Capabilities of a Four-Layered Feedforward Neural Network: Four Layers Versus Three

Shin'ichi Tamura and Masahiko Tateishi

***Abstract*—Neural-network theorems state that only when there are infinitely many hidden units is a four-layered feedforward neural network equivalent to a three-layered feedforward neural network. In actual applications, however, the use of infinitely many hidden units is impractical. Therefore, studies should focus on the capabilities of a neural network with a finite number of hidden units. In this paper, a proof is given showing that a three-layered feedforward network with $N-1$ hidden units can give any N input-target relations exactly. Based on results of the proof, a four-layered network is constructed and is found to give any N input-target relations with a negligibly small error using only $(N/2)+3$ hidden units. This shows that a four-layered feedforward network is superior to a three-layered feedforward network in terms of the number of parameters needed for the training data.**

***Index Terms*—Feedforward neural networks, mapping capability analysis, multilayer perceptrons, nonlinearities.**

I. Introduction

FEEDFORWARD neural networks are universal approximators. Indeed, a three- or four-layered feedforward neural network (an input layer, one or two hidden layers, and an output layer), which is only allowed connections between adjacent layers, can achieve any continuous mapping using sigmoid-like saturating unit output functions when there are infinitely many hidden units [1]–[3]. Considering this, it is obvious that with infinitely many hidden units a four-layered feedforward neural network is equivalent to a three-layered feedforward neural network. Actual applications, however, restrict use to feedforward neural networks having a finite number of hidden units. Which is more powerful then, a three-layered feedforward neural network or a four-layered feedforward neural network? In other words, if both types of feedforward networks with a finite number of hidden units were to have the same number of parameters, would there be any difference in performance?

The performance of a network can be divided into two main categories, one based on the generalization capabilities of the network and the other based on its mapping capabilities. Villiers and Barnard [4] studied the generalization capabilities of three- and four-layered feedforward networks given the same number of weights for classification tasks. Their conclusion was against the use of four-layered networks in all but the most esoteric applications. At present, however, there seems to be no theoretical basis that justifies these interesting results. Obradovic and Yan [5], in contrast, indicated the superiority of four-layered feedforward networks to three-layered networks in terms of mapping capabilities. They showed that the classification boundaries of four-layered networks, which are at most polynomial in the number of training samples, are strictly more general than those of three-layered networks.

In this paper, we investigate the mapping capabilities of a four-layered feedforward network with a finite number of hidden units for the training of input-target pairs. We obtain results on the number of hidden units a four-layered feedforward network requires to train input-target relations while achieving an arbitrarily small error.

The organization of this paper is as follows. Section II gives a proof stating that a three-layered feedforward neural network with $N-1$ hidden units can give any N input-target relations exactly [6], [7]. Based on results of the proof in Section II, we construct a four-layered feedforward network capable of giving any N input-target relations, and obtain results on the sufficient number of hidden units in Section III. Finally, our conclusion is presented in Section IV.

II. Proof of Three-Layered Neural Network Capabilities

It has already been proven that a three-layered feedforward neural network with $N-1$ hidden units can give any N input-target relations exactly [6], [7]. In this paper, we give another proof of the above fact [8]. The proof focuses on sigmoidal nonlinearity and clarifies the nonlinear deformation mechanism of a three-layered feedforward network with sigmoidal nonlinearity. Shown in Fig. 1 is the network we consider. The network has M linear input units, $N-1$ hidden units, and one linear output unit. Each of the linear units in the input layer just passes its input to form its output. Each hidden unit has a sigmoid function, $1/(1+\exp(-x))$ for its nonlinearity. The linear output unit adds its bias to its input to form its output. The input units are numbered from one to M and the hidden units are numbered from one to $N-1$. Let us define some symbols:

$(\mathbf{x}^{(k)}, t^{(k)})$	kth input-target training pair;
$\mathbf{x}^{(k)}$	kth M-dimensional input vector;
$t^{(k)}$	kth target value $(k=1,2,\cdots,N)$.
$\mathbf{t}$	target vector, $t_k = t^{(k)}\,(k=1,2,\cdots,N)$;
$o_k^{\langle i\rangle}$	ith hidden unit output with $\mathbf{x}^{(k)}$ being the network input, $(i=1,2,\cdots,N-1,\ k=1,2,\cdots,N)$;

Manuscript received August 23, 1994; revised May 31, 1995, March 18, 1996, and July 20, 1996.

The authors are with Research Laboratories, Nippondenso Co., Ltd., Aichi 470-01, Japan.

Publisher Item Identifier S 1045-9227(97)01750-5.

$s(x)$	sigmoid function, $1/(1+\exp(-x))$;
w_{ij}	link weight from jth input unit to ith hidden unit, $(i=1,2,\cdots,N-1,\ j=1,2,\cdots,M)$;
b_i	bias of ith hidden unit;
$\mathbf{W}$	link weight vector from the hidden layer to the output unit,
W_i	link weight from the $(i-1)$th hidden unit to the output unit $(i=2,\cdots,N)$;
W_1	bias value of the output unit.

For the N input-target training pairs, the network must satisfy the following equation:

$$\begin{aligned} O\mathbf{W} &= \mathbf{t}, \quad O_{k,1}=1 \quad (k=1,2,\cdots,N),\\ O_{k,i} &= o_k^{\langle i-1\rangle} \quad (i=2,\cdots,N, k=1,2,3,\cdots,N). \end{aligned} \tag{1}$$

Let us write the matrix O as follows:

$$O = (\mathbf{1}, \mathbf{o}^{\langle 1\rangle}, \mathbf{o}^{\langle 2\rangle}, \cdots, \mathbf{o}^{\langle N-1\rangle}) \tag{2}$$

where $\mathbf{o}^{\langle i\rangle}$ are the ith column vectors and $\mathbf{1}$ is a column vector whose elements are all one. What is to be proved is that O is invertible, or full-rank. We assume that we are working in standard Euclidean space. Let $\mathbf{w}^{\langle i\rangle}$ and (,) be a link weight vector from all the input units to the ith hidden unit and inner product, respectively. The output of the ith hidden unit with $\mathbf{x}^{(k)}$ being the input to the network can be written as follows:

$$o_k^{\langle i\rangle} = s((\mathbf{w}^{\langle i\rangle}, \mathbf{x}^{(k)}) + b_i) \quad (k=1,2,\cdots,N). \tag{3}$$

$o_k^{\langle i\rangle}$ is the kth element of the column vector, $\mathbf{o}^{\langle i\rangle}$, and $(\mathbf{w}^{\langle i\rangle}, \mathbf{x}^{(k)})$ is the input to the ith hidden unit. Let us consider a curve $\mathbf{c}(b_i)$, in Euclidean space R^N

$$\mathbf{c}(b_i) = \mathbf{o}^{\langle i\rangle} = \begin{pmatrix} s(b_i + (\mathbf{w}^{\langle i\rangle}, \mathbf{x}^{(1)}))\\ s(b_i + (\mathbf{w}^{\langle i\rangle}, \mathbf{x}^{(2)}))\\ \vdots\\ s(b_i + (\mathbf{w}^{\langle i\rangle}, \mathbf{x}^{(N)})) \end{pmatrix}$$
$$b_i \in I = [a,b](a<b). \tag{4}$$

For each $i(i=1,2,\cdots,N-1)$ we can assume that $(\mathbf{w}^{\langle i\rangle}, \mathbf{x}^{(k)}) \neq (\mathbf{w}^{\langle i\rangle}, \mathbf{x}^{(k-)})$ for $k \neq k'$. Or to meet this condition, we can easily change the weight vector by perturbing the weight vector.

We show that the curve does not belong to any subspace whose dimension is less than N. Let us suppose that the curve belongs to a subspace of dimension $N-1$. Then there exists a vector $\mathbf{n}$ which is orthogonal to this subspace

$$\begin{aligned} (\mathbf{n}, \mathbf{c}(b_i) - \mathbf{c}(a)) &= n_1 \cdot s(b_i + d_1) + n_2 \cdot s(b_i + d_2)\\ &\quad + \cdots + n_N \cdot s(b_i + d_N) - z = 0\\ d_k &= (\mathbf{w}^{\langle i\rangle}, \mathbf{x}^{(k)}) \quad (k=1,2,\cdots,N)\\ z &= (\mathbf{n}, \mathbf{c}(a)), \quad \text{for } \forall b_i \in I = [a,b](a<b). \end{aligned} \tag{5}$$

Let us assume that n_N is not zero. By some deformation and change of variables of the equation above, we obtain

$$s(b_i) = \sum_{p=1}^{N-1} \alpha_p \cdot s(b_i + e_p) + z' \tag{6}$$

where $\alpha_p = -n_p/n_N$, $z' = z/n_N$, $e_p = d_p - d_N \neq 0$, $e_p \neq e_{p'}$ for $p \neq p'$, $b_i \in I' = [a + d_N,\ b + d_N]$.

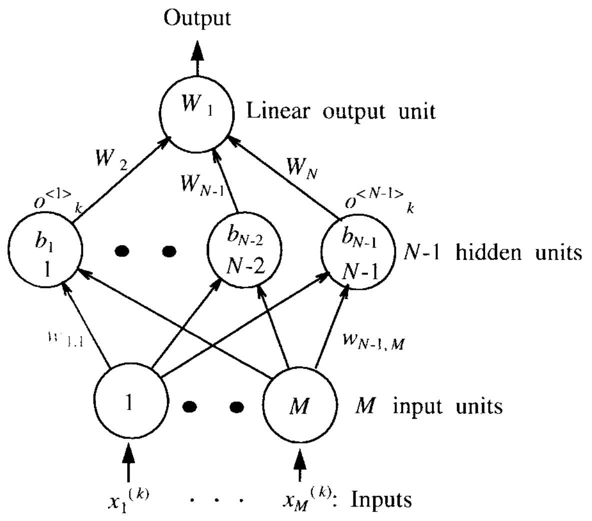

Fig. 1. Three-layered feedforward network.

Interestingly, this formula states that, if we choose α_p and z' appropriately, a sigmoid function can be represented by a weighted sum of $N-1$ shifted sigmoid functions and a bias. The parameters $\alpha_p\,(p=1,2,\cdots,N-1)$ and z' must satisfy the following linear equations for arbitrarily fixed $B \in I'$:

$$\sum_{p=1}^{N-1} \alpha_p \cdot s(B + e_p) + z' = s(B) \tag{7a}$$

$$\sum_{p=1}^{N-1} \alpha_p \cdot s^{(1)}(B + e_p) = s^{(1)}(B) \tag{7b}$$

$$\sum_{p=1}^{N-1} \alpha_p \cdot s^{(2)}(B + e_p) = s^{(2)}(B) \tag{7c}$$

$$\sum_{p=1}^{N-1} \alpha_p \cdot s^{(3)}(B + e_p) = s^{(3)}(B) \tag{7d}$$

$$\vdots$$

where $s^{(n)}(\cdot)$ is the nth derivative of a sigmoid function. Because a sigmoid function is not a polynomial of finite order, these linear equations continue on to infinity. There are, however, only N free parameters: $\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_{N-1}, z'$. This presents a contradiction. Therefore, the curve $\mathbf{c}(b_i)\, b_i \in I$, can not belong to any subspace in R^N if the dimension of the subspace is less than N.

Hence from any interval, $[a,b]$ of R, it is possible to choose N bias values $b_i^{(1)}, b_i^{(2)}, \cdots, b_i^{(N)}$, for the ith hidden unit such that the corresponding vectors $\mathbf{c}(b_i^{(1)})$, $\mathbf{c}(b_i^{(2)}), \cdots, \mathbf{c}(b_i^{(N)})$ span R^N. This means that the column vectors $\mathbf{o}^{\langle 1\rangle}, \mathbf{o}^{\langle 2\rangle}, \cdots, \mathbf{o}^{\langle N-1\rangle}$ can be made linear-independent by tuning the biases of the hidden units properly. Thus it has been shown that O can be made full-rank and, in making O full-rank, the following remarks can be derived.

1) Link weights from the input layer to the hidden layer can almost all be chosen arbitrarily.
2) The bias value of each unit in the hidden layer can be chosen from any interval of R.

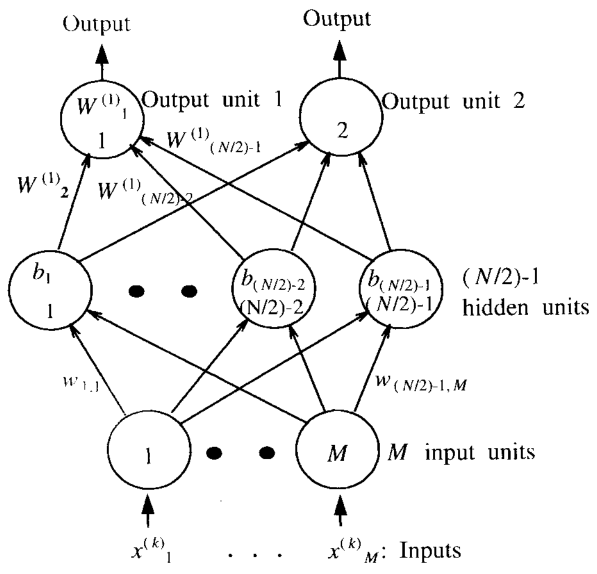

Fig. 2. Three-layered subnetwork.

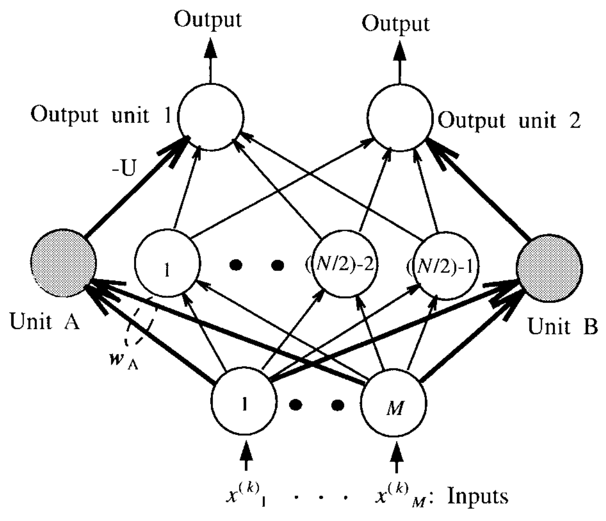

Fig. 3. Units A and B.

III. Construction of a Four-Layered Neural Network

For any N input-target pairs $(\mathbf{x}^{(k)}, t^{(k)})$; $\mathbf{x}^{(k)} \in R^M$, $t^{(k)} \in R$, we will show that, using significantly fewer hidden units than a three-layered network, a four-layered network can give the input-target relations with an arbitrarily small error. For simplicity, we assume that N is even.

Let us first consider the subnetwork shown in Fig. 2 and the following subtargets:

$$t'^{(k)} = t^{(k)}/C + 0.5 \in (0,1) \quad (k = 1,2,\cdots,N) \tag{8}$$

where C is a positive constant and is determined so that $t'^{(k)} \in (0,1)$.

The subnetwork has M linear input units, $(N/2)-1$ hidden units and two output units. The output units, in this case, are nonlinear units (i.e., each output unit has a sigmoid function). The output units are numbered one and two. The link weights between the input layer and the hidden layer are set using random numbers.

It is obvious that the finite N input vectors can be separated into two groups consisting of $N/2$ input vectors with a hyperplane in the input space. Let us denote these two input vector groups as V_1 and V_2 and assume that the index, k, for the subtargets is reindexed so that $t'^{(k)} (k = 1,2,\cdots,N/2) \in V_1$ and $t'^{(k)} (k = (N/2)+1,\cdots,N) \in V_2$. As described in Section II, by adjusting the bias values of the hidden units in the subnetwork, it is possible for an $N/2 \times N/2$ matrix, O, formed by the hidden layer outputs with all of the vectors of V_1 being inputs to the subnetwork, to be made full-rank. Thus, one of the output units, for example, unit 1, can assign each input vector of V_1 to the corresponding subtarget value. The bias value of output unit 1, and the link weights between the hidden layer and output unit 1 are determined by solving the following equation:

$$O\mathbf{W}^{(1)} = \mathbf{u}, \quad u_k = s^{-1}(t'^{(k)}) \quad (k = 1,2,\cdots,N/2) \tag{9}$$

where $s^{-1}(\cdot)$ is the inverse function of a sigmoid function, $W^{(1)}_1$ is the bias value of output unit 1, and $W^{(1)}_i (i = 2,3,\cdots,N/2)$ are the link weights from the $(i-1)$th hidden unit to output unit 1.

Let us consider the V_2 inputs. Since the bias values of the hidden units are adjusted for V_1 inputs, there is no guarantee that another $N/2 \times N/2$ matrix, O', which is formed by the hidden layer outputs with all of the vectors of V_2 being inputs to the subnetwork, will be full-rank, or invertible.

Let us write the bias values tuned for the V_1 inputs as follows:

$$\mathbf{B} = (b_1, b_2, \cdots, b_{(N/2)-1})^t \quad (t\text{: transpose}) \tag{10}$$

and assume that O' is not full-rank. As described in Section II, in order to make O' full-rank, each bias value of the hidden units should be chosen from any interval of R. Hence, for an arbitrarily small $e > 0$, we can choose $\mathbf{B}'$, which makes O' full-rank

$$\mathbf{B}' = (b'_1, b'_2, \cdots, b'_{(N/2)-1})^t, \quad b'_i \in [b_i - e, b_i + e] \quad (i = 1,2,\cdots,(N/2)-1). \tag{11}$$

Since the determinant of O is a continuous function of the bias values and is not zero at $\mathbf{B}$, if we set e sufficiently small, the determinant of O at $\mathbf{B}'$ does not become zero. Thus, it has been shown that we can always choose bias values that make both O and O' full-rank at the same time. Hence, output unit 2 can also assign each input vector of V_2 to the corresponding subtarget value. The bias value of output unit 2 and the link weights between the hidden layer and output unit 2 are determined in the same way. At this stage, output units 1 and 2 produce incorrect outputs for the V_2 inputs and the V_1 inputs, respectively.

Now let us add two hidden units to this subnetwork and denote these units as unit A and unit B, which are shown in Fig. 3. Like the other hidden units, these units are fully connected from the input units. However, unit A is connected only to output unit 1 and unit B is connected only to output unit 2.

Let us consider unit A. By setting the link weight vector, $\mathbf{w}_A$, which is the link weights from the input layer to unit A, to be equal to $T\mathbf{n}$, where $\mathbf{n}$ is a normal vector of the hyperplane described above and T is a large positive constant, and by adjusting the bias value of Unit A so that outputs of unit A become 0.5 with vectors on the hyperplane being inputs to the subnetwork (Fig. 4), unit A can be made a detector for V_1 and V_2 (i.e., unit A produces almost zero and almost one for the V_1 inputs and the V_2 inputs, respectively). Here the direction of the normal vector should be interpreted appropriately. Setting the link weight from unit A to output unit 1 to be $-U$, where U is also a large positive constant, output unit 1 receives from unit A:

1) small negative values, $-e_i\,(e_i > 0)$ for the V_1 inputs $(i = 1, 2, \cdots, N/2)$;
2) large negative values, $-E_i\,(E_i > 0)$ for the V_2 inputs $(i = (N/2) + 1, (N/2) + 2, \cdots, N)$.

We can make E_i and e_i arbitrarily large and small, respectively, by setting T and U sufficiently large. E_i goes to infinity and e_i goes to zero. Thus, for an input vector of V_2, unit A is able to send an arbitrarily large negative value to output unit 1, and all the other hidden units send fixed constant values. Since output unit 1 has a sigmoid function, the outputs of output unit 1 can be arbitrarily close to zero for the V_2 inputs. However, for the V_1 inputs, since unit A is able to send a value that is arbitrarily close to zero to output unit 1, the outputs of output unit 1 can be arbitrarily close to the corresponding subtarget values. In the same way, unit B can be tuned for output unit 2. We have thus far constructed a network whose performance is as follows.

1) For the V_1 inputs:
   The outputs of output unit 1 are arbitrarily close to the corresponding subtarget values and the outputs of output unit 2 are arbitrarily close to zero.
2) For the V_2 inputs:
   The outputs of output unit 1 are arbitrarily close to zero and the outputs of output unit 2 are arbitrarily close to the corresponding subtarget values.

To complete the construction, let us add one more linear unit, i.e., the final output unit. The complete four-layered neural network is shown in Fig. 5. This output unit is fully connected from output units 1 and 2 of the subnetwork. The link weights from output units 1 and 2 to the output unit are set to C of (8), and the bias value of the output unit is set to $0.5C$ in order to restore the original target values. Accordingly, the four-layered neural network can produce almost the same corresponding target value for each of the N input vectors. The error between the target value and the actual network output can be made arbitrarily small. Hence, it has been shown that a four-layered feedforward neural network with $(N/2) + 3$ hidden units can give any N input-target relations with an arbitrarily small error.

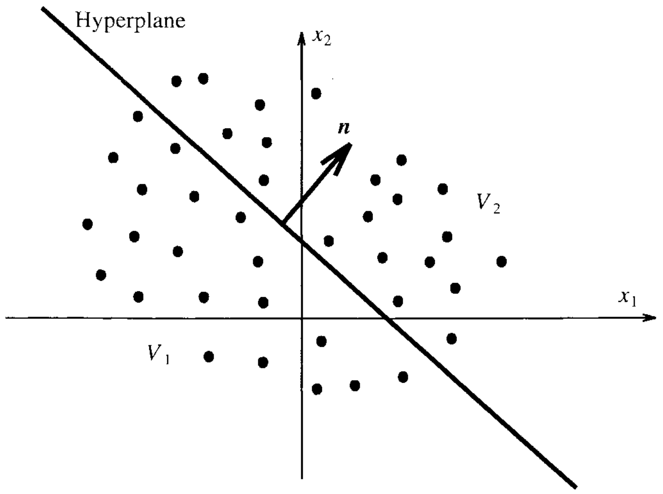

Fig. 4. Input space and the separating hyperplane (two-dimensional case).

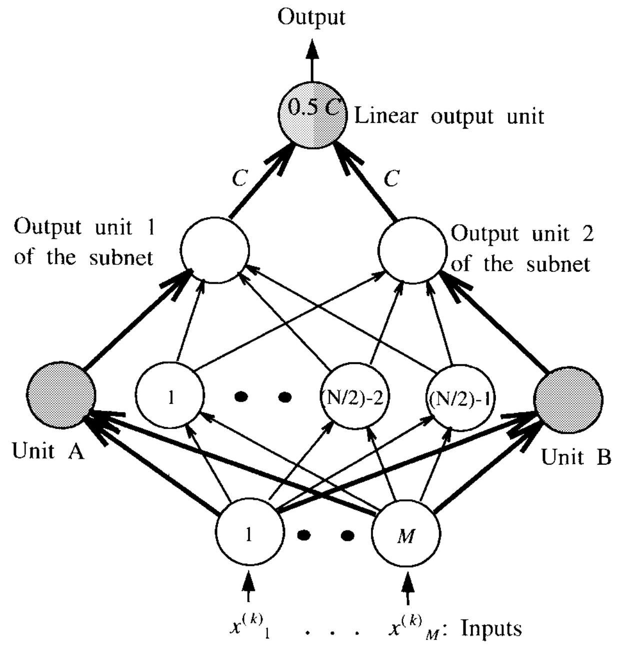
Output
0.5 C
Linear output unit
C
C
Output unit 1 of the subnet
Output unit 2 of the subnet
1
(N/2)-2
(N/2)-1
Unit A
Unit B
1
M
$x^{(k)}_1$ . . . $x^{(k)}_M$: Inputs

Fig. 5. Complete four-layered feedforward network.

IV. Conclusion

We have given another proof of the fact that a three-layered feedforward neural network with $N - 1$ hidden units can give any N input-target relations exactly. Based on results of the proof, we have constructed a four-layered feedforward neural network with $(N/2) + 3$ hidden units and shown that such a network can give any N input-target relations with an arbitrarily small error. This clearly shows the difference in capabilities between a four-layered feedforward neural network and a three-layered feedforward neural network and indicates the superiority of the four-layered network for the training of input-target pairs.

Acknowledgment

The authors would like to thank T. Okabe, Executive Managing Director of the Laboratories, T. Hattori, Associate

Director, H. Kurahashi, and M. Minami for their constant help and encouragement. Thanks are also due to the reviewers for their insightful comments and suggestions.

References

[1] K. Funahashi, "On the approximate realization of continuous mapping by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[2] G. Cybenko, "Continuous valued neural networks with two hidden layers are sufficient," *Math. Contr., Signals, Syst.*, vol. 2, pp. 303–314, 1989.

[3] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[4] J. de Villiers and E. Barnard, "Backpropagation neural nets with one and two hidden layers," *IEEE Trans. Neural Networks*, vol. 4, pp. 136–141, Jan. 1992.

[5] Z. Obradovic and P. Yan, "Small depth polynomial size neural networks," *Neural Computa.*, vol. 2, pp. 402–404, Winter 1990.

[6] M. A. Sartori *et al.*, "A simple method to derive bounds on the size and to train multilayer neural networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 467–471, July 1991.

[7] S. C. Huang and Y. F. Huang, "Bounds on number of hidden neurons in multilayer perceptrons," *IEEE Trans. Neural Networks*, vol. 2, pp. 47–55, Jan. 1991.

[8] S. Tamura, "Capabilities of a three-layer feedforward neural network," in *Proc. IJCNN91*, Nov. 1991, pp. 2757–2762.

Shin'ichi Tamura received the B.E. degree in mechanical engineering from the University of Tokyo, Japan, in 1979.

From 1986 to 1990 he was a Senior Researcher of ATR Interpreting Telephony Research Laboratories. Since 1992 he has been with Nippondenso Co., Ltd. His research interests include speech recognition, speech signal processing, and neural networks.

Masahiko Tateishi received the B.E. degree in electrical engineering from Nagoya University, Japan, in 1984.

From 1984 to 1988 he engaged in the development of the operating system for Sequential Inference Machine at the Institute for New Generation Computer Technology (ICOT). Since 1991 he has been with Nippondenso Co., Ltd. His current research interest is speech processing using recurrent neural networks.