Contents lists available at SciVerse ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Training multi-layered neural network neocognitron

Kunihiko Fukushima [*]

Fuzzy Logic Systems Institute, Iizuka, Fukuoka, Japan

ARTICLE INFO

Article history:
Received 6 December 2011
Received in revised form 6 July 2012
Accepted 4 January 2013

Keywords:
Visual pattern recognition
Hierarchical network
Neocognitron
Learning
Add-if-silent
Interpolating-vector

ABSTRACT

This paper proposes new learning rules suited for training multi-layered neural networks and applies them to the *neocognitron*. The neocognitron is a hierarchical multi-layered neural network capable of robust visual pattern recognition. It acquires the ability to recognize visual patterns through learning. For training intermediate layers of the hierarchical network of the neocognitron, we use a new learning rule named *add-if-silent*. By the use of the add-if-silent rule, the learning process becomes much simpler and more stable, and the computational cost for learning is largely reduced. Nevertheless, a high recognition rate can be kept without increasing the scale of the network. For the highest stage of the network, we use the method of *interpolating-vector*. We have previously reported that the recognition rate is greatly increased if this method is used during recognition. This paper proposes a new method of using it for both learning and recognition. Computer simulation demonstrates that the new neocognitron, which uses the add-if-silent and the interpolating-vector, produces a higher recognition rate for handwritten digits recognition with a smaller scale of the network than the neocognitron of previous versions.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

The author previously proposed an artificial neural network *neocognitron* for robust visual pattern recognition (Fukushima, 1980). Its architecture was initially suggested by neurophysiological findings on the visual systems of mammals (e.g., Hubel & Wiesel, 1962, 1965). It is a hierarchical multi-layered network and acquires the ability to robustly recognize visual patterns through learning.

The neocognitron consists of layers of S-cells, which resemble simple cells of the visual cortex, and layers of C-cells, which resemble complex cells. These layers of S-cells and C-cells are arranged alternately in a hierarchical manner.

Input connections of S-cells are variable and are modified through learning. After learning, S-cells come to work as feature-extracting cells, and extract local features from the input pattern presented to the input layer (or photoreceptor array).

C-cells, whose input connections are fixed, exhibit an approximate invariance to the position of the stimuli presented within their receptive fields. We can also express through this operation that the response of a layer of S-cells is spatially blurred in the succeeding layer of C-cells.

The C-cells in the highest stage work as recognition cells, which indicate the result of pattern recognition. After learning, the neocognitron can recognize input patterns robustly, with little effect from deformation, change in size, or shift in position.

Varieties of modifications, extensions and applications of the neocognitron, as well as varieties of related networks, have been reported so far (e.g., Elliffe, Rolls, & Stringer, 2002; LeCun et al., 1989; LeCun, Bottou, Bengio, & Haffner, 1998; Lo et al., 1995; Riesenhuber & Poggio, 1999; Satoh, Kuroiwa, Aso, & Miyake, 1999). They are all hierarchical multi-layered networks and have an architecture of *shared connections*, which is sometimes called a *convolutional net*. They also have a mechanism of pooling outputs of feature-extracting cells. The pooling operation can also be interpreted as a blurring operation. In the neocognitron, the pooling operation, which is done by C-cells, is performed by a weighted sum of the outputs of feature-extracting S-cells. In some networks, the pooling is realized by simply reducing the density of cells in higher layers. In some other networks, it is replaced by a MAX operation.

In the learning phase of the neocognitron, input connections of feature-extracting cells (S-cells) are modified by the presentation of training patterns. Several methods of training the neocognitron have been proposed so far, but the best method has not been revealed yet.

Generally speaking, in the learning of a hierarchical multi-layered network, the training of an intermediate layer cannot progress independently of other layers. The result of training of an intermediate layer largely affects the training for the succeeding layers. For training intermediate layers of the neocognitron of previous versions, competitive learning with *winner-take-all* rule (Fukushima, 1980, 2003) or *winner-kill-loser* rule (Fukushima, 2010; Fukushima, Hayashi, & Léveillé, 2011) have been mainly used. This paper proposes another learning rule, which is named *add-if-silent*, and uses it for training intermediate layers of the

[*] Correspondence to: 634-3, Miwa, Machida, Tokyo 195–0054, Japan. Tel.: +81 44 988 5272; fax: +81 44 988 5272.
E-mail address: fukushima@m.ieice.org.
URL: http://www4.ocn.ne.jp/~fuku_k/index-e.html.

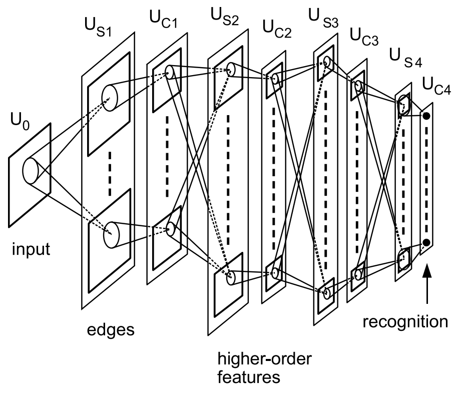

Fig. 1. The architecture of the neocognitron.

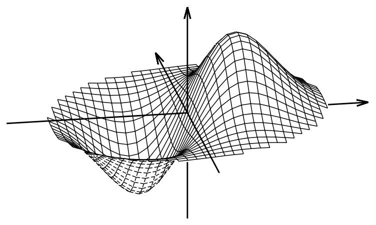

Fig. 2. The connections to an S-cell of U$_{S1}$ that extract vertical edges.

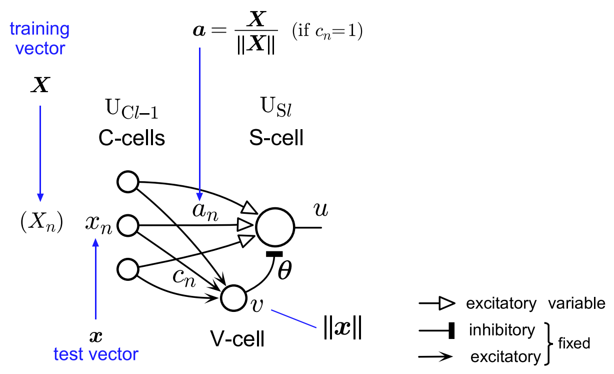

Fig. 3. Connections converging to an S-cell.

neocognitron. By the use of the add-if-silent rule, the learning process becomes much simpler and more stable. Nevertheless, a high recognition rate can be kept without increasing the scale of the network.

It is already known that the recognition rate can be increased by the method of the *interpolating-vector* for classification of patterns at the highest stage of the neocognitron (Fukushima, 2007). In the previous neocognitron, however, the method of the interpolating-vector was used only during recognition, and the training of the highest stage was done by a supervised winner-take-all method. This paper proposes a method for applying the interpolating-vector, not only in the recognition phase, but also in the learning phase. We then show that, by applying the interpolating-vector also during learning (namely, a supervised interpolating-vector), we can obtain a higher recognition rate with a smaller scale of the network (that is to say, with a smaller computational cost).

With computer simulation for recognizing handwritten digits, we show that the new neocognitron, which is trained with the add-if-silent rule for intermediate layers and with the method of the interpolating-vector for the highest stage, produces a higher recognition rate than conventional ones with a smaller scale of the network.

2. Outline of the network

This section explains the outline of the network. More detailed mathematical expressions are found in Appendices A–G.

2.1. Network architecture

The neocognitron consists of layers of S-cells, which resemble simple cells in the visual cortex, and layers of C-cells, which resemble complex cells. These layers of S-cells and C-cells are arranged alternately in a hierarchical manner.

Fig. 1 shows the architecture of the network that is discussed in this paper. In the figure, U$_{Sl}$, for example, indicates the layer of S-cells of the lth stage. The network has four stages of S- and C-cell layers.

Each layer of the network is divided into a number of sub-layers, called *cell-planes*, depending on the feature to which cells respond preferentially. In Fig. 1, each rectangle drawn with thick lines represents a cell-plane. Incidentally, a cell-plane is a group of cells that are arranged retinotopically and share the same set of input connections (Fukushima, 1980). As a result, all cells in a cell-plane have receptive fields of an identical characteristic, but the locations of the receptive fields differ from cell to cell.

2.2. Edge extraction

The stimulus pattern is presented to the input layer, U$_0$. The output of U$_0$ is sent directly to U$_{S1}$. An S-cell of this layer resembles a simple cell in the primary visual cortex, and responds selectively to an edge of a particular orientation. In the computer simulation discussed later, U$_{S1}$ has $K_{S1} = 16$ cell-planes, whose preferred orientations are chosen at an interval of 22.5°. As a result, the contours of the input image are decomposed into edges of every orientation in U$_{S1}$.

When a line drawing is presented to U$_0$, for example, edges on both sides of the line are extracted in U$_{S1}$. This situation can be seen in Fig. 11 below.

S-cells of U$_{S1}$ are made of analog threshold elements. An analog threshold element calculates weighted sum of its inputs and cuts off the negative component by a threshold operation. In other words, an S-cell of U$_{S1}$ extracts an oriented edge directly from U$_0$ using a linear spatial filter, where the negative component of the filtered output is suppressed to zero.

The shape the linear spatial filter is derived from directional differentiation of a two-dimensional Gaussian curve (See Appendix B) (Mallot, 2000). Fig. 2 shows its shape. To increase the orientation selectivity of the S-cells, there is a mechanism of weak lateral inhibition among S-cells of different preferred orientations.

2.3. S-cells with subtractive inhibition

Each S-cell of layer U$_{Sl}$ ($l \geq 2$) receives excitatory signals directly from a group of C-cells, which are cells of the preceding layer U$_{Cl-1}$. It also receives an inhibitory signal through a V-cell, which accompanies the S-cell. The V-cell receives fixed excitatory connections from the same group of C-cells as does the S-cell, and always responds with the average intensity (root-mean-square) of the output of the C-cells. Fig. 3 illustrates these connections, which converge to an S-cell.

In the conventional neocognitron, the inhibitory signals from V-cells were made to work in a divisional manner. In the present neocognitron, however, the inhibition works in a subtractive manner, by which the neocognitron becomes much more robust against background noise (Fukushima, 2011).

Let a_n be the strength of the excitatory connection to the S-cell from the nth C-cell, whose output is x_n. The output u of the S-cell

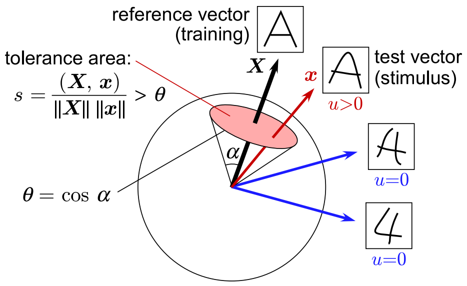
reference vector (training)
tolerance area:
$s = \frac{(\boldsymbol{X}, \boldsymbol{x})}{\|\boldsymbol{X}\| \, \|\boldsymbol{x}\|} > \theta$
$\boldsymbol{X}$
$\boldsymbol{x}$
test vector (stimulus)
$u>0$
α
$\theta = \cos \alpha$
$u=0$
$u=0$

Fig. 4. Tolerance area of an S-cell in the multi-dimensional feature space.

is given by

$$u = \frac{1}{1-\theta} \cdot \varphi \left[\sum_n a_n x_n - \theta \, v \right], \tag{1}$$

where $\varphi[\ ]$ is a function defined by $\varphi[x] = \max(x, 0)$. Namely, $\varphi[\ ]$ is a nonlinear function like a half-wave rectifier. The strength of the inhibitory connection is θ, which determines the threshold of the S-cell ($0 \leq \theta < 1$). The response of the V-cell is given by

$$v = \sqrt{\sum_n c_n \, {x_n}^2}, \tag{2}$$

where c_n is the strength of the fixed excitatory connection from the nth C-cell.

The value of c_n is used as a kind of weight, which compensates the effect of distance in the retinotopic space between the post-synaptic S-cell and a presynaptic C-cell. Namely, it is a monotonically decreasing function of distance between the locations of the receptive fields of pre- and post-synaptic cells. To simplify discussion, however, we discuss here the case of $c_n = 1$. The case of $c_n \neq 1$ is discussed in Appendix C. All discussions below hold even under $c_n \neq 1$, if we use the *weighted* inner product, which is defined in Appendix C.

We now use vector notation $\boldsymbol{x} = (x_1, x_2, \ldots, x_n, \ldots)$ to represent the input signal to the S-cell, namely, the response of presynaptic C-cells, from which the S-cell receives excitatory signals. We sometimes call $\boldsymbol{x}$ the test vector. With this vector notation, the response of the V-cell of (2) can be expressed as

$$v = \|\boldsymbol{x}\|, \tag{3}$$

which is the norm of vector $\boldsymbol{x}$ (see also Appendix C).

Connection $\boldsymbol{a} = (a_1, a_2, \ldots, a_n, \ldots)$ is given by

$$\boldsymbol{a} = \boldsymbol{X} / \|\boldsymbol{X}\|, \tag{4}$$

where $\boldsymbol{X} = (X_1, X_2, \ldots, X_n, \ldots)$ is the training vector that the S-cell has learned. In the case of layers U_{S2} and U_{S3}, $\boldsymbol{X}$ is the response of presynaptic C-cells at the moment when the S-cell is generated. Incidentally, concrete methods of training S-cells are discussed later in Sections 3 and 4.

Substituting (3) and (4) to (1), we can express the response of the S-cell by

$$u = \|\boldsymbol{x}\| \cdot \frac{\varphi[s - \theta]}{1 - \theta}, \tag{5}$$

where

$$s = \frac{(\boldsymbol{X}, \boldsymbol{x})}{\|\boldsymbol{X}\| \cdot \|\boldsymbol{x}\|}. \tag{6}$$

In the multi-dimensional feature space, s shows a kind of similarity between $\boldsymbol{X}$ and $\boldsymbol{x}$, which is defined by the normalized inner product of $\boldsymbol{X}$ and $\boldsymbol{x}$. As illustrated in Fig. 4, if similarity s is larger than threshold θ, the S-cell yields a non-zero response (Fukushima, 2011). The area that satisfies $s > \theta$ in the multi-dimensional feature space is named the tolerance area of the S-cell. We call $\boldsymbol{X}$, which is the training vector, the reference vector of the S-cell. Using a neurophysiological term, we can also express that $\boldsymbol{X}$ is the preferred (optimal) feature of the S-cell.

2.4. Learning

The input connections to S-cells, except in layer U_{S1}, are variable and are modified through learning. After having finished learning, S-cells come to work as feature-extracting cells. In a higher stage, they extract more global features.

Training of the network is performed from lower stages to higher stages: after the training of a lower stage has been completely finished, the training of the succeeding stage begins. For the training of S-cells of U_{Sl} ($l \geq 2$), the response of C-cells of the preceding layer U_{Cl-1} works as a training stimulus. The same set of training patterns is presented to U_0, for the training of all stages.

As mentioned above, all cells in a cell-plane share the same set of input connections. This condition of shared connections has to be kept even during learning, when input connections to S-cells are to be generated or renewed. To keep this condition, a *seed-cell* is employed when a training pattern is presented. The seed-cell is a cell that works like a seed in crystal growth. If a seed-cell is chosen from a cell-plane, its input connections are modified depending on the responses of the C-cells presynaptic to it. Since all cells in the cell-plane share the same set of input connections, all other cells in the cell-plane follow the seed-cell and come to have the same input connections as the seed-cell.

Concrete methods of choosing and training seed-cells are discussed below in Sections 3 and 4.

2.5. C-cell layers

In each stage of the hierarchical network, the output of layer U_{Sl} is fed to layer U_{Cl}. Except the highest stage, U_{Sl} and U_{Cl} has the same number of cell-planes, and there is a one-to-one correspondence between cell-planes of the two layers.

Each C-cell has fixed excitatory connections from a group of S-cells of the corresponding cell-planes of S-cells. Through these connections, each C-cell averages the responses of S-cells whose receptive field locations are slightly deviated. In other words, S-cells' response is spatially blurred in the succeeding cell-planes of C-cells. The averaging operation is essential, not only for endowing neural networks with an ability to recognize deformed patterns robustly, but also for smoothing additive random noise contained in the responses of S-cells.

Similarly to previous neocognitrons, C-cells of U_{C1} and U_{C2} have inhibitory surround in their input connections (Fukushima, 2003). Namely, the excitatory connections, which produce a blur, are surrounded annularly by inhibitory connections, where each C-cell receives both excitatory and inhibitory signals from S-cells of the same preferred feature. Different from conventional neocognitrons, however, the inputs to C-cells are averaged, not by linear summation, but by root-mean-square (Fukushima, 2011).

3. Learning with add-if-silent

3.1. Add-if-silent rule

For the training of S-cells of intermediate layers, namely layers U_{S2} and U_{S3}, we use a new learning rule, which is named *add-if-silent*. If all post-synaptic cells are silent for a training stimulus, as shown in Fig. 5, a new S-cell is generated and added to the layer. The strength of the input connections of the generated S-cell is proportional to the response of the presynaptic C-cells at this moment.

Different from other learning rules shown in Fig. 8 below, however, once the S-cell is generated and added to the network, the input connections to the S-cell do not change any more.

Fig. 6 illustrates how new S-cells are generated by the add-if-silent rule in the multi-dimensional vector space. The circle around

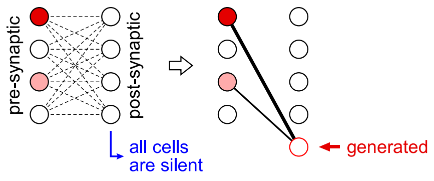

Fig. 5. The add-if-silent rule. A new cell is generated when all post-synaptic cells are silent. In the figure, the response strength of each cell is represented by the depth of the color. Incidentally, generation of a new cell occurs not only under the add-if-silent rule, but also under some other rules, which are shown in Fig. 8 below.

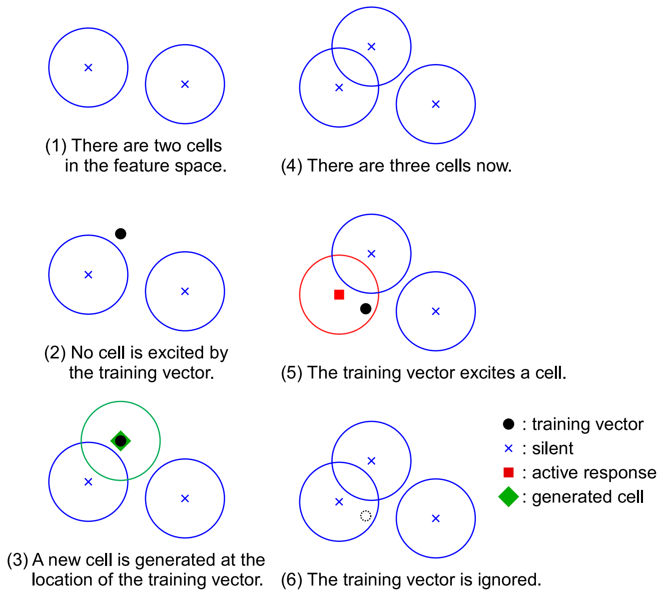
(1) There are two cells in the feature space.
(4) There are three cells now.
(2) No cell is excited by the training vector.
(5) The training vector excites a cell.
● : training vector
× : silent
■ : active response
◆ : generated cell
(3) A new cell is generated at the location of the training vector.
(6) The training vector is ignored.

Fig. 6. The process of generating new cells by the add-if-silent rule in the multi-dimensional feature space is illustrated in a time sequence. The circle around a cell represents the tolerance area of the cell.

an S-cell represents the tolerance area of the S-cell. (1) Suppose there are already two S-cells in the feature space. (2) A training vector is presented. Since it is located outside the tolerance areas of all S-cells, it elicits no response from any S-cell. (3) Hence a new S-cell (reference vector) is generated at the location of the training vector. In other words, the training vector is adopted as the input connections of the generated S-cell. (4) There are three S-cells now. (5) Another training vector is presented. Since it is located in the tolerance area of an S-cell, a response is elicited from the S-cell. (6) Since there is a non-silent S-cell, a new S-cell cannot be generated, and the training vector is ignored. Thus distances between S-cells in the feature space are always kept to be larger than the radius of the tolerance area, which is determined by the threshold θ of the S-cells. In the illustration of Fig. 4 above, the radius of the tolerance area is $\alpha = \cos^{-1}\theta$ in the feature space.

When applying the add-if-silent rule to the neocognitron, a slight modification is required, because each layer of the neocognitron consists of a number of cell-planes. All cells in a cell-plane share the same set of input connections, and this condition of shared connections has to be kept even during learning.

Individual cells in a cell-plane have their receptive fields at different locations in U_0. Here we indicate the location of a cell using the location of its receptive field center. Using this indication, we can say that there are a number of cells (namely, one cell from each cell-plane) at the same location in a layer.

If all feature-extracting S-cells are silent in a certain small area when a training stimulus is presented during learning, a new S-cell is generated and added to the network. In the neocognitron, generation of a new S-cell means a generation of a new cell-plane. The newly added S-cell learns this training stimulus, and all other cells in the cell-plane come to have the same set of input connections as the added S-cell.

We now explain the rule in more detail. If there is any location where all post-synaptic S-cells within a certain distance (D_l) around it are silent, the location becomes a candidate at which a new S-cell can be generated. Incidentally, an area of radius D_l has a shape of a hypercolumn in a layer (Fukushima, 1980, 1988). Among these candidate locations, we search the location where the sum of the responses of the presynaptic C-cells is the largest. We then generate a new S-cell at the location, and input connections to the S-cell are determined in proportion to the responses of the individual presynaptic C-cells. In other words, the new S-cell learns the training stimulus presented to it. Generation of a new S-cell means a generation of a new cell-plane. The new S-cell takes the place of the seed-cell of the new cell-plane, and all S-cells in the cell-plane follows the seed-cell and come to have the same set of input connections as the seed-cell.

After the generation of the new cell-plane, if there still remains any location where responses of the post-synaptic cells are silent in spite of non-silent inputs, the same process of generating cell-planes is repeated until the whole area of the layer is covered by non-silent S-cells.

After that, we proceed to the presentation of the next training stimulus.

3.2. Dual threshold for S-cells

The ability to recognize patterns robustly is influenced by the selectivity of feature-extracting S-cells, which is controlled by the threshold of the S-cells.

For S-cells of intermediate layers (U_{S2} and U_{S3}) of the neocognitron, we use *dual threshold*. Namely, we use a high threshold for S-cells during learning, and a low threshold during recognition (Fukushima & Tanigawa, 1996).

As shown in (5) above, a feature-extracting S-cell is active when $s > \theta$, and it is silent when $s \leq \theta$. Namely, in the feature space shown in Fig. 4, the S-cell is active if and only if the distance $\alpha = \cos^{-1} s$ between the reference vector and the training vector is smaller than $\cos^{-1}\theta$. Under the add-if-silent rule, new S-cells cannot be generated if a training stimulus elicits a response from any S-cell. Hence, after learning, during which a sufficient number of training vectors have been presented, S-cells come to distribute uniformly in the feature space, and each stimulus comes to excite only one (or at most a small number of) S-cell. This means that S-cells come to behave like grandmother cells in the layer.

Since a new S-cell is generated if all S-cells are silent, the distance α between adjacent reference vectors becomes, not smaller than, but approximately equal to, $\cos^{-1}\theta$. This guarantees creating a situation where S-cells distribute uniformly in the multi-dimensional vector space. In order to produce a sufficient number of feature-extracting S-cells in a layer, the threshold θ of S-cells needs to be high enough during learning.

For recognizing deformed patterns robustly, however, a behavior like grandmother cells is not desirable for S-cells of a layer. If the threshold is high, each stimulus feature might elicit a response from only one S-cell. When the stimulus feature is slightly deformed, the S-cell stops responding, and another S-cell comes to respond instead of the first one. This decreases the ability of the network to recognize deformed patterns robustly.

Fig. 7 illustrates this situation in the multi-dimensional feature space. The dotted circle around a stimulus vector shows the size of the tolerance areas of S-cells. In other words, S-cells that are located within the dotted circle can respond to the stimulus vector.

Fig. 7(a) illustrates a situation where the threshold of S-cells is as high as in the learning phase. Since the size of the tolerance area is small, only one (or at most a small number of) S-cells can

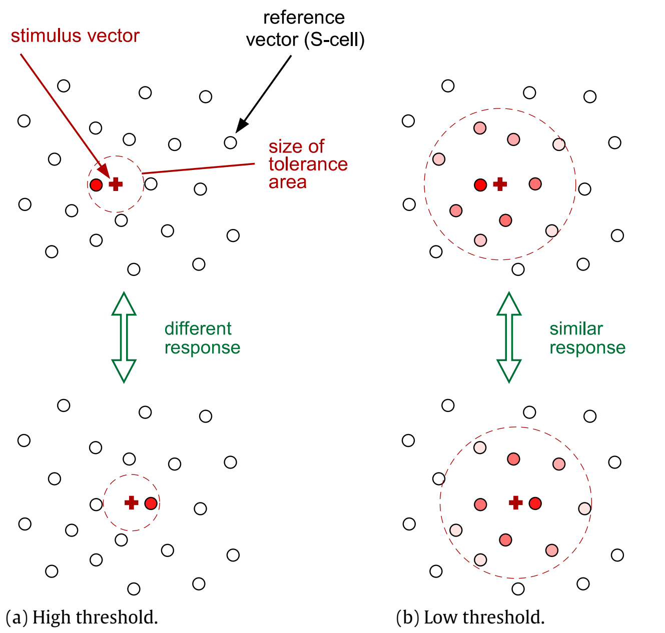

Fig. 7. Response of S-cells in the multi-dimensional feature space during recognition. The dotted circle around a stimulus vector shows the size of the tolerance areas of S-cells. (a) When the threshold of S-cells is as high as in the learning phase, only one S-cell can respond to a stimulus vector. A slight shift of the stimulus vector produces a completely different response of the layer. (b) When the threshold is low, many S-cells respond to a stimulus vector. Even if the stimulus vector shifts a little, most of the S-cells keep responding, and the response of the entire layer does not change so much.

responds to a stimulus vector. It should be noted here that, in the learning phase, we have managed to produce this situation, where each training vector elicits a response from only one S-cell.

If, as a result of deformation of the input image, the stimulus vector shifts slightly, the S-cell stops responding and another S-cell comes to respond instead. A slight shift of the stimulus vector thus produces completely different responses, which have no similarity to each other. Hence the feature-extracting cells in the succeeding layer cannot judge that they are similar responses.

If the threshold is low, however, S-cells respond even to features somewhat deformed from their reference vectors. This makes a situation like a population coding of features rather than grandmother cell theory: many S-cells respond to a single feature if the response of an entire layer is observed. In other words, each stimulus elicits non-zero responses from a number of S-cells, and these S-cells jointly represent the stimulus. Even if the feature is slightly deformed, many of the S-cells still keep responding. Only a small number of S-cells change their responses (Fig. 7(b)). This situation of low threshold in the recognition phase usually endows the network with an ability of generalization and produces a better recognition rate of the neocognitron.

Hence we use the dual threshold of S-cells for the learning and the recognition phases (Fukushima, 2010; Fukushima & Tanigawa, 1996). In the recognition phase after having finished the learning, the threshold of S-cells is set to a lower value θ^{R} than the threshold θ^{L} for the learning.

The use of a dual threshold can also be interpreted in another way. In this interpretation, we assume that S-cells take always the same threshold θ^{R} both during learning and recognition. During learning, a new S-cell is generated, if there is no S-cell whose response is larger than $(\theta^{\mathrm{L}}-\theta^{\mathrm{R}})/(1-\theta^{\mathrm{R}})\cdot\|\boldsymbol{x}\|$. Since the response of the V-cell v is equal to $\|\boldsymbol{x}\|$, this condition can be rephrased that a new S-cell is generated, if there is no S-cell whose response u is larger than $(\theta^{\mathrm{L}}-\theta^{\mathrm{R}})/(1-\theta^{\mathrm{R}})\cdot v$.

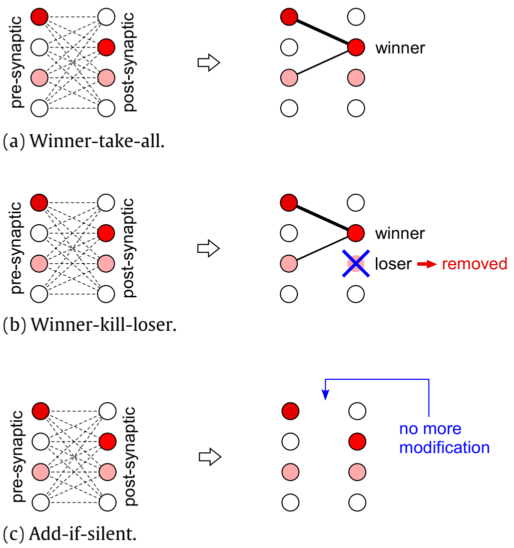

Fig. 8. Several learning rules in comparison with the add-if-silent rule. The figure shows how the cells that have already been generated have their input connections modified under different learning rules. Under the add-if-silent rule, the connections to a post-synaptic cell are not modified any more after the generation of the cell. In this figure, the response of each cell is represented by the depth of the color.

3.3. Comparison with other learning rules

Several methods, other than add-if-silent, have ever been proposed and tried to be used in the neocognitron to train intermediate layers of S-cells. Most of them use unsupervised competitive learning.

The process of generating new S-cells for these competitive learning rules is the same as that for the add-if-silent rule. What is different from the add-if-silent rule is the process of modifying input connections to S-cells after their generation.

Fig. 8 illustrates and compares several rules for unsupervised learning. The figure shows how the cells that have already been generated have their input connections modified under various learning rules. Different from the add-if-silent rule shown in Fig. 8(c), under other learning rules shown in Fig. 8(a) and (b), input connections to S-cells continue to be modified after their generation throughout the learning phase.

Under the winner-take-all rule, shown in Fig. 8(a), post-synaptic cells compete each other, and the cell from which the largest response is elicited becomes the winner. Only the winner can have its input connections strengthened. The amount of strengthening of a connection to the winner is proportional to the response of the presynaptic cell from which the connection is leading. Incidentally, most of the conventional neocognitrons (Fukushima, 1980, 2003) use this learning rule.

The winner-kill-loser rule (Fukushima, 2010; Fukushima et al., 2011), shown in Fig. 8(b), resembles the winner-take-all rule in the sense that only the winner learns the training stimulus. In the winner-kill-loser rule, however, not only does the winner learn the training stimulus, but also the losers are simultaneously removed from the network. Losers are defined as cells whose responses to the training stimulus are smaller than that of the winner, but whose activations are nevertheless greater than zero. The idea behind this learning rule is as follows: If a training stimulus elicits non-zero responses from two or more S-cells, it means that the preferred features of these cells resemble each other, and that they work redundantly in the network. To reduce this redundancy, only the winner has its input connections renewed to fit more to the training vector, while the other active cells, namely the losers, are removed from the network.

Under the add-if-silent rule, shown in Fig. 8(c), input connections to a cell are created only at the moment when the cell is generated. Once the cell has been generated and added to the network, its input connections are not modified any more afterward, whatever training stimuli are given to the network.

3.4. Merits of add-if-silent rule

One of the largest merits of the add-if-silent rule is the simplicity of its algorithm. As a result, computational cost during learning can be made much smaller than other learning rules: once an S-cell is generated and added to the network, we need not train it anymore. When designing a neocognitron, we can remove several parameters that control the training after generation of S-cells. This makes designing neocognitron much easier.

Another advantage of the add-if-silent rule is that the process of learning progresses more stably. With the winner-kill-loser rule (Fukushima, 2010; Fukushima et al., 2011), which produces a higher recognition rate than the winner-take-all rule, the number of cell-planes continues to increase and decrease throughout the learning period and does not completely stabilize. With the add-if-silent rule, the number of cell-planes just increases monotonically, and the increase stops when the multi-dimensional feature space have been covered with reference vectors.

As can be seen from the computer simulation discussed below in Section 5.1, the recognition rate of the neocognitron trained by the add-if-silent rule is almost the same as that of the neocognitron trained by other learning rules (see Fig. 13 below). The number of cell-planes generated in the network is almost the same or a little smaller, which helps the reduction of computational cost (see Figs. 14 and 15 below).

We can conclude from this that, by the use of the add-if-silent rule, the learning algorithm can be simplified, and the computational cost for learning is reduced. Nonetheless, a high recognition rate can be kept without increasing the scale of the network.

We now discuss why a good recognition rate can be obtained with a simpler algorithm of the add-if-silent rule.

The process of generating new cells is the same, both under the add-if-silent rule and under the conventional competitive learning rules (winner-take-all and winner-kill-loser). What is different from them is the learning process performed after the generation of individual cells. Under the add-if-silent rule, once an S-cell is generated and added to the network, the input connections to the S-cell need not change any more.

On the contrary, under the conventional competitive learning rules, an S-cell continues to learn training stimuli that are presented after the generation of the S-cell. This means that S-cells keep learning so as make their reference vectors fit to the training vectors more accurately.

The final classification of input patterns, however, is not made by an intermediate layer, but by the highest stage of the network. The role of the intermediate layer is to represent an input pattern accurately, not by the response of a single cell, but by the population coding, where the pattern given to the input layer is represented by the response of a number of cells. In the case of population coding, best-fitting of individual cells to training stimuli is not necessarily important. It is enough if the input stimulus can be accurately represented by the response of the population of the whole cells in the layer.

4. Interpolating-vector

In the highest stage of the network, the method of the *interpolating-vector* is used for analyzing the response of U_{S4} to classify input patterns.

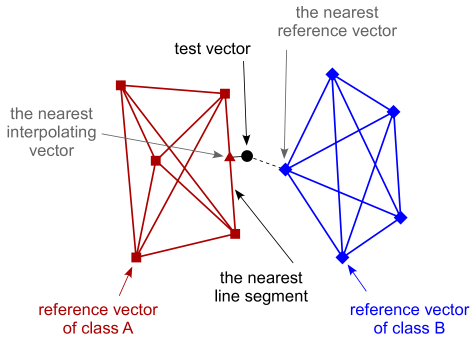
the nearest reference vector
test vector
the nearest interpolating vector
the nearest line segment
reference vector of class A
reference vector of class B

Fig. 9. Recognition by the method of the interpolating-vector. The label of the nearest line segment, instead of the nearest reference vector, shows the result of the pattern recognition.

In the original paper on the method of the interpolating-vector (Fukushima, 2007), we proposed to use interpolating vectors only during recognition, and to use a conventional supervised competitive learning to train S-cells of U_{S4}. In contrast to the previous neocognitron, the present neocognitron uses interpolating vectors, not only in the recognition phase, but also in the learning phase.

In both learning and recognition phases, the response of C-cells of U_{C3} become input signals to S-cells of layer U_{S4}. Here we use vector notation to represent these input signals. During learning, the response of C-cells works as a training vector. Each training vector has a label indicating the class to which the vector belongs. When an S-cell is initially generated and learns a training vector, the S-cell (hence its reference vector) is assigned the label of the training vector. The purpose of the learning is to produce a small set of reference vectors (S-cells) so as to represent the larger set of training vectors. The reference vectors work as representatives of the training vectors.

Before explaining the method of creating reference vectors of S-cells, we first discuss the recognition phase, where reference vectors have already been produced. The process of creating reference vectors is discussed later in Section 4.2.

4.1. Classification by interpolating-vector

After having finished the learning, by which all reference vectors have been produced, we use the method of the interpolating-vector to recognize patterns more robustly. Each stimulus (namely, the response of presynaptic C-cells) that is to be classified is called a test vector here.

The basic idea of the method of the interpolating-vector is as follows. We assume a situation where virtual vectors, which are named *interpolating vectors*, are densely placed along the line segments connecting every pair of reference vectors of the same label. From these interpolating vectors, we choose the one that has the largest similarity to the test vector. The label (or the class name) of the chosen vector is taken as the result of pattern recognition.

Actually, we do not need to generate an infinite number of interpolating vectors. We just assume line segments connecting every pair of reference vectors of the same label. The line segments are assigned the same labels as the reference vectors on both sides. We then measure distances (based on similarity) to these line segments from the test vector, and choose the nearest line segment (Fig. 9). The label of the line segment, instead of the nearest reference vector, shows the result of the pattern recognition.

Mathematically, this process can be expressed as follows. Let $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ be two reference vectors of the same label. An interpolating vector $\boldsymbol{\xi}$ for this pair of reference vectors is given by a linear combination of them:

$$\boldsymbol{\xi} = p_i \frac{\boldsymbol{X}_i}{\|\boldsymbol{X}_i\|} + p_j \frac{\boldsymbol{X}_j}{\|\boldsymbol{X}_j\|}, \quad (p_i + p_j = 1). \tag{7}$$

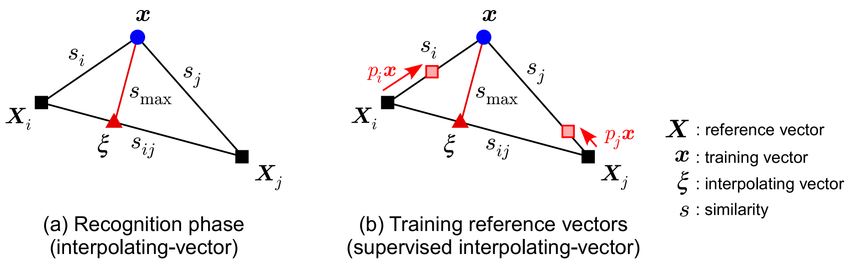
$\boldsymbol{x}$
s_i
s_j
$s_{\max}$
$\boldsymbol{X}_i$
ξ
s_{ij}
$\boldsymbol{X}_j$
(a) Recognition phase (interpolating-vector)
$p_i\boldsymbol{x}$
$p_j\boldsymbol{x}$
(b) Training reference vectors (supervised interpolating-vector)
$\boldsymbol{X}$: reference vector
$\boldsymbol{x}$: training vector
ξ : interpolating vector
s : similarity

Fig. 10. The method of the interpolating-vector.

Similarity s between the interpolating vector $\boldsymbol{\xi}$ and a test vector $\boldsymbol{x}$ is

$$s = \frac{(\boldsymbol{\xi}, \boldsymbol{x})}{\|\boldsymbol{\xi}\| \cdot \|\boldsymbol{x}\|} = \frac{p_i s_i + p_j s_j}{\sqrt{p_i^2 + 2p_i p_j s_{ij} + p_j^2}}, \tag{8}$$

where

$$s_i = \frac{(\boldsymbol{X}_i, \boldsymbol{x})}{\|\boldsymbol{X}_i\| \cdot \|\boldsymbol{x}\|}, \qquad s_j = \frac{(\boldsymbol{X}_j, \boldsymbol{x})}{\|\boldsymbol{X}_j\| \cdot \|\boldsymbol{x}\|}, \qquad s_{ij} = \frac{(\boldsymbol{X}_i, \boldsymbol{X}_j)}{\|\boldsymbol{X}_i\| \cdot \|\boldsymbol{X}_j\|}. \tag{9}$$

Among various combinations of p_i and p_j, similarity s takes a maximum value

$$s_{\max} = \sqrt{\frac{s_i^2 - 2s_i s_j s_{ij} + s_j^2}{1 - s_{ij}^2}} \tag{10}$$

at

$$p_i = \frac{s_i - s_j s_{ij}}{(s_i + s_j)(1 - s_{ij})}, \qquad p_j = \frac{s_j - s_i s_{ij}}{(s_i + s_j)(1 - s_{ij})}. \tag{11}$$

Since threshold θ of S-cells is always zero in layer U_{S4}, we can see from (5) that s_i and s_j are proportional to the responses of S-cells whose reference vectors are $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, respectively. To be more specific, the response of the ith and jth S-cells are given by

$$u_i = \|\boldsymbol{x}\| \cdot \frac{(\boldsymbol{X}_i, \boldsymbol{x})}{\|\boldsymbol{X}_i\| \cdot \|\boldsymbol{x}\|} = \|\boldsymbol{x}\| \cdot s_i,$$

$$u_j = \|\boldsymbol{x}\| \cdot \frac{(\boldsymbol{X}_j, \boldsymbol{x})}{\|\boldsymbol{X}_j\| \cdot \|\boldsymbol{x}\|} = \|\boldsymbol{x}\| \cdot s_j. \tag{12}$$

Hence, if we have calculated s_{ij} in advance, we can easily get $\|\boldsymbol{x}\| \cdot s_{\max}$ from the responses of S-cells, u_i and u_j, by (10). Namely,

$$\|\boldsymbol{x}\| \cdot s_{\max} = \sqrt{\frac{u_i^2 - 2s_{ij} u_i u_j + u_j^2}{1 - s_{ij}^2}}. \tag{13}$$

Since the value of $\|\boldsymbol{x}\|$ is the same for all S-cells that have receptive fields at the same location, we can easily find the pair of S-cells (reference vectors) that have the maximum similarity $s_{\max}$.

We can interpret that $s_{\max}$ represents similarity between test vector $\boldsymbol{x}$ and the line segment that connects a pair of reference vectors $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ (Fig. 10(a)). Among all line segments that connect every pairs of reference vectors of the same label, we choose the one that has the largest similarity to the test vector. The label (or the class name) of the chosen line segment is taken as the result of pattern recognition. (If there exists only one reference vector of a class, the similarity between the reference vector and the test vector is taken as $s_{\max}$ for the class.)

4.2. Creating reference vectors

Every time when a training vector is presented during learning, we first try to classify it using the method of the interpolating-vector.

If the result of the classification is wrong, or if all S-cells are silent, the training vector is adopted as a new reference vector and is assigned a label of the class name.

If the result of the classification is correct, we choose the line segment that shows the largest similarity to the training vector $\boldsymbol{x}$. The two reference vectors, $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, on both sides of the line segment learn the training vector $\boldsymbol{x}$ (Fig. 10(b)). The amounts of increase of $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ are $p_i\boldsymbol{x}$ and $p_j\boldsymbol{x}$, respectively, where p_i and p_j are given by (11).[1]

We call this training method a *supervised interpolating-vector*. Incidentally, in the previous version of the interpolating-vector (Fukushima, 2007), the method of the supervised winner-take-all was used in the learning phase for creating reference vectors, and the interpolating-vector was used only in the recognition phase.

4.3. Application to the neocognitron

When applying the interpolating-vector to layer U_{S4} of the neocognitron, small modifications are required. In the neocognitron, layer U_{S4}, like other layers, consists of a number of cell-planes. All S-cells in a cell-plane share the same set of input connections, but they have receptive fields at different locations. To keep shared connections during learning, we use seed-cells. Different from other layers, each cell-plane of U_{S4} has a label indicating the class to which it belongs. This means that all S-cells in a cell-plane have the same label.

In the recognition phase, we first look at a group of S-cells whose receptive fields are at the same location. This means that the group contains one S-cell from each cell-plane. We search, from every group, a pair of S-cells that yields the maximum value of $\|\boldsymbol{x}\| \cdot s_{\max}$. We then choose the largest $\|\boldsymbol{x}\| \cdot s_{\max}$ among all groups. The label of the pair of S-cells that yields the largest $\|\boldsymbol{x}\| \cdot s_{\max}$ is the result of pattern recognition by the neocognitron.

In the learning phase (under the supervised interpolating-vector), if the recognition result is correct, the pair of the S-cells takes the place of seed cells. This means that two cell-planes of U_{S4} are modified at a time.

If the result of recognition is wrong, or all S-cells are silent in U_{S4}, a new cell-plane is generated and is assigned the label of the training pattern. The seed-cell for the newly generated cell-plane is chosen at the location where a weighted sum of the input signals is the largest, where the spatial distribution of the weight has a shape like a dome.

5. Computer simulation

We test the behavior of the neocognitron by computer simulation using handwritten digits (free writing) randomly sampled from the ETL1 database.[2] Incidentally, the ETL1 is a large database of segmented handwritten characters written by about 1400 different writers. The scale of the network and the parameters for the simulation are shown in Appendix A.

[1] If $p_i < 0$, namely, $p_j > 1$, we set $p_i = 0$ and $p_j = 1$. Similarly, if $p_i > 1$, namely, $p_j < 0$, we set $p_i = 1$ and $p_j = 0$.

[2] http://www.is.aist.go.jp/etlcdb/#English.

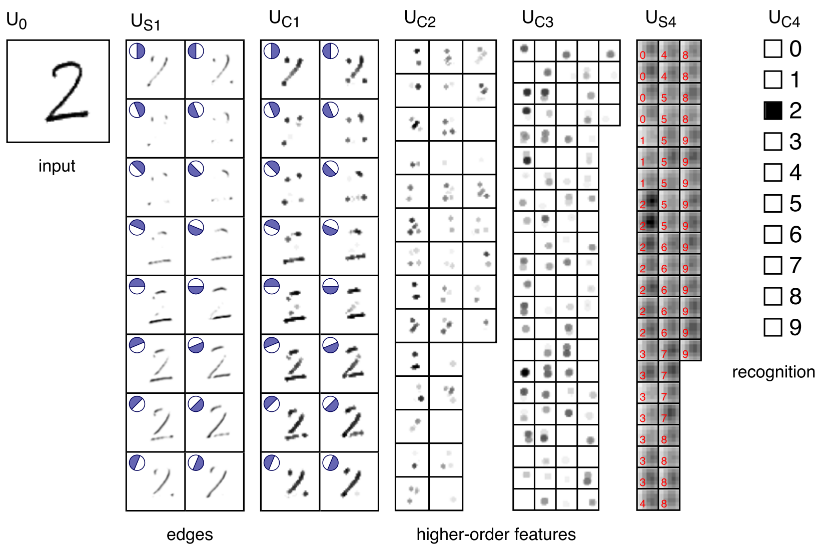

Fig. 11. An example of the response of the new neocognitron. The half disk drawn in each cell-plane of U_{S1} and U_{C1} shows the orientation of the edge extracted by the cell-plane. The rightmost layer, U_{C4}, shows that the input pattern is recognized correctly as '2'.

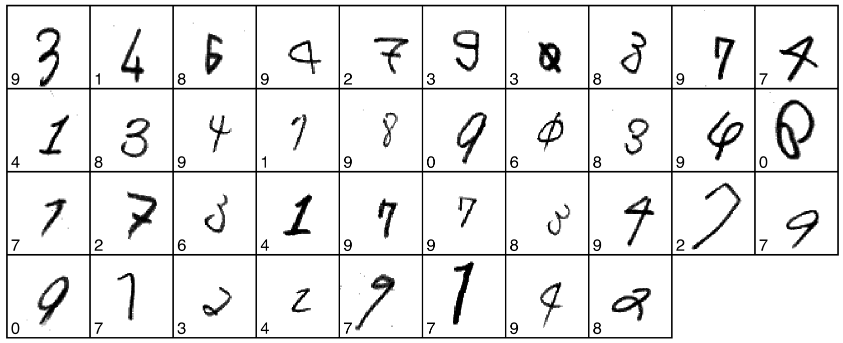

Fig. 12. Erroneously recognized patterns among randomly sampled 5000 blind test patterns. The digit in the lower left of each pattern indicates how the pattern was recognized.

Fig. 11 shows a typical response of the network that has finished learning. For the learning, we used a training set consisting of 3000 patterns (300 patterns for each digit) randomly sampled from the ETL1 database. The training set is presented to the neocognitron only once for training each layer of intermediate stages (U_{S2} and U_{S3}), and twice for training U_{S4}. The responses of all layers in the network, except U_{S2} and U_{S3}, are displayed in series from left to right in the figure. The leftmost layer, U_0, is the input layer. The rightmost layer, U_{C4}, shows the final result of recognition. The numerals drawn in layer U_{S4} indicate the labels (class names) assigned to individual cell-planes. In this example, it is seen that input pattern '2', which is presented to U_0, is recognized correctly.

Incidentally, Fig. 12 shows erroneously recognized patterns by this network among randomly sampled 5000 blind test patterns. (See below for more information on the experimental condition.)

5.1. *Comparison of learning rules for intermediate layers*

To show how the add-if-silent rule is effective for the learning of intermediate layers (U_{S2} and U_{S3}), we test the performance of the neocognitron by computer simulation under different learning rules. The learning rules compared are: (a) add-if-silent (red solid line in Figs. 13–15), which is proposed in this paper; (b) winner-kill-loser with triple threshold (blue dashed line) (Fukushima et al., 2011); (c) winner-take-all (dark-green dotted line), which is used in the conventional neocognitrons (Fukushima, 1980, 2003). In these simulations, the method of the interpolating-vector is used for the highest layer U_{S4} in both learning and recognition phases. (Namely, the supervised interpolating-vector for learning and the interpolating-vector for recognition.)

Optimal values of thresholds of S-cells differ under different learning rules. We then searched optimal thresholds, which produces the highest recognition rate for individual learning rules, by changing the four threshold values (θ_2^L, θ_2^R, θ_3^L and θ_3^R), where we used a training set of 3000 patterns and a test set of 5000 patterns. In the search, we used the hill climbing method. The figures below show the results under the threshold values thus obtained.

Fig. 13 shows how the error rate of the neocognitron changes with the size of the training set. Since no rejection is allowed here in the recognition, recognition rate is equal to (100% − (error rate)). The training set is presented only once for the training of each layer of intermediate stages (U_{S2} and U_{S3}), and twice for the training of U_{S4}. The test set consists of 5000 patterns (500 patterns for each digit) for all sizes of training set. We made this experiment seven times for each condition, using different learning and test sets randomly sampled from the ETL1 database, and averaged the results of the seven experiments. The error bars in the figure represent standard deviation. We can see from the figure that, in most of the cases, the recognition error does not differ so much by the learning rules for intermediate layers, although the add-if-silent and winner-kill-loser rules produce a slightly better recognition rate than the winner-take-all rule.

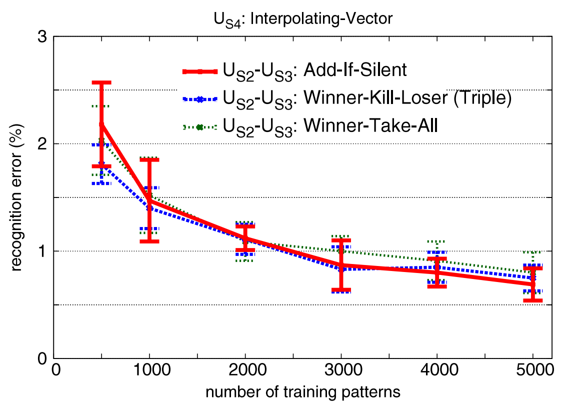

Fig. 13. Recognition error vs. number of training patterns. Comparison of the training method for U$_{S2}$ and U$_{S3}$: the add-if-silent (red solid line), the winner-kill-loser with triple threshold (blue dashed line), and the winner-take-all (dark-green dotted line). We made the simulation seven times for each condition, using different learning and test sets randomly sampled from the ETL1 database, and averaged the results of the seven experiments. The error bars in the figure represent standard deviation.

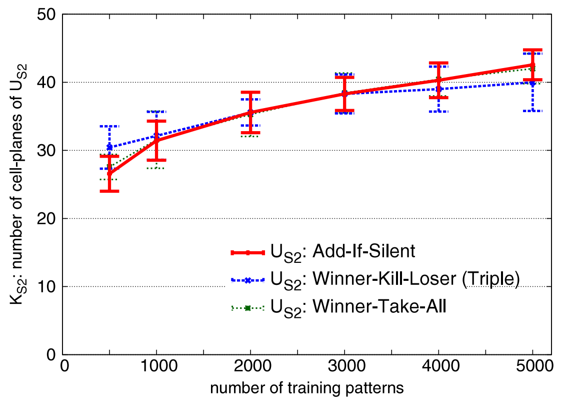

Fig. 14. K_{S2}, the final number of cell-planes generated in U$_{S2}$, vs. the number of training patterns. Comparison between different training rules for U$_{S2}$.

Figs. 14 and 15 show the final number of cell-planes, K_{Sl} ($l = 2$ and 3), that is generated in U$_{Sl}$ after having finished the learning with each size of training sets. They show the cases where intermediate layers, U$_{S2}$ and U$_{S3}$, are trained with the add-if-silent (red solid line), the winner-kill-loser with triple threshold (blue dashed line), and the winner-take-all (dark-green dotted line). Although the number of cell-planes K_{S3} is smaller under the add-if-silent rule than other learning rules, K_{S2} does not differ so much under different learning rules.[3] We can see from these experiments that the add-if-silent rule can produce a high recognition rate without increasing the scale of the network. Incidentally, the computational cost for calculating the response of U$_{Sl}$ is approximately proportional to $K_{Sl-1} \times K_{Sl}$.

5.2. Comparison of the learning and recognition methods for the highest stage

We now test how the method of the interpolating-vector is effective for the learning and recognition in the highest stage U$_{S4}$.

[3] In a previous experiment (Fukushima et al., 2011), which used the winner-kill-loser (triple threshold) for intermediate layers and the supervised winner-take-all for the highest stage, we had a smaller K_{S3} than in the present simulation. In the previous experiment, however, we choose thresholds of S-cells so as to minimize the recognition error when we use the winner-kill-loser (triple threshold) for intermediate layers and the supervised winner-take-all for the highest stage. In contrast to it, the blue dashed lines in Figs. 13–15 show the results obtained by a network, for which thresholds of S-cells are determined so as to minimize the recognition error when we use the interpolating-vector for the highest stage.

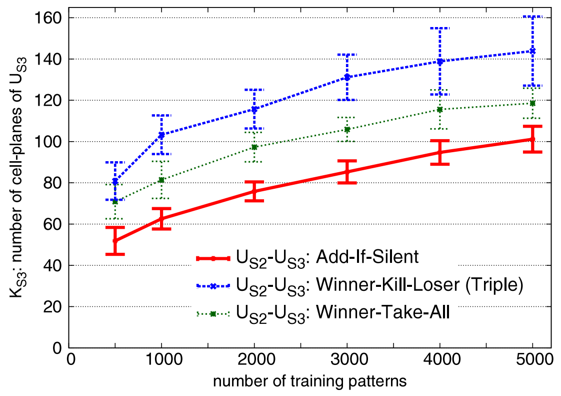

Fig. 15. K_{S3}, the final number of cell-planes generated in U$_{S3}$, vs. the number of training patterns. Comparison between different training rules for U$_{S2}$–U$_{S3}$.

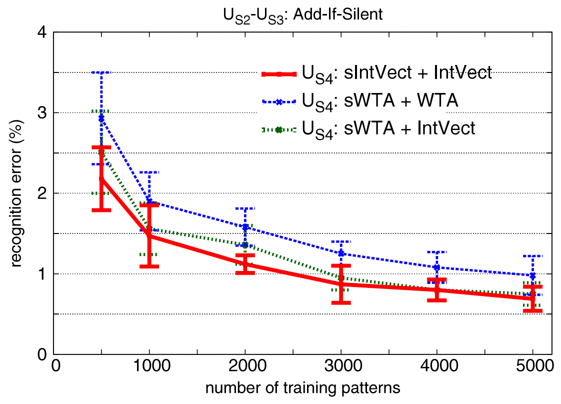

Fig. 16. Recognition error vs. the number of training patterns under different methods of learning and recognition for U$_{S4}$. The red solid line (sIntVect + IntVect) shows the case of learning by the supervised interpolating-vector and recognition by the interpolating-vector. The blue dashed line (sWTA + WTA) shows the case of learning by the supervised winner-take-all and recognition by the winner-take-all. The dark-green dotted line (sWTA + IntVect) shows the case of learning by the supervised winner-take-all and recognition by the interpolating-vector. In all cases, the add-if-silent rule is used for the learning of layers U$_{S2}$ and U$_{S3}$.

Fig. 16 shows how the recognition error changes with the size of training set under different learning and recognition methods for U$_{S4}$.

The red solid line (sIntVect + IntVect) shows the case where the supervised interpolating-vector is used for learning and the interpolating-vector for recognition. The blue dashed line (sWTA + WTA) shows the case where the supervised winner-take-all is used for learning and the winner-take-all is used for recognition. The dark-green dotted line (sWTA + IntVect) shows the case where the supervised winner-take-all is used for learning and the interpolating-vector is used for recognition. In all cases, the add-if-silent rule is used for the learning of layers U$_{S2}$ and U$_{S3}$. Similar to other simulations discussed above, the training set is presented only once for the training of each layer of intermediate stages (U$_{S2}$ and U$_{S3}$), and twice for the training of U$_{S4}$. Each curve shows the averaged result of seven trials.

We can see from this figure that the recognition error can be made much smaller by the interpolating-vector (sIntVect + IntVect) than by the winner-take-all (sWTA + WTA). Incidentally, in most of the conventional neocognitrons (Fukushima, 1980, 2003, 2010, 2011; Fukushima et al., 2011), the supervised winner-take-all was used for training U$_{S4}$.

Fig. 16 also shows the case where the method of the interpolating-vector is used only for recognition and the supervised winner-take-all is used for training (sWTA + IntVect). Incidentally, this is the case of the neocognitron of a previous version that used the interpolating-vector (Fukushima, 2007). The

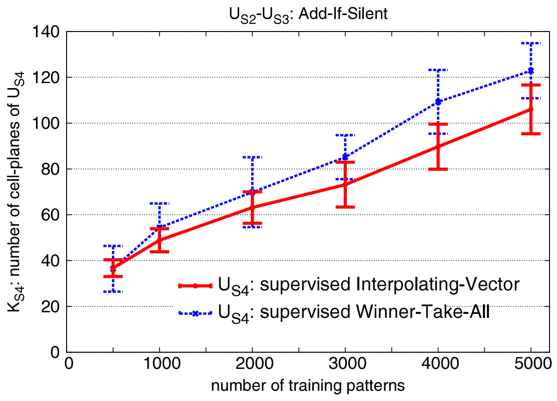

Fig. 17. K_{S4}, the final number of cell-planes generated in U_{S4}, vs. the number of training patterns. Comparison between supervised interpolating-vector (red solid line) and supervised winner-take-all (blue dashed line) for the learning.

recognition error is slightly large but not so bad compared to the case where the method of the interpolating-vector is used for both learning and recognition (sIntVect + IntVect). This shows that the good recognition rate largely owes the use of the interpolating-vector in the recognition phase. As shown in Fig. 17 below, the use of the supervised interpolating-vector in the learning mainly contributes to reducing the number of generated cell-planes, K_{S4}.

Although Fig. 16 only shows the result, where the add-if-silent rule is used for the learning of layers U_{S2} and U_{S3}, we also confirmed that the recognition error is always smaller by the interpolating-vector than by the winner-take-all, regardless of the training method for the intermediate layers.

Fig. 17 shows how K_{S4}, the number of cell-planes generated in layer U_{S4}, changes with the number of training patterns under different learning methods. We can see from the figure that K_{S4} can be made smaller by the supervised interpolating-vector than by the supervised winner-take-all. It should be noted here that, in layer U_{S4}, most of the computational cost in the recognition is spent for calculating the response of S-cells, and the cost for classifying patterns from the response of the S-cells is relatively small. Hence we can roughly say that the computational cost is nearly proportional to K_{S4}, regardless of the recognition method.

We can conclude that the improvement of recognition rate owes largely to the use of the interpolating-vector in the recognition, while learning by the supervised interpolating-vector contributes to a reduction of the network scale.

6. Discussion

This paper proposed new learning rules suited for training multi-layered neural networks and applied them to the *neocognitron*. For training intermediate layers of the hierarchical network, we used a new learning rule named the *add-if-silent*. For the highest stage of the network, we used the method of the *interpolating-vector*, not only during recognition, but also during learning.

Computer simulation has demonstrated that the new neocognitron produces a higher recognition rate with a smaller scale of the network than the neocognitron of previous versions.

By the use of the add-if-silent rule, the learning algorithm is considerably simplified, and the computational cost for learning has been reduced. Parameters required in designing the network can also be reduced. Nonetheless, we can have almost the same or a slightly better recognition rate by the add-if-silent rule without increasing the scale of the network.

The add-if-silent rule somewhat resembles the model proposed by Serre, Oliva, and Poggio (2007), in the sense that the response of the preceding layer to a certain training pattern is imprinted as a feature to be extracted by a cell (namely, as the reference vector of the cell). Still there is a significant difference. In their model, features to be imprinted are sampled just randomly without any criteria. In contrast to their model, the add-if-silent rule guarantees that the features are chosen keeping a certain minimum distance, which is determined by the threshold of the cells. Namely, in the feature space shown in Fig. 4, the distance $\alpha = \cos^{-1} s$ between imprinted features is always kept to be $\alpha > \cos^{-1} \theta$.

To examine the ability of the method by Serre et al. (2007), we simulated a neocognitron, in which only the training method of intermediate layers are replaced from the add-if-silent rule to their method. Here, this neocognitron is called *their* network, while the neocognitron discussed in this paper is called *our* network. The difference between their and our networks resides only in the training method for intermediate layers. Similarly to our network, training of their network is performed from lower stages to higher stages. The method of the interpolating-vector was used for the highest stage both during learning and recognition. The thresholds of S-cells during recognition were set to the same values as those for our network.

To train intermediate layers of their network, we decided K_{Sl}, the number of cell-planes of U_{Sl} ($l = 2$ and 3), in advance. We then chose a training set of 3000 patterns randomly sampled from the ETL1 database. For training U_{Sl}, we chose K_{Sl} patterns at random from the training set of 3000 patterns and presented them one by one to U_0. For each of the K_{Sl} patterns, a new cell-plane is generated in U_{Sl}, where the location of the seed-cell was chosen at random from the places where activities of presynaptic cells were not below a certain level. After the training of U_{S2} and U_{S3} had been completely finished, we performed the training of the highest stage using the training set of 3000 patterns. We then tested the behavior of the network, using a blind test set of 5000 patterns. We performed this experiment seven times, using different learning and test sets randomly sampled from the ETL1 database.

As shown in Figs. 14 and 15, the average numbers of cell-planes generated in *our* network were $K_{S2} \approx 38$ and $K_{S3} \approx 85$, when it was trained with a training set of 3000 patterns. We then tested *their* network by setting $K_{S2} = 38$ and $K_{S3} = 85$. The error rate of their network was $2.43 \pm 0.57\%$, and $K_{S4} = 180.4 \pm 37.5$ cell-planes were generated in U_{S4}. The error rate was very large compared with that of our network. Since K_{S4} was large, the computational cost was also large. The large number of K_{S4} shows that important features had not been extracted sufficiently in the intermediate layers.

Thinking of a possibility of insufficient numbers of cell-planes, we then tried to double the number of cell-planes of the intermediate layers, namely, $K_{S2} = 76$ and $K_{S3} = 170$. The error rate was now $2.14 \pm 0.30\%$, and we had $K_{S4} = 159.6 \pm 21.5$. Although they decreased, we still had a large error rate and a large number of K_{S4}.

These results show clearly that the add-if-silent rule can generate feature-extracting cells of intermediate layers much more efficiently than choosing their preferred features at random.

It has been known that the use of the interpolating-vector during recognition contributes largely to the increase in the recognition rate. This paper has proposed to use the interpolating-vector, not only in the recognition phase, but also for training the highest stage. Incidentally, in the neocognitron of a previous version that used the interpolating-vector, the method of the interpolating-vector was used only for classifying patterns in the recognition phase (Fukushima, 2007). By the use of the interpolating-vector also for the learning (namely, the supervised interpolating-vector), we can produce feature extractors that are more suited for classifying by the interpolating-vector. As a result, the number of cell-planes can be reduced in the highest stage. In short, classification by the interpolating-vector greatly increases the recognition rate, and learning by the supervised interpolating-vector contributes to the reduction of the network scale.

Since the use of the add-if-silent rule is very useful for the learning of intermediate layers, we tested if this is the case also for the highest stage. We tried to omit the process of modifying connections to S-cells that have already been generated in the highest stage, but we failed to get a good result. This is probably because, in the highest stage, the response of each S-cell is directly used to classify input patterns, while, in the intermediate layers, it is enough that the response of the whole S-cells, not the response of a single S-cell, represent the input pattern accurately.

It is recommended to use different learning rules for the intermediate and for the highest stages of hierarchical multi-layered network. The add-if-silent rule is useful for the intermediate layers, where the input pattern is represented by population coding. In the highest stage, before which feature extraction from the input pattern has already been finished, the classification by the interpolating-vector becomes a powerful tool.

Acknowledgments

The author thanks Prof. Isao Hayashi (Kansai University), Dr. Hayaru Shouno (University of Electro-Communications) and Dr. Masayuki Kikuchi (Tokyo University of Technology) for stimulating discussions on the neocognitron.

This work was partially supported from Kansai University by Strategic Project to Support the Formation of Research Bases at Private Universities: Matching Fund Subsidy from MEXT, 2008–2012.

Appendix A. Scale of the network

The new neocognitron discussed in this paper is a four-stage network as was shown in Fig. 1.

Fig. A.18 shows a one-dimensional cross-section of connections between cell-planes. Although there are a number of cell-planes in each layer, only one cell-plane is drawn in each layer. The number of connections converging to a cell and the radius of their spatial spread are the same for all cell-planes of a layer.

The figure also shows the total number of cells (not counting inhibitory V-cells) in each layer. The number of cells in each cell-plane is pre-determined for all layers when designing the network.

The reduction in density of cells in cell-planes, namely the thinning-out of the cells, is made from a S-cell layer to the C-cell layer of the same stage. The ratio of the thinning-out from U_{Sl} to U_{Cl} is 2 : 1 (in both horizontal and vertical directions) in all stages except U_{C4}.

Layer U_{S1} consists of edge-extracting cells. The number of its cell-planes is $K_{S1} = 16$, which is determined when designing the network (see Appendix B). For other stages higher than it ($l \geq 2$), the number of cell-planes in each S-cell layer (K_{Sl}) is determined automatically by the learning, depending on the training set, which is chosen by random sampling from the database.

In each stage except the highest one, the same number of cell-planes is generated in the C-cell layer as in the S-cell layer. Namely, $K_{Cl} = K_{Sl}$ ($l \leq 3$). The recognition layer U_{C4}, however, has only $K_{C4} = 10$ cell-planes, which correspond to the 10 digits to be recognized, and each cell-plane contains only one C-cell.

The sizes and the shapes of connections converging to single cells (namely, radiuses A_{Sl}, A_{Cl}, etc.) are shown below in connection with mathematical equations. In the equations, the unit of length is different from layer to layer: namely, the pitch of cells in the cell-plane of the preceding layer is taken as the unit of length. It should be noted here that, if two layers have the same radius based on the pitch of the cells in their respective layers, the actual size of the connections measured with the scale of the input layer is larger in the higher layer, because the density of cells is lower in the higher layer.

Appendix B. Extraction of oriented edges

In the neocognitron proposed in this paper, S-cells of U_{S1} extract oriented edges using linear filters.

S-cells of this layer consist of analog-threshold-elements, which calculates weighted sum of their inputs and cut off the negative components by a threshold operation. They extract oriented edges directly from the response of U_0 by a linear spatial filter. There is a mechanism of weak lateral inhibition among S-cells of different preferred orientations.

These operations can be expressed mathematically as follows. Let $u_{S1}(\boldsymbol{n}, k)$ be the output of an S-cell of the kth cell-plane of layer U_{S1}, where $\boldsymbol{n}$ represents the location of the receptive field center of the cells. The output $u_{S1}(\boldsymbol{n}, k)$ is obtained from $u'_{S1}(\boldsymbol{n}, k)$ by lateral inhibition:

$$u_{S1}(\boldsymbol{n}, k) = \varphi\left[u'_{S1}(\boldsymbol{n}, k) - \frac{2}{K_{S1}} \sum_{\kappa=0}^{K_{S1}-1} u'_{S1}(\boldsymbol{n}, \kappa)\right], \tag{B.1}$$

where

$$u'_{S1}(\boldsymbol{n}, k) = \varphi\left[\sum_{|\boldsymbol{\nu}| \leq A_{S1}} a_{S1}(\boldsymbol{\nu}, k) \cdot u_0(\boldsymbol{n} + \boldsymbol{\nu})\right]. \tag{B.2}$$

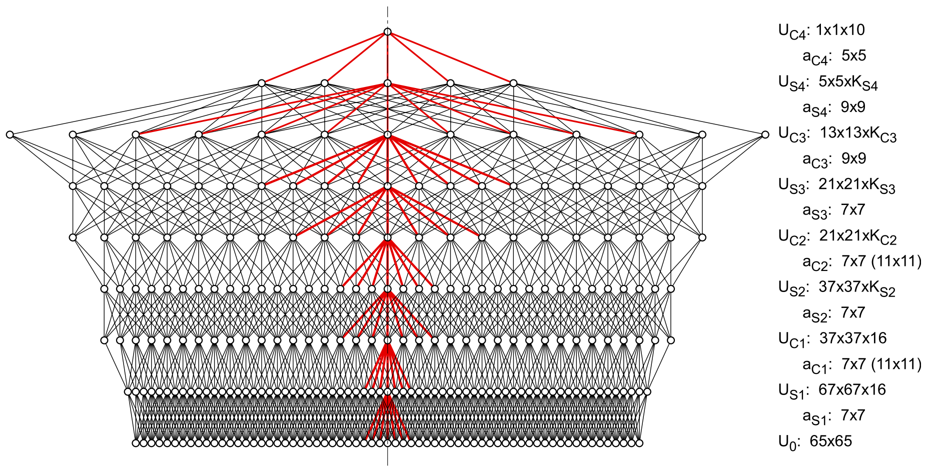

Fig. A.18. Arrangement of cells and connections in the network. A one-dimensional cross-section of connections between cell-planes is drawn. Since the spatial spread of connections converging to a cell is not square but actually is circular, only approximate numbers of connections are shown in the figure. Only positive connections are drawn for a_{C1} and a_{C2}. To clearly show the input connections converging to a single cell, a cell is chosen from each cell-plane, and its input connections are drawn with heavy lines.

Function $\varphi[\ ]$ is defined by $\varphi[x] = \max(x, 0)$. In the computer simulation, we chose $K_{S1} = 16$ and $A_{S1} = 3.5$.

The input connection $a_{S1}(\boldsymbol{\nu}, k)$ to the S-cell is given by

$$a_{S1}(\boldsymbol{\nu}, k) = \left\{\nu_x \cos\left(\frac{2\pi k}{K_{S1}}\right) - \nu_y \sin\left(\frac{2\pi k}{K_{S1}}\right)\right\} \exp\left(-\frac{\|\boldsymbol{\nu}\|^2}{2\sigma^2}\right),$$
$$\boldsymbol{\nu} = (\nu_x, \nu_y), \tag{B.3}$$

where σ is a positive constant, and we chose $\sigma = 0.02$. A proportional constant that determines the scale of the amplitude of $a_{S1}(\boldsymbol{\nu}, k)$ is abbreviated from this equation. It can be any positive value, because the relative strength of the output of the S-cell is normalized in the succeeding C-cell layer. Incidentally, Eq. (B.3) is derived from directional differentiation of a two-dimensional Gaussian curve $G_\sigma(x, y)$. Namely, we can get connection $a_{S1}(\boldsymbol{\nu}, k)$ ($k = 0$) for extracting vertical edges, for example, by sampling the following function at discrete rectangular grids.

$$-\frac{\partial}{\partial x} G_\sigma(x, y) = \frac{x}{2\pi\sigma^4} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \tag{B.4}$$

The shape of this function is shown in Fig. 2 above.

Appendix C. Weighted inner product

In (2) and Fig. 3, the values of c_n, namely the fixed excitatory connection to the V-cell from the nth C-cell, is used as a kind of weight, which compensates the effect of distance in the retinotopic space between the post-synaptic V-cell and a presynaptic C-cell. Namely, it is a monotonically decreasing function of distance between the locations of the receptive fields of pre- and post-synaptic cells. The same weight is applied also to the input connections to the S-cell, to which the V-cell accompanies.

Here we define *weighted* inner product of arbitrary two vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ by

$$(\boldsymbol{y}, \boldsymbol{z}) = \sum_n c_n y_n z_n, \tag{C.1}$$

where the strength of the input connections to the V-cell, c_n, is used as the weight for the inner product. We also define the norm of an arbitrary vector $\boldsymbol{y}$, using the *weighted* inner product, by $\|\boldsymbol{y}\| = \sqrt{(\boldsymbol{y}, \boldsymbol{y})}$. It should be noted here that, if $c_n = 1$, (C.1) reduces to the conventional inner product.

If we use this norm, which is defined by the *weighted* inner product, the response of the V-cell of (2) can be expressed by (3) even in the case of $c_n \neq 1$.

In the case of $c_n \neq 1$, connection a_n is given by

$$a_n = c_n X_n / \|\boldsymbol{X}\| \tag{C.2}$$

instead of (4). However, all discussions in the text, including (3), (5) and (6), hold if we use *weighted* inner product defined above.

Appendix D. Response of S-cells

All S-cells, except the ones in the first stage U_{S1}, have the same characteristics.

Let $u_{Sl}(\boldsymbol{n}, k)$ and $u_{Cl}(\boldsymbol{n}, k)$ be the output of an S-cell and a C-cells of the kth cell-plane of the lth stage, respectively, where $\boldsymbol{n}$ represents the location of the receptive field center of the cells. Layer U_{Sl} ($l \geq 2$) contains not only S-cells but also V-cells, whose output is represented by $v_l(\boldsymbol{n})$. The outputs of S-cells and V-cells are given by

$$u_{Sl}(\boldsymbol{n}, k) = \frac{1}{1 - \theta_l}$$
$$\cdot \varphi\left[\sum_{\kappa=1}^{K_{Cl-1}} \sum_{|\boldsymbol{\nu}| \leq A_{Sl}} a_{Sl}(\boldsymbol{\nu}, \kappa, k) \cdot u_{Cl-1}(\boldsymbol{n} + \boldsymbol{\nu}, \kappa) - \theta_l \cdot v_l(\boldsymbol{n})\right], \tag{D.1}$$

where

$$v_l(\boldsymbol{n}) = \sqrt{\sum_{\kappa=1}^{K_{Cl-1}} \sum_{|\boldsymbol{\nu}| \leq A_{Sl}} c_{Sl}(\boldsymbol{\nu}) \cdot \left\{u_{Cl-1}(\boldsymbol{n} + \boldsymbol{\nu}, \kappa)\right\}^2}. \tag{D.2}$$

Integer values K_{Sl} and K_{Cl-1} represent the number of cell-planes of the layers of S-cells of the lth stage and C-cells of the $(l-1)$th stage, respectively.

Parameter $a_{Sl}(\boldsymbol{\nu}, \kappa, k)$ (≥ 0) is the strength of excitatory connection coming from C-cell $u_{Cl-1}(\boldsymbol{n} + \boldsymbol{\nu}, \kappa)$ of the preceding stage. It should be noted here that all cells in a cell-plane share the same set of input connections, hence $a_{Sl}(\boldsymbol{\nu}, \kappa, k)$ is independent of $\boldsymbol{n}$. A_{Sl} denotes the radius of summation range of $\boldsymbol{\nu}$, that is, the size of spatial spread of input connections to an S-cell.

Parameter $c_{Sl}(\boldsymbol{\nu})$ represents the strength of the fixed excitatory connections to the V-cell, and is a monotonically decreasing function of $|\boldsymbol{\nu}|$. It is also used as a weighting function for training connections $a_{Sl}(\boldsymbol{\nu}, \kappa, k)$, as is shown in (E.5) and (F.4) below.

In the computer simulation, the shape of $c_{Sl}(\boldsymbol{\nu})$ is determined by a Gaussian function, but its value is truncated to zero in the outside of the connecting area of radius A_{Sl}. Namely,

$$c_{Sl}(\boldsymbol{\nu}) = \begin{cases} \gamma_{Sl}^{(|\boldsymbol{\nu}|^2/A_{Sl}^2)} & |\boldsymbol{\nu}| \leq A_{Sl} \\ 0 & |\boldsymbol{\nu}| > A_{Sl}. \end{cases} \tag{D.3}$$

The radius of input connections to S- and V-cells is: $A_{S2} = A_{S3} = 3.5$ and $A_{S4} = 4.5$. The parameter that determines the deviation of the Gaussian function is $\gamma_{Sl} = 0.7$ for $l = 2, 3, 4$.

The positive constant θ_l is the threshold of the S-cell. It determines the selectivity in extracting features (See Section 2.3). In the computer simulation, thresholds θ_l^{R} for the recognition phase are: $\theta_2^{\mathrm{R}} = 0.45$, $\theta_3^{\mathrm{R}} = 0.40$ and $\theta_4^{\mathrm{R}} = 0.0$. Thresholds θ_l^{L} for the learning phase are: $\theta_2^{\mathrm{L}} = 0.58$, $\theta_3^{\mathrm{L}} = 0.53$ and $\theta_4^{\mathrm{L}} = 0.0$.

Appendix E. Training S-cells of intermediate layers

In the hierarchical network of the neocognitron, training (or learning) is performed from lower stages to higher stages: after the training of a lower stage has been completely finished, the training of the succeeding stage begins. The same set of training patterns is used for the training of all stages.

Training of intermediate layers, U_{S2} and U_{S3}, is made by the add-if-silent rule. Different from neocognitrons of previous versions, in the new neocognitron, which uses the add-if-silent rule, input connections to S-cells are not changed, once the S-cells have been generated in the network.

Since a layer of the neocognitron consists of cell-planes, generation of an S-cell means the generation of a cell-plane.

If there is any location where C-cells of the preceding stage are active but all S-cells are silent around there (within radius D_l from the location), a new cell-plane is generated. At each presentation of a training pattern, generation of new cell-planes is repeated until all locations where presynaptic C-cells are active are covered with active S-cells.

Specifically, we first prepare a virtual cell-plane called a *seed-selecting plane* for the layer. Every time when a training pattern is presented, the seed-selecting plane is set to an initial value of

$$u_{\mathrm{seed}l}(\boldsymbol{n}) = \varphi\left[\sum_{\kappa=1}^{K_{Cl-1}} u_{Cl-1}(\boldsymbol{n}, \kappa) - \theta_{\mathrm{seed}l}\right]. \tag{E.1}$$

Threshold $\theta_{\mathrm{seed}l}$ in this equation is determined by

$$\theta_{\mathrm{seed}l} = \epsilon \cdot \max_{\boldsymbol{n}}\left[\sum_{\kappa=1}^{K_{Cl-1}} u_{Cl-1}(\boldsymbol{n}, \kappa)\right], \tag{E.2}$$

where ϵ is a small positive constant. In the computer simulation, we chose $\epsilon = 0.3$.

The response of the seed-selecting plane at $\boldsymbol{n}$ is suppressed to zero if any S-cell is active within a radius of D_l around $\boldsymbol{n}$. Namely,

$$u_{\mathrm{seed}l}(\boldsymbol{n}) = 0, \quad \text{if} \sum_{\kappa=1}^{K_{Sl}} \sum_{|\boldsymbol{\nu}|\le D_l} u_{Sl}(\boldsymbol{n}+\boldsymbol{\nu},\kappa) > 0, \tag{E.3}$$

where K_{Sl} is the number of the cell-planes that have been generated by that time. In the computer simulation, we chose $D_l = 2.5$ for both $l = 2$ and 3.

If the response of the seed-selecting plane is not completely zero, we generate a new cell-plane: We search the location $\hat{\boldsymbol{n}}$ at which $u_{\mathrm{seed}l}(\boldsymbol{n})$ is maximum, and make it the location of the *seed-cell* of the newly generated cell-plane. Let $\hat{k}$ be the sequence number of the newly generated cell-plane. The input connections of the seed-cell, which become the input connections of all S-cells of the cell-plane, is set to the values given by

$$a_{Sl}(\boldsymbol{\nu},\kappa,\hat{k}) = a'_{Sl}(\boldsymbol{\nu},\kappa,\hat{k})/b_{Sl}(\hat{k}), \tag{E.4}$$

where

$$a'_{Sl}(\boldsymbol{\nu},\kappa,\hat{k}) = c_{Sl}(\boldsymbol{\nu}) \cdot u_{Cl-1}(\hat{\boldsymbol{n}}+\boldsymbol{\nu},\kappa), \tag{E.5}$$

and

$$\begin{aligned} b_{Sl}(\hat{k}) &= \sqrt{\sum_{\kappa=1}^{K_{Cl-1}} \sum_{|\boldsymbol{\nu}|\le A_{Sl}} \frac{\left\{a'_{Sl}(\boldsymbol{\nu},\kappa,\hat{k})\right\}^2}{c_{Sl}(\boldsymbol{\nu})}} \\ &= \sqrt{\sum_{\kappa=1}^{K_{Cl-1}} \sum_{|\boldsymbol{\nu}|\le A_{Sl}} c_{Sl}(\boldsymbol{\nu}) \cdot \left\{u_{Cl-1}(\hat{\boldsymbol{n}}+\boldsymbol{\nu},\kappa)\right\}^2}. \end{aligned} \tag{E.6}$$

It should be noted here that $b_{Sl}(\hat{k})$ represents the norm of the training vector.

The response of the seed-selecting plane is suppressed again by the response of the new cell-plane through (E.3), and search the location of a seed-cell to generate and train another cell-plane. The process of generating new cell-plane is repeated until the value of the seed-selecting plane becomes completely zero. We then proceed to the presentation of the next training pattern.

Appendix F. S-cells of the highest stage

F.1. Interpolating-vector

Based on the responses of S-cells of the highest stage U_{S4}, the input pattern is classified by the method of the interpolating-vector. We first discuss the recognition phase, where S-cells have already been trained. The process of training S-cell is discussed later in Appendix F.2. During learning, each cell-plane of U_{S4} is assigned a label indicating the class name of the training patterns that are used to train the cell-plane.

Now, let two S-cells, $u_{S4}(\boldsymbol{n}, i)$ and $u_{S4}(\boldsymbol{n}, j)$, have the same label (or are assigned the same class name). They are cells of the ith and the jth cell-planes, and their response is given by (D.1). Since threshold $\theta_4 = 0$ for this layer, (D.1) reduces to

$$u_{S4}(\boldsymbol{n},k) = \sum_{\kappa=1}^{K_{C3}} \sum_{|\boldsymbol{\nu}|\le A_{S4}} a_{S4}(\boldsymbol{\nu},\kappa,k) \cdot u_{C3}(\boldsymbol{n}+\boldsymbol{\nu},\kappa), \quad (k = i, j). \tag{F.1}$$

Their responses correspond to u_i and u_j in Eq. (12) above.

Similarity s_{ij} between the ith and the jth reference vectors, namely, between the ith and the jth cell-planes, is given by

$$s_{ij} = \sum_{\kappa=1}^{K_{C3}} \sum_{|\boldsymbol{\nu}|\le A_{S4}} \frac{a_{S4}(\boldsymbol{\nu},\kappa,i) \cdot a_{S4}(\boldsymbol{\nu},\kappa,j)}{c_{S4}(\boldsymbol{\nu})}. \tag{F.2}$$

We calculate s_{ij} for every pair of cell-planes (or reference vectors) of the same label in advance, and store the values.

Since s_{ij} have thus been stored, we can easily calculate $\|\boldsymbol{x}\| \cdot s_{\max}$ from (13). Among all pairs of S-cells that have the same label, we search the pair that gives the largest value of $\|\boldsymbol{x}\| \cdot s_{\max}$. The search is made, not at a single location $\boldsymbol{n}$, but from all S-cells that have receptive fields at different locations $\boldsymbol{n}$. The label of the pair of S-cells thus selected represents the final result of pattern recognition.

The result of the pattern recognition thus obtained is displayed by the response of U_{C4}. In the present neocognitron, which are trained to classify handwritten 10 digits, layer U_{C4} has 10 cell-planes, each of which consists of only one cell. If the result of the recognition is k, C-cell of the k-th cell-plane, namely $u_{C4}(k)$, takes the largest value of $\|\boldsymbol{x}\| \cdot s_{\max}$ that is obtained above, while other C-cells of layer U_{C4} are made zero.

F.2. Training S-cells of the highest stage

Every time when a training pattern is presented to input layer U_0 during learning, we first try to classify it by the interpolating-vector using the responses of layer U_{S4}.

If the result of the classification is wrong, or if all S-cells are silent, we generate a new cell-plane and add it to U_{S4}. At the same time, the generated cell-plane is assigned a label indicating the class name of the training pattern. The process of generating a new cell-plane resembles that for intermediate layers, which was discussed in Appendix E, but D_l in (E.3) is as large as to cover the whole area of cell-planes.

Specifically, we first prepare a virtual cell-plane called a seed-selecting plane for the layer. The response of the seed-selecting plane is given by

$$u_{\mathrm{seed}4}(\boldsymbol{n}) = \sum_{\kappa=1}^{K_{C3}} \sum_{|\boldsymbol{\nu}|\le A_{S4}} c_{S4}(\boldsymbol{\nu}) \cdot u_{C3}(\boldsymbol{n}+\boldsymbol{\nu},\kappa). \tag{F.3}$$

We search the location $\hat{\boldsymbol{n}}$ at which $u_{\mathrm{seed}4}(\boldsymbol{n})$ is maximum, and make it the location of the seed-cell of the newly generated cell-plane. The input connections to the seed-cell is determined by (E.4)–(E.6). Unlike S-cells of other stages, however, the input connections to the S-cells of the cell-plane will be modified further during learning, as discussed below.

If the result of the classification is correct, we choose the pair of S-cells that have yielded the largest value of $\|\boldsymbol{x}\| \cdot s_{\max}$. Let the pair of S-cells be $u_{S4}(\hat{\boldsymbol{n}}, i)$ and $u_{S4}(\hat{\boldsymbol{n}}, j)$. Then these two S-cells take the place of seed-cells and learn the training vector, by modifying the input connections to them.

To be more specific, the input connections $a_{S4}(\boldsymbol{\nu},\kappa,\hat{k})$ to these seed-cells ($\hat{k} = i, j$) are determined through auxiliary variables $a'_{S4}(\boldsymbol{\nu},\kappa,\hat{k})$. The auxiliary variables $a'_{S4}(\boldsymbol{\nu},\kappa,\hat{k})$ to these seed-cells are increased by the following amounts:

$$\Delta a'_{Sl}(\boldsymbol{\nu},\kappa,\hat{k}) = p_{\hat{k}} \cdot c_{Sl}(\boldsymbol{\nu}) \cdot u_{Cl-1}(\hat{\boldsymbol{n}}+\boldsymbol{\nu},\kappa), \quad (\hat{k} = i, j), \tag{F.4}$$

where $c_{S4}(\boldsymbol{\nu})$ is given by (D.3). Values of p_i and p_j are obtained by

$$\begin{aligned} p_i &= \frac{u_{S4}(\hat{\boldsymbol{n}},i) - s_{ij} \cdot u_{S4}(\hat{\boldsymbol{n}},j)}{(1-s_{ij})(u_{S4}(\hat{\boldsymbol{n}},i) + u_{S4}(\hat{\boldsymbol{n}},j))}, \\ p_j &= \frac{u_{S4}(\hat{\boldsymbol{n}},j) - s_{ij} \cdot u_{S4}(\hat{\boldsymbol{n}},i)}{(1-s_{ij})(u_{S4}(\hat{\boldsymbol{n}},i) + u_{S4}(\hat{\boldsymbol{n}},j))}, \end{aligned} \tag{F.5}$$

which are reduced from (11). The value of s_{ij} is calculated by (F.2). If $p_i < 0$, namely, $p_j > 1$, we set $p_i = 0$ and $p_j = 1$. Similarly, if $p_i > 1$, namely, $p_j < 0$, we set $p_i = 1$ and $p_j = 0$.

Then the input connections $a_{S4}(\boldsymbol{\nu}, \kappa, \hat{k})$ to these seed-cells ($\hat{k} = i, j$), and consequently to all S-cells in the same cell-planes as the seed-cells, are calculated from the value of $a'_{S4}(\boldsymbol{\nu}, \kappa, \hat{k})$, which is obtained from (F.4), by

$$a_{S4}(\boldsymbol{\nu}, \kappa, \hat{k}) = a'_{S4}(\boldsymbol{\nu}, \kappa, \hat{k}) / b_{S4}(\hat{k}), \quad (\hat{k} = i, j) \tag{F.6}$$

where

$$b_{S4}(\hat{k}) = \sqrt{\sum_{\kappa=1}^{K_{C3}} \sum_{|\boldsymbol{\nu}| \le A_{S4}} \frac{\left\{a'_{S4}(\boldsymbol{\nu}, \kappa, \hat{k})\right\}^2}{c_{S4}(\boldsymbol{\nu})}}. \tag{F.7}$$

Appendix G. C-cell layers

In each stage ($l \le 3$), except the highest stage, the layer of C-cells have the same number of cell-planes as the layer of S-cells. There is a one-to-one correspondence between the cell-planes of the S- and C-cell layers in the same stage. Each C-cell of a cell-plane receives signals from S-cells of the corresponding cell-plane.

We first calculate an average of input signals from presynaptic S-cells, not by a linear summation, but by a root-mean-square. Let $u'_{Cl}(\boldsymbol{n}, k)$ be the averaged input to a C-cell of the kth cell-plane of layer U_{Cl}, where $\boldsymbol{n}$ is the location of its receptive field:

$$u'_{Cl}(\boldsymbol{n}, k) = \sqrt{\varphi\left[\sum_{|\boldsymbol{\nu}| \le A_{Cl}} a_{Cl}(\boldsymbol{\nu}) \cdot \{u_{Sl}(\boldsymbol{n} + \boldsymbol{\nu}, k)\}^2\right]}. \tag{G.1}$$

Then the response of the C-cell is given by

$$u_{Cl}(\boldsymbol{n}, k) = \frac{u'_{Cl}(\boldsymbol{n}, k)}{\sigma_C \cdot u_{\text{Cmax}l} + u'_{Cl}(\boldsymbol{n}, k)}, \tag{G.2}$$

where

$$u_{\text{Cmax}l} = \max_{\boldsymbol{n}, k} \left\{u'_{Cl}(\boldsymbol{n}, k)\right\}. \tag{G.3}$$

Parameter σ_C, which determines the degree of saturation, is $\sigma_C = 0.3$ for all stages.

For the 1st and the 2nd stages ($l = 1, 2$), the connections $a_{Cl}(\boldsymbol{\nu})$ have an excitatory center $a^+_{Cl}(\boldsymbol{\nu})$ and an inhibitory surround $a^-_{Cl}(\boldsymbol{\nu})$:

$$a_{Cl}(\boldsymbol{\nu}) = a^+_{Cl}(\boldsymbol{\nu}) - a^-_{Cl}(\boldsymbol{\nu}) \quad (l = 1, 2), \tag{G.4}$$

where $a^+_{Cl}(\boldsymbol{\nu}) > 0$ in a disk $|\boldsymbol{\nu}| \le A_{Col}$, and $a^-_{Cl}(\boldsymbol{\nu}) > 0$ in an annulus $A_{Col} < |\boldsymbol{\nu}| \le A_{Cl}$.

Different from the 1st and the 2nd stages, the connections to a C-cell of the 3rd stage lack an inhibitory surround and have only an excitatory center. Namely,

$$a_{C3}(\boldsymbol{\nu}) = a^+_{C3}(\boldsymbol{\nu}). \tag{G.5}$$

In the computer simulation, we chose

$$a^+_{Cl}(\boldsymbol{\nu}) = \begin{cases} \gamma_{Cl}^{(|\boldsymbol{\nu}|^2/A^2_{Col})} & |\boldsymbol{\nu}| \le A_{Col} \\ 0 & \text{otherwise} \end{cases} \quad (l = 1, 2, 3), \tag{G.6}$$

$$a^-_{Cl}(\boldsymbol{\nu}) = \begin{cases} \beta_{Cl} \cdot \gamma_{Cl}^{(|\boldsymbol{\nu}|^2/A^2_{Col})} & A_{Col} < |\boldsymbol{\nu}| \le A_{Cl} \\ 0 & \text{otherwise} \end{cases}$$

$$(l = 1, 2), \tag{G.7}$$

where positive constant β_{Cl} is determined so as to satisfy

$$\sum_{|\boldsymbol{\nu}| \le A_{Cl}} a_{Cl}(\boldsymbol{\nu}) = 0, \quad (l = 1, 2). \tag{G.8}$$

The radiuses of the excitatory center and the inhibitory surround are: $A_{Co1} = A_{Co2} = 3.5, A_{C1} = A_{C2} = 5.5, A_{Co3} = A_{C3} = 4.5$. The parameter that determines the spatial decay is $\gamma_{Cl} = 0.7$ for $l = 1, 2, 3$.

References

Elliffe, M. C. M., Rolls, E. T., & Stringer, S. M. (2002). Invariant recognition of feature combinations in the visual system. *Biological Cybernetics*, *86*, 59–71.

Fukushima, K. (1980). Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193–202.

Fukushima, K. (1988). Neocognitron: a hierarchical neural network capable of visual pattern recognition. *Neural Networks*, *1*(2), 119–130.

Fukushima, K. (2003). Neocognitron for handwritten digit recognition. *Neurocomputing*, *51*, 161–180.

Fukushima, K. (2007). Interpolating vectors for robust pattern recognition. *Neural Networks*, *20*(8), 904–916.

Fukushima, K. (2010). Neocognitron trained with winner-kill-loser rule. *Neural Networks*, *23*(7), 926–938.

Fukushima, K. (2011). Increasing robustness against background noise: visual pattern recognition by a neocognitron. *Neural Networks*, *24*(7), 767–778.

Fukushima, K., Hayashi, I., & Léveillé, J. (2011). Neocognitron trained by winner-kill-loser with triple threshold. In B.-L. Lu, L. Zhang, & J. Kwok (Eds.), *Lecture notes in computer science, LNCS*: *vol. 7063*. *Neural information processing — ICONIP 2011, part II* (pp. 628–637). Berlin, Heidelberg: Springer-Verlag.

Fukushima, K., & Tanigawa, M. (1996). Use of different thresholds in learning and recognition. *Neurocomputing*, *11*(1), 1–17.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology (Lond.)*, *160*(1), 106–154.

Hubel, D. H., & Wiesel, T. N. (1965). Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *Journal of Neurophysiology*, *28*(2), 229–289.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*, 541–551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lo, S. B., Chan, H., Lin, J., Li, H., Freedman, M. T., & Mun, S. K. (1995). Artificial convolution neural network for medical image pattern recognition. *Neural Networks*, *8*(7/8), 1201–1214.

Mallot, H. A. (2000). *Computational vision: information processing in perception and visual behavior*. Cambridge, Mass.: MIT Press, Chapter 4, translated by J.S. Allen.

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*(11), 1019–1025.

Satoh, S., Kuroiwa, J., Aso, H., & Miyake, S. (1999). Recognition of rotated patterns using a neocognitron. In L. C. Jain, & B. Lazzerini (Eds.), *Knowledge based intelligent techniques in character recognition* (pp. 49–64). CRC Press.

Serre, T., Oliva, A., & Poggio, T. (2007). A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Sciences of the United States of America*, *104*(15), 6424–6429.