REVIEW

Neural processor in artificial intelligence advancement

Manu Mitra

Systems Engineer/Executive – IT Operations, India

ABSTRACT

A neuron network is a computational model based on structure and functions of biological neural networks. Information that flows through the network affects the structure of the neuron network because neural network changes- or learns, in a sense-based on that input and output. Although neural network being highly complex (for example change of weights for every new data within the time frame) an experimental model of high level architecture of neural processor is proposed. Neural Processor performs all the functions that an ordinary neural network does like adaptive learning, self-organization, real time operations and fault tolerance. In this paper, analysis of neural processing is discussed and presented with experiments, graphical representation including data analysis.

***Keywords**: Neuron; neuron processor; neuron network; artificial intelligence*

ARTICLE INFO
Received: September 12, 2017
Accepted: December 18, 2017
Available online: May 14, 2018

***CORRESPONDING AUTHOR**
Manu Mitra, Electrical Engineering Department University of Bridgeport 126 Park Avenue, Bridgeport, CT – 06604, USA; manu.ieee@gmail.com; mmitra@my.bridgeport.edu

CITATION
Mitra M. Neural processor in artificial intelligence advancement. Journal of Autonomous Intelligence 2018; 1(

1. Introduction

Neural Network are set of algorithms, modeled loosely after the human brain that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, such as images, sound, text or time series *etc*.

First, a collection of software "neurons" are created and connected together, allowing them to send messages to each other. Then, the network is asked to solve a problem, which it attempts to do over and over, each time strengthening the connections that leads to success and diminishing those that leads to failure.

An average neural system has couple of dozens to thousands or even greater number of artificial neurons. These units are organized in a progression of layers, each of which interfaces with the layers on either side. Some of them, known as input units that are intended to get different types of data from the outside world. That system helps to find out about, perceive, or general process. Different units sit on the inverse side of the system and flag how it reacts to the data; those are known as yield units. In the middle of the information units and yield units are at least one layer of shrouded units, which together, frame most of the manufactured cerebrum. Most of the neural systems are completely associated which each concealed unit and each yield unit is associated with each unit in the layers either side. The associations between one unit and another are broken by a number called a weight, which can be either positive (in the event that one unit energizes another) or negative (on the off chance that one unit stifles or hinders another). The higher the weight, the more impact one unit has on another. (This relates to the way genuine cerebrum cells trigger each other crosswise over little holes called neurotransmitters).

Data processing system in the neural network consist of large number of simple, highly interconnected processing elements (artificial neurons) in an architecture stimulated by the structure of the cerebral cortex of the brain[1–3].

2. High level architecture of a neural processor

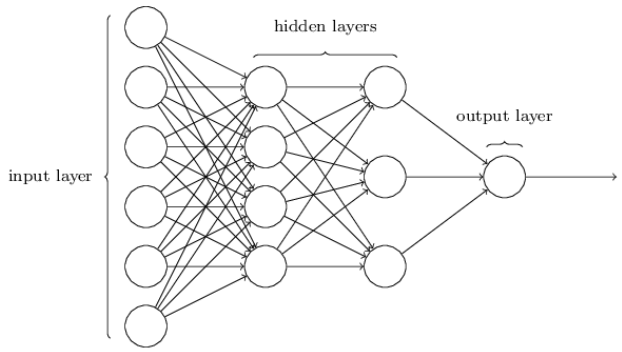


Figure 1. Depicts a traditional representation of Neural Network

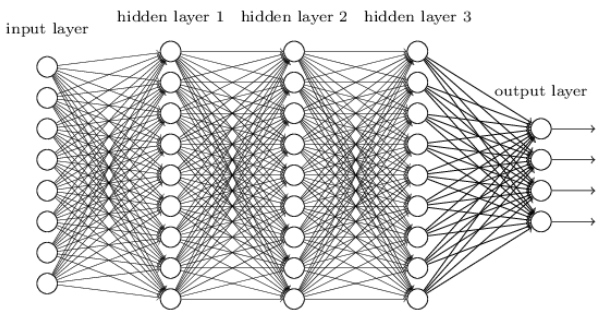


Figure 2. Depicts a complex neural network with three hidden layers, nine set of neurons and four output layers.

2.1. Operations & maintenance processor

Figure 3. Proposed high level architecture for a Neural Processor

In majority of systems, such as embedded and distributed, administrator controls the total framework model from reassure to specialized architecture. When outlining particular framework, operations and support processor ought to give view about the organization of the architecture.

Operations and Maintenance Processor system have a lot of impact on the superficial quality of the system.

Functions of an Operations and Maintenance Processor:

●It maintains method based screens interface to the operator to control the system.

●Offers statistics screens to determine the health of the system

●Aids in interface to notify the operator of serious problems that need operator intervention.

●Assists in Periodic and on-demand reports about the health and performance of the system

●Support configuring and dimensioning of the system

2.2. Central manager

In Central Manager, configuration is utilized to characterize an essential processor that deals with the operations of entire framework. Central Manager keeps up framework level perspective course of action. Normally, the Central Manager interacts with Element Processor (as shown in the **Figure 4**).

Functions of a Central Manager:

●It sustains health of the system by checking the sanity of highest level segments. Global fault handling is implemented at this level.

●Assigns resources that are shared at global level.

●Communicates the system startup and other activities that involve system wide communications

●Connects with Operations & Maintenance processor for operator interface for system wide commands/statistics.

●Handles the network with highest level segment managers

2.3. Element processor

Element can be defined as collection of different processors and or many micro controllers that implement vital part of the system's functionality. Element Manager is a processor in the system that manages all other processors belonging to that specific segment. It can further manage sub-modules. Sub-modules further manage their own processor.

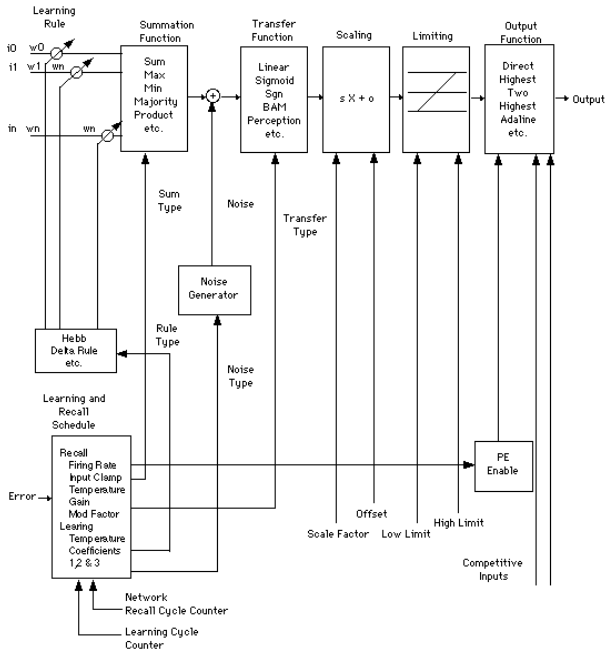


Figure 4. Function of each single element in the neuron processor[5]

Element Processor has been discussed in detail in **Figure 4**.

Functions of an Element Processor:

●Manages commands and instruction requests from the Central Manager (valid to highest level Element Managers)

●Manages commands and requests from a higher level Element Manager (valid to sub-module managers)

●Maximum level Element Managers participate in the global network interfacing them with the Central Manager.

●Element level fault handling is carried out by the Element Manager. Intra-module faults are not reported to the Central Manager.

●It controls the state and startup of Terminal Managers and other processors within the segment.

●Executes most of the transaction processing. For example, in a switching system, most of the call processing will be handled.

●It also synchronizes transaction processing actions within the element.

●Element Manager checks the local intra-module network.

●Reins all Operations & Maintenance processor requests for operator initiated instructions.

2.4. Terminal manager

Terminal Manager is an equipment board that backings number of units of similar sets. For instance, a computerized trunk administrator card may deal with numerous advanced trunk circuits. The Terminal Manager has solitary microchip that handles every one of the component. Hence Element Manager distributes out individual terminals. The fault handler monitors health of the Terminal Manager. If at some point when a Terminal Manager comes up short, the fault handling takes care of fragment of the product as far as individual terminal fails.

Function of Terminal Manager:

●It aids in partition of hardware and software resources on the board into terminals.

●Manages terminal configuration and de-configuration requests from the controlling Element Manager.

●It also manages operator status change requests from the operator. (The Terminal Manager does not directly connects with the Operations & Maintenance processor; it receives the requests from the controlling Element Manager).

●Observes the status of individual devices implementing the terminals. Terminal Manager also exchanges health messages with micro-controller and Digital Signal Processing (DSP) controlled terminals.

●It conceals the rest of the system from exact hardware implementation of terminals (*e.g.* the Terminal Manager might consist of eight DSPs implementing four terminals per DSP.)

2.5. Device controller

Device Controller is a processor and/or micro controller which interacts with the lowest level of hardware. Instances of Device Controllers are:

●Digital Signal Processing implementing terminals in a Terminal Manager

●A microprocessor and or micro controller controlling a transducer, heat sensor *etc*.

●Micro-controller operating a stepper motor

A Device Controller can connect with other processors with any of the below interfaces:

●First In First Out (FIFO)

●Dual Ported Random Access Memory (RAM)

●Serial link

●Ethernet

●Direct Memory Access (DMA) Operations

Function of a Device Controller:

●It offers a message level device programming connection to higher layer software

●Manages interrupts and device communications, thus offloading the main processor from device level processing

Retains other processors clear to the actual device operation[4].

3. High level architecture of an element processor in a neural processor

3.1. Major components of neural network element

Element Processor is one of the major module in the Neuron Processor that is indicated in the **Figure 3**. It plays vital role in neuron processing system and because of the complexity and too many variables below are few of many components.

Component 1. Weighting Factors: A neuron regularly gets various simultaneous data sources. Each information has its own particular relative weight which gives the information that impacts on component's summation capacities. These weights play an important role such as changing synaptic qualities of natural neurons. In a case, however, a few information sources are made remarkable than others with the goal that they affect the handling component as they merge to create a neural reaction.

Weights are versatile coefficients inside the system that decide the centralization of information motion as recorded by the artificial neuron. They measure information's association quality. These qualities can be adjusted for a few preparing sets and as indicated by a system's express topology or through its learning rules.

These weights can be seen in the trials performed and they continue changing for different time allotment.

Component 2. Summation Function: The first step in a processing element's process is to calculate the weighted sum of all of the inputs. Scientifically, the inputs and the corresponding weights are vectors which can be represented as (X_1, X_2 . . . X_n) and (Y_1, Y_2 . . . Y_n). The total input signal is the dot, or inner, product of these two vectors. This basic summation function is found by proliferating each component of the X vector by the corresponding component of the w vector and then totaling up all the products. Input1 = X_1 * Y_1, input2 = X_2 * Y_2, *etc*., are added as input1 + input2 + . . . + input n. The outcome is a single number, not a multi-element vector.

The summation capacity can be more composite than recently input information and weight whole of items. The information and weighting coefficients can be merged in a few diverse routes on to the exchange work. Nevertheless a basic item summing, the summation capacity can choose most extreme, dominant part, or various normalizing calculations. The unequivocal calculation for consolidating neural sources of information is controlled by the engineering and model.

Some summation capacities have an extra procedure to result before it is conveyed on to the exchange work. This procedure is additionally called the initiation work. The reason of utilizing an enactment work is to permit the summation yield to change time allotment. Initiation works are basically limited to inquire about. A large portion of the present system applications utilize a “personality” actuation work, which is same to not having one. Also, such capacity is conceivable to be a part of the system in general as opposed to every individual handling component segment.

Component 3. Transfer Function: The result of the summation work, quite often weighted total, is transmuted to working yield through algorithmic procedure known as the exchange work. In this exchange work, summation aggregate can be coordinated to choose the neural yield. In the event that the sum is more important than the edge, the preparing component creates a flag. In this event whole information and weight items are not as much as the edge, no flag (or some inhibitory flag) is created. The two sorts of reaction are vital.

Component 4. Scaling and Limiting: After the processing element's transfer function, the outcome can transit through extra processes called scale and limit. This scaling simply proliferates a scale factor times the transfer value, and then adds an offset. Limiting is the mechanism which insures that the scaled result does not exceed an upper or lower bound. This limiting is addition to hard limits that the original transfer function may have accomplished.

Component 5. Output Function: Each processing element is allowed one output signal which it may output to hundreds of other neurons. This is just like biological neuron, where there are many inputs and only one output action. Normally, output is directly equivalent to the transfer function's end result. Some network topologies, however, adjust the transfer outcome to incorporate trace among next processing elements. Neurons are allowed to participate with

each other, inhibiting processing elements unless they have great strength. Traces can occur at one or both of two levels. First, competition determines which artificial neuron will be active, or offers an output. Second, traces inputs help determine which processing element will participate in the learning or adaptation process.

Component 6. Error Function and Back-Propagated Value: In most learning networks the variance between the current outcome and the expected outcome is planned. This raw error is then transmuted by the error function to match a certain network architecture. The simplest architectures use this error directly, but some square the error while holding its sign, some cube the error, and other paradigms modify the raw error to fit their specific determinations. The artificial neuron's error is then typically propagated into the learning function of another processing element. This error term is occasionally called the current error.

Component 7. Learning Function: The reason of having learning capacity is to adjust the variable associated weights on the contributions of each preparing component rendering to some neural based calculation. This procedure of changing the weights that are associated with some foreseen result can be known as the adaption work, and additionally the learning mode. There are two sorts of learning: directed and unsupervised. Regulated learning requires a guide. The coach might be a preparation set of information or an onlooker who reviews the execution of the system results. In any case, having a guide is learning by support. At the point when there is no outside guide, the framework oversees itself by some inside criteria composed into the system. This is called learning by doing[5].

All these components are integrated into one system called element and this can be called as Element processor in the neural processor (**Figure 5**).

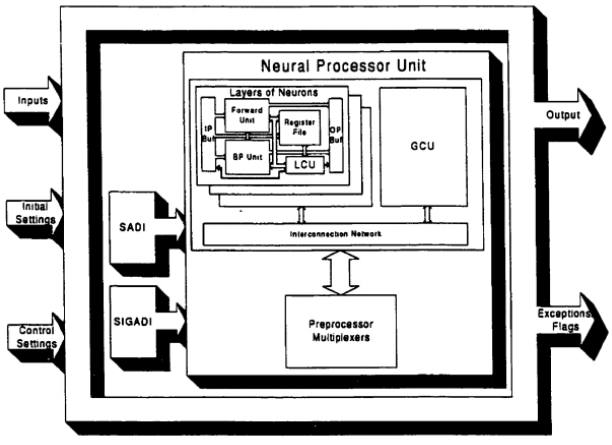


Figure 5. A Block Diagram of a neuron processor[6]

A typical block diagram of a neuron processor can look like above figure. Author would like to Thank Balamurugan Balasubramanian for the image.

4. Neural processing experiments (Part I)

Below is the experimental model of cat and mouse chase to represent if a neural processor is implemented.

There are three windows (from left) First window shows mouse and cat status and its fitness. Second window (background) is the graphical representation on how mouse gets the cheese and cat gets the mouse. Third window shows the graph generated for every iteration and each iteration last for 30 sec.

Three attempt were taken, each attempt consist of ten iteration and each iteration lasted for 30 sec[7]. These graph depicts the change in the values for the output for the for cat and mouse fitness (**Figure 6**).

5. Neural processing experiments (Part II)

As per reviewer remarks, author agrees that Neural processing experiments are performed on neural networks i.e. one of the major part of the neural processor but not the output of comprehensive neuron processor.

5.1. High Dimensional Projection

In High dimensional Projection there are sets of neurons of different weights and weights changes for change of time frame.

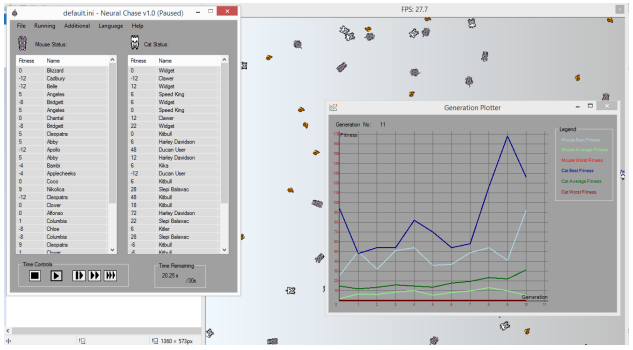
There are two windows 1) Attractor Network which contains set of neural network of different weights and 2) Projection of Attractor Network which has Projection X and Projection Y and graph changes based on the data points and the time frame for that network while considering weights of neurons changes (**Figure 7** and **Figure 8**)[8].

5.2. Actor critic model

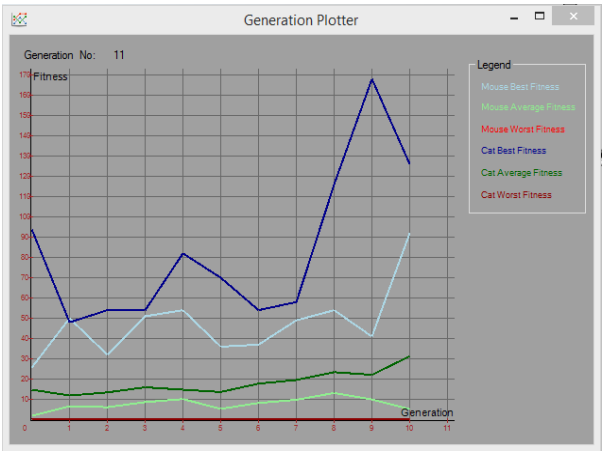
In Actor Critic Model there are parallel neurons with each of them connected to each other and one output.

There are five windows as shown in the below Figure 1) Information 2) Simulations 3) TD Network 4) Odor Word 5) TD, Value, Reward Window (**Figures 9, 10 and 11**).

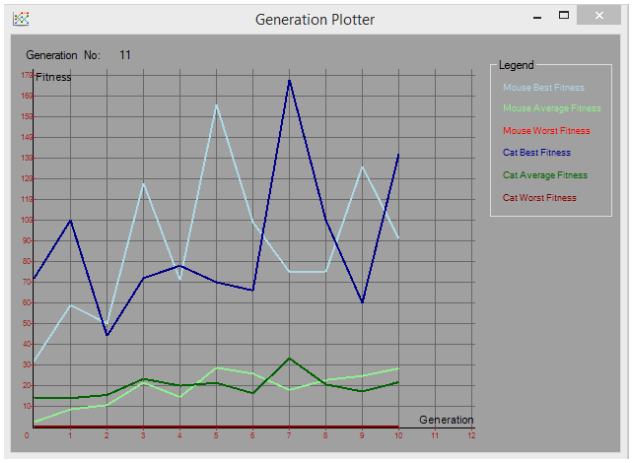
(A)



(B)



(C)



(D)

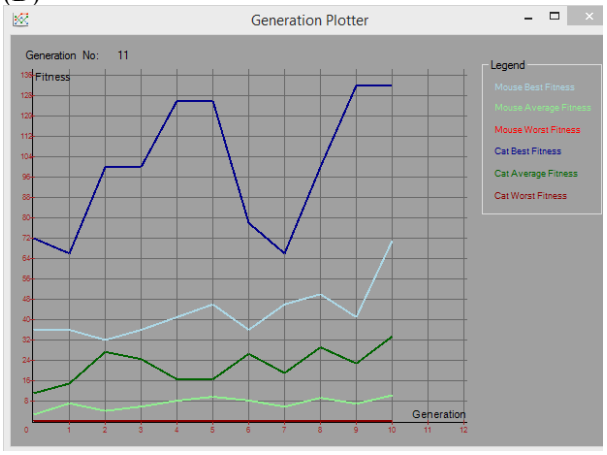


Figure 6. (**A**) shows three windows. First window (from left) mouse and cat status. Second window (background) is the graphical representation on how mouse gets the cheese and how cart gets the mouse. Third window shows the graphs generated. (**B**) Attempt I. (**C**) Attempt II. (**D**) Attempt III.

5.3. Sorn Model

In Sorn model there are various neurons and the flashing neuron (yellow color) shows the change of the weight.

Once the simulation was executed and as the time frame changes the neuron weight changes (shown as flashing yellow color of the neuron). Two simulations were taken, one was at time frame of 10 ms and other for 20 ms.

The use of this model explains that if there are large number of neurons then the time required to complete a task will be significantly low compared to other models (**Figure 12**).

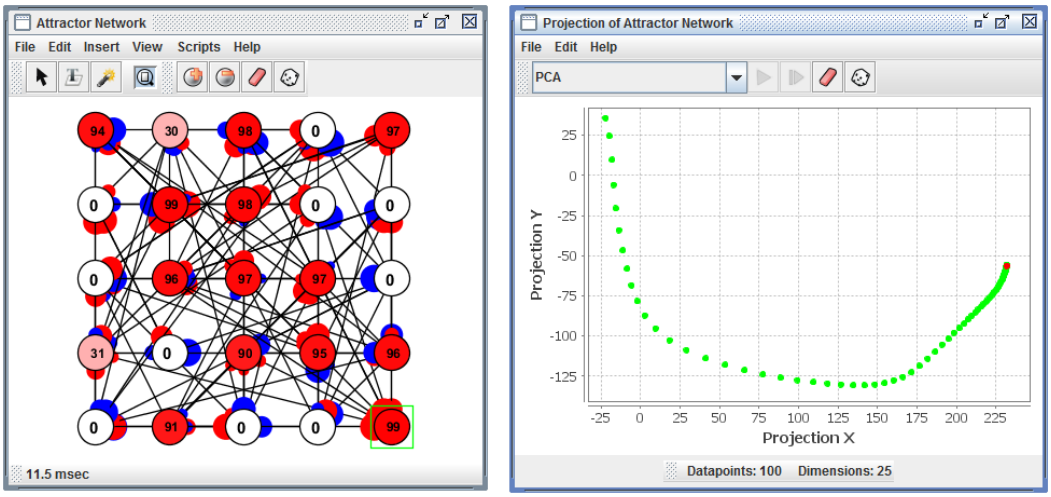
6. Results of the experiments

Based on the experiments performed a Neural Processor can be constructed and proposed high level architecture is shown. Simply to put that various iterations are required for the neuron network until the network learns by itself within the time frame to achieve the results. For example in the Actor Critic Model shows the number of iterations required for the mouse to get the cheese within the timeframe (noise not included) can be one of the example how neuron processor works for robotic models.

7. Applications, advantages and disadvantages

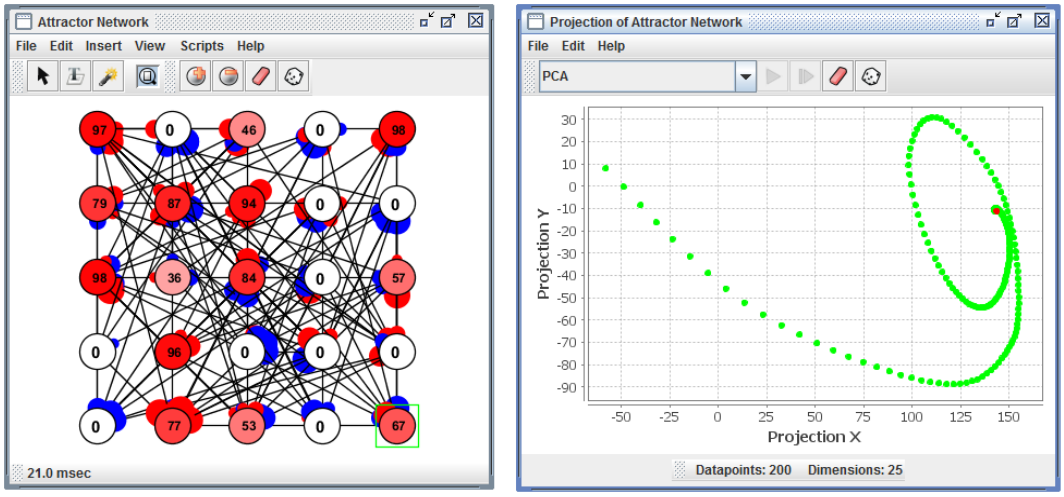
7.1. Applications

There are many applications of a neuron processor and most important application as per



(**A**) (**B**)

Figure 7. (**A**) shows the sets of neuron connected and 7(**B**) shows the plotted graph for 100 Data points Graph is plotted on 100 data points with the time frame of 11.5 msec



(**A**) (**B**)

Figure 8. (**A**) shows the sets of neuron connected and 8(**B**) shows the plotted graph for 200 Data points Graph is plotted on 200 data points with the time frame of 21.0 msec

author point of view is that it can be used in robotic systems.

The following are the sample list of applications to start with but there are many other useful tasks that neuron processor can perform that are not listed here [9].

1) Can be used in Intelligence System of a Robot acting as a major brain system of a robot system

2) Because of self-learning capability can be used in noise reduction in digital systems

3) Usage in Digital Signal Multiprocessor, Digital Image Multiprocessor, Video Signal Processing, Speech recognition for better results

4) Machine learning applications

5) Visual processing systems

6) Hardware and Software systems

7.2. Advantages

One of the major applications of a neuron processor is that once successfully implemented in robotics system; Robots can perform tasks much more accurately, precisely and not to mention that by using artificial intelligence such as neuron processor; robots can think for themselves and solve problems that occur in our day to day life.

7) Solving problems by robots that are more accurate and precise than human

8) Improved Artificial Intelligence in Robotic systems

9) Adaptive learning, self-organization, real time

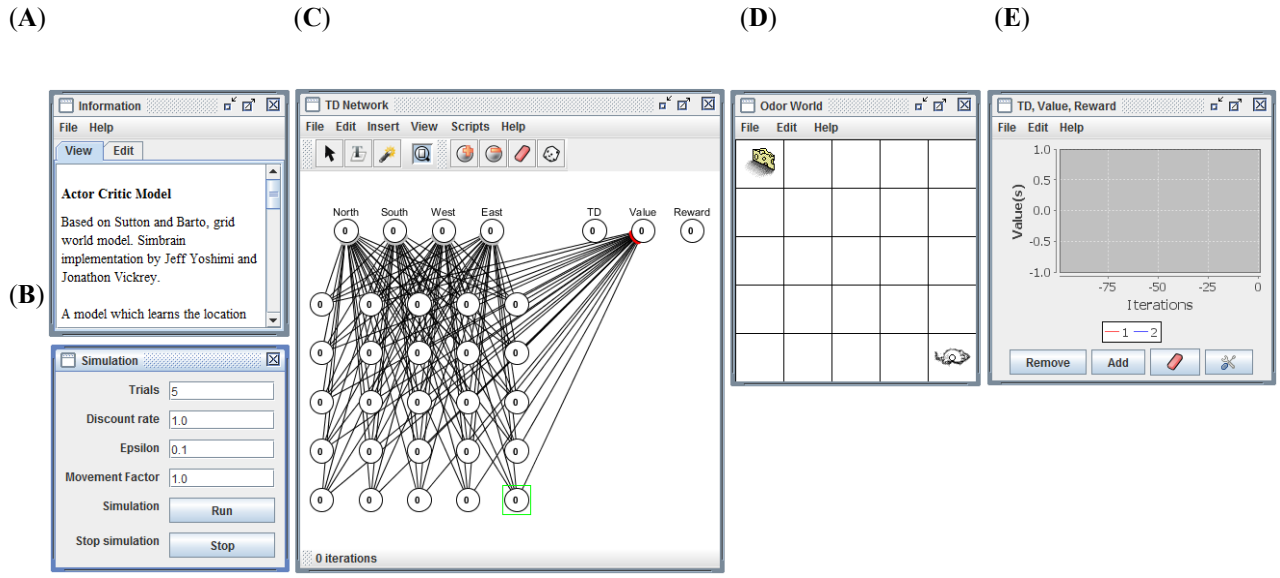


Figure 9. (**A**) Information. (**B**) Simulation window. (**C**) The various neuron connected to get output value. (**D**) Mouse to get the cheese in how many iterations for the neuron network to achieve it. (**E**) The graph for TD Network, Value and Reward.

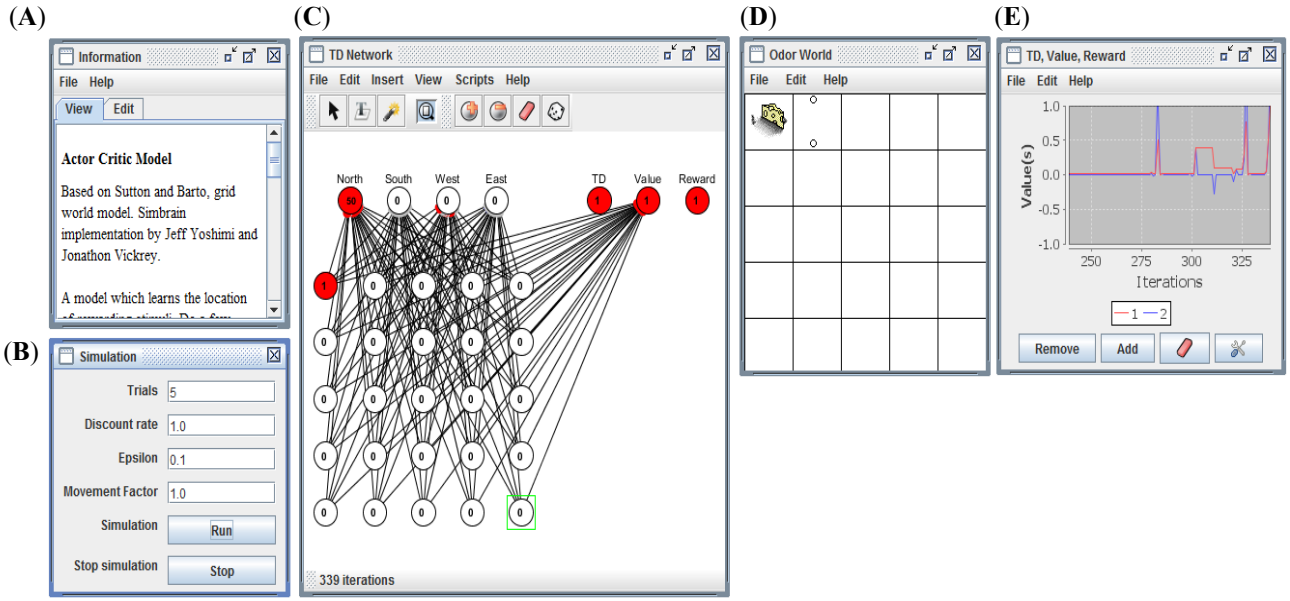


Figure 10. First Attempt (**A**) Information. (**B**) Simulation window. (**C**) The various neuron connected and the weights for the neurons have been changed for five trails making 339 iteration for the time frame of 339. (**D**) Mouse to get the cheese in how many iterations for the neuron network to achieve it. (**E**) The updated graph for TD Network, Value and Reward

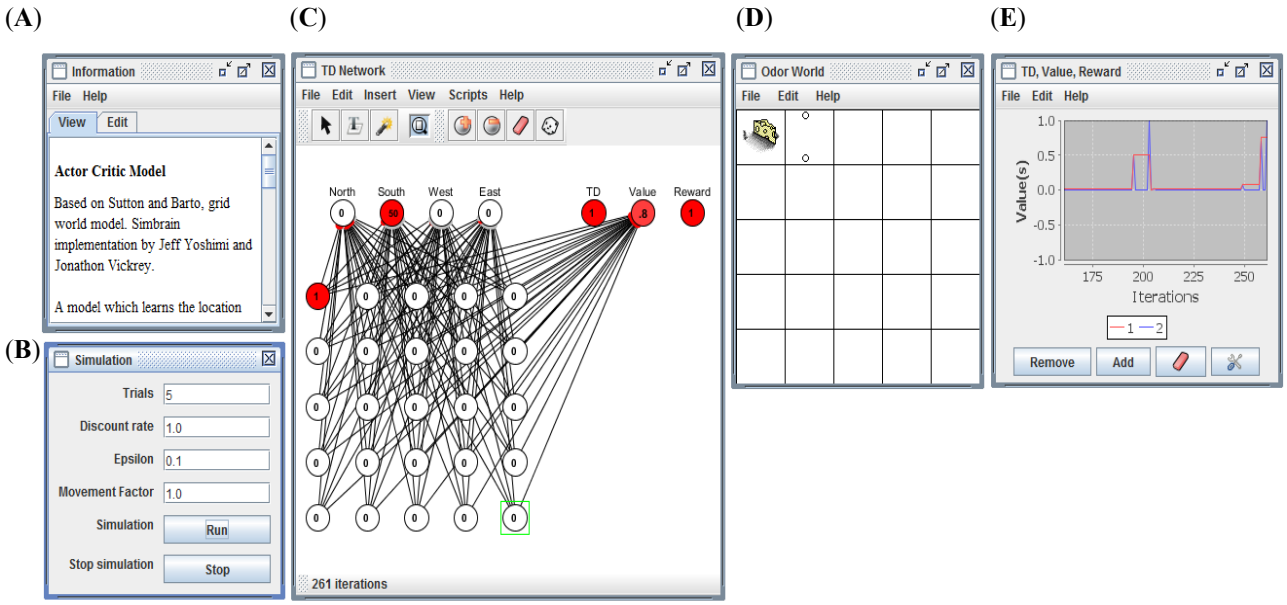


Figure 11. Second Attempt (**A**) Information. (**B**) Simulation window. (**C**) The various neuron connected and the weights for the neurons have been changed for five trails making 261 iteration for the time frame of 261. (**D**) Mouse to get the cheese in how many iterations for the neuron network to achieve it. (**E**) The updated graph for TD Network, Value and Reward

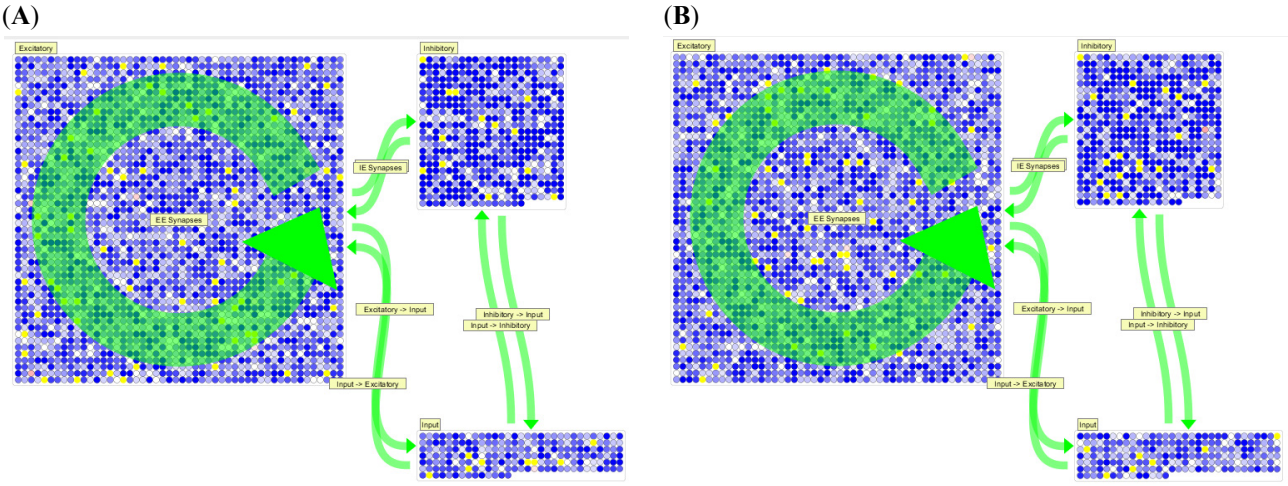


Figure 12. (**A**) shows sorn model where there are neurons that trigger with elapse of time (time taken for 10 ms). (**B**) shows sorn model where there are neurons that trigger with elapse of time (time taken for 20 ms)

operation, fault tolerance

10) Industrial applications such as process controls

11) Software applications such as Data validations

12) Risk Management

7.3. Disadvantages

Below are some of the disadvantages.

13) Robots can think for themselves causing discrepancies

14) Expensive technology

15) Complicate design

16) Initial design cost is very high.

17) Limited external control mechanisms.

18) Misuse of the technology

8. Data analysis

Below are the data analysis that are performed[10].

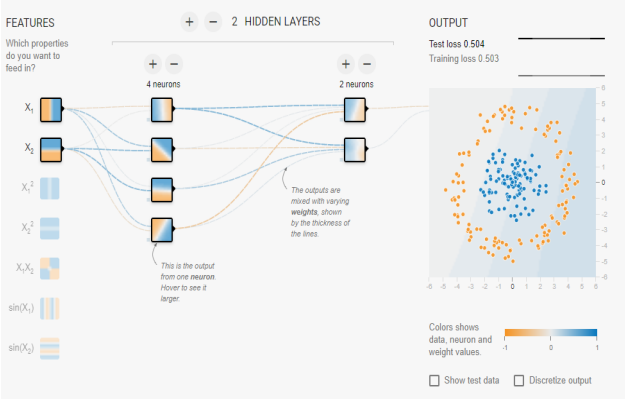


Figure 13. (**A**) depicts data analysis on two input (X_1, X_2) neurons four neurons, two hidden layers.

Epoch: 0
Ratio of Training to test data: 50%
Noise: 0
Batch size: 10
Learning Rate: 0.03
Problem Type: Classification
Activation: Tanh
Regularization: None
Output
Test Loss: 0.504
Training Loss: 0503

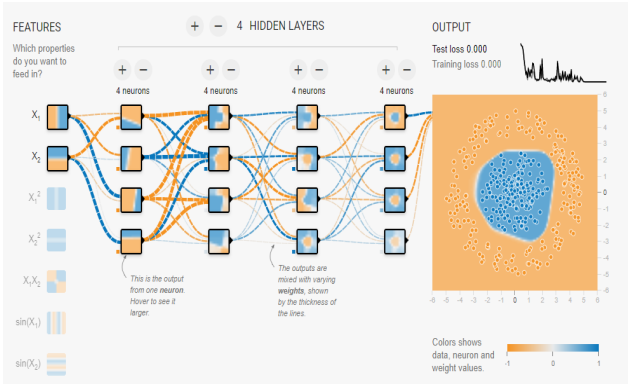


Figure 13. (**B**) depicts data analysis on two input (X_1, X_2) neurons four sets of neurons, four hidden layers.

Epoch: 100
Ratio of Training to test data: 80%
Noise: 0
Batch size: 10
Learning Rate: 0.3
Problem Type: Classification
Activation: Tanh
Regularization: None
Output
Test Loss: 0.000
Training Loss: 0000

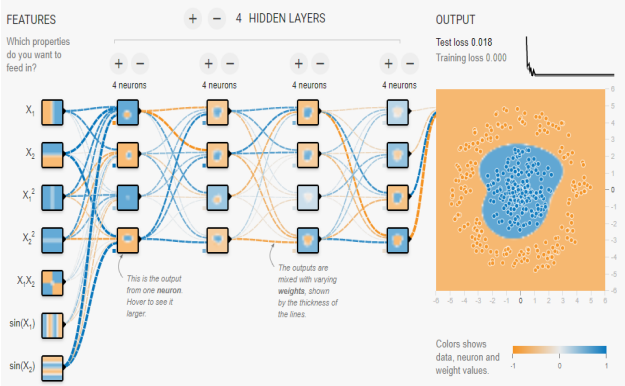


Figure 13. (**C**) depicts data analysis on seven inputs (X_1, X_2, X_1^2, X_2^2, X_1X_2, sin(X_1), Sin(X_2) neurons four sets of neurons, four hidden layers.

Epoch: 100
Ratio of Training to test data: 80%
Noise: 0
Batch size: 10
Learning Rate: 0.3
Problem Type: Classification
Activation: Tanh
Regularization: None
Output
Test Loss: 0.018
Training Loss: 0000

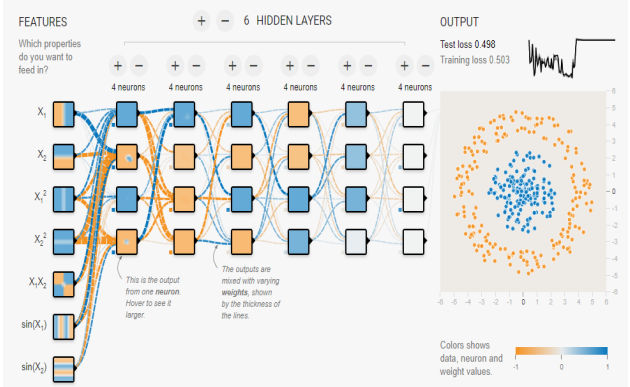


Figure 13. (**D**) depicts data analysis on seven inputs (X_1, X_2, X_1^2, X_2^2, X_1X_2, sin(X_1), Sin(X_2) neurons four sets of neurons, six hidden layers.

Epoch: 100
Ratio of Training to test data: 80%
Noise: 0
Batch size: 10
Learning Rate: 0.3
Problem Type: Classification
Activation: Tanh
Regularization: None
Output
Test Loss: 0.498
Training Loss: 0.503

Figure 13. (**E**) depicts data analysis on seven inputs (X_1, X_2, X_1^2, X_2^2, X_1X_2, sin(X_1), Sin(X_2) neurons eight sets (max that tool can effort) of neurons, six hidden layers.

Epoch: 10
Ratio of Training to test data: 80%
Noise: 0
Batch size: 10
Learning Rate: 0.3
Problem Type: Classification
Activation: Tanh
Regularization: None
Output
Test Loss: 0.157
Training Loss: 0.191

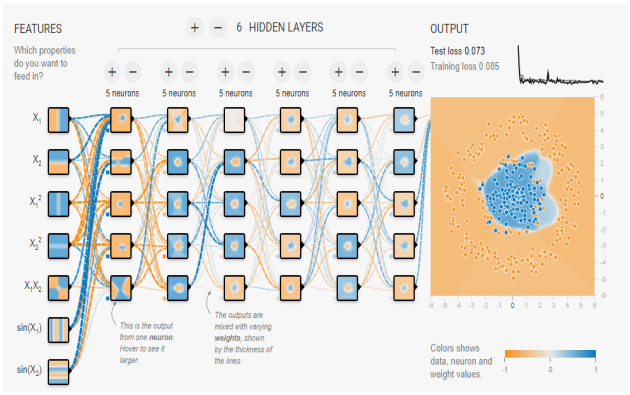


Figure 13. (**F**) depicts data analysis on seven inputs (X_1, X_2, X_1^2, X_2^2, X_1X_2, sin(X_1), Sin(X_2) neurons five sets of neurons, six hidden layers with noise

Epoch: 100
Ratio of Training to test data: 80%
Noise: 20
Batch size: 10
Learning Rate: 0.3
Problem Type: Classification
Activation: Tanh
Regularization: None
Output
Test Loss: 0.073
Training Loss: 0.085

Figure 13. (**G**) depicts data analysis on seven inputs (X_1, X_2, X_1^2, X_2^2, X_1X_2, sin(X_1), Sin(X_2) neurons five sets of neurons, six hidden layers with noise [Different data set]

Epoch: 100
Ratio of Training to test data: 80%
Noise: 20
Batch size: 10
Learning Rate: 1
Problem Type: Classification
Activation: Sigmoid
Regularization: None
Output
Test Loss: 0.496
Training Loss: 0.521

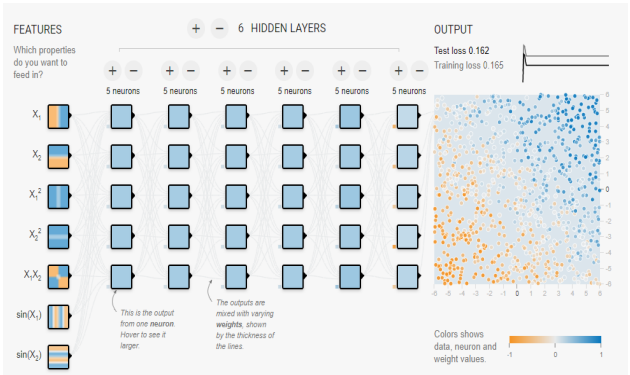


Figure 13. (**H**) depicts data analysis on seven inputs (X_1, X_2, X_1^2, X_2^2, X_1X_2, sin(X_1), Sin(X_2) neurons five sets of neurons, six hidden layers with noise [Different data set]

Epoch: 100
Ratio of Training to test data: 80%
Noise: 50
Batch size: 10
Learning Rate: 1
Problem Type: Regression
Activation: Sigmoid
Regularization: L1
Regularization Rate: 0.001
Output
Test Loss: 0.162
Training Loss: 0.165

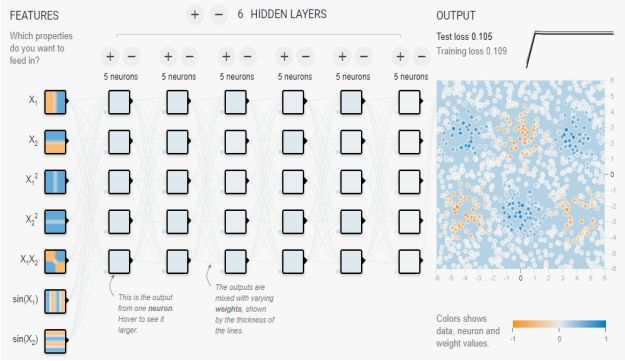


Figure 13. (**I**) depicts data analysis on seven inputs (X_1, X_2, X_1^2, X_2^2, X_1X_2, sin(X_1), Sin(X_2) neurons five sets of neurons, six hidden layers with noise [Different data set]

Epoch: 10
Ratio of Training to test data: 80%
Noise: 10
Batch size: 10
Learning Rate: 1
Problem Type: Regression
Activation: ReLU
Regularization: L2
Regularization Rate: 0.3
Output
Test Loss: 0.105
Training Loss: 0.109

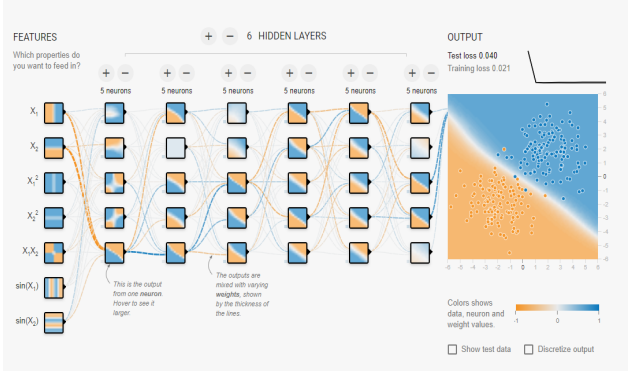


Figure 13. (**J**) depicts data analysis on seven inputs (X_1, X_2, X_1^2, X_2^2, X_1X_2, sin(X_1), Sin(X_2) neurons five sets of neurons, six hidden layers with noise [Different data set]

Epoch: 10
Ratio of Training to test data: 50%
Noise: 20
Batch size: 10
Learning Rate: 0.3
Problem Type: Classification
Activation: Linear
Regularization: L1
Regularization Rate: 0.003
Output
Test Loss: 0.040
Training Loss: 0.021

8.1. Graphical representation of the Data analysis

Notes: Orange and blue are utilized all through the perception in various routes, however when all is said in done orange shows negative esteems while blue shows positive esteems.

The information focuses (represented by little circles) are at first shaded orange or blue, which relate to positive one and negative one.

In the concealed layers, the lines are hued by the weights of the associations between neurons. Blue demonstrates a positive weight, which implies the system is utilizing that yield of the neuron as given. An orange line demonstrates that the system is appointing a negative weight.

In the yield layer, the specks are hued orange or blue contingent upon their unique esteems. The foundation shading demonstrates what the system is anticipating for a specific zone. The power of the shading indicates how certain that expectation is.

8.2. Interpretation of the results

Interpretation of the above analysis is based on very broad range because many variables, factors and data sets are taken into consideration. It also includes supervised and unsupervised training of the neuron network.

As per the author point of view these factors are to be taken into consideration when designing a neuron processor including ratio of training test data, noise, problem type, activation, test loss, training loss.

9. Conclusion

What is claimed in this are

1.An architecture of a neuron processor that performs supervised and unsupervised learning or adaptive learning that aids to get the output as expected.

2.Neuron Processor can be used as acting brain system in the robots.

3.When using Neuron Processor in the robotic system it not only improves the accuracy and preciseness of the work performed by the robots because of adaptive learning method but also robots can think for themselves to solve the problem.

4.Based on the experiments performed and data analysis for the neural networks a Neuron Processor can be achieved based on the events, training data.

Neuron Processor can be improved based on the complexity of the neuron network such as increasing the test data, increasing neuron sets, training data, *etc*.

Acknowledgment

Author would like to thank Prof. Navarun Gupta, Prof. Hassan Bajwa and Prof. Linfeng Zhang for their academic support. Author also thanks anonymous reviewers for their comments.

References

1. Shbier M. Introduction to artificial neuron networks. Available from: https://tmohammed.files.wordpress.com/2012/03/w1-01-introtonn.ppt.
2. Woodford C. Introduction to neural networks. Available from: http://www.explainthatstuff.com/introduction-to-neural-networks.html.
3. Introduction to deep neural networks. Available from: https://deeplearning4j.org/neuralnet-overview.
4. Processor Architecture Patterns. Available from: http://www.eventhelix.com/RealtimeMantra/Patterns/processor_architecture_patterns.htm#.Wa7v8rJ97Dd.
5. Artificial neural networks technology. Available from: http://irrigation.rid.go.th/rid15/ppn/Knowledge/Artificial%20Neuron%20Networks%20Technology/4.0%20Neural%20Networks%20Components.htm
6. Block Diagram Image Credit. Available from: http://research.library.mun.ca/9042/1/Balasubramanian_Balamurugan.pdf.
7. Coss J. Neural chase. Available from: https://sourceforge.net/projects/neuralchase/.
8. Simbrain simulator for neuron processing. Available from: http://simbrain.net/index.html.
9. Danese G, Leporati F, Ramat S. A parallel neural processor for real-time applications. Available from: http://brahms.di.uminho.pt/discip/MInf/ac0203/ICCA03/ParallNeuProc.pdf.
10. Tensor flow for neurons. Available from: http://playground.tensorflow.org.