2003 International Symposium on Advanced Intelligent Systems
September 25-28, 2003, Jeju, Korea

# Sigma-Pi$_t$ Cascaded Hybrid Neural Network and its Application to the Spirals and Sonar Pattern Classification Problems

Eduardo Masato Iyoda, Hajime Nobuhara, Kazuhiko Kawamoto, Shin'ichi Yoshida, and Kaoru Hirota
Department of Computational Intelligence and Systems Science
Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology
4259 Nagatsuta, Midori-ku, Yokohama 226-8502, Japan
Email: {iyoda, nobuhara, kawa, shin, hirota}@hrt.dis.titech.ac.jp

***Abstract*— A cascade structured neural network called Sigma-Pi$_t$ Cascaded Hybrid Neural Network ($\sigma\pi_t$-CHNN) is proposed. It is an extended version of the Sigma-Pi Cascaded extended Hybrid Neural Network ($\sigma\pi$-CHNN), where the classical multiplicative neuron ($\pi$-neuron) is replaced by the translated multiplicative ($\pi_t$-neuron) model. The learning algorithm of $\sigma\pi_t$-CHNN is composed of an evolutionary programming method, responsible for determining the network architecture, and of a Levenberg-Marquadt algorithm, responsible for tuning the weights of the network. The $\sigma\pi_t$-CHNN is evaluated in 2 pattern classification problems: the 2 spirals and the sonar problems. In the 2 spirals problem, $\sigma\pi_t$-CHNN can generate neural networks with 10% less hidden neurons than that in previous neural models. In the sonar problem, $\sigma\pi_t$-CHNN can find the optimal solution for the problem, i.e., a network with no hidden neurons. These results confirm the expanded information processing capabilities of $\sigma\pi_t$-CHNN, when compared to previous neural network models.**

## I. Introduction

Sigma-Pi Cascaded extended Hybrid Neural Network ($\sigma\pi$-CHNN) [1] is the evolutionary neural architecture that can employ both additive and multiplicative compositions, and different activation functions in each neuron of the network. Although $\sigma\pi$-CHNN has been shown to perform well in nonlinear regression problems, it uses a conventional multiplicative neuron ($\pi$-neuron) [2] as one of its components. It has been shown that the $\pi$-neuron has some disadvantages that may possibly limit its applicability in complex problems [3]. Furthermore, $\sigma\pi$-CHNN uses a standard genetic algorithm to search for neural architectures, which causes the learning algorithm to perform slowly.

To overcome these 2 possibly limiting properties of $\sigma\pi$-CHNN, an extended architecture called Sigma-Pi$_t$ Cascaded Hybrid Neural Network ($\sigma\pi_t$-CHNN) is proposed. Instead of using $\pi$-neuron, $\sigma\pi_t$-CHNN uses the translated multiplicative neuron ($\pi_t$-neuron) [3] model, which has been shown to perform better than the $\pi$-neuron, thus increasing the computational power of the model. Moreover, to speed up the learning procedure, $\sigma\pi_t$-CHNN employs a modified version of EPNet [4], an evolutionary algorithm specially developed to optimize neural networks. The weights of the network are adjusted by the Optimized Levenberg-Marquadt with Adaptive Momentum (OLMAM) algorithm [5].

The proposed architecture is evaluated in 2 pattern classification problems: the 2 spirals and the sonar problems. In the spirals problem, the results show that the $\sigma\pi_t$-CHNN model can achieve better performance in terms of computational cost, reducing the number of hidden neurons in the order of 10%, compared to that in former neural architectures. In the sonar problem, $\sigma\pi_t$-CHNN is able to find the global optimum, i.e., an architecture with no hidden neurons.

In section II, the $\sigma\pi_t$-CHNN architecture is presented in detail. Section III describes the learning algorithm of $\sigma\pi_t$-CHNN. Section IV shows the results obtained in the spirals and sonar problems.

## II. Sigma-Pi$_t$ Cascaded Hybrid Neural Network ($\sigma\pi_t$-CHNN)

Sigma-Pi Cascaded extended Hybrid Neural Network ($\sigma\pi$-CHNN) [1] model consists in architecture where the neurons are organized in a cascaded structure, i.e., each neuron can be connected to all of its precedent nodes (neurons and inputs). Each neuron can employ additive or multiplicative operators and the activation functions are chosen out of a finite set of 8 candidate functions. Even though $\sigma\pi$-CHNN has been successfully tested in nonlinear regression problems, it employs a classical multiplicative neuron ($\pi$-neuron) [2] as one of its components. It has been shown that the $\pi$-neuron presents 2 properties that may limit its applicability in complex problems [3]: (1) an excessive number of parameters is required and (2) decision surfaces generated by $\pi$-neuron are always centered at the origin of its input space. These 2 properties can potentially affect the overall performance of a $\sigma\pi$-CHNN.

To prevent these possible drawbacks, the $\pi$-neuron is replaced by the translated multiplicative neuron ($\pi_t$-neuron) [3], a multiplicative neuron model designed to overcome the disadvantages of $\pi$-neuron. The resulting architecture is called Sigma-Pi$_t$ Cascaded Hybrid Neural Network ($\sigma\pi_t$-CHNN), depicted in Fig. 1. The network in Fig. 1 is structurally similar to the original $\sigma\pi$-CHNN model, but the output is produced by a cascade of additive and translated multiplicative neurons.

An additive neuron, or $\sigma$-neuron, is defined by

$$y = f_k\left(w_0 + \sum_{i=1}^{m} w_i p_i\right), \qquad (1)$$

where $w_i$, $i = 0, \ldots, m$, are the adjustable weights, $p_i$, $i = 1, \ldots, m$, are the neuron's inputs, and $f_k(\cdot)$, $k \in \{0, \ldots, 7\}$ is the neuron's activation function, chosen out of the 8 candidate

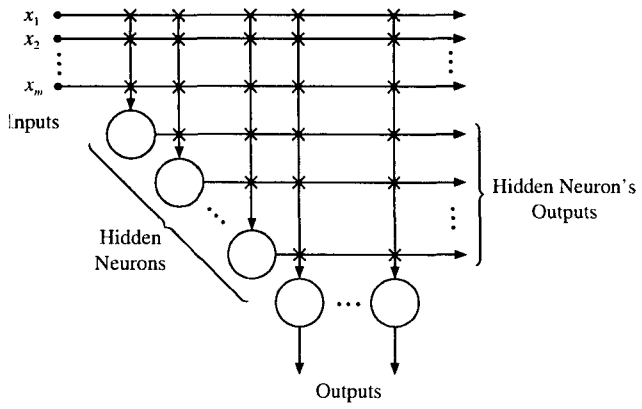


Fig. 1. A fully connected Sigma-Pi$_t$ Cascaded Hybrid Neural Network. Each cross represents a connection weight and each neuron can be a $\sigma$-neuron or a $\pi_t$-neuron (see (1) and (2)).

functions in Table I. The candidate functions in Table I are the same as those used in the $\sigma\pi$-CHNN model.

A translated multiplicative neuron, or $\pi_t$-neuron [3], is defined by

$$y = f_k\left(b\prod_{i=1}^{m}(p_i - t_i)\right), \tag{2}$$

where $b$ is a scaling factor and $t_i$, $i = 1,\ldots,m$, are the coordinates of the center of decision surface generated by the neuron. The activation function $f_k(\cdot)$, $k \in \{0,\ldots,7\}$, is also chosen out of the 8 candidates in Table I.

TABLE I
THE 8 CANDIDATES FOR ACTIVATION FUNCTION.

| *Function* | *Name* | *Expression* |
|---|---|---|
| $f_0$ | Hyperbolic tangent | $f_0(x) = \tanh(x)$ |
| $f_1$ | Logistic | $f_1(x) = \frac{1}{1+e^{-x}}$ |
| $f_2$ | Linear | $f_2(x) = x$ |
| $f_3$ | Cosine | $f_3(x) = \cos(x)$ |
| $f_4$ | Gaussian | $f_4(x) = e^{-x}$ |
| $f_5$ | Mexican hat | $f_5(x) = 2e^{-x^2}(1-2x^2)$ |
| $f_6$ | Sinc | $f_6(x) = \begin{cases} 1, \text{ for } x = 0 \\ \sin(x), \text{ otherwise} \end{cases}$ |
| $f_7$ | Null | $f_7(x) = 0$ |

The architecture shown in Fig. 1 is a fully connected $\sigma\pi_t$-CHNN. Note that not all the connections in Fig. 1 may be necessary. Moreover, to choose appropriate activation functions (from the candidate set in Table I) and aggregation type ($\sigma$ or $\pi_t$) for each neuron in the network is not a simple task. Therefore, a learning algorithm based on evolutionary computation associated with a second order Levenberg-Marquadt method is proposed to adjust both the architecture and weights of a $\sigma\pi_t$-CHNN.

## III. LEARNING IN $\sigma\pi_t$-CHNN

The evolutionary part of the original $\sigma\pi$-CHNN's learning algorithm consists in a standard, general purpose genetic algorithm, which causes the algorithm to run slowly. Thus, to improve the convergence times of the learning algorithm, a modified version of Evolutionary Programming Network (EPNet) [4] is used to evolve $\sigma\pi_t$-CHNN architectures. EPNet is specially designed to encourage fast evolution of small neural networks that present good behavior. Because the original EPNet evolves only classical neural networks (using only additive neurons and a single activation function in all the neurons), it was necessary to change some steps of EPNet to make it capable of determining the full architecture of a $\sigma\pi_t$-CHNN. To accomplish this, 2 new mutation operators are inserted in EPNet:

1) Node type change: this operator randomly changes, with the uniform distribution, the type ($\sigma$ or $\pi_t$) a certain amount of neurons of the network. The probability of changing the type of a single neuron is a user defined parameter of the method.
2) Activation function change: this operator changes the activation function of a certain amount of neurons of a $\sigma\pi_t$-CHNN. The new activation function for a neuron is chosen, with uniform random probability, from the 8 candidate functions in Table I. The probability of changing the activation function of a neuron is also a user defined parameter.

These 2 new operators are performed before node and connection mutations are tried, because changing neuron types and activation functions seem to be less disruptive for the behaviour of a $\sigma\pi_t$-CHNN than the other mutation operators.

In the original EPNet, the weights of a network are tuned by a combination of backpropagation and simulated annealing. To further accelerate the learning algorithm, the weights in $\sigma\pi_t$-CHNN are adjusted by a partial training mutation algorithm, composed of a Gaussian mutation operator combined to the Optimized Levenberg-Marquadt with Adaptive Momentum (OLMAM) [5] algorithm. This operator is applied as follows:

1) Train the network using OLMAM a certain number of steps. If the network performance does not improve significantly, then go to step 2; otherwise go to step 3. Here, when the number of correct classified patterns increases after training, a significant improvement is considered to have been achieved.
2) Apply a Gaussian mutation operator and train it again using OLMAM for some epochs.
3) If network's performance is better than before, accept the changes; otherwise try the other mutation operators.

The OLMAM algorithm replaces the Scaled Conjugate Gradient (SCG) [6] algorithm used in the original $\sigma\pi$-CHNN. This is because there is empirical evidence that OLMAM can overcome local minima in a network's error surface [5].

The codification scheme used in $\sigma\pi_t$-CHNN is the same as that used in the original $\sigma\pi$-CHNN [1]: each neuron is encoded in a vector, containing information about the type of each neuron, the activation function used, and the connections with other neurons in the network.

```
Randomly initialize population
Partial_training()
Rank-based selection
while (not termination criterion) {
  if (partial_training()) not successful
    if (change_node_type()) not successful
      if (change_activation_function())
      not successful
        if (delete_node()) not successful
          if (delete_connection())
          not successful
            add_node_connection()
  Obtain new generation
}
Further training
```

Fig. 2. Modified EPNet [7]. The node type change and activation function change are new mutation operators. The partial training operator is performed by the OLMAM [5] algorithm and the Gaussian mutation operator.

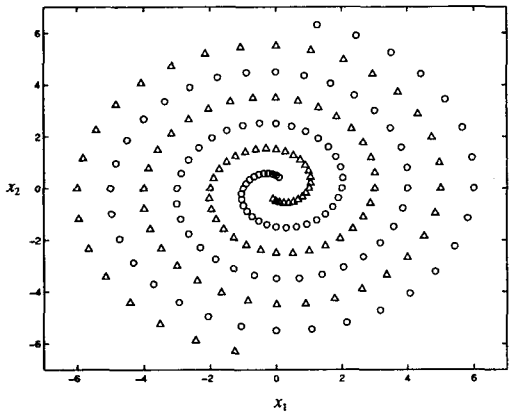


Fig. 3. The two spirals problem.

Figure 2 depicts the $\sigma\pi_t$-CHNN's learning algorithm. For details of the other mutation operators, refer to [4].

## IV. Application of $\sigma\pi_t$-CHNN in the 2 Spirals and Sonar Pattern Classification Problems

To evaluate the performance of $\sigma\pi_t$-CHNN, 2 pattern classification problems are chosen: the 2 spirals problem and the sonar problem. These 2 problems are chosen because they are considered difficult to solve and are commonly applied in neural networks research, thus allowing easier comparison with other neural architectures.

### A. 2 Spirals Problem

The 2 spirals problem consists in classifying a set of points, arranged in a spiral configuration, in 2 different classes (see Fig. 3). Although the data for this problem are generated artificially, it is considered hard to solve because of the peculiar distribution of data in a 2 dimensional space. The 2 spirals problem has been widely applied as a benchmark for neural network architectures [5][8]. The objective of this experiment is to infer $\sigma\pi_t$-CHNN's ability of generating complex decision surfaces and to compare it with other neural network architectures.

TABLE II

Results for the Spirals Problem.

| *Architecture* | *Classification Accuracy* | *Number of Hidden Neurons* |
|---|---|---|
| CASCOR | 100% | 12 |
| $\sigma\pi$-CHNN | 100% | 12 |
| $\sigma\pi_t$-CHNN | 100% | 11 |

The parameters for the learning algorithms are chosen as follows:

- Maximum number of hidden neurons: 15
- Maximum number of generations: 1000
- Population size: 30
- Probability of node type change: 0.5
- Probability of activation function change: 0.5
- Probability of hidden node deletion: 0.3
- Probability of hidden node addition: 0.3
- Probability of node and connection addition: 0.3

Table II shows the results obtained. The performance of $\sigma\pi_t$-CHNN is compared with Cascade Correlation (CASCOR) [8] architecture and with $\sigma\pi$-CHNN. All the considered networks can achieve 100% of classification accuracy, but note that $\sigma\pi_t$-CHNN's learning algorithm can generate a network that achieves this result using less hidden neurons than the others, confirming that the introduction of $\pi_t$-neuron can improve the learning capabilities of a neural network.

### B. Sonar Problem

The sonar problem [9] consists in classifying sonar signals in 2 categories – metal cylinders (mines) and rocks. The sonar problem has also been used to evaluated neural network architectures and learning algorithms. Although this problem is known to be linear, it is usually hard to solve it using a conventional multilayer perceptron (MLP) with no hidden neurons [5]. In [9], it is reported that only with the addition of 12 hidden neurons, a MLP is able to solve this problem with a success rate of 100%. It has been argued that the nonuniform distribution of input data makes this problem difficult to solve without hidden nodes. Here, the objective is to infer $\sigma\pi_t$-CHNN's ability of finding the global optimum of a problem.

The parameters for the learning algorithms are chosen as follows:

- Maximum number of hidden neurons: 3
- Maximum number of generations: 100
- Population size: 10
- Probability of node type change: 0.5
- Probability of activation function change: 0.5
- Probability of hidden node deletion: 0.3
- Probability of hidden node addition: 0.3
- Probability of node and connection addition: 0.3

Table III shows the results obtained. Here, all the considered architectures can perfectly solve the sonar problem, i.e., achieve 100% of classification accuracy. Moreover, a Multilayer Perceptron (MLP) with no hidden neurons, when trained with OLMAM algorithm, can also solve this problem.

TABLE III

RESULTS FOR THE SONAR PROBLEM.

| *Architecture* | *Classification Accuracy* | *Number of Hidden Neurons* |
|---|---|---|
| MLP with backprop. | 100% | 12 |
| MLP with OLMAM | 100% | 0 |
| $\sigma\pi_t$-CHNN | 100% | 0 |

But note that this result was obtained with the prior knowledge of linearity of data distribution, i.e., it was known in advance that such an architecture would solve this problem. On the other hand, $\sigma\pi_t$-CHNN's learning algorithm does not make any assumption on data distribution, i.e., the learning process is responsible for finding an appropriate network architecture that fits the problem at hand. This is an important property for those problems in which it is not possible to make any assumptions on the problem's data distribution. Finally, note that the learning algorithm of $\sigma\pi_t$-CHNN is able to find a global optimum for this problem, i.e., a network with no hidden neurons.

## V. CONCLUSIONS

An extension for the Sigma-Pi Cascaded extended Hybrid Neural Network Model ($\sigma\pi$-CHNN) [1], called Sigma-Pi$_t$ Cascaded Hybrid Neural Network ($\sigma\pi_t$-CHNN), is proposed. In $\sigma\pi_t$-CHNN, instead of the classical multiplicative neuron ($\pi$-neuron), the translated multiplicative neuron ($\pi_t$-neuron) [3] is employed as one of the network components, along with the additive neuron ($\sigma$-neuron). The activation functions for each neuron of a $\sigma\pi_t$-CHNN are chosen out of 8 candidate functions. A modified version of EPNet [4] is used to optimize the network architecture, whereas the Optimized Levenberg-Marquadt with Adaptive Momentum (OLMAM) [5] algorithm is used to adjust the weights of the network. To evaluate the capabilities of $\sigma\pi_t$-CHNN, the 2 spirals and sonar pattern classification problems are used. In the 2 spirals problem, $\sigma\pi_t$-CHNN can generate neural networks with 10% less hidden neurons than that in previous neural models, whereas an optimal solution, i.e., a solution with no hidden neurons, is found by $\sigma\pi_t$-CHNN in the sonar problem.

The results obtained in the 2 spirals problem show that $\sigma\pi_t$-CHNN can generate smaller networks than previous approaches, thus confirming that $\sigma\pi_t$-CHNN has improved representation capability, when compared to previous neural models. Moreover, the results obtained in the sonar problem show that the learning algorithm of $\sigma\pi_t$-CHNN is capable of finding an optimal solution for a learning problem. Therefore, it can be concluded that $\sigma\pi_t$-CHNN has the potential to be successfully applied in challenging real world problems.

As subsequent application, $\sigma\pi_t$-CHNN will be evaluated in different application domains, such as image processing and automatic control. For further improving the performance of the learning algorithm, parallel implementations of evolutionary neural networks may be investigated. Another research direction is to extend the proposed approach to recurrent neural network architectures, and apply it in the approximation of nonstationary mappings.

## REFERENCES


[1] E. M. Iyoda, K. Hirota, and F. J. Von Zuben, "Sigma-pi cascade extended hybrid neural networks," *Journal of Advanced Computational Intelligence*, vol. 6, pp. 126–134, Oct. 2002.

[2] B.-T. Zhang, "A bayesian evolutionary approach to the design and learning of heterogeneous neural trees," *Integrated Computer-Aided Engineering*, vol. 9, no. 1, pp. 73–86, 2002.

[3] E. M. Iyoda, H. Nobuhara, and K. Hirota, "Translated multiplicative neuron: An extended multiplicative neuron that can translate decision surfaces," in *Proceedings of the IEEE International Conference on Computational Cybernetics*, (Siófok, Hungary), Aug. 2003. To appear.

[4] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, pp. 694–713, May 1997.

[5] N. Ampazis and S. J. Perantonis, "Two highly efficient second order algorithms for training feedforward networks," *IEEE Transactions on Neural Networks*, vol. 13, pp. 1064–1074, Sept. 2002.

[6] M. F. Møller, "Scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, no. 4, pp. 525–533, 1993.

[7] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, pp. 1423–1447, Sept. 1999.

[8] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," Tech. Rep. CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Aug. 1991.

[9] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, vol. 1, pp. 75–89, 1988.