Function Approximation Performance of Fuzzy Neural Networks

Rita Lovassy[1], László T. Kóczy[2,3], László Gál[2,4]

[1] Inst. of Microelectronics and Technology, Kandó Kálmán Faculty of Electrical Engineering, Óbuda University, Budapest, Hungary

[2] Inst. of Informatics, Electrical and Mechanical Eng., Faculty of Engineering Sciences, Széchenyi István University, Győr, Hungary

[3] Dept. of Telecommunication and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary

[4] Dept. of Technology, Informatics and Economy, University of West Hungary, Szombathely, Hungary

e-mail: lovassy.rita@kvk.uni-obuda.hu, koczy@sze.hu, koczy@tmit.bme.hu, laci.gal@gmail.com

Abstract: In this paper we propose a Multilayer Perceptron Neural Network (MLP NN) consisting of fuzzy flip-flop neurons based on various fuzzy operations applied in order to approximate a real-life application, two input trigonometric functions, and two and six dimensional benchmark problems. The Bacterial Memetic Algorithm with Modified Operator Execution Order algorithm (BMAM) is proposed for Fuzzy Neural Networks (FNN) training. The simulation results showed that various FNN types delivered very good function approximation results.

Keywords: fuzzy flip-flop neurons; Fuzzy Neural Networks; Bacterial Memetic Algorithm with Modified Operator Execution Order

1 Introduction

Over the last few decades many different approaches to the hybridization of neural networks and fuzzy systems have been introduced and studied [9], [11], [22] as new neuro-fuzzy structures. Based on this idea, in this paper the concept of the fuzzy flip-flop neuron is introduced. The function approximation capability of the novel Fuzzy Flip-Flop-based Neural Networks (FNN), as a new type of neural networks is studied. A comparison of the effect of applying some selected fuzzy operations in the investigation of the fuzzy flip-flop (F^3)-based neurons, and the Multilayer Perceptrons (MLPs) based on them, are presented. The proposed

network is a structure consisting of the same types of F^3. The proposed training method is a particular combination of evolutionary and gradient based (global and local search) algorithms. The effect of fuzzy operations types and fuzzy neurons numbers are studied. Illustrative examples are presented in order to demonstrate the success of this work in terms of the function approximation capability of the proposed FNNs.

The outline of this paper is as follows. In Section II we present the fuzzy J-K and D flip-flops in general. The Fuzzy Flip-Flop based Neural Network as a novel implementation possibility of multilayer perceptron neural networks is proposed in Section III. The FNNs neuron element may be any fuzzy flip-flop derived from the original F^3 with two additional modifications (feedback and fixed internal state) with more or less sigmoid transfer characteristics. After simplifications the fuzzy J-K and D flip-flop neurons block diagram for a fix Q value is given. The Bacterial Memetic Algorithm with Modified Operator Execution Order (BMAM) is carried out in Section IV. We apply this method to demonstrate that the proposed FNNs built up from fuzzy J-K and D flip-flops based on algebraic, Łukasiewicz, Yager, Dombi and Hamacher operations can be used for learning and approximating a battery cell charging characteristics, two dimensional trigonometric functions, and two and six dimensional benchmark problems. The target activation function is *tansig*, a MATLAB built-in sigmoid transfer function. Finally, the FNNs function approximation performance comparison thought simulation results are discussed in Section V, followed by a brief Conclusion and References.

2 Fuzzy J-K and D Flip-Flops

The concept of fuzzy flip-flop was introduced in the middle of 1980's by Hirota (with his students) [7]. The Hirota Lab recognized the essential importance of the concept of a fuzzy extension of a sequential circuit and the notion of fuzzy memory. From this point of view they proposed alternatives for "fuzzifying" digital flip-flops. The starting elementary digital units were the binary J-K flip-flops. Their definitive equation was used both in the minimal disjunctive and conjunctive forms. As fuzzy connectives do not satisfy all Boolean axioms, the fuzzy equivalents of these equations resulted in two non-equivalent definitions, "reset and set type" fuzzy flip-flops, using the concepts of fuzzy negation, t-norm and t-conorm operations. In [8] Hirota et al. recognized that the reset and set equations cannot be easily used as elements of memory module, because of their asymmetrical nature. In their paper [19] Ozawa, Hirota and Kóczy proposed a unified form of the fuzzy J-K flip-flop characteristic equation, involving the reset and set characteristics, based on min-max and algebraic norms. A few years later, the hardware implementation of these fuzzy flip-flop circuits in discrete and continuous mode was presented in [20]. The next state Q_{out} of a fuzzy J-K flip-

flop is characterized as a function of both the present state Q and the two present inputs J and K. The unified formula of the fuzzy J-K flip-flop was expressed as follows [19]:

$$Q_{out} = (J \vee \overline{K}) \wedge (J \vee Q) \wedge (\overline{K} \vee \overline{Q}) \tag{1}$$

The over bar denotes the standard negation (e.g. $\overline{K} = 1 - K$); furthermore, $\wedge$ and $\vee$ denote fuzzy operations (t-norm and t-conorm, labeled in the next as *i* and *u*). $J, K, Q, Q_{out} \in [0,1]$. In [12] a F^3 derived from fuzzy J-K flip-flop where $\overline{Q}$ is fed back to *K* (*K* = 1 - *Q*) is proposed. The characteristic equation of a fuzzy J-K flip-flop with feedback is

$$Q_{out} = (J \; u \; Q) \; i \; (J \; u \; Q) \; i \; (Q \; u \; (1-Q)) \tag{2}$$

The concept of a novel fuzzy D flip-flop type was introduced in [13]. Connecting the inputs of the fuzzy J-K flip-flop in a particular way, namely by applying an inverter in the connection of the input *J* to *K*, case of *K* = 1 - *J*, a fuzzy D flip-flop is obtained. Substitute $\overline{K} = J$ in equation (1) and let *J* = *D*, the fundamental equation of fuzzy D flip-flop is

$$Q_{out} = (D \; u \; D) \; i \; (D \; u \; Q) \; i \; (D \; u \; (1-Q)) \tag{3}$$

In our previous papers [13], [14] the unified fuzzy J-K flip-flop based on various norms combined with the standard negation was analyzed in order to investigate, whether and to what degree they present more or less sigmoid (s-shaped) $J \to Q_{out}$ characteristics in particular cases, when *K* = 1 - *Q* (unified fuzzy J-K flip-flop with feedback), *K* = 1 - *J* (new fuzzy D flip-flop derived from the unified fuzzy J-K one) with fixed value of *Q*. We conducted extensive investigations and found that the $J \to Q_{out}$ transfer characteristics of fuzzy J-K flip-flops with feedback based on Łukasiewicz, Yager, Dombi and Hamacher norms, as well as the $D \to Q_{out}$ characteristics of (new) fuzzy D flip-flops of Łukasiewicz, Yager and Hamacher operations, show quasi sigmoid curvature for selected *Q* values. The fuzzy J-K and D F^3s based on algebraic norms as well as the fuzzy D F^3s based on Dombi norms have non-sigmoid behavior.

3 Fuzzy Neural Networks

3.1 Fuzzy Flip-Flop Neurons

We proposed the construction of a neuron unit, a combinational sigmoid generator derived from arbitrary fuzzy J-K flip-flop where $\overline{Q}$ is fed back to K and (old) Q is fixed (Figure 1). In this approach, the output of fuzzy J-K flip-flop neuron depends from the value of Q_{fix} and input values of J. Substitute $\overline{K} = Q$ (1 - $K = Q$) in the unified formula of the fuzzy J-K flip-flop (equation 1), for a fix Q value, the characteristic equation of fuzzy J-K flip-flop neuron is

$$Q_{out} = (J \; u \; Q_{\text{fix}}) \; i \; (J \; u \; Q_{\text{fix}}) \; i \; \left(Q_{\text{fix}} \; u \; \left(1 - Q_{\text{fix}}\right)\right) \tag{4}$$

where i and u denotes various t-norms and t-conorms.

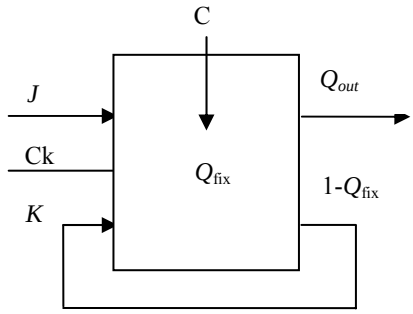

Figure 1
Fuzzy J-K flip-flop neuron

The clocked fuzzy J-K flip-flop neuron circuit can be implemented using hardware blocks (denoted by i, u and n symbols) to realize various t-norms, t-conorms and fuzzy negations [24]. Since t-norms and t-conorms are functions from the unit square into the unit interval, the fuzzy J-K flip-flop neuron block diagram differs from the binary J-K flip-flop structure. The input J is driven by a synchronized clock pulse in the sample-and-hold (S/H) circuit (Figure 2) which could be a simple D flip-flop used as register.

We proposed the construction of the fuzzy D flip-flop neuron (Figure 3), which is a combinational sigmoid generator. This unit is derived from arbitrary fuzzy J-K flip-flop by connecting the inputs of the fuzzy J-K flip-flop in a particular way, namely by applying an inverter in the connection of the input J to K.

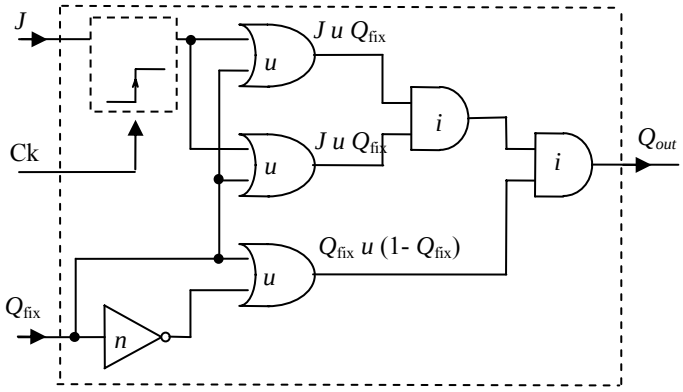

Figure 2
Fuzzy J-K flip-flop neuron block diagram

Starting from the fundamental equation of the fuzzy J-K flip-flop and substituting $\overline{K} = J$ in equation (1) and letting $D = J$ for a fix Q value, the characteristic equation of fuzzy D flip-flop neuron is

$$Q_{out} = (D\ u\ D)\ i\ (D\ u\ Q_{fix})\ i\ \left(D\ u\ \left(1 - Q_{fix}\right)\right) \tag{5}$$

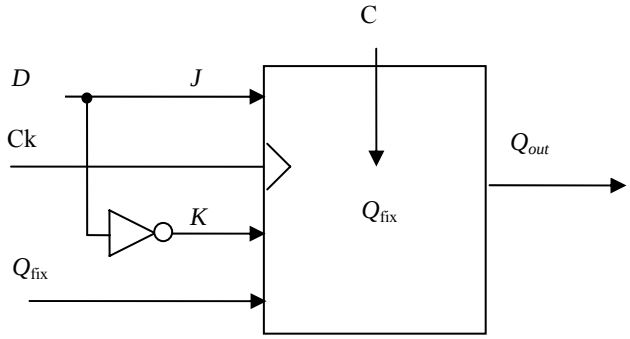

Figure 3
Fuzzy D flip-flop neuron

Interconnecting the blocks of fuzzy operations in a different way, the fuzzy D flip-flop neuron block diagram is obtained (Figure 4).

The use of four-layered (that have two sigmoid hidden layers) neural network as universal approximators of continuous functions have been investigated by Funahashi [4] and Hecht-Nielsen [6]. Kurkova [10] studied the function approximation capabilities of multilayer feedforward networks with sigmoid activation, analyzing also their computational complexity issues.

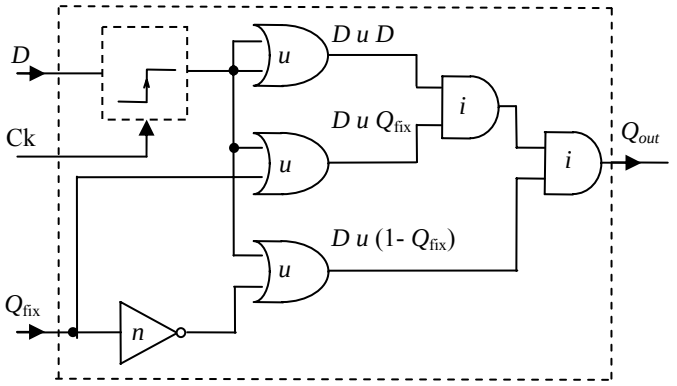

Figure 4
Fuzzy D flip-flop neuron block diagram

The neuro-fuzzy system proposed is based on two hidden layers where the approximation process becomes more manageable. The first hidden layer neurons are used to partition the input space into regions and learn the local features. A neuron in the second layer learns the global features for a particular region of the input space and outputs zero elsewhere [4]. The FNNs constituted from fuzzy flip-flop neurons are supervised feedforward network, applied in order to approximate various test functions. The functions to be approximated are represented by a set of input/output pairs.

In this approach the weighted input values are connected to input *J* of the fuzzy flip-flop neuron based on a pair of t-norm and t-conorm, having quasi sigmoid transfer characteristics. The output signal is then computed as the weighted sum of the input signals transformed by the transfer function. Based on previous hardware implementation results [19], FNNs with fixed structure can be stated as easily implemented in real hardware neural networks.

The nonlinear characteristics exhibited by fuzzy neurons are represented by quasi sigmoid transfer functions given by fuzzy J-K and D flip-flop neurons based on algebraic, Łukasiewicz, Yager, Dombi and Hamacher operations. The proposed network activation function is the same at each hidden layer, from unit to unit. The function approximation goodness is strongly dependent on the number of fuzzy neurons in the hidden layers [15], [16].

4 Bacterial Memetic Algorithm with Modified Operator Execution Order

The process of biological evolution has inspired a large amount of optimization algorithms. It has been shown that evolutionary algorithms work efficiently for solving nonlinear and constrained optimization problems. These methods do not use derivatives of the functions, such as the gradient-based training algorithms. Similarly to biological recombination, these methods are based on the search for a large number of solutions.

In particular, the Bacterial Memetic Algorithm with Modified Operator Execution Order (BMAM) [5] evolutionary approach is proposed for FNNs training. This chapter presents how the bacterial evolutionary algorithm [18] can be improved with another optimization method, the Levenberg-Marquardt (LM) technique, to achieve better results in the function approximation process, developing an effective hybrid combination.

The BMAM is a memetic algorithm which eliminates the imperfection of traditional evolutionary and LM algorithm. The evolutionary algorithm is able to find the global optimum region but miss the local optimum solution. The LM algorithm is fast and efficient for training feedforward neural networks and is able to find the local optimum, but is very sensitive to the initial position of the search space.

The learning of the FNNs is formulated as a parameter optimization problem, using the mean square error as the fitness evaluation set-up. The algorithm starts with a random population of initial solutions to the optimization problem. The solutions are coded as an array of floating point or integer values. The basic steps embrace the bacterial mutation operation and the LM method. In this application the population number was initialized according to the network size.

During simulations 30 generations of 5 individuals with 5 clones were chosen to obtain the best fitting variable values, with the lowest performance. Then the same part, or parts, of the chromosome is chosen and mutated randomly. The LM method nested into evolutionary algorithm is applied 3 times for each individual. The selection of the best clone is made and transfers its mutated parts to the other clones. The part choosing-mutation-LM method-selection-transfer cycle is repeated until all the parts are mutated, improved and tested. The best individual remains in the population and all other clones are deleted. This process is repeated until all individuals have gone through the modified bacterial mutation. Then the Levenberg-Marquardt method is applied 7 times for each individual executing several LM cycles during the bacterial mutation after each mutation step.

Gene transfer operation is done 3 times for a partial population. The number of the gene transfers in a generation is the algorithm parameter; it could be 0. The quasi optimal values can be identified at the end of the BMAM training algorithm.

5 Numerical Simulation and Results

The FNNs architecture is predefined, depending on the input function complexity. In this approach the choice of an optimal network design for a given problem is a guessing process. In particular, the application of a recently improved BMAM algorithm is applied for training various FNNs with different structures. This new, complex software is able to train all the FNNs parameters, eliminating completely the imprecision caused by training them with the LM algorithm. The simulation results published in our previous papers obtained under the same conditions could turn out to be different because the LM method is very sensitive to the initial values of the search space. The FNNs approximate a one-dimensional real-life application, two dimensional trigonometric functions, two benchmark problems whose dates were selected from the input/output test points of a gas furnace benchmark data set, and a six dimensional non-polynomial function. The test functions are arranged in the order of complexity.

A Simple Real - Life Application: Approximation of a Nickel-Metal Hydride Battery Cell Charging Characteristics

In this particular case, the FNNs approximate a Nickel-Metal Hydride (NiMH) Battery Cell charging characteristics [2], a one-input real-life application.

The nickel-metal hydride batteries can be repeatedly charged and discharged for more than 500 cycles. The charging process duration can be different, from 15 minutes to 20 hours. The charge characteristics are affected by current, time and temperature. In this experiment it was more than 1 hour. The test function is a characteristic between the battery capacity input and the cell voltage. The battery type was GP 3.6V, 300 mAH, 3x1.2V NiMH, charged for 1.5 hours with 300 mA and 25˚C.

B The Box-Jenkins' Gas Furnace Benchmark Data Set

The gas furnace data set presented by Box and Jenkins in 1970 is a frequently used benchmark data set. The set consists of 296 input-output data; a pair is sampled at every 9 seconds. The input signal represents the flow rate of the methane in a gas furnace, while the output of the model corresponds to the CO_2 concentration in the gas mixture flowing out of the furnace under a steady air supply [17]. We used as the most studies [21], [23] the inputs $y(k\text{-}1)$ and $u(k\text{-}4)$ which have the highest correlation with the output $y(k)$.

C Two - Input Trigonometric Functions

We used the next two two-dimensional polynomial input functions as test functions

$$y_1 = \left(\sin\left(c_1 \cdot x_1\right)^5 \cdot \cos\left(c_2 \cdot x_2\right)^3\right)/2 + 0.5 \tag{6}$$

$$y_2 = e^{-\frac{r^2}{100}} \cdot \cos\left(\frac{r}{2}\right); \text{ where } r = \sqrt{x_1^2 + x_2^2}, \tag{7}$$

$x_1,\ x_2 \in [-20, 20]$

D Six Dimensional Non-Polynomial Function

This widely used target function originates from paper [1] and is given by the following expression

$$y_3 = x_1 + x_2^{0.5} + x_3 x_4 + 2e^{2(x_5 - x_6)} \tag{8}$$

where $x_1,\ x_2 \in [1,\ 5],\ x_3 \in [0,\ 4],\ x_4 \in [0,\ 0.6],\ \ x_5 \in [0,\ 1],\ x_6 \in [0,\ 1.2]$.

The real-life application (denoted by 1D) is approximated with a 1-2-2-1 FNN size, described by a set of 543 input/output samples selected equidistantly from a set of 2715 test points. To approximate the gas furnace benchmark data set we proposed a 1-2-2-1 FNN (2D-gas) size given by 296 dates. A 1-20-20-1, respectively a 1-15-15-1 feedforward neural networks structure based on F^3 neurons were proposed to approximate the two two-input trigonometric functions, (equations (6) and (7), labeled as 2D-trig and 2D-hat), represented by 1600 input/output samples. To approximate the six-dimensional benchmark problem we proposed a 1-10-10-1 FNN (6D) size given by 200 samples.

The number of neurons was chosen after experimenting with different size hidden layers. Smaller neuron numbers in the hidden layer result in worse approximation properties, while increasing the neuron number in a complex FNN structure results in better performance, but longer simulation time.

In [15] we proposed a method to find the optimal *Q* and fuzzy operation parameter pairs for J-K and D type F^3 neurons based on algebraic, Yager, Dombi and Hamacher norms by training a 1-8-8-1 FNN with the Bacterial Memetic Algorithm. The optimal variable values depend on the fuzzy flip-flop neuron and fuzzy operation types. A change of the operations and parameter values in the characteristic equations of the fuzzy J-K and D flip-flops leads to the modification of the slope of the transfer function, which will affect the learning rate in the implementation of neural networks. In the next, the algebraic, Łukasiewicz, Yager, Dombi and Hamacher operations and two different fuzzy flip-flop neuron types will be compared from the point of view of the respective fuzzy-neural networks approximation capability.

Figures 5 and 6 compare the function approximation performance of J-K and D FNNs in case of various test functions. Tables I and II present the 10 runs average approximation goodness, by indicating the median MSE (mean squared error) of the various FNNs training values. During evaluation we compared the median MSE values, considering them as the most important indicators of trainability. The median is a robust estimate of the center of a data sample, since outliers have little

effect on it. According to the numerical illustrations the average of 10 runs MSE values the sequence is almost the same in every test function cases. By extensive simulation experiments it is proved that the function approximation goodness of FNNs based on fuzzy J-K flip-flop neuron with Dombi and Łukasiewicz norms are the best ones.

The error of approximation of the battery cell and Box-Jenkins' gas furnace benchmark dates characteristics obtained by traditional *tansig* function is irrelevantly less than that obtained in our simulations. As can be seen from Figure 6, and comparing the simulation results from Table II the function approximation by D FNNs may be considered sufficiently good in case of Łukasiewicz and Yager type fuzzy operations.

Conclusions

In this paper, we found that the fuzzy J-K flip-flop neurons based on Dombi and Łukasiewicz as well as the fuzzy D flip-flop neurons based on Łukasiewicz and Yager norms are the most suitable ones for constructing FNNs in order to approximate various test functions. As these FNN types produced more or less low MSE values in all simulation experiments. Thus, we proposed the construction of real hardware fuzzy neural networks constructed of the above-mentioned F^3 neuron types. The accuracy of the approximations not only depends on the network structures and parameters selected, such as the number of layers, and of the hidden units, but is strongly influenced by the fuzzy flip-flop neuron and fuzzy operation type. In the future we intend to propose new types of fuzzy flip-flop based on pliant inequality [2], and to improve the function approximation capability of FNNs.

Acknowledgement

Research supported by the National Scientific Research Fund Grant OTKA K75711, further by Széchenyi István University Main Research Direction Grant 2010, and Óbuda University Grants.

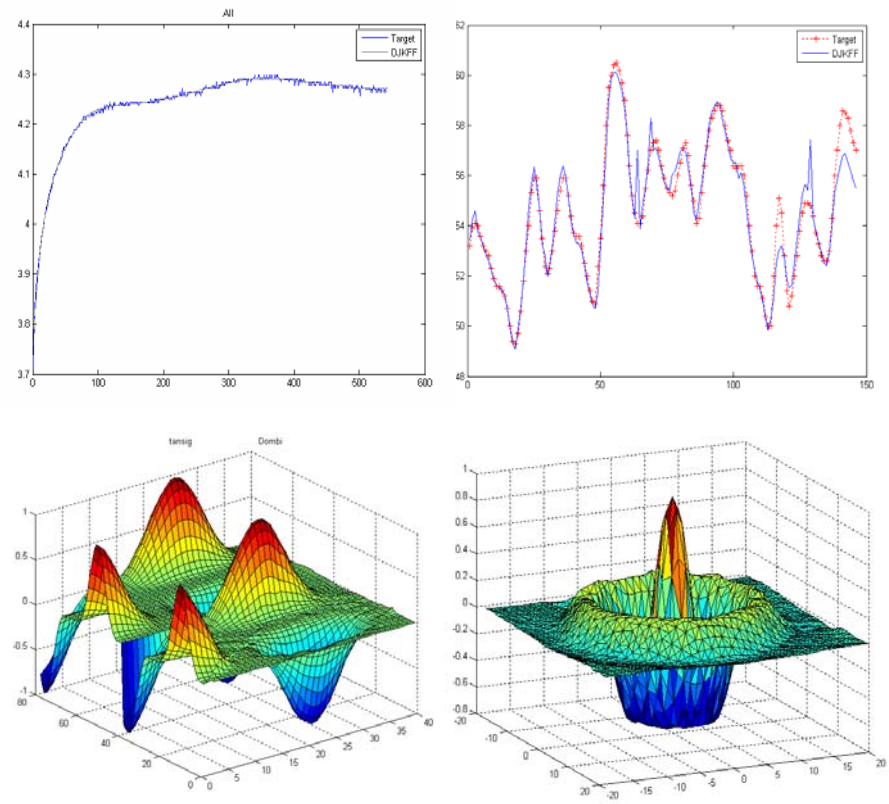

Figure 5
Function approximation capabilities of J-K FNNs

Table I
Training Median MSE Values for J-K Type FNNs

Fuzzy op.	Bat. cell 1-2-2-1 FNN	2D-gas 1-2-2-1 FNN	2D-trig 1-20-20-1 FNN	2D-hat 1-15-15-1 FNN	6D 1-10-10-1 FNN
tansig	$1.32\text{x}10^{-5}$	$7.71\text{x}10^{-2}$	$9.07\text{x}10^{-7}$	$4.26\text{x}10^{-7}$	$1.12\text{x}10^{-4}$
Algebraic	$3.32\text{x}10^{-4}$	$1.07\text{x}10^{0}$	$4.32\text{x}10^{-2}$	$3.17\text{x}10^{-2}$	$9.69\text{x}10^{-1}$
Łukasiewicz	$\mathbf{7.11\text{x}10^{-5}}$	$\mathbf{7.27\text{x}10^{-2}}$	$3.71\text{x}10^{-4}$	$9.46\text{x}10^{-4}$	$5.78\text{x}10^{-1}$
Yager	$1.47\text{x}10^{-4}$	$8.23\text{x}10^{-1}$	$1.53\text{x}10^{-2}$	$1.49\text{x}10^{-2}$	$5.92\text{x}10^{-1}$
Dombi	$\mathbf{3.52\text{x}10^{-5}}$	$\mathbf{7.30\text{x}10^{-2}}$	$\mathbf{8.75\text{x}10^{-6}}$	$\mathbf{1.76\text{x}10^{-4}}$	$2.98\text{x}10^{-1}$
Hamacher	$2.59\text{x}10^{-4}$	$8.27\text{x}10^{-1}$	$2.18\text{x}10^{-2}$	$2.86\text{x}10^{-2}$	$7.43\text{x}10^{-1}$

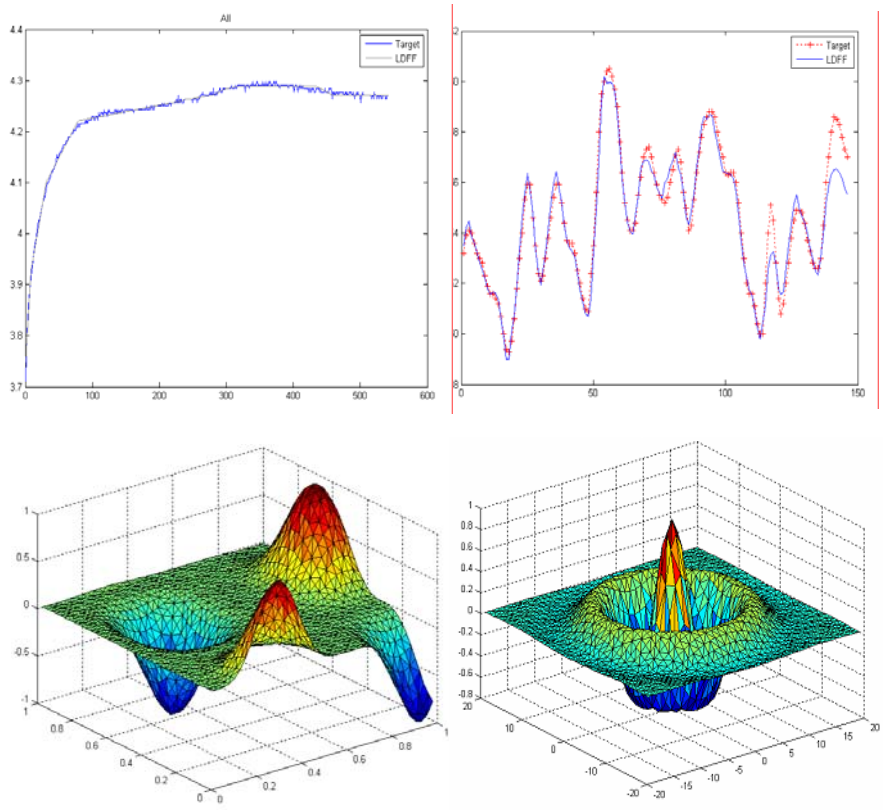

Figure 6
Function approximation capabilities of D FNNs

Table II
Training Median MSE Values for D Type FNNs

Fuzzy op.	Bat. cell 1-2-2-1 FNN	2D-gas 1-2-2-1 FNN	2D-trig 1-20-20-1 FNN	2D-hat 1-15-15-1 FNN	6D 1-10-10-1 FNN
tansig	$1.32\text{x}10^{-5}$	$7.71\text{x}10^{-2}$	$9.07\text{x}10^{-7}$	$4.26\text{x}10^{-7}$	$1.12\text{x}10^{-4}$
Algebraic	$1.38\text{x}10^{-4}$	$1.18\text{x}10^{0}$	$2.71\text{x}10^{-2}$	$2.94\text{x}10^{-2}$	$6.56\text{x}10^{-1}$
Łukasiewicz	**$4.95\text{x}10^{-5}$**	**$7.26\text{x}10^{-2}$**	**$7.48\text{x}10^{-4}$**	**$1.52\text{x}10^{-3}$**	$7.15\text{x}10^{-1}$
Yager	$1.09\text{x}10^{-4}$	**$7.69\text{x}10^{-2}$**	$8.21\text{x}10^{-3}$	$1.86\text{x}10^{-2}$	**$1.45\text{x}10^{-1}$**
Dombi	$6.41\text{x}10^{-4}$	$1.25\text{x}10^{0}$	$2.93\text{x}10^{-2}$	$3.23\text{x}10^{-2}$	$1.58\text{x}10^{0}$
Hamacher	$1.12\text{x}10^{-4}$	$7.91\text{x}10^{-2}$	$5.25\text{x}10^{-3}$	$1.48\text{x}10^{-2}$	$2.52\text{x}10^{-1}$

References

[1] J. Botzheim, B. Hámori, L. T. Kóczy, A. E. Ruano: Bacterial Algorithm Applied for Fuzzy Rule Extraction, IPMU 2002, Annecy, France, pp. 1021-1026

[2] J. Dombi: Pliant Arithmetics and Pliant Arithmetic Operations, Acta Polytechnica Hungarica, Vol. 6, No. 5, 2009, pp. 19-49

[3] R. Fan: NiMH Battery Charger Reference Design, Designer Reference Manual, 2003

[4] K. I. Funahashi: On the Approximate Realization of Continuous Mapping by Neural Networks, Neural Networks, 2, 1989, pp. 183-192

[5] L. Gál, J. Botzheim, L. T. Kóczy: Improvements to the Bacterial Memetic Algorithm used for Fuzzy Rule Base Extraction, Proc. of CIMSA 2008, Computational Intelligence for Measurement Systems and Applications, Istanbul, Turkey, 2008, pp. 38-43

[6] R. Hecht-Nielsen: Theory of the Backpropagation Neural Network, in Proc. of the Neural Networks, IJCNN, International Joint Conference on Neural Networks, 1989, pp. 593-605

[7] K. Hirota, K. Ozawa: Concept of Fuzzy Flip-Flop, Preprints of 2nd IFSA Congress, Tokyo, 1987, pp. 556-559

[8] K. Hirota, K. Ozawa: Fuzzy Flip-Flop as a Basis of Fuzzy Memory Modules, In: M. M. Gupta et al., eds., Fuzzy Computing. Theory, Hardware and Applications, North Holland, Amsterdam, 1988, pp. 173-183

[9] J.-S. R. Jang, C. T. Sun, E. Mizutani: Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence, Upper Saddle River, NJ: Prentice Hall, 1997

[10] V. Kurkova: Kolmogorov's Theorem and Multilayer Neural Networks Neural Networks, 5. 1992, pp. 501-506

[11] R. S. T. Lee: Fuzzy-Neuro Approach to Agent Applications, Springer, 2006

[12] R. Lovassy, L. T. Kóczy, L. Gál: Analyzing Fuzzy Flip-Flops Based on Various Fuzzy Operations, Acta Technica Jaurinensis Series Intelligentia Computatorica, Vol. 1, No. 3, 2008, pp. 447-465

[13] R. Lovassy, L. T. Kóczy, L. Gál: Applicability of Fuzzy Flip-Flops in the Implementation of Neural Networks; Proc. of CINTI 2008, 9th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, 2008, pp. 333-344

[14] R. Lovassy, L. T. Kóczy, L. Gál: Function Approximation Capability of a Novel Fuzzy Flip-Flop-based Neural Network, Proc. of IJCNN 2009, International Joint Conference on Neural Networks, Atlanta, USA, 2009, pp. 1900-1907

[15] R. Lovassy, L. T. Kóczy, L. Gál: Parameter Optimization in Fuzzy Flip-Flop Based Neural Networks, in Proc. of International Symposium on Innovations in Intelligent Systems and Applications, INISTA 2009, Trabzon, Turkey, 2009, pp. 205-209

[16] R. Lovassy, L. T. Kóczy, L. Gál: Fuzzy Flip-Flops as Basic Components in Fuzzy Neural Networks-abstract, Tenth International Conference on Fuzzy Set Theory and Applications, FSTA 2010, Liptovsky Ján, the Slovak Republic, 2010, pp. 92-93

[17] S. Lukasik, P. Kowalski, M. Charytanowicz, P. Kulczycki: Fuzzy Model Identification Using Kernel-Density-based Clustering, in: Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Applications. Volume II, Krassimir Atanassov et al. (eds.), EXIT Publishing House, Warsaw (Poland), 2008, pp. 135-146

[18] N. E. Nawa, T. Furuhashi: Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm, IEEE Trans. on Fuzzy Systems, 7(5), 1999, pp. 608-616

[19] K. Ozawa, K. Hirota, L. T. Kóczy, K. Omori: Algebraic Fuzzy Flip-Flop Circuits, Fuzzy Sets and Systems 39/2, North Holland, 1991, pp.215-226

[20] K. Ozawa, K. Hirota, L. T. Kóczy: Fuzzy Flip-Flop, In: M. J. Patyra, D. M. Mlynek, eds., Fuzzy Logic. Implementation and Applications, Wiley, Chichester, 1996, pp. 197-236

[21] W. Pedrycz: An Identification Algorithm in Fuzzy Relation Systems, Fuzzy Sets Syst. 13, 1984, pp. 153-167

[22] L. Rutkowski: Flexible Neuro-Fuzzy Systems, Kluwer 2004

[23] C. W. Xu, Y. Z. Lu: Fuzzy Model Identification and Self-Learning for Dynamic Systems, IEEE Trans. Syst. Man and Cyber., 17, 1987, pp. 683-689

[24] A. H. Zavala, O. C. Nieto, I. Batyrshin, L. V. Vargas: VLSI Implementation of a Module for Realization of Basic t-norms on Fuzzy Hardware, Proc. of FUZZ-IEEE 2009, IEEE Conference on Fuzzy Systems, Jeju Island, Korea, August 20-24, 2009, pp. 655-659