Supervised Competitive Learning

Thomas H. Fuller, Jr. and Takayuki D. Kimura,

WUCS-93-45

Reprinted from
Journal of Intelligent Material Systems and Structures
(pages 232-246)

March 1994

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis, MO 63130-4899

This work was supported by the Kumon Machine Project.

Supervised Competitive Learning

Thomas H. Fuller, Jr.[1] and Takayuki D. Kimura
Department of Computer Science
Washington University in St. Louis

Journal of Intelligent Material Systems and Structures, October, 1993

ABSTRACT

Supervised Competitive Learning (SCL) assembles a set of learning modules into a supervised learning system to address the stability-plasticity dilemma. Each learning module acts as a similarity detector for a prototype, and includes prototype resetting (akin to that of ART) to respond to new prototypes. SCL has usually employed backpropagation networks as the learning modules. It has been tested with two feature abstractors: about 30 energy-based features, and a combination of energy-based and graphical features (about 60). About 75 subjects have been involved. In recent testing (15 college students), SCL recognized 99% (energy features only) of test digits, 91% (energy) and 96.6% (energy/graphical) of test letters, and 85% of test gestures (energy/graphical). SCL has also been tested with fuzzy sets as learning modules for recognizing handwritten digits and handwritten gestures, recognizing 97% of test digits, and 91% of test gestures.

[1] Assistant Professor, Department of Mathematics and Computer Science, Principia College

Table of Contents

1 Introduction
- 1.1 The Adaptable User Interface
- 1.2 Stability versus plasticity
- 1.3 Supervised Competitive Learning

2 Related work
- 2.1 The ART model of Carpenter and Grossberg
- 2.2 Other adaptive learning models
- 2.3 Off-line handwritten character recognition
- 2.4 On-line handwritten character recognition
- 2.5 Energy-based feature abstraction
- 2.6 Grid-based feature abstraction

3 The SCL model
- 3.1 Requirements and environment
- 3.2 SCL model and algorithm
- 3.3 Refinements to the model and algorithm
- 3.4 Contributions of SCL as compared with similar learning models
- 3.5 Energy-based feature abstraction in SCL
- 3.6 Grid-based feature abstraction in SCL
- 3.7 Additional considerations in feature computation
- 3.8 Network architecture

4 Experiments
- 4.1 Early recognition performance with backpropagation
- 4.2 Recognition performance with fuzzy logic
- 4.3 Training schedules, preprocessing and the complexity of individual prototype units
- 4.4 Current results
- 4.5 Recognition speed
- 4.6 Behavior of prototypes in gathering "kindred" exemplars
- 4.7 Confidence
- 4.8 SCL and gestures
- 4.9 Freeing the recognizer from "boxed" input

5 Conclusions

6 Future directions

7 Acknowledgement

8 References

List of Figures

1 SCL schematic
2 The numeral "5" in the feature grid
3 Digit recognizer with exemplars gathered by the prototypes
4 Subset of a letter recognizer with exemplars gathered by the prototypes

1 Introduction

1.1 The Adaptable User Interface

For decades, computers have inched toward more "user-friendly" access. Still most people do not find them friendly enough to apply to the many areas of potential benefit. This continues a historical pattern. Increasing processor power wrought little change in the "friendliness" until the paradigm of the user's relation to the processes shifted. Such a shift occurred with the advent of the mouse and the graphical user-interface introduced in the late 60's and widely realized in the late 80's. Significantly, neither the mouse nor the GUI individually was sufficient to cause the change -- *both* were necessary. This decade is bringing half of a new paradigm to the fore: the pen-based user interface. What must accompany this change of medium (as was the mouse) to radically transform the user interface? We feel that an Adaptable User Interface (AUI) should emerge as the "rest" of the paradigm. An AUI models a plastic system that gains knowledge of its user's behaviors and applies this knowledge to interpret user inputs.

A critical barrier to pen-based systems of any kind looms over the many promises — the recognition of handwritten input. It is a regular theme in the estimates of pen-computing in the industry and popular press. (See for example Schwartz 1992). Pundits prophesy that once this obstacle is removed, the pen just might revolutionize the use of computers. (Cf. Jerney 1993, Wagoner 1993, Baran 1992.)

It would be most desirable for an AUI to not only recognize input, but to improve its recognition of characters (letters, digits, gestures) from regular users. Supervised Competitive Learning offers one method for allowing the computer to regularly improve on its recognition task.

1.2 Stability *versus* plasticity

When an adaptive learning system such as a backpropagation artificial neural net (ANN) is used to encode input patterns from an evolving environment, it suffers the stability-plasticity dilemma formulated by Grossberg [Grossberg 1986] for the competitive learning paradigm: How can a learning system remain plastic in response to significant events and yet remain stable in response to irrelevant or routine events? How can it maintain previous knowledge while continuing to gain new?

Handwritten character recognition furnishes an example. Suppose a system has been successfully trained to recognize the handwritten character "7" by a person who writes "7" consistently with two strokes (European style). Now, the same system is to be trained by another person who writes "7" with one stroke. After adapting to the one-stroke "7," it may not be able to recognize the two-stroke "7" as well as it used to. A similar problem arises when a system learns alphabetic characters after "mastering" numeric characters.

1.3 Supervised Competitive Learning

SCL compounds a set of learning modules into a supervised learning system. Each prototype learning module (*prototype* for short) acts as a similarity detector for one class of exemplar

patterns considered sufficiently similar. SCL adopts a prototype resetting mechanism (akin to that of ART) to create new prototypes. Any learning model can be used for component modules; backpropagation nets and pattern classification models based on fuzzy logic are two natural candidates.

In the remaining sections we review related work, describe the architecture of SCL (including the abstraction of relevant features), report results of our experiments with the recognition of handwriting and gestures (using backpropagation and fuzzy logic as similarity detectors), and suggest future directions for research.

2. Related work

Many researchers have been active in various aspects of this field. This review groups relevant research by its relation to our own (adaptive learning, feature abstraction, etc.).

2.1 The ART model of Carpenter and Grossberg

Adaptive Resonance Theory (ART) was proposed by Carpenter and Grossberg [1988] as a possible solution to the stability-plasticity dilemma in the competitive learning paradigm. It consists of two sets of processing nodes: the attention subsystem and the orienting subsystem. The nodes in the attention subsystem compete with each other when activated by an input pattern. The winning node represents the learned category of the input pattern and also carries the prototype (attention) pattern associated with the category. The orienting subsystem compares the prototype with the input, and if the two are significantly different, it resets (disables) the winning node, which triggers a new round of competition; the reset response assumes that the input pattern does not truly belong to a category represented by the current winner. If all prototype patterns in the attention subsystem are sufficiently different from the input, the input pattern itself becomes the prototype of a new node. The degree of similarity is controlled by the *vigilance* parameter.

The ART model assumes no teaching input and performs unsupervised learning. It organizes itself to group "similar" input patterns into the same category. Category proliferation is controlled by the vigilance parameter. An ART system with low vigilance will permit grouping of patterns that are only grossly similar, and a system with high vigilance will try to form separate categories for patterns that have only minor differences. In the ART2/BP network, Sorheim uses the ART2 [Carpenter and Grossberg 1987] model to build a supervised backpropagation network in his attempt to resolve the stability-plasticity dilemma [Sorheim 1991]. A simple backpropagation net is connected to each output unit of the ART2 subsystem. The competitive learning occurs in the ART2 subsystem, and no competition exists among the backpropagation nets.

2.2 Other adaptive learning models

SCL is akin to Kohonen's topology-preserving maps in general [Kohonen 1982], and Learning Vector Quantization in particular [Kohonen 1988, 1990]. LVQ processors depend on their neighbors to establish boundaries, and require more processing units (ten per category instead of

the one to four prototypes typical of SCL). Also SCL prototypes are topologically independent; a classification category may be represented by prototypes scattered widely (and disconnected) over the feature space (cf. Section 4 below). In this respect our work is closer to that of Reilly et al. [1982]. Their work recognized the value of multiple prototype formation as well as the retention of training examples for each prototype, though employing different control mechanisms. A potentially unlimited number of prototypes may be formed (a new one from any new pattern in an unclaimed region of the feature space). Each such prototype completely defines the associated response. SCL (as ART) combines "exemplar" patterns in the formation of more general prototypes. This would be expected to significantly reduce the number of prototypes necessary to categorize incoming patterns. Our experiments tend to confirm this. Reilly et al. were able to recognize 98% of handwritten digits in a test group (isolated from the pool of training digits).

Fukushima's Neocognitron [1988] uses a hierarchy of progressively more abstract feature detectors (e.g. eleven "cell planes" in a digit recognizer). Pixel data is entered at the "lowest" level. At an intermediate level, such features as horizontal, vertical, and diagonal strokes are detected (akin to the more recent work of Fontaine and Shastri, discussed in the next section, and similar to the "grid" features of Grothe [1991] and Lin [1982], and used by SCL). The classification categories themselves form the highest level.

2.3 Off-line handwritten character recognition

Due to the sponsorship and data standardization of the postal systems of the United States and Great Britain, much attention has been given to off-line recognition of handwritten letters and digits, often called optical character recognition (OCR). Despite nearly four decades of research, results have remained modest. Some of the best results to date on USPS ZIP code digit images are reported by Fontaine and Shastri [1992] using a system that converts the static representation of a character into a time-varying signal. This greatly simplifies the dimensionality of the problem, and led to impressive results: 96% recognition of test digits with no rejections. If 15% of the digits are rejected, the recognition rate rises to 99%.

Stonham and Nellis [1992] experimented with handwritten addresses made available from the British post. Their work involved "discriminators" compounded from Random Access Memories [Aleksander and Norton 1990]. Their system uses a masking technique for balancing the pixels between white and dark to accommodate the RAMs, and turns off certain discriminators if not recently used. They report recognition of 93.95 of test letters and digits (a substantially more difficult task than just digits) in context-free recognition, and higher results with the addition of context-related data (Viterbi maps, etc.).

2.4 On-line handwritten character recognition

The advent of pen-based computing is accelerating the already roused interest in the recognition of characters in real time. In a broad survey of the state of the art, Tappert et al. [1990] note that the recognition task is simplified by having actual temporal data. (For example, it is much simpler that the clever device of superimposed temporalizing by Fontaine and Shastri.) The on-line recognizer observes the number, order, and speed of the individual strokes composing the

input pattern. This may be a mixed blessing since characters that appear identical when complete may be formed by very different processes. (For example, a nine may be formed from the bottom to the top, or vice versa, or by separate strokes for the stem and the circle.) They also note that an on-line recognizer can always request clarification of ambiguous data, and that users tend to adapt (even unconsciously) to forms that are more palatable to the recognizer.

Much of the significant research in this area is regrettably obscured by the commercial potential of pen-based computers. Gibbs [1993] gives vendor-provided data on several commercial recognizers. With welcome candor, Go Corporation has published its work in handwriting recognition within the PenPoint Operating System [Carr and Shafer 1991]. GO's recognition engine appears typical of commercially crafted recognizers (as opposed to research vehicles). It includes hundreds of prototypes for its alphabet of interaction: 82 symbols (52 uppercase and lowercase letters, 10 digits, and 20 punctuation marks) and 50 gestures. The system allows for training sessions by a particular user in any of these symbols and gestures. These training sessions are distinct, that is, not integrated into the normal recognition activity.

In the recognition of what Carr and Shafer describe as "neatly printed" characters (after some training of the users), the PenPoint system recognizes on average of 94% of the 62 symbol set that omits punctuation. (Although, there is rich provision for context support such as Viterbi mappings, dictionary support, and constrained fields, the performance reported in this paper focuses on the hardest task: context-free recognition.) Oddly, when the input is constrained to 10 digits, the recognition barely improves, to 92-95%. When constrained to 26 lowercase characters, the recognition rate is 89-93%. Since these results were reported, Go has released an improved version of the recognition engine. We are currently in the process of compiling our own data on its performance.

The Penpoint recognition engine requires about 64 KB of code and about 128 KB of static data. On a 16 MHz 80286-based system, it can perform about 3 recognitions per second. This is close to the fastest sustainable writing speed with segmented characters. Other recognizers in the Gibbs [1993] review claim speeds (on 16 MHz 80286-based systems) in the range of 4 to 20 recognitions per second. The highest recognition performance for the Western alphabet, 98%, was claimed by Communication Intelligence Corporation. (In addition to Gibbs, see also Wagoner 1993.)

These performance data set the stage for considering the usefulness of the techniques introduced by the research vehicle, SCL. Before considering that, we turn to the issue of feature abstraction; no recognizer can outperform the abstraction of features available to it.

2.5 Energy-based feature abstraction

The learning environment places a number of demands on the interface, and especially on the choice of handwriting recognizers. The system must accommodate a potentially very large market. More than a million children in 18 countries now take Kumon classes. (For more information about Kumon and the Kumon Machine which will use SCL, see Fuller [1991]) Yet the system must respond to the particularities of the individual user. This requires a recognizer that is general and universal enough to permit any Kumon instructor, parent, or student to enter

initial data such as name, address, goals, etc. Yet as the learning sessions commence, it must respond quickly and accurately to the input from a relatively small number of users -- typically just a child or a few in one family. In school settings this might include a class sharing a system.

Much of the research about handwriting recognition (as noted above) has centered on off-line recognition and consequently has rarely considered the dynamic features associated with the actual creation of the characters. Our research suggests that some dynamic features of handwriting, especially those described below, might be unexpectedly uniform across international populations, and strongly stable in individuals through time, thus meeting both elements of the above demands.

Our system samples the velocity and acceleration of the pen as it creates digits, letters, and gestures on the two-dimensional digitizing tablet. (Although the writing instrument is more properly a *stylus* than a pen, these systems are universally known as pen-based, and certainly not stylus-based. Therefore we use "pen.") The germ of the notion of decoding handwriting with energy-based features was in the method used by document examiners. A forged signature may be detected by microscopically examining its sharp turns (such as the top of a "t" or the bottom of a "v"). At normal writing speed, the centrifugal force of the pen throws off tiny droplets of ink as the pen "rounds a bend." At the slower speed of most forgers, these droplets are absent or much closer to the inked line. The forger is able to mimic the graphical (static) features of the original, but not the dynamic. Conversely, the real signer may vary the pattern somewhat from signature to signature, but as this work and that of others (e.g. Wagoner 1993) shows, the velocity and acceleration features tend to remain constant.

In October of 1991, the authors were discussing the CELP algorithm for compressing voice (Codebook-Enhanced Linear Prediction). It derives its advantage to some degree from the consistent use of musculature and vocal apparatus by human speakers. It occurred to us that this might also be true of writers. For example, we write a "9" as successfully with our eyes closed as with them open. The graphical feedback, while useful for positioning the digit, is unnecessary for creating it. The signal is apparently encoded in what is popularly called "muscle memory" -- conceivably in the form of amount of energy to be dissipated in particular muscles to achieve the task. The notion of such "ballistic" muscular action (i.e. unmediated by feedback) was first proposed by Lashley in 1917, and applied to signature verification by von der Gon and Thuring in 1965. Herbst and Liu [1982] used this model to construct a signature verifier. They observed

"Feedback control in response to error signals from the eyes or the hand requires on the order of 200 ms while the time it takes for the individual muscle forces used to generate the strokes is 30 to 100 ms. These rapid motions are characterized by the fact that their durations are predetermined by the brain, and that the agonist and antagonist muscles for each degree of freedom are alternately energized."

It should be possible to recover this energy encoding via the velocity and acceleration of the pen to a degree sufficient to determine the character or gesture being created. (Cf. Thomas [1972], 289-293]. Acceleration of the hand on the writing surface was used by Herbst and Liu [1982] in their signature verification system (cf. Wagoner [1993]). They used quite sophisticated hardware: x- and y- accelerometers, and resistance foil-strain gauge to capture "downward" pressure as well; then the data was passed through a linear phase recursive digital filter.

Tappert et al. [1990] indicate the difficulty of making useful comparisons given the wide diversity of experimental environments (e.g., writer dependency, size of alphabet, hardware capabilities). The best reported results for Western alphabets were in the range of 95% to 97% with recognition requiring from .1 to 1.0 seconds per character. This is a bit better than the results reported by GO with PenPoint. Recently some commercial providers have claimed as high as 98% recognition [Gibbs 1993].

2.6 Grid-based feature abstraction

For all the advantages of energy-based features and their strong success in identifying digits, it became clear that a stronger recognizer could be constructed from combinations of feature sets than from any single feature set alone. To our dynamic features (velocity and acceleration), we needed to add some static features. Many have been described in the literature.

Fukushima's Neocognitron [1988] includes several layers of neural discriminators to detect features such as the orientation of pattern subsets. These are progressively associated in more abstraction in higher layers. Nellis and Stonham [1992] employ a very different learning engine (random access memory units combined into associators), but indicate the power of clever static feature abstraction. Their work also supports the effectiveness of varying prototypes and of disabling infrequently-used classifiers. Recognizers may be combined hierarchically as well (cf. Fontaine and Shastri [1992], Kohonen [1990]).

Ultimately we settled on a relatively unsophisticated system for considering static features. Rob Grothe [1991] suggested a simple 3 by 3 grid overlaying the bounding rectangle of the (smoothed) input pattern (cf. Fontaine and Shastri [1992]. Lin [1982] used a similar 2 by 2 grid.). Our grid-based features are described below.

3 The SCL model

3.1 Requirements and environment

SCL is intended to be a general input recognition system for pen computers used by Kumon mathematics students. Currently about a million learners (75% elementary) employ Kumon as a supplement to the mathematics taught in regular schools [Fuller, 1991]. Input patterns vary from alphanumeric characters to geometric shapes such as circles and rectangles. Users vary from young children to adults. Thus the system must be open-ended; its implementation demands adaptive coding for a complex environment, embracing different character sets and different handwriting styles. It must remain constantly in the learning (training) mode. When the system mistakes the input pattern, the user may be asked to enter the correct response; the system will then train itself until it recognizes the new pattern consistently. SCL should learn new characters rapidly enough (perhaps though background processing) not to interfere with the stream of input from the user. Finally, it is desirable for SCL to be able to recognize digits that are not written in preprinted boxes; that is, it should be able to recognize multi-stroke digits in real time, performing its own segmentation of the input strokes into digits.

3.2 SCL model and algorithm

The schematic definition of SCL is shown in Figure 1. SCL receives the input pattern X and outputs the category name C. If the system fails to produce the correct category name, then the user gives the correct name to the system as the teaching value Y. The system learns the association between X and Y so that it may respond correctly to a similar input X next time.

An SCL recognizer consists of a set of N (>0) prototype units (attention subsystem) and a Selector (orienting subsystem). Each prototype unit, n_i ($1 \le i \le N$), is responsible for identifying all the input patterns that belong to a particular subcategory (prototype). When the input pattern X is given, the output value, $n_i(X)$, from the unit n_i represents the certainty of X belonging to the subcategory of the unit. Or equivalently, it represents the similarity between the input pattern and the combined prototype pattern of that subcategory. We assume that the output of each prototype unit is normalized to [-1.0,1.0]; i.e., $-1.0 \le n_i(X) \le 1.0$.

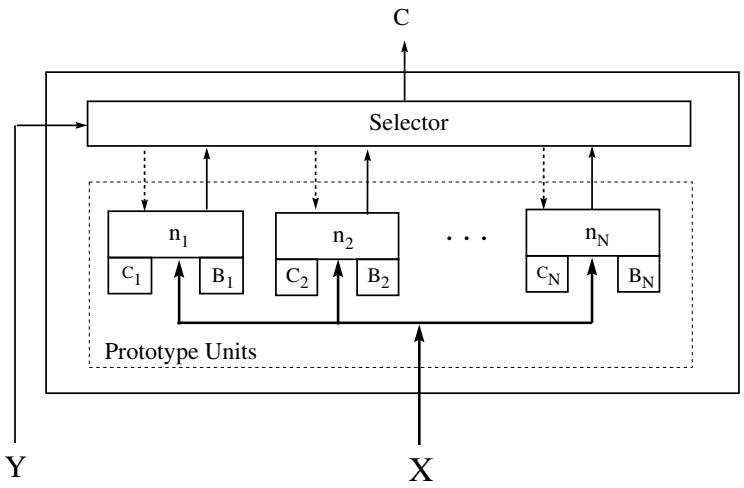

Figure 1 SCL schematic

The Selector selects, as the winning unit, the unit whose output is larger than a threshold value and the largest among those above the threshold. Then it produces the name of the category to which the winning unit belongs as the output of the SCL system. If there is no winner, then the input X is considered to be a member of a new subcategory, and a spare (unused) prototype unit is assigned to represent it. If no spare unit is available, the unit winning least frequently, presumably representing the least significant subcategory of input patterns, will be assigned to represent the new subcategory.

If the winning unit is wrongly selected, the teaching value, Y, is used to train the prototype units as follows: The units of the category Y that respond to X with low outputs will be trained to increase their outputs for X up to σ_H. Those units representing categories other than Y that respond to X with high outputs will be trained to decrease their outputs down to σ_L. We assume that each unit is trainable to produce high output values for members of its subcategory and to produce low output values for non-members.

Associated with each prototype unit, n_i, the system maintains the following information: the name of the category, C_i , to which the subcategory belongs, a set of *exemplar* input patterns, B_i, that are selected by n_i, and the frequency, f_i , the number of times n_i has won the competition. Initially each unit has the null category name, Λ.

The algorithm for the Selector is given below:

	Parameters:	$-1.0 < \sigma_L < \rho < \sigma_H < 1.0$
	Initialization:	$C_i = \Lambda,\ f_i = 0,\ B_i := \phi$ (the empty set) for $1 \le i \le N$.
Recognize:	1.	Get an input pattern X and present it to the n_i.
	2.	$K := \{ i \mid C_i \ne \Lambda$ and $n_i(X) > \rho \}$.
	3.	If $K = \phi$ then Produce Λ; Goto 7.
	4.	Find j such that $n_j(X) = \max\{ n_i(X) \mid i \in K \}$.
	5.	Produce C_j.
	6.	If accepted then Goto 1.
Train:	7.	Get the correct category name Y.
	8.	$K := \{ i \mid C_i = Y$ and $n_i(X) > \rho \}$.
	9.	If $K = \phi$ then Goto 14.
	10.	Find j such that $n_j(X) = \max\{ n_i(X) \mid i \in K \}$.
Add exemplar:	11.	$B_j := B_j \cup \{X\};\ f_j := f_j + 1$.
	12.	Train n_j with $\{(X,1)\} \cup \{(u,1) \mid u \in B_j\}$ until $n_j(X) > \sigma_H$.
	13.	For all i such that $i \notin K$ and $n_i(X) \ge \sigma_L$, train n_i with $\{(X,-1)\} \cup \{(u,1) \mid u \in B_i\}$ until $n_i(X) < \sigma_L$; Goto 1.
Add prototype:	14.	$K := \{ i \mid C_i = \Lambda \}$.
	15.	If $K \ne \phi$ then Select $j \in K$; Goto 17.
	16.	Find j such that $f_j = \min\{ f_i \mid 1 \le i \le N \}$.
Initialize unit:	17.	$f_j := 0$; $B_j := \phi$; $C_j := Y$; Goto 11.

When the input pattern X is received, each unit predicts the certainty that X belongs to that unit's subcategory. The unit, n_j, with the largest output value greater than the *vigilance* ρ, is selected as the winner, and its category name C_j is given as the output. If the output is correct (Λ never being correct), no prototypes are changed. Otherwise, the system receives the correct category name Y from the user. If there is no winner, but a unit remains with the empty name Λ, it becomes the winner with Y as its category. If there are no more units with Λ, then the unit with the smallest frequency count will become the winner after re-initializing its settings. The winning unit updates its history set B_j by concatenating X to it and increments its frequency count f_j. Then the winning unit is positively trained on all the patterns in B_j until the output value $n_j(X)$ becomes greater than the *high confidence value*, σ_H, for the input pattern X. Those losing units n_i whose category name is not Y and whose output value $n_i(X)$ is greater than σ_L, get

positively trained on all the patterns in B_i as well as negatively trained on X until $n_i(X)$ becomes less than the *low confidence value* σ_L. Losing units with category name Y are not trained.

3.3 Refinements to the model and algorithm

Extensive experimentation led to a number of refinements in the training regimen. In early experiments, SCL tended to be overly sensitive to small differences in the patterns, resulting in nearly as many prototypes as exemplars. This could partly be compensated by low values of ρ, but was further strengthened by dividing the training into phases with different rules. The initial training (while gathering exemplars) was limited (usually into 1000 or 2000 backpropagations) for each new prototype. Exactly half of this new training was for the newest prototype; the rest was for prototypes that either responded too weakly to their own patterns ($<\sigma_H$) or too strongly to other patterns ($>\sigma_L$). After the available patterns have all been presented, some number of these patterns (fewer than half) have been assimilated into the exemplar sets, B_j. The prototypes then undergo training without competing for new exemplars.

This method helped, but the performance reached a plateau caused by unfortunate collection of exemplars. Since the system starts "from scratch" with no prototypes, the first prototype for a category is quite indiscriminate. As it settles down, the added wisdom of more mature weights may cause the first few exemplars to become embarrassing oddities compared to its current profile. For an actual example, one digit prototype, n_s, gathered eights made with two circles ("snowman eights"), but since it was the only eight collector around, it eventually gathered other eights made by starting at the central intersection ("inside-out eights"). By this point in the training, another prototype, n_i, had started gathering the snowman eights which were no longer within ρ similarity of the original n_s prototype. Obviously, it would be desirable to have n_s give up its snowman eights to n_i. This desirable result is achieved by permitting one, two, or three "shuffles" during the training. A shuffle frees all exemplars ($B_i := \phi$ for $1 \le i \le N$). Prototypes retain weight values to compete in a new round of exemplar-gathering. Those which gather no exemplars are removed. The others generally gather a much better coordinated set of exemplars, leading to better classification performance.

3.4 Contributions of SCL as compared with similar learning models

The distinctive aspect of Supervised Competitive Learning is precisely its "competitiveness." As with all competitive systems, the prototype modules compete to recognize the input pattern. SCL modules also compete to gain a collection of exemplars, and they compete for training time. Modules with more exemplars (larger |Bj|) receive proportionately more training backpropagations (until reaching the σ_H and σ_L criteria). The system is self-initializing. It builds classes from the stream of incoming patterns, and does not depend on "exemplary" exemplars to perform robustly, as do many pattern recognizers, such as the Neocognitron. The model of Reilly et al. assigns precisely one exemplar/prototype per "neuron" and then trains it by appropriately extending or contracting its range of influence (its "turf"). SCL's training regimen rewards prototypes which successfully draw more exemplars, presumably encouraging the generalization that emerges from the natural "averaging" of the backpropagation through the shared weights.

Next we briefly examine the derivation of features used as input to SCL. This is divided into the energy-based and grid-based features.

3.5 Energy-based feature abstraction in SCL

We adopt the convention introduced in PenPoint [GO Corporation 1991] of defining incoming patterns (digits, letters, or gestures) as "scribbles." A scribble S is composed of one or more strokes Ks, which in turn are each composed of one or more points.

$$S = (K_1\ K_2 \ldots K_k)$$
$$\text{where each } K_s = (P_{s_1}\ P_{s_2} \ldots P_{s_{p_s}})$$

where k is the number of strokes in S, and p_s is the number of points in stroke K_s.

The intent is to represent the average velocity and acceleration in the x and y directions for each of the sampling intervals of the scribble. The algorithm assumes that the point sampling period of the input digitizer is constant over time. Actually, it stays close to 5 ms, but varies slightly. This doesn't materially affect the algorithm's success since variances are averaged out over many samples.

The points of all strokes of the scribble are gathered into unified arrays

$$X = (\ x_{P11}\ x_{P12} \ldots x_{P1p1}\ x_{P21} \ldots x_{Pkp_i}\ )$$
$$Y = (\ y_{P11}\ y_{P12} \ldots y_{P1p1}\ y_{P21} \ldots y_{Pkp_i}\ )$$

and p_t is the total number of points in all strokes. The differences between successive points are loaded into velocity arrays V_X and V_Y. The differences between successive velocities are stored in acceleration arrays A_X and A_Y. The number of velocity sampling intervals V_{samp} is fixed for a given network, but networks were built for several values of V_{samp} from 4 to 9. A typical value is 6. V_{samp} determines the boundary points b_m of the sampling intervals. For each velocity interval, we calculate the average velocity, and set the corresponding velocity feature equal to this average divided by the maximum velocity for the entire scribble.

$$V_{Xmax} = \max (V_{X_j}) \qquad 0 \le j < p_t$$

$$V_{Xavg_m} = \left(\sum_{j = b_{m-1}}^{b_m} V_{X_j} \right) / \left(V_{Xmax} * (b_m - b_{m-1}) \right)$$

Only this final operation uses real values; the prior calculations may be handled as short integers. Similar normalized velocities are calculated for the y velocity vector, V_Y, and for the x and y acceleration vectors, A_X and A_Y. These normalized ratios serve as real-valued inputs to the SCL recognizer. See Fuller [1992] for a fuller explanation of the use of energy-based features.

3.6 Grid-based feature abstraction in SCL

Although the use of energy-based features gives quite respectable recognition performance in many settings, it was not found satisfactory for recognizing letters. The diversity of handwritten letters greatly surpasses that of numerals, as the following examples from our data illustrate. In the following paragraph, the 4-digit numbers refer to the tag numbers by which we cross reference the writers and sequence of writing. The lower case letters in parentheses indicate the intended character.

Character 2128 (which is actually a 'c') could be mistaken for 'l' or 'e.' Compare it with this 'e' 5147, . In this pair, 1238 (n), the first, strongly resembles 3179 'h.' Consider the following word:

These are the letters 5168 (g) 5108 (a) 5170 (g)1289 (s), that is, the word "gags." The following 4308 (u) resembles 'n.' This 5103 (a) resembles a 'q.' Finally, this 6280 (r) would pass for a 'v' much more easily than an 'r.'

Not surprisingly, human recognition of context-free handwritten characters peaks at 97 to 98%. We humans compensate by using context information to supply reasonable guesses. We needn't belabor this point. Ask any school teacher or pharmacist!

Velocity and acceleration are first and second order features of input patterns. In view of the difficulties illustrated above, it was decided to present some "zeroth order" graphical features to SCL. Several were tried including grid-based orientation features and angular position, velocity, and acceleration. After numerous experiments, we settled on a simple grid-based scheme suggested by Grothe [1991].

For each input scribble, the bounding rectangle is divided into thirds in each direction, creating nine zones (numbered 0 to 8 horizontally from the top left as seen in Figure 2). For each consecutive pair of points in each stroke (but not crossing stroke boundaries), we calculate the zone into which the majority of the segment falls. Within each zone are four orientations enumerated 0 to 3: horizontal, right diagonal, vertical, left diagonal. We determine the arctangent of the segment (making the y difference positive to limit the angle to the first two quadrants). Then we add the length of the segment to the selected orientation of this zone as follows:

horizontal:	$0 <= \arctan ß < \pi/8$ OR $7\pi/8 <= \arctan ß <= \pi$
right diagonal:	$\pi/8 <= \arctan ß < 3\pi/8$
vertical:	$3\pi/8 <= \arctan ß < 5\pi/8$
left diagonal:	$5\pi/8 < \arctan ß < 7\pi/8$

These comparisons are done with pre-calculated constant ratios rather than by calls to the arctangent function in the interest of efficiency. When this has been completed for all point pairs, the values are normalized by dividing each of the 36 length sums by the total length of all 36 values.

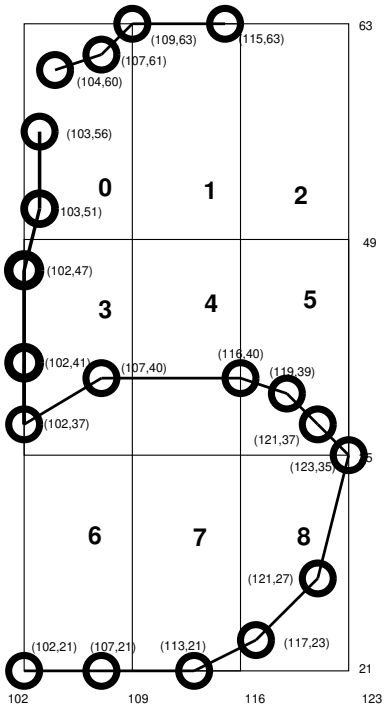

Figure 2 The numeral "5" in the feature grid

3.7 Additional considerations in feature computation

In actual practice, several refinements proved necessary. Since the on-line system is vulnerable to the whims of the user, some safeguards are required. Input is limited to 1000 points per stroke and 10 strokes per scribble; data beyond these are discarded. Furthermore, the feature computation would be very unwieldy for 10,000 points since SCL must recognize characters in real time. We ran tests using larger quantities of sampled points, but recognition was not improved by using any more than 60 sample points. If the incoming scribble has more than 60 points, the preprocessor performs a stroke-wise reduction to 60 points (for the entire scribble) by selecting points at constant intervals throughout the strokes.

It was also found helpful to exponentially smooth out anomalies in the digitized input. This is similar to weighting the past several points in decreasing degree, but requires the retention of only the last (smoothed) point. This technique is widely used by statisticians to smooth data in quest of significant trends. For an example, see pages 163-5 of Fuller [1987].

Finally, the selection of the velocity and acceleration intervals will not necessarily coincide with stroke boundaries. This is potentially useful in making the "virtual" stroke between the end of one stroke and the beginning of the next susceptible to the usual measures of velocity and acceleration. As shown below, this assumption yielded fine results. However, the happenstance

nature of the "fall" of the break (e.g., in a two-stroke "4") introduces an unnecessary oddity in the input feature vector. In later work, we adjusted the sampling interval to "stretch" to the nearest stroke boundary, and abandoned all data related to the "virtual" strokes between the collected ("actual") strokes. This resulted in both improved performance of the recognizers and reduced complexity (fewer hidden units) in the prototype units.

3.8 Network architecture

Network architecture began with simple considerations of primitive backpropagation neural nets [Rumelhart and McClelland 1986] with extensions to include connections directly from input to output [Kimura 1990].

Each unit is an acyclic backpropagation net consisting of 3 layers; typically, 27 input units, 2 hidden units, and 1 output unit. The input layer is fully connected to both the hidden layer and the output layer. The hidden layer is fully connected to the output layer. There are no lateral connections, that is, no connections within the same layer. We used the activation and error functions of Kalman and Kwasny [1991, 1992], namely,

$$a = \lambda(x) = (1 - e^{-x}) / (1 + e^{-x}) \qquad \text{(activation function)}$$
$$e = \Sigma (c^2 / (\lambda'(x)) = \Sigma (c^2 / (1 - a^2) \qquad \text{(error function)}$$

c is the difference between the *training value* (1.0 if correct, -1.0 if incorrect) and the actual activation value of the prototype unit (which ranges from -1.0 to 1.0). The learning rate and the momentum value (η and α in the tables below) were typically fixed to 0.0005 to 0.001 and 0.4 to 0.9, respectively. Momentum tends to let the networks escape from local minima. (Cf. [Becker and le Cun 1988], [Fahlman 1988], and [Shynk and Roy 1988].)

4 Experiments

4.1 Early recognition performance with backpropagation

The first task for SCL was to recognize handwritten digits, 0 - 9, collected on a pen-based Lombard computer from GO Corporation. The data is captured by the x and y movement of the pen on a digitizing tablet. For example, the raw data for the five scribble shown in Figure 2 above is sampled by the tablet as:

```
108 5 2
15  103,56 103,51 102,47 102,41 102,37 107,40 116,40 119,39
    121,37 123,35 121,27 117,23 113,21 107,21 102,21
4   104,60 107,61 109,63 115,63
```

This scribble has the identification tag 108, value 5, and consists of 2 strokes. The first stroke consists of 15 points and the second 4 points. The velocity is divided into 7 intervals and the acceleration is divided into 6 intervals in these early tests. The average X and Y components of velocity and acceleration for each section are divided by the maximum values to generate 26 floating point features, and the stroke count is used as the 27th feature.

SCL was limited to 40 prototype units (N = 40). These early experiments used only energy-based features to recognize handwritten digits (600 test digits after "Trials" backpropagations on 800 training digits) collected from 20 subjects on the Lombard computer. A trial is one backpropagation for one prototype unit. In the first experiment below, for example, since 19 prototypes in total were generated, each unit was trained an average of 22,000 times. After the training iterations, the system was tested by 600 digit patterns obtaining 92.7% recognition. The column "Time" is training time on a NeXTstation (25 MHz MC68040-based; rated at 15 MIPS).

SCL Parameters				Hidden	Prototypes		Time	
η	σ_H	ρ	σ_L	units	created	Trials	(min)	Recognition
(GO-collected digits -- Energy only)								
.0005	0.5	-0.10	-0.5	4	19	211,761	<20	92.7%
.0005	0.5	-0.10	-0.5	4	26	2,791,989	<250	94.8%
(GO-collected digits -- Energy/Graphical)								
.0010	0.3	0.05	-0.5	4	35	648,287	32	97.5%

Table 1 Early SCL Results

Several later experiments used a combination of 57 to 63 real-valued features (stroke count, 20 to 26 energy-based features, and 36 grid-based features). These experiments with our actual pen-based prototype (Kumon Machine, or KM) used 1600 digits (1280 train, 320 test) and 4160 letters (3328 train, 832 test). Improvements to the preprocessing of feature extraction and the reduction to two hidden units significantly sped up learning convergence. The 63 features, training set, and testing set are identical to those used for SCL fuzzy logic-based digit training reported in Table 3 below, permitting comparison of SCL under the two types of learning modules.

SCL Parameters				Hidden	Prototypes		Time	
η	σ_H	ρ	σ_L	units	created	Trials	(min)	Recognition
(KM-collected digits -- Energy only)								
.0010	0.9	0.10	-0.5	2	39	712,466	9	98.4%
(KM-collected digits -- 63 Energy/Graphical features)								
.0010	0.8	0.10	-0.5	2	29	65,921	2	97.5%
.0010	0.9	0.05	-0.5	2	35	906,315	19	99.7%
(KM-collected letters -- Energy only)								
.0010	0.9	0.10	-0.5	2	144	3,383,493	<200	91.1%
(KM-collected letters -- Energy/Graphical)								
.0010	0.9	-0.10	-0.5	2	139	2,079,554	139	95.2%
.0010	0.9	-0.05	-0.5	2	143	3,271,052	181	95.7%

Table 2 Improved SCL Results

4.2 Recognition performance with fuzzy logic

In another paper [Wang, Kimura, and Fuller 1992], we reported the results of using prototype classifiers based on fuzzy logic. Each such prototype unit responded with a number between zero and one indicating the degree of membership in the set defined by the exemplars associated with that unit. Each new exemplar requires only a recalculation of the weights akin to a rolling average. This greatly reduces the CPU load. Such fuzzy logic has no counterpart to the negative training that was found necessary to prevent "greedy" prototype detectors from seizing neighboring turf. For this reason, the test results were not quite as high as those for backpropagation. The results are reported below as Table 3. Similar tests with a gesture set resulted in 91% recognition. See Wang [1992] for a fuller description of SCL with fuzzy logic.

Parameters				Confidence	Number of	Training	Recognition Rate	
α	E	F	z	Value ρ	Prototypes	Time (Min.)	Train(800)	Test(600)
-1	1	1	1	0.8	88	4	96.62%	95.67%
-1	1	1	1	0.85	88	4	96.88%	96.83%
-1	1	1	1	0.9	157	11	99.50%	97.17%

Table 3 SCL Results using fuzzy logic prototype units

4.3 Training schedules, preprocessing and the complexity of individual prototype units

In our earliest experiments with SCL, the individual recognizers embedded significant complexity in the form of 4 to 8 hidden units. In experiments with recurrent neural networks after the model of Servan-Schreiber, Cleeremans, and McClelland [1988], we found that we could maintain comparable performance with simpler prototype units. In fact, by concentrating the temporal data within a pool of shared hidden units, the individual prototype units could be reduced to no hidden units, or linear classifiers.

This is emblematic of a trend found throughout this investigation. As preprocessing and scheduling of training have become more sophisticated, the complexity of the prototype recognizers has been sharply reduced. The reduced complexity fully compensates the additional time required for preprocessing and testing for inclusion in the training schedule. The overall training and recognition times have fallen significantly.

No single element can account for the improvements, but several milestones deserve mention. As described above, the earliest versions of the energy-based features were blind to the mapping of the segmentation imposed by the sampling process and the natural segmentation of the constituent strokes of the character. Since many characters have only a single stroke, this practice was less than disastrous (and simplified the preprocessing). Later, SCL adjusted the sampling segments to coincide with stroke boundaries, and simultaneously to disregard the implicit acceleration associated with the movement from the end of one stroke to the beginning of another. (This movement may be considered a "virtual" stroke.) This added to the preprocessing burden,

but improved training and recognition performance.

The greatest improvement came through the shuffling of exemplars described above. This led to a more "mature" selection of exemplars (i.e. exemplars which are all closer to the final weight space of the trained prototype units). Experiments involving various schedules of exemplar competition and training (including one, two and three shuffles with appropriate training between them) show no significant differences. Although "shuffling" and subsequent retraining adds to the "overhead" of the training task, it has led to simplified prototype units (the prototype units reported below are linear) and more rapid weight convergence.

4.4 Current results

The following table summarizes results incorporating the above-described improvements. We also report a more broadly-based performance statistic: the five-fold cross validation after Weiss and Kulikowski [1991]. For a file of either digits or letters, we divide the file into fifths, then run five training sessions where one fifth is held in reserve for testing and the other four fifths are used for training. This number is shown in Table 4 as the average of the five sessions.

SCL Parameters				Hidden	Prototypes		Time	
η	σ_H	ρ	σ_L	units	created	Trials	(min)	Recognition
(KM-collected digits -- Energy only)								
.0010	0.7	0.00	-0.5	0	31	1,178,887	4	100.0%
Average for all five sessions: 99.0%								
(KM-collected digits -- 63 Energy/Graphical features)								
.0010	0.7	-0.15	-0.5	0	28	118,768	<1	99.0%
.0010	0.7	-0.15	-0.5	0	29	755,302	5	100.0%
Average for all five sessions: 99.5%								
(KM-collected letters -- Energy/Graphical)								
.0010	0.7	-0.05	-0.5	0	113	452,454	4	90.0%
.0010	0.7	0.00	-0.5	0	121	7,635,982	71	97.8%
Average for all five sessions: 96.6%								

Table 4 Current SCL Results

4.5 Recognition speed

Each recognition requires a forward propagation through all the prototypes in the search for the maximal activation. Thus, the recognition time is approximately O(p+n) where p is the time for preprocessing and n is the number of prototype units. With 122 prototypes, lowercase alphabetic recognition on a 25 MHz MC68040-based NeXTstation takes about 15 ms of which about 2 ms are required for preprocessing. This corresponds to about 65 characters per second. A numeric recognizer (with a third the number of prototypes) requires the same 2 ms for preprocessing and about 4 ms for the forward propagations, or 6 ms total.

4.6 Behavior of prototypes in gathering exemplars

Figures 3 and 4 illustrate SCL's performance in gathering exemplars that are "similar" in the combined energy-based and grid feature space. Figure 3 shows all 27 prototype units of a digit recognizer. Note that Prototype 13 garnered one-stroke fives and left the two-stroke fives to Prototypes 12 and 14. Among the eights, Prototype 21 gathered the "usual" eights, Prototype 22 gathered the eights made by starting at the center and forming the stroke in an upward counterclockwise direction, and Prototype 23 gathered eights formed by starting at the center and upward in a clockwise direction. Prototype 24 gathered "curly" nines. Prototype 25 gathered "usual" nines, and Prototype 26 garnered two-stroke nines.

Following are several examples of training set members selected by prototype units as being "similar" among the letters used for training. Note that with a trivial exception (one letter), only two prototypes were enough to categorize all examples of the letter 'r.'

4.7 Confidence

An independent measure of confidence in a recognition helps to predict rejection rates if particular levels of performance are mandatory. If a recognizer responded "perfectly" to an input, the appropriate prototype unit would show the highest possible activation for the sigmoidal function, 1.0. Other units would respond with the lowest activation, -1.0. We define the *confidence* of a recognition as the difference between the highest activation for a category and highest activation for any second category divided by this limit of 2.0. In other words, the confidence of a recognition r is

$$C_r = (\lambda_i - \lambda_j) / 2 \quad \lambda_i \geq \lambda_j \geq \lambda_k \text{ all k: } C_k \neq C_i \text{ and } k \neq i,j$$

where λ_i is the activation output by n_i, the i^{th} prototype unit

SCL exhibits a high correlation between confidence and correctness. Recent results with digits (the average of the five-fold cross validation is 99.5%) limit the relevance of this measure, but it is valuable in the recognition of letters and gestures. The following table illustrates the correlation between confidence and correctness. In this particular test, SCL identified 97.4% of the test set (lowercase letters) correctly. That is, it correctly identified 760 of the 780 test letters. In 13 of the 20 cases of misclassification, the second highest guess is correct. For 373 of the 780 recognitions (47.8% of the test set), the confidence was above 0.45; for this group — nearly half of the test set — the recognizer was 100% correct. For 596 of the 780 recognitions (76.4% of the test set), the confidence was above 0.25; the recognizer was 99.5% correct for these. For 735 of the 780 (94.2%), the confidence was above 0.10 and the recognition rate was 98.5%.

Figure 3. Digit recognizer with exemplars gathered by the prototypes (*Next page*).

Figure 4. Subset of a letter recognizer with exemplars gathered by the prototypes (*Second page following*)

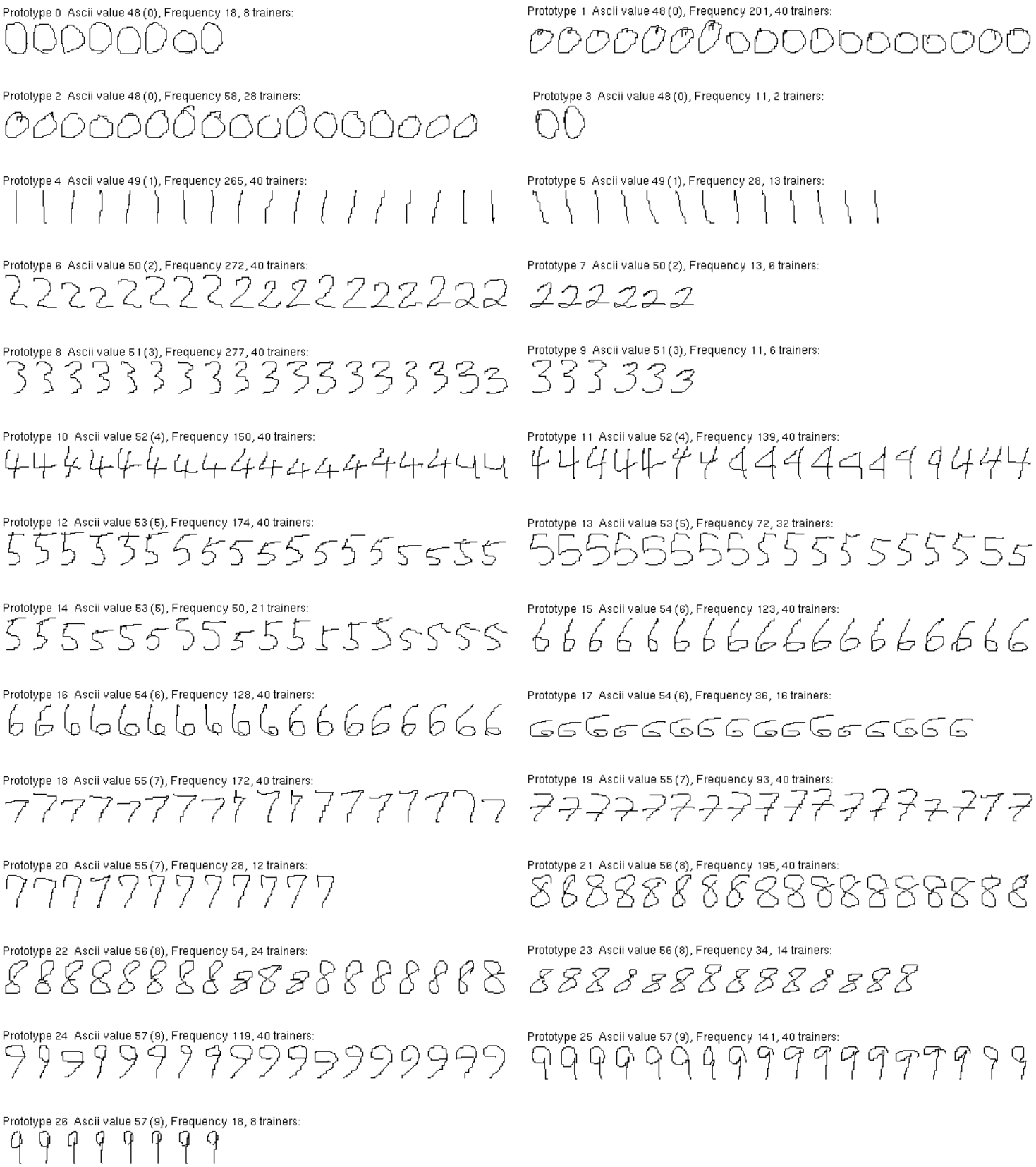

Figure 3. Digit recognizer with exemplars gathered by the prototypes

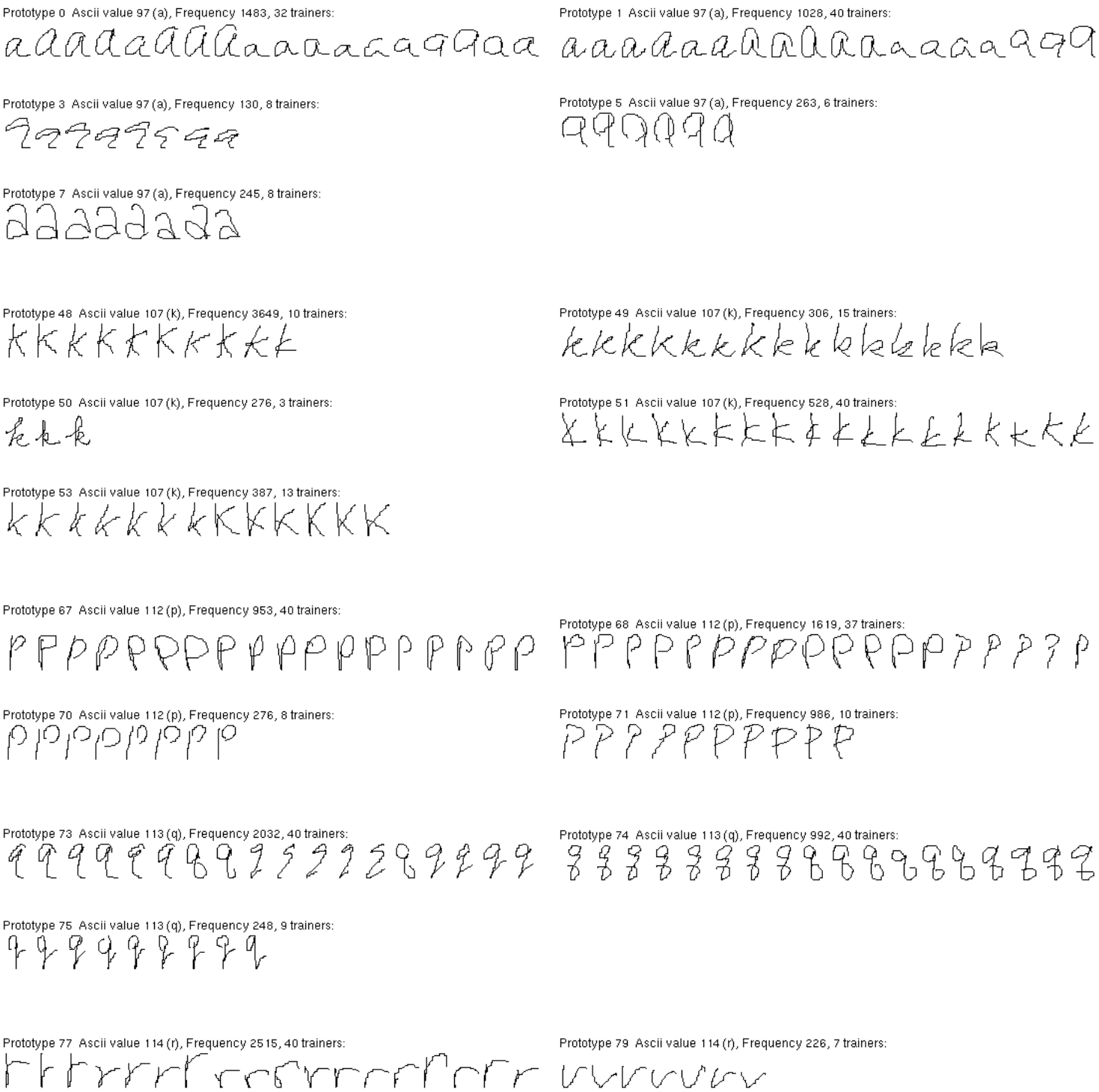

Figure 4. Subset of a letter recognizer with exemplars gathered by the prototypes

Confidence factor	right of (in this range)	total	ratio		cumulative % of test set		cumulative performance	
1.00:	0 of	0	0.0%	+=>	0.0%	0 of	0	0.0%
0.95:	4 of	4	100.0%	+=>	0.5%	4 of	4	100.0%
0.90:	2 of	2	100.0%	+=>	0.8%	6 of	6	100.0%
0.85:	18 of	18	100.0%	+=>	3.1%	24 of	24	100.0%
0.80:	17 of	17	100.0%	+=>	5.3%	41 of	41	100.0%
0.75:	20 of	20	100.0%	+=>	7.8%	61 of	61	100.0%
0.70:	30 of	30	100.0%	+=>	11.7%	91 of	91	100.0%
0.65:	39 of	39	100.0%	+=>	16.7%	130 of	130	100.0%
0.60:	63 of	63	100.0%	+=>	24.7%	193 of	193	100.0%
0.55:	56 of	56	100.0%	+=>	31.9%	249 of	249	100.0%
0.50:	54 of	54	100.0%	+=>	38.8%	303 of	303	100.0%
0.45:	***70 of***	***70***	***100.0%***	***+=>***	***47.8%***	***373 of***	***373***	***100.0%***
0.40:	54 of	56	96.4%	+=>	55.0%	427 of	429	99.5%
0.35:	66 of	66	100.0%	+=>	63.5%	493 of	495	99.6%
0.30:	51 of	51	100.0%	+=>	70.0%	544 of	546	99.6%
0.25:	***49 of***	***50***	***98.0%***	***+=>***	***76.4%***	***593 of***	***596***	***99.5%***
0.20:	47 of	50	94.0%	+=>	82.8%	640 of	646	99.1%
0.15:	34 of	35	97.1%	+=>	87.3%	674 of	681	99.0%
0.10:	***50 of***	***54***	***92.6%***	***+=>***	***94.2%***	***724 of***	***735***	***98.5%***
0.05:	36 of	45	80.0%	+=>	100.0%	760 of	780	97.4%

Table 5 Confidence versus Recognition Performance

The high correlation between confidence and success could be used to establish a threshold of acceptance in the recognition process. If the confidence falls below say 0.10, it may be appropriate to show a question mark and ask for re-entry. In the above case, this would result in a 5.8% rejection rate, and 98.5% accuracy on those accepted. Given SCL's success with the second guess, it would be easy to show both the first and second guesses in such cases (perhaps each half size, one above the other), and accept the letter tapped by the user.

The current recognition speed has assuaged earlier concerns about keeping up with the writer in recognition, altough real-time training remains a concern. (Letter recognition requires about 15 ms on the NeXTstation with a 25 MHz MC68040, and about 120 ms on a 80386 without FPU.) If it does prove necessary to speed up recognition, the trustworthiness of the confidence factor allows a hierarchy of "short-cut" recognizers to return a value if the confidence is high enough, and otherwise to work through a secondary recognizer on questionable cases.

4.8 SCL and gestures

Taysi [1992] used an early version of SCL to build a gesture-based graphical editor for a simplified prototype of the visual programming language, Hyperflow [Kimura 1992]. His program, Box and Arrow Editor (BAE) used 34 gestures for all aspects of the editor (opening and saving files, editing, aligning, asking for help, superimposing a grid on the editing space, etc.). He collected 5460 gestures from adults and children, using 3120 to train SCL and the remaining 2340 to test it. SCL showed 85% accuracy on the test data after training (forming from 80 to 147 prototypes).

As Taysi notes, the system's recognition performance fell into two clear categories. For several very closely related gestures (especially the "line," "dashed line," and "thick line" gestures which were very close in appearance), SCL was unsuccessful in consistently disambiguating the patterns (60 - 70% or worse recognition rate). For more complex gestures, SCL was able to achieve nearly 100% recognition. His work also shows the challenges associated with gathering consistent gesture data from children (including some under eight years old).

4.9 Freeing the recognizer from "boxed" input

Currently SCL recognizes input drawn within a box for each character. It would be especially desirable in math worksheets to eliminate this requirement. (The number of boxes may be construed as a hint about the number of digits in the answer.) One approach to unboxed recognition is to recognize a digit based only on its first stroke, and further, to recognize which strokes are the first of two strokes, and which are full digits. To test this possibility, SCL was allowed to train only on the first stroke of each scribble and was required to build different *categories* for one-stroke and two-stroke digits. This would enable it to learn from just the first stroke which digit is intended and whether or not another stroke will follow as part of this digit. (When this occurs, the following stroke is merely discarded). SCL was able to recognize *both* the digit value and the number of strokes for 94% of the test scribbles. This affords one quick path to an unboxed recognizer of modest performance. More experimentation in this area would likely yield improvements to this performance, since these results came from earlier versions of SCL achieving only 98% recognition with boxed input.

5 Conclusions

The experiments (99.6% for digits and 96.6% for letters, and comparable results for a subset of the gestures) illustrate the success of SCL in negotiating the stability/plasticity dilemma. SCL adapts to new environments without losing the recognition capability obtained in the old environment. While cautiously acknowledging its specialized task domain (recognizing members of a small alphabet each written within a box), we feel that SCL demonstrates recognition performance that approaches that of humans in both speed and success. The SCL architecture gathers similar exemplars for training. The correlation of confidence to success permits the assignment of an accuracy threshold (say 99.5%). The design also shows some promise as a recognizer of unsegmented digits, which will be especially helpful for worksheets in mathematics.

The mixed results with gestures (many drawn by children) point to two areas demanding further investigation and improvement. It is not clear how SCL scales up to recognition tasks involving more categories (e.g. gestures, or upper and lower-case letters with punctuation). Since SCL's principal application involves use by children, further testing is required with younger writers. Finally, SCL training times have decreased, but it is not yet clear that it can meet the constraint of constant, real-time learning in a manner transparent to the user.

6 Future directions

Currently SCL builds various prototypes which differ by the weights applied to the input feature vector. We are exploring prototype collections which differ by the input vectors as well. For example, one prototype unit might consider energy-based features and another will consider grid features. We expect to further pursue the promise of unboxed input for digit recognition.

We also expect to expand SCL to build recognizers which differ by network architecture. One recognizer may include no hidden units, another five hidden units, and a third six layers of five hidden units used recurrently. In fact SCL has already been tested with simple recurrent nets modelled after Servan-Schreiber et al. [1988]. These diverse collections of recognizers will require careful arbitration to select the final vote on each pattern. We are currently investigating the Weighted Majority Algorithm of Littlestone and Warmuth [1991] for this purpose.

The high correlation between confidence and success suggests experiments with a hierarchy of progressively more complex recognizers. The confidence could be used to assess the success of a suggested recognition early in the processing, that is, higher in the tree of recognizers. If the confidence exceeds a given threshold, the category name is returned. Otherwise the recognition descends the tree (possibly using specialized discriminatory features to resolve the ambiguity). This could also exploit another yet-unused feature of SCL: its success with second or third guesses. Overall, this suite of capabilities could lead to a robust Adaptable User Interface capable of learning the idiosyncrasies of each user.

We will continue to study SCL in recognition tasks where on-line constant learning by the system is performed. We will continue to test SCL in broader areas of alphanumeric and gesture recognition. So far SCL has been limited to tests involving about 75 subjects. We plan to investigate SCL with on-line training of unrecognized characters generated by a wider circle of writers (especially more children). Broader gesture-based applications will require more sophisticated gesture sets. We expect that users of pen computers will constantly enlarge and modify the gesture vocabulary. SCL's adaptability leads to natural and desirable collaboration between the user and the system. This may be the seed of Adaptable User Interfaces (AUIs) in future pen-based computers.

7 Acknowledgement

The Kumon Machine Project is jointly funded by the Kumon Institute of Education and Osaka Gas Information Systems. Their continuing support made this research possible and is gratefully acknowledged.

8 References

Aleksander, I. and H. Morton (1990). An introduction to neural computing. Chapman and Hall.

Baran, N. (1992). The Outlook for Pen Computing. Byte. September, 1992. 159-164.

Becker, S. and Y. le Cun (1988). Improving the convergence of back-propagation learning with second order methods in Touretzky, D. G. Hinton and T. Sejnowski (eds.) Proceedings of the 1988 Connectionist Models Summer School (CMU, June 17-26, 1988). San Mateo, California: Morgan Kaufmann Publishers. 29-37.

Carpenter, G. A. and Grossberg, S. (1987). ART2: Self-Organizing of Stable Category Recognition Codes for Analog Input Patterns. Applied Optics, 4919-4930.

Carpenter, G. A. and Grossberg, S. (1988). The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network. Computer 21:3, 77-88.

Carr, R. and D. Shafer (1991). The Power of PenPoint. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc.

Fahlman, S. E. (1988). Faster learning variations on backpropagation: An empirical study. in (eds.) Touretzsky, D., G. Hinton, and T. Sejnowski. Proceedings of the 1988 Connectionist Models Summer School (CMU, June 17-26, 1988). San Mateo, California: Morgan Kaufmann, 1988. 38-51.

Fukushima, K. (1988). Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition. Neural Networks, Vol 1., 119-130.

Fuller, T (1987). Microcomputers in Production and Inventory Management. Homewood, Illinois: Dow Jones-Irwin.

Fuller, T. (1991). The Kumon Approach to Learning Mathematics: An Educator's Perspective. Technical Report WUCS 91-49. Washington University in St. Louis.

Fuller, T. (1992). Energy-related Feature Abstraction for Handwritten Digit Recognition. Proceedings of the Fourth Midwest Artificial Intelligence and Cognitive Science Conference, May 1992, 71-76.

Gibbs, M. (1993). Handwriting Recognition: A Comprehensive Comparison. Pen, March/April 1993, 31-35.

GO Corporation (1991). PenPoint API Reference. Foster City, California: GO Corporation.

Grossberg, S. (1986). Competitive Learning: From Interactive Activation to Adaptive Resonance. Cognitive Science 11, 23-63.

Grothe, R. (1991). Private communication with the authors, October, 1991.

Herbst, N. M. and C. N. Liu (1982). Automatic Signature Verification, in Suen, C. Y. and R. De Mori (eds.) Computer Analysis and Perception, Volume 1: Visual Signals. Boca Raton, Florida: CRC Press, Inc. 83-105.

Jerney, J. (1993). The (General) Magical Mystery Tour. WPDO News, Worldwide PenPoint Developers Organization, Vol. 2, No. 2 (March, 1993).

Kalman, B. L. and Kwasny, S. C. (1991). A Superior Error Function for Training Neural Nets. Proceedings of the International Joint Conference on Neural Networks, Seattle, Washington, 1991, Vol. 2, 49-52.

Kalman, B. L. and Kwasny, S. C. (1992). Why Tanh: Choosing a Sigmoidal Function. Proceedings of International Joint Conference on Neural Networks, Baltimore, MD, June 1992.

Kimura, T.D. (1990). An Algebraic Proof for Backpropagation in Acyclic Neural Networks. Proceedings of International Joint Conference on Neural Networks 1990, San Diego, CA, June 1990, 554-557.

Kimura, T. D. (1992). Hyperflow: A Visual Programming Language for Pen Computers. Proceedings of the 1992 IEEE Workshop on Visual Languages, September 15-18, 1992, Seattle, Washington. 125-132.

Kohonen, Teuvo (1982). Self-Organized Formation of Topologically Correct Maps. Biological Cybernetics, Vol 43, 59-69.

Kohonen, Teuvo (1988). An Introduction to Neural Computing. Neural Networks, Vol.1, 3-16.

Kohonen, Teuvo (1990). The Self-Organizing Map. Proceedings of the IEEE, 78:9, 1464-1478.

Lin, W. C. (1982). Computer Processing of Hand-Drawn Sketches and Diagrams. in Suen, C. Y. and R. De Mori (eds.) Computer Analysis and Perception, Volume 1: Visual Signals. Boca Raton, Florida: CRC Press, Inc. 107-149.

Nellis, J. and T. J. Stonham (1992). A fully integrated hand- printed character recognition system using artificial neural networks. Second IEE Conference of Artificial Neural Networks, Bournemouth, U.K., IEE Publication No. 349, 219-224.

Reilly, D. L., L. N. Cooper, and C. Elbaum (1982). A Neural Model for Category Learning. Biological Cybernetics Vol. 45, 35-41.

Schwartz, John (1992). The Revolution That Wasn't. Newsweek: October 19, 1992, 51.

Servan-Schreiber, David, Axel Cleeremans, and James L. McClelland (1988). Encoding Sequential Structure in Simple Recurrent Networks. Carnegie Mellon University, November, 1988.

Shynk, J. J. and S. Roy (1988). The LMS Algorithm with Momentum Updating. IEEE Symposium on Circuits and Systems, Espoo, Finland. New York: IEEE. 2651-2654.

Sorheim, E. (1991). ART2/BP Architecture for Adaptive Estimation of Dynamic Processes. Advances in Neural Information Processing Systems, Vol. 3 ed. by D. Touretzky, 169-75.

Tappert, Charles C., Ching Y. Suen and Toru Wakahara. "The State of the Art in On-Line Handwriting Recognition." IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 8 (August, 1990), 787-808.

Taysi, B. M. (1992). Gesture System for a Graph Editor. Master's Thesis, Washington University in St. Louis, August, 1992.

Thomas, Jr. G. B (1972). Calculus and Analytic Geometry. Reading, Massachusetts: Addison-Wesley Publishing Co., Inc. {Cf. Thomas 1972, 289-293 for energy integration calulation.}

Wagoner, J. (1993). Second Annual Reader's Choice Award Winners. Pen, March/April 1993, 11-21.

Wang, C. (1992). A Fuzzy Approach to Handwritten Digit Recognition for Pen-based Computers. Master's Thesis, Washington University in St. Louis, December, 1992.

Wang, C., T. Kimura, and T. Fuller (1992). Supervised Competitive Learning, Part II: SCL with Fuzzy Logic, in Dagli, C, et al. (ed.) Intelligent Engineering through Artificial Neural Networks, Volume 2. New York: ASME Press. 351-6.

Weiss, S. M. and C. A. Kulikowski (1991). Computer Systems That Learn. Morgan Kaufmann Publishers, Inc.