



## Article

# A Formalism of the General Mathematical Expression of Multilayer Perceptron Neural Networks

Castro Gbêmémali Hounmenou<sup>1,3,\*</sup>  and Kossi Essona Gneyou<sup>2</sup> and Romain Glèlè Kakai<sup>3</sup> 

<sup>1</sup> Institut de Mathématiques et de Sciences Physiques, University of Abomey-Calavi, Dangbo, (Benin); castrohounmenou@gmail.com (H.G.C.)

<sup>2</sup> Laboratoire de Modélisations Mathématiques et Applications, University of Lome, Lome (Togo); kgneyou@gmail.com (G.E.K.)

<sup>3</sup> Laboratoire de Biomathématiques et d'Estimations Forestières, University of Abomey-Calavi, Abomey-Calavi (Benin); glele.romain@gmail.com (G.K.R.)

\* Correspondence: castrohounmenou@gmail.com (H.G.C.)

**Abstract:** Neural networks models are mostly represented by oriented graphs where only the components, constitutive elements of the graph, are transcribed into mathematical expression. Indeed, accurate knowledge of the full expression of the model is required in certain situations such as selecting among several reference models, the one that best fits the available data or comparing the explanatory and predictive performance of an established model with respect to some reference models. In this paper, we establish a formalism of the mathematical expression for multilayer perceptron neural network in a general framework, MLP- $p-n-q$ , with  $p$ ,  $n$  and  $q$  natural integers and show its restriction to cases where one has a hidden layer and multivariate outputs (MLP- $p-1-q$ ), and then a single output (MLP- $p-1-1$ ). Then, we give some specific cases of the most commonly used models. An application case is presented in the context of solving a nonlinear regression problem.

**Keywords:** Expression of multilayer network models, oriented graph, multivariate model, nonlinear regression

## 1. Introduction

In statistical analysis framework, attempt to reproduce wide variety of situations even the rarest is sorted in order to improve practices and risk management [1]. For example, in agriculture, to anticipate the impact of climate change on cotton productivity, one can vary the climatic and pest parameters for extreme situations, and see the yield behavior, and thus analyze its consequences on the life of producers and the local economy from the expression (1) developed by the Agricultural Research Centre for Cotton and Fibers of the Republic of Benin [2] :

$$\begin{aligned}\hat{\phi}(\mathbf{x}) = & -24550 + 0.6339x_1 - 839\log(x_2) - 298x_3 + 218.3x_4 + \\ & + 743.4x_5 + 16450x_6^2 - 0.2115\log(x_7) - 163.5x_8^2\end{aligned}\quad (1)$$

where the input variables are  $\mathbf{x} = (x_1 = \text{Precipitation}, x_2 = \text{Temperature}, x_3 = \text{Average humidity}, x_4 = \text{Insulation}, x_5 = \text{Evapotranspiration}, x_6 = \text{Average Helicoverpa density per hectare}, x_7 = \text{Average pest density per hectare and } x_8 = \text{Number of treatments})$  and the output estimate variable is  $\hat{\phi}(\mathbf{x}) = (\text{Cotton yield})$ . On the other hand, data can be artificially generated from an existing model and can be used for a particular purpose [3]. The set of these generated data can be exploited as a database to develop another model or to serve in simulation studies. For instance, knowing the distribution of the input variables of the equation (1), we can generate  $\hat{\phi}(\mathbf{x})$  and create a database of dimension  $(100.000 \times 8)$ . In addition, one can consider an established model and then compare its explanatory and predictive performance with respect to one or more reference models. Also, it may be necessary to compare several reference models and select the one that best fits the data available. In forestry, [4] tested five reference models to predict the relationship between wood density ( $WD$ ) of *A. auriculiformis* and tree diameter ( $D$ ): Linear model :

$WD = aD + b$  [5]; polynomial of the second degree :  $WD = aD^2 + bD + c$  [6]; exponential :  $WD = aD \exp(-bD)$  [7]; asymptotic by Michaelis-Menten :  $WD = \frac{aD}{1+bD}$  [7] and again exponential :  $WD = a \exp(bD)$  [7] where  $a$ ,  $b$  and  $c$  are the parameters of the models. The appropriate model that best fits this data was selected based on the performance of criteria such as the Akaike Information Criterion (AIC) [8], the coefficient of determination and the overall significance of the model coefficients.

However, multilayer perceptron neural networks (MLPs), one of connexionist methods, are oriented graphs and are widely used to solve real-world tasks. These models do not offer this flexibility, especially when there is at least one hidden layer and more than one output. Only their graphical form is more abundant in the literature sometimes with mathematical expression of the components [9,10]. The full expression of the functional form of three-layer MLP models with a single output is nevertheless available [11–14]. This limits the use of MLPs for simulation studies, or for the generation of artificial data from an existing model or for the comparison of several reference models from real data.

[15] have shown that the performances of two networks whose synaptic weights differ only slightly can have very different performances after learning. This is explained by the fact that the two choices of weights can be on either side of a dividing line between two basins of different minima. To bridge this gap we propose to establish a formalism of the mathematical expression of neural networks with multiple hidden layers and multivariate outputs. We show its restriction to the special cases that derive from it and give some special cases of the most commonly used models.

In Section 2, we give some definitions in the setup and establish our main results. In Section 3, we apply to a numerical study and Section 4 is a conclusion.

## 2. Artificial neural network models, Definitions and Main Results

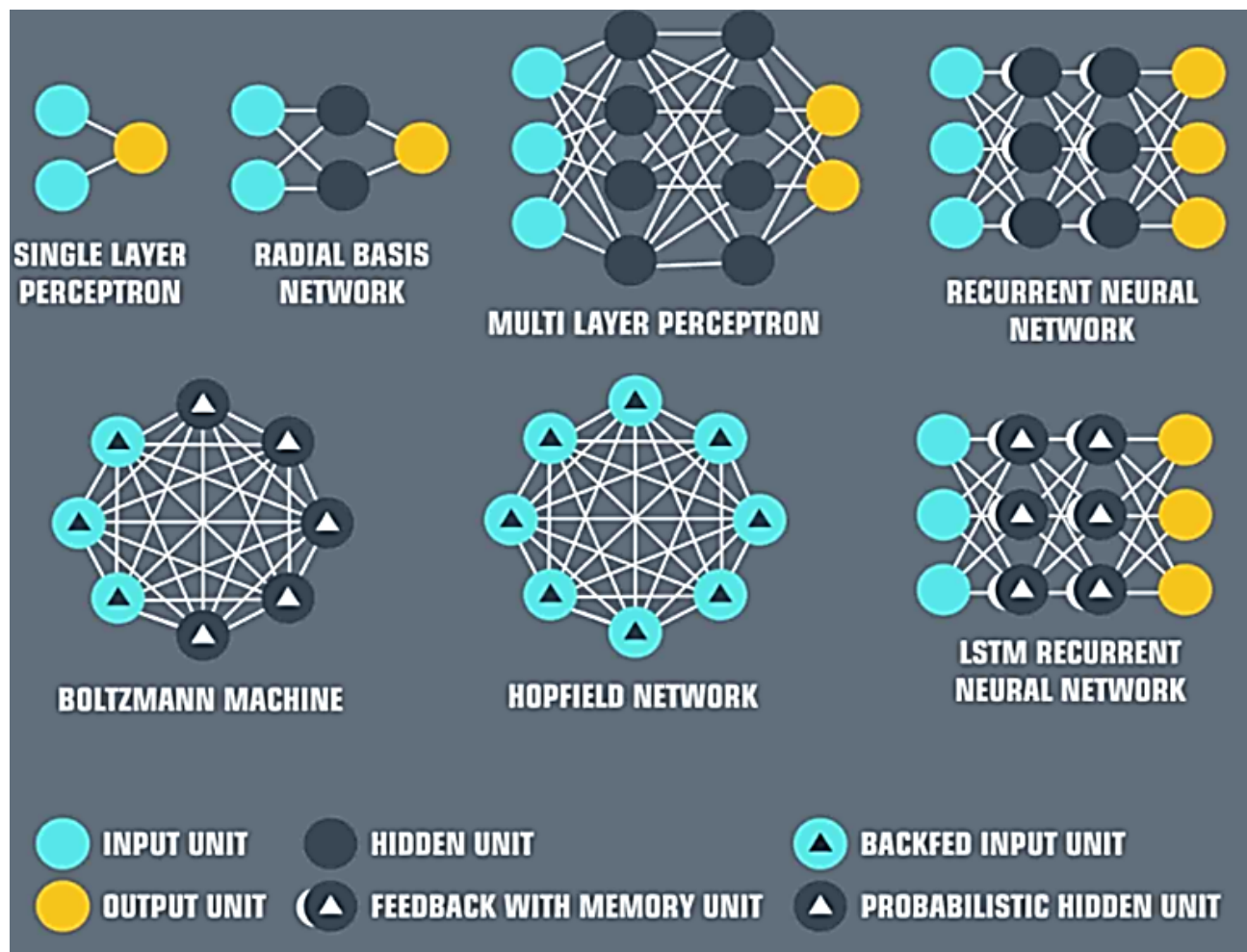
Artificial neural networks (ANNs), a subset of artificial intelligence methods are mathematical models inspired by the neurosciences and function like the human brain to learn complex things. They are widely used as a framework for advanced tasks such as prediction [16], pattern recognition [17] and many other applications in various disciplines [18]. This framework functions in a similar way to the brain, receives certain inputs, processes them and produces certain outputs [19] (see Fig.1 ).

Let  $\mathcal{X}$  be the domain of observations or inputs  $\mathbf{x} \in \mathbb{R}^p$ ,  $p \in \mathbb{N}$ . The labels, also called responses or outputs  $\mathbf{y}$  associated with the inputs take their values from a set denoted  $\mathcal{Y}$ . Two cases are to be distinguished: regression tasks if  $\mathcal{Y} \subseteq \mathbb{R}^q$ ,  $q \in \mathbb{N}^*$  (it is multivariate framework with  $q > 2$  responses, bivariate with  $q = 2$  responses and simple with  $q = 1$  response) and classification tasks when  $\mathcal{Y}$  is discrete, i.e. isomorphic to a finite subset of  $\mathbb{N}$ . The pairs  $(\mathbf{x}, \mathbf{y})$  are assumed to be independent and identically distributed (i.i.d.) draws of a joint probability distribution  $\mathbb{P}_{(\mathbf{X}, \mathbf{Y})}$  characterizing the phenomenon studied, where  $\mathbf{X}$  represents a random vector associated with the observations and  $\mathbf{Y}$  is the random vector corresponding to the multivariate responses. The distribution  $\mathbb{P}_{(\mathbf{X}, \mathbf{Y})}$  is fixed but unknown. Then, on the basis of these considerations, an artificial neural network is a mathematical function  $\mathbf{F}_\theta$  depending on a set of parameters  $\theta \in \Theta$  that associates to any point  $\mathbf{x}$  of the input space  $\mathcal{X}$  a point  $\mathbf{y}$  of the output space  $\mathcal{Y}$  such as:

$$\begin{aligned} \mathbf{F}_\theta &: \mathcal{X} \rightarrow \mathcal{Y} \\ \mathbf{x} &\mapsto \mathbf{F}_\theta(\mathbf{x}). \end{aligned} \quad (2)$$

There are several types of neural network models including multilayer perceptron (MLP), see Fig.1.

MLP are oriented graphs made up of a set of units (neurons) organized in successive layers (first layer is called input layer; last layer, output layer and middle layer(s), hidden



**Figure 1.** An Overview of artificial neural network models

layer(s)), which are interconnected with the weights (model parameters). The connections are always directed from lower layers to upper and the neurons of a same layer are not interconnected as shown in figure 2. They are denoted MLP- $p$ - $n$ - $q$ , where  $p$ ,  $n$  and  $q$  are positive natural numbers, which represent the number of inputs, the number of hidden layers and the number of responses respectively.

Figure 2(a) represents the architecture of a MLP-4-2-3, i.e. a MLP with 4 inputs, 2 hidden layers, 3 outputs, with synaptic weights (theoretical parameters), 2(b) represents the architecture of a MLP with 3 inputs, 2 hidden layers and 3 outputs (MLP-3-2-3) without weight representation [20] and 2(c) represents a MLP-4-2-1 with weight values obtained after learning [21].

Classically, a neuron can be defined as a neural network unit, still called a single perceptron, it is a real value parametrized function. It is characterized by its state, its connections with other neurons and its transfer functions. Its characteristics and functioning are displayed in figure 3. A layer of neurons is the juxtaposition of several neurons sharing the same inputs but each having their own vector of coefficients and their own output.

We have the following theorem which gives the general expression of MPL- $p$ - $n$ - $q$ .

**Theorem 1.** Let  $\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_h, \dots, \mathbf{f}_{n+1}$  be vectors of real value smooth functions, not necessarily linear from  $\mathbb{R}^{c_h}$  to  $\mathbb{R}^{c_h}$ . Let  $n$  be the total number of hidden layers,  $c_h$  the


$$\begin{aligned} \mathbf{F} \cdot (\mathbf{x}) &= \mathbf{f}_{n+1} \left( \mathbf{w}^{(n+1)} \mathbf{f}_n \left( \mathbf{w}^{(n)} \mathbf{f}_{n-1} \left( \mathbf{w}^{(n-1)} \dots \mathbf{f}_h (\mathbf{w}^{(h)} \dots \mathbf{f}_2 (\mathbf{w}^{(2)} \mathbf{f}_1 (\mathbf{w}^{(1)} \mathbf{x} + \right. \right. \right. \\ &\quad \left. \left. \left. + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \dots + \mathbf{b}^{(h)}) \dots + \mathbf{b}^{(n-1)}) + \mathbf{b}^{(n)} \right) + \mathbf{b}^{(n+1)} \right). \end{aligned} \quad (3)$$

$\mathbf{F} \cdot (\mathbf{x}) = (F_{1,\theta_1}(\mathbf{x}), \dots, F_{c_{n+1},\theta_{c_{n+1}}}(\mathbf{x}))$  is a vector of  $c_{n+1}$  outputs and the number of parameters  $\mathcal{N} = (\mathbf{w}^{(h)}, \mathbf{b}^{(h)}, 1 \leq h \leq n+1)$  is given by:

$$n_\theta = \sum_{\substack{1 \leq i \leq n \\ 2 \leq j \leq n+1 \\ i \leq j}} c_i c_j + \sum_{j=1}^{n+1} c_j \quad (4)$$

Figure 2 the smallest unit in the construction of graphs of multilayer perceptron neural networks (MLPs) is the neuron. The network is fully connected, so each unit receives connections from all units in the previous layer. This means that each unit has its own bias, and there is a weight for each pair of units in two consecutive layers. Thus the proof of Theorem 1 will be done in 3 steps :

In **Step 1** we classically use the definition of a neuron and give its mathematical expression, then we show its behavior in any layer of the network except the input layer.

In **Step 2** we give the mathematical expression for a layer of neurons in the network except the input layer and

in **Step 3** we give the mathematical expression for  $n$  hidden layers and a multivariate output.

Proof of Theorem 1 :

**Step 1:** We consider any neuron  $i$  of the first hidden layer (see Fig 3),  $\mathbf{x} \in \mathbb{R}^p$  is a realization of the random vector  $\mathbf{X}$  with  $p$  components ( $p \in \mathbb{N}, 1 \leq j \leq p$ );  $w_{ij}$  are the synaptic weights and represent the connection between the neuron  $i$  and the neurons of the previous layer. The activation of the neuron  $i$ ,  $s_i \in \mathbb{R}$  is obtained by inner product of the vector  $\mathbf{x} \in \mathbb{R}^{p+1}$  and the vector of weights  $\mathbf{w}_i = (w_{i0}, \dots, w_{ij}, \dots, w_{ip}) \in \mathbb{R}^{p+1}$ :

$$s_i = \sum_{j=0}^p w_{ij} x_j = \sum_{j=1}^p w_{ij} x_j + b_{i0} \quad (5)$$

with  $b_{i0} = w_{i0}$  called bias and  $x_0 \equiv 1$ . The activation function or transfer function,  $f$  is a real value function, continuous, not necessarily non-linear and class  $C^\infty$  defined from  $\mathbb{R}$  to  $\mathbb{R}$  and applied to equation (5) gives:

$$y_i = f(s_i) = f(\mathbf{w}_i \mathbf{x}) \quad (6)$$

$f(s_i)$  is called the state of the neuron  $i$  i.e. the output of the neuron  $i$  and can be used as input for the other neurons of the next layer or for the label  $y$ .

on other hand, suppose we are on any other hidden layer noted  $h$ , except the first hidden layer, the state of the neuron  $i$  on this layer noted  $f_h(s_i^{(h)})$  is written by analogy to the equations (5) and (6) as:

$$f_h(s_i^{(h)}) = \sum_{j=0}^{c_h} w_{ij}^{(h)} f_{h-1}(s_j^{(h-1)}) \quad (7)$$

with  $f_h$  and  $f_{h-1}$  are activation functions;  $c_h$  is the number of neurons of the layer  $h$ ;  $h-1$  is the number of the layer that precedes  $h$ ;  $s_i^{(h)}$  is the activation of the neuron  $i$ .

Knowing the expression of a neuron  $i$  in any layer  $h$  of the network, except the input layer, we give the expression of a layer of  $c_h$  neurons.



**Step 2 :** Let  $p = c_0$  and  $c_1$  be two natural integers, we note  $\mathbf{w} = (\mathbf{w}_1, \dots, \mathbf{w}_{c_1}) \in \mathbb{R}^{c_1(c_0+1)}$  with  $\forall i \in \{1, \dots, c_1\}, \mathbf{w}_i = (w_{i1}, \dots, w_{ic_0}; b_i) \in \mathbb{R}^{c_0+1}$ . A layer  $h = 1$  of  $c_1$  neurons and of  $c_0$  inputs is a function :  $\mathbf{f}_1 : \mathbb{R}^{c_1(c_0+1)} \times \mathbb{R}^{c_0} \longrightarrow \mathbb{R}^{c_1}$  verifying :  $\forall i \in \{1, \dots, c_1\}, f_i$  is a neuron, and  $\forall \mathbf{w} \in \mathbb{R}^{c_1(c_0+1)}$  and  $\mathbf{x} \in \mathbb{R}^{c_0+1}$ ,

$$\mathbf{f}_1(\theta \mathbf{x}) = \left( f_1(\mathbf{w}_1 \mathbf{x}), \dots, f_i(\mathbf{w}_i \mathbf{x}), \dots, f_{c_1}(\mathbf{w}_{c_1} \mathbf{x}) \right) \quad (8)$$

$$= \mathbf{f}_1(\mathbf{w}^{(1)} \mathbf{x} + b^{(1)}) \quad (9)$$

In the equation (9),  $\mathbf{w}^{(1)} \in \mathcal{M}_{(c_1, c_0+1)}$  and  $b^{(1)} \in \mathbb{R}^{c_1}$ .

For any layer  $h \neq 1$  of  $c_h$  neurons with  $h \in \llbracket 2, n+1 \rrbracket$  and of  $c_{h-1}$  inputs is from the equation (7) and band by analogy with equation (9), the function  $\mathbf{f}_h$  is written in (10) :

$$\begin{aligned} \mathbf{f}_h : \mathbb{R}^{c_h} &\rightarrow \mathbb{R}^{c_h} \\ \mathbf{s}^{(h)} &\mapsto \mathbf{f}_h(\mathbf{s}^{(h)}) \end{aligned} \quad (10)$$

with  $\mathbf{f}_h(\mathbf{s}^{(h)}) = (f_h(s_1^{(h)}), \dots, f_h(s_{c_h}^{(h)}))$  and  $\mathbf{s}^{(h)} = (s_1^{(h)}, \dots, s_{c_h}^{(h)})$ ,  $f_h$  is the activation function of the  $h^{th}$  layer of the MLP from oriented graphs.

The combination of steps 1 and 2 gives the general mathematical expression for a MLP with  $n$  hidden layers and a multivariate output.

**Step 3 :** Consider a multilayer Perceptron neural network MPL- $p$ - $n$ - $q$  defined from  $\mathcal{X} \subseteq \mathbb{R}^p$  into  $\mathcal{Y} \subseteq \mathbb{R}^q$  with an input layer of  $p$  inputs,  $n$  hidden layers where each layer may have  $c_h$  hidden neurons and a output layer of  $q$  outputs. From steps 1 and 2, the general expression of a MPL- $p$ - $n$ - $q$  is a function of the form

$$\begin{aligned} \mathbf{F}(\mathbf{x}) = & \mathbf{f}_{n+1} \left( \mathbf{w}^{(n+1)} \mathbf{f}_n \left( \mathbf{w}^{(n)} \mathbf{f}_{n-1} \left( \mathbf{w}^{(n-1)} \dots \mathbf{f}_h(\mathbf{w}^{(h)} \dots \mathbf{f}_2(\mathbf{w}^{(2)} \mathbf{f}_1(\mathbf{w}^{(1)} \mathbf{x} + \right. \right. \right. \\ & \left. \left. \left. + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \dots + \mathbf{b}^{(h)}) \dots + \mathbf{b}^{(n-1)}) + \mathbf{b}^{(n)} \right) + \mathbf{b}^{(n+1)} \right) \end{aligned} \quad (11)$$

where  $n+1$  is the position of the last layer of the MLP neural network;  $c_h$  is the number of neurons in the layer  $h$ ; the number of hidden layers is between 1 and  $n$ ;  $\forall h \in \llbracket 1, n+1 \rrbracket, \mathbf{w}^{(h)} \in \mathcal{M}_{(c_h, c_{h-1})}$ ,  $b^{(h)} \in \mathbb{R}^{c_h}$ ,  $c_0 = p$ ,  $c_{n+1} = q$  and  $\mathbf{F}(x) = (F_{1, \theta_1}(x), \dots, F_{c_{n+1}, \theta_{c_{n+1}}}(x))$  is a vector of  $c_{n+1}$  outputs, with  $\mathcal{P} = (\mathbf{w}^{(h)}, \mathbf{b}^{(h)}, 1 \leq h \leq n+1)$ , the set of network parameters where  $\mathbf{w}^{(h)}$  are the weights and  $\mathbf{b}^{(h)}$ , the bias or thresholds of the  $h$ -th layer of MLPs. The number of MLP neural network parameters is given by:

$$n_\theta = \sum_{\substack{1 \leq i \leq n \\ 2 \leq j \leq n+1 \\ i \leq j}} c_i c_j + \sum_{j=1}^{n+1} c_j. \quad (12)$$

■

The following two corollaries give special cases of a MPL- $p$ -1- $q$  (MLP with multiple inputs, a single hidden layer and a multivariate output) and MPL- $p$ -1-1 (MLP with multiple inputs, a single hidden layer and single output).

**Corollary 1.** Under the conditions of Theorem 1, if  $n = 1$  and  $q \geq 2$ , we have a MPL-p-1-q defined from  $\mathcal{X} \subseteq \mathbb{R}^p$  into  $\mathcal{Y} \subseteq \mathbb{R}^q$  whose expression is given by :

$$\mathbf{F} \cdot (\mathbf{x}) = \mathbf{f}_2 \left( \mathbf{w}^{(2)} \mathbf{f}_1 (\mathbf{w}^{(1)} \mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)} \right) \quad (13)$$

where  $\mathbf{F} \cdot (\mathbf{x}) = (F_{1,\theta_1}(\mathbf{x}), \dots, F_{c_2,\theta_{c_2}}(\mathbf{x}))$  is a vector of outputs, with  $\mathcal{Z} = (\mathbf{w}^{(1)}, \mathbf{b}^{(1)}; \mathbf{w}^{(2)}, \mathbf{b}^{(2)})$  or

$$\theta = (w_{11}^{(1)}, \dots, w_{1c_0}^{(1)}, \dots, w_{c_1}^{(1)}, \dots, w_{c_1 c_0}^{(1)}; b_{10}^{(1)}, \dots, b_{c_1 0}^{(1)}; w_{11}^{(2)}, \dots, w_{1c_1}^{(2)}, \dots, w_{c_2 1}^{(2)}, \dots, w_{c_2 c_1}^{(2)}; b_{10}^{(2)}, \dots, b_{c_2 0}^{(2)})$$

and

$$n_\theta = c_1(c_0 + c_2 + 1) + c_2. \quad (14)$$

**Corollary 2.** Under the conditions of Theorem 1, if  $n = 1$  and  $q = c_{n+1} = 1$ , we have a MLP-p-1-1 defined from  $\mathcal{X} \subseteq \mathbb{R}^p$  into  $\mathcal{Y} \subseteq \mathbb{R}$  whose expression is given by:

$$F_\theta(\mathbf{x}) = f_2 \left( \mathbf{w}^{(2)} \mathbf{f}_1 (\mathbf{w}^{(1)} \mathbf{x} + \mathbf{b}^{(1)}) + b^{(2)} \right) \quad (15)$$

where  $\mathbf{F} \cdot (\mathbf{x}) = F_{1,\theta_1}(\mathbf{x})$  is a single output, with  $\mathcal{Z} = (\mathbf{w}^{(1)}, \mathbf{b}^{(1)}; \mathbf{w}^{(2)}, b^{(2)})$  or

$$\theta = (w_{11}^{(1)}, \dots, w_{1c_0}^{(1)}, \dots, w_{c_1}^{(1)}, \dots, w_{c_1 c_0}^{(1)}; b_{10}^{(1)}, \dots, b_{c_1 0}^{(1)}; w_{11}^{(2)}, \dots, w_{1c_1}^{(2)}; b_{10}^{(2)}) \text{ and}$$

$$n_\theta = c_1(c_0 + 2) + 1. \quad (16)$$

**Remark 1.** From most commonly used models, we deduce the following cases of specific models :

**Case 1 :** From equation (11), if  $p = c_0 (\geq 2)$ ,  $n = 0$  and  $q = c_1$ , then  $\mathbf{F} \cdot (\mathbf{x}) = \mathbf{f}_1(\mathbf{w}^{(1)} \mathbf{x} + \mathbf{b}^{(1)})$ . It follows that

(i) if  $\mathbf{f}_1$  is a single linear function and  $q = c_1 = 1$ , then  $F_\theta(\mathbf{x}) = \mathbf{w}^{(1)} \mathbf{x} + b^{(1)}$  is the multiple linear regression expression with  $n_\theta = c_0 + 1$ ;

(ii) If  $\mathbf{f}_1$  is a single sigmoid function and  $q = c_1 = 1$ , then

$$F_\theta(\mathbf{x}) = \frac{\exp(\mathbf{w}^{(1)} \mathbf{x} + b^{(1)})}{1 + \exp(\mathbf{w}^{(1)} \mathbf{x} + b^{(1)})}$$

is the binary logistic regression expression with  $n_\theta = c_0 + 1$ ;

(iii) If  $\mathbf{f}_1$  is a vector of  $c_1$  linear functions and  $q = c_1$ , then

$$\mathbf{F} \cdot (\mathbf{x}) = (F_{1,\theta_1}(\mathbf{x}), \dots, F_{i,\theta_i}(\mathbf{x}), \dots, F_{c_1,\theta_{c_1}}(\mathbf{x}))$$

is the multivariate linear regression expression with  $n_\theta = c_1(c_0 + 1)$  and

$$\forall i \in \llbracket 1, c_1 \rrbracket, F_{i,\theta_i}(\mathbf{x}) = \mathbf{w}_i^{(1)} \mathbf{x} + b_i^{(1)}$$

is the expression of linear regression function;

(iv) If  $\mathbf{f}_1$  is a vector of  $c_1$  softmax functions and  $q = c_1$ , then

$$\mathbf{F} \cdot (\mathbf{x}) = (F_{1,\theta_1}(\mathbf{x}), \dots, F_{i,\theta_i}(\mathbf{x}), \dots, F_{c_1,\theta_{c_1}}(\mathbf{x}))$$

is the multiclass logistic regression expression with  $n_\theta = c_1(c_0 + 1)$  and

$$\forall i \in \llbracket 1, c_1 \rrbracket, F_{i, \theta_i}(\mathbf{x}) = \frac{\exp(\mathbf{w}_i^{(1)} \mathbf{x} + b_i^{(1)})}{\sum_{l=1}^{c_n+1} \exp(\mathbf{w}_l^{(1)} \mathbf{x} + b_l^{(1)})}$$

is the expression of softmax functions.

**Case 2:** From equation (11), if  $p = c_0 \geq 2$ ,  $n = 1$  and  $q = c_2$ , then

$$\mathbf{F}_\bullet(\mathbf{x}) = \mathbf{f}_2 \left( \mathbf{w}^{(2)} \mathbf{f}_1(\mathbf{w}^{(1)} \mathbf{x} + b^{(1)}) + b^{(2)} \right).$$

It follows that

(a) if  $\mathbf{f}_2$  is a vector of single linear functions,  $\mathbf{w}^{(2)}$  is a matrix composed of only 1, then

$$\mathbf{F}_\bullet(\mathbf{x}) = (F_{1, \theta_1}(\mathbf{x}), \dots, F_{i, \theta_i}(\mathbf{x}), \dots, F_{c_2, \theta_{c_2}}(\mathbf{x}))$$

is multiclass of additive models with  $n_\theta = c_1(c_0 + c_2 + 1) + c_2$  and

$$\forall i \in \llbracket 1, c_2 \rrbracket, F_{i, \theta_i}(\mathbf{x}) = \sum_{i=1}^{c_1} f_1(\mathbf{w}_i^{(1)} \mathbf{x} + b_i^{(1)}) + b_i^{(2)}$$

is a class of additive models equivalent to projection pursuit models (see [22]);

(b) From (a), if  $b_i^{(1)} = 0$ ,  $p = c_0 = c_1$  and  $\mathbf{w}_{ij}^{(1)} = \delta_{ij}$  (the Kronecker of  $\delta$ ), then

$$F_{i, \theta_i}(\mathbf{x}) = \sum_{i=1}^{c_n} f_1(\mathbf{x}_i) + b_i^{(2)}$$

is a generalized additive model [23];

(c) if  $\mathbf{f}_2$  is a vector of single linear functions,  $\mathbf{f}_1$  is a vector of single sigmoid functions,  $n = 1$  and  $q = c_2$ , then

$$\mathbf{F}_\bullet(\mathbf{x}) = \mathbf{w}^{(2)} \mathbf{f}_1(\mathbf{w}^{(1)} \mathbf{x} + b^{(1)}) + b^{(2)}$$

with  $n_\theta = c_1(c_0 + c_2 + 1) + c_2$  which corresponds to the model discussed by [24] and used by [25]. This form of model is applied in many works and in various fields. When we want positive outputs, we apply the exponential function,  $\exp(\cdot)$  to  $\mathbf{f}_2$ .

**Remark 2.** Activation function also called transfer function determines the relationship between inputs and outputs of a node and a network. In general, the activation function introduces a degree of nonlinearity that is valuable for most ANN applications. [26] identifies general conditions for a continuous function to be qualified as an activation function. Basically, any differentiable function can be qualified as an activation function in theory. In practice, only a small number of "well-behaved" (bounded, monotonically increasing, and differentiable) activation functions are used. A few popular ones are highlighted in Table ???. There are some heuristic rules for the selection of the activation function. Generally, a network may have different activation functions for different nodes in the same or different layers. Yet almost all the networks use the same activation functions particularly for the nodes in the same layer. In addition, the choice of the activation function strongly depends on the problem we are trying to solve or what our network is trying to learn. The nonlinear activation functions are mostly included in designing of MLP models for solving complex problems.



Table 1: Usual activation functions

Name	Function $f$	Derivative of $f$ , $f'$	Range
<i>Identity</i>	$f(x) = x$	$f'(x) = 1$	$(-\infty, +\infty)$
<i>Sigmoid</i>	$f(x) = \frac{1}{1+\exp(-x)}$	$f'(x) = f(x)(1-f(x))$	$(0, 1)$
<i>Gaussian</i>	$f(x) = \exp(-x^2)$	$f'(x) = -2x \exp(-x^2)$	$(0, 1)$
<i>Softplus</i>	$f(x) = \ln(1 + \exp(x))$	$f'(x) = \frac{1}{1+\exp(x)}$	$(0; \infty)$
<i>Cos</i>	$f(x) = \cos(x)$	$f'(x) = -\sin(x)$	$(-1, 1)$
<i>Sin</i>	$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$(-1, 1)$
<i>ReLU</i>	$f(x) = x\mathbb{1}_{x>0}$	$f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases}$	$(0, \infty)$
<i>Tanh</i>	$f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	$f'(x) = 1 - [f(x)]^2$	$(-1, 1)$

*ReLU*: Rectified linear unit, *GELU*: Gaussian Error Linear Unit.

### 3. Numerical study

In this section data have been artificially generated from an existing MLP model and are related to a real world situation. The generation procedure is described in 8 steps (from a) to h), see below in Resolution). It can be used when developing a model and needing to test it with real data from a given domain, but not available. Also, it can be used in simulation studies. It serves to reproduce the complexity of the relationship between the original data and to get the data close enough to the real data sought.

#### Practical case:

Consider the multilayer perceptron neural network model (MLP-4-1-1) obtained in the Insect as Feed for West Africa project [27] to predict  $y = \text{food economic efficiency}$  as a function of  $\mathbf{x} = (x_1 = \text{dose with three modality } (0, 50 \text{ and } 100), x_2 = \text{age}, x_3 = \text{food consumption}, x_4 = \text{weight})$  to assess the effect of maggot meal on the growth and economic performance of *Guinea fowl*. The resulting model is:

$$\mathbb{E}(y_t) = \sum_{i=1}^{11} \mathbf{w}_i^{(2)} \frac{\exp(\langle \mathbf{x}_t, \mathbf{w}_i^{(1)} \rangle + b_i^{(1)}) - \exp[-(\langle \mathbf{x}_t, \mathbf{w}_i^{(1)} \rangle + b_i^{(1)})]}{\exp(\langle \mathbf{x}_t, \mathbf{w}_i^{(1)} \rangle + b_i^{(1)}) + \exp[-(\langle \mathbf{x}_t, \mathbf{w}_i^{(1)} \rangle + b_i^{(1)})]} + b^{(2)} \quad (17)$$

where  $y_t$  represents the  $t^{\text{th}}$  observation ( $t \in \llbracket 1, n \rrbracket, n \in \mathbb{N}^*$ );  $\mathbf{w}_i$  is the weight vector associated with the  $i$ -th neuron in the hidden layer;  $b_i$  is the bias of the  $i$ -th neuron in the hidden layer of MLP model. The optimal parameters are  $\theta = (\mathbf{w}^{(1)}, \mathbf{b}^{(1)}, \mathbf{w}^{(2)}, b^{(2)})$  with

$$\mathbf{w}^{(1)} = \begin{bmatrix} -0.06 & -0.87 & 0.33 & -0.10 & -0.15 & 0.08 & -0.13 & 0.60 & -0.07 & 0.04 & -0.17 \\ -0.06 & 0.41 & -0.38 & 0.29 & -0.18 & -0.31 & 0.22 & -0.44 & -0.12 & 0.16 & 0.28 \\ -0.13 & 0.47 & -0.16 & -0.27 & 0.22 & 0.20 & -0.36 & 1.42 & -0.13 & 0.27 & -0.48 \\ 0.22 & 0.35 & 0.21 & 0.03 & 0.15 & 0.01 & 0.27 & 0.55 & 0.32 & -0.43 & 0.37 \end{bmatrix}$$

$$\mathbf{b}^{(1)} = \begin{bmatrix} 0.31 & 0.65 & 0.45 & -0.82 & -0.39 & -0.38 & -0.86 & -0.01 & -0.79 & 0.79 & -0.43 \end{bmatrix};$$

$$\mathbf{w}^{(2)} = \begin{bmatrix} 0.01 & -0.38 & 0.15 & -0.03 & -0.05 & 0.04 & -0.04 & 0.61 & -0.01 & -0.02 & -0.05 \end{bmatrix};$$

$$b^{(2)} = -0.82.$$

In vector form  $\theta = (-0.06, -0.06, -0.13, 0.22, -0.87, 0.41, 0.47, 0.35, 0.33, -0.38, -0.16, 0.21, -0.10, 0.29, -0.27, 0.03, -0.15, -0.18, 0.22, 0.15, 0.08, -0.31, 0.20, 0.01, -0.13, 0.2, -0.36, 0.27, 0.60, -0.44, 1.42, 0.55, -0.07, -0.12, -0.13, 0.32, 0.04, 0.16, 0.27, -0.4, -0.17, 0.28, -0.48, 0.37; 0.31, 0.65, 0.45, -0.82, -0.39, -0.38, -0.86, -0.01, -0.79, 0.79, -0.43; 0.01, -0.38, 0.15, -0.03, -0.05, 0.04, -0.04, 0.61, -0.01, -0.02, -0.05; -0.82)$  and the total number of parameters is  $n_\theta = 67$ .

These parameters are obtained under certain configurations, namely: number of hidden neurons = 11, learning rate = 0.5, training data size:  $n_A = 54$  and test data size is  $n_T = 18$ , normalization method = min-max, algorithm used = standard back-propagation, and the evaluation criteria values based on the test data are : coefficient of determination,  $R^2 = 64.03$ , root mean square error  $RMSE = 0.01$  and correlation coefficient,  $r = 80.02$ .

In the context of this practical case study, it is intended to generate data almost identical to the unavailable real data.

### Resolution:

In this study, the theoretical model used is the model (15) with the activation function in the hidden layer, the hyperbolic tangent function and the identity function in the output layer. Based on equation (17), then by taking into account the parameters  $\theta$  and the distribution of the input variables:  $X_1 = \text{resampling techniques}$ ,  $X_2 \sim \mathcal{N}(\mu = 4.5, \sigma^2 = 2.30)$ ,  $X_3 \sim \mathcal{N}(\mu = 29.95, \sigma^2 = 13.04)$  and  $X_4 \sim \mathcal{N}(\mu = 239.76, \sigma^2 = 117.11)$ , we obtain the output  $y$  for a size  $n \in \mathbb{N}^*$  by programming under the R language.

The following steps are used:

- Generation of the input variables according to their respective distribution of size  $n$ .
- Creation of a data matrix for the input variables  $\mathbf{x}$ .
- Normalization of the input data using the Min-Max normalization method.
- Computation of the inner product for each input variable and the weights associated with a neuron  $i \in \llbracket 1, 11 \rrbracket$ ,  $\langle \mathbf{x}_t, \mathbf{w}_i^{(1)} \rangle + b_i^{(1)}$ .
- Application of the activation function, hyperbolic tangent for all hidden layer neurons,
 
$$\mathbf{Z}_{t,i} = \frac{\exp(\langle \mathbf{x}_t, \mathbf{w}_i^{(1)} \rangle + b_i^{(1)}) - \exp[-(\langle \mathbf{x}_t, \mathbf{w}_i^{(1)} \rangle + b_i^{(1)})]}{\exp(\langle \mathbf{x}_t, \mathbf{w}_i^{(1)} \rangle + b_i^{(1)}) + \exp[-(\langle \mathbf{x}_t, \mathbf{w}_i^{(1)} \rangle + b_i^{(1)})]}.$$
- Computation of the inner product between the hidden neuron outputs and the weights associated with the output neuron,  $y_t = \langle \mathbf{Z}_t, \mathbf{w}^{(2)} \rangle + b^{(2)}$ .
- Denormalization of the output variable  $y_t$
- Reconstitution of the database  $(\mathbf{X}_t, y_t)_{t \in \llbracket 1, n \rrbracket}$ .

The first ten rows obtained are in the table below :

Table 2: Data obtained close enough to the real data

$x_1$	$x_2$	$x_3$	$x_4$	$y$
100	3.93	52.87	281.77	2.80
100	3.95	37.26	238.46	2.80
0	3.94	24.04	129.60	4.15
0	3.97	19.10	226.19	4.48
0	3.93	14.73	144.31	4.36
50	3.96	16.05	268.13	3.62
50	3.98	9.55	72.86	4.29
50	3.91	45.03	282.61	3.06
50	3.88	40.79	268.85	2.89
0	3.93	33.42	242.00	3.70

$y = (\text{food economic efficiency in FCFA/kg of feed})$ ,  $\mathbf{x} = (x_1 = \text{dose in } \%, x_2 = \text{age in week}, x_3 = \text{food consumption in grams/dr}, x_4 = \text{weight in Kg})$

The MLP-4-1-1 model used in this way was able to simulate data fairly close to the guinea fowl zoo-economy, taking into account the complexity between these data and can be used to test other models.

#### 4. Conclusion

In this paper, we have proposed a general expression for multilayer perceptron neural network (MLP) models. From this general form, we have given some specific cases of the most commonly used models. This expression can be used not only instead of oriented graphs to present MLP models in scientific work, but also in simulation studies. A procedure for generating artificial data from an existing MLP model is given. A practical case was carried out involving the artificial generation of data from an existing MLP model in the field of poultry breeding and in the context of nonlinear regression. These artificial data are fairly close to the original data and retain the complexity of the relationship between them.

**Author Contributions:** Conceptualization, H.G.C.; methodology, H.G.C., G.E.K. and R.G.K.; software, H.G.C.; validation, H.G.C., G.E.K. and R.G.K.; formal analysis, H.G.C.; resources, R.G.K.; writing—original draft preparation, H.G.C.; writing—review and editing, H.G.C., G.E.K. and R.G.K.; visualization, H.G.C.; supervision, G.E.K. and R.G.K.; project administration, R.G.K.; funding acquisition, R.G.K. All authors read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The authors confirm that the data supporting the findings of this work are available within the article.

**Acknowledgments:** The authors are very grateful to the Editor-in-Chief, the Associate Editor and the two anonymous Reviewers for the careful review and valuable comments on earlier versions of this paper. He's work is supported by African Centre of Excellence in Mathematics and Applications (CEA-SMA). We thank this institution and its donors.

**Conflicts of Interest:** The authors declare no conflict of interest.

#### References

1. Rubinstein, R.Y.; Melamed, B. *Modern simulation and modeling*; Vol. 7, Wiley New York, 1998.
2. CRA-CF. Rapport de l'étude de la modélisation des rendements du Coton sur les sites de Alafiarou, Angaradebou et Gogounou. Centre de recherches agricoles coton et fibres (CRA-CF) de l'Institut national des recherches agricoles du Bénin (INRAB), 2008, pp. 1–56.
3. Kroese, D.P.; Taimre, T.; Botev, Z.I.; Rubinstein, R.Y. *Solutions manual to accompany simulation and the Monte Carlo Method*; 2007; pp. 1–188.
4. Tounouewa, J. Caractérisation du bois de *Acacia auriculiformis* et potentiel de valorisation comme bois d'œuvre au Bénin, Afrique de l'Ouest. PhD thesis, École Doctorale des Sciences Agronomiques et de l'Eau, Université de Parakou, 2020.
5. Silva, J.P.M.; Fernandes, M.R.d.M.; Gonçalves, A.F.A.; Silva, G.F.d.; Cabacinha, C.D.; others. Estimation of the Basic Wood Density of Native Species Using Mixed Linear Models. *Floresta e Ambiente* **2019**, *26*.
6. Githiomi, J.; Kariuki, J. Wood basic density of *Eucalyptus grandis* from plantations in central rift valley, Kenya: variation with age, height level and between sapwood and heartwood. *Journal of Tropical Forest Science* **2010**, pp. 281–286.
7. Oddi, F.J.; Miguez, F.E.; Ghermandi, L.; Bianchi, L.O.; Garibaldi, L.A. A nonlinear mixed-effects modeling approach for ecological data: Using temporal dynamics of vegetation moisture as an example. *Ecology and evolution* **2019**, *9*, 10225–10240.
8. Hu, S. Akaike information criterion. *Center for Research in Scientific Computation* **2007**, *93*.
9. Koacadagli, O. Bayesian Learning based Gaussian Approximation for Artificial Neural Networks. *Turkish Journal of Forecasting* **2017**, *1*, 54–65.
10. Rynkiewicz, J. Estimation and test for multi-dimensional regression models. *Communications in Statistics-Theory and Methods* **2007**, *36*, 2655–2671.
11. Hounmenou, C.G.; Tohou, R.; Gneyou, K.E.; Glèlè Kakaï, R. Empirical determination of optimal configuration for characteristics of a multilayer perceptron neural network in nonlinear regression. *Afrika Statistika* **2020**, *15*, 2413–2429.
12. Li, M.; Wang, J. An empirical comparison of multiple linear regression and artificial neural network for concrete dam deformation modelling. *Mathematical Problems in Engineering* **2019**, *2019*.
13. Pasini, A. Artificial neural networks for small dataset analysis. *Journal of thoracic disease* **2015**, *7*, 953.
14. Shirvany, Y.; Hayati, M.; Moradian, R. Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations. *Applied Soft Computing* **2009**, *9*, 20–29.
15. Kolen, J.; Pollack, J. "Backpropagation is sensitive to initial Conditions" TR 90-JK-BPSIC **1990**.
16. Zhang, G.; Patuwo, B.E.; Hu, M.Y. Forecasting with artificial neural networks: The state of the art. *International journal of forecasting* **1998**, *14*, 35–62.
17. Egmont-Petersen, M.; de Ridder, D.; Handels, H. Image processing with neural networks-a review. *Pattern recognition* **2002**, *35*, 2279–2301.

18. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938.
19. Bengio, Y.; Goodfellow, I.; Courville, A. *Deep learning*; Vol. 1, MIT press Massachusetts, USA, 2017.
20. Mahmoodi, M.; Naderi, A. Applicability of artificial neural network and nonlinear regression to predict mechanical properties of equal channel angular rolled Al5083 sheets. *Latin American Journal of Solids and Structures* **2016**, *13*, 1515–1525.
21. Adisa, O.M.; Botai, J.O.; Adeola, A.M.; Hassen, A.; Botai, C.M.; Darkey, D.; Tesfamariam, E. Application of artificial neural network for predicting maize production in South Africa. *Sustainability* **2019**, *11*, 1145.
22. Friedman, J.H.; Stuetzle, W. Projection pursuit regression. *Journal of the American statistical Association* **1981**, *76*, 817–823.
23. Hastie, T.J.; Tibshirani, R.J. *Generalized additive models*; Vol. 43, CRC press, 1990.
24. Barron, A.R. Complexity regularization with application to artificial neural networks. In *Nonparametric functional estimation and related topics*; Springer, 1991; pp. 561–576.
25. Nychka, D.; Ellner, S.; Gallant, A.R.; McCaffrey, D. Finding chaos in noisy systems. *Journal of the Royal Statistical Society: Series B (Methodological)* **1992**, *54*, 399–426.
26. Chen, T.; Chen, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks* **1995**, *6*, 911–917.
27. IFWA. Assess the effect of maggot meal flour on the growth and economic performances of *Guinea fowl*. Insect as Feed for West Africa project in Laboratoire de Recherche Avicole et de Zoo-Economie, Abomey-Calavi, Bénin, 2017.