

Neural Networks in Designing Fuzzy Systems for Real World Applications *

S. K. Halgamuge, M. Glesner
Darmstadt University of Technology
Institute of Microelectronic Systems
Karlstr. 15, D-64283 Darmstadt, Germany
Tel.: ++49 6151 16-5136
Fax.: ++49 6151 16-4936
email: saman@microelectronic.e-
technik.th-darmstadt.de

Abstract—A special multilayer perceptron architecture known as FuNe I is successfully used for generating fuzzy systems for a number of real world applications. The FuNe I trained with supervised learning can be used to extract fuzzy rules from a given representative input/output data set. Furthermore, optimization of the knowledge base is possible including the tuning of membership functions. The new method employed to identify the rule relevant nodes before the rules are extracted makes FuNe I suitable for applications with large number of inputs.

Some of the real world applications in areas of state identification and image classification show encouraging results in a shorter development time. Expert knowledge is not compulsory but can be included in the automatically extracted knowledge base. The generated fuzzy system can be implemented in hardware very easily. A flexible prototype board is developed with a FPGA chip in order to run applications with up to 128 inputs and 4 outputs in realtime (1.25 million rules per second).

Keywords:

Fuzzy Logic; Neural Networks; Expert Systems; Pattern Recognition; Industrial Quality Control

I. INTRODUCTION

Fuzzy systems and neural networks, both model-free systems, contain their own advantages and drawbacks. One area of combining them, popularly known as fuzzy-neural networks, seeks maximization of the desirable properties and the reduction of disadvantages in both systems.

Several methods of fusion from fuzzy systems and neural networks are reported in literature [15], [12], [8], [6], [18], [19]. Different learning strategies are used in those applications e.g. unsupervised learning [17], supervised learning [20] and differential competitive learning [18].

This paper presents the structure and new features of the fuzzy-neural system FuNe I, which was successfully employed in a number of real world industrial applications.

Section II illustrates some of the techniques used to design major parts of fuzzy systems as trainable feedforward neural networks. Section III presents the FuNe I fuzzy model and section IV illustrates its rule generation method. Section V highlights the optimization possibilities of the resulting system and section VI summarizes some of the real world industrial applications of FuNe I. Section VII shows that FuNe I can be implemented in conventional as well as special fuzzy processors and finally section VIII discloses some of the future developments.

II. FUZZY SYSTEMS AS FEEDFORWARD NEURAL NETWORKS

The McCulloch-Pitts model of a Neuron is mathematically described as:

$$O_i = a[f(B_i, W_{i1} * I_1, W_{i2} * I_2 \dots)], \quad (1)$$

where a is the activation function, $W_{i1}, W_{i2}, \dots$ are weights incorporated with inputs $I_1, I_2, \dots$, B_i is the bias weight or threshold of the neuron and f is the function estimating the net input. The activation function a can be linear, sigmoidal or

*published in "Fuzzy Sets and Systems", North Holland, Vol. 65, NO. 1, PP 1-12, 1994

hyperbolic tangential. The function f usually performs summation.

Fuzzy Systems can be mapped into feed forward type of neural networks. Those systems are denoted as fuzzy-neural networks in this paper. The neurons used in many fuzzy-neural networks have a slightly different structure. The activation a is restricted to be either linear or sigmoidal. For the function f multiplication and the soft versions of minimum ([2]) and maximum can also be used instead of the summation.

Soft minimum (Min), and soft maximum (Max) can be described as:

$$\text{Min}_K(I_1..I_n) = \frac{\sum_{i=1}^n I_i \cdot e^{-K \cdot I_i}}{\sum_{i=1}^n e^{-K \cdot I_i}} \quad (2)$$

$$\text{Max}_K(I_1..I_n) = \frac{\sum_{i=1}^n I_i \cdot e^{K \cdot I_i}}{\sum_{i=1}^n e^{K \cdot I_i}} \quad (3)$$

where K is a variable, that can be virtually increased to infinity, together with the network reaching the convergence. For the remainder of this paper f is only specifically mentioned, if other than summation.

A. Rule Neurons

The premise of a fuzzy rule can be implemented as a rule neuron. Three types of rules are considered for implementation:

- simple rules with premises containing a single fuzzy variable
- conjunctive rules with many fuzzy variables in premises
- disjunctive rules with many fuzzy variables in premises

The implementation of Min-Max, Product-Sum or Lukasiewicz T-norms and T-Conorms are straightforward using the neuron model used. In FuNe I, conjunction and disjunction operators are implemented as Min and Max, respectively; e.g. the premise of a conjunctive rule with two inputs can be implemented with a single neuron using the parameters: a = linear, $B_i = 0$, f = Min and weights $W_{i1} = W_{i2} = 1$.

B. Antecedent Membership Functions

Virtually any membership function can be obtained using a multi-layer perceptron network

that has to be separately trained. But using a lesser number of neurons, and exploiting the possibilities of shifting, scaling and reflecting the sigmoidal transfer function, a satisfactory solution can be reached, without elaborate training. Considering three possible adjectives: Low (L), Medium (M), and High (H), the formation of the membership functions is illustrated over leaf (see also Fig. 1). Two sigmoidal functions are useful in creation of membership functions:

$$a_1[I_i, C, \alpha] = \frac{1}{1 + e^{-C(I_i - \alpha)}} \quad (4)$$

$$a_2[I_i, C, \alpha] = \frac{1}{1 + e^{C(I_i - \alpha)}} \quad (5)$$

The sigmoid (4) is the mirror reflection of the sigmoid (5) on the Y axis. Gain C is a positive variable, used to change the steepness of the sigmoid curve (e.g. if C goes to infinity, the sigmoid curve tends to be the step function), and α is a positive variable employed in shifting the sigmoids:

- Low:

$$L = a_2[I_i, C_L, \alpha_L] \quad (6)$$

- High:

$$H = a_1[I_i, C_H, \alpha_H] \quad (7)$$

- Medium:

Medium can be realized in different ways. One way is to use two sigmoid neurons from both types. A third linear neuron with f = Min is connected to the two sigmoid neurons by fixed connection weights of unity.

$$M1 = a_1[I_i, C_{M1}, \alpha_{M1}] \quad (8)$$

$$M2 = a_2[I_i, C_{M2}, \alpha_{M2}] \quad (9)$$

$$M = \text{Min}\{M1, M2\} \quad (10)$$

Another way of implementing Medium is to subtract one shifted sigmoid neuron from another shifted sigmoid neuron using a third linear neuron:

$$M1 = a_1[I_i, C_{M1}, \alpha_{M1}] \quad (11)$$

$$M2 = a_1[I_i, C_{M2}, \alpha_{M2}] \quad (12)$$

$$M = \sum \{M1, -M2\} \quad (13)$$

It is to be noted that $\alpha_{M1} < \alpha_{M2}$ in both cases.

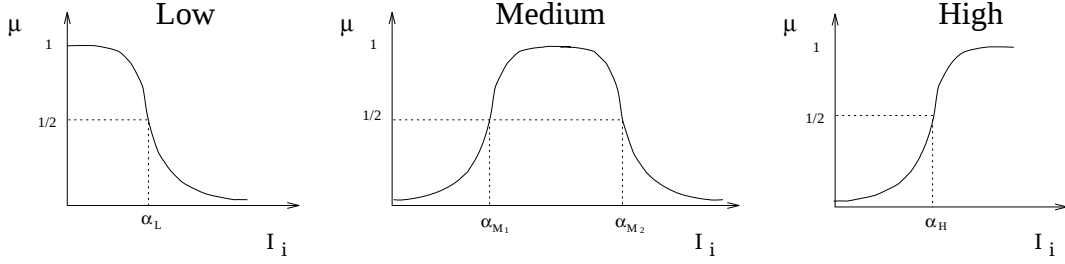


Figure 1: Antecedent Membership Functions

C. Consequent Membership Functions

Let $\mathbf{U}$ denote the finite set of possible normalized output values from the rule inference of a fuzzy system:

$$\mathbf{U} = \{U_1, U_2, \dots, U_n\} \quad (14)$$

where $\forall i: 0 \leq U_i \leq 1$. The fuzzy output must be defuzzified to get the crisp output U_{out} .

The set of maximum membership values μ (from the envelope shown in Fig. 2) corresponding to the set $\mathbf{U}$ can be denoted as :

$$\mu = \{\mu_1, \mu_2, \dots, \mu_n\} \quad (15)$$

Fig. 2 shows the creation of consequent membership values using two layers of linear neurons having $f = \text{Maximum}$. Each neuron in the first layer denotes an output membership function. Both layers are connected with weights corresponding to maximum possible membership values of the Low, Medium and High adjectives. Whenever a rule strength reaches the highest possible value (in the normalized case 1), the fuzzy output corresponds to consequent membership curves shown in dashed lines. Lower rule strengths represent the values of an appropriately scaled curve (e.g. solid curve μ_i). The curve $\mu_i^{\alpha_i}$ is the modified fuzzy output, Where $\alpha_i \in \alpha_I$ and α_I is the set *variable confidence measure*. This can be considered as an addition of nonlinear noise to the fuzzy output. The transparent neural network for obtaining the modified output is illustrated in [7]

Consequent membership functions can be tuned by adjusting the weights connecting the layers in a way similar to the tuning of antecedent membership functions.

D. Defuzzification

Standard defuzzification methods such as Center of Gravity (COG) can be implemented as neural networks [15].

$$U_{out}^{COG} = \frac{\sum_{i=1}^n \mu_i \cdot U_i}{\sum_{i=1}^n \mu_i} \quad (16)$$

But the limitations in standard defuzzification strategies compelled the authors to seek a parameterized approach, that can be optimized with theoretical or application oriented considerations. Customized defuzzification in general is advantageous to the application of a standard defuzzification method, and no standard method has so far proved to be the best, independent of the application.

The easiest way of implementing application oriented defuzzification is to use a trainable perceptron neural network. All the rule strengths are weighted and added to the output neuron with sigmoidal activation function. This is also called black box defuzzification in [6].

An extended family, Customizable Basic Defuzzification Distributions (**CBADD**) that can be implemented as neural networks is introduced in [7].

The defuzzified output according to CBADD is:

$$U_{out}^{CBADD} = \frac{\sum_{i=1}^n \mu_i^{\alpha_i} \cdot U_i}{\sum_{i=1}^n \mu_i^{\alpha_i}} \quad (17)$$

The proposed CBADD method reduces to BADD proposed by Yager et. al. [4] as a special case if $\forall i: \alpha_i = \alpha$. In contrast to the α in BADD, the variable confidence measure α_I in CBADD can also be negative. Whenever α_i is negative, $\mu_i^{\alpha_i} \geq 1$. It can be easily found by scaling (to

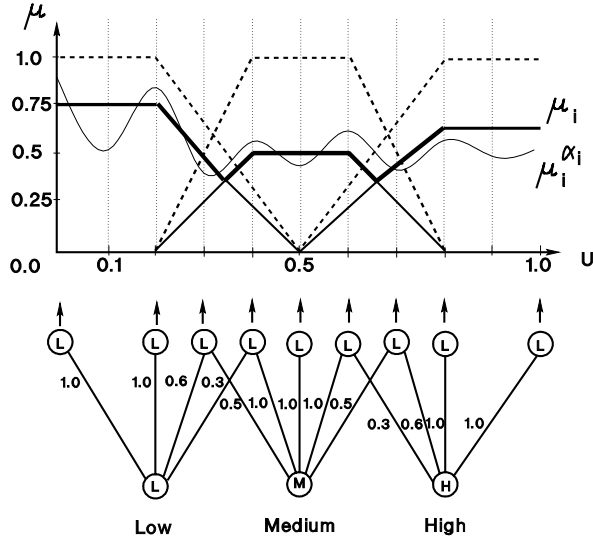


Figure 2: Consequent Membership Functions

bring back the maximum of $\mu_i^{\alpha_i}$ below a certain level) a function $\mu_i^{\alpha_i^*}$ which provides the same defuzzified output as the $\mu_i^{\alpha_i}$. Although the function $\mu_i^{\alpha_i^*}$ can be directly obtained with a non negative set of α_i , faster results may be obtained in neural learning by allowing α_i to take negative values.

Although the problem of local minima is unavoidable due to the use of gradient descent learning, the test results prove good approximation abilities in training with artificial sample data for different existing defuzzification methods on random fuzzy output curves and with application data from a fuzzy controlled reverse driving support of a truck with a long trailer.

CBADD is very flexible, which makes it ideal to employ a neural network to determine α_i from the training data of the application. A transparent trainable neural network is proposed for the implementation of CBADD in [7].

III. FU_{NE} I FUZZY MODEL

FuNe I and FuNe II are fuzzy-neural architectures, that construct and tune fuzzy systems from representative training data. The fuzzification methods used in FuNe I and FuNe II are identical and the extraction of “if” and “if not” rules are possible in both of the methods. While FuNe II uses the conventional fuzzy model with the

proposed parameterizable CBADD method with consequent membership functions, FuNe I has a modified structure. This modified fuzzy system generated from FuNe I training with gradient descent methods (e.g. backpropagation algorithm) is called FuNe I FS.

Assume for example rules R1, R2 and R3 and inputs $X = A1$ and $Y = B1$, where $Out1$, $Out2$ and $Out3$ denote the outputs and L, M and H are Low, Medium and High respectively. In FuNe I both positive type (if) and negative type (if not) rules are considered. The rule strengths are positively or negatively weighted depending on the type of the rule and directly summed up to the output neurons with sigmoidal activation functions:

R1: $W_1 * \text{If } X \text{ is L AND } Y \text{ is M THEN } Out1$

R2: $W_2 * \text{If } X \text{ is H OR } Y \text{ is L THEN } Out2$

R3: $W_3 * \text{If } X \text{ is M THEN } Out3$

If the membership functions of X and Y are μ_{L_X} , μ_{M_X} , μ_{H_X} , μ_{L_Y} , and μ_{M_Y} then the evaluation of the antecedents for rules are:

$$K_1 = \cap \{\mu_{L_X}(A1), \mu_{M_Y}(B1)\} \quad (18)$$

$$K_2 = \cup \{\mu_{H_X}(A1), \mu_{L_Y}(B1)\} \quad (19)$$

$$K_3 = \mu_{M_X}(A1) \quad (20)$$

where $\cap$ and $\cup$ are fuzzy set conjunction and disjunction denoted as the T-norm and T-Conorm, and K_1 , K_2 and K_3 are the strengths of rules R1, R2 and R3 respectively.

The FuNe I fuzzy model differs from conventional fuzzy models as it transfers the weighted sum of the fired rules into the crisp output as follows:

$$Out\ i = a_i \left(\sum_{j=1}^r W_{ij} * K_j \right) \quad (21)$$

where r is the number of rules, W_{ij} represents the weight of the connection from j th rule node to the i th output. The activation of the output neuron (a_i) is a sigmoidal function. In the proposed architecture, W_{ij} can also adopt negative values. This provides the effect of complementary rules to the output.

By considering the fact that the rules are weighted and summed to the output, the conjunctive and disjunctive rules with premises containing more than two fuzzy variables can be approximated by rules with maximum of two fuzzy variables in each of the premises.

IV. METHODS OF FUZZY RULE EXTRACTION

Several researchers explore the possibilities of rule extraction from a representative data set [16], [19], [12], [14], [22], [10]. Some of them “identify” the rule base from initially selected rules, reducing the problem of generating rules to a problem of forming the initial rule base.

The method of rule base extraction proposed in [21] employs backpropagation learning for positioning membership functions appropriate to the predetermined conjunctive rule structure. According to the rule structure proposed, each rule contains all the inputs in the premise and only the parameters of the membership functions are trained. This method is extended in [22] with the introduction of fuzzy basis functions and the application of orthogonal least-squares learning algorithm instead of backpropagation.

The advantage in this method is fast learning. But the loss of transparency due to the proposed rule structure is unavoidable specially if the number of rules or number of inputs are high, e.g. if number of inputs is 23 and the number of rules selected is 20, then each input is fuzzified with 20 membership functions and each rule will have all 23 inputs in the premise.

The method proposed in [14] is simple since almost all the possible fuzzy rules are considered in the classification procedure. A grade of certainty is assigned to each rule by training the network. The disadvantages are that no optimization such as tuning of membership functions is possible and the amount of rules overlapping or the existence of redundant rules in the rule base increases rapidly with the increase of number of inputs.

Horikawa et al. [12] presented a fuzzy modelling method using gradient descent learning. The identification of the premise and the consequence part for the proposed three different fuzzy models are the main features in this work. They proposed a new method for identification of rules from an initial rule base. The initial rule base can be created either using the expert knowledge or allowing

all the possible combinations. However, the latter method of creating the initial rule base will be impossible, if the number of inputs is high. A rule is selected according to the accuracy and the generality caused. Accuracy is determined by the summed squares of error and the generality is evaluated by the summed squares of error obtained by cross validating the train and test data sets.

The fuzzy-neural model used in FuNe I is similar to the Horikawa method of evaluation of premise and to the creation of antecedent membership functions. However, the initial rule base is not required in FuNe I, due to its unique architecture and the fuzzy model. Therefore, neither the absence of expert knowledge nor the higher number of inputs cause difficulties in generating a fuzzy system. In comparison to the method in [12], FuNe I uses a method based on trained weights.

If the repeated appearance of an input in the premise is avoided, the number of all possible simple, conjunctive and disjunctive rules depends upon the number of inputs (N_I) and the number of adjectives (N_A) selected:

$$N_I * N_A + 2 * \left(\sum_{i=2}^{N_I} N_A^i * \binom{N_I}{i} \right) \quad (22)$$

If the maximum number of variables in the premise is limited to 2, possible number of rules reduces to $N_I * N_A (N_A (N_I - 1) + 1)$. Therefore, Horikawa approach is only applicable to problems having a limited number of input variables and a limited number of adjectives selected.

FuNe I employs a different approach in finding the initial rule base. It identifies rule relevant nodes for conjunctive and disjunctive rules for each output first. This method reduces the initial rule base drastically. After a number of training steps with “frozen” membership functions, which allows the FuNe I training network (Fig.3) to avoid local minima in generating its rules, the network is trained with “free moving” membership functions. Only the dark lines in fuzzification and defuzzification blocks in Figure represent variable weights, and other connection have fixed unity weights. The dark circles represent neurons with sigmoid activation functions and the other neurons have linear activation functions. The white circles with $\cap$ ($\cup$) have Soft Min

(Soft Max) net input calculation and other neurons have summing inputs.

A. Identification of Rule Relevant Nodes

Since FuNe I contains positive weighted (if) as well as negative weighted (if not) rules, it is necessary to obtain two lists of rule relevant nodes for each operation (conjunction and disjunction) at each output.

Let us consider an example with three adjectives ($N_A = 3$). The fuzzified values of a crisp input I_i are L_i , M_i and H_i . Following steps (see also Fig. 4(a)) are to be taken in deciding whether the i^{th} input has any influence on a conjunctive (in this case Min) rule:

1. Connect the fuzzification layer to the node C_i (layer C in Fig. 3), that selects the maximum from the strongest membership values from all the inputs but the input i :

$$C_i = \text{Max}_{[j \neq i]}^{\forall j} [\text{Max}_j(L_j, M_j, H_j)] \quad (23)$$

2. Make connections to corresponding nodes R_{L_i} , R_{M_i} (layer R in Fig. 3) and R_{H_i} as follows:

$$R_{L_i} = \text{Min}(C_i, L_i) \quad (24)$$

$$R_{M_i} = \text{Min}(C_i, M_i) \quad (25)$$

$$R_{H_i} = \text{Min}(C_i, H_i) \quad (26)$$

3. The weights W_{L_i} , W_{M_i} and W_{H_i} connecting corresponding nodes to the output are initialized at random.
4. After training, connecting weights are analysed to decide which of the three fuzzy inputs is relevant to the if or if-not Min rule.

The last step for an example with 7 inputs is illustrated in table 1.

Extraction of rule relevant nodes is itself a rule based method. Table 1 illustrates some of the rules incorporated in this process. An exhibitive (inhibitive) weight is denoted by the symbol + (-). This does not necessarily mean that the weights are quantized. The symbol ++ (- -) indicates the weight considered is stronger than the other weights of the same input. A user definable relative similarity measure is used to decide upon the parameters of comparison.

i	W_{L_i}	W_{M_i}	W_{H_i}	positive nodes	negative nodes
1	+	+	+		
2	-	-	-		
3	++	+	+	L_3	
4	-	- -	-		M_4
5	-	+	+		L_5
6	-	-	+	H_6	
7	-	++	+	M_7	

Table 1: Lists of Min Rule relevant Nodes

If all three weights are “equally” exhibitive ($i = 1$) or inhibitive ($i = 2$) none of the three corresponding nodes influence the output other than providing an offset. Therefore, the inputs 1 and 2 can be ignored for that particular output. Given a corresponding node having a stronger exhibitive ($i = 3$) or inhibitive ($i = 4$) weight compared to other corresponding nodes of an input, it should be obviously taken to the positive or negative list of rule relevant nodes of the considered output.

Extraction of the Max-rule relevant nodes is analog to this method. A similar table can be used with:

$$C_i = \text{Min}_{[j \neq i]}^{\forall j} [\text{Max}_j(L_j, M_j, H_j)] \quad (27)$$

Corresponding nodes R_{L_i} , R_{M_i} and R_{H_i} can be obtained by replacing Min by Max in equations (24), (25) and (26).

All extracted Min and Max rule relevant nodes can be considered as candidates for Simple rule nodes. The results obtained by Min rule extraction for the example considered is given in Fig. 4(b).

V. OPTIMIZATION OF FuNe-I-FS

After the most challenging task of generating a fuzzy system, i.e. the creation of an initial knowledge base is accomplished, the optimization using the same training data set can be started. The number of rules generated in an initial fuzzy system may be too large for a particular application, and the user might want to define a limit, a relative weight, that can be used as a measure to remove weaker rules. If many generated rules including some of the important rules are removed, then the performance of the fuzzy system will be

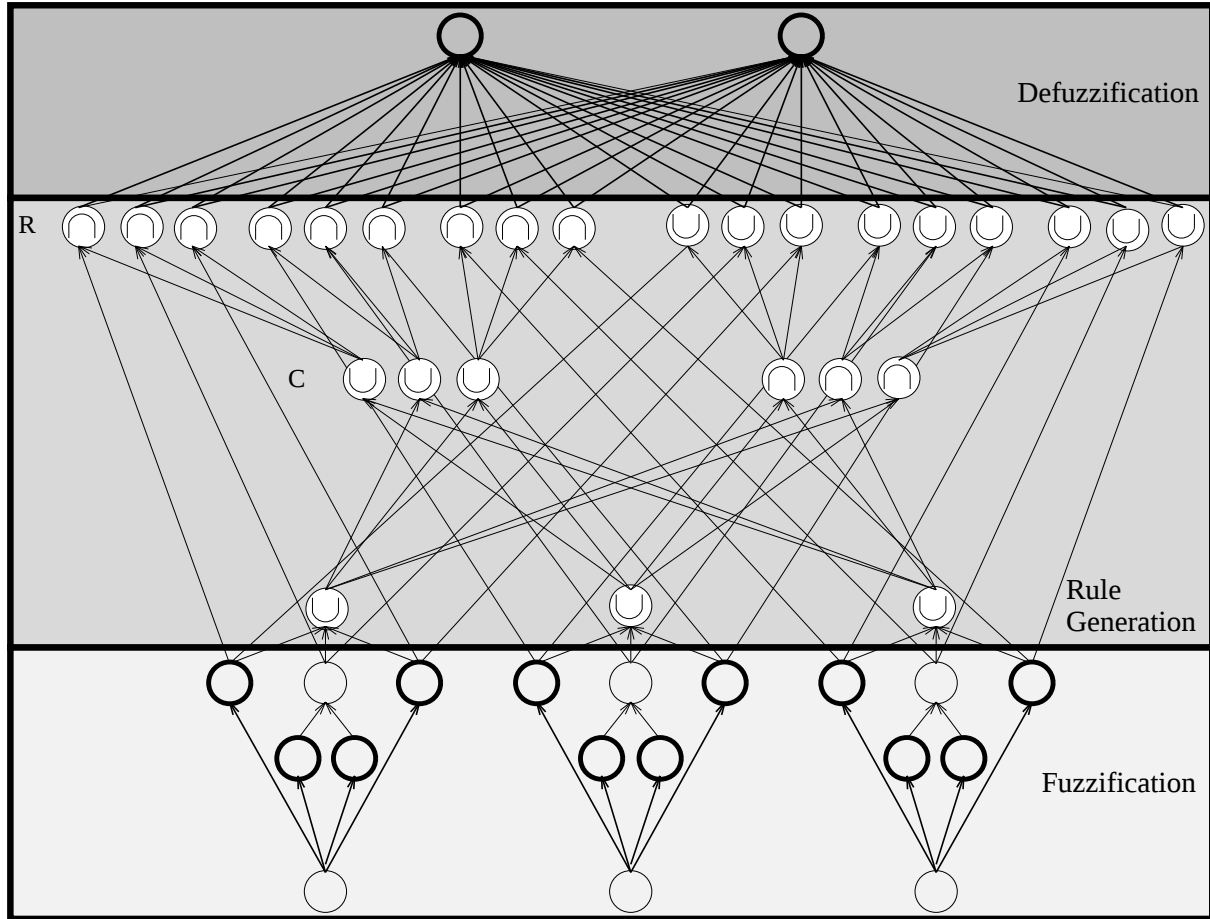


Figure 3: FuNe I training network

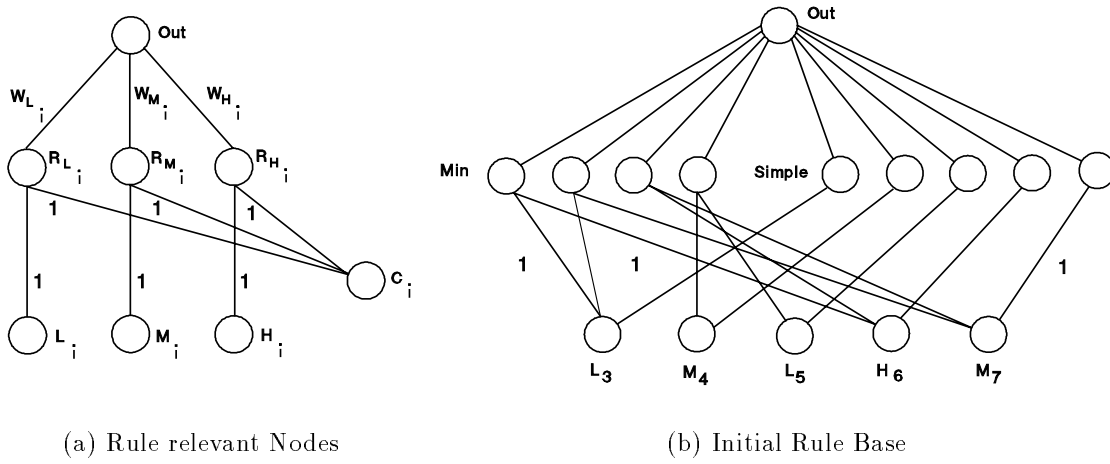


Figure 4: Extraction of Rules

lower. Therefore, a solution having the least number of rules and the best possible performance can be found iteratively. The method of optimization can be “on-line”, with reinforcement learning [2] or “off-line” as used in FuNe systems.

A. Parameter Tuning

The initially created antecedent membership functions illustrated in section II B, can be tuned using training data. Changing α values create a shift in membership functions and C values can be used to tune the slopes of sigmoid curves of the membership functions. In case of FuNe II consequent membership functions can also be tuned by allowing weights between two layers of neurons in Fig. 2 to be learned by training.

The extraction of rule relevant nodes reduces the possible number of rules drastically, but a second run of learning leads to further rule reduction. The rule nodes with weak weights can be removed by optimizing the FuNe-I-FS at last.

B. Rule Optimization

The optimization of rules is a compulsory process, since the number of rules to be processed is the major time limitation for realtime implementation of FuNe-I-FS.

An already generated or created rule base can be reduced effectively by training the FuNe-I-FS with input/output data. The weights connecting rules to the output layer can be pruned to obtain a smaller number of rules with high influence on outputs. An alternative method of optimizing the rule base is to use fuzzy entropy methods.

C. Posterior Input Feature Reduction

The input features for neural networks and fuzzy systems are generally selected on an intuitional basis. In classical neural techniques there is hardly any method to analyse the effectiveness of an input. FuNe I training automatically does it on the performance of the system. After the extraction of knowledge base, superfluous inputs which do not appear in rules can be removed. Furthermore, inputs in the premises of rules with weak weights can be checked for possible preprocessing mistakes, if expert opinion is opposed to the rule strength generated.

D. Knowledge Integration

Knowledge integration into the FuNe-I-FS is possible in different ways. Experts can integrate a priori knowledge as fuzzy rules and membership functions into the system. Another method is to include on-line reinforcement learning [2].

Although human expertise on technical process is biased compared to the automatically extracted knowledge, the importance of using or testing such knowledge should not be undermined. The expert opinion can be included to the FuNe training process before or after the generation of the FuNe-I-FS.

The expert knowledge can be added as new fuzzy rules to the FuNe-I-FS. The rule weights can be initialized and the system can be trained again. The optimization of the rule block should follow this step. This can also be included before the process of knowledge extraction as antecedent membership functions or rules.

VI. SUMMARY OF APPLICATIONS OF FU NE I

Basic requirement for applying FuNe I is the existence of a representative data set for training. FuNe I was first tested with a benchmark example known as classification of Iris species of Anderson [1]. The data contain a set of four dimensional vectors, each of which represents sepal length, sepal width, petal length and petal width of one of three iris subspecies Setosa, Versicolor, and Virginica. Measurements are taken from 50 plants from each subspecies. The data set is divided into training and test sets, each of them having 75 vectors. FuNe I is trained with the training set and the resulted FuNe I-FS is applied to the test set. As described earlier a solution having least number of rules and the best performance on the test set is attempted. A 99% classification rate could be achieved with 13 rules. Each training and optimization step was presented 600 times (epochs) with the training data set. Among the correctly classified 74 patterns of the test set, only 3 patterns are weakly classified, i.e. with output (classification strength) between 0.51 and 0.6 (Table 2). The ideal strength of classification is 1.

However, a more compact solution can be reached with 3 input and 7 rules and a lower (96%) classification rate on iris test data. The training data set can be classified completely by

classification strength	number of patterns
0.51–0.6	3
0.61–0.7	2
0.71–0.8	3
0.81–0.9	11
0.91–1.1	55

Table 2: Strength of Classification for Iris data

all the generated systems.

Three strong extracted rules having positive weights on outputs are shown below:

W_1 * If *petal length* is Low AND *petal width* is Low THEN Setosa

W_2 * If *petal length* is Medium AND *petal width* is Medium THEN Versicolor

W_3 * If *petal length* is High AND *petal width* is High THEN Virginica

FuNe I is successfully applied in number of real world problems in the areas of Image Classification and state identification.

A. State Identification

Identification of different possible states in mechatronical systems is a known method for estimating state dependent parameters. One example is the identification of road surface state of a moving car.

Physical limits of driving dynamics are determined by the friction between tyre and road. According to the measurements of friction taken from different real road surfaces, the maximum value of the friction coefficient μ is ranging from a low value (on ice) to a high value (on well textured dry asphalt) [3]. Therefore, automatic evaluation of the maximum possible μ is of importance for driving safety, specially in considering its use for the anti-lock braking system (ABS). Since this is closely related to the state of the road, identification of the state of road surface is performed with FuNe I [9]. The deformation of the tyre thread elements is transferred by a Hall sensor to the preprocessing unit, where the sensor signal is filtered and 7 input features for FuNe I are extracted. The FuNe-I-FS is generated from FuNe

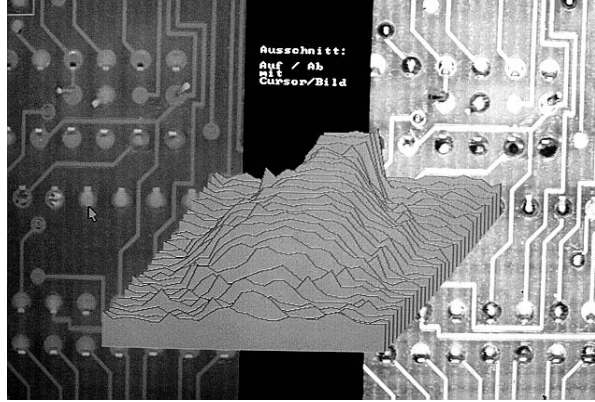


Figure 5: Solder joint images

I training with data drawn by driving in different known road surfaces.

B. Image Classification

From three applications of FuNe I in image classification can be reported: classification of solder joint images [10], underwater sonar image recognition [11] and handwritten digit recognition.

In classification of solder joint images 3D surface information (Fig.5 left) and 2D grey-level information (Fig.5 right) from a circuit board provided by a laser scanner system was used to automatically distinguish good solder joints from bad ones. After segmentation and centering of an individual solder joint 23 features were extracted as inputs to the network. FuNe I generated 27 Min, 4 Max and 13 Simple rules from training data. The resulting FuNe-I-FS was successfully tested (99%) with a different test data set. The posteriori feature reduction resulted in a reduction of the number of inputs from 23 to 13.

Underwater sonar image recognition is useful in identifying different objects in the deep sea. The priority was given in distinguishing artificial objects such as a metal cylinder (e.g. metal can) from natural objects (e.g. rock). The images are pre-processed into numerical data sets with 64 input features before classification [11]. This application shows the ability of the generated fuzzy system to recognize artificial objects, though training data used are incomplete.

Handwritten numerical character recognition is

another area where FuNe I is applied. Numerical characters from several writers are preprocessed and 36 input features are selected. After the normalization of the image at the beginning, the center of gravity (G) is found. The main guide in preprocessing is the 8 scan lines drawn from G, each in 45° of difference (see also Fig. 6). The lengths from G to two cutting points in each scan line are processed. Furthermore the number of gray points and corner points are also considered as input features.

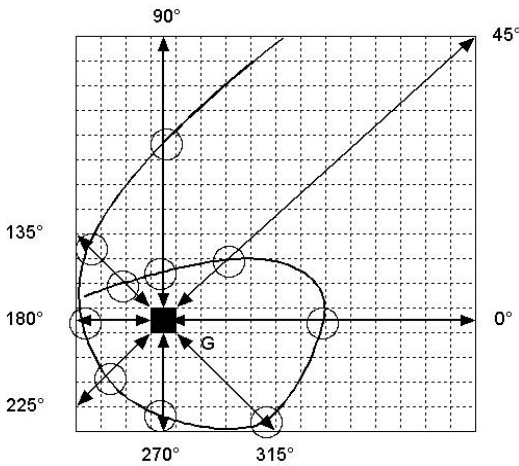


Figure 6: Preprocessing of numerical character

With the generated knowledge base, a 100% classification could be achieved for 9 out of 10 numerical digits. Only 99% classification could be achieved for the digit 6. In some situations this digit is classified as 5. This problem could be localised easily by observing the extracted rule base of the FuNe-I-FS. Expert help can be included in the system for the better separation of the digit 6 from 5.

VII. REAL TIME SOLUTIONS FOR FU NE I SYSTEM

Since real-time solutions for conventional fuzzy systems can be easily achieved due to the existence of special fuzzy processors such as FC 110 [13] and Fuzzy-166 [5], our first effort was to implement the resulting FuNe-I-FS in such a processor.

After successful implementation of some of the real world applications (solder joint classification

and the Iris problem) using a FC110 fuzzy processor, the possibility of implementing FuNe-I-FS in standard microcontrollers and digital signal processors is explored.

An application specific design in a Field Programmable Gate Array (FPGA) was tested for the benchmark example classification of Iris species. The FuNe-I-FS for IRIS application implemented in a prototype board containing a single Xilinx 4005 FPGA needs $10\mu S$ for each classification with a $10MHz$ clock. The same fuzzy system implemented in FC110 requires $94\mu S$ per classification.

A more flexible compiler is created to transfer FuNe-I-FS directly to net listings for Xilinx 4006 FPGA. With the use of this automatic tool for down loading into the prototype board built, it is possible to run a FuNe-I-FS with up to 128 crisp inputs, 256 rules and 4 crisp outputs in real-time. The speed reached with $10MHz$ is 1.25 Million Rules per Second. The sigmoidal membership functions are approximated with piecewise linear functions (up to 8 lines per membership function), which are much easier to implement in hardware.

VIII. DISCUSSION AND FUTURE WORK

The number of inputs and outputs in real world applications mentioned in this paper clearly shows the superiority of the FuNe I method compared to its commercial counterparts. FuNe-I-FS can be generated and implemented in hardware very fast if a representative data set exists. It is possible to extend the automatically extracted knowledge base with expert aid.

The successful application in real world problems confirms the abilities of this system. Hardware implementation of FuNe-I-FS is easier than that of a conventional fuzzy controller due to the use of “black box” defuzzification.

However, the generated FuNe-I-FS is static and will not be learned on-line. Therefore, the authors are working on extending the generated system to act as an adaptive fuzzy system (Adaptive FuNe I FS). The gradient descent learning algorithms are capable of solving many problems, if sample data are preprocessed appropriately. But the suitability of a neural learning algorithm is essentially problem dependent. Therefore, authors are working towards the expansion of FuNe I giving user to select the neural algorithm for the extraction

of fuzzy system, depending on the application.

IX. ACKNOWLEDGEMENT

The authors thank the Department of Corporate Production and Logistics (ZPL1 IF 43) of SIEMENS AG Munich for supporting the work with data from their laser scanner system, "Drittes Physikalisches Institut" of the University of Goettingen for providing us with their sonar data files, DFG-SFB special research area IMES for application data on road surface recognition and colleagues W. Poechmueller and H.J.-Herpel for their involvement in applying FuNe I for real world problems.

REFERENCES

- [1] E. Anderson. The Irises of the Gaspé Peninsula. *Bull. Amer. Iris Soc.*, 59:2–5, 1935.
- [2] H. R. Berenji and P. Khedkar. Learning and Tuning Fuzzy Logic Controllers Through Reinforcements. *IEEE Transactions on Neural Networks*, 3(5):724–740, 1992.
- [3] B. Breuer, U. Eichhorn, and J. Roth. Measurement of Tyre/Road Friction Ahead of the Car and Inside the Tyre. In *AUEC JAPAN*, Japan, 1992.
- [4] D. P. Filev and R. R. Yager. A Generalized Defuzzification Method via Bad Distributions. *International Journal of Intelligent Systems*, 6, 1991.
- [5] INFORM GmbH. *FUZZY-166 Hybrid Fuzzy Processor*. Aachen, Germany, 1992.
- [6] S. K. Halgamuge and M. Glesner. A Fuzzy-Neural Approach for Pattern Classification with the Generation of Rules based on Supervised Learning. In *Neuro-Nimes 92*, pages 165–173, Nanterre, France, November 1992. ISBN 2-906899-79-8.
- [7] S. K. Halgamuge and M. Glesner. The Fuzzy Neural Controller FuNe II with a New Adaptive Defuzzification Strategy Based on CBAD Distributions. In *European Congress on Fuzzy and Intelligent Technologies'93*, pages 852–855, Aachen, Germany, September 1993. Verlag der Augustinus-Buchhandlung. ISBN 3-86073-176-9.
- [8] S. K. Halgamuge and M. Glesner. Fuzzy Neural Fusion Techniques for Industrial Applications. In *ACM Symposium on Applied Computing (SAC'94)*, Phoenix, USA, March 1994.
- [9] S. K. Halgamuge, H.-J. Herpel, and M. Glesner. An Automotive Application With Neural Network Based Knowledge Extraction. In *Mechatronic Computer Systems for Perception and Action' 93*, pages 295–299, Halmstad, Sweden, June 1993. Center of Computer Studies. ISBN 91-630-1847-0.
- [10] S. K. Halgamuge, W. Poechmueller, and M. Glesner. A Rule based Prototype System for Automatic Classification in Industrial Quality Control. In *IEEE International Conference on Neural Networks' 93*, pages 238–243, San Francisco, USA, March 1993. IEEE Service Center; Piscataway. ISBN 0-7803-0999-5.
- [11] S. K. Halgamuge, W. Poechmueller, S. Ting, M. Hoehn, and M. Glesner. Identification of Underwater Sonar Images Using Fuzzy-Neural Architecture FuNe I. In *International Conference on Artificial Neural Networks'93*, pages 922–925, Amsterdam, The Netherlands, September 1993. Springer. ISBN 3-540-19839-3.
- [12] S. Horikawa, T. Furuhashi, and Y. Uchikawa. On Fuzzy Modeling Using Fuzzy Neural Networks with the Backpropagation Algorithm. *IEEE Transactions on Neural Networks*, 3(5), 1992.
- [13] Togai Infralogic Inc. *FC110 Togai Fuzzy Processor*. Irvine, U.S.A., 1991.
- [14] H. Ishibuchi, K. Nozaki, and H. Tanaka. - Pattern Classification by Distributed Representation of Fuzzy Rules. In *IEEE International Conference on Fuzzy Systems*, pages 643–650, San Diego, USA, 1992.
- [15] A. Kawamura and N. Watanabe and H. Okada and K. Asakawa. A Prototype of Neuro-Fuzzy Cooperation System. In *IEEE International Conference on Fuzzy Systems*, pages 1275–1282, San Diego, USA, 1992.

- [16] E. Khan and P. Venkatapuram. Neufuz: Neural Network Based Fuzzy Logic Design Algorithms. In *Second IEEE International Conference on Fuzzy Systems*, San Francisco, USA, March 1993.
- [17] T. Kohonen. *Self-Organization and Associative Memory*. Springer Verlag, 1989.
- [18] B. Kosko. *Neural Networks and Fuzzy Systems*. Prentice-Hall, USA, 1992.
- [19] C. T. Lin and C. S. G. Lee. Real-Time Supervised Structure/Parameter Learning for Fuzzy Neural Network. In *IEEE International Conference on Fuzzy Systems*, pages 1283–1291, San Diego, USA, 1992.
- [20] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, USA, 1986.
- [21] L. X. Wang and J. M. Mendel. Backpropagation Fuzzy System as Nonlinear Dynamic System Identifiers. In *IEEE International Conference on Fuzzy Systems*, pages 1409–1418, San Diego, USA, 1992.
- [22] L. X. Wang and J. M. Mendel. Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least Squares Learning. *IEEE Transactions on Neural Networks*, 3(5):807–814, 1992.