Combining Flat and Structured Representations for Fingerprint Classification with Recursive Neural Networks and Support Vector Machines

Yuan Yao[a] Gian Luca Marcialis[b] Massimiliano Pontil[c,a]
Paolo Frasconi[d] Fabio Roli[b]

[a]*Department of Mathematics, City University of Hong Kong, Hong Kong*

[b]*DIEE University of Cagliari, Italy*

[c]*DII, University of Siena, Italy*

[d]*DSI, University of Florence, Italy*

Abstract

We present new fingerprint classification algorithms based on two machine learning approaches: support vector machines (SVM), and recursive neural networks (RNN). RNN are trained on a structured representation of the fingerprint image. They are also used to extract a set of distributed features of the fingerprint which can be integrated in the SVM. SVM are combined with a new error-correcting code scheme. This approach has two main advantages: (a) It can tolerate the presence of ambiguous fingerprint images in the training set, and (b) It can effectively identify the most difficult fingerprint images in the test set. By rejecting these images the accuracy of the system improves significantly. We report experiments on the fingerprint database NIST-4. Our best classification accuracy is of 95.6 percent at 20 percent rejection rate and is obtained by training SVM on both FingerCode and RNN-extracted features. This result indicates the benefit of integrating global and structured representations and suggests that SVM are a promising approach for fingerprint classification.

Key words: Fingerprint classification, error-correcting output codes, recursive neural networks, support vector machines.

1 Introduction

The pattern recognition problem studied in this paper consists of classifying fingerprint images into one out of five categories: whorl (W), right loop (R), left loop (L), arch (A), and tented arch (T). These categories were defined during early investigations about fingerprint structure [1] and have been used extensively since then. The task is important because classification can be employed as a preliminary step for reducing complexity of database search in the problem of automatic fingerprint matching [2,3]: If a query image can be classified with high accuracy, the subsequent matching algorithm only needs to compare stored images belonging to the same class.

Many approaches to fingerprint classification have been presented in the literature and this research topic is still very active. Overviews of the literature can be found in [4–7]. These approaches can be coarsely divided into two main categories: "flat" and "structural" approaches. Flat approaches are characterized by the use of "decision-theoretic" (or statistical) techniques for pattern classification, namely, a set of characteristic measurements, called feature vector, is extracted from fingerprint images and used for classification. Methods in this category include detection of singular points [8], connectionist algorithms such as self-organizing feature maps [9], neural networks [10,11], and hidden Markov models [12]. Structural approaches presented in the literature rely on syntactic or structural pattern-recognition techniques. In this case fingerprints are described by production rules [13] or relational graphs. Parsing or (dynamic) graph matching algorithms [14,4] are used for classification. The structural approach has received less attention until now. However, a simple visual analysis of the "structure" of fingerprint images allows one to see that structural information can be very useful for distinguishing some fingerprint classes (e.g., for distinguishing fingerprints belonging to class A from the ones of class W - see Figure 1). Accordingly, we will attempt to "integrate" flat and structured representations and evaluate the practical benefits of their combination. Concerning this issue, very few works investigated the potentialities of such integration [15,16].

In this paper, we propose new fingerprint classification algorithms based on two machine learning approaches: support vector machines (SVM), and recursive neural networks (RNN). SVM is a relatively new technique for pattern classification and regression that is well-founded in statistical learning theory [17]. One of the main attractions of using SVM is that they are capable of learning in *sparse, high-dimensional spaces* with very few training examples. They have been successfully applied to various classification problems (see [18] and references therein [1]). A RNN is a connectionist architecture designed for

[1] For an updated list of SVM applications see `www.clopinet.com/`

solving the supervised learning problem when the instance space is comprised of labeled graphs [19]. This architecture can exploit structural information in the data, which, as explained above, may help discriminating between certain classes. In this paper RNN are also used to extract a distributed vectorial representation of the relational graph associated with a fingerprint. This vector is regarded as an additional set of features subsequently used as inputs for the SVM classifier.

An important issue in fingerprint classification is the problem of ambiguous examples: some fingerprints are assigned to two classes simultaneously, i.e. they have double labels (these images are also called "cross-referenced"). In order to address this issue, we designed an error-correcting code [20] scheme of SVM classifiers based on a new type of decoding distance. This method presents two main advantages. First, it allows a more accurate use of ambiguous examples because each SVM is in charge of generating only one codebit, whose value discriminates between two disjoint sets of classes. Then, if a fingerprint has labels all belonging to the same set for a particular codebit, we can keep this example in the training set without introducing any labeling noise. The second advantage of our system is his capability to deal with rejection problems. This is due to the concept of *margin* inherent to the SVM, which is incorporated in the decoding distance.

The system is validated on the NIST database 4 [21]. We present three series of experiments. As a base fingerprint representation we use FingerCode features[2], a flat representation scheme proposed in [3]. In the first set of experiments, FingerCode features are combined with a structural representation of fingerprints based on relational graphs. In this case, a connectionist architecture that integrates flat and structural representations achieves 87.9 percent accuracy at 1.8 percent rejection rate. In the second experiment, SVM are trained on FingerCode [3] preprocessed images, achieving 89.1 percent accuracy with 1.8 percent rejection. This result is only 0.9 percent worse than the accuracy obtained in [3] using the same features and a two stages k-NN/MLP classifier. Interesting SVM's accuracy is much better than separate accuracies of both k-NN and MLP. Finally, the SVM is trained on both FingerCode and RNN-extracted features. In so doing, the performance is improved to 90.0 percent at 1.8 percent rejection rate. By allowing 20 percent rejection rate, accuracy increases to 95.6 percent. This result indicates the benefit of integrating global and structural representations for fingerprint classification.

The paper is organized as follows: In Section 2 we briefly describe SVM's theory and introduce our new error-correcting code classification method. Section 3 discusses the method we designed for the extraction of the structural

`isabelle/Projects/ SVM/applist.html`

[2] A short introduction on FingerCode feature is given in Section 5.1.

features. The RNN is presented in Section 4. In Section 5 we report the experimental results. Finally, we report our conclusions in Section 6.

2 Support vector machines

In this section we briefly overview the main concepts of support vector machines (SVM) [17] for pattern classification. More detailed accounts are [17,22,23,18].

2.1 Binary classification

SVM perform pattern recognition for two-class problems by determining the separating hyperplane[3] with maximum distance to the closest points of the training set. These points are called *support vectors*. If the data is not linearly separable in the input space, a non-linear transformation $\Phi(\cdot)$ can be applied which maps the data points $\mathbf{x} \in \mathbb{R}^n$ into a high (possibly infinite) dimensional space $\mathcal{H}$ which is called feature space. The data in the feature space is then separated by the optimal hyperplane as described above.

The mapping $\Phi(\cdot)$ is represented in the SVM classifier by a kernel function $K(\cdot, \cdot)$ which defines an inner product in $\mathcal{H}$, i.e. $K(\mathbf{x}, \mathbf{t}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{t})$. The decision function of the SVM has the form:

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i c_i K(\mathbf{x}_i, \mathbf{x}), \tag{1}$$

where ℓ is the number of data points, and $c_i \in \{-1, 1\}$ is the class label of training point $\mathbf{x}_i$. Coefficients α_i in Equation (1) can be found by solving a quadratic programming problem with linear constraints. The support vectors are the nearest points to the separating boundary and are the only ones for which α_i in Equation (1) can be nonzero.

An important family of admissible kernel functions are the Gaussian kernel:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|/2\sigma^2\right),$$

with σ the variance of the gaussian, and the polynomial kernels:

$$K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^d,$$

[3] SVM theory also includes the case of non-separable data, see [17].

with d the degree of the polynomial. For other important examples of kernel functions used in practice see [23,17].

Let ρ be the distance of the support vectors to the hyperplane. This quantity is called *margin* and it is related to the coefficients in Equation (1),

$$\rho = \left(\sum_{i=1}^{\ell} \alpha_i \right)^{\frac{1}{2}} . \qquad (2)$$

The margin is an indicator of the separability of the data. More precisely, the expected error probability of the SVM is bounded by the average (with respect to the training set) of $\frac{R^2}{\ell \rho^2}$, where R the radius of the smallest sphere containing the support vector in the feature space [17]. In the case that the Gaussian kernel is used, R can be upper bounded by 1. Then, in this case the effect of the radius on the expected error probability of the SVM can be discarded.

2.2 Multi-class classification with error-correcting codes

Many real-world classification problems involve more than two classes. Attempts to solve q-class problems with SVM have involved training q SVM, each of which separates a single class from all remaining classes [17], or training $\frac{q(q-1)}{2}$ machines, each of which separates a pair of classes [24–26]. The first type of classifiers are usually called *one-vs-all*, while classifiers of the second type are called *pairwise* classifiers. When the one-vs-all classifiers are used, a test point is classified into the class whose associated classifier has the highest score among all classifiers. In the case of pairwise classifiers, a test point is classified in the class which gets most votes among all the possible classifiers [25].

Classification schemes based on training one-vs-all and pairwise classifiers are two extreme approaches: the first uses all the data, the second the smallest portion of the data. In practice, it can be more effective to use intermediate classification strategies in the style of error-correcting codes (ECC) [20,27]. In this case, each classifier is trained to separate a subset of classes from another disjoint subset of classes (the union of these two subsets does not need to cover all the classes). For example the first set could consist of classes A and T and the second of classes R,L and W. By doing so, we associate each class with a row of the “coding matrix” $M \in \{-1, 0, 1\}^{q \times s}$, where s denotes the number of classifiers. $M_{ij} = -1$ or 1 means that points in class i are regarded as negative or positive examples for training the $j-$th classifier. $M_{ij} = 0$ says that points in class i are not used for training the $j-$th classifier.

Three sets of SVM classifiers were used to construct the coding matrix: 5 one-vs-al classifiers, 10 two-vs-three classifiers and 10 pairwise classifiers. This allows a more accurate use of ambiguous examples, since each SVM is only in charge of generating one codebit, whose value discriminates between two disjoint sets of classes. If a fingerprint has labels all belonging to the same set for a particular codebit, then clearly we can keep this example in the training set without introducing any labeling noise. As an example consider fingerprints with double labels A and T. These examples are discarded by the one-vs-all classifiers of class A and T, and the pairwise classifier A-vs-T. However they are used to train the classifier AT-vs-RLW.

A test point is classified in the class whose row in the coding matrix has minimum distance to the vector of outputs of the classifiers. Let $\mathbf{m}$ be a row of the coding matrix, $\mathbf{f}$ the vector of outputs of the classifiers, and γ_i the margin of the $i-$th classifier. The simplest and most commonly used distance is the Hamming distance $d(\mathbf{f}, \mathbf{m}) = \sum_{i=1}^{s} 1 - \text{sign}(f_i m_i)$. We will also use two other distance measures which take in account the margin of the classifiers: The margin weighted Euclidean distance, $d(\mathbf{m}, \mathbf{f}) = \sum_{i=1}^{s} (\gamma_i f_i - m_i)^2$, and the soft margin distance proposed in [27], $d(\mathbf{f}, \mathbf{m}) = \sum_{j=1}^{s} |1 - f_i m_i|_+$. In the latter expression, the function $|x|_+$ is equal to x, when $x > 0$, and zero otherwise.

In Section 5 we will show that our ECC of SVM achieves a performance of 89.1 percent when fingerprints are represented with FingerCode features [3] and a performance of 90.0 percent when we use both FingerCode and the structural features below. At the same time we found that, when only the first label is used as a target (like standard systems do), the performance drops down to 88.4 percent and 89.4 percent respectively. This result indicates the advantage of our ECC-based approach to better handling cross-referenced fingerprints during training.

3 Extraction of structural features

In this section we discuss the algorithm we have used to extract the structured representation of the fingerprint image.

3.1 Flow diagram of the structural classification module

As pointed out in the Introduction, a visual analysis of fingerprint images allows one to note that fingerprint's "structure" can be extracted by segmenting the fingerprint into regions characterized by homogeneous ridge directions. To explain this aspect, consider the sample segmented directional images in

Figure 1 [4]. These images clearly tell us that structural information can be very useful for distinguishing fingerprint classes A and W (see Figure 1.a). On the other hand, structural information is not effective for distinguishing fingerprints belonging to classes L,R, and T (see Figure 1.b). The fingerprint structure can be characterized by local attributes of the image regions (area, average directional value, etc.) and by the geometrical and spectral relations among adjacent regions (relative positions, differences among directional average values, etc.). Therefore, the developed module extracts and represents the structural information of fingerprints by segmenting the related directional images and by converting such segmented images into relational graphs whose nodes correspond to regions extracted by the segmentation algorithm. Graph nodes are then characterized by local characteristics of regions and by the geometrical and spectral relations among adjacent regions.

Class A Class W Class L Class R Class T

(a) (b)

Fig. 1. (a) Examples of segmented fingerprint images related to the A and W classes. It is easy to see that structural information is very useful for distinguishing such fingerprint classes. (b) Examples of segmented fingerprint images related to the L,R, and T classes. It is easy to see that structural information is not very useful for distinguishing such fingerprint classes.

The main steps of our structural approach to fingerprint classification are as follows:

- Computation of the directional image of the fingerprint. This directional image is a 28x30 matrix. Each matrix element represents the ridge orientation within a given block of the input image. The directional image was computed using the algorithm proposed in [28].
- Segmentation of the directional image into regions containing ridges with similar orientations. To this end, the segmentation algorithm described in [4] was used.
- Representation of the segmented image by a directed positional[4] acyclic graph (DPAG). The representation of fingerprint structure is completed by characterizing each graph node with a numerical feature vector containing local characteristics of the related image region (area, average directional value, etc.) and some geometrical and spectral relations with respect to adjacent regions (relative positions, differences among directional average values, etc).

[4] Positional here means that for each vertex v, a bijection $P : \mathcal{E} \rightarrow I\!N$ is defined on the edges leaving from v.

- Classification of the above DPAG by a RNN made up of two multilayer perceptrons (MLP) neural nets. This neural network model is briefly described below (see [19] for more details).

Fig. 2. Left: segmented fingerprint image. Right: Relational graph.

3.2 Image segmentation and relational graph construction

In order to segment directional fingerprint images, we used an algorithm explicitly designed for such task [28]. This algorithm implements a very sophisticated "region growing" process. It starts from the central element of the directional image and scans the image according to a square spiral strategy. At each step, the segmentation algorithm uses a cost function to decide about the creation of a new region. The resulting segmentation is related to a minimum of such cost function. Details about this segmentation algorithm can be found in [4]. The construction of relational graphs from segmented images is performed by the following main steps:

- The image region containing the "core" point is selected as starting region for the graph construction, that is, a graph node associated to such region is initially created;
- The regions that are adjacent to such core region are then evaluated for the creation of new nodes. Nodes are created for adjacent regions which are located in one of the following spatial positions with respect to the core region: North, North East, East, South East, South, South West, West, North West;
- The process above is repeated for each of the new nodes until that all the regions of the segmented images have been considered;
- The graph nodes created by the algorithm above are finally characterized by a numerical feature vector containing local characteristics of the related image regions (area, average directional value, etc) and some geometrical and spectral relations with respect to adjacent regions (relative positions, differences among directional average values, etc).

Algorithm 1 is used to derive relational graphs from segmented directional images. Figure 2 depicts a relational graph that the algorithm derived from a segmented directional image.

Algorithm 1. `rgExtract(R)` Construct relational graphs from segmented directional images.

Input: *$R = [R_1, R_2, \ldots, R_n]$ is the set of regions of the segmented image.*
Output: *$\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the relational graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$.*
Notation: *For $k = 0, \cdots, 7$ we shall denote the "reference angles" for each position by $\beta_k = k\pi/4$, and the "reference interval" by $I_k = [\beta_k - \pi/8, \beta_k + \pi/8]$. An edge in a DPAG is denoted (u, v, k) indicating that v is the k-th child of u. Finally, for each region R, Xbar[R] and Ybar[R] denote the x-y coordinates of the region's center of mass.*

```
$\mathcal{V} \leftarrow R$
$\mathcal{E} \leftarrow \emptyset$
for $j \leftarrow 1, \ldots, n$ do
    Color[$R_j$] $\leftarrow$ WHITE;
end for
Let $Q$ be a FIFO queue of regions (initially empty);
Let $R_i$ be the region containing the core of the fingerprint;
Enqueue($Q$,$R_i$);
while $Q$ is not empty do
    $R_c \leftarrow$ Dequeue($Q$);
    Color[$R_c$] $\leftarrow$ BLACK;
    $R'_c \leftarrow \{R_j : R_j$ is adjacent to $R_c\}$
    for all $R_j \in R'_c$ do
        if Color[$R_j$] = WHITE then
            Enqueue($Q$, $R_j$);
            $\alpha_j \leftarrow$ slope of the vector from (Xbar[R_c], Ybar[R_c]) to (Xbar[R_j],
            Ybar[R_j]); Note that $\alpha_j \in [0, 2\pi]$.
            for $k \leftarrow 0, \cdots 7$ do
                if $\alpha_j \in I_k$ then
                    $d_{jk} \leftarrow |\alpha_j - \beta_k|$;
                else
                    $d_{jk} \leftarrow \infty$;
                end if
            end for
        end if
    end for
    for $k \leftarrow 0, \cdots, 7$ do
        $h \leftarrow \arg\min_j \{d_{jk}\}$;
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(R_c, R_h, k)\}$;
    end for
end while
return $\mathcal{G}$
```

4 Recursive neural networks

Research in connectionist models capable of representing and learning structured (or hierarchically organized) information begun in the early 90's with recursive auto-associative memories (RAAM) [29]. Since then, several other architectures have been proposed, including holographic reduced representations (HRR) [30], and recursive neural networks (RNN) [31,32,19]. A selection of papers in this research area recently appeared in [33]. RNN are capable of solving the supervised learning problems such as classification and regression when the output prediction is conditioned on a hierarchical data structure, like the structural representation of fingerprints described above. The input to the network is a labeled DPAG $\boldsymbol{U}$, where the label $U(v)$ at each vertex v is a real-value feature vector associated with a fingerprint region, as described in Section 3.2. A hidden state vector $X(v) \in R^n$ is associated with each node v. This vector contains a distributed representation of the subgraph dominated by v (i.e., all the vertices that can be reached starting a directed path from v). The state vector is computed by a state transition function which combines the state vectors of v's children with a vector encoding of the label of v:

$$X(v) = f(X(w_1), \cdots, X(w_k), U(v)) \tag{3}$$

being $\{w_1, \cdots, w_k\}$ the ordered set of v's children. Computation proceeds recursively from the frontier to the supersource (the vertex dominating all other vertices). The base step for Equation (3) is $X(w) = 0$ if w is a missing child. Transition function f is computed by a multilayer perceptron, which is replicated at each node in the DPAG, sharing weights among replicas. Note that the state vector is similar to the context layer used in simple recurrent networks [34], except that it encodes its context as a recursive data structure, and not simply a sequence. Pollack's RAAM [29] used similar distributed compositional representations but can only encode and decode trees and cannot solve supervised learning problems.

In the case of classification we assume that the input graph possess a supersource s (i.e. a node from which every other node can be reached by a directed path). Classification with recurrent neural networks is then performed by adding an output function g that takes as input the hidden state vector $X(s)$ associated with the supersource s:

$$Y = g(X(s)) \tag{4}$$

Function g is also implemented by a multilayer perceptron. The output layer in this case uses the *softmax* function (normalized exponentials), so that Y can be interpreted as a vector of conditional probabilities of classes given the input graph, i.e. $Y_i = P(C = i|\boldsymbol{U})$, being C a multinomial class variable. Training relies on maximum likelihood. The training set consists of T pairs $\mathcal{D} = \{(\boldsymbol{U}_1, c_1), \cdots, (\boldsymbol{U}_t, c_t), \cdots (\boldsymbol{U}_T, c_T),\}$ where c_t denotes the class of the t-th fingerprint in the dataset. According to the multinomial model, the log-likelihood has the form

$$l(\mathcal{D}; \theta) = \sum_t \log Y_{c_t} \tag{5}$$

where θ denotes the set of trainable weights and t ranges over training examples. Optimization is performed with a gradient descent procedure, where gradients are computed by the back-propagation through structure algorithm [31,32].

We remark that the state vector $X(s)$ at the supersource is a distributed representation of the entire input DPAG and encodes features of the input DPAG deemed to be relevant for discrimination amongst classes. In the subsequent experiments, the components of the state vector at the supersource are thus used as additional features that may help to discriminate between some class pairs.

5 Experimental results

5.1 *Dataset and FingerCode features*

Our system was validated on the NIST Database 4 [21]. This Database consists of 4000 images analyzed by a human expert and labeled with one *or more* of the five structural classes W, R, L, A, and T (more than one class is assigned in cases where ambiguity could not be resolved by the human expert). Previous works on the same dataset either rejected ambiguous examples in the training set (loosing in this way part of the training data), or used the first label as a target (potentially introducing output noise).

Fingerprints were represented with the structured representation discussed in Section 3 as well as with FingerCode features. FingerCode is a representation scheme described in [3] and consists of a vector of 192 real features computed in three steps. First, the fingerprint core and center are located. Then the algorithm separates the number of ridges present in four directions (0°, 45°, 90°, and 135°) by filtering the central part of a fingerprint with a bank of Gabor filters. Finally, standard deviations of grayscale values are computed on 48 disc sectors, for each of the four directions.

As in [3], a few fingerprint images were rejected[5] both during training (1.4 percent) and testing (1.8 percent). Thus, when not explicitly said, the evaluation performance of the methods discussed below is intended to be at 1.8 percent rejection rate. For the same reason we were not able to compute the performance at zero percent rejection rate, which is not expected to increase. However, this does not reduce the interest of the discussion. Our main goal here is to show the novel methodology offered by our machine learning approach: (a) the advantage of ECC to better exploit cross-referenced (i.e. double labeled) images for training, (b) the strength of the SVM in improving accuracy versus rejection curves and (c) the benefit of integrating flat and structural representations. In particular, as we explained in Section 2.2, our ECC scheme provides a novel approach to deal with cross-referenced images during training which can be easily implemented/extended to work on new image representations.

5.2 Results using RNN and integration of flat and structural classifiers

In order to investigate the potentialities of the combination of flat and structural methods, we coupled our structural approach (Section 3) with the vector-based approach proposed in [3]. Several strategies for combining classifiers were evaluated in order to combine the flat and structural approaches [35–37]. We firstly assessed the performance of simple and commonly used combination rules, namely, the majority voting rule and the linear combination. Such classifier combination rules works well if the classifiers exhibit similar accuracies and make "independent" errors. As the flat and the structural classifiers considered exhibit very different accuracies (86.0% vs. 71.5%; see Section 5.2.1) and their classifications were correlated in a complex way (namely, correlation of classifications strongly varies in the feature space), such simple combination rules performed poorly. Accordingly, we combined classifiers adopting the so-called "metaclassification" (or "stacked") approach, which uses a trainable rule or an additional classifier for the combination. In particular, after various experiments with different trainable combination rules proposed in the literature [37], a k-nearest neighbor classifier was selected for the combination.

5.2.1 Comparison between vector-based and structural classification

We have trained a multi-layer perceptron (MLP) using the FingerCode feature vector as input. The best performance on the test set were obtained with a MLP architecture with 28 hidden units. Table 1a shows the corresponding confusion matrix. The overall accuracy is 86.0 percent.

[5] This is due to the failure of FingerCode in reliably locating the fingerprint core.

Table 1
Confusion matrices: (a) MLP using FingerCode; (b) RNN using relational graphs; (c) combination of MLP and RNN. Rows denote the true class, columns the assigned class.

	W	R	L	A	T
W	354	24	13	4	1
R	8	349	0	6	27
L	8	2	349	6	15
A	0	8	4	347	72
T	55	8	12	2	292

(a)

	W	R	L	A	T
W	323	28	36	5	1
R	24	305	16	6	64
L	44	11	266	7	57
A	3	13	12	333	68
T	21	44	49	41	179

(b)

	W	R	L	A	T
W	359	19	10	3	2
R	6	345	0	7	29
L	4	2	342	5	18
A	0	4	0	361	65
T	0	8	9	46	312

(c)

Table 1b shows the confusion matrix of the RNN trained on the structural features discussed in Section 3. In this case the overall accuracy is 71.5 percent.

Tables 1a and 1b point out that the accuracy of the structural classifier is much lower than the one of the flat classifier. This is mainly due to the large degree of confusion among L, R and T classes. On the other hand, as expected, the best performance of the structural classifier are related to the discrimination between A and W classes.

Afterwards, we analyzed the degree of complementarity between the two classifiers discussed above. To this end, we computed the performance of an ideal "oracle" that, for each input fingerprint, always selects the one of two classifiers, if any, that classifies correctly such fingerprint. Such oracle applied to the two classifiers provided an overall accuracy of 92.5 percent. This accuracy value obviously represents a very tight upper bound for any combination method applied to the two classifiers. Yet, it points out the potential benefit of the combination of the flat and structural classifiers.

5.2.2 Combined flat and structural classification

A k-nearest neighbor classifier (with a value of the k parameter equal to 113) was used for combing the flat and structural classifiers. Such metaclassifier takes the outputs of the two classifiers as inputs and provides the final fingerprint classification as output. Table 1c depicts the confusion matrix for this experiment. Notice that such combination outperforms the best single classifier (i.e., the MLP classifier using the FingerCode representation; see Table 1a), so pointing out that the exploitation of structural information allows increasing classification performances. The accuracy of this classifier is 87.9 percent. In particular, we note that such combination improves the performances related to A and W classes, confirming the intuition suggested by Figure 1.

5.3 Results with SVM

We used the three types of multi-class classification schemes discussed in section 2.1 which are based on the combination of binary SVM. SVM have been trained using the SVMFu code[6] on a 550MHz Pentium-II PC. Training on 2000 examples takes about 10s for pairwise classifiers and 20s for one-vs-all classifiers.

One-vs-all SVM. We trained five one-vs-all SVM classifiers using both Gaussian kernels and polynomials of degree between 2 and 6. The best result was obtained with the Gaussian kernel with $\sigma = 1$: 88.0 percent at 1.8 percent rejection rate; the confusion matrix is reported in Table 2a. The best polynomial SVM was of degree 3 and achieved a performance of 84.5 percent only. Then, in the remaining experiments we used only the Gaussian kernel.

Pairwise SVM. We trained the ten pairwise SVM. The test set accuracy increases to 88.4 percent at 1.8 percent rejection rate, improving the MLP accuracy reported in [3] of 2 percent. The confusion matrix is reported in Table 2b.

Error-correction SVM scheme. Three sets of SVM classifiers were used to construct the coding matrix: 5 one-vs-al classifiers, 10 two-vs-three classifiers and 10 pairwise classifiers. The three kinds of decoding distances discussed in Section 2 were compared: (i) Hamming distance: 88.0 percent, (ii) Margin weighted Euclidean distance: 89.1 percent, (iii) Soft margin distance: 88.8 percent (all results are at 1.8 percent rejection rate). This results confirm the advantage of taking in account the margin of the classifiers inside the decoding distance. The confusion matrix for the margin weighted Euclidean distance is reported in Table 2c.

We have also trained the ECC of SVM for the four classes task (classes A and T merged together) using the margin weighted Euclidean distance. The obtained accuracy is of 93.7 percent at 1.8 percent rejection rate. For comparison, the accuracy reported in [3] for the sole MPL's is 92.1 percent, while the cascade of k-NN and MLP yields 94.8 percent.

[6] This software can be downloaded at `http://five-percent-nation.mit.edu/SvmFu`.

Table 2
Confusion matrices: (a) One-vs-all SVM; (b) Pairwise SVM; (c) ECC SVM with margin weighted Euclidean decoding). Rows denote the true class, columns the assigned class.

	W	R	L	A	T
W	356	23	14	3	1
R	4	344	1	7	33
L	4	2	356	6	13
A	0	2	5	371	55
T	0	7	7	48	302

(a)

	W	R	L	A	T
W	359	24	15	3	1
R	4	341	1	6	30
L	5	0	356	6	15
A	0	2	4	363	58
T	0	7	9	38	317

(b)

	W	R	L	A	T
W	362	22	11	3	0
R	4	350	3	8	27
L	7	2	357	5	11
A	0	3	3	398	32
T	0	13	9	51	283

(c)

5.3.1 Analysis of the margin

We have measured the margin as in Equation (2) and the number of support vectors of each SVM classifier used in our experiments (the training error of each individual classifier was always equal to zero). The number of support vector ranges between 1/5 and 1/2 of the number of training points. As expected the margin decreases for those classifiers which involve difficult pairs of classes. Among the pairwise classifiers, the A-T classifier has the smallest margin. The margin of the T-vs-all and A-vs-all is also small. However the margin of the AT-vs-RLW classifier increases, which might explain why our error-correcting strategy works well.

5.3.2 Rejection vs. accuracy

Let d_1 be the minimum distance of the vector of outputs of the classifiers from the coding row, and d_2 the second minimum distance. Rejection can be decided by looking at the difference $\Delta = d_2 - d_1$. This quantity can be seen as the margin of the multiclass classifier. A large value of Δ indicates high confidence in classification; when Δ is smaller than a given threshold we reject the data. The rejection rate is controlled by this threshold. Table 3 shows the accuracy-rejection tradeoff obtained in our experiments. Notice that the accuracy of the system increases sharply with the rejection rate. At 20 and 32.5 percent rejection, the system shows a moderate improvement over the best results in [3].

Table 3
Accuracy vs. rejection rate for the ECC of SVM trained on FingerCode features.

Rejection Rate:	1.8%	8.5%	20.0%	32.5%
5 Classes:	89.1%	90.6%	93.9%	96.2%
4 Classes:	93.7%	95.4%	97.1%	98.4%

5.4 Using both vectorial and structured features with SVM

We have trained SVM on both FingerCode and RNN-extracted features and used the ECC scheme with margin weighted Euclidean decoding. The confusion matrix is summarized in Table 4. The performance is improved to 90.0 percent at 1.8 percent rejection rate. If we compare this performance to the performance obtained with FingerCode features only (89.1 percent), we observe the benefit of integrating global and structural representations.

Table 4
Confusion matrix for the ECC of SVM trained on both FingerCode and RNN-extracted features.

	W	R	L	A	T
W	366	18	8	2	0
R	5	354	0	7	29
L	6	1	357	2	13
A	0	2	2	396	33
T	1	8	12	48	294

This effect is especially clear in the accuracy vs. rejection rate results. As shown in Table 5, the accuracy sharply increases with the rejection rate, improving significantly over the results obtained with FingerCode features only (see Table 3).

Table 5
Accuracy vs. rejection rate for the ECC of SVM trained on the union of FingerCode and RNN-extracted features

Rejection Rate:	1.8%	8.5%	20.0%	32.5%
5 Classes:	90.0%	92.2%	95.6%	97.6%
4 Classes:	94.7%	96.6%	98.4%	99.2%

Finally, Table 6 summarizes the performance results of the different methods presented above. From there one can clearly observe the advantage offered by the ECC approach as well as and the effect of integrating flat and structural features.

Table 6
Summary of the results of the different proposed methods. We have used the following notation: SVM1 for One-vs-all SVM, SVM2 for Pairwise SVM, and SVM3 for the ECC of SVM.

Method	RNN	MLP	k-NN	SVM1	SVM2	SVM3	SVM & RNN
Performance:	71.5%	86.0%	87.9%	88.0%	88.4%	89.1%	90.0%

6 Conclusions

In this paper we have studied the combination of flat and structured representations for fingerprint classification. An algorithm for extracting a structural representation of fingerprint images was presented. RNN were used to process this structural representation and to extract a distributed vectorial representation of the fingerprint. This vectorial representation was integrated with other global representation, showing significant improvement over global features only. Experiment were performed on the NIST Database 4. The best performance was obtained with an error-correcting code of SVM. This method can tolerate the presence of ambiguous examples in the training set and shown to be precise to identify difficult test images, then sharply improving the accuracy of the system at a higher rejection rate.

A number of questions and future research directions are still open. The main one is how to train directly SVM on structured representation.

Acknowledgment: We wish to thank Anil Jain for providing us the dataset of preprocessed NIST-4 fingerprints. This work was partially supported by CERG Grant 9040457.

References

[1] E.R. Henry. *Classification and Uses of Finger Prints.* Routledge, London, 1900.

[2] R.S. Germain, A. Califano, and S. Colville. Fingerprint matching using transformation parameter clustering. *IEEE Computational Science and Engineering*, 4(4):42–49, 1997.

[3] A.K. Jain, S. Prabhakar, and L. Hong. A multichannel approach to fingerprint classification. *PAMI*, 21 (4):348–359, 1999.

[4] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni. Fingerprint classification by directional image partitioning. *Transactions on Pattern Analysis Machine Intelligence*, 21(5):402–421, 1999.

[5] A.K. Jain, R. Bolle, and S. Pankainti. *Biometrics: Personal Identification in Networked Society.* Kluwer Academic Pub., Norwell, MA, 1999.

[6] L.C Jain, U. Halici, I. Hayashi, S.B Lee, and Tsutsui (Eds.). *Intelligent Biometric Techniques in Fingerprint and Face Recognitioni.* CRC Press, USA, 1999.

[7] A. K. Jain and S. Pankanti. *Fingerprint Classification and Recognition*, chapter The Image and Video Processing Handbook. Academic Press, 2000.

[8] K. Karu and A.K. Jain. Fingerprint classification. *Pattern Recognition*, 29(3):389–404, 1996.

[9] U. Halici and G. Ongun. Fingerprint classification through self-organizing feature maps modified to treat uncertainty. *Proceedings of the IEEE*, 84(10):1497–1512, 1996.

[10] K. Moscinska and G. Tyma. Neural network based fingerprint classification. In *Third International Conference on Neural Networks*, pages 229–232, 1993.

[11] H.V. Neto and D.L. Borges. Fingerprint classification with neural networks. In *Proceedings 4th Brazilian Symposium on Neural Networks*, pages 66–72. IEEE Press, 1997.

[12] A. Senior. A hidden markov model fingerprint classifier. In *Proc. 31st Asilomar Conf. Signal, Sistems, and Computers*, pages 305–310, 1997.

[13] B. Moayer and K.S. Fu. A syntactic approach to fingerprint pattern recognition. *Pattern Recognition*, 7:1–23, 1975.

[14] D. Maio and D. Maltoni. A structural approach to fingerprint classification. In *of the 13th International Conference on Pattern Recognition*, volume 3, pages 578–585, 1996.

[15] R. Cappelli, D. Maio, and D. Maltoni. Combining fingerprint classifiers. In *Proceedings First International Workshop on Multiple Classifier Systems (MCS2000)*, pages 351–361, Cagliari, 2000.

[16] G.L. Marcialis, F. Roli, and P. Frasconi. Fingerprint classification by combination of flat and structural approaches. *Proc. 3rd Int. Conf. on Audio- and Video-Based Biometric Person Authentication*, 2001.

[17] V. N. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[18] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[19] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Trans. on Neural Networks*, 9(5):768–786, 1998.

[20] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 1995.

[21] C.I. Watson and C.L. Wilson. *Fingerprint database.* National Institute of Standards and Technology, April 1992. Special Database 4, FPDB.

[22] C. Burges. A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery.* Kluwer Academic Publishers, Boston, 1998. (Volume 2).

[23] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[24] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, Denver, Colorado, 2000.

[25] Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1997.

[26] M. Pontil and A. Verri. Support vector machines for 3-d object recognition. *IEEE Trans. PAMI*, pages 637–646, 1998.

[27] Robert E. Schapire, Yoram Singer, and Erin Lee Young. Reducing multiclass to binary: A unifying approach for margin classifiers. Technical report, AT&T Research, 2000.

[28] M.J. Donahue and S.I. Rokhlin. On the use of level curves in image analysis. *Image Understanding*, 57(3):185–203, 1993.

[29] J. B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–106, 1990.

[30] Tony A. Plate. Holographic reduced representations. *IEEE Trans. on Neural Networks*, 6(3):623–641, 1995.

[31] C. Goller and A. Küchler. Learning task-dependent distributed structure-representations by backpropagation through structure. In *IEEE Int. Conf. on Neural Networks*, pages 347–352, 1996.

[32] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3), 1997.

[33] P. Frasconi, M. Gori, and A. Sperduti. Special section on connectionist models for learning in structured domains. *IEEE Trans. on Knowledge and Data Engineering*, 13(2), 2001.

[34] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[35] G. Giacinto and F. Roli. Ensembles of neural networks for soft classification of remote sensing images. In *European Symposium on Intelligent Techniques*, pages 166–170, 1997.

[36] G. Giacinto, F. Roli, and L. Bruzzone. Combination of neural and statistical algorithms for supervised classification of remote-sensing images. *Pattern Recognition Letters*, 21(5), 2000.

[37] J. Kittler and F. Roli, editors. *Proc. of the First International Workshop on Multiple Classifier Systems*, volume 1857 of *LNCS.* Springer-Verlag, 2000.