

# NEURAL NETWORK-BASED CONTEXT DRIVEN RECOGNITION OF ON-LINE CURSIVE SCRIPT

PREDRAG NESKOVIC and LEON N COOPER  
*Physics Department and Institute for Brain and Neural Systems  
Brown University  
Providence, RI 02912  
email: pedja@cns.brown.edu and Leon\_Cooper@Brown.edu*

Most of the state-of-the-art systems for cursive script recognition are based on a combination of neural networks (NN) and hidden Markov models (HMMs)<sup>1,2</sup>. The post-processing stage is almost exclusively modeled using HMMs and the dynamic programming (DP) technique (the Viterbi algorithm) is used to efficiently search the space of possible segmentations. In this work we introduce a neural network-based model for representing handwritten patterns as an alternative to HMMs. In addition, we present a new algorithm that uses context information to segment, modify and organize bottom up information in order to achieve successful recognition.

## 1 Introduction

It is generally accepted that our experience influences our perception of the outside world. What we expect heavily shapes what we see or hear. Many of the ambiguities present at the low level of recognition or processing seem impossible to resolve without taking into account cognitive level expectations. A simple and clear illustration of the importance of context during the recognition process is the segmentation of cursive script. Segmenting script from a language that we do not know seems impossible without knowing the language itself (Fig. 1).

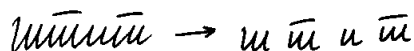


Figure 1: Segmentation of a pattern representing a word written in Cyrillic.



Figure 2: “l r t e” or “b i t e”?

In a language that we do understand, segmentation and identification is influenced by our expectations. The pattern in Fig. 2 can be equally well segmented into “l r t e” and “b i t e” in an out of context situation.

Inspired by some of the properties of the human visual system, such as the

importance of feedback connections from higher processing levels and selective attention, we have constructed a working neural network-based system that explicitly uses context information to segment, identify and organize bottom up information in order to achieve successful recognition.

We have applied our system to a database of cursive script compiled by David Rumelhart<sup>3</sup>. Our dataset consists of words written by 100 different writers, where each writer wrote 1000 words. The size of the dictionary is 1000 words.

## 2 Overview of the System

**Preprocessing.** A raw data file, representing a word from a dictionary, contains information about  $x$  and  $y$  pen positions recorded every 10 milliseconds. The input signal is first transformed into *strokes*<sup>4</sup>, which are defined as lines between points with zero velocity in the  $y$  direction. The preprocessor then filters out strokes with lengths that are less than some threshold length. Each stroke is characterized by a set of features<sup>3</sup>: the  $x$  and  $y$  size of the stroke, the net motion of the pen half way through the stroke, the velocity in the  $x$  direction at the end of the stroke, and the ratio of the frequency in the  $x$  direction  $\omega_x$  to the frequency in the  $y$  direction,  $\omega_y$ . The preprocessor extracts features from the strokes and outputs the feature vector to the segmentation network.

**The Segmentation Network.** We have built a multi-layer feedforward network based on a weight sharing technique to detect letters - the segmentation network. This particular architecture was proposed by Rumelhart<sup>3</sup>. Similar networks can also be found in the literature by the name Time Delay Neural Network (TDNN)<sup>2</sup> and Space Displacement Neural Network (SDNN)<sup>5</sup>.

One of the most useful properties of this architecture is the original and easy training procedure. While the segmentation network learns to detect isolated letters, it is trained on entire unsegmented words. The only information the network needs is whether a letter of a particular class is present or not within the pattern but not the exact location of the letter<sup>a</sup>.

The output layer of the network consists of units called *letter detectors*. The outputs of the letter detectors form a *detection matrix*. Elements of the detection matrix represent recognition probabilities of the letters from the alphabet. Each row of the detection matrix represents one letter and each column corresponds to the position of the letter within the pattern. The units that are in the same row have restricted and overlapping receptive fields and share the same weights. We will denote by  $d^c(x)$  the output of the letter

---

<sup>a</sup>The details of the training procedure have been described by Rumelhart<sup>3</sup>.

detector that represents the letter of the  $c$ -th class,  $c \in [1, \dots, 26]$ , and the  $x$  represents the column number. If it is known which dictionary word the pattern represents, and over which letter the letter detector is located, then we can use the equivalent notation  $d_n^i(x)$  where now the index  $n$  represents the dictionary word, and the index  $i$  numbers the letters within the word.

Since the section of the input pattern that is supplied to a letter detector often contains insufficient or ambiguous information, the outputs of the letter detectors are not very reliable. Therefore, the output of the segmentation network, the detection matrix, represents the *over-segmented* pattern. Only some of the elements of the detection matrix represent correct letters and the goal is to find them.

**The Binding Network.** The final stage of information processing, usually referred to as post-processing, is performed by the *binding* network. The input to the binding network consists of the elements of the detection matrix and the task of the binding network is to select, or bind, a subset of the elements from the detection matrix such that they represent a dictionary word. It is important to stress that the set of the selected elements from the detection matrix that represents a given dictionary word is equal to the number of the letters in the word and not to the number of columns of the detection matrix (in contrast to HMMs) <sup>6</sup>.

The post-processing stage, in most of the state-of-the-art systems, is almost exclusively modeled using HMMs and the DP algorithm is used for finding the “best” sequence of the elements from the detection matrix. In the rest of the paper we will introduce a new model for representing handwritten patterns. In addition, we will present an algorithm that uses context information for searching the space of possible segmentations.

### 3 Motivation for Constructing the Binding Network

We will now describe a simplified scenario of the process of human word recognition and use it as the motivation for constructing our system. Upon the presentation of an unknown pattern it is rarely the case that all the letters of the word are immediately correctly identified and located, unless the pattern represents a short and unambiguous word. It is more often the case that the pattern is “probed” or investigated at different locations, letters identified one by one, or in small groups, and the word slowly discovered. We will assume that upon the presentation of the pattern one letter is identified with higher confidence than other letters and we will call that letter the *central* letter. The pattern is then perceived from the perspective of the central letter, meaning that other letters are identified, and their locations estimated, from the point

of view (i.e. from the location) of the central letter.

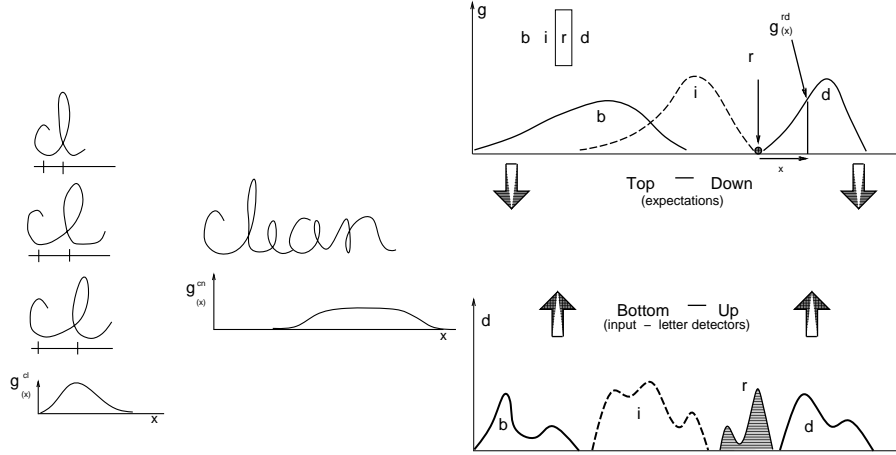


Figure 3: Left: Given the location of the letter “c” the location of the nearest neighbor letter “l” is described with a fairly sharp probability profile. Right: The uncertainty in estimating the location of the letter “n” increases with the number of letters between the letter “n” and the central letter “c”.

Figure 4: Top: A set of probability functions that estimate the locations of the letters of the word “bird” given the location of the central letter “r”. Bottom: Probabilities of detecting the letters of the word “bird” coming from the letter detectors.

One of the consequences of this approach is that the locations of some letters of the word can not be accurately estimated. Locations of the letters that are further away from the central letter are harder to estimate since the uncertainty is accumulating with each letter of the word, as shown in Fig. 3.

Given the central letter, and the view of the pattern from the perspective of the central letter, a group of dictionary words is associated with the pattern. For example, if the letter “c” is identified as the central letter, the words “act”, “ice” and “account” might be associated with the pattern since they all contain the letter “c”. The dictionary word that is associated the most with the pattern, say the word “act”, then *imposes* the structure on the pattern in the sense that the context of the word “act” requires that the letter “a” is located in a certain region to the left of the central letter “c” and the letter “t” is located in a certain region to the right of the letter “c”. Recognition now becomes an *active* process, meaning that we start actively looking for certain letters at specific locations.

**Word Representation.** We will represent a dictionary word as a collection of letters arranged at specific spatial locations. Recognition of a word then consists of a) detection of the letters and b) estimation of their locations. As

we have seen, the segmentation network can be used to detect isolated letters. On the other hand, the location estimate of every letter of a given dictionary word (from the perspective of the central letter) can be found from the training data. We will denote with symbol  $g_n^{ij}(x_i, x_j)$  the (pairwise) probability of finding the  $j$ -th letter of the  $n$ -th dictionary word at location  $x_j$  with respect to the location  $x_i$  of the  $i$ -th letter<sup>b</sup>. One of the simplest ways for approximating pairwise probabilities is to first find a distribution of widths for each letter from the alphabet and then from single letter distributions calculate nearest neighbor pairwise probabilities. Knowing the nearest neighbor location estimates, it is then easy to propagate them and find the pairwise probabilities between any two letters of every dictionary word<sup>6</sup>.

Once the location estimates are found, they are used to select a set of letters from the detection matrix such that their locations fit as closely as possible the expected locations. This is illustrated in Fig. 4, where a “template” (top-down information), representing the word “bird” from the perspective of the letter “r”, is matched against the input (bottom-up information).

#### 4 The Binding Network

**The Architecture.** We will now introduce the labeling of the units of the network. With each dictionary word we associate one unit from the network and call it a *word unit*. A word can be observed from different views, i.e. from the perspective of every letter of the word. Each view of the word is represented with one *complex unit*. A complex unit consists of *simple units* and a *central unit*, and the simple units receive inputs from the detection matrix. An example of the complex unit representing the word “bird” from the point of view of the letter “r” is illustrated in Fig. 5. The location estimates of the letters of the word “bird”, given the location of the letter “r”, are mapped into the weights of the corresponding simple units, Fig. 5.

Every simple unit is class dependent and the region where it expects to see the letter to which it is selective is called the *receptive field* (RF). A simple unit, of the  $n$ -th word unit, combines the contextual expectation ( $g_n^{ij}(x_i, x_j)$ ) and the information supplied by the input ( $d^c(x)$ ), and chooses the *location* of the letter to which it is selective by finding the maximum of its input elements weighted by the “expectation” weights

$$s_n^{ij}(x_i, x_j) = \max_{x_j \in RF} [g_n^{ij}(x_i, x_j) d^c(x_j)], \quad (1)$$

where  $c$  is the class of the  $j$ -th letter of the  $n$ -th dictionary word. The outputs of all the simple units, of a given complex unit, are summed up and

<sup>b</sup>Alternatively, we can use the notation  $g_n^{ij}(x)$  where  $x = (x_j - x_i)$ .

weighted by the detection probability of the central letter into the activation of the complex unit

$$c_n^i(\vec{x}) = d_n^i(x_i) \sum_{j=1, j \neq i}^{L_n} s_n^{ij}(x_i, x_j), \quad (2)$$

where  $d_n^i(x_i)$  is the activation of the central letter of the  $i$ -th complex unit,  $L_n$  represents the number of letters in the  $n$ -th dictionary word, and  $\vec{x} = (x_1, \dots, x_{L_n})$  is a particular configuration of detection matrix elements. Finally, the outputs of all the complex units are supplied to a common word unit representing the  $n$ -th dictionary word independently of the point of view

$$w_n(\vec{x}) = \sum_{i=1}^{L_n} c_n^i(\vec{x}). \quad (3)$$

Note that the activation of the word unit is a function of a specific choice of elements from the detection matrix,  $\vec{x}$ , a specific segmentation of the word. *The goal of the binding algorithm is to find a set of letters from the detection matrix such that the expression given by Eq. (3) is maximized for each dictionary word.*

It is important to note that the spatial arrangement of the receptive fields of the simple units with respect to each other is fixed, but the position of the complex unit over the detection matrix is not. With each selection of the central letter, the network is repositioned so that all the central units (of all the complex units) are placed over the central letter <sup>c</sup>. This is illustrated in the example shown in Fig. 6.

**The Algorithm.** In the beginning of the recognition process an element from the detection matrix, e.g. the one with the highest activation value (an element that represents the letter “t” in column 11 in Fig. 6), is selected as the central letter. The binding network is positioned over the detection matrix in such a way that the central units, of all the complex units, receive an input from the location of the central letter. Every simple unit then selects a corresponding letter (an element from the detection matrix) from its receptive field according to Eq. (1). The activations of the simple units propagate and excite all the complex units which translates into activations of all the word units. The word units with activations above a certain threshold value are selected as the candidate group words thus significantly reducing the effective size of the dictionary.

While all the complex units are initially activated, only one complex unit (from every word unit) wins and activations of other complex units, from the

---

<sup>c</sup>This is similar to the eye movements during the human recognition.

same word unit, are suppressed. Consider now a particular word unit, e.g. the one with the highest activation. One of its complex units, the one with the best match, segments the pattern into a set of letters (a “tentative” segmentation). As it can be seen from the example in Fig. 6, the pattern can be segmented into letters “a-c-t” in many different ways. Every complex unit proposes *one* segmentation, combining high level expectations, and the real input provided by the detection matrix. In the example from Fig. 6 the complex unit CU3 segmented the pattern into the letters “a-c-t” from columns 5, 7 and 11.

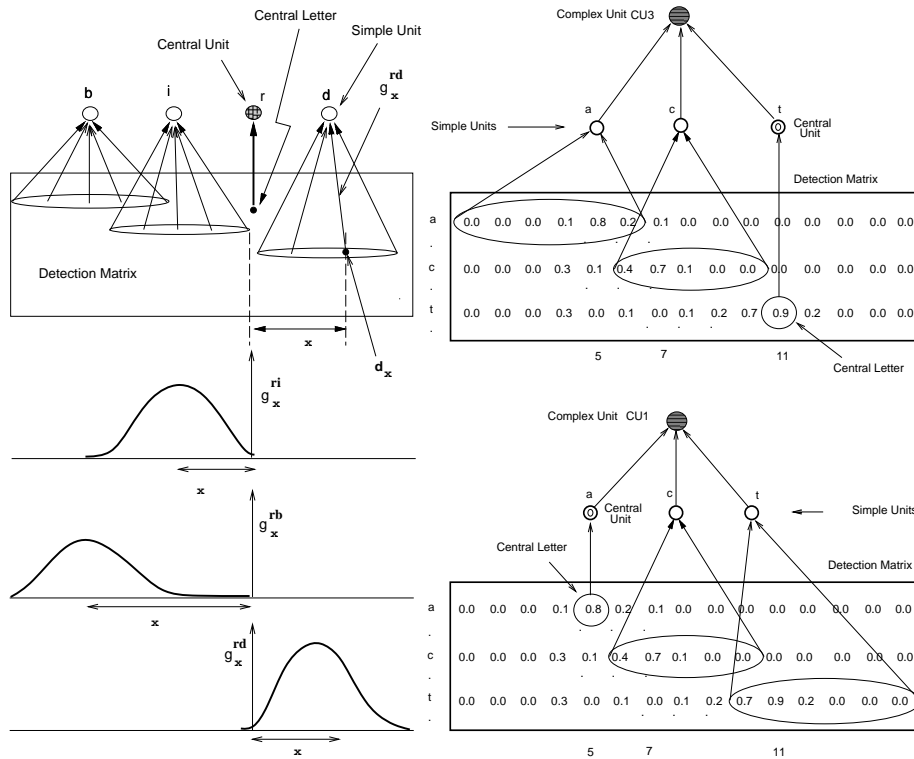


Figure 5: Top: A set of simple units that belong to the complex unit that represents the word “bird” from the point of view of the letter “r”. Bottom: Location estimates are mapped into the weights of the simple units. Figure 6: Two complex units, representing the word “act” from the point of view of the letter “t” (top example), and the letter “a” (bottom example), positioned over different locations of the detection matrix

The network then selects the next central letter, called the *target* letter. The simple unit with the highest activation provides the location (and the

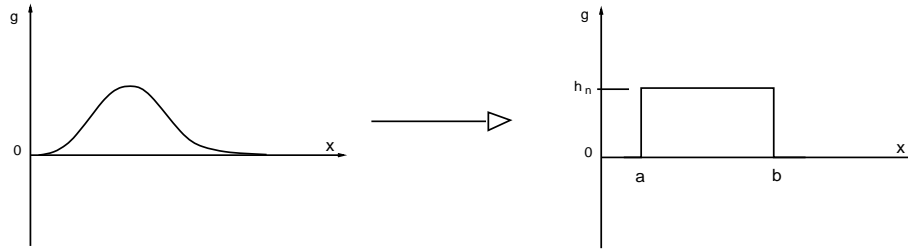


Figure 7: Approximation of the letter location estimates.

class) of the target letter (the letter “a” in Fig. 6). This means that, using context information, the network effectively suppresses the activations of all the elements of the detection matrix except the ones that are within the receptive fields of the simple units.

This tremendously reduces the search time and we say that the dictionary word *imposes* the structure on the input and its expectations guide the recognition. The network is then repositioned over the target letter and another complex unit with highest activation wins. The complex unit chooses its own segmentation and the word value(s) are updated. The goal of repositioning the network is thus to improve the location estimates and to re-confirm the segmentations proposed by different complex units.

Since the simple units operate independently of each other and have overlapping receptive fields, some conflicts may arise during the recognition process. 1) The segmentation “proposed” by a complex unit can be wrong (e.g. the letters can be out of order), and 2) segmentations of the pattern, performed by different complex units, can be in conflict with each other. One of the questions that naturally arises is: how much and whether the performance depends on the choice of the first central letter? It can be shown that the network has the mechanisms to correct itself, if it has chosen the wrong starting letter or proposed an incorrect segmentation. Therefore, the choice of the initial letter is not of crucial importance. The details of those procedures are beyond the scope of this paper and can be found elsewhere<sup>6,7</sup>.

## 5 Implementation and Results

In order to improve recognition estimates of individual letters, we used several segmentation networks and averaged their results. Each segmentation network was trained on 70 writers and an independent group of writers was used as a cross validation set.

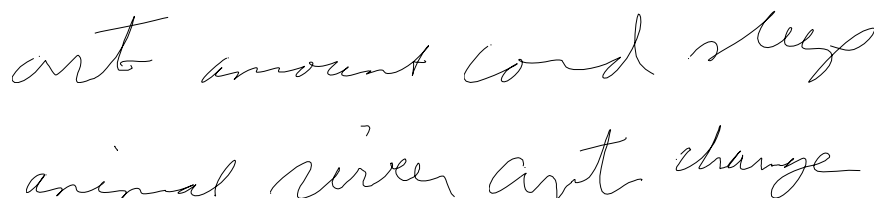


Figure 8: Some of the examples from the database that have been correctly recognized by our system: “art”, “amount”, “cord”, “sleep”, “animal” and “river”. The last two words “ant” and “change” were incorrectly recognized as “art” and “charge”.

The pairwise probabilities, the weights  $g_n^{ij}(x)$ , were approximated as illustrated in Fig. 7. The strengths of all the weights that belong to a given simple unit were set to the same value  $h_n = 1/L_n$ , where  $L_n$  is the number of letters of the  $n$ -th dictionary word. The receptive field of the corresponding simple unit was defined with two parameters: the left boundary  $a$  and the right boundary  $b$ , which are the same for all the nearest neighbor letters. The lower and upper limits for the  $i$ -th and  $j$ -th non-adjacent interaction terms can be approximated as  $a_{ij} = |j - i| \cdot a$  and  $b_{ij} = |j - i| \cdot b$ , respectively. It is important to mention that we have introduced beginning (B) and end (E) “letters” to mark the beginning and end of the pattern, and their detection probabilities are set to some constant value

The system’s performance depends on the writer, on his style and the clarity of his writing, and sometimes the variations among different writers can be above 20%. Some of the examples from the database are illustrated in Fig. 8. The output of the system is the ranked set of words. For good writers the correct word is in the top 5 words over 97% of the time, and the correct word is the top ranked word over 93% of the time. For bad writers the correct word is in the top 5 words over 90% of the time, and the correct word is the top ranked word over 70% of the time.

The only available results on the same dataset were reported by David Rumelhart<sup>3</sup>. The main difference between his system and ours is that his post-processor is based on DP algorithm. The results of his system are as follows: For some writers the top ranked word is selected over 90%, whereas for some writers the correct word is top ranked no more than about 60% of the time. On a reasonably large group of writers, the correct word is in the top five almost 95% percent of time.

It is very hard to compare the results of Rumelhart’s system and ours since the results vary significantly with the choice of the writer and we do not know

which group of writers was used as a testing set by Rumelhart. Therefore, we implemented the post-processing module based on DP algorithm and tested it on the same group of writers we used for evaluating the binding algorithm. Our system outperformed the DP-based postprocessor on every writer and, in some cases, improved the recognition by more than 13%.

It is important to stress that new dictionary words can be easily added to the dictionary, and the system does not require retraining on the new words. The only information about a new word that has to be supplied to the system is the ordering of the letters. Knowing the pairwise probabilities of the letters, it is easy to calculate the location estimates between *any* two letters of the new word and therefore it is easy to construct the weights of all the simple units of a given (new) dictionary word.

### Acknowledgments

Supported in part by the Office of Naval Research. The authors thank Brian Blais, Nathan Intrator, David Mumford and the members of Institute for Brain and Neural Systems for helpful conversations.

1. Y. Bengio, Y. LeCun, C. Nohl, and C. Burges. Lerec: A NN/HMM hybrid for on-line handwriting recognition. *Neural Computation*, 7:1289–1303, 1995.
2. M. Schenkel, I. Guyon, and D. Henderson. On-line cursive script recognition using time delay neural networks and hidden markov models. *Machine Vision and Applications*, 8:215–223, 1995.
3. David E. Rumelhart. Theory to practice: A case study – recognizing cursive handwriting. In E. B. Baum, editor, *Computational Learning and Cognition: Proceedings of the Third NEC Research Symposium*. SIAM, Philadelphia, 1993.
4. J. Hollerbach. An oscillation theory of handwriting. *Biological Cybernetics*, 39:139–156, 1981.
5. O. Matan, C. Burges, Y. LeCun, and J. Denker. Multi-digit recognition using a space displacement neural network. In *Advances in Neural Information Processing Systems*, volume 4, pages 488–495, 1992.
6. P. Neskovic. *Feedforward, Feedback Neural Networks With Context Driven Segmentation And Recognition*. PhD thesis, Brown University, Physics Department, May 1999.
7. P. Neskovic, L. N Cooper, and D. Reilly. Feedforward feedback multiple neural networks with context driven recognition. *U.S. Patent*, January 28 1999. Filed.