Adaptive Boosting of Support Vector Machine Component Classifiers Applied in Face Detection

S.M. Valiollahzadeh; A. Sayadiyan; and F. Karbassian

ABSTRACT

Face detection has found an important role in many computer vision applications. But automatic face detection is a difficult task because of broad range of variability due to illumination, facial expression, occlusion, scale, and aging. In this paper a novel algorithm, which combines adaptive boosting (AdaBoost) with Support Vector Machine (SVM) as weak component classifiers is proposed. 2D non-standard haar features where used to represent the face. In the offline phase feature selection is used to extract the most discriminative features from the set of overcomplete 32000 features. In the online phase the selected features are used in face detection algorithm. Experiments using frontal images from MIT–CMU test set show 98.2% correct face detection superior to other popular AdaBoosted methods both in accuracy and speed.

KEYWORDS

Face Detection, AdaBoost, Support Vector Machine (SVM), ComponentLearn, AdaBoostedSVM

1. INTRODUCTION

Automatic face detection has been studied widely over the past decade due to its ever-increasing applications such as video surveillance, human-computer interaction, face image database management, and many others. It is also the first task in face recognition [1], [2], which has a lot of commercial applications such as security access control, matching of controlled-format photographs like passports, ID cards, driver's licenses [3]-[6].

Face and nonface classification is a difficult task as faces are nonrigid objects and have a high degree of variability due to illumination, face rotation, facial expression, occlusion, scale, location, and aging.

Face detection methods generally learns statistical models of face and nonface images, and then apply two-class classification rules to discriminate between face and nonface patterns [4], [5].

In this paper, using skin color, the region of interest in a color image is significantly reduced. The image is then converted into grayscale and the slide-window technique is used to scan the image to find faces. To overcome the difficulty encountered by size, growing slide-window is utilized. Large amount of non-standard 2D haar features (almost 32000), are extracted to represent the region of interest and feature selection is used to find most discriminative ones. AdaBoostedSVM is utilized to increase the accuracy of classification. It is shown from experimental results that proposed method is superior to conventional AdaBost algorithms both in accuracy and speed. In section 2, background information is introduced. Section 3 discusses on using skin color to reduce the region of interest. Feature selection is discussed in section 4. In section 5, the face detection algorithm is proposed. Section 6 presents experimental results and conclusion is given in section 7.

2. BACKGROUND

Earlier face detection research was based on correlation or template matching, matched filtering, subspace methods, deformable templates, etc. [7], [8]. For comprehensive surveys of these methods, see Refs. [2], [6]. In contrast, recent face detection approaches, however, rely on statistical analysis and machine learning techniques to find the relevant characteristics of face and nonface images [2], [9]. The learned characteristics are in the form of distribution models [10] or discriminant functions [11]. Meanwhile, dimensionality reduction is usually carried out for computational efficiency.

One of the major developments in machine learning in the past decade is the ensemble method, which finds a highly accurate classifier by combining many moderately accurate component classifiers. Two of the commonly used techniques for constructing ensemble classifiers are Boosting [12] and Bagging [13]. In Comparison with Bagging, Boosting outperforms when the data do not have much noise [14], [15]. Among popular Boosting methods, AdaBoost [16] establishes a collection of weak component classifiers by maintaining a set of weights over training samples and adjusting them adaptively after each Boosting iteration. The weights of the misclassified samples by current component classifier will be increased while the weights of the correctly classified samples will be decreased. To implement the weight updates in AdaBoost, several algorithms have been proposed [17]. The success of AdaBoost can be attributed to its ability to enlarge the margin [18], which could enhance AdaBoost's generalization capability. Decision trees [19] or neural networks [20], [21] have already been employed as component classifiers for AdaBoost. These studies showed good generalization performance; however, the suitable tree size determination is still a problem when decision trees are used as component classifiers. Besides, controlling the complexity in order to avoid overfitting is also a problem, when Radial Basis Function (RBF) neural networks are used as component classifiers. Moreover, one has to decide the optimum number of centers and also sets the width values of the RBFs.

Diversity is known to be an important factor which affects the generalization accuracy of ensemble classifiers [17], [22]. Kuncheva [23] and Windeatt [24] proposed several methods to quantify the diversity. There is an accuracy/diversity dilemma in AdaBoost [19], i.e. the more accurate two component classifiers become, the less they can disagree with each other. Only when the accuracy and diversity are well balanced, can the AdaBoost demonstrate excellent generalization performance. This phenomenon in existing AdaBoost algorithms is not well studied yet.

Support Vector Machine (SVM) was developed based on the theory of Structural Risk Minimization [25]. Using a kernel trick to map the training samples from an input space to a high dimensional feature space, SVM finds an optimal separating hyperplane in the feature space and uses a regularization parameter, C, to control its model complexity, and training error. One of the popular kernels used by SVM is the RBF kernel, including a parameter known as Gaussian width, σ. In contrast to the RBF networks, SVM with the RBF kernel (RBFSVM in short) can automatically determine the number and location of the centers and the weight values [26]. Also, it can effectively avoid overfitting by selecting proper values of C and σ. From the performance analysis of RBFSVM, σ is a more important parameter compared to C. Although RBFSVM cannot learn well when a very low value of C is used, its performance largely depends on the σ value if a roughly suitable C is given. Therefore over a range of suitable C, the performance of RBFSVM can be conveniently altered by simply adjusting the value of σ.

3. SKIN COLOR

In order to assign each part of the image a probability of being face or not, one needs a reliable reference which must be adaptive with different color skins and to different lighting conditions. The RGB space representation of color images is not suitable for this aim. In RGB space, the triple component represents not only color but also luminance. Luminance may vary across a person's face due to ambient lighting and therefore is not a reliable measure for separating skin from nonskin.

Chromatic colors also known as pure colors can be effectively used to segment color images in many applications. The skin color distribution of different people is found to be clustered in a small area of chromatic color space. Skin color of different people is close and differs mainly in intensity [1].

In our algorithm for face detection, first the probability density function of human skin in chromatic color space is determined using samples of different races. As the skin color information is extracted from color images, the data is filtered using a low pass filter to reduce the noise effect. The skin color histogram can be represented by a Gaussian model. Better performance will yield using GMM model to predict the histogram. Utilizing obtained histogram the likelihood of skin for any pixel is determined. Hence the skin color model transforms a color image into a grayscale where the gray value shows the likelihood of skin of the pixel.

Using an appropriate threshold, the grayscale image can be transformed into a binary image of skin and nonskin regions. This significantly reduces the region of interest in color images.

4. FEATURE SELECTION

There are many motivations for using features rather than the pixels directly. One of the most common reasons is that feature-based systems operate much faster than the pixel-based systems. But selecting appropriate features to represent faces and proper classification of these features are two central issues to face detection systems.

In this paper, four types of 2D haar-like basis functions are used to build the feature pool [27]. More specifically three types of features have been used. Fig. 1 (a) and (b) show *two-rectangle features*. Their value is difference between the sums of pixels within two rectangular regions. The rectangular regions have the same size and are horizontally or vertically adjacent. A *three-rectangle* feature, shown in Fig. 1 (c), computes the sum of side rectangles minus the centered one. Finally a *four-rectangle feature*, shown in Fig. 1 (d), computes the difference between diagonal pairs of rectangles. The feature can be computed efficiently with integral image [28]. The main advantage of using these features is that they can be rescaled easily avoiding to calculate a pyramid of images and thus accelerates the detector greatly. Given that the base resolution of the detector is 32×32, the exhaustive set of rectangle features is quite large, over 32,000. Unlike the haar basis, the set of rectangle features is overcomplete.

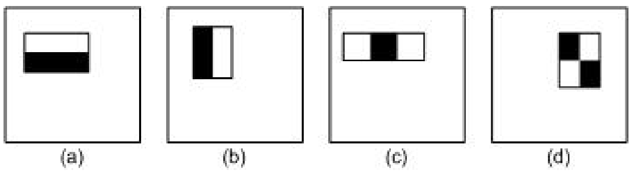

Figure 1: Rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles is subtracted from the sum of pixels in the dark rectangles. (a) and (b) *two- rectangle features,* (c*) three - rectangle feature,* and (d) *four-rectangle feature.*

For each scale level, we rescale the features and record the relative coordinate of the rescaled features to the top-left of integral image in look-up-table (LUT). Looking up the value of the rescaled rectangle's coordinate, features will be calculated with relative coordinate. Image variance σ is used to correct lighting [28]. Rescaling needs to round rescaled coordinates to nearest integer, which would degrade the performance [29]. To reduce the rounding error the features are normalized by acreage.

Fig. 2 shows how a rectangular sum can be calculated. Using the integral image, to calculate the value of rectangle *D*, one needs to follow a four array reference procedure. From definition of integral images the value at any point *x* is the sum of the pixels left and above of it. Hence the value of rectangle *D* can be found as $x_4 + x_1$ - $(x_2 + x_3)$. The difference between two rectangular sums can be computed in an eight array reference procedure, but as the *two-rectangle features* defined above involve adjacent rectangular sums they can be computed in six array references. *Three-rectangle* and *four-rectangle features* are computed in eight and nine array references respectively.

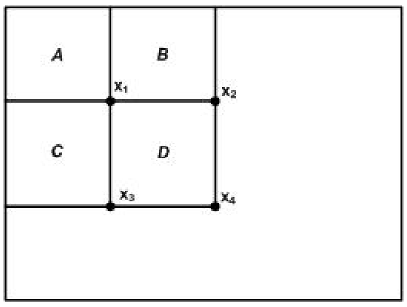

Figure 2: The sum of the pixels within rectangle *D* can be computed with four array references. The value of the integral image at location x_1 is the sum of the pixels in rectangle *A*. The value at location x_2 is *A*+*B*, at location x_3 is *A*+*C*, and at location x_4 is *A*+*B*+*C*+*D*. The sum within *D* can be computed as $x_4 + x_1 - (x_2 + x_3)$.

5. STATISTICAL LEARNING

This section describes boost-based learning methods to construct face/nonface classifier, and proposes a new boosting algorithm which improves boosting learning.

A. AdaBoost Learning

Given a set of training samples, AdaBoost [30] provides a probability distribution, *w*, over these samples. This distribution is initially uniform. Then, AdaBoost algorithm calls WeakLearn algorithm repeatedly in a series of cycles. At cycle *t*, AdaBoost provides training samples with a distribution w^t to the Weak Learn algorithm.

AdaBoost constructs a composite classifier by sequentially training classifiers while putting more and more emphasis on certain patterns.

For two class problems, a set of *N* labeled training examples $(y_1, x_1), \ldots, (y_N, x_N)$ are given where $y_i \in \{+1, -1\}$ is the class label associated with example x_i. For face detection, x_i is an image sub-window of a fixed size (for our system 32×32) containing an instance of the face $(y_i = +1)$ or nonface $(y_i = -1)$ pattern. In the notion of AdaBoost (see Algorithm 1), a stronger classifier is a linear combination of *M* weak classifiers.

In boosting learning, each sample x_i is associated with a weight w_i and the weights are updated dynamically using a multiplicative rule according to the errors in previous learning so that more emphasis is placed on those samples which are erroneously classified by the weak classifiers learned previously.

Greater weights are given to weak learners with lower errors. The important property of AdaBoost is that if the weak learners consistently have accuracy only slightly better than half, then the error of the final hypothesis decays to zero exponentially. This means that the weak learners need to be only slightly better than random.

Furthermore, since proposed AdaBoost with SVM provides a convenient way to control the classification accuracy of each weak learner, it also provides an opportunity to deal with the well-known accuracy/diversity dilemma in Boosting methods.

Algorithm 1. The AdaBoost Algorithm [30]

```
1. Input: Training sample
Input: a set of training samples with labels $(y_1, x_1), \ldots, (y_N, x_N)$,
ComponentLearn algorithm, the number of cycles $T$.

2. Initialize: the weights of training samples: $w_i^1 = 1/N$, for all
$i$=1, …, $N$

3. Do for $t$ =1, …, $T$
    (1)Use ComponentLearn algorithm to train the component
        classifier $h_t$ on the weighted training sample set.
    (2)Calculate the training error of $h_t$:
        $\varepsilon_t = \sum_{1=1}^{N} w_i^t, y_i \neq h_t(x_i)$.
    (3)Set weight of component classifier $h_t$:
        $h_t : \alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
    (4)Update the weights of training samples:
        $w_i^{t+1} = \frac{w_i^t \exp\{-\alpha_t y_i h_t(x_i)\}}{C_t}$
        where $C_t$ is a normalization constant, and
        $\sum_{i=1}^{N} w_i^{t+1} = 1$
4. Output: $f(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$.
```

B. SVM-based Approach for Classification

SVM relies on a linear separation in a high dimension feature space where the data have been previously mapped, in order to take into account the eventual nonlinearities of the problem.

If we assume the training set $X = (x_i)_{i=1}^{l} \subset \mathrm{R}^R$ where l is the number of training vectors, **R** stands for the real numbers set and *R* is the number of modalities, is labelled with two class targets $Y = (y_i)_{i=1}^{l}$, where :

$$y_i \in \{-1, +1\} \quad \Phi : \mathrm{R}^R \to F \tag{1}$$

Mapping the data into a feature space, maximizing the minimum distance in space *F* between $\Phi(X)$ and the separating hyperplane $H(w,b)$ is a good means of reducing the generalization risk [25], where

$$H(w,b) = \{f \in F \mid <w,f>_F + b = 0\},\quad (<> \textit{means} \text{ dot product}) \tag{2}$$

The optimal hyperplane can be obtained solving the convex quadratic programming (QP) problem [25]:

$$\begin{aligned} &\text{Minimize} && \frac{1}{2}\|w\|^2 + c\sum_{i=1}^{l}\xi_i \\ &\text{with} && y_i(<w,\Phi(X)>+b) \geq 1-\xi_i \quad i=1,\dots,l \end{aligned} \tag{3}$$

Constant C and slack variables X take into account the eventual non-separability of $\Phi(X)$ into F.

In practice, this criterion is softened to the minimization of a cost factor involving both the complexity of the classifier and the degree to which marginal points are misclassified, and the trade-off between these factors is managed through a margin of error parameter (usually designated C) which is tuned through cross-validation procedures.

Although the SVM is based upon a linear discriminator, it is not restricted to making linear hypotheses. Nonlinear decisions are made possible by a nonlinear mapping of the data to a higher dimensional space. The phenomenon is analogous to folding a flat sheet of paper into any three-dimensional shape and then cutting it into two halves, the resultant nonlinear boundary in the two-dimensional space is revealed by unfolding the pieces.

The SVM's non-parametric mathematical formulation allows these transformations to be applied efficiently and implicitly: the SVM's objective is a function of the dot product between pairs of vectors; the substitution of the original dot products with those computed in another space eliminates the need to transform the original data points explicitly to the higher space. The computation of dot products between vectors without explicitly mapping to another space is performed by a kernel function.

The nonlinear projection of the data is performed by this kernel functions. There are several common kernel functions that are used such as the linear, polynomial kernel $(K(x,y) = (<x,y>_{R^R} + 1)^d$ and the sigmoidal kernel $(K(x,y) = \tanh(<x,y>_{R^R} + a))$, where x and y are feature vectors in the input space.

The other popular kernel is the Gaussian (or "radial basis function") kernel, defined as:

$$K(x,y) = \exp\left(\frac{-|x-y|^2}{(2\sigma^2)}\right) \tag{4}$$

Where σ is a scale parameter, and x and y are feature-vectors in the input space. The Gaussian kernel has two hyper parameters to control performance C and the scale parameter σ . In this paper we used radial basis function (RBF).

C. AdaBoosted SVM-based Component Classifier

We combine SVM with AdaBoost to improve its capability in classification. When applying Boosting method to strong component classifiers, these classifiers must be appropriately weakened in order to benefit from Boosting [19].

Resampling is used to train AdaBoost [30]. One needs to train weak classifiers (SVM classifier) to obtain best Gaussian width, σ and the regularization parameter, C, for optimizing strong classifier (AdaBoost classifier). Hence, SVM with RBF kernel is used as weak learner for AdaBoost. A relatively large σ value, which corresponds to a SVM with RBF kernel with relatively weak learning ability, is preferred. Both resampling and reweighting can be used to train AdaBoost. The algorithm is as follows:

```
Algorithm 2. The AdaBoostedSVM Algorithm
1. Input: Training sample
Input: a set of training samples with labels $(y_1,x_1),\dots,(y_N,x_N)$,
The initial $\sigma = \sigma_{ini}$ , $\sigma_{\min}$ , $\sigma_{step}$
2. Initialize: the weights of training samples: $w_i^1 = 1/N$ , for all $i$=1, …, $N$
3. Do while $\sigma > \sigma_{\min}$
    (1)Use RBFSVM to train on the weighted training sample set.
    (2)Calculate the training error of $h_t$ :
        $\varepsilon_t = \sum_{1=1}^{N} w_i^t, y_i \neq h_t(x_i)$.
    (3) if $\varepsilon_t > .5$ ,decrease $\sigma$ value by $\sigma_{step}$ and goto(1)
    (4)Set weight of component classifier $h_t$ :
        $h_t : \alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
    (5)Update the weights of training samples:
        $w_i^{t+1} = \frac{w_i^t \exp\{-\alpha_t y_i h_t(x_i)\}}{C_t}$
      where $C_t$ is a normalization constant, and
        $\sum_{i=1}^{N} w_i^{t+1} = 1$
4. Output: $f(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$.
```

D. Face Detection System

The SVM-based component classifier and AdaBoost algorithm are used for the classification of each stage of cascaded classifiers.

Fig. 3 shows the architecture of our face detection system. Firstly the region of interest in color images is reduced using skin color information. Color images are then converted into grayscale. At the first step a 32×32 slide window scans the image. Subset of two dimensional non-standard haar wavelet features selected in the offline feature selection procedure is applied to weak classifiers. Strong classifier is constructed as a linear combination of weak classifiers utilizing AdaBoosted SVM-based component classifier learning. The detector is made of three strong classifiers in cascade. The slide-window is then resized and the procedure is repeated until the pyramidly-growing slide-window covers the whole image.

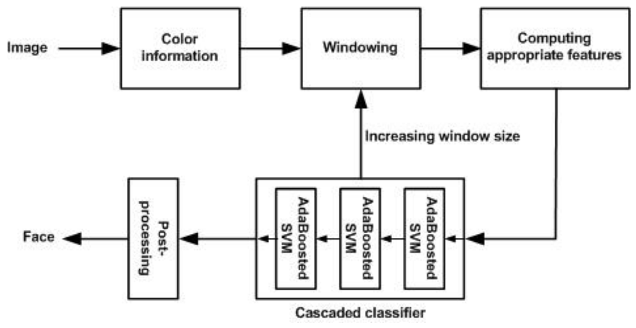

Figure 3: System architecture of AdaBoostedSVM method for face detetion.

6. EXPERIMENTAL RESULTS

To train the detector, a set of face and nonface training images from MIT CBCL data set were used. The pairwise recognition framework is evaluated on a compound face database with 2000 face images. Hand labelled faces scaled and aligned to a base resolution of 32 by 32 pixels by the center point of the two eyes. For nonface training set, an initial 10,000 nonface samples were selected randomly from 15,000 large images which contain no face.

The proposed algorithm is then tested on the MIT–CMU frontal face test set. Specifically the test set includes 92 images containing 282 faces. As our method addresses detection of frontal and real human face, those MIT–CMU test images containing large pose-angled faces, line-drawn faces, poker faces, or cartoon faces are not included in our experiments. Noting that test images used in our experiments are from diverse sources, while the training images are only from one database, the test data is able to test the generalization performance of our AdaBoostedSVM method.

Fig. 4 displays examples of multiple face detection. The reason of missed detection in Fig. 4 (a) is that our AdaBoostedSVM method is only trained by the upright frontal faces, i.e. our training set did not include any pose-angled face. Fig 4 (b) have three missed faces due to occlusion. These missed faces plus another one which is not presented here are only missed faces in our test set. These five missed faces among 92 images, containing 282 faces, lead to a high detection rate of 98.2%. Fig. 5 shows an example of detecting faces in multiple sizes, while examples of faces either very large or very small detected by our AdaboostedSVM classifier are presented in Fig. 6. Fig. 6 (a) displays the detection of the largest face in the test set. While Fig. 6 (d) shows the detection of the smallest face in the test set. Fig. 7 collects examples of detecting faces in images with poor quality.

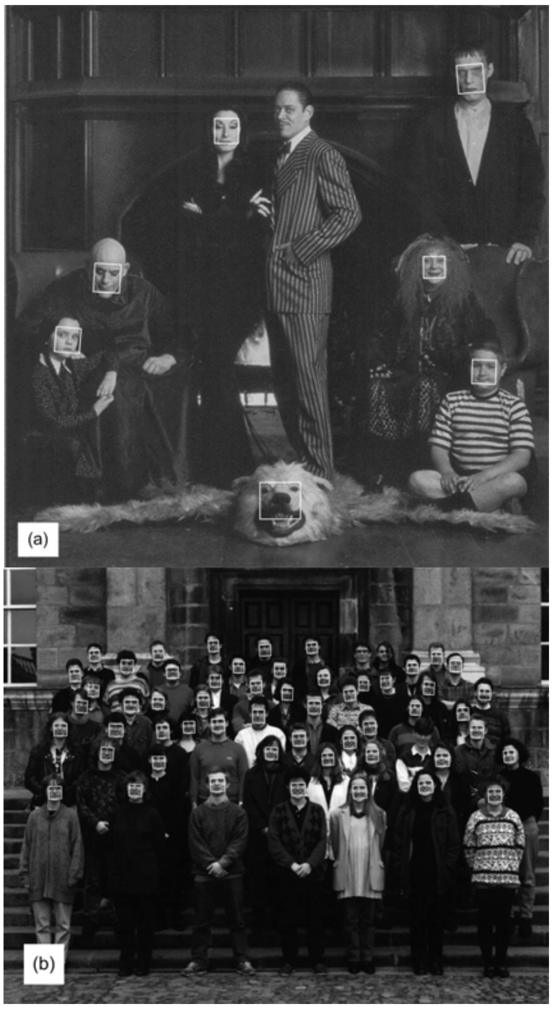

Fig 4: Example of detecting faces in images containing multiple faces.

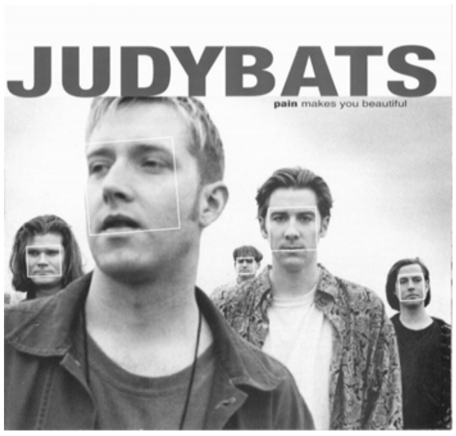

Fig 5: Multiple scale face detection.

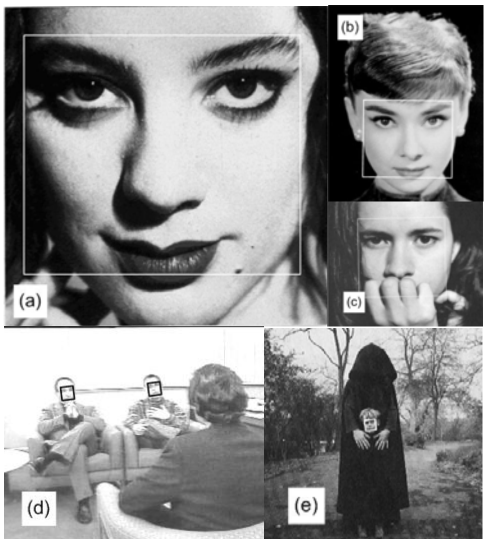

Fig 6: Examples of detecting faces that are either very large or very small.

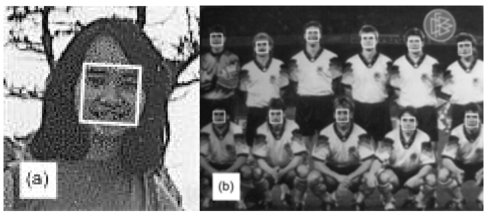

Fig 7: Example of detecting faces in images with poor quality.

In comparison to other commonly used Adaboost methods, our AdaBoosted SVM-based component classifier algorithm is superior in both detection rate and speed. Table 1 compares the proposed AdaBoostedSVM method with other proper AdaBoost methods i.e. Decision Trees, and neural networks.

TABLE 1
COMPARISON ERROR RATE (%) FOR SOME ADABOOST METHODS

Detector \ False detections	120	200
Adaboost with SVM	5.41	1.85
Adaboost with Decision Trees	9.81	2.42
Adaboost with neural networks	14.51	5.41

An ROC (Receiver Operator Characteristics) curve comparing the performance of our detector on the MIT–CMU test set is shown in Fig. 8.

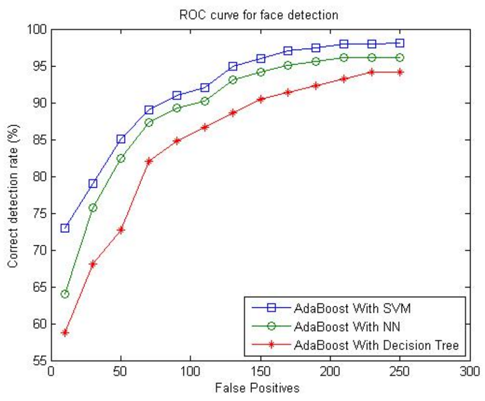

Fig 8: Comparison of ROC for frontal face detection results.

7. CONCLUSION

AdaBoost with properly designed SVM-based component classifiers is proposed in this paper, which is achieved by adaptively adjusting the kernel parameter to get a set of effective component classifiers. Experimental results on MIT–CMU database for face detection demonstrated that proposed AdaBoostedSVM algorithm performs better than other approaches of using component classifiers such as Decision Trees and neural networks both in accuracy and speed and have a high detection rate of 98.2%. It is also found that proposed AdaBoostedSVM algorithm demonstrated good performance on imbalanced classification problems. Based on the AdaBoostedSVM, an improved version is further developed to deal with the accuracy/diversity dilemma in Boosting algorithms, giving rise to better generalization performance. Experimental results indicate that the performance of the cascaded AdaBoost classifier with SVM is overall superior to those obtained by the neural network or Decision Tree.

8. REFERENCES

[1] R.-L. Hsu, M. Abdel-Mottaleb, and A.K. Jain, "Face detection in color images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 695-705, 2002.

[2] M.-H. Yang, D.J. Kriegman, and N. Ahuja, "Detecting Faces in Images: a Survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 34-58, 2002.

[3] Z. Jin, J.-Y. Yang, Z.-S. Hu, and Z. Lou, "Face recognition based on the uncorrelated discriminant transformation," *Pattern Recogn.*, vol. 34, pp. 1405-1416, 2001.

[4] A.K. Jain, R.P.W. Duin, and J. Mao, "Statistical pattern recognition: a review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 4–37, 2000.

[5] A. Pentland, "Looking at people: sensing for ubiquitous and wearable computing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp.107–119, 2000.

[6] R. Chellappa, C.L. Wilson, and S. Sirohey, "Human and machine recognition of faces: a survey, " *Proc. IEEE* vol. 83, pp.705–740, 1995.

[7] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," in *Proc.1994 Computer Vision and Pattern Recognition*, pp. 84–91.

[8] A. Yuille, P. Hallinan, and D. Cohen, "Feature extraction from faces using deformable templates," *Int'l J. Computer Vision*, vol. 8, pp. 99-111, 1992.

[9] E. Hjelmas, and B.K. Low, "Face detection: a survey," *Comput. Vision Image Underst.*, vol. 83, pp. 236–274, 2001.

[10] K.-K. Sung, and T. Poggio, "Example-based learning for View-Based human face detection," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 20, pp. 39-51, 1998.

[11] P. Shih, and C. Liu, "Face detection using discriminating feature analysis and Support Vector Machine," *Pattern Recogn.*, vol. 39, pp. 260-276, 2006.

[12] R.E. Schapire, "The boosting approach to machine learning: An overview," presented at MSRI Workshop on Nonlinear Estimation and Classification, 2002.

[13] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[14] D. Opitz, and R. Maclin, "Popular ensemble methods: an empirical study," *J. Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[15] E. Bauer, and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine Learning*, vol. 36, pp. 105–139, 1999.

[16] Y. Freund, and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Computer and System Sciences*, vol. 55, pp. 119–139, 1997.

[17] L.I. Kuncheva, and C.J. Whitaker, Using diversity with three variants of boosting: aggressive, conservative, and inverse, in *Proc. 2002 3rd International Workshop on Multiple Classifier Systems*.

[18] R.E. Schapire, and Y. Freund, "Boosting the margin: a new explanation for the effectiveness of voting methods," *Annals of Statistics*, vol. 26, pp. 1651–1686, 1998.

[19] T.G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, pp. 139-157, 2000.

[20] H. Schwenk , and Y. Bengio, "Boosting neural networks," *Neural Computation*, vol. 12, pp. 1869–1887, 2000.

[21] G. Rarsch, "Soft margins for AdaBoost," *Machine Learning*, vol. 42, pp. 287-320, 2001.

[22] P. Melville and R.J. Mooney, "Creating diversity in ensembles using artificial data," *Information Fusion*, vol. 6, pp. 99–111, 2005.

[23] L.I. Kuncheva, and C.J. Whitaker, "Measures of diversity in classifier ensambles and their relationship with the ensamble accyracy," *Machine Learning*, vol. 51, pp. 181-207, 2003.

[24] T. Windeatt, "Diversity measures for multiple classifier system analysis and design," *Information Fusion*, vol. 6, pp. 21-36, 2005.

[25] V. Vapnick, *Statistical Learning Theory*, New York: John Wiley & Sons, 1998.

[26] B. Scholkopf, K.-K. Sung, C.J.C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing Support Vector Machines with Gaussian kernels to radial basis function classifiers," *IEEE Trans. Signal Processing*, vol. 45, pp. 2758-2765, 1997.

[27] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," presented at International Conference on Computer Vision, 1998.

[28] P. Viola, and M.J. Jones, "Robust real-time face detection," *Int'l J. Computer Vision*, vol. 57, pp.137-154, 2004.

[29] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," 2003.

[30] R.E. Schapire, and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, pp. 297–336, 1999.