

Artificial Neural Networks for Document Analysis and Recognition

Marco Gori * Simone Marinai ◊ Giovanni Soda ◊

◊ DSI - Università di Firenze Via S.Marta, 3 - 50139 Firenze - Italy

Email: {simone,giovanni}@dsi.unifi.it

* DII - Università di Siena, Via Roma, 56 - 53100 Siena - Italy

Email: marco@dii.unisi.it

Technical Report N.1/2003 University of Florence

More material can be found in the home page of the tutorial: “Artificial Neural Networks for Document Analysis and Recognition” held at ICDAR01 and ICPR02 (<http://www.dsi.unifi.it/~simone/ANNxDAR>)

Contents

1	Introduction	4
1.1	Document processing tasks	5
1.2	Representation and modeling	6
2	Pattern representation and coding	7
3	Pre-processing	8
3.1	Text image restoration	8
3.2	Skew detection and thinning	10
3.3	Discussion	11
4	Layout Analysis	12
4.1	Pixel classification	13
4.2	Region classification	14
4.3	Page classification	15
4.4	Discussion	16
5	Character segmentation	17
5.1	Identification of touching characters	17
5.2	Location of cutting points	18
5.3	Discussion	19
6	Optical character recognition	20
6.1	Vector-based feature extraction	20
6.2	Encoding of structural features	21
6.3	Modular architectures	22
6.4	Discussion	25
7	Word recognition	26
7.1	Holistic word recognition	26
7.2	Heuristic over segmentation	28
7.3	Time Delay Neural Network	29
7.4	Discussion	31

8	Signature verification	32
8.1	Discussion	33
9	Conclusions	34

Abstract

Artificial neural networks have been extensively applied to document analysis and recognition. Most efforts have been devoted to the recognition of isolated handwritten and printed characters with widely recognized successful results. However, many other document processing tasks like pre-processing, layout analysis, character segmentation, region classification, word recognition, and signature verification have been effectively faced with very promising results.

This paper surveys most significant problems in the area of document processing where connectionist-based approaches have been applied. Similarities and differences between approaches belonging to different categories are discussed. A particular emphasis is given on the crucial role of the prior knowledge for the conception of both appropriate architectures and learning algorithms. A unified framework is proposed where methods reported in the literature are described and compared. Finally, the paper provides a critical analysis on the reviewed approaches and tries to depict most promising research guidelines in the field. In particular, a second generation of connectionist-based models are foreseen which are based on appropriate graphical representations of the learning environment.

Index Terms: Character segmentation, document image analysis and recognition, layout analysis, neural networks, optical character recognition, pre-processing, recursive neural networks, signature verification, time-delay neural networks, word recognition.

1 Introduction

Today, computers equipped with cameras or optical scanners can read documents and provide their faithful electronic reproduction. In spite of these technological achievements, however, stacks of documents still flood desks of most offices. Whereas documents can be read and accurately stored, the processing required for extracting information is still only in its infancy. Unfortunately, either the presence of noise or the hardly predictable document structure make it very hard to extract information automatically. In the last decade, the decreasing cost of document acquisition, storage and processing, and the renewal of interest in neural networks have given rise to many novel solutions to different tasks of document processing.

In spite of the emphasis on tasks like OCR, isolated word recognition, and graphical item recognition, the application of neural networks to other important tasks of document processing has not received much attention yet. An extensive bibliography on applications of artificial neural networks to document processing tasks¹ shows that most papers (65 %) deal with OCR-related tasks, and with word recognition (15 %), whereas other sub-tasks receive lower attention. To the best of our knowledge, important

¹The bibliography is maintained by the authors of this papers in the Web site of the “Artificial Neural Networks for Document Analysis and Recognition” Tutorial (<http://www.dsi.unifi.it/~simone/ANNxDAR>).

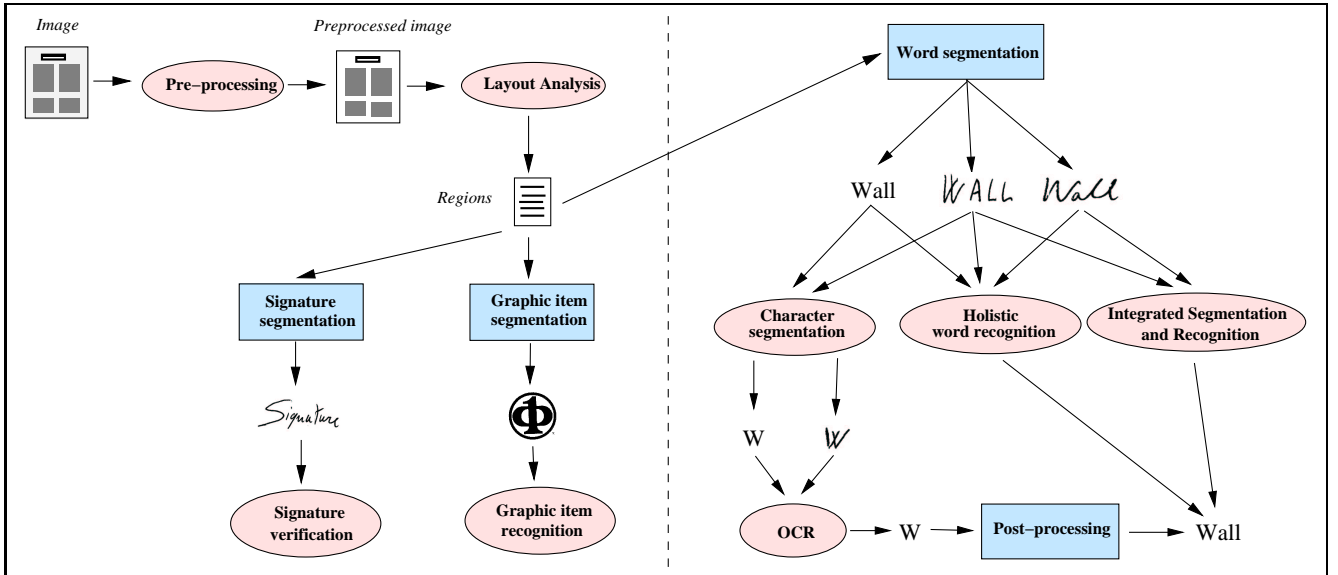


Figure 1: General view of the document processing data-flow. Oval boxes correspond to tasks approached with significant results using neural network-based methods. In the right part, printed, hand-printed, and handwritten instances of the word “Wall” are taken as examples to show different processing flows related to different writing sources.

complementary topics have not been the subject of surveys like for OCR and word recognition [1, 2, 3]. The purpose of this paper is to fill this gap providing an introduction to most significant problems of document processing where connectionist-based approaches have demonstrated their effectiveness. In order to narrow the scope of the paper, we decided to focus our analysis on applications dealing with document images only, thus leaving out the vast amount of literature related to the on-line processing of cursive words and signatures.

1.1 Document processing tasks

The relationships between different document processing tasks are summarized in Fig. 1. *Pre-processing* operations are used in order to improve the input image for subsequent analysis. Common tasks for the pre-processing of document images are de-skew, enhancement (noise reduction) and thinning. *Layout analysis* methods are aimed at extracting the physical and/or logical structure of the document image. In this task we also include page classification approaches. When dealing with the textual parts of a document, the words are usually located by non-connectionist methods like morphological processing and connected components clustering. This task of *word segmentation* is required for the recognition of printed, hand-printed, and cursive text. Word recognition can be based either on the segmentation and subsequent recognition of the individual word characters, or on the recognition of the whole word image as a single unit. In the former approach, based on a “divide-et-impera” scheme, a preliminary step of *character segmentation* is required and the isolated characters are subsequently recognized. In

the second approach (*holistic word recognition*) the whole word is considered as a single object to be recognized. In so doing errors due to wrong segmentations can be avoided. This approach can be effectively used in the presence of a limited dictionary like in postal applications and check reading. An alternative approach is based on *integrated segmentation and recognition*, where the two operations take place at the same time. *Graphical items* are recognized with methods similar to those applied to character recognition, whereas certain specific approaches are considered for dealing with *signature verification*.

The topic of document analysis and recognition is thoroughly dealt with in books and survey papers (see e.g. [4, 5, 6, 7, 8]) where specific material can be found on the above mentioned tasks.

1.2 Representation and modeling

In order to face the tasks described in Section 1.1 one needs to conceive appropriate representations of the data to be processed and select adequate neural network architectural and learning schemes. There is a huge literature dealing with the theoretical and practical aspects of Artificial Neural Networks (ANNs); very good introduction material can be found for instance in [9, 10, 11]. Most of the contributions appeared in the literature are based on traditional schemes and do not propose novel methodologies; their original contribution concerns the way neural networks are applied to the specific task. There are a number of straightforward applications to tasks like filtering and noise removal, character, word, and graphical item recognition. The massive experimentation carried out in the last few years demonstrates that, unfortunately, in spite of very interesting and promising results, critical issues of learnability and computational capabilities often arise when dealing with real-world problems.

The pattern representation typically plays a very crucial role for the effectiveness of the proposed solution. For instance, the idea of receptive fields to provide invariance under horizontal and vertical translations has been proven to increase significantly the recognition performance for OCR [2]. Such a solution, however, is just an attempt towards a more closer look at the issue of pattern representation, which has often been neglected. Basically, in many task, the input to be pre-processed has typically a flat representation based on a vector of features. Whereas this representation is appropriate for many tasks of pre-processing and textual and graphical recognition, structural representations seem to be more appropriate for all tasks which involve the whole document or any parts which exhibit relevant structure. The recent developments in the field of learning in structured domains (see e.g. [12, 13]) offer new unexplored and very promising research directions, some of which are reviewed in the following.

This paper can be roughly split into three main parts. In the first part (Section 2), we briefly discuss some aspects of pattern representation and graph processing by means of artificial neural networks. The second part (Sections 3-5) is devoted to the analysis of neural methods related to pre-processing and

segmentation of document images. The last part covers the “reading” of written documents. This part includes OCR (Section 6), word recognition (Section 7), and signature verification (Section 8). The paper ends with a final discussion in Section 9.

2 Pattern representation and coding

The input and output layers of Multilayer perceptrons (MLPs, e.g. [14]), and similarly the input layer of Self Organizing Maps (SOM, e.g. [15]), are usually problem-specific, since they act as “interfaces” with prior and subsequent modules of the overall neural-based system. The *input coding* is the process of mapping features to network input. This mapping is straightforward when we have a fixed number of features, but more complex when the number of features may vary from one pattern to another. Zoning, for example, can be used in OCR for encoding the feature position. Zoning consists in superimposing a grid on the character image and counting the number of features in each region of the grid. In its simplest implementation, zoning is frequently used for feeding a neural network with a sub-sampled image of variable-size characters. Mutual positions of structural features can be described by graphs, whose nodes correspond to features, and whose edges correspond to relationships between features. Encoding a graph into a fixed-size vector without information loss is possible only if an upper bound to the number of nodes and edges can be defined. Other methods can deal with graphs of arbitrary size, but the overall structure of the graph is lost in the process. Input coding of structural features for OCR is further discussed in Section 6.

When using MLPs for classification purposes, the membership of each training sample (*output coding*) is usually described by means of the so-called “one-hot coding”. Here one output unit is assigned to each class, and all the outputs are set to 0, with the exception of the one corresponding to the right class, which is fixed to 1. Some specific target values are considered when using MLPs acting as autoassociators ([16], pp. 55, 161). An autoassociator is an MLP with the same number of input and output units, and fewer neurons in the hidden layer. When fed with samples of the corresponding class, the autoassociator is trained to carry out an identity function. A classifier can be built by feeding several autoassociators in parallel (one for each class), and considering a decision module which interprets the distances between the reconstructed outputs (for each network) and the presented example. The lower the distance, the higher the similarity between the input sample and the corresponding autoassociator class.

Classic Backpropagation learning schemes have been recently extended to the case in which the patterns are represented by directed ordered acyclic graphs[12]. Unlike other more general graphical structures, this kind of abstract representation does not always come out in a natural way from the

original pattern, yet it offers one of the simplest mechanisms for expressing the pattern structure. The basic idea of the computational scheme is depicted in Fig. 2. For any given pattern an appropriate graphical representation is produced. In the specific example, the contour-tree algorithm [17] is used which associates a node to each contour following a recursive scheme, where for each contour, its internal contours are represented as children. The computation scheme consists of unfolding the neural architecture through the structures.

More general graphs can easily express complex patterns, but the corresponding extension of classic neural networks architectures and learning algorithms is shown to be less effective. In the simplest case, the computation carried out by neural networks on graphs is independent of the node. This is in fact a strong hypothesis which simplifies dramatically the learning scheme but, unfortunately, it is not very appropriate to all pattern recognition problems.

This extended view of neural networks operating on graphs gives rise to a new wave of connectionist-based techniques for pattern recognition, which is somehow in between traditional decision-theoretic and structural approaches. In this paper, we point out that this new approach, which is referred to as *adaptive graphic pattern recognition* and its application to tasks of pattern classification and image retrieval offer the most promising and viable research direction in the field. A general discussion on this new computational scheme can be found in [18]

It is worth mentioning that, so far, the application to pattern recognition has been essentially limited to non-stationary processing of DOAGs. Recent analyses on graphs different from DOAGs as well as models which allows one to incorporate non-stationarity, have not been applied so far to pattern recognition. The development of these issues is likely to be of crucial importance for further developments of adaptive graphical pattern recognition

3 Pre-processing

Pre-processing operations in document image analysis process the input image in order to produce an enhanced image that is more suitable for further analysis. Typical operations include binarization, noise reduction, and skew detection (e.g. [5], chapter 2). In this Section, we review most significant results on the use of neural networks for text image restoration, as well as for skew detection and character thinning.

3.1 Text image restoration

Text image restoration techniques are aimed at improving the quality of document images. This can be very useful for subsequent processing of the image, like layout analysis and OCR. These techniques

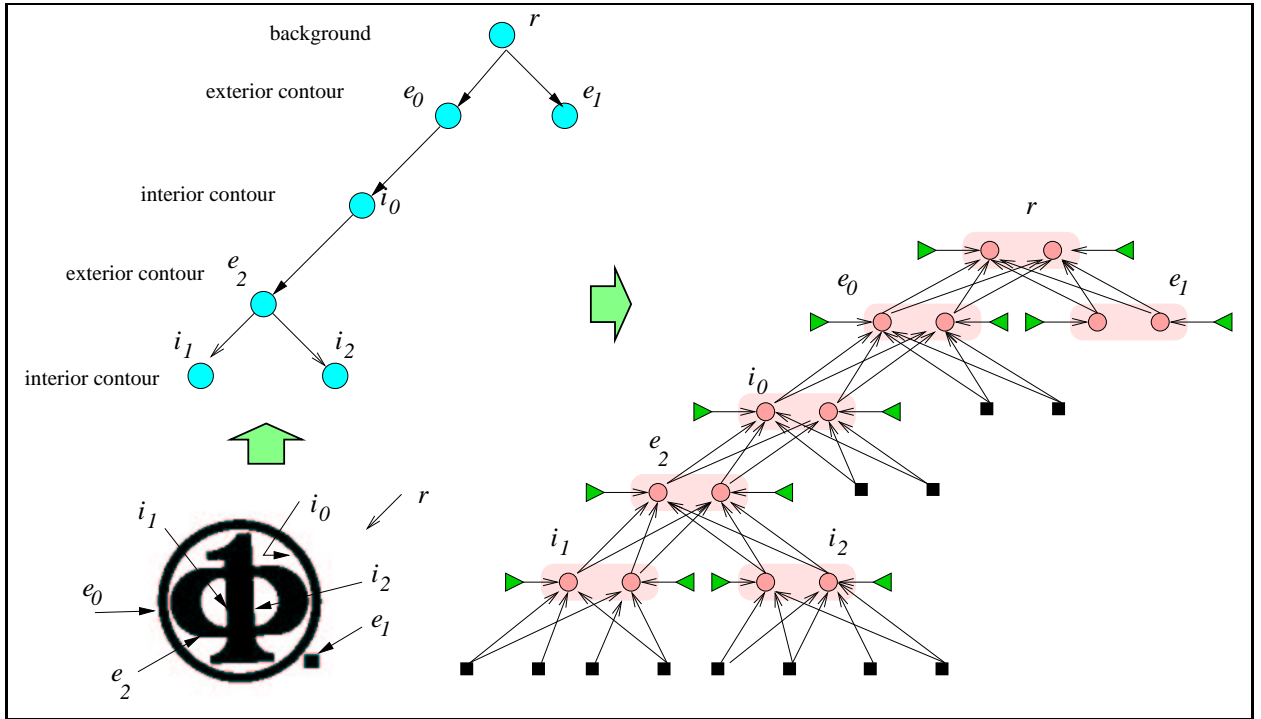


Figure 2: Graphical pattern representation based on the contour-tree algorithm and the corresponding connectionist processing.

are designed to clean broken and touching characters. Typical approaches to text image restoration include morphological filtering [19, 20], filling operations (e.g. the kFill algorithm [5], page 13), Kalman filtering [21], and line following methods [22]. From a computational point of view all these methods are quite expensive, since even the simpler ones require the input to be processed several times. In addition, each method is tuned to a specific kind of noise and cannot be easily extended in order to deal with other noise sources.

Unlike the previous methods, neural networks are frequently used for pre-processing tasks by learning the appropriate restoration filters from examples. Neural networks, and in particular the MLP, can be used to perform filtering operations (Fig. 3) by feeding an MLP with the pixels of a moving window of fixed size. Evidently a single MLP-based filter can hardly be used for cleaning a wide range of noisy documents. An adaptive approach to text image restoration using MLPs is proposed by Stubberud, Kanai and Kalluri [23]. The MLP is used as a filter with a square input window in order to model the inverse of the distortion process. The proposed method adapts the filter by re-training the MLP on each separate page to be processed. Initially, an OCR package is used in order to locate characters which, although distorted, can be identified with enough confidence. Subsequently, the MLP is trained by correlating each distorted character with the corresponding undistorted character, that is properly synthesized thanks to the information provided by the OCR. After training, the MLP is used to restore

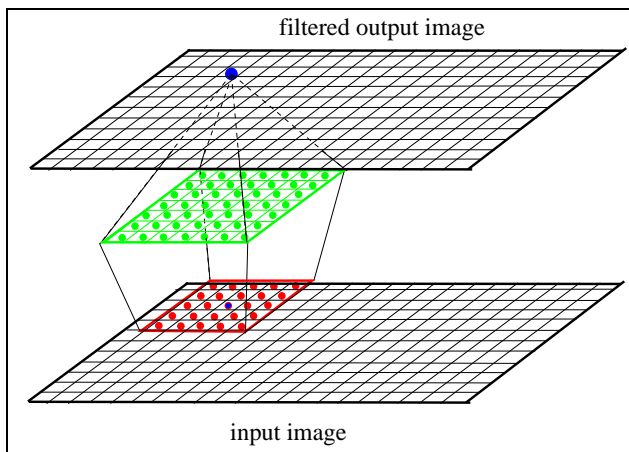


Figure 3: An MLP acting as a filter. The input is transformed onto the output by moving the input window in raster order.

the whole page. The performances are measured considering both the pixel accuracy, that is related to the number of correctly restored pixels and the OCR accuracy.

Broken and touching characters can be unintentionally generated when removing lines overlapping text in checks, forms, and music notation. In these applications the main problem consists of reconstructing the broken symbols after removing the lines. This is related to “image continuation,” which is the problem of restoring images that have undefined pixel values at known pixel locations [24]. In music drawing processing, Martin and Bellisant [25] propose a line removal method that uses an MLP to suggest whether a pixel of a staff line belongs also to an overlapping symbol. The input to the MLP is the histogram of chord lengths² computed at different orientations. Histograms of intersection pixels have two important peaks at most, one for the line and one for the symbol, while histograms of erasable pixels only have one peak around the zero orientation. In order to reduce the computational cost, chord lengths are calculated in small windows (50 by 50 pixels) around current pixels. The network architecture is designed to take local relationships between contiguous inputs into account. For this purpose, the hidden layer has the same size of the input layer, and each hidden node is connected to two adjacent input units only. The cyclic quality of the histogram is taken into account by considering one hidden neuron which is connected to the first and the last components of the input vector.

3.2 Skew detection and thinning

Skewed pages can affect document image segmentation as well as the reading of textual parts. Some de-skewing methods which use neural networks have been suggested for dealing with both handwritten pages [26] and isolated symbols [27]. In both cases a set of appropriate features describing the input is

²A chord is a line segment inscribed in the connected component which is made by the line and the overlapping symbol.

Task	Neural architecture	Input to neural network	Output of neural network	Refs.
Image restoration	MLP	Pixels in a window moved across the image	Restored value of current pixel	[23]
Removal of touching lines	MLP with receptive fields	Histogram of chord lengths	Membership of current pixel	[25]
Page de-skew	MLP	Features computed from the page	Skew angle	[26]
Symbol de-skew	MLP	Moments computed from the symbol	Skew angle	[27]
Symbol thinning	SOM	Normalized bitmap of the symbol	Thinned symbol	[28]

Table 1: Neural approaches for the pre-processing of document images.

obtained and, subsequently, an MLP with one output is trained to reproduce the mapping from these features to the estimated angle.

Thinning algorithms are used for computing features based on the symbol skeleton. Ahmed [28] proposes a clustering-based skeletonization algorithm (CBSA) implemented by using SOM neural networks. CBSA is articulated into two main steps: firstly, a set of clusters corresponding to adjacent pixels are located and, secondly, the skeleton is constructed by connecting the cluster centers. The first step is implemented in [28] by taking the clustering capabilities of SOM during network training into account. In order to perform clustering, the paper proposes a special map (the self-organizing graph) where the adjacency of neurons can change during learning.

3.3 Discussion

As pointed out in this Section, the main property of ANNs which is useful for pre-processing applications is their ability to learn from examples complex nonlinear input-output relationships. In particular, the learning from examples of restoration algorithms allows a rapid adaptation to different noise types and levels without the need for an expensive restoration system re-design. When compared to other learning algorithms, the main advantage of ANNs is that they demonstrate certain immunity to wrong patterns that can be unintentionally generated during an on-line generation of training patterns. In pre-processing applications, ANNs are mainly used for regression, which is quite common in filtering. Sometimes classification approaches can be considered as well, for instance when the neural network is used to decide whether a pixel belongs to the foreground or to the background. A complementary situation occurs in layout analysis (Section 4), where ANNs are mostly used as classifiers. Similarly to other filtering approaches, the choice of an appropriate input window size is a challenging problem. To partially overcome this problem, the use of multiresolution approaches can be taken into consideration, and effective multiresolution approaches employed in layout analysis are described in the next Section. It is important to point out that, as yet, there is no clear evidence of the effectiveness of ANN in

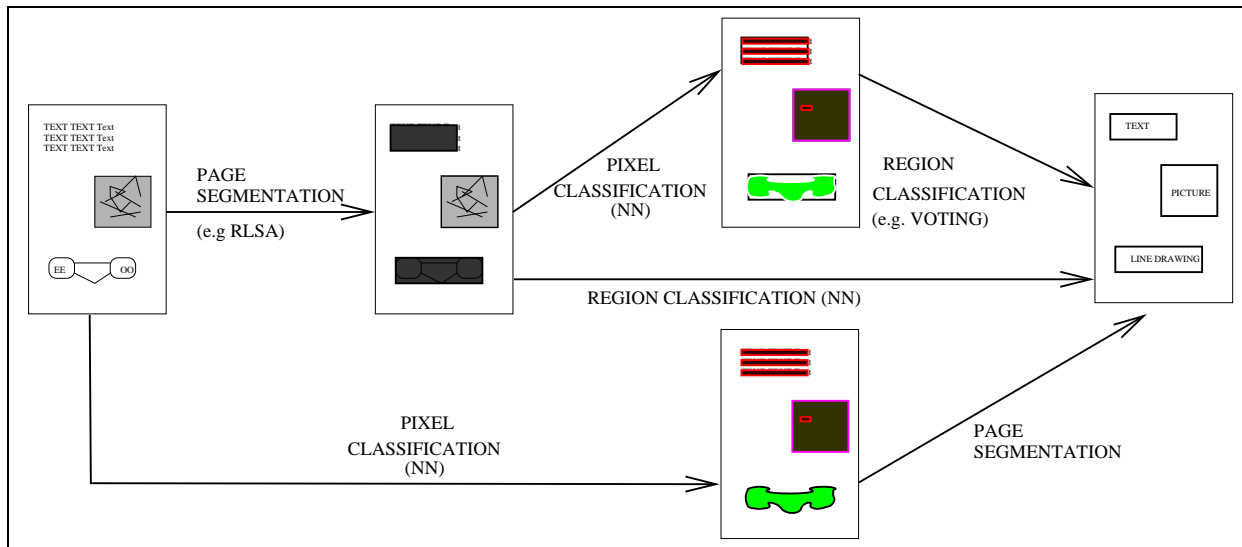


Figure 4: Neural classifiers in layout analysis. Different approaches described in Section 4 follow different links in the picture. Pixel classification – page segmentation [34]. Page segmentation – region classification [35, 36]. Page segmentation – pixel classification – region classification [37].

this field. In contrast to other document processing tasks, the methods proposed for approaching pre-processing problems report experimental results on small data sets, without performing an effective validation. Benchmarking of the results is required in order to emphasize benefits and limits of ANN-based approaches. In addition, there is the need to face problems related to the improvement of the recognition time in approaches based on the on-line training of ANNs.

4 Layout Analysis

Document layout analysis is performed for segmenting the document image into regions with homogeneous content, and assigning a meaning to regions. The segmentation step is called *physical layout analysis*, whereas the assignment of logical meaning is referred to as *logical layout analysis*, or *functional labeling*.

Segmentation algorithms in image processing are usually classified into three main categories: pixel classification, edge-based segmentation, and region-based segmentation. Document segmentation approaches pertain to the two categories of pixel classification and region-based segmentation. In pixel classification the methods are related to those proposed for image segmentation [29]. Most common segmentation methods in document processing belong to the family of region-based segmentation methods, comprising in turn bottom-up (e.g. the Run Length Smoothing Algorithm, RLSA [30] and the Document Spectrum [31]), and top-down (e.g. [32, 33]) methods.

In layout analysis, the classification capabilities of neural networks have been exploited at three levels:

pixel classification, *region classification*, and *page classification*. Fig. 4 and Table 2 summarize some neural approaches to layout analysis described in this Section.

4.1 Pixel classification

Pixel classification was initially applied to the binarization of document images by assigning each pixel either to the foreground or to the background (e.g. [38]). Then, pixel classification was extended in order to deal with additional classes (e.g. text, graphics, and line drawing). The simplest approach to binarization relies only on the current pixel. More complex approaches, like adaptive thresholding, analyze also the grey level of appropriate windows. Most pixel-based segmentation methods rely on the assumption that different types of regions have different textures.

Etemad *et al.* [39] use MLPs in conjunction with a multi-scale wavelet packet representation for document segmentation. The features are the local moments of wavelet packet components computed in local windows at different resolutions. These features are used in order to provide for each pixel various class estimations at different resolutions. The pixel class is computed by considering the MLP outputs that encode the class membership with a one-hot coding.

When the text size is fixed within one or more pages, the texture can be analyzed by considering a single resolution, and by feeding an MLP with the pixels in a small window around the pixel to be labeled. An example of this approach is given by Jain and Zhong [34], using an MLP to discriminate between three classes: background, halftone, and a class comprising text and line-drawing. Only a subset of the pixels of the input window are used as inputs so as to reduce the number of connections and improve generalization capabilities. Node pruning is performed to remove hidden units that do not significantly contribute to classification. As with most segmentation methods based on pixel classification, after pixel labeling the regions are extracted through the smoothing of the labeled image. Finally, text and line-drawing regions are subjected to further discrimination by examining the size of connected components in the regions.

Pixel classification has been recently approached by using a parallel modular architecture based on three sub-networks that are fed with the pixels in the same window [40]. Each sub-network is trained to extract relevant features of one class and is obtained by cutting the first three layers of a five-layer autoassociator. The autoassociators are independently trained to recognize pixels of the corresponding classes (text, continuous-tone, and screened-halftone).

4.2 Region classification

Region classification was initially performed by using global features computed from each region. Classification was carried out using linear classifiers operating on these features according to user-defined parameters [41, 42, 43, 44]. Similarly, when using neural networks for region classification, some features are computed from each region and are used as inputs to neural classifiers. Both local and global features can be taken into account.

Local features are computed for each pixel in the region. In [35] each region, extracted by an RLSA-based algorithm, is identified as *text/non-text* by a SOM-based classifier with local features. Several masks are used to extract textural patterns of the region. The features are the occurrences in the region of the given set of masks, and some values which express the relationships between these masks. When the computation of local features is expensive, a few random blocks within each region can be analyzed. In [37] the histogram of gradient vector directions and the histogram of luminance levels are the features taken into account for each block. Each block is classified by an MLP which yields the classes *printed characters*, *handwritten characters*, *photographs*, *painted images*, and *background*. Finally the region class is determined by a majority voting over the selected blocks.

Le, Toma and Wechsler [36] use global features for the classification of rectangular regions of black and white images, and compare four different kinds of neural networks with the same set of features. Similarly to non-connectionist methods [41, 42, 43, 44], the features are computed using the width and the height of the region, the number of black pixels, and the length of black runs. MLP with sigmoidal neurons, RBF networks, Probabilistic Neural Networks, and SOMs are compared and the authors claim that the RBF-based networks yield a better classification accuracy.

The logical classification of textual regions has been recently addressed in [45] with a recurrent neural network in order to exploit contextual information. The text blocks are extracted from the top-left to the bottom-right corners and are organized as a “temporal sequence.” For each step, the input contains quantitative information on the current block (dimensions, position, number of lines, and number of words) as well as information from the output of the previous block. The recurrent neuro-fuzzy system called RFasArt (Recurrent Fuzzy Adaptive System ART based) is adopted which uses the same general structure of the Fuzzy-ARTMAP [46]. Experimental results are reported for business letters and scientific papers. Examples of logical classes for business letters are *date*, *address*, and *sender*, while for scientific papers logical classes can be *title*, *authors*, and *abstract*.

A similar problem is the logical labeling of handwritten text lines in postal envelopes. In this domain, De Waard [47] proposes the use of recurrent neural networks for identifying sequences of connected components that correspond to the zip code.

Task	Classification level	Neural architecture	Input to neural network	Output of neural network	Refs.
Page segmentation	Pixel	MLP	Pixels in a window around current pixel	Background, halftone, text & line drawings	[34]
Page segmentation	Pixel	parallel combination of MLPs obtained by autoassociators	Pixels in a window around current pixel	Text, continuous-tone, screened-halftone	[40]
Multi-resolution segmentation	Pixel	MLP	Moments of wavelets computed in a window around current pixel	Image, text, graphics	[39]
Identification of text blocks - Global features	Region	MLP	Region size, number of black pixels, etc.	Text, non-text	[36]
Identification of text blocks - Local features	Region	SOM	Occurrences of pre-defined 3x3 masks in the region	Text, non-text	[35]
Region labeling - Voting of labels of few random windows	Region	MLP	Gradient vector and luminance computed in a window	Background, printed & handwritten characters, photo, image	[37]
Logical labeling of text blocks	Region	Recurrent neural network	Block size, number of lines, etc., and label of previous block	Date, Address, etc.	[45]
Logical labeling of text lines	Region	Recurrent neural network	Features of connected components	Postal code or not	[47]
Form classification	Page	MLP	Line crossings	Form identifier	[48]
Page classification in digital libraries	Page	MLP	MXV tree encoding	Page class (title, index, regular, etc.)	[49]

Table 2: Neural approaches in layout analysis.

4.3 Page classification

Page classification has been addressed with different aims and using different methods. Earlier applications concerned form classification methods that frequently rely on the presence of ruling lines in the pre-printed form layout [48, 50, 51]. Other approaches are considered in office applications. In this case typical classes are business letters and technical papers [52, 53]. In the last few years the classification of journal and book pages has received a significant attention [54, 55]. The position of layout objects can be described either with zoning or with a tree-based representation of the page layout.

Taylor *et al.* [48] propose an MLP-based form classification system having input features corresponding to line crossings. Nine types of crossings are possible if all the intersections between horizontal and vertical ruling lines are considered. The crossings can be found at any position on the page, and forms can be distinguished considering the position of these features. Zoning-based encoding is considered with a 3 by 3 grid, obtaining nine equally-sized non-overlapping regions. The number occurrences of each kind of crossing is considered for each region, and after a normalization of the maximum value, a feature vector containing 81 values is computed. The 21 classes are identified with a one-hot coding. An improved zoning has been recently proposed in [55], where the classifier is based on decision-trees.

Page layout is generally of a hierarchical nature, and can be represented by graphs [56] or trees [32]. Neural networks can deal with structural representations using two main approaches: either by coding

the representation into a fixed-size feature vector, or by using recursive neural networks. An example of the first approach has been recently proposed in [49]. Here document images are represented by means of a MXY tree where the cuts are made along horizontal and vertical lines, in addition to classical cuts along spaces considered in XY trees [32]. MXY trees are encoded into a fixed-size representation by considering the occurrences of tree-patterns (containing three nodes) in the tree corresponding to each page. The basic idea that is behind this encoding is the observation that similar layouts frequently contain similar sub-trees in the corresponding MXY tree. Trees composed by three nodes can have two basic structures: the first one is composed by a root and two children, the second one is composed by a root, a child, and a child of the second node.

4.4 Discussion

Neural networks are used in layout analysis mainly for pattern classification, in contrast to pre-processing tasks, where they are used as non-linear trainable filters. Pixel classification approaches are in a certain sense on the borderline between filtering and classification, because as with filtering methods, the input window is moved across the image; however the output of the neural network is a discrete decision among few classes. The most apparent advantage of neural networks acting as classifiers, which is common to other classification methods, is their ability to learn the decision function from examples. This is an advantage when compared with approaches where the classification algorithm is hand-tuned by the user (e.g. see Section 4.2). An additional benefit of neural networks over other learning approaches (e.g. decision trees) is that the network training is slightly affected by few wrongly labeled patterns in the training set. When dealing with pixel classification problems, erroneous training samples can be caused by frontier pixels that could be assigned to more than one class, since the window centered on these pixels covers more regions. By using MLPs with one-hot coding for pixel classification it is also possible to assign to each pixel a similarity measure with respect to every class. As outlined in this section, this feature has been used in [39] in order to integrate the decisions of pixel classifiers at various resolutions.

Perspective uses of ANNs in layout analysis are related to graph-based representations of the page layout. Page layout is generally of a hierarchical nature and can be represented by graphs (e.g. [56]) or trees (e.g. [32]). With few exceptions analyzed in this section, ANNs have not been used with these hierarchical representations. The use of recursive neural networks seems to be very promising for dealing with this kind of representations.

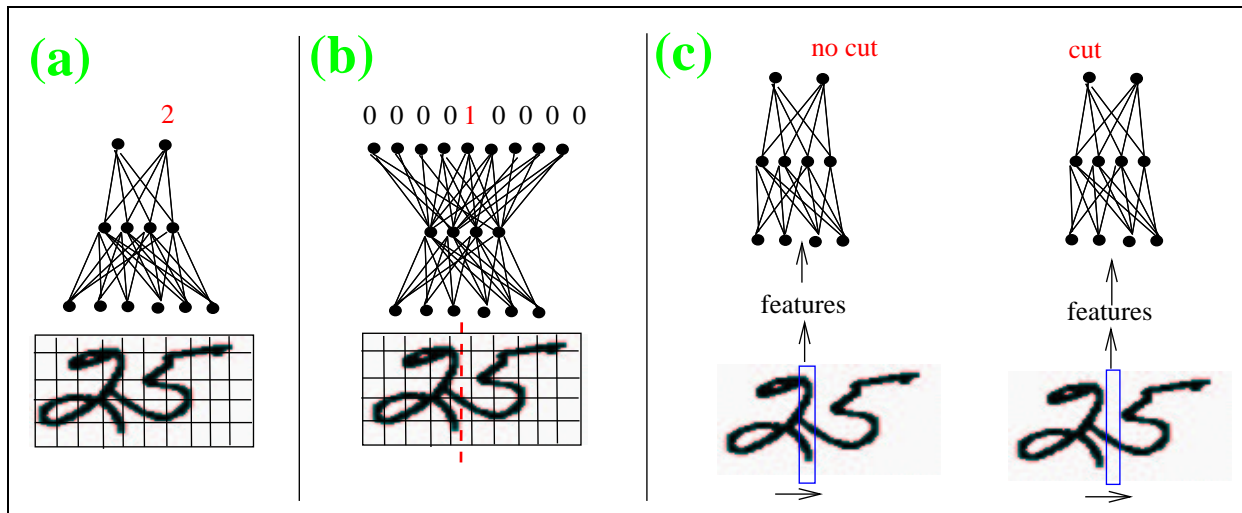


Figure 5: Some neural approaches to character segmentation. In (a) and (b) an MLP is fed with a sub-sampled image. In (a) the MLP is used in order to establish the number of characters present in the image (one or two). In (b) the MLP suggests the cutting point for separating two touching characters. In (c) a set of features are extracted at each horizontal position in the image, and considered as MLP input which suggests whether the position could correspond to a cutting point or not.

5 Character segmentation

Character segmentation seeks to decompose the image of a sequence of characters into sub-images that correspond to individual symbols. Segmentation strategies can be divided into three main categories [7]. *Dissection methods*, which partition the input image into sub-images having “character-like” properties. *Recognition-based methods* that are based on the integration of segmentation and recognition. *Holistic approaches* which avoid segmentation by recognizing entire words as units. In this section we analyze dissection methods employing neural networks, whereas the other methods are discussed in Section 7 devoted to word recognition. Neural networks can be employed in two main sub-tasks of dissection methods (Figure 5): identification of touching characters, and location of cutting points (neural dissection).

5.1 Identification of touching characters

In dissection-based segmentation, sub-images corresponding to touching characters are generally detected either on the basis of the sub-image size or considering a rejection provided by a classifier trained to recognize isolated characters. A valuable alternative is the use of ANNs as discussed in the following.

Wang and Jean [57] propose the use of an MLP having two neurons in the output layer, and trained to distinguish isolated characters from pairs of touching characters. The network input is a normalized image corresponding to one or two characters (Figure 5 (a)). A training set containing more than 17,000

pairs of touching and non-touching characters was artificially generated starting from scanned samples of more than 30 common fonts. Each touching pair considered in training was generated by the simple concatenation of two characters. This approach to the artificial generation of training samples has some obvious advantages over a more immediate strategy of looking for touching and non-touching pairs in some “real” documents. However, a major concern is the generalization capabilities of the network, that is the ability to detect touching pairs found in scanned documents and not artificially generated. The results reported in [57] are encouraging, since all the touching pairs not detected using traditional approaches (based on aspect-ratio and classifier rejection) were correctly detected.

The previous approach can be extended to the more general problem of estimating the length (number of characters) of connected strings. For instance Lu, Chi and Siu [58] train an MLP to classify sub-images on the basis of the expected number of connected digits. The MLP is fed with features extracted from the sub-image, and has four outputs, corresponding to four string lengths (one to four digits). The features are related to the horizontal foreground-background and background-foreground transitions in the image, as well as to feature points extracted from the skeleton image. The dataset considered in the experiments was built starting from a NIST database. Since the number of connected strings with three and four characters is very small (only 48 connected 4-digit strings are available), some additional 3-digit and 4-digit strings are produced by artificially merging existing samples. In this way a balanced training set was obtained.

5.2 Location of cutting points

Neural networks can be used for locating cutting points as well. Eastwood *et al.* [59], for example, propose using an MLP for the segmentation of cursive words. Similarly to neural filters (Section 3), an MLP is “moved” across the input image (in this case only a horizontal displacement is taken into account) resulting in a labeling of the corresponding horizontal position (Figure 5 (c)). Ten simple features (based on the vertical projection profile, the histogram of line crossings, the position of holes and the upper contour of the word) are computed for each x position in the word. At each position the features are fed to an MLP with one output node which is trained to recognize segmentation points based on local information.

If the sub-image to be segmented is constrained to contain at most two characters, then a sub-sampling of the whole sub-image can be considered as an MLP input (Figure 5 (b)). This is the approach considered in [60]. Zoning is applied to these sub-images, which are linearly scaled in order to fit into a 30 by 60 frame. A feature vector of 72 elements is computed from the scaled image by counting the number of black pixels in small, non-overlapping windows of 5 by 5 pixels. One-hot coding is used on the 60 outputs of the MLP to describe the position of the cutting point in the scaled image. During

Task	Number of characters	Neural architecture	Input to neural network	Output of neural network	Refs.
Length estimation	1-2	MLP	Normalized sub-image	Number of characters	[57]
Length estimation	1-4	MLP	Features computed from the whole image	Number of characters	[58]
Neural dissection	2	MLP	Normalized sub-image	Cutting position	[60]
Neural dissection	Variable	TDNN	Features computed at each x position	Current x is a cutting point	[59]

Table 3: Neural approaches for segmentation of touching characters.

sub-image segmentation, the highest output is selected as a cutting point, and used to produce two characters. The segmented characters are recognized by a classifier. In the case of rejection, the next cutting point (suggested by the MLP output), is considered and a further classification is attempted on the new hypothesized individual characters. In this method, training samples are collected and labeled by hand.

5.3 Discussion

In neural dissection a simple sub-sampling of the image can be considered as input only when no more than two characters are expected to be found in the processed sub-image (Table 3). This is related to the difficulty of training a single network being able of locating more than one cutting point in a given sub-image. Roughly 20,000 different combinations of three characters can be expected (although only a small fraction of them will occur in real documents). The training of a network able to discriminate among so many different situations requires a huge number of training patterns, and even the convergence of the training appears to be very difficult. Another aspect common to all the sub-problems considered in neural dissection methods, is the collection of training patterns. Touching characters are not very frequent in average quality documents. For this reason the search for a large number of training patterns can require a significant human effort. To solve this problem most methods use appropriate algorithms to automatically generate synthetic touching pairs of characters.

In spite of the difficulties for training samples collection, the use of ANNs for the segmentation of touching characters has some advantages over more traditional approaches. The first advantage is the ability to adapt to different levels of character overlapping. The lower execution time is another benefit when compared to more complex algorithms.

By comparing neural dissection methods with the word recognition approaches analyzed in Section 7 we can discover several similarities that could be exploited in new approaches. For instance, Spatial Delay Neural Networks (SDNN, Section 7.3) are convolutional networks (derived from TDNN) that are used for character recognition in word strings. Since there are several similarities between dissection methods and TDNN-based word recognition, it seems realistic to take into account also SDNN for

approaching the task of character segmentation.

The high training cost of neural segmentation methods has limited their application to linear dissections, where the separating line is a vertical cut. This approach is appropriate for printed and hand-printed text [7]. When dealing with handwritten and cursive text more complex separation lines are needed and graph-based representations have been applied also to this task [61], and could be used in conjunction with recursive neural networks.

6 Optical character recognition

Thanks to the strong learning capabilities in sub-symbolic environments, a typical representation frequently adopted in OCR is based on a simple sub-sampling of the input which yields a fixed-size vector (zoning). The main advantages of sub-sampling are the dimension reduction of the pattern samples so as to exhibit good generalization to new examples, as well as the noise reduction provided by the low-pass filtering effect of sub-sampling. Several strategies can be considered when using prior knowledge in the neural classifier design. In this section we mainly analyze three crucial aspects for OCR: the feature extraction either vector- or graph-based with the corresponding learning algorithms, and the adoption of modular classifiers.

6.1 Vector-based feature extraction

Prior knowledge concerning basic properties of handwritten and printed characters can be taken into consideration both in designing effective feature extractors and in improving the learning mechanism. Two examples of these properties in OCR are worth mentioning: the presence of linear and/or circular shapes in characters, and the role of edges in providing character information.

Handwritten and printed characters are composed of linear and circular shapes. Feature extractors can take advantage of this property by searching for these shapes in the character image. The Hough transform [62] has been used for locating linear prototypes and ellipsoids in characters [63], as well as for locating lines in Bengali script [64]. With an alternative approach the location of specific shapes can be “forced” during training by presenting the neural network with some artificially-generated patterns. This approach is suitable for learning feature maps in convolutional networks, and was applied in Neocognitron training [65, 66]. In the neural network proposed in [65], the first layer is trained to recognize line patterns by building 12 cell planes, one cell plane for each direction of lines. Subsequent layers are trained to recognize more complex (e.g. circular) patterns. Each cell plane is forced to recognize a given kind of pattern which is pre-defined by a “teacher”.

Similar relationships between feature extraction methods and learning mechanisms can be highlighted

when dealing with symbol edges. In character images most of the information is provided by edges, the signal in the background or in the strokes being of lesser importance. Features related to edges in the image can be extracted by computing the gradient of the image [67] or by taking pixel distance features [68, 69] into account. On the other hand the use of information relative to the input image gradient can be inserted into the learning procedure. Gori *et al.* [70] propose an improved approach to the use of autoassociators to classify graphical items in presence of spot noise. A weighted norm is used instead of the Euclidean norm to measure the input-output accuracy of the neural network. The weights depend on the gradient of the images so as to give less importance to uniform color regions, like the spots. A modified learning algorithm (Edge-Backpropagation) is derived from the classical Backpropagation by considering the weighted error function.

6.2 Encoding of structural features

Structural features describe the symbols in terms of sub-parts and relationships between these sub-parts. There are two ways for using structural features together with connectionist-based classifiers. The first strategy is based on the mapping of the variable-size data into a fixed-size vector that can be processed by the MLP. The second class of methods uses recursive neural networks for processing the structural features.

Handwritten characters can be described by analyzing the stroke contour (e.g. [71]) or skeleton (e.g. [72]). Amin *et al.* [72] describe the character with a graph whose nodes correspond to sub-patterns extracted from the skeleton (breakpoints, straight lines, curves); edges, on the other hand, correspond to mutual positions between sub-patterns. The proposed graph encoding is based on the assignment of some pre-defined slots of the feature vector to each node and edge of the graph. Several values of the feature vector correspond to each node or edge in order to represent its features. This approach is appropriate when the maximum graph size is bounded. One advantage is that the mapping from the graph to the vector is without information loss, and that numeric attributes in nodes and arcs can be considered. Another method which can be employed when node and arc attributes have discrete values, consists in computing the occurrences of all possible combinations of node and edge attribute values. In [73], for example, the handwritten character skeleton is represented by a set of circular arcs, and described by means of an ARG (Attributed Relational Graph [74]), whose nodes (described by 3 features) represent the circular arcs, and whose edges represent relationships between pairs of connected arcs (described by a 2-tuple). In the proposed encoding, each combination of values of the 3-tuple, as well as each combination of values of the 2-tuple corresponds to one item of the feature vector. The value of each item is the number of occurrences of the relative tuple in the character description. Variable size graphs can be taken into account with this method, nevertheless the structure of the graph is lost

during its encoding, and only discrete attributes can be taken into consideration for both nodes and arcs of the graph.

Other approaches rely on recursive neural networks [12] which can process graphs instead of simple sequences like recurrent networks. In [75] recursive neural networks are used to recognize logos described with contour-trees [17] which contain the topological structural information of logos (see Section 2 and Fig. 2). Nodes in the tree are labeled by a feature vector describing the corresponding contour with numeric values. One limitation of this approach is that the maximum number of children for each node must be known in advance.

6.3 Modular architectures

Parallel and serial combinations are the main categories of modular classifiers that we discuss in this Section.

With reference to Fig. 6, in parallel classifiers ANNs can be used for implementing either the individual experts or the combiner. Connectionist-based experts are described by Strathy and Suen [68] (Fig. 6 (b)), which propose a digit recognizer based on a majority vote combination of three neural classifiers. Two of the networks have the same architecture, but the initial random weights are different. The third network is fed with a smaller resolution digit image. A neural network can be used for implementing the combiner (Fig. 6 (c)). Lee and Srihari propose a Decision Combination Neural Network (DCNN) [76]. In the DCNN the input neurons (outputs of individual classifiers) are directly connected with output neurons. In this way there is a clear meaning for the connections between neurons, that identify the contribution to the overall decision of each classifier. A parallel classifier can be designed relying only on neural networks (Fig. 6 (d)). An example of this architecture for the recognition of unconstrained handwritten digits is proposed by Mui *et al.* [77]. The network contains ten small independent sub-nets, each of which is responsible for a particular class. The sub-nets are fed in parallel, and the combiner implements a “winner takes all” strategy. Independent networks dedicated to separate classes are considered also in autoassociator-based classifiers (Fig. 6 (e)). When the number of classes is high, the use of a single large MLP can give rise to generalization problems due to the high number of output units. In order to improve the recognition performance of the classifier, prior knowledge of the classes can be taken into account. Characters, for example, can be split into upper-case, lower-case, digits, and special symbols; a gating network can activate the appropriate classifier (Fig. 6 (f)). In [78] this architecture is contemplated for building a two-stage OCR system. In the first step a gating network with four outputs selects the specialized networks that will be used for more accurate classification. One or more networks are activated, and the “winner” character is selected among all the specialized networks. Likewise, in [79, 80], a gating network pre-sorts Hangul syllables into six global types.

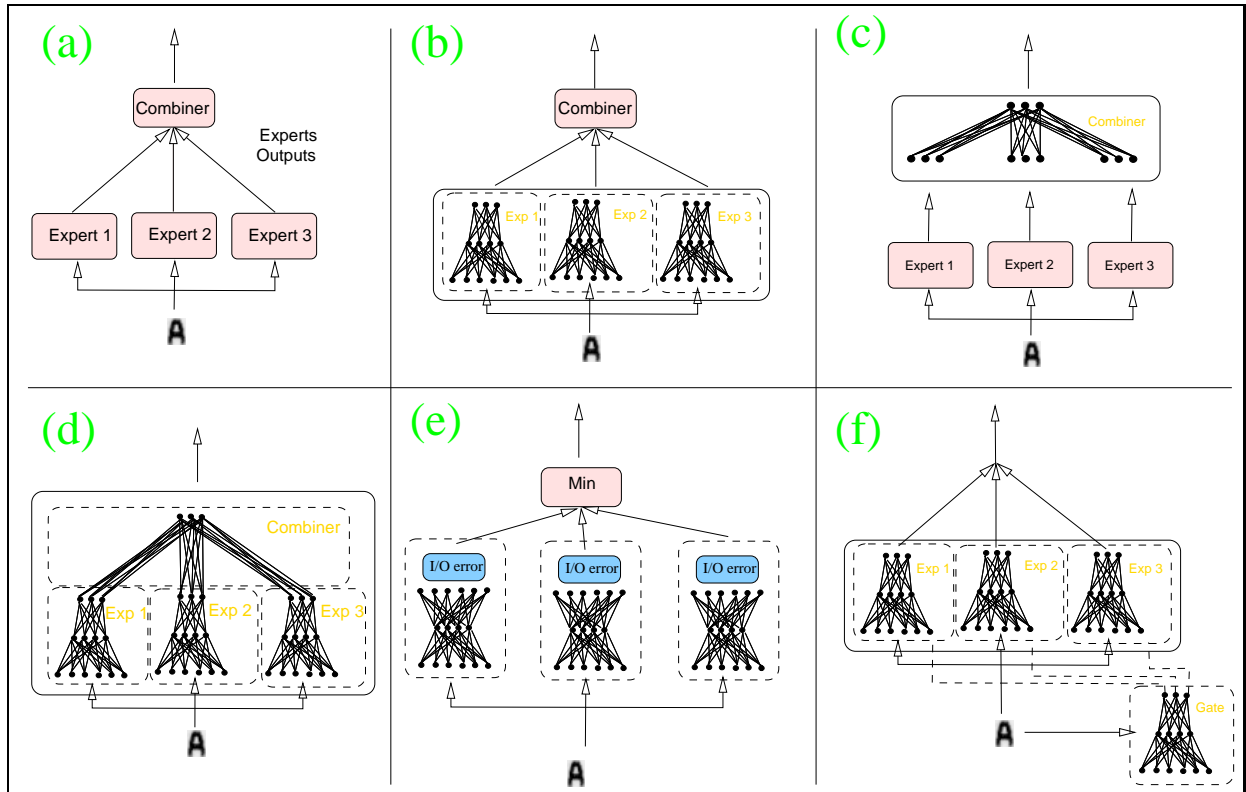


Figure 6: The use of neural networks in modular parallel classifiers for OCR. (a) represents the general architecture containing a set of experts and a combiner. (b-d) show the use of neural classifiers for the individual experts, the combiner or both modules. The autoassociator-based classifier is represented in (e). A gating network, activating different modules, is shown in (f).

In serial combinations, the decision concerning the selection of subsequent classifiers can either be made on-line, or off-line. In the first case the classifier to be activated depends on the input pattern. In the second case the architecture is defined during the learning process of the overall classifier. The outputs of an MLP-based classifier trained with one-hot coding can be used for selecting the most appropriate verification modules in a serial combination. Two examples of this approach are proposed by Takahashi and Griffin [81], and by Francesconi *et al.* [82]. In [81], after the preliminary classification by one MLP, an additional verification is made considering the three top classes. The latter is performed as a linear tournament: the one-to-one comparison between the first and second candidate classes is carried out, and the “winner” is verified with the third candidate. In the absence of a one-to-one verifier, the class with the higher recognition rank is the winner by default. One limit of this approach is the high number of networks to be trained, however only a few verifiers are needed to cover most errors. A serial combination of an MLP and an autoassociator-based classifier is described in [82]. The MLP is the first classifier of the combination and has a twofold role. First, it acts as a filter for the classifier combination, since the autoassociators are invoked only for symbols rejected by the MLP. Second, it selects the classes to be considered by the second classifier, to reduce the number of autoassociators to be activated. Basically, the autoassociators are used when the MLP rejects a pattern and the classes considered are those corresponding to the MLP outputs greater than a threshold value. The threshold is computed as a percentage of the highest MLP output in order to adapt the class selection to the MLP’s recognition confidence.

Structure adaptation methods automatically adjust the network structure to the uneven distribution of classes in the pattern space. These methods are very suitable when designing a classifier with many classes. Clustering algorithms are frequently used for grouping together most similar classes. Some methods analyze the confusion table of the first classifier in order to find most confused classes. Examples of this approach are proposed in [83], where maxima in the confusion table are searched, and in [84], where the confusion table of the first stage, a feature-based OCR, is manually analyzed. Nine sets of most confused classes are found (e.g. {S,5,6}; {B,D,O,8}), and recognized in the second stage by appropriate MLPs. Alternatively, clustering algorithms can be applied to the characters belonging to the learning set. For instance Su *et al.* [85] group together similar characters by means of a clustering algorithm, and organize the classifiers so that most indistinguishable classes (e.g. “4” and “A”) are recognized by the last network. Each network is subsequently trained to recognize the patterns belonging to its classes, and to reject patterns that should be recognized by subsequent classifiers.

Competitive learning is also based on clustering, and frequently taken into consideration when the number of classes is high [86, 87, 88, 89]. In a particular case a structure adaptation method was adopted for the recognition of Korean characters using a self organizing neural tree [86]. The basic

Method	Modular architecture	Architecture of experts	Architecture of combiner	Peculiarities	Refs.
Parallel networks	Parallel	MLP	<i>Voting</i>	Different initial weights	[68, 92]
Neural decision	Parallel	—	MLP with two layers	No hidden layers in the decision network	[76]
Cluster network	Parallel	MLP		Connectionist combination	[93]
Single class networks	Parallel	MLP		Networks are trained independently	[77]
Autoassociators	Parallel	MLP acting as autoassociator	<i>Minimum I/O error</i>	Invariant recognition	[94, 70]
Partitioning of classes	Gated Parallel	MLP	<i>Selected by gating network</i>	Separated networks for different classes	[79, 78]
Verification	Serial	MLP	—	Discriminators between 2 or 3 classes	[81, 95, 84]
Verification with autoassociators	Serial	MLP and autoassociators	—	Combines a parallel classifier with a serial one	[82]
Automatic selection of classes	Serial	MLP	—	Structure adaptation clustering of classes	[85, 83]
Neural tree	Hierarchical	SOM	—	Structure adaptation	[86]

Table 4: Main features of neural modular classifiers for OCR. Cursive descriptions in the combiner column refer to non-connectionist methods; this column is obviously empty for serial and hierarchical combinations.

idea is to automatically find a network structure and size suitable for the classification of large-set and complex patterns. The tree-structured network is based on sub-networks that are logically connected to nodes in the previous level. In fact sub-networks define with higher resolution regions of the pattern space containing more classes. In this architecture, structure adaptation is based on three operations: sub-network generation, merging of similar nodes, and deletion of nodes that have not been activated for a long time during learning. Another hierarchical structure adaptation SOM for the recognition of handwritten digits is proposed in [90]. It is worthwhile to observe that node deletion in SOM structure adaptation methods is strictly related to pruning methods in MLPs (e.g. [91] for handwritten characters or [34] for applications in layout analysis).

6.4 Discussion

Optical character recognition (either printed or handwritten) in its off-line version has been one of the first benchmarks for neural networks trained by supervised learning algorithms. As a result of this interest, most papers describing connectionist techniques in document image analysis propose solutions to this problem or to related ones, like graphic symbol recognition. Some authors have analyzed connectionist-based OCRs by comparing the performance of various neural and non-neural classifiers on some data-sets (e.g. [1, 2, 3]). The influence of the training set size on the generalization capabilities of MLPs trained for the recognition of handwritten characters was also evaluated in several papers ([1, 2, 96, 97]). Various connectionist models dealing with size-normalized images of handwritten digits have been compared by LeCun *et al.* [2]. In [97], Martin and Pittman examined the generalization of

MLPs trained on a data-set containing handwritten characters; the results clearly show that recognition performance increases when the learning set size increases.

In the near future we can expect that the recent interest in modular classifiers will be extended also to the neural-based classifiers analyzed in Section 6.3. Also in this case OCR problems will be used as first benchmarks for new architectures and learning mechanisms. Structural descriptions have been already considered for symbol recognition, and appropriate encodings have been taken into account (Section 6.2). These descriptions are likely to be used for testing novel approaches to connectionist-based processing of structural descriptions.

7 Word recognition

The segmentation-based approach to word recognition reaches its limits when the location of segmentation points is impossible or unreliable, as in cursive handwriting. One alternative is to use a holistic-based word recognition approach, where the words of a small dictionary are recognized as single units. When the problem at hand requires a larger lexicon, then segmentation-based reading is appropriate, nevertheless segmentation requires some feedback from recognition. Integrated segmentation and recognition (ISR) techniques are related to the contextual development of segmentation and recognition modules. IRS takes three main approaches into account: Heuristic Over Segmentation (HOS), Time-Delay Neural Networks (TDNN) and Space Displacement Neural Networks (SDNN), which are closely related with TDNN, and therefore will be analyzed in the same Section.

7.1 Holistic word recognition

Holistic word recognition (Fig. 7 (a)) is effective when the words belong to a small lexicon. When reading bank checks the number of basic words required to fill an amount is limited to a few dozen, depending on the language. For instance, 32 words are needed for writing legal amounts of checks written in English [98], whereas a set of 30 words can be used for French checks [99]. When individual words in the cursive script can be segmented we can presume to work with a finite lexicon, and use holistic approaches.

Recently, an MLP-based holistic word recognition method was proposed by Kim *et al.* [98] for the recognition of cursive words in handwritten checks. The MLP input is based on zoning and crossing features. Zoning features are the number of black pixels in each region. Crossing features describe the number of crossings between the word strokes and 25 uniformly spaced horizontal and vertical lines. The 32 classes are encoded with a one-hot representation. Amin and Mansoor [100] proposed a similar approach for the recognition of Arabic words.

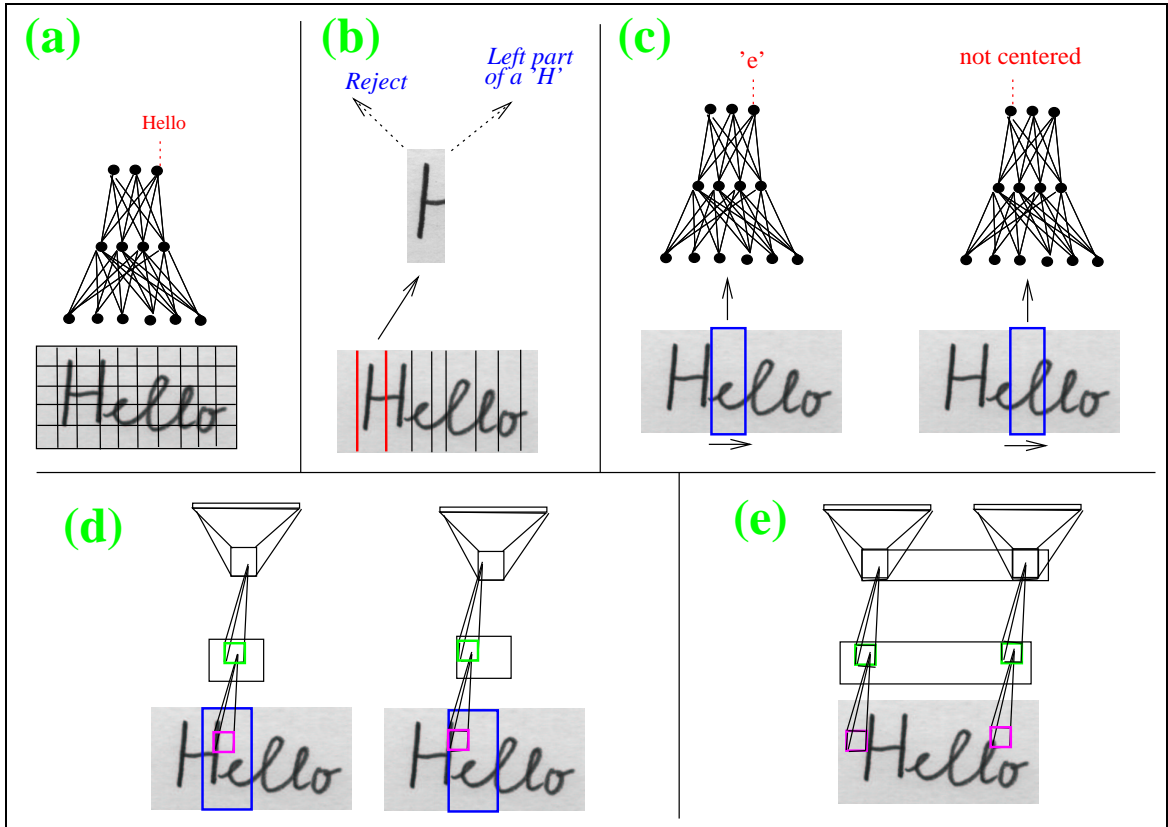


Figure 7: Some approaches to word recognition. (a) Holistic word recognition where the word image is the input, and each output is assigned to a word class. (b) Two possible interpretations of broken characters in HOS. (c) “Non centered” output in TDNN. (d) In convolutional TDNN, the same computation can take place several times when the network is shifted horizontally. (e) In SDNN the network is replicated over the input in the horizontal direction.

Dodel and Shinghal [101] propose the integration of a symbolic classifier and an MLP for check reading. The symbolic classifier recognizes the words by feeding the MLP with normalized images of words or parts of words. The MLP is trained to recognize 19 patterns corresponding to words (such as “one”, “six”, “nine”) or portions of words (e.g. “ree”, or “hree” from the word “three”).

Keyword spotting is another application where holistic word recognition can be appropriate. The objective of keyword spotting is to retrieve keywords in document images using image properties only [102]. Keyword spotting can be considered as a verification problem and autoassociator-based classifiers are appropriate. Cesarini *et al.* [103] locate keywords in printed documents by learning an autoassociator-based recognizer for each keyword. The method allows the location of words containing touching characters. Nevertheless words printed in different fonts generally require different autoassociators.

Word-level information (e.g. the font of a word) can be useful for word recognition. Font classification is proposed by Jung *et al* [104], and the method is based on the recognition of the “blobs” (upper and lower parts of characters) in the word. Different types of blobs can be considered for a given font: for instance in Courier there are 3 ascenders, 4 descenders, and 7 serifs. Each MLP output is assigned to

each kind of blob (for 7 fonts a total of 78 outputs are considered), whereas the input is a vector with 36 values. Each value indicates the number of pixels having a given local slope (possible values are 0, 45, 90, 135 degrees) in each region obtained by superimposing a uniform 3 by 3 grid to the blob image.

7.2 Heuristic over segmentation

In heuristic over segmentation, a segmentation algorithm is applied to a word image to locate a large number of candidate cutting points. Subsequently, a recognizer is employed to score the alternative segmentations generated and to find the best character sequence. The basic idea behind this approach is to over-segment the word in the hope of including all the correct segmentation points among the proposed ones. The best interpretation of the word relies on an overall optimization of a cost that is based on the recognition of “potential” characters. Dynamic programming approaches and Hidden Markov Model (HMM)-based methods are frequently taken into account for optimizing the recognition in relation to a given lexicon [105].

An example of the integration of HMM and neural network in HOS is the word recognizer proposed by Knerr and Augustin [106]. In this method, the letters are decomposed into graphemes, which are described by appropriate features, and an MLP is used for computing each grapheme’s posterior class probability. The word recognizer is part of a check reading system which operates using a reduced lexicon. Consequently, each word can be modeled by an independent HMM, which can be used to compute the likelihood for the corresponding word class. A similar integration of HMM with neural network is described in [99], where a hybrid HMM/RBF system is proposed for the recognition of handwritten words. The HMM models the alignment of letters onto segments produced by the segmentation step; the RBF network is used in order to estimate the emission probabilities associated with Markov states from the bitmaps of segments. Two main approaches can be considered for word recognition using HMMs. In the model discriminant approach we consider a separate HMM for each class, whereas in the path discriminant approach a unique HMM is used for modeling all the word classes (e.g. [107, 108]). The model discriminant approach can be considered when dealing with small dictionaries [99, 105]).

When using neural classifiers in HOS an appropriate training strategy must be adopted for dealing with broken characters due to segmentation errors. Two main strategies can be considered (Fig. 7 (b)). The first approach assumes that the correct segmentation points are definitely among the proposed ones. In this case a broken character will be rejected as noise by the classifier, to look for another combination of cutting points that will produce the correct character image. In the second approach the maximum information possible is extracted from erroneously segmented characters. To this purpose appropriate sub-classes are introduced for each character: in the presence of a broken symbol, the classifier should label it as a cut symbol of the correct class.

An example of the first approach is illustrated by Bromley and Denker [109], who propose using an MLP-based classifier trained to reject images produced by erroneous segmentations (“rubbish”). Training on rubbish did not cause degradation when the neural network was tested on good digits only, while it improved the ability of the network to reject cut symbols. The paper by Gilloux *et al.* [99] exploits the second approach and proposes a system for the recognition of cursive words in French checks. One RBF is used to classify word segments that correspond to complete and broken characters. The output layer contains three neurons for each character, corresponding to character segmentations (left part, right part, complete). This method provides different labels for complete or cut characters. A different strategy relies on training MLPs with fuzzy outputs instead of crisp ones (corresponding to the classic one-hot coding). In this case the output is forced to represent ambiguities in characters to be recognized. Gader *et al.* [110] compare crisp and fuzzy neural networks for the recognition of handwritten words in a HOS-based approach. The basic approach is based on training the network to reflect the ambiguity of a cut “o” that could be interpreted as a “c”. For this reason, the target output assigned to each training sample contains non-zero values not only for the “right” class, but also for similar classes. The similarity is estimated by selecting the closest samples in the pattern space: each target output is computed considering the percentage of neighboring characters in the corresponding class.

7.3 Time Delay Neural Network

Time-Delay Neural Networks (TDNN) are used to deal with temporal sequences. The output of a TDNN depends on its current and previous inputs, which are delayed by one or more time units. If the input signal is a vector of m values and we take a delay of n time units into account, then a TDNN can be implemented with an MLP having $n \cdot m$ input units ([111] page 256). TDNNs have been used for on-line word recognition: in this case the meaning of “time” is quite straightforward. In off-line word recognition the horizontal axis in the window containing the word is considered as a temporal scale. Even in the case of TDNNs, the recognition output provided by the neural network must be processed to compute the most likely word. HMMs are frequently used for this task, and in this case a path discriminant approach is usually considered for dealing with larger lexica (e.g. [112]).

Several strategies have been proposed for the actual implementation of the “scanning window” in TDNN, as well as for the information provided by the neural network at each position.

In the approach proposed by Martin [113] a fixed-size window scans a word horizontally, and the pixels of the window feed a TDNN (Fig. 7 (c)). One output neuron is active when the window covers portions of two digits (the “non-centered” output). The other output neurons describe the digit membership with a one-hot encoding when the window is centered over a digit. During training, when the center

of the window is close to one digit center, the output of the corresponding class is set to its maximum value. The “non-centered” output gets its maximum value when the center of the window is close to the halfway point between two character centers. Between these two extremes, the target values vary linearly with distance, creating a trapezoidal function. The experiments are made on a subset of a NIST database containing handwritten fields whose character classes are known from the ground-truth, whereas the horizontal center of each digit is set by hand.

A similar method is described in [114]. In this case the network architecture is based on recurrent neural networks, and the sliding window is moved in steps of 2 pixels over the input image to speedup the process.

From a computational point of view, a window scanning all the positions of the input word is not very efficient, since most positions correspond to non-centered points. To solve this problem, in [115] the neural network is trained not only to determine whether a character is centered, but also to make corrective jumps (saccades) to the current and to the next characters. The network output contains four parts. The first two parts are a “non-centered” node, and a set of nodes corresponding to character classes (similarly to [113]). In addition, two groups of nodes encode the distance (in pixels) to the current and to the next characters. As pointed out by Shustorovich and Thrasher [116] the last two groups of nodes do not behave linearly over small displacements in the input. When the input frame center moves across the midpoint between two characters the next character becomes the current one and the output encoding suddenly changes. In [116], in order to solve this problem, a group of output neurons describe the character positions in the window. This group replaces both the “non-centered” output and the information about the character position. The classification is performed in a subsequent step by a second network. This approach looks very similar to segmentation methods described in Section 5.2. The main difference is that in [116] the characters are not actually cut from the input image, instead a portion of the image centered on a character (containing also parts of neighboring characters) is fed to the classifier.

One of the main drawbacks of Time-Delay Neural Networks is their computational cost, since the network needs to be activated at every position in the input image. When TDNNs are based on a convolutional architecture, then *Space Displacement Neural Networks* (SDNN) are a good alternative (e.g. [117]). Two sub-networks of a convolutional network can look at identical locations on the input image when the TDNN is horizontally shifted (Fig. 7 (d)). Since convolutional sub-networks share the weights, the output of the two sub-networks (at different positions) will be the same. To reduce the computational burden, it is possible to build a larger convolutional network whose structure is identical to the original network, with the exception that the feature maps are expanded in the horizontal dimension. In the single character network, the output layer is a vector that is connected to a zone of

Method	Category	Input to neural network	Output of neural network	Peculiarities	Refs.
Words recognized as single entities	Holistic	Subsampled word image	One-hot coding of classes	Words belong to a reduced lexicon	[100, 99]
Autoassociators	Holistic	Subsampled word image	Similarity of the image with the word class	One network for each word	[103]
Font information	Holistic	Useful for word recognition	[104]		
MLP trained on "rubbish"	HOS	Subsampled character or broken character	One-hot coding with reject	Cut characters are rejected	[109]
MLP trained with cut characters	HOS	Subsampled character or broken character	Three outputs for each class	Cut characters have their own class	[99]
Fuzzy MLP	HOS	Subsampled character or broken character	One output for each class	Learning constrained to represent ambiguities	[110]
Positioning information coded in MLP	TDNN	Window moving horizontally over the word	One output encodes "not-centered" position	Not-centered positions equivalent to an additional class	[113, 114]
Saccadic MLP	TDNN	Window moved according to MLP output	Position of near characters	Jumps to current and next character	[115, 116]
Large convolutional neural networks	SDNN	Word image	Character classes at various positions	Complexity reduced w.r.t. convolutional TDNN	[117]

Table 5: Neural network approaches to word recognition.

width W in the last feature layer. If this network is replicated over the input in the horizontal direction, then the output layer will be replicated, and each output vector will be connected to a different zone of width W in the last feature layer (Fig. 7 (e)). In this way we obtain a group of output vectors each encoding the class membership of a character at the corresponding position in the input window. Since the width of handwritten characters is highly variable, more groups of output vectors are placed at different horizontal positions, and connected to different width zones of the last feature layer. In [117], for example, four groups of output vectors are built. The output vectors are processed by the Viterbi algorithm which selects the group of output vectors providing the highest score.

7.4 Discussion

Word recognition approaches are tightly connected with other tasks described in this survey. Holistic word recognition is related to methods used for isolated character recognition, and zoning is frequently considered as feature extraction. A significant difference concerns the number of classes that must be managed by the classifiers. In OCR the classes are pre-defined and known in advance, whereas a limited lexicon can be taken into account for word recognition only in some applications. The basic idea of TDNNs, which are based on a sliding window feeding an MLP, is very similar to the sliding window approach that is used for character segmentation (Section 5). The main difference (and the reason why we analyze the two approaches in different Sections) is the purpose of the classification performed by the "moving" neural network. In character segmentation the network is used for deciding whether the corresponding point is a cutting location or not, whereas in the case of TDNNs described above the network provides a character classification in addition to the cutting information.

Similarly to OCR applications, the use of connectionist architectures for word recognition is a mature field well investigated, and with several effective approaches available. Improvements in the future will probably be related with the application of modular architectures also in this domain, and with the application of neural architectures to challenging problems like cursive text recognition. Like for signature verification, the use of autoassociator-based classifiers and RBF networks is essential for dealing with problems of verification.

8 Signature verification

Signature identification and verification are very important topics in document processing systems (e.g. in check reading applications), that has been the subject of several surveys analyzing also neural approaches (e.g. [118, 119, 120]).

Signature *identification* deals with assigning a signature to a class (corresponding to a given writer). Signature *verification* aims at verifying the correctness of the signature provided by an individual. Both problems are of difficult solution due to the high level of shape variation of signatures in a given class. Most work has been done concerning signature verification. In this section we focus attention on the connectionist approaches to off-line signature verification.

Four classes of forgeries are usually considered having increasing levels of difficulty with regard to their automatic identification [121, 122]: *random forgeries* are produced without knowing the name of the signer; in *simple forgeries* the name is known, but the shape of the correct signature is unknown to the counterfeiter; this shape is known, on the other hand, when producing *skilled forgeries*; whereas *tracing forgeries* are obtained by carefully copying the original signature.

Since training sets are typically quite small, and the variability of signatures in a class is quite high, prior knowledge is often used for designing feature extraction algorithms. Both global and local features can be taken into account; global features are more appropriate when dealing with random forgeries, whereas local features are useful for more difficult forgeries [122]. A possible solution for increasing the number of training samples is the use of artificially generated samples (e.g. [123]).

Many connectionist approaches have considered the problem of recognizing random forgeries as a preliminary step to a more complete verification system. In random forgeries verification the class of genuine signatures is denoted by ω_1 , whereas random forgeries are denoted by ω_2 . In this case the training set can be built up simply by collecting signatures of different writers, without requiring actual forgeries. A simple neural approach to the recognition of random forgeries is based on the use of a feature vector-fed MLP having two outputs, one which corresponds to the class ω_1 , and the other which corresponds to the class ω_2 (e.g. [121]).

In a data-set where each signer provides the same number of samples, the number of signatures belonging to class ω_1 is lower than the number of patterns of class ω_2 . In this case training samples of the ω_2 class are usually randomly selected from the signatures of other writers, so as to obtain a balanced data set of positive and negative samples. Plamondon and Lorette [124] as well as Murshed *et al.* [125], point out the limits of a random selection of training samples from patterns of the ω_2 class. To solve these problems, random forgeries are identified without prior knowledge of class ω_2 [125]. In the proposed approach, the system is trained using only the genuine signatures of one writer. Basically the signature is divided into m equal-size regions, each assigned to a Fuzzy ARTMAP network which acts like an expert examining a region of the signature. Appropriate regions are extracted using a writer-specific grid. A combination of SOM-based classifiers and MLPs is proposed in [126] in order to select patterns belonging to class ω_2 . Initially two SOMs are applied to the input signature to group together the most similar signatures corresponding to one neuron of the SOM output layer. Subsequently a MLP classifier is used to verify a given signature. It is worthwhile noting that the training samples of class ω_2 correspond to signatures that are clustered together by the SOM.

One possible way for improving the performance of signature verification systems is the use of modular architectures. Parallel architectures in which both experts and the combiner are neural are described in [123, 127]. In both systems each expert is an MLP having a different set of features as input: the three MLPs used in [123] are fed with local shape features at different resolutions; the three MLPs used in [127] are fed with moment features, upper and lower envelopes of the signature. Combiners are an MLP [123] and a set of adalines, one for each signer [127]. Features used for the identification of random and skilled forgeries are different, since in the first case the measurements must be able to identify large differences, whereas in the latter case only small details must be detected. This property can be incorporated by designing a verifier that is based on a serial combination of classifiers. In this approach the first module of the cascade is dedicated to the detection of random forgeries, whereas the latter one is used to identify simple/skilled forgeries (e.g. [121, 122]).

8.1 Discussion

Two related problems affect the design of signature verification systems: the collection of training samples and the choice of classifiers that must be able to discriminate between ω_1 and ω_2 classes. In particular we must emphasize the limits of MLP-based classifiers when used for classification purposes. As discussed in [128] MLPs are not adequate for applications of pattern recognition requiring a reliable rejection and, especially, they are not adequate for pattern verification tasks. To this purpose other architectures like autoassociator-based classifiers and RBF networks seem to be more appropriate. The comparison of MLP-based classifiers with other neural architectures more effective in verification prob-

lems should consequently be investigated in the near future, together with other modular architectures.

9 Conclusions

Artificial neural networks have been massively used for many tasks in document image processing. ANNs are frequently presented as “universal methods” able of solving many tasks with little changes. This is partially true for some applications of Optical Character Recognition, where simple neural architectures have been proven to provide satisfactory results. However, successful applications take the prior knowledge of the domain into account in the design of neural document processing systems.

Most work was devoted to the well known problem of isolated character recognition (OCR), either printed or handwritten. In this domain, many years of research have demonstrated that static neural networks can be used for recognizing characters with satisfactory results. Recently, more emphasis has been put to the development of modular neural classifiers, in order to increase the recognition performance. Modular classifiers have been used also in other document processing tasks, and with the continuously increase of computation capabilities, they appear to become more and more attractive.

Word recognition is another task where neural networks have been extensively used with excellent results. In this domain, successful applications came out from appropriate integrations of ANNs with other techniques capable of handling different interpretations of character sequences. Hidden Markov Models (HMM) are useful techniques which have been applied to word recognition also independently from the use of ANNs.

On the other hand, neural networks have been used in order to solve problems in nearly all the document processing tasks. However in the latter case most approaches rely on the use of simple MLPs, and often appropriate motivations for the choice of both the model and the architecture are missing. Worth of attention exceptions have been analyzed in the paper, showing the links existing between approaches used in different domains.

The confidence values which are the outputs of MLP are frequently taken into account for rejecting dubious patterns. Two alternatives for handling outliers are RBF, and autoassociators. The advantages of using RBF for region classification in layout analysis have been analyzed in Section 4, and similar conclusions for the case of OCR have been discussed in Section 6. Autoassociators have been used also for both pixel classification in layout analysis as well as for graphic symbol recognition. Another way for dealing with problems of uncertainty concerning labels assigned to classes is to use self-organizing approaches that are capable of discovering “new” classes not taken into consideration during learning (Section 4). Pattern rejection is also very important in signature verification applications.

One peculiarity of the use of neural networks in pattern classification is that frequently the feature

extraction can be implicitly performed by the network itself. This quality is extensively considered in methods related to symbol and character recognition (Section 6). However, by using pruning algorithms, the users of these classifiers can implement a feature selection mechanism dealing with “hidden” features in pixel classification as well.

Another issue which is relevant in most document processing tasks, is the collection of training samples. Neural network performances heavily depends on the amount of training data, and to their similarity to actual data coming from real applications. The problem has been solved in OCR by manually collecting huge amounts of data in standard data-sets which are currently widely available. Other document processing tasks require a special care in the collection of training samples, and standard data-sets are not yet available for tasks like character segmentation. In some domains the only solution was the use of artificial generation of training samples. However, in the latter case a special attention has to be paid to the generalization to actual data of neural networks trained on artificial data. We expect that the use of artificially generated training samples will increase in the next years, similarly to other research fields in pattern recognition.

Logical labeling of text blocks can be performed simply by taking the context of the blocks into account. For this purpose recurrent neural networks have been used in order to deal with these sequences. Higher level tasks, such as page classification, require dealing with more complex structures (e.g. trees and graphs); for these cases recursive neural networks represent some of the most promising methods to be taken into consideration in future applications.

In our opinion, the most challenging tasks for the future research are an extended use of modular architectures in all the tasks, and the use of connectionist models able to deal with structural information, which are very common in various document processing tasks.

Acknowledgments

We thank all the members of the DANTE research group for their support to most of the research we have been carrying out in the field. We are also very grateful to the participants of the tutorial “Artificial Neural Networks for Document Analysis and Recognition” (ICDAR 2001 - Seattle, September 2001, and ICPR 2002 - Québec, August 2002) for their constructive questions and comments.

References

- [1] D. J. Burr, “Experiments on neural net recognition of spoken and written text,” *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1162–1168, 1988.

- [2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [3] M. D. Garris, C. L. Wilson, and J. L. Blue, "Neural network-based systems for handprint OCR applications," *IEEE Trans. Image Processing*, vol. 7, no. 8, pp. 1097–1112, 1998.
- [4] R. Kasturi and L. O’Gorman, "Document image analysis: A bibliography," *Machine Vision and Applications*, vol. 5, no. 5, pp. 231–243, 1992.
- [5] L. O’Gorman and R. Kasturi, *Document Image Analysis*. Los Alamitos, California: IEEE Computer Society Press, 1995.
- [6] G. Nagy, "Twenty years of document image analysis in PAMI," *IEEE Trans. PAMI*, vol. 22, no. 1, pp. 38–62, 2000.
- [7] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE Trans. PAMI*, vol. 18, no. 7, pp. 690–706, 1996.
- [8] R. Plamondon and S. N. Srihari, "On-line and off-line handwriting recognition: a comprehensive survey," *IEEE Trans. PAMI*, vol. 22, no. 1, pp. 63–84, 2000.
- [9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.
- [10] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4–22, 1987.
- [11] B. Widrow, "30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation," *IEEE Trans. Neural Networks*, vol. 78, no. 9, pp. 1415–1442, 1990.
- [12] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Trans. Neural Networks*, vol. 9, no. 5, pp. 768–786, 1998.
- [13] P. Frasconi, M. Gori, and A. Sperduti, "Guest editors’ introduction: special section on connectionist models for learning in structured domains," *IEEE Trans. Knowledge and Data Engineering*, vol. 13, no. 2, pp. 145–147, 2001.
- [14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing* (D. E. Rumelhart and J. L. McClelland, eds.), vol. 1, ch. 8, pp. 318–362, Cambridge: MIT Press, 1986.

- [15] T. Kohonen, “The self-organizing map,” *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
- [16] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge: MIT Press, 1986.
- [17] G. Cortelazzo, G. A. Mian, G. Vezzi, and P. Zamperoni, “Trademark shapes description by string-matching techniques,” *Pattern Recognition*, vol. 27, no. 8, pp. 1005–1018, 1994.
- [18] P. Frasconi, M. Gori, A. Kuechler, and A. Sperduti, “From sequences to data structures: Theory and applications,” in *A field guide to dynamical recurrent networks* (J. Kolen and S. Kremer, eds.), New York: IEEE Press, 2001. Chapter 19.
- [19] J. Liang, R. Haralick, and I. T. Phillips, “Document image restoration using binary morphological filters,” in *Proc. SPIE - Doc. Rec. III*, pp. 274–285, 1996.
- [20] Y. Zhang, R. P. Loce, and E. R. Dougherty, “Document restoration and enhancement using optimal iterative and paired morphological filters,” in *Proc. SPIE - Doc. Rec. IV*, pp. 109–123, 1997.
- [21] M.-Y. Yoon, S.-W. Lee, and J. Kim, “Faxed image restoration using Kalman filtering,” in *Proc. 3rd Int’l Conf. Doc. Anal. Rec.*, pp. 677–680, 1995.
- [22] A. P. Whichello and H. Yan, “Linking broken character borders with variable sized masks to improve recognition,” *Pattern Recognition*, vol. 29, no. 8, pp. 1429–1435, 1996.
- [23] P. Stubberud, J. Kanai, and V. Kalluri, “Adaptive image restoration of text images that contain touching or broken characters,” in *Proc. 3rd Int’l Conf. Doc. Anal. Rec.*, pp. 778–781, 1995.
- [24] N. Billawala, P. E. Hart, and M. Peairs, “Image continuation,” in *Proc. 2nd Int’l Conf. Doc. Anal. Rec.*, pp. 53–57, 1993.
- [25] P. Martin and C. Bellisant, “Low-level analysis of music drawing images,” in *Proc. 1st Int’l Conf. Doc. Anal. Rec.*, pp. 417–425, 1991.
- [26] N. Rondel and G. Burel, “Cooperation of multilayer perceptrons for the estimation of skew angle in text document images,” in *Proc. 3rd Int’l Conf. Doc. Anal. Rec.*, pp. 1141–1144, 1995.
- [27] R. Palaniappan, P. Raveendran, and S. Omatu, “New invariant moments for non-uniformly scaled images,” *Pattern Analysis and Applications*, vol. 3, no. 2, pp. 78–87, 2000.
- [28] P. Ahmed, “A neural network based dedicated thinning method,” *Pattern Recognition Letters*, vol. 16, no. 6, pp. 585 – 590, 1995.

- [29] K. Chen, D. Wang, and X. Liu, "Weight adaptation and oscillatory correlation for image segmentation," *IEEE Trans. Neural Networks*, vol. 11, no. 5, pp. 1106–1123, 2000.
- [30] J. Fisher, S. Hinds, and K. D'Amato, "A rule-based system for document image segmentation," in *Proc. 10th Int'l Conf. Pattern Recognition*, pp. 567–572, 1990.
- [31] L. O'Gorman, "The document spectrum for page layout analysis," *IEEE Trans. PAMI*, vol. 15, no. 11, pp. 1162–1173, 1993.
- [32] G. Nagy and S. Seth, "Hierarchical representation of optically scanned documents," in *Proc. 7th Int'l Conf. Pattern Recognition*, pp. 347–349, 1984.
- [33] T. Watanabe, Q. Luo, and N. Sugie, "Structure recognition methods for various types of documents," *Machine Vision and Applications*, vol. 6, no. 6, pp. 163–176, 1993.
- [34] A. K. Jain and Y. Zhong, "Page segmentation using texture analysis," *Pattern Recognition*, vol. 29, no. 5, pp. 743–770, 1996.
- [35] C. Strouthopoulos and N. Papamarkos, "Text identification for document image analysis using a neural network," *Image and Vision Computing*, vol. 16, no. 12/13, pp. 879–896, 1998.
- [36] D. X. Le, G. R. Thoma, and H. Wechsler, "Classification of binary document images into textual or nontextual data blocks using neural network models," *Machine Vision and Applications*, vol. 8, no. 5, pp. 289–504, 1995.
- [37] S. Imade, S. Tatsuta, and T. Wada, "Segmentation and classification for mixed text/image documents using neural networks," in *Proc. 2nd Int'l Conf. Doc. Anal. Rec.*, pp. 930–934, 1993.
- [38] T. Taxt, P. J. Flynn, and A. K. Jain, "Segmentation of document images," *IEEE Trans. PAMI*, vol. 11, no. 12, pp. 1322–1329, 1989.
- [39] K. Etemad, D. S. Doermann, and R. Chellappa, "Multiscale segmentation of unstructured document pages using soft decision integration," *IEEE Trans. PAMI*, vol. 19, no. 1, pp. 92–96, 1997.
- [40] K. Nakamura, S. Yamamoto, and T. Itoh, "Document image segmentation into text, continuous tone and screened-half-tone region by the neural networks," in *IS&T / SPIE*, pp. 358–361, 1998.
- [41] K. Y. Wong, R. G. Casey, and F. M. Wahl, "Document analysis system," *IBM J. Research and Development*, vol. 26, no. 6, pp. 647–656, 1982.
- [42] D. Wang and S. N. Srihari, "Classification of newspaper image blocks using texture analysis," *Computer Vision, Graphics, and Image Processing*, vol. 47, no. 3, pp. 327–352, 1989.

- [43] K.-C. Fan and L.-S. Wang, "Classification of document blocks using density features and connectivity histogram," *Pattern Recognition Letters*, vol. 16, pp. 955–962, 1995.
- [44] F. Y. Shih and S. S. Chen, "Adaptive document block segmentation and classification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 26, no. 5, pp. 797–802, 1996.
- [45] G. Sainz Palmero and Y. Dimitriadis, "Structured document labelling and rule extraction using new recurrent fuzzy-neural systems," in *Proc. 5th Int'l Conf. Doc. Anal. Rec.*, pp. 181–184, 1999.
- [46] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: a neural network architecture for incremental learning of analog multidimensional maps," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 698–713, 1992.
- [47] W. P. de Waard, "Neural techniques and postal code detection," *Pattern Recognition Letters*, vol. 15, no. 2, pp. 199 – 206, 1994.
- [48] S. L. Taylor, R. Fritzson, and J. A. Pastor, "Extraction of data from preprinted forms," *Machine Vision and Applications*, vol. 5, no. 5, pp. 211–222, 1992.
- [49] F. Cesarini, M. Lastri, S. Marinai, and G. Soda, "Encoding of modified X-Y trees for document classification," in *Proc. 6th Int'l Conf. Doc. Anal. Rec.*, pp. 1131–1136, 2001.
- [50] J. Lin, C.-W. Lee, and Z. Chen, "Identification of business forms using relationships between adjacency frames," *Machine Vision and Applications*, vol. 9, no. 2, pp. 56–64, 1996.
- [51] Y. Ishitani, "Flexible and robust model matching based on association graph for form image understanding," *Pattern Analysis and Applications*, vol. 3, no. 2, pp. 104–119, 2000.
- [52] S. L. Taylor, M. Lipshutz, and R. W. Nilson, "Classification and functional decomposition of business documents," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 563–566, 1995.
- [53] A. Dengel and F. Dubiel, "Clustering and classification of document structure -a machine learning approach-," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 587–591, 1995.
- [54] J. Hu, R. Kashi, and G. Wilfong, "Document image layout comparison and classification," in *Proc. 5th Int'l Conf. Doc. Anal. Rec.*, 1999.
- [55] C. K. Shin and D. S. Doermann, "Classification of document page images based on visual similarity of layout structures," in *Proc. SPIE - Doc. Rec. Retr. VII*, pp. 182–190, 2000.

- [56] J. Yuan, Y. Y. Tang, and C. Y. Suen, “Four directional adjacency graphs (FDAG) and their application in locating fields in forms,” in *Proc. 3rd Int’l Conf. Doc. Anal. Rec.*, pp. 752–755, 1995.
- [57] J. Wang and J. Jean, “Segmentation of merged characters by neural networks and shortest path,” *Pattern Recognition*, vol. 27, no. 5, pp. 649–658, 1994.
- [58] Z. K. Lu, Z. Chi, and W. C. Siu, “Length estimation of digits strings using a neural network with structure based features,” *SPIE/IS&T Journal of Electronic Imaging*, vol. 7, pp. 79–85, January 1998.
- [59] B. Eastwood, A. Jennings, and A. Harvey, “A low level feature based neural network segmenter for fully cursive handwritten words,” in *Proc. 4th Int’l Conf. Doc. Anal. Rec.*, p. 523, 1997.
- [60] J. H. Bae, K. Jung, J. Kim, and H. Kim, “Segmentation of touching characters using an MLP,” *Pattern Recognition Letters*, vol. 19, no. 8, pp. 701–709, 1998.
- [61] S. W. Lee, D.-J. Lee, and H.-S. Park, “A new methodology for gray-scale character segmentation and recognition,” *IEEE Trans. PAMI*, vol. 18, no. 10, pp. 1045–1050, 1996.
- [62] J. Illingworth and J. Kittler, “A survey of the Hough transform,” *Computer Vision, Graphics, and Image Processing*, vol. 44, no. 1, pp. 87–116, 1988.
- [63] W. Utschick, P. Nachbar, C. Knobloch, and J. A. Nossek, “The evaluation of feature extraction criteria applied to neural network classifiers,” in *Proc. 3rd Int’l Conf. Doc. Anal. Rec.*, pp. 315–318, 1995.
- [64] S. Sural and P. K. Das, “An MLP using Hough transform based fuzzy feature extraction for bengali script recognition,” *Pattern Recognition Letters*, vol. 20, no. 8, pp. 771–782, 1999.
- [65] K. Fukushima and N. Wake, “Handwritten alphanumeric character recognition by the Neocognitron,” *IEEE Trans. Neural Networks*, vol. 2, no. 3, pp. 355–365, 1991.
- [66] K. Fukushima and S. Miyake, “Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position,” *Pattern Recognition*, vol. 15, no. 6, pp. 455–469, 1982.
- [67] G. Srikantan, S. W. Lam, and S. N. Srihari, “Gradient-based contour encoding for character recognition,” *Pattern Recognition*, vol. 29, no. 7, pp. 1147–1160, 1996.
- [68] N. W. Strathy and C. Y. Suen, “A new system for reading handwritten zip codes,” in *Proc. 3rd Int’l Conf. Doc. Anal. Rec.*, pp. 74–77, 1995.

- [69] I.-S. Oh and C. Y. Suen, "A feature for character recognition based on directional distance distribution," in *Proc. 4th Int'l Conf. Doc. Anal. Rec.*, pp. 288–292, 1997.
- [70] M. Gori, M. Maggini, S. Marinai, J. Sheng, and G. Soda, "Edge-backpropagation for noisy logo recognition," *Pattern Recognition*, vol. 36, no. 1, pp. 103–110, 2003.
- [71] H. Takahashi, "A neural net OCR using geometrical and zonal-pattern features," in *Proc. 1st Int'l Conf. Doc. Anal. Rec.*, pp. 821–828, 1991.
- [72] A. Amin, H. Al-sadoun, and S. Fischer, "Hand-printed arabic character recognition system using an artificial network," *Pattern Recognition*, vol. 29, no. 4, pp. 663–675, 1996.
- [73] L. Cordella, C. De Stefano, and M. Vento, "A neural network classifier for OCR using structural descriptions," *Machine Vision and Applications*, vol. 8, no. 5, pp. 336–342, 1995.
- [74] M. A. Eshera and K. S. Fu., "An image understanding system using attributed symbolic representation and inexact graph-matching.," *IEEE Trans. PAMI*, vol. 8, no. 5, pp. 604–617, 1986.
- [75] E. Francesconi, P. Frasconi, M. Gori, S. Marinai, J. Sheng, G. Soda, and A. Sperduti, "Logo recognition by recursive neural networks," in *Proc. 2nd Int'l Workshop GREC*, pp. 104–117, 1997.
- [76] D. Lee and S. N. Srihari, "Dynamic classifier combination using neural network," in *Proc. SPIE - Doc. Rec. II*, pp. 26–37, 1995.
- [77] L. Mui, A. Agarwal, A. Gupta, and P. S.-P. Wang, "An adaptive modular neural network with application to unconstrained character recognition," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 8, no. 5, pp. 1189–1204, 1994.
- [78] J. Mao, K. Mohiuddin, and T. Fujisaki, "A two-stage multi-network OCR system with a soft pre-classifier and a network selector," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 78–81, 1995.
- [79] S.-B. Cho, J. O. Kwon, Y. B. Kwon, and J. H. Kim, "NETeye: A Neural Network system for recognizing multi-font/multi-size Hangul (Korean script) documents," in *Proc. 1st Int'l Conf. Doc. Anal. Rec.*, pp. 812–820, 1991.
- [80] S.-B. Cho and J. H. Kim, "Recognition of large-set printed Hangul (Korean script) by two-stage backpropagation neural classifier," *Pattern Recognition*, vol. 25, no. 11, pp. 1353–1360, 1992.
- [81] H. Takahashi and T. D. Griffin, "Recognition enhancement by linear tournament verification," in *Proc. 2nd Int'l Conf. Doc. Anal. Rec.*, pp. 585–588, 20–22 October 1993.

- [82] E. Francesconi, M. Gori, S. Marinai, and G. Soda, "A serial combination of connectionist-based classifiers for OCR," *Int'l J. Doc. Anal. Rec.*, vol. 3, no. 3, pp. 160–168, 2001.
- [83] R. Y.-M. Teo and R. Shingal, "A hybrid classifier for recognizing handwritten numerals," in *Proc. 4th Int'l Conf. Doc. Anal. Rec.*, pp. 283–287, 1997.
- [84] J. Wang and J. Jean, "Resolving multifont character confusion with neural networks," *Pattern Recognition*, vol. 26, no. 1, pp. 175–188, 1993.
- [85] H. Su, W. Wang, X. Li, and S. Xia, "Hierarchical neural network for recognizing hand-written characters in engineering drawings," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 46–49, 1995.
- [86] H.-H. Song and S.-W. Lee, "A self-organizing neural tree for large-set pattern classification," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 1111–1114, 1995.
- [87] K. Miyahara and F. Yoda, "Printed japanes character recognition based on multiple LVQ neural networks," in *Proc. 2nd Int'l Conf. Doc. Anal. Rec.*, pp. 250–253, 1993.
- [88] S.-W. Lee and J.-S. Kim, "Multi-lingual, multi-font and multi-size large-set character recognition using self-organizing neural networks," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 28–33, 1995.
- [89] Y. Waizumi, N. Kato, K. Saruta, and Y. Nemoto, "High speed rough classification for handwritten characters using hierarchical vector quantization," in *Proc. 4th Int'l Conf. Doc. Anal. Rec.*, pp. 23–27, 1997.
- [90] S. B. Cho, "Neural-network classifiers for recognizing totally unconstrained handwritten numerals," *IEEE Trans. Neural Networks*, vol. 8, no. 1, pp. 43–53, 1997.
- [91] K. Chung and J. Yoon, "Performance comparison of several feature selection methods based on node pruning in handwritten character recognition," in *Proc. 4th Int'l Conf. Doc. Anal. Rec.*, pp. 11–15, 1997.
- [92] T. K. Ho, "Recognition of handwritten digits by combining independent learning vector quantization," in *Proc. 2nd Int'l Conf. Doc. Anal. Rec.*, pp. 818–821, 20–22 October 1993.
- [93] S. W. Lee, "Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network," *IEEE Trans. PAMI*, vol. 18, no. 6, pp. 648–652, 1996.
- [94] H. Schwenk and M. Milgram, "Transformation invariant autoassociation with application to hand-written character recognition," in *Proc. NIPS*, pp. 991–998, 1996.

- [95] M. Gori, S. Marinai, and G. Soda, "Handwritten character recognition using the championship algorithm," in *1st Italian Workshop Digital Image Proc. by Neural Net.*, (Rome), pp. 49–54, 1993.
- [96] U. H.-G. Kressel, "An empirical study on the impact of the learning set size for handwritten digit recognition," in *Int. Conf. Artificial Neural Networks*, 1991.
- [97] G. L. Martin and J. A. Pittman, "Recognizing hand-printed letters and digits using backpropagation learning," *Neural Computation*, vol. 3, no. 2, pp. 258–267, 1991.
- [98] J. H. Kim, K. K. Kim, and C. Y. Suen, "An HMM-MLP hybrid model for cursive script recognition," *Pattern Analysis and Applications*, vol. 3, no. 4, pp. 314–324, 2000.
- [99] M. Gilloux, B. Lemarié, and M. Leroux, "A hybrid radial basis function network/hidden markov model handwritten word recognition system," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 394–397, 1995.
- [100] A. Amin and W. Mansoor, "Recognition of printed arabic text using neural networks," in *Proc. 4th Int'l Conf. Doc. Anal. Rec.*, pp. 612–615, 1997.
- [101] J.-P. Dodel and R. Shinghal, "Symbolic/neural recognition of cursive amounts on bank cheques," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 15–18, 1995.
- [102] D. S. Doermann, "The indexing a retrieval of document images: a survey," *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 287–298, 1998.
- [103] F. Cesarini, M. Gori, S. Marinai, and G. Soda, "INFORMys: A flexible invoice-like form reader system," *IEEE Trans. PAMI*, vol. 20, no. 7, pp. 730–745, 1998.
- [104] M.-C. Jung, Y.-C. Shin, and S. N. Srihari, "Multifont classification using typographical attributes," in *Proc. 5th Int'l Conf. Doc. Anal. Rec.*, (Bangalore, India), pp. 353–356, IEEE Computer Society Press, 1999.
- [105] T. Steinherz, E. Rivlin, and N. Intrator, "Offline cursive script word recognition - a survey," *Int. Journal Document Analysis and Recognition*, vol. 2, no. 2/3, pp. 90–110, 1999.
- [106] S. Knerr and E. Augustin, "A neural network-hidden markov model hybrid for cursive word recognition," in *Proc. 14th Int'l Conf. Pattern Recognition*, pp. 1518–1520, 1998.
- [107] M. Y. Chen, A. Kundu, and S. N. Srihari, "Variable duration hidden markov model and morphological segmentation for handwritten word recognition," *IEEE Trans. Image Processing*, vol. 4, no. 12, pp. 1675–1688, 1995.

- [108] M. Dehghan, K. Faez, and M. Ahmadi, "A hybrid handwritten word recognition using self-organizing feature map, discrete HMM, and evolutionary programming," in *Int. Joint Conference on Neural Networks*, pp. 515–520, 2000.
- [109] J. Bromley and J. S. Denker, "Improving rejection performance on handwritten digits by training with rubbish," *Neural Computation*, vol. 5, no. 3, pp. 367–370, 1993.
- [110] P. D. Gader, M. Mohamed, and J.-H. Chiang, "Comparison of crisp and fuzzy character neural networks in handwritten word recognition," *IEEE Trans. Fuzzy Systems*, vol. 3, no. 3, pp. 357–363, 1995.
- [111] L. M. Fu, *Neural networks in computer intelligence*. New York, NY: McGraw-Hill, 1994.
- [112] R. Seiler, M. Schenkel, and F. Eggimann, "Off-line cursive handwriting recognition compared with on-line recognition," in *Proc. 13rd Int'l Conf. Pattern Recognition*, pp. 505–509, 1996.
- [113] G. Martin, "Centered-object integrated segmentation and recognition of overlapping handprinted characters," *Neural Computation*, vol. 5, no. 3, pp. 419–429, 1993.
- [114] S.-W. Lee and E.-J. Lee, "Integrated segmentation and recognition of connected handwritten characters with recurrent neural network," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 413–416, 1995.
- [115] G. L. Martin, R. Mosfeq, D. Chapman, and J. A. Pittman, "Learning to see where and what: training a net to make saccades and recognize handwritten characters," in *Proc. NIPS*, pp. 441–447, 1993.
- [116] A. Shustorovich and C. W. Thrasher, "Neural network positioning and classification of handwritten characters," *Neural Networks*, vol. 9, no. 4, pp. 685–693, 1996.
- [117] O. Matan, J. C. Burges, Y. LeCun, and J. S. Denker, "Multi-digit recognition using a space displacement neural network," in *Proc. NIPS*, pp. 488–495, 1992.
- [118] F. Leclerc and R. Plamondon, "Automatic signature verification: the state of the art," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 8, no. 3, pp. 643–660, 1994.
- [119] L. Lee, "Neural approaches for human signature verification," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 1055–1058, IEEE Computer Society Press, 1995.

- [120] R. Sabourin, R. Plamondon, and G. Lorette, "Off-line signature identification with handwritten signature images: survey and perspectives," in *IAPR Workshop on Syntactic and Structural Pattern Recognition*, pp. 377–391, 1990.
- [121] J. P. Drouhard, R. Sabourin, and M. Godbout, "A neural network approach to off-line signature verification using directional PDF," *Pattern Recognition*, vol. 29, no. 3, pp. 415–424, 1996.
- [122] C. Sansone and M. Vento, "Signature verification: increasing performance by a multi-stage system," *Pattern Analysis and Applications*, vol. 3, no. 2, pp. 169–181, 2000.
- [123] K. Huang and H. Yan, "Off-line signature verification based on geometric feature- extraction and neural-network classification," *Pattern Recognition*, vol. 30, no. 1, pp. 9–17, 1997.
- [124] R. Plamondon and G. Lorette, "Automatic signature verification and writer identification: the state of the art," *Pattern Recognition*, vol. 22, no. 2, pp. 107–131, 1989.
- [125] N. A. Murshed, F. Bortolozzi, and R. Sabourin, "Off-line signature verification, without a priori knowledge of class ω_2 . A new approach," in *Proc. 3rd Int'l Conf. Doc. Anal. Rec.*, pp. 191–195, 1995.
- [126] H. Cardot, M. Revenu, B. Victorri, and M.-J. Revillet, "A static signature verification system based on a cooperating neural networks architecture," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 8, no. 3, pp. 679–692, 1994.
- [127] R. Bajaj and S. Chaudhury, "Signature verification using multiple neural classifiers," *Pattern Recognition*, vol. 30, no. 1, pp. 1–8, 1997.
- [128] M. Gori and F. Scarselli, "Are multilayer perceptrons adequate for pattern recognition and verification?," *IEEE Trans. PAMI*, vol. 20, no. 10, pp. 1121–1132, 1998.