Learning Features for Fingerprint Classification

Xuejun Tan, Bir Bhanu and Yingqiang Lin
Center for Research in Intelligent Systems
University of California, Riverside, CA 92521, USA
{xtan, bhanu, yqlin}@cris.ucr.edu

Abstract. In this paper, we present a fingerprint classification approach based on a novel feature-learning algorithm. Unlike current research for fingerprint classification that generally uses visually meaningful features, our approach is based on Genetic Programming (GP), which learns to discover composite operators and features that are evolved from combinations of primitive image processing operations. Our experimental results show that our approach can find good composite operators to effectively extract useful features. Using a Bayesian classifier, without rejecting any fingerprints from NIST-4, the correct rates for 4 and 5-class classification are 93.2% and 91.2% respectively, which compare favorably and have advantages over the best results published to date.

1 INTRODUCTION

The *Henry System* is a systematic method for classifying fingerprints into five classes: Right Loop (R), Left Loop (L), Whorl (W), Arch (A), and Tented Arch (T). Figure 1 shows the examples of each class. This system of fingerprint classification is commonly used by almost all the developers and users. The most widely used approaches for fingerprint classification are based on the number and relations of the singular points (SPs), which are defined as the points where a fingerprint's orientation field is discontinuous. Using SPs as reference points, Karu and Jain [8] present a classification approach based on the structural information around SPs. Most other research uses a similar method: first, find the SPs and then use a classification algorithm to find the difference in areas, which are around the SPs for different classes. Several representations based on principal components analysis (PCA) [10], self-organizing map (SOM) [11] and Gabor filters [12] are used. The problems with these approaches are: (a) it is not easy to detect the SPs and some fingerprints do not have SPs; (b) the uncertainty in the location of SPs is large, which has great effect on the classification performance since the features around the SPs are used. Cappelli et al. present a structural analysis of a fingerprint's orientation field [9]. Jain and Minut propose a classification algorithm based on finding the kernel that best fits the flow field of the given fingerprint [15]. Both approaches are unnecessary to find the SPs. Researchers have also tried different methods to combine different classifiers to improve the classification performance. Senior [13] combines Hidden Markov Models (HMM), decision trees and PCASYS (a standard fingerprint classification algorithm) [10]. Yao et al. [14] present new fingerprint classification algorithms based on two machine learning approaches: support vector machines (SVMs) and recursive neural networks (RNNs). The features used in those approaches are well-defined conventional known features. Unconventional features discovered by the computer are never used in fingerprint classification.

In most imaging applications, the task of finding a good feature is equivalent to finding a good point in the search space of *composite operators*, where a composite operator consists of primitive operators and it can be viewed as a selected combination of primitive operations applied on images. Our Genetic Programming (GP) based approach may try many unconventional ways of combining primitive operations that may never be imagined by humans and yield exceptionally good results. The parallelism of GP and the speed of computers allow the search space explored by GP to be much larger than that by human experts. As the search goes on, GP will gradually shift the population of composite operators to the portion of the space containing good composite operators.

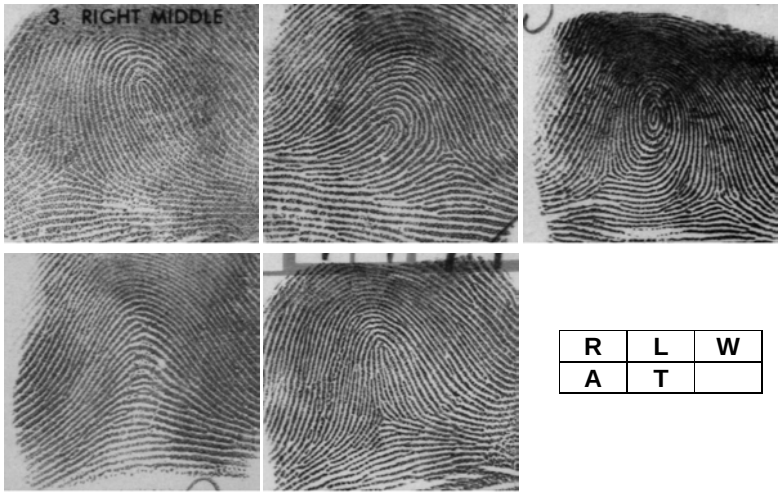

Fig. 1. Examples of fingerprints from each class of *Henry System* for fingerprint classification.

Genetic Programming (GP) was first proposed by Koza in [1]. Poli [2] used GP to develop effective image filters to enhance and detect features of interest or to build pixel-classification-based segmentation algorithms. Stanhope and Daida [3] used GP paradigm for the generation of rules for target/clutter classification and rules for the identification of objects. Howard et al. [4] applied GP for automatic detection of ships in low-resolution SAR imagery using an approach that evolves detectors. Roberts and Howard [5] used GP to develop automatic object detectors in infrared images.

The *contributions* of our work are: (a) we develop an approach to learn the composite operator based on primitive features automatically. This may help us to find some useful unconventional features, which are beyond the imagination of humans. The primitive operators defined in this paper are very basic and easy to compute. (b) Primitive operators are separated into computation operators and feature generation operators. Features are computed wherever feature generation operators are used. (c) Results are shown on the entire NIST-4 fingerprint database and they are compared with the other published research.

2 TECHNICAL APPROACH

Figure 2 shows the block diagram of our approach. During the training, GP is used to generate compositor operators, which are applied to the primitive features generated from the original orientation field. Feature vectors used for fingerprint classification are generated by composite operators. A Bayesian classifier is used for classification. During training, fitness value is computed according to the classification result and used for evolution. During testing, the learned composite operator is applied directly to generate feature vectors. Note that, in our approach, we do not need to find the reference points. The major design considerations of GP include:

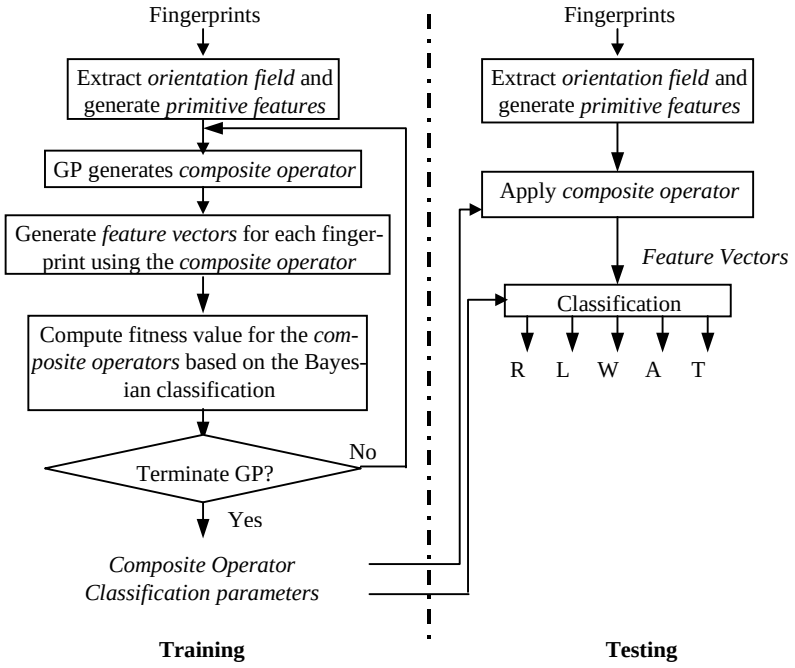

Fig. 2. Block diagram of our approach.

- **The Set of Terminals:** For a fingerprint, we can estimate the orientation field [6]. The set of terminals used in this paper are called primitive features, which are generated from the orientation field. Primitive features used in our experiments are : 1) original orientation image; 2) mean, standard deviation, min, max and median images obtained by applying 3×3 and 5×5 templates on orientation image; 3) edge images obtained by applying sobel filters along horizontal and vertical directions on orientation image; 4) binary image obtained by thresholding the orientation image with a threshold of 90. Note that, local orientation $\theta \in [0, 180)$; and 5) images obtained by applying *sin* and *cos* operations on the orientation image. These 16 images

are input to the composite operators. GP determines which operations are applied on them and how to combine the results.

Table 1. Primitive operators used in our approach.

Primitive Operator		Meaning
Computation Operators	ADD_OP, SUB_OP, MUL_OP and DIV_OP	A+B, A–B, A×B and A/B. If the pixel in B has value 0, the corresponding pixel in A/B takes the maximum pixel value in A.
	MAX2_OP and MIN2_OP	max(A,B) and min(A,B)
	ADD_CONST_OP, SUB_CONST_OP, MUL_CONST_OP and DIV_CONST_OP	A+c, A-c, A×c and A/c
	SQRT_OP and LOG_OP	$sign\ (A) \times \sqrt{\lvert A \rvert}$ and $sign\ (A) \times \log(\lvert A \rvert)$.
	MAX_OP, MIN_OP, MED_OP, MEAN_OP and STD_OP	max(A), min(A), med(A), mean(A) and std(A), replace the pixel value by the maximum, minimum, median, mean or standard deviation in a 3×3 block
	BINARY_ZERO_OP and BINARY_MEAN_OP	threshold/binarize A by zero or mean of A
	NEGATIVE_OP	-A
	LEFT_OP, RIGHT_OP, UP_OP and DOWN_OP	left(A), right(A), up(A) and down(A). Move A to the left, right, up or down by 1 pixel. The border is padded by zeros
	HF_DERIVATIVE_OP and VF_DERIVATIVE_OP	HF(A) and VF(A). Sobel filters along horizontal and vertical directions
Feature Generation Operators	SPE_MAX_OP, SPE_MIN_OP, SPE_MEAN_OP, SPE_ABS_MEAN_OP and SPE_STD_OP	max2(A), min2(A), mean2(A), mean2($\lvert A \rvert$) and std2(A)
	SPE_U3_OP and SPE_U4_OP	$\mu_3(A)$ and $\mu_4(A)$. Skewness and kurtosis of the histogram of A
	SPE_CENTER_MOMENT11_OP	$\mu_{11}(A)$. First order central moments of A
	SPE_ENTROPY_OP	H(A). Entropy of A
	SPE_MEAN_VECTOR_OP and SPE_STD_VECTOR_OP	mean_vector(A) and std_vector(A). A vector contains the mean or standard deviation value of each row/column of A

- **The Set of Primitive Operators:** A primitive operator takes one or two input images, performs a primitive operation on them and outputs a resultant image. Suppose 1) A and B are images of the same size and c is a constant, $c \in [-100,+100]$; 2) for operators, which take two images as input, the operations are performed on the pixel-by-pixel basis. Currently, there are two kinds of primitive operators in our approach: computation operators and feature generation operators. Table 1 explains the meaning of these operators in detail. For computation operators, the output is an image, which is generated by applying the corresponding operations on the input image. However, for feature generation operators, the output includes an image and a real number or vector. The output image is the same as the input image and passed as the input image to the next node in the composite operator. The real number or vector is

the elements of the feature vector, which is used for classification. Thus, the size of the feature vectors depends on the number of the feature generation operators that are a part of the composite operator.

- **Generation of New Composite Operator:** Composite operators are represented by binary trees whose internal nodes represent the primitive operators and leaf nodes represent the primitive features. The search of GP is done by performing reproduction, crossover and mutation operations. The initial population is randomly generated. The reproduction operation used in our approach is based on tournament selection. To perform crossover, two composite operators are selected on the basis of their fitness values. One internal node in each of these two parents is randomly selected, and the two subtrees with these two nodes as root are exchanged between the parents. In this way, two new composite operators are created. Once a composite operator is selected to perform mutation operation, an internal node of the binary tree representing this operator is randomly selected, and the subtree rooted at this node is deleted, including the node selected. Another binary tree is randomly generated and this tree replaces the previously deleted subtree. The resulting new binary tree replaces the old one in the population. We use *steady-state* GP in our experiments. A detailed description of it can be found in Koza [1].

- **The Fitness Measure:** During training, at every generation for each composite operator proposed by GP, we compute the feature vector and estimate the Probability Distribution Function (PDF) for each class using all the available feature vectors. Suppose the feature vectors for each class have normal distribution, $v_{i,j}$, where $i = 1,2,3,4,5$ and $j=1,2,\ldots n_i$, n_i is the number of feature vectors in the training for class i, ω_i . Then, for each i, we estimate the mean μ_i and covariance matrix Σ_i by all $v_{i,j}$, and the PDF of ω_i can be expressed as:

$$p(x|\omega_i) = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} exp(-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)) \quad (1)$$

According to Bayesian theory, we have

$$v \in \omega_k, \textit{iff. } p(v|\omega_k)\cdot p(\omega_k) = \max_{i=1,2,3,4,5}(p(v|\omega_i)\cdot p(\omega_i)) \quad (2)$$

where $x \in \{v_{i,1}\ v_{i,2} \ldots v_{i,n_i}\}$, n is the size of feature vector and v is a feature vector for classification. During training, we estimate $p(x \mid \omega_i)$, then use the entire training set to do the classification. The Percentage of Correct Classification (PCC) is taken as the fitness value of the composite operator.

$$\textit{Fitness Value} = \frac{n_c}{n_s} \times 100\% \quad (3)$$

where n_c is the number of correctly classified fingerprints in training set and n_s is the size of training set. Note that, if $|\Sigma_i|=0$ for ω_i in equation (1), we simply let the fitness value of the composite operator be 0. During testing, we still use equation (2) to obtain the classification results on the testing set, however, none of the testing fingerprints is used in the training.

- **Parameters and Termination:** The key parameters are maximum size of composite operator (150), population size (100), number of generations (100), crossover rate (0.6), and mutation rate (0.05). The GP stops whenever it finishes the pre-specified number of generations.

Composite Operator for 5-class classification, size 61:
((SUB_OP) ((MIN_OP) ((HF_DERIVATIVE_OP) ((HF_DERIVATIVE_OP) ((ADD_CONST_OP) ((MUL_OP) ((SPE_STD_VECTOR_OP) ((STDV_OP) ((SPE_CENTER_MOMENT11_OP) ((SQRT_OP) ((SUB_CONST_OP) ((VF_DERIVATIVE_OP) ((MEAN_OP) ((INPUT_OP: 0)))))))) ((SUB_CONST_OP) ((HF_DERIVATIVE_OP) ((SUB_CONST_OP) ((HF_DERIVATIVE_OP) ((ADD_CONST_OP) ((SUB_CONST_OP) ((ADD_CONST_OP) ((MUL_OP) ((SPE_STD_OP) ((MEAN_OP) ((LOG_OP) ((SPE_MEAN_VECTOR_OP) ((SQRT_OP) ((RIGHT_OP) ((SPE_MIN_OP) ((ABS_OP) ((MEAN_OP) ((INPUT_OP: 0))))))))))) ((SUB_CONST_OP) ((SPE_MEAN_VECTOR_OP) ((SPE_STD_VECTOR_OP) ((SPE_MIN_OP) ((STDV_OP) ((SPE_CENTER_MOMENT11_OP) ((SPE_U3_OP) ((SPE_STD_VECTOR_OP) ((SPE_MIN_OP) ((STDV_OP) ((SPE_CENTER_MOMENT11_OP) ((SPE_U3_OP) ((UP_OP) ((SPE_MEAN_OP) ((INPUT_OP: 1) ((SUB_CONST_OP) ((SPE_MEAN_OP) ((SQRT_OP) ((SUB_CONST_OP) ((SPE_U3_OP) ((SPE_U4_OP) ((SPE_STD_VECTOR_OP) ((SPE_MIN_OP) ((STDV_OP) ((SPE_CENTER_MOMENT11_OP) ((SQRT_OP) ((SUB_CONST_OP) ((VF_DERIVATIVE_OP) ((INPUT_OP: 13))))))))))))))))

Fig. 3. Learned composite operator for 5-class classification.

3 EXPERIMENTS

3.1 Database

The database used in our experiments is the NIST Special Database 4 (*NIST-4*) [11]. The size of the fingerprint images is 480×512 pixels with a resolution of 500 *DPI*. *NIST-4* contains 2000 pairs of fingerprints. Some sample fingerprints are shown in Figure 1. We use the first 1000 pairs of fingerprints for training and the second 1000 pairs of fingerprints for testing. In order to reduce the effect of overfitting, for the 1000 pairs of fingerprints in training, we only use first 500 pairs to estimate the parameters for each class and use the entire training set to validate the training results.

Table 2. Confusion matrix of the testing results for 5 and 4-class classifications.

	R	L	W	A	T
R	**356**	4	16	3	10
L	12	**372**	15	3	24
W	4	6	**369**	0	2
A	8	9	0	**416**	19
T	23	3	1	8	**337**

	R	L	W	A/T
R	**381**	4	11	23
L	11	**375**	5	40
W	1	4	**382**	1
A/T	20	12	3	**741**

3.2 Experimental Results

We performed the experiments 10 times and took the best result as the learned composite operator. Figure 3 shows the best composite operator for 5-class classifica-

tion, whose size is 61. Out of these 61 operators, there are 21 feature generation operators and the length of the feature vector is 87. The size of the best composite operator for 4-class classification is 149, which is much larger and not convenient to be shown directly. Obviously, these composite operators are not easy to be constructed by humans. Note that, it is possible to perform feature selection to reduce the size of feature vectors. During training, our approach runs slowly. Usually, it takes about 60 minutes for one generation to evolve. However, in testing, since it only needs to apply composite operator to the corresponding primitive operators, it runs very fast. On a SUN Ultra II workstation with a 200MHZ CPU, the average run-times for one fingerprint test for 5-class and 4-class classifications are 40ms and 71ms, respectively.

Table 3. Classification results on NIST-4.

Approaches	Class #	Error rate %	Reject rate %	Dataset	Comments
Karu and Jain 1996 [8]	5	14.6	zero	4000 images, no training	Decision based on topological information
	4	8.6			
Jain and Minut 2002 [15]	4	8.7	zero	Same as above	Hierarchical kernel fitting
Jain et al. 1999 [12]	5	14.6	**1.8**	Training: First 2000 images Testing: Second 2000 images	KNN
	4	8.5			
	5	13.6			Neural Network
	4	7.9			
	5	**10.0**			KNN+NN, two stage classifier
	4	**5.2**			
Senior 2001 [13]	4	Average 8.5[1]	zero	Same as above	Neural Network fusion with priors
Yao et al. 2003 [14]	5	10.0	1.8	Same as above	SVM+RNN
	4	5.3			
This paper	5	**8.4**	**zero**	Same as above	GP based learned features + Bayesian classifier
	4	**6.7**			

Table 2 shows the confusion matrix of our testing results on the second 1000 pairs of fingerprint in NIST-4. Note that, because of bad quality, the ground truths of some fingerprints provided by NIST-4 fingerprint database contain 2 classes, i.e. the ground truths of f0008_10 include class T and L. As other researchers did in their experiments, we only use the first ground truth label to estimate the parameters of the classifier. However, in testing, we use all the ground truth labels and consider a test as correctly classified if the output of the system matches to one of the ground truths. The PCC is 93.2% and 91.2% for 4 and 5-class classifications respectively. The classes R, L, W, A and T are uniformly distributed in NIST-4. However, in nature, the frequencies of their occurrence are 31.7%, 33.8%, 27.9%, 3.7% and 2.9%, respectively. From Table 2, we observe that most of the classification errors are related to classes A and T. Considering that A and T occur less frequently in nature, our ap-

[1] 3.1%, 4.2%, 4.5% and 22.3% for R,L,W and A/T respectively.

proach is expected to perform better in real world. Table 3 shows the results on NIST-4 database reported by other researchers. Considering that we have not rejected any fingerprints from NIST-4, our result is one of the best. For the 5-class classification, our result has 1.6% advantage over the result shown in [12], although in [12] the reject rate is 1.8%.

4 CONCLUSIONS

In this paper, we proposed a learning algorithm for fingerprint classification based on GP. Our experimental results show that the primitive operators selected by us are effective and GP can find good composite operators, which are beyond humans' imagination, to extract the feature vectors for fingerprint classification. The experimental results on NIST-4 fingerprint database show that our approach is one of the best approaches. Without rejecting any fingerprints, the experimental results show that our approach is promising and has advantages over the best results reported in the literatures.

Acknowledgments: This work is supported in part by a grant from SONY, DiMI, I/O Software and F49620-02-1-0315. The contents and information do not necessarily reflect the positions or policies of the sponsors.

References

1. J.R. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press, 1994.
2. R. Poli, Genetic programming for feature detection and image segmentation, Evolutionary Computation, T.C. Forgarty Ed., pp. 110-125, 1996.
3. S.A. Stanhope and J.M. Daida, Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery, Proc. Evolutionary Programming VII, pp. 735-744, 1998.
4. D. Howard, S.C. Roberts, and R. Brankin, Target detection in SAR imagery by genetic programming, Advances in Eng. Software, 30(5), pp. 303-311, May 1999.
5. S.C. Roberts and D. Howard, Evolution of vehicle detectors for infrared line scan imagery, Proc. Evolutionary Image Analysis, Signal Processing and Telecommunications, pp. 110-125, 1999.
6. A.M. Bazen and S.H. Gerez, Systematic methods for the computation of the directional fields and singular points of fingerprints, IEEE Trans. on PAMI, vol. 24, no. 7, pp. 905-919, July 2002.
7. C.I. Watson and C.L. Wilson, NIST special database 4, fingerprint database, U.S. National Institute of Standards and Technology, 1992.
8. K. Karu and A.K. Jain, Fingerprint classification, Pattern Recognition, 29(3), pp. 389-404, 1996.
9. R. Cappelli, A. Lumini, D. Maio and D. Maltoni, Fingerprint classification by directional image partitioning, IEEE Trans. PAMI, vol. 21, no. 5, pp. 402-421, 1999.
10. G.T. Candela, P.J. Grother, C.I. Watson, R.A. Wilkinson and C.L. Wilson, PCASYS --- a pattern-level classification automation system for fingerprints, Technical Report NISTIR 5647, NIST, Apr. 1995.
11. U. Halici and G. Ongun, Fingerprint classification through self-organizing feature maps modified to treat uncertainties, Proc. IEEE, vol. 84, no. 10, pp. 1497-1512, Oct. 1996.
12. A.K. Jain, S. Prabhakar and L. Hong, A multichannel approach to fingerprint classification, IEEE Trans. on PAMI, vol. 21, no. 4, pp. 348-359, Apr. 1999.
13. A. Senior, A combination fingerprint classifier, IEEE Trans. on PAMI, 23(10), pp. 1165-1174, 2001.
14. Y. Yao, G.L. Marcialis, M. Pontil, P. Frasconi and F. Roli, Combining flat and structured representations for fingerprint classification with recursive neural networks and support vector machines, Pattern Recognition, vol. 36, no. 2, pp. 397-406, Feb. 2003.
15. A.K. Jain and S. Minut, Hierarchical kernel fitting for fingerprint classification and alignment, Proc. ICPR, vol. 2, pp. 469-473, 2002.