Pattern Recognition ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Optimization of a training set for more robust face detection

Jie Chen[a], Xilin Chen[b,*], Jie Yang[c], Shiguang Shan[b], Ruiping Wang[b], Wen Gao[a,b]

[a]School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China
[b]Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[c]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

ARTICLE INFO

Article history:
Received 12 September 2008
Received in revised form 23 December 2008
Accepted 2 February 2009

Keywords:
Resampling
Genetic algorithm
Manifold
Face detection

ABSTRACT

The performance of a learning-based method highly depends on the quality of a training set. However, it is very challenging to collect an efficient and effective training set for training a good classifier, because of the high dimensionality of the feature space and the complexity of decision boundaries. In this research, we study the methodology of automatically obtaining an optimal training set for robust face detection by resampling the collected training set. We propose a genetic algorithm (GA) and manifold-based method to resample a given training set for more robust face detection. The motivations behind lie in two folds: (1) dynamic optimization, diversity, and consistency of the training samples are cultivated by the evolutionary nature of GA and (2) the desirable non-linearity of the training set is preserved by using the manifold-based resampling. We demonstrate the effectiveness of the proposed method through experiments and comparisons to other existing face detectors. The system trained from the training set by the proposed method has achieved 90.73% accuracy with no false alarm on MIT+CMU frontal face test set—the best result reported so far to our knowledge. Moreover, as a fully automatic technology, the proposed method can significantly facilitate the preparation of training sets for obtaining well-performed object detection systems in different applications.

1. Introduction

Over the past decades, the problem of human face detection has been thoroughly studied in the computer vision community for its fundamental challenges and interesting applications, such as video surveillance, human computer interaction, face recognition, and face data management. The goal of face detection is to determine whether there are any faces within a given image, and return the location and extent of each face in the image if one or more faces are present [39]. Recently, the emphasis has been laid on applications of machine learning techniques (e.g., [29,32,38]). Many previous methods can be found in the survey by Yang et al. [39]. After the survey, one of the most important progresses is the boosting-based method proposed by Viola and Jones [36] and its variations (e.g., [9,18,19,23,26,30,37,40]). Another progress is the convolutional face finder proposed by Garcia and Delakis [16].

However, compared with the great deal of efforts on classifier design when given a static training set, very little attention has been paid to the optimal use of the training set by resampling, and even less work has been done to systematically select training samples for face detection although collecting a training set is prerequisite for a learning task. It is often time consuming to collect a well-distributed training set [20]. In this paper, we present a method of automatically obtaining an optimal training set by resampling the positive samples. Unlike the previous methods, our method adds variations on positive samples by genetic algorithm (GA) extremely (i.e., from 6000 to 100,000) and then remove redundant samples iteratively by manifolds until obtaining a dense and balanced training set. The optimized training set is then used for more robust face detection. We demonstrate the effectiveness of the proposed method through experiments and comparisons to other existing face detectors. The system trained from the training set by the proposed method has achieved 90.73% accuracy with no false alarm on MIT+CMU frontal face test set [29].

The rest of this paper is organized as follows: Section 2 presents some related work. Section 3 describes how to resample a training set by GA and manifolds. Section 4 shows experimental results of the proposed method. Section 5 gives the conclusions.

2. Related work

In this section, we briefly review some related work. We start with the existing resampling method, and then present the related work on GA and manifolds.

* Corresponding author. Tel.: +86 10 6260 0754.
E-mail addresses: chenjie@jdl.ac.cn (J. Chen), xlchen@jdl.ac.cn (X. Chen), yang+@cs.cmu.edu (J. Yang), sgshan@jdl.ac.cn (S. Shan), rpwang@jdl.ac.cn (R. Wang), wgao@jdl.ac.cn (W. Gao).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.02.006

Resampling is one of basic issues in statistics. The theoretical foundations of resampling techniques are presented in [13]. The basic idea is to generate a subset from existing samples. The most widely used methods are jackknife (leave one out) [12], bagging [8], and arcing [14], etc. The jackknife is superior for small datasets. The basic idea of jackknife is to roundly select one sample for testing and the other for training a classifier. This procedure is carried out n times for a set with n samples. Bagging generates several training sets from an original training set and then trains a component classifier from each of those training sets [8]. In contrast, arcing is to adaptively resample and combine so that the sample weights during the resampling are increased for those ones most often misclassified [14]. Different from the traditional resampling techniques, the resampling method in our case not only selects a subset, but also generates new samples.

The resampling methods are also widely used in face detection and recognition applications. In order to expand the non-face training examples, Sung and Poggio [32] proposed the bootstrap to obtain more non-face examples during training. Lu and Jain [25] utilized the resampling techniques to generate several sample subsets from the original training dataset. Zhou and Jiang [42] showed that classifiers could benefit from the enhanced training set by adding virtual views. Kirby and Sirovich [21] used the Karhunen–Loève procedure to reconstruct human faces, which resulted in an extension of the data. Moreover, one challenge in face recognition domain is so called "one sample per person" problem: Only one sample per class is available to a system [33]. To solve this problem, Torre et al. [35] presented a method to combine various representations of the same image so as to exploit their specific merit. Beymer and Poggio [5] proposed a method to generate novel views of a single face image under different poses.

Evolutionary computation is a general stochastic search methodology, which inspired from evolution mechanisms discovered by Charles Darwin [17]. Recently, it has been used for various optimization problems in different areas, such as data mining, image processing, pattern recognition, and signal processing. Atkinson-Abutridy et al. [1] propose a novel approach for knowledge discovery from texts. Au et al. [2] develop the method of data mining by evolutionary learning, which introduces classification rules to predict the likelihood of each classification. Bhattacharyya et al. [6] present a domain-related structuring of the representation and incorporation of semantic restrictions for genetic programming (GP)-based search of trading decision models. Brameier and Banzhaf [7] introduce a new form of linear GP and show that GP achieves comparable performance in classification and generalization. Cano et al. [10] compare four evolutionary and some nonevolutionary instance selection algorithms. The experimental results suggest that the evolutionary instance selection algorithms outperform the nonevolutionary ones. Zhou et al. [41] employ gene expression programming to learn classification rules. Liu and Tang [24] use the evolutionary search for faces from line drawings to tackle the face identification problem by a variable-length GA.

GAs are a branch of the evolutionary computation [17]. They start from an initial population, often represented as bit strings, which evolves over successive generations. Those individuals who represent better solutions to approach the target win more chances to "reproduce" than those individuals which are relatively far from the target. They are mated with other solutions by crossing parts of a solution string with another. The reproduced strings are also mutated. Over time, they reproduce by crossing high fitness solutions at random points to weed out poor fitness solutions. These operations randomly sample a huge state space very efficiently.

A manifold provides a low dimensional description for data in high dimensional space. It enables us to visualize data, perform classification and cluster efficiently. Some prevailing approaches are isometric feature mapping (Isomap) [34], local linear embedding (LLE) [28], and Laplacian Eigenmap [4]. Meanwhile, Isomap is one of the representative techniques [34]. It is intuitive, well understood, and can produce reasonable mapping results. In this algorithm, firstly, distances between neighboring data points are calculated and neighborhood graph is constructed. Secondly, for each pair of non-neighboring data points, Isomap finds the shortest path through neighborhood graph, subject to the constraint that the path must hop from neighbor to neighbor. The length of this path (so called "the estimated geodesic distance") is an approximation to the true distance between its end points (so called "geodesic distance"), as measured within the underlying manifold. Finally, the classical multidimensional scaling is used to construct low-dimensional embedding.

3. Expanding a training set

Different from the bootstrap method in [32], which approaches the boundary between faces and non-faces by only expanding non-face set during training stage, we try to optimize a training set by expanding a *face* set. Comparing with the unlimited non-faces, the idea of using expanded faces combined with bootstrap for non-faces seems more reasonable. Moreover, it can significantly improve the performance of a trained detector and speed up the procedure of the preparation of training set [11].

Fig. 1 shows a comparison between a collected training set and the optimized training set by the proposed method. One can find that many face and non-face clusters are heavily interlaced. A human face is a 3D object which imposes many variations on face images. Many other objects may look like faces. It is impossible for a training set

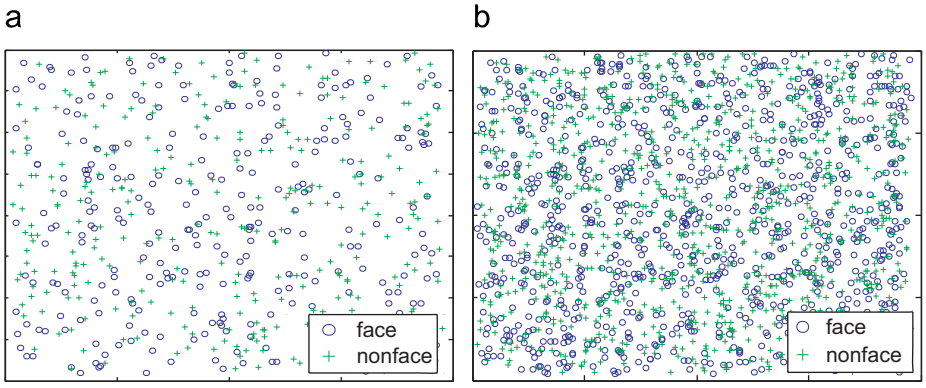

Fig. 1. Why do we optimize the training set? (a) A random collected face and non-face sets (projected on a 2D plane using manifold) and (b) the optimized face set by the proposed method (projected on a 2D plane using manifold).

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

to cover all these variations without careful selections. In addition, many samples are collected from web sites, videos or digital cameras without knowing the distribution of faces in the image space in advance (an example shown in Fig. 1(a)). Some locations in the feature space might be much denser while others much sparser, which results in the bias of a trained detector. What we want to do is to fill in the face example space firstly by GA and then resample it by manifolds to obtain a dense and balanced training set (i.e., it can better characterize the target distribution of faces) as shown in Fig. 1(b).

An overview of the proposed method is shown in Fig. 2. Initially, all of collected images are aligned coarsely, preprocessed, and divided into three sets: training, validating, and testing. The training set is then employed as an initial population to perform the GA operations. All the intermediate solutions of the current generation are evaluated by a classifier called Sparse Network of Winnows (SNoW) [38] which is trained by the last generation and non-face samples. Fitter solutions survive while weaker ones perish. All the survival solutions are then resampled using manifolds. The remained solutions, together with the initial population, are composed of the next population which is utilized to retrain the SNoW classifier again for the evaluation of the next intermediate solutions. Once the termination criterion is satisfied, the iterative GA operations are stopped and the last population is of the ultimate solutions.

A key issue of the proposed method genetic algorithm and manifold (GAM) is how to generating an effective sufficient dataset from a relatively small initial population. In general, it is easy to expand a dataset by interpolating if it is a small one but spans the face space well as shown in Fig. 3(a). However, it is usually difficult to collect such a delicate dataset. In practice, some commonly collected sets are shown in Fig. 3(b), such as S_1, S_2, or S_3. The detailed analysis for these commonly collected sets is as follows:

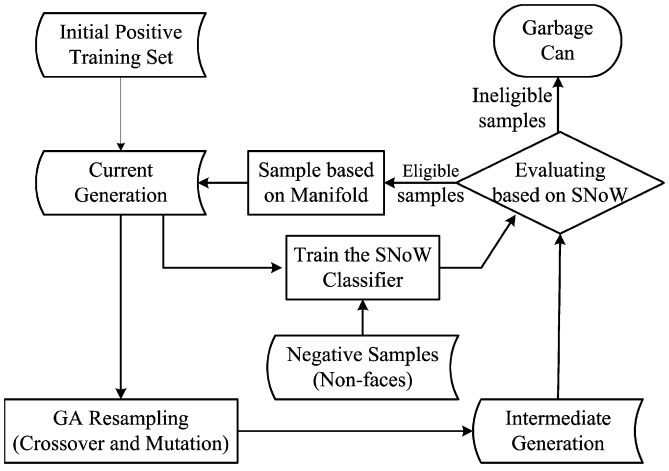

Fig. 2. A flow chart of the proposed method.

Let F be the entire face space. For each $\boldsymbol{f} \in F$, f depends on several factors. Let H be an imaging function. Formally, $\boldsymbol{f} = H(x_0, x_1, \ldots, x_{p-1}, x_p, \ldots, x_{p+q-1})$, where the first p variables denote the intrinsic variations of face, such as the facial shape and albedo, etc.; and the last q variables mean the extrinsic variations, such as the lighting conditions and pose variations, etc. Based on its definition, F can be divided into two separated datasets: a *core* face set C and its supplementary set E, i.e., $F = C \cup E$. Here, the core set C is composed of samples with the intrinsic variations and *normalized* extrinsic conditions (i.e., uniform lighting, frontal pose, and well-focused); the set E is composed of face samples with non-normalized extrinsic variations. Formally, $E = Lighting(C) \cup Pose(C) \cup Blurring(C) \cup \ldots$, where $Lighting(C)$ denotes the set of samples generated by relighting the samples in C, and so for $Pose(C)$ and $Blurring(C)$.

Based on the analysis above, what we expect to do is to approach F by interpolating and extrapolating from a collected dataset. In this paper, the interpolation is achieved by crossover operator, and the extrapolation is achieved by mutation operators such as relighting and blurring (For the detailed description, please refer to Section 3.3.). Given a collect set S (e.g., S_1, S_2 or S_3), it can be decomposed into two parts: $S = S_{sub\_1} \cup S_{sub\_2}$, where $S_{sub\_1} = S \cap C$ and $S_{sub\_2} = S \cap E$. Let t_i be the cardinality of S_{sub_i} ($i = 1, 2$). Usually, the larger t_1 is, the better the optimized training set can approach the entire face set F. In summary, an effective and sufficient dataset can be approached from a relatively small set that do not necessarily (sparsely) span the entire face space. Of course, the degree of approximation depends on t_1, i.e., the size of the intersection of the core set and the collected dataset. There is no constraint for t_2 (i.e., the size of the intersection of the non-core set and the collected dataset), although the optimized training set would benefit from the increase of t_2.

Fig. 4 illustrates that the proposed method can evidently generate new samples and valid face samples from existing ones. Although some of these samples might be very close to the existing ones, they can be deleted by the following resampling procedure. Furthermore, we expect the generated samples are valid in the sense of human perception. However, it is intractable in practice for human beings to check thousands of generated face samples during the GA evolution. Alternatively, an evolutionary classifier, i.e., SNoW, is employed to complete the task. To simulate human perception, the classifier should be balanced between conservation and generalization. In our case, the generalizability of SNoW is well controlled by the last population, since the newly added population is limited compared to the size of the existing generation.

As mentioned above, by generating some new and valid samples using GA and resampling the resulting generations by manifolds, we

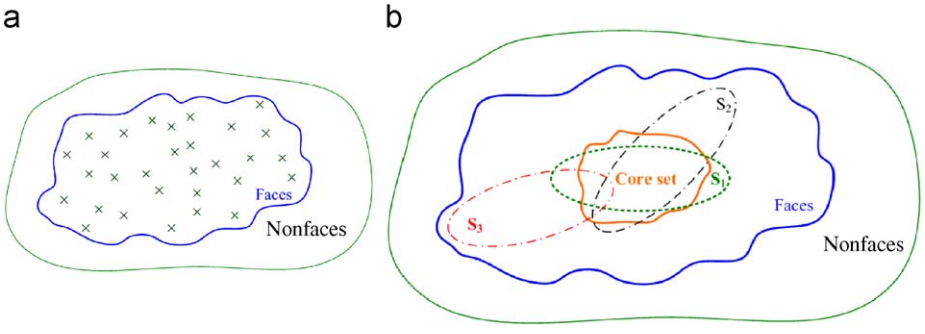

Fig. 3. An illustration of the collected initial population: (a) a small dataset but spanning the face space well, where the symbol "×" denotes a sample and (b) an illustration of some commonly collected datasets—S_1, S_2, and an unbalanced one S_3, where the *core set* is composed of samples with the intrinsic variations, such as the shape and albedo of faces.

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

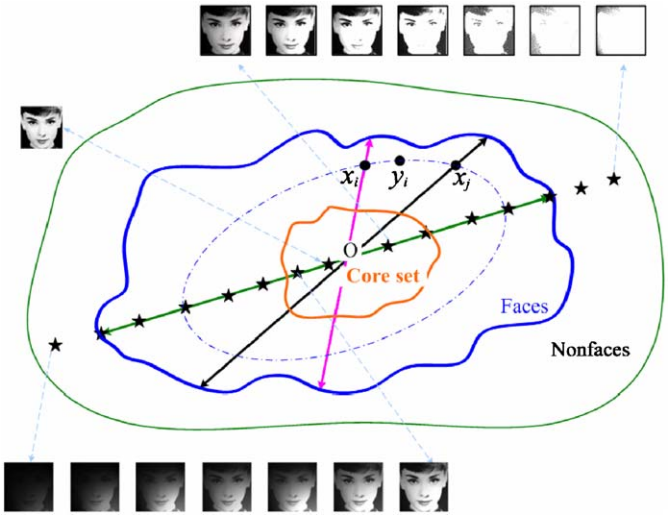

Fig. 4. An illustration of a dataset expansion by interpolating and extrapolating from a collected dataset. Here, we illustrate some examples generated by extrapolation (/mutation) through relighting as the "⋆" symbols, and the generation of y_i by interpolating (/crossover) x_i and x_j.

optimize the collected training set as shown in Figs. 1 and 4. In this section, after discussing the face variations and preprocessing, we describe the details of the proposed method.

3.1. Variations of face images

A face image is a 2D projection of a 3D head or a 2D face image under a certain lighting condition. A lot of factors can make a face image look differently from the others. All these factors are divided into two categories: intrinsic factors, such as the shape of basic components; and extrinsic factors, such as lighting, head pose, lens distortion and camera location. In our method, we simulate the intrinsic factors by crossover operator of GA and the extrinsic factors by mutation operator. Note that we focus on near-frontal faces, i.e., in-plane rotation, in this paper.

3.2. Preprocessing face samples

To begin GA operations, we need to construct an initial population, a validation set and test set to keep watching on the evolution of resulting generation for training a classifier. The original database in our case consists of 6000 faces which are collected from Internet, and covers wide variations in poses, facial expressions, and lighting conditions. The basic method to make a face detector robust to affine transform is to enlarge a face set by rotating, translating, and scaling [29].

By applying the preprocessing mentioned above, we obtain 30,000 face samples. The database is then randomly divided into three subsets: a training set (which consists of 15,000 images), a validating set (5000 images), and a testing set (10,000 images). However, the "training set" is optimized by the proposed method. We call this "training set" as "the *original* training set" in order to distinguish it from the optimized one by GA.

3.3. Expanding a training set by GA

It is often time consuming to collect a well-distributed training set. To this problem, we attempt to optimize the collected training set. Specifically, we expand the collected training set using GA, then cut the redundant samples in the expanded training set by manifolds and fill in the over sparse region with the aid of manifold embedding. In this section, we describe details of GA procedure. In Section 3.4, we will present a manifold-based method to reshape the distribution of generated solutions.

3.3.1. GA procedure for face generation

The main procedure of GA includes: encoding of a pattern, selecting of an initial population, operating of crossover and mutation, and evaluating of fitness. We explain them in detail in our application:

(1) *Pattern encoding.* As discussed in [38], given an image I with the size $w \times h$, for a pixel (x, y), assume that its intensity is $I(x,y)\,(0 \leqslant I(x,y) \leqslant 255)$. To embed both its location and intensity into an encoded string, we have $l(i) = l(y \times w + x) = 256(y \times w + x) + I(x,y)$. Thus, an individual $\boldsymbol{A}$ is represented as a string $(l_0)(l_1)(l_2)\cdots(l_i)\cdots(l_{w\times h-1})(f_j)$, where $(l_i)\,(0 \leqslant i \leqslant w \times h - 1)$ denotes a gene of an individual, and f_j is the fitness value of this individual. Note that both width w and height h are 20 in our experiments.

(2) *Initial population.* In most cases, the initial population is randomly generated [17]. In this proposed method, however, we create a population in which each individual is generated by encoding a normalized face sample. It is actually the encoded original training set as discussed in Section 3.2.

(3) *Crossover and Mutation.* Crossover and mutation are basic operations in GA. Here, we use the "roulette selection" to choose individuals for these operations. The roulette selection is based on the fitness value of each individual image—the higher an individual's fitness is, the more chances it has.

In our scheme, we consider "1-point" crossover in order to manipulate conveniently the fitness of solutions. Furthermore, every two parents crossover at fixed locus by the probability P_c, i.e., we break down each parent into four smaller pieces without overlapping: forehead, eye, nose, mouth, as demonstrated in Fig. 5(a), and the process of crossover are shown in Fig. 5(b). This might be explained by an intuition why someone says an individual, for example, as having *another person's eyes* [21]. For the details about how to partition sample set for GA, please refer to Section 3.3.2.

Mutation, in our case, is accomplished by sharpening, blurring or relighting with the probability P_m. The procedure of sharpening or blurring is that, first of all, a sub-image, about a quarter to half of its parent, is obtained from its parent. It is then sharpened or blurred randomly. Sequentially, we recombine the transformed sub-image and the original part to reproduce its child. To avoid the trace brought about by the recombination, the intermediate solutions are smoothed as shown in Fig. 5(c).

To simulate the lighting variations, during mutation operations, we use two kinds of strategies: one is the point light source model described in [22]. Specifically, we mutate a sample by changing its brightness plane, which is define as: $I(x,y) = \alpha x + \beta y$. For example, for $\alpha = 1$ and $\beta = 0$, the face is illuminated from 45° from the left. The other is the same as described in [3,27], which is used to simulate more complex diffuse light fields by a configuration of nine point light source directions. In summary, we mutate the selected samples by a harmonic images model. We first estimate the illuminations of the input samples and then re-render them to new lighting conditions. Some generated examples are shown in Section 4.1.

Following Srinivas and Patnaik's idea [31], the probabilities P_c and P_m are modified adaptively in the proposed method. The GA with the probabilities P_c and P_m adaptively modified is called as the adaptive GA (AGA), and the traditional GA is called as the standard GA (SGA). The modulated P_c and P_m for AGA are computed as

$$P_c = \begin{cases} k_1(f_{\max} - f_c)/(f_{\max} - \bar{f}), & f_c \geqslant \bar{f} \\ k_3, & f_c < \bar{f} \end{cases}, \quad (1)$$

$$P_m = \begin{cases} k_2(f_{\max} - f_m)/(f_{\max} - \bar{f}), & f_m \geqslant \bar{f} \\ k_4, & f_m < \bar{f} \end{cases}, \quad (2)$$

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

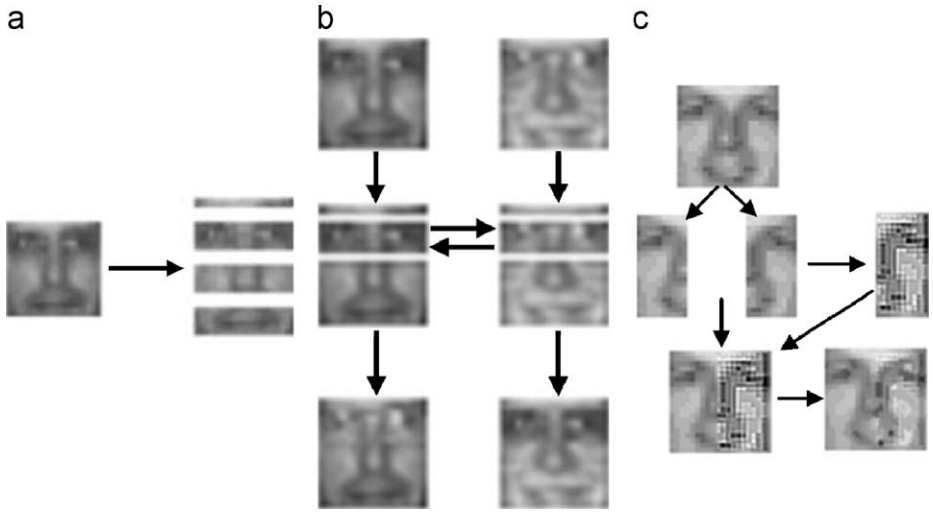

Fig. 5. Crossover and mutation: (a) convert each parent into a sequence of observation vectors for crossover, (b) crossover, and (c) mutation.

where $f_{\max}$ is the maximum fitness value of the current population and $\bar{f}$ is its average fitness; f_c is the bigger one of the two parents' fitness value for crossover; f_m is the fitness of mutated parent; $k_1 = k_3 = 1$, $k_2 = k_4 = 0.5$. AGA reduces P_c and P_m of those individuals whose fitness value are bigger than the average of the current population. This makes the AGA converge faster than SGA. Moreover, the probabilities P_c and P_m of those individuals whose fitness value are smaller than the average are increased to avoid the local solutions of SGA.

Using adaptive probabilities P_c and P_m in AGA seems contradictory to the idea of "roulette selection" and inconsistent with the idea of SGA. However, our goal is to expand the training set and increase its diversities, and the stop criterion in our case lays more emphasis on the performance of a detector trained from the entire population other than the fitness of individuals as in SGA. Therefore, using the "roulette selection" and the adaptive probabilities P_c and P_m is a tradeoff between expanding the training set and increasing its diversities.

Specifically, we use the "roulette selection" to select parents according to the fitness of the individuals. However, whether the selected individuals are employed to crossover or mutate depends on the probability P_c or P_m. Therefore, we can balance the reproduction of the individuals with the high and low fitness. The individuals with high fitness are selected with the higher probabilities and are provided relatively fewer choices to reproduce. In contrast, the individuals with low fitness are selected with the lower probabilities and are provided relatively more choices to reproduce. In turn, the average fitness of the resulting population by AGA is much smaller than that of the case by SGA [31]. It means that the resulting population by AGA contains more diversities than that of the case by SGA. Therefore, in most cases, AGA outperforms SGA significantly, especially for those problems that are highly multimodal. Indeed, faces can be modeled well by multimodal [32].

(4) *Fitness Evaluation*. A classifier is used as fitness function to measure the solutions. In the proposed method, the SNoW classifier is selected. The details of training SNoW and how to evaluate the solutions are discussed in Section 3.3.3.

3.3.2. Sample partition for GA

A few constraints are imposed on the input of GA operations in order to avoid some unexpected intermediate solutions. For instance, the parents for crossover should be under the similar poses. Therefore, as shown in Fig. 6, we divide the initial population evenly into six subsets: the first is those within $[-15^\circ, -10^\circ)$ and denoted by ω_1; the second is those within $[-10^\circ, -15^\circ)$ and denoted by ω_2; …; and the last is those within $[10^\circ, 15^\circ]$ and denoted by ω_6. Note that the degree here is of in-plane rotation. All of these six subsets are then used as the initial population of GA operations.

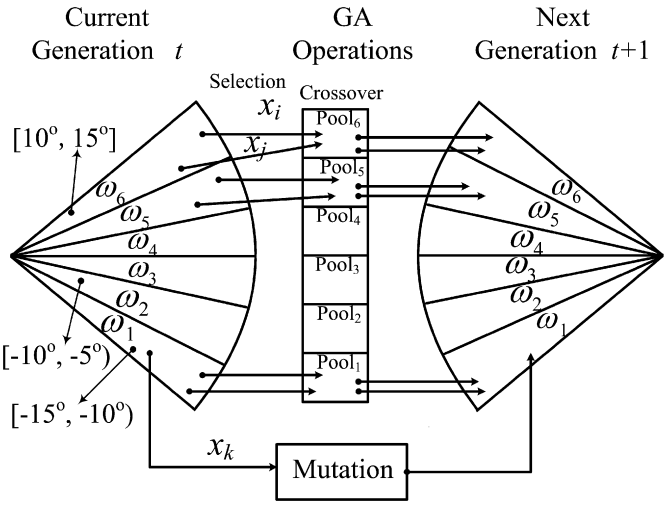

Fig. 6. An illustration of the GA operation process. An initial population is divided evenly into six subsets. Those individuals within the same subset crossover and some parents of the current generation mutate.

As shown in Fig. 6, these selected individuals are put into mating pools, and those individuals within the same subset crossover with the probability P_c (it is computed as Eq. (1)). For example, two individuals, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, selected from ω_6, are put into Pool_6. After crossover operations, their offspring is put back into ω_6 again. These operations are applied to all of the other subsets. Some parents of the current generation (not their children in our method) are mutated by the probability P_m (it is computed as Eq. (2)). For example, $\boldsymbol{x}_k$, selected from ω_1, is mutated and its child is put back into ω_1.

After each generation is reproduced, only partial intermediate solutions are remained. The remained solutions are no more than θ_p of current population (in our experiments, $\theta_p = 30\%$). That is, for those intermediate solutions whose fitness is larger than a survived threshold, if their number is beyond θ_p of current population, we keep only θ_p of them. Or we keep all of those solutions whose fitness

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

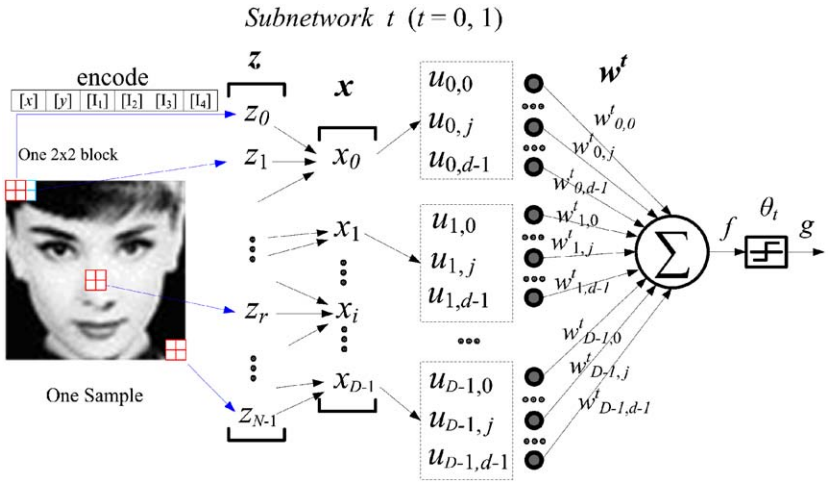

Fig. 7. An illustration of the architecture of SNoW.

value is larger than the threshold θ_f (in our experiments, $\theta_f = 0.55$). Herein, the fitness is a normalized value attached as demonstrated in Section 3.3.3. In this scheme, after every 10 generations, the population will contain $15,000 \times 1.3^{10} \approx 206,787$ individuals. This amount is much larger than that of the original one. In order to keep its size in control, we cut down its size and keep a reasonable amount of solutions (85,000 solutions in our case). How to perform this operation is discussed in Section 3.4. After the resampling, we have 100,000 individuals, which including the 15,000 ones of the initial population and their 85,000 children.

Herein, an issue is to determine the survived thresholds. In this research, they are determined experientially. On one hand, for the parameter θ_p, it is a tradeoff between the size of the resulting generations and their diversities. Specifically, if θ_p is too small, it is difficult to maintain the diversities of solutions since most of the samples are deleted. In contrast, if θ_p is too large, the population expands too quickly. On the other hand, as for the parameter θ_f, it is also a tradeoff between the diversities and reasonability of solutions. Specifically, if θ_f is too small, the diversities of samples become abundant while some unreasonable solutions are included. In contrast, if θ_f is too large, the diversity becomes less, which decreases the performance of a trained classifier.

We use SNoW to evaluate a new generation. Specifically, we use the current detector SNoW to measure the fitness of intermediate solutions, and the qualified intermediate solutions (together with the initial population) and a negative set are used as a new training set to train a newer classifier SNoW. Note that the newly trained classifier is utilized to evaluate the intermediate solutions of the next generation and is also tested on the validation set as discussed in Section 3.2. Moreover, we keep comparing these results of each generation. The GA operation is terminated when the difference of verified results between two neighbor generations is less than a preset threshold.

3.3.3. Solution evaluation based on SNoW

The SNoW is a sparse network of linear functions that utilizes the Winnow update rules. It is particularly suitable to learn tasks in domains where the number of features used for decisions is very large, but may be unknown a priori [38]. In our method, we use SNoW instead of AdaBoost because the training of SNoW is much faster than that of AdaBoost. In addition, Yang et al. trained a well-performed face detector using SNoW [38]. In this part, we describe the SNoW classifier, the feature, and the usage to evaluate the offspring during the GA operation.

(1) *SNoW classifier.* A SNoW classifier is illustrated in Fig. 7. This learning architecture is characterized by its sparsely connected units, the allocation of features and links in a data driven way, the decision mechanism and the utilization of an efficient update rule.

An input sample is divided into N overlapped 2×2-blocks. For each block, we concatenate its coordinate and spatially pixel intensities as a feature z_r ($r = 0,1,\ldots,N-1$). Subsequently, each feature z_r is quantized as a new feature x_i ($i = 0, 1,\ldots,D-1$). Herein, x_i is an m_1+m_2 bits binary string, where the first m_1 ($D \leqslant 2^{m_1}$) bits denote for its location and the next m_2 bits for its pattern modes. SNoW is a sparse network of linear units over a common pre-defined feature space (e.g., x_i, $i = 0,1,\ldots,D-1$). A node $u_{i,\,j}$ ($j = 0,1,\ldots,d-1$, $d \leqslant 2^{m_2}$), in the input layer is addressed by the input x_i, where d is the number of the possible pattern modes. A node $u_{p,\,q}$ ($p = 0, 1,\ldots,D-1$, $q = 0,1,\ldots,d-1$) is called active if a feature x_p is an input and its pattern mode is equal to q. Note that, for an input sample, only D nodes in the input layer of a subnetwork are active. In turn, $u_{p,\,q}$ activates the weight $w^t_{p,q}$ ($t = 0, 1$). The output of subnetwork t is $f = \sum_{p=0}^{D-1} w^t_{p,q} x_p = \sum_{p=0}^{D-1} w^t_{p,q}$ since we use the binary feature. Herein, $w^t_{p,q}$ is the weight on the edge connecting the pth feature of the subnetwork t. For the classification case, $g = 1$ if and only if $f > \theta_t$, else $g = 0$, where θ_t is the threshold.

The sparse connection of SNoW ensures that each x_i only activates one node $u_{i,j}$. Thus, for a given sample, only a few nodes are active (e.g., D nodes for a sample in our case). Different samples are encoded by different x_i ($i = 0,1,\ldots,D-1$), which leads to the different links between the input features and the nodes. The update rule for the weights is a variant of Littlestone's Winnow update rule, which is on-line and mistake-driven [38].

In our application, only two subnetworks are used, one for a face pattern and the other for a non-face pattern (i.e., subnetwork 0 for non-face and subnetwork 1 for face). A given sample is processed autonomously by each target subnetwork; that is, a sample labeled as a face is used as a positive sample for the face target and as a negative sample for the non-face target, and vice versa.

(2) *Features for SNoW.* The feature of SNoW can be varied with a task. In our application, to train the classifier, we use a feature pattern as described in [15] to characterize the samples as illustrated in Fig. 8. The feature of each possible 2×2-block is quantized into one scalar value. The location of each 2×2-block is coarsely encoded as 6 bits, so $m_1 = 6$; the intensity of each pixel is quantized as 3 bits, so $m_2 = 12$. Hence, we obtain an 18-bit feature string.

By this means, we totally have $2^{18} = 262,144$ possible binary features, which enlarge the feature space further and make the

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

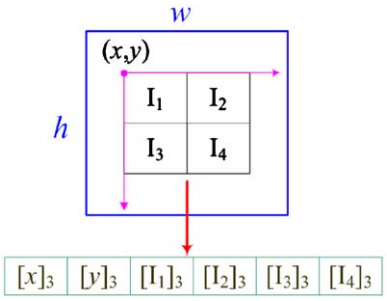

Fig. 8. A quantization method of grey features, where (x, y) is the coordinate of the 2×2-block; $\mathbf{I}_j$ $(j = 1, 2, 3, 4)$ are 4 pixels in this block; the subscript "3" in $[x]_3$, $[y]_3$ and $[\mathbf{I}_j]_3$ denotes that each component is encoded by 3-bit resolution in the feature representation.

two-class problem (face and non-face) more linear separable while compared with the feature space in [38] (102,400 possible binary features).

(3) *Training and evaluation.* The training process of an evaluation function (or classifier) SNoW is as follows. For each generation, we use the initial population plus the solutions of the latest generation as positive samples. For the negative samples, we start with 15,000 non-face examples from 12,156 images of landscapes, trees, buildings, etc. Although it is extremely difficult to collect a typical set of non-face examples, the bootstrapping [32] is used to include more non-face examples during training. All positive and negative samples are encoded and used as an input to SNoW as shown in Fig. 7. For the detailed training procedure please refer to [38].

Subsequently, we use the trained classifiers of SNoW to evaluate the intermediate solutions. Each resulting classifier is used to evaluate the successive generation. For each solution, as shown in Fig. 7, we encode it and input to the subnetwork 1 of SNoW (i.e., the subnetwork for face). The output (i.e., f) of subnetwork 1 is normalized and assigned to be the fitness value for the input solution.

Note that (1) for each generation, we use subnetwork 1 to evaluate the solutions and both subnetwork 1 and 0 to test on the validation set and test set; (2) As to the initial population, let the fitness value $f_j = 1$ for each individual; and (3) the second generation is evaluated by the classifier SNoW trained only by the initial population and negative examples.

3.4. Resampling by manifolds

We use a manifold-based method to resample the resulting generations after the reproduction by GA as discussed in Section 3.3.2 to prevent it from expanding excessively and guarantee its reasonable distribution. In this research, we use Isomap since it can discover the globally optimal manifold embedding of the input data. Furthermore, it is an isometric mapping, which preserves the distance in the low-dimensional manifold embedding well [34].

After discovering the manifold embedding of a set, we can easily find out the over-dense and over-sparse regions. For the over-dense regions, we subsample the samples based on the geodesic distances between samples. Here, we use the geodesic distance due to the fact that, to reflect intrinsic similarities of two arbitrary points (samples) on a nonlinear manifold, the geodesic distance is more reasonable than the Euclidean distance in the high dimensional input space (for example, on the "Swiss roll" manifold as shown in Fig. 9(a), the geodesic distance illustrated by the solid curve $\widehat{AB}$ is more reasonable than the Euclidean distance illustrated by dashed line $\overline{AB}$) [34]. We believe that the smaller the geodesic distance between two points (samples) is, the more intrinsic similarities they have. In other words, as shown in Fig. 9(b), when the distances are smaller than a given threshold, one point (samples) can be deleted.

For the over-sparse regions of the resulting face embeddings computed by Isomap, we generate some virtual samples to fill in these regions. The virtual samples are reconstructed using the neighbors and the corresponding weights. The neighbors can be found by K-nearest neighbors (KNN) in the low-dimensional manifold embedding, and the weights are the reciprocals of the geodesic distances between the virtual samples and their neighbors.

3.4.1. Subsampling on over-dense regions

We calculate the estimated geodesic distances in the high-dimensional space between pairs of samples using Isomap as in [34]. These distances are then used to subsample the generated populations. By this means, we can obtain a smaller representative subset from a huge amount of data. The algorithm of subsampling based on the Isomap is as follows:

Algorithm 1. Subsample based on the Isomap.

Input: Sample set $\boldsymbol{S} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}\}$ (herein, n is the number of data points), the estimated geodesic distance set $\boldsymbol{G} = g_{ij}$, $(i, j = 0, \ldots, n-1)$ between pairs of data points, and the number of samples, T, to be deleted;
Output: The subsampled set $\boldsymbol{S}$.
Step 1: Sort all of the estimated geodesic distances in $\boldsymbol{G}$ in ascending order;
Step 2: Determine the value of the threshold *Thr* for deleting examples according to T;
Step 3: **For** $t = 0, \ldots \ldots, T-1$:
  If one of the estimated geodesic distances g_{ij} between an example $\boldsymbol{x}_i$ and its adjacent point $\boldsymbol{x}_j$ $(i, j = 0,1, \ldots, n-1)$ is smaller than a given threshold Ξ:
    Count the number of the estimated geodesic distances in the sorted sequence, where the counted distances are relative to $\boldsymbol{x}_i$ or $\boldsymbol{x}_j$ and less than the given threshold Ξ;
    Delete the data point $\boldsymbol{x}_i$ from $\boldsymbol{S}$ if the counted number of $\boldsymbol{x}_i$ is larger than that of $\boldsymbol{x}_j$;
    Remove those estimated geodesic distances between the deleted data point and others from $\boldsymbol{G}$;
  End if;
End For;
Step 4: Get the subsampled set $\boldsymbol{S}'$.

Here, the parameter T is determined as: $T = n-m$; where n is the size of the current population and m is the number of the remained samples. For example, in our case, m is 80,000, and subtracting m (= 80,000) from the given population size, we get the value of the parameter T. In addition, the parameter Ξ is determined experientially, i.e., $\Xi = \max(g_{ij}) \times (T/n)$, where $\max(g_{ij})$ is the maximum of all the geodesics in $\boldsymbol{G}$. Note that in our implementation, the samples to be deleted are only marked, which does not affect the $\boldsymbol{G}$ matrix in the following iterations. Eventually, all the marked samples are deleted simultaneously after all the samples are visited.

For example, as shown in Fig. 9(b), the data points 1 and 2 are deleted during subsampling. As to the data point 3, it is reserved since the geodesic distances between point 3 and its neighborhoods are larger than the given threshold. From the right part of Fig. 9(b), the remained samples still basically maintain the data's intrinsic geometric structure.

3.4.2. Interweaving on over-sparse regions

As shown in Fig. 9(c), after the subsampling, there are still some over-sparse regions (i.e., holes) due to the unbalance of the database (generated populations), such as the location of circle O_1 and O_2, in

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

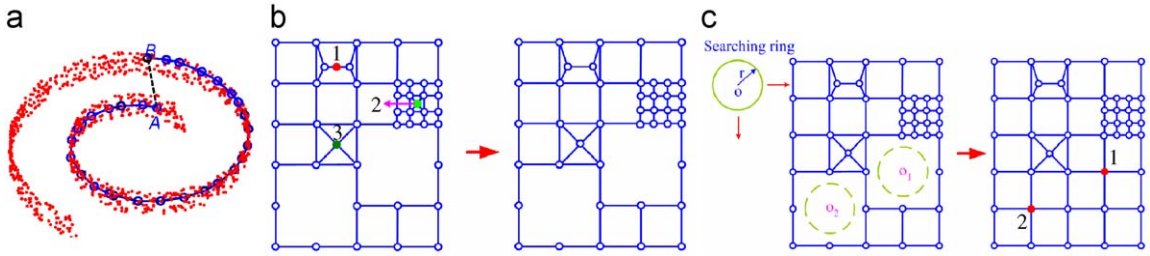

Fig. 9. An illustration of resampling a set by manifolds: (a) a "Swiss roll" manifold, where the dashed line $\overline{AB}$ denotes the Euclidean distance between point A and B in the high dimensional input space; and the solid curve $\widehat{AB}$ shows the geodesic distance along the low-dimensional manifold, (b) subsampling based on the estimated geodesic distances, and (c) the illustration of interweaving among the manifold embedding.

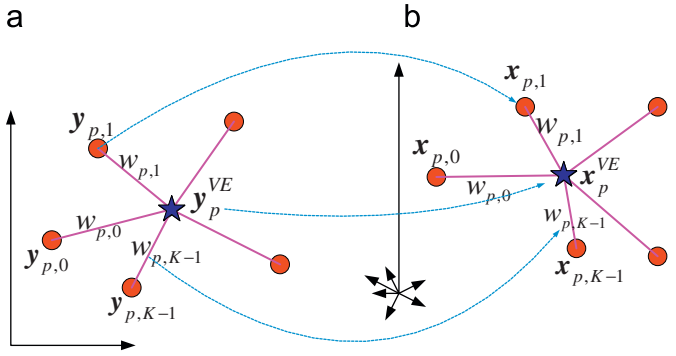

Fig. 10. An illustration of reconstruction of a virtual sample: (a) a virtual example $\boldsymbol{y}_p^{VE}$ and its K neighbors $\{\boldsymbol{y}_{p,0},\boldsymbol{y}_{p,1},\ldots,\boldsymbol{y}_{p,K-1}\}$ in a low-dimensional embedding and (b) reconstruction of vector $\boldsymbol{x}_p^{VE}$ by w_{pj} and its K neighbors $\{\boldsymbol{x}_{p,0},\boldsymbol{x}_{p,1},\ldots,\boldsymbol{x}_{p,K-1}\}$ in a high-dimensional data space.

the low-dimensional manifold embedding. To search these "holes" in the embedding, we first calculate the median d_m of all of the Euclidean distances between pairs of points y_i $(i=0,1,\ldots,n-1)$ and its K neighbors y_j $(j=0,\ldots,K-1)$. The median d_m is then used as a radius of the "searching ring" as shown in Fig. 9(c). Subsequently, moving the searching ring by a given experiential step size along the embedding, we can find some holes inside the manifold embedding. The centers of these holes, such as points 1 and 2, are the places where we generate virtual samples.

In our implementation, we search the holes in 2D for convenient computation. That is, we run the searching ring along the embedding in its 2D projection computed by Isomap. Among this projection, one can easily search the "holes" and control the "holes" found within the bound of the 2D embedding. We have also run this operation in the higher dimensional projection (e.g., 3D, 4D, etc.) and find the difference among the resulting classifiers is limited.

After finding the holes in the manifold embedding of the generated populations, the next step is to generate some virtual examples to fill in these holes. The algorithm is as follows (cf. Fig. 10):

Algorithm 2. Filling in holes.

Input: The embedding of face manifolds by Isomap;
Output: The embedding with some holes filled.
Step 1: In the low-dimensional manifold embedding, calculate its K neighbors $\{\boldsymbol{y}_{p,0},\boldsymbol{y}_{p,1},\ldots,\boldsymbol{y}_{p,K-1}\}$ of a virtual-example embedding vector $\boldsymbol{y}_p^{VE}$ $(p=0,\ldots,m-1)$, where an embedding vector corresponds to the center of a found hole, and the upper index "*VE*" denotes *virtual example* in the low-dimensional manifold embedding and m is the number of virtual face examples;
Step 2: Compute the weights ω_{pj} $(j=0,\ldots,K-1)$, which are the reciprocals of the geodesic distances between the virtual samples and their neighbors;
Step 3: For the K neighbors $\{\boldsymbol{y}_{p,0},\boldsymbol{y}_{p,1},\ldots,\boldsymbol{y}_{p,K-1}\}$ of a virtual-example embedding vector, get their corresponding original data points $\{\boldsymbol{x}_{p,0},\boldsymbol{x}_{p,1},\ldots,\boldsymbol{x}_{p,K-1}\}$ and compute the reconstructed vector $\boldsymbol{x}_p^{VE}$ by ω_{pj} and its K neighbors: $\boldsymbol{x}_p^{VE}=\sum_{j=0}^{K-1}\omega_{pj}x_{pj}$.

Whether do the found holes in the 2D embedding locate inside the data space? The reasons lie in the following two aspects: Theoretically, the basis of our "hole-filling" algorithm is run in the local linearity data space, which is also the fundamental of manifold. Intuitively, some observed facts also suggest that the faces distribution is continuous in the high dimension space. For example, component exchanging, local deformation (e.g., expression), and/or 3D morphing also generate legal faces. Therefore, in our case, using the neighbors and their corresponding weights to generate the virtual samples sounds reasonable.

Herein, an issue is to determine how many virtual samples should be generated. In general, it depends on two factors: one is the diversity of the collected dataset, and the other is the density of the final dataset. Specifically, on one hand, if the diversity of dataset increases, we usually generate more virtual samples and vice versa. On the other hand, if the dataset distributes sparsely, we generate more samples in comparison with the case when the dataset distributes densely.

4. Experimental results

In this section, some experiments are designed to evaluate the proposed method. We first show how our method improves the detection accuracy. Subsequently, we compare the performance of our detector with the state-of-the-art results on MIT+CMU frontal face test set [29].

4.1. Resampling a training set based on GAM

As discussed in Section 3, we perform the GA operations on an initial population and evaluate all the intermediate solutions of current generation by SNoW. After every 10 generations, we use a manifold-based method to resample the population to keep its size in control and optimize its distribution.

In general, it is usually difficult to model the manifolds from a large dataset by Isomap because the modeling needs to calculate the eigenvalues and eigenvectors of a large matrix. For example, if the number of a set is n (it is equal to 100,000 in our implementation), the matrix is $n\times n$ (i.e., 100,000×100,000). It is obviously difficult to calculate the eigenvalues and eigenvectors of such a large matrix. In order to avoid this problem, we divide a large set into some subsets according to their fitness, i.e., those solutions with similar fitness

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

Fig. 11. Some face samples generated by GAM.

are fallen into the same subset to ensure the compactness of a local manifold embedding. We have three reasons to use the fitness as a criterion to partition dataset: (1) the sample fitness contains the clustering similarity; (2) the fitness can be obtained easily because it has been attached for each solution during GA; and (3) it turns out to be simple and effective experimentally.

Subsequently, we learn a manifold of each subset by Isomap and compute the estimated geodesic distances in the high-dimensional space between pairs. These estimated geodesic distances are then used directly to subsample by deleting some examples from the database. In this way, we can obtain a smaller representative subset of a huge dataset. After checking the distribution of the resulting faces embedding, we generate more samples as discussed in Section 3.4.2. In our case, we use KNN mode and $K = 12$ for Isomap.

Our experimental results show that GAM converges after 40 iterations. The last population of GAM is used as the ultimate solutions. It consists of about 100,000 individuals, which includes 15,000 original samples and 85,000 their children. Some solutions generated by GAM are shown in Fig. 11. The first row is some examples by crossover. The second row is by mutation, which are four samples relighted under different lighting conditions. The third row is some virtual samples. Furthermore, we also show some generated false samples but automatically discarded by SNoW in the fourth row, where some due to inaccurate label and others due to the extreme lighting conditions of original samples or occlusion. Note that the generated samples do not change the symmetry of faces but the expressions might not be the same as the collected initial generation due to the crossover between two parents.

4.2. The relationship between GAM and other classifier

In this section, we use another classifier, i.e., AdaBoost classifier, to thest the generalization performance of the optimized training set and verify whether the generated solutions are independent of any special classifier though they are evaluated by SNoW during GA iteration. Simultaneously, we discuss the effects of the distribution and size of the resulting training set by GAM on the trained AdaBoost classifier [36].

4.2.1. A comparison of face detectors with different training sets

The goals of GAM are to expand the size of a training set and to improve its distribution by resampling to significantly improve the performance of an object detector. In order to illustrate the effectiveness of GAM on the trained classifier, we prepare seven different training sets to train seven different detectors. In our implementation, we use AdaBoost as a classifier and test the seven trained detectors on the MIT+CMU frontal face dataset, which consists of 130 images showing 507 upright faces. Here, the first training set is the initial population NoGAM (i.e., the original training set which consists of 15,000 samples as described in Section 3.2). The other six training sets are reproduced by GAM (after 40 generations) but in different sizes, as illustrated in Fig. 12(a). Meanwhile, GAM-*N* consists of *N* thousand individuals. For example, GAM-100 consists of 100,000 individuals. We use 100,000 non-faces as negative samples. Each detector is then trained using a bootstrapping approach similar to what described in [32] to increase the number of samples in the non-face set. The bootstrapping is carried out several times on a set of 16,536 images containing no faces.

The ROC curves of the seven detectors are illustrated as in Fig. 12(a). One can find that we obtain a detection rate of 90.73% with no false alarm using the detector trained on the set by GAM-100. It shows that the training set can be also used to train other classifiers rather than SNoW only. Similarly, from the ROC curves in Fig. 12(a), one can find out that GAM-15 works better than NoGAM (note that both GAM-15 and NoGAM contain 15,000 positive samples). We believe that this performance gain comes from the enriched variations and optimized distribution of face samples generated by GAM. In addition, from Fig. 12(a), we can find out that the performance of a detector trained on GAM-*M* is better than one trained on GAM-*N* ($N < M$). However, the difference becomes less and less when $N \geqslant 100$. This phenomenon shows that a well prepared training set is depended on two factors: a proper distribution and the size of samples.

4.2.2. A comparison of face detectors with different resampling schemes

To show the effects of two parts of our method—GA and manifold learning—on improving the accuracy of a face detector, we compare three classifiers trained on three different expanded training sets. The first one is expanded by GA and sampled by a manifold learning, which we call it *GAM*. The second one is expanded by GA and sampled by a clustering algorithm, which we call it *GAC* (i.e., GA+cluster). It is obtained by sampling the expanded solutions using the *K*-means clustering algorithm and the Euclidian distances among solutions as the similarity measure and $K = 12$. The third one is expanded by some basic image processing operations (such as rotation and shifting) and sampled by Isomap algorithm, which we call it *IPM* (i.e., image processing and manifold). It is obtained by randomly rotating

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

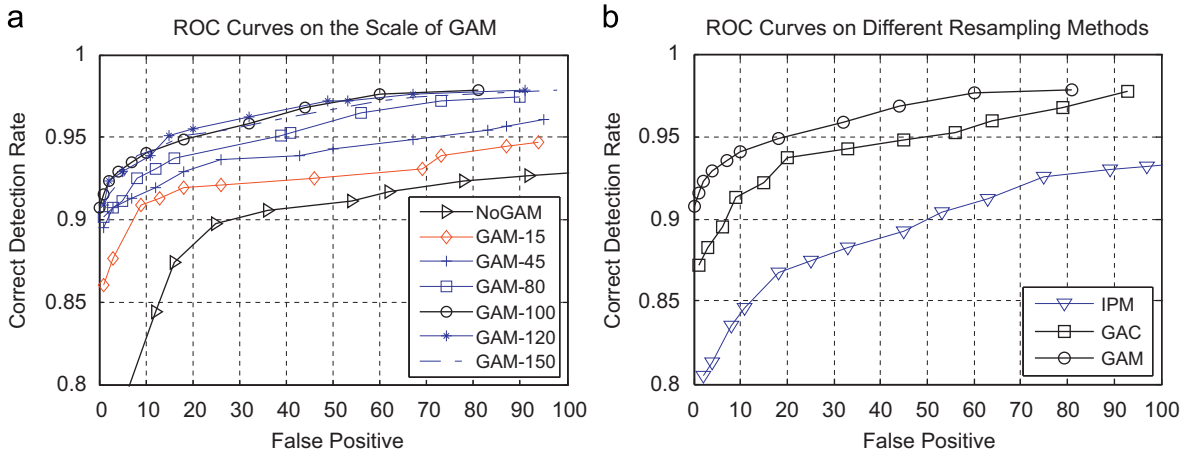

Fig. 12. A comparison of face detectors: (a) the ROC curves for the trained detectors by GAM but with the different size of the resulting population on the MIT+CMU frontal face test set and (b) the ROC curves on the validation set based on the four different subsampling methods: randomly sampling, evenly sampling based on the sorted sequence in decreasing order of the fitness value, sampling based on the manifold, and sampling by a clustering algorithm.

each faces in plane ranging from -15° to $+15^\circ$ and then shifting it from 0 to 1 pixel. Each training set is composed of 100,000 faces.

Similarly, we use 100,000 non-faces as negative samples and bootstrapping approach. The ROC curves of three detectors are illustrated in Fig. 12(b). From the ROC curves one can find out that the performance of the detector GAM is the best and the detector GAC outperforms the detector IPM obviously. This indicates that expanding the training set by GA is more effective to improve the performance of a face detector. In addition, the GAM outperforms the GAC shows the value of sample expansion by manifolds.

4.3. GAM with a small initial population

Although it is difficult to determine the minimum size of an initial population for GAM to work properly, we use a small subset (only 600 face samples) as input of GAM. Firstly, we randomly draw a subset (e.g., 10%, 600 samples) from the *original* 6000 faces (as mentioned in Section 3.2). By applying the preprocessing (i.e., affine transformations), we obtain 5000 face samples. The database is then divided into three subsets: training set (3000 samples), validating set (1000 samples), and testing set (1000 samples). During GA, the remaining ratio of solutions are less than θ_p (It is equal to 50%, here) of the current population. Note that the remaining ratio here is larger than the ratio for the case as mentioned in Section 3.3.2 because the former is composed of a smaller size (only 3000 samples) compared to the latter (15,000 samples). Otherwise, we keep all of those solutions whose fitness value is larger than the threshold θ_f (it is equal to 0.55, here). After every 10 generations, the population will contain $3000 \times 1.5^{10} \approx 172,995$ individuals. We cut down the population to 97,000 individuals as described in Section 3.4. After the resampling operations, we obtain 100,000 individuals, including 3000 individuals from the initial population and their 97,000 children.

We use four different face training sets to train an AdaBoost-based classifier. The first dataset is the subset including only 600 faces (we call it *NoGAM-Orig-Subset*). The second is the set after preprocessing, which consists of 3000 faces (we call it *NoGAM-PreProcess-Subset*). The third contains 100,000 face samples generated by GAM (we call it *GAM-Subset*). The fourth also contains 100,000 face images generated by GAM from the original training set (i.e., GAM-100 as mentioned in Section 4.2.1).

The resulting AdaBoost-based detectors, trained on these four different datasets, are also evaluated on the MIT+CMU frontal face test set. Their performances are compared in Fig. 13. From these

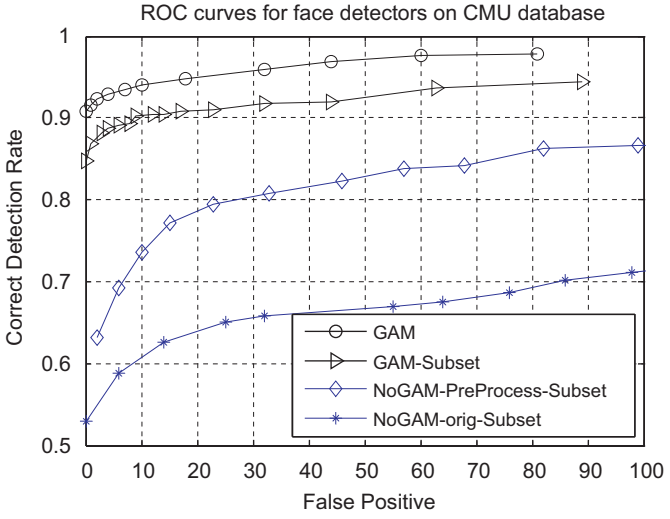

Fig. 13. The ROC curves of AdaBoost-based detectors trained from different small initial sets for GAM.

ROC curves, one can conclude that GAM-subset works much better than NoGAM-Orig-Subset (about 30% higher for 10 false alarms) and NoGAM-PreProcess-Subset (about 17% higher for 10 false alarms). Especially, the results of GAM-Subset are comparable to that of GAM, although the former is expanded from 600 samples while the latter from 6000 samples. It demonstrates that GAM can still work elegantly even with a small initial population.

As to the distribution of the initial population for GAM, our experience shows that some variations, such as out-of-plane rotation, skins and well-focused face images, etc., are necessary, while other variations such as lighting, shadow, blurring, and various organs, are optional since the proposed method can generate these variations automatically. Herein, we need well-focused face samples for GAM because too blurred samples imply little useful statistical information.

4.4. A comparison with the state-of-the-art results

As shown in Fig. 14, we discuss a performance comparison of the proposed method with some existing face detection algorithms

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

reported results on the MIT+CMU test set, such as in Brubaker [9], Garcia [16], Li [23], Schneiderman [30], and Viola [36].

It is obvious that our detector (the AdaBoost-based classifier trained by the optimized training set, i.e., GAM-100 mentioned in Section 4.2.1) compares favorably with the others. However, different criteria (e.g., number of training examples involved, execution time, and the number of scanned windows during detection, etc.) can be used to favor one over another, which makes it difficult to evaluate the performance of different methods even though they use the same benchmark datasets [39]. Fig. 15 shows some examples of the detected faces from MIT+CMU frontal face test set, web site, sports video games and video surveillance, etc.

In general, the selection of training examples and execution time change the performance of a detector significantly. We provide some experimental results on how to organize the training set and explore the performance of a detector. Specifically, for the training examples involved, one can draw a conclusion from Figs. 12(a) and 13 that the performance of a trained face detector is significantly influenced by the different training set. For better views, we consider the face detection accuracies with false alarms (FAs) being equal to 10 and show them in Table 1. Each row in this table corresponds to the different training sets which are the same as Figs. 12(a) and 13. Specifically, the first three rows are taken from Fig. 13 and the others from Fig. 12a. One can find out that the detector trained with the set GAM-100 generated by the proposed method performs the best. For more details, please refer to Sections 4.2 and 4.3.

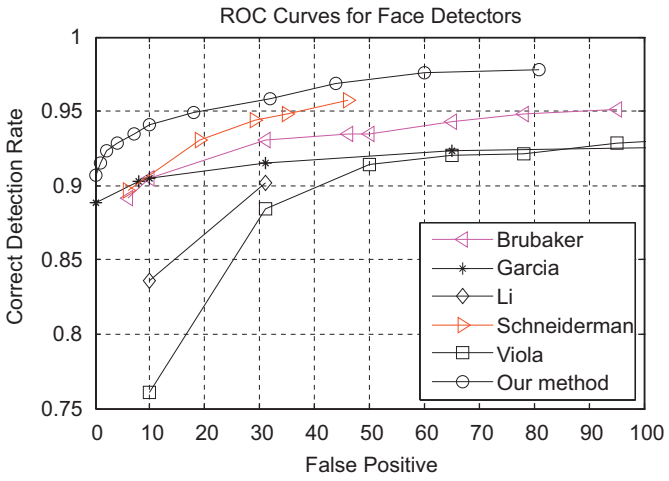

Fig. 14. A performance comparison of the proposed method with other published methods.

Table 1
Face detection accuracies varying with the training examples involved (FAs = 10).

Different training sets	Accuracies (%)
NoGAM-Orig-Subset	60.65
NoGAM-Preprocess-Subset	73.57
GAM-Subset	90.34
NoGAM	82.84
GAM-15	90.93
GAM-45	91.12
GAM-80	92.50
GAM-100	94.08
GAM-120	94.08
GAM-150	94.28

Table 2
Face detection accuracies varying with the total execution time on the MIT+CMU frontal face test set (FA = 0).

Scaling factor (α_1)	Step size (s_x, s_y)	Total execution time (s)	Accuracies (%)
0.5	(1, 1)	7.53	65.88
1	(2, 2)	14.88	82.64
1	(1, 2)	16.05	86.98
1	(2, 1)	26.87	90.53
1	(1, 1)	31.25	90.73

For the execution time, we show some experimental results in Table 2. Here, we change the execution time of a detector by the two approaches: (1) resizing the input image by a scaling factor α_1 before the detector runs across the image and (2) varying the step size (s_x, s_y) of the detector when it scans the sub- windows across the input image. Specifically, for the first approach, there are two scaling factors to change the execution time of a detector: one is the scaling factor α_1 before the detector runs; the other is the scaling factor α_2 used during the detector running to build the input image pyramid. Here, we change the first scaling factor (α_1) and keep the other factor α_2 as a constant ($\alpha_2 = 1.2$) since the factor α_1 changes the execution time more significantly than the factor α_2. For the second approach, the step sizes in Table 2 mean that a detector can be moved in steps of several pixels across the image in horizontal and vertical directions respectively. For example, the step sizes ($s_x = 1$, $s_y = 2$) in the third row mean that the detector moves in steps of one pixel in horizontal direction and two pixels in vertical direction when scanning the sub-windows across the image.

Here, the detector is the AdaBoost-based classifier trained by the optimized training set (i.e., GAM-100 mentioned in Section 4.2.1). The test set is the MIT+CMU frontal face test set. Here, the execution time in Table 2 is the total time that the detector runs on the test set. The first row shows the results with the scaling factor $\alpha_1 = 0.5$ and steps ($s_x = 1$, $s_y = 1$). Thus the minimum detectable faces are 40×40

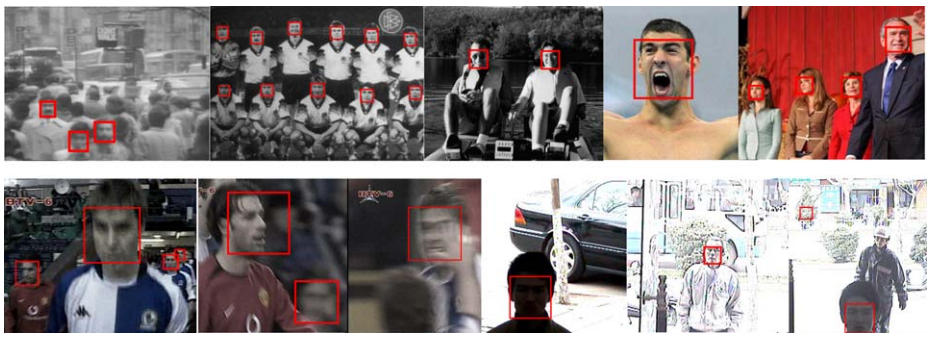

Fig. 15. Some testing results of our trained face detector.

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

since they are 20×20 for the original images. By image resizing the detector can work much faster and it is often safe for the automatic face recognition since the face resolution for recognition is usually larger than 40×40. However, the performance drops greatly since many faces in the MIT+CMU testing set are smaller than 40×40. Another method to make a detector run faster is to vary the step size horizontally and/or vertically as shown in Table 2 from the second row to the fourth row (and $\alpha_1 = 1$). One can find out that varying the step size horizontally outperforms varying the step size vertically. In addition, running the detector pixelwise obtains the best performance. However, the performance differences resulted from the different step sizes (e.g. from the second row to the fifth row in Table 2) is related to the test set. We believe that the significant difference in Table 2 is brought about by the diverse faces of the MIT+CMU test set.

5. Conclusions

In this paper, we have proposed a novel method to optimize a training set by expanding the size and reshaping the distribution of a face sample set using GAM. The optimized training set can improve the performance of a classifier. In our case, the proposed method can generate new face samples by crossover and mutation operations. These generated samples can simulate wide variations, such as face variations in daily life and image variations due to lighting and quality conditions. The solutions are further resampled by manifolds to control the size of a population and optimize its distribution. We use the optimized face set to train an AdaBoost-based classifier and test the trained classifier on the MIT+CMU frontal face test set, and a detection rate 90.73% is achieved without a false alarm. Furthermore, the experimental results show that the proposed method improves the performance of an AdaBoost-based classifier significantly, especially with low false alarms. For example, the face detector trained from the optimized dataset is 17.98% better for 10 false alarms compared with that of the AdaBoost-based face detector proposed by Viola and Jones.

While most of the researchers pay much attention to the design of the classifiers and resampling (by bootstrap, for instance) the negative samples (/non-faces), the experimental results of this paper indicate that optimization of the training set by resampling of positive samples (/faces) can also improve the performance of the classifier significantly. Furthermore, the proposed method is a fully automatic procedure and can significantly facilitate the preparation of a training set in order to obtain a well-performed face detector. The proposed method can also be applied to other object detection and recognition tasks for automatically obtaining an optimal training set and enhancing detection and recognition performance of a classifier.

Acknowledgments

This research is partially sponsored by Natural Science Foundation of China under contract 60533030, U0835005, 60772071, National Basic Research Program of China (973 Program) under contract 2009CB320902, the 100 Talents Program of CAS, and Hi-Tech R & D Program of China under contract 2007AA01Z163.

References

[1] J. Atkinson-Abutridy, C. Mellish, S. Aitken, A semantically guided and domain-independent evolutionary model for knowledge discovery from texts, IEEE Transactions on Evolutionary Computation 7 (2003) 546–560.
[2] W.H. Au, K.C.C. Chan, X. Yao, A novel evolutionary data mining algorithm with applications to churn prediction, IEEE Transactions on Evolutionary Computation 7 (2003) 532–545.
[3] R. Basri, D. Jacobs, Lambertian reflectance and linear subspaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2003) 218–233.
[4] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: Advances in Neural Inform. Processing Systems, vol. 14, MIT Press, 2002, pp. 585–591.
[5] D. Beymer, T. Poggio, Face recognition from one example view, in: Proceedings of the Fifth International Conference on Computer Vision, 1995, pp. 500–507.
[6] S. Bhattacharyya, O.V. Pictet, G. Zumbach, Knowledge-intensive genetic discovery in foreign exchange markets, IEEE Transactions on Evolutionary Computation 6 (2002) 169–181.
[7] M. Brameier, W. Banzhaf, A comparison of linear genetic programming and neural networks in medical data mining, IEEE Transactions on Evolutionary Computation 5 (2001) 17–26.
[8] L. Breiman, Bagging predictors, Machine Learning (1996) 123–140.
[9] S.C. Brubaker, M.D. Mullin, J.M. Rehg, Towards optimal training of cascaded detectors, in: Proceedings of the European Conference on Computer Vision, 2006, pp. 325–337.
[10] J.R. Cano, F. Herrera, M. Lozano, Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study, IEEE Transactions on Evolutionary Computation 7 (2003) 561–575.
[11] J. Chen, X. Chen, W. Gao, Expand training set for face detection by GA resampling, in: Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004, pp. 73–79.
[12] A.C. Davison, D.V. Hinkley, Bootstrap Methods and Their Application, Cambridge University Press, Cambridge, UK, 1997.
[13] B. Efron, The Jackknife, the Bootstrap and Other Resampling Plans, Society of Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1982.
[14] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the 13th International Conference on Machine Learning, 1996, pp. 148–156.
[15] B. Fröba, A. Ernst, Fast frontal-view face detection using a multi-path decision tree, in: Proceedings of Audio- and Video-based Biometric Person Authentication, 2003, pp. 921–928.
[16] C. Garcia, M. Delakis, Convolutional face finder: a neural architecture for fast and robust face detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 1408–1423.
[17] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Reprinted in 1992, MIT Press.
[18] C. Huang, H. Ai, Y. Li, S. Lao, High-performance rotation invariant multiview face detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 671–686.
[19] C. Huang, H. Ai, T. Yamashita, S. Lao, M. Kawade, Incremental learning of boosted face detector, in: Proceedings of the IEEE International Conference on Computer Vision, 2007
[20] A.K. Jain, R.P.W. Duin, J.C. Mao, Statistical pattern recognition: a review, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 4–37.
[21] M. Kirby, L. Sirovich, Application of the karhunen–Loève procedure for the characterization of human faces, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (1990) 103–108.
[22] V. Krueger, Gabor wavelet networks for object representation, Ph.D. Dissertation, Christian-Albrechts University, Kiel, Germany, January 2001.
[23] S. Li, Z. Zhang, Floatboost learning and statistical face detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 1112–1123.
[24] J. Liu, X. Tang, Evolutionary search for faces from line drawings, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005) 861–872.
[25] X. Lu, A.K. Jain, Resampling for face recognition, in: Proceedings of Internal Conference on Audio- and Video-Based Biometric Person Authentication, 2003, pp. 869–877.
[26] M.-T. Pham, T.-J. Cham, Fast training and selection of Haar features using statistics in boosting-based face detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2007.
[27] R. Ramamoorthi, P. Hanrahan, On the relationship between radiance and irradiance: determining the illumination from images of a convex Lambertian object, Journal of the Optical Society of America A 18 (2001) 2448–2459.
[28] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
[29] H.A. Rowley, S. Baluja, T. Kanade, Neural network-based face detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 23–38.
[30] H. Schneiderman, Feature-centric evaluation for efficient cascaded object detection, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, pp. 29–36.
[31] M. Srinivas, L.M. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transactions on System, Man, and Cybernetics, Part A 24 (4) (1994) 656–667.
[32] K.K. Sung, T. Poggio, Example-based learning for view-based human face detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 39–51.
[33] X. Tan, S. Chen, Z.-H. Zhou, F. Zhang, Face recognition from a single image per person: a survey, Pattern Recognition 39 (2006) 1725–1745.
[34] J.B. Tenenbaum, V. Silva, J. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.
[35] F. Torre, R. Gross, S. Baker, V. Kumar, Representational oriented component analysis (ROCA) for face recognition with one sample image per training class, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp. 266–273.
[36] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001, pp. 511–518.
[37] S. Yan, S. Shan, X. Chen, W. Gao, J. Chen, Matrix-structural learning (MSL) of cascaded classifier from enormous training set, in: Proceedings of the IEEE

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006

Computer Society Conference on Computer Vision and Pattern Recognition, 2007.
[38] M.H. Yang, D. Roth, N. Ahuja, A SNoW-Based Face Detector, Advances in Neural Information Processing Systems, vol. 12, MIT Press, Cambridge, MA, 2000 pp. 855–861.
[39] M.H. Yang, D. Kriegman, N. Ahuja, Detecting faces in images: a survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 34–58.
[40] J. Yuan, J. Luo, Y. Wu, Mining compositional features for boosting, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2008.
[41] C. Zhou, W. Xiao, T.M. Tirpak, P.C. Nelson, Evolving accurate and compact classification rules with gene expression programming, IEEE Transactions on Evolutionary Computation 7 (2003) 519–531.
[42] Z.H. Zhou, Y. Jiang, NeC4.5: neural ensemble based C4.5, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 770–773.

About the Author—JIE CHEN received the M.S. and Ph.D. degrees from the Harbin Institute of Technology, Harbin, China, in 2002 and 2007, respectively. He is currently working as a PostDoc in Machine Vision Group, Department of Electrical and Information Engineering, University of Oulu, Finland.
His research interests include pattern recognition, computer vision, machine learning, and watermarking.

About the Author—XILIN CHEN received the B.S., M.S., and Ph.D. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 1988, 1991, and 1994, respectively.
He was a Professor with the Harbin Institute of Technology from 1999 to 2005. He was a Visiting Scholar with Carnegie Mellon University, Pittsburgh, PA, from 2001 to 2004. He has been with the Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing, since August 2004.
He is also with the Key Laboratory of Intelligent Information Processing, CAS. His research interests are image processing, pattern recognition, computer vision, and multimodal interfaces.
Dr. Chen has served as a program committee member for more than 20 international and national conferences. He has received several awards, including China's State Scientific and Technological Progress Award in 2000, 2003, and 2005 for his research work.

About the Author—JIE YANG received the Ph.D. degree in electrical engineering from the University of Akron, Akron, OH, in 1994.
He is currently a Senior Systems Scientist at the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. He pioneered the hidden Markov model for human performance modeling in his Ph.D. dissertation research. He joined the Interactive Systems Laboratories in 1994, where he has been leading research efforts to develop visual tracking and recognition systems for multimodal human–computer interaction. He developed adaptive skin color modeling techniques and demonstrated software-based real-time face tracking system in 1995. He has been involved in the development of many multimodal systems in both intelligent working spaces and mobile platforms. He has been working on automatic detection of text from natural scenes over last six years, with applications to automatic sign translation and intelligent driving assistant systems. Dr. Yang is an Associate Editor of the IEEE Transactions on Multimedia. His current research interests include multimodal interfaces, computer vision, and pattern recognition.

About the Author—SHIGUANG SHAN received the B.S. and M.S. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 1997 and 1999, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing, in 2004.
He is currently an Associate Researcher and serves as the Vice-Director with the Digital Media Center, Institute of Computing Technology, CAS. He is also with the Key Laboratory of Intelligent Information Processing, CAS. He is also the Vice-Director with ICT-ISVision Joint R&D Laboratory for Face Recognition. His research interests cover image analysis, pattern recognition, and computer vision. He is particularly focusing on face recognition-related research topics.

About the Author—RUIPING WANG received the B.S. degree in applied mathematics from Beijing Jiaotong University, Beijing, China, in 2003. He is currently working toward the Ph.D. degree in the Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing. He is currently with the Key Laboratory of Intelligent Information Processing, CAS. His research interests include computer vision, pattern recognition, and machine learning.

About the Author—WEN GAO received the B.Sc. degree from the Harbin University of Science and Technology, Harbin, China, in 1982, the M.Sc. degree from the Harbin Institute of Technology, Harbin, in 1985, all in computer science, and the Ph.D. degree in electronics engineering from the University of Tokyo, Tokyo, Japan, in 1991.
He was with the Harbin Institute of Technology, in 1985, where he served as a Lecturer, Professor, and the Head of the Department of Computer Science until 1995. He was with Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing, from 1996 to 2005. During his professor career with CAS, he was also appointed as Director with the Institute of Computing Technology, Executive Vice-President with the Graduate School, as well as the Vice President with the University of Science and Technology of China. He is currently a Professor with the School of Electronics Engineering and Computer Science, Peking University, Beijing. He has published four books and over 300 technical articles in refereed journals and proceedings in the areas of multimedia, video compression, face recognition, sign language recognition and synthesis, image retrieval, multimodal interface, and bioinformatics.
Dr. Gao is an Associate Editor of IEEE Transactions on Circuits and Systems for Video Technology, an Associate Editor of the IEEE Transactions on Multimedia, an Editor of the *Journal of Visual Communication and Image* Representation, and the Editor-in-Chief of the *Journal of Computer* (*in Chinese*). He received the Chinese National Award for Science and Technology Achievement in 2000, 2002, 2003, and 2005.

Please cite this article as: J. Chen, et al., Optimization of a training set for more robust face detection, Pattern Recognition (2009), doi: 10.1016/j.patcog.2009.02.006