How to Design Multiplierless Neural Networks for Deep Learning?

Maja Lutovac Banduka and Miroslav Lutovac

April 27, 2024

How to Design Multiplierless Neural Networks for Deep Learning?

Maja Lutovac Banduka
RT-RK, Computer Based Systems
Belgrade, Serbia
majamlutovac@gmail.com

Miroslav Lutovac
Academy of Engineering Sciences of Serbia
Belgrade, Serbia
lutovac@gmail.com

Abstract— **The paper is an extended version of the paper accepted at MECO2024 with a procedure for designing neural networks. The accepted paper describes the efficient implementation of neural networks based on symbolic analysis. The main advantage is the reduction of processing latency by replacing general-purpose multipliers with a small number of summing components.**

Keywords - symbolic analysis; sensitivity; probability

I. INTRODUCTION

A symbolic analysis of classical neural networks for deep learning is presented in [1]. A modification of the analysis and an extended version will be published in [2]. The main advantage of [1] and [2] is that the sensitivity of the decision is very low in relation to the values of the adjustment coefficients. This property is due to a very large number of coefficients in relation to the number of derived classes.

The low sensitivity of the derived class implies that it is not necessary to calculate the precise values of the system parameters; of course, at the end of the training process. A larger quantization step can be used for the final coefficient values at the end of training. We expect to detect a class with a probability greater than 40% and much higher probability than the probability of any other class.

The final values of the system parameters can be implemented with a small number of adders. This means that instead of general purpose multipliers we can use adders. This is also important because the delay of each multiplier is reduced to the delay of the adder component.

II. TYPE OF COMPONENTS

A. *The Simplest Multiplier Component without Adder*

The simplest component is a binary shift and 0. Multiplication by ½ is just a binary shift of one bit. A binary shift of two bits is in the case of multiplication by ¼. A binary shift of three bits is in the case of multiplication by 1/8, while the binary shift is four bits in the case of multiplication by 1/16. We can use to multiply with a binary shift by two bits, in a cascade of another binary shift by two bits.

B. *The Multiplier Implemented as Two-input Adder*

A multiplication component implemented as a two-input adder can be used to implement this type of multiplication component. Note that the input to the adder can be +1 or -1 for a two-input adder.

C. *The Multiplier Implemented with Two Additions*

A multiplication component implemented as a three-input adder can be used to implement this type of multiplication component. A multiply component implemented as a cascaded two-input adder can be used to implement this type of multiply component. Note that the input to the adder can be either +1 or -1.

D. *The Multiplier Implemented as Two Three-input Adders*

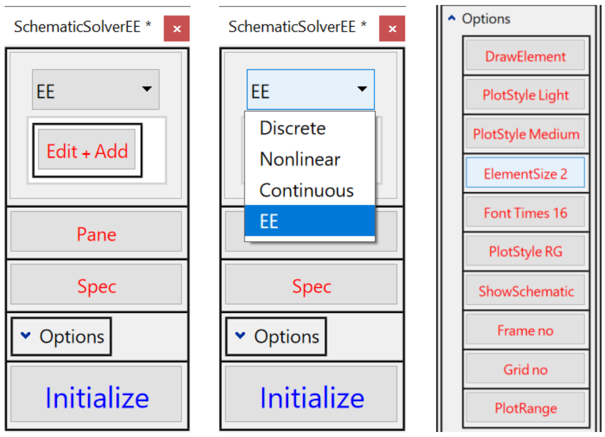
A multiplying component implemented as two cascaded three-input adders can be used to implement this type of multiplying component.

III. Published Part of the Method

The main motivation for this work is the problem of using a higher language environment as used in [3] and [4]. Sometimes, the built-in classification function starts with a neural network model, but the model output is not successful. We use the application package [5] that runs in the higher language environment.

A. GUI for Generating Network

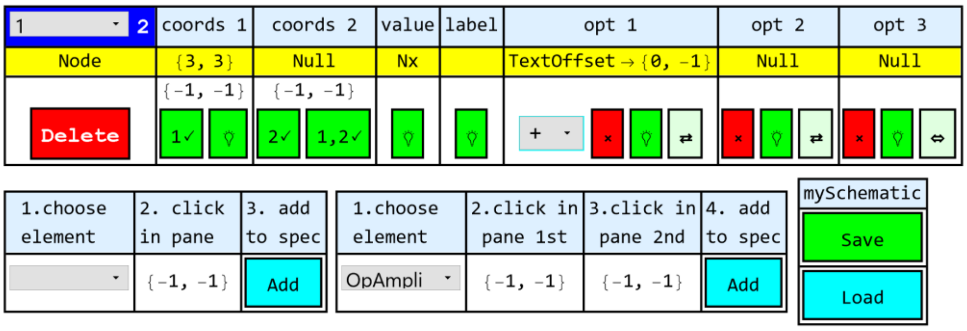
The GUI starts by initializing the primary palette as presented in [5], which calls drawing knowledge. Different palettes can be used for different systems, as illustrated in Fig. 1. Element options can also be changed using the primary palette extension.



a) b) c)

Figure 1. Palette a) main palette, b) selection of different systems, c) selection of different options.

In Fig. 2, we can see GUI functions, one to change element properties and another to add a new element between two nodes.



1 ▾ 2	coords 1	coords 2	value	label	opt 1	opt 2	opt 3
Node	{3, 3}	Null	Nx		TextOffset → {0, -1}	Null	Null
Delete	{-1, -1} 1✓	{-1, -1} 2✓ 1,2✓			+ ▾		

1.choose element	2. click in pane	3. add to spec
▾	{-1, -1}	Add

1.choose element	2.click in pane 1st	3.click in pane 2nd	4. add to spec
OpAmpli ▾	{-1, -1}	{-1, -1}	Add

mySchematic
Save
Load

Figure 2. GUI for adding a new element and modifying element parameters.

The original software is embedded with additional functionality, to moving elements to the left or to the right, and up and down, to rotate elements by 90^0, 180^0, or 270^0. Also, any element of the schematic description can be easily replaced by another element. The system solution is obtained in the time or frequency domain by simply clicking a button. Although the system in [3] represents a more complex schematic electro-technical system, the same procedure can be used for continuous-time systems, discrete-time systems, or nonlinear systems. Of course, it is not possible to use the transformed method to solve nonlinear systems.

Note that the software is written using a computer algebra system that is free on small computers. The original code can be changed and adopted according to the user's wishes.

B. Embedded Knowledge to Transform Elements

The original software was embedded with new functionality in [6]. Instead of an existing element specification in the netlist, as in [5], being replaced by another element with a simple change of element name, one element specification can be replaced by multiple components. In this way, the schematic description is adapted to a specific further analysis, such as a small or large signal model, where the frequency transformation is used to solve the system in the frequency domain or in the time domain including clipping signal distortions or saturation regions.

C. *Embedded Knowledge for Nonlinear Systems*

Embedded knowledge is generally used for nonlinear complex systems with many repeating blocks. The initial drawing of one neuron is copied several times in one layer, thus providing everything needed by many neurons in the layer. Then the entire layer is copied several times to generate a more complex neural network as presented in [1]. All system parameters are represented by symbols. Due to its complexity, the system cannot be solved manually. We can use some known system parameters as derived in [3]. For numerical values from [3], we prove that this symbolic solution produces the same class probability.

The symbolic system response allows simple replacement of the activation function with a known or unknown pure function. Using the parameters obtained by the random number generator, we prove that all classes are equally possible. The most likely system parameters can be used if we know the preferred class.

D. *Sensitivity analysis*

The sensitivity analysis of system parameters is further done in [2]. The result is that all parameters can be varied without significantly affecting the class probabilities.

Just to repeat what is used for automatic analysis. Only a simple diagram of a neuron is drawn. Then the scheme is enlarged according to the prescribed number of classes or neurons in one layer and the number of layers. The minimum number of layers is three. Two layers are hidden and one layer is for output.

The software used is an original application package written in a higher symbolic language. Then, additional knowledge is added according to the requested tasks.

A schematic description is not just a pictorial representation of the system. The transformation of the schematic description into many different forms can be used as a system netlist, a mathematical description of the system's response to symbolic inputs, or to generate a software or hardware implementation. All system parameters are defined as symbols, which can be symbolically replaced by numbers at any time when numerical values are needed, such as plotting responses.

E. *MECO 2024 paper*

Finally, we reach the MECO 2024 paper.

The numerical values from [3] describe the mathematical approach of a neural network for deep learning, representing the system in a matrix format. All numerical values are used as a starting point for this new schematic symbolic approach of MECO 2024 work [7].

The motive of this work is to find a general purpose multiplier implementation with a small number of adder components. The netlist of the neural network is created without multipliers with symbolic values of the system.

F. *A New Part of the Method*

The new part of the previous method is to identify the most suitable combination of binary shifts.

This means that we are trying to identify some kind of general structure of the adder. In this paper, we propose the following form for the value of the multiplier:

$$\left(\left\{\begin{matrix}0\\ \pm\frac{1}{2}\end{matrix}\right. + \left\{\begin{matrix}0\\ \pm\frac{1}{8}\end{matrix}\right. + \left\{\begin{matrix}0\\ \pm\frac{1}{16}\end{matrix}\right.\right)\left(\left\{\begin{matrix}0\\ 1\end{matrix}\right. + \left\{\begin{matrix}0\\ \pm\frac{1}{4}\end{matrix}\right. + \left\{\begin{matrix}0\\ \pm\frac{1}{8}\end{matrix}\right.\right). \qquad (1)$$

The component used is a three-input adder with weighting factors of -1, 0, and 1.

The symbolic values of each layer are of the form:

$$\boldsymbol{W} = \begin{vmatrix} w_1 & w_2 & w_3 & w_4 \\ w_5 & w_6 & w_7 & w_8 \\ w_9 & w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} & w_{16} \end{vmatrix}, \boldsymbol{B} = \begin{vmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{vmatrix}. \qquad (2)$$

Each numerical value from [3] is replaced by binary shifts and an adder element. The system parameters of the first hidden layer, the second hidden layer and the output layer are replaced by a small number of adder elements.

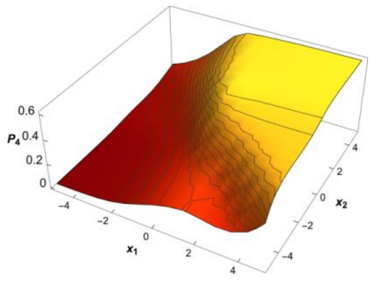
We can then analyze the probability of the four output classes in terms of the input values x_1 and x_2, which are from the range {-5, 5}. The chosen activation function is tanh as in [3]. Using the results obtained in [3] we have:

$$\boldsymbol{P} = \{p_1, p_2, p_3, p_4\} = \{0.184, 0.032, 0.184, 0.600\}. \qquad (3)$$

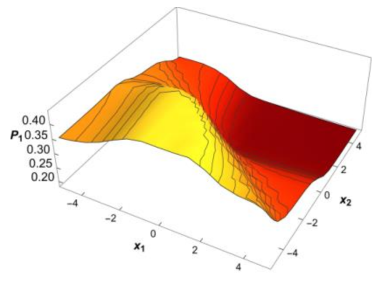
The results obtained using the quantized values are:

$$\boldsymbol{P}_q = \{p_{1q}, p_{2q}, p_{3q}, p_{4q}\} = \{0.19, 0.03, 0.18, 0.60\}. \quad (4)$$

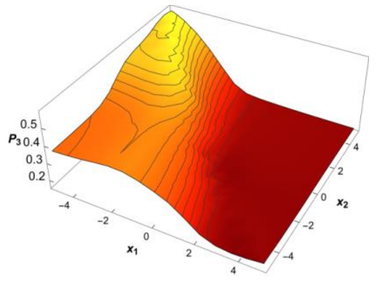
The probabilities as 3D plots and contour plots for the first, third and fourth classes are presented in Fig. 3 and 4. For input values $x_1>0$ and $x_2>0$, the most probable detected is the fourth class. For $x_1<0$ and $x_2>0$ the most probable detected is the first class. For $x_1>0$ and $x_2<-0.2$ the most probable detected is the third class.



a)

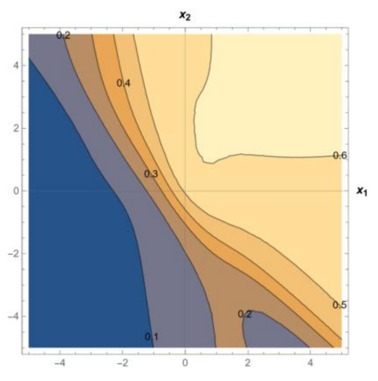


b)

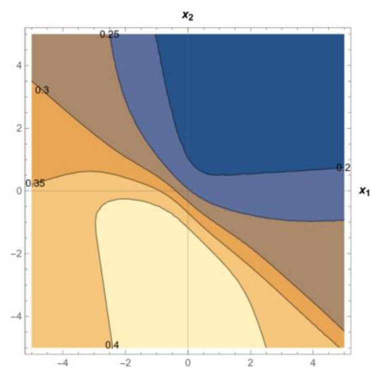


c)

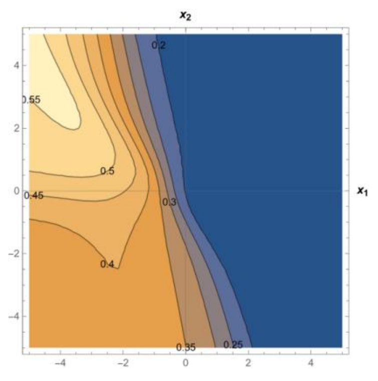
Figure 3. 3D probability diagram of (a) the fourth class, (b) the first class, and (c) the third class.



a)



b)



c)

Figure 4. Probability contour plot of (a) the fourth class, (b) the first class, and (c) the third class.

IV. Conclusions

The main disadvantage of neural networks is that it is observed as a black box system that can be viewed in terms of its inputs and outputs, without any knowledge of its inner workings. The advantage of this work is that symbolic analyzes provide visual identification of class probabilities.

References

[1] M. Lutovac-Banduka, I. Franc, V. Milićević, N. Zdravković, and N. Dimitrijević, "Symbolic analysis of classical neural networks for deep learning," Preprints 2023, 2023110446, November 2023, published Online https://doi.org/10.20944/preprints202311.0446.v1.

[2] V. Milićević, M. Lutovac-Banduka, I. Franc, N. Zdravković, and N. Dimitrijević, "Symbolic analysis of classical neural networks for deep learning," International Journal for Quality Research, vol. 19, no. 1, 10.24874/IJQR19.01-06, pp.1–15, February 2025, in press.

[3] E. Bernard, Introduction to machine learning, Champaign, IL, USA: Wolfram Media, 2022, pp.1–424.

[4] S. Wolfram, An Elementary Introduction to the Wolfram Language, 3rd ed.; Wolfram Media: Champaign, IL, USA, 2023, pp.1–376.

[5] M. Lutovac Banduka, D. Milosevic, Y. Cen, A. Kar, and V. Mladenovic, "Graphical user interface for design, analysis, validation, and reporting of continuous-time systems using Wolfram language," JCSC, 2023, Art. no. 2350244, vol. 32, no. 14, pp.1–15, February 2023.

[6] M. Lutovac-Banduka, A. Simović, V. Orlić, and A. Stevanović, "Dissipation minimization of two-stage amplifier using deep learning," Serbian Journal of Electrical Engineering, vol. 20, no. 2, pp.129–145, June 2023.

[7] M. Lutovac-Banduka, M. Lutovac, "Multiplierless neural networks for deep learning," MECO, June 2024.