# Suitability of Fuzzy Systems and Neural Networks for Industrial Applications

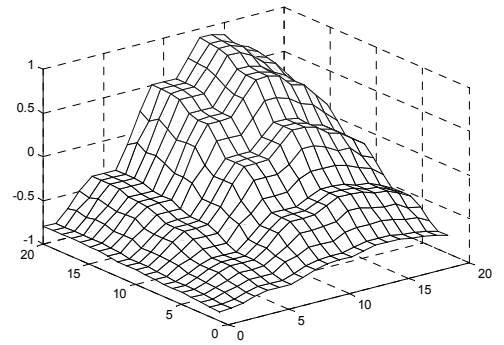
Bogdan M. Wilamowski
Electrical and Computer Engineering, Auburn University,
420 Broun Hall, Auburn, Alabama, USA
*wilam@ieee.org*

***Abstract*-The presentation provides a comparison of fuzzy and neural systems for industrial applications. Both neural networks and fuzzy systems perform nonlinear mapping and both systems internally operate within a limited signal range between zero and one. Neural networks can handle basically an unlimited number of inputs and outputs while fuzzy systems have one output and number of inputs is practically limited to 2 or 3. The resulted nonlinear function produced by neural networks is smooth while functions produced by fuzzy systems are relatively rough. At the same time the design of fuzzy systems transparent and easy to follow while the development of neural networks is much more labor intensive. It is shows that most commonly used neural network architecture of MLP – Multi Layer Perceptron is also one of the least efficient ones. Also most commonly used EBP – Error Back Propagation algorithm is not only very slow, but also it is not able to find solutions for optimal neural network architectures. EBP can solve problems only when large number of neurons is used, but this way neural network loses its generalization property. Performances of both fuzzy systems and neural networks are compared leading to the conclusion that neural networks can produce much more accurate nonlinear mapping and they may require less hardware**

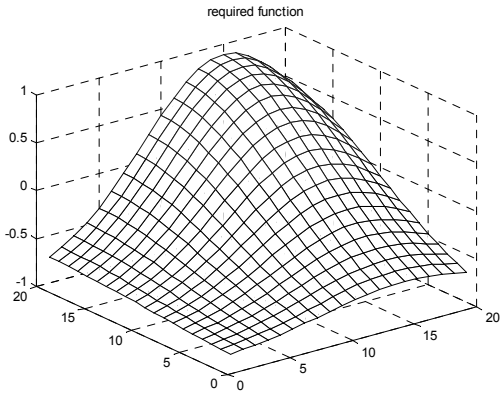
## I. INTRODUCTION

Neural and fuzzy systems found already a prominent place in industrial control [1-3]. Both neural networks and fuzzy systems perform nonlinear mapping and both systems internally operate within a limited signal range between zero and one. In general, all parameters of fuzzy systems are designed, while parameters of neural networks are being obtained by a training process. It is relatively easy for humans to follow the computation process of fuzzy systems, while it is almost impossible to do as in the case of neural networks. Neural networks can handle basically an unlimited number of inputs and outputs while fuzzy systems have one output and number of inputs is practically limited to 2 or 3. The resulted nonlinear function produced by neural networks is smooth while functions produced by fuzzy systems are relatively rough (see Fig. 1).

It is relative easy to design fuzzy systems based on a designer's intuition. In case of neural networks a designer may face many challenges. The first challenge is that it is difficult to decide how many neurons to use and how they have to be connected. Second problem, which often leads to frustration, is how to train neural networks. As a result far from optimum neural network architectures are selected and learning algorithms which are not able to produce a good solution are used. These issues will be discussed in section 2 of this presentation. However, if difficulties with neural networks are solved then neural networks generate not only a better and smoother control surface, but also its microcontroller implementations require shorter code and faster operation [4]. Of course it is possible to merge these two technologies by developing neuro-fuzzy systems.




(a)




(b)

Fig. 1. Comparison of control surfaces obtained with (a) fuzzy controller with 6 and 5 membership function for both inputs (b) neural controller with 3 neurons

## II. NEURAL NETWORKS

Artificial neural networks consist of many neurons with a sigmoid activation function as shown in Fig. 2. These neurons are connected in such a way that the feed forward signal flow is assured. Connecting weights may have both positive and

negative values and the resulted neuron excitation *net* is calculated as a sum of products of incoming signals multiplied by weights:

$$net = \sum_{i=1}^{n} w_i x_i \qquad (1)$$

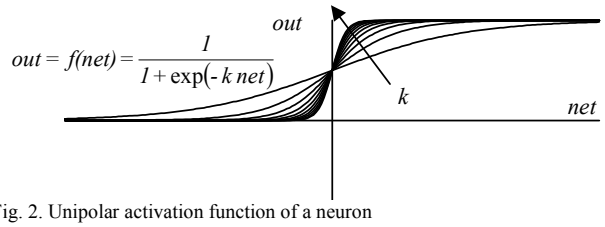



Fig. 2. Unipolar activation function of a neuron

*A. Neural Network Architectures*

The simplest NN architecture is the conterpropagation network [5]. The most commonly used architecture is MLP – Multi Layer Perceptron as shown in Fig. 3 [6-8] The only advantage of MLP is that it is relatively easy to write software for this architecture. Unfortunately when using MLP topology more neurons are needed to solve problems. A better option is to use MLP with connections across layers. This architecture is not only more powerful but it can be trained faster, assuming that we have written a proper software. The most powerful architecture is the cascade architecture, also known as FCN – Fully Connected Network shown in Fig. 4.

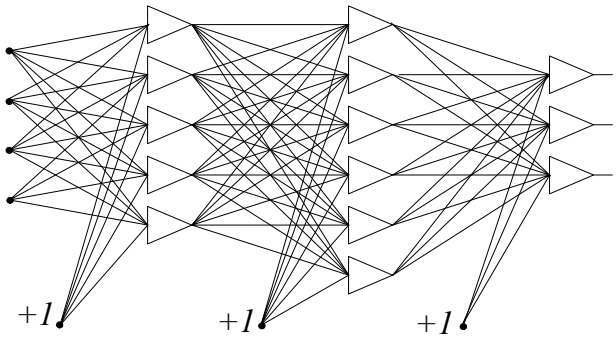



Fig. 3. MLP - Multi Layer Percepton architecture for neural networks

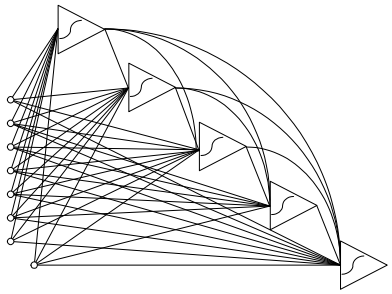
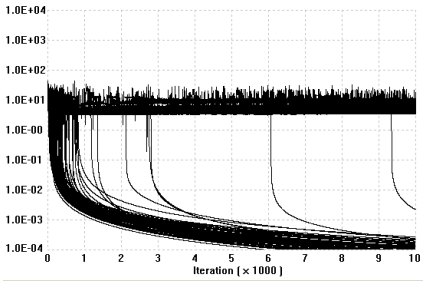


Fig. 4. FCN – Fully Connected Network or cascade network architecture

Recently significant attention is devoted to the deep neural network architectures [9,10]. As it was shown in [11,12] the power of neural networks increases almost exponentially with teir depth. Unfortunately such deep architectures are difficult to train.

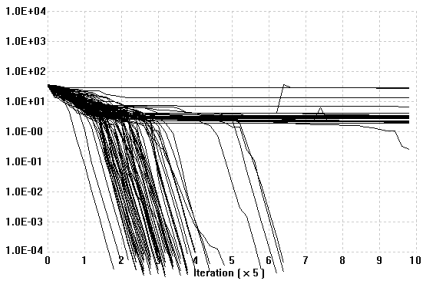
*B. Neural Network Training*

The most common training algorithm is EBP – Error Back Propagation [13-16]. It is relatively simple and it does not require a lot of computer resources. This algorithm however seldom leads to a good solution and is extremely slow. The EBP show some advantages for MLP networks [5,17] Much better results can be obtained with the LM – Levenberg-Marquardt Algorithm [18]. Even better results can be obtained with NBN – Neuron by Neuron algorithm [19,20]. Note that in the LM algorithm an N by N matrix must be inverted in every iteration. This is the reason why for large size neural networks the LM algorithm is not practical. Also most of implementations of LM algorithms (like popular MATLAB NN Toolbox) are developed only for MLP [21]. The Neuron by Neuron (NBN) algorithm was developed in order to eliminate most disadvantages of the LM algorithm. Detailed descriptions of the algorithm can be found in [17,22-24].



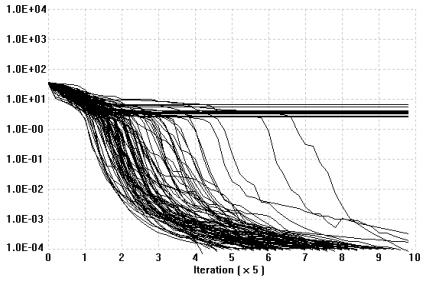

| | |
|---|---|
| Number of iterations | 7956 |
| Training time | 3297ms |
| Success rate | 46% |

Fig. 5. Result of parity-4 training using EBP algorithm with 4-3-3-1 architecture




| | |
|---|---|
| Number of iterations | 17 |
| Training time | 15ms |
| Success rate | 69% |

Fig. 6. Result of parity-4 training using LM algorithm with 4-3-3-1 architecture

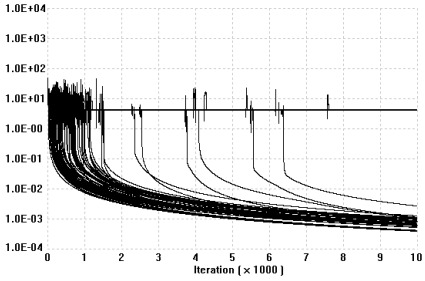



| | |
|---|---|
| Number of iterations | 34 |
| Training time | 37ms |
| Success rate | 82% |

Fig. 7. Result of parity-4 training using NBN algorithm with 4-3-3-1 architecture

Figures 5 through 7 show the results of training of MLP architecture for the Parity- 4 problem. One may notice that EBP training requires 200 times more iterations and 100 times longer time to train. Also the success rate is smaller than in the case of other algorithms. LM algorithm is slightly faster but it has a smaller success rate in comparison to NBN algorithm. For the FCN architecture, with 3 neurons connected in cascade, the parity problem can be solved with significantly smaller number of neurons (3 instead of 7).

Unfortunately EBP algorithm cannot solve this problem with less than 10,000 iterations. The standard LM algorithm is not suitable for FCN architecture; but NBN algorithm can solve this problem in a short time with the success rate of 98%. Training results for Parity-4 problem are summarized in Table I.

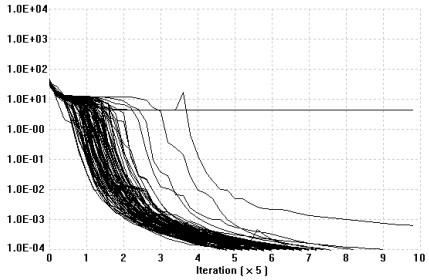



| | |
|---|---|
| Success rate | 0% |

Fig. 8. Result of parity-4 training using EBP algorithm with 4-1-1-1 architecture

For MLP topologies it seems that the EBP algorithm is most robust and has the largest success rate for random weight initialization. At the same time the EBP algorithm requires over 200 times larger number of iterations to converge.

If the number of neurons in FCN is increased to 4 then also EBP algorithm can solve the problem with success rate of 98%. The NBN algorithm can solve this problem in 150 times shorter time with 100% success rate. Results are shown in Table II. One may also notice that with increasing of network complexity neural networks are losing their ability for generalization (if neural network is used for nonlinear mapping).




| | |
|---|---|
| Number of iterations | 30 |
| Training time | 19ms |
| Success rate | 98% |

Fig. 9. Result of parity-4 training using NBN algorithm with 4-1-1-1 architecture

TABLE I
COMPARISON OF SOLUTIONS OF PARITY-4 PROBLEM WITH VARIOUS ALGORITHMS FOR TWO DIFFERENT ARCHITECTURES

| | Iterations | Time [ms] | Success rate |
|---|---|---|---|
| **MLP with 4331 architecture without connections across layers (desired error=0.0001)** | | | |
| EBP ( α=3) | 7956.4 | 3296.6 | 46% |
| LM (Gauss-Jordan) | 16.5 | 14.9 | 69% |
| NBN (Gauss-Jordan) | 33.6 | 37.07 | 82% |
| **FCN with 4111 architecture with connections across layers (desired error=0.0001)** | | | |
| EBP ( α=3) | N/A | N/A | 0% |
| LM (Gauss-Jordan) | *Is not able to handle FCN with connections across layers* | | |
| NBN (Gauss-Jordan) | 30.2 | 19.1 | 98% |

TABLE II
COMPARISON OF SOLUTIONS OF PARITY-4 PROBLEM WITH VARIOUS ALGORITHMS ON 4-1-1-1-1 TOPOLOGY

| Type size (pts.) | Averages from 100 runs | | |
|---|---|---|---|
| | Success rate | iterations | Computing time |
| EBP | 98% | 3977.15 | 1382.78ms |
| LM | N/A | N/A | N/A |
| NBN | 100% | 12.36 | 8.15ms |

For experiments shown in Table III the LM algorithm was modified so not only MLP networks but all arbitrarily connected neural networks could be trained. Also in both LM and NBN algorithms the matrix inversion, the Gauss-Jordan method, was replaced by the LU decomposition. As a consequence the training time was significantly reduced.

One may notice that if too large neural networks are used the system can find solutions which produce very small error for the training patterns, but for patterns which were not used for training errors actually could be much larger than in the case of much simpler network.

TABLE III
COMPARISON OF SOLUTIONS OF VARIOUS INCREASED COMPLEXITY PROBLEMS USING VARIOUS ALGORITHMS AND FCN ARCHITECTURES

| | Iterations | Time [ms] | Success rate |
|---|---|---|---|
| **Parity 4 problem with 4111 architecture (desired error=0.001)** | | | |
| EBP ( α=1) | 17505 | 3384.6 | 93% |
| LM (modified) | 14.6 | 0.16 | 98% |
| NBN (modified) | 20.6 | 1.01 | 99% |
| **Parity 8 problem with 411111 architecture (desired error=0.001)** | | | |
| EBP ( α=1) | failed | failed | 0% |
| LM (modified) | 32.1 | 152.2 | 8% |
| NBN (modified) | 40.6 | 192.7 | 28% |
| **Parity 12 problem with 411111111 architecture (desired error=0.001)** | | | |
| EBP ( α=1) | failed | failed | 0% |
| LM (modified) | 77.3 | 9656. | 3% |
| NBN (modified) | 66.7 | 14,068. | 14% |

What many people are not also aware of is that not all popular algorithms can train every neural network. Surprisingly, the most popular EBP (Error Back Propagation) algorithm cannot handle more complex problems while other more advanced algorithms can. Also, in most cases neural networks trained with popular algorithms such as EBP produce far from optimum solutions [5,17].

***Boolean***

| | | AND | | | OR |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

***Fuzzy***

| | | MIN | | | MAX |
|---|---|---|---|---|---|
| 0.3 | 0.1 | 0.1 | 0.3 | 0.1 | 0.3 |
| 0.2 | 0.8 | 0.2 | 0.2 | 0.8 | 0.8 |
| 0.2 | 0.9 | 0.2 | 0.2 | 0.9 | 0.9 |
| 0.8 | 0.9 | 0.8 | 0.8 | 0.0 | 0.9 |

Fig. 10. Comparison Boolean algebra with fuzzy logic .

## III. FUZZY SYSTEMS

The fuzzy set system theory was developed by Zadeh [25]. Fuzzy logic is similar to Boolean algebra, but it operates on analog values between zero and one. Also, instead of AND and OR operators the MIN and MAX operators are used as is shown in Fig. 10.

In order to solve problem of nonlinear mapping two similar approaches are usually taken: Mamdani[26] and TSK[27,28]. Block diagrams for these two controllers are shown in Figures 11 and 12.

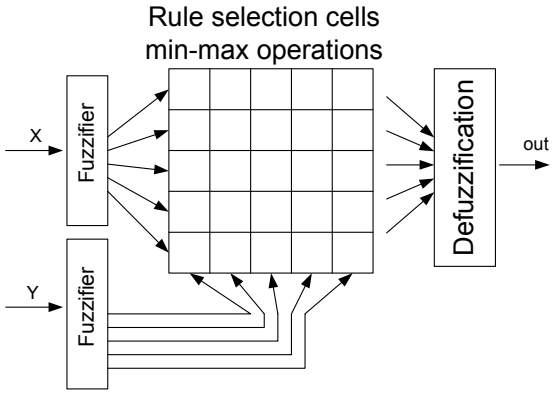



Fig. 11. Block diagram of a Mamdani type fuzzy controller

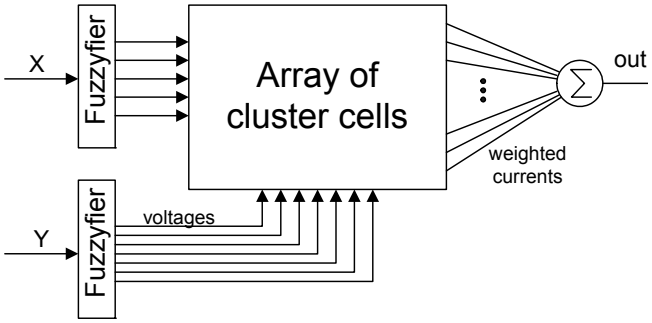



Fig. 12. TSK (Takagi-Sugeno-Kang) fuzzy architecture.

## IV. NEURO-FUZZY SYSTEMS

Most commonly used neuro-fuzzy architecture is shown in Fig 13 [29]. It has neural network topology, but its operation does not reassemble biological neuron operations. This concept of neuro-fuzzy architecture requires signal multiplication and division and as result it is not easy to implement this concept in hardware.

It is, however, possible to implement fuzzy systems using typical neurons with sigmoid activation functions. One such implementation may follow the the concept of conterpropagation neural networks [30]. Actually TSK fuzzy systems have a very similar topology. Unfortunately, the conterpropagation networks operate correctly only on normalized inputs. Normalization of inputs lead to removal of important information so it cannot be used. However, by

increasing the dimensionality of input dimensionality it is possible to project input data on sphere (or hypersphere) without losing important information, The neuro-fuzzy system based on the conterpropagation network with input pattern transformation is shown in Fig. 14. `In the network of Fig. 14 each neuron is responsible for one fuzzy rule.

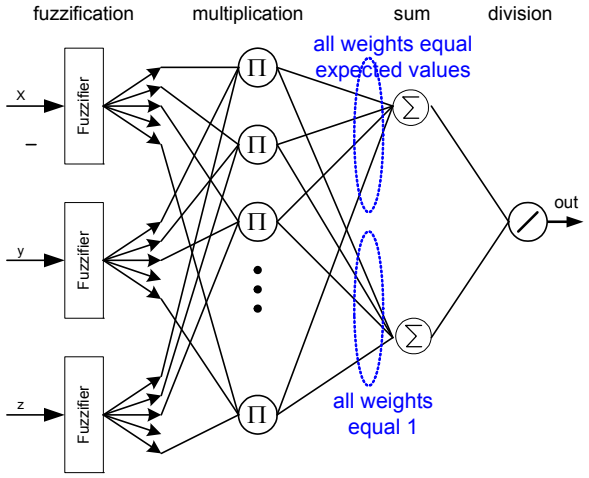



Fig. 13. Classical Neuro-Fuzzy Architecture.

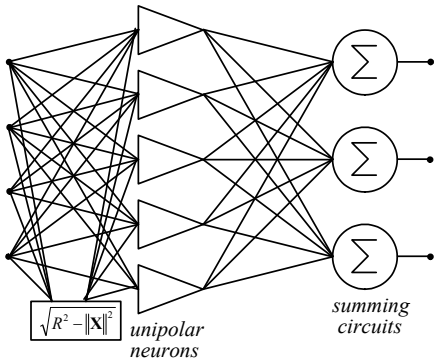



Fig. 14. Fuzzy controller based on conterpropagation network

All neurons in Fig. 14 have a unipolar activation function and if the system is properly designed, then for any input vector in certain areas only the neuron of this area produces +1 while all remaining neurons have zero values. In the case of when the input vector is close to a boundary between two or more regions, then all participating neurons are producing fractional values and the system output is generated as a weighted sum. For proper operation it is important that the sum of all outputs of the second layer must be equal to +1. In order to assure the above condition, an additional normalization block can be introduced, in a similar way as it is done in TSK fuzzy systems as shown in Fig. 12.

Another concept of replacing fuzzy systems with neural networks is shown in Fig. 15 [31,32]. This network can be considered as a fuzzy system with sigmoid membership functions [32].

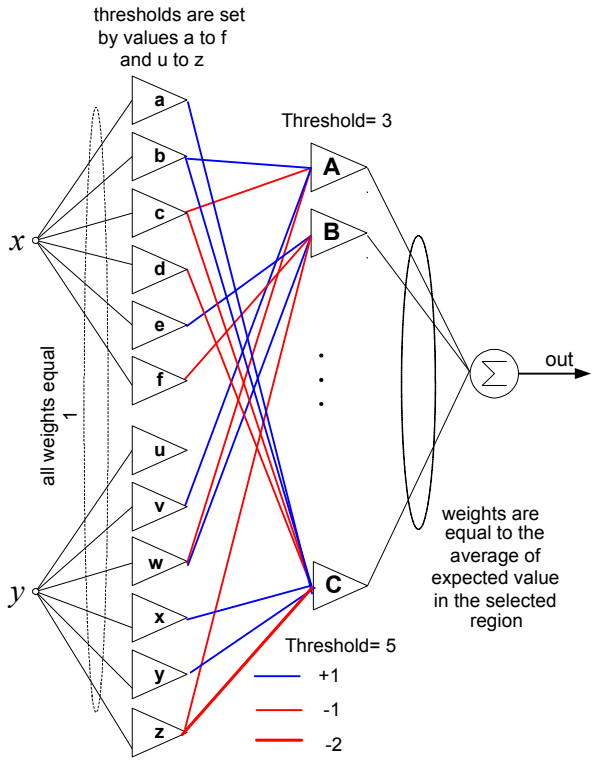



Fig. 15. Simple neural networks performing the function of TSK fuzzy system.

It is shown above that a simple neural network of Fig. 15 can replace a fuzzy system. All parameters of this network are directly derived from requirements specified for a fuzzy system and there is no need for a training process.
One may observe that if the training process is allowed then the network architecture of Fig. 15 can be significantly simplified. Let us compare in the following subsections the commonly used neural network architectures

## V. Comparison of Neural Networks and Fuzzy Systems

Fuzzy systems utilize the expert information in the form of a set of rules. There are several reasons for using fuzzy systems in control engineering practice. First, the dynamics of the system under interest is generally complicated, but sometimes its behavior can be defined more easily in linguistic terms. Second, fuzzy systems are suitable architectures for modification and tuning process, which provides some kind of adaptiveness through the on-line adjustment of parameters. The major advantage of fuzzy logic based systems is their ability to utilize expert knowledge and perception based information.

Artificial neural networks are well known by their property of performing complex nonlinear mappings. Earlier works on the mapping properties of these architectures have shown that neural networks are universal approximators [33].

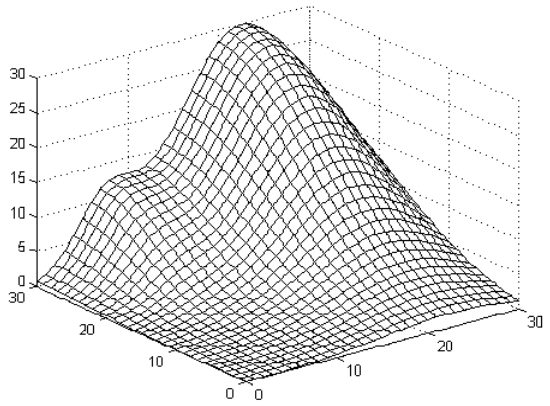



Fig. 16. Required control function and a comparison of the results obtained with microcontroller implementation using fuzzy and neural systems

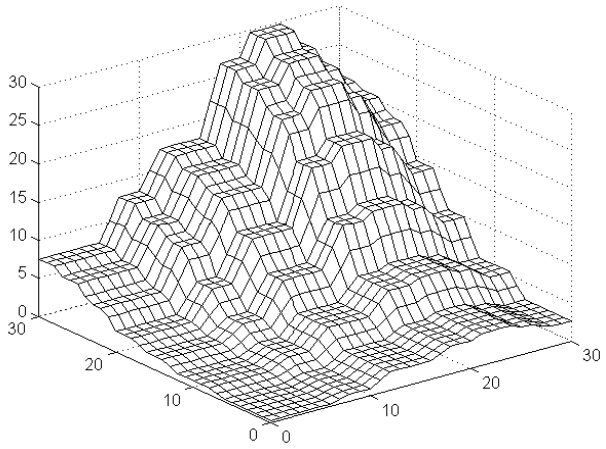



Fig. 17. Control surfaces obtained with Motorola microcontroller HC11 using fuzzy approach with trapezoidal membership functions (7 functions per input) and TSK defuzzification [1]

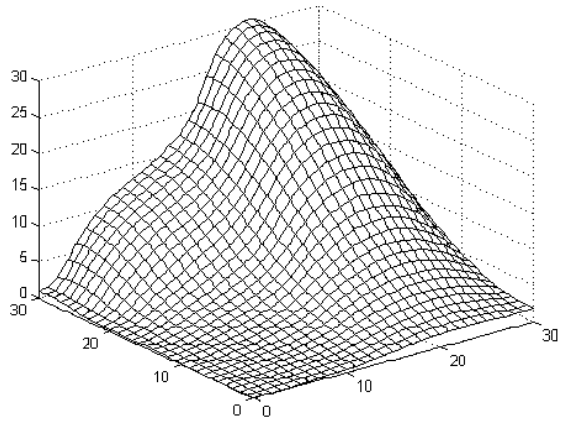



Fig. 18. Control surfaces obtained with Motorola microcontroller HC11 using fuzzy approach with six neurons 2-1-1-1-1-1 architecture and Elliot activation function. [1]

Figures 16 to 18 show a comparison of fuzzy and neural networks based system implemented in Motorola HC11 microcontroller. Motorola's 68HC711E9 is a low cost, 8-bit microprocessor; the on-board features of which are 512 bytes of RAM and EEPROM and 12K bytes of UV erasable EPROM. The processor was used with an 8 MHz crystal, allowing an internal clock frequency of 2 MHz.

Currently, fuzzy controllers are the most popular choice for hardware implementation of complex control surfaces because they are easy to design. Neural controllers are more complex and harder to train, but provide an outstanding control surface with much less error than that of a fuzzy controller.

A drawback of neural controllers is that the design process is more complicated than that of fuzzy controllers. However, this difficulty can be easily overcome with proper design tools. One severe disadvantage of a fuzzy system is its limited ability of handling problems with multiple inputs. In the case of neural networks such a limitation does not exist. Furthermore, control surfaces obtained from neural controllers also do not exhibit the roughness of fuzzy controllers that can lead to unstable or rough control.

TABLE IV
COMPARISON OF SOLUTIONS OF VARIOUS INCREASED COMPLEXITY PROBLEMS USING VARIOUS ALGORITHMS AND FCN ARCHITECTURES

| | Fuzzy System (Zadeh) | Fuzzy System (TSK) | Neural Network 2-1-1-1 | Neural Network 2-1-1-1-1-1 |
|---|---|---|---|---|
| Length of code | 2324 | 1502 | 680 | 1119 |
| Time (ms) | 1.95 | 28.5 | 1.72 | 3.3 |
| MSE Error | 0.945 | 0.309 | 0.000578 | 0.000093 |

Both neural networks and fuzzy systems are capable of approximating any nonlinear function, but their implementation in silicon is not easy. Both neural and fuzzy systems have their advantages and limitations.

One may notice that TSK fuzzy controller can be easily replaced by neural network with very simple architecture. In this case the intuitive fuzzy rules can be used as patterns to train neural networks. This approach is not only very simple but it also produces a smooth control surface. In most cases these neural networks which are replacing fuzzy systems require hardware.

## VI. CONCLUSION

The paper describes basic concepts of neural networks and fuzzy systems. It is shown that most commonly used neural network architecture of MLP – Multi Layer Perceptron is also one of the least efficient ones. Also most commonly used EBP – Error Back Propagation algorithm is not only very slow, but also it is not able to find solutions for optimal neural network architectures.

EBP can solve problems only when large number of neurons is used, but this way neural network loses its generalization property. Performances of both fuzzy systems and neural networks are compared leading to the conclusion that neural networks can produce much more accurate nonlinear mapping that are simple to implement. At the end of the presentation several concepts of neuro-fuzzy systems are compared.

## REFERENCES


[1] B. K. Bose, "Neural network applications in power electronics and motor drives – An introduction and perspective," *IEEE Trans. on Industrial Electronics*, Vol. 54, No. 1, pp. 14-33, February 2007.

[2] T. Orlowska-Kowalska, M. Dybkowski, K. Szabat, "Adaptive Sliding-Mode Neuro-Fuzzy Control of the Two-Mass Induction Motor Drive Without Mechanical Sensors ," *IEEE Trans. on Industrial Electronics,* vol. 57, no. 2, pp. 553 - 564 , Feb 2010.

[3] *Industrial Electronics Handbook,* 2nd Edition, (editors: B. M. Wilamowski and J.D.Irwin) *vol. 5 – Intelligent Systems,* CRC Press 2011.

[4] B. M Wilamowski and J. Binfet " Microprocessor Implementation of Fuzzy Systems and Neural Networks ", *International Joint Conference on Neural Networks (IJCNN'01),* pp. 234-239, Washington DC, July 15-19, 2001

[5] B. M. Wilamowski, "Special Neural Network Architectures for Easy Electronic Implementations" *POWERENG'09 - International Conference on Power Engineering, Energy and Electrical Drivers*, Lisbon, Portugal, March 18-20.

[6] P. D. Wasserman, Neural Computing Theory and Practice. Van Nostrand Reinhold, New York, 1989.

[7] J.M. Zurada, Artificial Neural Systems, PWS Publishing Company, St. Paul, MN, 1995

[8] S. Haykin, Neural Networks – A Comprehensive Foundation, Prentice Hall 1999.

[9] K. Chen and A. Salman, "Learning Speaker-Specific Characteristics with a Deep Neural Architecture," *IEEE Trans. on Neural Networks*, vol. 22, no. 11, pp. 1744-1756, 2011.

[10] I. Arel, D. C. Rose and T. P. Karnowski, "Deep Machine Learning – A New Frontier in Artificial Intelligence Research," *IEEE Computational Intelligence Magazine*, vol. 5, no. 4, pp. 13-18, 2010.

[11] B. M. Wilamowski, Hao Yu, and Kun Tao Chung "Parity-*N* Problems as a Vehicle to Compare Efficiency of Neural Network Architectures" *Industrial Electronics Handbook, vol. 5 – Intelligent Systems,* 2nd Edition, chapter 10, pp. 10-1 to 10-8, CRC Press 2011.

[12] David Hunter, Hao Yu, Michael S. Pukish, Janusz Kolbusz, Bogdan M. Wilamowski, "Selection of Proper Neural Network Sizes and Architectures - A Comparative Study" *IEEE Trans. on Industrial Informatics,* vol 8, n.2, May 2012.

[13] D. E Rumelhart., G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors". *Nature*, vol. 323, pp. 533-536, 1986.

[14] P.J. Werbos "Back-propagation: Past and Future". *Proceeding of International Conference on Neural Networks*, San Diego, CA, 1, 343-354, 1988.

[15] Scott E. Fahlman. Faster-learning variations on back-propagation: An empirical study. In T. J. Sejnowski G. E. Hinton and D. S. Touretzky, editors, 1988 Connectionist Models Summer School, San Mateo, CA, 1988. Morgan Kaufmann

[16] B. M. Wilamowski and H. Yu, "Neural Network Learning without Backpropgation," *IEEE Trans. on Neural Networks*, vol. 21, no. 11, pp. 1793-1803, Nov. 2010.

[17] B. M. Wilamowski, "Advanced Learning Algorithms", *INES'09 – 13-th International Conference on Intelligent Engineering Systems,* Barbados, April 16-18, 2009 pp. 9-17.

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

[19] B. M. Wilamowski, N. J. Cotton, O. Kaynak, G. Dundar, "Computing Gradient Vector and Jacobian Matrix in Arbitrarily Connected Neural Networks," *IEEE Trans. on Industrial Electronics,* vol. 55, no. 10, pp. 3784-3790, Oct 2008

[20] B. M. Wilamowski, N. Cotton, J. Hewlett, and O. Kaynak, "Neural Network Trainer with Second Order Learning Algorithms", 11th *INES 2007 -International Conference on Intelligent Engineering Systems,* Budapest, Hungary, June 29 2007-July 1 2007, pp. 127-132.

[21] Howard Demuth and Mark Beale. Neural Network Toolbox, User's Guide, Version 4. The MathWorks, Inc., Natick, MA, revised for version 4.0.4 edition, October 2004 http://www.mathworks.com.

[22] Hao Yu and B. M. Wilamowski, "C++ Implementation of Neural Networks Trainer", 13-th International Conference on Intelligent Engineering Systems, INES-09, Barbados, April 16-18, 2009

[23] B. M. Wilamowski and H. Yu " Improved Computation for Levenberg Marquardt Training", *IEEE Trans. on Neural Networks* vol. 21, 2010

[24] B. M. Wilamowski and H. Yu, "Neural Network Learning without Backpropgation," *IEEE Trans. on Neural Networks*, vol. 21, no. 11, pp. 1793-1803, Nov. 2010.

[25] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, Vol. 8, pp. 338-353, 1965.

[26] E. H. Mamdani, "Application of Fuzzy Algorithms for Control of Simple Dynamic Plant," *IEEE Proceedings*, Vol. 121, No. 12, pp. 1585-1588, 1974.

[27] M. Sugeno and G. T. Kang, "Structure Identification of Fuzzy Model," *Fuzzy Sets and Systems*, Vol. 28, No. 1, pp. 15-33, 1988.

[28] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Application to Modeling and Control," *IEEE Transactions on System, Man, Cybernetics*, Vol. 15, No. 1, pp. 116-132, 1985.

[29] D. Rutkowska, Y. Hayashi "Neuro-fuzzy systems approaches" *Int. J. Advanced Computational Intelligence*, vol. 3, no 3, pp. 177-185, 1999.

[30] R. Hecht-Nielsen, R. 1987. Counterpropagation networks. Appl. Opt. 26(23):4979-4984

[31] B. M. Wilamowski, "Methods of Computational Intelligence for Nonlinear Control Systems" *ICCAE' 05 International Conference on Control, Automation and System,* June 2-5, 2005, Gyeonggi-Do, Korea, pp. P1-P8

[32] J. Kolbusz, S. Paszczyński and B.M. Wilamowski, "Network traffic model for industrial environment", *IEEE Transaction on Industrial Informatics,* vol. 2, No. 4, pp. 213-220, 2006.

[33] B. M. Wilamowski and O. Kaynak, "Oil Well Diagnosis by sensing Terminal Characteristics of the Induction Motor," *IEEE Transactions on Industrial Electronics,* Vol 47, No 5, pp. 1100-1107, October 2000.