American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-7, Issue-5, pp-152-161
www.ajer.org
Research Paper Open Access

Review Of Artificial Neural Network: An Emerging Problem Solving Tool In Computing

Okereke Eze Aru, Michael Okpara
Department of Computer Engineering,
University of Agriculture, Umudike, Umuahia-Abia State, Nigeria
Corresponding Author: Okereke Eze Aru

ABSTRACT: *An Artificial Neural Network (ANN) is an information processing system which is inspired by the way biological nervous systems, like the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working together to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. This research work looks at the learning in biological systems as involves adjustments to the synaptic connections that exist between the neurons and its similarity with the ANNs paradigm. These biological methods of computing are considered to be the next major advancement in the Computing Industry, hence worthy of study. Systems combining both fuzzy logic and neural networks with their different architectures in different application areas are considered in this study. The neural network learns by adjusting its weights and bias (threshold) iteratively to yield desired output. These are also called free parameters. For learning to take place, the Neural Network is trained first. The training is performed using a defined set of rules also known as the learning algorithm. Neural networks should not, however, be heralded as a substitute for statistical modeling, but rather as a complementary effort or an alternative approach to fitting non-linear data.*

KEYWORDS: *Artificial Neural Networks, learning, Human Brain, Fuzzy-Logic, Learning Algorithm, etc*

Date of Submission: 27-04-2018 Date of acceptance: 12-05-2018

I INTRODUCTION

Artificial Neural Networks are electronic models based on the neural structure of the brain. The brain learns from experience. The most basic element of the human brain is a cell which, unlike the rest of the body, doesn't appear to regenerate. These cells are known as neurons. There are 100 billion of these cells in the brain and each can connect with up to 200,000 other neurons (Jain & Mao, 1996). There are multiple connections between them.

The individual neurons are complicated. They have a myriad of parts, sub-systems, and control mechanisms. They convey information via a host of electrochemical pathways. Together these neurons and their connections form a process which is not binary, not stable, and not synchronous. The artificial neural networks try to replicate only the most basic elements of this complicated, versatile, and powerful organism. But for the software engineer who is trying to solve problems, neural computing was never about replicating human brains. It is about machines and a new way to solve problems.

II ARTIFICIAL NEURAL NETWORK PROCESSING ELEMENTS

The fundamental processing element of a neural network is a neuron. This building block of human awareness encompasses a few general capabilities. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then outputs the final result. Fig. 1 shows the relationship of these four parts.

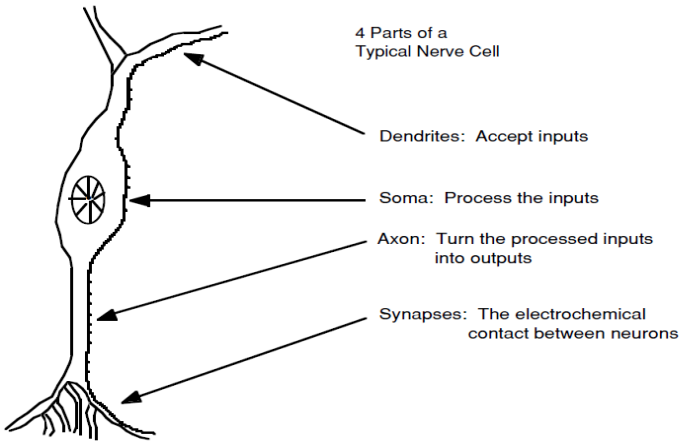


Fig. 1: Model of a Biological Neuron (Jain & Mao, 1996).

These input channels receive their input through the synapses of other neurons. The soma then processes these incoming signals over time and then turns that processed value into an output which is sent out to other neurons through the axon and the synapses. Currently, the goal of Artificial Neural Networks is not to recreate the brain rather to engineer solutions to problems that have not been solved by traditional computing. Hence, the basic unit of Artificial Neural Networks (ANN), the artificial neurons, simulate the four basic functions of natural neurons (Sasikumar & Balakrushna, 2011). Fig. 2 shows a fundamental representation of an artificial neuron.

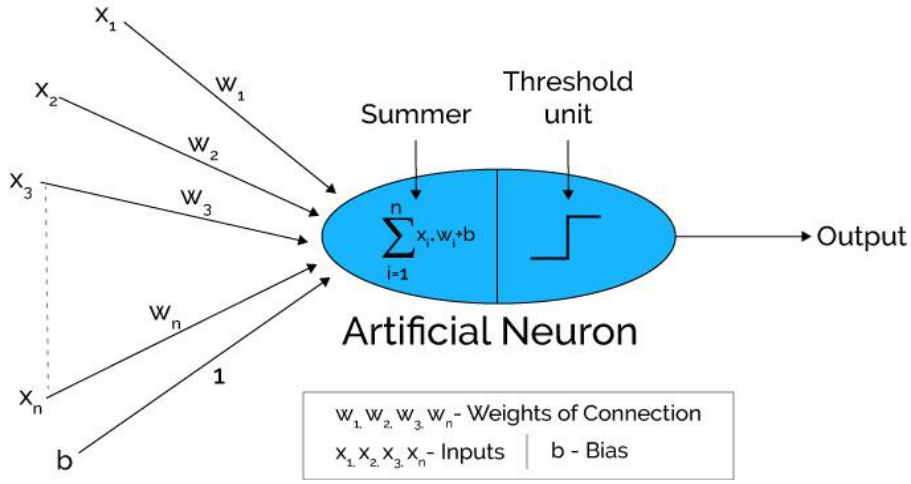


Fig. 2: Model of Artificial Neuron (Renner and Lacher, 2000).

$I = \sum w_i x_j - summation \quad Y = f(t) - Transfer$

Where $X_o, X_1, \ldots X_n$ = Inputs , $W_o, W_1, \ldots W_n$ = Weights

In Fig. 2, various inputs to the network are represented by the mathematical symbol, x_n. Each of these inputs are multiplied by a connection weight. These weights are represented by w_n . In the simplest case, these products are simply summed, fed through a transfer function to generate a result, and then output.

The mathematical representation of a neuron depicted above can be described as:

$$Y = f\left(\left(\sum_{i=1}^{n} x_i\, w_i\right) + b\right) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 1$$

where x_1 , x_2 ,....x_n represent an input vector, and w_1 , w_2,....w_n represent the weights (or strengths) of the incoming synapses (or interconnections). The bias (b) performs an affine transformation of the linearly combined input signals, and the activation function (f) applies to produce the final output (Y) from the neuron. Fig. 3 is a more detailed schematic of the artificial neuron. In Fig. 3, inputs enter into the processing element from the upper left. The first step is for each of these inputs to be multiplied by their respective weighting factor (w_n). Then these modified inputs are fed into the summing function, which usually just sums these products. Yet, many different types of operations can be selected. These operations could produce a number of different

values which are then propagated forward; values such as the average, the largest, the smallest, the ORed values, the ANDed values, etc. Furthermore, most commercial development products allow software engineers to create their own summing functions via routines coded in a higher level language (C is commonly supported). Sometimes the summing function is further complicated by the addition of an activation function which enables the summing function to operate in a time sensitive way. Either way, the output of the summing function is then sent into a transfer function. This function then turns this number into a real output via some algorithm. It is this algorithm that takes the input and turns it into a zero or a one, a minus one or a one, or some other number. The transfer functions that are commonly supported are sigmoid, sine, hyperbolic tangent, etc. This transfer function also can scale the output or control its value via thresholds. The result of the transfer function is usually the direct output of the processing element (Dave and George, 1992). An example of how a transfer function works is shown in Fig. 4. This sigmoid transfer function takes the value from the summation function, called sum in the Fig.4, and turns it into a value between zero and one.

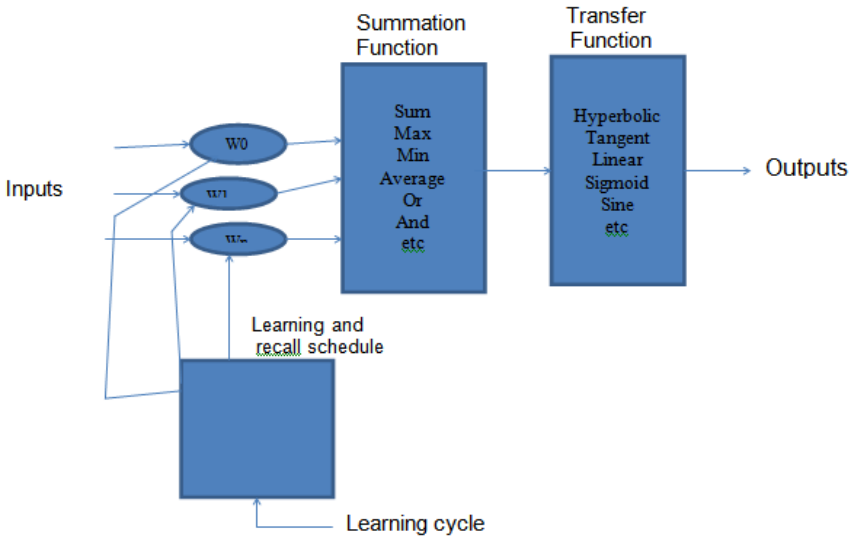


Fig. 3: A model of a Processing Element (Fraser, 1957)

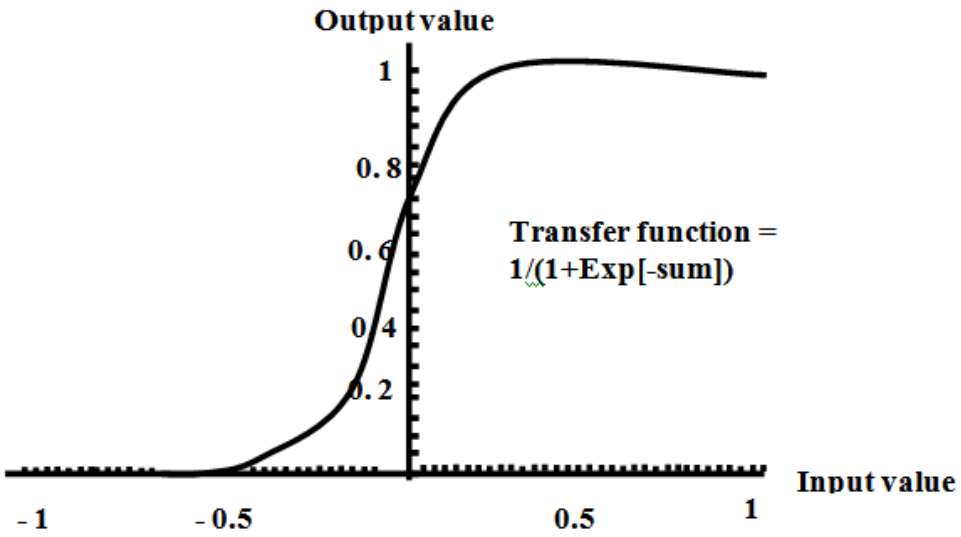


Fig. 4: Sigmoid Transfer Function ((Agarwal *et al.*, 2004))

Finally, the processing element is ready to output the result of its transfer function. This output is then input into other processing elements, or to an outside connection, as dictated by the structure of the network. All Artificial Neural Networks are constructed from this basic building block - the processing element or the Artificial Neuron. It is variety and the fundamental differences in these building blocks which partially cause the implementing of Neural Networks to be an "art."

III THE GENERAL STRUCTURE OF A NEURAL NETWORK MODEL

The most commonly used structure is in the Fig. 5. The neural network model is formed in three layers, called the input layer X1, hidden layer X2, and output layer X3. Each layer consists of one or more nodes, represented in this diagram by the small circles. The lines between the nodes indicate the flow of information from one node to the next. In this particular type of neural network, the information flows only from the input to the output (that is, from left-to-right). The nodes of the input layer are passive, meaning they do not modify the data. They receive a single value on their input, and duplicate the value to their multiple outputs. In comparison, the nodes of the hidden and output layer are active. This means they modify the data. Each value from the input layer is duplicated and sent to all of the hidden nodes. This is called a fully interconnected structure. The values entering a hidden node are multiplied by weights, a set of predetermined numbers stored in the program. Before leaving the node, this number is passed through a nonlinear mathematical function called a sigmoid. This is an "s" shaped curve that limits the node's output. That is, the input to the sigmoid is a value between -∞ and +∞, while its output can only be between 0 and 1. The active nodes of the output layer combine and modify the data to produce the two output values of this network.

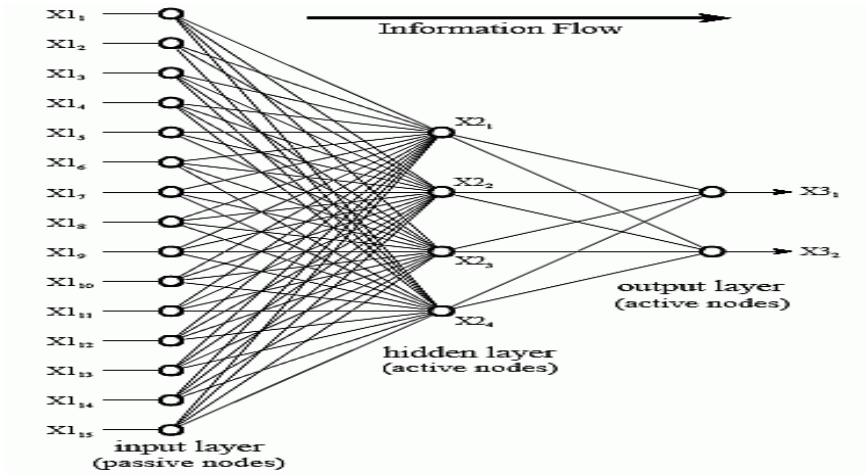


Fig. 5: Sample ANN Model (Aigbe, P.E., 2008).

IV COMPONENTS OF AN ARTIFICIAL NEURON

This section describes the seven major components which make up an artificial neuron. These components are valid whether the neuron is used for input, output, or is in one of the hidden layers.

Component 1. (Weighting Factors): A neuron usually receives many simultaneous inputs. Each input has its own relative weight which gives the input the impact that it needs on the processing element's summation function. These weights perform the same type of function as do the varying synaptic strengths of biological neurons.

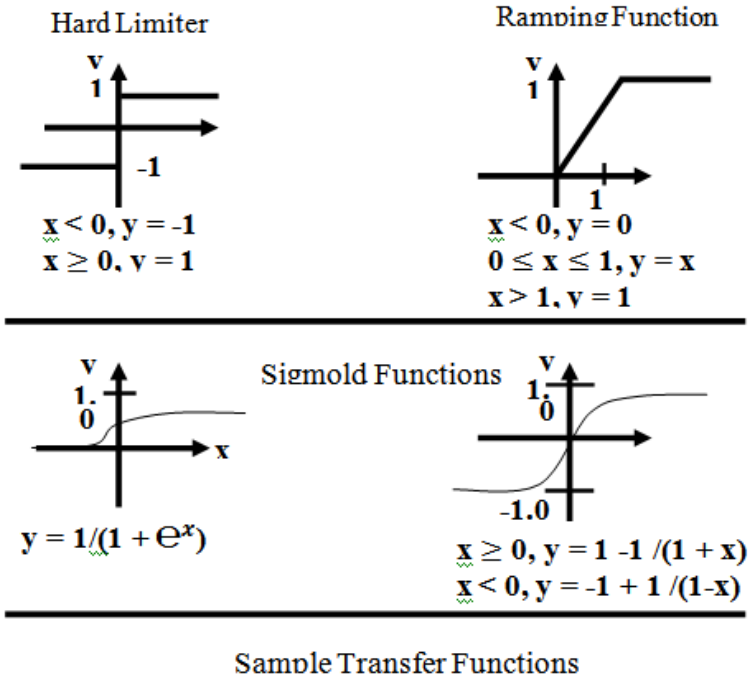
Component 2. (Summation Function): The first step in a processing element's operation is to compute the weighted sum of all of the inputs. Mathematically, the inputs and the corresponding weights are vectors which can be represented as (X11, X12 . . . X1n) and (W21, W22 . . . W2n). The total input signal is the dot, or inner, product of these two vectors. This simplistic summation function is found by multiplying each component of the i vector by the corresponding component of the w vector and then adding up all the products.

Component 3. (Scaling and Limiting): After the processing element's transfer function, the result can pass through additional processes which scale and limit. This scaling simply multiplies 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.

Component 4. (Output Function (Competition)): Each processing element is allowed one output signal which it may output to hundreds of other neurons. This is just like the biological neuron, where there are many inputs and only one output action. Normally, the output is directly equivalent to the transfer function's result.

Component 5. (Error Function and Back-Propagated Value): In most learning networks the difference between the current output and the desired output is calculated. This raw error is then transformed by the error function to match particular network architecture. The most basic architectures use this error directly, but some square the error while retaining its sign, some cube the error, and other paradigms modify the raw error to fit their specific purposes. The artificial neuron's error is then typically propagated into the learning function of another processing element. This error term is sometimes called the current error (Binitha & Sathya, 2012).

Component 7. (Transfer Function): The result of the summation function, almost always the weighted sum, is transformed to a working output through an algorithmic process known as the transfer function. In the transfer function shown in Fig. 6, the summation total can be compared with some threshold to determine the neural output.



Sample Transfer Functions

Fig. 6: Sample Transfer Functions (Andrew Whinston, 1996)

Component 7.
Learning Function: The purpose of the learning function is to modify the variable connection weights on the inputs of each processing element according to some neural based algorithm. This process of changing the weights of the input connections to achieve some desired result can also be called the adaption function, as well as the learning mode.

V TRAINING AN ARTIFICIAL NEURAL NETWORK

Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins.

There are two approaches to training - supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help. The vast bulk of networks utilize supervised training. Unsupervised training is used to perform some initial characterization on inputs (Boufardea & Garofalakis, 2012). However, in the full blown sense of being truly self learning, it is still just a shining promise that is not fully understood, does not completely work, and thus is relegated to the laboratory.

VI NEURO FUZZY SYSTEMS

The neuro-fuzzy term was born by the fusing of these two techniques. As researcher combine these two tools in different way, then, some confusion was created on the exact meaning of this term. Still there is no absolute consensus but in general, the neuro-fuzzy term means a type of system characterized for a similar structure of a fuzzy controller where the fuzzy sets and rules are adjusted using neural networks tuning techniques in an iterative way with data vectors (input and output system data) , (Lin, C.T. & Lee, C.S.,(1991)) as shown in Fig. 7. Such systems show two distinct ways of behavior.

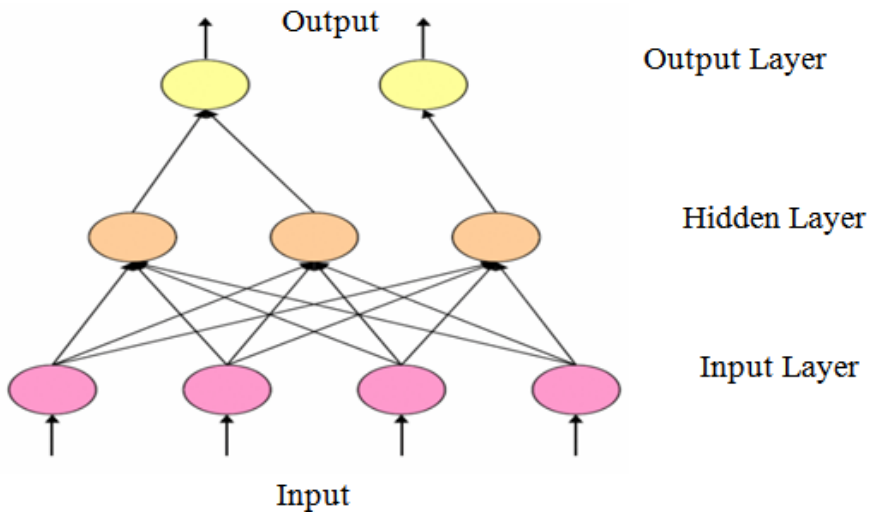


Fig. 7: The Structure of a Neuro-Fuzzy System (Lin & Lee, 1991)

In a first phase, called learning phase, it behaves like neural networks that learns its internal parameters off-line. Later, in the execution phase, it behaves like a fuzzy logic system.

Separately, each one of these techniques possess advantages and disadvantages that, when mixed together provides better results than the ones achieved with the use of each isolated technique.

Since the moment that fuzzy systems become popular in industrial application, the community perceived that the development of a fuzzy system with good performance is not an easy task. The problem of finding membership functions and appropriate rules is frequently a tiring process of attempt and error. This leads to the idea of applying learning algorithms to the fuzzy systems. The neural networks, that have efficient learning algorithms, had been presented as an alternative to automate or to support the development of tuning fuzzy systems. The first studies of the neuro-fuzzy systems started in early 90', (Jang,1991), (Berenji, 1992) and (Nauck, 1993). The majority of the first applications were in process control. Gradually, its application spread for all the areas of the knowledge like, data analysis, data classification, imperfections detection and support to decision-making, etc. Neural networks and fuzzy systems can be combined to join its advantages and to cure its individual illness. Neural networks introduce its computational characteristics of learning in the fuzzy systems and receive from them the interpretation and clarity of systems representation. Thus, the disadvantages of the fuzzy systems are compensated by the capacities of the neural networks. These techniques are complementary, which justifies its use together.

6.1 Types of Neuro-Fuzzy Systems

In general, all the combinations of techniques based on neural networks and fuzzy logic can be called neuro-fuzzy systems. The different combinations of these techniques can be divided, in accordance with Jang (1992), in the following classes:

***Cooperative Neuro-Fuzzy System*:** In the cooperative systems there is a pre-processing phase where the neural networks mechanisms of learning determine some sub-blocks of the fuzzy system. For instance, the fuzzy sets and/or fuzzy rules (fuzzy associative memories or the use of clustering algorithms) to determine the rules and fuzzy sets position. After the fuzzy sub-blocks are calculated the neural network learning methods are taken away, executing only the fuzzy system.

***Concurrent Neuro-Fuzzy System*:** In the concurrent systems the neural network and the fuzzy system work continuously together. In general, the neural networks pre-processes the inputs (or pos-processes the outputs) of the fuzzy system.

Hybrid Neuro-Fuzzy Systems: In this category, a neural network is used to learn some parameters of the fuzzy system (parameters of the fuzzy sets, fuzzy rules and weights of the rules) in an iterative way. The majority of the researchers use the neuro-fuzzy term to refer only hybrid neuro-fuzzy system. Nauck (1995) definition: "A hybrid neuro-fuzzy system is a fuzzy system that uses a learning algorithm based on gradients or inspired by the neural networks theory (heuristical learning strategies) to determine its parameters (fuzzy sets and fuzzy rules) through the patterns processing(input and output)".

A neuro-fuzzy system can be interpreted as a set of fuzzy rules. This system can be created from input/ output data or initialised with a prior knowledge in the same way of fuzzy rules. The resultant system by fusing fuzzy systems and neural networks has as advantages of learning through patterns and the easy interpretation of its functionality. There are several different ways to develop hybrid neuro-fuzzy systems, therefore, being a

recent research subject, each researcher has defined its own particular models. These models are similar in its essence, but they present basic differences. Many types of neuro-fuzzy systems are represented by neural networks that implement logical functions. This is not necessary for the application of a learning algorithm into a fuzzy system, however, the representation trough a neural networks is more convenient because it allows to visualize the flow of data through the system and the error signals that are used to update its parameters. The additional benefit is to allow the comparison of the different models and visualize its structural differences.

6.2 Architectures of Neuro-Fuzzy Network System

The techniques of artificial intelligence based in fuzzy logic and neural networks are frequently applied together. The reasons to combine these two paradigms come out of the difficulties and inherent limitations of each isolated paradigm. Generically, when they are used in a combined way, they are called Neuro-Fuzzy Systems (Viharos1 & Kis1, 2014). This term, however, is often used to assign a specific type of system that integrates both techniques. This type of system is characterized by a fuzzy system where fuzzy sets and fuzzy rules are adjusted using input output patterns. There are several different implementations of neuro-fuzzy systems, where each author defined its own model.

The *Fuzzy Adaptive Learning Control Network Architecture (FALCON)*

FALCON is an architecture of five layers as it is shown in Fig. 8. There are two linguistics nodes for each output. One is for the patterns and the other is for the real output of the FALCON. The first hidden layer is responsible for the mapping of the input variables relatively to each membership functions. The second hidden layer defines the antecedents of the rules followed by the consequents in the third hidden layer. According to Kosko (1992), FALCON uses a hybrid learning algorithm composed by a unsupervised learning to define the initial membership functions and initial rule base and it uses a learning algorithm based on the gradient descent to optimize/adjust the final parameters of the membership functions to produce the desired output.

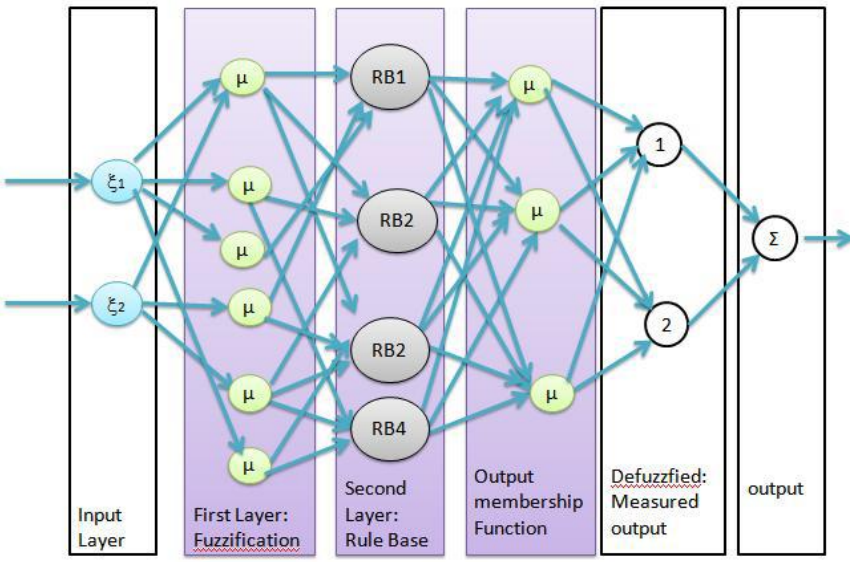


Fig. 8: FALCON architecture.

The *Adaptive Network based Fuzzy Inference System Architecture (ANFIS)*

(Jang,1993) proposed architecture and learning algorithms which is combination of fuzzy logic with neural networks for drawing inference . It is a model that is proficient in constructing input-output mapping accurately based on both human knowledge using data in the form of fuzzy if-then rules and predetermined input output data pairs. ANFIS implements a Takagi Sugeno fuzzy inference system and it has five layers as shown in Fig. 9. The first hidden layer is responsible for the mapping of the input variable relatively to each membership functions. The operator T-norm is applied in the second hidden layer to calculate the antecedents of the rules. The third hidden layer normalizes the rules strengths followed by the fourth hidden layer where the consequents of the rules are determined. The output layer calculates the global output as the summation of all the signals that arrive to this layer. ANFIS uses back propagation learning to determine the input membership functions parameters and the least mean square method to determine the consequents parameters. Each step of the iterative learning algorithm has two parts. In the first part, the input patterns are propagated and the parameters of the consequents are calculated using the iterative minimum squared method algorithm, while the parameters of the premises are considered fixed. In the second part, the input patterns are propagated again and

in each iteration the learning algorithm back propagation is used to modify the parameters of the premises, while the consequents remain fixed.

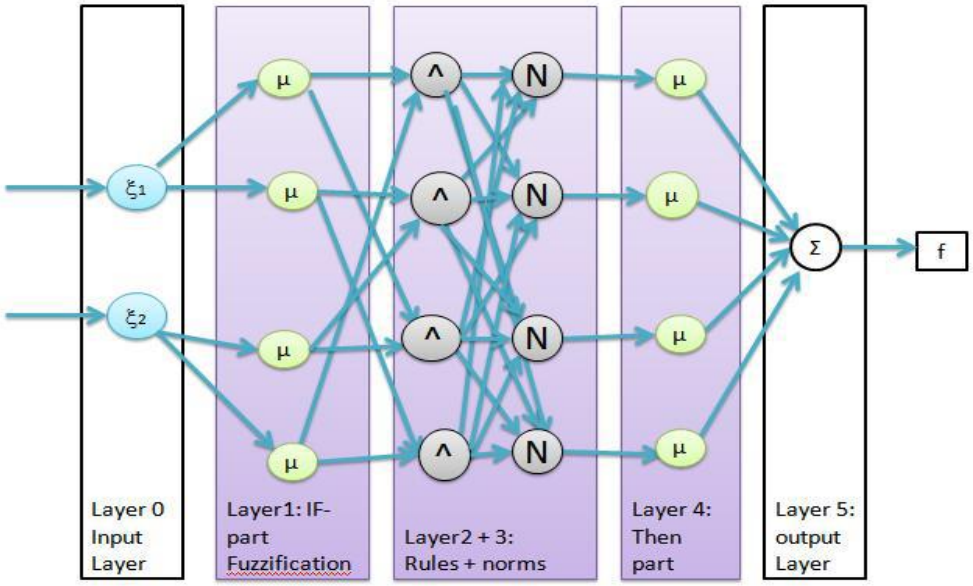


Fig. 9: ANFIS architecture (Lei Y., He Z., Zi Y. & Hu Q., 2007).

The Generalized Approximate Reasoning based Intelligence Control Architecture (GARIC)

GARIC implements a neuro-fuzzy system using two neural network modules, ASN (Action Selection Network) and AEN (Action State Evaluation Network). The AEN is an adaptative evaluator of ASN actions. The ASN of the GARIC is an advanced network of five layers.

Fig. 10 illustrates GARIC-ASN structure. The connections between the layers are not weighted.

The first hidden layer stores the linguistics values of all input variables. Each input can only connect to the first layer, which represents its associated linguistics values. The second hidden layer represents the fuzzy rule nodes that determine the compatibility degree of each rule using a softmin operator. The third hidden layer represents the linguistics values of the output variables. The conclusions of each rule are calculated depending on the strength of the rules antecedents calculated in the rule nodes. GARIC uses the mean of local mean of maximum method to calculate the output of the rules. This method needs for a numerical value in the exit of each rule. Thus, the conclusions should be transformed from fuzzy values for numerical values before being accumulated in the final output value of the system. GARIC uses a mixture of gradient descending and reinforcement learning for a fine adjustment of its internal parameters.

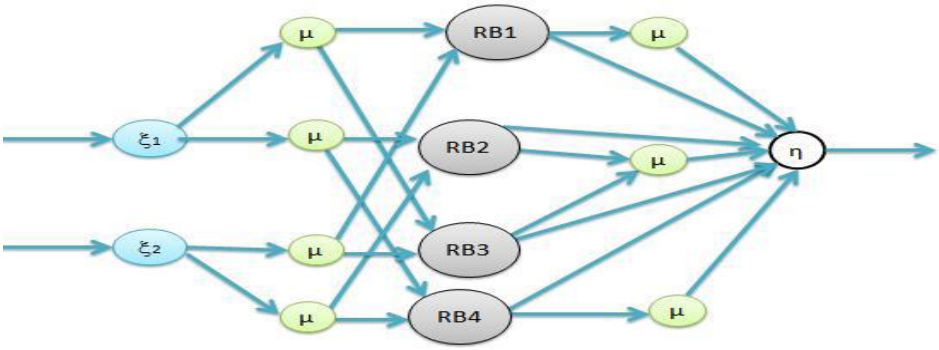


Fig. 10: GARIC Architecture. (Takagi & Hayashi, 1991)

The ***Neural Fuzzy Controller Architecture (NEFCON)***

NEFCON was drawn to implement a Mamdani type inference fuzzy system as illustrated in Fig. 11. The connections in this architecture are weighted with fuzzy sets and rules using the same antecedents (called shared weights), which are represented by the drawn ellipses. They assure the integrity of the base of rules. The input units assume the function of fuzzyfication interface, the logical interface is represented by the propagation function and the output unit is responsible for the defuzzyfication interface. The process of learning in

architecture NEFCON is based in a mixture of reinforcement learning with back propagation algorithm. This architecture can be used to learn the rule base from the beginning, if there is no prior knowledge of the system, or to optimise an initial manually defined rule base. NEFCON has two variants NEFPROX (for function approximation) and NEFCLASS (for classification tasks).

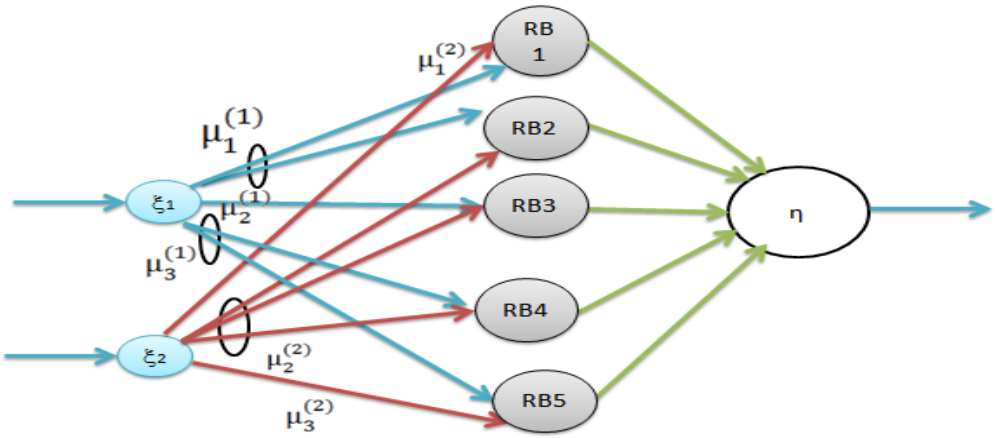


Fig. 11: NEFCON Architecture (Nauck & Kruse,1994).

Evolving Neural Fuzzy Network Architecture (EFuNN)

In EFuNN all nodes are created during the learning phase. The first layer passes data to the second layer that calculates the degrees of compatibility in relation to the predefined membership functions. The third layer contains fuzzy rule nodes representing prototypes of input- output data as an association of hyper-spheres from the fuzzy input and fuzzy output spaces. Each rule node is defined by two vectors of connection weights, which are adjusted through a hybrid learning technique. The fourth layer calculates the degree to which output membership functions are matched the input data and the fifth layer carries out the defuzzyfication and calculates the numerical value for the output variable.

Dynamic Evolving Neural Fuzzy Network (DMEFUNN).

This is a modified version of the EFuNN with the idea of not only the winning rule node's activation is propagated but a group of rule nodes that is dynamic selected for every new input vector and their activation values are used to calculate the dynamical parameters of the output function as shown in Fig. 12. While EFuNN implements Mamdani type fuzzy rules, dmEFuNN implements Takagi Sugeno fuzzy rules.

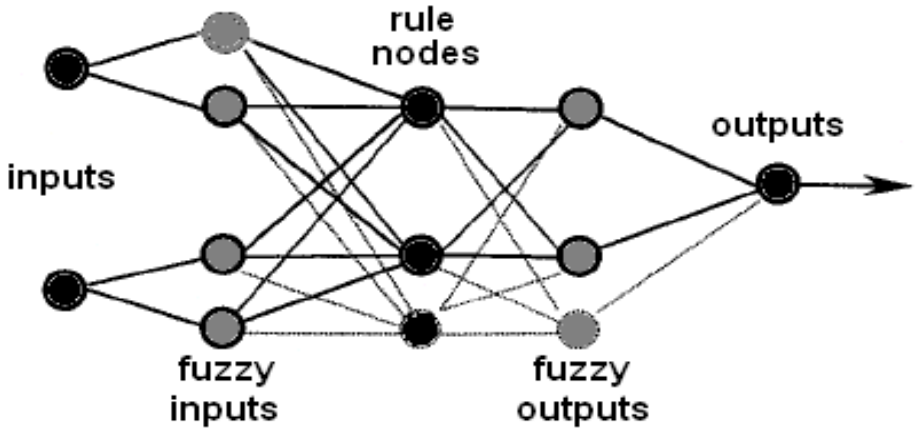


Fig. 12: EfuNN Architecture (Neha Kashyap & Nidhi Agarwal, 2014).

To get a more detail description of this architecture, beyond the specific pointed references made, a detailed survey was made by Abraham in 2000 where it can be found a detailed description of several well known neuro-fuzzy architectures and their respective learning algorithms.

7.0 **Conclusion**:

Artificial neural networks are computers whose architecture is modeled after the brain. They typically consist of many hundreds of simple processing units which are wired together in a complex communication network. Each unit or node is a simplified mode l of a real neuron which sends off a new signal if it receives a sufficiently strong input signal from the other nodes to which it is connected. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. Prediction for the future rests on some sort of evidence or established trend which, with extrapolation, clearly takes us into a new realm. Neural Networks will fascinate user-specific systems for education, information processing, entertainment, genetic engineering, neurology and psychology. ANNs, like people, learn by example. NN's ability to learn by example makes them very flexible and powerful. Perhaps the most exciting aspect of neural networks is the possibility that some day 'conscious' networks might be produced

REFERENCES

[1]. Agarwal, A.,Singh. R. D., (2004) Runoff modelling through back propagation artificial neural network with variable rainfall-runoff data. Water Resources Management. 2004, 18(3): 285-300.

[2]. Aigbe, P.E. (2008) Design and Implementation of a Web Based University Admission and Placement Neural Network Model. Unpublished M.Sc. Thesis. Ahmadu Bello University Zaria, Kaduna State

[3]. Andrew Whinston (1996) Instructor's manual with test bank to accompany decision support systems: a knowledge-based approach. West Publishing, 703, 1996.

[4]. Berenji, H.R. (1992) A Reinforcement Learning Based Architecture for Fuzzy Logic Control. *International Journal of Approximate Reasoning*6, 267-292.

[5]. Binitha, S., and Sathya, S. S. (2012) A Survey of Bio inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering*, 2(2) 2231-2307.

[6]. Boufardea, E., and Garofalakis, J. (2012) A Predictive System for Distance Learning Based on Ontologies and Data Mining, COGNITIVE 2012: The fourth *International Conference on Advanced Cognitive Technologies and Applications*, 151-158

[7]. Dave, A. and George, M. (1992) Artificial Neural Networks Technology, A DACS State-of-the-Art Report. Contract Number F30602-89-C-0082 (Data & Analysis Center for Software),ELIN: A011.August 20 1992

[8]. Fraser, A. S. (1957) Simulation of genetic systems by automatic digital computers: Effects of linkage on rates of advance under selection. *Australian Journal of Biological Science* 10, 492–499.

[9]. Jain, A. K. and Mao, J. (1996) Artificial neural networks: A tutorial. Appeared in IEEE Computer,

[10]. Jang, R. (1991) "Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm", Proc. Ninth Nat. Conf. Artificial Intell., pp. 762-767, 1991.

[11]. Jang, R. (1992) "Neuro-Fuzzy Modelling: Architectures, Analysis and Applications", PhD Thesis, University of California, Berkley.

[12]. Jang, J.-S.R. (1993)ANFIS: adaptive-network-based fuzzy inference systems. IEEE Transactions on Systems, Man, and Cybernetics 23 (3),665–685. systems.IE (3),665–685.

[13]. Kosko, B. (1992) "Neural Networks and Fuzzy Systems: A Dynamical System Approach to Machine Intelligence", Prentice Hall, Englewood Cliffs, New Jersey.

[14]. Lei Y., He Z., Zi Y. & Hu Q., (2007) "Fault diagnosis of rotating machinery based on multiple ANFIS combination with GAs", Mechanical Systems and Signal Processing, vol. 21: (5), pp. 2280-2294, 2007.

[15]. Lin, C.T. & Lee, C.S.,(1991) Neural-Network-Based Fuzzy Logic Control And Decision Systems. IEEE Transactions on Computers, Vol. 40, No. 12, Pp. 1320-1336.

[16]. Linn, Baker, and Dunbar, (1991)Complex, Performance-Based Assessment: Expectations and Validation Criteria." Educational Researcher 20, 8: 15–21. (ERIC Document Reproduction Service No. EJ 436 999).

[17]. Nauck, D. and Kruse, R., (1993) A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error back- propagation. In: Proceedings of the IEEE International Conference on NeuralNetworksICNN'93, SanFrancisco,CA,pp.1022–1027.

[18]. Nauck, D. and Kruse, R. (1994) NEFCON-I: An X window based simulator for Neural Fuzzy Controllers. In proceeding of IEEE International Conference on Neural Networks, 1994 at IEEE WCCT94, Pages 1638-1643. Orlando

[19]. Nauck, D. (1995) "Beyond Neuro-Fuzzy Systems: Perspectives and Directions". Proceeding of the Third European Congress on Intelligent Techniques and Soft Computing (EUFIT'95), Aachen.

[20]. Neha Kashyap and Nidhi Agarwal(2014) International Journal of Advanced Engineering Research and Technology (IJAERT), ISSN: 2348–8190

[21]. ICRTIET-2014 Conference Proceeding, 30th -31st August 2014. pages 81-85

[22]. Renner R.S. and Lacher R.C. (2000) Combining Constructive Neural Networks for Ensemble Classification. Proceedings of the Fifth Joint Conference on Information Science, 887-891

[23]. Sasikumar, G. and Balakrushna K. T. (2011) Study of Image Recognition Using Cellular Associated Artificial Neural Networks. Proceedings of the International Multi Conference of Engineers and Computer Scientists 2011 Vol I, IMECS 2011, March 16-18, 2011, Hong Kong

[24]. Takagi, H. & Hayashi, I. (1991) Neural Network-driven fuzzy reasoning, International Journal of Approximate Reasoning, Volume 5 (1991) 191 212.

[25]. Viharos, Zs. J. & Kis, K. B.(2014) Survey on Neuro-Fuzzy Systems and their Applications in Technical Diagnostics. 13th IMEKO TC10 Workshop on Technical Diagnostics Advanced measurement tools in technical diagnostics for systems' reliability and safety. June 26-27, 2014, Warsaw, Poland

Okereke Eze Aru. " Review Of Artificial Neural Network: An Emerging Problem Solving Tool In Computing"American Journal of Engineering Research (AJER), vol. 7, no. 5, 2018, pp.152-161.