International Journal for Quality Research 19(1) 85-100
ISSN 1800-6450

Vladimir Milićević [1]
Igor Franc
Maja Lutovac Banduka
Nemanja Zdravković
Nikola Dimitrijević

Article info:
Received 13.03.2024.
Accepted 10.09.2024.

DOI – 10.24874/IJQR19.01-06

SYMBOLIC ANALYSIS OF CLASSICAL NEURAL NETWORKS FOR DEEP LEARNING

Abstract: *Deep learning is usually based on matrix computing with a large number of hidden parameters that are not visible outside the computing module. A deep learning algorithm can be implemented in hardware or software as a non-linear system. It is common for researchers to visualize a computing module and monitor its hidden parameters. In this paper, we propose, as a proof of concept, to start the system design by drawing a single neuron. A more complex scheme of the neural network is obtained by using the copy, move, and paste commands for the simplest unit. The number of neurons and layers can be chosen arbitrarily. When the scheme is complete, implementation code is automatically executed using symbolic inputs, system parameters, and symbolic activation functions. This cannot be done manually because the system response is extremely complex. With the symbolic expression of outputs obtained from inputs and parameters, including pure symbolic activation functions, many other properties can be derived in closed form, such as classification with respect to a single system parameter, activation function, or inputs. This unique original method can help scientists and programmers design complex machine learning algorithms and understand how deep learning algorithms work. This paper presents several examples with new achievements. The proposed algorithm can be implemented in any programming language with symbolic computing. Although it was developed for a classical neural network, the same methodology can be used for any type of neural network.*

Keywords: *artificial neural networks, closed-form expression, feature extraction, machine learning.*

1. Introduction

Machine learning, as a subfield of artificial intelligence (AI), refers to computers learning to do things on their own (Bernard, 2022). Machine learning (ML) is the core of commonly used AI to derive complex algorithms and methods for classification, clustering and forecasting (Mukhamediev *et al*, 2022). It has many applications in several fields of basic science and engineering that solve practical problems, such as nonlinear problems, numerical methods, analytical methods, error analysis and mathematical models (Juraev & Noeiaghdam, 2022). A review of a variety of experiments on extracting structure from machine learning data is presented in (He, 2023). Efficient continuous approximation is proposed in (Yun, 2019). Some

[1] Corresponding author: Vladimir Milićević
Email: milicevic.v@mfkv.kg.ac.rs

applications with a nonlinear process have demonstrated different neural network-based models (Ren et al., 2022). A theoretical framework based on non-linear activation functions is explained in (Gnjatović et al., 2022). A popular two-layer activation function, a rectified linear unit (ReLU), for neural networks is introduced in (Liu & Cai, 2023). The trends of the mathematical explanations for the theoretical aspects of artificial neural networks (ANNs), with special attention to activation functions, can be used to derive the defining features of each design scenario (Zhao & Huang, 2022). Other researchers have developed cost estimation techniques using a statistical approach (Refonaa et al., 2022). Graph representation learning is a suitable learning model for practical prediction tasks (Jegelka, 2022). Symbolic regression, as the task of predicting a mathematical expression, is a difficult task; neural networks have been used in prediction but are still less powerful (Kamienny et al., 2022).

The biggest disadvantage of machine learning is that most machine learning-based algorithms cannot be explained. So far, AI has successfully solved practical problems that are intellectually difficult for humans but relatively simple for computer calculations. Success is possible if we use formal mathematical rules. The challenge for AI is to solve problems that are easy for humans but formally hard for programs using formal mathematical rules. Hierarchy allows a computer to solve complex problems by starting with simple solutions. A neural network is deep when it has at least two hidden layers. For this reason, artificial intelligence based on neural networks with two hidden layers is called deep learning (Goodfellow et al., 2016). Hard-coded knowledge is not adequate for artificial intelligence systems; AI needs to be trained to acquire its own knowledge from raw data (Goodfellow et al., 2016). We can use mathematical mapping to produce output values from input values. A complex function is defined from simpler functions. Some authors noticed that simpler artificial networks still can provide solutions as biological neurons (Pavone & Plebe, 2021). To solve decision-making problems in both computer science and engineering, we can use the Optimization Machine Learning Toolkit as an open-source software package incorporating neural networks (Ceccon et al., 2022). Software tools can simplify the training of neural networks using machine learning to solve problems ranging from simpler to more complex optimization problems.

Increased interest in solving visual computing problems using neural networks is evident (Xie et al, 2022). Machine learning tools are part of the Wolfram Language that performs classification, regression, dimensionality reduction and neural network processing. The author of (Bernard, 2022) takes a "show, don't tell" approach and we have used the same examples to show how to use the Wolfram Language (Wolfram, 2023). Using a visual programming language (the SchematicSolver application package (Lutovac et al, 2016), which requires the software Mathematica 9 (Wolfram, 2023)), we can develop our own algorithms to better understand the architecture of complex networks. Mathematical education at the end of high school should be sufficient to understand mathematical content (Bernard, 2022). A visual programming interface or GUI (graphical user interface) can provide fast interactive design, symbolic processing, exact simulation, verification and reporting (Lutovac-Banduka et al., 2023; Raska et al., 2024). This way, we skip the gap between theory and practice in continuous-time systems. A mathematical representation of a complex system can be obtained by using simpler parts. Automated symbolic computations are also possible (Lutovac-Banduka et al., 2023).

AI models as "black boxes" are often not comprehensible (Setzu et al., 2021). Transfer learning is a powerful approach that leverages knowledge from one domain to enhance

learning in another domain; it can provide practical insights for researchers and practitioners interested in applying transfer learning techniques (Ghelani, 2021). Even though AI has good predictive performance, it also has poor properties as there is a lack of explanation for people to understand how it works (Swamy et al., 2022). To date, there are tools and case studies that aim to explain the black boxes of AI in healthcare (Srinivasu et al., 2022).

Deep learning using convolutional neural networks (CNNs) is used to solve the problem of brain tumor diagnosis; but its clinical applications still face some critical issues (Xie et al., 2022). The smart visualization of medical image models is based on multi-detector computed tomography, which can provide a clearer view of changes in the brain (Simović et al., 2022).

Trends in machine learning techniques have been applied to robotic manipulation (Vuong, 2021). This can overcome the limitations associated with a common user attempting to access the deep knowledge of robot programming. For simulation and remote control of robotic manipulators, we can use widespread and inexpensive devices and the Android operating system (Lutovac Banduka, 2016). A review on deep learning-based smart processing, robotic sensor networks and data management algorithms is reported in (Lăzăroiu et al., 2022). A simulator used to verify motion commands and test new human centrifuge algorithms is presented in (Lutovac Banduka et al., 2013).

Analogous designs implementing machine learning applications are explained in (Liu et al., 2022). The design of analog circuits requires a large amount of human knowledge (Rojec et al., 2022). An example of using deep learning as a subset of machine learning is presented in (Lutovac-Banduka et al., 2023). The application of using an artificial intelligent method is described in (Thanoon, 2024).

The purpose of this paper is to explain machine learning models using algorithm visualization. The main goal of this work is to present the final results as expressions in a closed form in which developers can change and monitor hidden parameters, input values, or selection of proper activation function.

2. Materials

Deep learning involves learning using an artificial neural network. Humans have about one hundred billion biological neurons in their brain and about ten thousand times more connections. A neuron has its inputs x_1, x_2, ..., x_i, via input branches.

The neuron then calculates an output y_j, which is sent to other neurons through tiny intercellular connections.

Artificial neural networks use the same basic principles but are much simpler than biological neurons. For the given input values x_1, x_2, ..., x_i, an artificial neuron has only one output, whose value we can calculate according to the following formula:

$$y_j = f\left(w_1x_1+w_2x_2 + \cdots + w_ix_i + b_j\right). \quad (1)$$

The system parameters, w_1, w_2, ..., w_i, change during neuron operation and are interpreted as the strengths of the connection between neurons. They are called weights or weight parameters. The parameter b_j is called the bias and is interpreted as the threshold at which a neuron is activated. The nonlinear function f is called the activation function or the transfer function.

An illustration of the calculation of an artificial neuron (which gives the same output expression as it theoretically should be) is presented in Fig. 1 (drawn using SchematicSolver ver. 2.3 (Lutovac et al., 2014), which is written using the Wolfram Language and Mathematica ver. 13 (Wolfram, 2023)).

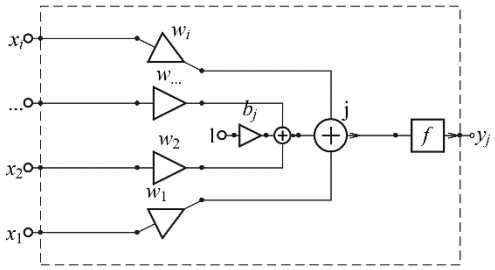

Figure 1. Illustration of the computation made by an artificial neuron

The first part is a linear combination of the inputs, and then the nonlinearity f is applied. Nonlinearity allows neurons to model a nonlinear system. Biological neurons are either activated or not activated. Therefore, it is beneficial to use some kind of activation function. Artificial neurons either use a logistic sigmoid function, a hyperbolic tangent function or a Ramp function (rectified linear unit, ReLU), which works pretty well in deep neural networks (Bernard, 2022), as shown in Fig. 2.

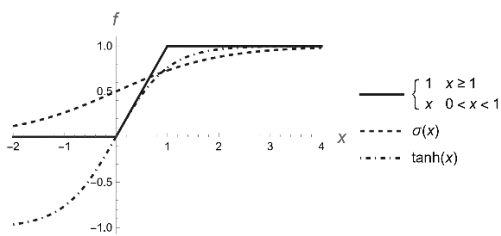

Figure 2. Classic activation function as a rectified linear unit ReLU, a logistic sigmoid σ(x), or a hyperbolic tangent function tanh(x)

3. Methods

In Fig. 3, the architecture of an artificial neuron is drawn, where the input signals x_1, x_2, ..., x_i are separated from the parameters that are modified during the operation of the neuron. Fig. 4 shows a block diagram of a neuron that depends exclusively on the input signals, and the parameters w_1, w_2, ..., w_i are listed as inputs through which the output is calculated.

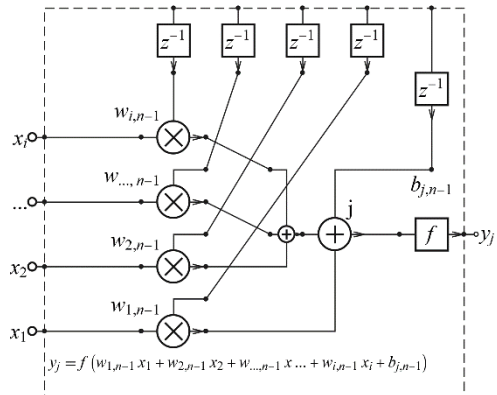

Figure 3. Classic artificial neuron with memory elements

It is important to note that unlike linear systems, multipliers with constant values are not used because the parameters w_1, w_2, ..., w_i, and b_j are not constants. They are modified during the operation of the neuron.

By connecting a large number of artificial neurons, a network can be created. The circles in this network represent numerical values called activations. The linear combination of the inputs and bias are passed through the nonlinearity function. The simplest network has two inputs and one output. The graph of such a network is directed and acyclic, so the output can be calculated simply by following the arrows.

Each neuron has an input with its own weight parameter and one bias parameter. For training purposes, we can define a minimum cost function. The architecture does not change; only the numerical values of the parameters (weights and biases) are changed. A similar architecture is used with a pair of neurons in the layers; each neuron in one layer sends its output to each neuron in the next layer.

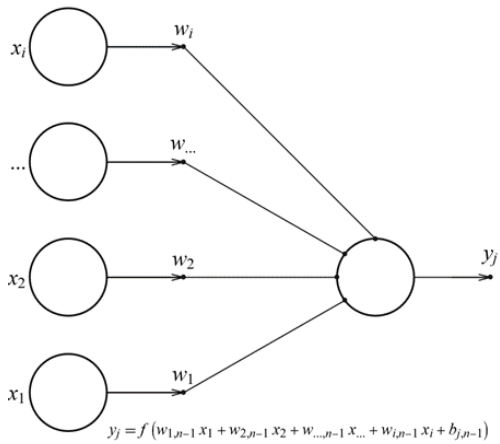

Figure 4. Classic Classic artificial neuron - simplified version

Let the network consist of several layers and let there be four neurons in each layer. Four input signals come to the first layer. The output of each neuron generates only one output, and since there are 4 neurons, 4 output signals are generated. The output signals of the first layer become the input signals for the second layer, which also has 4 neurons and generates 4 output signals. The output signals of the second layer are the input signals of the third layer, which also has 4 neurons with 4 output signals. The output signals of the third layer are the outputs of the neural network. The first and second layers are called hidden layers because the outputs of the neurons in these layers are not seen outside the network.

The number of input signals to the first layer must not be greater than 4, and to ensure that some of the 4 network inputs do not affect the network outputs, the weight parameters are set to 0 from the network inputs in the first layer to the adders.

Assume that there are only two numerical values as the inputs and the first hidden layer generates four numerical values as the outputs. Now, the second hidden layer has 4 numerical values as the inputs, and it generates 4 output values which are the values of the third-layer neurons. These four inputs generate 4 output numerical values that become the outputs of the neural network, and therefore, the third layer is not a hidden layer because the results are visible. The activation function generates one of the 4 classes to which the processing result belongs, and the activation function is unambiguously mapped to the outputs, that is, the ones that have all positive results (all probabilities must be positive), and the sum of all probabilities (outputs) used as a nonlinear function of the last layer must be 1 (sum of probabilities of all classes must be 1). Such a network is used to train a classifier. In our example, there are four possible classes, and the output values would then be the class probabilities. For a regression task, you would have only one output, which is the class that is most likely to occur.

In our example, two hidden layers are included, although there could be many more. A network with at least two hidden layers is called a deep network.

3.1 Matrix representation

As an illustration of how a neural network works, we analyze a network that has 4 neurons in three layers as a minimal deep learning network (the inputs are not layer, the output layer is not hidden layer). Usually, a mathematical approach is used to calculate the matrices, which are easily implemented in numerical programs. The input layer contains 4 weight coefficients for each neuron, so a total of 2×4=8 parameters for the weights are used to calculate the output values of the first hidden layer. One bias parameter per output is added. Visually, the weight parameters are represented as a 4×4=16 matrix (for four neurons, each with 4 weights), and a 1×4=4 matrix (for four neurons, each with one bias parameter). For the input values, we use a 1×4=4 matrix with numerous values that can be used as input signals to the network or as output signals of the previous layer. The matrix representation is as follows, which is determined according to formula (Bernard, 2022):

$$\boldsymbol{Y} = f(\boldsymbol{W} \cdot \boldsymbol{X} + \boldsymbol{B}) \quad (2)$$

The matrix representations of parameters **W** and **B** are:

$$\boldsymbol{W} = \begin{vmatrix} w_1 & w_5 & w_9 & w_{13} \\ w_2 & w_6 & w_{10} & w_{14} \\ w_3 & w_7 & w_{11} & w_{15} \\ w_4 & w_8 & w_{12} & w_{16} \end{vmatrix}, \ \boldsymbol{B} = \begin{vmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{vmatrix}. \quad (3)$$

The matrix representations of the activation function $f(\boldsymbol{A})$, inputs $\boldsymbol{X}$, and outputs $\boldsymbol{Y}$ of one layer are:

$$f(\boldsymbol{A}) = tanh(\boldsymbol{A}), \ \boldsymbol{X} = \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{vmatrix}, \ \boldsymbol{Y} = \begin{vmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{vmatrix}. \quad (4)$$

3.2 Matrix computation

The activation function is chosen to be a hyperbolic tangent. For only two inputs, numerous values are arbitrarily chosen, while the remaining two inputs are set to 0. All parameters are obtained via a random number generator. In order to always obtain the same numerical value, the function SeedRandom [123] is used. Now, the numerical values for the first layer are rounded to 3 decimal places, according to reference (Bernard, 2022):

$$\boldsymbol{W} = \begin{vmatrix} -0.089 & 0.956 & 0 & 0 \\ 0.887 & 0.924 & 0 & 0 \\ -0.395 & -0.067 & 0 & 0 \\ -0.877 & -0.229 & 0 & 0 \end{vmatrix}, \boldsymbol{B} = \begin{vmatrix} -0.140 \\ 0.557 \\ -0.903 \\ 0.257 \end{vmatrix}, \quad (5)$$

$$\boldsymbol{X} = \begin{vmatrix} 1.2 \\ 4.5 \\ 0 \\ 0 \end{vmatrix}, \ \boldsymbol{Y} = \begin{vmatrix} 0.999398 \\ 0.999981 \\ -0.932446 \\ -0.949308 \end{vmatrix} \quad (6)$$

To make sure that the third and fourth inputs do not affect the result, some weight parameters are set to 0. Therefore, this is a network with only two inputs.

The output values of the first layer are represented as a matrix **Y**. These values become the inputs for the next layer, so the following values are obtained for the second layer:

$$\boldsymbol{W} = \begin{vmatrix} -0.444 & -0.820 & 0.753 & -0.782 \\ -0.468 & 0.837 & -0.660 & -0.801 \\ -0.059 & -0.194 & 0.943 & -0.370 \\ -0.748 & -0.455 & 0.211 & 0.344 \end{vmatrix},$$

$$\boldsymbol{B} = \begin{vmatrix} -0.515 \\ -0.016 \\ 0.126 \\ 0.098 \end{vmatrix} \quad (7)$$

$$\boldsymbol{X} = \begin{vmatrix} 0.999398 \\ 0.999981 \\ -0.932446 \\ -0.949308 \end{vmatrix}, \boldsymbol{Y} = \begin{vmatrix} -0.940069 \\ 0.938869 \\ -0.575210 \\ -0.925832 \end{vmatrix}. \quad (8)$$

The output values of the second layer are shown as the matrix **Y**. These values become the inputs to the last layer, so the following values are obtained for the last layer:

$$\boldsymbol{W} = \begin{vmatrix} -0.037 & 0.289 & -0.165 & 0.409 \\ 0.923 & 0.808 & 0.741 & 0.263 \\ -0.215 & -0.127 & 0.137 & -0.034 \\ -0.679 & 0.927 & -0.908 & -0.007 \end{vmatrix},$$

$$\boldsymbol{B} = \begin{vmatrix} -0.008 \\ -0.975 \\ -0.023 \\ -0.843 \end{vmatrix} \quad (9)$$

$$\boldsymbol{X} = \begin{vmatrix} -0.9400769 \\ 0.938869 \\ -0.575210 \\ -0.925832 \end{vmatrix}, \boldsymbol{Y} = \begin{vmatrix} 0.600 \\ 0.184 \\ 0.031 \\ 0.184 \end{vmatrix} \quad (10)$$

The output values of the last layer do not use the hyperbolic tangent of the activation function, but the so-called `softmax` function we are using to calculate the probability of the occurrence of an output:

$$\frac{e^{y_i}}{e^{y_1}+e^{y_2}+e^{y_3}+e^{y_4}}, \ i=\{1, 2, 3, 4\} \quad (11)$$

The fourth outcome has a probability of 0.600, i.e., 60.0%, while the first and third outcomes have a probability of 0.184 (18.4%) and the second outcome has a probability of 0.031 (3.1%). All probabilities must be positive, and the sum of all probabilities must be 1 (the first decimal place may have a difference due to rounding).

A classifier that has parameters, as in this example, shows that the most likely result is class 4. The weight parameters w_1, w_2, ..., w_i, and the bias parameter b_j change during the operation of the neurons during the testing phase. In total, in the analyzed network with two hidden layers and one output layer with 4 neurons, there are 3×4×4=48 weight parameters and 3×1×4=12 bias parameters, i.e., the analyzed network has a total of 48+12=60 parameters, which are changed during neural network processing.

4. Results

In computing, it is common to use a matrix data structure. It is even more common for people to use an architectural sketch.

4.1 Symbolic computation

Firstly, we draw the architecture of one layer consisting of four neurons using software (Lutovac, et al., 2014), as shown in Fig. 5.

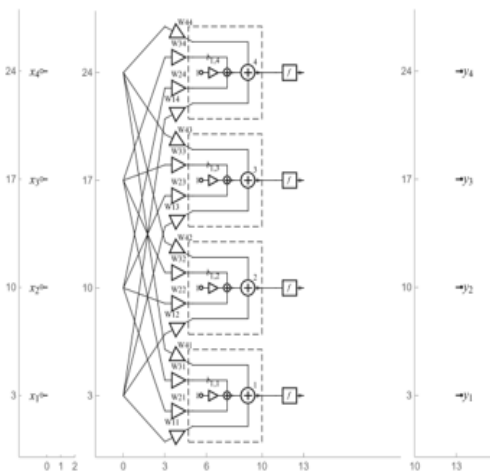

Figure 5. Classic artificial neural network with a single layer: 4 inputs, 4 neurons in the middle stage, and 4 outputs

The inputs and outputs are drawn separately for a neural network with one layer, as shown in Fig. 5. Using one middle layer, the neural network is obtained by joining the input, the output and the one middle layer, as shown in Fig. 6.

To draw a neural network with three layers, an already-drawn middle layer, as shown in Fig. 5, is used, which is copied three times to the right with a raster of 13 (how wide the base layer is), and the output layer is shifted to the right by 3×13=39. The symbolic parameter names and activation functions are changed with each copy. The complete scheme is obtained by connecting the inputs, the outputs and all layers and is shown in Fig. 7.

It is important to note that the activation function of the final layer is different, g, so that the output values are equal to the class probabilities.

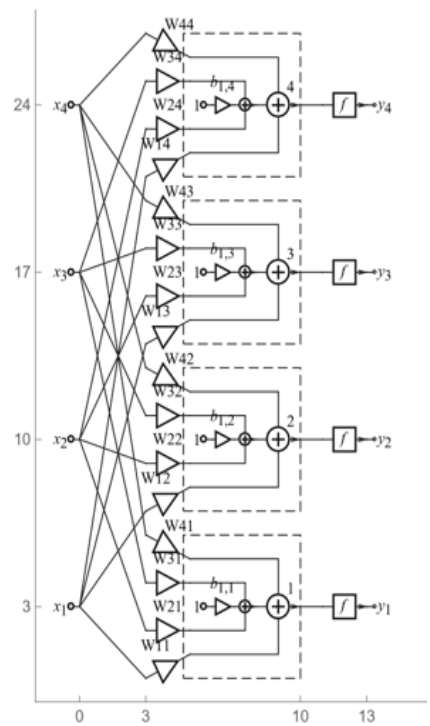

Figure 6. Classic artificial neural network with one layer

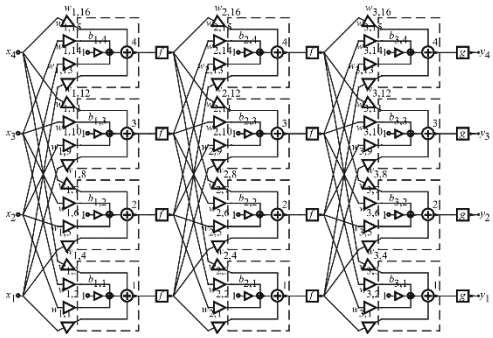

Figure 7. Classic artificial neural network with three layers

4.2 Symbolic analysis

By clicking on the button to implement the schematic obtained from the image drawn using SchematicSolver, an implementation code is obtained, a small part of which is shown in Fig. 8.

The symbolically derived expressions for the four classes use almost 316KB as it is shown in Fig. 9. Such a performance in the closed form is not possible to derive manually. In Fig. 10, it can be seen that the graph with all levels is very complex. The symbolically derived expressions can be used for further analysis, which is not possible when using a purely numerical approach. By changing the symbolic values with the numerical parameters from the matrix approach, the same probabilities are obtained, and there is a 60.0% probability that the classification number is four, as with matrix computing.

```
{g[b21 + f[b11 + f[b1 + w11 x1 + w21 x2 + w31 x3 + w41 x4] w111 +
      f[b2 + w12 x1 + w22 x2 + w32 x3 + w42 x4] w121 +
      f[b3 + w13 x1 + w23 x2 + w33 x3 + w43 x4] w131 +
      f[b4 + w14 x1 + w24 x2 + w34 x3 + w44 x4] w141] w211 +
   f[b12 + f[b1 + w11 x1 + w21 x2 + w31 x3 + w41 x4] w112 +
      f[b2 + w12 x1 + w22 x2 + w32 x3 + w42 x4] w122 +
      f[b3 + w13 x1 + w23 x2 + w33 x3 + w43 x4] w132 +
      f[b4 + w14 x1 + w24 x2 + w34 x3 + w44 x4] w142] w221 +
   f[b13 + f[b1 + w11 x1 + w21 x2 + w31 x3 + w41 x4] w113 +
      f[b2 + w12 x1 + w22 x2 + w32 x3 + w42 x4] w123 +
```

Figure 8. A small part of the symbolic response of classic artificial neural network with 3 layers

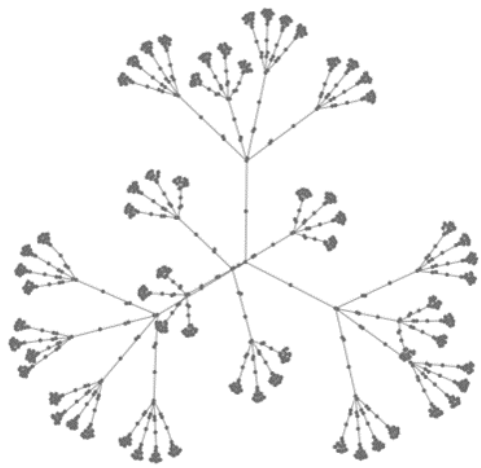

Figure 9. Tree graph with different levels at different depths

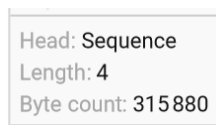

Figure 10. Size of the tree graph

4.3 Symbolic computation

In the same way, a neural network with four layers is obtained, of which three are hidden layers. It has the same values for the parameters as a network with two hidden layers, but with the addition of randomly generated values for newly added layer, a probability of 63.8% is obtained for class 2:

$$\boldsymbol{Y} = \begin{vmatrix} 0.14015 \\ 0.638097 \\ 0.118381 \\ 0.103371 \end{vmatrix}. \tag{12}$$

The complexity of the closed-form final expressions in symbolic notation is shown in Fig. 11. This derivation exceeds manual execution.

The symbolically obtained response of the neural network enables further analysis without the use of schematics. With 100,000 randomly generated parameters, the probabilities for all outputs were determined, as shown in Fig. 12.

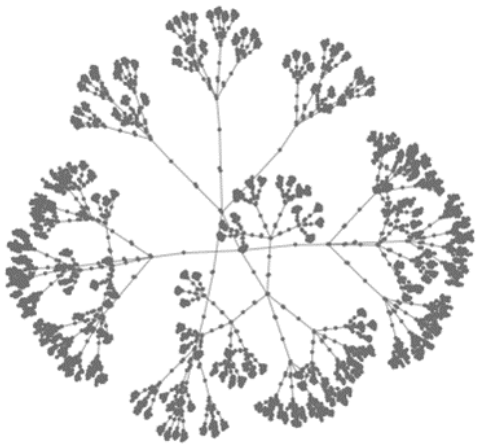

Figure 11. The tree graph of classic artificial neural network with 4 layers

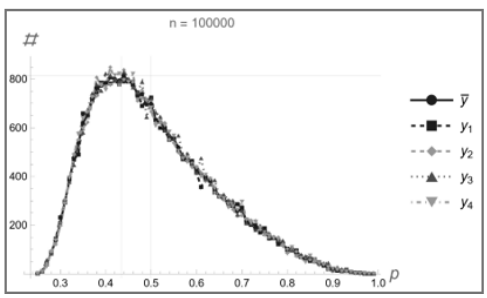

Figure 12. The tree graph of classic artificial neural network with 4 layers.

The mean probability is determined, as shown in Fig. 13.

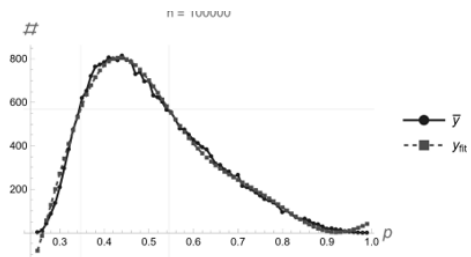

Figure 13. Mean value of probability

For the mean value, we derive an expression in the closed form as follows:

$$y(x) = 96.267 + 86.935\cos(16.728\,x) - 344.78\log(x) + 592.72\,log(x)\,sin(9.2\,x). \quad (13)$$

The most likely probability was calculated when the maximum number of classes was detected, which was a probability of 43.3% when 800 cases were detected. For a value of 0.7071 in relation to the maximum range in which the largest number of certain classes was calculated, the range of probability was between 34.7% to 54.4%. Half of all the classes were detected within this range. It is important to note that there cannot be a probability of less than 25% when a class is detected because at least one other class would have a probability of more than 25%. Each class was detected approximately 25,000 times, and no case was detected where all classes were equally likely. In all cases, the same input signal values were used $x_1 = 1.2$, $x_2 = 4.5$, $x_3 = 0$, $x_4 = 0$.

For all classes and all probabilities, all parameter values are equally likely, that is, there is no parameter value where the probability of detecting one of the four classes is more pronounced.

5. Application

The symbolic representation of a neural network can be used for further analysis.

5.1 The influence of any parameter

We assume all parameters are frozen except for the last layer. We can then set all parameters to have the same values when calculating the matrix. In that case, the probability of the four outcomes is already given by equation (11).

Suppose we would like to find how much the outputs would change for different sets of parameters that are in line with the neurons, for example, the parameters $w_{3,1}$ and $w_{3,10}$. The class probabilities of all outputs as functions of these parameters are shown in Fig. 14 and Fig. 15.

The probability of the output class y_4 as a function of the parameters $w_{3,1}$ and $w_{3,10}$ is always higher than any other probability. This means that with small changes in these

parameter values, the expected output is always y_4, assuming all other parameters are frozen.

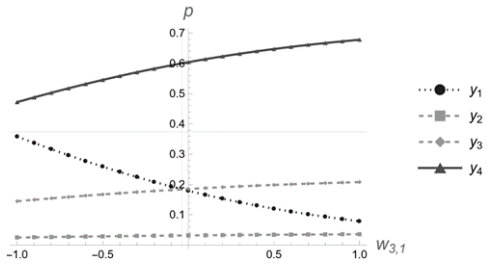

Figure 14. Hyperbolic tangent activation function: probabilities of all outputs as a function of $\boldsymbol{w_{3,1}}$

This property gives us the idea that the parameters can be changed in larger steps. The most expected value of a parameter can be implemented with a small number of shifts and add operations, and thus, we can replace all multipliers with a few shifts and adders. This is important for fast algorithms since the inherent delay of a multiplier is reduced to that of an adder element.

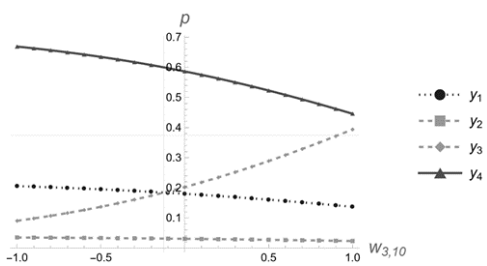

Figure 15. Hyperbolic tangent activation function: probabilities of all outputs as a function of $\boldsymbol{w_{3,10}}$.

The adaptation of the parameters as multiplication constants during the training phase can be replaced by shifts and two-input sums during hardware implementations of ANN (artificial neural network).

5.2 The role of the activation function

Again, we assume all parameters are frozen except for the last layer. We can set all parameters to have the same values when calculating the matrix. The probability matrix for the four outputs is different with the logistic sigmoid $\sigma(k)$ as the activation function:

$$\boldsymbol{Y} = \begin{vmatrix} y_4 \\ y_3 \\ y_2 \\ y_1 \end{vmatrix} = \begin{vmatrix} 0.115 \\ 0.263 \\ 0.289 \\ 0.333 \end{vmatrix} \neq \begin{vmatrix} 0.600 \\ 0.184 \\ 0.031 \\ 0.184 \end{vmatrix}. \qquad (14)$$

Suppose we would like to find how much the outputs change for different sets of parameters that are in line with the neurons. The class probabilities of all outputs as functions of some parameters are shown in Fig. 16 and Fig. 17.

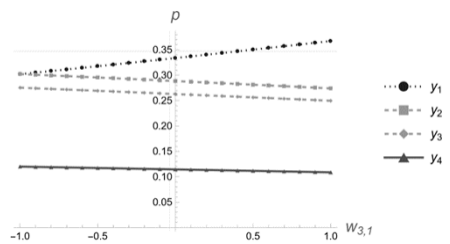

Figure 16. Logistic sigmoid activation function: probabilities of all outputs as a function of $\boldsymbol{w_{3,1}}$

The probability of the output class y_4, as a function of the parameters $w_{3,1}$ and $w_{3,10}$ is always less than any other probability. This means that with small changes in these parameter values, the expected output is not y_4, assuming all other parameters are frozen.
It follows that the parameter $w_{3,10}$ can be used to make y_3 the most expected class. Otherwise, the output y_1 is the most expected class.

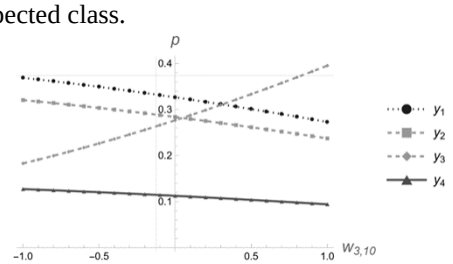

Figure 17. Logistic sigmoid activation function: probabilities of all outputs as a function of $\boldsymbol{w_{3,10}}$

The only modification in the program is the change in the activation function, and everything else is the result of evaluating the same matrix using the logistic sigmoid activation function.

5.3 Training properties

Fig. 18 illustrates the class probabilities of all outputs as functions of the input value x_1 when using the hyperbolic tangent activation function. For positive input values x_1, the probabilities are approximately constant and the expected output of the class is y_4, assuming all other parameters are frozen. In this case, there is no need to change the parameters.

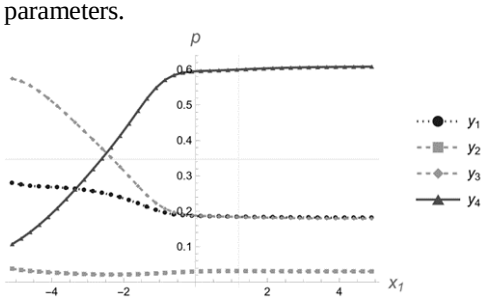

Figure 18. Hyperbolic tangent activation function: class probabilities of all outputs as a function of $\boldsymbol{x_1}$

Fig. 19 shows the class probabilities of all outputs as a function of the input value x_2 when using the hyperbolic tangent activation function. For positive input values x_2, the probabilities are approximately constant and the expected class output is again y_4, assuming all other parameters are frozen. In this case, there is no need to change the parameters.

Fig. 20 shows the class probabilities of all outputs as a function of the input value x_1 when using the logistic sigmoid activation function. For this activation function, the expected output of the class is no longer x_4, assuming all other parameters are frozen. In this case, there is no need to change the parameters except to increase the probabilities of other outputs.

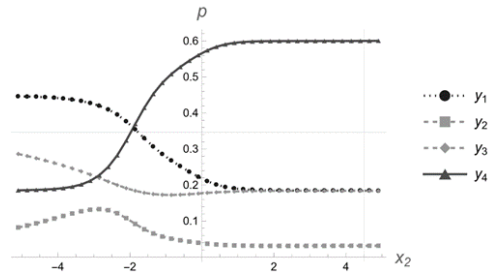

Figure 19. Hyperbolic tangent activation function: class probabilities of all outputs as a function of $\boldsymbol{x_2}$

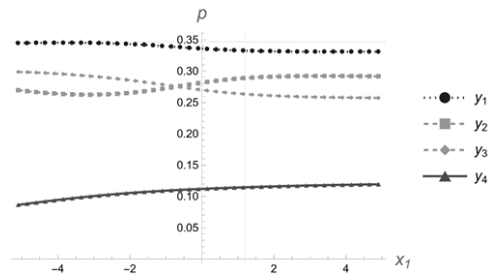

Figure 20. Logistic sigmoid activation function: class probabilities of all outputs as a function of $\boldsymbol{x_1}$

Fig. 21 illustrates the class probabilities as functions of the input value x_2 when using the logistic sigmoid activation function. In this case, the probabilities of other outputs can be increased by changing the parameters.

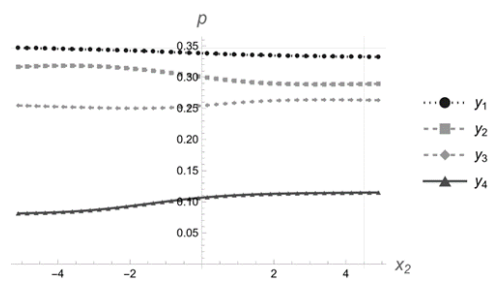

Figure 21. Logistic sigmoid activation function: class probabilities of all outputs as a function of $\boldsymbol{x_2}$

For positive and negative input values x_1 and x_2, the probabilities are approximately constant. Of course, the tree class probabilities are in the 25% to 35% range.

It is important to note that the input values can be in the value range of -5 to 5. In other cases, the output value of each neuron, that is, the input of any neuron in the hidden layers, can be in the range of -1 to 1.

6. Discussion

The most popular approach to using artificial neural networks (ANNs) and machine learning (ML) is to use mathematical matrix knowledge and black-box models. Researchers with extensive mathematical knowledge are familiar with such an approach and do not need to use different methods. However, researchers without extensive mathematical knowledge and knowledge of the matrix representation of black-box models usually do not have the technical skills to apply deep learning (DL) ANNs to solve practical problems. Therefore, they need a different approach. One such approach is presented in this paper

Graphical programming is useful for non-technical users as they can see the ANN architecture's component parts. Instead of using a black-box model, a researcher can visually identify the architecture of an ANN and perform analysis to investigate the impact of each constituent part of the ANN.

The proposed solution is based on symbolic manipulations with symbolically defined parameters of the system schema. The system schema contains everything needed for a different ANN presentation (graphical for drawing scheme, mathematical for the closed-form response of the system, programing for setting programing code for software implementation, embedded code for hardware implementation, visualization of the plotting the system response for numeric values). A schematic view is not just a pictorial representation of a system. By means of a simple transformation, the system's response to symbolic input values generates an output. Another advantage is that this original method generates a software or hardware implementation. Since all system parameters are defined as symbols, they can be symbolically replaced by numbers at any time. For example, to plot the response, we need numerical values.

An ANN response may be too complex for human derivation. It will be difficult to rewrite such a response. Therefore, this original method is a kind of knowledge-based system. The complexity of the system can be increased or simplified to better understand the properties of the ANN. For example, in a tree-layered ANN with four neurons in each layer, we can freeze most of the 48 parameters and plot the response as a function of a single parameter. If the result is not satisfactory, we can choose another activation function. As expected by experts, the appropriate parameters and activation function are used to obtain the expected response of the ANN.

Although the higher programing language is used, the same approach can be developed using a more user-friendly programming language.

The disadvantage of using ML as a black box is overcome by designing a non-linear system using visual programming. During the testing or validation process, system parameters can be viewed for verification purposes. The final results, presented as closed-form expressions, can be derived and optimized according to the required task. Future development includes developing a fully automated design based on the required number of neurons per layer as well as the number of layers. To the best of the authors' knowledge, there is no similar approach in the literature.

7. Conclusion

An original algorithm for implementing classical neural networks, as a concept of proofs, is presented. This original method provides a symbolic response that can be easily transformed, for example, into a numerical response for plotting. The algorithm is developed using one of the best software for symbolic computing and a graphical user interface for drawing simple

constituent parts of systems. A more complex system can be generated via visual programming using the copy-move-paste method. Future work is planned for a different neural network structure and more activation functions.

7.1 Supplementary Materials

Programs are available from the corresponding author or https://www.preprints.org/manuscript/202311.0446/v1.

The code is in the form of readable text with the extension. nb. The first step is to set up a working directory and import the knowledge as a package or text file with .m extension. The next few lines of the program tell how the scheme is visible in the images. The inputs, outputs and single-layer schemes are defined as separate lists. Replacement rules are used for drawings. The starting layer can be translated and copied several times using the `Do` command. Each symbol can be represented by different mathematical notations.

References:

Bernard, E. (2022). *Introduction to machine learning*. Champaign, IL: Wolfram Media.

Ceccon, F., Jalving, J., Haddad, J., Thebelt, A., Tsay, C., & Misener, R. (2022). OMLT: Optimization & machine learning toolkit. *Journal of Machine Learning Research, 23*(1), 15829–15836. Retrieved from https://www.jmlr.org/papers/volume23/22-0277/22-0277.pdf

Ghelani, D. (2022). Explainable AI: Approaches to make machine learning models more transparent and understandable for humans. *International Journal of Computer Science and Technology, 6*(4), 45–53.

Gnjatović, M., Maček, N., Saračević, M., Adamović, S., Joksimović, D., & Karabašević, D. (2022). Cognitively economical heuristic for multiple sequence alignment under uncertainties. *Axioms, 12*(3), 1–15. https://doi.org/10.3390/axioms12010003

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, MA: MIT Press.

He, Y. H. (2023). Machine-learning mathematical structures. *International Journal of Data Science and Mathematical Sciences, 1*(1), 23–47. https://doi.org/10.48550/arXiv.2101.0631

Jegelka, S. (2022). Theory of graph neural networks: Representation and learning. In *Proceedings of the International Congress of Mathematicians* (virtual event, originally planned Saint Petersburg, Russia), Helsinki, Finland, July 6–14. https://doi.org/10.48550/arXiv.2204.07697

Juraev, D., & Noeiaghdam, S. (2022). Modern problems of mathematical physics and their applications. *Axioms, 11*(2), 1–6. https://doi.org/10.3390/axioms11020045

Kamienny, P. A., d'Ascoli, S., Lampe, G., & Charton, F. (2022). End-to-end symbolic regression with transformers. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, New Orleans, USA, November 28–December 9. *Advances in Neural Information Processing Systems, 35*, 10269–10281.

Lăzăroiu, G., Adronie, M., Iatagan, M., Gaemanu, M., Stefanescu, R., & Dijmarescu, I. (2022). Deep learning-assisted smart process planning, robotic wireless sensor networks, and geospatial big data management algorithms in the Internet of Manufacturing Things. *Internet of Manufacturing Things, 11*(5), 1–26. https://doi.org/10.3390/ijgi11050277

Liu, M., & Cai, Z. (2023). Adaptive two-layer ReLU neural network: II. Ritz approximation to elliptic PDEs. *Computers & Mathematics with Applications, 113*, 103–116. https://doi.org/10.1016/j.camwa.2022.03.010

Liu, S. C., Strachan, J. P., & Basu, A. (2022). Prospects for analog circuits in deep networks. In P. Harpe, K. Makinwa, & K. Baschirotto (Eds.), *Advances in analog circuit design: Analog circuits for machine learning, current/voltage/temperature sensors, and high-speed communication* (Vol. 1, pp. 49–61). Springer: Heidelberg, Germany. Presented at the Workshop AACD 2021. https://doi.org/10.48550/arXiv.2106.12444

Lutovac Banduka, M. (2016). Robotics first: A mobile environment for robotics education. *The International Journal of Engineering Education, 32*(2A), 818–829.

Lutovac, M., Tosic, D., & Evans, B. (2016). SchematicSolver, Mathematica application package: Symbolic signal processing, software implementation, mouse-driven interactive drawing tool. Retrieved from https://www.wolfram.com/products/applications/schematicsolver

Lutovac-Banduka, M., Kvrgić, V., Ferenc, G., Dimić, Z., & Vidaković, J. (2013). 3D simulator for human centrifuge motion testing and verification. Presented at the *Mediterranean Conference on Embedded Computing*, Budva, Montenegro, June 15–20. https://doi.org/10.1109/MECO.2013.6601345

Lutovac-Banduka, M., Milošević, D., Cen, Y., Kar, A., & Mladenović, V. (2023). Graphical user interface for design, analysis, validation, and reporting of continuous-time systems using Wolfram language. *Journal of Circuits, Systems and Computers, 32*(14), Article 2350244. https://doi.org/10.1142/S0218126623502444

Lutovac-Banduka, M., Simović, A., Orlić, V., & Stevanović, A. (2023). Dissipation minimization of two-stage amplifier using deep learning. *Serbian Journal of Electrical Engineering, 20*(2), 129–145. https://doi.org/10.2298/SJEE2302129L

Mukhamediev, R., Popova, Y., Kuchin, Y., Zaitseva, E., Kalimodayev, A., Symagulov, A., ... Yelis, M. (2022). Review of artificial intelligence and machine learning technologies: Classification, restrictions, opportunities and challenges. *Mathematics, 10*(15), 1–25, Article 2552. https://doi.org/10.3390/math10152552

Pavone, A., & Plebe, A. (2021). How neurons in deep models relate with neurons in the brain. *Algorithms, 14*, 1–15, Article 272. https://doi.org/10.3390/a14090272

Raska, P., Ulrych, Z., Baloun, J., Malaga, M., & Lenc, L. (2024). Using adaptive neural networks for optimising discrete event simulation. *International Journal of Simulation Modelling, 23*(2), 227–238. https://doi.org/10.2507/IJSIMM23-2-678

Refonaa, J., Huy, D. T. N., Trung, N. D., Thuc, H. V., Raj, R., Haq, M. A., & Kumar, A. (2022). Probabilistic methods and neural networks in structural engineering. *International Journal of Advanced Manufacturing Technology, 125*(3–4), 1–9. https://doi.org/10.1007/s00170-022-09335-5

Ren, Y. M., Alhajeri, M. S., Luo, J., Chen, S., Abdullah, F., Wu, Z., & Christofides, P. D. (2022). A tutorial review of neural network modeling approaches for model predictive control. *Computers & Chemical Engineering, 165*, 1–71, Article 107956.

Rojec, Ž., Fajfar, I., & Burmen, Á. (2022). Evolutionary synthesis of failure-resilient analog circuits. *Mathematics, 10*(1), 1–20, Article 156. https://doi.org/10.3390/math10010156

Setzu, M., Guidotti, R., Monreale, A., Turini, F., Pedreschi, D., & Giannotti, F. (2021). GLocalX: From local to global explanations of black box AI models. *Artificial Intelligence, 294*, 1–15, Article 103457. https://doi.org/10.1016/j.artint.2021.103457

Simović, A., Banduka, M. L., Lekić, S., & Kuleto, V. (2022). Smart visualization of medical images as a tool in the function of education in neuroradiology. *Diagnostics, 12*, 1–19, Article 3208. https://doi.org/10.3390/diagnostics12123208

Srinivasu, P. N., Sandhya, N., & Jhaveri, R. H. (2022). From black-box to explainable AI in healthcare: Existing tools and case studies. *Mobile Information Systems, 2022*, 1–20, Article 8167821. https://doi.org/10.1155/2022/8167821

Swamy, V., Radmehr, B., Krco, N., Marras, M., & Kaser, T. (2022). Evaluating the explainers: Black-box explainable machine learning for student success prediction in MOOCs. Presented at the *15th International Conference on Educational Data Mining*, Durham, England, July 24–27.

Thanoon, M. I. (2024). Artificial intelligence-based smart class attendance system: An IoT infrastructure. *International Journal for Quality Research, 18*(1), 187–198. https://doi.org/10.24874/IJQR18.01-12

Vuong, Q. (2021). Machine learning for robotic manipulation. *arXiv Preprint, arXiv-2101.00755*, 1–15. https://doi.org/10.48550/arXiv.2101.00755

Wolfram, S. (2023). *An elementary introduction to the Wolfram language* (3rd ed.). Champaign, IL: Wolfram Media.

Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., ... Sridhar, S. (2022). Neural fields in visual computing and beyond. *State of the Art Report, 41*(2), 1–36. https://doi.org/10.1111/cgf.14505

Xie, Y., Zaccagna, F., Rundo, L., Testa, C., Agati, R., Lodi, R., ... Tonon, C. (2022). Convolutional neural network techniques for brain tumor classification (from 2015 to 2022): Review, challenges, and future perspectives. *Diagnostics, 12*(8), 1–46, Article 1850. https://doi.org/10.3390/diagnostics12081850

Yun, B. I. (2019). A neural network approximation based on a parametric sigmoidal function. *Mathematics, 7*(3), 1–12, Article 262. https://doi.org/10.3390/math7030262

Zhao, F., & Huang, S. L. (2022). On the universally optimal activation function for a class of residual neural networks. *Applied Mathematics, 2*(4), 574–584. https://doi.org/10.3390/appliedmath2040033

Vladimir Milićević
Faculty of Mechanical and Civil Engineering in Kraljevo, University of Kragujevac,
Kraljevo, Serbia
milicevic.v@mfkv.kg.ac.rs
ORCID 0000-0002-5587-2717

Igor Franc
Faculty of Mechanical and Civil Engineering in Kraljevo, University of Kragujevac,
Kraljevo, Serbia
franc.i@mfkv.kg.ac.rs
ORCID 0009-0000-4609-1081

Maja Lutovac Banduka
RT-RK LLC (former Department of RT-RK Institute, Computer Based Systems), Belgrade, Serbia
maja.lutovac-banduka@rt-rk-com
ORCID 0000-0003-4446-706X

Nemanja Zdravković
Belgrade Metropolitan University,
Belgrade, Serbia
nemanja.zdravkovic@metropolitan.ac.rs
ORCID 0000-0002-2631-6308

Nikola Dimitrijević
Belgrade Metropolitan University,
Belgrade, Serbia
nikola.dimitrijevic@metropolitan.ac.rs
ORCID 0000-0002-6595-9277