Published in *Proceedings of INNS World Congress on Neural Networks*,
WCNN'94-SanDiego, June 4-9, Vol.3, pp.262-267 (1994).

(a) **The title of the article**
Ability of the 3D Vector Version of the Back-Propagation to Learn 3D Motion

(b) **The authors' full names**
Tohru Nitta

(c) **Current Affiliations**
Neuroscience Research Institute,
National Institute of Advanced Industrial Science and Technology (AIST),
AIST Tsukuba Central 2, 1-1-1 Umezono, Tsukuba-shi, Ibaraki, 305-8568 Japan.
E-mail: tohru-nitta@aist.go.jp

Ability of the 3D Vector Version of the Back-Propagation to Learn 3D Motion

Tohru Nitta
Electrotechnical Laboratory,
1-1-4 Umezono, Tsukuba Science City, Ibaraki, 305 Japan.

Abstract :
The 3D vector version of the back-propagation algorithm (called "3DV-BP") is a natural extension of the complex-valued version of the back-propagation algorithm (called "Complex-BP"). The Complex-BP can be applied to multi-layered neural networks whose weights, threshold values, input and output signals are all complex numbers, and the 3DV-BP can be applied to multi-layered neural networks whose threshold values, input and output signals are all 3D real valued vectors, and whose weights are all 3D orthogonal matrices. It has already been reported that an inherent property of the Complex-BP is its ability to learn "2D motion". This paper shows in computational experiments that the 3DV-BP has the ability to learn "3D motion", which corresponds to the ability of the Complex-BP to learn "2D motion".

1 Introduction

One of the most popular neural network models is the multi-layered network and the related back-propagation training algorithm, called here, "Real-BP" [7]. Back-propagation networks have many successful applications.

The "Complex-BP" is a complex valued version of the back-propagation algorithm, which can be applied to multi-layered neural networks whose weights, threshold values, input and output signals are all complex numbers [1, 3]. This algorithm enables the network to learn complex valued patterns naturally. It has already been reported that an inherent property of the Complex-BP is its ability to learn "2D motion" [1, 3]. And also, the Complex-BP has been applied to the interpretation of optical flow (motion vector field calculated from images) and estimation of motion which are important tasks in computer vision [5, 6].

The "3DV-BP" is a three-dimensional vector version of the back-propagation algorithm which can be applied to multi-layered neural networks whose threshold values, input and output signals are all 3D real valued vectors, and whose weights are all 3D orthogonal matrices [2]. This algorithm is a natural extention of the Complex-BP algorithm. This paper shows in computational experiments that the 3DV-BP has the ability to learn "3D motion", which corresponds to the ability of the Complex-BP to learn "2D motion".

Hereafter, we shall refer to a real valued (usual) neural network used by the Real-BP as a "Real-BP network", a complex valued neural network used by the Complex-BP as a "Complex-BP network", and a three-dimensional vector valued neural network used by the 3DV-BP as a "3DV-BP network".

This paper is organized as follows: Section 2 describes the 3DV-BP algorithm, and Section 3 deals with the empirical analyses of the ability of the 3DV-BP network model to learn 3D motion . The paper will end with our conclusions.

2 The "3DV-BP" Algorithm

This section briefly describes the 3DV-BP algorithm [2]. It can be applied to multi-layered neural networks in which threshold values, input and output signals are all 3D real valued vectors, and whose weights are all 3D orthogonal matrices, and the output function F of a neuron can be defined as

$$F(\boldsymbol{A}) = \begin{bmatrix} f(a_1) \\ f(a_2) \\ f(a_3) \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \tag{1}$$

where $f(u) = 1/(1 + \exp[-u])$, that is, each component of an output $F(\boldsymbol{A})$ of a neuron means the sigmoid function of each component a_m of the net input $\boldsymbol{A}$ to the neuron, respectively ($m = 1, 2, 3$). The learning rule has been obtained by using a steepest descent method.

3 Ability to Learn 3D Motion

We wiil now present some illustrative examples to show that an adaptive network of 3D valued neurons can be used to learn 3D motion such as rotation, similar transformation, and translation. Due to space limitations, we will restrict the presentation of our results to similar transformations, although similar work has been carried out on rotations, and parallel displacement [4].

We used a 1-6-1 three-layered network, which transformed a point (x_1, x_2, x_3) into another point (x'_1, x'_2, x'_3) in 3-dimensional space. Although the 3DV-BP network generates a value $\boldsymbol{X} = {}^t[x_1\ x_2\ x_3]$ within the range $0 \leq x_1, x_2, x_3 \leq 1$, for the sake of convenience, we present it in the figures given below as having a transformed value within the range $-1 \leq x_1, x_2, x_3 \leq 1$.

We also conducted experiments with a 3-15-3 network with real valued weights and thresholds, to compare the 3DV-BP with the Real-BP. The first component of a 3-vector was input into the first input neuron, the second component was input into the second input neuron, and the third component was input into the third input neuron. The output from the first output neuron was interpreted as the first component of a 3-vector, and the output from the second output neuron was interpreted as the second component, and the output from the third output neuron was interpreted as the third component.

The learning constant ε used in these experiments was 0.5. The initial components of the weights and the thresholds were chosen to be random real numbers between $-$ 0.3 and 0.3. We determined that learning finished when

$$\sqrt{\sum_p \sum_{k=1}^{N} ||\boldsymbol{T}_k^{(p)} - \boldsymbol{O}_k^{(p)}||^2} = 0.05 \tag{2}$$

held, where $||\boldsymbol{x}|| \overset{\text{def}}{=} \sqrt{x_1^2 + x_2^2 + x_3^2}$, $\boldsymbol{x} = {}^t[x_1 \ \ x_2 \ \ x_3]$; $\boldsymbol{T}_k^{(p)}$, $\boldsymbol{O}_k^{(p)} \in \boldsymbol{R}^3$ denote the desired output value, the actual output value of the neuron k for the pattern p, i.e. the left side of (2) means the error between the desired and actual output patterns ($\boldsymbol{R}$ denotes the set of real numbers); N denotes the number of neurons in the output layer. We regarded presenting a set of learning patterns to the neural network as one learning cycle. In this connection, time complexity per learning cycle of the 1-6-1 three-layered network for the 3DV-BP was nearly equal to that of the 3-15-3 three-layered network for the standard BP, as seen in Table 1. Furthermore, the space complexity (i.e. the number of parameters) was almost half that of the standard BP.

The experiments described in this section, consisted of two parts - a training step, followed by a test step.

3.1 Learning of a Simple 3D Motion

Figs. 1 and 2 show the result of an experiment on a simple similar transformation.

The training step consisted of learning a set of (3D orthogonal matrix valued) weights and (3D vector valued) thresholds, such that the input set of 11 points (with equal intervals) lying along the straight line

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = t \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad (0.0 \leq t \leq 1.0) \tag{3}$$

gave as output, half-scaled straight line points (Fig. 1). The output training points also lay along the same straight line (equation (3)), but the range was $0.0 \leq t \leq 0.5$. To avoid complexity, we omitted the points and showed only the lines joining them in the figures.

In a second (test) step, the 48 input points (with equal intervals) lying on three squares would hopefully be mapped to an output set of points lying on three half-scaled squares. The actual output test points for the 3DV-BP did, indeed, almost lie on the squares (but, with an error) (Fig. 2).

To compare how a real valued network would perform, the 3-15-3 (real valued) network mentioned above was trained using the same pairs of training points lying along equation (3). The same 48 test points lying on the three squares were then input with this real network. All points were “mapped” onto straight lines, as shown in Fig. 2.

3.2 Learning of a More Complex 3D Motion

This subsection shows that the 3DV-BP can make more complicated transformation.

Fig. 3 shows how the training points mapped onto each other. Those 11 points (with equal intervals) lying along the straight line indicated by “Input Pattern 1”, mapped onto points along the same line, but with a scale reduction factor of 2. Those 11 points (with equal intervals) lying along the straight line indicated by “Input Pattern 2” mapped onto points along the same line, but with a scale reduction factor of 10. All the training points lie along the straight line

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = t \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad (-1.0 \leq t \leq 1.0), \tag{4}$$

where "Input Pattern 1" for $0.0 \leq t \leq 1.0$, "Output Pattern 1" for $0.0 \leq t \leq 0.5$, "Input Pattern 2" for $-1.0 \leq t \leq 0.0$, and "Output Pattern 2" for $-0.1 \leq t \leq 0.0$.

In the test step, by presenting the 60 points lying on the three circles $x^2 + z^2 = 1$, $y^2 + z^2 = 1$ and $x^2 + y^2 = 1$, the actual output points took the patterns as shown in Fig. 4.

It appears that this 3DV-BP network has learned to generalize the reduction factor α as a function of the position in three-dimensional space, i.e. a point ${}^t[x \quad y \quad z]$ is transformed into another point $\alpha^t[x \quad y \quad z]$, where $\alpha({}^t[x \quad y \quad z]) \approx 0.5$ for $x, y, z \geq 0$, and $\alpha({}^t[x \quad y \quad z]) \approx 0.1$ for $x, y, z \leq 0$.

4 Conclusions

We investigated by computational experiments the characteristics of the 3DV-BP algorithm which is a natural extension of the Complex-BP algorithm. It was learned that the 3DV-BP had the ability to learn "3D motion" such as similar transformation as its inherent property, which corresponded to the ability of the Complex-BP to learn "2D motion". We expect that applications for the 3DV-BP algorithm will be found in such areas as 3D image processing.

Acknowledgements

The author expresses his thanks to Dr.K.Ohta, Director of the Computer Science Division, and Dr.T.Higuchi, Chief of the Computational Models Section, for having an opportunity to do this study and their continual encouragement.

References

[1] Nitta, T. and Furuya, T. (1991). A Complex Back-Propagation Learning. *Transactions of Information Processing Society of Japan*, Vol. 32, No. 10, pp. 1319-1329 (in Japanese).

[2] Nitta, T. (1993). A Three-Dimensional Back-Propagation. *Proc. INNS World Congress on Neural Networks*, Portland, Vol. 3, pp. 572-575.

[3] Nitta, T. (1993). A Complex Numbered Version of the Back-Propagation Algorithm. *Proc. INNS World Congress on Neural Networks*, Portland, Vol. 3, pp. 576-579.

[4] Nitta, T. (1993). Ability of the 3D Vector Version of the Back-Propagation to Learn 3D Motion. *ETL Technical Report*, TR-93-24 (in Japanese).

[5] Miyauchi, M. and Seki, M. (1992). Interpretation of Optical Flow through Neural Network Learning. *Proc. IEEE International Conference on Communication Systems /International Symposium on Information Theory and its Applications*, Singapore, pp. 1247-1251.

[6] Miyauchi, M., Seki, M., Watanabe, A. and Miyauchi, A. (1992). Interpretation of Optical Flow through Neural Network Learning. *Proc. IAPR Workshop on Machine Vision Applications*, Tokyo, pp. 523-528.

[7] Rumelhart, D. E. et al. (1986). *Parallel Distributed Processing*, Vol. 1, p. 547, MIT press.

Network	Time complexity			Space complexity		
	$\times$ and $\div$	$+$ and $-$	Sum	Weights	Thresholds	Sum
3DV-BP 1-6-1	255	141	396	36	21	57
Standard BP 3-15-3	264	141	405	90	18	108

Table 1 The Computational Complexity of the 3DV-BP and the Standard BP. Time complexity means the sum of the four operations performed per learning cycle. Space complexity means the sum of the parameters (weights and thresholds).

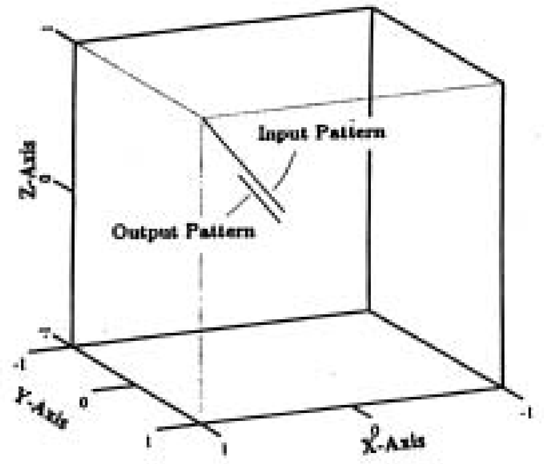

Fig. 1 Learning pattern (Simple transformation).

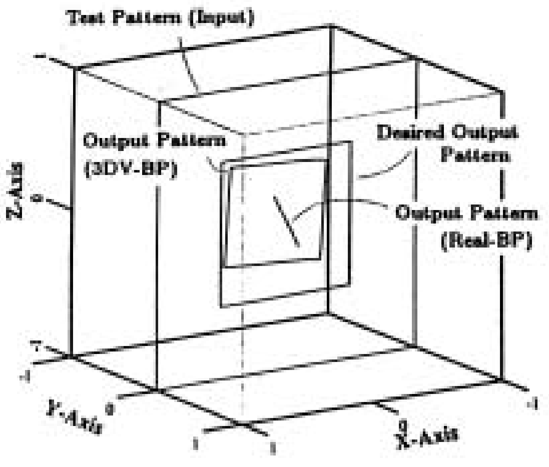

Fig. 2 (a) Test pattern 1 (Simple transformation).

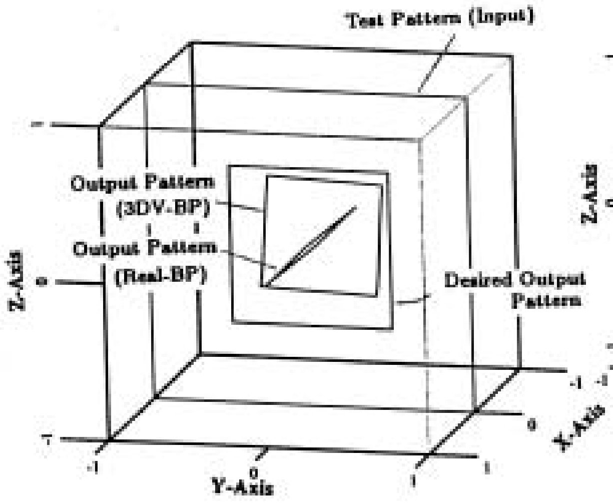

Fig. 2 (b) Test pattern 2 (Simple transformation).

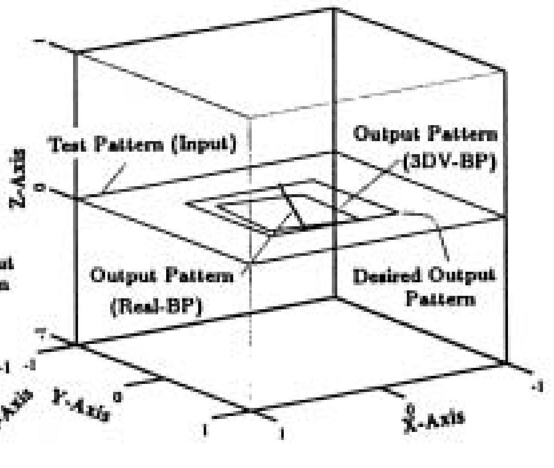

Fig. 2 (c) Test pattern 3 (Simple transformation).

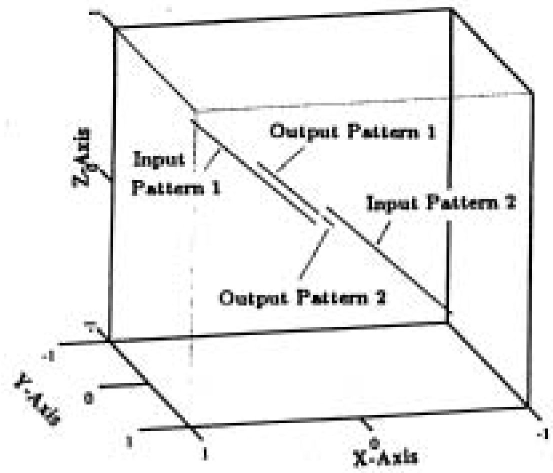

Fig. 3 Learning patterns (Complex transformation).

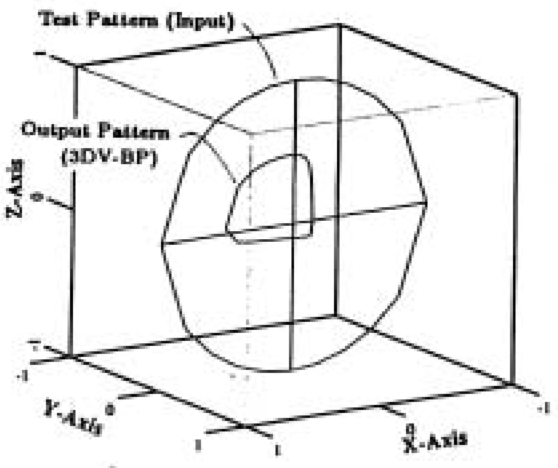

Fig. 4 (a) Test pattern 1 (Complex transformation).

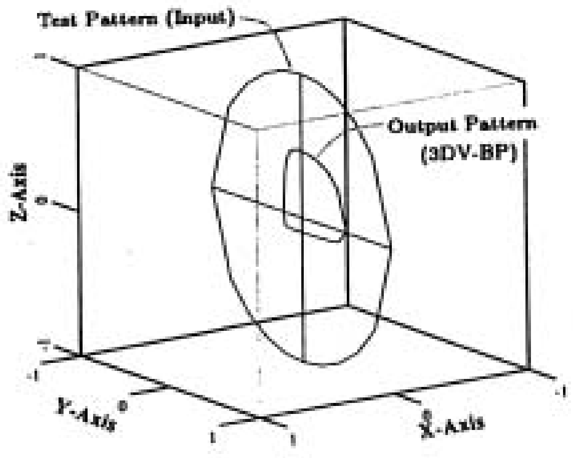

Fig. 4 (b) Test pattern 2 (Complex transformation).

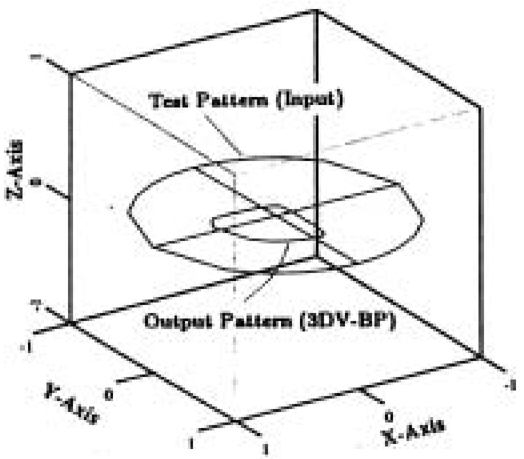

Fig. 4 (c) Test pattern 3 (Complex transformation).