Partially Connected Feedforward Neural Networks Structured by Input Types

Article *in* IEEE Transactions on Neural Networks · February 2005

DOI: 10.1109/TNN.2004.839353 · Source: PubMed

2 authors:

Sanggil Kang
Inha University
95 PUBLICATIONS **618** CITATIONS
SEE PROFILE

Can Isik
Syracuse University
84 PUBLICATIONS **729** CITATIONS
SEE PROFILE

Partially Connected Feedforward Neural Networks Structured by Input Types

Sanggil Kang and Can Isik, *Senior Member, IEEE*

***Abstract*—This paper proposes a new method to model partially connected feedforward neural networks (PCFNNs) from the identified input type (IT) which refers to whether each input is coupled with or uncoupled from other inputs in generating output. The identification is done by analyzing input sensitivity changes as amplifying the magnitude of inputs. The sensitivity changes of the uncoupled inputs are not correlated with the variation on any other input, while those of the coupled inputs are correlated with the variation on any one of the coupled inputs. According to the identified ITs, a PCFNN can be structured. Each uncoupled input does not share the neurons in the hidden layer with other inputs in order to contribute to output in an independent manner, while the coupled inputs share the neurons with one another. After deriving the mathematical input sensitivity analysis for each IT, several experiments, as well as a real example (blood pressure (BP) estimation), are described to demonstrate how well our method works.**

***Index Terms*—Fully connected neural network (FCNN), input sensitivity, input type (IT), partially connected neural network (PCNN).**

I. Introduction and Background

NEURAL NETWORKS (NNs) are systems in which computational units analogous to the human brain are interconnected. Feedforward neural networks (FNNs) are the class of NNs whose structure has input, hidden, and output layers. FNNs can be defined as NNs with acyclic connections from the input layer to the output layer.

For input–output mapping (IOM) problems [1], [2] fully connected feedforward neural networks (FCFNNs) (Fig. 1) have been commonly used as a matter of course, since they usually do not need a priori information about data. We refer here to the information about data as being of the input type (IT), which refers to whether an input is coupled or uncoupled with other inputs for generating outputs. For instance, if some inputs are multiplicatively related among them for generating outputs, those can be considered as the coupled inputs in this paper. If an input is additively related with the other inputs, that can be considered as the uncoupled input.

FCFNNs usually work on complicated IOM problems [3] even when data is corrupted; it has been proved that they can be applied to real applications [4]–[6]. Nevertheless, we need to study them further because of their black-box learning style.

Manuscript received December 22, 2002; revised July 8, 2003. This work was supported in part by the Welch–Allyn Company and in part by the CASE Center, Syracuse University.

S. Kang is with Laboratory for Multimedia Computing, Communications, and Broadcasting, Information and Communications University, Daejeon, 305-714, South Korea (e-mail: sang@icu.ac.kr).

C. Isik is with the Electrical Engineering and Computer Science Department, Syracuse University, Syracuse, NY13244 USA (e-mail: cisik@syr.edu).

Digital Object Identifier 10.1109/TNN.2004.839353

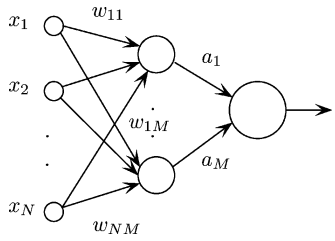


Fig. 1. Example of FCFNNs.

Due to the learning method, several concerns need to be considered when FCFNNs are used for the mapping problems, such as understanding of the input–output relationship, simplification of networks, and generalization.

A. Understanding of the Input–Output Relationship

The black-box style learning can be defined as the method to adjust the internal weights in a network in the sense that the error between the output from the network and the true output (target) is reduced. Usually, the iteration of the adjustment lasts until the error reaches to a predetermined value. During the process of the weight adjustment, people try to vary the number of neurons and layers to construct better performing networks without considering the intrinsic relationship of input and output. Thus, it is difficult to explain how the modeled FCFNNs from the input and output work well for the nonlinear mapping problems. By identifying each IT using the input sensitivity analysis, the problem can be solved to some extent, which will be demonstrated in later sections.

B. Simplification of Networks

Due to the learning style, the structure of the trained FCFNNs usually have unnecessary connections which induces the issue of the complexity of the networks and causes the slow training time, especially for large networks.

The complexity problem has attracted the interest of researchers because of the advantages that would be obtained by solving it. One critical advantage is that the simpler the system, the better it is. One way to reduce the complexity of the networks is to reduce the number of redundant connections, nodes [7]–[11], or input features [12]–[14].

Input feature reduction can be accomplished by eliminating input features with very small input sensitivities as long as the performance of the networks is not affected by doing so.

The reduction of the connections or nodes can be achieved by removing the weights that contribute the least to the network outputs. To our best knowledge, most reduction methods have been done during training networks. For example, Cun *et al.* [7]

introduced a novel method to remove unimportant connections with the least salient weights when generating outputs. Salience is determined by measuring the output deviation of NNs while training them by setting one weight at a time to zero. The smaller the deviation, the less important is a weight in the network. Hassibi *et al.* [8] developed Cun's method by considering second-order derivatives for calculating the salience of weights. They saved computational burden by avoiding repeat training of NNs to measure salience. Duckro *et al.* [9] proposed a statistical estimation of the salience of weights from multiple NNs, using the bootstrap method [15]. They showed that their method works better for test data than [9], using several empirical examples. Karnin [10] introduced a method for connection reduction in which, for each connection, the sensitivity of the error function to a change in the weight update is calculated while training NNs [e.g., fully connected neural network (FCNN)]. After completing the training, he sorted the sensitivities of all the connections in descending order and retrained the NNs without the lowest sensitivity connections [e.g., partially connected neural network (PCNN)]. If there is no significant difference in performance between the FCNN and the PCNN, the lowest sensitivity connections can be pruned. Sietsma *et al.* [11] found that NNs could be pruned by removing noncontributing or slightly contributing neurons for simple pattern recognition problems. For example, if the targets are binary with "0" or "1," the networks is trained for all neurons to have "0" or "1" as an output from those neurons. In this case, the output from a neuron can be identical to that from any other neuron, so neurons may make no contribution to outputs for input patterns. Those redundant and noncontributing neurons can therefore be trimmed. From their experiment, they found that for a pattern recognition problem a pruned NN is more robust to noise to the inputs than the original networks. Two open problems in most of the connection reduction and input feature reduction methods addressed above are: 1) that the input feature reduction methods cannot reduce size of NNs if there are no noisy or redundant inputs and 2) that connection reduction methods have difficulty in understanding the network structure after eliminating unnecessary connections. However, our method first analyzes data and then builds an appropriate structure for an NN according to the result of the analysis, as will be demonstrated in Section II. Therefore, our method can not only reduce network complexity but also understand the input–output relationship. Sometimes the generalization ability can be improved as a by-product.

C. Generalization

Noyes [16] mentioned that the generalization ability is the most valuable feature of NNs, which is the most attractive reason for studying them. Sometime, the generalization ability can not be as good as expected in some examples due to the network complexity or lack of the input–output relationship. As an approach for improving the generalization of networks, PCNNs have been modeled as explained in [16]. The PCNN can be defined as the reduced form of network whose structure is usually modeled by eliminating connections between adjacent nodes in the FCNNs. Elizondo *et al.* [17] addressed the issue

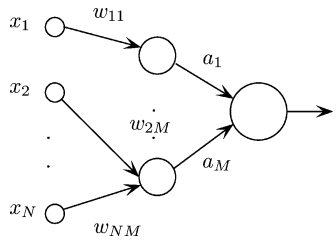


Fig. 2. Example of PCFNNs.

that PCNNs can improve the generalization capabilities with reduced hardware requirements and did a survey of methods for dealing with PCNNs. In [17], three groups of the reduction of the connections are demonstrated, such as ontogenic [18], nonontogenic [19], and hybrid methods. Ontogenic methods reduce connection in network topology while training networks, nonontogenic methods use an analysis of data, and hybrid methods use a combination of both. Usually, PCNNs [20], [21] have been used in special cases in which the input–output relationship is known to some extent. For example, Chung *et al.* [20] obtained a decision tree [22] by using the classifier program called C4.5 [23] in order to structure a PCNN by finding the number of neurons and how the neurons are supposed to be connected within the PCNN. They obtained a better performance when PCNNs rather than FCNNs were used. Wyeth *et al.* [24] used a PCNN for the development to date of an unmanned air vehicle based on a standard size 60 model helicopter. In the network, x-gyro, roll, and y-accelerometer are grouped, y-gyro, pitch, and x-accelerometer are grouped, and z-gyro and z-accelerometer are common to both. They found from a couple of experiments that PCNNs do not degrade training performance, compared with FCNNs.

In this paper, the addressed three concerns are handled by identifying the ITs from data. According to the identified ITs, the structure of networks is chosen. For example, if all inputs are coupled, an FCFNN is a good choice. However, if there is any uncoupled input, a PCFNN can be structured. In PCFNNs, uncoupled inputs do not share neurons in the hidden layer with other inputs while coupled inputs share neurons in the hidden layer with other coupled inputs. For example, x1 is an uncoupled input, while other inputs are coupled inputs (Fig. 2). The ITs are identified by analyzing the input sensitivities as amplifying the magnitude of the inputs.

The remainder of this paper is structured as follows. Section II presents the mathematical derivation of identification of input type (IIT). Section III presents several experiments and a couple of real examples in order to show how well our method of IIT works. Section IV presents the idea that PCFNNs structured according to the identified ITs can lessen the concerns about generalization and simplicity. Section V summarizes conclusions and potential future directions.

II. Identification of Input Types

A. Mathematical Representation

Mathematical representation is done by calculating the difference between the input sensitivities calculated from the FCFNN trained with the original inputs and the original targets

(also called desired outputs), and the input sensitivities calculated from the FCFNN with the amplified inputs by the amount of input amplification (AIA) and the original targets. For convenience, we separately derive the equations of the sensitivity changes of the uncoupled and coupled IOMs. The analysis is mathematically expressed for the multi-input–single-output case, while the multi-input–multi-output (MIMO) case can be straightforward. Before deriving the equations, we need to introduce the notation used in deriving the mathematical expression of the IITs.

Notation:

X	Input vector.
y	Target (true output).
f_i	Individual function of input i.
f_i'	Derivative of f_i.
h	Net function composed of the functions of the coupled inputs.
h'	Derivative of h.
x_i	Input i.
$\hat{y}$	Output from the trained NN with X and y.
W	Weight vector in the trained FCFNN.
g	Net function of the FCFNN trained with original input X and original target y.
Δx_i	AIA of x_i.
$X_{\Delta x_i}$	$= [x_1, x_2, \ldots, x_i + \Delta x_i, \ldots, x_n]$ Changed X in which only input x_i is expanded by Δx_i.
$W_{\Delta x_i}$	Weight vector in the trained NN with $X_{\Delta x_i}$ and the original targets.
$g_{\Delta x_i}$	Net function of the FCFNN trained with $X_{\Delta x_i}$ and the original target y.
$\delta^j_{\Delta x_i}$	Sensitivity change of the input j due to x_i.

a) Uncoupled Input–Output Mapping: Let us have an FCFNN model viewed by

$$\hat{y} = g(X, W) \tag{1}$$

which is the equation of an FCFNN trained with data from the uncoupled function as

$$y = \sum_{i=1}^{k} f_i(x_i) \tag{2}$$

which is an uncoupled IOM equation. The output from the FCFNN can be expressed in terms of the target y and error e such as

$$\hat{y} = y + e. \tag{3}$$

Now, let us define the net function of the FCFNN trained with the changed input vector $X_{\Delta x_i}$ whose element x_i is expanded by the AIA, and the original targets y can be expressed as

$$\hat{y}_{\Delta x_i} = g_{\Delta x_i}(X_{\Delta x_i}, W_{\Delta x_i}). \tag{4}$$

It can be reorganized as

$$\hat{y}_{\Delta x_i} = \hat{y} + g_{\Delta x_i}(X_{\Delta x_i}, W_{\Delta x_i}) - g(X, W). \tag{5}$$

If Δx_i is small, $g_{\Delta x_i}$ will be approximated, using the Taylor series expansion (TSE) [25]

$$\begin{aligned} g_{\Delta x_i}(X_{\Delta x_i}, W_{\Delta x_i}) &\approx g_{\Delta x_i}(X, W_{\Delta x_i} + \Delta x_i g'_{\Delta x_i}(X, W_{\Delta x_i}) \\ &\approx g(X, W) + \Delta x_i g'_{\Delta x_i}(X, W). \end{aligned} \tag{6}$$

In TSE, we considered only the first-order derivative in order to have the simple closed forms of the sensitivity change equation, because we place the emphasis on showing whether there is any sensitivity change by amplifying an input according to IT (refer to the Appendix). We can assume that $g_{\Delta x_i}$ is very close to g, because Δx_i is relatively small, compared to the inputs, and FCFNNs are well trained. In other words, it is assumed that $W_{\Delta x_i} \approx W$; g' means the derivative of g in terms of x_i. Therefore

$$\hat{y}_{\Delta x_i} = \hat{y} + e + \frac{\Delta x_i \partial(y + e)}{\partial x_i} = \hat{y} + \Delta x_i f_i'(x_i). \tag{7}$$

Now, the closed form of the theoretical sensitivity of each amplified input x_i by Δx_i from the FCFNN trained with the amplified input is calculated as

$$\begin{aligned} \frac{\partial \hat{y}_{\Delta x_i}}{\partial x_j} &\approx \frac{\partial \hat{y}}{\partial} \frac{(\Delta x_i f_j'(x_j))}{\partial x_j} \\ &= \begin{cases} \frac{\partial \hat{y}}{\partial x_i} + \Delta x_i' f_i'(x_i) + \Delta x_i f_i''(x_i), & \text{for } i = j \\ \frac{\partial \hat{y}}{\partial x_j}, & \text{for } i \neq j \end{cases}. \end{aligned} \tag{8}$$

The sensitivity of the original input x_i from the FCFNN trained with the original input in (1) is calculated as

$$\frac{\partial \hat{y}}{\partial x_j}, \ \forall j. \tag{9}$$

Therefore, the sensitivity changes due to Δx_i can be obtained by (8) and (9) as below

$$\Delta \delta^j_{\Delta x_i} \approx \begin{cases} \Delta x_i' f_i'(x_i) + \Delta x_i f_i''(x_i), & \text{for } i = j \\ 0, & \text{for } i \neq j \end{cases}. \tag{10}$$

b) Coupled Input–Output Mapping: In a manner similar to that of the uncoupled mapping, the mathematical equations of the sensitivity change for a coupled mapping can be obtained. From (7) we start to derive the equations of the sensitivity change, using the coupled mapping

$$y = h\left(\prod_{i=1}^{k} f_i(x_i)\right) \tag{11}$$

$$\hat{y}_{\Delta x_i} = \hat{y} + \Delta x_i f_i'(x_i) \cdot h'\left(\prod_{j=1}^{k} f_j(x_j)\right) \cdot \prod_{j \neq i}^{k} f_j(x_j). \tag{12}$$

Hence, the closed forms of the sensitivity equation from the FCFNN trained with amplified input x_i due to Δx_i are

$$\begin{aligned}\frac{\partial \hat{y}_{\Delta x_i}}{\partial x_j} \approx & \frac{\partial \hat{y}}{\partial x_j} + \frac{\partial \Delta x_i}{\partial x_j \cdot f_j'(x_j)} \\ & \cdot h'\left(\prod_{m=1}^{k} f_m(x_m)\right) \cdot \prod_{m\neq i}^{k} f_m(x_m) + \Delta x_i f_j''(x_j) \\ & \cdot h'\left(\prod_{m=1}^{k} f_m(x_m)\right) \cdot \prod_{m\neq i}^{k} f_m(x_m) + \Delta x_i f_m'^2(x_m) \\ & \cdot h''\left(\prod_{m=1}^{k} f_m(x_m)\right) \cdot \left(\prod_{m\neq j}^{k} f_m(x_m)\right)^2, \text{ for } j = i. \quad (13)\end{aligned}$$

$$\begin{aligned}\frac{\partial \hat{y}_{\Delta x_i}}{\partial x_j} \approx & \frac{\partial \hat{y}}{\partial x_j} + \Delta x_i f_i'(x_i) \cdot f_j'(x_j) \\ & \cdot h''\left(\prod_{m=1}^{k} f_m(x_m)\right) \cdot \prod_{m\neq i}^{k} f_m(x_m) \\ & + \Delta x_i f_i' f_j' h'\left(\prod_{m=1}^{k} f_m(x_m)\right) \\ & \cdot \prod_{m\neq i,j}^{k} f_m(x_m), \text{ for } j \neq i. \quad (14)\end{aligned}$$

The (13) and (14) can be expressed as

$$\frac{\partial \hat{y}_{\Delta x_i}}{\partial x_j} \approx \frac{\partial \hat{y}}{\partial x_j} + \Delta \delta^j_{\Delta x_i}. \quad (15)$$

Here, $\Delta \delta^j_{\Delta x_i}$ is the rest terms of $\partial \hat{y}_{\Delta x_i}/\partial x_j$ in (13) and (14). Therefore, $\Delta \delta^j_{\Delta x_i}$ can be considered as the sensitivity change due to Δx_i.

From (10) and (15), we can infer that the uncoupled inputs are not correlated with the amplification in the other inputs, but that the coupled inputs are correlated in the change of sensitivity due to the AIA in one of the coupled inputs.

The (10) and (15) can work well when there is no corruption in data (inputs and targets), because sensitivity analysis depends on the accuracy of the trained FCFNNs. However, data can often be corrupted while being extracted or collected in the real world, In which case the FCFNNs cannot be well trained and, thus, the sensitivity analysis will not be as correct as when there is no corruption of data. Nevertheless, our method is still applicable for identifying ITs through the relative comparison (described in Section III) of the changes in sensitivities.

Before doing experiments for identifying ITs, we would like to provide the algorithmic expression of the mathematical derivation to help readers understand the process of the experiment. The algorithmic expression is illustrated as in Fig. 3.

1. Train an NN with the original input and target values
2. Calculate MSE_0(Mean square error from the NN in 1
3. Calculate sensitivities of all inputs and save them
4. **For** each input i
   For AIA $\in$ predefined set of values
   4.1 Add AIA to i
   4.2 Train the NN until $MSE_0 \approx MSE_i$ (Mean square error from the NN at 4.2)
   4.3 Calculate input sensitivities of all inputs and save them
   End
   End
5. Analyze the saved sensitivities

Fig. 3. Algorithmic expression of IIT.

III. Experiments

Some examples (including simulated and real ones) of identifying ITs are discussed below. At the mathematical derivation we assumed that the AIA is very small so that the performances of the two NNs (one trained with the original inputs and targets, and the other trained with the inputs, including an amplified input, and the original targets) are very close. In the experiments, we chose the appropriate AIAs because some simulated functions are insensitive to such small AIAs that we cannot recognize sensitivity changes at first glance. We used 5%, 10%, 15%, 20%, and 25% of the inputs as AIAs, and averaged the sensitivity changes. The averaged absolute sensitivity change of input j due to the amplification of input i is denoted as $|\Delta \bar{\delta}^j_{\Delta x_i}|$ in tables. As shown in Fig. 3, the NNs, including an amplified input and the original targets, were trained until $\text{MSE}_0 + |\alpha| \leq \text{MSE}_i$, with α being a very small number. Here, MSE_0 means the mean square error (MSE) of the NN trained with the original inputs and targets. MSE_i means the MSE of the NN trained with the inputs, including an amplified input, and the original targets. Throughout all of the following examples, the simulated results will be written to the nearest hundredth for values of the input sensitivity.

The conditions of data and training parameters are summarized as follows.

- Uniformly distributed Inputs: three inputs (x_1, x_2, x_3) are generated with a range of [0 1], 1000 patterns for each input.
- Normally distributed random noise is added to an input in order to calculate the input sensitivities in a trained NN.
  Noise was randomly generated with zero mean and standard deviation $= 0.01$.
- Learning method: Levenberg–Marquardt backpropagation.
- $\alpha = 10^{-7}$, where α is a tolerance value in $\text{MSE}_0 + |\alpha| \leq \text{MSE}_i$.

The simulated functions are as follows.

Case I) $y = e^{x_1} + x_2^2 + x_3$, all inputs are uncoupled in generating target y.
Case II) $y = e^{x_1} \times x_2^2 \times x^3$, all inputs are coupled in generating target y.
Case III) $y = e^{x_1} \times x_2^2 + x_3$, inputs x_1 and x_2 are coupled, but x_3 is uncoupled in generating target y.

TABLE I
ABSOLUTE MEAN SENSITIVITY CHANGES FOR THE THREE CASES WHEN DATA ARE NOT CORRUPTED

Case	Amplified input	$\lvert\Delta\bar{\delta}^{1}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{2}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{3}_{\Delta x_i}\rvert$
I	x_1	0.34	0.00	0.00
	x_2	0.00	0.28	0.00
	x_3	0.00	0.00	0.13
II	x_1	0.09	0.10	0.06
	x_2	0.11	0.29	0.18
	x_3	0.06	0.16	0.10
III	x_1	0.15	0.16	0.00
	x_2	0.18	0.43	0.00
	x_3	0.00	0.00	0.12

A. No Corruption in Data

From Table I, we can see that the uncoupled inputs are not affected by the AIAs of other inputs, as in (10), but the coupled inputs are mutually affected by their AIA, as in (15). Therefore, we can clearly identify the types of inputs in all cases.

B. Corruption in Data

When targets are corrupted we cannot identify the ITs as clearly as when they are not corrupted, because FCFNNs are not trained as well as when there is no corruption of data. For instance, the sensitivity change in the uncoupled input by the AIA of the other inputs will not be zero as in (10). In this case, identifying ITs is more challenging. However, we can infer some characteristics of uncoupled and coupled inputs from exhaustive empirical experiments. We summarize the inferred characteristics as below:

1) Uncoupled Inputs: If the sensitivity of an input is not affected by the amplification in another input, and vice versa, these two inputs have a high possibility of being uncoupled.

2) Coupled Inputs:

- *No reflexivity:* If the sensitivities of two inputs are "mutually" affected by the amplifications of the two inputs, we can suspect that these two inputs are coupled. If sensitivities of two inputs are affected in one way, these two inputs are not considered to be coupled.
- *No Transitivity:* If two inputs are coupled to any one input, we cannot assert that those two inputs are coupled.

Using the simulated functions, we corrupted the target y by adding random noise with mean $=$ 0 and $standard\ deviation = \sigma$, which is chosen as the fraction of the mean value of the targets; σ can be converted in the percentile as $\gamma = 100 \times \sigma$. We used γ instead of σ in the following tables.

From Table II we can appropriately identify the types of inputs by analyzing the mean sensitivity changes when targets are corrupted tolerably, such as by 5% and 10% of the mean targets. When targets are corrupted too much (30% in our examples), inputs are not correctly identified as all inputs are coupled, because the input–output relationship deteriorates due to the large corruption in targets.

TABLE II
ABSOLUTE MEAN SENSITIVITY CHANGES FOR THE THREE CASES WHEN TARGETS ARE CORRUPTED

Case	$\gamma(\%)$	Amplified input	$\lvert\Delta\bar{\delta}^{1}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{2}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{3}_{\Delta x_i}\rvert$
I	10	x_1	0.37	0.04	0.04
		x_2	0.03	0.20	0.05
		x_3	0.04	0.02	0.15
	20	x_1	0.33	0.03	0.03
		x_2	0.03	0.25	0.07
		x_3	0.02	0.06	0.15
	30	x_1	0.81	3.10	2.78
		x_2	0.33	3.59	2.91
		x_3	0.42	3.20	3.00
II	10	x_1	0.09	0.10	0.07
		x_2	0.11	0.29	0.19
		x_3	0.07	0.15	0.11
	20	x_1	0.08	0.09	0.06
		x_2	0.10	0.28	0.18
		x_3	0.13	0.15	0.10
	30	x_1	0.08	0.10	0.09
		x_2	0.16	0.28	0.18
		x_3	0.14	0.15	0.17
III	10	x_1	0.15	0.16	0.01
		x_2	0.18	0.51	0.01
		x_3	0.02	0.01	0.14
	20	x_1	0.21	0.17	0.01
		x_2	0.20	0.50	0.01
		x_3	0.05	0.01	0.11
	30	x_1	0.33	0.14	0.18
		x_2	0.35	0.42	0.19
		x_3	0.16	0.16	0.36

C. Application: Blood Pressure Estimation

In this case, we can see that our method can be applicable for real mapping problems such as blood pressure (BP) estimation. Welch–Allyn [26], a medical instrument product company, provided clinical raw data from which inputs and targets for the experiment were extracted. Eleven inputs were extracted from the oscillometric pulse profile [27] for each patient. At this stage, we will not explain the methodology of the data collection because Welch–Allyn does not allow us to disclose its proprietary information in order to protect it from other competitive companies. Also, the methodology is out of the scope of this paper, so it does not hurt the purpose of this experiment. The 11 inputs are denoted as x_i, $i = 1, 2, 11$ with 1476 patterns (patients). These inputs may have been easily corrupted while being collected due to various sources of error such as patient movement, cuff flexibility, numeric approximation. The targets are systolic and diastolic pressures read by two nurses. If there were more than 5 *mmHg* differences between the two nurses' readings, they measured the BP again until their readings agreed within 5 *mmHg*. Still, the targets can be corrupted because of inaccuracy of the nurses' readings… We scaled data (inputs and targets) within 0–1 to help networks converge on training. We used 10% and 20% of the inputs as the AIAs. The $\alpha = 0.000\,005$ is used for the condition of $\text{MSE}_0 + |\alpha| \leq \text{MSE}_i$.

TABLE III
ABSOLUTE MEAN SENSITIVITY CHANGES FOR INPUTS USED IN THE SYSTOLIC ESTIMATION

Amplified input	$\lvert\Delta\bar{\delta}^{1}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{2}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{3}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{4}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{5}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{6}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{7}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{8}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{9}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{10}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{11}_{\Delta x_i}\rvert$
x_1	0.19	0.14	0.09	0.14	0.02	0.12	0.04	0.03	0.00	0.02	0.05
x_2	0.22	0.25	0.05	0.12	0.06	0.04	0.00	0.03	0.01	0.00	0.03
x_3	0.44	0.09	0.24	0.09	0.00	0.02	0.01	0.01	0.06	0.05	0.06
x_4	0.23	0.20	0.02	0.05	0.06	0.03	0.01	0.01	0.01	0.02	0.05
x_5	0.23	0.24	0.09	0.14	0.11	0.03	0.07	0.02	0.02	0.11	0.07
x_6	0.17	0.10	0.04	0.13	0.08	0.13	0.02	0.04	0.01	0.07	0.05
x_7	0.34	0.06	0.00	0.12	0.12	0.02	0.02	0.02	0.05	0.02	0.04
x_8	0.07	0.13	0.01	0.08	0.08	0.01	0.00	0.07	0.02	0.02	0.08
x_9	0.36	0.25	0.06	0.03	0.06	0.02	0.04	0.03	0.05	0.03	0.08
x_{10}	0.02	0.23	0.03	0.05	0.10	0.02	0.01	0.00	0.02	0.05	0.05
x_{11}	0.10	0.13	0.00	0.14	0.11	0.11	0.03	0.01	0.02	0.06	0.01

TABLE IV
ABSOLUTE MEAN SENSITIVITY CHANGES FOR INPUTS USED IN THE DIASTOLIC ESTIMATION

Amplified input	$\lvert\Delta\bar{\delta}^{1}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{2}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{3}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{4}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{5}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{6}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{7}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{8}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{9}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{10}_{\Delta x_i}\rvert$	$\lvert\Delta\bar{\delta}^{11}_{\Delta x_i}\rvert$
x_1	0.20	0.36	0.05	0.14	0.10	0.16	0.05	0.02	0.04	0.03	0.03
x_2	0.08	0.30	0.08	0.19	0.00	0.02	0.04	0.00	0.01	0.09	0.01
x_3	0.13	0.52	0.01	0.38	0.10	0.10	0.05	0.05	0.00	0.01	0.01
x_4	0.30	0.89	0.18	0.96	0.30	0.11	0.00	0.00	0.15	0.17	0.05
x_5	0.19	0.05	0.09	0.20	0.21	0.22	0.06	0.08	0.01	0.06	0.03
x_6	0.19	0.09	0.01	0.24	0.17	0.18	0.05	0.03	0.01	0.03	0.02
x_7	0.12	1.11	0.01	0.00	0.12	0.03	0.09	0.03	0.02	0.04	0.01
x_8	0.04	0.56	0.03	0.05	0.12	0.06	0.08	0.01	0.02	0.01	0.02
x_9	0.09	0.87	0.06	0.05	0.08	0.06	0.07	0.00	0.02	0.01	0.01
x_{10}	0.14	0.04	0.07	0.26	0.08	0.11	0.08	0.00	0.03	0.02	0.01
x_{11}	0.22	0.56	0.13	0.10	0.25	0.05	0.01	0.03	0.04	0.01	0.03

As noted above, both inputs and targets for BP estimation are corrupted, so it is a challenge to identify ITs correctly. Such corruption causes uncoupled inputs to be identified as coupled inputs, since NNs can be poorly trained because of the noisy environment. From exhaustive empirical experiments, we set the criteria for the identification of uncoupled and coupled inputs as follows.

1) Uncoupled inputs have small sensitivity changes when other inputs are amplified. That is, if an input has a relatively small sensitivity change due to the amplification of other inputs, it is considered to be uncoupled.
2) For lower ranked inputs with very small sensitivity changes for their own AIAs, it is difficult to identify types of those inputs. For convenience, those inputs are considered as uncoupled inputs in the experiments.
3) The sensitivities of coupled inputs are "mutually" and "notably" affected by the amplification in any one of them. "Notable" is defined such that the sensitivity change of an input is greater than 50% of the sensitivity change of the expanded input from our empirical experimentation.

1) Systolic BP Estimation: From Table III, the sensitivity changes (written in bold and italic) of inputs x_1, x_2, and x_4 are mutually and notably affected by the AIA of any one of them, and those of x_1 and x_6 (written in bold and italic) also are mutually and notably affected by the AIA of any one of them. Inputs x_1, x_2, and x_4 can be considered to be coupled, and inputs x_1 and x_6 can also be considered to be coupled. Inputs x_3, x_5, x_7, x_8, x_9, x_{10}, and x_{11} can be considered as uncoupled inputs by 1) and the contradiction of 3). The result of the identification of ITs for systolic estimation is summarized as:

- coupled inputs: (x_1, x_2, x_4), (x_1, x_6);
- uncoupled inputs: x_3, x_5, x_7, x_8, x_9, x_{10}, and x_{11}.

2) Diastolic BP Estimation: From Table IV, the sensitivities (which are written in bold and italic) of inputs x_1, x_2, x_5, and x_6 are mutually and notably affected by the AIAs of input x_4; therefore, inputs x_1, x_2, x_5, and x_6 can be considered to be coupled with x_4. Input x_3 is considered as an uncoupled input because the sensitivity change of x_3 due to the AIA of x_3 is very small, as mentioned in 2). Inputs x_7, x_8, x_9, x_{10}, and x_{11} can be considered as uncoupled inputs by 1) and the contradiction of 3). We summarize the results of the identification of ITs for diastolic estimation as follows:

- coupled inputs: (x_1, x_4), (x_2, x_4), (x_4, x_5), (x_4, x_6);
- uncoupled inputs: x_3, x_7, x_8, x_9, x_{10}, and x_{11}.

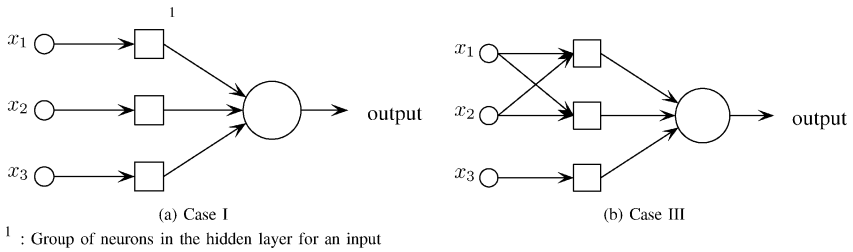


Fig. 4.Structure of PCFNNs Cases (a) I and (b) III.

IV. Empirical Comparison of FCFNNs and PCFNNs for the Three Concerns

According to the ITs identified in Section III for simulated and BP examples, we compare performances for only test data (people are usually not interested in the performance of training data.), learning times, and the number of connections in the networks. The root-mean-square error (RMSE) was used as a measurement of the comparison of the performances.

A. Simulated Examples

For Cases I and III in Section III, we made the training conditions of PCFNNs (Fig. 4) as close to those of FCFNNs as we could. Such conditions include the learning method, number of epochs, number of neurons in the hidden layer, the same initial weights for identical connections between PCFNNs and FCFNNs, etc. For a comparison of performances, we first trained an FCFNN with the predetermined training epoch, and then trained a PCFNN until the MSE of the PCFNN was very close to that of the FCFNN. The MSE of those networks for the test data is listed in Table V.

The conditions of data and training parameters are summarized as follows:

- uniformly distributed Inputs: Three inputs (x_1, x_2, x_3) are generated with a range of [0 1], 2000 patterns (1,000 for training, the rest for test) for each input;
- learning method: Gradient descent with momentum and adaptive backpropagation;
- the number of training epoch: 500;
- minimum value of the targets:
  Case I $= 1.12$;
  Case III $= 0.02$;
- maximum value of the targets:
  Case I $= 4.64$;
  Case III $= 3.33$.

As shown in Table V, for these examples the performances of PCFNNs are not degraded and are even a little bit improved for both "no corruption" and "corruption" in the target data, compared to those of FCFNNs. Also, the number of neurons in the hidden layer is saved by 67% and 44% for Case I and Case III, respectively. These savings reduce the training time (Table V). This example demonstrates that PCFNNs can sometimes improve performance with a simple structure, compared to FCFNNs.

B. BP Estimation

For systolic and diastolic BP estimation, we can structure PCFNNs as shown in Figs. 5 and 6. For systolic estimation, inputs x_1, x_2, and x_4 share neurons in the hidden layer, because they are coupled for generating systolic pressure. Also, inputs x_1 and x_6 share neurons in the hidden layer. The rest of the inputs do not share neurons because they are considered as uncoupled. For diastolic estimation, input x_4 shares neurons with x_1, x_2, x_5, and x_6.

TABLE V
Performances of PCFNNs and FCFNNs for the Simulated Examples

Case	γ (%)	# of neurons in the hidden layer	RMSE of the test data		Saved numbers of neurons by PCFNNs	Saved training time by PCFNNs (sec)
			PCFNNs	FCFNNs		
I	0	9	0.095	0.1	18(67%)	5.2(31%)
		15	0.1	0.12	30(67%)	7.3(26%)
	10	9	0.26	0.3	18(67%)	5.2(31%)
		15	0.25	0.25	30(67%)	7.3(26%)
	20	9	0.52	0.58	18(67%)	5.2(31%)
		15	0.49	0.53	30(67%)	7.3(26%)
	30	9	0.78	0.85	18(67%)	5.2(31%)
		15	0.79	0.81	30(67%)	7.3(26%)
III	0	9	0.001	0.002	12(44%)	2.1(13%)
		15	0.0009	0.001	20(44%)	2.8(11%)
	10	9	0.1	0.17	12(44%)	2.1(13%)
		15	0.1	0.17	20(44%)	2.8(11%)
	20	9	0.2	0.21	12(44%)	2.1(13%)
		15	0.19	0.22	20(44%)	2.8(11%)
	30	9	0.32	0.41	12(44%)	2.1(13%)
		15	0.32	0.37	20(44%)	2.8(11%)

We compared the performances of FCFNNs and PCFNNs for both systolic and diastolic pressure estimation with the same number of neurons in the hidden layer. First, we trained many FCFNNs by changing the initial weights and biases and picked a network that had the best performance for test data of the trained FCFNNs. Second, we trained PCFNNs with the same conditions used for training the best FCFNN until the MSE for the training data was very close to that of the FCFNN, as we changed the initial weights and biases. We listed the root RMSE from FCFNNs and PCFNNs for the test data. The errors from those networks were unscaled with maximum and minimum values of the raw data. The unit of errors is mmHg.

For efficient use of a data set that is not large enough to split a training and test set, we suggest a multiple NN approach where all data will be used for training and testing without crosscontamination. The data is subdivided into five groups. For group

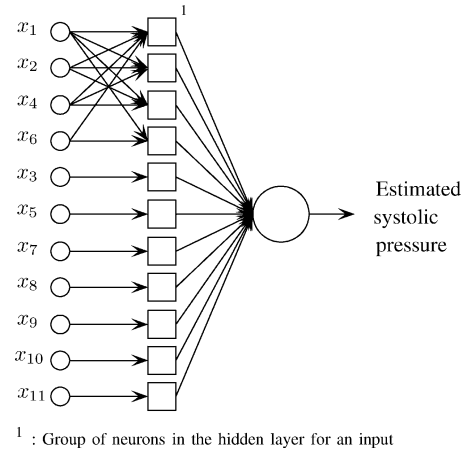


Fig. 5. Structure of PCFNN for the systolic pressure estimation.

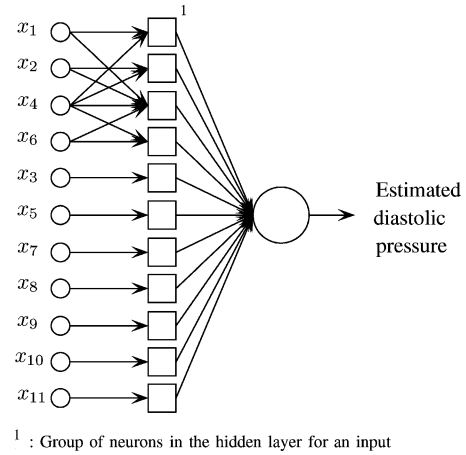


Fig. 6. Structure of PCFNN for the diastolic pressure estimation.

i ($i = 1,2,\ldots,5$), the test data set is obtained by picking the ith patient and every 6th patient from the ith patient up to the last patient. The rest is the training data set. To help readers understand, we viewed the algorithm for splitting test and training data sets as in Fig. 7. Thus, we obtained five different groups of training and test data from 1476 patterns.

For each group, 1181 patterns were used for training NNs and 295 patterns for testing NNs.

For the systolic BP estimation shown in Table VI, the overall performance of the PCFNNs for each data group is better than that of the FCFNNs by 0.68 ($= 5.52 - 4.84$) mmHg, which can be considered as quite an improvement in BP estimation by the experts. We also notice that the PCFNNs save 109 (85%) connections in the hidden layer, which causes the training time to be reduced by 28.98 (46.5%) s as shown in the table, compared to that for FCFNNs. From the results, we can conclude that PCFNNs can successfully replace FCFNNs for systolic BP estimation.

For the diastolic BP estimation shown in Table VII, the overall performance of the PCFNNS is improved by 0.6 ($= 4.76 - 4.16$) mmHg. The PCFNNs also saved 110 connections in the hidden layer, which saved training time by 28.6 s, compared to FCNNs. From the results, we can also conclude that PCFNNs can successfully replace FCFNNs for diastolic BP estimation.

For i = 1 to 5

Test data = (i^{th} patient, and every 6^{th} patient from the i^{th} patient)

Training data = ($\overline{Test\ data}$)

End

Fig. 7. Algorithm for splitting test and training data sets.

TABLE VI
PERFORMANCES OF PCFNNS AND FCFNNS FOR SYSTOLIC PRESSURE ESTIMATION

Group	# of neurons in the hidden layer	RMSE of the test data		Saved numbers of neurons by PCFNNs	Saved training time by PCFNNs (sec)
		PCFNN	FCFNN		
1	12	4.47	5.07	113	29.1(47%)
2	11	4.64	5.08	103	28.8(46%)
3	12	5.36	6.39	113	29.1(47%)
4	12	4.96	5.66	113	29.1(47%)
5	11	4.78	5.43	103	28.8(46%)
Overall		**4.84**	**5.52**	**109(85%)**	**28.98(46.5%)**

TABLE VII
PERFORMANCES OF PCFNNS AND FCFNNS FOR DIASTOLIC PRESSURE ESTIMATION

Group	# of neurons in the hidden layer	RMSE of the test data		Saved numbers of neurons by PCFNNs	Saved training time by PCFNNs (sec)
		PCFNN	FCFNN		
1	12	4.15	4.75	114	28.7(46%)
2	12	4.12	4.52	103	28.7(46%)
3	12	4.18	4.56	114	28.7(46%)
4	12	4.22	5.12	114	28.7(46%)
5	11	4.13	4.82	103	27.95(46.05%)
Overall		**4.16**	**4.76**	**110(85%)**	**28.6(46%)**

V. CONCLUSION AND FUTURE WORK

A. Summary

We have presented here that FCFNNs can be applied to input–output mapping problems by identifying ITs. The identification of IT was done by analyzing the sensitivity changes of the input based on the amplitude of that input. If data (inputs and targets) are not corrupted, the types of inputs are clearly identified, as in (10) and (15). For instance, uncoupled inputs are not affected by the AIA of other inputs, but coupled inputs are mutually affected by their AIA. However, when data are corrupted, the IT cannot be clearly identified. In such case, the criteria for the identification of uncoupled and coupled inputs were stated in Section III.

$$g_{\Delta x_i}(X_{\Delta x_i}, W_{\Delta x_i}) \approx g_{\Delta x_i}(X, W_{\Delta x_i}) + \Delta x_i g'_{\Delta x_i}(X, W_{\Delta x_i}) + \frac{1}{2}\Delta x_i^2 g''_{\Delta x_i}(X, W)$$
$$\approx y + e + \frac{\Delta x_i \partial(y+e)}{\partial x_i} + \frac{1}{2}\frac{\Delta x_i^2 \partial^2(y+e)}{\partial^2 x_i} \approx \hat{y} + \Delta x f_i'(x_i) + \frac{1}{2}\Delta_i^2 f_i''(x_i) \quad \text{(A.16)}$$

$$\frac{\partial \hat{y}_{\Delta x_i}}{\partial x_j} \approx \begin{cases} \frac{\partial \hat{y}}{\partial x_i} + \Delta x_i' f_i'(x_i) + \Delta x_i(1+\Delta x_i') f_i''(x_i) + \frac{1}{2}\Delta x_i f_i'''(x_i), & \text{for } i = j \\ \frac{\partial \hat{y}}{\partial x_j}, & \text{for } i \neq j \end{cases} \quad \text{(A.17)}$$

$$\Delta\delta_{\Delta x_i}^j \approx \begin{cases} \Delta x_i' f_i'(x_i) + \Delta x_i(1+\Delta x_i') f_i''(x_i), & \text{for } i = j \\ 0, & \text{for } i \neq j \end{cases}. \quad \text{(A.18)}$$

B. Contributions and Advantages

To our best knowledge, it has been difficult to obtain *a priori* information concerning data until we proposed our method for identifying ITs, thus, our work may become a foundation stone in identifying ITs and for topics related to it. This is one of the critical contributions of our work.

To our knowledge, PCFNNs have rarely, compared to FCFNNs, been used in mapping problems. From the identified ITs we can build an appropriate NN. For example, if all inputs are coupled, an FCFNN is a good choice. However, if there exists at least one uncoupled input, a PCFNN can successfully replace an FCFNN, with a small number of connections between the input layer and the hidden layer.

Our method of identifying types of inputs has several advantages.

- Understanding the relationship of inputs and outputs. We can understand how inputs are related to one another for generating outputs.
- Solving the complexity problem. From the identified ITs, we can model a PCFNN with small connections and without degrading, and with even better generalization, compared to the FCFNN.
- Saving training time. Since there are fewer connections in an PCFNN, the training time can be reduced as demonstrated in the experiments.
- Mathematical expression. It was proved theoretically and demonstrated with examples that our method is capable of identifying ITs when data are not corrupted.
- Applicable to real problems. We showed by the results of BP estimation that the proposed method can be applied to real IOM problems.

C. Future Work

Despite encouraging results from our methods, further work remains in order to improve them. As we have explained, uncoupled inputs can be affected by amplification of other inputs when both inputs and targets are corrupted, so it is difficult to clearly identify ITs. When two inputs are mutually and notably affected by the amplification in any one of them, those two inputs can be considered to be coupled to each other. It is not easy to determine the percentage of the sensitivity change of the amplified input that should be considered notable as a threshold of coupled inputs. In BP estimation, 50% of the sensitivity change of the amplified input was chosen as the threshold in our experiment. We need further study in order to come up with a more robust technique for deciding the threshold of identifying coupled and uncoupled inputs.

Our methods are restricted to three-layered FNNs even though they resolve many of the practical continuous mappings we need to find a means of applying our methods to NNs with more than three layers.

APPENDIX

Let us see what the impact of the sensitivity changes if we add the second derivative part in the derivation of TSE at (5) and (6) in Section II. For decoupled IOM, see (A.16)–(A.18) at the top of the page.As shown in (A.18), the amplified input only has a sensitivity change when we add the second derivative part, as there is only the first derivative part (6) in Section II). For the coupled IOM, we will have the same result that only one the amplified input but also the coupled inputs with it have sensitivity changes.

REFERENCES

[1] M. Mangeas, A. S. Weigend, and C. Muller, "Forecasting electricity demand using nonlinear mixture of experts," in *Proc. World Congr. Neural Networks*, vol. 2, 1995, pp. 48–53.

[2] D. M. Bates, *Nonlinear Regression Analysis and its Applications*. New York: Wiley, 1998.

[3] D. Cohen and J. Shawe-Taylor, "Feedforward networks-a tutorial," in *Proc. Meeting Neural Computing*, 1989, pp. 1–13.

[4] N. Petkov, "Systolic simulation of multilayer, feed-forward neural networks," *Parallel Process. Neural Syst. Comput.*, pp. 303–306, 1990.

[5] L. Mussone, "A review of feed-forward neural networks in transportation research," *Elektrotechnik und Informationstechnik*, vol. 116, no. 6, pp. 360–365, 1999.

[6] H. Mori, K. Itou, H. Uematsu, and S. Tsuzuki, "An artificial neural net based method for predicting power system voltage harmonics," *IEEE Trans. Power Del.*, vol. 7, no. 1, pp. 402–409, Jan. 1992.

[7] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," *Adv. Neural Inform. Process. Syst.*, vol. 2, pp. 598–605, 1990.

[8] B. Hassibi, D. G. Stork, and C. Wolff, "Optimal brain surgeon: Extensions and performance comparisons," *Adv. Neural Inform. Process. Syst.*, vol. 6, pp. 263–270, 1994.

[9] D. E. Duckro, D. W. Quinn, and S. J. Gardner, "Neural network pruning with Tuckey Kramer multiple comparison procedure," *Neural Computat.*, vol. 14, pp. 1149–1168, 2002.

[10] E. D. Karnin, "A simple procedure for pruning backpropagation trained neural netwowrks," *IEEE Trans. Neural Netw.*, vol. 1, no. 2, pp. 239–242, Jun. 1990.

[11] J. Sietsma and R. J. F. Dow, "Neural net pruning- why and how," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 1, 1988, pp. 325–333.

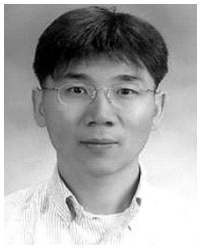
[12] S. Viaene, B. Baesens, G. Dedene, J. Vanthienen, and J. Vandenbulcke, "Sensitivity based pruning of input variables by means of weight cascaded training," in *Proc. PADD*, 2000, pp. 141–159.

[13] A. H. Rambhia, R. Glenny, and J. N. Hwang, "Critical input data channels selection for progressive work exercise test by neural network sensitivity snalysis," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 1999, pp. 1097–1100.

[14] A. H. Sung, "Ranking input importance in neural network modeling of engineering problems," in *Proc. IEEE Int. Joint Conf. Neural Network*, 1998, pp. 316–321.

[15] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*. London, U.K.: Chapman & Hall, 1998.

[16] J. L. Noyes, *B3.5 Handbook of Neural Computation*. London, U.K.: Chapman & Hall, 1997.
[17] D. Elizondo and E. Fiesler, "A survey of partially connected neural networks," *Int. J. Neural Syst.*, vol. 8, no. 5 and 6, pp. 535–558, 1997.
[18] E. Fiesler, "Comparative bibliography of onto-genic neural networks," in *Proc. Int. Conf. Artificial Neural Networks*, 1994, pp. 793–796.
[19] D. Elizondo, E. Fiesler, and J. Korczak, "Non-ontogenic sparse neural networks," in *Proc. Int. Conf. Neural Networks*, 1995, pp. 290–295.
[20] Y. Y. Chung, M. T. Wong, and N. W. Bergmann, "High-speed neural network based classifier for real-time application," in *Proc. ICSP '98*, pp. 506–509.
[21] A. Rajapakse, K. Furuta, and S. Kondo, "Identification of nonlinear dynamic models with particially connected neural networks trained using orthogonal least square estimation," *Trans. Inst. Electr. Eng. Japan*, vol. 3, pp. 335–343, 1999.
[22] I. K. Sethi, "Entropy nets: From decision trees to neural networks," *Proc. IEEE*, vol. 78, no. 3, pp. 1605–1613, Mar. 1990.
[23] J. M. Zurada, A. Malinowski, and U. Shiro, "Perturbation method for deleting redundant input of perceptron networks," *Neurocomput.*, vol. 14, no. 2, pp. 177–193, 1996.
[24] G. F. Wyeth, G. Buskey, and J. Roberts, "Flight control using an artificial neural network," in *Proc. Australian Conf. Robotics and Automation*, 2000, pp. 65–70.
[25] J. Chen and S. M. Kang, "Model-order reduction of weekly nonlinear MEMS devices with Taylor series amplification and Arnoldi process," in *Proc. 43rd IEEE Midwest Symp. Circuits and Systems*, 2000, pp. 248–251.
[26] Welch-Allyn: A Medical Instrument Product Company [Online]. Available: www.welchallyn.com
[27] J. G. Webster, *Medical Instrumentation:Application and Design*, 3rd ed. New York: Wiley, 1993.



Sanggil Kang received the M.S. degree from Columbia University, New York, in 1995 and Ph.D. degree from Syracuse University, Syracuse, NY, in 2002.

He is currently a Research Professor at Information and Communications University, Daejeon, South Korea. His research interests include multimedia systems, signal processing, and neural networks.

Can Isik (S'83–M'85–SM'90) received the B.S. and M.S. degrees in electrical engineering from Middle East Technical University, Ankara, Turkey, in 1980 and 1988, respectively, and the Ph.D. degree in electrical engineering from the University of Florida, Gainesville, in 1985.

He is currently the Senior Associate Dean for Academic Student Affairs in the College of Engineering and Computer Science, Syracuse University, Syracuse, NY. His research interests include neural networks, fuzzy logic, and their applications in modeling, controls, indoor environmental systems, and medical instrumentation.

Dr. Isik is a Member of Eta Kappa Nu, Golden Key, and is listed in *Who is Who in American Education*.