www.sjm06.com

Serbian Journal of Management 21 (1) (2026) 37 - 48

Serbian Journal of Management

SATURATED LINEAR UNIT AS AN UNIVERSAL SYMMETRIC ACTIVATION FUNCTION FOR DEEP LEARNING

Maja Lutovac Banduka[a], Vladimir Milićević[b*], Igor Franc[b], Nemanja Zdravković[c] and Nikola Dimitrijević[c]

[a]RT-RK LLC (former Department of RT-RK Institute, Computer Based Systems), 2v Dunavska, 11158 Belgrade, Serbia
[b]Faculty of Mechanical and Civil Engineering, University of Kragujevac, Dositejeva 19, 36000 Kraljevo, Serbia
[c]Faculty of Information Technology, Belgrade Metropolitan University, Tadeuša Košćuška 63, 11158 Belgrade, Serbia

(Received 31 March 2025; accepted 14 April 2025)

Abstract

There is a number of symmetric activation functions used in artificial neural networks for deep learning. In this paper, we propose a universal activation function based on the Saturated Linear Unit (SaLU) that can be used instead of any known symmetric activation function. It is not necessary for classification tasks to have an exact calculation of the probability of detected classes. The classification decision is made based on the highest probability for the input values. We propose, as a proof of concept, that the two most commonly used hyperbolic tangent and algebraic sigmoid activation functions can be effectively replaced by SaLU by choosing a single parameter. Moreover, the theoretical step function can also be replaced by SaLU for a wider transition range. All derivations use symbolic processing. Also shown is a visualization of the range of inputs that result in a suitable classification. This can help scientists and programmers design complex machine learning algorithms and understand how deep learning algorithms work.

Keywords: artificial neural networks, closed-form solutions, classification decision, feature extraction, machine learning algorithms, deep learning

1. INTRODUCTION

Artificial Neural Networks (ANNs) have an acyclic learning process in which they adjust a large number of network parameters so that the desired outputs are correctly obtained from the inputs (Khan et al, 2018). Network propagation is usually the feed-

* *Corresponding author: milicevic.v@mfkv.kg.ac.rs*

DOI: 10.5937/sjm21-57891

forward with frozen ANN parameters. The feed-back propagation has the ability to memorize adapted network parameters using an appropriate algorithm. These two processes take place independently without affecting each other.

The last stage of the feed-forward part of the artificial neuron in the ANN is an activation nonlinear function with theoretically only two output values [fire node; not fire node], [on; off], [1; 0], or [1; 1]. The most common strategy is to use the first derivative of the activation function in the feed-back network. Neuron outputs become insensitive to the network parameters with a sharp transition between the two possible values. For example, suppose we have five possible neuron outputs, there are only five possible results regardless of the network parameters, see Table 1.

The output of activation function of the last layer is usually the so-called Softmax function:

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

Assuming the Softmax is applied after the activated function, the highest probability of any output is approximately 0.70, the lowest probability of any output is approximately 0.04.

Table 1. Probability using Softmax

Outputs $\{y_1, y_2, y_3, y_4\}$	Softmax $\{p_1, p_2, p_3, p_4\}$
{1, 1, 1, 1}	{0.25, 0.25, 0.25, 0.25}
{-1, 1, 1, 1}	{0.04, 0. 32, 0. 32, 32}
{-1, -1, 1, 1}	{0.06, 0.06, 0. 44, 0. 44}
{-1, -1, -1, 1}	{0.10, 0.10, 0.10, 0.70}
{-1, -1, -1, -1}	{0.25, 0.25, 0.25, 0.25}

We chose a symmetric activation function for the analysis because we expect the same contribution from positive and negative input values. If the input values are only positive, we use zero centering by subtracting the mean calculated value. Also, the input data of the train and test sets are divided by the same value so that the sum of the input values to the network is equal to the number of artificial neurons in the layer. In this way, we maximize the dynamic range of the ANN and use the same range of input values for all layers.

Therefore, in the rest of the work we use only hyperbolic tangent or algebraic sigmoid activation functions, as the most popular in practice. We avoid using any non-symmetric function (such as Sigmoid, ReLU, Noisy ReLU, Leaky ReLU/PReLU, Randomized Leaky ReLU, Exponential Linear Unit (Khan et al., 2018) because we expect the same contribution from positive and negative network input values.

We are looking for a function that is similar to ReLU in that it is fast to compute, but with a symmetric property. Note, the activation function should be limited to -1 for negative input values and to 1 for positive input values. In this way, the computation of the activation function is just passing the input to the function unit that is without any computation other than multiplying by a constant. The linear property of the activation function for values between -1 and 1 avoids sharp transitions.

2. MATERIALS

In this paper, we analyze two inputs and four outputs of an artificial neural network with two hidden and one output layer (deep learning requires at least two hidden layers) and four neurons per layer. We use the proof of concept only to demonstrate the main

properties. The neural network has its inputs x_1, x_2, x_3, and x_4, via input branches, but two inputs are set to 0, x_3=0 and x_4=0. The neuron of each layer L calculates the output $y_{(L,j)}$, $i \in \{1, 2, 3, 4\}$, which is sent to the other four neurons of the next layer. The outputs of the each neuron in a layer are the inputs to the next layer, which is for the ith neuron:

$$y_{L,i} = f_L(w_{L,1}x_{L,1}+w_{L,2}x_{L,2}+w_{L,3}x_{L,3}+w_{L,4}x_{L,4} + b_{L,i}) \quad (2)$$

The inputs to the network are inputs to the first layer.

$$x_{1,1} = x_{1,} \quad x_{1,2} = x_2 \quad x_{1,3} = x_3 \quad x_{1,4} = x_4 \quad (3)$$

The outputs of the layer $L \in \{1, 2, 3\}$ are inputs to the next layer.

$$L \in \{1,2,3\},\ i \in \{1,2,3,4\},\ x_{L+1,i} = y_{L,i}. \quad (4)$$

For a two inputs that are set to 0, x_3=0 and x_4=0, we can additionally specify some parameters, for example $w_{1,3}$=0, $w_{1,4}$=0, $b_{1,3}$=0, $b_{1,4}$=0. Note that the first layer has 16 network parameters (4 inputs × 4 neurons = 16 w parameters) and four bias parameters. (4 neurons = 4 b parameters).

A deep learning ANN with minimal 3 layers has in total, 3 layers×16=48 w parameters and 3 layers×4=12 b parameters, i.e., the analyzed ANN has the total of 48+12=60 parameters, which are changed during neural network learning stage. Some system parameters may be 0, but still the feedback propagation must adjust about 50 network parameters.

The activation function can be the same for all hidden layers f_L. The last activation function is usually Softmax function.

3. ENVIRONMENT

In (Bernard, 2022), a numerical example of an ANN with two inputs and four outputs, two hidden layers and an output layer and with a hyperbolic tangent activation function for the hidden layers and a Softmax function for the output layer is presented. In this paper, we use the same numerical values as in (Bernard, 2022). The programming code in (Bernard, 2022) is available in the Wolfram language (Wolfram, 2023). We adapt the visual programming of Add-Ons Schematic Solver application package (Lutovac & Tosic, 2014), which requires the software Mathematica ver. 9 (Wolfram, 2023), with built-in additional functions so that it works in Mathematica ver. 13 for ANN.

More details on symbolic ANN are available at (Milićević et al., 2025). An artificial neuron is first drawn and then copied three times to determine the layer; the layer is copied twice to create more than one layer. All ANN symbols must have unique values. The nonlinear activation function f is saved as a symbol.

Each element of an ANN can be replaced by another element, or a set of related elements, as described in (Lutovac Banduka & Lutovac, 2025). Each ANN symbol can be replaced by another symbol or expression.

4. METHODS

We propose a saturated activation function called Saturated Linear Unit (SaLU) which is similar to ReLU. A symmetric SaLU satisfies the following conditions:

$$f(0) = 0 \quad (5)$$

$$f(0) = 0 \tag{6}$$

$$-1 \le f(x) \le 1 \tag{7}$$

$$0 \le f'(x) \tag{8}$$

A linear property means a constant slope for -$a \le x \le a$, where a is a constant:

$$\begin{cases} f(x) = -1, & x < -a \\ f(x) = \dfrac{1}{2a}x, & -a \le x \le a \\ f(x) = 1, & a < x \end{cases} \tag{9}$$

The first derivative f'(x) is a constant 1/2a or 0:

$$\begin{cases} f'(x) = 0, & x < -a \\ f'(x) = \frac{1}{2a}, & -a \le x \le a \\ f'(x) = 0, & a < x \end{cases} \tag{10}$$

The introduced SaLU are illustrated in Figure 1. The three selected SaLUs and the most commonly used activation functions are presented in Figure 2. The first derivative of the functions plotted in Figure 2 is shown in Figure 3.

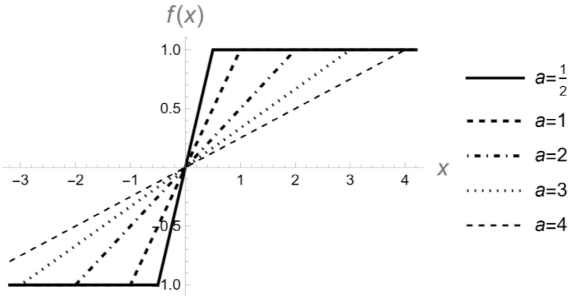

Figure 1. SaLU for different values of a

It follows from Figure 2 that the hyperbolic tangent activation function can be approximated by SaLU for a=1.49. Also, we can conclude that the algebraic sigmoid activation function can be approximated by SaLU for a=2.35.

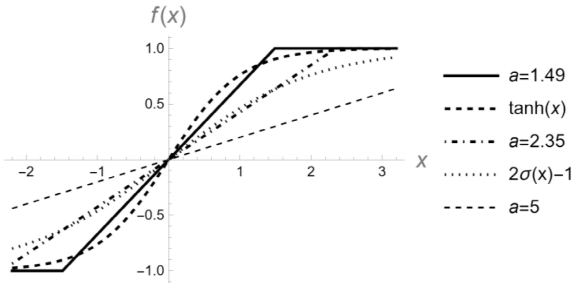

Figure 2. SaLU, hyperbolic tangent and algebraic sigmoid activation functions.

We can bring out a similar conclusion for the first derivative.

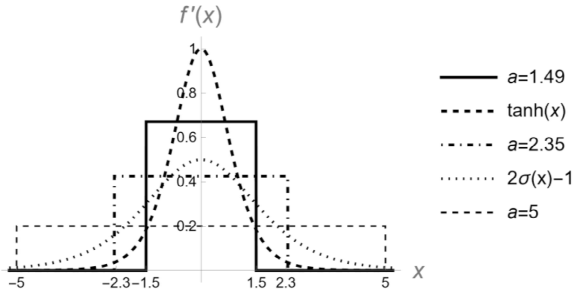

Figure 3. The first derivative of SaLU, hyperbolic tangent and algebraic sigmoid activation functions.

In the next section we will test the probability and show that approximately the same probability is obtained with the hyperbolic tangent activation function and SaLU for a=1.49. The same applies to the algebraic sigmoid activation function and SaLU for a=2.35.

Note that SaLU must be defined as a pure function.

5. RESULTS

Assume that the network propagation is a feed forward with frozen parameters obtained using a hyperbolic tangent activation function and known inputs. Feed-back propagation has the network parameters adjusted using an appropriate algorithm. All network parameters are frozen. Now we test the results using another activation function. The probabilities are shown in Table 2.

Table 2. probability using Softmax

Activation function	Softmax $\{p_1, p_2, p_3, p_4\}$
SaLU, a=1.49	{0.179, 0.033, 0.188, 0.600}
Hyperbolic Tangent	{0.184, 0.031, 0.184, 0.600}
Algebraic Sigmoid	{0.260, 0.061, 0.276, 0.403}
SaLU, a=2.35	{0.257, 0.060, 0.278, 0.405}
SaLU, a=5.00	{0.328, 0.098, 0.345, 0.229}

The SaLU probabilities for a=1.49 and the hyperbolic tangent activation function are approximately the same. The SaLU probabilities for a=2.35 and the algebraic sigmoid activation function are approximately the same. The same can be concluded from Figure 4.

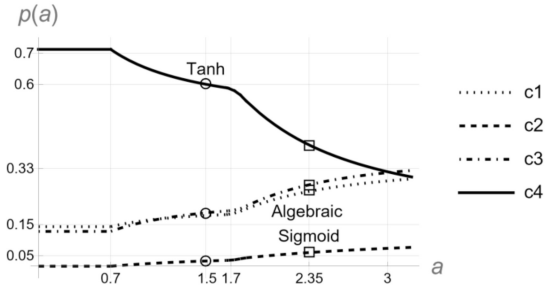

Figure 4. The probabilities of SaLU for parameter a from 0 to 3

The letter c indicates the corresponding class at the ANN output.

A very important feature emerges from Figure 4. The probabilities are constant for a from 0 to 0.7. This means that a sharp transition of the activation function is not a desirable solution, because the probability is constant for a wider range of a. Besides, the derivation is not possible for very small a.

Another important feature emerges from Figure 4. The probabilities are not differentiated for such that saturation is not applicable. For example, for $a \geq 5$, the input to the activation function is always between -5 and 5; the outputs of the previous layers contribute a maximum of ±4, plus ±1 bias parameter.

Note that in this example the ANN parameters are obtained for a hyperbolic tangent activation function. For the tested examples, the fourth output of the ANN has the highest probability (60%), but this does not mean that it will be the same for different inputs to the ANN.

The proper functioning of ANN is to separate the other classes, not just the fourth class. Therefore, we need to test the correct class detection for different inputs to the ANN.

6. VISUALIZATION

Assuming that the value of a is similar to the response when the activation function is the hyperbolic tangent (a=1.49), the 3D probability plot of SaLU is presented in Figure 5.

A benchmark value of 0.43 is used to separate the different detected classes. The probability is set to 0 for any class probability less than 0.43.

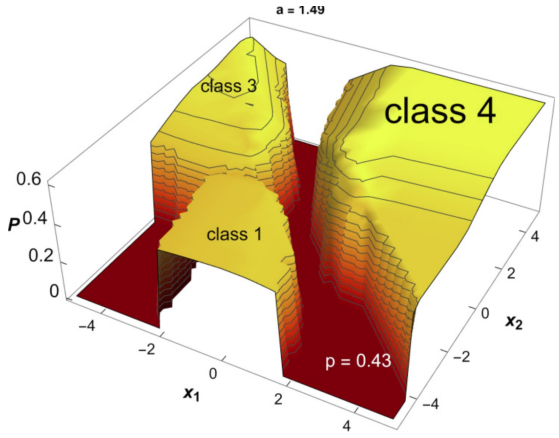

Figure 5. The 3D plot of probabilities of SaLU for parameter a=1.49

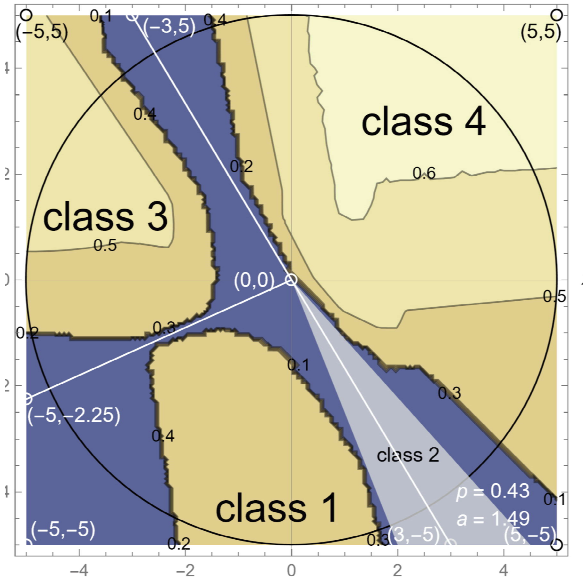

Figure 6. The contour plot of probabilities of SaLU for parameter a=1.49

7. MAPPING

Instead of calculating the probability for each new input to the ANN, we can map the input domain to the outputs of the detected classes. The contour diagram of probabilities for ANN parameters from (Bernard, 2022) is shown in Figure 6.

Prominent cases of ANN input are mapped to the corresponding regions. All detected classes have a probability greater than 43% for the benchmark value of 0.43. Note that class 2 is not detected with a higher probability than any other class.

We can divide the entire input domain by lines, so that for any inputs we can discover the corresponding class. Classes 1, 3 and 4 are bounded by a polygon with two-dimensional points (x_1, x_2)

c1 = {(-5,-5), (-5,-2.25), (0,0), (3,-5), (-5,-5)},
c3 = {(-5,-2.25), (-5,5), (-3,5), (-5,-2.25)},
c4 = { (3,-5), (0,0), (-3,5), (5,5), (5,-5), (3,-5)}.

Class 2 may have maximum values limited by points

c2 = {(2,-5), (0,0), (4.5,-5), (2,-5)}.

Let's test several inputs as shown in Table 3.

Table 3. Probability using Mapping

Input values (x_1, x_2, x_3, x_4)	Softmax $\{p_1, p_2, p_3, p_4\}$
(-2, -4, 0, 0)	{**0.43**, 0.10, 0.40, 0.07}
(2.3, -5, 0, 0)	0.36, **0.25**, 0.19, 0.20}
(-4, 2, 0, 0)	{0.32, 0.05, **0.54**, 0.09}
(2.5, 2.5, 0, 0)	{0.18, 0.03, 0.19, **0.60**}
(0, 0, 0, 0)	{0.30, 0.04, 0.26, **0.40**}

Class 2 has the highest probability, p=0.25, for input (2.3, -5, 0, 0), but is still smaller than class 1, p=0.36. Therefore, the mapping can be used for fast detection of classes only for classes 1, 3, and 4. Class 4 has the highest probability, p=0.40, for input (0, 0, 0, 0). Mapping using a polygonal description can be used to find the class most likely to occur, although we can expect a small error in the decision.

8. DISCUSSION

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). 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).

AI based on ANN with at least two hidden layers is called deep learning (Goodfellow et al., 2016). AI models as "black boxes" are often not comprehensible (Setzu et al., 2021). Transfer learning is a powerful approach to make machine learning models more transparent and understandable for humans (Wiratsin & Ragkhitwetsagul, 2025).

A review on deep learning-based smart processing, robotic sensor networks and data management algorithms is reported in (Lăzăroiu et al., 2022).

We can compare our results with the survey paper [A1] (Dubey et al., 2021) where a large number of state-of-the-art activation functions were analyzed:

1. Logistic Sigmoid,
   a. ESwish – E-Swish Sigmoid,
   b. ISigmoid – Improved logistic Sigmoid,
   c. PSF – Parametric Sigmoid Function,
   d. SiLU – Sigmoid-weighted Linear Unit,
   e. sSigmoid – Scaled Sigmoid,
   f. Swish – Swish Sigmoid,

2. Tanh,
   a. LiSHT – Linearly Scaled Hyperbolic Tangent,
   b. pTanh – penalized Tanh,
   c. P-TELU – Parametric Tan Hyperbolic Linear Unit,
   d. sTanh – scaled Hyperbolic Tangent,

3. ReLU – Rectified Linear Unit,
   a. AB-ReLU – Average Biased ReLU,
   b. BReLU – Bounded ReLU,
   c. CReLU – Concatenated ReLU,
   d. D-ReLU – Dynamic ReLU,
   e. DisReLU – Displaced ReLU,
   f. DualReLU – Dual ReLU,
   g. EReLU – Elastic ReLU,
   h. FReLU – Flexible ReLU,
   i. Leaky ReLU,
   j. L-ReLU – Lipschitz ReLU,
   k. MeLU – Mexican ReLU,
   l. NLReLU – Natural-Logarithm ReLU,
   m. PairedReLU for image super-resolution,
   n. PReLU – consider slope for negative input,
   o. ReLTanh – Rectified Linear Tanh,
   p. RMAF – ReLU-Memristor-like AF,
   q. RReLU – Randomized ReLU,
   r. RTReLU – Random Translation ReLU,
   s. vReLU – V-shaped ReLU,

4. ELU – Exponential Unit,
   a. CELU – Continuously differentiable ELU,
   b. EELU – Elastic ELU,
   c. ELiSH – Exponential Linear Sigmoid SquasHing,
   d. FELU – Fast ELU,
   e. HardELiSH – HardSigmoid and Linear,
   f. MPELU – Multiple PELU,
   g. PDELU – Parametric Deformable ELU,
   h. PELU – Parametric ELU,
   i. SELU – Scaled ELU,
   j. ShELU – Shifted ELU,

5. Learning Adaptive Activation Functions,
   a. APL – Adaptive Piecewise Linear,
   b. BLU – Bendable Linear Unit,
   c. PLU – Piecewise Linear Unit,

6. Mish – Miscellaneous Activation Functions,
   a. KAF – Kernel-based non-parametric AF,
   b. SAAF – Smooth Adaptive AF,

c. PAU – Padé Activation Unit,
d. Polynomial functions,
e. RePU – Rectified Power Unit,
f. GELU – Gaussian Error Linear Unit probabilistic functions,
g. SLU – Softplus Linear Unit,
h. RSP – Rand Softplus,

7. NLP – Natural Language Processing,

8. ReSech – Rectified Hyperbolic Secant,

9. Hexpo function,

10. Elliott activation functions,

11. SRS – Soft-Root-Sign,

12. MTLU – Multi-bin Trainable Linear Unit,

13. SAF – cubic spline interpolation,

14. BDAA – Bi-modal Derivative Adaptive Activation,

15. VAF – Variable Activation Function.

Problems that occur with unbalanced input signals can be solved by subtracting the mean value and scaling to the maximum value. Note, the output range of the SaLU activation function is between (-1, 1). Therefore, the input range of the activation function is multiplied by the number of neurons plus ±1 for the bias parameter. In this way, SaLU solves almost all the lacks of many activation functions.

According to [A1] (Dubey et al., 2021), ReLU has become the most modern AF because of its simplicity. Therefore, we suggest SaLU to use simplicity of the computation. The disadvantage of [A1] (Dubey et al., 2021) is that the number of nodes in a layer is not analyzed.

The introduction of linear saturated activation functions in a neural network as a universal approximation is presented in [A2] (Stursa at al., 2019). Experimental results [A2] (Stursa at al. 2019) showed that there are only small differences in performance and convergence speed between linear saturated activation functions and ReLU. The disadvantage of [A2] (Stursa at al., 2019) is that only one hidden layer is considered and that no relation of the slope of the function in relation to the number of neurons in the layer was found. Also, [A2] (Stursa at al., 2019) indicates that the error decreases with more neurons in the hidden layer. In a SaLU with at least two hidden layers, the number of neurons should be related to the slope of the activation function, so that 1/3 is limited only linearly, and the rest of the input to the activation function is saturated.

The algorithm should be capable of parallel processing [A3] (Wang et al., 2019). The SaLU approach provides parallel processing of each artificial neuron. This means that 4 parallel processes can be executed for each neuron in the layer. Since each output of the activation function is connected to all the artificial neurons of the next layer, all the computations of the previous layer should be completed before the computation of the next layer. This means that there is no need to have fewer neurons than in the layer with the highest number of neurons. All general multipliers, as well as the slope of the linearity of the activation function, can be replaced by a small number of adders and thereby speed up the calculation (Lutovac Banduka & Lutovac, 2024).

A saturated non-monotonic activation function was proposed in [A4] (Chen et al.,

2023). ReLU can be activated for positive inputs without distortion, while non-monotonic activation functions provide non-linearity for negative inputs to the activation function. Experiments show an improved classification that is in the range below 5%. If the input values for the ANN are multiplied by -1, the dominant part will be the negative inputs, not the positive ones. For example, we can rotate the image by 90 degrees or 180 degrees, or subtract all values by the same amount, before using any new algorithm, to prove correct classification. With SaLU and subtracting the mean from all inputs, we are sure that the algorithm works well.

The output layer has no SaLU activation function; so the maximum output value is 5 and the minimum value is -5. Then softmak is applied; the highest probability of the output is approximately 1.00, the lowest probability of the other outputs is approximately 0.00, as shown in the Table 4.

Table 4. Probability of the last layer using Softmax

Outputs {y1, y2, y3, y4}	Softmax {p1, p2, p3, p4}
{5, 5, 5, 5}	{0.25, 0.25, 0.25, 0.25}
{-5, 5, 5, 5}	{0.00, 0. 33, 0. 33, 33}
{-5, -5, 5, 5}	{0.00, 0.00, 0.50, 0.50}
{-5, -5, -5, 5}	{0.00, 0.00, 0.00, 1.00}
{-5, -5, -5, -5}	{0.25, 0.25, 0.25, 0.25}

Note that the input value for the Softmax is equal to the number of neurons in the last layer plus ±1 for the bias parameter. The bias and weighting parameters of the output layer should be adjusted according to the expected probability of the training examples.

For example, for SaLU and a=1.49, we should expect to have {179, 033, 188, 600} classes. If the number of those 1000 classes is different, we should change the parameters of the last layer and a, to obtain the corresponding probabilities.

In [A5] (Lau et al., 2018) it was shown that adaptive activation functions obtained the lowest misclassification rate because the ANN can learn through training examples. Saturated activation functions required pre-training to eliminate the vanishing gradient problem during ANN training. SaLU uses scaling of the input values by the number of neurons in the layer and any known algorithm to find an initial guess for the corresponding probability. An initial guess can be obtained by using random values for all parameters. If the initial guess is not satisfactory, new random values are used. Also, the value of a can be used as an adaptive parameter.

The event-driven H∞ control problem of continuous-time nonlinear systems related to a two-player zero-sum game is presented in [A6] (Yang et al., 2020). An adaptive architecture with a unique critical ANN is used. Note, SaLU can be used for discrete time as well as continuous time ANN. The value a can be used as adaptive parameter.

An algorithm for deep learning frameworks can be embedded as shown in [A7] (Lee et al., 2023). The SaLU algorithm is embedded in the software as plain text and can be modified according to specific reasons. SaLU code is not prepackaged software. An arbitrary number of neurons can be specified for each layer, and an arbitrary number of hidden layers can be used. The limits of the symmetric saturated function can be changed to values other than 1. The only prerequisite is that the activation function must be defined as a pure function.

A dropout neuron is introduced to prevent the over-fitting phenomenon in [A8] (Zhang et al., 2022). A schematic diagram of the dropout operation can be easily done in

SaLU by setting some neuron parameters to 0. All weighting parameters are listed as symbols, and any number can be set using rule substitution.

Some nonlinear process applications have shown the use of different ANN models [A9] (Ren et al., 2022). Note that a nonlinear system cannot use frequency transformation. The model is represented by schematics and workflow. Obviously, drawing a scheme consisting of a hundred elements is not so easy to do by hand. Therefore, our algorithm for programming the schematic description of large ANNs is the only acceptable solution. The schematic description of our algorithm contains everything needed to draw the schematic, to calculate the ANN response, to perform the mathematical expression.

The most popular approach to using 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 black-box models usually do not have the skills to understand learning ANNs to solve practical problems. Therefore, they need a different approach. One such approach is presented in this paper

Instead of using a black-box model, a researcher can visually identify the architecture of an ANN and perform symbolic analysis to investigate the impact of each constituent part of the ANN.

The proposed solution is based on symbolic manipulations. By means of a simple transformation, the system's response to symbolic ANN inputs generates an output. 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 for inputs, ANN parameters, and the activation function.

Therefore, this original method is a kind of knowledge-based system. For example, in a three-layer 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 or another parameter based on expected response of the ANN.

During the testing or validation process, system parameters can be viewed for verification purposes. 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.

9. CONCLUSION

An original algorithm for implementing artificial 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 visualization. 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.

Acknowledgment

Parts of this research were funded by The Ministry of Science, Technological Development and Innovation of the Republic of Serbia, Contract no. 451-03-137/2025-03/200108.

ЗАСИЋЕНА ЛИНЕАРНА ЈЕДИНИЦА КАО УНИВЕРЗАЛНА СИМЕТРИЧНА АКТИВАЦИОНА ФУНКЦИЈА ЗА ДУБИНСКО УЧЕЊЕ

Маја Лутовац Бандука, Владимир Милићевић, Игор Франц, Немања Здравковић, Никола Димитријевић

Извод

Постоји више симетричних активационих функција које се користе у вештачким неуронским мрежама за дубинско учење. У овом раду предлажемо универзалну активациону функцију засновану на засићеној линеарној јединици (SaLU), која се може користити као замена за било коју познату симетричну активациону функцију. За задатке класификације није неопходно прецизно израчунавање вероватноће детектованих класа. Одлука о класификацији доноси се на основу највеће вероватноће за дате улазне вредности. Као доказ концепта, показујемо да се две најчешће коришћене активационе функције — хиперболички тангенс и алгебарски сигмоид — могу ефикасно заменити функцијом SaLU избором једног параметра. Штавише, теоријска степ-функција такође може бити замењена функцијом SaLU, уз шири опсег прелазне области. Сви изводи заснивају се на симболичкој обради. Такође је приказана визуелизација опсега улазних вредности које доводе до одговарајуће класификације. Ово може помоћи научницима и програмерима у пројектовању сложених алгоритама машинског учења и у разумевању начина рада алгоритама дубинског учења.

Кључне речи: вештачке неуронске мреже, решења у затвореном облику, одлука о класификацији, екстракција карактеристика, алгоритми машинског учења, дубинско учење

References

Khan, S., Rahmani, H., Shah, S., & Bennamoun, M. (2018). A Guide to Convolutional Neural Networks for Computer Vision, Switzerland, Springer Nature, 2018, 54-56.

Bernard, E. (2022). Introduction to Machine Learning, Champaign, IL, USA: Wolfram Media

Wolfram, S. (2023). An Elementary Introduction to the Wolfram Language, 3rd ed., Champaign, IL, USA: Wolfram Media

Lutovac, M.D., & Tosic, D. (2014). Schematic Solver, Mathematica application package, symbolic signal processing, software implementation, mouse driven interactive drawing tool, https://library.wolfram.com/infocenter/TechNotes/4814/

Milićević, V., Franc, I., Lutovac Banduka, M., Zdravković, N., & Dimitrijević, N. (2025). Symbolic analysis of classical neural networks for deep learning, International Journal for Quality Research, 19 (1), 85-100.

Lutovac Banduka, M., & Lutovac, M. (2024). Multiplierless neural networks for deep learning, in Proc. 13th Mediterranean Conference on Embedded Computing, Budva, Montenegro, Jun. 11-14, 2024, 262-265.

Zhao, F., & Huang, S. L. (2022). On the universally optimal activation function for a class of residual neural networks, AppliedMath, 2 (4), 574-584.

Kamienny, P.A., d'Ascoli, S., Lample, G.,

& Charton, F. (2022). End-to-end symbolic regression with transformers," presented at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022), New Orleans, USA, Nov. 28th - Dec. 9th 2022, Adv. Neural Inf. Proc. Syst., 35, 10269-10281.

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

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, 103457, 294, 1-15

Wiratsin, O., & Ragkhitwetsagul, C. (2025). Effectiveness of Explainable Artificial Intelligence (XAI) Techniques for Improving Human Trust in Machine Learning Models: A Systematic Literature Review, IEEE Access, 13, 121326-121350, 2025, doi: 10.1109/ACCESS.2025.3575022

Lăzăroiu, G., Andronie, M., Iatagan, M., Geamanu, M., Stefanescu, R., & Diamarescu, I. (2022). Review deep learning-assisted smart process planning, robotic wireless sensor networks, and geospatial big data management algorithms in the internet of manufacturing things, ISPRS International Journal of Geo-Information, 11 (5), 277.

Dubey, S.R., Singh S.K., & Chauduri, B.B. (2021). Activation functions in deep learning: A comprehensive survey and benchmark, arXiv preprint arXiv:2109.14545.

Stursa, D. & Dolezel P. (2019). Comparison of ReLU and linear saturated activation functions in neural network for universal approximation, 22nd International Conference on Process Control (PC19), 11-14 June 2019, Strbske Pleso, Slovakia

Wang, J.H., Cao, J., Li, W., Yu, P., & Huang, K. (2019). A novel parallel accelerated CRPF algorithm," Applied Intelligence, 50 (3), 849-859.

Chen, J., & Pan. Z. (2023). Saturated Non-Monotonic Activation Functions, arXiv preprint arXiv:2305.07537.

Lau, M.M., & Hann Lim, K. (2018). Review of Adaptive Activation Function in Deep Neural Network," 2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES). 03-06 December 2018, Sarawak, Malaysia

Yang, X., Gao, Z., & Zhang, J. (2020). Event-driven H∞ control with critic learning for nonlinear systems, Neural Networks, 132, 30-42.

Lee, Y., Park, C., & Kang, S. (2023). Deep Embedded Clustering Framework for Mixed Data, IEEE Access, 11, 33-40.

Zhang, N., Zhao, J., Ma. L., Kong, H., & Li, H. (2022). Tool Wear Monitoring Based on Transfer Learning and Improved Deep Residual Network, IEEE Access, 10, 119546-119557.

Ren, Y.M., Alhajeri, M.S., Luo, J., & Chen, S. (2022). A tutorial review of neural network modeling approaches for model predictive control, Computers & Chemical Engineering, 165 (3), 107956.