Computer Science Department Technical Report
Machine Perception Laboratory
University of California
Los Angeles, CA 90024-1596

VISUAL RECOGNITION OF SCRIPT CHARACTERS NEURAL NETWORK ARCHITECTURES

Josef Skrzypek
Jeff Hoffman

December 1989
CSD-890075

MPL

Machine
Perception
Lab

UCLA
Computer Science
Department

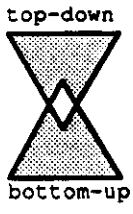
Visual Recognition of Script Characters Neural Network Architectures

Josef Skrzypek
Jeff Hoffman

TR 89-10 Nov 15 1989

MPL

Machine
Perception
Lab



Visual Recognition of Script Characters Neural Network Architectures.

Josef Skrzypek
Jeff Hoffman
Machine Perception Laboratory
Computer Science Department
University of California
Los Angeles, CA 90024

Abstract

Visual recognition of script characters is introduced in the context of the neural network paradigm and the results of applying one specific neural architecture are analyzed. First, computer classification of script characters is partitioned into preprocessing, recognition and postprocessing techniques which are briefly reviewed in terms of suitability for implementation as neural net architectures. The second part of the paper introduces one example of neural net solution to script recognition. Handwriting is assumed to be defined as concatenation of ballistic hand movements where characters can be represented as functions of position and velocity. We adapt a hypothesized model of human script generation where characters can be composed from a limited number of basic strokes which are learned using visual and positional feedback. The neural representation of these characters is used for assembling motor program during writing and it can be used for their visual recognition during reading. A modified, three-layer "backpropagation" algorithm is used to learn features of each single character that are independent of writing style. Preliminary results suggest 80% recognition rate.

1 INTRODUCTION

1.1 Scope of the Paper

Practical benefits of automating character recognition process in application to man-computer interactions are well understood and a number of algorithms exists to verify signatures [1], identify writers, recognize printed characters [2,3], numerals [4,5,6], hand-printed characters [3], cursive script [7], English and Kanji characters, etc. However, a system that could input hand written documents for further processing including understanding of its content, has not been realized as yet and engineering attempts based on integration of available techniques seem to result in solutions that can not operate in real time. In this paper we are only interested in visual recognition of the handwritten text regardless of writing style, for the purpose of further cognitive analysis of its content. This differs from such problems as verification of signature or identification of a writer (see [1]) where special purpose devices can be instrumented to extract such attributes of handwriting as presure and acceleration. Because of the wide variety of fonts and hand written styles and because of the noise inherent in any electronic system, the simplest solution of directly matching data against a stored template becomes a difficult problem and perhaps, analytically intractable.

The purpose of this paper is to introduce the problem of visual recognition of script in the context of neural network paradigm. We first introduce a background material by reviewing some existing techniques and assumption underlying character recognition. These are analyzed in terms of applicability to script recognition and potential advantages of implementing them in "neural" architectures. Most of the existing algorithms for recognition can be

classified into distinct methodologies by identifying them with three separate functions; *preprocessing, recognition, and postprocessing.* Within each function we try to identify fundamentally different methodologies and we relate them to the possible neural network solutions. In our view many preprocessing techniques need not be specific to character recognition therefore, it is not clear that specilized "neural" architectures for preprocessing script will be of interest to vision community. In fact, it is conceivable that "neural" nets developed for "general" vision might make it unnecessary to, for example, skeletonize the original character image.

Next, we review algorithms for recognition. It is our believe that here some specific solution must emerge that are not general to vision but only apply to character recognition. This is also the area were we introduce our "neural" architecture for script recognition. Our solution is based on assumptions depicted in Fugure 1. and discussed in details in the second part of the paper. Briefly, each script character is composed of a few basic strokes of ballistic hand movements that can be modeled as an oscillation of a two-spring system [8]. During first attempts to write characters, style-invariant features are extracted by learning, under both visual and positional/tactile feedback. These features are components of a neural representation that allows to express characters in their minimal configuration of basic strokes. This representation is accessible during the visual recognition and it is also used when generating motor program for execution of handwriting. The adult script characters might appear to differ from this representation because of changes in hand mechanics and other psychological variants of a writer at the given time. To simplify the problem we focus on recognition only, which presupposes the existence of apriori acquired "templates" or memories of each character; we are not attempting to optimize or model the human learning process. Consistent with this model, we decompose script characters into horizontal and vertical frequency waveforms, from which the style-invariant features are extracted and fed to the neural network.

In this paper we are focusing only on the recognition of single characters, but it should be clear that when information about them is only partially available higher level cognitive processes might be invoked to infer missing data before recognition can take place. These problems are partially addressed by the postprocessing functions which are briefly reviewed in the first part of the paper for completeness.

1.2 Neural net algorithms for character recognition.

Serial techniques require autonomous functions that rely only on input from previous stages of computation. For perceptual tasks, a serial approach does not seem adequate to explain the levels of performance seen in human subjects. It has been suggested [9,10,11] that a parallel set of interacting processes is necessary to achieve high rates of recognition over different styles of input (i.e. accents and speed of spoken language, varied writing styles and sizes). .sp There are many reasons why "neural" networks could be useful in perceptual tasks as well as other problems where input data is naturally organized in a massively parallel representations [11,12,13]. "Neural" architectures have a potential to speed up computation while assuring fault tolerant performance. Usually a correct decision made at the output of a neural net depends on pattern of activity in a subset of all neuron. Thus representation does not have to be the exact connectivity between neurons but only an approximate pattern of activity related to the topology of the network. Neural systems exhibit inordinate amount of plasticity. This guarantees graceful degradation in case of erroneous decisions or novel stimuli. Consequently the network is always capable of some response that can always be made to converge on the desired output. For many real-world problems such as script recognition, it is extremely difficult to enumerate a priori all possible rules that apply to specific case. Equally impossible is a task of completely specifying all contextual conditions that might affect such rules. Hence, the precise analysis of all numerical decisions that might have to be made are not feasible and an approximate response of a neural net is better than no response at all.

Neural networks appear to be better suited for realization of very complex systems. For example, information in "analog" neurons expressed as a temporal variation of signal amplitude makes interfaces between modules easier than multidimensional symbols such as words. Thus "connectionist" paradigm allows integration to be based physically on signals going through the links between nodes, while in symbol processing systems a complex communication interfaces are needed. Nevertheless, there is no clear formal way at the present time to develop separate modules and have a priori specification of link weights in order to connect the modules into a complex system. In other words, a nontrivial problem which has not been addressed satisfactorily in traditional AI, is the

questions of programming systems to perform perceptual or cognitive tasks. Here, the "neural" net paradigm offers interactive solutions that are based on primitive forms of supervised learning or some form of self-organization of a neural network in the process of learning. All of these advantages make "neural" net techniques ideally suited for pattern classification, a key component of any recognition system [11,14].

1.3 Performance Criteria

In order to propose improvements in character recognition techniques one must be able to evaluate the performance of existing algorithms. Since, we lack standards in this area, individual investigators are not always consistent with evaluation of their own algorithms. Below we attempt to summarize several criteria which could be used to this purpose.

RECOGNITION RATE criteria may include correct or incorrect recognition rate, and rejection rate. A rejected character can be better than an incorrectly recognized character since it can be flagged by the system for further review by the user. *CONSTANCY* under rotation, translation and scaling can be considered as another criterion of performance. Systems that are not invariant to these conditions must constrain their input to only certain formats. Interestingly, human perceptual system is not rotationally invariant at the preattentive level, and therefore, humans must consciously rotate the image in order to achieve recognition [15,16]. Constancy problems are of general interest to vision and are not specific to recognition of script. Some neural net solutions to these problems are discussed in the later sections. System must display sufficient *NOISE IMMUNITY* so that noisy images of characters are recognizable. This parameter can be implicitly expressed in the recognition rate of an algorithm that has been tested on noisy character images. Humans excel in this parameter, and no recognition algorithm has been devised that comes close to human capabilities unless character shape is severely constrained. Some algorithms are only usable for Chinese characters, some for numerals and some only for machine printed characters. Therefore, a *GENERALITY* of an algorithm is a very important parameter. There are certain format 'characters' which should be handled in a recognition algorithm. They include, an underline, a dash across "t", dot over "i" + "j", and other special emphasis charcters. Current algorithms do not take these into account, because it would involve integrating higher level information (e.g. recognizing the underline in the whole word) with lower level algorithms.

1.4 Parameters versus functions.

Handwritten characters can be represented in terms of parameters or functions or both. Since in our approach we are interested only in visual input, a limited number of parameters and functions of handwritten text will apply. Basic function is a position of a character trace in x and y at different points in time. However, since the time information underlying character generation is lost in a static image this function is difficult to compute. The data are successive points expressed in x,y coordinates, obtained from sampling script characters at some regular intervals. Form this data we can build higher level features that represent strokes. At the higher level of processing, recognition may be aided by symmetry or other characteristic points that remain invariant regardless of the writing style. Handwriting velocity, computed as a first derivative of position can add new information about the dynamics of the hand in the process of writing. Additional information can be obtained from acceleration which can be computed from the second derivative of position. Other functions such as pressure and force can not be extracted from the visual input and are not considered here (but see [17]). In continuous writing, additional information can be obtained from considering spaces between characters when the hand is in motion but the pen is in the air. It is conceivable that in human perception these invisible "traces" are processed by the so called "illusory contour" neurons [18] as an additional aid in script recognition.

The parameters that can be extracted directly from the script by visual means include local curvature, starting direction and peaks with respect to some local reference. Some of the more global parameters that apply more to text than to characters are number of breaks when the pen is lifted, maximal and minimal excursions for each continuous segment of characters and perhaps proportions. An interesting feature set has been proposed by Sabourin and Plamondon [19], where they computed an angular orientation of the signature intensity gradient. This

was assumed to reflect the stroke orientation. Another approach to features extraction, derived from traditional techniques used in graphoanalyzis is to divide the text into upper, middle and lower zones and to performed analysis of subimages referenced on initial and end strokes [20]. A variations on this approach is to divide the image into vertical and horizontal segments and then analyze their content [21].

The approaches to handwriting recognition based on extraction of parameters versus computing functions can not be qualitatively compared at the present time. Clearly it seems that parametric description would offer advantages in storage requirement, but this may be offset by increased computational expense in composing and recognizing characters. Many of these problem are basic to pattern recognition in general and it is not clear that their solution will emerge from analysis of handwritten text. These problems include question as to which feature set is best and offers most discriminant power? How to determine and control thresholds for binary images and for making decision based on processed data? How to best combine parametric and functional approaches to recognition and many others. Our approach to character recognition as detailed in later sections combines both parametric and function based techniques. We use a position function to rerepresent the character in the form of X and Y oscillation waveforms from which we extract parameters related to basic strokes.

2 PREPROCESSING STAGES.

Most character recognition algorithms use many discrete preprocessing stages which prepare, enhance and extract relevant information. These include correction for nonuniformity of acquisition devices, localization of a character in the image, character - background segmentation, filtering, thresholding and perhaps data reduction by signal to noise enhancement, etc. Preprocessing operations can be classified according to at least three objectives all of which have the common goal of converting the data into a form useful for the final recognition stage. One objective is to enhance and/or blurr certain parameters. For instance, noise might be removed while character borders might be enhanced by using thresholding and filtering. Another objective is to translate the low-level spatial data into a higher level symbolic notation; for example, extracting lines, curves, loops, and strokes [22,7] from the original image and linking them into outlines of characters. Examples of these operations include feature extractions and segmentation. Finally, an objective might be to add a performance criteria that is missing from the recognition algorithm. For instance, a particular algorithm might not be rotationally invariant, but this invariancy can be established by rotating the character to a standard orientation in the preprocessing stage.

Many problems addressed by preprocessing stages remain only partially solved. For example, in case of a very simple script, the localization problem can be solved by using variable size window operator [21] but it is in general a difficult problem for connected text. Thresholding techniques for enhancing characters are in general insufficient for noisy background images with partially missing traces, uneven trace width etc. Intensity discontinuity operator such as Sobel or Difference of Gaussian can be applied to binary version of the original image to extract characters. However, setting of binary threshold remains in general an unsolved problem [21].

2.1 Input Image

Acquisition of script characters from static, grey-level images is a more desirable method because of the need to process previously written documents. Here, documents are read by scanning them optically. In general, the output of this device is a number recording light reflected from the paper at each point. This 'grey level image' is usually processed to yield a 'binary image' by thresholding as described in the next section. Visual means of acquiring data implies problems not present in the graphics tablet. First of all, some spatial information may be degraded during transformation from a gray level image to a binary image. Secondly, algorithms for segmenting the input into separate characters must be used since sloppy written characters might be overlapping or touching; pressure-based system can segment the characters by noting the temporal sequence of the character strokes [23]. With visual acquisition, the user doesn't have control of the format of the input. Finally, although it is not too difficult to acquire an image of a character optically, most of the information about the dynamic process of script generation is degraded or missing. It is conceivable that humans can recover some of this information from the

saccadic eye movements during scanning [24], but these ideas have not been incorporated into machine recognition as yet.

2.2 Noise reduction by thresholding and filtering

An example of a preprocessing algorithms is a simple thresholding which converts, a grey level image into binary values. The intensity of each pixel in the grey level image is compared to a threshold value. If the intensity is larger than the threshold then a zero is substituted for the intensity (denoting no image point), otherwise a one is substituted for the intensity. This results in a binary image. Most of the noise is filtered out by this process since noise usually yields intensity values small in comparison to the actual image intensity. Moreover, the amount of information in the stored image is reduced, yielding a saving in storage space and a simpler recognition algorithm. The disadvantage of the thresholding algorithm is that it marks many valid points as noise points and vice versa. One simple solution is to vary the threshold of a point depending on its neighboring pixels. For example, if all the neighboring points are part of the character (or are likely to be part of the character), the threshold can be lowered. Since noise tends to come in single pixel spurts while the character is many pixels wide, this will reduce the likelihood of a bad decision. Other possible solutions include averaging over a neighborhood of points and varying the threshold depending on local measures of business [25].

Thresholding operations in general have been of considerable interest within neural net paradigm. One generalization is that lateral interactions within neural layers are amenable to various space and temporal adaptive threshold algorithms as for example automatic gain control in early stages of visual processing [26]. The basic principle here is that threshold can be automatically adjusted depending on the value of neighboring units. The adjustment can be a function of multiple spatial scales. The complete process can be viewed as automatic shifting of the operating characteristic for a " neuron" along the domain of the feature. Basic to this operation is concentric organization of the center surround receptive fields. In some sense this can also be viewed as predictive coding where the surround predicts what the center should be and the difference between center and surround encodes the error in prediction. "Neural" networks are very appealing for thresholding based on above principles, because it is very easy to implement a concurrent filtering operation using the same "neurons".

A filtering step is usually added to the algorithm to reduce the noise. The type of filter used ultimately depends on the noise statistics of an image and of the system. Undesired result of low-pass filtering is blurring effect on character boundary (this is critical in some algorithms). In addition, filters implemented as a convolution, are computationally expensive on sequential machines. There are many solutions to these problems in neural net paradigm. Analog neural nets with build in lateral inhibition can perform convolution-like operation as a relaxation with convergence being practically instantaneous, while at the same time enhancing any discontinuities [27,26]. Many other filtering operation can be performed in this mode. In general, interactions between layers of neurons can be considered as two-dimensional mappings between arrays of instantaneous values. These can include space invariant image processing that allows restoration of motion-blurred images or removal of other translational and rotational distortion [28].

2.3 Trace detection: region growing and border following.

Region growing is the process of defining a small region and then expanding it (i.e. making it grow) by adding as many neighboring pixels as possible such that they satisfy a certain criteria. For instance, the algorithm might search for four pixels in a square which are larger than a certain threshold, thereby finding a region. Then, pixels are added to the region by comparing each pixel that is adjacent to the region to a second threshold. The output of this algorithm is an image which is decomposed into many different regions. In general, the algorithm tries to find two regions: the character region and background region. The location of the character region can then be fed to a recognition algorithm. This might replace the thresholding stage when thresholding isn't robust enough against noise. Boundary detection by region growing seem naturally suited for neural implementation. Lateral comparison of neighboring "neuron" activities can spread rapidly to determine the uniformity of a region [29].

The dual to region growing is border following. There are many different algorithms to find and follow a border (see [30,31]). The simplest is the turtle algorithm which operates on a binary image. The border is found by locating a transition from white to dark at which point a 'turtle' follows the border and marks it. This algorithm usually creates closed borders which can be a very useful property in some recognition schemes. More sophisticated algorithms operating on a grey level image use filters (usually directional) to follow the boundary of a character. For example the Sobel operator sums up pixels in a window (using weights) to determine if there is an edge in a certain direction [32]. One disadvantage of this method is that it requires at least two passes - one with a filter sensitive in the x direction and the other sensitive to the y direction. This of course suggests the use of the Laplacian of the Gaussian which can be implemented as resistive neural networks [27]. Another approach to determining discontinuities has been recently developed by Gleeson and Skrzypek [33] which differs from all other edge detection algorithms based on local differentiation. Here, we make no assumptions about the location or the type of discontinuities but we take advantage of the fact that away from discontinuities, pixel values are highly correlated. Image approximation with a nearly second order function fails at the contrast boundaries. Statistics generated in the process of improving approximation can be used to predict the occurrence of a boundary. A higher level approach for organizing neural nets to infer curves from images is introduced by Zucker [34]. The tangent field is first computed by labeling relaxation from discrete tangent measurements at each contrast discontinuity. This is followed by dynamically fitting splines to the tangent field. The curves can be recovered to subpixel accuracy, making skeletonization unnecessary. The method is motivated by end-stopped receptive fields and neural structures at the infero-temporal cortex level.

2.4 Performance improvement by normalization and skeletonization.

Because size invariance is desirable, the character height and width must be normalized if the length of line segments is used in the recognition algorithm. This can be done by finding the character height and width using either the border following or region growing algorithms and then normalizing to a chosen height and/or width. Some recognition algorithms require the character to be oriented correctly. Current approaches make a character positioning dependent on the needs of the recognition algorithm. There are already some neural network based algorithms that give rotationally invariant recognition of a simple objects [35]. Neural net solutions to size invariancy are based on findings that early visual processing in biological systems, follows space-invariant mapping from the retina to the cortex which can be captured by Log-polar function [36]. Combination of this mapping with a saccadic response directed by the focus of attention results in size and translation invariant mappings. The foveation response recenters the target while the log-polar function gives size invariance. A combination of "adaptive resonance network" [29] with a neural net that mimicks the log-polar function and foveal response was recently tested on noisy object [66]. The system gave rotation, position and scale invariant recognition by classifying correctly all object some of which were partially obscured and embed in a 50% noisy background.

A character image is sometimes skeletonized. This is a process in which the character is thinned to its basic shape [30,31]. It is important in this process to preserve the shape and connectivity of the character. A typical thinning algorithm, the medial axis transformation has the properties that the connectivity of the skeleton is preserved, the original image can be exactly reconstructed, and the geometry of the skeleton is invariant under rotation [30]. The ('minor') problem with this algorithm is that it is theoretically impossible to implement it exactly on a digital computer [37]. Another skeletonization technique is Hilditches' algorithm [38] which involves many passes over the image. Each pass deletes image points that have neighboring image points. The skeleton resulting from this algorithm however, depends on the orientation of the character. Finally another variation is line thinning by line following [39,31].

In essence, skeletonization algorithms examine and manipulate pixels within some local (3x3) window. With larger windows, (5x5) it is possible to reduce the noise sensitivity and to better preserve straight edges. Since, larger kernels require more computing, they are good candidates for implementations in neural architectures as some forms of convolutions [4,40].

2.5 Data abstraction by segmentation and feature extraction.

Some aspects of segmentation, the dividing of the image into separate characters, may be performed in the preprocessing stage. This is easily done with machine printed characters and sometimes with hand printed characters. However, segmentation of joined script characters requires interaction between the character recognition scheme and the feature extraction scheme. In general, representation of the context as available from the recognition level, is a very difficult problem in vision. Various forms of relaxation labeling schemes have been used to iteratively remove labels that are not congruent with other local labels and with global (context) labels. Unfortunately there are no general methods to decide what contextual labels are. One possibility is that a backpropagation neural net could encode and learn global descriptors of the context without the recourse to relaxation [41].

Feature extraction is the most critical issue in character recognition. Some recognition algorithms try to match the different lines and curves of a character against a template. In that case, the preprocessing stage, segments the character into a set of features (e.g. a set of lines and curves). Most of the previous algorithms use features selected subjectively by the designer according to intuitive hints about significance of topology and geometry in shape perception. Neural network techniques show promise in their ability to independently extract important features of the input data, useful in classifying object types. This approach has the advantage of adaptively choosing the appropriate discriminatory features depending on writing style. A disadvantage is the inability to perform this function in real time because current neural nets can require long training periods.

3 RECOGNITION ALGORITHMS

In order to achieve recognition some comparison must be performed between functions or parameters extracted from the trace of a character and the previously stored memory. This comparison preceeds a decision about identification of a character. It is difficult if not impossible to apply a simple correlation procedure based on fixed number of temporal samples because with constant sampling the same characters can be stretched out or compressed. Additional random variations can take place due to the biomechanical problems or even mood or hesitation of a writer. Some of these problem can be in part compensated for by using adaptive sampling techniques [42]. All other problems caused by translation, rotation and scaling have been traditionally dealt with by using some normalization procedure [2,19,23,43].

3.1 Algorithm Classifications.

In general, most recognition algorithms involve comparison of a character to a template followed by a decision reflecting quality of the match. Characters consist of features that are useful in distinguishing between classes of characters. Input patterns of features represent points in some multidimensional space. A pattern classifier, partitions the multidimensional input space into some decision regions that could be used to classify the input. We can categorize all algorithms depending on whether the template is a direct replica of the input data or some transform of it.

Direct recognition algorithms recognize characters by comparing their topological features with templates where the comparison can be based on geometrical, structural or statistical aspects of a character. For example, a statistical approach may use number of horizontal and vertical intensity changes [44]. Geometrical aspects may be based on shape parameters such as the minima and maxima of character segments [45]. Finally, character structures such as loops can be used to describe and recognize script letters as a concatenation of loops and arcs with a relatively high degree of accuracy [22,7]. In general, direct recognition methods are computationally and algorithmically simpler than the transform based techniques.

Certain aspects of direct recognition techniques map naturally to neural nets and are consistent with general ideas about the structure and function of perceptual system. Feature detectors [46,47] and their aggregation according to Gestalt laws are such examples [48]. A more difficult problem for neural nets is rapid development of

a template. One solution to this problem is ART network developed by Carpenter and Grossberg [29]. Another possible model for rapid learning has been recently introduced by Lynch et.al., [49]. The problems of making comparison and reaching a decision about the match has been also addressed within a neural network paradigm but not specifically in application to cursive script recognition [50,35,51,4,52]

Transform methods rerepresent the character image into a separate domain (e.g. the Fourier domain) and compare the character to its template in that domain. Fourier shape descriptors could be designed as a function of the first few Fourier coefficients of the character boundary [44]. Template matching in the frequency domain seem consistent with the idea that neural structures of the perceptual system could process spatial frequencies [53] instead of features. However, neural net architecture for transforming spatial data from characters into frequency domain have not been developed as yet.

A further classification of algorithms can be made based on the kind of characters they are designed to read. For instance, there are special algorithms for machine printed characters and numerals, for handprinted characters, and for script. Also, the language and therefore the alphabet, is an important consideration when designing an algorithm. This can be seen from the many algorithms designed specifically for the Chinese alphabet (which consists of thousands of characters). However, there are more distinct feature groups and context may not play as large a role as in roman script recognition. These algorithms are very different from their Roman-Script counterparts in that they employ very sophisticated data structures and tree algorithms in order to effect the recognition [54].

3.2 General Techniques.

Many algorithms employ a unique method to perform the recognition. However, some general techniques are used in various forms by most of the algorithms. The most general group of classifiers which incorporates recent advances in neural networks, is based on building decision regions bounded by hyperplanes computed from the input space. These include, perceptrons [55], multilayer - backpropagation nets [56], Boltzman machines [57] and decision tree classifiers [58]. A different category of classifiers, which are most plausible from biological perspective, build decision region bounded by the properties of neuronal functions determined by the overlapping receptive fields. Examples of such classifiers are Cerebellar Model Articulation Controller (CMAC) [59,60] and a multi-map classifier [61,62]. A subgroup of classifiers within this category performs classification by identifying exemplars that are nearest to the input. These include, k-nearest neighbor classifier [63], learning vector classifier [62] and adaptive resonance theory classifier (ART) [29]. The differences between various classifiers are in training time and memory requirements. For example, Boltzman machine and back-propagation classifiers require very long training sessions, while the ART and k-nearest neighbor classifiers require a lot memory for classification. In this respect, decision trees seem to be optimal.

Decision trees classifiers, used by many of the newer algorithms, allow complex decisions to be made by a number of simple and local decisions. In a decision tree, a classifier is used to decide which group of characters the image belongs to. That group is then further subdivided by a different classifier. This process continues indefinitely until the group is composed of a single character. For example, Gu, et. al., [54] used Walsh coefficients as the classifiers to a decision tree in order to recognize 3000 noisy Chinese characters with a 99.5% recognition rate. The computational demand of a decision tree is low and they can be implemented using fine grain parallelism. They can use signal or symbols as an input. However their training is not biologically motivated, and requires access to all training examples simultaneously.

There are a number of different problems that must be dealt with when designing the decision rules for trees. For instance, a tree might overlap [64]; an initial grouping might be performed at the main node by dividing according to which characters contain horizontal lines and which contain vertical lines. In that case, 'H' belongs to both groups. In the Chinese character set, which is very large, this can result in a dramatic increase of a tree size. This problem can be avoided by redesigning the classifier so that there are no overlapping regions; a very difficult task with a large character set.

Another tree design question is whether to make the decisions completely binding. For instance, a decision

might be made for a membership in a certain character group. However, other classifiers might disagree with this classification and determine that the character belongs to a different group. Certain tree structures allow a branching back to the group whereas other don't. A difficult problem with this approach is who finally makes the decision? This might be interesting application area for neural net paradigm.

Concatenated Classifiers, often used in conjunction with decision trees attempt to increase accuracy by using more than one type of classifier. Some classifiers are good at differentiating certain characters but not others. In that case, one-type of a classifier can be used to divide the characters into groups and individual characters. Then, another type of classifier is used to make the final decision. For example, Fourier descriptors used for the first grouping couldn't differentiate between rotationally symmetric numerals (for instance: a 2 and a 5) [44]. Therefore, a second descriptor- average number of intensity crossings in x and y directions was used to differentiate those numerals. Similarly, Shridhar and Badreldin [65] were able to recognize handwritten numerals with a 98.8% accuracy with a first subdivision using the number of horizontal and vertical intensity crossings and a second subdivision using the approximate location of the peak of the character contour. Considering that recognition of numerals involves only 10 characters, as opposed to 26 characters in the alphabet the problem is not as complex.

Learning techniques are used in many algorithms designed to recognize hand written characters. Typically, a user will input many examples of a character. Using these examples, the algorithm can 'learn' what a character looks like. Usually this means that the algorithm is able to learn how much the data differs from the template and this information can subsequently be used during the matching process. This technique, which is usually used during an initial 'learning' stage, creates a system that is efficient for a single user or a small group of users. An alternate approach to learning used by [66], allows the algorithm to learn while performing the recognition. This is done by assuming that the algorithm makes a correct decision and then updating the stored statistics of the character with the statistics of the character just recognized. In order to de-emphasize wrong decisions, the algorithm can reduce allowed variance bounds as well as increase them.

Shape and Size Constraints based techniques are usually not acceptable from the users point of view. However, it is sometimes the only way to create an algorithm with a very high accuracy. Shingal and Suen [67] describe results from research they performed into the effect of different handwriting styles on recognition rates. One of their conclusions is that recognition rates depend significantly on writing styles and therefore, certain styles should be taught in order to increase accuracy of recognition.

Most of the traditional techniques are linear techniques. There is good reason to assume however, that the optimum technique is non-linear. This can be seen by noting that a perturbation of a character form will almost always involve a large number of pixels. Therefore, the location of the next pixel to be matched depends on the location of the pixel appearing before it. Using this logic, it might be possible to find some type of matching algorithm that would use the information from local pixels to modulate the matching process [68]. Other heuristics are possible, for instance, a pattern match in one part of the character is more important than no match in a different part of the character (since noise or perturbations could have caused the no match condition).

Neural Network Classifiers are the most recent developments in the field of handwritten character recognition. Neural network based feature extraction can select the most discriminating points among a set of characters. This information can be used by a second network to classify the characters, taking other factor (such as letter and word context) into account. This approach requires a large amount of training time, however after that initial investment, future adaptation would be minimal, and operation could be real-time.

All neural net classifiers applied to character recognition are based on learning, and the multilayer architectures appear to concatenate different functions. The best known neural-net classifiers that are trained under supervision using gradient-descent training methods are backpropagation algorithms [56,14,69]. In general the properties of a three-layer backprop can be described in term of operation that various layers can perform on n-feature space. The first layer partitions the input space into hyperplanes. The hidden layer nodes, combine the hyperplanes of the first layer into a product that represent a convex region of the feature space. Nodes could typically have a sigmoidal nonlinearity as a transfer function. Iterative training, allows to backpropagate the error between the desired and computed output and this is used to modify the weights. Training is considered completed when the error value reaches some predetermined level (for review see [70,13,56,14]). Despite the initial hope and the resounding

popularity, there is no evidence that backpropagation resembles biological processes [71]. The backpropagation has been used successfully in many application and all attempts to use neural nets in character recognition are based on this techniques. A disadvantage of backpropagation is long training period, which can be reduced and performance improved if the size of the net is tailored to the problem [69].

A simple neural net classifier based on back-propagation [50] uses grey level image where learning features are vectors of stroke's lengths. Piecewise linear, neurons organized in multiple layers of full connectivity after being trained on the representative sample of inputs can achieve recognition rates that are not as good as standard nearest neighbour classifier. Another example of neural network applied to recognition of handprinted, block characters, was shown to achieve recognition rates comparable to conventional nearest neighbour classifier [70]. The peak rates reached 94% with a network of 20 hidden units that learned graded-value activities in a specialized 13-segment, bar-mask, character sensor. This is comparable to Bayes [63] classifier which is theoretically the best performance one can expect. Unfortunately input sample in [70] was limited to one writer.

Le Cun et. al., [4,51] developed a hardware/software, neural-net system for recognition of handwritten isolated numerals obtained from postal zip codes. The system combines skeletonization, feature extraction and classification as one huge feed-forward network. The architecture consists of three hidden layers, besides the input and the output layers. Grey level images are used at the input and the system can correctly classify 94% of test examples after only 15 training sessions. The maximum rate of classification was 10 digit per second which could be improved to 30 digit per second after normalization of input images. The most interesting feature of this neural network is breaking hidden units into separate maps that combine only local sources of information. This is similar to our approach of clustering hidden nodes, which we found to improve performance by constraining learning to local features [69].

An example of a different approach to modelling handwritten character recognition is provided in a recent paper by Morasso [7]. This work investigates the use of self-organizing maps to model cursive script handwriting. Written words are broken down into strokes bounded by points of zero vertical velocity. Each stroke was then represented by a five point polygonal curve which was used as input to a so-called graphotopic map, similar in design to Kohonen's phonetic typewriter [62]. Through training (with 200 words or about 1600 characters), an organized map of curved segments was formed on a 15x15 matrix. The segments on the resulting map were organized by pen direction, pen pressure (up or down), stroke length and amount of curvature.

4 POSTPROCESSING

The preprocessing and recognition stages output a string of characters in binary format. In general, this string will contain errors. Errors can be classified as substitution errors, deletion errors, and addition errors depending on whether a character was substituted, added, or deleted. In most character recognition systems, substitution errors are the most common, and therefore, much of the literature attempts to solve that problem. In the script recognition systems, all three types of errors are possible due to the connectivity of the letters. Viterbi algorithm is applicable to correction of substitution errors. Additions and deletions have been dealt with specialized dictionaries to search for such errors [72].

In a typical error correction scheme, the recognition algorithm outputs error probabilities along with the character. If the probability of error is very large, then the error correction routine will try to find the most likely substitute for the character. There are at least two general approaches to the correction problem [73]. In *STATISTICAL ERROR CORRECTION*, letter transition properties for the English language are used. For example, probability that a follows b is computed from English text beforehand. These probabilities are usually computed with a depth of two; i.e. the probability that this character is an 'a' given that the last *two* characters were 'b' and 'c'. Using the Viterbi algorithm, the word with the minimum distance can be computed by searching the trellis using the transitional probabilities as the distance measure. The major disadvantage to this algorithm is that it doesn't guarantee that the resulting string is a valid or correct word. In a *CONCEPT DRIVEN CORRECTION* scheme, special properties such as syntax, semantics, or valid words are used to guess the word. For instance, an

inputted word can be compared to a dictionary of legal words. If the word is not valid then the most likely word is computed and the word is replaced. Although this dictionary can be large the search can be narrowed down if certain assumptions are made about the errors (for instance, only letter substitutions need to be considered in most cases).

Both of these methods assume that the text will follow certain patterns. This assumption is repeatedly violated with proper nouns and other special text. However, it should be noted that human capabilities are also severely tested when trying to recognize unfamiliar names and the words which are not legible. In addition, these methods do not take the context of a word into account. Statistical recognition favors high incidence words which may or may not fit a given context.

Although the computation involved in the transitional probability algorithm is minimal, it does very poorly at replacing the word with a valid word. Conversely, the dictionary algorithm is very slow (due to the large number of legal words), but it always replaces the incorrect word with a valid word - which because of the large constraint inherent in the dictionary, has a higher probability of being the correct word. It is therefore obvious that some type of hybrid approach to the problem would be expedient. There are two ways to combine algorithms, by cascading and by integrating [73].

In a cascaded approach, a word can be obtained by computing the letter transitional probabilities. The resulting word can be checked with the dictionary to determine whether it is a legal word. If the word is illegal, then a new word can be tried. Although this further constrains the dictionary, it still results in a large number of comparisons.

The second hybrid approach, integration, involves searching both the Viterbi trellis (for the transitional probabilities) and the dictionary simultaneously. The dictionary is stored as a linked list of characters (called tries). Each node has a successor and a brother. The successor is the next letter in the word whereas the brother is another possible word with the same prefix. For example, "S' can appear with two possible continuations (S)mooth and (S)krzypek. With this dictionary, the set of possible letters after a given prefix can be computed. This set is used to constrain the trellis to search only the valid combinations thereby greatly constraining the search. The problem becomes difficult when there are many possible continuation of words which don't exist in the dictionary Comparison between these algorithms shows that for a given sample, better transition algorithm recognized 30% words correctly [73]. This rate increased to 83% and 86% for Dictionary algorithm and Integrated algorithm correspondingly.

This integration could be carried one step further by combining it with the character recognition algorithm. Here, the trellis and dictionary can be used to constrain the set of possible characters. Although this might not increase efficiency for machine or even hand printed characters, it could benefit the recognition of script characters by improving their segmentation. A typical algorithm might feed the recognition routine a list of expected characters and the algorithm would search only for those expected characters. A number of problems associated with this method can be addressed. For instance, although in general, proper nouns won't be included in the dictionary, they can be flagged by the recognition of a capital letter at the beginning of the word. Moreover, AI techniques can be used to 'understand' the sentence and predict the part of speech of the word and perhaps even the word itself!

One approach to using AI techniques in recognition of recursive script was introduced by Bozinovic and Srihari [45]. They presegmented script characters by finding the minima then used topological features to hypothesize the character and attach probabilities to each possibility, then they used a simple dictionary look up to compute the correct string. They found (with a small character subset) that although in may cases the algorithm converged to the correct word quite quickly, there were some cases where the algorithm spent a lot of time searching the dictionary for possible words. Nevertheless, they did demonstrate the need and the possibility of combining lexical analysis (string correcting techniques) with the recognition algorithm.

At the present time there are no examples where neural networks were used to perform postprocessing in script recognition. Perhaps one reason is that computing contextual information is a difficult problem in itself. In addition it seems that segmentation of a connected, cursive script is a difficult problem. Furthermore, symbolic processing as known in traditional AI, at the level of words or meaning behind a sentence, can not be easily mapped onto

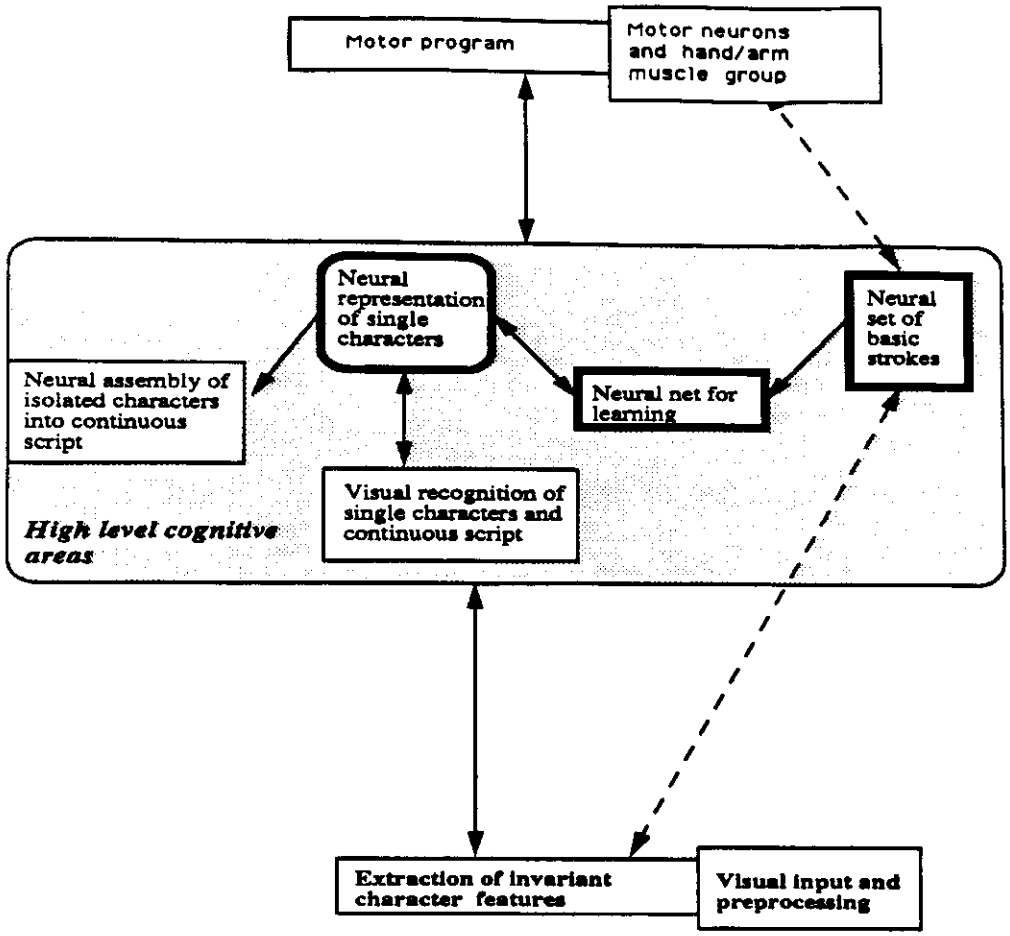


Figure 1: Functional block diagram showing organization of the proposed neural representation of characters.

neural nets. It is intuitively clear that context information would be very important in segmentation. Perceptual completion of familiar patterns was modeled as an interactive activation of context effects in letter perception [74]. The study illustrated how "neural" net models can resolve multiple, parallel constraints to recognize familiar patterns and fill in missing portions. Unfortunately, the study was limited to recognition of four letter words and the mothod of computing context information in general was not elucidated.

5 SCRIPT GENERATING PROCESS - ASSUMPTIONS

One of the key problems in character recognition is selection of features that could be independent of style or font. It is hoped that a combinations of such features can uniquely differentiate various characters. The examples of such features, selected in past rather intuitively, include lines, cusps, holes, loops etc. Our approach is also feature based, but the motivation for feature selection derives from a model of handwriting.

The functional block diagram on which we based our approach is shown in Figure 1. In this diagram we make many assumptions which are reasonable in view of our belief that preprocessing stages are not specific to character recognition but are subserve the vision process in general. The three, thick-line boxes within the shaded box called High Level Cognitive Areas are the focus of our interest and below we try to justify some of the connectivity between them.

Neurophysiological as well as psychophysical (for review see [75]) evidence suggests that extensive local connec-

tivity, cortical magnification factor [76,61] and asynchronous simultaneous processing [77] in the layer of cells allow to synthesize a data driven architecture that achieves location, size and orientation invariance. Gaps in traces or boundaries such as when the pen is lifted although the hand is following a trace, can be completed or filled in by combining lateral agonistic and antagonistic interactions between neuronal feature detectors of various modalities. This is similar to combination of spreading activation [52] with local feedback [26]. It is conceivable that "illusory contour" neurons [18] also play some role in this process. Features can be extracted at multiple spatial scales [48,26] and they can be clustered in a scale space. Segmentation is possible by localizing and integrating regions of the image by "saccading" through locally enhanced features that attract foveal attention as compared to surrounding regions where activities are low. Finally, it is conceivable to have recognition and a primitive form of visual learning in such simple structures provided some mechanism of short-term memory is available [78]. Most of these functions are localized to boxes that deal with visual input in Figure 1.

The motivation of our approach to the problem of recognizing cursive script is based on the assumption that handwritten characters are formed from a limited number of primitive hand motions [22,8,7]. The strokes resulting from these primitive hand motions have a separate neural representation as indicated by the right-most box in the middle of Figure 1. This differs from traditional approaches in that our selection of recognizable features which define characters are constrained by the accepted model of handwriting. Hence the dashed arrow from the motor neurons and/or motor program area to the box called "Neural set of basic strokes". The characteristic features are invariant under rotation, translation, changes in size and/or style. These features can be used for categorizing visually sensed characters and for controlling motor activities during writing. We assume that after the initial learning period, motor programs underlying handwriting result in ballistic hand movements, where there is no instantaneous positional feedback. Higher levels of the central nervous system issue appropriate commands which through some intermediate neural structure control the fine behaviour of the motor neurons and hand muscles in a feedforward manner. This simplifies the question as to at which point positional and visual feedback become important in this process. The representation which we used is derived from the Oscillation Theory of cursive script [8]. It can be learned by a modified "backpropagation" neural network, thus overcoming many image preprocessing problems.

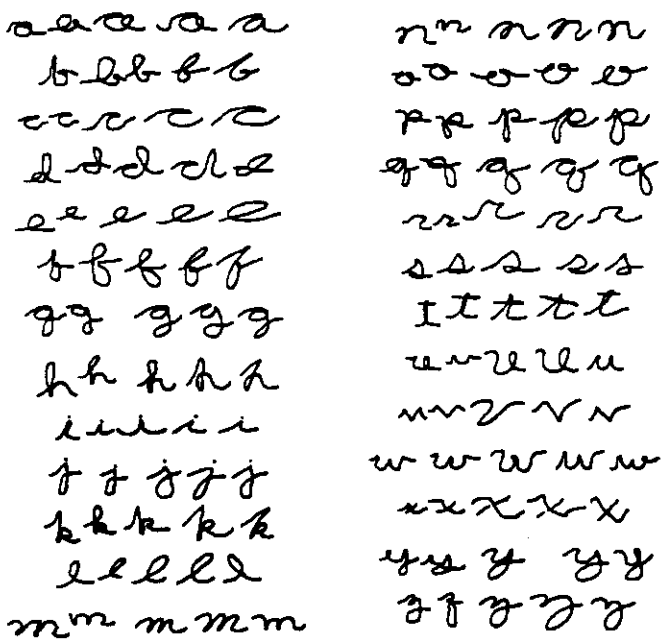


Figure 2: Example characters from "a" to "z" used in training and testing the clustered neural network for script recognition.

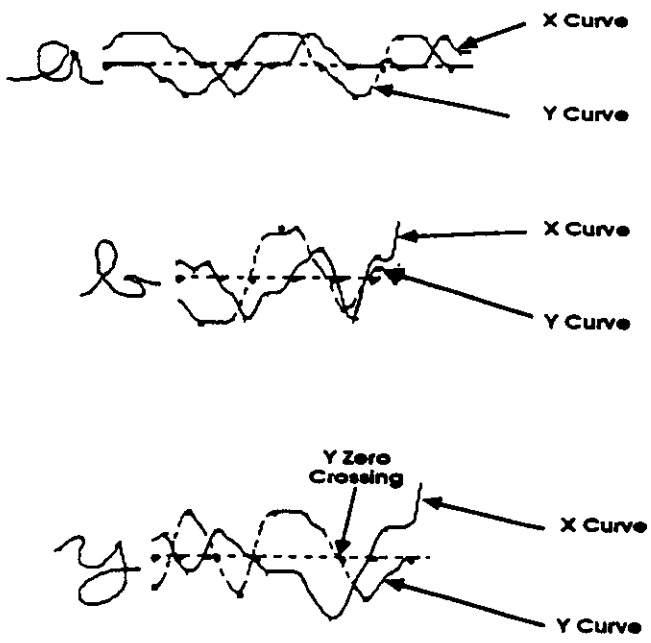


Figure 3: Sample X and Y waveforms derived from characters in the testing and training sets.

Another assumption that we make is that characters are not stored in in iconic form, to be recognized later by matching with specific templates each time the character is presented. Simply there are too many variations due to fonts and writer styles. A more effective way is to encode patterns according to some invariant properties. For example spatial relationships between parts of a character are invariant. These relationships have iconic coordinate representation in the visual domain that is perhaps localized within parietal lobe. At a higher level the same coordinate information is translated into symbolic domain, resulting in categorical representation, where parts are related to each other with verbal symbols such as "above", "below", "adjacent" etc. This is consistent with neurophysiological results [79], which suggest two separate visual pathways. One path, including parietal lobe is concerned with spatial vision where the main function is description of location and spatial relationships between object/parts. The other, pathway, concerned with shape analysis, revolves around the question "what?" and includes inferior temporal lobe. Both of the system might have their own memory and the recognition of the object may involve, sequential task of combining parts of object according to separately stored description of organization of the object. In other words, parts of a shape are stored separately from the description of spatial relations among parts and only one reference point at the time specifies location of one part with respect to other parts [80]. In this context we assume that a few basic strokes, out of which each script character is composed are stored in one location and the font and writing-style independent description of how strokes should be organized into characters are perhaps stored separately (Figure 1.).

6 PROCEDURE

Each hand written character is first generated as a binary image. Sample characters were created via three procedures which have been used in previous studies (see [81,22]). Figure 2 shows example of characters between "a" and "z" used for the training and testing sets. First, data was mapped onto bitmaps (approximately 60x60 pixels) using a mouse and a graphical interface. Second, similar bitmaps were created with a digitizing board. Third, some sample bitmaps (approximately 150x200) were generated by a simulation of Hollerbach's Oscillation Theory [8]. These bitmaps are used to create the training and testing sets for a backward error propagation network according to the procedure described below. In total, 364 characters bitmaps were generated, half reserved for the

training set, and the other half for the testing set.

A sequence of points is extracted from the bitmap representation of the character using a modification of the line following technique described in [31]. This procedure, which we have labeled dynamic feature extraction (see Figure 3), attempts to recreate the method of producing the character by traversing the points in their original order, while thinning the line to a nominal width. The resulting values can be used to create two separate graphs of horizontal (X) and vertical (Y) oscillations that are principal to localization of the modulation points in both dimensions. To overcome the differences in size we normalize both X and Y graphs to values between zero and one. The structure inherent in the learning network is used to accommodate any shifts in the modulation points. In the end the modulation points from both X and Y graphs are used as a minimal representation of characters that are fed to the neural network (see Figure 3).

These graphs were analyzed to detect and localize the major positive and negative peaks, as well as zero vertical velocity crossings. Each peak is listed by its relative position on the time axis, along with an indicator whether it occurred on the X or Y velocity graph, and its relative magnitude (-1 to +1). The zero crossings are recorded with a relative position on the time axis (for the Y graph only), and a relative magnitude of 1.0. This list of peaks is preceded by the number of points scanned in the image (this value is used to normalize the peak positions), and the letter that is represented (for the net "supervisor"). Supplemental data can then be added to the input representation. In our experiments, the only information added was the indication of the presence or absence of a cross-stroke on the letter. This improved performance by helping to differentiate among letter pairs such as 'l' and 't' and 'c' and 'x'.

Our approach is based only on the relative velocity profile derived from the curvature of the input script. This differs from Morasso's [7] approach where he segments the written word at the Y zero crossing points of the velocity profile while using the curved segments from the input. Also, the graphotopic map cannot directly recognize a given character, but may provide a compact representation based on the temporal sequence of activated map cells.

The formatted character data is presented to a modified three-layer backward error propagation network [69]. The net is actually composed of several small networks, of the same size, that share a common set of input nodes. Each small network has independent hidden and output layers, and the output of the whole "cluster" of networks is the average of the corresponding output node values. The averaging function is performed by a fourth or "judge" layer. An example of the structure of a cluster is shown in Figure 4; it consists of ten separate backpropagation networks each constructed with six hidden nodes and twenty-six output nodes. One possible refinement to increase recognition accuracy using the cluster architecture would be to implement a more sophisticated judge layer. Better solutions would include a judge with weighted links and memory in order to select a sub-network's output depending on its strength.

The activation function performed by each node is the familiar sigmoid (1/1+exp(-a)), where a is the sum of all the weighted inputs to the node. Output strength is measured in the range 0, +1. When presented with novel data, the network tends to activate several output nodes (three to four) with varying strengths. These strength values indicate the closeness of the match between the input data and the character representation learned from the training set data.

7 RESULTS

Dynamic feature extraction essentially localizes the character in the bitmap and normalizes its size. Additionally, the output produced by the line-following method of extracting points is not affected by character slant or orientation, as long as the system can locate the beginning stroke. The preprocessed character features are stored as a sequence of peak values, and the average number of features for all examples in both the training and testing set is about sixteen. The order of examples in the testing set is also irrelevant, as the network is newly activated at each presentation.

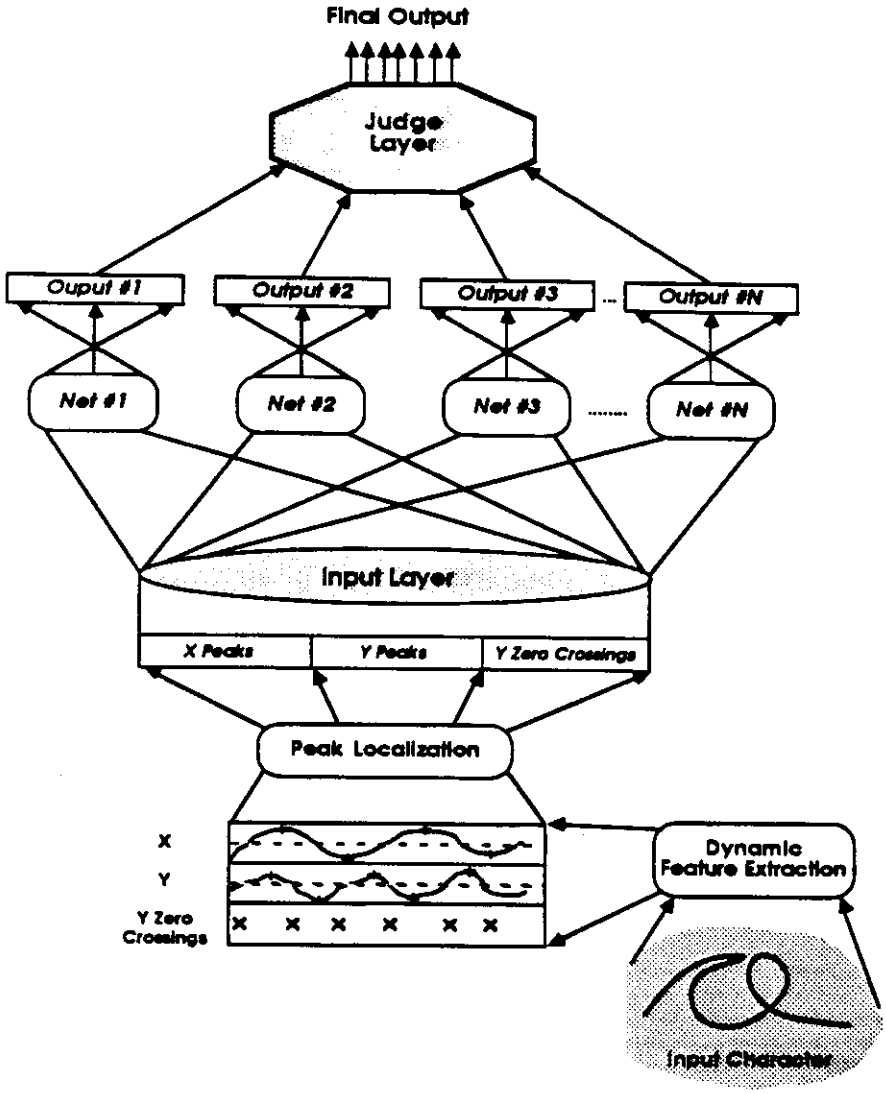


Figure 4: Overall diagram of the cluster network structure and dynamic feature extraction.

The sequence of peak points chosen by the preprocessing step are identified only by their occurance on the X or Y waveforms. There is no explicit coupling of X or Y peaks with non-peak values on the other waveform. The use of Y zero crossings gives the network some reference to the shape of the Y graph, but there is no information concerning the slope of the crossing This problem is being addressed in [82]. Note that the coupling of X and Y values does not directly follow from Hollerbach's Oscillation Theory [8].

Clusters networks converge in approximately one half the number of epochs as single networks. An epoch is a period in which all examples are presented to the network exactly once. Convergence occurs when the network has learned to correctly classify the training examples within a specified tolerance for error. In our trials, the cluster networks converged in approximately 1000 to 1200 epochs with a mean squared error of 0.001 (about three percent) as compared to single networks which require an average of 2000 epochs to reach this error value. The learning curves for a cluster and a single network are compared in Figure 5.

Cluster networks easily recognize the characters in the training set (100 percent), however, with novel data, the correct character is contained in the output group about eighty percent of the time. This is a significant (ten to fifteenpercent) improvement over the performance of single backward error propagation networks. The errors are generally of substitution character (about twenty percent), with a less than one percent omission rate. Recognition

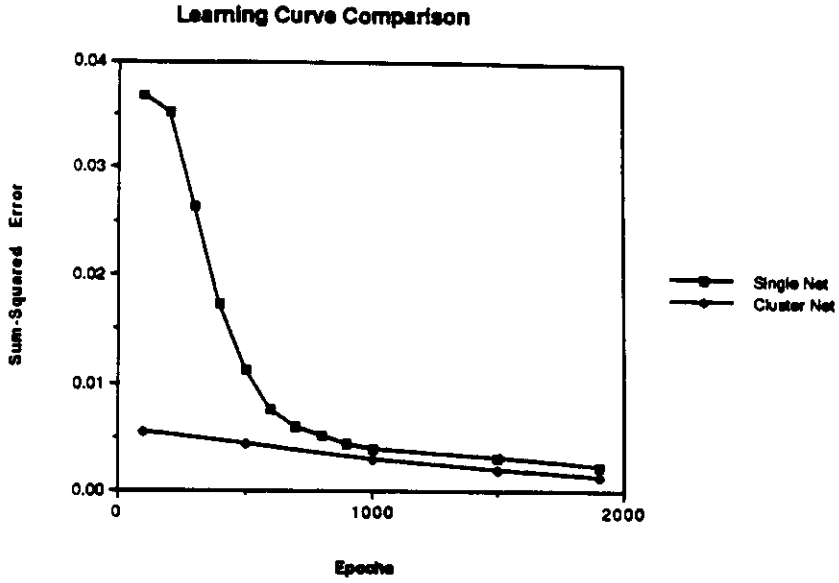


Figure 5: Learning curves for Cluster and single Backward Error Propagation networks.

has been improved further by the use of supplemental information involving the cross-stroke needed to complete the t and the x. The study by Corrieu and DeFalco [83] has shown that children can classify about eighty percent of printed script characters presented to them.

As in other backward error propagation networks [69,70], there is an optimal number of hidden units necessary to maximize generalization when the network is presented with novel examples. This character recognition task performs optimally with fifty to sixty hidden nodes, divided among seven to ten sub-networks. Generalization tends to decline for over one hundred hidden nodes, and similarly, training becomes poor for fewer than thirty hidden nodes.

The cluster networks tend to confuse such letter groups as (b,f,h,k), (e,i,r,l), (g,p,q,y,z) and (m,n,u,w). This agrees with data from Bouma [84], who defined seven major confusion classes: (a,s,z,x), (e,o,c), (n,m,u), (r,v,w), (d,h,k,b), (t,i,l,f), and (g,p,j,y,q). The letters most easily recognized by the network are in the group (e,g,j,l,m,p,t,x) and those most difficult are (a,b,c,h,n,o,q,r,u). The Corrieu and DeFalco study found that the group (c,e,i,j,m,w,x,y) were most easily recognized by the children and the group (b,h,k,n,p,q,r,s,u) among the most difficult. While the data sets compared above are similar, their disagreement can be accounted for in the difference between cursive and printed script.

Figures 6 and 7 show activitiies in each subnet of a cluster during training and during the test of the net. At the present time we have no explanation why some of the subnets would display similar activities during learning leters as for example "j,k,l", nor can we hypothesize uneven activities by the same subnets during testing phase with "j.k.l". These analysis will be addressed in our future work [82].

A system of this nature is designed to work interactively with other (possibly network-based) higher-level classifiers. These would include links to systems that recognize different word shapes, linguistic properties and semantic information. Word shapes are useful indicators to frequently used words (the, it, a). Linguistic properties include the part of speech as well as the orthographic rules governing letter juxtaposition. Semantic information can narrow down the choice of candidate words, and hence impact letter identification. Taken together, these various classification systems could be used to identify handwritten word patterns, and hence, the constituent letters.

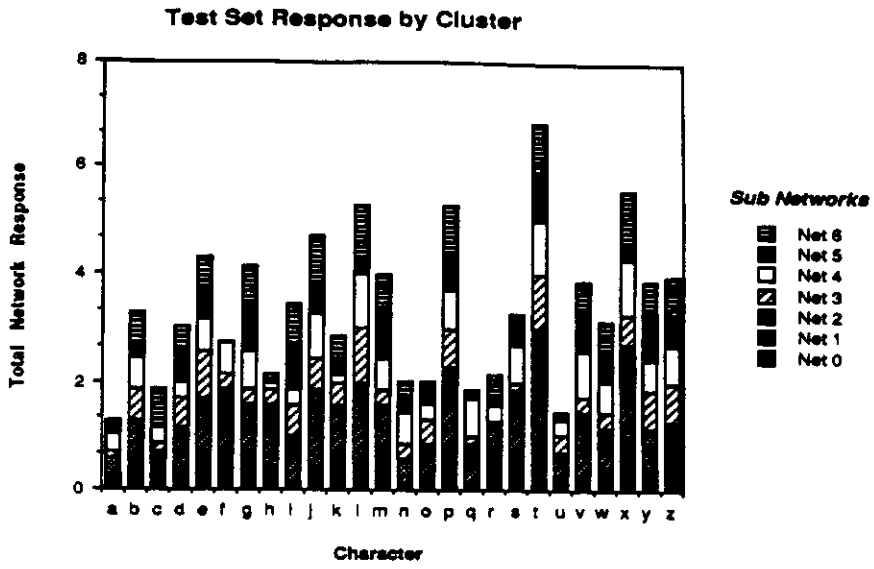


Figure 6: Activities within each sub-network of a cluster during test on each character in the alphabet.

8 CONCLUSION

In the future we would like to extend the script recognition problem in the following way; instead of deciding which features are important to learn, we want to set up a stochastic relaxation network that could differentiate the strokes itself by learning the correspondence between strokes within the internal model of a character. In other words given that we know the model, ie script characters of various styles, network must first develop its own representation for basic components that are constrained by visual feature detectors. The recognition of any character would then become a very fast neural learning implemented as a relaxation, constrained by the existing models. Further extenssion would be to develop an interactive activation model that could select features through competitive learning, by satisfying multiple constraints of letter-feature detectors, motorprograms for character generation and contextual information at the word or sentence level.

Many algorithms for characters recognition have very limited applicability. Some algorithms apply to the characteristics of Chinese characters while others to the characteristics of English script. It seems however, that humans have no structural preference, that could be detected in all alphabets; There is wide variety of alphabets and the structural differences in individual letters within those alphabets. Moreover, the evidence that humans 'turn' objects over in their minds in order to match rotated objects [15] implies that the object representation is a spatial one. It is also interesting to note that when reading words that are upside down, there is an initial period when the reading is very slow. After this initial 'training' period people adapt and are able to read at a normal pace. This fact can be explained by assuming that during the initial training period, the letters are rotated mentally and then recognized. Afterwards, a representation of the inverted character is stored for immediate recognition. Finally, it is also interesting that objects that complicate the character structure for instance, serifs, don't impede recognition. Some recognition algorithms need a special preprocessing step to eliminate the serifs. Humans however, seem to have a preference for serifs. This would seem to rule out statistical methods for the human recognition system since serifs can easily change any statistical properties for the character.

In conclusion, to approach human recognition capabilities some radically different approach is needed. This should involve a spatial representation of the character that can be manipulated and deformed easily. Moreover, this method should be integrated with lexical analysis and other high level routines (such as underline recognizers).

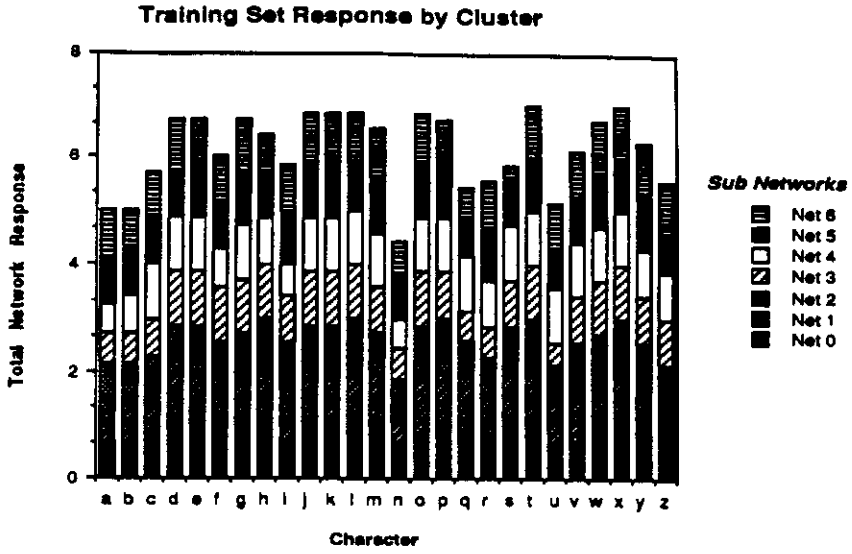


Figure 7: Activities within each sub-network of a cluster during training with each character in the alphabet.

Obviously, this algorithm must be highly parallel and have a large memory cache at its disposal.

Acknowledgements

Support for the UCLA Machine Perception Laboratory environment is provided in part by generous grants from IBM and Hewlett Packard. We sincerely acknowledge support by ARCO-UCLA Grant #1, MICRO-Hughes grant #541122-57442, ONR grant #N00014-86-K-0395, ARO grant DAAL03-88-K-0052 and PMTC-ATI grant #N00123-87-D-0364. Special thanks to Prof. J.Vidal and the students of the MPL for critical comments of previous versions of this manuscript.

References

[1] R Plamondon and G. Lorette. Automatic signature verification and writer identification—the state of the art. *Pattern Recognition*, 22(2):107–131, 1989.

[2] S.T. Kahan, T. Pavlidis, and W. Baird. On recognition of printed characters of any font and size. *IEEE Pattern Analysis and Machine Intelligence*, 9:274–285, 1987.

[3] A. Rajavelu, M.T. Musavi, and M.V. Shrivaikar. A neural network approach to character recognition. *Neural Networks*, 2:387–393, 1989.

[4] Y. Le Cun, L.D. Jackel, B. Boser, J.S. Denker, H.P. Graf, I. Guyon, D. Henderson, R.E. Howard, and W. Hubbard. Handwritten digit recognition: applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989.

[5] L. Lam and C.Y. Suen. Structural classification and relaxation matching of totally unconstrained handwritten zip-code numbers. *Pattern Recognition*, 21:19–32, 1988.

[6] I. Guyon, I. Poujand, L. Personnaz, G. Dreyfus, J. Denker, and Y. Le Cun. Comparing different neural network architectures for classifying handwritten digits. In *Proceedings of the International Joint Conference on Neural Networks, Washington, DC*, pages 127–132, June 1989.

[7] P. Morasso. Neural models of cursive script handwriting. In *Proceedings of the International Joint Conference on Neural Networks, Washington, DC*, pages 539–542, June 1989.

[8] J. Hollerbach. An oscillation theory of handwriting. *Biological Cybernetics*, 39:139–156, 1981.

[9] Marslen-Wilson W. and Tyler LK. The temporal structure of spoken language understanding. *Cognition*, 8(1):1–71, 1980.

[10] V.W. Berninger, Chen A.C.N, and R.D. Abbott. A test of the multiple connections model of reading acquisition. *Internationall Journal of Neuroscience*, 42:283–295, 1988.

[11] J.L. McClelland, D.E. Rumelhart, and G.E. Hinton. The appeal of pdp. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pages 3–44, MIT Press, 1986.

[12] E. Mesrobian, M. Stiber, and J. Skrzypek. *SFINX - structure and function in neural connections.* Technical Report UCLA-MPL-TR-89-8, Machine Perception Laboratory, University of California, Los Angeles, November 1989.

[13] R.P. Lippman. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4–22, 1987.

[14] R.P. Lippman. Pattern classification using neural networks. *IEEE Communications Magazine*, 27:47–64, 1989.

[15] L. A. Cooper and R. N. Shepard. Turning something over in the mind. *Scientific American*, 251(6):106–115, December 1984.

[16] S.C. Fischer and Pellegrino J.W. Hemisphere differences for components of mental rotation. *Brain and Cognition*, 7:1–15, 1988.

[17] M. Ammar, Y. Yoshida, and T. Fukumara. A new effective approach for off-line verification of signatures by using pressure features. In *8th International Conference on Pattern Recognition*, pages 566–569, Paris, 1986.

[18] R.E. von der Heydt, Peterhans, and G. Baumgartner. Illusory contours and cortical neuron responses. *Science*, 224:1260–1262, 1984.

[19] Sabourin. R and R. Plamondon. Preprocessing of handwritten signatures from image gradient analysis. In *8th International Conference on Pattern Recognition*, pages 576–579, Paris, 1986.

[20] P.C. Chuang. Machine verification of handwritten signature image. In J.S. Jackson and R.W. De Vore, editors, *1977 International Conference on Crime Countermeasures,*, pages 105–109, University of Kentucky, 1977.

[21] R.N. Nagel and A. Rosenfeld. Computer detection of freehand forgeries. *IEEE Transactions on Computers*, 26:895–905, 1977.

[22] S. Edelman and T. Flash. A model of handwriting. *Biological Cybernetics*, 57:25–36, 1987.

[23] W. W. Loy and I. E. Landau. An on-line procedure for recognition of handprinted alphanumeric characters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(4):422–427, July 1982.

[24] M.K. Babcock and J.J. Freyd. Perception of dynamic information in static handwritten forms. *American Journal of Psychology*, 101(1):111–130, 1988.

[25] P. A. Dondes and A. Rosenfeld. Pixel classification based on gray level and local business. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(4):79–84, January 1982.

[26] J. Skrzypek. A unified computational architecture for preprocessing visual information in space and time. In *Proceedings of SPIE Conference on Computer Vision*, pages 258–263, 1985.

[27] C. Koch, J. Marroquin, and Yuille A. Analog neuronal networks in early vision. In *Proceedings National Academy of Science USA*, 1986.

[28] H.A. Mallot and W. von Seelen. Neural mappings and space invariant image processing. In *Abstracts, 1st Annual INNS Conference*, page 514, Boston, 1988.

[29] G.A. Carpenter and S. Grossberg. The art of adaptive pattern recognition by self-organizing neural network. *IEEE Computer*, 21:77–88, March 1988.

[30] D.H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, New Jersey, 1982.

[31] O. Baruch. Line thinning by line following. *Pattern Recognition Letters*, 8(4):271–276, November 1988.

[32] A. Iannino and S. D. Shapiro. An iterative generalization of the sobel edge operator. In *International Conference on Pattern Recognition and Image Processing*, pages 130–137, 1979.

[33] R.J. Gleeson and J. Skrzypek. Boundaries as unpredictable discontinuities. In *SPIE Image Understanding and the Man-Machine Interface II*, Los Angeles, 1989.

[34] S.W. Zucker. The organization of curve detection: coarse tangent fields and fine spline coverings. In *Abstracts, 1st Annual INNS Conference*, page 534, Boston, 1988.

[35] S. Busenberg and L. Rossi. Optimization of rotationally invariant object recognition in neural network. In *Abstracts, 1st Annual INNS Conference*, page 483, Boston, 1988.

[36] E.L. Schwartz. On mathematical structure of retinotopic mapping of primate striate cortex. *Science*, 227:1066, 1985.

[37] T. Wakayama. A core line tracing algorithm based on maximal square moving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(1):68–74, January 1982.

[38] N. J. Naccache and R. Shinghal. An investigation into the skeletonization approach of hilditch. *Pattern Recognition*, 17(3):279–283, 1984.

[39] F. Banch. *Pattern Recognition Letters*, 8:271–276, November 1988.

[40] I. Heisey and J. Skrzypek. Color constancy and early vision: connectionist model. In *IEEE First International Conference on Neural Networks*, San Diego, CA, 1987.

[41] W.A. Wright. Contextual image segmentation with a neural network. In *Abstracts, 1st Annual INNS Conference*, page 531, Boston, 1988.

[42] M Yasuhara and M. Oka. Signature verification experiment based on nonlinear time alignment: a feasibility study. *IEEE Trans Systems, Man and Cybernetics*, 17:212–216, 1977.

[43] Y. Sato and K. Kogura. On-line signature verification based on shape, motion and handwriting pressure. In *6th International Conference on Pattern Recognition*, pages 823–826, 1982.

[44] A. Badreldin and M. Shridhar. High accuracy character recognition algorithm using form and topological descriptors. *Pattern Recognition*, 17:515–524, 1984.

[45] R. Bozinovic and S. N. Srihari. Knowledge based cursive script interpretation. In *International Conference on Pattern Recognition*, pages 774–776, 1984.

[46] H.B. Barlow. General principles: the senses considered as physical instruments. In H.B. Barlow and J.D. Mollon, editors, *The Senses*, Cambridge University Press, 1982.

[47] O.G Selfridge and U. Neisser. Pattern recognition by machine. *Scientific American*, 203:60–68, 1960.

[48] J. Skrzypek and E. Mesrobian. Textural segmentation: gestalt heuristics as a connectionist hierarchy of feature detectors. In *IEEE Conference of the Engineering in Medicine and Biology*, Boston, 1987.

[49] G. Lynch, R. Grangner, M. Baudry, and J. Larson. Cortical encoding of memory: hypotheses derived from analysis and simulation of physiological learning rules and anatomical structures. In L. Nadel, L. Cooper, P. Culicover, and R.M. Harnish, editors, *Neural Connections and Mental Computation*, pages 247–289, MIT Press, Cambridge, MA, 1988.

[50] M.V. Shirvaikar and M.T. Musavi. A neural network classifier for character recognition. In *Abstracts, 1st Annual INNS Conference*, page 524, Boston, 1988.

[51] J.S. Denker, W.R. Gardner, H.P. Graf, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, H.S. Baird, and I. Guyon. Neural network recognizer for handwritten zip code digits. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, Morgan Kaufman Publishers, San Mateo, CA, 1989.

[52] S. Grossberg. Competitive learning: from interactive activation to adaptive resonance. *Cognitive Science*, 11:23–63, 1987.

[53] Campbell and Robson. Application of fourier analysis to the visibility of gratings. *Journal of Physiology*, 197:551–556, 1968.

[54] Y. X. Gu, Q. R. Wang, and C. Y. Suen. Application of a multilayer decision tree in computer recognition of chinese characters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(1):83–89, January 1983.

[55] F. Rosenblatt. *Principles of Neurodynamics*. Spartan, New York, 1962.

[56] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, 1986.

[57] D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. A learning algorithm for boltzman machines. *Cognitive Science*, 9:147–160, 1985.

[58] J.R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[59] J.S. Albus. *Brains, behavior and robotics*. Byte Books, New Hampshire, 1981.

[60] F.H. Glanz and W.T. Miller. Shape recognition using a cmac-based learning system. In *Intelligent Robots and Computer Vision*, SPIE, Cambridge, MA, 1987.

[61] A. Rojer and E. Schwartz. A multiple-map model for pattern classification. *Neural Computation*, 1(1):104–115, 1989.

[62] Kohonen T. An introduction to neural computing. *Neural Networks*, 1:3–16, 1988.

[63] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. J. Wiley and Sons, New York, 1973.

[64] C. Y. Suen and Q. R. Wang. Analysis and description of a decision tree based on entropy and its application to large character set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(4):406–417, July 1984.

[65] A. Badreldin and M. Shridhar. A true classification algorithm for handwritten character recognition. In *International Conference on Pattern Recognition*, pages 615–618, 1984.

[66] T. Sagawa, E. Tanaka, M. Suzuki, and M. Fajita. An unsupervised learning of handprinted character with linguistic information. In *International Conference on Pattern Recognition*, pages 766–769, 1984.

[67] R. Shingal and C. Y. Suen. A method for selecting constrained hand-printed character shapes for machine recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(1):74–78, January 1982.

[68] K. Asai and J. Tsukumo. Non linear method for handprinted character recognition. In *International Conference on Pattern Recognition*, pages 770–773, 1984.

[69] W. Lincoln and J. Skrzypek. Synergy of clustering multiple back-propagation networks. 1989. To appear in NIPS Proceedings, Denver, CO.

[70] D.J. Burr. Experiments on neural net for recognition of spoken and written text. *IEEE Transactions on Acoustics, Speech And Signal Processing*, 36(7):1162–1168, 1988.

[71] R. Crick. The recent excitment about neural nets. *Nature*, 337:129–132, 1989.

[72] R. Bozinovic and S. N. Srihari. A string correction algorithm for cursive script recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(6):655–663, 1982.

[73] J. J. Hull, S. N. Srihari, and R. Choudhari. An integrated algorithm for text recognition: comparison with a cascaded algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(4):384–395, July 1983.

[74] J.L. McClelland and D.E.Rumelhart. An interactive activation model of context effects in letter perception: part 1 an account of main findings. *Psychological Review*, 88:375–407, 1981.

[75] E. R. Kandel and J. H. Schwartz. *Principles of Neural Science.* Elsevier, New York, 1985.

[76] E. L. Schwartz. Computational anatomy and functional architecture of striate cortex: a spatial mapping approach to perceptual coding. *Vision Research*, 20:645–669, 1980.

[77] J. L. Adams. *Principles of Complementarity, Cooperativity, and Adaptive Error Control in Pattern Learning and Recognition: A Physiological Neural Network Model Tested by Computer Simulation.* Technical Report, University of California, Los Angeles, 1989. Department of Neuroscience.

[78] M. Seibert and A.M. Waxman. Early vision applications of feature-map diffusion-enhancement nets. In *Abstracts, 1st Annual INNS Conference*, page 523, Boston, 1988.

[79] L.G. Ungerlaider and M. Mishkin. *Analysis of visual behavior,*. MIT Press, 1982.

[80] S.M. Kosslyn. Aspects of cognitive neuroscience of mental imagery. *Science*, 240:1621–1626, 1988.

[81] R.M. Bozinovic and S.N. Srihari. Off-line cursive script recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1):68–83, January 1989.

[82] J. Hoffman and J. Skrzypek. In preparation.

[83] P. Corrieu and S. De Falco. Segmental vs. dynamic analysis of letter shape by preschool children. *European Bulletin of Cognitive Psychology*, 9(2):189–198, 1989.

[84] H. Bouma. Visual recognition of isolated lower-case letters. *Vision Research*, 11:459–474, 1971.