Identification of Time Series Signals Using Dynamical Neural Network with GA-based Training

Kunihiko Nakazono
University of the Ryukyus
Senbaru 1, Nishihara, Okinawa. 903–0213
nakazono@tec.u-ryukyu.ac.jp

Kouhei Ohnishi
Keio University
Hiyoshi 3–14–1, Yokohama. 222–8522
ohnishi@sd.keio.ac.jp

Hiroshi Kinjo
University of the Ryukyus
Senbaru 1, Nishihara, Okinawa. 903–0213
kinjo@tec.u-ryukyu.ac.jp

Abstract

In this paper, we propose a dynamical neural network (DNN) having the properties of inertia, viscosity, and stiffness and its training algorithm based on a genetic algorithm (GA). In a previous study, we proposed a modified training algorithm for the DNN based on error backpropagation method. However, in the former method it was necessary to determine the values of the DNN parameters by trial and error. In the proposed DNN, the GA is designed to train not only the connecting weights but also the parameters of the DNN. Simulation results show that the DNN trained by GA obtains good training performance for time series patterns generated from unknown system.

1 Introduction

Recently, recurrent neural networks and spiking neural networks have attracted more research interest than layered neural networks having static mapping capability [1, 2, 3, 4]. The recurrent neural network is a possible candidate for improving the system dynamics because it incorporates a feedback structure in the neuron unit and takes time delayed inputs into consideration. Research on spiking neural networks is also ongoing. Spiking neural networks treat spike trains and process the signals based on spike pulses. However, the network structure in recurrent neural networks and spiking neural networks is complex compared to that in layered neural networks with a training algorithm.

Here, we propose a dynamical neural network (DNN) that realizes a dynamical property and has a network structure with the properties of inertia, viscosity, and stiffness without time delayed input elements. In a previous study, the proposed DNN was constructed with a training algorithm that used error backpropagation method [5]. However, that algorithm modified only the connecting weights and the property parameters for the DNN had to be determined by trial and error. We design a GA-based training [6] both the connecting weights and the parameters of the DNN.

The validity of the proposed DNN was verified by identifying periodic functions such as a simple one-period sine waveform and several periodic sine waveforms [7]. In this paper, it is verified by identifying the time series signals of linear system and nonlinear system. Simulation results show that the proposed DNN provides higher performance than the conventional neural network.

2 Structure of DNN

In this paper, a DNN is configured using a neuron having the properties of inertia, viscosity, and stiffness. In this neuron model, we assume the image output from neuron possesses the properties of inertia, viscosity, and stiffness, and that the output is propagated in the next neuron. The proposed DNN is composed of three hierarchy layers and the proposed neuron adopts a hidden layer and an output layer. The structure of the DNN is shown in Figure 1.

The equations for the DNN are expressed as follows.

$$y_i = u_i, \quad (i = 1, 2, \cdots, N_I) \tag{1}$$

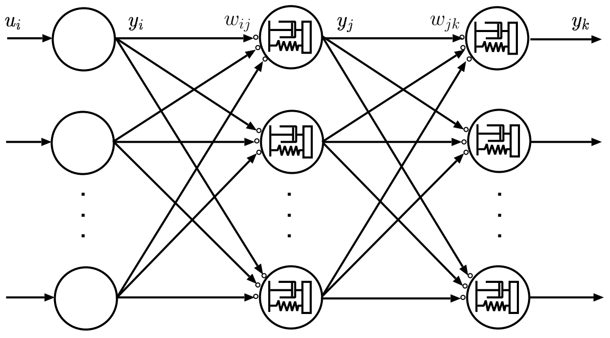

Figure 1: Structure of DNN

$$y_j = K_j f_j(net_j) + D_j \dot{f}_j(net_j) + M_j \ddot{f}_j(net_j) \tag{2}$$

$$net_j = \sum_{i=1}^{N_I} w_{ij} y_i, \quad (j = 1, 2, \cdots, N_J) \tag{3}$$

$$y_k = K_k f_k(net_k) + D_k \dot{f}_k(net_k) + M_k \ddot{f}_k(net_k) \tag{4}$$

$$net_k = \sum_{j=1}^{N_J} w_{jk} y_j, \quad (k = 1, 2, \cdots, N_K) \tag{5}$$

Here, u_i shows input value to the DNN, and y_i, y_j, and y_k show output values in input, hidden, and output layers, respectively. The Connecting weight from unit i in input layer to unit j in hidden layer is denoted by w_{ij}. Similarly, w_{jk} is a connecting weight from unit j in hidden layer to unit k in output layer. The total sum of products of the connecting weight and the output value is denoted by net. M_j, D_j, and K_j are the property parameters of inertia, viscosity, and stiffness, respectively. N_I, N_J, and N_K are the number of neurons in input, hidden, and output layers, respectively. The threshold function uses a sigmoid function in the range of $[-1, 1]$.

3 Training algorithm based on GA

The DNN is trained using a GA in an off-line process. Figure 2 shows the flowchart of the evolution process in the DNN. The evolution algorithm for the DNN is as follows.

STEP1: Produce the initial DNNs at random. The connecting weights w_{ij} and w_{jk} in the range of $[-1, 1]$ and the property parameters M_j, D_j, K_j, M_k, D_k, and K_k of DNNs in the range of $[0, 10]$

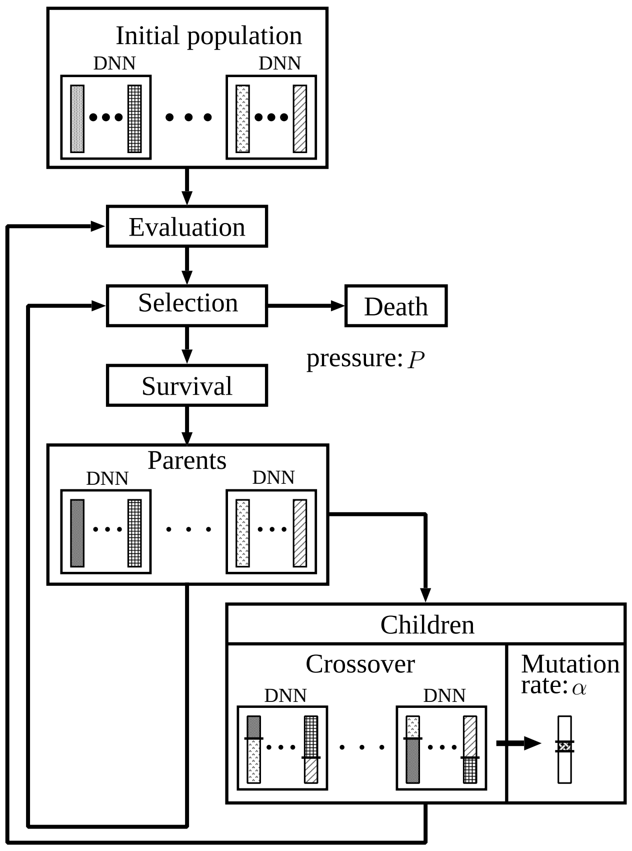

Figure 2: Flowchart of GA-based training

are transformed to the chromosome. The genetic code is transformed to the binary code (16 bit).

STEP2: Sum all of the fitnesses for the DNNs.

STEP3: Select the parent DNNs by means of roulette wheel parent selection.

STEP4: Perform a crossover operation for the chromosome to produce new DNNs.

STEP5: Perform a mutation operation for some additional new DNNs.

STEP6: Sum all of the fitnesses for the DNNs including the new DNNs. Go back to STEP2 until the evolution process arrives at generation 10,000.

Further information regarding the parameters of the GA is shown in Table 1.

During the GA-based training process, an error function E is used to evaluate the performance of each

Table 1: Simulation parameters of GA

Initial DNNs	400 individuals
Selection	Roulette wheel parent selection $P = 0.6$
Crossover	One-point crossover
Mutation	Bit mutation $\alpha = 0.10$
Final generation	10,000

DNN. The error function E is described by the following equation as

$$E = \frac{1}{2}\sum_k e(t)^2 = \frac{1}{2}\sum_k (d(t) - y(t))^2 \qquad (6)$$

where $d(t)$ is the desired signal. The fitness of DNN is expressed in terms of the inverse of the error function E. The connecting weights and property parameters of the DNN are modified in order to maximize the fitness function determined by the error function in Equation (6).

4 Numerical simulation

The effectiveness of the proposed DNN is verified by numerical simulation in order to identify time series signal. The method by which a time series signal from an unknown system can be identified is shown in Figure 3. The DNN is structured to have a single input

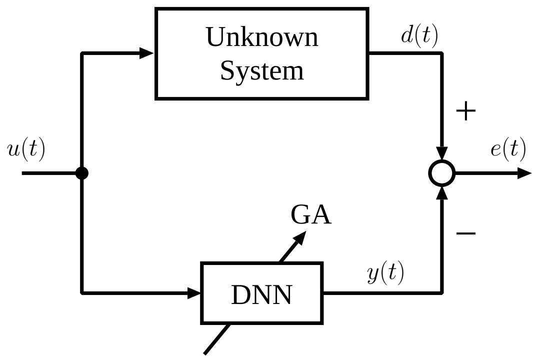

Figure 3: System identification of time series signal

and single output (SISO). The input signal $u(t)$ of the DNN and the unknown system is a random number of normal distribution (mean 0.5 and variance 0.5). The desired signal, namely the training data $d(t)$, is the output signal of the unknown system.

In numerical simulation, the validity of the proposed DNN is verified by identifying the unknown system such as linear system expressed in Equation (7) and nonlinear system expressed in Equation (8).

- Simulation 1 (linear system)

$$d(t) = 0.1d(t-1) + 0.2d(t-2) + 0.3u(t) + 0.4u(t-1) \quad (7)$$

- Simulation 2 (nonlinear system)

$$d(t) = 0.1d(t-1) + 0.2d(t-2) + \sin\left(\frac{\pi u(t)}{4}\right) \quad (8)$$

4.1 Evolution process

In GA simulation, we set the GA parameters shown in Table 1 and the number of neurons in hidden layer is 12 units. The input signal $u(t)$ uses 1,000 sampling data. In simulation 1 and simulation 2, the evolution processes of the best DNN with the GA-based training are shown in Figure 4.

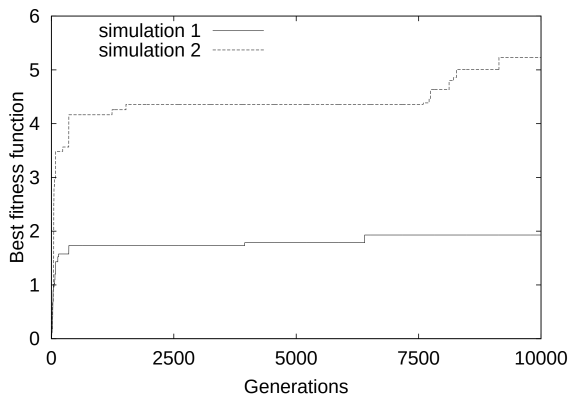

Figure 4: Evolution process

It is observed that either of the evolution processes provide performance to some extent. The fitness values increase gradually and the evolutions almost stagnate at generation 7,000 in simulation 1 and at generation 9,000 in simulation 2, respectively.

4.2 Simulation 1 (linear system)

The result of the regenerating signal using the trained DNN is shown in Figure 5. The figure shows the regenerating signal in range of $[100, 200]$.

The output of the DNN deviated negligibly from the desired signal, but the DNN could not cope with a quick transition.

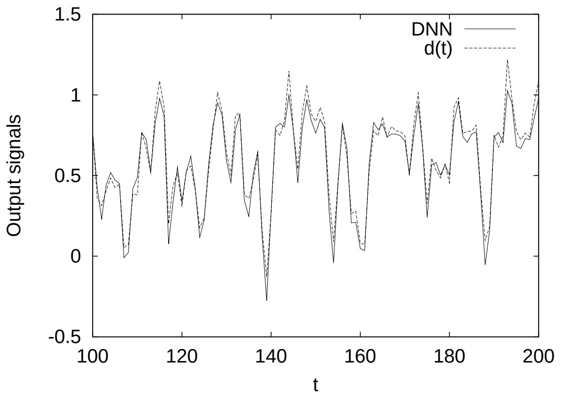

Figure 5: Regenerated result (simulation 1)

4.3 Simulation 2 (nonlinear system)

The result of the regenerating signal using the trained DNN is shown in Figure 6. The figure shows the regenerating signals in range of $[100, 200]$.

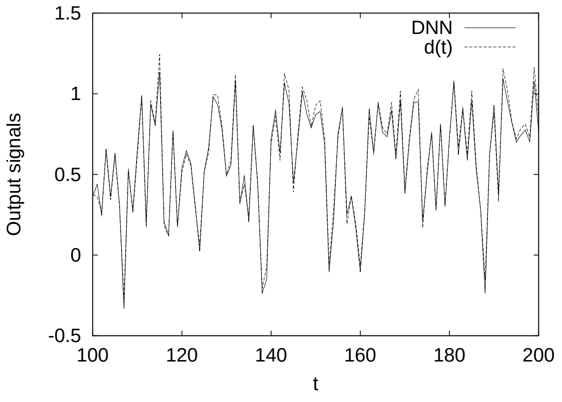

Figure 6: Regenerated result (simulation 2)

The output $y(t)$ of the best DNN deviated negligibly from the desired signal $d(t)$. The DNN trained by GA obtained good training performance for time series signal generated from the output of the nonlinear system.

5 Conclusion

In this paper, the proposed DNN, exhibiting the effectiveness of dynamical neuron with properties of inertia, viscosity, and stiffness, was configured. The training algorithm adopt the GA-based training method. Simulation results showed that the DNN trained by the GA realized good training performance for time series signals generated from either of unknown systems with linearity and nonlinearity.

In future work, we will try to identify a unknown system with strong nonlinearity.

References

[1] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group (1989), Parallel Distributed Processing, *The MIT Press.*

[2] R. J. Williams and D. Zipser (1989), A Learning Algorithm for Continually Running Fully Recurrent Neural Networks, *Neural Computation*, 1, No.2, pp.270–280.

[3] H. Kinjo, K. Nakazono, and T. Yamamoto (1997), Pattern Recognition for Time Series Signals Using Recurrent Neural Networks by Genetic Algorithms (in Japanese), *Trans. of ISCIE*, Vol. 10, No. 6, pp.304–314.

[4] W. Mass and C. M. Bishop (1999), Pulsed Neural Networks, pp. 16-53, *MIT Press.*

[5] K. Nakazono, K. Ohnishi, H. Kinjo, and T. Yamamoto (2003), Identification of Periodic Function Using Dynamical Neural Network, *AROB 8th '03*, Vol.2, pp. 633–636.

[6] Edited by L. Davis (1991), Handbook of Genetic Algorithms, *Van Nostrand Reinhold.*

[7] K. Nakazono, K. Ohnishi, and H. Kinjo (2004), System Identification Using Dynamical Neural Network with GA-based Training, *AROB 9th '04*, Vol.1, pp. 75–78.