TR - H - 132

# Synergy of Modular Neural Networks by Fuzzy Logic

*Sung-Bae Cho*

1995. 3. 6

ATR人間情報通信研究所

〒619-02　京都府相楽郡精華町光台2-2　☎ 0774-95-1011

**ATR Human Information Processing Research Laboratories**

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-774-95-1011

Facsimile: +81-774-95-1008

# Synergy of Modular Neural Networks by Fuzzy Logic

Sung-Bae Cho*

*ATR Human Information Processing Research Laboratories*
*2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan*

March 3, 1995

**Abstract**

The concept of combining modular neural networks has been recently exploited as a new direction for the development of highly reliable neural network systems in the area of artificial neural networks. In this paper we present an efficient method of combining the modular networks based on fuzzy logic, especially the fuzzy integral. This method nonlinearly combines objective evidences, in the form of network outputs, with subjective evaluation of the reliability of the individual neural networks. Also, for more effective aggregation, we adopt the extension of the fuzzy integral with ordered weighted averaging operators. The experimental results with the recognition problem of on-line handwriting characters show that the performance of individual networks could be improved significantly.

## 1 Introduction

In the past several years, there has been a tremendous growth in the complexity of the recognition, estimation and control problems expected from neural networks. In solving these problems, we are faced with a large variety of learning algorithms and a vast selection of possible network architectures. After all the training, we choose the best network with a winner-takes-all cross-validatory model selection. However, recent theoretical and experimental work indicates that we can improve performance by considering methods for combining neural networks [1, 2, 3, 4, 5, 6]. One of the key issues of this approach is how to combine the results of the various networks to give the best estimate of the optimal result. There are a number of possible schemes for automatically optimizing the choice of individual networks and/or combining architectures.

A straight-forward approach is to decompose the problem into manageable ones for several different subnetworks and combine them via a gating network that decides which of the subnetworks should be used for each case. Hampshire and Waibel [7] have described a system of this kind that can be used when the decomposition into subtasks is known prior to training, and Jacobs *et al.* [3] have also proposed a supervised learning procedure for systems composed of many separate networks, each of which learns to handle a subset of the complete set of training instances. The subnetworks are local in the sense that the weights in one expert are decoupled from the weights in other subnetworks. However, there is still some indirect coupling because if some other network changes its weights, it may cause the gating network to alter the responsibilities that get assigned to the subnetworks.

*Tel: +81-7749-5-1076 Fax: +81-7749-5-1008 E-mail: sbcho@hip.atr.co.jp.

An alternative one is to independently generate a number of networks for possible generalizers and utilize all of them for obtaining robust output. While a usual scheme chooses one best network from amongst the set of candidate networks based on a winner-takes-all strategy, this approach keeps multiple networks and runs them all with an appropriate collective decision strategy. This is different from the aforementioned "adaptive mixtures of local experts" [3], in the sense that here networks do not decompose the task but learn globally the same task with different points of view. Several methods for combining evidence produced by multiple information sources have been applied in statistics, management sciences, and pattern recognition [8, 9]. A general result from the previous works is that averaging separate networks improves generalization performance for the mean squared error [6]. If we have networks of different accuracy, however, it is obviously not good to take their simple average or simple voting.

To give a solution to the problem, this paper presents a fusion method that considers the difference of performance of each network in combining the networks, which is based on the notion of fuzzy logic, especially the fuzzy integral. This method combines the outputs of separate networks with importance of each network, which is subjectively assigned as the nature of fuzzy logic. Also, we demonstrate the superior performance of the presented method and compare with conventional averaging methods by thorough experiments. Although a serious theoretical investigation is beyond the scope of this paper, we will demonstrate the effectiveness of the method by experimental results on a difficult OCR problem. For more details, refer the forthcoming publications made by the author [10, 11, 12].

The rest of this paper is organized as follows. Section 2 formulates the modular neural networks and considers possible methods for combining them. In section 3, we present the proposed architecture combining the modular neural networks with the fuzzy integral, and extends it with ordered weighted averaging (OWA) operators. Shown in section 4 are results with the recognition of on-line handwriting characters.

# 2 Backgrounds

## 2.1 Neural Network as Bayes Classifier

A neural network can be considered as a mapping device between an input set and an output set. Mathematically, a neural network represents a function $F$ that maps $I$ into $O$; $F : I \rightarrow O$, or $y = f(x)$ where $y \in O$ and $x \in I$. Since the classification problem is a mapping from the feature space to some set of output classes, we can formalize the neural network, especially two-layer feedforward neural network trained with the generalized delta rule, as a classifier.

Fig. 1 shows a two-layer neural network classifier with $T$ neurons in the input layer, $H$ neurons in the hidden layer, and $c$ neurons in the output layer. Here, $T$ is the number of features, $c$ is the number of classes, and $H$ is an appropriately selected number.[1] The network is fully connected between adjacent layers. The operation of this network can be thought of as a nonlinear decision-making process: Given an unknown input $X = (x_1, x_2, \ldots, x_T)$ and the class set $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$, each output neuron estimates the

[1] There has been a long debate as to how to determine $H$ as appropriate for any given problem. This has motivated the development of several constructive training techniques, such as Fahlman's *Cascade Correlation*.

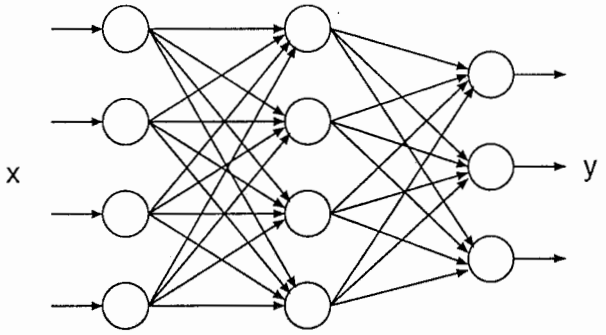



Figure 1: A two-layered neural network architecture.

probability $P(\omega_i|X)$ of belonging to this class by

$$P(\omega_i|X) \approx f\left\{\sum_{k=1}^{H} w_{ik}^{om} f\left(\sum_{j=1}^{T} w_{kj}^{mi} x_j\right)\right\}, \tag{1}$$

where $w_{kj}^{mi}$ is a weight between the $j$th input neuron and the $k$th hidden neuron, $w_{ik}^{m}$ is a weight from the $k$th hidden neuron to the $i$th class output, and $f$ is a sigmoid function such as $f(x) = 1/(1+e^{-x})$. The neuron having the maximum value is selected as the corresponding class. The key point is the training process determining the weight values, $w_{kj}^{mi}$ and $w_{ik}^{om}$.

On the other hand, the outputs of neural networks are not just likelihoods or binary logical values near zero or one. Instead, they are estimates of Bayesian *a posteriori* probabilities [13]. With a squared-error cost function, the network parameters are chosen to minimize the following:

$$E\left[\sum_{i=1}^{c}(y_i(X) - d_i)^2\right] \tag{2}$$

where $E[\cdot]$ is the expectation operator, $\{y_i(X) \mid i = 1,\ldots,c\}$ the outputs of the network, and $\{d_i \mid i = 1,\ldots,c\}$ the desired outputs for all output neurons. Performing several treatments in this formula allows it to cast in a form commonly used in statistics that provides much insight as to the minimizing values for $y_i(X)$:

$$E\left[\sum_{i=1}^{c}(y_i(X) - E[d_i|X])^2\right] + E\left[\sum_{i=1}^{c} var[d_i|X]\right] \tag{3}$$

where $E[d_i|X]$ is the conditional expectations of $d_i$, and $var[d_i|X]$ is the conditional variance of $d_i$.

Since the second term in (3) is independent of the network outputs, minimization of the squared-error cost function is achieved by choosing network parameters to minimize the first expectation term. This term is simply the mean-squared error between the network outputs $y_i(X)$ and the conditional expectation of the desired outputs. For a 1 of $M$ problem, $d_i$ equals 1 if the input $X$ belongs to class $\omega_i$ and 0 otherwise. Thus, the conditional expectations are the following:

$$\begin{aligned} E[d_i|X] &= \sum_{j=1}^{c} d_i P(\omega_j|X) \\ &= P(\omega_i|X) \end{aligned} \tag{4}$$

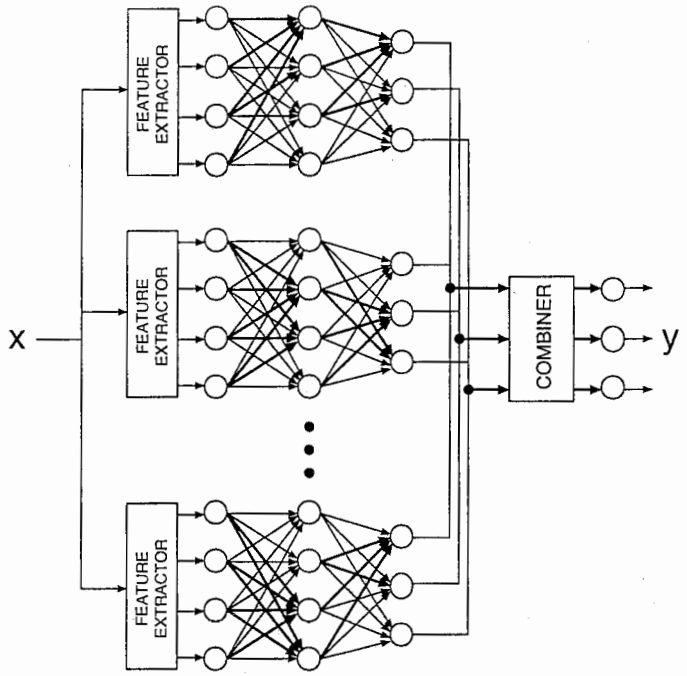



Figure 2: The modular neural network scheme combined by fusion method.

which are the Bayesian probabilities. Therefore, for a 1 of $M$ problem, when network parameters are chosen to minimize a squared-error cost function, the outputs estimate the Bayesian probabilities so as to minimize the mean-squared estimation error.

## 2.2 Modular Neural Networks

The network presented in the previous section trains on a set of example patterns and discovers relationships that distinguish the patterns. A network of a finite size, however, does not often load a particular mapping completely or it generalizes poorly. Increasing the size and number of hidden layers most often does not lead to any improvements. Furthermore, in complex problems such as character recognition, both the number of available features and the number of classes are large. The features are neither statistically independent nor unimodally distributed.

The basic idea of the modular network scheme is to develop $n$ independently trained neural networks with relevant features, and to classify a given input pattern by utilizing combination methods to decide the collective classification [1, 14] (see Fig. 2). Then it naturally raises the question of obtaining a consensus on the results of each individual network or expert.

In the meantime, we shall sketch how the modular neural network scheme generates an improved regression estimate [6]. Suppose that we have two finite data sets whose elements are all independent and identically distributed random variables: a training data set $A = \{(x_m, y_m)\}$ and a cross-validatory data set $CV = \{(x_l, y_l)\}$. Further suppose that we have used $A$ to generated a set of functions, $F = f_i(x)$, each element of which approximates $f(x)$. We would like to show that the MNN estimator, $f_{\mathrm{MNN}}(x)$, produces an improved approximation to $f(x)$.

Define the misfit of function $f_i(x)$, the deviation from the true solution, as $m_i(x) \equiv$

$f(x) - f_i(x)$. The mean square error can now be written in terms of $m_i(x)$ as

$$MSE[f_i] = E[m_i^2].$$

The average mean square error is therefore

$$\overline{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n} E[m_i^2].$$

Define the MNN regression function, $f_{\text{MNN}}(x)$, as

$$f_{\text{MNN}}(x) \equiv \frac{1}{n}\sum_{i=1}^{n} f_i(x) = f(x) - \frac{1}{n}\sum_{i=1}^{n} m_i(x).$$

If we now assume that the $m_i(x)$ are mutually independent with zero mean, we can calculate the mean square error of $f_{\text{MNN}}(x)$ as

$$\begin{aligned}
\text{MSE}[f_{\text{MNN}}] &= E\left[\left(\frac{1}{n}\sum_{i=1}^{n} m_i\right)^2\right] \\
&= \frac{1}{n^2}E\left[\sum_{i=1}^{n} m_i^2\right] + \frac{1}{n^2}E\left[\sum_{i\neq j} m_i m_j\right] \\
&= \frac{1}{n^2}E\left[\sum_{i=1}^{n} m_i^2\right] + \frac{1}{n^2}\sum_{i\neq j} E[m_i]E[m_j] \\
&= \frac{1}{n^2}E\left[\sum_{i=1}^{n} m_i^2\right],
\end{aligned}$$

which implies that

$$\text{MSE}[f_{\text{MNN}}] = \frac{1}{n}\overline{\text{MSE}}.$$

This is a powerful result because it tells us that by averaging regression estimates, we can reduce our mean square error by a factor of $n$ when compared to the population performance.

## 2.3 Synergy Methods

There might be two general approaches to combining the modular neural networks: One is based on fusion technique and the other on voting technique. In the methods based on the fusion technique, the classification of an input $X$ is actually based on a set of real value measurements:

$$P(\omega_i|X), \quad 1 \leq i \leq c.$$

They represent the probabilities that $X$ comes from each of the $c$ classes under the condition $X$. In the modular network scheme, each network $k$ estimates by itself a set approximations of those true values as follows:

$$P_k(\omega_i|X), \quad 1 \leq i \leq c, \;\; 1 \leq k \leq n.$$

One simple approach to combine the results on the same $X$ by all $n$ networks is to use the following average value as a new estimation of combined network:

$$P(\omega_i|X) = \frac{1}{n}\sum_{k=1}^{n} P_k(\omega_i|X), \quad 1 \leq i \leq c. \tag{5}$$

We can think of such a combined value as an averaged Bayes classifier. This estimation will be improved if we give the judge the ability to bias the outputs based on *a priori* knowledge about the reliability of the networks:

$$P(\omega_i|X) = \sum_{k=1}^{n} r_k P_k(\omega_i|X), \quad 1 \le i \le c, \tag{6}$$

$$\text{where} \sum_{k=1}^{n} r_k = 1. \tag{7}$$

Another alternative is to use the maximum value of $P_k(\omega_i|X)$ denoted by $P_m(\omega_i|X)$, to replace the correspondent average value. Since $\sum_{i=1}^{c} P_m(\omega_i|X) \neq 1$, we use the following normalized values as the new estimations:

$$P(\omega_i|X) = \frac{P_m(\omega_i|X)}{\sum_{j=1}^{c} P_m(\omega_j|X)}, \quad 1 \le i \le c. \tag{8}$$

The other method based on voting techniques considers the result of each network as an expert judgement. A variety of voting procedures can be adopted from group decision making theory: unanimity, majority, plurality, Borda count, and so on. In particular, we will introduce the two of them: majority voting and Borda count.

The majority voting rule chooses the classification made by more than half the networks. When there is no agreement among more than half the networks, the result is considered an error. To appreciate the network performance, let's assume that all neural networks arrive at the correct classification with a certain likelihood $1-p$ and that they make independent errors. The chances of seeing exactly $k$ errors among $n$ copies of the network is then

$$\binom{n}{k} p^k (1-p)^{n-k} \tag{9}$$

which gives the following likelihood of the majority rule being in error

$$\sum_{k>n/2}^{n} \binom{n}{k} p^k (1-p)^{n-k}. \tag{10}$$

It can be shown by induction for odd $n$ (or separately for even $n$) that provided $p < 1/2$, (10) is monotonically decreasing in $n$. In other words, if each network can get the correct answer more than half the time, and if network responses are independent, then the more networks used, the less the likelihood of an error by a majority decision rule. In the limit of infinite $n$, the coordinated error rate goes to zero.

For any particular class $c$, the Borda count is the sum of the number of classes ranked below $c$ by each network; Let $B_j(c)$ be the number of classes ranked below the class $c$ by the $j$th network. Then, the Borda count for class $c$ is $B(c) = \sum_{j=1}^{n} B_j(c)$. The final decision is given by selecting the class label whose Borda count is the largest.

## 3 Fuzzy Approach to Network Fusion

In this section, we shall describe the neuro-fuzzy architecture that utilizes the fuzzy integral for combining the modular neural networks. This method might produce better classification results, especially when we can assign the importance to each network.

### 3.1 Overview of Fuzzy Integral

The fuzzy integral is a nonlinear functional that is defined with respect to a fuzzy measure, especially $g_\lambda$-fuzzy measure introduced by Sugeno [15]. The ability of the fuzzy integral to combine the results of multiple sources of information has been established in several previous works [16, 17, 18]. In the following we shall introduce some definitions of it and present an effective method for combining the outputs of multiple networks with regard to subjectively defined importances of individual networks.

*Definition 1*: A set function $g : 2^X \rightarrow [0, 1]$ is called a fuzzy measure if
1) $g(\emptyset) = 0$, $g(X) = 1$,
2) $g(A) \leq g(B)$ if $A \subset B$,
3) If $\{A_i\}_{i=1}^{\infty}$ is an increasing sequence of measurable sets, then

$$\lim_{i \rightarrow \infty} g(A_i) = g(\lim_{i \rightarrow \infty} A_i).$$

Note that $g$ is not necessarily additive. This property of monotonicity is substituted for the additivity property of the measure.

From the definition of a fuzzy measure $g$, Sugeno introduced the so-called $g_\lambda$-fuzzy measures satisfying the following additional property: For all $A, B \subset X$ and $A \cap B = \emptyset$,

$$g(A \cup B) = g(A) + g(B) + \lambda g(A)g(B), \quad \text{for some } \lambda > -1.$$

It affords that the measure of the union of two disjoint subsets can be directly computed from the component measures.

*Example 1*: Consider the following case of $Y = \{y_1, y_2, y_3\}$ together with density values $g^1 = 0.34$, $g^2 = 0.32$, and $g^3 = 0.33$. Using the equation 14 (which will be introduced below), the Sugeno measure $g$ must have a parameter $\lambda$ satisfying $0.0359\lambda^2 + 0.3266\lambda + 0.001 = 0$. The unique root greater than $-1$ for this equation is $\lambda = 0.0305$, which produces the following fuzzy measure on the power set of $Y$:

| Subset $A$ | $g_{0.0305}(A)$ |
|---|---|
| $\emptyset$ | 0 |
| $\{y_1\}$ | 0.34 |
| $\{y_2\}$ | 0.32 |
| $\{y_3\}$ | 0.33 |
| $\{y_1, y_2\}$ | 0.6633 |
| $\{y_2, y_3\}$ | 0.6532 |
| $\{y_1, y_3\}$ | 0.6734 |
| $\{y_1, y_2, y_3\}$ | 1.0 |

As expected, the subset of criteria $\{y_1, y_3\}$ is more important for confirming the hypothesis than either subsets $\{y_1, y_2\}$ or $\{y_2, y_3\}$.

Using the notion of fuzzy measures, Sugeno developed the concept of the fuzzy integral, which is a nonlinear functional that is defined with respect to a fuzzy measure, especially $g_\lambda$-fuzzy measure [15, 16, 17].

*Definition 2*: Let $X$ be a finite set, and $h : X \rightarrow [0, 1]$ be a fuzzy subset of $X$. The fuzzy integral over $X$ of the function $h$ with respect to a fuzzy measure $g$ is defined by

$$\begin{aligned} h(x) \circ g(\cdot) &= \max_{E \subseteq X} \left[ \min \left( \min_{x \in E} h(x), g(E) \right) \right] \\ &= \sup_{\alpha \in [0,1]} \left[ \min \left( \alpha, g(h_\alpha) \right) \right] \end{aligned} \tag{11}$$

where $h_\alpha$ is the $\alpha$ level set of $h$,

$$h_\alpha = \{x \mid h(x) \geq \alpha\}. \tag{12}$$

The following properties of the fuzzy integral can be easily proved [17].
1) If $h(x) = c$, for all $x \in X$, $0 \leq c \leq 1$, then

$$h(x) \circ g(\cdot) = c.$$

2) If $h_1(x) \leq h_2(x)$ for all $x \in X$, then

$$h_1(x) \circ g(\cdot) \leq h_2(x) \circ g(\cdot).$$

3) If $\{A_i \mid i = 1, \ldots, n\}$ is a partition of the set $X$, then

$$h(x) \circ g(\cdot) \geq \max_{i=1}^{n} e_i,$$

where $e_i$ is the fuzzy integral of $h$ with respect to $g$ over $A_i$. For further details on the properties of the fuzzy integral and associated fuzzy measures for aggregating information, see the recent publication made by Yager [18].

To get some intuition for the fuzzy integral we consider the following interpretation. $h(y)$ measures the degree to which the concept $h$ is satisfied by $y$. The term $\min_{y \in E} h(y)$ measures the degree to which the concept $h$ is satisfied by all the elements in $E$. Moreover, the value $g(E)$ is a measure of the degree to which the subset of objects $E$ satisfies the concept measured by $g$. Then, the value obtained from comparing these two quantities in terms of the min operator indicates the degree to which $E$ satisfies both the criteria of the measure $g$ and $\min_{y \in E} h(y)$. Finally, the max operation takes the biggest of these terms. One can interpret the fuzzy integral as finding the maximal grade of agreement between the objective evidence and expectation.

### 3.2 Fuzzy Integral for Network Fusion

The calculation of the fuzzy integral when $Y$ is a finite set is easily given. Let $Y = \{y_1, y_2, \ldots, y_n\}$ be a finite set and let $h : Y \rightarrow [0, 1]$ be a function. Suppose $h(y_1) \geq h(y_2) \geq \ldots \geq h(y_n)$, (if not, $Y$ is rearranged so that this relation holds). Then a fuzzy integral, $e$, with respect to a fuzzy measure $g$ over $Y$ can be computed by

$$e = \max_{i=1}^{n} [\min\ (h(y_i),\ g(A_i))] \tag{13}$$

where $A_i = \{y_1, y_2, \ldots, y_i\}$.

Note that when $g$ is a $g_\lambda$-fuzzy measure, the values of $g(A_i)$ can be determined recursively as

$$\begin{aligned} g(A_1) &= g(\{y_1\}) = g^1 \\ g(A_i) &= g^i + g(A_{i-1}) + \lambda g^i g(A_{i-1}), \quad \text{for } 1 < i \leq n. \end{aligned}$$

$\lambda$ is given by solving the equation

$$\lambda + 1 = \prod_{i=1}^{n} (1 + \lambda g^i) \tag{14}$$

where $\lambda \in (-1, +\infty)$, and $\lambda \neq 0$. This can be easily calculated by solving an $(n-1)$st degree polynomial and finding the unique root greater than $-1$. Thus the calculation of

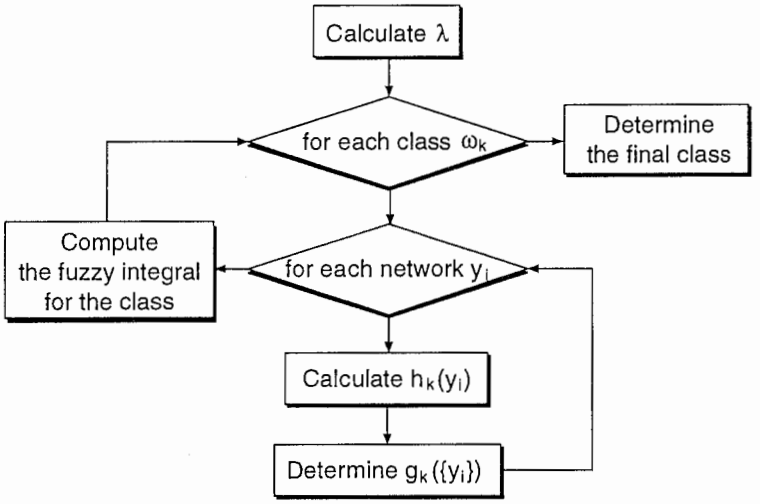



Figure 3: Algorithm of the network fusion by the fuzzy integral.

the fuzzy integral with respect to a $g_\lambda$-fuzzy measure would only require the knowledge of the density function, where $i$th density, $g^i$, is interpreted as the degree of importance of the source $y_i$ towards the final evaluation.

Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$ be a set of classes of interest. Note that each $\omega_i$ may, in fact, be a set of classes by itself. Let $Y = \{y_1, y_2, \ldots, y_n\}$ be a set of neural networks, and $A$ be the object under consideration for recognition. Let $h_k : Y \rightarrow [0, 1]$ be the partial evaluation of the object $A$ for class $\omega_k$, that is, $h_k(y_i)$ is an indication of how certain we are in the classification of object $A$ to be in class $\omega_k$ using the network $y_i$, where a 1 indicates absolute certainty that the object $A$ is really in class $\omega_k$ and 0 implies absolute certainty that the object $A$ is not in $\omega_k$.

Corresponding to each $y_i$ the degree of importance, $g^i$, of how important $y_i$ is in the recognition of the class $\omega_k$ must be given. These densities can be subjectively assigned by an expert, or can be induced from data set. The $g^i$'s define the fuzzy density mapping. Hence $\lambda$ is calculated using (14) and thereby the $g_\lambda$-fuzzy measure, $g$, is constructed. Now, using (13) to (14), the fuzzy integral can be calculated. Finally, the class $\omega_k$ with the largest integral value is chosen as the output class. Fig. 3 illustrates the details of how the consensus is formed.

*Example 2*: Using the Example 1, how the consensus decision is performed by the fuzzy integral can now be described for a two class problem, which discriminates handwriting characters 6 and 4. Suppose that we obtain the network outputs for an input image as shown in Fig 4: $h(y_1) = 0.6$, $h(y_2) = 0.7$, and $h(y_3) = 0.1$, for class 1. For class 2, $h(y_1) = 0.8$, $h(y_2) = 0.3$, and $h(y_3) = 0.4$. The following table shows how the consensus is formed, where $H(E) = min(h(y_i), g(A_i))$.

| Class | $h(y_i)$ | $g(A_i)$ | $H(E)$ | $max[H(E)]$ |
|---|---|---|---|---|
| 1 | 0.7 | $g(\{y_2\}) = g^2 = 0.32$ | 0.32 | |
| | 0.6 | $g(\{y_2, y_1\}) = g^2 + g^1 + \lambda g^2 g^1 = 0.66$ | 0.6 | $\surd$ |
| | 0.1 | $g(\{y_2, y_1, y_3\}) = 1.0$ | 0.1 | |
| 2 | 0.8 | $g(\{y_1\}) = g^1 = 0.34$ | 0.34 | |
| | 0.4 | $g(\{y_1, y_3\}) = g^1 + g^3 + \lambda g^1 g^3 = 0.67$ | 0.4 | $\surd$ |
| | 0.3 | $g(\{y_1, y_3, y_2\}) = 1.0$ | 0.3 | |

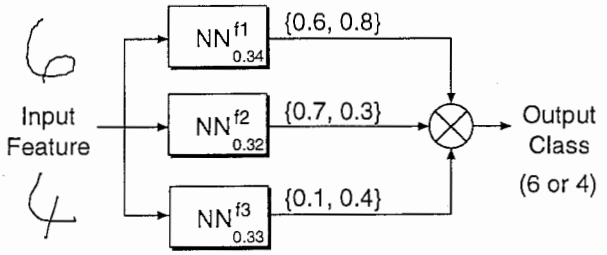



Figure 4: A simple example of network outputs for a two class problem.

Finally, the class 1 is selected as output. In the meantime, in case that we use the weighted average instead of the fuzzy integral, the class 2 is chosen as the correct class because the class 2 yields 0.5 ($0.34 \times 0.8 + 0.32 \times 0.3 + 0.33 \times 0.4$) whereas the class 1 produces 0.46 ($0.34 \times 0.6 + 0.32 \times 0.7 + 0.33 \times 0.1$). This example shows how the minute differences of $g^i$ conspire so as to dramatically change the performance compared to simple averaging.

### 3.3 Extension of Fuzzy Integral with OWA Operators

In [18] Yager extended the fuzzy integral with two special families of OWA operators, S-OWA-AND and S-OWA-OR.

*Definition 3*: A mapping $F$ from

$$I^n \rightarrow I(\text{where } I = [0,1])$$

is called an OWA operator of dimension $n$ if associated with $F$ is a weighting vector $W$,

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_n \end{bmatrix}$$

such that

1) $W_i \in (0,1)$

2) $\sum_i W_i = 1$

and where

$$F(a_1, a_2, \cdots, a_n) = W_1 b_1 + W_2 b_2 + \cdots + W_n b_n,$$

where $b_i$ is the $i$th largest element in the collection $a_1, a_2, \cdots, a_n$.

In [19] Yager shows how different assignment of the weights allows implementation of different quantifiers. For example, $W^*$, with $W_1 = 1$ and $W_i = 0$, $i \neq 1$ provides the max operator. $W_*$ with $W_n = 1$ and $W_i = 0$ for $i \neq n$ gives us the min operator. In addition, $W_i = 1/n$ gives us the average $1/n \sum a_i$. This shows the more of the weights near the bottom the more "and-like" the aggregation while the more of weights near the top the more "or-like" the aggregation.

There are two special families of OWA operators which are useful for extending the fuzzy integral [18]. These are called the S-OWA-AND and S-OWA-OR operators. The S-OWA-AND operators are defined such that

$$\hat{F}_\alpha(a_1, \cdots, a_n) = \frac{1-\alpha}{n} \sum_i a_i + \alpha \min_i a_i.$$

The S-OWA-AND operators provide for *and-like* aggregations. In the formulation for the fuzzy integral we can obtain the effect of S-OWA-AND operators by replacing $\min_{x\in E} h(x)$ with

$$\frac{1-\alpha}{\mathrm{Card}E}\sum_{x\in E} h(x) + \alpha \min_{x\in E} h(x).$$

The parameter $\alpha$ lies in the unit interval. The closer $\alpha$ is to one the more it becomes *and-like* aggregation.

On the contrary, S-OWA-OR operator provides for an *or-like* aggregation. This operator is defined such that

$$\tilde{F}_\beta(a_1, \cdots, a_n) = \frac{1-\beta}{n}\sum_i a_i + \beta \max_i a_i.$$

This provides for an *or-like* aggregation. Here again the parameter $\beta$ lies in the unit interval and the closer $\beta$ is to 1, the more like a pure *or* the operation. This S-OWA-OR operator can be used to provide a further generalization of the fuzzy integral. Let us denote $\min(\min_{x\in E} h(x), g(E))$ as $H(E)$. The value of the fuzzy integral is requiring that *at least one* subset $E$ of $X$ satisfy $H(E)$. With $n$ the cardinality of $X$ we can change the aggregation to

$$\frac{1-\beta}{2^n}\sum_{E\subset X} H(E) + \beta \max_{E\subset X} H(E).$$

With this change, depending on the choice of $\beta$, we are requiring that *some* or *few* of the $E$ satisfy $H(E)$ rather than just one.

## 4 Experimental Results

In order to give an idea of practical application of the presented method to pattern recognition, a data set of handwriting characters was used as a source of both training and test samples. Handwriting characters were inputed to the computer (SUN workstation) by a Photron FIOS-6440 LCD tablet, which samples at the rate of 80 dots per second. The tasks were to classify Arabic numerals, uppercase letters, and lowercase letters collected from 13 different writers. The writers were told to draw the numerals and letters into prepared square boxes in order to facilitate segmentation.

An input character consists of a set of strokes, each of which begins with a pen-down movement and ends with a pen-up movement. Several preprocessing algorithms were applied to successive data points within each stroke to reduce quantization noise and fluctuations due to the writer's pen motion. The processes used were wild point reduction, dot reduction, hook analysis, three point smoothing, peak preserving filtering, and $N$ point normalization [20]. Data points, representing single characters, were resampled with a fixed number of regularly spaced points. Then, a sequence of preprocessed data points was approximated by a sequence of 8-directional straight-line segments—the chain code, as used by Freeman [21].

To evaluate the performance of the proposed method, we implemented three different networks, each of which is a two-layered neural network having a different number of input neurons and 20 hidden neurons. $\mathrm{NN}_1$, $\mathrm{NN}_2$ and $\mathrm{NN}_3$ have 10, 15, and 20 input neurons, respectively. In each case, the network makes a decision based on its resolution. For example, $\mathrm{NN}_1$ uses sparsely sampled inputs, and in doing so is able to overcome variations in input noise. $\mathrm{NN}_3$, on comparison, uses a finer view of the input image. The selection of the features is largely adhoc and no attempt was made to find an optimal coding scheme although this is an important issue in character recognition schemes. Our objective here is

Table 1: The recognition rates of the fuzzy integral for different densities (%).

| Case | $g^1$ | $g^2$ | $g^3$ | Numeral | Upper | Lower |
|---|---|---|---|---|---|---|
| 1 | 0.1 | 0.2 | 0.3 | 78.4 | 71.6 | 65.4 |
| 2 | 0.1 | 0.3 | 0.2 | 79.0 | 73.4 | 66.8 |
| 3 | 0.2 | 0.1 | 0.3 | 78.8 | 72.2 | 64.8 |
| 4 | 0.2 | 0.3 | 0.1 | 79.4 | 73.8 | 69.2 |
| 5 | 0.3 | 0.1 | 0.2 | 79.8 | 74.0 | 66.2 |
| 6 | 0.3 | 0.2 | 0.1 | 80.2 | 75.2 | 70.4 |

Table 2: Fuzzy densities and the corresponding $\lambda$.

| Subject | $g^1$ | $g^2$ | $g^3$ | $\lambda$ |
|---|---|---|---|---|
| Numerals | 0.3430 | 0.3330 | 0.3230 | $-0.0149$ |
| Uppercases | 0.3447 | 0.3312 | 0.3240 | 0.0003 |
| Lowercases | 0.3370 | 0.3321 | 0.3312 | $-0.0009$ |

to evaluate and compare different fusion methods through an example which has a certain complexity and practical significance.

Each of the three networks was trained by the EBP algorithm with 40 samples per class, validated with another 500 samples, and tested on ten sets of additional samples collected from different 10 writers: The training process was stopped when the recognition rate over the validation set was optimized. This process and early stopping mechanism were adopted mainly for preventing networks from overtraining. The initial parameter values used for training were: Learning rate is 0.4 and momentum parameter is 0.6. An input vector is classified as belonging to the output class associated with the highest output activation. Each of the following experiments consisted of 10 trials in which the different data were made from different writers.

First, the behavior of the fuzzy integral of a function $h$ with respect to a $g_\lambda$-fuzzy measure, $g$ is examined. Table 1 shows these results. Here, each case shows a set of fuzzy densities corresponding to three networks and the recognition rates of numerals, uppercase letters, and lowercase letters using the fuzzy integral on the three networks. Using (14), the Sugeno measure $g$ must have a parameter $\lambda$ satisfying $0.006\lambda^2 + 0.11\lambda - 0.4 = 0$. The unique root greater than $-1$ for this equation is $\lambda = 3.109$.

As expected, the recognition results in the table depend on the $g$ values. When the $g$ values change, the new fuzzy integral value will change depending on how these changes are balanced with respect to the source corresponding to the fuzzy integral value. We assigned the fuzzy densities $g^i$, the degree of importance of each network, based on how good these networks performed on validation data. We computed these values as follows:

$$g^i = \frac{p_i}{\sum_j p_j} \cdot d\text{sum}, \tag{15}$$

where $p_i$ is the performance of network $\text{NN}_i$ for the validation data and $d$sum is the desired sum of fuzzy densities. The real values of these densities with the corresponding $\lambda$ are shown in table 2.

Table 3 reports the results of network fusion using the fuzzy integral on three different networks for numerals. In this table the value in the parentheses represent the confidence of the evaluation result. As can be seen, cases 2 and 3 were misclassified by $\text{NN}_3$ and $\text{NN}_2$,

Table 3: Results of network fusion using the fuzzy integral on three different networks for numerals.

| Data index | Actual class | Partial decision | | | Fuzzy integral decision |
|---|---|---|---|---|---|
| | | $NN_1$ | $NN_2$ | $NN_3$ | |
| 1 | 5 | 5 (0.9859) | 5 (0.8995) | 5 (0.9941) | 5 (0.9598) |
| 2 | 6 | 6 (0.9968) | 6 (0.9985) | 5 (0.3301) | 6 (0.6877) |
| 3 | 8 | 8 (0.9999) | 0 (0.0022) | 8 (0.9996) | 8 (0.6668) |
| 4 | 2 | 2 (0.9922) | 2 (0.9998) | 2 (0.9920) | 2 (0.9946) |
| 5 | 7 | 8 (0.0162) | 8 (0.0087) | 7 (0.9615) | 7 (0.3205) |
| 6 | 9 | 8 (0.0001) | 7 (0.1195) | 7 (0.4780) | 7 (0.1991) |
| 7 | 7 | 6 (0.0137) | 7 (0.9903) | 7 (0.9988) | 7 (0.6630) |
| 8 | 7 | 2 (0.1342) | 7 (0.9677) | 7 (0.0023) | 7 (0.3233) |
| 9 | 1 | 1 (0.9999) | 1 (0.9972) | 1 (0.9993) | 1 (0.9988) |
| 10 | 0 | 6 (0.7116) | 6 (0.4098) | 8 (0.8098) | 6 (0.3753) |
| 11 | 7 | 1 (0.1794) | 8 (0.0003) | 1 (0.0080) | 1 (0.0625) |
| 12 | 4 | 4 (0.9998) | 4 (0.9999) | 9 (0.9964) | 4 (0.6694) |
| 13 | 9 | 9 (0.9965) | 9 (0.9958) | 8 (0.0740) | 9 (0.6691) |
| 14 | 3 | 3 (0.9987) | 3 (0.9912) | 3 (0.9999) | 3 (0.9966) |
| 15 | 7 | 8 (0.0365) | 0 (0.0460) | 7 (0.4831) | 7 (0.1610) |
| 16 | 5 | 5 (0.9311) | 3 (0.1304) | 3 (0.6245) | 5 (0.3265) |
| 17 | 2 | 8 (0.3470) | 2 (0.9983) | 8 (0.2092) | 2 (0.3327) |
| 18 | 8 | 8 (0.9899) | 0 (0.9669) | 8 (0.6815) | 8 (0.8384) |
| 19 | 0 | 1 (0.0353) | 4 (0.0004) | 4 (0.0004) | 1 (0.0118) |
| 20 | 9 | 9 (0.7519) | 9 (0.3799) | 9 (0.9540) | 9 (0.6953) |
| 21 | 0 | 8 (0.8032) | 0 (0.9994) | 0 (0.9993) | 0 (0.6662) |
| 22 | 4 | 4 (0.9997) | 9 (0.9170) | 4 (0.9871) | 4 (0.7099) |
| 23 | 9 | 9 (0.9989) | 1 (0.9944) | 9 (0.9902) | 9 (0.6705) |
| 24 | 4 | 4 (0.9998) | 4 (0.9999) | 4 (0.9995) | 4 (0.9997) |
| 25 | 8 | 8 (0.9998) | 0 (0.9882) | 8 (0.8353) | 8 (0.6204) |

Table 4: Means and standard deviations of recognition rates (%).

| | Numeral | | Uppercase | | Lowercase | |
|---|---|---|---|---|---|---|
| Nets | Mean | S.D. | Mean | S.D. | Mean | S.D. |
| $NN_1$ | 82.6 | 6.36 | 73.2 | 8.95 | 73.9 | 7.73 |
| $NN_2$ | 81.2 | 7.16 | 68.6 | 9.14 | 71.8 | 8.86 |
| $NN_3$ | 81.0 | 7.15 | 70.8 | 10.60 | 72.1 | 9.30 |
| $NN_{all}$ | 77.6 | 6.31 | 72.1 | 8.93 | 74.7 | 10.01 |
| Voting | 84.9 | 8.31 | 74.0 | 9.28 | 74.6 | 7.97 |
| Average | 86.9 | 7.24 | 75.2 | 9.95 | 78.2 | 8.85 |
| Fuzzy | 88.1 | 7.14 | 76.1 | 9.85 | 80.3 | 7.24 |

respectively. However, in the final evaluations they were correctly classified. In cases 5 and 17, one network with strong evidence overwhelmed the other networks, producing correct classification. Furthermore, in case 15, the fuzzy integral made a correct decision despite that the partial decisions from the individual neural networks were completely inconsistent. The effect of misclassification by the other networks has given rise to small fuzzy integral values for the correct classification in this case.

Table 4 summarizes the recognition rates of numerals, uppercase letters, and lowercase letters for the three networks and the representative synergy methods like majority voting, average, and the fuzzy integral. All results are averaged over ten different sets of the data. In this table, $NN_1$ to $NN_3$ represent the three individual networks, and $NN_{all}$ a large network trained with all the features used by each network.

Although the network learned the training set almost perfectly in all three cases, the performances on the test sets are quite different. Furthermore, we can see that the performance did not improve by training a large network with considering all the features used by each network. This is a strong evidence that modular neural network might produce better result than conventional single network approach. The following test can further support to determine whether the fuzzy integral method is superior to the conventional method or not.

For a given test problem, let $f_i^a$ denote the solution at convergence for method $a$ using test data $i$. To test whether methods $a$ and $b$ have the same mean solution value, we compute the following statistic:

$$t = \frac{\sqrt{n}\bar{x}}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}} \tag{16}$$

where $n = 10$, $x_i = f_i^a - f_i^b$, and $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. (In this case the method $b$ is of the fuzzy integral.) From this value we can reject the null hypothesis that $H_0 : \bar{x} \leq 0$ in favor of the alternative that $\bar{x} > 0$ with significance level $\alpha$, where $\alpha = \Phi(t)$ and $\Phi(t)$ can be obtained from the table of percentage points of the $t$-distribution.

Since it follows $t$-distribution, an $\alpha$ point can be computed as the threshold $t_\alpha$, where $\alpha$ could be 95, 99.95 or 99.9%, Then, if

$$|t| > t_\alpha \tag{17}$$

the null hypothesis is rejected at a $100\% - \alpha$ level of significance, i.e., the fuzzy integral method is superior to the conventional method. Otherwise, the null hypothesis is accepted, i.e., we cannot say the fuzzy integral method improves the performance significantly.

Table 5: $t$-test. (degree of freedom = 9, $t_{0.05} = 1.833$, $t_{0.025} = 2.262$, $t_{0.01} = 2.821$. "A" stands for Average method, and "WA" for Weighted Average. "Yes" indicates that the hypothesis is rejected for the task at the associated level of significance.)

| Task | $t$ | Significance Level | | |
|---|---|---|---|---|
| | | 5 % | 2.5 % | 1 % |
| Numerals (A) | −2.908 | Yes | Yes | Yes |
| Numerals (WA) | −2.575 | Yes | Yes | No |
| Uppercase (A) | −2.193 | Yes | No | No |
| Uppercase (WA) | −1.300 | No | No | No |
| Lowercase (A) | −3.806 | Yes | Yes | Yes |
| Lowercase (WA) | −3.361 | Yes | Yes | Yes |

Table 5 shows the results of the test with $n = 10$ for all three tasks. In this comparison, $f_i^b$ is of the fuzzy integral, and $f_i^a$ of the average method as mentioned in the equation (5) or the weighted average method in (6). These methods were chosen for comparisons because they had produced the best results among the conventional methods mentioned in this paper. In this comparison, the degree of freedom is $(n-1) = 9$, and the threshold $t_\alpha$ with $\alpha = 95$, 97.25, and 99% is 1.833, 2.262 and 2.821, respectively. It is seen from table 5 that all the values of $t$, except for the weighted average of the uppercase letter task, are greater than $t_\alpha$ with $\alpha = 95\%$. Therefore, for all the cases except that, "no-improvement" hypothesis is rejected at a 5% level of significance. Similarly, other cases can be tested. This is a strong evidence that the proposed method is superior to the conventional methods.

Table 6 shows the confusion matrices of the network outputs for the numeral recognition task. The performance for the combined outputs is much better than either of the individual networks, leading to a reduction in error rate significantly. We also see a strongly diagonal confusion matrix for the combined output, indicating that complementary nature of the confusions made by the individual networks.

Fig. 5 shows the recognition rates of the presented method with the OWA operators for the three tasks. The results indicate that the performance of the fuzzy integral might be enhanced if we select the $\alpha$ and $\beta$ parameters appropriately.

## 5 Concluding Remarks

This paper has presented a neuro-fuzzy architecture that produces an improved performance on real-world classification problem, especially handwriting character recognition. One of the important advantages of the method is that not only is the classification results combined but that the relative importance of the different networks is also considered. The experimental results for classifying a large set of on-line handwriting characters show that it improves the generalization capability significantly. This indicates that even these straightforward, computationally tractable approach can significantly enhance pattern recognition.

Future efforts will concentrate on refining the feature extraction to capture more information, and testing the efficacy of this fuzzy neural system on larger data sets. The complementary nature of the neural network and the fuzzy logic lead us to believe that a further refined fuzzy neural system will significantly improve the state-of-the-art pattern recognizers, especially in noisy environments.

Table 6: Confusion matrices of the three individual networks and the combined output for the numeral recognition task. Each vertical column is labeled by the target output, and each horizontal row represents the output by the network.

(a) Network 1

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 30 | 0 | 0 | 0 | 1 | 0 | 12 | 2 | 5 | 0 |
| 1 | 0 | 27 | 0 | 2 | 1 | 0 | 4 | 2 | 0 | 1 |
| 2 | 1 | 2 | 38 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 3 | 2 | 3 | 0 | 39 | 0 | 3 | 0 | 1 | 3 | 0 |
| 4 | 0 | 0 | 1 | 0 | 48 | 0 | 1 | 1 | 1 | 2 |
| 5 | 1 | 0 | 0 | 3 | 0 | 42 | 0 | 0 | 2 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 0 | 41 | 0 | 0 | 0 |
| 7 | 1 | 4 | 0 | 7 | 2 | 0 | 0 | 36 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 3 | 2 | 0 | 1 | 61 | 0 |
| 9 | 0 | 2 | 0 | 2 | 0 | 1 | 0 | 4 | 0 | 47 |

(b) Network 2

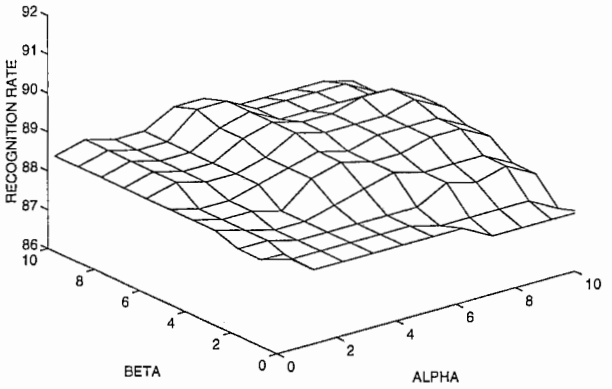
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 24 | 0 | 0 | 0 | 5 | 2 | 6 | 0 | 12 | 1 |
| 1 | 2 | 25 | 0 | 2 | 5 | 0 | 0 | 3 | 0 | 0 |
| 2 | 0 | 0 | 41 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 46 | 0 | 2 | 0 | 0 | 2 | 1 |
| 4 | 0 | 0 | 1 | 1 | 45 | 0 | 1 | 0 | 0 | 6 |
| 5 | 1 | 0 | 0 | 0 | 2 | 37 | 3 | 0 | 4 | 1 |
| 6 | 0 | 0 | 1 | 0 | 0 | 0 | 39 | 0 | 1 | 1 |
| 7 | 0 | 3 | 1 | 5 | 4 | 0 | 0 | 37 | 0 | 1 |
| 8 | 5 | 0 | 0 | 1 | 0 | 1 | 2 | 1 | 58 | 0 |
| 9 | 0 | 1 | 0 | 2 | 0 | 5 | 0 | 1 | 1 | 46 |

(c) Network 3

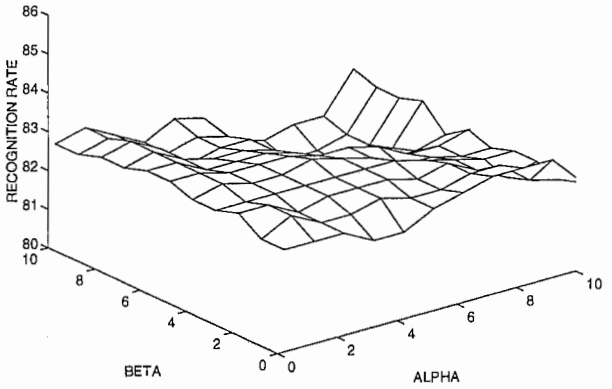
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 24 | 5 | 1 | 0 | 1 | 2 | 11 | 1 | 5 | 0 |
| 1 | 0 | 27 | 0 | 1 | 4 | 0 | 1 | 2 | 2 | 0 |
| 2 | 0 | 2 | 39 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 1 | 2 | 0 | 41 | 1 | 0 | 1 | 0 | 5 | 0 |
| 4 | 1 | 0 | 0 | 0 | 42 | 1 | 1 | 2 | 4 | 3 |
| 5 | 1 | 0 | 0 | 2 | 0 | 44 | 0 | 0 | 1 | 0 |
| 6 | 3 | 1 | 1 | 0 | 2 | 0 | 34 | 0 | 0 | 1 |
| 7 | 0 | 7 | 1 | 6 | 0 | 1 | 1 | 34 | 1 | 0 |
| 8 | 5 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 59 | 0 |
| 9 | 3 | 1 | 0 | 1 | 0 | 1 | 0 | 2 | 7 | 41 |

(d) Combined Output

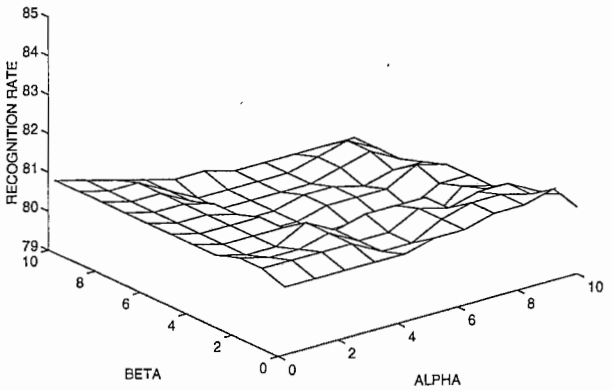
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 32 | 0 | 0 | 0 | 2 | 0 | 10 | 1 | 5 | 0 |
| 1 | 0 | 28 | 0 | 2 | 1 | 0 | 3 | 2 | 0 | 1 |
| 2 | 0 | 1 | 42 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 2 | 4 | 0 | 43 | 0 | 2 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 | 47 | 0 | 1 | 0 | 2 | 3 |
| 5 | 1 | 0 | 0 | 1 | 0 | 44 | 0 | 0 | 2 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 0 | 41 | 0 | 0 | 0 |
| 7 | 1 | 4 | 0 | 6 | 2 | 0 | 0 | 38 | 0 | 0 |
| 8 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 63 | 0 |
| 9 | 0 | 3 | 0 | 3 | 0 | 2 | 0 | 2 | 0 | 46 |




(a) Numerals




(b) Uppercases




(c) Lowercases

Figure 5: The recognition rates with OWA operators.

## Acknowledgements

The author would like to thank Dr. K. Shimohara and Dr. Y. Tohkura at ATR HIP laboratories for continuous encouragement. This work was supported in part by a grant from the Korea Science and Engineering Foundation (KOSEF) and Center for Artificial Intelligence Research (CAIR), the Engineering Research Center (ERC) of Excellence Program.

## References

[1] Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Trans. Patt. Anal. Mach. Inte. **12** (1990) 993–1001.

[2] Scofield, C., Kenton, L., Chang, J.: Multiple neural net architectures for character recognition. Proc. Compcon. (1991) 487–491 (San Francisco, CA, IEEE Computer Society Press).

[3] Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Comp. **3** (1991) 79–87.

[4] Wolpert, D.: Stacked generalization. Neural Net. **5** (1992) 241–259.

[5] Cho, S.-B., Kim, J.H.: Two design strategies of neural network for complex classification problems. Proc. 2nd Int. Conf. Fuzzy Logic & Neural Net. (1992) 759–762.

[6] Perrone, M.P., Cooper, L.N.: When networks disagree: Ensemble methods for hybrid neural networks. Neural Net. for Speech and Image Proc. (1993) (Mammone, R.J. ed., Chapman-Hill, London).

[7] Hampshire II, J.B., Waibel, A.: Connectionist architectures for multi-speaker phoneme recognition. Adv. Neural Info. Proc. Syst **2** (1990) 203–210.

[8] Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Trans. Syst. Man. Cyber. **22** (1992) 688–704.

[9] Benediktsson, J.A., Swain, P.H.: Consensus theoretic classification methods. IEEE Trans. Syst. Man. Cyber. **22** (1992) 418–435.

[10] Cho, S.-B.: Cooperation of modularized neural networks by fuzzy integral with OWA operators. 3rd Int. Conf. Fuzzy Logic, Neural Nets, and Soft Computing (1994) 95–96.

[11] Cho, S.-B.: Neural network ensemble aggregated by fuzzy logic. IEEE/Nagoya University World Wisemen/women Workshop on Fuzzy Logic and Neural Networks/Genetic Algorithms (1994) 46–52.

[12] Cho, S.-B., Kim, J.H.: Combining multiple neural networks by fuzzy integral for robust classification. IEEE Trans. Syst. Man. Cyber. **25** (1995) 380–384.

[13] Richard, M.D., Lippmann, R.P.: Neural network classifiers estimate Bayesian *a posteriori* probabilities. Neural Comp. **3** (1991) 461–483.

[14] Shlien, S.: Multiple binary decision tree classifiers. Patt. Recog. **23** (1990) 757–763.

[15] Sugeno, M.: Fuzzy measures and fuzzy integrals: A survey. Fuzzy Automata Dec. Proc. (1977) 89–102 (Amsterdam: North Holland).

[16] Leszeynski, K., Penczek, P., Grochulskki, W.: Sugeno's fuzzy measures and fuzzy clustering. Fuzzy Sets Syst. **15** (1985) 147–158.

[17] Tahani, H., Keller, J.M.: Information fusion in computer vision using the fuzzy integral. IEEE Trans. Syst. Man. Cyber. **20** (1990) 733–741.

[18] Yager, R.R.: Element selection from a fuzzy subset using the fuzzy integral. IEEE Trans. Syst. Man. Cyber. **23** (1993) 467–477.

[19] Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Trans. Syst. Man. Cyber. **18** (1988) 183–190.

[20] Tappert, C.C., Suen, C.Y., Wakahara, T.: The state of the art in on-line handwriting recognition. IEEE Trans. Patt. Anal. Mach. Intel. **12** (1990) 787–808.

[21] Freeman, H.: Computer processing in line drawing images. Comp. Surv. **6** (1974) 57–98.