On the Design of Cascades of Boosted Ensembles for Face Detection

S. Charles Brubaker Jianxin Wu Jie Sun Matthew D. Mullin James M. Rehg

Abstract

Cascades of boosted ensembles have become popular in the object detection community following their highly successful introduction in the face detector of Viola and Jones [25]. Since then, researchers have sought to improve upon the original approach by incorporating new methods along a variety of axes (e.g. alternative boosting methods, feature sets, etc). We explore several axes that have not yet received adequate attention in this context: cascade learning, stronger weak hypotheses, and feature filtering. We present a novel strategy to determine the appropriate balance between false positive and detection rates in the individual stages of the cascade, enabling us to control our experiments to a degree not previously possible. We show that while the choice of boosting method has little impact on the detector's performance and feature filtering is largely ineffective, the use of stronger weak hypotheses based on CART trees can significantly improve upon the standard results.

1 Introduction

Object detection is one of the classic problems in computer vision, having applications to surveillance, robotics, multimedia processing, and HCI. Developing a generic object detection system is still an open problem, but there have been important successes over the past several years for some visual patterns. Among the most influential systems is the face detector of Viola and Jones [25], which can be credited with the widespread popularity of cascaded detectors.

The influence of the Viola-Jones face detector, however, extends beyond the use of cascades to their particular approach for learning cascades of boosted ensembles (CoBEs). The key elements of the original approach are

1. The cascade structure, which enables the detector to be fast and helps achieve a low false positive rate.
2. The use of Adaboost [5] to combine weak hypotheses into a strong one.
3. Thresholding on single feature values to form weak hypotheses (threshold-based hypotheses).
4. Feature selection from a large set of features, each of which might be only weakly discriminative in itself.

This seminal work has spawned a large body of literature that explores various aspects of the CoBE architecture. Despite this great interest, however, there is no concensus about which design decisions are most important to the final performance of the detector. Here we examine each of the key elements outlined above and evaluate the importance of the major design decisions in a systematic and controlled way.

Cascade Learning The stages of the cascade are trained sequentially, as the output of one stage affects the training examples given to the next. Deciding when to stop training one stage and move on to the next and knowing the appropriate operating point on a stage's ROC curve are critical steps in the training of a cascade. Despite the guidelines provided in [25] and [24], however, these decisions are often made by hand in practice. *Is there a fully automatic way of making these decisions that still yields state of the art results?*

We present a novel method for cascade learning, in which the user specifies detection and false positive rate goals for the whole cascade and the system adaptively chooses the detection and false positive rates for the individual stages to meet these goals. We show that the method is robust in the sense that a single set of parameters yields excellent performance over a variety of detection strategies and that it is capable of producing state of the art results. Because the method is fully automatic, it enables us to control the cascade learning and rigorously compare strategies for the other learning tasks in the CoBE architecture.

Boosting Several previous papers attribute improved detection results to their use of different boosting methods. Unfortunately, these changes are often coupled with other changes to the detector, making a fair comparison difficult. *Is the particular boosting method an important factor in the performance of a CoBE face detector?*

Of the boosting algorithms we implemented, none consistently outperformed the others in our experiments. How-

ever, we did observe that depending on the type of weak learner, the choice of boosting algorithm can affect the number of boosting iterations required to meet the goal criteria for a stage. For instance, when CART-based [1] hypotheses are used, Realboost [19] requires fewer iterations than several other boosting methods. For threshold-based hypotheses, however, Realboost actually requires many more iterations than the other methods.

Weak Hypotheses The original Viola and Jones strategy for forming weak hypotheses is to threshold on single features. The advantage of this approach is that the hypotheses are fast to train and evaluate; however, every individual hypothesis has high error. *What is the effect of using more powerful methods to obtain more discriminative weak hypotheses? Are fewer total features required? Are better results achieved?*

Combining several features into a single weak hypothesis based on CART classifiers consistently and significantly improved the detection performance in our experiments. As the learning task becomes more difficult in the later stages, ensembles of threshold-based hypotheses are unable to achieve the needed performance, whereas the ensembles of the CART-based hypotheses are. Our experiments also show that using histograms over single features gives good results for the early stages with fewer iterations of Adaboost, but does not produce better results in the later stages. Thus, the detector runs faster, but the classification peformance is not significantly better than when threshold-based hypotheses are used.

Feature Selection While several authors have already experimented with using more discriminative feature sets within the CoBE framework (e.g. [12, 7]), we focus on reducing the size of the feature set with filtering techniques before the boosting framework is applied. As we discuss in section 6, feature filtering can have a dramatic effect on the training time of the ensemble, and is useful in prototyping detectors. Despite the popularity of feature filtering in other fields and the attention it has received in the machine learning literature [9], however, we are unaware of any published attempts to apply feature filtering in the CoBE architecture. *Is there an effective way to filter the features so that the weak learner has fewer to examine? Can other algorithms improve upon the greedy feature selection strategy of Adaboost?*

For the feature set from the original Viola and Jones paper [25], we found that several standard filtering methods are largely ineffective. Typically, Adaboost itself plays a feature selection role, using only a small subset of the available features to form the weak hypotheses. Filtering methods that are significantly faster than Adaboost seem to select a redundant feature set, where the individual features all tend to misclassify the same examples, and offer no advantage over random selection. Methods that have running times closer to that of Adaboost perform better than random selection, but offer no advantage over the Adaboost's inherent greedy feature selection strategy. Thus, randomly filtering out as few features as the computing resources permit seems to be as good a strategy as any other.

In summary, the major contributions of this paper are

- an improved cascade learning algorithm that views the actual detection performance as a random variable and enables us to perform controlled experiments on other aspects of the detector; and
- a controlled empirical analysis showing that several popular approaches to feature filtering are largely ineffective in the CoBE context,
- the choice of boosting algorithm has more impact on the detector's speed than it does on classification performance, and
- the use of stronger weak hypotheses can significantly improve the detector's results.

2 Architecture Overview

Here we present a generic framework that encompasses much of the variation in the literature of CoBEs. At the highest level, a CoBE detector is a cascade of increasingly specialized stages, each one being trained to reject the false positives of previous stages, while detecting all positive instances. When an instance enters the detector, it is examined by the first stage, which either rejects the instance immediately, or passes it on to the next stage for further scrutiny. This process is repeated for subsequent stages until the instance is either rejected or it passes all stages, thereby being detected. An individual stage consists of an ensemble of weak classifiers, whose outputs are combined by a weighted vote. Each weak classifier is based on a small subset of the image features, which can be any function computed over a sub-window of the image.

The cascade architecture was designed to handle the rare event nature of the face detection problem. In practice, detectors are scanned across images in a brute force fashion. To reliably detect faces without littering the image with false positives therefore requires a false positive rate on the

order of 10^{-7}. Because of the conjunctive nature of the cascade, however, not all negative instances need be rejected at once by a single stage. Instead, each stage of the cascade is left with the potentially easier task of carving away part of the negative class, leaving fewer and fewer false positives. Its conjunctive nature also makes the detector fast, because most instances are non-faces and will be rejected by the first few stages of the cascade before the detector has invested much time on them.

To express these ideas mathematically, let X be the instance space, and let $c : X \rightarrow \{0, 1\}$ be the target concept. The stages of the cascade $\{s_i\}_{i=1}^{N}$ consist of a set of weak classifiers $\{h_{ij}\}_{j=1}^{M_i}$ and a threshold θ_i such that for all $x \in X$

$$s_i(x) = \chi[\sum_{j=1}^{M_i} h_{ij}(x) > \theta_i], \quad (1)$$

where χ is the indicator function. Here we follow the convention of using confidence rated classifiers that return an unbounded real value, instead of $\{0, 1\}$ or $\{-1, 1\}$, removing the need for a weighing coefficient [1]. The hypotheses for the entire cascade can then be expressed as $\bigwedge_{i=1}^{N} s_i(x)$. It is important to note, however, that while a stage concept s_i is defined over all X, it will only ever be applied to the set $X_i = \{x \in X : \wedge_{k=1}^{i-1} s_k(x)\}$. Thus, a stage that performs well on such a subset might perform poorly on all of X and still fulfill its role in the detector.

2.1 Training a CoBE

A generic version of the original Viola-Jones training algorithm is presented in the LEARN-COBE procedure. The subroutines serve as placeholders for any number of solutions to the subproblem in question. Before training a stage, we first apply the standard bootstrapping practice [22] to acquire appropriate training and validation data. Positive examples that would be rejected by the current cascade are removed from the training and validation sets, and false positives of the current cascade are extracted from an image corpus, in which the object is either absent or blacked out, to form the negative examples (BOOTSTRAP). Thus, these data sets consist only of instances not rejected by any stage of the cascade that has already been trained. Since the set of features available is often too large (134,736 for the original Viola-Jones set), we then filter the set down to a manageable size (FILTER-FEATURES). To build the ensemble of classifiers, we first learn a classifier (WEAK-LEARN) based on the current set of weights. We then reweigh the examples (REWEIGH-EXAMPLES), giving the misclassified examples more weight. Finally, we search for a suitable θ_i to balance the detection versus false positive tradeoff (FIND-BEST-THRESHOLD [2]). To assess the peformance, we apply the stage to the validation set to calculate the false positive and detection rate pair $\langle \hat{f}_i, \hat{d}_i \rangle$ (VALIDATE). This measurement and the analogous measurements for all previous stages are used to predict an overall cost for the cascade that is to be minimized (PREDICT-COST). If this cost is low enough, then we are done with the current stage and proceed to the next one; otherwise, we repeat the cycle of learning a new hypothesis, reweighing the examples, and evaluating the ensemble.

Let F be the set of features and E the set of examples. We denote the weights for E as W. No more than L iterations of Adaboost are permitted. G refers to the goal cost for the cascade, and $\langle \hat{f}_i, \hat{d}_i \rangle$ denotes the false positive and detection rate pair for the ith stage.

procedure LEARN-COBE()
  $C \leftarrow \emptyset$ { C initialize an empty cascade}
  for each stage i **do**
    $E \leftarrow$ BOOTSTRAP()
    $F' \leftarrow$ FILTER-FEATURES()
    $s_i \leftarrow \emptyset$
    $W \leftarrow$ INITIALIZE-WEIGHTS()
    repeat
      $h \leftarrow$ WEAK-LEARN()
      $W \leftarrow$ REWEIGH-EXAMPLES()
      $s_i \leftarrow s_i \cup h$
      $\theta_i \leftarrow$ FIND-BEST-THRESHOLD()
      $\langle \hat{f}_i, \hat{d}_i \rangle \leftarrow$ VALIDATE()
    until $|s_i| > L$ or PREDICT-COST() $\leq G$
    $C \leftarrow C \cup \langle s_i, \theta_i \rangle$
  end for

It should be noted that in choosing the stage thresholds θ_i, the goal should not be to maximize the performance of the stage in isolation, but rather to maximize the performance of the cascade as a whole. We will shown how to address this problem in section 4.

Next, we turn to the WEAK-LEARN routine, which outlines the standard procedure for training weak classifiers. The form of the classifier defines partitions P of the instance space considered by the algorithm. For instance, if the classifier thresholds a single feature to make its decision, then the set P could contain all dichotomies produced by all possible thresholds of all features in F'. For a given partition p we denote the corresponding disjoint sets of the instance space as $\{X_j^p\}$, where j indexes over the sections of the partition. We denote the weight of the positive and

negatives examples in these sets as $\{W_j^+\}$ and $\{W_j^-\}$. The WEAK-LEARN routine builds a hypothesis for each partition p by deciding on a confidence value for each subset of the instance space X_j^p based on the weights W_j^+ and W_j^-. It then evaluates each hypothesis and returns the best hypothesis for use in the ensemble.

Let P be the set of partions of the examples induced by the classifier's form; e.g. thresholding on a single feature.

procedure WEAK-LEARN()
  $\epsilon \leftarrow \infty$
  for all $p \in P$ **do**
    $h_p \leftarrow$ CONFIDENCE(p)
    if $\epsilon >$ ERROR(h_p) **then**
      $h_{\text{best}} \leftarrow h_p$
      $\epsilon \leftarrow$ ERROR(h_p)
    end if
  end for
  return h_{best}

Although not all changes made to the original Viola and Jones implementation strictly fit into the above architecture, we believe it provides a useful abstraction of the CoBE approach.

3 Previous Work

Despite the critical importance of the FIND-BEST-THRESHOLD and PREDICT-COST functions to the performance of the final detector, these aspects of the training process have received comparatively little attention in the literature. A preliminary version of our approach was published in [21]. In comparison, our new algorithm views the actual cascade performance as a random variable and uses a smaller and more intuitive set of parameters (see sections 4.3 and 4.4 for details).

Huitao Luo has recently published a method for adjusting the stage thresholds after the full cascade has been trained [16]. While the success of this method illustrates the importance of the stage thresholds for classification performance, it does not address how the thresholds should be chosen in the cascade training phase (FIND-BEST-THRESHOLD), which critically influences the bootstrapped data, or when it is appropriate to begin training a new stage (PREDICT-COST).

Despite the great interest in cascaded detectors, there have been few controlled, comparative studies that address questions about which factors have the greatest impact on cascade performance. In this regard, the study of Lienhart et al [14] is closest to the analysis presented in this paper. Our work differs both in its methodology and in its conclusions. The stopping criterion for a stage proposed in [14] is based on a fixed performance goal on the training data. In contrast, we use a probablistic framework based on validation data, as discussed section 4.4. Another important difference lies in the evaluation methodology. The ROC curves in [14] are produced by varying a post-processing parameter that regulates the density of detected windows required for a location to be labeled as a face. We produce ROC curves by varying the thresholds of the last several stage classifiers as described in section 4.1. Our method allows us to account for the effects of post-processing as well (see section 7).

Our major results also differ from [14] in several important respects. Lienhart et al found that Gentleboost outperforms Realboost and Discrete Adaboost. We found that these three boosting algorithms did not produce significantly different results but, depending on the weak learner, they could produce detectors of varying speed. Our results also show a much more striking difference between threshold-based and CART-based hypotheses. Indeed, in all of our experiments, the use of CART weak classifiers produced the most consistent and significant improvements.

Much of the early research on the CoBE architecture focused on the boosting algorithm. In their 2002 paper, Viola and Jones observe that the goal of a stage in the cascade is not to minimize error, but to retain very high detection rates, while accepting modest false positive rates if necessary [24]. They propose Asymmetric Adaboost, which changes the REWEIGH-EXAMPLES routine to keep most of the weight on the positive examples (instead of treating positive and negative examples equally), ensuring that a high percentage is detected by each weak classifier.

In a more dramatic change, [28] uses forward feature selection (FFS), which does not re-weigh the data between iterations, but instead greedily chooses the hypothesis that most improves the performance of the entire ensemble. [27] introduces the Linear Asymmetric Classifier (LAC) algorithm, which is used to re-weigh hypotheses after they have been selected by FFS or Adaboost. Like Asymmetric Adaboost, this method is designed for the requirements of classification in the cascade context.

Li and Zhang have proposed another alternative boosting algorithm in their paper on FloatBoost [13], which instead of greedily adding hypotheses to the ensemble, allows backtracking to eliminate the less useful or even hurtful hypotheses. In other respects, the algorithm proceeds as RealBoost.

Liu and Shum [15] found that using KL-boost combined with weak classifiers based on histograms of 1D projections

in feature space improved detection performance over the original approach. However, it is not clear whether it is the changes to the weighing scheme or the means of forming the weak hypotheses that is critical to the improvement.

A more radical departure from the LEARN-COBE routine is due to Xiao et al [29]. Inspired by the observation that the operating point of a stage may not minimize error, they allow the hypothesis formed by the minimum error threshold of the previous stage to play the role of a weak hypothesis in the next stage of the cascade. Having thus produced a cascaded detector, they convert it to a single weighted voting scheme and train an SVM to relearn the confidence (vote) weights.

Others have changed the feature set while keeping the other key aspects of the CoBE architecture. Lienhart et al [14] proposed another Haar-like feature set including diagonal features that can also be quickly computed via an integral image. Froba and Ernst [7] use a modified census transform and achieve state of the art performance using only three cascade stages. Levi and Weiss [12] also achieve state of the art performance using a small number of training examples with features based on edge oriented histograms.

CoVEs A class of detectors closely related to the CoBE family are the cascades of voting ensembles. By a voting ensemble, we mean a classifier of the form [3]

$$\sum_i h_i(x) > \theta, \tag{2}$$

where x is an instance and h_i returns an unbounded real. Included in this class are the cascade of semi-Naive Bayes classifiers used by Schneiderman [20], the cascades of SVMs used by Heisele et al [10], and the linear classifiers of Keren et al [11], Elad et al [2], and Romdhani et al [17]. The critical difference between these detectors and the CoBEs is that they do not use boosting. Nevertheless, the cascade learning algorithm and some of the empirical results may have implications for these architectures as well.

For a more detailed history of face detection see [30].

4 Cascade Learning

Two of the most important decisions in building a cascade of boosted ensembles are:

1. When to stop training a stage to move on to the next one.
2. How to balance the detection versus false positive trade-off within a stage.

In terms of our LEARN-COBE algorithm these decisions are determined by the function FIND-BEST-THRESHOLD, which chooses θ_i, fixing the stage's operating point, and by the function PREDICT-COST which determines when to move on to the next stage of the cascade.

4.1 The Cascade ROC

Before addressing how to make these decisions, we review the effect of the individual stage operating points on the overall cascade performance and explain how to build a ROC curve for the entire cascade. We believe that this cascade ROC curve can be a useful tool for visualizing the cascade training process and that it is a novel contribution of this paper.

If the true false positive and detection rates for the stages are $\{f_i\}$ and $\{d_i\}$, then the false positive and detection rates for the whole cascade are $F = \prod_{i=1}^{N} f_i$ and $D = \prod_{i=1}^{N} d_i$. This is not a statement of independence, but a factorization of the probability that all stages accept an instance; i.e.,

$$\Pr[s_1(x), \ldots, s_N(x) | c(x) = y] = \prod_{i=1}^{N} \Pr[s_i(x) | s_{i-1}(x), \ldots, s_1, c(x) = y], \tag{3}$$

where y is either -1 or 1 (face or non-face). Knowing the relationship between $\{f_i\}$ and $\{d_i\}$ will enable us to reason about F and D and thus about the overall performance of the cascade.

For a single ensemble classifier, the ROC curve is formed by adjusting the θ_i parameter and counting the number of detections and false positives. To build a ROC curve for a cascade, we need a way in which the detector scores the instances so that we can adjust an analogous threshold for the entire cascade. We accomplish this by assigning higher scores to instances that are accepted by more stages in the cascade. For instances rejected by the same stage s_i, ties are broken according to the score assigned by the stage $s_i(x)$. For accepted instances, ties are broken according the score in the last stage. With this definition of a cascade scoring, we can plot not just the ROC curve for the full cascade, but the ROC curve for partial cascades $C_n = \cup_{i=1}^{n} \langle s_i, \theta_i \rangle$, where n is less than the total number of stages N. Such a plot is shown for several partial cascades in figure 1.

Notice how the curves are identical in the upper right corner, where the instances are being rejected by a shared (early) part of the cascade. The points where the curves diverge correspond to the θ_i thresholds of the stages, as the new stage is applied in one partial cascade where it is not in the other.

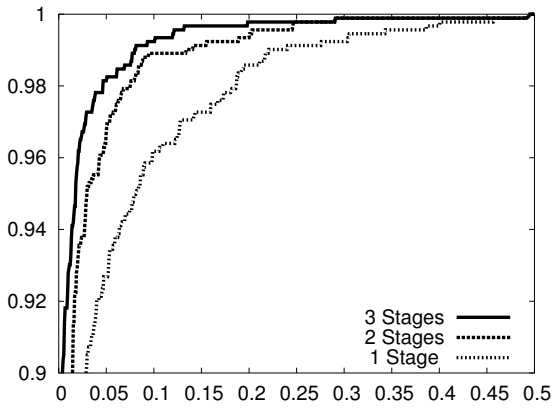


Figure 1: ROC Curve for several partial cascades. The curves diverge at points corresponding to the stage thresholds.

Another way to think about a cascade ROC curve is that it shows scaled versions of the individual stage ROC curves that are pieced together. Tracing the three stage cascade from the right (high false positive) end of figure 1, we start with the ROC curve for the first stage. After the threshold for the first stage is crossed, then the ROC curve for the second stage is shown, but it is scaled so that the operating point $\langle 1, 1\rangle$ corresponds to the point $\langle f_1, d_1\rangle$. For all subsequent stages, $n+1$, the stage ROC curve is scaled to $\langle \prod_{i=1}^{n} f_i, \prod_{i=1}^{n} d_i\rangle$. The choice of θ_i determines where the cascade ROC curve stops following one stage's ROC curve and starts following the next.

Thus, we can think about the LEARN-COBE algorithm as building this ROC curve in this stagewise manner. The addition of a weak hypothesis to the ensemble changes the shape of the curve. The FIND-BEST-THRESHOLD procedure then selects a point on the curve to serve as the starting point for the remaining stages and PREDICT-COST evaluates whether this point is good enough. If it is not, the curve is reshaped by adding another weak hypothesis, and the process is repeated. The challenge here is that the shape of the next stage's ROC curve is not known until the choice of θ_i is made, the bootstrapping is performed, and new weak hypotheses are trained.

4.2 Fixed Stage Goal

The standard approach outlined in [25, 24] is to choose a goal operating point $\langle F_g, D_g\rangle$ and then take its Lth root to obtain $\langle f_g, d_g\rangle$, where L is the intended number of stages in the cascade. Each stage is constrained to achieve one of f_g or d_g (typically f_g works better) and then terminates when either the other goal criterion is achieved or the maximum number of boosting iterations is exceeded.

This goal-based strategy leaves something to be desired, however. First, it rigidly fixes the number of stages in the cascade before any training is done. Second, it does not permit any trade-off between the detection and false positive rates within the stages. For instance, when selecting the threshold of a stage, one might be able to significantly improve the false positive rate at a small expense to the detection rate, improving the chances of meeting the goal criteria. The extra leeway on the false positive criterion might also be used at a later stage to improve a stage's detection at the expense of the false positive rate. By fixing one element of the operating point, this strategy precludes taking advantage of such trade-offs.

The most natural way to trade-off the detection and false positive rates of a stage is to use a Bayes risk $\eta f_i + (1 - d_i)$. This approach, however, offers little control over how the overall cascade performs. Depending on the choice of η and the shapes of the stages' ROC curves, the stage operating points tend to gravitate towards one extreme or the other, so that when the whole cascade is trained, one is left with good detection and a poor false positive rate or a good false positive rate and poor detection rate. Controlling the η parameter to obtain a goal operating point for the entire cascade is not only non-intuitive, but it is also possible that no single choice will give acceptable results on all stages, as argued in [21].

4.3 Cascade Indifference Curve

In [21], we introduced the notion of the indifference curve and presented a novel method for cascade learning that retains the benefits of both these approaches, while addressing many of their weaknesses. Although there are well established methods for choosing the appropriate trade-off between detection and false positive rates for single-stage classifiers, it is far from clear what it means for one operating point to be better than another in the cascade context. Our solution is to assume that the operating point for the current stage can be replicated in any number of subsequent stages. To compare two operating points, we simply assume as many repetitions as are necessary to bring the false positive rate to a fixed value and compare the predicted detection rates.

For instance, suppose that we are comparing the operating points $a = \langle 0.7, 0.99\rangle$ and $b = \langle 0.49, 0.97\rangle$. We assume that a can be repeated in the next stage, making it equivalent to $a^2 = \langle 0.49, 0.98\rangle$. We call this the *repeatability assump-*

tion. The operating point a^2 has the same false positive rate as b but a higher detection rate. We conclude, therefore, that a is a better operating point than b.[4] It is this type of reasoning that underlies the FIND-BEST-THRESHOLD decision.

To decide when to move on to the next stage (PREDICT-COST), we use a cost function that is a linear combination of the predicted detection rate $\mathcal{C}_b$ at some small false positive rate and the predicted number of features applied per window $\mathcal{C}_c$

$$\text{cost}_{\text{old}} = \mathcal{C}_b + \lambda \mathcal{C}_c.$$

One difficulty with this approach is that it choses operating points with high false positive rates when they are coupled with perfect or extremely high detection rates. Achieving the desired false positive rate of approximately 10^{-7} with such operating points requires an impractical number of stages. To cope with this problem, the method requires that the user specify a range of acceptable false positive rates for the stages $[f_{\min}, f_{\max}]$.

The result is that the user must specify a set of rather unintuitive parameters. The quantities $f_{\min}$ and $f_{\max}$ may make sense to someone understands the inner workings of the algorithm, but they are not directly related to the quantites he cares about most (e.g. how well does it perform? how fast does it run?). Even more difficult is the λ parameter, which balances speed and the predicted detection rate. If λ is too large, then the detector will not perform well. If too small, then the detector will be unnecessarily slow. Finding a value that gives the desired result can require extensive experimentation.

4.4 Cascade Learning with Beta Variables

We now describe a novel alternative method for cascade learning which is governed by a much more intuitive set of parameters. The inputs to the learning method are:

1. A goal operating point for the entire cascade $\langle F_g, D_g \rangle$.
2. A ratio η that reflects the relative importance of the false positive and detection criteria.
3. A maximum number of stages L.

The cascade learner then builds the fastest detector it can while achieving the goal performance with high probability.

A key element of our approach is that the algorithm views the performance of the cascade $\langle F, D \rangle$ as a random variable and treats the empirical results on validation data for the individual stages, $\{\hat{f}_i\}$ and $\{\hat{d}_i\}$, as evidence. During training, we use the repeatability assumption to infer the probability that the cascade will meet the goal criteria. Each stage is then trained to use the minimum number of features that ensure that this probability of success is sufficiently high.

4.4.1 Cost Function

Because a reasonable goal might not be known a priori, the algorithm must be robust to unattainable goals and produce results that are as close as possible. Depending on the attainability of the goal, therefore, we adjust our cost function. For simplicity, assume that $\eta > 1.0$, meaning that the false positive criterion is more important. We consider the following cases

1. If $\Pr[D < D_g] < \gamma$ and $\Pr[F > F_g] < \gamma$, then
$$\text{cost} = \Pr[D < D_g] + \eta \Pr[F > F_g].$$
2. If $\Pr[F > F_g] < P_t$, then cost $= 2 + \eta - D$.
3. Otherwise, cost $= 2 + \eta + F$.

The first cost function is suitable when both goals are attainable with some substantial probability, say γ. However, when this is not possible, then the function provides no incentive to trade a small decrease in the false positive rate for a large improvement in the detection rate (an analogous statement holds if $\eta < 1.0$, giving detection greater importance). Therefore, if both criteria cannot be met with probability γ, then we constrain the false positive rate to be met with probability γ and maximize the detection rate. Finally, if the criterion for false positive rate cannot be met with probability γ, we simply minimize the false positive rate. Typically, this means that the false positive rate is reduced to zero, effectively terminating the training process.

4.4.2 Cost Prediction

Minimizing this cost function requires the ability to compute $\Pr[D > D_g]$ and $\Pr[F < F_g]$. We will only treat the detection criterion, because the false positive one is analogous. Consider the likelihood $\Pr[\hat{d}_i | d_i]$, where $\hat{d}_i$ is the measured detection rate over M positive examples. Given the true detection rate d_i, the probability of m out of M examples being detected is just the binomial distribution

$$\left(\begin{array}{c} M \\ m \end{array} \right) (1 - d_i)^{M-m} d_i^m.$$

Taking a uniform prior $\Pr[d_i]$ over $[0, 1]$ and applying Bayes

Assume that the cascade has already been trained through stage i and that we are predicting the cost if the measured operating point of the next stage is $\langle \hat{f}_{i+1}, \hat{d}_{i+1} \rangle$.

PREDICT-COST-SAMPLE *maintains a set of sampled operating points for the currently trained cascade $\{\langle F_i^k, D_i^k \rangle\}_{k=1}^K$. All measurements are made with validation sets of M negative examples and the same number of positive examples.*

procedure PREDICT-COST-SAMPLE()
  for $j = i+1$ to N **do**
    for $k = 1$ to K **do**
      $F_j^k \leftarrow F_{j-1}^k \cdot \beta_{\hat{f}_i}$, where $\beta_{\hat{f}_i}$ is a random beta deviate with parameters $\hat{f}_i M + 1$ and $(1 - \hat{f}_i)M + 1$.
      $D_j^k \leftarrow D_{j-1}^k \cdot \beta_{\hat{d}_i}$, where $\beta_{\hat{d}_i}$ is a random beta deviate with parameters $\hat{d}_i M + 1$ and $(1 - \hat{d}_i)M + 1$.
    end for
    $G_f \leftarrow |\{k : F_j^k > F_g\}|/M$
    $G_d \leftarrow |\{k : D_j^k < D_g\}|/M$
    $\text{cost}_j \leftarrow \text{COST}(G_f, G_d)$.
  end for
  return $\min_j \text{cost}_j$.

PREDICT-COST-APPROX *maintains the first and second moments of the distribution for of cascade operating points $\langle S_i^f, S_i^d \rangle$ and $\langle T_i^k, T_i^d \rangle$. $I(z; a, b)$ refers to the regularized beta function.*

procedure PREDICT-COST-APPROX()
  for $j = i+1$ to N **do**
    $\langle S_j^f, S_j^d \rangle \leftarrow \langle S_{j-1}^f \frac{\hat{f}_i M+1}{M+2}, S_{j-1}^d \frac{\hat{d}_i M+1}{M+2} \rangle$
    $\langle T_j^f, T_j^d \rangle \leftarrow \langle T_{j-1}^f \frac{(\hat{f}_i M+1)(\hat{f}_i M+2)}{(M+2)(M+3)}, T_{j-1}^d \frac{(\hat{d}_i M+1)(\hat{d}_i M+2)}{(M+2)(M+3)} \rangle$
    $G_f \leftarrow I\left(F_g; \frac{(S_j^f - T_j^f)S_j^f}{T_j - S_j^f}, \frac{(S_j^f - T_j^f)(1-S_j^f)}{T_j - S_j^f}\right)$,
    $G_d \leftarrow 1 - I\left(D_g; \frac{(S_j^d - T_j^d)S_j^d}{T_j - S_j^d}, \frac{(S_j^d - T_j^d)(1-S_j^d)}{T_j - S_j^d}\right)$,
    $\text{cost}_j \leftarrow \text{COST}(G_f, G_d)$.
  end for
  return $\min_j \text{cost}_j$.

Either of the procedures above can be used for PREDICT-COST.

procedure FIND-BEST-THRESHOLD()
  choose θ_i for which PREDICT-COST() is lowest.

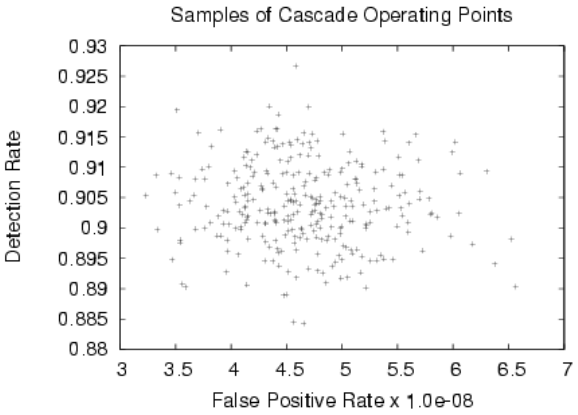


Figure 2: Samples generated by PREDICT-COST-SAMPLE of the operating point for a fully trained cascade.

rule gives

$$\begin{aligned} \Pr[d_i|m, M] &= \frac{\Pr[m|d_i, M]\Pr[d_i]}{\int_0^1 \Pr[m|p, M]\Pr[p]dp} \\ &= \frac{(1-d_i)^{M-m} d_i^m}{\int_0^1 (1-p)^{M-m} p^m dp}, \end{aligned}$$

which is precisely the beta distribution with parameters $m+1$ and $M - m + 1$.

Therefore, conditioned on the validation measurements, D is the product of beta variables. The exact distribution only admits a clean analytic form in a few specialized cases [8], but it can easily be approximated. One strategy is to sample from the distribution for D by taking a sample from the distribution d_i for each stage and taking their product. The quantity $\Pr[D > D_g]$ can be estimated by counting the fraction of samples greater than D_g. This method is used in the PREDICT-COST-SAMPLE procedure. A final set of samples for a fully trained cascade is shown in figure 2.

Another approximation strategy due to Fan [3] is to compute the first and second moments of distribution of D and then to approximate it as a beta distribution having this first and second moments. The first and second moments of the product of beta variables are given by

$$S = \prod_{i=1}^N \frac{a_i}{a_i + b_i} \quad \text{and} \quad T = \prod_{i=1}^N \frac{a_i(a_i+1)}{(a_i+b_i)(a_i+b_i+1)},$$

respectively. The parameters for the beta distribution with these statistics are

$$a = \frac{(S-T)S}{T - S^2} \quad \text{and} \quad b = \frac{(S-T)(1-S)}{T - S^2}.$$

This method is used in the PREDICT-COST-APPROX procedure.

Either of these procedures allows us to estimate the cost once the cascade is fully trained, but it is still not clear how to train the cascade. We therefore adapt the repeatability assumption and apply it to the validation data, meaning that if we can achieve $\langle \hat{f}_i, \hat{d}_i \rangle$ on a validation set for the current stage, then we assume that we can achieve the same result for all subseqent stages. Therefore, as we are training the ith stage, we use the results on the validation set to estimate $\langle \hat{f}_j, \hat{d}_j \rangle$ for all previous stages ($j < i$), but we use the results for the ith stage on validation data for any subsequent stages ($j > i$). The operating point having the lowest cost according to this estimate is chosen for each stage, as shown in the FIND-BEST-THRESHOLD procedure.

The advantage of this approach is that it offers the control of the goal-based strategy over the cascade's overall performance, while allowing some subtle tradeoffs between detection and false positive rates in the stages. Moreover, it can "remember" past trade-offs to help decide whether a new trade-off will improve the chances of achieving the cascade's goal operating point. Note that though we specify a maximum number of stages, we do not specify a minimum. If the learner predicts better performance with fewer stages, then it will plan for fewer stages.

5 Boosting Methods

Although many modifications to Adaboost have been proposed and applied to face detection, most of the methods can be implemented with the WEAKLEARN routine of section 2.1 simply by varying the CONFIDENCE and ERROR functions. Recall that the strong hypothesis is formed by taking a weighted vote of weak hypotheses, so that stage decisions are made according to equation 1.

Each hypothesis h_{ij} is a classifier that encodes a partition of the input space and returns the confidence for the section of the partition in which the instance x falls. During training phase, the partition of the instance space is selected according to which induces the lowest error as measured by the ERROR function. The confidence for each section of the partition is determined by the CONFIDENCE function.

Discrete Adaboost, Realboost, and Gentleboost (the methods implented in this study) are summarized in terms of these functions in figure 3. Consider a section j of the partition. Let W_j^+ be the total weight of positives examples in the section and let W_j^- be defined analogously. The ERROR measure and the CONFIDENCE for section j are given in the table. The graphs show the ERROR summand and CONFIDENCE for section j as a function of the fraction of the positive weight in the partition $W_j^+/(W_j^+ + W_j^-)$. We will not review the various assumptions behind these formulae or review their derivation, but refer the reader to [19, 6].

It is interesting to observe how the methods treat pure (high or low $W_j^+/(W_j^+ + W_j^-)$) sections. Realboost seems to encourage these pure partitions by retaining a higher error for less pure partitions. It also places great confidence in these high partitions by giving pure partitions much more weight than less pure ones. This contrasts with Discrete Adaboost, which gives all partitions equation weight up to a sign.

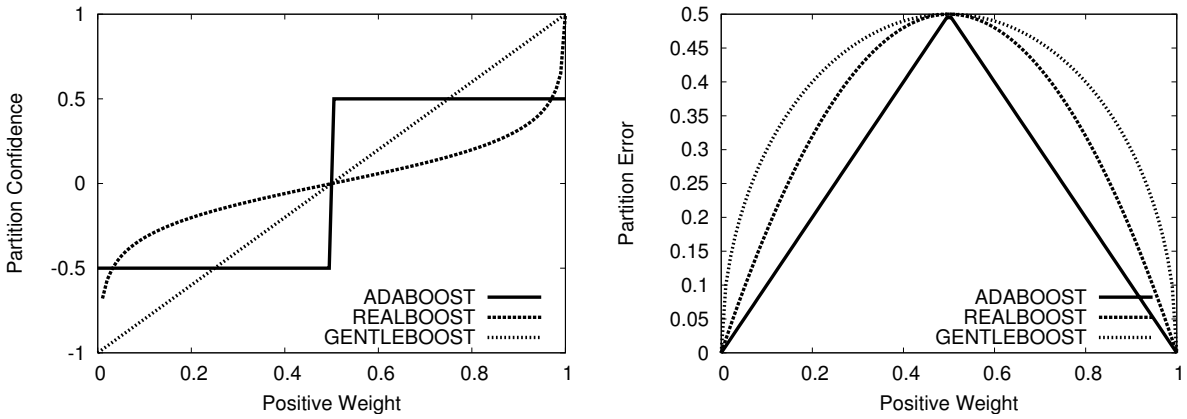
6 Weak Learning

Although thresholding on a single feature has been the dominant practice in CoBEs for object detection, Adaboost does not restrict how the weak learning takes place. The thresholding strategy may be efficient in terms of training or execution time, but it seems doubtful that such a simple weak learner would give the best results.

The act of thresholding converts a real-valued feature to binary, throwing away information. Even if the optimal decision based on a single feature could be described by a threshold (as opposed to something more complicated such as a union of intervals), the real-valued feature still might contain valuable information about the confidence associated with such a decision. With this in mind, we try to preserve some of this information by discretizing the feature values and using histogram learning to build hypotheses. The disadvantage of this approach is that because of the coarse discretization some ability to make a fine distinction between feature values is lost.

Another weakness of thresholding on single features can be seen by the following XOR-like example. Let x and y be two features and let the target distribution be uniform over $[-1, 1] \times [-1, 1]$, with the only positive class in $([-1, 0] \times [-1, 0]) \cup ([0, 1] \times [0, 1])$ and the only negative class elsewhere. Regardless of the number of iterations, Adaboost cannot learn the concept if the weak hypotheses are formed by examining only x or y, nor can any voting scheme for that matter.

With this example in mind, a natural way to make weak hypotheses stronger is to combine several features. In our experiments, we learn CART trees by greedily adding the feature-threshold pair that results in the greatest reduction in the ERROR measure, filling out the binary tree to a fixed depth. With these more powerful hypotheses, Adaboost can solve the trivial XOR-like problem given above. As was the



Method	CONFIDENCE	ERROR
Adaboost	$\frac{\text{sign}(W_j^+ - W_j^-)}{2} \cdot \log \frac{1-\epsilon}{\epsilon}$	$\sum_j \min\{W_j^+, W_j^-\}$
Gentleboost	$\frac{W_j^+ - W_j^-}{W_j^+ + W_j^-}$	$\sum_j 2 \cdot \frac{W_j^+ W_j^-}{W_j^+ + W_j^-}$
Realboost	$\frac{1}{2} \log \frac{W_j^+}{W_j^-}$	$\sum_j \sqrt{W_j^+ W_j^-}$

Figure 3: Confidence and Error Measures used by several boosting algorithms. In the table, W_j^+ and W_j^- refer to the total positive and negative weight in the part of the section j in question. In the graphs, positive weight is defined as $W_j^+/(W_j^+ + W_j^-)$.

case for histograms, the use of CART trees also allows the hypothesis to return a variety of confidences.

sectionFeature Selection One of the greatest obstacles to wider use of cascades of boosted ensembles is that they take a long time to train. Chief among the reasons for this slowness is that in every round of boosting the WEAK-LEARN routine examines every example for every feature. Since reducing the example corpus weakens the generalization, reducing the feature pool via the FILTER-FEATURES routine is an attractive option.

To actually improve the training time, however, the filtering algorithm itself must be faster than Adaboost. Unfortunately, few filtering algorithms offer an asymptotic improvement in training time. Nevertheless, asymptotically equivalent methods often admit implementation speed-ups, which make the actual run-time faster than the worst-case analysis time would indicate. Moreover, because Adaboost's greedy selection of features is not optimal, limiting the feature pool available to Adaboost may actually improve the results. The idea is that Adaboost may produce a better classifier when it is presented with a small set of features, all of which are good, rather than a large set containing these same good features in addition to many spurious ones.

For purposes of this discussion, therefore, we divide filtering techniques into two broad categories:

Fast Filters: This category consists primarily of ranking schemes which examine each feature once and sort according to some measure of the feature's discriminative power. These filters are typically much faster than Adaboost and run in $O(|F| \log |F|)$ time.

Slow Filters: This category includes methods that examine each feature in F before choosing the next feature to add to the selected pool F'. These filters run in $O(|F'||F||E|)$ time and are about as fast as Adaboost with a thresholding weak learner.

Notice that the running times given above assume that the precomputation strategy of [27] has been used. With this strategy, the evaluation of a feature, either for selection or for use in a weak classifier, can be performed in $O(|E|)$ time.

In this context, we hope for the filters from the first category to improve the training time significantly without diminishing the quality of the results. On the other hand, we hope for filters from the second category to improve the

quality of the results and perhaps offer a modest improvement in training time.

6.1 Fast Filters

RND The most obvious filtering strategy is simply to select a subset of the features at random. This method can be counted on to maintain most of the diversity of the original feature set and to avoid any of the systematic mistakes associated with other methods. However, in cases where there are but a few useful features among many useless ones in the feature set, the method can select the good features only by being lucky. The running time for this filter is $O(|F'|)$.

RANK Another simple strategy is to rank the features, taking only those that are most discriminative in themselves. For instance, one can measure each feature's discriminative power by first finding a threshold that maximizes the corresponding binary feature's mutual information with the class label. The features having the greatest mutual information with the class label are retained, while the rest are filtered out. The problem with this method is that the selected features might be redundant, meaning that they tend to classify the same instances correctly [9]. In an extreme case, the set of instances classified correctly by at least one feature might be no larger than the set classified correctly by the single best feature. That is, one feature would be as discriminative as the entire selected pool.

6.2 Slow Filters

One natural way, to avert this redundancy problem is to iteratively add the feature that contains the most information about the class label given the features that have already been selected. That is, iteratively add the feature that contains the most "new" information about the class label. Unfortunately, it is impractical to consider the joint distribution of examples over many feature values.

CMIM One strategy used by Vidal-Naquet and Ullman [23], as well as Fleuret [4], to cope with this problem is to consider only pairwise distributions of examples. We assume that the feature has been converted to binary by selecting the threshold that gives the greatest mutual information with the class label, just as was done for the ranking method. We call these binary features $\{\omega_j\}$. Let F'_n be the set of selected features when n features have been selected, and let $P_n = F - F'_n$ be the set of remaining features. In the first iteration, F'_1 is initialized to contain only the feature with the greatest mutual information with the class label C, i.e $\arg\max_{\omega_j \in P_0} I(\omega_j; C)$. In all subsequent iterations, we add the feature that contributes the most pairwise mutual information with the class label; i.e. for $n > 1$,

$$\arg\max_{\omega_i \in P_n} \left\{ \min_{\omega_j \in {F'}_n} I(\omega_i; C|\omega_j) \right\}$$

is added to F'_n to form F'_{n+1} and the feature is deleted from P_n to form P_{n+1}. Intuitively, one can think of the quantity $I(\omega_i; C|\omega_j)$ as an asymmetric distance, so that the algorithm picks the feature that is furthest away from all previously selected features, thus obtaining a set that spans the feature space.

Although this method avoids the particular redundancy associated with ranking, other types of redundancy may persist in the feature set. The problem lies is the fact that the algorithm only considers pairs of features. Again, we consider the extreme case, where a feature $\omega_3 \in P_n$ is the XOR of $\omega_1, \omega_2 \in {F'}_n$. Although the inclusion of ω_3 would add no new information to the selected feature pool, the selection criterion would not capture this fact, as both $I(\omega_3; C|\omega_1)$ and $I(\omega_3; C|\omega_2)$ could be non-zero. Thus, considering the class distribution over pairs of features does not eliminate the redundancy problem.

Fleuret [4] has recently published a faster implementation of the algorithm based on a lazy evaluation of the conditional information. The key observation is that $\max_{\omega_i \in P_n} \min_{\omega_j \in {F'}_n} I(\omega_i; C|\omega_j)$ can only decrease as more features are selected. Thus, once we have evaluated this quantity for an iteration, we have an upper bound of the quantity for all subsequent iterations. We only need to update the quantity when this upper bound no longer assures us that another feature is better. In practice, this makes a dramatic difference in the computational cost.

FFS Another alternative to considering the joint distribution over large feature sets is to assume that the utility of a feature set can be summarized in the majority vote of the binarized features. With this assumption, Forward Feature Selection (FFS) [28] is a natural, greedy approach to feature filtering. Instead of adding the feature with minimum error with respect to a changing weight distribution, as Adaboost does, FFS keeps the weight distribution over the examples constant and adds the feature that most improves the majority vote classification of the ensemble of features as a whole.

7 Results

All experiments presented in this paper are performed on the CMU-MIT face detection data set. We use the original

Viola-Jones features and the automatic cascade training procedure described in section 4.4. Unless otherwise indicated, all experiments were performed with a goal detection rate of $D_g = 0.95$, a goal false positive rate of $F_g = 3 \cdot 10^{-8}$, $\eta = 1.0$ (though the false positive goal is favored when both are unachievable), an overall cost of $G = 0.05$, and a maximum number of hypotheses in an ensemble of $L = 200$. Although this methodology is not guaranteed to produce the best ROC curves, we hope this consistency will help make comparisons with other work straightforward.

The detector is applied to an image by scanning a detection window over the image at a sequence of scales, each scale being a factor of 1.25 larger than the previous one. Naturally, the detector returns positive results on overlapping windows, so some post-processing is required to clean up the output. Our post-processing step is to define a graph over the positive windows by placing edges between windows where intersection is a high percentage (63%) of the larger window. This ensures that windows that actually detecting the same thing have an edge between them. We then return one averaged window for each sufficiently large connected component in the graph. An example of this process is shown in figure 4.

The best comparison between two detectors is realized by visualizing the full ROC curve [5], which shows the full detection versus false positive tradeoff. Unfortunately, visualizing an ROC curve for every comparison made in this section is impractical. Therefore, in some cases we will summarize the overall performance of a detector by averaging the detection rate over 0-130 false positives. This measure is analogous to the standard "area under curve", used when the detection rate is plotted against the false positive rate, rather than the number of false positives. Because there are 130 images in the CMU-MIT data set, this upper bound on the number of false positives represents an average of one false positive per image.

To account for the effects of post-processing, we produce a ROC curve for the entire CMU-MIT data set for several rule sets and then take the highest detection rate for each number of false positives. Thus, for every point on the ROC curve, there is a post-processing rule that achieves that result.

To measure the speed of the detector, we calculate the average number of features applied to the detection window as it is scanned over entire data set. Note that this can be different from the number of hypotheses applied when CART trees are used in the detector.

7.1 Feature Selection

7.1.1 "Fast" Filters

As discussed in section 6.1, we examine two fast feature selection methods: random selection and ranking by mutual information. Each of these methods was used to reduce the feature pool by 90% (RND13473 and RANK13473) and 99% (RND1347 and RANK1347) during the training of several detectors. We performed a full factorial analysis over the amount of filtering (90% and 99%), the filtering method (random selection or ranking), and the boosting method (Adaboost, RealBoost, or GentleBoost), training a total of twelve detectors. Results for Adaboost are shown in figure 5.

In both cases, random selection gives comparable performance to the ranking method. Table 5 shows that these results hold regardless of whether Adaboost, Realboost, or Gentleboost is used. At first, this may seem counterintuitive. The ranking method does, after all, include the most discriminative features. How can a random selection of features produce detectors that perform just as well? The answer is the redundancy problem identified in section 6.1. The "best" features tend to misclassify the same examples, making it difficult for Adaboost to learn an ensemble of hypotheses that classifies these examples correctly. This confirms the well-known problems of ranking for feature selection [9].

We illustrate this redundancy phenomenon by plotting the cumulative distribution of the fraction of features that misclassify an example. Mathematically, we define this distribution by letting V be the set of either positive or negative validation examples and letting

$$M(s) = \frac{|\{f \in F' : s \,\text{ismisclassifiedby}\, f\}|}{|F'|}$$

be the fraction of selected features that misclassify the example $s \in V$. We then plot the distribution

$$R(x) = \Pr_{s \in V}\left[M(s) \leq x\right].$$

One can think of this distribution as being formed by plotting the fraction of incorrect features for each example on the real line, and then counting the fraction of these points that are less than x to form the cumulative distribution $R(x)$. This distribution is similar to the cumulative distribution of the margin, a more familiar concept in the machine learning literature (e.g. [18]). In our plots, we remove the need for choosing a decision boundary by plotting the positive and negative examples separately, while still showing the same phenomena.

Figure 4: Detected windows before (left) and after (right) post-processing.

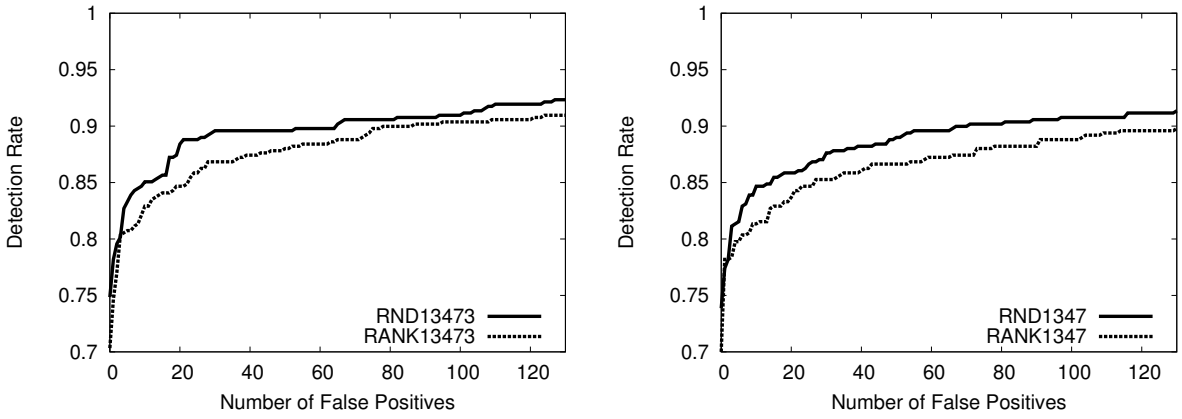


Figure 5: Comparison of ROC curves for detectors trained with random feature selection (RND) and ranking by mutual information (RANK) for 90% feature reduction (left) and 99% feature reduction (right). Results for disrete Adaboost shown.

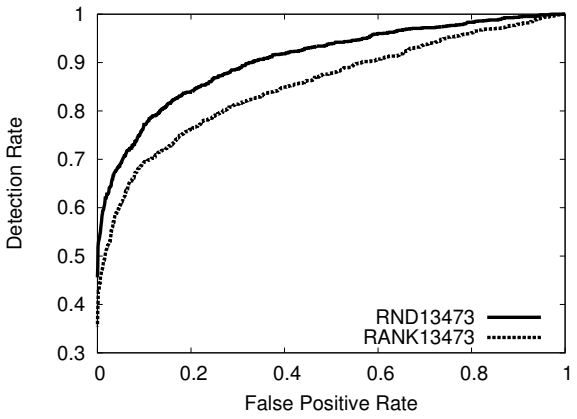


Figure 7: ROC curves of the voting ensemble formed by the features selected by the RND and RANK filters for the 5th stage. The ROC curve for the RND features is clearly better, but Adaboost is able to make the RANK method more competitive by changing the feature threshold and using more features.

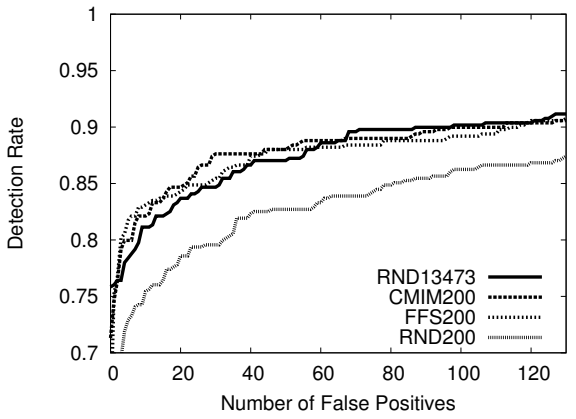


Figure 8: ROC curves for detectors using CMIM and FFS filtering methods. Results for random feature pools also shown for comparision. Both "slow" filtering methods perform than random selection (RND200), but do no better than not filtering at all (RND13473). Results shown with Gentleboost.

In figure 6, we compare these distributions for positive and negative examples for a representative stage (the fifth one) of the detectors trained with Adaboost. We observe that there is much greater variance in the number of features that misclassify an example when the ranking method is used. For instance, most positive examples are misclassified by between 10% and 35% of the the features when random selection is used, but this range widens to 5% to 80% when ranking is used.

This observation is a clear indication of the redundancy problem in the ranking method. Some examples are almost always classified correctly, while others are almost always misclassified. Note that although $R(x)$ is greater when the ranking method is applied for many x, this is not an indication that the feature set is better. If we consider a voting ensemble of the selected features, there is no point on the ROC curve where the ranking method is better than the random one, as shown in figure 7. This means that Adaboost's task of combining these features to meet the performance goals for the stage is more difficult when ranking selection is used. In fact, for the fifth stage shown in the figures, 190 iterations of Adaboost are required, instead of the 101 iterations when random selection is used.

7.1.2 "Slow" Filters

To assess the asymtotically slower methods, conditional mutual information maximization (CMIM) and forward feature selection (FFS), we first randomly selected 10% of the features and then used the methods to filter down to 200 features. For a baseline comparison we also trained a detector with 200 randomly selected features (RND200). The ROC curves for the resulting detectors trained with Gentleboost are shown in figure 8. Both FFS and CMIM produce ROC curves comparable to the one produced by RND13473. That is, the detectors perform as well as they would if no filtering had been applied at all. Thus, although these methods offer a modest improvement in training time, they do not outperform the greedy selection naturally employed by Adaboost.

Our results also show that CMIM and FFS are effective at eliminating the redundancy problem associated with ranking. Figure 9 shows the distributions $R(x)$ for the fifth stage of detectors employing the filtering methods. The distributions show approximately the same amount of variance as RND in the number of misclassifications, though the means differ more markedly.

7.2 Alternative Weak Learners

7.2.1 CART-based Hypotheses

Our experiments show that CART-based detectors offer improved detection rates with only small drops in speed. The ROC curve of figure 10 shows the improvement coming from using CART trees of depth 2, 4, and 6, as opposed to stumps (i.e. threshold-based hypotheses). This result is especially significant, because using deeper CART trees is

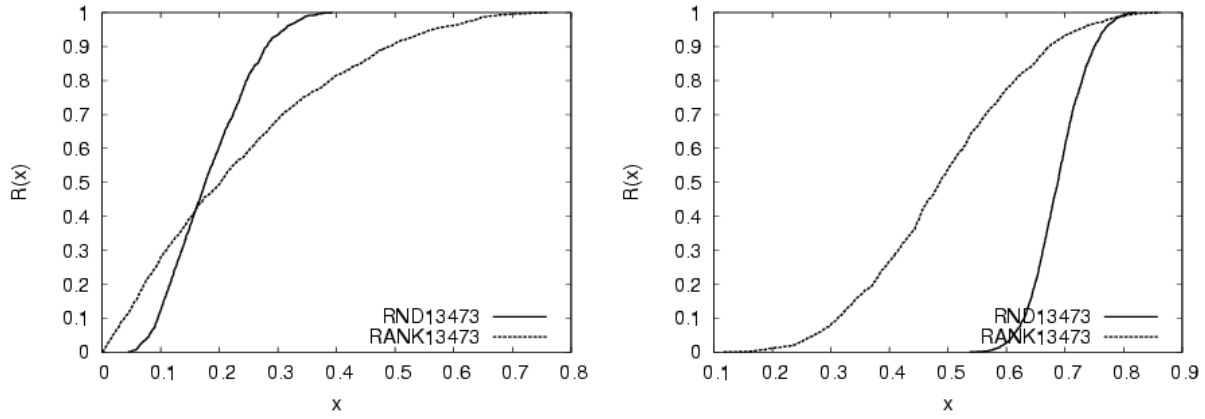


Figure 6: The distributions $R(x)$, where x is the fraction of features that misclassify an example, for positive examples (left) and negative examples (right) for the fifth stage of detectors using random (RND) and ranking (RANK) feature selection. Notice the greater variance in the RANK distribution. Results for the 5th stage shown.

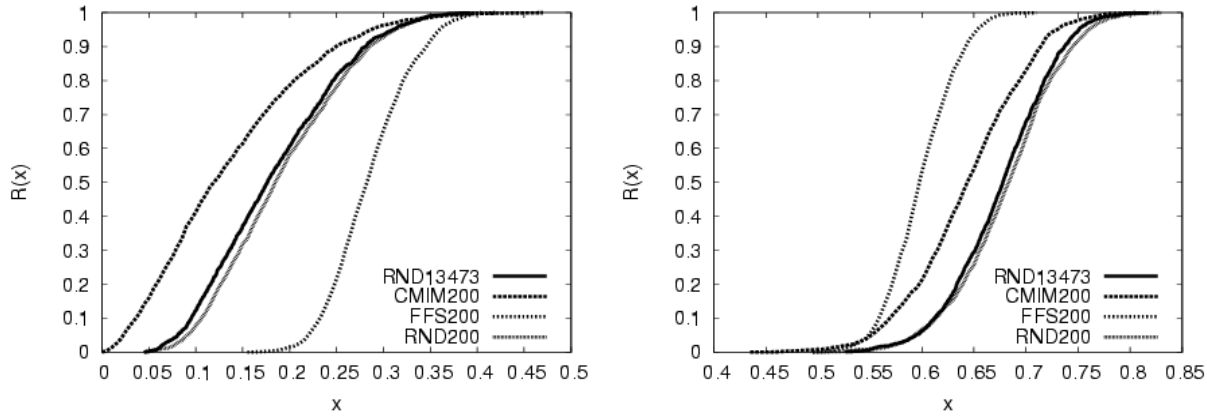


Figure 9: The distribution $R(x)$, where x is the fraction of features that misclassify an example, for positive examples (left) and negative examples (right) for the fifth stage of detectors using CMIM and FFS with analgous distributions shown for random selection. Notice how the distributions for CMIM and FFS do not have a large variance and are comparable to the RND distributions with slightly shifted means.

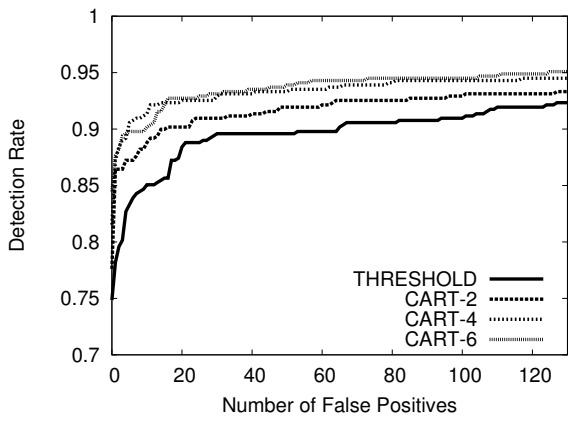


Figure 10: CART depths up to 4 significantly and consistently improve performance. Results shown for discrete Adaboost.

the only change in this study that shows results better than our result for the original Viola and Jones strategy. A comparison among several standard detectors is given in table 1.

Table 2 shows that CART-based hypotheses improve results across the various boosting algorithms with a minimal loss in speed. Unfortunately, the measurements for the average number of features applied the image do not show much consistency, creating some uncertainty on the latter point. We attribute this to the fact that the boosting algorithms seek to minimize the misclassification error, or some closely related quantity, rather than trying to meet the stage criteria directly.

One might reasonably argue that the above comparison is unfair to threshold-based detectors, because with the limit of 200 hypotheses per stage the detectors using non-stump CART trees are allowed to apply more features to the detection window. For example a hypothesis using 4 deep CART trees can apply up to 800 features to the detection window in a given stage, whereas the threshold-based detector can only apply 200 per stage. Therefore, we trained two more detectors that allow 400 and 800 iterations of discrete Adaboost and compared the results to the detectors using 2-deep and 4-deep CART trees. The results are shown in figure 11. Although allowing more threshold-based hypotheses per stage does improve the results, better detection results are achieved if these "extra" features are combined with others in CART trees.

7.2.2 Histogram-base Hypotheses

Our experiments show that histogram based hypotheses are more powerful than theshold-based hypotheses in the early stages of the cascade, requiring fewer hypotheses to meet the goal criteria for a stage. However, in the later stages, as the task of separating the faces from the false positives of the currently trained partial cascade becomes harder, the ensemble of histogram-based hypotheses is no better at meeting the goal criteria than threshold-based ensembles are. The result is that histogram-based detectors are faster at runtime, but do not produce better detection performance. Table 3 shows that this result holds over the various boosting methods employed in our experiments.

7.3 Variations on Boosting

Table 5 shows the full set of results obtained in this study. Surprisingly, this reveals that the method of boosting has no significant effect on the overall performance of the detector. The boosting method does, however, seem to have an signifiant, though not always consistent, effect on the number of features that need to be applied to the image. Thus, we may say that the boosting method affects the efficiency of the detector.

Table 4 shows the number of features applied per window averaged over the experiments in this study for methods of boosting and types of weak hypotheses. For all three categories of weak hypotheses (threshold-based, CART-based, and histogram-based) Gentleboost seems to be more efficient than Adaboost. Realboost, however, is the most efficient when CART-based hypotheses are used and the least efficient otherwise. A plausible explanation for this latter result is that the ERROR and CONFIDENCE combination associated with Realboost (see section 5) is well suited to the case where the partitions are relatively pure (close to all positive or to all negative), as one would expect for deeper CART trees, but that they are poorly suited to impure partitions, as one would expect for threshold-based or coarse histograms.

7.4 Cascade Learning

The effectiveness of our cascade learning strategy (see section 4.4) has been demonstrated by the fact that all detectors presented in the paper were trained using this method. Our improved algorithm is powerful enough to produce state of the art results without any manual intervention and is robust enough to produce reasonable results even when the feature pool is reduced to 200 randomly selected features.

	False Positives							
Detector	6	10	21	46	50	65	78	95
Viola-Jones	–	0.761	0.884	–	0.914	0.920	0.921	0.929
Schneiderman	0.897	–	–	0.957	–	–	–	–
Lienhart et al	–	0.82	–	–	0.90	–	–	–
CART-4 w/ Realboost	0.891	0.905	0.929	0.935	0.935	0.943	0.948	0.951

Table 1: A comparison of detection rates for several standard detectors. Results for Viola-Jones are from [26], results for Schneiderman are from [20], and results for Lienhart are from the graphs in [14].

CART Depth	Boosting	# of Features for first 3 stages	Avg. # of Features applied per window	Avg. Detection Rate
1	Adaboost	16, 41, 82	138.0	0.889
2	Adaboost	48, 50, 52	135.6	0.910
4	Adaboost	24, 44, 88	107.3	0.926
6	Adaboost	54, 66, 54	155.4	0.929
1	Realboost	29, 59, 83	162.3	0.868
2	Realboost	24, 48, 42	75.0	0.903
4	Realboost	20, 24, 44	75.6	0.930
6	Realboost	24, 24, 54	75.0	0.929
1	Gentleboost	17, 36, 32	93.3	0.866
2	Gentleboost	14, 60, 40	79.6	0.915
4	Gentleboost	44, 112, 48	126.4	0.929
6	Gentleboost	36, 78, 120	155.4	0.936

Table 2: Deeper CART-based detectors produce better detection results, sometimes at the expense of speed.

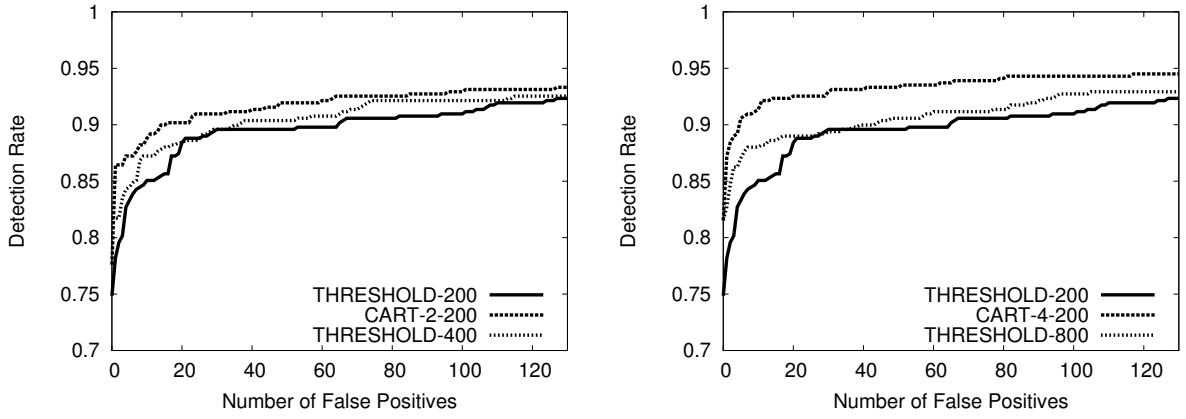


Figure 11: CART-based hypotheses produce better detectors, when we control for the maximum number of features that can be applied to a detection window. The notation HYPO-E indicates that the hypotheses were HYPO-based and that maximum ensemble length was fixed at E.

Weak Learner	Boosting	# of Features for first 3 stages	Avg. # of Features applied per window	Avg. Detection Rate
Threshold	Adaboost	16, 41, 82	138.0	0.889
Histogram	Adaboost	14, 39, 48	89.7	0.877
Threshold	RealBoost	29, 59, 83	125.5	0.877
Histogram	RealBoost	33, 32, 70	114.8	0.875
Threshold	GentleBoost	17, 36, 32	93.3	0.866
Histogram	GentleBoost	13, 17, 23	57.2	0.870

Table 3: Histogram-based detectors are consistently faster but do not consistently produce better results than threshold-based detectors. Average Detection rate is taken over 0 to 130 false positives.

Category	Adaboost	Realboost	Gentleboost
Threshold-based	142.8	147.2	84.7
CART-based	132.8	75.5	95.8
Histogram-based	89.7	114.8	57.2

Table 4: Average number of features applied per detection window in each of the weak learning categories. Gentleboost is consistently more efficient than discrete Adaboost. RealBoost is the least efficient except when CART-based hypotheses are used.

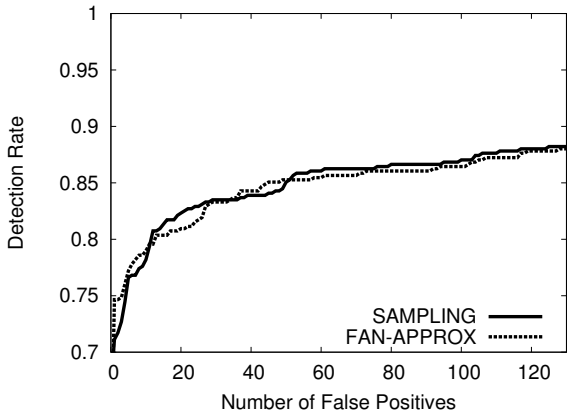


Figure 12: The sampling strategy and Fan's approximation strategy produce comparable results. Detectors were trained using a random 10% feature set, threshold-based hypotheses, and discrete Adaboost.

Two methods were presented for estimating the overall cascade operating point. For consistency, only the sampling method was used for experiments presented thus far, but as shown in figure 12, the two methods seem to produce similar results.

The reader may have noticed that the early stages of the cascades include more hypotheses than other systems. In most other systems, the number of hypotheses in a stage is determined by hand, and in an effort to create a fast detector the early stages are forced to contain only a few hypotheses. Such intervention is compatible with the cascade learning strategy presented here, so long as the cascade learner is provided with measurement of the earlier stages' detection and false positive rates.

8 Conclusion

We have described a novel algorithm for fully-automatic cascade training based on a probabilistic prediction of cascade performance. The approach removes much of the guess-work associated with training cascades of boosted ensembles in the past, and we hope that it will provide a framework for controlled experiments comparing cascaded detectors in the future.

Using this approach, we analyzed the influence of several other factors on the performance of the final detector. We found that the most effective way to improve the performance of a CoBE detector is to learn stronger weak hypotheses. Combining the original Viola and Jones features into CART trees produces significantly improved results. We expect that others ways of creating stronger weak hypotheses would be effective as well.

Although feature filtering would seem to hold great promise as a way to speed up the training process or improve the detector performance, several standard techniques, including ranking by mutual information, conditional mutual information maximization, and forward feature selection, were ineffective in our experiments. We also show that when the cascade learning is properly controlled, the method of boosting has little impact on the overall performance of the detector, though it can have an effect on its efficiency.

Feature Selection	Feature Pool Size	Weak Learner	Boosting	Avg. # of Features Applied	Avg. Detection Rate
Random	13473	Threshold	Adaboost	138.0	0.889
Ranking	13473	Threshold	Adaboost	168.8	0.872
Random	1347	Threshold	Adaboost	147.0	0.880
Ranking	1347	Threshold	Adaboost	157.7	0.860
CMIM	200	Threshold	Adaboost	143.1	0.868
FFS	200	Threshold	Adaboost	115.2	0.875
Random	200	Threshold	Adaboost	148.9	0.841
Random	13473	Threshold	Realboost	125.5	0.877
Ranking	13473	Threshold	Realboost	162.3	0.868
Random	1347	Threshold	Realboost	144.6	0.852
Ranking	1347	Threshold	Realboost	196.7	0.848
CMIM	200	Threshold	Realboost	127.4	0.871
FFS	200	Threshold	Realboost	134.5	0.868
Random	200	Threshold	Realboost	186.1	0.830
Random	13473	Threshold	Gentleboost	71.1	0.881
Ranking	13473	Threshold	Gentleboost	92.4	0.872
Random	1347	Threshold	Gentleboost	78.5	0.874
Ranking	1347	Threshold	Gentleboost	191.1	0.834
CMIM	200	Threshold	Gentleboost	65.4	0.870
FFS	200	Threshold	Gentleboost	93.5	0.860
Random	200	Threshold	Gentleboost	101.2	0.829
Random	13473	CART-2	Adaboost	135.6	0.910
Random	13473	CART-4	Adaboost	107.3	0.926
Random	13473	CART-6	Adaboost	155.4	0.929
Random	13473	CART-2	Realboost	75.0	0.903
Random	13473	CART-4	Realboost	75.6	0.930
Random	13473	CART-6	Realboost	75.0	0.929
Random	13473	CART-2	Gentleboost	61.0	0.905
Random	13473	CART-4	Gentleboost	100.8	0.920
Random	13473	CART-6	Gentleboost	125.8	0.932
Random	13473	Histogram	Adaboost	89.7	0.877
Random	13473	Histogram	Realboost	114.8	0.875
Random	13473	Histogram	Gentleboost	57.2	0.870
Random	13473	Threshold	Adaboost	148.0	0.898 [6]
Random	13473	Threshold	Adaboost	153.2	0.900 [7]

Table 5: Full table of all results of this study.

Notes

[1] Schapire and Singer use the same convention in section 4 of [19], saying the weight coefficients are "folded into" the hypothesis.

[2] We use the same "validation" data in the FIND-BEST-THRESHOLD step as we do in the VALIDATE step. Although a purist would object, this works well in practice.

[3] A more strict definition of a voting ensemble might require that the classifier be of the form $\sum_i a_i h_i(x) > \theta$, where h_i is a concept returning either 0 or 1. This restriction causes a vote's weight to be fixed; whereas the h in the confidence-rated hypotheses used in equation 2 might adjust their weight according to x. If the confidence-rated hypotheses return only two distinct values, then the two definitions are equivalent, but if they can take on more values, then the classifier described in equation 2 does not meet the more strict definition given here.

[4] A set of points that are equivalent in this sense form an indifference curve.

[5] Technically, we will be plotting the detection rate versus the number of false positives on the CMU-MIT data set, rather than a true ROC curve which plots the detection rate versus the false positive rate.

[6] Maximum Ensemble length is not 200 but 400.

[7] Maximum Ensemble length is not 200 but 800.

References

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[2] M. Elad, Y. Hel-Or, and R. Keshet. Pattern detection using a maximal rejection classifier. *Pattern Recognition Letters*, 23(12):1459–1471, 2002.

[3] D.-Y. Fan. The distribution of the product of independent beta variables. *Communications in Statistics - Theory and Methods*, 20:4043–4052, 1991.

[4] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, pages 1531–1555, 2004.

[5] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[6] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting, 1998.

[7] B. Froba and A. Ernst. Face detection with the modified census transform. In *The Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 91–96, May 2004.

[8] A. K. Gupta and S. Nadarajah, editors. *Handbook of Beta Distribution and its applications*. Marcel Dekker, Inc., 2004.

[9] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, pages 1157–1182, 2003.

[10] B. Heisele, T. Serre, S. Mukherjee, and T. Poggio. Feature reduction and hierarchy of classifiers for fast object detection in video images. In *Proc. CVPR*, volume 2, pages 18–24, 2001.

[11] D. Keren, M. Osadchy, and C. Gotsman. Antifaces: A novel, fast method for image detection. *IEEE Trans. on PAMI*, 23(7):747–761, 2001.

[12] K. Levi and Y. Weiss. Learning object detection from a small number of examples: the importance of good features. In *Proc. CVPR*, 2004.

[13] S. Z. Li and Z. Q. Zhang. Floatboost learning and statistical face detection. *IEEE Trans. on PAMI*, 26(9), 2004.

[14] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Technical report, MRL, Intel Labs, 2002.

[15] C. Liu and H. Shum. Kullback-leibler boosting. In *Proc. CVPR*, volume I, pages 587–594, 2003.

[16] H. Luo. Optimization design of cascaded classifiers. In *Proc. CVPR*, 2005.

[17] S. Romdhani, P. Torr, B. Schoelkopf, and A. Blake. Computationally efficient face detection. In *Proc. ICCV*, pages 695–700, 2001.

[18] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.

[19] R. E. Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[20] H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection. In *Proc. CVPR*, 2004.

[21] J. Sun, J. M. Rehg, and A. Bobick. Automatic cascade training with perturbation bias. In *Proc. CVPR*, 2004.

[22] K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Trans. on PAMI*, 20(1):39–51, 1998.

[23] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *Proc. ICCV*, 2003.

[24] P. Viola and M. Jones. Fast and robust classification using asymmetric AdaBoost and a detector cascade. In *NIPS 14*, 2002.

[25] P. Viola and M. J. Jones. Robust real-time object detection. CRL 2001/01, Compaq Cambridge Research Laboratory, Feb 2001.

[26] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

[27] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Fast asymmetric learning for cascade face detection. Technical report, Georgia Institute of Technology, 2005.

[28] J. Wu, J. M. Rehg, and M. D. Mullin. Learning a rare event detection cascade by direct feature selection. In *NIPS 16*, 2004.

[29] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. In *Proc. ICCV*, volume I, pages 709–715, 2003.

[30] M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Trans. on PAMI*, 24(1):34–58, 2002.