Invited Paper

Recent advances in the deep CNN neocognitron

Kunihiko Fukushima\textsuperscript{1a)}

\textsuperscript{1} Fuzzy Logic Systems Institute

680-41 Kawazu, Iizuka, Fukuoka 820-0067, Japan

\textsuperscript{a)} fukushima@m.ieice.org

Received October 19, 2018; Published October 1, 2019

Abstract: Deep convolutional neural networks (deep CNN) show a large power for robust recognition of visual patterns. The neocognitron, which was first proposed by Fukushima (1979), is a network classified to this category. Its architecture was suggested by neurophysiological findings on the visual systems of mammals. It acquires the ability to recognize visual patterns robustly through learning. Although the neocognitron has a long history, improvements of the network are still continuing. This paper discusses the recent neocognitron, focusing on differences from the conventional deep CNN. Some other functions of the visual system can also be realized by networks extended from the neocognitron, for example, recognition of partly occluded patterns, the mechanism of selective attention, and so on.

Key Words: deep CNN, pattern recognition, neocognitron, add-if-silent rule, interpolating-vector

1. Introduction

Recently, deep convolutional neural networks (deep CNN) have become very popular in the field of visual pattern recognition (e.g. [1]). The neocognitron, which was first proposed by Fukushima [2–4], is a network classified to this category. It is a hierarchical multi-layered network. Its architecture was suggested by neurophysiological findings on the visual systems of mammals. It acquires the ability to recognize visual patterns robustly through learning. Although the neocognitron has a long history, improvements of the network are still continuing.

For training intermediate layers of the neocognitron, the learning rule called \textit{AiS} (Add-if-Silent) is used. Under the AiS rule, a new cell is generated and added to the network if all postsynaptic cells are silent in spite of non-silent presynaptic cells. The generated cell learns the activity of the presynaptic cells in one-shot. Once a cell is generated, its input connections do not change any more. Thus the training process is very simple and does not require time-consuming repetitive calculation.

In the deepest layer, a method called \textit{IntVec} (Interpolating-Vector) is used for classifying input patterns based on the features extracted by intermediate layers. For the recognition by the IntVec, we search, in the multi-dimensional feature space, the nearest plane (or line) that is made of a trio (or pair) of reference vectors. Computer simulation shows that recognition error can be made much smaller by the IntVec than by the WTA (Winner-Take-All) or even by the SVM (support vector machine).
Some other functions of the visual system can also be realized by networks extended from the neocognitron, for example, recognition of partly occluded patterns, the mechanism of selective attention, and so on.

Sections from 2 to 5 discuss the recent neocognitron, focusing on differences from the conventional deep CNN. Sections 6 and 7 show some networks extended from the neocognitron.

2. Network architecture of the neocognitron

As shown in Fig. 1, the neocognitron consists of a cascaded connection of a number of stages preceded by an input layer $U_0$. Each stage is composed of a layer of S-cells followed by a layer of C-cells. After learning, S-cells come to work as feature-extracting cells. C-cells pool the response of S-cells in retinotopic neighborhood. Incidentally, S-cells behave like simple cells in the visual cortex, and C-cells behave like complex cells. In the figure, $U_{S_l}$ and $U_{C_l}$ indicate the layer of S-cells and the layer of C-cells of the $l$th stage, respectively.

![Fig. 1. The architecture of the neocognitron.](image)

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. A cell-plane is a group of cells that are arranged retinotopically and share the same set of input connections. 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.

The input connections to S-cells are variable and are modified through learning. After learning, each S-cell comes to respond selectively to a particular visual feature. The feature extracted by an S-cell is determined during learning. Generally speaking, local features of the input pattern are extracted in shallow layers, and more global features are extracted in deeper layers.

The output of layer $U_{S_l}$ is fed to layer $U_{C_l}$. C-cells pool the response of S-cells. Each C-cell has fixed excitatory connections from a group of S-cells whose receptive field locations are slightly deviated. C-cells thus average the responses of S-cells spatially. We can also interpret that S-cells’ response is spatially blurred by C-cells. Although deformation of an input pattern causes shift of local features, C-cells can absorb the effect of shift by spatial averaging.

In the neocognitron, the pooling by C-cells is done by spatial averaging (L2 pooling), not by MAX-operation. The spatial averaging is useful, not only for robust recognition of deformed patterns, but also for smoothing additive random noise contained in the responses of S-cells. To reduce the computational cost by decreasing the density of cells, down-sampling of cells is done in layers of C-cells. Spatial blur (i.e., low-pass filter for spatial frequency) before down-sampling is also important for suppressing aliasing noise caused by coarse down-sampling. The output of layer $U_{C_l}$ is fed to layer $U_{S_{l+1}}$ of the next stage.

Based on the features extracted in the intermediate stages, the final classification of the input pattern is made in the deepest stage.
3. Feature extraction by S-cells

3.1 Response of an S-cell

S-cells work as feature extractors. Let vector $a$ be the strength of excitatory connections to an S-cell from presynaptic C-cells, whose outputs are $x$ (Fig. 2). The V-cell receives signals from the same group of C-cells as does the S-cell and calculates the norm of $x$. Namely, $v = \|x\|$. 

![Connections converging to an S-cell.](image)

The output $u$ of the S-cell is given by

$$u = \frac{1}{1 - \theta} \cdot \varphi \left( (a, x) - \theta \|x\| \right)$$

where $\varphi[\cdot]$ is a rectified linear function, which is defined by $\varphi[x] = \max(x, 0)$. Connection $a$ is given by $a = \frac{X}{\|X\|}$, where $X$ is the training vector (or a linear combination of training vectors) that the S-cell has learned. Then, (1) reduces to

$$u = \|x\| \cdot \frac{\varphi[s - \theta]}{1 - \theta} \quad \text{where} \quad s = \frac{(X, x)}{\|X\| \cdot \|x\|}$$

In the multi-dimensional feature space, $s$ shows a kind of similarity between $X$ and $x$, which is defined by the inner product of $X$ and $x$. If similarity $s$ is larger than $\theta$, the S-cell yields a non-zero response [4]. Thus $\theta$ determines the threshold of the S-cell. The area that satisfies $s > \theta$ in the multi-dimensional feature space is named the tolerance area of the S-cell (Fig. 3). We call $X$ the reference vector of the S-cell. It represents the preferred feature of the S-cell.

![Tolerance area of an S-cell in the feature space [4].](image)

Now, the distance between two vectors can be defined by similarity $s$. In this paper, the nearest vector means the vector that has the largest similarity.

3.2 Robustness to background noise

Different from the neocognitron of old versions, the inhibition from V-cells works, not in a divisional manner, but in a subtractive manner. This is effective for increasing robustness to background noise [5].
Figure 4 illustrates a situation when the background noise is a faint image of other character. As shown in the lower left of the figure, each S-cell of an intermediate layer has a small receptive field and observes only a part of the input pattern.

If inhibition works in a divisional manner, an S-cell, whose receptive field (left circle on $U_0$) covers only a part of the faint background character, responds strongly despite of the weak intensity of the stimulus. As a result, S-cells in the intermediate layer respond as though an input pattern shown in the lower right in Fig. 4 is presented. Correct recognition of the foreground character is largely interfered by the strong responses from S-cells elicited by these irrelevant features.

On the contrary, if the inhibition works in a subtractive manner, the faint background noise elicits only negligibly small responses from S-cells. Thus, the responses of S-cells are little affected by the background noise. Incidentally, when the receptive field of an S-cell covers parts of both background and foreground characters together as shown by the upper circle in $U_0$, the S-cell’s response is almost the same, either with divisional or subtractive inhibition.

Figure 5 shows how the intensity of the background noise affects the recognition error by the neocog-
nitron. If S-cells have subtractive inhibition, the recognition error remains small until the intensity of the background noise becomes very large. If S-cells have divisional inhibition, however, even a slight background noise produces a large recognition error.

We also tested the effect of other types of background noise and had similar results. In one case, line segments are randomly located in the background. This emulates a situation where the image of printed entry column is seen under the target digits. In another case, a white noise is superimposed on the test digit.

4. Training intermediate layers

4.1 Add-if-Silent

For training intermediate layers of the neocognitron, an unsupervised learning rule called AiS (Add-if-Silent) is used [4, 6]. The learning of a layer starts after the learning of preceding layers has been finished. During learning, training patterns from a training set are presented one by one to the input layer $U_0$, and the response of layer $U_{C_l-1}$ works as a training stimulus for $U_{Sl}$.

Under the AiS rule, a new cell is generated and added to the network if all postsynaptic cells are silent in spite of non-silent presynaptic cells (Fig. 6). The strength of the input connections (namely, reference vector) of the generated S-cell is determined to be proportional to the response of the presynaptic C-cells at this moment. The learning is done in one-shot. Once a cell is generated, its input connections do not change any more.

Fig. 6. The add-if-silent rule. A new cell is generated when all post-synaptic cells are silent. The strength of the connections to the generated cell is set to be proportional to the intensity of the response of presynaptic cells.

Thus the training process is very simple and does not require time-consuming repetitive calculation. Presenting each training pattern only once is enough to complete the learning. This means that the learning by the AiS is a process of choosing a small set of reference vectors from the large set of training vectors.

Figure 7 illustrates an example of the progress of learning in time sequence. At $t = 0$, a new cell is generated, because there is no post-synaptic cell yet at this initial state. At $t = 1$, another new cell is generated, because the existing cell, which was generated at $t = 0$, is silent. At $t = 2$, the training

Fig. 7. Progress of learning by the AiS.
stimulus is just ignored and no cell is generated, because at least one cell shows a response to the training stimulus. At $t = 3$, all existing cells are silent, and a new cell is generated.

We now look at this learning process in the multi-dimensional feature space (Fig. 8). Under the AiS rule, no cell can be generated any more within the tolerance areas of existing S-cells, whose size is determined by the threshold $\theta$ of S-cells. As a result, reference vectors of generated S-cells come to distribute uniformly in the feature space after presentation of a large enough number of training vectors.

![Fig. 8.](image)

We can express this situation as follows. During the learning with the AiS, S-cells behave like grandmother cells (namely, gnostic cells). In other words, each training vector elicits a response from only one (or only a small number of) S-cell. This is useful for producing a uniform distribution of reference vectors of the generated S-cells in the feature space.

4.2 Dual threshold for S-cells

During the recognition phase, however, behavior like grandmother cells is not desirable for robust recognition of deformed patterns. If an S-cell, which has been the only active S-cell in a layer, stops responding by a slight deformation of the test vector, and if another S-cell comes to respond instead, the layer comes to exhibit a completely different response.

For robust recognition of deformed patterns, it is desirable that some number of cells respond together to an input pattern. In other words, situation like a sparse population coding is required.

We then use dual threshold for S-cells: After finishing the learning, the threshold of S-cells is set to a lower value than the threshold for the learning [7]. We can thus produce a situation like a population coding during the recognition phase.

We can understand that S-cells of a low threshold produces a blur (or pooling) in the feature space, while C-cells produces a blur in the retinotopic space.

The reason why a good recognition rate can be obtained with simple algorithm of the AiS can be explained as follows. The final classification of input patterns is made, not by an intermediate layer, but by the deepest layer of the network. The role of intermediate layers is to represent an input pattern accurately, not by the response of a single cell, but by the population coding. In the case of
population coding, best-fitting of individual cells to training stimuli is not necessarily important. It is enough if the input pattern is accurately represented by the response of the whole cells.

4.3 AiS with feedback
To apply the AiS (Add-if-Silent) rule to a neocognitron, a slight modification is required because the neocognitron is a CNN (convolutional neural network). Each layer of the neocognitron consists of a number of cell-planes. In a cell-plane, all cells are arranged retinotopically, and share the same set of input connections. This condition of shared connections has to be kept even during the learning.

In the neocognitron, generation of a new S-cell means the generation of a new cell-plane. All cells in the cell-plane come to have the same input connections as the generated S-cell.

Suppose a training pattern is presented to input layer $U_0$. Here, the response of the C-cells of $U_{CL-1}$ works as the training stimulus for $U_{SL}$. The AiS rule is applied at the retinotopic location where all post-synaptic S-cells are silent in spite of non-silent presynaptic C-cells. If there are a number of such locations, we have to choose one of them. We use negative feedback signals from non-silent S-cells for this purpose [8, 9]. The strength of negative feedback connections from an S-cell is the same as that of the feed-forward connections converging to the S-cell. Negative feedback signals from active postsynaptic S-cells thus inhibit the presynaptic C-cells that have already contributed for eliciting responses from S-cells. We then apply the AiS rule at the location where the response of inhibited C-cell is the largest.

After generation of a new cell-plane with the AiS rule, if there still remains any area in which presynaptic C-cells have not been inhibited yet, the same process of generating a cell-plane is repeated. We can thus choose all retinotopic locations where important features exist.

It should be noted here, however, that the feedback is used only for determining the retinotopic location where the AiS rule is to be applied, and that the reference vector of the generated S-cell (seed-cell) is set to be proportional to the response of the presynaptic C-cells before being suppressed by the negative feedback.

5. Deepest layer
In the deepest layer (namely, highest layer) $U_{SL}$, training S-cells of is done by a supervised learning. The reference vectors of S-cells are created in such a way that a large number of training vectors of each class can be represented by a small number of reference vectors. Each reference vector is made of a weighted sum of training vectors of the same class and has a label of the class name.

Here we first discuss the method of recognition, before discussing the detailed method of learning.

5.1 Interpolating-vector
5.1.1 Int-2
Test patterns presented to input layer $U_0$, are classified in the deepest layer $U_{SL}$, based on the features extracted by intermediate layers (Fig. 1). For this purpose, a method named Interpolating-Vector (IntVec) is used [10].

Let $x$ be the input signals to an S-cell of $U_{SL}$, which are the response of C-cells of $U_{CL-1}$. In the deepest layer, threshold $\theta$ of S-cells is set to $\theta = 0$. Hence, from (2), the response of an S-cell is given by $u = s \cdot \|x\|$, where $s = (X, x)/\{\|X\| \cdot \|x\|\}$ is the similarity between $x$ and reference vector $X$ of the S-cell.\footnote{For the economy of computational cost, analysis of the response of S-cells of $U_{SL}$ is actually performed, not at all retinotopic locations, but only at the location where $v = \|x\|$ (the response of the V-cell) is the largest. This means that $\|x\|$ is the same for all S-cells to be analyzed.}

In the multidimensional feature space, we assume lines connecting every pair of reference vectors of the same label (Fig. 9). Every line is assigned the same label as the reference vectors that span the line. We then measure distances (based on similarity $s$) to these lines from test vector $x$.

Among all lines that connect every pair of reference vectors of the same label, we search the one that has the largest similarity to the test vector. In other words, we search the line nearest to the test vector.
vector. The label of the nearest line (namely, the line that has the largest similarity to \( x \)), instead of the nearest reference vector, shows the result of pattern recognition.

The process of searching the nearest line can be expressed mathematically as follows. Let \( X_i \) and \( X_j \) be two reference vectors of the same label (Fig. 10 (Int-2)). Let \( \xi \) be a vector that is given by a linear combination of this pair of vectors. It is named an interpolating vector, and represents a point on the line connecting the two vectors:

\[
\xi = p_i \frac{X_i}{\|X_i\|} + p_j \frac{X_j}{\|X_j\|}, \quad (p_i + p_j = 1)
\]  

Under possible combinations of \( p_i \) and \( p_j \), the similarity between \( \xi \) and test vector \( x \) takes a maximum value

\[
s_{\text{line}} = \sqrt{s_i^2 - 2s_is_js_{ij} + s_j^2} \quad (1 - s_{ij})
\]  

at

\[
p_i = \frac{s_i - s_is_{ij}}{(s_i + s_j)(1 - s_{ij})}, \quad p_j = \frac{s_j - s_is_{ij}}{(s_i + s_j)(1 - s_{ij})}
\]

where

\[
s_i = \frac{(X_i, x)}{\|X_i\| \cdot \|x\|}, \quad s_j = \frac{(X_j, x)}{\|X_j\| \cdot \|x\|}, \quad s_{ij} = \frac{(X_i, X_j)}{\|X_i\| \cdot \|X_j\|}
\]

We can interpret that \( s_{\text{line}} \) represents similarity (or distance) between test vector \( x \) and line \( X_iX_j \).

**Fig. 9.** Recognition by the IntVec (Int-2). The test vector is classified, not to class B, but to class A, because the nearest line is chosen instead of the nearest reference vector [4].

**Fig. 10.** IntVec from two vectors (Int-2) and from three vectors (Int-3) [9].

Figure 11 shows some examples of patterns that were recognized, erroneously by the WTA, but correctly by the IntVec (Int-2).

Why the IntVec is powerful? With the IntVec, a sequence of deformed patterns are emulated by a sequence of patterns generated by linear interpolation (Fig. 12). If this operation is applied directly
to input patterns, however, the accuracy of emulation cannot be so high. In the IntVec, this operation is applied, not directly to input patterns, but to extracted features. Hence emulation with a high accuracy becomes possible.

Not only in the neocognitron but also in most deep neural networks, the recognition error can usually be reduced by increasing the number of training patterns. It is reported that, if a sufficiently large number of training patterns are not available, even the use of artificially generated training patterns to cover the shortage is effective. We can understand that the IntVec produces this situation, not during the training phase, but during the recognition phase. Namely, the number of extracted features is virtually increased during the recognition phase, without increasing the number of training patterns themselves.

Computer simulation has shown that recognition error can be made much smaller by the IntVec than by the WTA or even by the SVM [9, 11].

This method of IntVec is named \textit{Int-2}, because it uses two reference vectors.

5.1.2 Extension to Int-3 and Int-4

The Int-2 can be extended to \textit{Int-3} [9] or \textit{Int-4} [11]. In the Int-3, we assume planes spanned by every trio of reference vectors of the same label (Fig. 10 (Int-3)). Similarly in the Int-4, we assume hyperplanes spanned by every tetrad of reference vectors of the same label. Although we use planes or hyperplanes instead of lines, the rest of the process is the same as that for the Int-2. The computational cost increases a little, but the Int-3 yields a better recognition rate than the Int-2. The Int-4 produces a much better recognition rate.

The mathematical process of the Int-3 can be expressed as follows. Let $X_i$, $X_j$, and $X_k$ be three reference vectors of the same label. Let $\xi$ be an interpolating vector that is given by a linear combination of this trio of vectors. It represents a point on plane $X_iX_jX_k$. 

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig11.png}
\caption{Some examples of patterns (middle row) that were recognized, erroneously by the WTA (upper row), but correctly by the IntVec (Int-2, bottom row). (modified from [10])}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig12.png}
\caption{Pattern recognition by the IntVec. In the IntVec, a sequence of deformed patterns is emulated by a sequence of patterns generated by linear interpolation. This operation is applied, not directly to input patterns, but to extracted features.}
\end{figure}
\[ \xi = p_i \frac{X_i}{\|X_i\|} + p_j \frac{X_j}{\|X_j\|} + p_k \frac{X_k}{\|X_k\|}, \quad (p_i + p_j + p_k = 1) \] (7)

Under possible combinations of \( p_i, p_j \) and \( p_k \), the similarity between \( \xi \) and \( x \) takes maximum value

\[ s_{\text{plane}} = \sqrt{s S^{-1} s^T} \] (8)

at

\[ p = \frac{S^{-1} s^T}{(1 1 1) S^{-1} s^T} \] (9)

where \( S^{-1} \) is the inverse matrix of \( S \), and the following notation is used:

\[ p = (p_i, p_j, p_k) \] (10)
\[ s = (s_i, s_j, s_k) \] (11)
\[ S = \begin{pmatrix}
  s_{ii} & s_{ij} & s_{ik} \\
  s_{ji} & s_{jj} & s_{jk} \\
  s_{ki} & s_{kj} & s_{kk}
\end{pmatrix} \] (12)

where

\[ s_i = \frac{(X_i, x)}{\|X_i\| \cdot \|x\|}, \quad s_j = \frac{(X_j, x)}{\|X_j\| \cdot \|x\|}, \quad s_k = \frac{(X_k, x)}{\|X_k\| \cdot \|x\|} \]
\[ s_{ij} = \frac{(X_i, X_j)}{\|X_i\| \cdot \|X_j\|}, \quad s_{ik} = \frac{(X_i, X_k)}{\|X_i\| \cdot \|X_k\|}, \quad s_{jk} = \frac{(X_j, X_k)}{\|X_j\| \cdot \|X_k\|} \] (13)

Hence, \( s_{ii} = s_{jj} = s_{kk} = 1, \ s_{ji} = s_{ij}, \ s_{ki} = s_{ik} \) and \( s_{kj} = s_{jk} \).

We can easily extend these expressions to IntVec of any number of reference vectors; not only for Int-3 (and Int-2), but also for Int-4 or more.

### 5.2 Margined WTA

One of the important roles of learning is to produce a compact set of reference vectors (namely, S-cells) that can accurately represent the large set of training vectors. To use IntVec, we have to search the nearest line or plane among all combinations of two or three reference vectors. If the number of reference vectors can be made small, we can easily search the nearest line or plane without consuming a heavy computational cost. This is important especially when the Int-3 or Int-4 is used for recognition.

Similarly to intermediate layers, S-cells are generated during learning. In the deepest layer \( U_{SL} \), however, each S-cell (namely, its reference vector) has a label indicating the class name, to which it is to be classified. Hence the training of the deepest layer is done by a supervised learning.

The result of classification by the IntVec (as well as by other methods, say, WTA or SVM), is mainly affected by reference vectors located near borders between different classes. Hence it is desired that the reference vectors of each class are generated so as to distribute more densely near class borders than near the center of its cluster.

#### 5.2.1 Generating new cells by mWTA

Learning of the deepest layer is performed in two steps. In the first step, S-cells (reference vectors) are just generated, but their connections are not tuned. Tuning connections (namely, tuning reference vectors) starts in the second step, when all S-cells have already been generated in the layer.

One of the simplest methods for generating cells is the use of WTA (Winner-Take-All). Every time when a training vector is presented, we search the nearest reference vector (namely, the largest-output S-cell). If the nearest reference vector has a label different from that of the training vector, we judge that the classification by the WTA is wrong. If the classification by the WTA is wrong, a new reference vector (namely, a new S-cell) is generated at the location of the training vector. In other words, the
training vector itself is adopted as the new reference vector. If the classification is correct, however, the training vector is just ignored, and no reference vector is generated.

Here we propose, not to use the WTA directly, but to introduce a certain amount of margin for the WTA. We call this **margined Winner-Take-All (mWTA)** [9, 11]. In other words, in the search for the nearest reference vector during the learning, we impose a handicap on reference vectors of classes other than the training vector.

Figure 13 illustrates this situation in the multidimensional feature space. Let \( x \) be a training vector of class A. We search the nearest reference vectors of class A and B (B means all classes other than A), and compare the distances to them. Under the conventional WTA, we judge whether \( x \) is classified correctly or erroneously, depending on the distances to A and B. In the case of Fig. 13, \( x \) is judged to be recognized correctly by the WTA, because it is located below the broken line, which represents the border of Volonoi partition.

![Fig. 13.](image)

**Fig. 13.** Suppose a training vector \( x \) of class A is presented. We search the nearest reference vectors of classes A and B. Under the mWTA, the border hyperplane is shifted toward the nearest vector of A, from the hyperplane that has equal distances from A and B. Then, \( x \) is judged to be classified erroneously if it is located above the shifted hyperplane.

Under the mWTA, however, the border hyperplane is shifted downward (namely, toward A) by a certain amount. If the training vector \( x \) is located above the shifted hyperplane, \( x \) is judged to be classified erroneously, and a new reference vector (namely, a new S-cell) of class A is generated. Training vector \( x \) is adopted as the reference vector of the generated cell. This means that training vectors located near the border between different classes have a larger chance of being adopted as new reference vectors than training vectors located near the center of a cluster. This helps to produce a situation where reference vectors distribute more densely near the border between different classes than near a center of a cluster. It should be noted here that shift of border hyperplane is made only during the first step of the learning, and is not done during the recognition when the learning has been finished.

Thus, during the first step of learning, reference vectors are just chosen (or sampled) from vectors in the training set.

### 5.2.2 Tuning connections

Different from intermediate layers, the tuning of chosen reference vectors is effective for reducing the recognition error, because, in the deepest layer, the test vector is classified based on similarities to individual labeled reference vectors. The tuning is done by adding training vectors to reference vectors selected by the IntVec.

It is important, however, that fine tuning of reference vectors (namely, input connections to S-cells)
should not start from the beginning of the learning. The tuning is done by adding training vectors to selected reference vectors. If we start tuning from the beginning of the learning, when a sufficiently large number of reference vectors have not been generated yet, there is a large risk of distorting reference vectors that have already been generated. Even if a trio (in the case of Int-3) of reference vectors happened to span the nearest plane, there is a possibility that the three reference vectors are not relevant enough to the training vector.

We then separate the learning process of the deepest layer into two steps (Fig. 14). In the first step, cells are generated following the process of mWTA discussed above in 5.2.1. Thus, reference vectors are just chosen (or sampled) from vectors in the training set. The generated reference vectors are not tuned during the first step.

We start tuning reference vectors from the second step, when a large number of reference vectors have already been generated. The amount of tuning is controlled depending on the relative location of $x$. The nearer $x$ to $B$ is, the larger become the amount of tuning.

6. Recognition of partly occluded patterns

Various extensions and modifications of the neocognitron have been proposed. Here, we show some of them.

Human beings are often able to read or recognize a letter or word contaminated by ink stains that partly occlude the letter. If the stains are completely erased and the occluded areas of the letter are changed to white, however, we usually have difficulty in reading the letter, which now has some missing parts. For example, the patterns in Fig. 15(a), in which the occluding objects are not visible, are almost illegible, but the patterns in Fig. 15(b), in which the occluding objects are visible, are much easier to read.

![Fig. 14. Two-step learning. 1st step: generate new reference vectors by mWTA. 2nd step: tune generated vectors by IntVec. In the figure, $s_A$ and $s_B$ are similarity between $x$ and the nearest reference vectors of classes A and B, respectively. $s_{AB}$ is similarity between these nearest reference vectors of classes A and B. (modified from [11])](image)

We start tuning reference vectors from the second step, when a large number of reference vectors have already been generated. The amount of tuning is controlled depending on the relative location of $x$. The nearer $x$ to $B$ is, the larger become the amount of tuning.

**Fig. 15.** (a) Patterns partly occluded by invisible masking objects are difficult to recognize. (b) It becomes much easier to recognize when the occluding objects are visible [12].
Why can we recognize occluded patterns easier when the occluding objects are visible? Visual patterns have various local features, such as edges and corners. The visual system of animals extracts these features in its lower stages and tries to recognize a pattern using the information of extracted local features. When a pattern is partly occluded, a number of new features, which do not exist in the original pattern, are generated.

If the occluding objects are not visible, the visual system has difficulty in distinguishing which features are relevant to the original pattern, and which are not. These irrelevant features largely disturb the correct recognition by the visual system.

On the other hand, if the occluding objects are visible, the visual system can easily distinguish relevant from irrelevant features, and can ignore irrelevant features. Since the visual system has a large tolerance to partial absence of relevant features, it can recognize the partly occluded patterns correctly, even though some relevant features are missing.

Figure 16 shows another example of stimuli, in which the perception is largely affected by the placement of occluding objects. The black parts of the patterns are actually identical in shape between the left and right figures. We feel, however, as though different black patterns are occluded by gray objects. Namely, we perceive a partly occluded pattern ‘R’ in the left figure, while an occluded pattern ‘B’ in the right.

![Fig. 16. Identical patterns are perceived differently by the placement of different gray objects [13].](image1)

Here we show a network that behaves like a human vision [12]. The features extracted near the contour of occluding objects are usually irrelevant to the occluded pattern. To eliminate these irrelevant features, a new layer $U_M$, named a *masker layer*, is added to a neocognitron as shown in Fig. 17. The masker layer detects and responds only to occluding objects. The shape of the occluding objects appears in $U_M$, in the same shape and at the same location as in the input layer $U_0$. There are retinotopically ordered and slightly diverging inhibitory connections from layer $U_M$ to all cell-planes of layer $U_{S1}$. The inhibitory signals from $U_M$ suppress the responses to features irrelevant to the occluded pattern. Hence only local features relevant to the occluded pattern are transmitted to higher (namely, deeper) stages of the network. The neocognitron, like biological visual systems, has a tolerance to

![Fig. 17. Layer $U_M$, which is called the masker layer, is added to a neocognitron. This network can recognize partly occluded patterns correctly. (modified from [12])](image2)
partial absence of relevant features. Thus, the neocognitron with masker layer can recognize partly occluded patterns correctly. Figure 18 shows how the signals from relevant and irrelevant signals flow in the network.

![Visual system: Neural network model](image)

**Fig. 18.** Process of recognizing an occluded pattern. (modified from [13]).

For example, the pattern in the left of Fig. 15(a), in which the occluding object is not visible, is erroneously recognized as ‘I’, but the pattern below it, in which the occluding object is visible, is correctly recognized as ‘A’. This network can also recognize patterns in Fig. 16 as ‘R’ and ‘B’, like human beings.

7. Selective attention by backward signals

7.1 Outline of the selective attention model

Although the neocognitron has considerable ability to recognize deformed patterns, it does not always recognize patterns correctly when two or more patterns are presented simultaneously. The selective attention model has been proposed to eliminate these defects [14]. In the selective attention model, top-down (i.e., backward) connections are added to the neocognitron-type network, which had only bottom-up (i.e., forward) connections. (Here, bottom-up means the direction of signal flow from input layer $U_0$ to the deepest layer).

When a composite stimulus, consisting of two patterns or more, is presented, the model focuses its attention selectively to one of the patterns, segments it from the rest, and recognizes it. After the identification of the first segment, the model switches its attention to recognize another pattern. The model also has the function of associative recall. Even if noise or defects affect the stimulus pattern, the model can recognize it and restore the complete pattern from which the noise has been eliminated and defects corrected. These functions can be successfully performed even for deformed versions of training patterns, which have not been presented during learning.

With the selective attention model, not only recognition of patterns, but also the filling-in process for defective parts of imperfect input patterns works on the deformed and shifted patterns themselves. The selective attention model can repair a deformed pattern without changing its basic shape and its location. The deformed patterns themselves can thus be repaired at their original locations, preserving their deformation.

7.2 Architecture of the selective attention model

We now explain the architecture of the model in more detail. As illustrated in Fig. 19, cells in the top-down path are arranged making pairs with the cells in the bottom-up path. In the figure, $W$ indicates a layer of cells in the top-down path, while $U$ indicates a layer of cells in the bottom-up
Fig. 19. Interaction of bottom-up and top-down signals in the selective attention model. (modified from [4])

path. The top-down connections also make a mirror image with the bottom-up connections. The difference between the top-down and bottom-up connections is only in the direction of signal flow.

The bottom-up signals manage the function of pattern recognition, while the top-down signals manage the function of selective attention and associative recall. The output of the highest stage of the bottom-up path is sent back to lower stages through the top-down path and reaches the recall layer $U_0$ at the lowest stage. The bottom-up and top-down signals interact with each other at every stage of the hierarchical network, and the top-down signals are controlled so as to trace the same route as the bottom-up signals.

In the bottom-up path, which has the same architecture as the neocognitron, a C-cell receives excitatory connections from a group of S-cells. In a usual operating condition, however, it is only a small number of S-cells that actually send non-zero bottom-up signals. If top-down signals from a $W_C^l$-cell (C-cell in the $l$th stage of the top-down path) simply flow through strong connections, we have only blurred signals in layer $W_{S^l}$. To make the top-down signals flow retracing the same route as the bottom-up signals, $U_{S^l}$-cells send gate signals to corresponding $W_{S^l}$-cells.

At the same time, the top-down signals, that is, the signals for selective attention, have a facilitating effect on the bottom-up signals by controlling the gain of $U_C$-cells. When two or more patterns are simultaneously presented to the input layer, a number of cells (recognition cells) might be activated at first in the highest stage of the bottom-up path. However, these recognition cells, except one, stop responding gradually while signals are circulating through the feedback loop because of competition by lateral inhibition. Then only the bottom-up signals relevant to a single pattern are kept flowing by the facilitation from the top-down signals. This means that attention is selectively focused on only one of the patterns in the stimulus.

The lowest stage $W_0$ of the top-down path works as the recall layer, where the output of associative recall and the result of segmentation appear. Guided by the bottom-up signal flow, the top-down signals reach exactly the same locations at which the input pattern is presented. The response of the recall layer $W_0$ is fed back positively to the input layer $U_0$.

When some part of the input pattern is missing and a feature which is supposed to exist there fails to be extracted in the bottom-up path, the top-down signal flow is interrupted there and cannot go down any more, because gate signals from the bottom-up path do not come. In such a case, the threshold of $U_{S^l}$-cells around there is automatically lowered, and the $U_{S^l}$-cells try to extract even vague traces of the undetected feature. Incidentally, the fact that a feature has failed to be extracted is detected by the condition that a $W_C^l$-cell in the top-down path is active but that the corresponding $U_{S^l}$-cells in the bottom-up path are not. Once a feature is thus extracted in the bottom-up path, the top-down signal now can be transmitted further to lower stages through the route unlocked by the newly activated bottom-up cell. Hence a complete pattern in which defective parts are interpolated emerges in the recall layer. From this pattern, noise and blemishes have been eliminated, because
top-down signals are not fed back there.

7.3 Responses of the selective attention model

Figure 20 shows some examples of the response of the selective attention model in a time sequence. Layer $U_{C3}$ (C-cell layer of the highest stage of the bottom-up path) shows the result of pattern recognition. The segmented and/or restored pattern appears one by one in $W_0$. Incidentally, training patterns that were used to train the network are shown in the right of Fig. 20(a).

Figure 20(a) shows the response to a stimulus consisting of two juxtaposed patterns, ‘2’ and ‘3’. In

(a) Each pattern in the stimulus is sequentially recognized ($U_{C3}$) and segmented ($W_0$).

(b) From an ambiguous stimulus, several patterns are recognized sequentially. Each pattern is segmented and missing parts are gradually restored in $W_0$.

(c) From a noisy stimulus, pattern ‘2’ is recognized ($U_{C3}$) and gradually restored in $W_0$.

(d) Even if two identical patterns ‘2’ are presented together in parallel, they are separately recognized and restored sequentially.

Fig. 20. Some examples of the response of the selective attention model shown in a time sequence. Training patterns that were used to train the network are shown in the right of figure (a).
the recognition layer $U_{C3}$, the cell corresponding to pattern ‘2’ happens to be activated at first ($t = 0$). This signal is fed back to the recall layer $W_0$ through a top-down path, but the middle part of the segmented pattern ‘2’ is missing because of interference from the closely adjacent ‘3’. However, the interference soon decreases and the missing part recovers, because the signals for pattern ‘3’, which is not being attended to, are gradually attenuated without receiving facilitation by gain-control signals ($t = 4$). At $t = 5$, the top-down signal-flow is interrupted for a moment to switch attention. Since the gain-control signals from the top-down cells stop, the bottom-up routes for pattern ‘2’, which have so far been facilitated, now lose their conductivity because of fatigue. The recognition cell for pattern ‘3’ is now activated. Since top-down signals are fed back from this newly activated recognition cell, pattern ‘3’ is segmented and emerges in $W_0$.

Figure 20(b) shows how several patterns in an ambiguous stimulus are recognized and segmented sequentially. Pattern ‘4’ is isolated first, pattern ‘2’ next, and finally pattern ‘1’ is extracted. The recalled pattern ‘4’ initially has one part missing ($t = 0$), compared with the training pattern shown in the right of Fig. 20(a). However, the missing part is soon restored ($t = 4$). Each pattern is thus segmented and missing parts are gradually restored in $W_0$.

Figure 20(c) shows the response to a greatly deformed pattern with several parts missing and contaminated by noise. Because of the large difference between the stimulus and the training pattern, no response is elicited from the recognition layer $U_{C3}$ at first ($t = 0$). Accordingly, no top-down signal reaches the recall layer $W_0$. The no-response detector detects this situation, and a threshold-control signal is sent to all feature-extracting cells ($U_S$-cells) in the network, which makes them respond more easily even to incomplete features. Thus, at time $t = 2$, the recognition cell for ‘2’ is activated in $U_{C3}$, and top-down signals are fed back from it. In the pattern now sent back to the recall layer $W_0$, noise has been completely eliminated, and some missing parts have begun to be restored. This partly restored signal, namely the output of the recall layer $W_0$, is again fed back positively to the input layer $U_0$. The interpolation continues gradually while the signal circulates through the feedback loop, and finally the missing parts of the stimulus are completely filled in. It should be noted here that the horizontal bar at the bottom of pattern ‘2’ is shorter in the restored pattern than in the training pattern. This means that the length of the bar of the stimulus pattern is kept intact even after restoration. The missing parts are restored quite naturally, where the style of writing of the stimulus pattern is kept as faithful as possible, and only indispensable missing parts are restored.

As shown in Fig. 20(d), even if two or more identical patterns are presented together, they are separately recognized and restored in a time sequence.

### 8. Conclusions

This paper has discussed recent advances of the neocognitron and several networks extended from it. The neocognitron is a network suggested from the biological brain.

The author feel that the deep learning is not the only way to realize networks like, or superior to, the biological brain. To make further advances in the research, it is important to learn from the biological brain. There should be several algorithms that control the biological brain. It is now important to find out these algorithms and apply them to the design of more advanced neural networks.

### References

[1] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.

[2] K. Fukushima, “Neural network model for a mechanism of pattern recognition unaffected by shift in position — Neocognitron,” (in Japanese), *Trans. IEICE*, vol. J62-A, no. 10, pp. 658–665, October 1979.

[3] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, April 1980.

[4] K. Fukushima, “Artificial vision by multi-layered neural networks: Neocognitron and its advances,” *Neural Networks*, vol. 37, pp. 103–119, January 2013.
[5] K. Fukushima, “Increasing robustness against background noise: Visual pattern recognition by a neocognitron,” *Neural Networks*, vol. 24, no. 7, pp. 767–778, September 2011.

[6] K. Fukushima, “Training multi-layered neural network neocognitron,” *Neural Networks*, vol. 40, pp. 18–31, 2013.

[7] K. Fukushima and M. Tanigawa, “Use of different thresholds in learning and recognition,” *Neurocomputing*, vol. 11, no. 1, pp. 1–17, May 1996.

[8] K. Fukushima, “One-shot learning with feedback for multi-layered convolutional network,” *ICANN 2014, LNCS 8681*, eds. S. Wermter, C. Weber, W. Duch, T. Honkela, P. Koprinkova-Hristova, S. Magg, G. Palm, and A.E.P. Villa, pp. 291–298, Springer International Publishing, Switzerland, 2014.

[9] K. Fukushima and H. Shouno, “Deep convolutional network neocognitron: Improved interpolating-vector,” *IJCNN 2015*, pp. 1603–1610, July 2015.

[10] K. Fukushima, “Interpolating vectors for robust pattern recognition,” *Neural Networks*, vol. 20, no. 8, pp. 904–916, October 2007.

[11] K. Fukushima, “Margined winner-take-all: New learning rule for pattern recognition,” *Neural Networks*, vol. 97, pp. 152–161, January 2018.

[12] K. Fukushima, “Recognition of partly occluded patterns: A neural network model,” *Biological Cybernetics*, vol. 84, no. 4, pp. 251–259, 2001.

[13] K. Fukushima, “Restoring partly occluded patterns: A neural network model,” *Neural Networks*, vol. 18, no. 1, pp. 33–43, January 2005.

[14] K. Fukushima, “Neural network model for selective attention in visual pattern recognition and associative recall,” *Applied Optics*, vol. 26, no. 23, pp. 4985–4992, December 1987.