SYNTHETIC ERROR SIGNAL PROPAGATION IN MULTI-LAYER PERCEPTRON SYSTEMS

Dejan Đukić*
[0000-0001-7581-148X]

Alfa University,
Belgrade, Serbia

Abstract:

Today's artificial intelligence (AI) systems usually take the form of the artificial neural networks (ANN). One of the earliest ANN structures is the well-known multilayer perceptron. Determination of the values of the parameters of the multilayer perceptron is often performed by the error gradient descent method, called the gradient backpropagation method. In this method, the gradient of the criterion function, usually the sum of squares of output errors, is computed layer by layer from the output towards the input. In the classical form of this method, the gradient values for the parameters of the hidden and the input layers depend on the previously computed gradient values. This may lead to either the gradient values of deeper layers decreasing towards zero (the gradient collapse), or growing to very large values (the gradient explosion). In either case, the parameter optimisation becomes impossible. In this work, this problem has been addressed by decoupling the error gradients with respect of the parameters in one layer from the gradient values in other layers. This has been achieved by estimating the output error for each layer, here called the layer synthetic error, and then computing the corresponding gradient values for each layer separately, thus decoupling the optimisation of the parameters of one layer from that of another layer. The synthetic error method has been compared to the conventional backpropagation method in an experiment. The goal of the experiment has been the training of the ANN on real-world data, using both methods. The results obtained in the experiment are very encouraging, as the synthetic error method has shown some clear advantages.

Keywords:

Multi-Layer Perceptron, Backpropagation, Synaptic Parameter Training, Deep Learning, Artificial Intelligence.

INTRODUCTION

For the past several decades, the human activities rely increasingly on technologies with the common name of artificial intelligence (AI). Nowadays, the AI tasks are predominantly performed by bio-mimetic technologies, amongst which the most important is the technology of the artificial neural networks (ANN) [1], [2], [3], [4]. In the ANNs, data transformations are performed by simple interconnected processing elements. Each processing element, the artificial neuron, performs some elementary algebraic transformation and passes its result to the neighbouring processing elements.

Correspondence:
Dejan Đukić

e-mail:
dejan.djukic@alfa.edu.rs

DOI: 10.15308/Sinteza-2026-247-253

The knowledge in the artificial neural network is, as is the case in biological networks, stored in the form of interconnections between the processing elements, and, in particular, in the values of the parameters regulating the passing of the results between the neurons. The process of training the ANN for a particular task is achieved by optimising the values of the parameters of the network.

Artificial neural networks come in many forms and variants. One of the very first ANN structures is the multilayer perceptron network. It has been proposed already in 1943 [5], and it may be considered to be the foundational structure for all the subsequent developments in this domain. The processing element in the multi-layer perceptron network is the perceptron. The transformation performed by the perceptron is the computation of a weighted sum of the input values, and applying a non-linear function to the computed sum [6], [7]. Multi-layer perceptron networks have the ability to approximate functions with arbitrary accuracy, and today they have been in use in a wide range of applications [8], [9], and [10].

The process of network learning determines the values of the network parameters, which are the weighting values of the inputs to the perceptron. The weighting coefficients of the inputs to artificial neurons are called the synaptic coefficients, and the non-linear function transforming the sum is the activation function, which reflects the vocabulary of the biological neural networks. The activation function has initially been the hard limiter (step) function, but this activation function has an inconvenient property of preventing the use of the classical optimisation methods, and in particular the gradient descent method. Thus, smooth versions of the activation function, amenable to the optimisation methods, have been proposed [11], [12].

In the gradient backpropagation method, the gradient of the criterion function, usually the sum of squares of output errors, is computed layer by layer from the output towards the input [13]. The method has proven to be effective in parameter optimisation, however, it has some serious drawbacks when applied to multi-layer perceptron networks. Namely, the gradient values for the parameters of the hidden and the input layers depend on the previously computed gradient values [14]. This may lead to either the gradient values of deeper layers decreasing towards zero (the gradient collapse [15], [16]), or growing to very large values (the gradient explosion [17], [18]). In either case, the parameter optimisation becomes impossible.

In this article, the problem of the mentioned drawbacks of the gradient backpropagation method (the gradient collapse and the gradient explosion) has been addressed. The main idea is that the computation of the error gradients with respect to the parameters ought to be decoupled from one layer to another. This has been achieved by estimating the output errors for each layer. These estimated error values are here called the synthetic errors. The synthetic errors are estimated by a new subtle modification of the conventional gradient backpropagation method. When the values of the synthetic errors for each layer are known, the classical gradient descent method may be applied to optimise the parameters of one layer independently of the gradient values for other layers. With the help of the synthetic error values, the gradient optimisation is confined only to the currently treated layer. This prevents the issues of the gradient collapse and the gradient explosion. Although producing an estimate of the synthetic error requires additional computational effort, the benefit of applying the method of the synthetic error propagation in the training multilayer perceptron networks is that the parameter optimisation becomes faster and more reliable.

Section 2 of this work briefly explains the conventional gradient back-propagation method. Section 3 follows with a presentation of the synthetic error propagation modification of the gradient backpropagation. In Section 4, the results of a numerical experiment have been presented. In the experiment performed, the two methods, the conventional back-propagation and the synthetic error propagation, have been compared. The goal of the experiment has been a training of the ANN for the task of time series prediction, using both of these two methods. The data set used in the experiment is the widely known time series of the number of sunspots observed during the past two centuries. The results obtained in the experiment are very encouraging, as the synthetic error method has shown some clear advantages in the shortening the training time, and in the reducing the prediction error. In the concluding section, the performance of the synthetic error propagation method has been critically evaluated, and some directions have been given regarding the future investigations and steps for its enhancement.

2. GRADIENT BACK-PROPAGATION

A single perceptron performs a simple transformation of its inputs.

$$y = \sigma(s) = \sigma(\sum_j a_j x_j - a_0) \quad (1)$$

where x_j is the vector of inputs, y is the perceptron output, σ is the activation function of the perceptron, s is the weighted sum of inputs, called the synaptic sum, a_j is the vector of synaptic coefficients, one per input signal, and a_0 is the biasing coefficient of the synaptic sum.

The gradient back-propagation method is presented here for a three-layer perceptron network, though it is clear that it may be extended to networks with an arbitrary number of layers.

Let a three-layer perceptron network have $m = n(0)$ inputs, $n(1)$ neurons in the first layer, $n(2)$ neurons in the second layer, and $n = n(3)$ neurons in the third, i.e., the output layer. Thus, the input layer has $m = n(0)$ inputs and $n(1)$ outputs, the second layer has $n(1)$ inputs and $n(2)$ outputs, and the third layer has $n(2)$ inputs and $n = n(3)$ outputs. The outputs of the layers have been computed using the following formulae:

$$x_i^{(1)} = \sigma^{(1)}\left(s_i^{(1)}\right) = \sigma^{(1)}\left(\sum_{j=1}^{n(0)} a_{ij}^{(1)} x_j - a_0^{(1)}\right) \quad (2)$$

$$x_i^{(2)} = \sigma^{(2)}\left(s_i^{(2)}\right) = \sigma^{(2)}\left(\sum_{j=1}^{n(1)} a_{ij}^{(2)} x_j^{(1)} - a_0^{(2)}\right) \quad (3)$$

$$y_i = x_i^{(3)} = \sigma^{(3)}\left(s_i^{(3)}\right) = \sigma^{(3)}\left(\sum_{j=1}^{n(2)} a_{ij}^{(3)} x_j^{(2)} - a_0^{(3)}\right) \quad (4)$$

For each set of input values x fed to the network, a corresponding set of the required network output values $\hat{y}$ has been compared to the output values produced by the network y. For each output, an error value has been computed as $e_i = \widehat{y_i} - y_i$. The criterion function for the optimisation is the sum of the squares of the error values of the outputs, $C = \sum_{i=1}^{n} e_i^2$, and the goal of the optimisation is its minimisation.

The gradient values for the parameters of the third layer, i.e. the output layer, are computed readily as:

$$\frac{\partial C}{\partial a_{ij}^{(3)}} = 2e_i \frac{\partial \sigma^{(3)}\left(s_i^{(3)}\right)}{\partial s_i^{(3)}} x_j^{(2)} \quad (5)$$

$$\frac{\partial C}{\partial a_{i0}^{(3)}} = -2e_i \frac{\partial \sigma^{(3)}\left(s_i^{(3)}\right)}{\partial s_i^{(3)}} \quad (6)$$

Going deeper into the layers, the gradient values get more and more complicated:

$$\frac{\partial C}{\partial a_{ij}^{(2)}} = 2\sum_{p=1}^{n(3)} e_p \frac{\partial \sigma^{(3)}\left(s_p^{(3)}\right)}{\partial s_p^{(3)}} a_{pi}^{(3)} \frac{\partial \sigma^{(2)}\left(s_i^{(2)}\right)}{\partial s_i^{(2)}} x_j^{(1)} \quad (7)$$

$$\frac{\partial C}{\partial a_{i0}^{(2)}} = -2\sum_{p=1}^{n(3)} e_p \frac{\partial \sigma^{(3)}\left(s_p^{(3)}\right)}{\partial s_p^{(3)}} a_{pi}^{(3)} \frac{\partial \sigma^{(2)}\left(s_i^{(2)}\right)}{\partial s_i^{(2)}} \quad (8)$$

$$\frac{\partial C}{\partial a_{ij}^{(1)}} = 2\sum_{p=1}^{n(3)} e_p \frac{\partial \sigma^{(3)}\left(s_p^{(3)}\right)}{\partial s_p^{(3)}} \sum_{q=1}^{n(2)} a_{pq}^{(3)} \frac{\partial \sigma^{(2)}\left(s_q^{(2)}\right)}{\partial s_q^{(2)}} a_{qi}^{(1)} \frac{\partial \sigma^{(1)}\left(s_i^{(2)}\right)}{\partial s_i^{(1)}} x_j \quad (9)$$

$$\frac{\partial C}{\partial a_{i0}^{(1)}} = -2\sum_{p=1}^{n(3)} e_p \frac{\partial \sigma^{(3)}\left(s_p^{(3)}\right)}{\partial s_p^{(3)}} \sum_{q=1}^{n(2)} a_{pq}^{(3)} \frac{\partial \sigma^{(2)}\left(s_q^{(2)}\right)}{\partial s_q^{(2)}} a_{qi}^{(1)} \frac{\partial \sigma^{(1)}\left(s_i^{(2)}\right)}{\partial s_i^{(1)}} \quad (10)$$

Evidently, the expressions for computing the gradient values of each deeper layer include the expressions for the computation of the gradient values of the previous layers, i.e. of the layers nearer to the output. Once the trend in the gradient values is set in a direction, decrease or increase, it will have a tendency, respectively, of decreasing or increasing more and more. These decreases or increases in the gradient values carry the names of the gradient collapse and the gradient explosion.

The optimisation of the network parameters, in its simplest form, is just an iterative adjustment of the value of the corresponding parameter, using the computed gradient values. E.G.

$$a_{ij}^{(3)}(t+1) = a_{ij}^{(3)}(t) - \Delta^{(3)}(t) \frac{\partial C}{\partial a_{ij}^{(3)}}(t) \quad (11)$$

where $\Delta^{(3)}(t)$ is the iteration step. The expressions for the adjustment of other parameters are analogous and have been omitted here for the reasons of brevity.

3. SYNTHETIC ERROR PROPAGATION

In a manner similar to that presented in the previous section, it is also possible to compute the gradient of the criterion function with respect to the layer's input values. Then, it is possible to use these gradient values to adjust the input values of that layer in order to reduce the criterion function. The guiding idea here is that the layer parameters may be better adjusted when the correct input and output values have been presented to the layer. The adjusted layer inputs are thought to be more correct than the inputs present before the optimisation step, and this could lead to a faster parameter optimisation process.

The gradient of the criterion function with respect to the input signals to layer 3, i.e. to the output layer, is:

$$\frac{\partial C}{\partial x_j^{(2)}} = 2\sum_{i=1}^{n(3)} e_p \frac{\partial \sigma^{(3)}\left(s_i^{(3)}\right)}{\partial s_i^{(3)}} a_{ij}^{(3)} \quad (12)$$

The value of signal $x_j^{(2)}$ should then be decreased by the computed adjustment value:

$$e_j^{(2)} = \Delta_e^{(2)} \frac{\partial C}{\partial x_j^{(2)}} \quad (13)$$

These adjustment values are the values of the synthetic error estimates of the inputs to layer 3. As the inputs to layer 3 are the outputs produced by layer 2, it is possible to adjust the parameters of layer 2 by using (5) and (6) and the computed synthetic error (13). These expressions are simpler and do not depend on the values of the gradients of other layers. The synthetic error estimates of layer 2 form a new criterion function $C^{(2)} = \sum_{i=1}^{n(2)} e_i^{(2)}$, from which it is possible to estimate the gradients of the inputs to layer 2.

$$\frac{\partial C^{(2)}}{\partial a_{ij}^{(2)}} = 2e_i^{(2)} \frac{\partial \sigma^{(2)}\left(s_i^{(2)}\right)}{\partial s_i^{(2)}} x_j^{(1)} \tag{14}$$

$$\frac{\partial C^{(2)}}{\partial a_{ij}^{(2)}} = -2e_i^{(2)} \frac{\partial \sigma^{(2)}\left(s_i^{(2)}\right)}{\partial s_i^{(2)}} \tag{15}$$

Clearly, the values of the gradients of the criterion function of layer 2 do not depend on the gradients of the output criterion function. The values of parameters $a_{ij}^{(2)}$ are adjusted by applying an expression similar to (11).

$$a_{ij}^{(2)}(t+1) = a_{ij}^{(2)}(t) - \Delta^{(2)}(t) \frac{\partial C^{(2)}}{\partial a_{ij}^{(3)}}(t) \tag{16}$$

Progressing through the layers towards the input, we see that the inputs to layer 2 are the outputs of layer 1. The gradient of the criterion function of layer 2 with respect to the inputs of layer 2, i.e. with respect to the outputs of layer 1, are:

$$\frac{\partial C^{(2)}}{\partial x_j^{(1)}} = 2 \sum_{i=1}^{n(2)} e_p \frac{\partial \sigma^{(2)}\left(s_i^{(2)}\right)}{\partial s_i^{(2)}} a_{ij}^{(2)} \tag{17}$$

With this gradient value, the error of the outputs of layer 1 has been estimated, and the value of signal $x_j^{(1)}$ has been modified by the adjustment value:

$$e_j^{(1)} = \Delta_e^{(1)} \frac{\partial C^{(2)}}{\partial x_j^{(1)}} \tag{18}$$

This is the synthetic error estimate of the outputs of layer 1. As has been the case in previous layers, the sum of squares of the synthetic errors of layer 1, form the criterion function for layer 1, $C^{(1)} = \sum_{i=1}^{n(1)} e_i^{(1)}$. With this value, again, it is possible to use (5) and (6) and (18) to compute the derivatives of the criterion function with respect to the parameters of layer 1, and then to use this value to adjust the values of the synaptic coefficients of layer 1.

$$\frac{\partial C^{(1)}}{\partial a_{ij}^{(1)}} = 2e_i^{(1)} \frac{\partial \sigma^{(1)}\left(s_i^{(1)}\right)}{\partial s_i^{(1)}} x_j \tag{19}$$

$$\frac{\partial C^{(1)}}{\partial a_{ij}^{(1)}} = -2e_i^{(1)} \frac{\partial \sigma^{(1)}\left(s_i^{(1)}\right)}{\partial s_i^{(1)}} \tag{20}$$

In case when the synthetic errors of the layer outputs have been correctly estimated, it is supposed that the values of the synaptic coefficients adjusted layer by layer using the synthetic error will converge faster to their optimal values.

Here, it has to be mentioned that, although there is a clear analytical justification for the use of the gradient descent in parameter optimisation, it is, in fact, an iterative approximation method. The synthetic error method proposed here does not replace completely this iterative approximation process, but it has been designed to improve its performance. The advantages of the synthetic error propagation method have been shown experimentally in the following section.

4. EXPERIMENTAL RESULTS

In order to test the claims announced in the introduction, a numerical experiment has been conducted. The data set used in the experiment is the well-known time-series of monthly observed numbers of sunspots, collected during a bit more than two centuries. The graphic view of this data has been represented in Figure 1 - Monthly number of sunspots. The data set has been scaled and translated to fit the limits of 0.1 and 0.9, as required by the unipolar sigmoid activation function, which has been used in a multi-layer perceptron network employed in the experiment.

The experimental goal has been the prediction of the number of sunspots for 12 months in advance, from the observation of the number of sunspots during the previous year (12 months). The structure of the neural network used is a three-layer perceptron network, having 12 inputs and one output. The number of hidden neurons has been left as an experimental variable.

In the notation of section 2, we have $m = n(0) = n(1) = 12$, $n(2)$ has been left open, and $n = n(3) = 1$. In all the network layers, the activation function has been the unipolar sigmoid function:

$$\sigma^{(1)}(x) = \sigma^{(2)}(x) = \sigma^{(3)}(x) = \frac{1}{1+e^{-x}}$$

In all cases, the iteration step of the gradient method has been chosen to be equal for all the layers for both methods, the gradient back-propagation and the synthetic error propagation. That is: $\Delta^{(1)}(t) = \Delta^{(2)}(t) = \Delta^{(3)}(t) = 0.1$. For the synthetic error method, the layer output adjustment step has been chosen to be much larger, and equal for all the layers: $\Delta^{(1)}(t) = \Delta_e^{(2)}(t) = 1$.

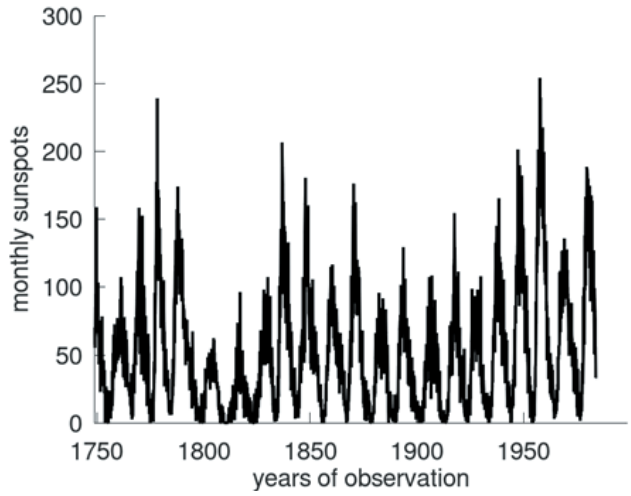

Figure 1. Monthly number of sunspots

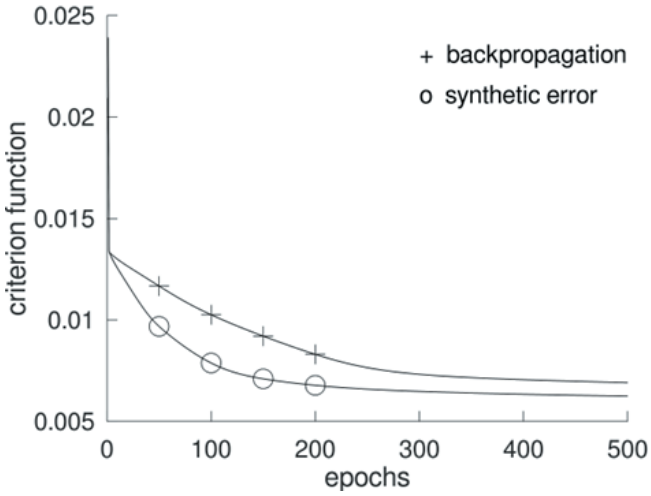

Figure 2. Network criterion function for 9 hidden neurons

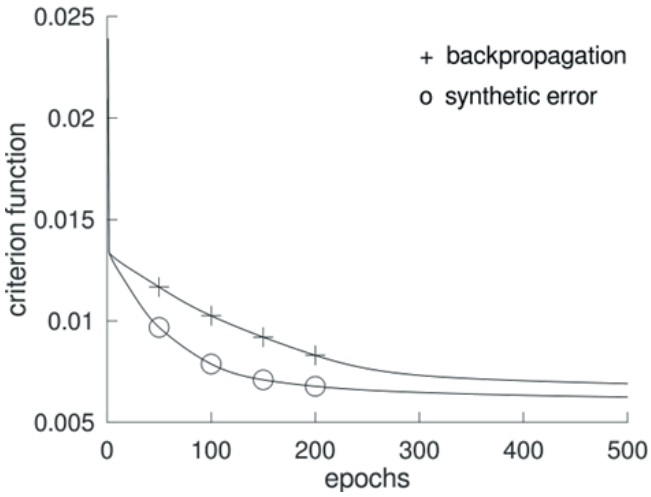

Figure3. Network criterion function for 13 hidden neurons

Two perceptron networks, one for each of the methods, and for various numbers of neurons in the hidden layer, have been trained using supervised learning on the dataset. During one learning epoch, all the data of the dataset have been presented to the network. During an epoch, the synaptic coefficients of the network have been adjusted according to the method applied to that network, the gradient backpropagation or the synthetic error propagation. At the end of each epoch, the average of the criterion function during the epoch has been computed. This value is presented in the figures in this section.

The trials have consistently shown a clear reduction of the criterion function value when the synthetic error method has been compared to the conventional back-propagation method. This has been shown in the cases of three-layer perceptron networks with 9 hidden neurons, shown in Figure 2 - Network criterion function for 9 hidden neurons., with 13 hidden neurons, shown in Figure 3 - Network criterion function for 13 hidden neurons., and also in the case of a network with 20 hidden neurons, as shown in Figure 4 - Network criterion function for 20 hidden neurons. Although these are only preliminary results of this research topic, they are extremely encouraging. The synthetic error method has shown clear advantages, achieving lower values of the criterion function for a lower number of epochs of network training.

5. CONCLUSION

A new idea for improving the gradient backpropagation algorithm, widely used in supervised training of multi-layer perceptron networks, has been presented in this work. It has been provisionally named the synthetic error propagation. The essence of this new method is that it estimates the errors of the outputs of the network layers preceding the output layer, i.e. the hidden and the input layers. This error is used to compute the criterion function for each layer separately, and from this its gradient with respect of the parameters of that layer. The parameter optimisation, i.e. the network training, is being performed within each layer separately, which prevents the problems of gradient collapse and of gradient explosion. The idea has been tested on a well-known data set, the monthly sunspot counts, and a multi-layer perceptron network has been trained to predict the number of sunspots from past observations. The results of the application of the synthetic error propagation have been compared to the results obtained by the conventional gradient back-propagation method. It may clearly be observed that a shortening of the number of epochs of network training takes place when the synthetic error method is being used.

There is still a considerable amount of research to be completed in order to confirm the viability of the idea, to find its best modalities and its limitations in application. Some of the topics that may lead to an improvement are trials on multi-layer perceptron networks with multiple hidden layers, the application of some sort of forward-backward traversal in the process of optimisation,

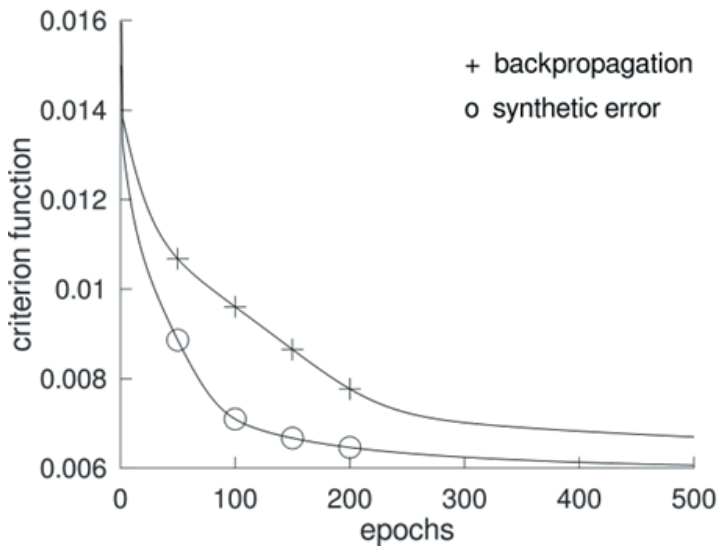

Figure 4. Network criterion function for 20 hidden neurons

and finding the right compromise between the accuracy of the synthetic error estimation and the rate of decrease of the criterion function. Despite the fact that there is still a lot of work to be done, the author strongly believes in the usefulness of the synthetic error propagation approach and presents the work performed so far with confidence.

REFERENCES

[1] S. Popović, S., L. Kopanja, S. Đukić Popović, and D. Djukic, "Neural networks and their application in object recognition," in *Proc. 12th Int. Conf. on Applied Internet and Information Technologies (AIIT 2022)*, Zrenjanin, 2022 , pp. 40-47.

[2] W. Lan, J. Dang, Y. Wang, S. Wang, " Pedestrian detection based on yolo network model," in *Proc. 2018 IEEE International Conference on Mechatronics and Automation, ICMA 2018*, 2018, pp. 1547–1551, doi: 10.1109/ICMA.2018.8484698.

[3] S. Popovic, S. D. Popovic, N. Denic, D. Djukic, and J. Stojanovic, "Risk Management Innovations through Neural Network Integration in Automated Boiler Combustion Systems," in *J. Soft Comput. and Decis. Anal.*, vol. 3, no. 1, pp. 129-135, 2025, doi: 10.31181/jscda31202565.

[4] H. Li, J. Li and X. Han, "Robot Vision Model Based on Multi-Neural Network Fusion," in *2019 IEEE 3rd Inf. Techol. Netw. Electron. Autom. Conf. (ITNEC)*, 2019, pp. 2571–2577, doi: 10.1109/itnec.2019.8729210.

[5] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," in *Bull. Math. Biohys.*, vol. 5, no. 4. pp. 115-133, 1943.

[6] F. Blayo and M. Verleysen, *Les réseaux de neurones artificiels*, Paris, Presses universitaires de France, 1996.

[7] D. Rumelhart, G. Hinton and R. Williams, "Learning representations by back-propagating errors," in *Nature*, vol. 323, 1986, pp. 533–536, doi: 10.1038/323533a0.

[8] W. Zhang, Z. Yin, Z. Sheng, Y. Li, W. Ouyang, X. Li and B. Cui, "Graph attention multi-layer perceptron," In *Proc. 28th ACM SIGKDD Conf. on knowl. discovery and data mining*, Aug. 2022, pp. 4560-4570, doi: 10.1145/3534678.3539121.

[9] J. Gaudart, B. Giusiano and L. Huiart, "Comparison of the performance of multi-layer perceptron and linear regression for epidemiological data," in *Comput. statist. and data anal.*, vol. 44, no. 4, 2024, pp. 547-570, doi: 10.1016/S0167-9473(02)00257-8.

[10] P. Coulibaly, F. Anctil and B. Bobée, "Prévision hydrologique par réseaux de neurones artificiels: état de l'art," in *Canadian J. of civil eng.*, vol.26, no. 3, 1999, pp. 293-304, doi: 10.1139/l98-069.

[11] D. Djukic and S. Popovic, "Bilateral exponential as sigmoid in multilayered neuronal networks," in *Proc. Alfatech 2024*, Belgrade, Serbia, 2024, doi: 10.5281/zenodo.12685976.

[12] R. Kruse, S. Mostaghim, C. Borgelt, C. Braune and M. Steinbrecher, "Multi-layer perceptrons," in *Computational intelligence: a methodological introduction*, Springer International Publishing, 2022, pp. 53-124, doi: 10.1007/978-3-030-42227-1_5.

[13] S. Sapna, A. Tamilarasi and M. P. Kumar, "Backpropagation learning algorithm based on Levenberg Marquardt Algorithm," in *Comp. Sci. Inform. Technol. (CS and IT)*, vol. 2, 2012, pp. 393-398, doi: 10.5121/csit.2012.2438.

[14] M. Rashid, M. A. Khan, M. Alhaisoni, S. H. Wang, S. R. Naqvi, A. Rehman and T. Saba, "A sustainable deep learning framework for object recognition using multi-layers deep features fusion and selection," in *Sustainability*, vol.12, no. 12, 2020, pp. 5037, doi: 10.3390/SU12125037.

[15] D. Beaglehole, P. Súkeník, M. Mondelli and M. Belkin, "Average gradient outer product as a mechanism for deep neural collapse," in *Advances in Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 130764-130796, doi: 10.52202/079017-4156.

[16] J. Zhou, C. You, X. Li, K. Liu, S. Liu, Q. Qu and Z. Zhu, "Are all losses created equal: A neural collapse perspective," in *Advances in Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 31697-31710, doi: 10.52202/068431.

[17] Z. Zhang, C. Luo and J. Yu, "Towards the gradient vanishing, divergence mismatching and mode collapse of generative adversarial nets," in *Proc. of the 28th ACM Int. Conf. on Inf. and Knowl. Manage.*, Nov. 2019, pp. 2377-2380, doi: 10.1145/3357384.3358081.

[18] S. Kanai, Y. Fujiwara and S. Iwamura, "Preventing gradient explosions in gated recurrent units," in *Advances in Neural Inf. Process. Syst.*, vol. 30, 2017.