Comparison of ReLU and linear saturated activation functions in neural network for universal approximation

Dominik Stursa
Faculty of Electrical Engineering and Informatics
University of Pardubice
Pardubice, Czech Republic
dominik.stursa@student.upce.cz

Petr Dolezel
Faculty of Electrical Engineering and Informatics
University of Pardubice
Pardubice, Czech Republic
petr.dolezel@upce.cz

***Abstract*—Activation functions used in hidden layers directly affect the possibilities for describing nonlinear systems using a feedforward neural network. Furthermore, linear based activation functions are less computationally demanding than their nonlinear alternatives. In addition, feedforward neural networks with linear based activation functions can be advantageously used for control of nonlinear systems, as shown in previous authors' publications. This paper aims to compare two types of linear based functions - symmetric linear saturated function and the rectifier linear unit (ReLU) function as activation functions of the feedforward neural network used for a nonlinear system approximation. Topologies with one hidden layer and the combination of defined quantities of hidden layer neurons in the feedforward neural network are used. Strict criteria are applied for the conditions of the experiments; specifically, the Levenberg-Marquardt algorithm is applied as a training algorithm and the Nguyen-Widrow algorithm is used for the weights and biases initialization. Three benchmark systems are then selected as nonlinear plants for approximation, which should serve as a repeatable source of data for testing. The training data are acquired by the computation of the output as a reaction to a specified colored input signal. The comparison is based on the convergence speed of the training for a fixed value of the error function, and also on the performance over a constant number of epochs. At the end of the experiments, only small differences between the performance of both applied activation functions are observed. Although the symmetric linear saturated activation function provides the lesser median of the final error function value across the all tested numbers of neurons in topologies, the ReLU function seems to be also capable of use as the activation function for nonlinear system modeling.**

***Index Terms*—Feedforward neural network, linear saturated activation function, rectified linear activation function, nonlinear system identification**

I. Introduction

Various successful applications of artificial neural networks exist these days. For instance, neural networks are applied in process identification, function approximation, pattern recognition, time series prediction and many other examples through the various fields, as summarized in [1] or in [2],

Generally, an artificial neural network is a complex structure with many parameters needed to be selected in order to find the best behavior and results. For example, topology, method of weight initialization, particular activation function selection, and training algorithm implementation.

New various topologies of neural networks have been proposed and tested recently. However a classical feedforward neural network still remains as one of the most popular architectures. One of the main advantages of the feedforward neural network architecture is the huge set of methods for training, starting with a well-known backpropagation training algorithm introduced by [3] and Levenberg-Marquardt algorithm, which is considered as one of the most effective algorithms for batch training, as presented in [4]. In addition, the process of the selection of suitable topology is well examined and naturally causes better efficiency of specific applications.

Every neuron in an artificial neural network is characterized, among others, by its own activation function. For the learning process, most of algorithms need to know a derivation of the activation function in order to work sufficiently. Thus, it is a good practice to implement continuous and smooth types of activation functions. However, authors in [5], [6] showed that a particular selection of the activation function can provide a very suitable tool for control of nonlinear systems. To be more specific, if a piecewise linear activation function is used, the neural network is able to provide a linear model of the nonlinear system in a very effective way. This model can be then used for local tuning of any type of controller.

Another significant effect to the performance of the network is caused by the initialization of weights at the beginning of the training. Since sophisticated methods mostly do not provide stable performances generally, the initial weights are settled either randomly, or using some simple methods as can be found in [7].

The main idea of article is based on possibility for nonlinear system approximation by an artificial neural network with linear based activation functions. Specifically, the concept was introduced in [5] and [6], where symmetric linear saturated functions as activation functions are proposed. However, other possibilities are available, too. Thus, the application of ReLU activation is naturally suggested, since ReLU functions are often used in various other topologies of neural networks with

positive results.

The paper is structured as follows. Firstly, the problems to be solved are formulated in next section. Then, the conditions and parameters of the experiments are comprehensively defined and, at the end of the paper, the results are discussed.

II. Problem formulation

As an addition to the approximation procedure with piecewise-linear feedforward neural network introduced in [6], where a symmetric linear saturated activation function is used in hidden layer, a ReLU activation function is considered in this contribution, since it can be used for a nonlinear system approximation in a very similar way. In addition, its evaluation is probably even more efficient in comparison to a symmetric linear saturated activation function. The performance results between both approaches are then compared.

The standard multilayer feedforward neural network with only one hidden layer is capable of approximating any mathematical function with a specified accuracy as proven in [8]. According to that source, hidden layer neurons need to contain squashing activation functions and the output layer neuron should contain a non-squashing activation function. In the use of practical applications, monotonic and continuous functions in hidden layer neurons are mostly used, as recommended in [9]. For the purposes of this research, a symmetric linear saturated activation function and the ReLU activation function are considered for neurons in the hidden layer, and the linear identical activation function is always used in the output layer. Therefore, the mentioned premises are fulfilled, although the ReLU function is just squashed from below.

The results published in [6] indicate that the approximations of a nonlinear system with a feedforward neural network, when linear based activation functions are implemented, are achievable. In the mentioned research, only the symmetric linear saturated activation function is considered. The aim of this paper is to propose an approach with the ReLU activation function instead of the symmetric linear saturated activation function, since it seems to be even less computationally demanding. However, its approximation qualities for the purposes of modeling of nonlinear dynamical systems need to be examined. Therefore, as the initial step in this field, a comparison of their efficiency and performance for three benchmark systems is considered.

A. Tested activation functions

As mentioned above, two linear based activation functions for the purposes of approximation of benchmark systems are examined in this paper. Although these activation functions are not as robust and suitable for approximation as some smooth nonlinear activation functions, they are less computationally demanding and the main reasons for using them in automation and process control are described in [5] and [6].

1) Symmetric linear saturated activation function: This activation function is proposed in [5] as a crucial part of the artificial network called a piecewise-linear neural network (PWLNN).

The symmetric linear saturated activation function is defined as follows.

$$y_i = \begin{cases} 1 & \text{for} \quad y_{a,i} > 1 \\ y_{a,i} & \text{for} \quad -1 \leq y_{a,i} \leq 1 \\ -1 & \text{for} \quad y_{a,i} < 1 \end{cases}, \tag{1}$$

where y_i is the output from the activation function while $y_{a,i}$ is its input.

2) ReLU activation function: Rectified linear unit (ReLU) used as an activation function performs a threshold operation for every input value, where any values less than zero is set to zero. For the input value bigger or equal to zero it acts as a linear identical function. The ReLU activation function is defined as follows.

$$y_i = \begin{cases} y_{a,i} & \text{for} \quad y_{a,i} \geq 0 \\ 0 & \text{for} \quad y_{a,i} < 0 \end{cases}, \tag{2}$$

where y_i is the output from the activation function while $y_{a,i}$ is its input.

B. Experiment procedure

The neural network and its design is a complex procedure, which involves initialization of weights and biases, training and testing data set acquisition, and pruning and validating of a neural network.

Initialization of weights and biases can significantly affect the training speed and even a final state of the trained neural network. In our experiments, the Nguyen-Widrow method is implemented as defined in [7].

The data sets for our experiments consist of input and output values of three nonlinear benchmark systems and they are described in the following section.

Generally, neural network training means using a set of observations to find optimal (in some sense) values of weights and biases of a trained neural network. In this paper, the Levenberg-Marquardt training algorithm is used, since it is broadly considered as one of the most effective algorithms for a lot of the applications as described in [10].

Optimization of the topology is not considered here and various topologies are presented below. In addition, feedforward neural networks with just one hidden layer and a set of specified numbers of neurons are used for the purposes of our experiments, since this type of topology is, in general, capable enough to approximate any mathematical function.

Two sets of experiments are performed - speed during training for a fixed error function value and a training performance over a defined constant number of epochs.

1) Performance: Feedforward neural networks, settled with mentioned approaches, are trained over a defined number of epochs and the final error function value is observed at the end of the training procedure.

2) Convergence speed: In this type of experiment, various network topologies are trained to achieve a defined goal, (a specific error value), while the number of epochs needed to achieve the goal is measured.

III. Conditions of the numerical experiments

A. Data for training

Three benchmark nonlinear systems (adapted from [11]), each with one input and one output, are chosen as systems for identification. These nonlinear systems are described by the following equations.

$$y(n) = 0.9x(n) + 0.8sin(\pi x(n)) + 0.5cos(\pi x(n-1)) + 0.1sin(\pi x(n-2)) + 0.08cos(\pi x(n-4)) - 0.3cos(\pi x(n-15)), \quad (3)$$

$$y(n) = \frac{y(n-1)}{1+y^2(n-1)} + |x(n)|^3, \quad (4)$$

$$y(n) = 0.6sin^3(x(n)) - 0.1cos(4\pi x(n-4)) + 1.21. \quad (5)$$

The training data for each system are acquired by the computation of the output in reaction to a specified colored input signal. The colored input signal is generated by passing white noise with zero mean and variance equal to 0.001 through the following model (used in [11])

$$M(z) = \frac{1 + 0.5z^{-1} + 0.81z^{-2}}{1 - 0.59z^{-1} + 0.4z^{-2}}. \quad (6)$$

Experiments are performed with three datasets where each one is composed by the set of input values and corresponding set of output values. In the first and the third system, only previous and actual values of the input signal in combination with the actual output value are used. The second system output depends on the previous output value and the actual input value.

The data from the datasets are divided into three subsets for neural network training process purposes. The training set contains 70 % of the samples, while both testing and validation sets contain 15 % of the data.

B. Used topologies

Feedforward neural networks with one hidden layer are used for training. The exact topologies are shown in Fig. 1. The number of hidden layer neurons is variable for all sets of trainings. The number of hidden layer neurons is continuously being increased in the sequence of 2, 3, 4, 6, 8, 10 and 15 neurons for every dataset. Thus, for example two neurons in the hidden layer are used for the first training set for each system.

C. Parameters for the experiments

The Levenberg-Marquardt algorithm is used for all trainings and the Nguyen-Widrow algorithm is implemented to set the initial weights and biases of the networks.

Both types of experiments are related to the mean square error, defined as follows.

$$E_{val} = \frac{1}{N}\sum_{i=1}^{N}[o(i) - y(i)]^2, \quad (7)$$

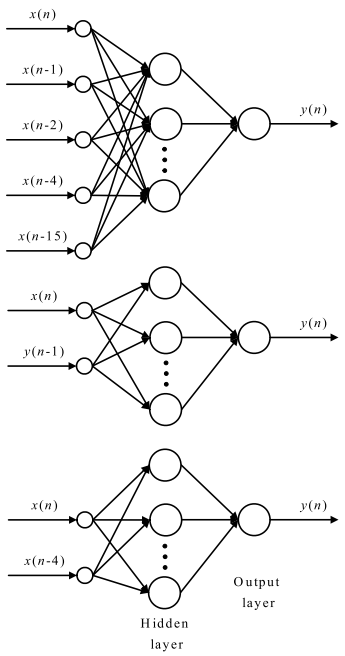

Fig. 1. Topologies used in experiments

where N is the number of the samples in the validation set, $o(i)$ is the i^{th} desired output and $y(i)$ is the actual output from the net.

As mentioned above, two types of experiments are considered in this research. The parameters of the first type (performance) are summarized in Table I, while the second type experiment (convergence speed) parameters are shown in Table II.

IV. Results and discussion

The experiments according to the information summarized in the previous section are performed using a special hardware suitable for parallel processing. In particular, the Jetson AGX Xavier Developer Kit is used - see Fig. 2.

As a development tool, Python 3.6 with TensorFlow library (see [12]) is implemented. The results of the experiments are shown in Figs. 3 to 8 below. The central lines in the box graphs, shown in the figures, are medians; the edges of the boxes are 25^{th} and 75^{th} percentiles; and the whiskers extend to

TABLE I
Parameters for the performance experiments

Training algorithm	Levenberg-Marquardt algorithm
Initialization	Nguyen-Widrow method
Number of experiments	200
Number of samples	10000
Maximum epochs	1000
Desired value of E_{val}	0
Stopping criterion	Maximum epochs reached
Adaptive coefficient μ	0.001
Increment μ	10
Decrement μ	0.1

Fig. 2. Jetson AGX Xavier Developer Kit

the most extreme data points (except outliers). In the captions of the figures, SLS means symmetric linear saturated activation function, ReLU means rectified linear unit activation function.

The results of the performance experiments are also summarized in Table III. Medians for each system and both activation functions are showed there. Looking at the results, one can say, that both activation functions lead to the successful approximation of every system. For the first and second system, the error decreases with a higher number of hidden layer neurons. Comparing medians in these two systems, the symmetric linear saturated function provides slightly better results. Medians of the third system, through all topologies and both activation functions, are very similar.

Table IV summarizes the results of the second set of the experiments. Medians for each system and both activation functions are shown in the table. The required goal is not always met only for the third system, where epochs needed to train the FFNN are increasing with more neurons in the hidden layer. For other systems both activation functions provide very similar results.

V. Conclusion

The complexity of training of the neural networks with linear based activation functions is discussed in this paper. To be more specific, a comparison of symmetric linear saturated and rectified linear unit activation functions for the purposes of nonlinear system approximation is examined. The experiments indicate that there are only small differences in both performance and convergence speed between the results of the compared activation functions. Therefore, the ReLU function seems to be capable of nonlinear system modeling as appropriately as the symmetric linear saturated activation function. This observation could bring a decent computational complexity decrease in process control and automation applications.

TABLE II
Parameters for the convergence speed experiments

Training algorithm	Levenberg-Marquardt algorithm
Initialization	Nguyen-Widrow method
Number of experiments	200
Number of samples	10000
Maximum epochs	1000
Desired E_{val} for system (3)	$9 \cdot 10^{-6}$
Desired E_{val} for system (4)	$1 \cdot 10^{-6}$
Desired E_{val} for system (5)	$8 \cdot 10^{-4}$
Stopping criterion	Desired E_{val} reached
Adaptive coefficient μ	0.001
Increment μ	10
Decrement μ	0.1

Acknowledgment

The work has been supported by the IGA Funds of the University of Pardubice, Czech Republic. This support is very gratefully acknowledged.

References

[1] A. Kramer and F. Morgado-Dias, "Applications of artificial neural networks in process control applications: A review," in *2018 International Conference on Biomedical Engineering and Applications (ICBEA)*, July 2018, pp. 1–6.

[2] Y. Li and W. Ma, "Applications of artificial neural networks in financial economics: A survey," in *2010 International Symposium on Computational Intelligence and Design*, vol. 1, Oct 2010, pp. 211–214.

[3] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[4] M. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, Nov 1994.

[5] P. Dolezel and I. Taufer, "Piecewise-linear artificial neural networks for pid controller tuning," *Acta Montanistica Slovaca*, vol. 17, no. 3, pp. 224–233, 2012.

[6] P. Dolezel and J. Heckenbergerova, "Computationally simple neural network approach to determine piecewise-linear dynamical model," *Neural Network World*, vol. 27, no. 4, pp. 351–371, 2017.

[7] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," 1990, pp. 21–26.

[8] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[9] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999, iSBN: 0132733501.

[10] S. Kollias and D. Anastassiou, "An adaptive least squares algorithm for the efficient training of artificial neural networks," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 8, pp. 1092–1101, 1989.

[11] S. Zhang and W. X. Zheng, "Mean-square analysis of multi-sampled multiband-structured subband filtering algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 3, pp. 1051–1062, March 2019.

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

TABLE III
MEDIANS OF RESULTING E_{val} FOR PERFORMANCE EXPERIMENTS

Neurons	2	3	4	6	8	10	15	15
System (3) with SLS	7.59E-05	2.92E-05	1.82E-05	1.09E-05	6.97E-06	4.37E-06	1.99E-06	1.99E-06
System (3) with ReLU	1.07E-04	6.40E-05	2.70E-05	1.22E-05	7.61E-06	4.71E-06	2.19E-06	2.19E-06
System (4) SLS	3.09E-06	2.44E-06	7.05E-07	2.49E-07	1.06E-07	5.18E-08	1.77E-08	1.77E-08
System (4) with ReLU	2.11E-05	2.82E-06	9.29E-07	7.01E-07	2.56E-07	2.08E-07	5.70E-08	5.70E-08
System (5) with SLS	8.84E-04	8.77E-04	8.77E-04	8.75E-04	8.78E-04	8.75E-04	8.73E-04	8.73E-04
System (5) with ReLU	8.80E-04	8.76E-04	8.79E-04	8.72E-04	8.72E-04	8.74E-04	8.78E-04	8.78E-04

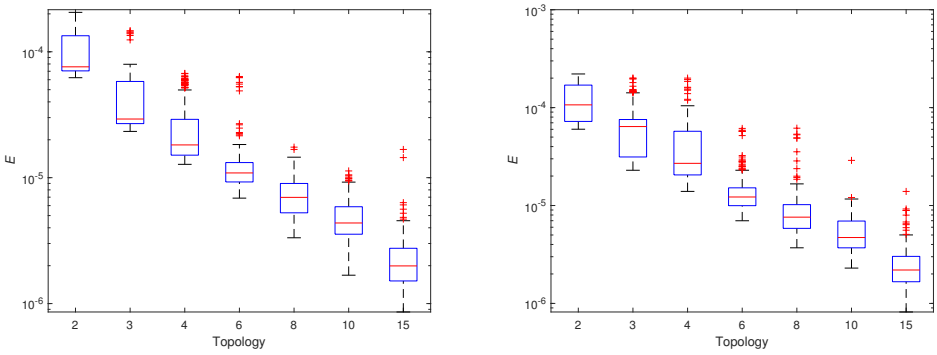

Fig. 3. Final values of E_{val} for system (3). Neural networks with SLS (left) and with ReLU (right).

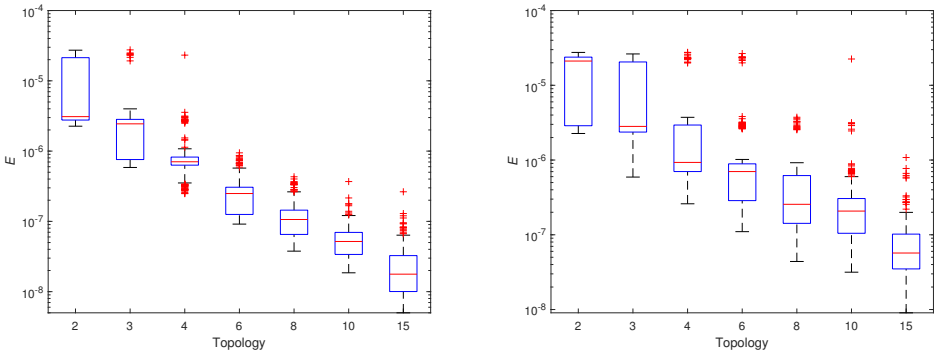

Fig. 4. Final values of E_{val} for system (4). Neural networks with SLS (left) and with ReLU (right).

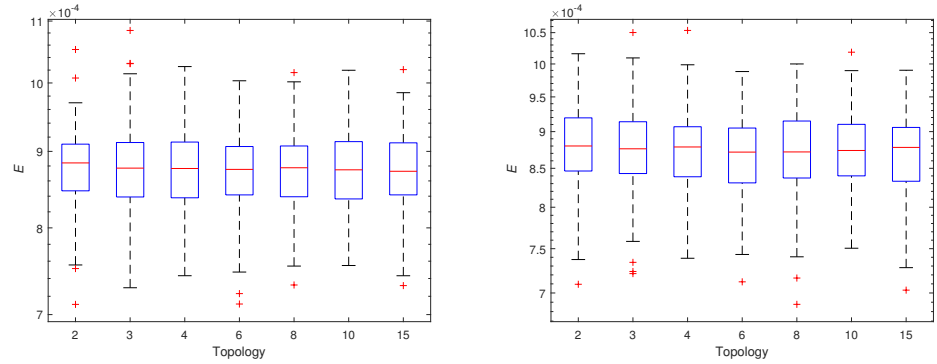

Fig. 5. Final values of E_{val} for system (5). Neural networks with SLS (left) and with ReLU (right).

TABLE IV
MEDIANS OF RESULTING NUMBERS OF EPOCHS FOR CONVERGENCE SPEED EXPERIMENTS

Neurons	2	3	4	6	8	10	15
System (3) with SLS	42	51	67	76	13	9	8
System (3) with ReLU	42	53	59	70	17	10	8
System (4) with SLS	32	21	10	8	7	7	6
System (4) with ReLU	28	28	17	10	7	7	6
System (5) with SLS	72	88	102	114	152	190	254
System (5) with ReLU	44	49	65	77	111	159	295

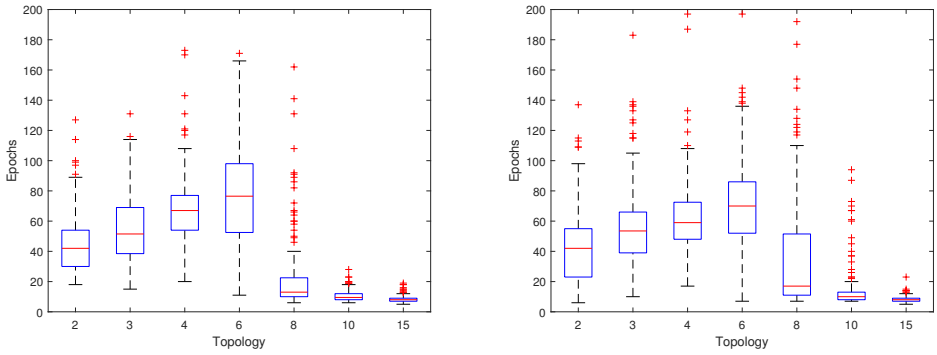

Fig. 6. Final numbers of epochs for system (3). Neural networks with SLS (left) and with ReLU (right).

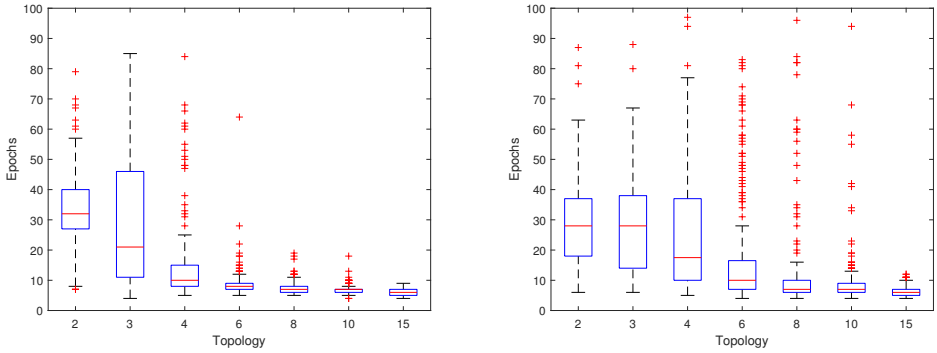

Fig. 7. Final numbers of epochs for system (4). Neural networks with SLS (left) and with ReLU (right).

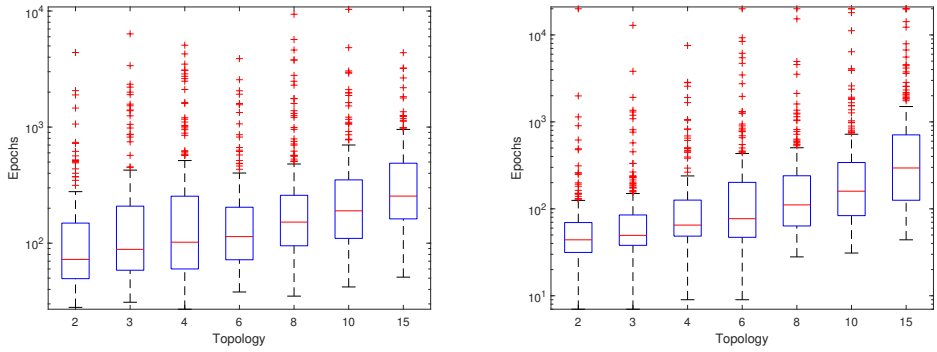

Fig. 8. Final numbers of epochs for system (5). Neural networks with SLS (left) and with ReLU (right).