

# Artificial Neural Networks

Subjects: [Computer Science](#), [Artificial Intelligence](#)

Contributor: Axel Escamilla-García , Genaro M. Soto-Zarazúa , Manuel Toledano-Ayala , Edgar Rivas-Araiza , Abraham Gastélum-Barrios

Artificial Intelligence (AI) researches and builds intelligent software and machines, provides a particular solution to a particular defined complex problem. To carry out these tasks, it uses models and algorithms such as genetic algorithms, particle swarm optimization, artificial neural networks (ANNs), and hybrid models (two or more of the above). ANNs are flexible to accommodate non-linear and non-physical data; however, they require large multidimensional data set to reduce the risk of extrapolation. ANNs are widely used in different branches of science for their learning ability and adaptability to various settings.

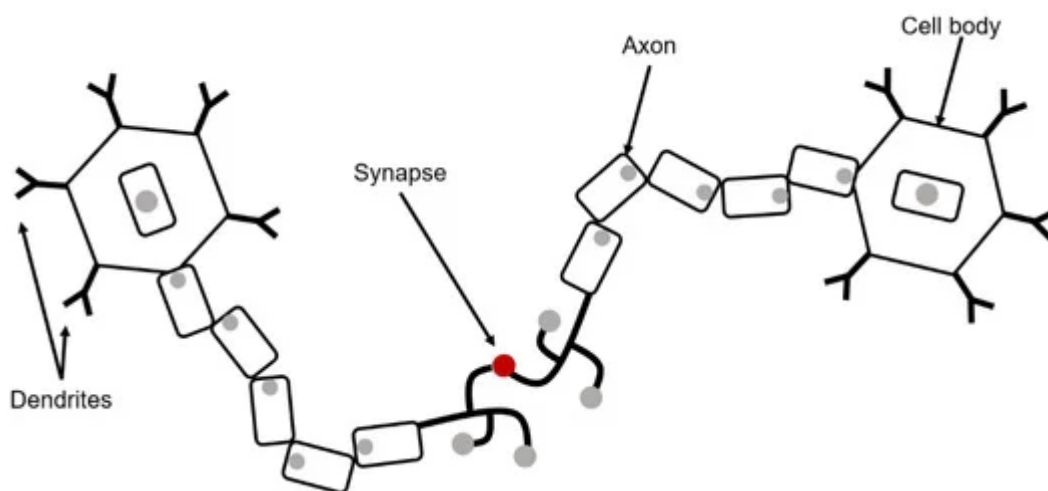
Feedforward Neural Networks

Recurrent Neural Networks

Learning of Artificial Neural Networks

Activation Function

An ANN is a ML algorithm based on the concept of a human neuron [\[1\]](#). It is a biologically inspired computational model, consisting of processing elements (neurons) and connections between them with coefficients (weights) attached to the connections [\[2\]](#). ANNs are inspired by the brain structure and for this reason it is important to define the main components under which a neuron, dendrites, cell body, and axon works. Dendrites are a network that carries electrical signals to the cell body. The cell body adds and collects the signals. The axon carries the signal from the cell body to other neurons using a long fiber. When the axon of a cell comes in to contact with a dendrite of another cell it is known as a synapse. Therefore, the functions of neuronal networks are established through the arrangement of neurons and individual synaptic forces [\[3\]](#). [Figure 1](#) presents a general schematic of a biological neuron with each element that makes it up.



**Figure 1.** Structure of a biological neuron.

Neural structures develop through learning; however, they constantly change, strengthening or weakening the synaptic junctions. Although ANNs are inspired by the brain, they are not that complex. However, the greatest similarities are primarily that both networks are interconnected and the functions of the networks are determined by the connections between neurons [4].

Neurons receive inputs such as impulses. The peak rate generated over time and the average peak generation rate in several runs, are some measures used to describe neuron activity. In ANNs, a neuron is identified by the speed at which it generates these peaks. A neuron connects to other neurons in the previous layer through adaptive synaptic weights. Knowledge is generally stored as a set of connection weights. When these connection weights are modified in an orderly manner and with a suitable learning method, a training process is carried out. The learning method consists of presenting the input to the network and the desired output, adjusting the weights so that the network can produce the desired output. After training the weights will have relevant information, whereas before training it is redundant and meaningless [5].

Figure 2 presents the simple neuron structure. The processing of the information in a neuron begins with the inputs  $X_n$ , they are weighted and added up before going through some activation function to generate its output, this process is represented as  $\xi = \sum X_i \cdot W_i$ . For each of the outgoing connections, this activation value is multiplied by the specific weight  $W_n$  and transferred to the next node. If it considers a linear activation, the output would be given by  $y = a(wx + b)$  [6].

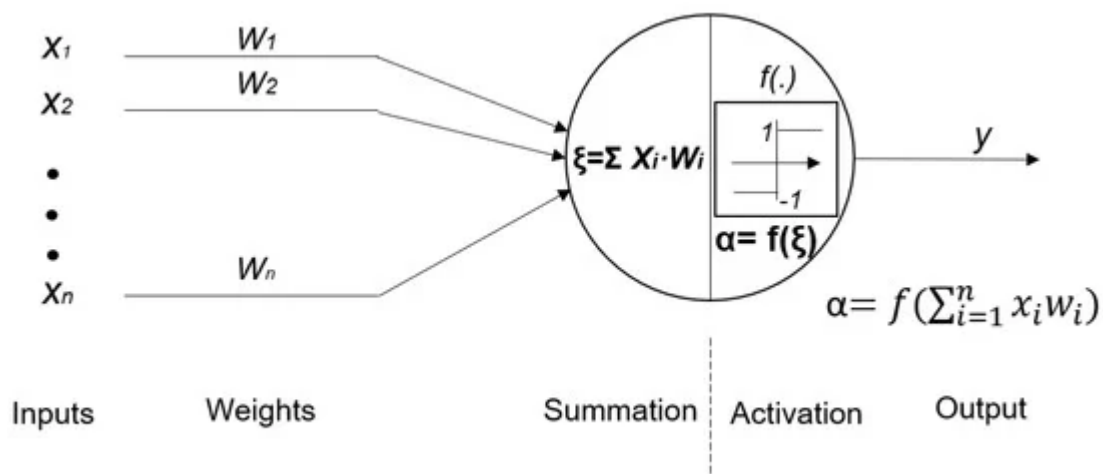


Figure 2. The basic scheme of a neuron.

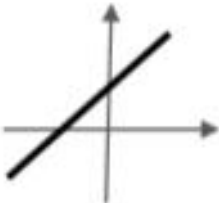
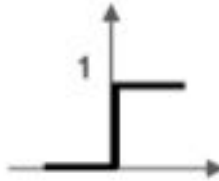

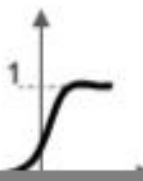
## The Activation Function of an Artificial Neural Network

The activation function is a function that receives an input signal and produces an output signal after the input exceeds a certain threshold. That is, neurons receive signals and generate other signals [7]. The neuron start is only performed when the sum of the total inputs is greater than the neuron threshold limit, then the output will be transmitted to another neuron or environment. This threshold limit determines whether the neuron is activated or

not, the most common activation or transfer functions are the linear, binary step, piecewise linear, sigmoid, Gaussian and hyperbolic tangent functions [8].

[Table 1](#) shows the activation functions commonly used in NNs. The behavior of neurons is defined by these functions. If it transfers a function that is linear and the network is multi-layered, it can be represented as a single-layer network, since it is product of weight matrices of each layer and will only produce positive numbers over the entire range of real numbers. On the other hand, non-linear transfer functions (sigmoid function) between layers allow multiple layers to provide new capabilities, adjusting the weights to obtain a minimum error in each set of connections between layers [9][10]. Linear functions are generally used in the input and output layers, while non-linear activation functions can be used for the hidden and output layers [11].

**Table 1.** Activation functions for layers in artificial neural networks.

Name	Graphic	Function
Linear		$f(x) = w \cdot x + b$
Binary step		$\begin{aligned} \text{if } x \geq 0, & \quad \text{then } f(x) = 1, \\ \text{if } x < 0, & \quad \text{then } f(x) = 0, \end{aligned}$
Piecewise linear		$\begin{aligned} \text{if } x \geq x_{\text{upper}}, & \quad \text{then } f(x) = 1, \\ \text{if } x_{\text{upper}} > x > & \quad \frac{x_{\text{upper}}}{x}, \\ \text{if } x \leq x_{\text{lower}}, & \quad \text{then } f(x) = 0, \end{aligned}$
Sigmoid		$f(x) = \frac{1}{1 + e^{-x}}$ <p style="text-align: right;">interval (0,1)</p>

The most used non-linear activation functions are sigmoid and hyperbolic tangents. Hyperbolic and sigmoid tangents are mainly used because they are differentiable and make them compatible with the back-propagation algorithm. Both activation functions have an “S” curve, while their output range is different [12]. The sigmoid function is the most used activation function in ANNs. This function varies from 0 to +1, although the activation function

sometimes seeks to oscillate between  $-1$  and  $+1$ , in which case the activation function assumes an antisymmetric form with respect to the origin, defining it as the hyperbolic tangent function [\[13\]](#)[\[14\]](#)[\[15\]](#).

Direct implementation of sigmoid and hyperbolic tangent functions in hardware is impractical due to its exponential nature. There are several different approaches to the hardware approximation of activation functions, such as the piecewise linear approximation. Linear part approximations are slow, but they are the most common way of implementing activation functions [\[16\]](#). In addition, this uses a series of linear segments to approximate the trigger function. The number and location of these segments are chosen so that errors and processing time are minimized [\[17\]](#).

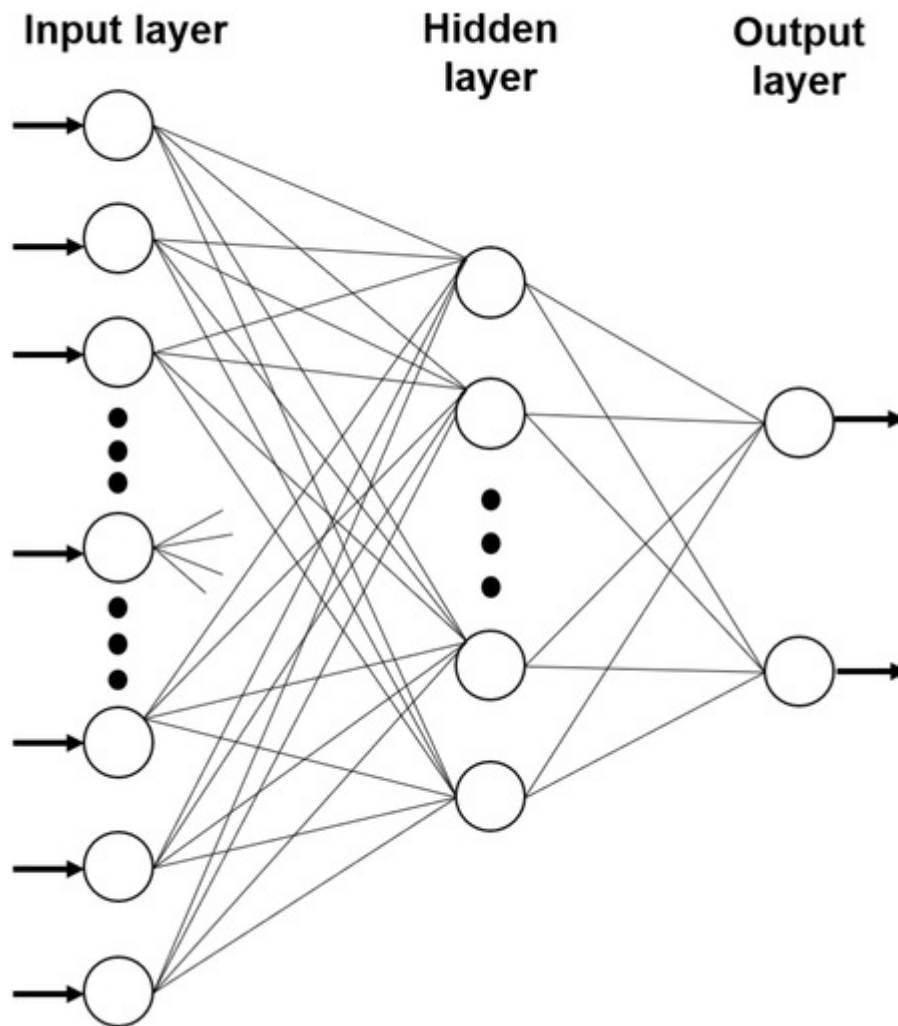
The Gaussian activation function can be used when finer control over the activation range is needed [\[10\]](#). Furthermore, it can uniformly perform continuous function approximations of various variables [\[18\]](#).

## Types of Artificial Neural Network

The ANNs are classified according to different criteria, we can establish that there are two types [\[19\]](#):

- Feedforward neural networks (FFNNs); **Feedforward Neural Networks**

- Recurrent neural networks (in discrete time) or differential (in continuous time); The neuron is the basic component of NNs. Neurons are connected to each other through synaptic weight [\[3\]](#). Considering a neural network with three layers such as in [Figure 3](#): an input layer, a hidden layer and an output layer the intermediate layer is considered self-organized Kohonen map, which consists of two layers of processing units (input and output), depending on the complexity of the network (there may be several hidden layers in each network) [\[20\]](#). In FFNNs, information progresses, from the input nodes to the hidden nodes and from the hidden nodes to the output nodes. When an input pattern is fed into the network, the units in the output layer compete with each other, and the winning output unit is the one whose input connection weights are closest to the input pattern, the number of neurons in the input and output layers is the same as the number of inputs and outputs of the problem [\[21\]](#). The learning method can be divided into two stages, the first stage is to determine the neuron of the hidden layer whose weight vector is the first input vector and the second refers to the training process. Initially, the Euclidean distance between the input and the weight vector of the first neuron will be calculated. If the distance is greater than a predetermined distance threshold value, a new hidden-layer neuron is created by assigning the input as the weight vector. Otherwise, the input pattern belongs to this neuron. During training, each pattern presented to the network selects the closest neuron on a Euclidean distance measure, modifying the winner's weight vector, and topological neighbors draws them in the direction of the input, the weights leaving the winning neuron and its neighbors are adjusted by the gradient descent method [\[22\]](#). Forward NNs fall into two categories based on the number of layers, either single layer or multiple layers [\[23\]](#).



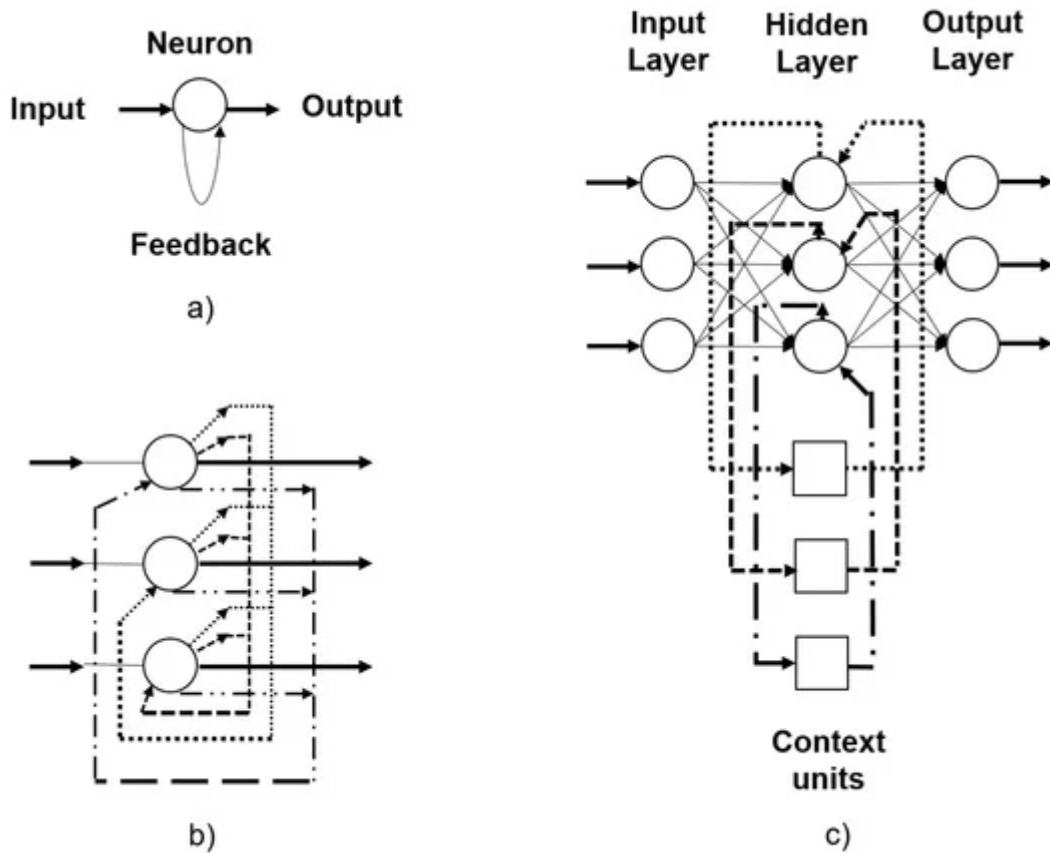
**Figure 3.** Feedforward neural network structure.

Back-propagation (BP) is a type of ANN training, used to implement supervised learning, tasks for which a representative number of sample inputs and correct outputs are known. BP is derived from the difference in desired and predicted, output; this is calculated and propagated backward [24]. First, network weights to a small random weight are initialized, the vector set of input data to the network are presented, the input propagated to generate the output, which is called the input advance phase, and the error comparing the estimated net output with the desired output calculated [25]. The weight will be corrected from the output to the input layer that is, in the backward direction in which the signals propagate when objects are introduced into the network. This is repeated until the error no longer improves [26].

## Recurrent Neural Networks

In recurrent neural networks (RNNs) the information goes back and forth as can be seen in [Figure 4a](#), for this reason, they are also called feedback networks. In these networks, the connections between nodes form a directed

cycle, where at least one path leads back to the initial neuron. In this type of network there are different types of structure [27]:



**Figure 4.** Recurrent neural networks structure: (a) Simple structure of a recurrent network, (b) Hopfield network structure, (c) Elman network structure.

- Hopfield network: each neuron is completely symmetrically connected with all other neurons in the network. If the connections are trained using Hebbian learning, then the Hopfield network can function as a solid memory and resistant to the alteration of the connection. Hebbian learning involves synapses between neurons and their strengthening when neurons on both sides of the synapse (input and output) have highly correlated outputs [28] as shown in [Figure 4b](#). There is a guarantee in terms of convergence for this network [29].

- Elman network: this is a horizontal network where a set of “context” neurons is added. In [Figure 4c](#) the context units are connected to the hidden network layer fixed with a weight. The subsequent fixed connections result in the context units always keeping a copy of the previous values of the hidden units, maintaining a state, which allows sequence prediction tasks [30].

- Jordan network: these are very similar to Elman’s networks. However, context units feed on the output layer instead of the hidden layer.

RNN is distinguished from a FFNN by the presence of at least one feedback connection. FFNNs do not have the intrinsic ability to process temporary information. There are two important considerations about why recurrent networks are viable tools for modeling: inference and prediction in noisy environments. In a typical recurrent network architecture, the activation functions of the hidden unit are fed back each time step to provide additional input. That is, the recurrent networks are built in such a way that the outputs of some neurons feed back to the same neurons or to the neurons in the previous layers [27]. Feedback from hidden units allows filtered data from the previous period to be used as additional input in the current period. This causes the network to work not only with the new data, but also with the past history of all entries, as well as their leaked equivalents. This additional filtered input history information acts as an additional guide to assess the current noisy input and its signal component. By contrast, filtered history never enters a FFNN. This is where recurring networks differ from a FFNN. Second, since recurrent networks have the ability to maintain the past history of filtered entries as additional information in memory, a recurrent network has the ability to filter noise even when the noise distribution can vary over time. In a FFNN a completely new training must be carried out with a new data set containing the new type of noise structure [19].

## Learning of Artificial Neural Networks

Learning is an essential part of NNs; this process defines the input-output relationship by looking for the most accurate prediction calculation. The learning process can be classified into two categories: supervised and unsupervised. Supervised learning knows the expected results and uses known or labeled data, while in unsupervised learning it is not necessary to have known data, and the learning is done through the discovery of internal structures and data representation [29][31].

Supervised learning consists of minimizing a cost function that accumulates the errors between the actual outputs of the system and the desired outputs, for the given inputs. To minimize this cost function, several methods are used, and the gradient descent as the error BP algorithm is the most used for its acceptable results in one layer and multilayer networks [32].

In unsupervised learning, it is based only on input data and the update of the weights is carried out internally in the network, the algorithms are designed for the self-organization of the ANNs and can be derived by Hebbian law, or the use of algorithms such as algebraic reconstruction technique [9].

The exposed learning techniques have allowed the development of advanced algorithms such as SOM (self-organizing maps) and SOTA (self-organizing tree algorithm), which are time series clustering algorithms based on unsupervised NNs [33]. SOM is a known data analysis tool for tasks like data visualization and clustering. One disadvantage of this tool is that the user must select the map size. This may lead to many experiments with different sized maps, trying to obtain the optimal result. Training and using these large maps may be quite slow [34]. While the SOTA permits classification in the initial levels of groups of patterns that are more separated from other and to classify patterns in final layers in a more accurate way [35].



These techniques open up the possibility of not only learning connection weights from examples, but also learning a neural network structure from examples. This is thanks to the fact that a neural network can be built automatically from the training data by SOTA methods [\[36\]](#).

---

## References

1. Yuanping Su; Lihong Xu; Towards discrete time model for greenhouse climate control. *Engineering in Agriculture, Environment and Food* **2017**, *10*, 157-170, 10.1016/j.eaef.2017.01.001.
2. Subana Shanmuganathan; Artificial Neural Network Modelling: An Introduction. *Advances in Intelligent Information and Database Systems* **2016**, *628*, 1-14, 10.1007/978-3-319-28495-8\_1.
3. Hagan, M.T.; Demuth, H.B.; Beale, M.H. *Neural Network Design*: Campus Pub. Serv. Univ. Color. Bookst.2002, 1284, 30–39.
4. Akbari, M.; Asadi, P.; Givi, M.K.B.; Khodabandehlouie, G. Artificial Neural Network and Optimization. In *Advances in Friction-Stir Welding and Processing*; Elsevier: Waltham, UK, 2014; pp. 543–599.
5. Soteris Kalogirou; Artificial neural networks in renewable energy systems applications: a review. *Renewable and Sustainable Energy Reviews* **2001**, *5*, 373-401, 10.1016/s1364-0321(01)00006-5.
6. Deb, A.K. *Introduction to Soft Computing Techniques: Artificial Neural Networks, Fuzzy Logic and Genetic Algorithms*. Soft Comput. Text. Eng. 2011, 3–24.
7. Han, S.-H.; Kim, K.W.; Kim, S.; Youn, Y.C. Artificial Neural Network: Understanding the Basic Concepts without Mathematics. *Dement. Neurocogn. Disord.* 2018, *17*, 83–89.
8. Profillidis, V.A.; Botzoris, G.N. Artificial Intelligence—Neural Network Methods. In *Modeling of Transport Demand*; Elsevier: Amsterdam, The Netherlands, 2019; pp. 353–382.
9. de B. Harrington, P. Sigmoid Transfer Functions in Backpropagation Neural Networks. *Anal. Chem.* 1993, *65*, 2167–2168.
10. Sibi, P.; Jones, S.A.; Siddarth, P. Analysis of Dierent Activation Functions Using Back Propagation Neural Networks. *J. Theor. Appl. Inf. Technol.* 2013, *47*, 1264–1268.
11. Özkan, C.; Erbek, F.S. The Comparison of Activation Functions for Multispectral Landsat TM Image Classification. *Photogramm. Eng. Remote Sens.* 2003, *69*, 1225–1234.
12. Zamanlooy, B.; Mirhassani, M. Ecient VLSI Implementation of Neural Networks with Hyperbolic Tangent Activation Function. *IEEE Trans. Very Large Scale Integr. Syst.* 2013, *22*, 39–48.
13. Csáji, B.C. Approximation with Artificial Neural Networks. *Fac. Sci. Etsv Lornd Univ. Hung.* 2001, *24*, 7.

14. Dalton, J.; Deshmane, A. Artificial Neural Networks. *IEEE Potentials* 1991, 10, 33–36.
15. Wang, S.-C. Artificial Neural Network. In *Interdisciplinary Computing in Java Programming*; Springer US: Boston, MA, USA, 2003; pp. 81–100.
16. Namin, A.H.; Leboeuf, K.; Wu, H.; Ahmadi, M. Artificial Neural Networks Activation Function HDL Coder. In *Proceedings of the 2009 IEEE International Conference on Electro/Information Technology*, Windsor, ON, Canada, 7–9 June 2009; pp. 389–392.
17. Basterretxea, K.; Tarela, J.M.; Del Campo, I. Approximation of Sigmoid Function and the Derivative for Hardware Implementation of Artificial Neurons. *IEE Proc. Circuits Devices Syst.* 2004, 151, 18–24.
18. Han, X.; Hou, M. Neural Networks for Approximation of Real Functions with the Gaussian Functions. In *Proceedings of the Third International Conference on Natural Computation*, Haikou, China, 24–27 August 2007; pp. 601–605.
19. Kwon, S.J. Artificial Neural Networks. *Artif. Neural Netw.* 2011, 1–426.
20. Sajja, P.S.; Akerkar, R. Bio-Inspired Models for Semantic Web. In *Swarm Intelligence and Bio-Inspired Computation*; Elsevier: Amsterdam, The Netherlands, 2013; pp. 273–294.
21. Suk, H., II. *An Introduction to Neural Networks and Deep Learning*, 1st ed.; Elsevier Inc.: Philadelphia, PA, USA, 2017.
22. Ben Nasr, M.; Chtourou, M. A Self-Organizing Map-Based Initialization for Hybrid Training of Feedforward Neural Networks. *Appl. Soft Comput.* 2011, 11, 4458–4464.
23. Sazlı, M.H. A Brief Review of Feed-Forward Neural Networks. *Commun. Fac. Sci. Univ. Ank. Ser. A2-A3* 2006, 50, 11–17. Available online: <https://pdfs.semanticscholar.org/5a45/fb35ffad65c904da2edeef443dcf11219de7.pdf> (accessed on 3 May 2020).
24. Wythoff, B.J. Backpropagation Neural Networks: A Tutorial. *Chemom. Intell. Lab. Syst.* 1993, 18, 115–155.
25. Saikia, P.; Baruah, R.D.; Singh, S.K.; Chaudhuri, P.K. Artificial Neural Networks in the Domain of Reservoir Characterization: A Review from Shallow to Deep Models. *Comput. Geosci.* 2019, 2019, 104357.
26. Zupan, J. Basics of Artificial Neural Network. In *Nature-inspired Methods in Chemometrics: Genetic Algorithms and Artificial Neural Networks*, 1st ed.; Leardi, R., Ed.; Elsevier Science: New York, NY, USA, 2003; Volume 23, pp. 199–229.
27. Poznyak, T.I.; Chairez Oria, I.; Poznyak, A.S. Background on Dynamic Neural Networks. In *Ozonation and Biodegradation in Environmental Engineering*; Elsevier: Amsterdam, The Netherlands, 2019; pp. 57–74.

28. Szandała, T. Comparison of Different Learning Algorithms for Pattern Recognition with Hopfield's Neural Network. *Procedia Comput. Sci.* 2015, 71, 68–75.
29. Neapolitan, R.E.; Neapolitan, R.E. *Neural Networks and Deep Learning*. *Artif. Intell.* 2018, 389–411.
30. Seker, S.; Ayaz, E.; Türkcan, E. Elman's Recurrent Neural Network Applications to Condition Monitoring in Nuclear Power Plant and Rotating Machinery. *Eng. Appl. Artif. Intell.* 2003, 16, 647–656.
31. Yang, H.H.; Murata, N.; Amari, S.I. *Statistical Inference: Learning in Artificial Neural Networks*. *Trends Cogn. Sci.* 1998, 2, 4–10.
32. Guyon, I. *Neural Networks and Applications Tutorial*. *Phys. Rep.* 1991, 207, 215–259.
33. Suárez Gómez, S.L.; Santos Rodríguez, J.D.; Iglesias Rodríguez, F.J.; de Cos Juez, F.J. Analysis of the Temporal Structure Evolution of Physical Systems with the Self-Organising Tree Algorithm (SOTA): Application for Validating Neural Network Systems on Adaptive Optics Data before on-Sky Implementation. *Entropy* 2017, 19, 103.
34. Pakkanen, J.; Iivarinen, J.; Oja, E. The Evolving Tree—a Novel Self-Organizing Network for Data Analysis. *Neural Process. Lett.* 2004, 20, 199–211.
35. Milone, D.H.; Sáez, J.C.; Simón, G.; Rufiner, H.L. Self-Organizing Neural Tree Networks. In *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, HongKong, China, 1 November 1998*; pp. 1348–1351.
36. Wen, W.X.; Jennings, A.; Liu, H. Learning a Neural Tree. In *Proceedings International Joint Conference on Neural Networks*; Citeseer: Berkeley, CA, USA, 1992.

---

Retrieved from <https://encyclopedia.pub/entry/history/show/2188>