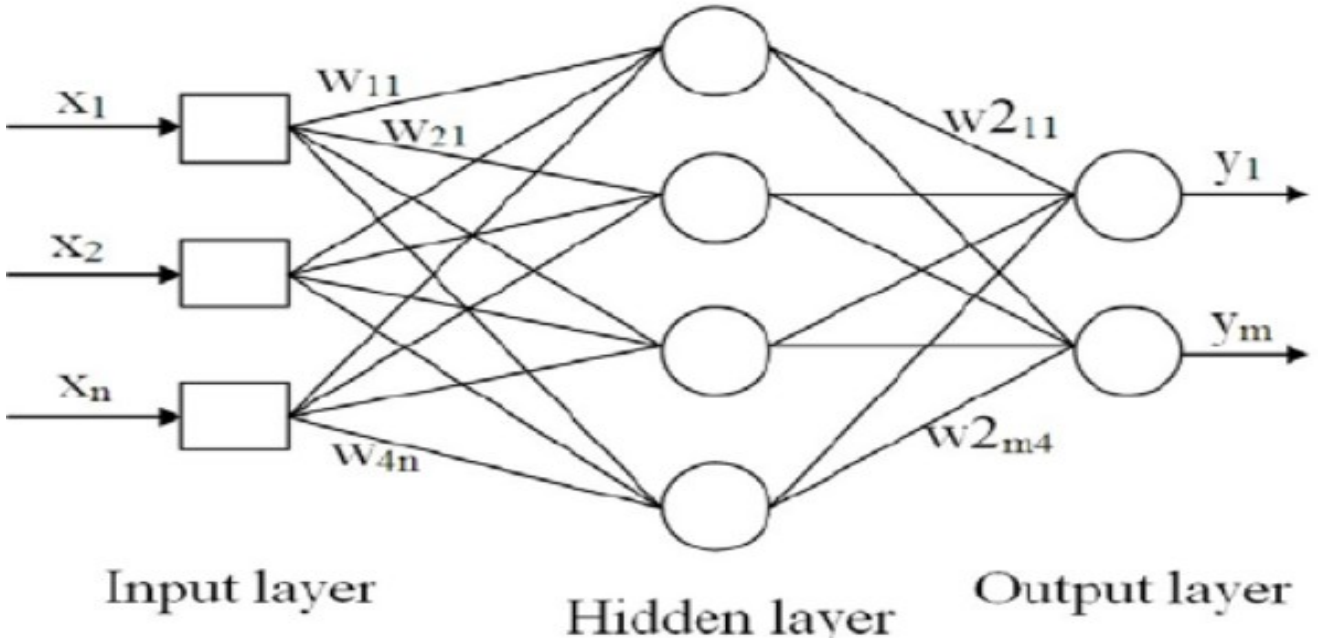
$$f(x) = \frac{1}{1+e^{-\beta x}}$$

$$f(x) = \tanh(\beta x)$$



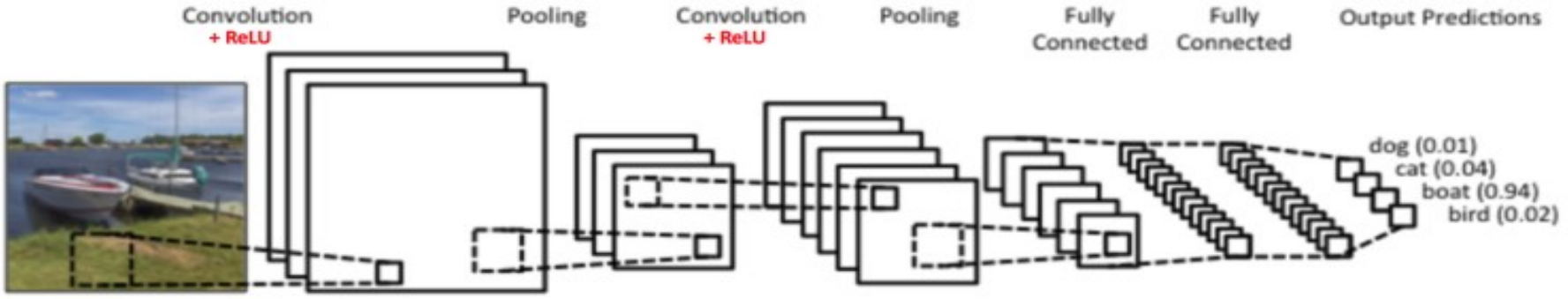
Very limited!

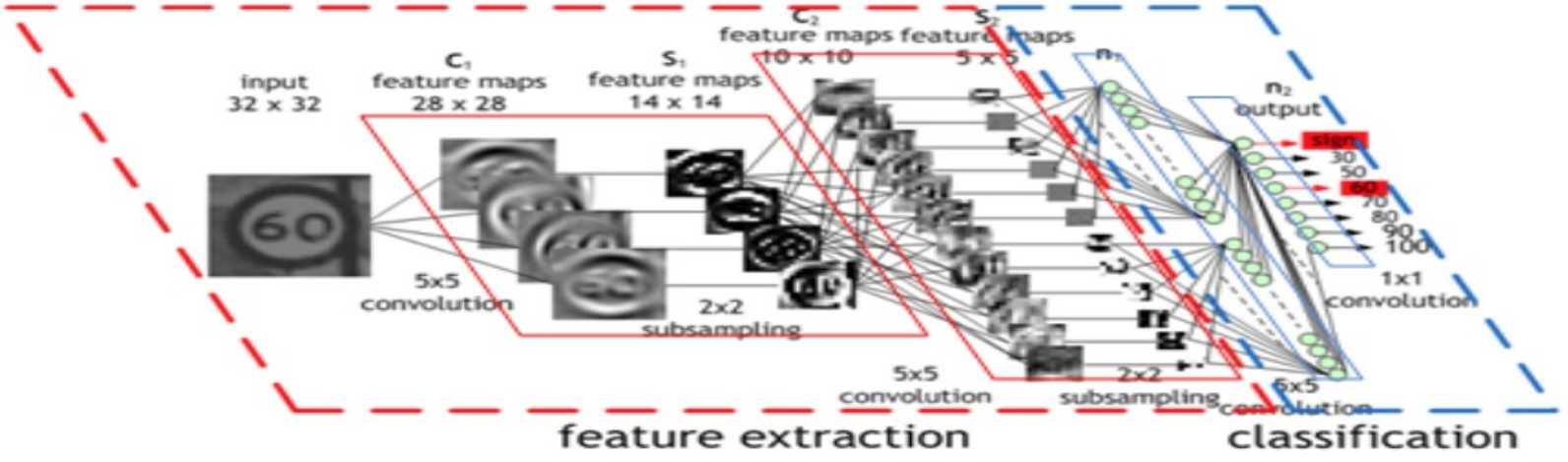
Multilayer perceptron



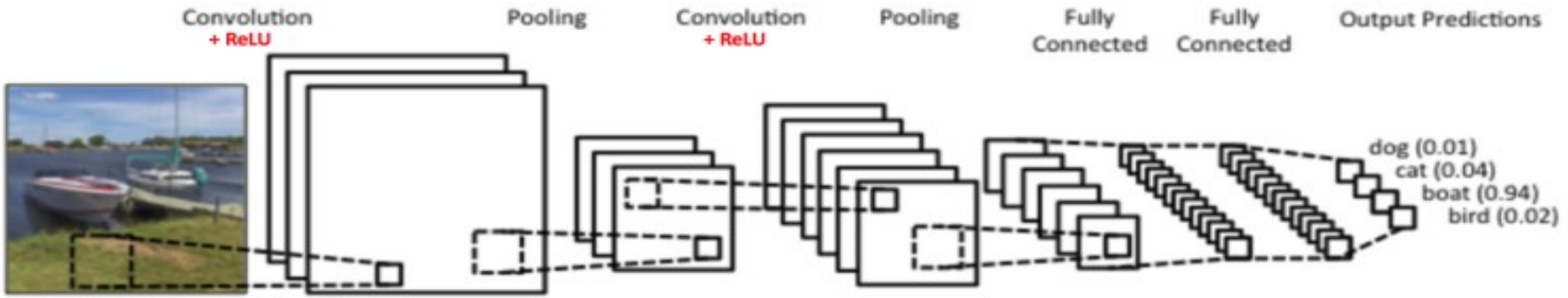
https://www.researchgate.net/profile/Junita_Mohamad-Saleh/publication/257071174/figure/fig3/AS:297526545666050@1447947264431/Figure-5-A-schematic-diagram-of-a-Multi-Layer-Perceptron-MLP-neural-network.png

Deep Neural Networks

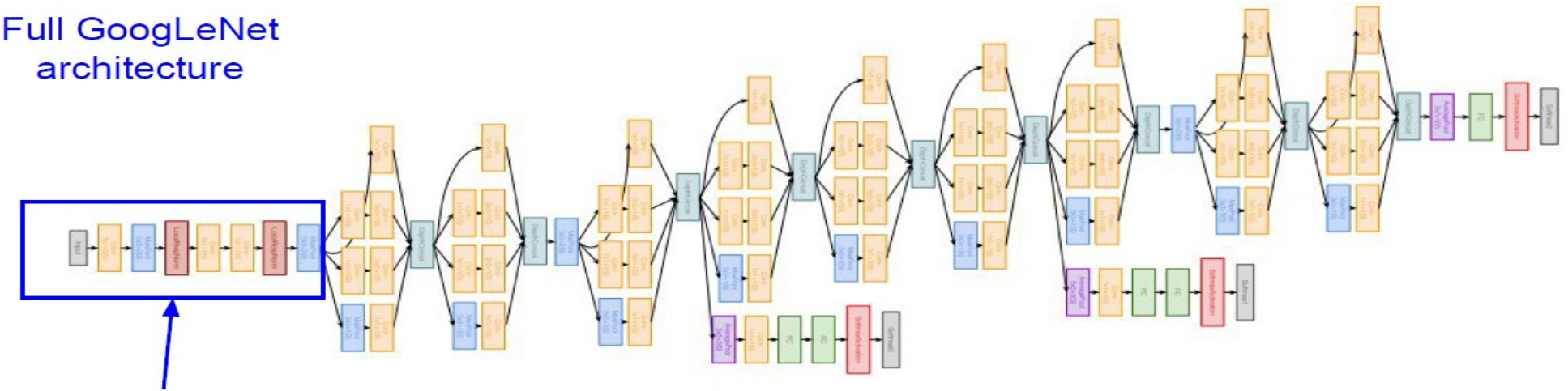




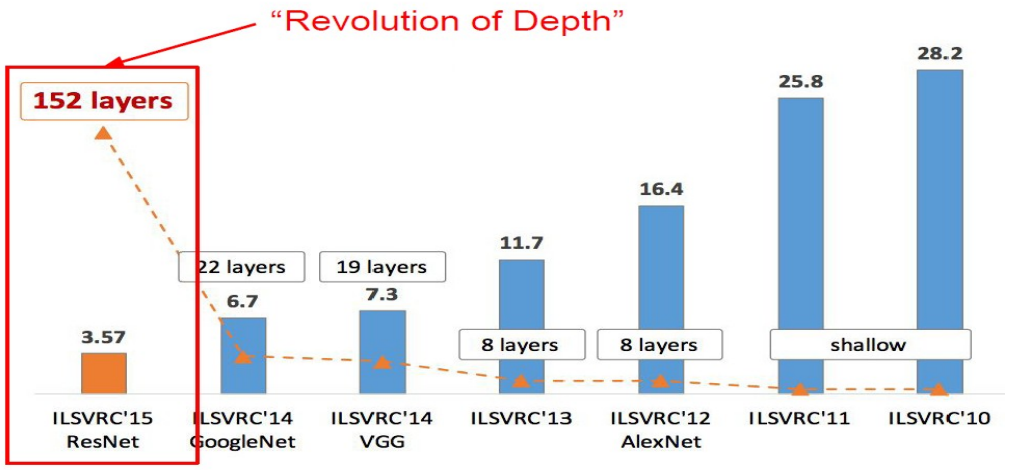
Deep Neural Networks

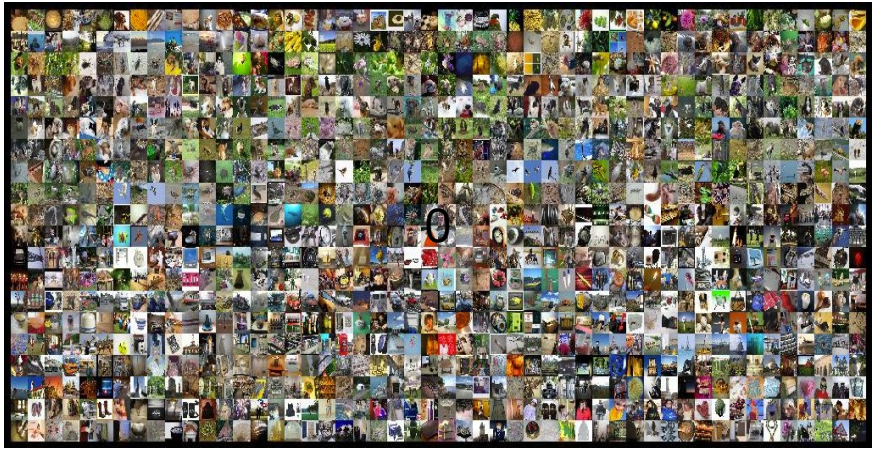


Full GoogLeNet architecture

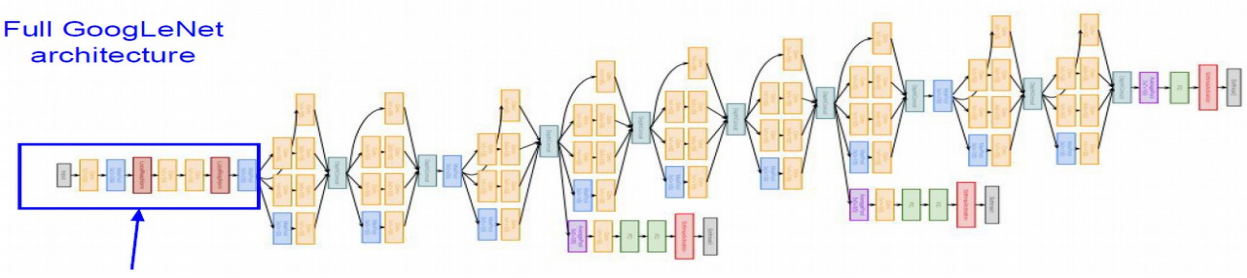


Deep Neural Networks

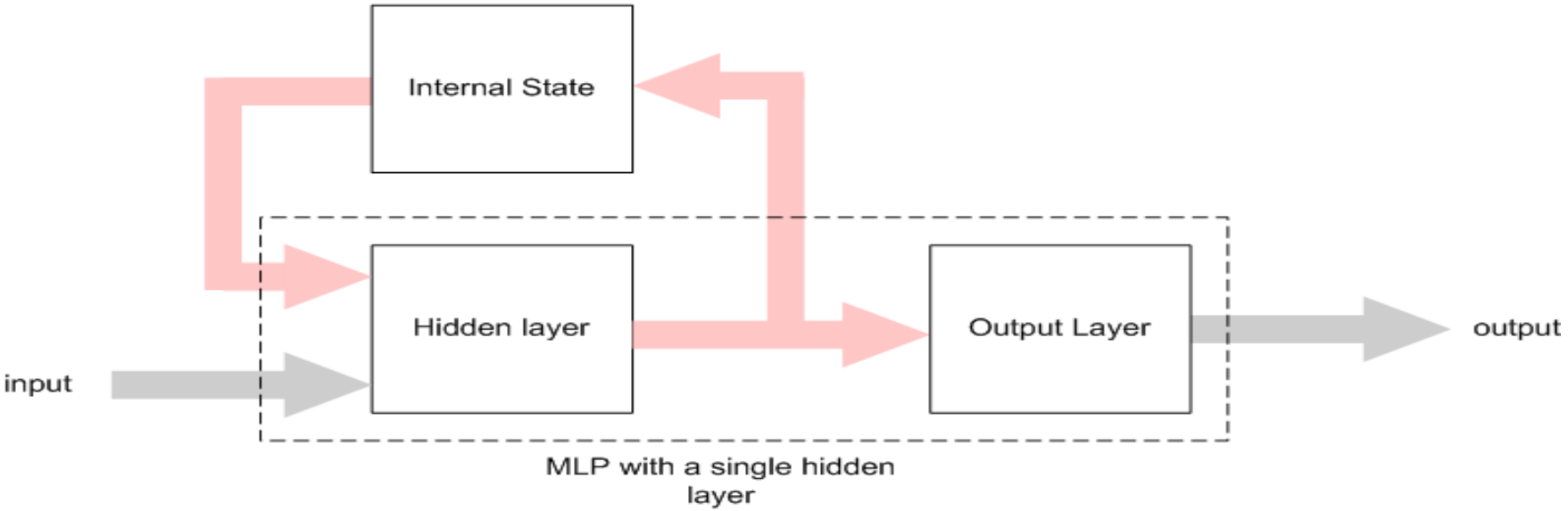




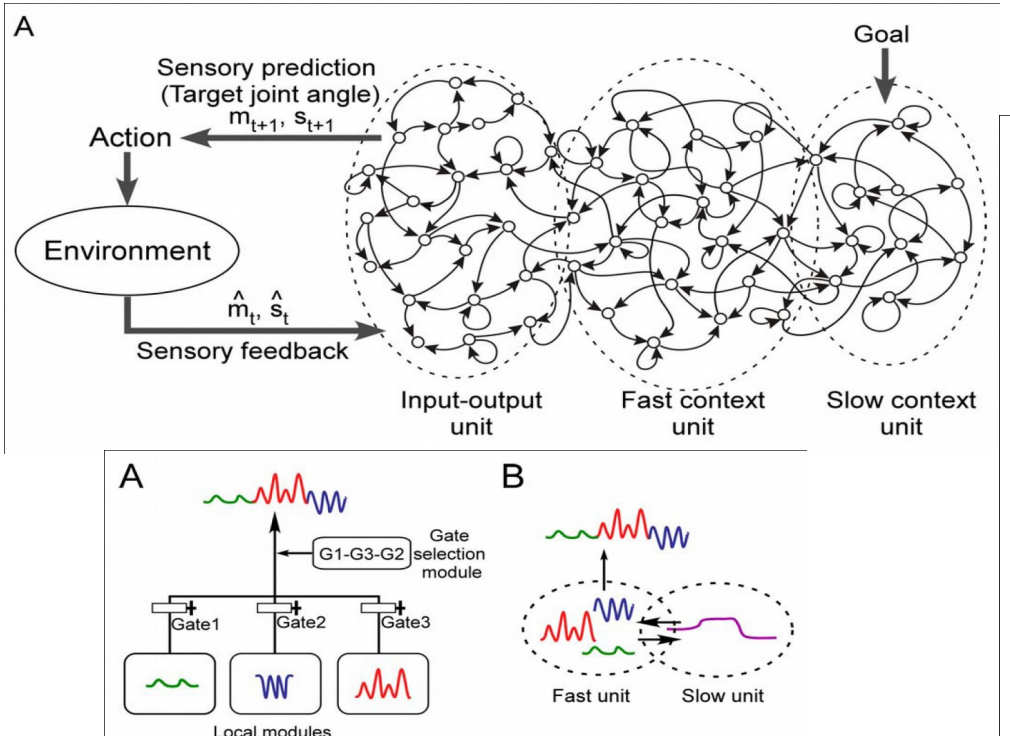
Very limited!

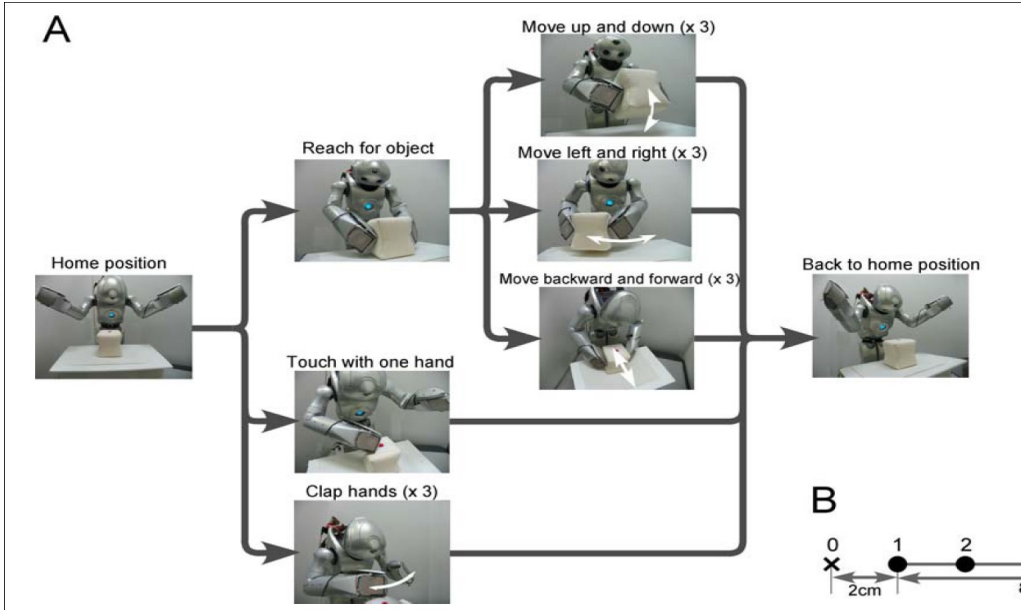


Recurrent neural networks



Recurrent neural networks





Yamashita, Yuichi, and Jun Tani. "Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment." PLoS Comput Biol 4.11 (2008): e1000220.

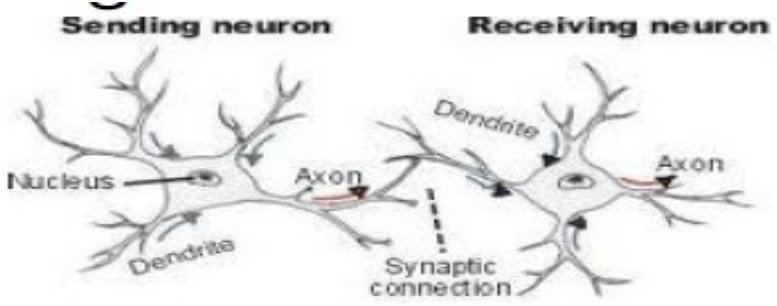
Hebb’s association principle

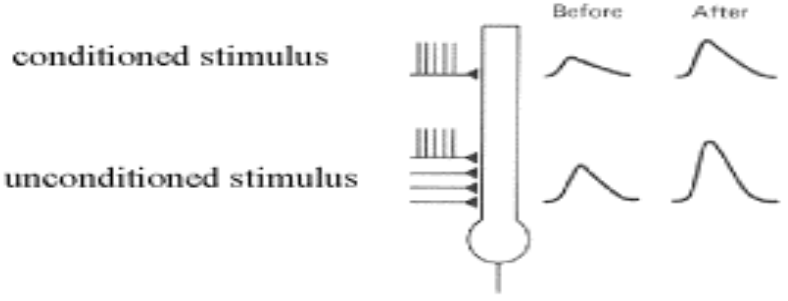
- Donald Hebb (1949) Cell A – B simultenous excitation: growth / metabolic change. Experiments (1966, 1973), confirming Hebb’s insight.
- The simple slogan to describe LTP is:

“Neurons that fire together, wire together.

Neurons that fire out of sync, fail to link.”

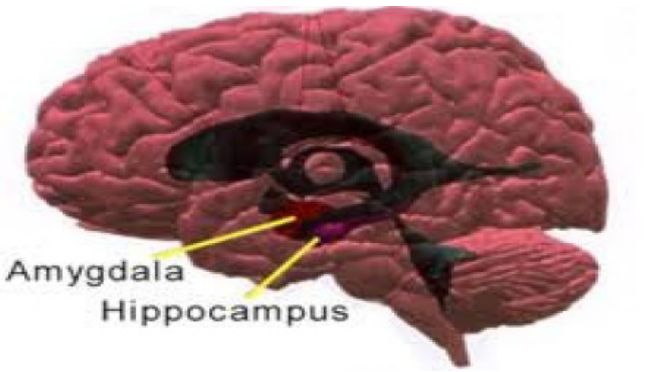
  - The neural network stores and retrieves associations, which are learned as synaptic connection.





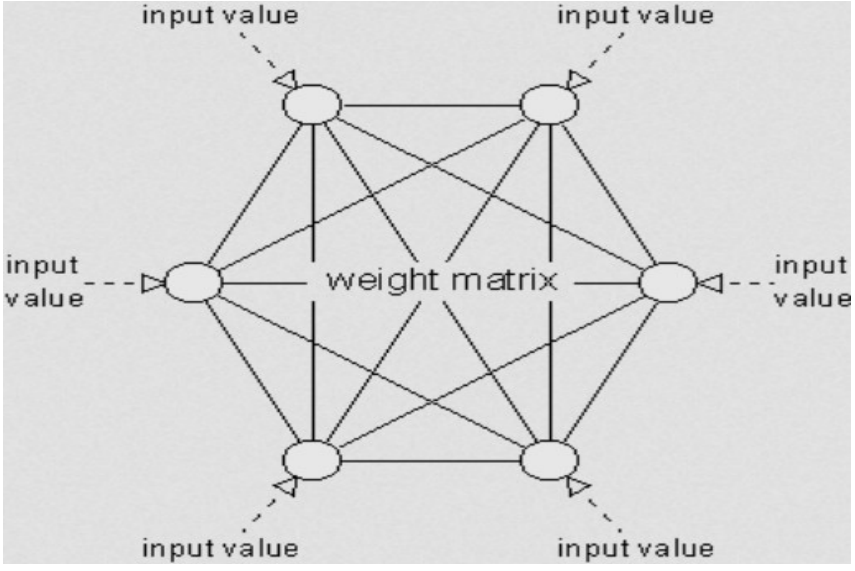
Hebb’s association principle and Human Learning

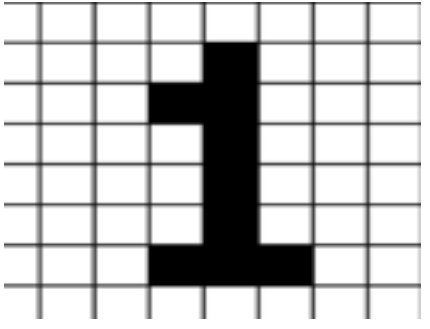
- Learning is to associate two events with each other.
- The main brain organ for learning/explicit memory is the hippocampus (of the limbic system) using Hebbian type.
- Human memory thus works in an associative or content-addressable way.



Hopfield Networks

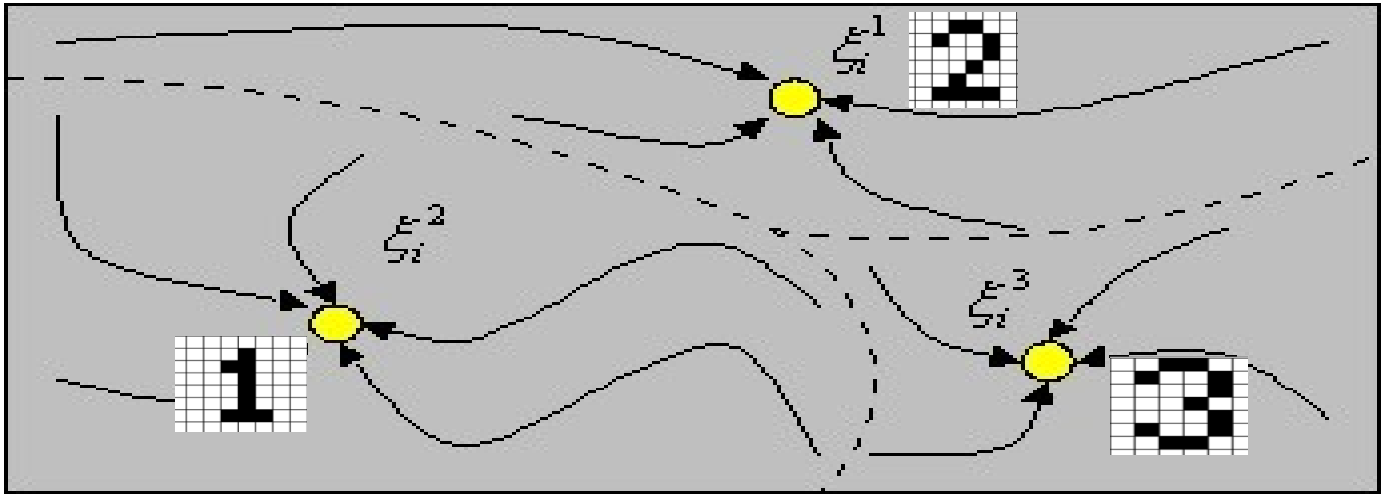
- A Hopfield Network is a model of associative memory. It is based on Hebbian learning but uses binary neurons.
- The associative memory problem is summarized as follows:
  - Store a set of p patterns P_i in such a way that when presented with a new pattern Q_i , the network responds by producing whichever one of the stored patterns most closely resembles Q_i .
- 0 or 1
- An associative memory can be thought as a set of attractors, each with its own basin of attraction.



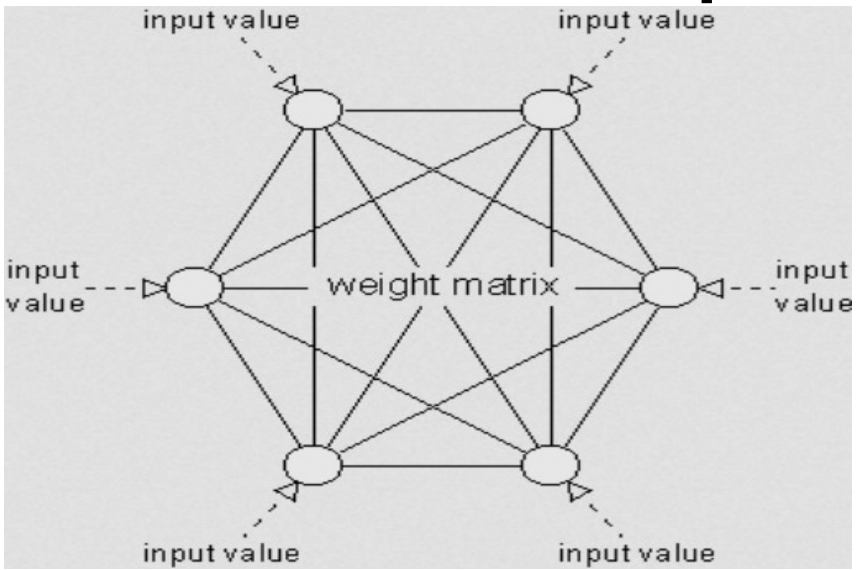


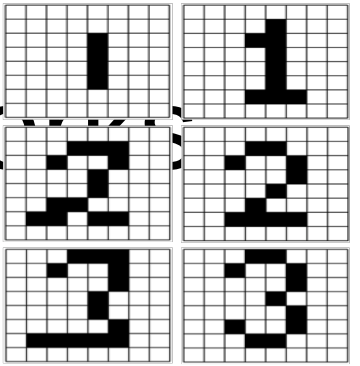
Hopfield Networks

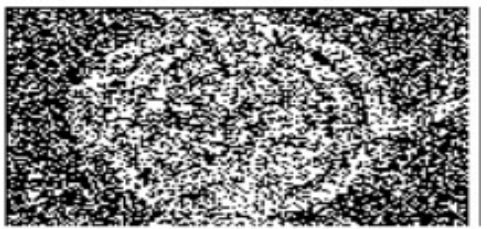
- The dynamics of the system carries a starting points into one of the attractors as shown in the next figure.

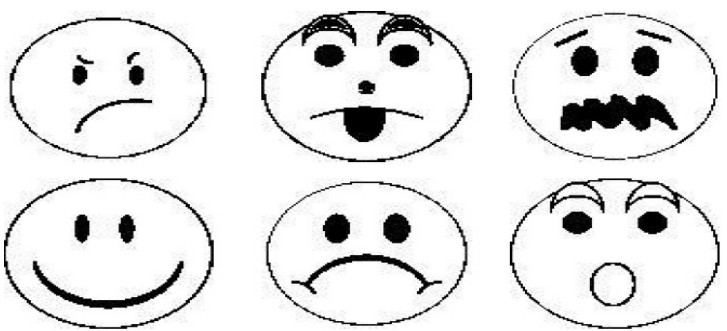


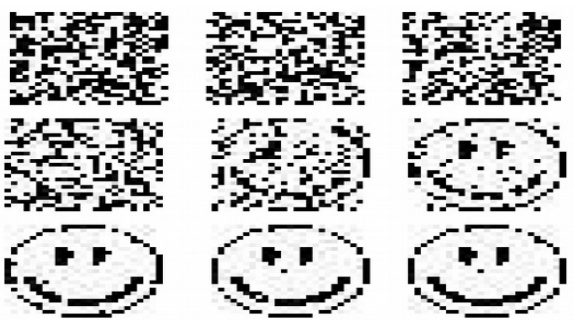
Hopfield Netwo







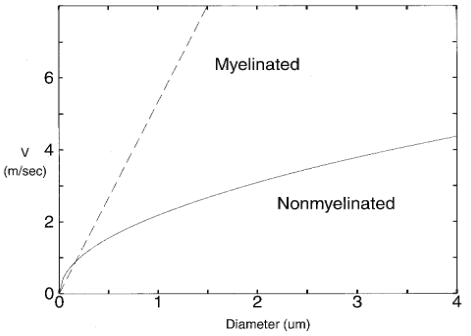


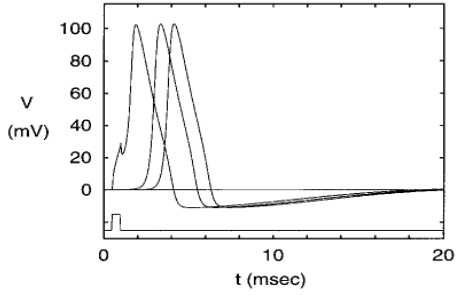


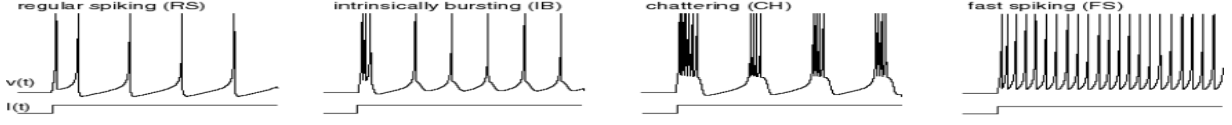
www.doc.ic.ac.uk/~ae/papers/Hopfield-networks-15.pdf

Spiking Neural Networks

- Increasing the level of realism
- Neuronal and synaptic state + concept of time.
- Neurons in the SNN do not fire at each propagation cycle, but rather fire only when a membrane potential reaches a specific value.
- When a neuron fires, it generates a signal which travels to other neurons which, in turn, increase/decrease their potentials.
- The current activation level (modeled as some differential equation) is normally considered to be the **neuron's state**, with incoming spikes **pushing this value higher**, and then either firing or decaying over time



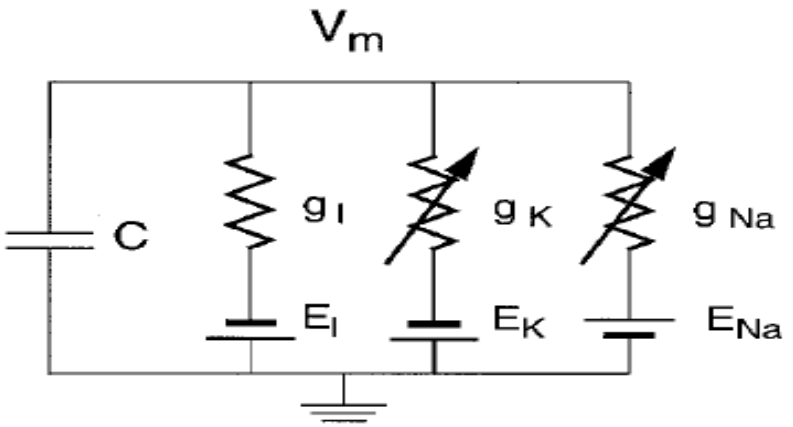




Hodgkin-Huxley Model

- Mathematical model that describes how action potentials in neurons are initiated and propagated
- Alan Lloyd Hodgkin and Andrew Fielding Huxley described the model in 1952 to explain the ionic mechanisms underlying the initiation and propagation of action potentials in the squid giant axon
- Terminology:
  - Channel: Flow of ions through membrane proteins
  - Concentration gradient: High sodium concentration outside the membrane
  - Reversal potential: Reduction of the gradient to zero
  - Electrical gradient: By sodium flow
  - Rest potential: -65mV
  - Threshold: At around -50mV, sodium channels open up

Hodgkin-Huxley Model



$$C\frac{dV_m}{dt} = g_l(E_l - V_m) + g_{\mathrm{Na}}(E_{\mathrm{Na}} - V_m) + g_{\mathrm{K}}(E_{\mathrm{K}} - V_m)$$

$$g_{\mathrm{Na}} = \bar{G}_{\mathrm{Na}} \cdot m(t)^3 h(t)$$
$$g_{\mathrm{K}} = \bar{G}_{\mathrm{K}} \cdot n(t)^4$$

$$\frac{dm}{dt} = \frac{m_\infty(V_m) - m}{\tau_m(V_m)}$$
$$\frac{dh}{dt} = \frac{h_\infty(V_m) - h}{\tau_h(V_m)}$$
$$\frac{dn}{dt} = \frac{n_\infty(V_m) - n}{\tau_n(V_m)}$$

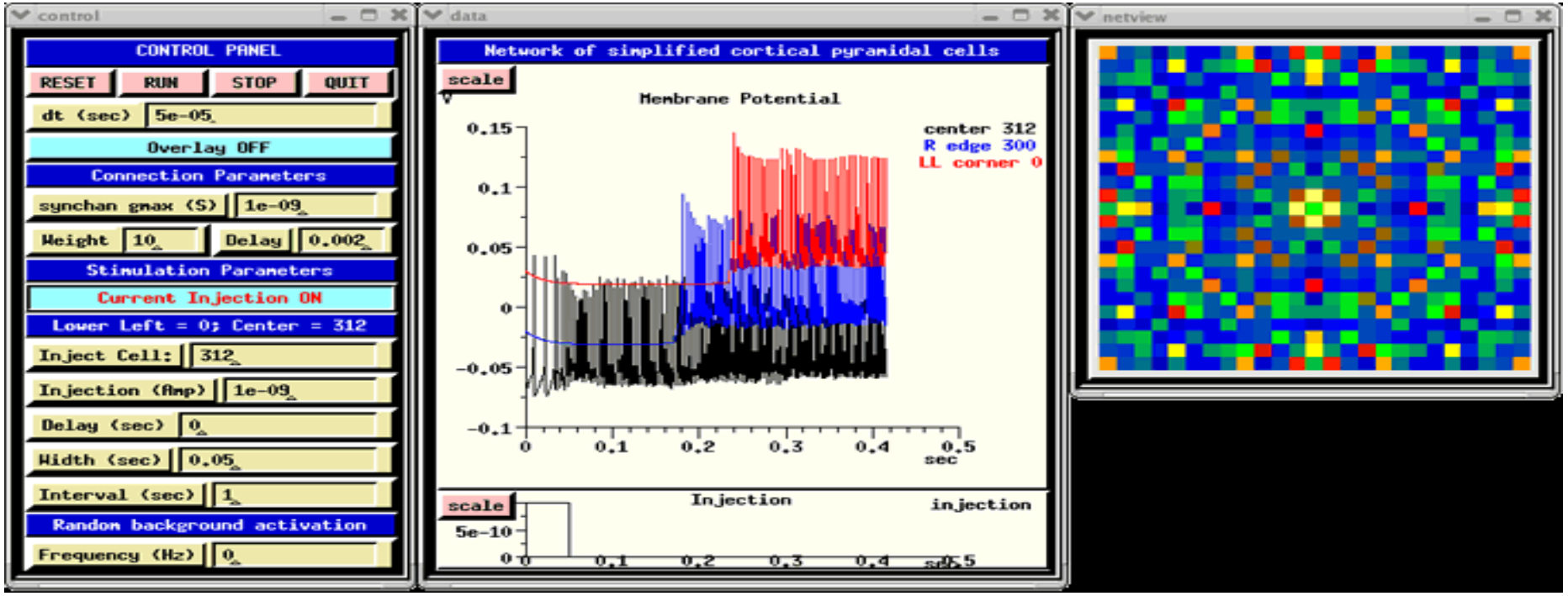
$$x_\infty = \frac{\alpha_x}{\alpha_x + \beta_x}$$
$$\tau_x = \frac{1}{\alpha_x + \beta_x}$$

$$\alpha_m = \frac{0.1(V_m - 40)}{e^{(V_m - 40)/10} - 1} \qquad \beta_m = 4e^{(V_m - 65)/18}$$
$$\alpha_h = 0.07e^{(V_m - 65)/20} \qquad \beta_h = \frac{1}{e^{(V_m - 35)/10} + 1}$$
$$\alpha_n = \frac{0.01(V_m - 55)}{e^{(V_m - 55)/10} - 1} \qquad \beta_n = 0.125e^{(V_m - 65)/80}$$

GENESIS Simulation System

- GEneral NEural SImulation System
- Models channels, cells and networks
- http://www.genesis-sim.org
- Creating a realistic model of a neuron:
  - Set the passive membrane parameters (membrane resistance and capacitance, axial resistance, and membrane resting potential for each of the compartments.
  - Populate the compartments with ionic conductances ("channels"), or other related neural elements.
  - Link compartments for the soma and dendrites together with appropriate messages to make a cell.

GENESIS Simulation System



NEURON Simulation Environment

- In *NEURON*, the neuron's geometry is described in terms of cylindrical **sections**
- Channel properties are set within sections
- You can add “cables”, and produce hierarchical structures
- Over a thousand papers published
- Hines, M. L. and Carnevale, N. T., 2001.
- http://www.neuron.yale.edu