Is the Neocognitron Capable of State-of-the-art Digit Recognition?

David R. Lovell, Tom Downs & Ah Chung Tsoi

Abstract— **We describe a sequence of experiments investigating the strengths and limitations of Fukushima's neocognitron as a handwritten digit classifier. Using the results of these experiments as a foundation, we propose and evaluate improvements to Fukushima's original network in an effort to obtain recognition performance on a par with state-of-the-art digit recognition systems.**

The neocognitron's performance is shown to be highly dependent on the choice of selectivity parameters and we present two methods to adjust these variables. Performance of the network under the more effective of the two selectivity adjustment techniques suggests that the network fails to exploit the features that distinguish different classes of input data. To avoid this shortcoming, the network's final layer cells were replaced by a nonlinear classifier (a multilayer perceptron) to create a hybrid architecture.

Tests of Fukushima's original system and the novel systems proposed in this paper suggest that the neocognitron is unable to achieve the performance of existing digit classifiers due to its reliance upon the supervisor's choice of selectivity parameters and training data. These findings pertain to Fukushima's *implementation* of the system and should not be seen as diminishing the practical significance of the concept of hierarchical feature extraction embodied in the neocognitron.

Keywords— **Neocognitron, handwritten character recognition, selectivity**

I. Introduction

The *neocognitron* [1, 2, 3, 4, 5, 6, 7] is a massively parallel, hierarchical neural network, designed, primarily, for 2-D pattern recognition. Proposed by Fukushima in 1979 [1], it was inspired by Hubel and Wiesel's serial model of biological vision [8] and, for the last decade, it has been acclaimed as a shift and distortion tolerant character recognition system.

Some of the neocognitron's biological plausibility was sacrificed in 1983 when Fukushima moved away from the original paradigm of self-organization and introduced a supervised training scheme in an effort to improve the network's handwritten character recognition performance [2]. However, to the best of our knowledge, there have been no concrete performance statistics published to indicate whether the desired improvement was achieved. Unlike Hubel and Wiesel's serial model of vision, which has undergone rigorous scrutiny to test its validity, the capabilities of the neocognitron have not been critically reviewed to any significant extent.

The neocognitron was proposed well before multi-layer perceptrons and backpropagation became popular. Why has so little empirical research been published on this system? One reason may be the relative complexity of the network (as mentioned in [9, p.199], [10, p.187]) and, in light of this, we present a review of the neocognitron's operation (Section II) followed by a concise definition of the system (Section III).

In Section IV, we examine how the neocognitron calculates the similarity between an input pattern and the pattern classes it has been trained to identify. Section IV highlights the importance of S-cell *selectivity* in obtaining good performance from the network.

Hildebrandt's method for adjusting selectivities is briefly reviewed in Section V before two new techniques are presented and evaluated using real-world digit data. The most effective of these methods is used as a basis for further improvements to the neocognitron — described in Section VI — in which the distinguishing features of different classes of digits are exploited to achieve more accurate classification. Effectively, in Section VI, we attempt to fine tune the neocognitron in order to answer the question posed in the title of this paper.

The paper concludes with a review of the empirical results obtained and the implications they have for the neocognitron, and its variants, as practical digit recognition systems.

II. Overview of the neocognitron

The neocognitron classifies input through a succession of functionally equivalent stages. Each stage extracts "appropriate" features from the output of the preceding stage and forms a compressed representation of those extracted features. This representation preserves the spatial location of the extracted features and becomes the input to the next stage.

Classification is achieved by steadily extracting and compressing feature representations until the input is transformed into a vector whose elements are measures of the similarity between the input pattern and the input classes that the neocognitron has been trained to recognize. In a *winner-take-all* fashion, the final layer unit with the highest output determines the class assigned to an input pattern.

Feature extraction is performed by arrays of *S-cells* (called *S-planes*) trained to respond to certain features *deemed by the supervisor* to characterize input patterns. Each S-cell receives input from a rectangular region of cells in each *C-cell* plane (*C-plane*) of the preceding stage. The set of weights between each S-cell and its input regions is the same for every S-cell within a given plane. This *weight sharing* ensures features are detected wherever they lie in the input cell plane.

The first author is currently with the Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK. The second and third authors are with The Intelligent Machines Laboratory, Department of Electrical and Computer Engineering, University of Queensland, Queensland 4072, Australia.

C-plane activity is a compressed approximation of the activity in the preceding S-planes. This compression of representation also provides a degree of translational invariance to the neocognitron [11].

III. Formal definition of the neocognitron

Despite the neocognitron's complexity, its formal description can still be organized into a logical progression of concepts. We address three major issues:

1. the *organization* of the cells in the neocognitron,
2. the *interconnections* between them,
3. the *functional description* of those cells.

We have adopted Fukushima's system of terminology (used in almost all neocognitron literature) but, for the sake of both completeness and clarity, new notation to describe training patterns is introduced in Section III-C (completeness) and abbreviated vector notation is defined in Section IV (clarity).

Since its inception, certain aspects of the neocognitron have been altered by Fukushima. We shall adhere to Fukushima's most recent *complete* description of the neocognitron [4] in the following discussion.

A. The "morphology" of the neocognitron

There are *three* types of processing element in the neocognitron: S-cells, C-cells and V-cells. Any individual S or C-cell is identified by four pieces of information:

- the *type* of cell (S or C)
- the *layer*, ℓ, that the cell belongs to
- the *cell-plane*, k, that it is part of
- the *location*, $\boldsymbol{n}$, of the cell within that cell-plane.

The outputs of S and C-cells are given the general notation $u_{S\ell}(\boldsymbol{n}, k)$ and $u_{C\ell}(\boldsymbol{n}, k)$.

V-cells provide information to S-cells about the amount of activity present within each S-cell's input regions. Only one V-cell plane (*V-plane*) per layer is necessary to store the values of weighted root-mean-square input region activity (see Figure 4), hence a particular V-cell is specified by

- the *type* of cell (*i.e.*, V)
- the *layer*, ℓ, that the cell belongs to
- the *location*, $\boldsymbol{n}$, of the cell within the V-plane.

The outputs of V-cells are given the general notation $u_{V\ell}(\boldsymbol{n})$.

The possible values of parameters ℓ, k and $\boldsymbol{n}$ are determined by the architecture of a specific network. In layer ℓ, S-planes are numbered 1 to $K_{S\ell}$; C-planes range from 1 to $K_{C\ell}$ (see Figure 1). A cell's location within an S, C or V-plane is specified by a 2-dimensional position vector, $\boldsymbol{n}$. This vector describes the position of a cell's receptive field centre in relation to the input cell plane, UC0, as depicted in Figure 2.

B. The synaptic organization of the neocognitron

The neocognitron is structured like a large sandwich of alternating S and C-plane layers. Only adjacent layers of cell-planes are directly connected and an S-cell is connected with cells in all immediately preceding C-planes. Individual links from C-cells to an S-cell[1] are identified by four pieces of information

- the *layer*, ℓ, of the S-plane that they connect to
- the *serial number*, k, of that S-plane
- the *serial number*, κ, of the C-plane from which the link originates
- the *location*, $\boldsymbol{\nu}$, within the connection region, A_ℓ, of the C-cell from which the link originates.

Weights from C to S-cells are given the general notation $a_\ell(\boldsymbol{\nu}, \kappa, k)$. Since all cells in S-plane k share the same weights, the connection $a_\ell(\boldsymbol{\nu}, \kappa, k)$ does *not* contain the argument $\boldsymbol{n}$ to define a specific S-cell as the destination of that link. The location of a link's source cell is identified by the position vector $\boldsymbol{\nu}$ (see Figure 3). S-cell weights have non-negative values, as do all other weights and parameters in the neocognitron.

S-cells also receive input from subsidiary V-cells (Figure 4). The degree that V-cells affect cells in a given S-plane, k, is determined by the positive value of the *inhibitory coefficient* $b_\ell(k)$.

Figure 4 shows that V-cells are linked with preceding C-planes in the same way that S-cells are. Unlike C-plane to S-cell weights though, connections between C-planes and any V-cell are fixed and specified as a function of a C-cell's position, $\boldsymbol{\nu}$, within the connection region A_ℓ. Each of the sets of weights (or *masks*) between a V-cell in layer ℓ and the previous C-planes is denoted $c_\ell(\boldsymbol{\nu})$.

Connections from S-cells to a C-cell are also fixed and expressed as a function of S-cell position within a C-cell

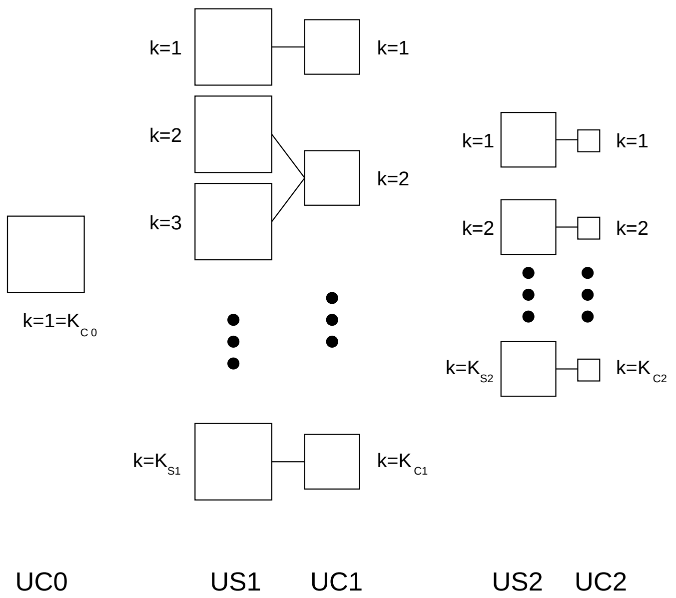

Fig. 1. Cell-planes are identified by a serial number k. When necessary to refer to two cell-planes in different layers, Fukushima uses κ to denote the second serial number. Layers of S-planes are labelled US1, US2, *etc.*, while layers of C-planes are labelled UC0, UC1 and so on.

[1] The prepositions *to* and *from* specify the direction of information flow along a connection between cells. The output of a *source* cell flows *to* a *destination* cell; a destination cell receives input *from* a source cell.

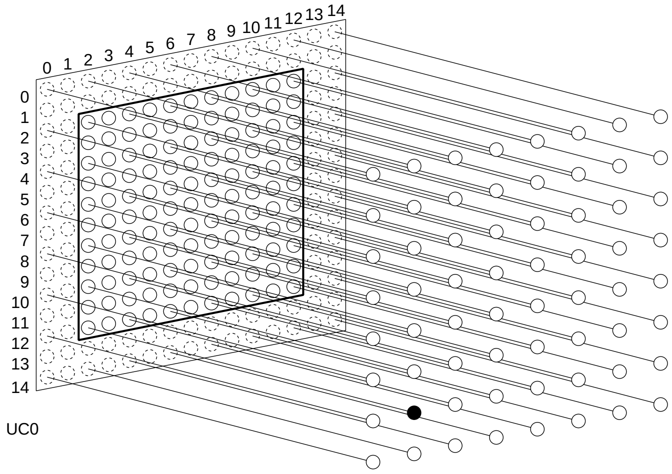

Fig. 2. A cell's position within its plane is given by the coordinates of its projection onto the input cell plane. This projection is the centre of the cell's receptive field and may lie outside the physical input cell-plane, UC0. Here the UC0 plane is 11×11 cells in size and the location of the black cell is at $n = (2, 12)$.

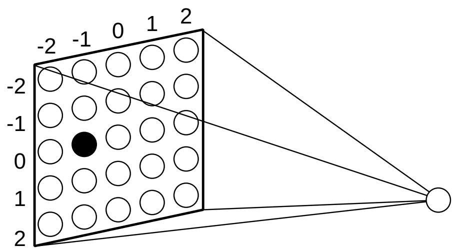

Fig. 3. The weight sharing mechanism used in the neocognitron makes it convenient to specify a particular connection in terms of ν, the position of a source cell within a destination cell's input region. In this 5×5 cell input region the black cell is at $\nu = (-1, 0)$.

input region, D_ℓ. A set of S-cell to C-cell weights is given the notation $d_\ell(\boldsymbol{\nu})$. Since a particular C-plane may receive input from one or more S-planes, S to C-plane connectivity is described by the factor $j(\kappa, k)$. If S-plane κ and C-plane k are connected, then $j(\kappa, k) > 0$, otherwise[2], $j(\kappa, k) = 0$.

The notation defined so far does not provide a way to express the spatial relationships and interconnectivity between cells. Rather than formalize this issue with more definitions, Fukushima presents this information diagrammatically, as shown in Figure 5. For simplicity, this diagram presents connection information as though there were but a single S and C-plane in each layer of the network — links between additional cell-planes obey the same scheme of interconnection. Figure 5 shows how the ratios of S and C-cells in layers two, three and four cause activity to converge to a single cell. Overlapping connections ensure this compression is achieved without undersampling. Note that the finite width of cell planes can cause cells at the edge of a plane to receive only partial connection to the previous layer.

[2]Fukushima implies that the actual value of $j(\kappa, k)$ for connected cell-planes is 1 [4].

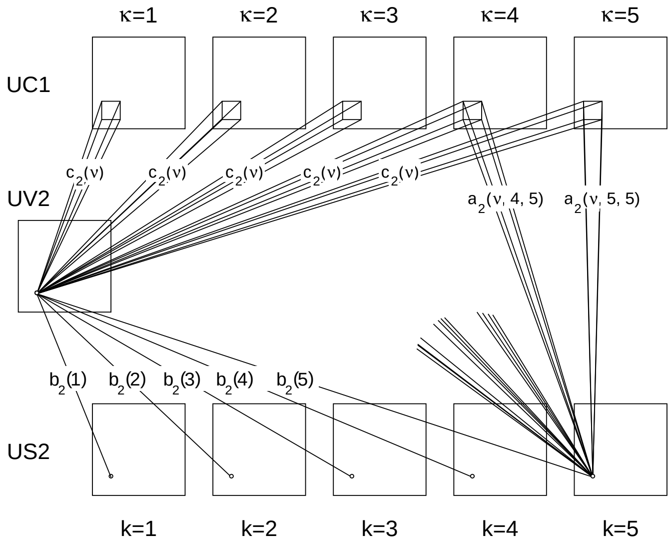

Fig. 4. V-cells have fixed weight links to input regions in all immediately preceding C-planes. Each V-cell's output is approximately equal to the magnitude of activity within its input regions. In layer ℓ, every cell in S-plane k receives inputs from a V-cell that has the same input regions. The strength of this input is weighted by the inhibitory coefficient $b_\ell(k)$.

C. The "cytology" of the neocognitron

Now the naming conventions used by Fukushima have been presented, we can define the equations governing S, C and V-cell function and the rules specifying the evolution of weights in the neocognitron. This section is deliberately terse; explanation of the following equations is relegated to Section IV so that *definition* and *interpretation* of cell function can remain distinct.

The output of an S-cell in the k^{th} S-plane of the ℓ^{th} layer of the neocognitron is given by

$$u_{S\ell}(\boldsymbol{n}, k) \stackrel{\text{def}}{=} r_\ell \times$$

$$\varphi\left[\frac{1 + \displaystyle\sum_{\kappa=1}^{K_{C\ell-1}} \sum_{\boldsymbol{\nu} \in A_\ell} a_\ell(\boldsymbol{\nu}, \kappa, k) \cdot u_{C\ell-1}(\boldsymbol{n} + \boldsymbol{\nu}, \kappa)}{1 + \dfrac{r_\ell}{r_\ell + 1} \cdot b_\ell(k) \cdot u_{V\ell}(\boldsymbol{n})} - 1\right]. \quad (1)$$

The function $\varphi(\cdot)$ is a *threshold-linear* transfer function, defined by

$$\varphi(x) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \leq x. \end{cases} \quad (2)$$

The *selectivity parameter*, r_ℓ, determines how closely the cell's input must correspond to the inputs it has been trained with in order to elicit a response.

The double summation in the numerator of Equation (1) is a *weighted sum* of the outputs of C-cells in the preceding layer. C-cell output is expressed as

$$u_{C\ell}(\boldsymbol{n}, k) \stackrel{\text{def}}{=}$$

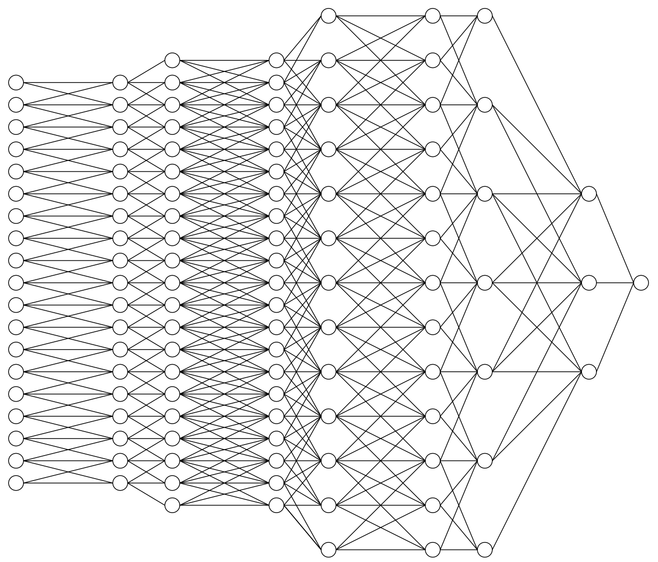

Fig. 5. This two dimensional 'slice' through the neocognitron shows the interconnections between one S and C-plane from each layer of the network. This information is based on the network described in [3].

$$\psi\left[\sum_{\kappa=1}^{K_{S\ell}} j(\kappa,k)\cdot\sum_{\boldsymbol{\nu}\in D_\ell} d_\ell(\boldsymbol{\nu})\cdot u_{S\ell}(\boldsymbol{n}+\boldsymbol{\nu},\kappa)\right], \quad (3)$$

where

$$\psi(x) \stackrel{\text{def}}{=} \frac{\varphi(x)}{1+\varphi(x)}. \quad (4)$$

The transfer function, $\psi(\cdot)$, limits C-cell output to the range $[0,1)$.

V-cells have an inhibitory effect and *normalize* an S-cell's response with respect to its input region activity. A V-cell's output is equal to the weighted root-mean-square value of the C-cell activity within its input regions:

$$u_{V\ell}(\boldsymbol{n}) \stackrel{\text{def}}{=} \sqrt{\sum_{\kappa=1}^{K_{C\ell-1}}\sum_{\boldsymbol{\nu}\in A_\ell} c_\ell(\boldsymbol{\nu})\cdot u^2_{C\ell-1}(\boldsymbol{n}+\boldsymbol{\nu},\kappa)}. \quad (5)$$

There are four different kinds of weights used in the neocognitron: $a_\ell(\boldsymbol{\nu},\kappa,k)$, $b_\ell(k)$, $c_\ell(\boldsymbol{\nu})$ and $d_\ell(\boldsymbol{\nu})$. The first two of these are determined during training, the last two are specified as

$$\begin{aligned} c_\ell(\boldsymbol{\nu}) &= {\gamma_\ell}^{|\boldsymbol{\nu}|} & (6)\\ d_\ell(\boldsymbol{\nu}) &= \overline{\delta}_\ell\cdot{\delta_\ell}^{|\boldsymbol{\nu}|}, & (7)\end{aligned}$$

where $0<\gamma_\ell,\delta_\ell\leq 1$ and $0<\overline{\delta}_\ell$.

Algorithm 1 uses pseudo-code to describe the training of the neocognitron; this allows us to present modifications to Fukushima's learning algorithm as straightforwardly as possible. To present Fukushima's *supervised* training algorithm[3] in this way, it is necessary to define further notation for the training exemplars.

Supervised training of the neocognitron requires that each S-plane be exposed to one or more training patterns. We define the set of training patterns for S-plane k in layer ℓ as

$$t_{k\ell} \stackrel{\text{def}}{=} \{t_{k\ell 1},\ldots,t_{k\ell m},\ldots,t_{k\ell|t_{k\ell}|}\}, \quad (8)$$

where $|t_{k\ell}|$ is the number of elements in $t_{k\ell}$. The *seed-cell* $\boldsymbol{n}_{k\ell m}$ is associated with the m^{th} training pattern. Both training patterns and their corresponding seed-cell locations are specified by the supervisor.

This concludes the formal specification of the neocognitron but there is still much more to tell. Several people (including Fukushima) have analysed how the neocognitron extracts features [13, 14, 15]. The following section gives an interpretation of Equations 1 to 7 and establishes concepts which will be useful to us later.

IV. The calculation of similarity

The neocognitron is based on the notion of similarity. On a small scale, individual S-cells calculate the similarity between the patterns of activity in their input regions and the features they have been trained to respond to. On a large scale, the outputs of the neocognitron represent the similarities between the input pattern and each of the different input classes that the network has been trained to recognize. Only the similarity calculated by S-cells has a direct mathematical representation (Equation (1)). This representation has a geometrical interpretation, with weights and inputs represented in Euclidean space. Before rendering this view of S-cell behaviour, we must define some additional vector notation.

If the connection region of an ℓ^{th} layer S-cell is defined as the set of all input cell position vectors in a preceding C-plane, *i.e.*,

$$A_\ell \stackrel{\text{def}}{=} \{\boldsymbol{\nu}_1,\boldsymbol{\nu}_2,\ldots,\boldsymbol{\nu}_{|A_\ell|}\}, \quad (9)$$

then the weight, mask and input vector of any cell in the k^{th} S-plane of layer ℓ can be written as

$$\boldsymbol{a}_\ell \stackrel{\text{def}}{=} \begin{bmatrix} a_\ell(\boldsymbol{\nu}_1,1,k)\\ \vdots\\ a_\ell(\boldsymbol{\nu}_{|A_\ell|},1,k)\\ a_\ell(\boldsymbol{\nu}_1,2,k)\\ \vdots\\ a_\ell(\boldsymbol{\nu}_{|A_\ell|},2,k)\\ \vdots\\ a_\ell(\boldsymbol{\nu}_{|A_\ell|},K_{C\ell-1},k)\end{bmatrix}, \boldsymbol{c}_\ell \stackrel{\text{def}}{=} \begin{bmatrix} c_\ell(1)\\ \vdots\\ c_\ell(|A_\ell|)\\ c_\ell(1)\\ \vdots\\ c_\ell(|A_\ell|)\\ \vdots\\ c_\ell(|A_\ell|)\end{bmatrix}$$

[3] Supervised training produces better recognition performance than self-organization [2] and since we are interested in maximizing the neocognitron's performance we shall only discuss this so-called *training-with-a-teacher* method of learning. The reader is referred to [12] for information about unsupervised learning in the neocognitron.

Algorithm 1: (Fukushima's Supervised Training Algorithm)

Assuming that all connections described by $a_\ell(\boldsymbol{\nu}, \kappa, k)$ and $b_\ell(k)$ are initially equal to zero, the process of training a neocognitron with layers 1 to L can be written as follows.

```
procedure train_neocognitron() {
   for ℓ = 1 to L {                              # For each layer of the neocognitron
      train_layer(ℓ) ;                           # learn S-cell weights.
   }
}

procedure train_layer(ℓ) {                       # Update S-plane weights
   for k = 1 to K_Sℓ {                           # for each S-plane in layer ℓ
      for m = 1 to |t_kℓ| {                      # and for each training pattern of the k^th S-plane.
         UC0 = t_kℓm ;                           # Load the m^th pattern into the input plane,
         for layer = 0 to ℓ−1 {                  # propagate activity from input to layer ℓ−1,
            activate(layer) ;
         }
         train_S_cell(n_kℓm, k, ℓ) ;             # then update the weights of the seed cell.
      }
   }
}

procedure train_S_cell(n_kℓm, k, ℓ) {            # Update the a_ℓ(ν,κ,k) and b_ℓ(k) weights
   for κ = 1 to K_Cℓ−1 {                         # for each C-plane in the preceding layer
      for all ν ∈ A_ℓ {                          # and for all input region cell positions.
         a_ℓ(ν,κ,k) = a_ℓ(ν,κ,k) + q_ℓ · c_ℓ(ν) · u_Cℓ−1(n_kℓm + ν, κ) ;
      }
   }
   b_ℓ(k) = b_ℓ(k) + q_ℓ · u_Vℓ(n_kℓm) ;
}
```

The procedure *activate*(ℓ) represents the propagation of activity from the S-cell inputs to the C-cell outputs of layer ℓ, according to the transformations defined by Equations (1) to (5). The parameter q_ℓ is a positive number known as the *learning rate* of layer ℓ. Note that this algorithm is completely deterministic.

$$\text{and} \quad \boldsymbol{u}_{C\ell-1} \stackrel{\text{def}}{=} \begin{bmatrix} u_{C\ell-1}(\boldsymbol{n}+\boldsymbol{\nu}_1, 1) \\ \vdots \\ u_{C\ell-1}(\boldsymbol{n}+\boldsymbol{\nu}_{|A_\ell|}, 1) \\ u_{C\ell-1}(\boldsymbol{n}+\boldsymbol{\nu}_1, 2) \\ \vdots \\ u_{C\ell-1}(\boldsymbol{n}+\boldsymbol{\nu}_{|A_\ell|}, 2) \\ \vdots \\ u_{C\ell-1}(\boldsymbol{n}+\boldsymbol{\nu}_{|A_\ell|}, K_{C\ell-1}) \end{bmatrix},$$

respectively. The vector of seed-cell inputs corresponding to the m^{th} training pattern of the k^{th} S-plane in layer ℓ is denoted by

$$\boldsymbol{u}^m_{C\ell-1} \stackrel{\text{def}}{=} \begin{bmatrix} u_{C\ell-1}(\boldsymbol{n}_{klm}+\boldsymbol{\nu}_1, 1) \\ u_{C\ell-1}(\boldsymbol{n}_{klm}+\boldsymbol{\nu}_2, 1) \\ \vdots \\ u_{C\ell-1}(\boldsymbol{n}_{klm}+\boldsymbol{\nu}_{|A_\ell|}, 1) \\ u_{C\ell-1}(\boldsymbol{n}_{klm}+\boldsymbol{\nu}_1, 2) \\ u_{C\ell-1}(\boldsymbol{n}_{klm}+\boldsymbol{\nu}_2, 2) \\ \vdots \\ u_{C\ell-1}(\boldsymbol{n}_{klm}+\boldsymbol{\nu}_{|A_\ell|}, 2) \\ \vdots \\ u_{C\ell-1}(\boldsymbol{n}_{klm}+\boldsymbol{\nu}_{|A_\ell|}, K_{C\ell-1}) \end{bmatrix}.$$

The inputs and weights of an S-cell are vectors in *pattern space* and a convenient measure of their similarity is the *cosine* of the angle between them. The degree of similarity between these two vectors is defined as

$$s(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) \stackrel{\text{def}}{=} \frac{\boldsymbol{a}_\ell{}^{\mathrm{T}} \boldsymbol{u}_{C\ell-1}}{|\boldsymbol{a}_\ell|\,|\boldsymbol{u}_{C\ell-1}|}, \tag{10}$$

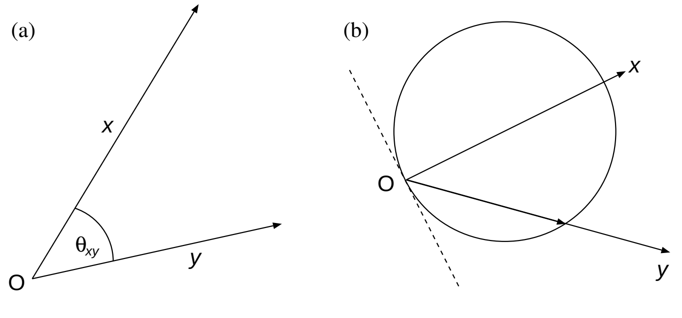

Fig. 6. (a) A two dimensional example of vectors x, y and the angle between them, θ_{xy}.
(b) The locus of $\cos\theta_{xy} \cdot \hat{y}$ as y takes all possible directions in the plane.

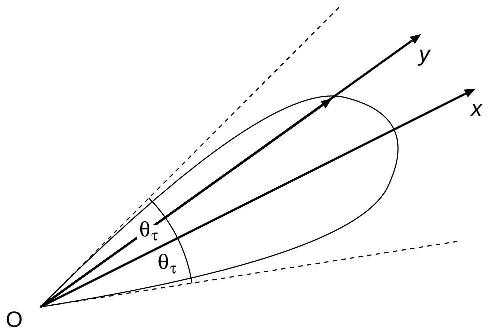

Fig. 7. The locus of $s''(x, y) \cdot \hat{y}$ for all directions of y. Since $\theta_\tau = \cos^{-1}\tau$, the higher the threshold, the narrower the locus about x becomes. Thus, high values of τ restrict the range of input vectors for which $s''(x, y)$ is greater than zero (*i.e.*, the *acceptance region*, $\mathcal{A}$). In two dimensions, $\mathcal{A}$ is a triangular region (between the dotted lines above), in three dimensions it is conical, and in higher dimensions the shape of the acceptance region is described as a *hypercone*.

that is, the cosine of $\theta_{\boldsymbol{a}_\ell \boldsymbol{u}_{C\ell-1}}$ (see Figure 6(a)). Figure 6(b) shows a 2-dimensional example of the *direction cosine* surface described when the similarity measure of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is projected in the direction of $\boldsymbol{y}$.

Since there may be no activity within a cell's input regions (*i.e.*, every element of $\boldsymbol{u}_{C\ell-1}$ could be 0), Equation (10) cannot be used directly as a similarity measure. The S-cell function described in Equation (1) is based upon a variation of Equation (11)

$$s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) \stackrel{\text{def}}{=} \frac{1 + {\boldsymbol{a}_\ell}^{\mathrm{T}} \boldsymbol{u}_{C\ell-1}}{1 + |\boldsymbol{a}_\ell|\,|\boldsymbol{u}_{C\ell-1}|}. \tag{11}$$

This modified similarity measure avoids the problem of zero length input vectors and is approximately equal to the similarity measure of Equation (10) when $|\boldsymbol{a}_\ell|\,|\boldsymbol{u}_{C\ell-1}| \gg 1$ (a condition which can be assured by using a large learning rate, *e.g.*, $q_\ell \approx 10^5$ as suggested in [4]). The relation between Equations (1) and (11) is not immediately apparent; subsequent equations should clarify the situation.

The purpose of an S-cell is to respond to an input *sufficiently* similar to the patterns it has been trained with. Fukushima has incorporated Equation (11) into the mathematical description of the S-cell so the degree of input and weight vector similarity necessary for non-zero S-cell response can be adjusted. By introducing a *threshold* parameter, $\tau \in [-1, 1]$, and a threshold-linear transfer function (Equation (2)), a further measure of similarity can be defined by

$$s''(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) \stackrel{\text{def}}{=} \varphi\left[\frac{s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) - \tau}{1 - \tau}\right]. \tag{12}$$

If $s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) \le \tau$, the non-negative function $\varphi(\cdot)$ ensures $s''(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) = 0$. If, however, the weight and input vectors are similar enough that $s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) > \tau$ then $0 < s''(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) \le 1$. Since $s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) \approx \cos\theta_{\boldsymbol{a}_\ell \boldsymbol{u}_{C\ell-1}}$ (for $|\boldsymbol{a}_\ell|\,|\boldsymbol{u}_{C\ell-1}| \gg 1$), the parameter τ defines a *threshold angle*, $\theta_\tau = \cos^{-1}\tau$. Figure 7 is a geometrical interpretation of Equation (12) and shows how τ can control the range of input vectors that make $s''(\cdot)$ positive. The volume of pattern space in which $s''(\cdot) > 0$ is referred to as the *acceptance region*, $\mathcal{A}$.

Equation (12) and parameter τ do not appear in other literature about the neocognitron. Instead of τ, Fukushima's S-cell description uses a *selectivity parameter*, r, to regulate the acceptance region. The threshold parameter is related to this quantity by $\tau = r/(r+1)$, where $r \ge -0.5$, hence

$$\begin{aligned} s''(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) &= \varphi\left[\frac{s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) - \dfrac{r}{r+1}}{1 - \dfrac{r}{r+1}}\right] \\ &= \varphi[(r+1)s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) - r] \end{aligned} \tag{13}$$

and, using some algebraic manipulation and the fact that $\varphi(r \cdot z) = r \cdot \varphi(z)$ for positive values of r, it can be shown that this equation is approximately equal to Equation (1).

What Equation (13) and Figure 7 show is that S-cell output (and hence network classification performance) depends strongly on selectivity. A high value of selectivity will cause an S-cell's acceptance region to be quite acute — the cell will only respond to inputs highly similar to the patterns it was trained with. Low selectivity produces a broad acceptance region and an S-cell that will respond to a wide range of inputs — possibly inputs quite unlike those it was trained to recognize. In the next section we consider how to adjust selectivity to maximize recognition.

V. Adjusting selectivity

Hildebrandt [15] was the first to propose a comprehensive[4] scheme for adjusting S-cell selectivities in the neocognitron. However, for reasons not initially apparent, Hildebrandt's Optimal Closed-Form Training (**OCFT**) algorithm failed to produce a network that performed well.

Without going into great detail[5], the central idea of **OCFT** is to adjust S-cell selectivity so the acceptance regions of cells in different S-planes (within a given layer) are as large as possible without overlapping. Hildebrandt reasoned that this would allow S-cells to tolerate the largest

[4] Selectivity adjustment was discussed briefly in [12].
[5] **OCFT** and its mode of failure are discussed fully in [16, 17].

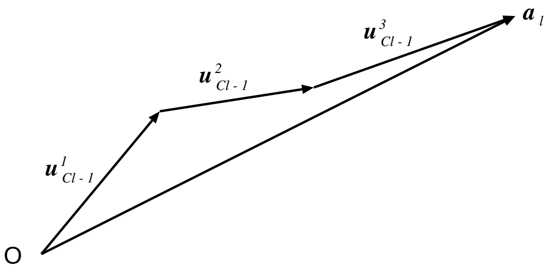

Fig. 8. Once training is complete, an S-cell's weight vector a_ℓ is proportional to the vector sum of the inputs presented during training $u^1_{C\ell-1} + u^2_{C\ell-1} + \cdots$

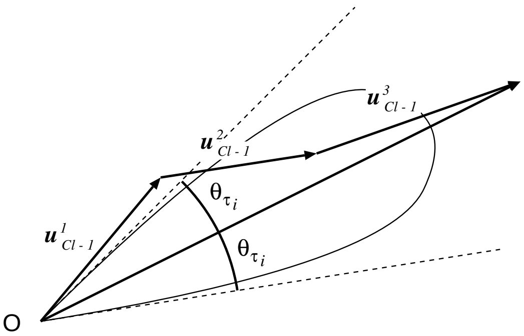

Fig. 9. If selectivity is chosen independently of training vectors, there is a risk of setting r_ℓ so high that one or more training vectors falls outside the acceptance region, as is the case for exemplar $u^1_{C\ell-1}$.

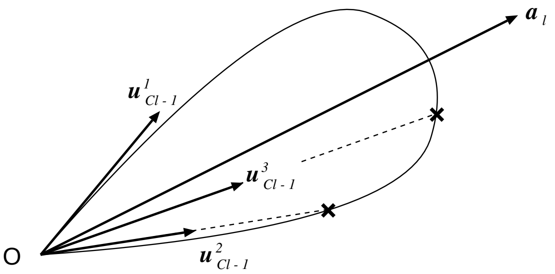

Fig. 10. For a given training pattern, the output of the S-cell is equal to the length of the line from the origin to the locus of S-cell output (the teardrop shaped lobe) in the direction of the appropriate input vector.

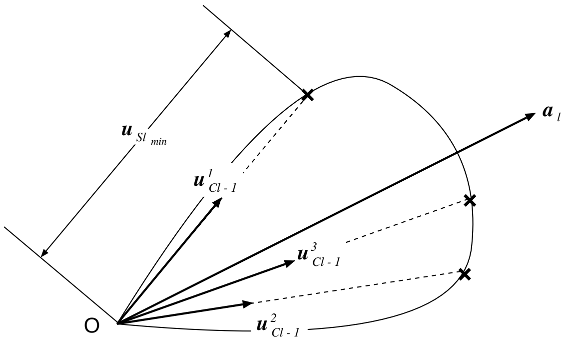

Fig. 11. **SOFT** adjusts the S-cell's selectivity to ensure the smallest output, in response to a training vector, is equal to $u_{S\ell_{\min}}$. Note the increased response of the S-cell to the second and third training patterns.

amount of distortion in input without compromising their ability to discriminate.

In practice [16], it was found that **OCFT** adjusted selectivities so that some S-cells would fail to recognize even the exemplars they were trained with, severely degrading overall performance. This *training feature rejection* occurred because **OCFT** determines selectivities on the basis of the *average* of S-plane training features. **OCFT** does not guarantee that selectivity will be adjusted so *individual* training vectors lie within the appropriate acceptance regions (as depicted in Figures 8 and 9).

A. Sub-Optimal Feature-based Training

One solution to the problem of training feature rejection is to adjust S-cell selectivity to *guarantee* a minimum S-cell response to *all* training patterns. This is the basis of the **S**ub-**O**ptimal **F**eature-based **T**raining (**SOFT**) algorithm described here.

The strategy of **SOFT** is to scale an S-cell's response to a training feature by adjusting the cell's selectivity from its initial value. This adjustment is carried out using the training pattern which elicits the weakest response from the S-cell. Using the general notation of Sections III and IV, the output of the seed cell in the k^{th} S-plane of layer ℓ, in response to the m^{th} training pattern, is $u_{S\ell}(\boldsymbol{n}_{k\ell m}, k)$. The initial selectivity of that cell is $r_{k\ell}$. We define the weakest response to a training pattern for this S-plane as

$$min_{\,u_{S\ell}(k)} \stackrel{\text{def}}{=} \min_m u_{S\ell}(\boldsymbol{n}_{k\ell m}, k). \tag{14}$$

SOFT adjusts the selectivity of the S-cell from $r_{k\ell}$ to $r'_{k\ell}$ so that the weakest S-cell response to a training pattern becomes $u_{S\ell_{\min}}$. The parameter $u_{S\ell_{\min}}$ is referred to as the *guaranteed minimum S-cell response* and satisfies $0 < u_{S\ell_{\min}} < 1$.

To explain how **SOFT** adjusts selectivity, we need to define S-cell *activation* as

$$x_{S\ell}(\boldsymbol{n}, k) \stackrel{\text{def}}{=} (r_{k\ell}+1)s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) - r_{k\ell}, \tag{15}$$

which is essentially the argument of $\varphi[\cdot]$ in Equation (13). In a similar fashion to Equation (14), the minimum S-cell activation in response to a training vector is defined as

$$min_{\,x_{S\ell}(k)} \stackrel{\text{def}}{=} \min_m x_{S\ell}(\boldsymbol{n}_{k\ell m}, k). \tag{16}$$

From Equation (11) we know $s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1})$ depends only on the angle formed between the vector of inputs to an S-cell, $\boldsymbol{u}_{C\ell-1}$, and that cell's weight vector, $\boldsymbol{a}_\ell$. Using Equation (15), we can calculate what the modified similarity measure of an S-cell must be, given that the activation and selectivity of the cell are $min_{\,x_{S\ell}(k)}$ and $r_{k\ell}$, respectively:

$$s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) = \frac{r_{k\ell} + min_{\,x_{S\ell}(k)}}{r_{k\ell}+1}. \tag{17}$$

The goal of **SOFT** is to adjust $r_{k\ell}$ to $r'_{k\ell}$ so that $min_{\,x_{S\ell}(k)} = u_{S\ell_{\min}}$. This means that

$$u_{S\ell_{\min}} = (r'_{k\ell}+1)s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) - r'_{k\ell}. \tag{18}$$

Rearranging this to isolate the required selectivity value gives

$$r'_{k\ell} \quad = \quad \frac{s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1}) - u_{S\ell_{\min}}}{1 - s'(\boldsymbol{a}_\ell, \boldsymbol{u}_{C\ell-1})}, \tag{19}$$

and substituting in Equation (17) gives an explicit expression for the value of $r'_{k\ell}$ in terms of the current selectivity and cell activation, and the desired output:

$$r'_{k\ell} \quad = \quad \frac{\left(\dfrac{r_{k\ell} + min_{\,x_{S\ell}(k)}}{r_{k\ell}+1}\right) - u_{S\ell_{\min}}}{1 - \left(\dfrac{r_{k\ell} + min_{\,x_{S\ell}(k)}}{r_{k\ell}+1}\right)}. \tag{20}$$

Now we can incorporate Equation (20) into a precise specification of the **SOFT** algorithm (see Algorithm 2).

B. Implementation issues

While **SOFT** alleviates the problem of training feature rejection, its implementation requires that three additional issues be addressed:

1. Instead of eliminating (or even reducing) the number of parameters that need to be chosen, **SOFT** *replaces* the set of selectivity parameters with a set of guaranteed minimum S-cell response parameters. We have no guidelines as to how to choose suitable values of $u_{S\ell_{\min}}$ (see Figure 12).
2. We cannot apply **SOFT** to S-cells that have only one training pattern (Figure 13). After training is complete, an S-cell with only one training pattern will have the weights $\boldsymbol{a}_\ell = q_\ell\, \boldsymbol{u}^1_{C\ell-1}$, and when we compute this cell's activation in response to its training vector we have $x_{S\ell}(\boldsymbol{n}_{k\ell 1}, k) = 1$. Equation (20) shows that this activation means $r'_{k\ell} \to \infty$ (for $0 \le u_{S\ell_{\min}} < 1$). If it were possible to implement such a value of $r'_{k\ell}$, the S-cell would respond only when the inputs to the cell were identical to the cell's weight vector, effectively removing any capacity for that S-cell to generalize.
3. The experimenter's choice of training features indirectly determines S-cell selectivity. A set of highly similar training vectors will produce an S-cell with a very narrow acceptance region. Certainly, all training vectors will elicit a non-zero response from that S-cell, but, unless the actual distribution of *typical* features is tightly clumped within the acceptance region, the cell will be far too selective (Figure 14).
   On the other hand, a training vector substantially dissimilar to other training vectors will cause an S-cell's acceptance region to balloon out to encompass it (Figure 15). An S-cell with such a low selectivity will be responsive to almost any feature and, therefore, probably not be of much use within the neocognitron. To use Hildebrandt's terms, finding the appropriate balance between generalization and discrimination is still a problem.

In the next section we determine whether these problems are a major handicap in applying **SOFT** to the neocognitron.

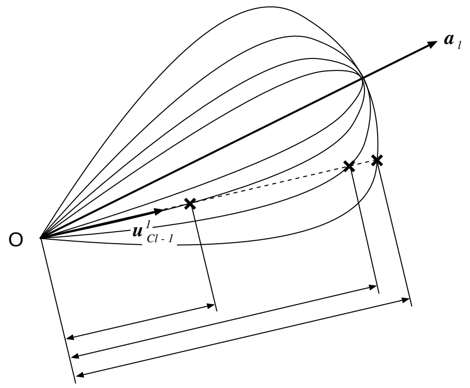

Fig. 12. By varying $r_{k\ell}$, we can produce a family of S-cell output loci, giving a range of feasible $u_{S\ell_{\min}}$ values. This figure shows how three different choices of $r_{k\ell}$ give three different non-zero S-cell outputs for the input vector $u^1_{C\ell-1}$. It is unclear which choice is best.

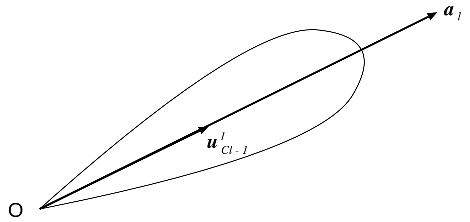

Fig. 13. If an S-cell has a single training vector, the cell's output in response to that vector will be 1, regardless of the value of selectivity.

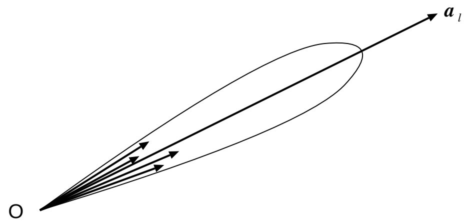

Fig. 14. Highly similar training vectors cause **SOFT** to produce a highly selective S-cell.

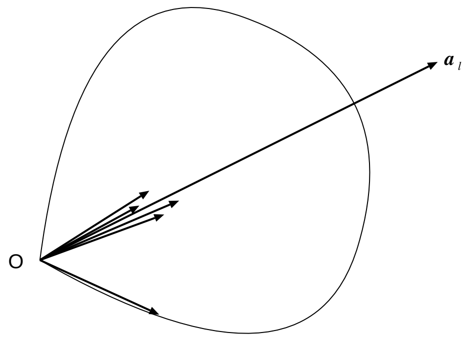

Fig. 15. One unusual training feature will radically alter the acceptance region produced by **SOFT**.

Algorithm 2: **(The Sub-Optimal Feature based Training Algorithm)**
As with Fukushima's supervised training algorithm, this training scheme commences with all connections $a_\ell(\boldsymbol{\nu}, \kappa, k)$ and $b_\ell(k)$ set to zero.

procedure *train_neocognitron()* {
  for $\ell = 1$ **to** L { — # For each layer of the neocognitron
    train_layer(ℓ) ; — # learn S-cell weights
    tune_layer(ℓ) ; — # then adjust S-cell selectivities.
  }
}

procedure *train_layer*(ℓ) { — # Update S-plane weights
  for $k = 1$ **to** $K_{S\ell}$ { — # for each S-plane in layer ℓ and
    for $m = 1$ **to** $|T_{k\ell}|$ { — # for each training pattern of
      — # the k^{th} S-plane.
      $UC0 = t_{k\ell m}$; — # Load the m^{th} pattern into the input plane,
      for *layer* $= 0$ **to** $\ell - 1$ { — # propagate activity from input...
        activate(layer) ; — # to layer $\ell - 1$,
      }
      train_S_cell($\boldsymbol{n}_{k\ell m}$, k, ℓ) ; — # then update the weights of the seed cell.
    }
  }
}

procedure *train_S_cell*($\boldsymbol{n}_{k\ell m}$, k, ℓ) { — # Update the $a_\ell(\boldsymbol{\nu}, \kappa, k)$ and $b_\ell(k)$ weights
  for $\kappa = 1$ **to** $K_{C\ell-1}$ { — # for each C-plane in the preceding layer
    for all $\boldsymbol{\nu} \in A_\ell$ { — # and for all input region cell positions.
      $a_\ell(\boldsymbol{\nu}, \kappa, k) = a_\ell(\boldsymbol{\nu}, \kappa, k) + q_\ell \cdot c_\ell(\boldsymbol{\nu}) \cdot u_{C\ell-1}(\boldsymbol{n}_{k\ell m} + \boldsymbol{\nu}, \kappa)$;
    }
  }
  $b_\ell(k) = b_\ell(k) + q_\ell \cdot u_{V\ell}(\boldsymbol{n}_{k\ell m})$;
}

procedure *tune_layer*(ℓ) { — # Tune the selectivities
  for $k = 1$ **to** $K_{S\ell}$ { — # for each S-plane in layer ℓ.
    $r_{k\ell} = 1$; — # Initialize selectivity of the k^{th} S-plane,
    $min_{x_{S\ell}(k)} = 1$; — # initialize minimum S-cell response, and
    for $m = 1$ **to** $|T_{k\ell}|$ { — # for each training pattern of
      — # the k^{th} S-plane
      $UC0 = t_{k\ell m}$; — # load the m^{th} pattern into the input plane
      for *layer* $= 0$ **to** ℓ { — # and propagate activity from input...
        activate(layer) ; — # to layer ℓ.
      }
      # If seed-cell's activation < current minimum then
      # update the current minimum S-cell response.
      if $x_{S\ell}(\boldsymbol{n}_{k\ell m}, k) < min_{x_{S\ell}(k)}$ **then**
      $min_{x_{S\ell}(k)} = x_{S\ell}(\boldsymbol{n}_{k\ell m}, \kappa)$;
    }
    # Adjust $r_{k\ell}$ so that the minimum S-cell response
    # to any training pattern will be $u_{S\ell_{\min}}$.
    $$r_{k\ell} = \left(\frac{r_{k\ell} + min_{x_{S\ell}(k)}}{r_{k\ell} + 1} - u_{S\ell_{\min}} \right) \Big/ \left(1 - \frac{r_{k\ell} + min_{x_{S\ell}(k)}}{r_{k\ell} + 1} \right) ;$$
  }
}

TABLE I

In this comparative table "KF-1988" refers to the network described in [3], "DL-1992" is the network described in [17].

Expt	Network	Training Alg.	Test Digits	% Correct	% Misclassified	% Rejected	% Reliability
1	KF-1988	Algorithm 1	Lovell's	52.50%	26.25%	21.25%	66.67%
2	KF-1988	**OCFT**	Lovell's	3.25%	10.00%	86.75%	24.53%
3	KF-1988	**SOFT**	Lovell's	74.00%	26.00%	0.00%	74.00%
4	KF-1988	**SOFT**	CEDAR	47.25%	52.00%	0.25%	47.61%
5	KF-1988	**SOFT**	CEDAR (thinned)	51.50%	48.50%	0.00%	51.50%
6	DL-1992	**SOFT**	CEDAR	55.75%	26.75%	17.50%	67.58%
7	DL-1992	**SOFT**	CEDAR (thinned)	58.75%	26.00%	15.25%	69.32%
8	DL-1992	**SHOP**	CEDAR (thinned)	75.75%	23.75%	0.50%	76.13%
9	**NCMLP**	**SHOP** + BP	CEDAR	84.73%	3.14%	12.13%	96.43%

C. Experiments with **SOFT**

As experiments with **OCFT** demonstrated [16], the effect of any changes to a complex system like the neocognitron must be assessed empirically. Table I shows the performance of the neocognitron for a variety of architectures, training algorithms and test data.

Initially, we did not have access to a database of test digits so a set of 400 digits was 'constructed' by the first author[6]. Under the assumption that test set performance is binomially distributed, this size of test set provides estimates within ±5% of the network's true recognition rate with 95% confidence.

Our first experiment (results of which were independently verified [?]) evaluated the performance of the neocognitron described by Fukushima[7] in 1988 [3]. (We refer to this version of the neocognitron as "KF-1988".) We then assessed this KF-1988's performance after **OCFT** and it became clear that Hildebrandt's scheme had some shortcomings (see rows 1 and 2 of Table I).

The next step was to apply **SOFT** to KF-1988 — this presented two difficulties. First, a number of S-planes (including all first layer S-planes) within Fukushima's network had single training patterns; we decided not to adjust the selectivities of these planes. Given that one of the final layer S-planes had a single training pattern, we also decided to set *all* final layer selectivities to 1.0, to ensure all S-planes in this layer had equal sized acceptance regions.

Second, we had to choose guaranteed minimum S-cell response parameters for layers 2 and 3 of the network. In the absence of any firm guidelines, we set $u_{S2_{min}} = 0.5$ and $u_{S3_{min}} = 0.75$.

As shown in row 3 of Table I, **SOFT** boosted the classification rate of Fukushima's network by over 20%. However, when this system was evaluated on real world data from the CEDAR database[8] [18] it was clear that the 'constructed' testing data was considerably easier to classify than actual handwritten digits (compare rows 3, 4 and 5 of Table I).

Handwritten character recognition systems often *thin* data before attempting classification. Row 5 of Table I shows thinning[9] gives a slight improvement in performance. To further enhance recognition, a new set of training patterns (based on features present in CEDAR database training digits) was created and a new network structure was implemented. (Full details of this new architecture — referred to as "DL-1992" — are given in [17, Appendix C].) The performance obtained is shown in rows 6 and 7 of Table I but, unfortunately, the degree of improvement was not enough to suggest that **SOFT** could produce a neocognitron capable of state-of-the-art digit recognition performance (*i.e.*, correct classification of around 90% with close to 100% reliability [20, 21]).

SOFT represents a significant improvement upon **OCFT** and salvages some of the ideas put forward by Hildebrandt. However, it is the authors' conviction that the geometric assumptions upon which **OCFT** and **SOFT** are based, should be put aside.

D. Selectivity Hunting to Optimize Performance

If **OCFT** and **SOFT** are not the best selectivity adjustment techniques, what issues should be considered in designing an effective method? The experiments described above, as well as results of some additional tests, led us to conclude that a suitable algorithm

1. needs to incorporate extensive amounts of real-world data into the adjustment process. Even though the neocognitron makes use of hand-crafted exemplars to determine S-cell weights, actual handwritten digits can still be used in the selectivity adjustment procedure.
2. must utilize a meaningful performance measure during training, *i.e.*, one directly related to the network's ability to generalize, not just its capacity to associate the correct output with each training pattern.
3. should not be based on unnecessarily restrictive assumptions about the distribution of patterns or features in input space; both **OCFT** and **SOFT** have demonstrated the dangers of making assertions in this regard.

[6] Lovell's test digits are public domain and are currently available via e-mail (contact `drl@eng.cam.ac.uk`).

[7] We gratefully acknowledge Professor Fukushima's kind assistance in providing us with his training data.

[8] Test data was drawn from the `TEST/BINDIGIS/BS` directory of the CD-ROM.

[9] Data was thinned using the Safe Point Thinning Algorithm [19]

4. must not introduce new parameters that have to be carefully chosen to obtain satisfactory behaviour from the network.

Taking these points into consideration, we propose a simple method of selectivity adjustment called **SHOP** — **S**electivity **H**unting to **O**ptimize **P**erformance. The concept behind this new algorithm is to take a number of identically structured neocognitrons, with different selectivity parameters, train them (using the same set of exemplars), then see which one is best at classifying a validation set of handwritten digits; the ideal selectivities are taken to be those of the network with the highest classification performance on the validation set.

Obviously, such a naïve method for determining selectivities must be subject to certain constraints and assumptions for it to be feasible. Instead of attempting to individually adjust the selectivity of each S-plane, **SHOP** maximizes the classification performance of the network with the constraint that all S-planes within a given layer have the same selectivity value. Also, because S-cell selectivity is a continuous variable, **SHOP** relies upon *sampling* the range of possible selectivities to obtain a finite number of performance estimates for a specific neocognitron.

SHOP is described in Algorithm 3, where R_ℓ is the set of all ℓ^{th} layer selectivities to be considered when testing the performance of the neocognitron.

E. *Problems with* SHOP

Although **SHOP** has the potential to make good use of real-world data and employs a more realistic measure of network performance than any of the algorithms discussed previously, the exhaustive search approach to finding a good set of selectivities has a number of drawbacks:

1. Execution time cannot be ignored in our assessment of **SHOP**'s feasibility. If t_T is the time taken to train the neocognitron, t_P is the time taken to propagate activity through the network and N_V is the number of examples in the validation set, then the time taken to execute **SHOP** is

$$t_{shop} = |R_1| \times |R_2| \times \cdots \times |R_L| \times (t_T + N_V\, t_P).$$

Fukushima [4] cites values of $t_T = 780s$, $t_P = 3.3s$ for a simulation of the neocognitron, written in FORTRAN and running on a SUN Sparcstation. If we specified 10 possible selectivity values for each layer of a four layer network and used a validation set with 400 patterns, **SHOP** would take about 8 months to come up with a good set of selectivities on Professor Fukushima's computer. In a commercial/industrial situation, a **SHOP** trained neocognitron would have to be implemented with dedicated hardware (*e.g.*, [22]).

2. The success of **SHOP** relies upon the stability of the neocognitron's performance with respect to changes in selectivity. If small changes in the selectivity of any layer cause the performance of the neocognitron to fluctuate wildly, then sampling the net's performance for a variety of selectivities will be of little use to us.

F. *Experiments with* SHOP

As indicated in the previous section, execution time was a major concern in our experiments with **SHOP**. The duration of each experiment was reduced by two orders of magnitude by restricting the first and fourth layer selectivity values tested. Since the relation between the input pattern and the outputs of the first layer S-planes was readily observable, a suitable r_1 value could be determined by trial and error. Hence, the set of first layer selectivities to be tested contained only one value: $R_1 = \{1.7\}$.

With respect to final layer selectivity it is important to remember that, while r_4 determines the *value* of the outputs of the network, it has no effect on which of the outputs is the largest. Since input patterns are classified on a winner-take-all basis, r_4 tends not to affect the classification performance of the network (providing it is not so high as to make US4 cells reject a large proportion of input patterns). The arbitrary restriction of $R_4 = \{1.0\}$ did not appear to cause any problems in the tests on **SHOP**. As well as saving time, restriction of first and fourth layer selectivities allowed the classification performance of the network to be readily visualized as a *performance surface* — a function of r_2 and r_3.

Figures 16 and 17 show that peak recognition performances obtained using **SHOP** on Fukushima's 1988 network are fairly poor. We reasoned that this may have been due to the stylized digit fragments used to train this system so, to test this hypothesis, we applied **SHOP** to the DL-1992 network (which used more "realistic" digit fragments during training). Figures 18 and 19 show this gave around 30% improvement in peak performance. Furthermore, the network's peak classification rate on the validation set of *unthinned* digits was slightly *in excess* of that achieved with the validation set of *thinned* digits — a finding contrary to Fukushima's comments that implied the the first layer of the neocognitron would have to be redesigned to cope with unthinned input patterns [4, Section V].

Performance surfaces are only useful if they can be used to *predict* good selectivity values. To see if this was the case, we compared the performance surface of Figure 19 to that given by a test set of 400 thinned CEDAR digits. Not only was the correlation between these two surfaces high ($\rho = 0.9961$), the maxima of both surfaces was achieved with the same (r_2, r_3) combination: $r_2 = 2.022$, $r_3 = 0.5990$. **SHOP** appears to provide an accurate prediction of good selectivity values. The peak performance obtained with the test digits is shown in row 8 of Table I.

Since **SHOP** allows us to systematically evaluate the neocognitron's behaviour, it can be used to investigate the effect of parameters other than selectivity. We know how to choose good values of selectivity (r_ℓ) and that the learning rate (q_ℓ) should be high, but we do not know how to choose the *mask parameters*, γ_ℓ, δ_ℓ and $\overline{\delta}_\ell$ (Equations (6) and (7)) or even (as Hildebrandt suggested [15, Section III-B]) whether such parameters are necessary. To explore this issue, we set all mask parameters of the DL-1988 network to 1.0 — effectively removing their influence on the network — and used **SHOP** with 400 thinned CEDAR digits

Algorithm 3: (The Selectivity Hunting to Optimize Performance Algorithm)

In the pseudo-code below, the procedure *train_neocognitron()* refers to Fukushima's original training algorithm (Algorithm 1).

The purpose of the *test_neocognitron()* procedure is to evaluate the performance of the network using a validation set of real-world data. It is up to the experimenter to decide exactly how the network's performance is to be measured. Correct classification rate and reliability are examples of statistics that could both be used to measure the network's performance

```
procedure SHOP() {
  for all r_1 ∈ R_1 {                       # Using every possible combination
    ⋮                                        # of selectivities,
      for all r_L ∈ R_L {                   # in all layers of the neocognitron,
        train_neocognitron() ;              # train the network...
        test_neocognitron() ;               # and then test it.
                                            # If the current network has
        if performance > best_so_far {      # the best performance (so far),
          best_so_far = performance ;       # update the best performance value
          R_best = (r_1, r_2, ..., r_L) ;   # and store the L-tuple of selectivities
        }                                   # used by the present network.
      }
    ⋮
  }
}
```

When the algorithm terminates, R_{best} contains the selectivities that elicited the highest performance from the network. These selectivities can then be used in practical implementations of the system.

to evaluate the network's performance surface. The surface obtained differed from the one shown in Figure 19 but the *peak classification performance* was not significantly different (a peak correct classification rate of 78.00%, with 78.20% reliability, occurred for $r_2 = 1.000$, $r_3 = 1.269$). Mask parameters *do not* seem to have a significant effect on the peak performance attainable.

VI. Further Improvements to the Neocognitron

We have investigated all parameters that affect cell function in the neocognitron. We know that by using a high learning rate ($q_\ell \approx 10^5$), appropriate mask parameter values ($\gamma_\ell, \delta_\ell \approx 0.7$–$1.0$; $\overline{\delta}_\ell \approx 1.0$–$4.0$) and selectivities determined by **SHOP**, a correct classification rate of around 76%, with 76% reliability, is feasible. If state-of-the-art performance is to be achieved with the neocognitron, it seems we must alter the network to attain it.

Researchers often decompose the problem of handwritten character recognition into two stages. *Feature extraction* obtains some numerical (or logical) measure of the characteristics of the input image, then the image undergoes *classification* and is labelled (or rejected) on the basis of the features extracted. Our attention has been focussed mainly on the neocognitron's feature extracting abilities and we have tended to ignore the 1% of cells in the network (*i.e.*, those in the final layer) that tell us what class has been assigned to an input pattern. For reasons explained in the previous section, **SHOP** does not alter the selectivities of these cells; examination of the network's outputs during operation gives an indication of the problems that occur as a result.

Figure 20 shows typical levels of output activity for fifty correctly classified input digits. Clearly, the neocognitron is not discriminating *effectively* between different classes of input. The outputs, as with all other cells in the network, simply indicate the degree to which certain features are present in the input image. Should some of the features detected by one cell also be detected by others, a number of cells can show high levels of activity at the same time. Consequently, the neocognitron is apt to confuse certain digits (such as '2's and '3's) because of the number of features they have in common.

We need final layer cells to exploit the idiosyncratic aspects of each kind of digit to obtain more robust classification. The multilayer perceptron (MLP) has been popular as a *distribution free* classifier [23]. Perhaps coupling the neocognitron and MLP would create a system that could perform distortion tolerant feature extraction *and* robust classification? We refer to this hybrid system as the **NCMLP** (**N**eo**C**ognitron plus **MLP**).

Put simply, the **NCMLP** takes a neocognitron that has been trained using **SHOP** and replaces the final layer C-cells with a two layer MLP (Figure 21). The MLP portion of the network is then trained to associate the outputs of the final layer S-cells with a single output that represents the class of the input image.

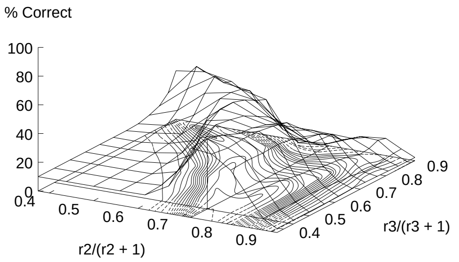

Fig. 16. The performance surface obtained with the KF-1988 network using a validation set of 400 unthinned CEDAR digits. Peak correct classification was 49.00% (54.60% reliability) for r_2 = 2.563, r_3 = 0.7805.

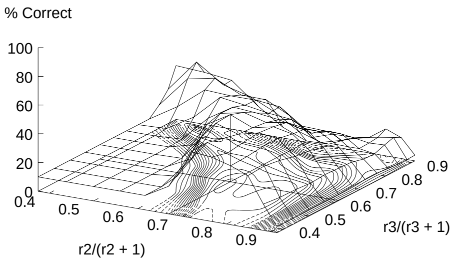

Fig. 17. The performance surface obtained with the KF-1988 1988 network using a validation set of 400 thinned CEDAR digits. Peak correct classification was 46.75% (49.34% reliability) for r_2 = 2.022, r_3 = 1.602.

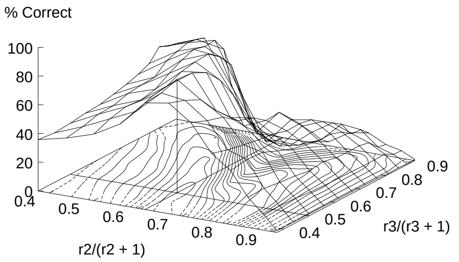

Fig. 18. The performance surface obtained with the DL-1992 network using a validation set of 400 unthinned CEDAR digits. A peak correct classification was 79.00% (79.40% reliability) for r_2 = 1.602, r_3 = 1.000.

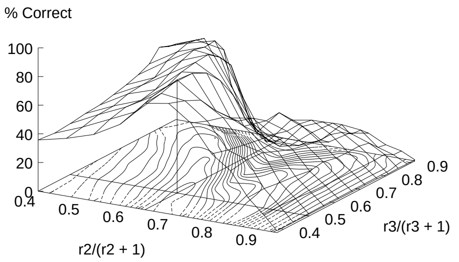

Fig. 19. The performance surface obtained with the DL-1992 network using a validation set of 400 thinned CEDAR digits. A peak correct classification was 78.75% (79.15% reliability) for r_2 = 2.022, r_3 = 0.5990.

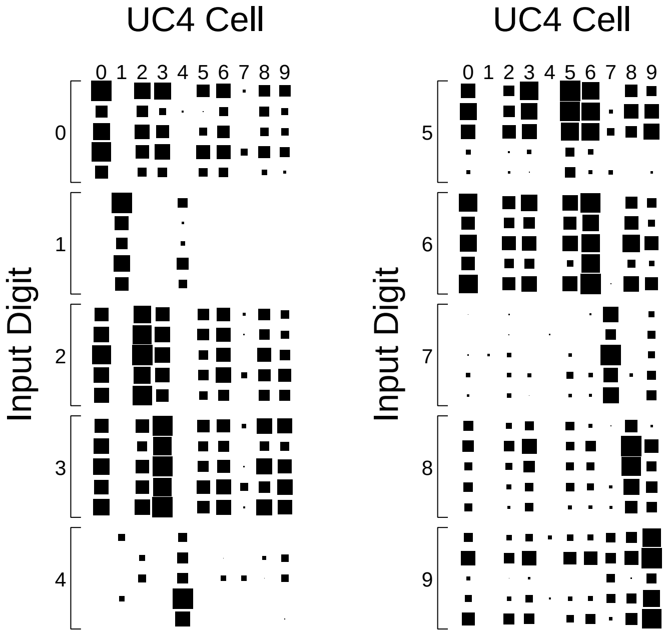

Fig. 20. Each row in the two large columns represents the outputs of the 10 final layer C-cells in the DL-1992 network after training with **SHOP** using unthinned CEDAR digits (see Figure 18). The class of input digit is indicated at the left of each column; UC4 cell numbers are shown at the top. The fifty digits that evoked these outputs were correctly classified. However, in many cases the margin between the largest and second largest output is small, making the classification vulnerable to error.

A. Problems with the NCMLP

On top of the variables attendant to the neocognitron, the **NCMLP** requires the experimenter to specify learning rate (η), momentum (α), number of hidden units, initial random weight variance, *etc.* One consolation is that the sheer volume of research involving MLP systems provides some empirical guidelines for selecting these parameters. But regarding the number of hidden units needed for the task at hand, the only practical answer seems to be to evaluate a variety of network architectures and choose the one that offers the best generalization performance (much in the same way that **SHOP** settles upon good selectivity values).

B. Experiments with the NCMLP

NCMLP training involves two phases: the first uses **SHOP** to determine effective selectivities for the neocognitron portion of the network; the second uses validation techniques in conjunction with backpropagation to determine the structure and weights for the MLP section.

We used **SHOP** and a validation set of 400 unthinned CEDAR digits to find selectivities for the DL-1992 network ($R_{\text{best}} = \{1.700, 1.602, 1.000, 1.000\}$). Determining an appropriate architecture for the MLP portion of the network was more involved.

To ensure inputs would not be classified on the basis of low or ambiguous outputs, we applied two thresholds, $t_{val} = 0.9$ and $t_{conf} = 0.1$, to the outputs of the **NCMLP**. t_{val} denotes the *threshold of validity* and t_{conf} the *threshold*

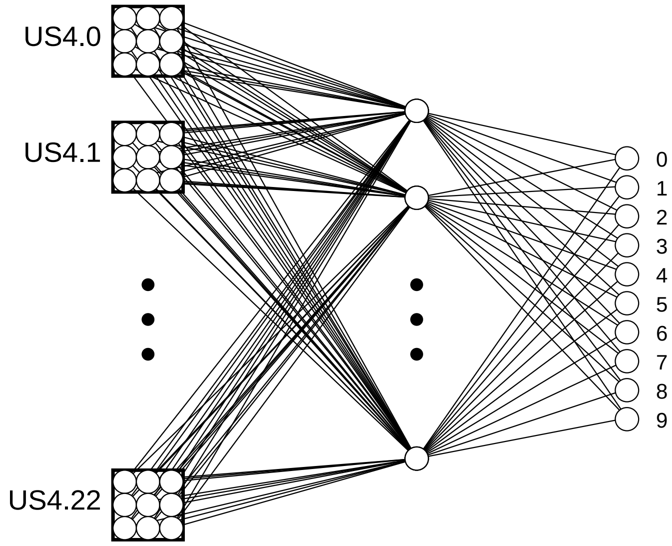

Fig. 21. The end portion of the **NCMLP** architecture. Final layer C-cells in a neocognitron are replaced by a two layer, fully connected MLP. The MLP receives input from the US4 cell planes and propagates activity through a layer of hidden units to the ten output units that indicate the **NCMLP**'s classification of the input image.

of confusion. For a given test digit, if no final layer unit's output was above t_{val} or more than one one unit's output was above t_{conf}, the digit was rejected. Otherwise, the digit was classed according to the unit with the highest output.

As well as thresholds to ensure rejection of illegible or ambiguous digits, the **NCMLP** was *trained* to reject images of ill-formed digits. Bromley and Denker [24] advocate the training of MLP type classifiers with "rubbish" subimages (*i.e.*, ambiguous digits, multiple or partial digits, and noise) to enhance rejection performance. The set of digits used to train the MLP portion of the network contained 440 examples of each digit[10] and 440 "rubbish" images.

Ten different MLP structures were evaluated; each had 207 inputs (19 US4 planes, each with 3×3 cells) and 10 outputs, and the number of hidden units varied between each network from 5, 10, 15,..., up to 50 units. Backpropagation training was applied to all networks using a learning rate of 1.0 and a momentum of 0.9. The performance of each architecture during training was monitored using a validation set of 4400 digits[11]

Under the assumption that validation set performance correlated highly with generalization ability[12], the most effective MLP was the one that achieved the highest classification rate on the validation set. This turned out to be a network with 45 hidden units, after 280 epochs of training. The test set[13] performance of this **NCMLP** is shown in row 9 of Table I.

[10] Taken from the **TRAIN/BINDIGIS/BR** section of the CEDAR CD-ROM.
[11] Again taken from the **TRAIN/BINDIGIS/BR** section of the CEDAR CD-ROM.
[12] An assumption justified by the correlation between validation and test set performances — $\rho > 0.98$ for the 10 different MLPs.
[13] An unbalanced test set of 2711 digits was taken from the **TEST/BINDIGIS/BS** section of the CEDAR CD-ROM.

There is an inherent difficulty in comparing the **NCMLP**'s performance to other digit recognition systems. As Suen *et al.* point out in their survey "...recognition systems cannot be compared simply by their reported performances since most systems are still tested on data bases with very different characteristics." [21, p.1176]. With this in mind, we note that the peak classification rate shown in row 9 of Table I is lower than that of the 19 systems cited by Suen *et al.* and it is ranked 15th in terms of reliability. While looking for ways to raise the **NCMLP**'s performance further still, it became clear that the neocognitron section of the network was not extracting all the features necessary for the MLP to distinguish between certain classes of digit. Additional training of the MLP would not remedy this situation; the feature extraction process would have to be altered either (a) by retaining the **NCMLP** paradigm and designing a more effective feature extraction network or (b) by rejecting the **NCMLP** model and trying to develop a system with a greater ability to learn to exploit distinguishing features of digits. The time and effort needed to redesign and retest the **NCMLP** renders the first option impractical. The second alternative has already been explored by Le Cun *et al.* [25, 26] who have applied gradient descent techniques and second derivative pruning methods to a hierarchical network very similar in structure to the neocognitron.

At this point we believe we have gone as far as possible in our investigation of the neocognitron and networks that are recognizably derived from it (yet distinct from existing systems). Our experiments have pointed us away from Fukushima's method of supervised training with digit fragments towards more effective, performance-driven learning schemes (*i.e.*, gradient descent style training). But, given the success of the **NCMLP** and the system described by Le Cun *et al.*, it seems likely that Fukushima's concept of a hierarchy of shared-weight feature extractors will be used in the classification of images by artificial neural networks for some time to come.

VII. Conclusions

In this paper we have critically reviewed the formulation and capabilities of Fukushima's neocognitron. To the best of the authors' knowledge, this is the first time an empirical assessment of the neocognitron with a substantial set of publicly available test data has been published.

The neocognitron's performance has been shown to be highly dependent upon the selectivity of the feature extracting S-cells and we have presented and evaluated two methods to set these parameters. The first technique, **SOFT**, avoids the shortcomings of Hildebrandt's **OCFT** algorithm by setting selectivity such that S-cells produce a guaranteed minimum response to their training patterns. Experiments with **SOFT** led to the proposal of **SHOP**, a selectivity adjustment algorithm that relies upon a validation set of real world digits to determine effective selectivities.

Analysis of final layer S-cell outputs after training with **SHOP** showed that classification implemented in the final layer of the network did not fully utilize the distinguishing features extracted in preceding stages. We addressed this problem with an extension to the neocognitron which uses an MLP as the final layer classifier. We showed how this **NCMLP** architecture could be trained using **SHOP** and backpropagation, as well as a validation set of digits to determine an appropriate number of hidden units for the MLP portion of the network.

Though the peak test performance of the **NCMLP** (84.73% correct with 96.43% reliability) was a significant improvement upon Fukushima's original network, this classification rate fell short of those obtained with state-of-the-art systems. Examination of S and C-cell outputs within the network suggested that the neocognitron fails to exploit the features that distinguish different classes of input. This result has implications for those conducting research with the supervised, unsupervised and selective attention versions of the neocognitron. Even if special consideration is given to the choice of the neocognitron's parameters it seems unlikely that systems of this type would be able to rival the performance of today's most advanced handwritten digit classifiers.

References

[1] K. Fukushima, "Neural network model for a mechanism of pattern recognition unaffected by shift in position — neocognitron," *Transactions IECE Japan*, vol. 62-A, no. 10, pp. 658–665, 1979.

[2] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: a neural network model for a mechanism of visual pattern recognition," *IEEE Trans. Systems, Man and Cybernetics*, vol. SMC-13, no. 5, 1983.

[3] K. Fukushima, "Neocognitron: A heirarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, pp. 119–130, 1988.

[4] K. Fukushima and N. Wake, "Handwritten alphanumeric character recognition by the neocognitron," *IEEE Trans. Neural Networks*, vol. 2, no. 5, pp. 355–365, May 1991.

[5] K. Fukushima and N. Wake, "An improved learning algorithm for the neocognitron," in *Artificial Neural Networks* (I. Aleksander and J. Taylor, eds.), vol. 2, pp. 497–504, Elsevier, 1992.

[6] K. Fukushima and N. Wake, "Improved neocognitron with bend detecting cells," in *IEEE/INNS International Joint Conference on Neural Networks*, vol. 4, (Baltimore), pp. 190–195, IEEE Press, 1992.

[7] K. Fukushima, "Improvement of the neocognitron and the selective attention model," in *Proc. World Congress on Neural Networks*, vol. 3, (New Jersey), pp. 634–647, International Neural Network Society, 1993.

[8] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiology*, vol. 160, pp. 105–154, 1962.

[9] R. Hecht-Nielsen, *Neurocomputing*. Addison-Wesley, 1990.

[10] A. I. Wasserman, *Neural Computing: Theory and Practice*. New York: Van Nostrand Reinhold, 1989.

[11] E. Barnard and D. Casasent, "Shift invariance and the neocognitron," *Neural Networks*, vol. 3, pp. 403–410, 1990.

[12] K. Fukushima and S. Miyake, "Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 16, no. 6, pp. 455–469, 1982.

[13] K. Johnson, C. Daniell, and J. Burman, "Feature extraction in the neocognitron," *Int. Joint Conf. on Neural Networks*, vol. 2, pp. 117–126, 1988.

[14] K. Fukushima, "Analysis of the process of visual pattern recognition by the neocognitron," *Neural Networks*, vol. 2, pp. 413–420, 1989.

[15] T. H. Hildebrandt, "Optimal training of thresholded linear correlation classifiers," *IEEE Trans. Neural Networks*, vol. 2, no. 6, pp. 577–588, November 1991.

[16] D. R. Lovell, A. C. Tsoi, and T. Downs, "Comments on "optimal training of thresholded linear correlation classifiers"," *IEEE Trans. Neural Networks*, vol. 4, no. 2, pp. 367–368, 1993.

[17] D. R. Lovell, *The Neocognitron as a System for Handwritten Character Recognition: Limitations and Improvements*. PhD thesis, University of Queensland, Brisbane, July 1994. Available via anonymous `ftp` from `archive.cis.ohio-state.edu` in the `pub/neuroprose/Thesis` directory.

[18] R. Fenrich and J. J. Hull, "Concerns in creation of image databases," in *Proc. International Workshop on Frontiers in Handwriting Recognition*, pp. 112–121, 1993.

[19] N. J. Naccache and R. Shinghal, "Spta: A proposed algorithm for thinning binary patterns," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 14, no. 3, pp. 409–418, May/June 1984.

[20] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of ocr research and development," *Proc. IEEE*, vol. 80, no. 7, pp. 1029–1058, 1992.

[21] C. Y. Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam, "Computer recognition of unconstrained handwritten numerals," *Proc. IEEE*, vol. 80, no. 7, pp. 1162–1180, 1992.

[22] A. M. Chiang and M. L. Chuang, "A ccd programmable image processor and its neural network applications," *IEEE J. Solid State Circuits*, vol. 26, no. 12, pp. 1894–1901, 1991.

[23] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, pp. 1–58, 1992.

[24] J. Bromley and J. S. Denker, "Improving rejection performance of handwritten digits by training with "rubbish"," *Neural Computation*, vol. 5, pp. 367–370, 1993.

[25] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, 1989.

[26] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems* (D. S. Touretzky, ed.), vol. 2, (San Mateo), pp. 396–404, Morgan Kaufmann, 1990.