Available online at www.sciencedirect.com
ScienceDirect
Neural Networks 20 (2007) 904–916
Neural Networks
www.elsevier.com/locate/neunet

Interpolating vectors for robust pattern recognition

Kunihiko Fukushima

Kansai University, Takatsuki, Osaka 569-1095, Japan

Received 15 January 2007; received in revised form 1 June 2007; accepted 1 June 2007

Abstract

This paper proposes a powerful algorithm for pattern recognition, which uses *interpolating vectors* for classifying patterns. Labeled reference vectors in a multi-dimensional feature space are first produced by a kind of competitive learning. We then assume a situation where virtual vectors, called interpolating vectors, are densely placed along line segments connecting all pairs of reference vectors of the same label. From these interpolating vectors, we choose the one that has the largest similarity to the test vector. Its label shows the result of pattern recognition. In practice, we can get the same result with a simpler process.

We applied this method to the *neocognitron* for handwritten digit recognition and reduced the error rate from 1.52% to 1.02% for a blind test set of 5000 digits.

Keywords: Pattern recognition; Interpolating vector; Labeled competitive learning; Neocognitron

1. Introduction

This paper proposes a powerful algorithm for pattern recognition, which uses *interpolating vectors* for classifying patterns in a multi-dimensional feature space.

Most of the visual pattern recognition systems extract features from input patterns first and then try to classify input pattern based on the extracted features. The features of a pattern can be represented by a vector in a multi-dimensional feature space. Various methods for classifying feature vectors have been proposed so far (Burges, 1998; Cover & Hart, 1968; Gray, 1984; Kohonen, 1995; Schölkoph et al., 1997). Many of them try to classify input patterns based on some kinds of similarities between test vectors (or input patterns) and labeled reference vectors (or code vectors), which have been produced from training vectors. A variety of techniques have been proposed, for generating labeled reference vectors and/or finding a reference vector that has the largest similarity to the test vector.

In the method proposed in this paper, labeled reference vectors are first produced by a kind of competitive learning. Different from conventional methods, however, we do not simply search for a reference vector that has the largest similarity to the test vector. We assume a situation where virtual vectors, called "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 infinite number of interpolating vectors. We just assume line segments connecting every pair of reference vectors of the same label. The line segments have 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. The label of the line segment shows the result of pattern recognition.

If some parts of borders between other classes are concave in the multi-dimensional feature space, some of the line segments might cross the concave borders and invade into the territory of other classes. To find out and eliminate such line segments, we have an adjusting phase after having generated reference vectors in the learning. During the adjusting phase, we test how the training vectors, which have been used to generate reference vectors, are classified. If a training vector is erroneously classified, line segments that cause erroneous classification of this training vector are all eliminated, because they are suspicious of crossing class borders.

E-mail address: fukushima@m.ieice.org.

0893-6080/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2007.06.003

To demonstrate the ability of this algorithm, we apply it to the *neocognitron* and show that the recognition rate can be increased further than that of the conventional neocognitron. Incidentally, the neocognitron, which the author proposed previously, is a neural network model for robust visual pattern recognition (Fukushima, 1980, 1988, 2003).

In the case of a neocognitron, the sets of input connections to individual S-cells of the highest stage, which correspond to labeled reference vectors, are first produced by a kind of competitive learning. After having finished the learning, the response of an S-cell can be interpreted as representing similarity between a test vector and the reference vector in a multi-dimensional feature space.

In the conventional neocognitron, the label (or the class name) of the largest-output S-cell, namely, the label of the reference vector that has the largest similarity to the test vector, is the result of pattern recognition. On the other hand, our proposed algorithm analyzes the response of the S-cells using interpolating vectors and largely reduces the error rate.

We first discuss, in Section 2, the use of interpolating vectors for pattern recognition in general. In Section 3, we briefly explain the neocognitron and discuss how to apply the interpolating vectors to the neocognitron. A more detailed description of the neocognitron used for the present experiment appears in Appendix. In Section 4, we demonstrate by computer simulation that the error rate of the neocognitron, which is used for handwritten digit recognition, is largely reduced by the use of interpolating vectors. Finally in Section 5, we make some discussions on the method of interpolating vectors.

2. Interpolating vectors

2.1. Multi-dimensional feature space

In visual pattern recognition, we assume a situation where the process of feature extraction has already been finished. Each input pattern, either training or test pattern, can be represented by a vector in a multi-dimensional feature space.

We define similarity s between arbitrary two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, using inner product $(\boldsymbol{x}, \boldsymbol{y})$ and norm $\|\boldsymbol{x}\| = \sqrt{(\boldsymbol{x}, \boldsymbol{x})}$ of the vectors, by

$$s = \frac{(\boldsymbol{x}, \boldsymbol{y})}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|}. \tag{1}$$

It is equal to the cosine of the angle between two vectors. If all elements of the vectors take non-negative values, like in the case of the neocognitron, s takes a value in the range of $0 \le s \le 1$.

2.2. Reference vectors

Each training vector has a label indicating the class to which the vector belongs. From a set of training vectors, we produce *reference vectors* for each class. Each reference vector has a label of the class name. A single reference vector usually represents a number of training vectors of a class. There can be more than one reference vectors for each class.

The reference vectors are created in such a way that a large number of training sets can be represented correctly by only a small number of reference vectors. This can greatly reduce the computational cost, which will be required by the use of interpolating vectors.

The learning progresses in two steps: an initial learning and an adjusting phase. In the initial learning, we produce reference vectors in such a way that each training vector of a class comes to have a largest similarity to a reference vector of the class. The generation of reference vectors is made by a kind of competitive learning, which was used to train the highest stage of the conventional neocognitron (Fukushima, 2003) (see Section 3.3).

The learning starts from a state where there is no reference vector. When a training vector of a class is presented at first, it is adopted as a reference vector and is assigned a label of the class name. If another training vector is presented afterward, similarities of the training vector to all reference vectors are measured. The reference vector that has the largest similarity to the training vector is taken as a winner of the competition.

If the label of the winner is the same as the class name of the training vector, the training vector is added to the reference vector of the winner, resulting in a modification of the reference vector. In other words, a reference vector $\boldsymbol{X}$ is produced by the total sum of all training vectors that have made $\boldsymbol{X}$ a winner:

$$\boldsymbol{X} = \sum_m \boldsymbol{x}^{(m)}, \tag{2}$$

where $\boldsymbol{x}^{(m)}$ is the mth training vector that have made $\boldsymbol{X}$ a winner.

If the label of the winner is different from that of the training vector, however, the reference vector, which became the winner and caused a wrong classification of this training vector, is not modified this time. A new reference vector is generated instead: the training vector itself is adopted as the new reference vector and is assigned a label of the class name of the training vector.

The process of finding the winner is equivalent to the process of finding the nearest reference vector in a space where distance is defined based on similarity. Each reference vector has its own territory determined by the Voronoi partition of the feature space. This process resembles the vector quantization (Gray, 1984; Kohonen, 1995) in this sense.

Generation of a new reference vector causes a shift of decision borders in the feature space, and some of the training vectors of other classes, which have been classified correctly before, might be misclassified now. If this situation occurs, additional reference vectors have to be generated again to readjust the decision borders. Thus, the decision borders are gradually adjusted to fit the real borders between classes.

Since training vectors that are located near the center of the territory of a class have a large tendency of being correctly classified, a single reference vector that is distant from its class border usually represents a large number of training vectors. As a result, the number of reference vectors generated is much smaller than the number of training vectors. Training vectors that are misclassified in the learning phase often come from near class borders. Hence reference vectors come to be

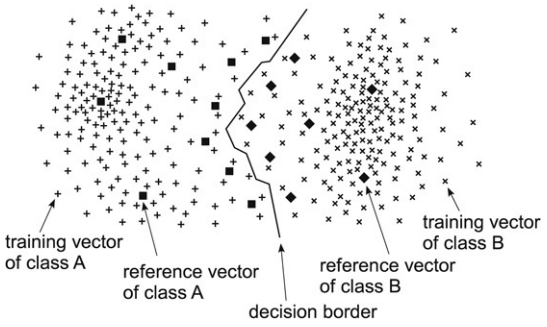

Fig. 1. Distribution of training and reference vectors in a multi-dimensional feature space.

distributed more densely near class borders as illustrated in Fig. 1.

We repeat this process until the generation of new reference vectors stops. Namely, every time when each round of presentation of a training set has finished in the initial learning phase, the number of newly generated reference vectors during that round, which is equal to the number of erroneously classified training vectors, is counted. If it reaches zero, the initial learning process ends.

Although a repeated presentation of a training vector set is required before the initial learning converges, the required number of repetition is not so large in usual cases. When this process was applied to a neocognitron, which is discussed later in Section 4, only six rounds of presentation of the training set was enough.

2.3. Interpolating vectors

After having finished the initial learning, by which all reference vectors have been produced, we use the method of interpolating vectors to recognize patterns more robustly. Each pattern that is to be classified is called a test vector in the feature space.

The basic idea of the method of interpolating vectors 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 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. The label of the line segment shows the result of 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

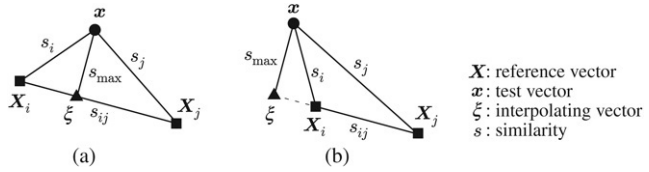

Fig. 2. Largest similarity $s_{\max}$ between test vector $\boldsymbol{x}$ and the line segment connecting a pair of reference vectors $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ of the same label.

by a linear combination of them:

$$\boldsymbol{\xi} = p\frac{\boldsymbol{X}_i}{\|\boldsymbol{X}_i\|} + q\frac{\boldsymbol{X}_j}{\|\boldsymbol{X}_j\|}. \tag{3}$$

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

$$s = \frac{(\boldsymbol{\xi}, \boldsymbol{x})}{\|\boldsymbol{\xi}\| \cdot \|\boldsymbol{x}\|} = \frac{ps_i + qs_j}{\sqrt{p^2 + 2pqs_{ij} + q^2}}, \tag{4}$$

where

$$s_i = \frac{(\boldsymbol{X}_i, \boldsymbol{x})}{\|\boldsymbol{X}_i\| \cdot \|\boldsymbol{x}\|}, \qquad s_j = \frac{(\boldsymbol{X}_j, \boldsymbol{x})}{\|\boldsymbol{X}_j\| \cdot \|\boldsymbol{x}\|},$$
$$s_{ij} = \frac{(\boldsymbol{X}_i, \boldsymbol{X}_j)}{\|\boldsymbol{X}_i\| \cdot \|\boldsymbol{X}_j\|}. \tag{5}$$

Among various combinations of p and q, similarity s takes a maximum value

$$s_{\max} = \sqrt{\frac{s_i^2 - 2s_i s_j s_{ij} + s_j^2}{1 - s_{ij}^2}}. \tag{6}$$

The ratio of p and q that gives this maximum value is

$$p : q = (s_i - s_j s_{ij}) : (s_j - s_i s_{ij}). \tag{7}$$

It should be noted here that similarity s defined by (4) is affected only by the ratio of p and q, and not by their absolute values, because $\boldsymbol{\xi}$ is normalized by its norm in the calculation of s.

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. 2(a)). Among all line segments that connect every pair 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.

In the search for the largest similarity here, we allow p or q to take a negative value, because better results are obtained usually in the computer simulation of the neocognitron. This means that line segments can extend beyond the reference vectors on both sides (Fig. 2(b)), and that the search is made among *interpolating* and *extrapolating* vectors (see Section 5 for further discussions).

2.4. Adjusting phase

If some parts of the border of a class are concave in the multi-dimensional feature space, however, some of the line

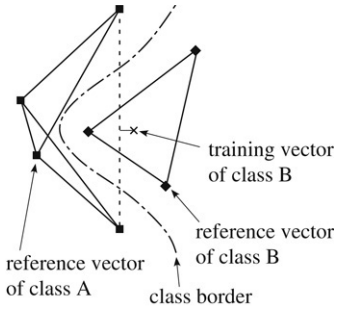

Fig. 3. Elimination of a line segment that crosses the concave border between classes. The training vector of class B (×) is nearer to the line segment of class A (dotted line) than to line segments of class B.

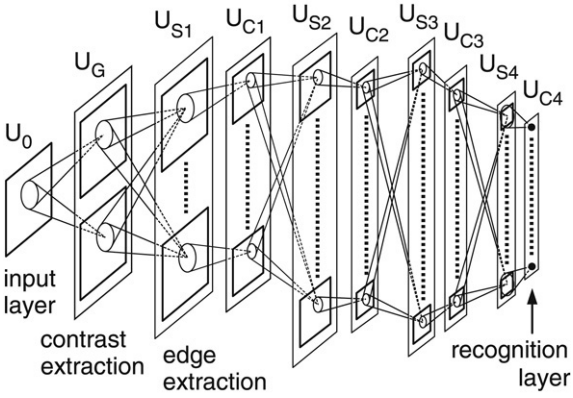

Fig. 4. The architecture of the neocognitron.

segments might cross the concave parts of the border and invade into the territory of other classes. Such line segments, whose example is illustrated by a dotted line in Fig. 3, will cause misclassification of the test vector. To find out and eliminate such line segments that are suspected of crossing class borders, we have an adjusting phase after having created reference vectors in the initial learning.

During the adjusting phase, we test how the training vectors, which have been used to generate reference vectors, are classified. If a training vector is erroneously classified, we suspect that the line segment nearest to the training vector crosses the class border. We then eliminate the line segment. Sometimes, more than one line segments might be eliminated by a single training vector. Thus, line segments that cause erroneous classification of training vectors are all eliminated in the adjusting phase of the learning.

Incidentally, if reference vectors have already been created in such a way that all training vectors can be classified correctly without using interpolating vectors (that is, by measuring only similarity to reference vectors), all training vectors will come to be classified correctly after having finished this adjusting phase.

3. Use of interpolating vectors for the neocognitron

3.1. Outline of the neocognitron

The neocognitron is a neural network model for robust visual pattern recognition (Fukushima, 1988, 2003). This section discusses how to apply the method of interpolating vectors to the neocognitron to improve the recognition rate further.

The neocognitron to which the method of interpolating vectors is applied is almost the same as the conventional neocognitron (Fukushima, 2003), except the highest stage of the hierarchical network. A more detailed description of the neocognitron, including mathematical expressions, is found in Appendix.

As illustrated in Fig. 4, the network consists of 4 stages of S- and C-cell layers: $U_0 \rightarrow U_G \rightarrow U_{S1} \rightarrow U_{C1} \rightarrow U_{S2} \rightarrow U_{C2} \rightarrow U_{S3} \rightarrow U_{C3} \rightarrow U_{S4} \rightarrow U_{C4}$. The stimulus pattern is presented to input layer U_0, and the result of pattern recognition appears in layer U_{C4}.

There are retinotopically ordered connections between cells of adjoining layers in the hierarchical network. Each cell receives input connections that lead from cells situated in a limited area on the preceding layer.

Cells of layer U_G resemble retinal ganglion cells and have either on-center or off-center receptive fields. The on-center cells extract positive contrast in brightness, and the off-center cells extract negative contrast from the image presented to the input layer U_0.

S-cells have modifiable input connections, which are determined by learning. They work as feature-extracting cells after having finished the learning. Namely, after having finished learning, each S-cell comes to respond selectively to a particular feature presented in its receptive field.

The S-cells of the first stage (U_{S1}) resemble simple cells in the primary visual cortex. They have been trained with supervised learning to extract edge components of various orientations from input images. More global features, such as parts of training patterns, are extracted in higher stages (U_{S2} and U_{S3}), which are self-organized using unsupervised competitive learning.

C-cells, which resemble complex cells in the visual cortex, are inserted in the network to allow for positional errors in the features of the stimulus. The input connections of C-cells, which come from S-cells of the preceding layer, are fixed and unmodifiable. Each C-cell receives excitatory input connections from a group of S-cells that extract the same feature, but from slightly different positions. The C-cell responds if at least one of these S-cells yields an output. Even if the stimulus feature shifts in position and another S-cell comes to respond instead of the first one, the same C-cell keeps responding. Thus, the C-cell's response is less sensitive to shift in position of the input pattern. We can also express that C-cells make a blurring operation, because the response of a layer of S-cells is spatially blurred in the response of the succeeding layer of C-cells.

In the whole network, with its alternate layers of S- and C-cells, feature extraction by S-cells and toleration of positional shift by C-cells are repeated. During this process, local features extracted in lower stages are gradually integrated into more global features. Since small amounts of positional errors of local features are absorbed, an S-cell in a higher stage comes to respond robustly to a specific feature even if the feature is slightly deformed or shifted.

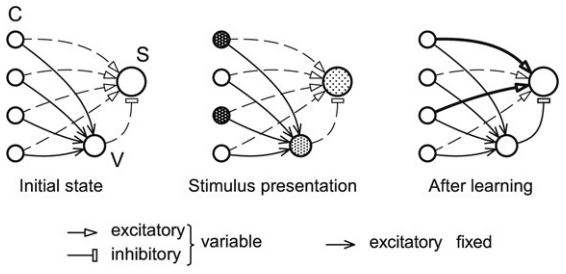

Fig. 5. Connections converging to an S-cell in the learning phase.

Each layer is divided into sub-layers, called *cell-planes*, according to the features to which the cells responds. The cells in each cell-plane are arranged retinotopically in a two-dimensional array and share the same set of input connections. In other words, the connections to a cell-plane have a translational symmetry. As a result, all the cells in a cell-plane have receptive fields of an identical characteristic, but the locations of the receptive fields differ from cell to cell. The modification of variable connections during the learning progresses under the restriction of shared connections.

Since cells in higher stages come to have larger receptive fields, the density of cells in each layer is designed to decrease with the order of the stage.

This neocognitron is initially trained to recognize patterns in an identical manner as for the conventional neocognitron (Fukushima, 2003). After having finished the conventional learning, we use interpolating vectors to analyze the response of U_{S4} and obtain a better recognition rate.

3.2. S-cells of the highest stage

Since the difference from the conventional neocognitron (Fukushima, 2003) mainly resides in the highest stage, we discuss here the characteristics of layer U_{SL} ($L = 4$) at the highest stage.

Let $u_{SL}(\boldsymbol{n}, k)$ be the output of S-cells of the kth cell-plane of the highest stage, where $\boldsymbol{n}$ represents the location of the receptive field center of the cells. Layer U_{SL}, like layers in other stages, contains not only S-cells but also V-cells, whose output is represented by $v_L(\boldsymbol{n})$. Fig. 5 illustrates the connections converging to an S-cell. The outputs of an S-cell and the V-cell, which is sending an inhibitory signal to the S-cell, are given by

$$u_{SL}(\boldsymbol{n}, k) = \frac{\sum_{\kappa=1}^{K_{CL-1}} \sum_{|\boldsymbol{\nu}|<A_{SL}} a_{SL}(\boldsymbol{\nu}, \kappa, k) \cdot u_{CL-1}(\boldsymbol{n}+\boldsymbol{\nu}, \kappa)}{b_{SL}(k) \cdot v_L(\boldsymbol{n})}, \quad (8)$$

$$v_L(\boldsymbol{n}) = \sqrt{\sum_{\kappa=1}^{K_{CL-1}} \sum_{|\boldsymbol{\nu}|<A_{SL}} c_{SL}(\boldsymbol{\nu}) \cdot \{u_{CL-1}(\boldsymbol{n}+\boldsymbol{\nu}, \kappa)\}^2}, \quad (9)$$

where $a_{SL}(\boldsymbol{\nu}, \kappa, k)$ (≥ 0) is the strength of variable excitatory connection that the S-cell receives 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 a single S-cell. Parameter $b_{SL}(k)$ (≥ 0) is the strength of variable inhibitory connection coming from the V-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 $|\boldsymbol{\nu}|$.

3.3. Initial learning of the highest stage

Before applying the method of interpolating vectors, we first train the S-cells of the highest stage (U_{SL}) using the same method as for the conventional neocognitron (Fukushima, 2003).

As can be seen from the discussions below, the process of the initial learning of the highest stage of the neocognitron is the same as the process of creating reference vectors, which was discussed in Section 2.2.

The initial learning of the highest stage U_{SL} begins, after the training of lower stages (U_{S2} and U_{S3}) has been completely finished. The training pattern set is the same as that for the lower stages. The training set is presented repeatedly until the generation of new cell-planes stops. Namely, every time when each round of presentation of the training set has finished, the number of newly generated cell-planes during that round, which is equal to the number of erroneously classified training patterns, is counted. If it reaches zero, the initial learning process ends.

Every time when a training pattern is presented, competition occurs among all S-cells in the layer. The maximum-output S-cell is taken as a winner of the competition. If the winner of the competition has the same label as the training pattern, the winner becomes the *seed cell* and learns the training pattern (Fukushima, 2003) (see Appendix A.4.1 for the meaning of a seed cell).

If the winner has a wrong label (or if all S-cells are silent), however, a new cell-plane is generated and is assigned a label of the class name of the training pattern.

Let cell $u_{SL}(\hat{\boldsymbol{n}}, \hat{k})$ be selected as a seed cell at a certain time. The variable connections $a_{SL}(\boldsymbol{\nu}, \kappa, \hat{k})$ to this seed cell, and, as a consequence of shared connections, to all S-cells in the same cell-plane as the seed cell, are increased by the following amount:

$$\Delta a_{SL}(\boldsymbol{\nu}, \kappa, \hat{k}) = c_{SL}(\boldsymbol{\nu}) \cdot u_{CL-1}(\hat{\boldsymbol{n}}+\boldsymbol{\nu}, \kappa). \quad (10)$$

The inhibitory connection $b_L(\hat{k})$ is determined directly from the values of the excitatory connections $a_{SL}(\boldsymbol{\nu}, \kappa, \hat{k})$. That is,

$$b_{SL}(\hat{k}) = \sqrt{\sum_{\kappa=1}^{K_{CL-1}} \sum_{|\boldsymbol{\nu}|<A_{SL}} \frac{\{a_{SL}(\boldsymbol{\nu}, \kappa, \hat{k})\}^2}{c_{SL}(\boldsymbol{\nu})}}. \quad (11)$$

3.4. Application of interpolating vectors to the neocognitron

After having finished the conventional learning (Fukushima, 2003), we use interpolating vectors to analyze the response of U_{SL} and obtain a better recognition rate.

We define a *weighted* inner product of arbitrary two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ by

$$(\boldsymbol{x}, \boldsymbol{y}) = \sum_{|\boldsymbol{\nu}|<A_{SL}} c_{SL}(\boldsymbol{\nu}) x(\boldsymbol{\nu}) y(\boldsymbol{\nu}), \tag{12}$$

using $c_{SL}(\boldsymbol{\nu})$ (>0), which represents the strength of fixed excitatory connections to a V-cell (see (9) and (10)), as the weight. We use this *weighted* inner product in the definition of similarity by Eq. (1).

Let $\boldsymbol{x}$ be the vector representing input signals to an S-cell $u_{SL}(\boldsymbol{n}, k)$. In other words, the response of a preceding C-cell $u_{CL-1}(\boldsymbol{n}+\boldsymbol{\nu}, \kappa)$ is the $\boldsymbol{\nu}$th element of vector $\boldsymbol{x}$.

Similarly, let $\boldsymbol{x}^{(m)}$ be the mth training vector to the S-cell. (To be more exact, $\boldsymbol{x}^{(m)}$ is the training vector to the mth seed cell of the cell-plane to which the S-cell belongs.) Let $\boldsymbol{X}$ be the total sum of all training vectors to the S-cell, which can be expressed by the same equation as (2).

Substituting (9)–(11) in (8), we have

$$u_{SL}(\boldsymbol{n}, k) = \frac{(\boldsymbol{X}, \boldsymbol{x})}{\|\boldsymbol{X}\| \cdot \|\boldsymbol{x}\|} = s. \tag{13}$$

We can interpret that the set of input connections to an S-cell (or a cell-plane) represents the *reference vector*. The response of the S-cell shows the similarity s between the test vector $\boldsymbol{x}$ and the reference vector $\boldsymbol{X}$. Thus, the process of the initial learning of the neocognitron is the same as the process of creating reference vectors in the multi-dimensional feature space: If a training vector is correctly classified in the initial learning, it is added up to the reference vector (or the input connections) of the winner. If a training vector is misclassified, however, the reference vector (or the input connections) of the winner, which caused a wrong recognition for this training vector, is not modified this time. A new cell-plane is generated instead, and the training vector is adopted as the reference vector of the new cell-plane.

Now, let two S-cells, $u_{SL}(\boldsymbol{n}, k_i)$ and $u_{SL}(\boldsymbol{n}, k_j)$, have the same label (or are assigned the same class name). Their responses correspond to s_i and s_j in Eq. (5).

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

$$s_{ij} = \frac{1}{b_{SL}(k_i) \cdot b_{SL}(k_j)} \times \sum_{\kappa=1}^{K_{CL-1}} \sum_{|\boldsymbol{\nu}|<A_{SL}} \frac{a_{SL}(\boldsymbol{\nu}, \kappa, k_i) \cdot a_{SL}(\boldsymbol{\nu}, \kappa, k_j)}{c_{SL}(\boldsymbol{\nu})}. \tag{14}$$

We calculate s_{ij} for every pair of cell-planes (or reference vectors) of the same label in advance, and store the values. We also put a mark, in the adjusting phase of the learning, to the pairs if the line segments connecting the pairs have caused a wrong classification for any training vectors and are suspected of crossing class borders.

Since s_{ij} have thus been stored, we can easily calculate $s_{\max}$ from (6). Among all pairs of S-cells that have the same label, we search for the pair that gives the largest value of $s_{\max}$. 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. In this search, however, we omit the pairs of S-cells, or the line segments, that are marked as suspicious of crossing borders.

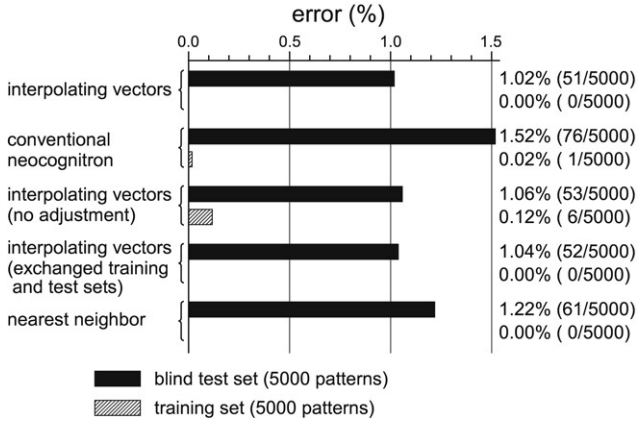

Fig. 6. Recognition errors under various conditions.

4. Computer simulation

We simulated a neocognitron with interpolating vectors on a computer. We tested its ability to recognize patterns, using handwritten digits (free writing) randomly sampled from the ETL1 database, which was published by former Electrotechnical Laboratory, Tsukuba, Japan. The scales and parameters of the network, and the method of learning for the lower stages of the network, are almost the same as those for the conventional neocognitron (Fukushima, 2003) (see Appendix).

We show here the results of an experiment, where we used a training set randomly sampled 5000 patterns (500 patterns for each digit). The error rate of the recognition after having finished the learning was measured by a blind test set, which was another set of randomly sampled 5000 patterns that were not used for the learning. Fig. 6 summarizes the recognition errors of the neocognitron under various conditions, which are discussed below.

When the method of interpolating vectors was used, the error rate was 1.02% (51/5000) with no rejection, for a blind test set of randomly sampled 5000 patterns, and 0% for the training set. Incidentally, the number of cell-planes (or reference vectors) generated in U_{S4} was 160, which was much smaller than the number of training patterns. Only six times of repeated presentation of the training set was enough before finishing the initial learning. The number of line segments was also reasonably small: it was 1294 after having finished the adjusting phase. The line segments eliminated during the adjusting phase were 16.

Fig. 7 shows a typical response of the network that has finished the learning. The leftmost side of the figure is the input layer U_0, and the rightmost is the recognition layer U_{C4}. Each square in a layer represent a cell-plane. The responses of

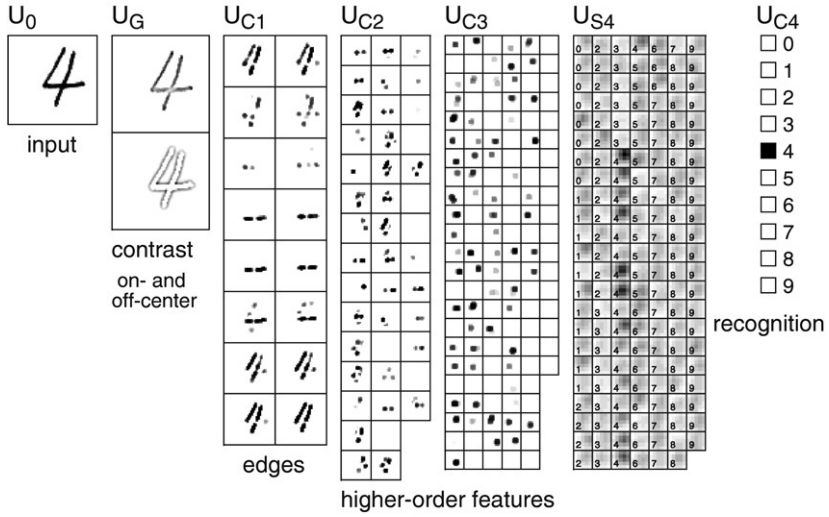

Fig. 7. An example of the response of the neocognitron with interpolating vectors. The input pattern (U_0) is recognized correctly as '4' (U_{C4}).

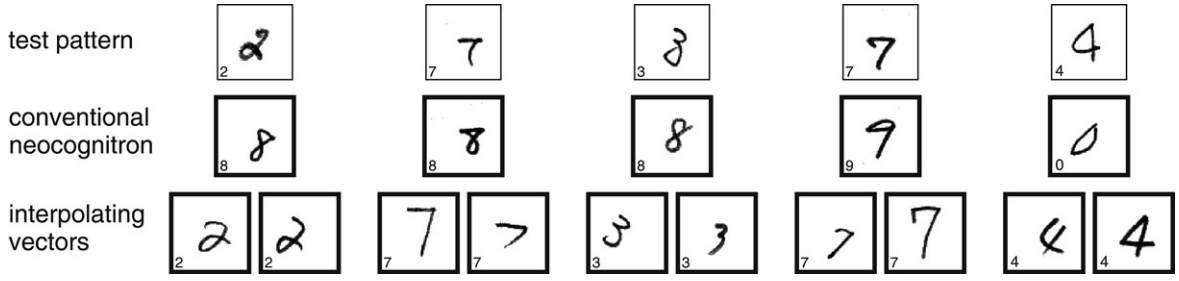

Fig. 8. Some examples of the test patterns (upper row) that were recognized, erroneously by the conventional neocognitron (middle row), but correctly by the use of interpolating vectors (bottom row).

U_{S1}, U_{S2} and U_{S3} are omitted from the figure, because they can be easily estimated from the responses of C-cell layers: a blurred version of the response of an S-cell layer appears in the corresponding C-cell layer, although it is slightly modified by the inhibitory surround in the connections. The cell-planes of U_{S4} in the display are not arranged in the order of their generation, but are rearranged in such a way that cell-planes of the same label are clustered. The digit written in each cell-plane of U_{S4} is its label (or the class name).

If the method of interpolating vectors was not used and the response of U_{S4} was analyzed by the method for the conventional neocognitron, the error rate was 1.52% (76/5000) for the blind test set, and 0.02% (1/5000) for the training set.[1] This means that, by the use of interpolating vectors, the erroneously recognized pattern decreased from 76 to 51.

Fig. 8 shows some examples of the test patterns that were recognized, erroneously by the conventional neocognitron, but correctly by the use of interpolating vectors. The figure shows which reference vectors (middle row: conventional neocognitron), and which line segments (bottom row: interpolating vectors), exhibited the largest similarity to the test patterns (top row): In each column of the display, although the test vector had a large similarity to the reference vector of a wrong class (shown in the middle row), it had a larger similarity to the line segment connecting the pair of reference vectors of the correct class (shown in the bottom row).

In this display, each reference vector is represented expediently by a training pattern that elicits the largest response from it (namely, the cell-plane of U_{S4}). It should be noted here that it is not possible to express accurately the input pattern that each reference vector really represent, because reference vectors are usually created by a linear combination of a number of training vectors. In other words, each reference vector is created, not by a direct linear combination of the training patterns presented to U_0, but by a linear combination of the responses of U_{C3}.

Incidentally, if the adjusting phase, which eliminates suspicious line segments, was omitted, the error rate was 1.06% (53/5000) for the blind test set, and 0.12% (6/5000) for the training set. It is seen that, even if adjusting phase is abbreviated, the error rate does not increase so much for the blind test patterns, but increases largely for the training patterns.

[1] The reason why the error rate for the training set is not completely zero can be explained as follows. As was mentioned in Section 2.2, the initial learning is continued until the number of newly generated reference vectors, which is equal to the number of erroneously classified training vectors, reaches zero during the last round of presentation of the training set. This usually, but not always, guarantees that all training vectors will be recognized correctly after having finished the initial learning, because reference vectors drift slightly even during the last round of presentation of the training set, where each training vector is added up to the reference vector of the winner.

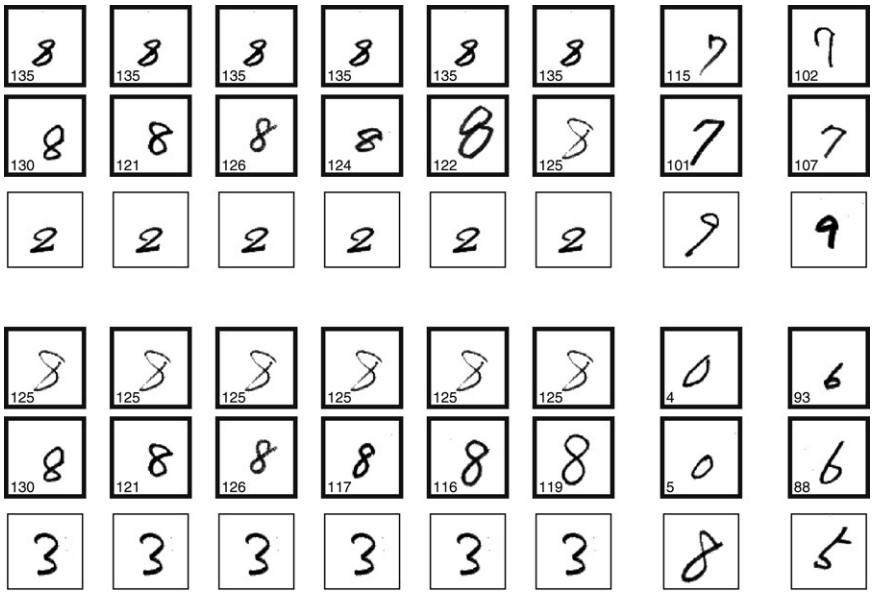

Fig. 9. The line segments eliminated during the adjusting phase. In each column, a pair of reference vectors on both sides of the eliminated line segments is enclosed by thick squares, and the training pattern that caused the elimination is enclosed by a thin square.

Fig. 9 shows the line segments eliminated during the adjusting phase. In each column in the figure, a pair of reference vectors on both sides of an eliminated line segment is enclosed by thick squares (the top and the middle patterns in the column), and the training pattern that caused the elimination is enclosed by a thin square (the bottom pattern). In some cases (six columns shown to the left side of the upper and the lower rows in the figure), one training vector eliminates a number of line segments, and all line segments eliminated by the training vector share the same reference vector on one side.

To see how the recognition rate is affected by the sampling of training and test patterns, we tried to exchange the training and the test sets. When the training and the test sets were exchanged, the error rate for the test set was 1.04% (52/5000), and 0% for the training set, if the method of interpolating vectors was used. The effect of the exchange was very little.

For comparison, we tried the nearest-neighbor method. It can be realized by the conventional neocognitron, if the learning method for U_{S4} is slightly modified. Namely, during the learning phase, a new cell-plane is always generated in U_{S4} for each training pattern. When the training set of 5000 patterns is used, 5000 cell-planes are generated, and an extremely large computational cost is required. The error rate for the blind test set was 1.22% (61/5000), which was worse than that by the interpolating vectors. Incidentally, we also tried k-nearest-neighbor methods ($k \geq 3$), but we were not able to get a better recognition than the 1-nearest-neighbor method. The error rate was 1.32% by the 3-nearest, and 1.42% by the 5-nearest-neighbor method. This means that the method of interpolating vectors yields a higher recognition rate with a much smaller computational cost than the nearest-neighbor method.

We also tested another type of k-nearest-neighbor method, where we searched k-nearest reference-vectors. In the conventional neocognitron, the label of the reference vector that has the largest similarity to the test vector shows the result of pattern recognition. This can be interpreted as a process of finding the 1-nearest reference-vector. We tried to extend the search to k-nearest reference-vectors, but the results were very bad. The error rate for the blind test set was 2.76% by the 3-nearest, and 3.56% by the 5-nearest reference-vectors. A reason for such bad error rates might be a mismatch between the process of k-nearest reference-vectors and the method of generating reference vectors. As has been discussed in Section 2.2, the reference vectors are generated in such a way that the 1-nearest reference-vector comes from the correct class for each training vector, but it is not regarded whether other reference vectors, such as the 2nd and the 3rd-nearest reference-vectors, come from the correct class or not. Since reference vectors are mainly generated near class borders rather than within the territories of individual classes, reference vectors of wrong classes, as well as the ones of the correct class, are distributed densely near class borders.

We also tried to apply the k-nearest-neighbor method, not to the analysis of the responses of U_{C3}, but to the direct analysis of input patterns U_0 themselves. The error rate for the blind test set was, of course, much worse. It was 16.40% by the 1-nearest, 15.48% by the 3-nearest, and 14.98% by the 5-nearest-neighbor method, respectively.

5. Discussions

This paper proposed the use of interpolating vectors for robust pattern recognition. To demonstrate its ability, we applied it to the neocognitron for handwritten digit recognition, and showed that the error rate is reduced from 1.52% to 1.02% for a blind test set of 5000 digits. The increase in computational cost by the use of interpolating vectors is very small, because the number of reference vectors (or cell-planes) generated in the learning is much smaller than the number of training vectors (or training patterns).

The error rate of the neocognitron is affected a little by various conditions, such as the size of the training set, threshold values of S-cells of lower stages, scale of the network, and so on. Decrease in error rate by the use of interpolating vectors, however, was observed always under all conditions ever tested.

Incidentally, the error rate of the neocognitron with interpolating vectors can be decreased further, if the method of creating reference vectors in the initial learning is slightly modified. In the conventional method for creating reference vectors, which was discussed in Section 2.2 and used in the computer simulation in Section 4, competition was allowed to occur among all reference vectors every time when a training vector was presented. In the modified method, the competition is restricted among reference vectors whose similarities to the training vector are larger than a certain threshold value, θ_L. In other words, the competition is restricted among S-cells whose responses are larger than θ_L.

When we took $\theta_L = 0.6$, 0.8, 0.85, 0.90, 0.95 and 1.00, the error rate of the neocognitron for the blind test set was 1.00% (50/5000), 1.00% (50/5000), 0.80% (40/5000), 0.76% (38/5000), 0.82% (41/5000) and 0.80% (40/5000) when interpolating vectors were used; and 1.48% (74/5000), 1.30% (65/5000), 1.26% (63/5000), 1.22% (61/5000), 1.22% (61/5000) and 1.22% (61/5000) when not used, respectively. The number of generated reference vectors was 161, 313, 642, 1881, 4195 and 5000; and the number of line segments after adjustment was 1312, 6131, 28 457, 231 321, 921 569 and 1 247 500, respectively. Incidentally, conventional learning method used in Section 4 can be interpreted as a special case where $\theta_L = 0$. It should also be noted that the nearest-neighbor method listed in Fig. 6 corresponds to the case where $\theta_L = 1.00$ is chosen without the use of interpolating vectors. Although the error rate can thus be further decreased by the use of a non-zero threshold value of θ_L, it is acquired at the expense of extremely large computational cost. Hence we chose a simpler case of $\theta_L = 0$ in this paper, because the computational cost is reasonably small, and also because it requires a smaller number of parameters when designing the network.

The method of interpolating vectors is well suited to the neocognitron. In the hierarchical network of the neocognitron, feature extraction by S-cells and toleration of positional shift by C-cells are repeated. The process of tolerating shift by the C-cells can also be interpreted as a blurring operation. Thus S-cells of the highest stage (U_{S4}) analyze the response of the C-cell layer of the preceding stage (U_{C3}), which has already been spatially blurred.

Feature vectors for intermediate deformed patterns between arbitrary two reference vectors (or reference patterns) can be well emulated by interpolating vectors (or interpolating patterns) produced by a linear combination of the two vectors, especially when the patterns are spatially blurred.

The same is true, not only for interpolating vectors, but also for extrapolating vectors. In Section 2.3, we mentioned that a better recognition rate is obtained if *extrapolating* vectors, as well as *interpolating* vectors, are used. (Although we call our method simply by the name of *interpolating* vectors, it is used as a general term including *interpolating* and *extrapolating* vectors).

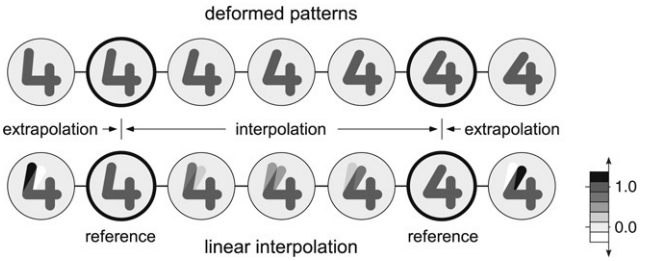

Fig. 10. Feature vectors of deformed patterns are well emulated by linear combinations of the pair of reference vectors. The use of extrapolating vectors, together with interpolating vectors, improves the recognition rate further.

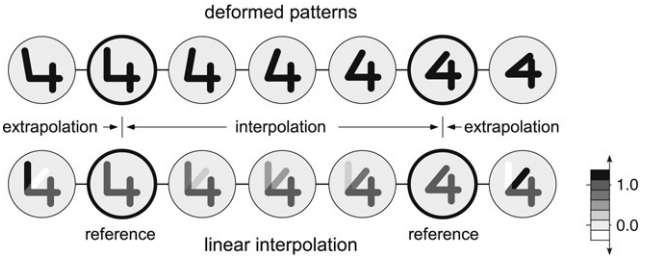

Fig. 11. It becomes difficult to emulate deformed patterns by interpolating/extrapolating vectors, when the similarity between the pair of reference vectors is not large enough, and/or when the amount of blur is not large enough.

This situation is illustrated in Fig. 10, where feature vectors are displayed expediently by original patterns presented to input layer U_0, instead of the actual responses of layer U_{C3}, for the sake of intuitive display. Lower half of Fig. 10 shows a series of interpolating and extrapolating vectors produced by linear combinations from a pair of reference vectors. Upper half of the figure illustrates examples of deformed patterns generated from the same pair of reference vectors. Each deformed pattern in the upper half shows a large similarity to the interpolating/extrapolating pattern below it. In other words, a deformed pattern (to be more exact, a feature vector for a deformed pattern) comes to show a large similarity to one of the interpolating or extrapolating vectors generated by a linear combination of the pair of reference vectors.

The left and right side of Fig. 10 illustrate examples of extrapolation. The pattern at the upper left of the figure, for example, is a pattern deformed from the left reference vector, not in the direction to its partner, but in the opposite direction. This pattern has a much larger similarity to the extrapolating vector (pattern) at the lower left, which is produced by a linear combination of the two reference vectors, where p or q in Eq. (3) is negative.

If the similarity between a pair of reference vectors is not large enough, or if the amount of blur is not large enough, however, interpolating or extrapolating vectors produced by linear combinations of reference vectors do not always emulate feature vectors of deformed patterns correctly enough, as is illustrated in Fig. 11.

We have demonstrated so far that the error rate can be decreased by the use of interpolating vectors that are distributed along lines connecting pairs of reference vectors. We then expected that the error rate might be decreased further, if interpolating vectors are allowed to be distributed on planes determined by triplets of reference vectors. An experiment under the same condition as that in Section 4, however, exhibited an error rate of 1.04% (52/5000) for the blind test set of 5000 patterns. Different from our expectation, the error rate by the use of triplets of reference vectors was slightly worse than the use of pairs of reference vectors. A possible reason for this result might be caused by a limitation of creating interpolating vectors by linear combinations. Creation of interpolating vectors by means of a method other than linear combinations might be a future problem to be considered.

Appendix. Neocognitron

The neocognitron was initially proposed as a neural network model of the mammalian visual system (Fukushima, 1980, 1988, 2003). It acquires the ability to recognize robustly visual patterns through learning. It has a hierarchical multilayered architecture similar to the classical hypothesis by Hubel and Wiesel (1962, 1965).

The neocognitron used for the simulation in this paper is almost the same as the conventional neocognitron (Fukushima, 2003). The main difference between the two resides in the highest stage, which is discussed in detail in the text. We then discuss below the mechanisms of lower stages, which are common to both the conventional and the present neocognitron.

A.1. Contrast-extracting cells

As was discussed in Section 3.1, the neocognitron consists of 4 stages of S- and C-cell layers (see Fig. 4).

The stimulus pattern is presented to the input layer (photoreceptor array) U_0. A layer of contrast-extracting cells (U_G), which correspond to retinal ganglion cells or lateral geniculate nucleus cells, follows input layer U_0.

Layer U_G has two cell-planes: one cell-plane consists of cells with concentric on-center receptive fields ($k = 2$ in Eq. (15) below), and the other consists of cells with off-center receptive fields ($k = 1$). The former cells extract positive contrast in brightness, whereas the latter extract negative contrast from the images presented to the input layer.

Let the output of a cell of U_0 be $u_0(\boldsymbol{n})$, where $\boldsymbol{n}$ represent the location of the cell. The output of a contrast-extracting cell of layer U_G, whose receptive field center is located at $\boldsymbol{n}$, is given by

$$u_G(\boldsymbol{n}, k) = \varphi\left[(-1)^k \sum_{|\boldsymbol{v}|<A_G} a_G(\boldsymbol{v}) \cdot u_0(\boldsymbol{n}+\boldsymbol{v})\right], \quad (k = 1, 2), \tag{15}$$

where $\varphi[]$ is a function defined by $\varphi[x] = \max(x, 0)$. Parameter $a_G(\boldsymbol{v})$ represents the strength of fixed connections to the cell and takes the shape of a Mexican hat. A_G denotes the radius of summation range of $\boldsymbol{v}$, that is, the size of spatial spread of the input connections $a_G(\boldsymbol{v})$.

The input connections to a single cell of layer U_G are designed in such a way that their total sum is equal to zero. In other words, the connection $a_G(\boldsymbol{v})$ is designed so as to satisfy

$$\sum_{|\boldsymbol{v}|<A_G} a_G(\boldsymbol{v}) = 0. \tag{16}$$

This means that the dc component of spatial frequency of the input pattern is eliminated in the contrast-extracting layer U_G. As a result, the output from layer U_G is zero in the area where the brightness of the input pattern is flat.

A.2. S-cells

S-cells have modifiable input connections, which are determined by learning. They work as feature-extracting cells after having finished the learning.

S-cells of layer U_{S1} come to work as edge-extracting cells like simple cells in the primary visual cortex. Layer U_{S1} has 16 cell-planes, each of which consists of edge-extracting cells of a particular preferred orientation. Preferred orientations of the cell-planes, namely, the orientations of the training patterns, are chosen at intervals of 22.5°.

S-cells of intermediate stages (U_{S2} and U_{S3}) come to extract more global features, such as parts of training patterns.

An S-cell receives variable excitatory connections from a group of C-cells of the preceding stage (see Fig. 5). The S-cell also receives a variable inhibitory connection from an inhibitory cell, called a V-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 of the output of the C-cells.

Let $u_{Sl}(\boldsymbol{n}, k)$ and $u_{Cl}(\boldsymbol{n}, k)$ be the output of S-cells and 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.[2] The output of the S-cell is given by[3]

$$u_{Sl}(\boldsymbol{n}, k) = \frac{1}{1-\theta_l} \cdot \varphi\left[\frac{\sum_{\kappa=1}^{K_{Cl-1}} \sum_{|\boldsymbol{v}|<A_{Sl}} a_{Sl}(\boldsymbol{v}, \kappa, k) \cdot u_{Cl-1}(\boldsymbol{n}+\boldsymbol{v}, \kappa)}{b_{Sl}(k) \cdot v_l(\boldsymbol{n})} - \theta_l\right], \tag{17}$$

[2] Although the pitch of cells in a cell-plane is the same for S- and C-cells, the locations of S- and C-cells are not necessarily exactly aligned. In some stages, they are staggered by half a pitch (Fukushima, 2003). In these stages, if S-cells are located on an integer grid $\boldsymbol{n}$, the locations $\boldsymbol{v}$ of C-cells relative to an S-cell are at the centers of the meshes of the grid. More specifically, if we write $\boldsymbol{v} = (v_x, v_y)$ in Eqs. (17) and (18), v_x and v_y do not take integer values but take integers plus 0.5. In the stage where S- and C-cells are exactly aligned, both v_x and v_y take integer values.

[3] Eq. (17) is slightly simplified mathematically than the corresponding equation used in the neocognitrons of previous versions (Fukushima, 1980, 1988, 2003).

where

$$v_l(\boldsymbol{n}) = \sqrt{\sum_{\kappa=1}^{K_{Cl-1}} \sum_{|\boldsymbol{\nu}|<A_{Sl}} c_{Sl}(\boldsymbol{\nu}) \cdot \{u_{Cl-1}(\boldsymbol{n}+\boldsymbol{\nu}, \kappa)\}^2} \tag{18}$$

is the output of the V-cell that sends inhibitory signals to the S-cell. Parameter $a_{Sl}(\boldsymbol{\nu}, \kappa, k)$ (≥ 0) is the strength of variable 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 a single S-cell. Parameter $b_{Sl}(k)$ (≥ 0) is the strength of variable inhibitory connection coming from the V-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 $|\boldsymbol{\nu}|$. The positive constant θ_l is the threshold of the S-cell and determines the selectivity in extracting features.

In (17) and (18) for $l = 1$, $u_{Cl-1}(\boldsymbol{n}, k)$ stands for $u_G(\boldsymbol{n}, k)$, and we have $K_{Cl-1} = 2$.

After having finished the learning, whose process is discussed in Appendix A.4, the output of the S-cell of Eq. (17) can be expressed also by vector notation like the one used in Section 3.4:

$$u_{Sl}(\boldsymbol{n}, k) = \frac{\varphi(s - \theta_l)}{1 - \theta_l}, \tag{19}$$

where s is the similarity between the reference vector $\boldsymbol{X}$ and the test vector $\boldsymbol{x}$.

This means that the response of an S-cell takes a maximum value equal to 1 when the test vector is identical to the reference vector, and becomes 0 if the similarity s is less than the threshold θ_l of the S-cell. In the multi-dimensional feature space, the area that satisfies $s < \theta_l$ becomes the tolerance area in feature extraction by the S-cell, and the threshold θ_l determines the radius of the tolerance area. The selectivity of an S-cell to its preferred feature (or the reference vector) can thus be controlled by the threshold θ_l.

Incidentally, Eqs. (17)–(19) hold also for the highest stage ($l = L$), where we have $\theta_L = 0$. In other words, they reduce to (8), (9) and (13), respectively, if we put $l = L$ and $\theta_L = 0$.

A.3. C-cells

The response of an S-cell layer U_{Sl} is spatially blurred in the succeeding C-cell layer U_{Cl}. Mathematically, the response of a C-cell of U_{Cl}, excluding the highest stage U_{C4}, is given by

$$u_{Cl}(\boldsymbol{n}, k) = \psi\left[\sum_{|\boldsymbol{\nu}|<A_{Cl}} a_{Cl}(\boldsymbol{\nu}) \cdot u_{Sl}(\boldsymbol{n}+\boldsymbol{\nu}, k)\right], \tag{20}$$

where $\psi[x] = \varphi[x]/(1 + \varphi[x])$. Parameter $a_{Cl}(\boldsymbol{\nu})$ represents the strength of the fixed connections converging from a group of S-cells to the C-cell, which spread within a radius of A_{Cl}. Basically, the input connections to a C-cell are excitatory and have a spatial distribution like a dome.

In some layers (U_{C1} and U_{C2}), however, inhibitory connections surround the excitatory ones (Fukushima, 2003). The concentric inhibitory surround in $a_{Cl}(\boldsymbol{\nu})$ endows the C-cells with the characteristics of end-stopped cells, and C-cells behave like hypercomplex cells in the visual cortex. In other words, an end of a line elicits a larger response from a C-cell than a middle point of the line. C-cells, whose input connections have inhibitory surrounds, can participate in extraction of bend points and end points of lines while they are making a blurring operation. This increases recognition rate of the network.

The inhibitory surrounds in the connections also have another merit. The blurring operation by C-cells, which usually is effective for improving robustness against deformation of input patterns, sometimes makes it difficult to detect whether a lump of blurred response is generated by a single feature or by two independent features of the same kind. For example, a single line and a pair of parallel lines of a very narrow separation generate a similar response when they are blurred. The inhibitory surround in the connections to C-cells creates a non-responding zone between the two lumps of blurred responses. This silent zone makes the S-cells of the next stage easily detect the number of original features even after blurring.

A.4. Self-organization of the network

The neocognitron is trained to recognize patterns through learning. Only S-cells in the network have their input connections modified through learning. In the specific network discussed in this paper, supervised learning is used for the lowest stage (U_{S1}), and unsupervised learning for intermediate stages (U_{S2} and U_{S3}).

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. The same set of training patterns is used for the training of all stages except layer U_{S1}. It should be noted that the same training set is used also for the learning of layer U_{S4} at the highest stage.

We discuss here the learning method for U_{S2} and U_{S3} first, and then the method for U_{S1}.

A.4.1. Competitive learning for intermediate layers

In the unsupervised learning, which is used for U_{S2} and U_{S3}, the self-organization of the network is performed using two principles. The first principle is a kind of *winner-take-all* rule: among the S-cells situated in a certain small area, which is called a "hypercolumn", only the one responding most strongly to a training pattern becomes the *winner*. The winner has its input connections strengthened.

The second principle for the learning is introduced in order that the connections are strengthened under the restriction of shared connections. The winner cell not only grows by itself, but also controls the growth of neighboring cells, working, so to speak, like a seed in crystal growth. To be more specific, all other S-cells in the cell-plane, from which the *seed cell* is selected, follow the seed cell, and have their input connections strengthened by having the same spatial distribution as those of the seed cell.

The amount of strengthening of each input connection to the winner, namely to the seed cell, is proportional to the intensity of the response of the cell from which the relevant connection leads.

Let cell $u_{Sl}(\hat{\boldsymbol{n}}, \hat{k})$ be selected as a seed cell at a certain time, the variable connections $a_{Sl}(\boldsymbol{\nu}, \kappa, \hat{k})$ to this seed cell, and, as a consequence of the shared connections, to all the S-cells in the same cell-plane as the seed cell, are increased by the following amount:

$$\Delta a_{Sl}(\boldsymbol{\nu}, \kappa, \hat{k}) = c_{Sl}(\boldsymbol{\nu}) \cdot u_{Cl-1}(\hat{\boldsymbol{n}} + \boldsymbol{\nu}, \kappa). \tag{21}$$

The inhibitory variable connection $b_l(\hat{k})$ from the V-cell is strengthened at the same time to the average strength of the excitatory connections $a_{Sl}(\boldsymbol{\nu}, \kappa, \hat{k})$. That is,

$$b_{Sl}(\hat{k}) = \sqrt{\sum_{\kappa=1}^{K_{Cl-1}} \sum_{|\boldsymbol{\nu}|<A_{Sl}} \frac{\{a_{Sl}(\boldsymbol{\nu}, \kappa, \hat{k})\}^2}{c_{Sl}(\boldsymbol{\nu})}}. \tag{22}$$

Incidentally, (21) and (22) are the same equations as (10) and (11), respectively.

The initial strength of the variable excitatory connections is very weak and nearly zero. Then, the variable excitatory connections to the S-cell grow into a *template* that exactly matches the spatial distribution of the response of the C-cells in the preceding layer. Through the excitatory connections, the S-cell receives signals indicating the existence of the relevant feature to be extracted. If an irrelevant feature is presented, the inhibitory signal from the V-cell becomes stronger than the direct excitatory signals from the C-cells, and the response of the S-cell is suppressed. The S-cell thus acquires the ability to extract a feature of the stimulus presented during the learning period.

Once an S-cell is selected as a seed cell and has learned to respond to a feature, the cell usually loses its responsiveness to other features. When a different feature is presented, a different cell usually yields the maximum output and learns the second feature. Thus, a *division of labor* among the cells occurs automatically.

For the learning of intermediate layers U_{S2} and U_{S3} of the neocognitron discussed in this paper, each training pattern of the training set was presented only once.

The neural networks' ability to recognize patterns robustly is influenced by the selectivity of feature-extracting cells, which is controlled by the threshold of the cells. Fukushima and Tanigawa (1996) have proposed the use of higher threshold values for feature-extracting cells in the learning phase than in the recognition phase, when unsupervised learning with a winner-take-all process is used to train neural networks. This method of dual threshold is used for the learning of layers U_{S2} and U_{S3}.

A.4.2. Supervised learning for the edge-extracting layer

In the supervised learning for U_{S1}, the *teacher* presents a training pattern to the input layer and points out the position of the feature that should be extracted. An S-cell whose receptive field center coincides with the position of the feature takes the place of the winner, and becomes the seed cell for that feature. The other process of learning is identical to that of the unsupervised learning, and occurs automatically.

To be more specific, the teacher presents a training pattern, namely a straight edge of a certain orientation, to the input layer of the network. The teacher then points out the location of the feature, which, in this particular case, can be an arbitrary point on the edge. The presentation of the training pattern, or an edge, to each cell-plane can be only once during the learning.

A.5. Parameters for the simulation

The scale of the network and the arrangement of cells in each layer is identical to the conventional neocognitron discussed by Fukushima (2003).

The total number of cells (not counting inhibitory V-cells) in each layer is as follows. U_0: 65×65, U_G: $71 \times 71 \times 2$, U_{S1}: $68 \times 68 \times 16$, U_{C1}: $37 \times 37 \times 16$, U_{S2}: $38 \times 38 \times K_{S2}$, U_{C2}: $21 \times 21 \times K_{C2}$, U_{S3}: $22 \times 22 \times K_{S3}$, U_{C3}: $13 \times 13 \times K_{C3}$, U_{S4}: $5 \times 5 \times K_{S4}$, and U_{C4}: $1 \times 1 \times 10$.

Although the number of cells in each cell-plane has been pre-determined for all layers, the number of cell-planes in each S-cell layer (K_{Sl}) is determined automatically in the learning phase depending on the training set. In each stage except the highest one, the number of cell-planes of the C-cell layer (K_{Cl}) is the same as K_{Sl}. The recognition layer U_{C4} has only $K_{C4} = 10$ cell-planes, and each cell-plane contains only one C-cell.

The sizes of connections converging to single cells (namely, radii A_G, A_{Sl}, A_{Cl}, etc.) are determined as follows, where the pitch of cells in the cell-plane of the preceding layer is taken as the unit of length.[4] In the contrast-extracting layer U_G, the radius of input connections $a_G(\boldsymbol{\nu})$, namely A_G in (15), is 3.3, and the positive center of the Mexican-hat is 1.2 in radius. For S-cell layers, the radius of input connections A_{Sl} is 3.4 for $l = 1, 2$ and 3. It is 4.9 for $l = 4$. As for C-cell layers, $A_{C1} = 9.4$, $A_{C2} = 7.4$, and $A_{C3} = 4.4$. The excitatory center of the connections $a_{Cl}(\boldsymbol{\nu})$ is 3.4 in radius for the lower layers ($l = 1, 2$). Connections $a_{C3}(\boldsymbol{\nu})$ for layer U_{C3} do not have inhibitory surrounds and consist of only excitatory components.

The radius of the hypercolumn used for the competition area in the learning is 3.1 for U_{S2} and U_{S3}.

The threshold values used in the experiment of Section 4 were as follows. For the edge-extracting layer U_{S1}, we chose $\theta_1 = 0.55$. For layers U_{S2} and U_{S3}, the thresholds in the recognition phase were: $\theta_2^{(\mathrm{R})} = 0.50$ and $\theta_3^{(\mathrm{R})} = 0.56$. Those in the learning phase were: $\theta_2^{(\mathrm{L})} = 0.64$ and $\theta_3^{(\mathrm{L})} = 0.67$. The numbers of cell-planes generated by the training set of 5000 patterns were $K_{S2} = 43$, $K_{S3} = 133$ and $K_{S4} = 160$ in the first experiment discussed in Section 4.

References

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2, 1–47.

[4] 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 stage, because the density of cells is lower in the higher stage.

Cover, T. M., & Hart, P. E. (1968). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *IT-4*(5), 515–516.

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. A computer program of this neocognitron in C language is available from Visiome Platform: http://platform.visiome.neuroinf.jp/.

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

Gray, R. M. (1984). Vector quantization. *IEEE ASSP Magazine*, *1*(2), 4–29.

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

Kohonen, T. (1995). *The self-organizing maps*. Berlin, Heidelberg, New York: Springer-Verlag.

Schölkoph, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., et al. (1997). Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, *45*, 2758–2764.