24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Relaxed Per-Stage Requirements for Training Cascades of Classifiers

Dariusz Sychel and **Przemysław Klęsk** and **Aneta Bera**[1]

Abstract. Historically first training algorithms for cascades of classifiers were guided by constant per-stage requirements (constraints) imposed on false alarm and detection rates. Those constant values were calculated according to the geometric mean rule, implied by a pair of final requirements predefined for the whole cascade.

We provide and prove a theoretical result demonstrating that the presence of slack between the constant requirements and actual rates observed while learning, allows to introduce new *relaxed* requirements for each successive stage and still complete the training procedure successfully (with final requirements satisfied). The relaxed requirements can be met more easily, using fewer features. This creates a potential possibility to reduce the *expected number of features* used by an operating cascade — the crucial quantity we focus on in the paper. Taking advantage of the relaxation, we propose new stage-wise training algorithms that apply two approaches: uniform or greedy. They differ in the way the slack cumulated so far becomes "consumed" later on.

Reported experimental results pertain to cascades trained to work as face or letter detectors, with Haar-like features or Zernike moments being the input information, respectively. The results confirm shorter operating times of cascades obtained by the proposed technique, owing to the reduction in the number of extracted features.

1 Introduction

Cascades of classifiers were in principle designed to work as classifying systems that operate under the following specific conditions: (1) very large number of incoming requests, (2) significant imbalance of classes. The natural scenario where these two conditions appear are detection procedures. Images, video sequences or sound data can be analyzed in order to detect some target objects or events of interest. Typically, detection procedures perform dense data scans using a sliding window, and thereby generate thousands of requests for classification in short periods of time. Apart from computer vision, cascades can be also applied in various batch classification jobs. In such scenarios, we expect the classification system to process in the background large portions of data incoming on regular basis, e.g.: medical samples, shopping transactions, satellite photos, system logs, etc..

The second condition pointed out — imbalance of classes — should not be seen as a difficulty but rather a favorable setting that makes the whole concept viable. Namely, a cascade should vary its computational efforts depending on the contents of an object to be classified. Obvious negatives (non-targets) should be recognized fast, using only a few extracted features. This is because negative objects constitute a vast majority of all objects (e.g. background regions in computer vision applications). On the other hand, positive objects (targets) or the ones resembling them, should be allowed to employ more time and computations based on hundreds or even thousands of features.

There exist a certain *average* value of the computational cost incurred by an operating cascade (in between the two mentioned extremes). This quantity can be defined mathematically as an *expected value* (we do this in Section 2.3) and, in fact, calculated explicitly for a given cascade in terms of:

- number of features applied on successive stages,
- false alarm rates on successive stages,
- detection rates (sensitivities) on successive stages,
- probability distribution from which the data is drawn.

Since the true probability distribution underlying the data is typically unknown in practice, the exact expected value cannot be determined. Interestingly though, it can be very accurately approximated using just the first two pieces of information in the list.

Training procedures for cascades are time-consuming. For a suitably complex problem the training may take days or even weeks to complete. As Viola and Jones noted in their pionieering works [22, 23], cascade training is a difficult combinatorial optimization problem which involves the following parameters: number of cascade stages, number of features on successive stages, selection of those features, and finally decision thresholds. It is worth to remark that this problem has not been ultimately solved yet. Viola and Jones tackled it by imposing: the number of stages (say K stages) and the *final requirements* the whole cascade should meet in order to be accepted by the user. These requirements were defined by a pair of numbers (A, D), where A denotes the largest allowed false alarm rate (FAR), and D the smallest allowed detection rate (sensitivity). Due to probabilistic properties of cascade structure, one can apply the geometric mean to translate the final requirements onto another pair of numbers — *per-stage requirements* — $(a_{\max}, d_{\min})$, where $a_{\max} = A^{1/K}$ and $d_{\min} = D^{1/K}$. If each stage satisfies such requirements then the whole cascade also satisfies the final requirements.

Many modifications to cascade training have been introduced over the years. Most of them try out different: feature selection approaches, subsampling methods, or are simply tailored to a particular type of features [9, 4, 13, 10, 21]. It is fair to remark that Haar-like, HOG and LBP features are by far the most common ones in literature. Some authors obtain modified cascades by designing new boosting algorithms that underlie the training process [18, 17, 15].

In our view, a particularly elegant algorithm was proposed by Saberian and Vasconcelos [15]. The authors try to optimize explicitly a Lagrangian representing the trade-off between cascade's error rate and the operating computational cost. The approach is based on

[1] Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, ul. Żołnierska 49, 71-210 Szczecin, Poland, email: {dsychel,pklesk,abera}@wi.zut.edu.pl

recursive formulas (both for the cascade response and the number of applied features) that are translated from discrete non-differentiable versions to smooth ones using a certain mathematical trick. The trick replaces indicator functions (i.e. zero-one functions that signal if a cascade stage is executed or not) by hyperbolic tangent functions that are smooth. In consequence, the gradient descent optimization can be performed. The resulting training procedure is not stage-wise. All stages are potentially kept open. Each gradient step can either: add a new weak classifier to any of the stages, or create a new stage by cloning the last existing one (operation proven to be neutral). The whole approach is analytically tractable, which is a great property, but there are some drawbacks too. First of all, the gradient descent can get stuck in local optima [19]. Secondly, it is difficult to guess a good value for the Lagrange multiplier. Last, but not least, the performed optimization is in a sense exhaustive. One has to check *all* variational derivatives based on features at disposal for *all* open stages to find the largest reduction of optimization criterion. In other words, the algorithm proposed in [15] becomes more and more complex as the training progresses. Successive steps require roughly $O(Kn)$ of work, where K is the current number of stages and n is the number of features.

Despite the development of deep learning techniques, recent literature shows that cascades of classifiers are still widely applied in detection systems or in expensive batch classification jobs. Let us enumerate a few examples: crowd analysis and people counting [1], human detection in thermal images [16], localization of white blood cells [5], eye tracking [11, 6, 8], detection of birds near high power electric lines [12].

The main contribution of this paper are two new cascade training algorithms based on the so-called *relaxed* per-stage requirements. Such requirements can be introduced by observing the actual rates (false alarm rate and detection rate) obtained in the progress of the algorithm, and the slack created by them with respect to final requirements. In Section 3 we prove a theorem that motivates the usage of relaxed requirements. We analyze some properties implied by this theorem and indicate how they can reduce the *expected number of features* in an operating cascade. The new algorithms themselves are stated in Section 4. The central role in both of them is played by the geometric mean that is updated in a specific way — uniformly or greedily — and allows to calculate relaxed requirements for subsequent stages. Experimental results, presented in Section 5, confirm the usefulness of our approach.

2 Preliminaries

2.1 Notation

Throughout this paper we shall use the following notation:

- K — number of cascade stages,
- $n = (n_1, n_2, \dots, n_K)$ — numbers of features used on successive stages,
- $(a_1, a_2, \dots, a_K)$ — FAR values on successive stages (false alarm rates),
- $(d_1, d_2, \dots, d_K)$ — sensitivities on successive stages (detection rates),
- A — required FAR for the whole cascade,
- D — required detection rate (sensitivity) for the whole cascade,
- $a_{\max} = A^{1/K}$ — per-stage FAR requirement (when calculated as a geometric mean as in Viola-Jones' approach),
- $d_{\min} = D^{1/K}$ — per-stage sensitivity requirement (when calculated as a geometric mean as in VJ approach),
- $F = (F_1, F_2, \dots, F_K)$ — ensemble classifiers on successive stages (the cascade),
- A_k — effective FAR observed up to k-th stage of cascade ($A_k = \prod_{1\leqslant i\leqslant k} a_i$),
- D_k — effective sensitivity observed up to k-th stage of cascade ($D_k = \prod_{1\leqslant i\leqslant k} d_i$),
- p — true probability of the positive class (unknown in practice),
- $1-p$ — true probability of the negative class (unknown in practice),
- $\#$ — set size operator (cardinality of a set),
- $\mathcal{D}$ — training data set,
- $\mathcal{V}$ — validation data set.

The probabilistic meaning of relevant quantities above is as follows. The final requirement on false alarm rate means that:

$$P\left(F(\mathbf{x}){=}+\,|y{=}-\right) \leqslant A. \tag{1}$$

The final requirement on detection rate (sensitivity) means that:

$$P\left(F(\mathbf{x}){=}+\,|y{=}+\right) \geqslant D. \tag{2}$$

Actual false alarm and detection rates observed during the training procedure are, respectively, equal to

$$\begin{aligned} a_k &= 1{-}P\left(F_k(\mathbf{x}){=}-\,|y{=}-, F_1(\mathbf{x}){=}\cdots{=}F_{k-1}(\mathbf{x}){=}+\right) \\ &= P\left(F_k(\mathbf{x}){=}+\,|y{=}-, F_1(\mathbf{x}){=}\cdots{=}F_{k-1}(\mathbf{x}){=}+\right), \\ d_k &= P\left(F_k(\mathbf{x}){=}+\,|y{=}+, F_1(\mathbf{x}){=}\cdots{=}F_{k-1}(\mathbf{x})=+\right). \end{aligned} \tag{3}$$

2.2 Classical cascade training (Viola-Jones style)

In algorithmic pseudocodes to follow in this paper we shall use the $\|$ symbol to denote concatenation of cascade stages. For example, when the current cascade is $F = (F_1, \dots, F_k)$ then the notation $F\|F_{k+1}$ should be understood as an extended cascade $(F_1, \dots, F_k, F_{k+1})$, but having no effect on F so far. Whereas, $F := F\|F_{k+1}$ means that the next stage has been in fact appended to the cascade, so that in effect it becomes $F = (F_1, \dots, F_k, F_{k+1})$.

The classical cascade training algorithm presented below (Algorithm 1) can be treated as reference for new propositions given in this paper. Please note, in the final line of the pseudocode, that we return $(F_1, F_2, \dots, F_k)$ rather than $(F_1, F_2, \dots, F_K)$. This is because the training procedure can potentially stop early, when $k < K$, provided that the final requirements (A, D) for the entire cascade are already satisfied i.e. $A_k \leqslant A$ and $D_k \geqslant D$.

The step "Adjust decision threshold" requires a more detailed explanation. The real-valued response of any stage can be suitably thresholded to obtain either some wanted sensitivity or FAR. Hence, the resulting $\{-1, +1\}$-decision of a stage is, in fact, calculated as the sign of expression[2]

$$F_k(\mathbf{x}) - \theta_k,$$

where θ_k represents the decision threshold. Suppose $(v_1, v_2, \dots, v_{\#\mathcal{P}})$ denotes a sequence of sorted, $v_i \leqslant v_{i+1}$, real-valued responses of a new cascade stage F_{k+1} obtained on positive examples (subset $\mathcal{P}$). Then, the $d_{\min}$ per-stage requirement can be satisfied by simply choosing:

$$\theta_{k+1} = v_{\lfloor (1-d_{\min})\cdot \#\mathcal{P}\rfloor}. \tag{4}$$

[2] zero value can be arbitrarily mapped to -1 or $+1$

Algorithm 1 Cascade of classifiers training algorithm with constant per-stage requirements

procedure TRAINVJCASCADE($\mathcal{D}$, A, D, K, $\mathcal{V}$)
  From $\mathcal{D}$ take subset $\mathcal{P}$ with all positive examples,
  and subset $\mathcal{N}$ with all negative examples.
  $F := ()$. ▷ initial cascade — empty sequence
  $a_{\max} := A^{1/K}$, $d_{\min} := D^{1/K}$. ▷ constant requirements
  $A_0 := 1$, $D_0 := 1$, $k := 0$.
  while $A_k > A$ **do**
    $n_{k+1} := 0$, $F_{k+1} := 0$, $A_{k+1} := A_k$.
    $a_{k+1} := A_{k+1}/A_k$.
    while $a_{k+1} > a_{\max}$ **do**
      $n_{k+1} := n_{k+1} + 1$.
      Train new weak classifier f using $\mathcal{P}$ and $\mathcal{N}$
      $F_{k+1} := F_{k+1} + f$.
      Adjust decision threshold θ_{k+1} for F_{k+1}
      to satisfy $d_{\min}$ requirement.
      Use cascade $F \| F_{k+1}$ on validation
      set $\mathcal{V}$ to measure A_{k+1} and D_{k+1}.
      $a_{k+1} := A_{k+1}/A_k$.
    $F := F \| F_{k+1}$.
    if $A_{k+1} > A$ **then**
      $\mathcal{N} := \emptyset$.
      Use current cascade F to populate set $\mathcal{N}$ with
      false detections sampled from non-face images.
    $k := k + 1$
  return $F = (F_1, F_2, \ldots, F_k)$.

2.3 Expected number of extracted features

2.3.1 Definition-based formula

A cascade stops operating after a certain number of stages. It does not stop in the middle of a stage. Therefore the possible outcomes of the random variable of interest, describing the disjoint events, are: $n_1, n_1 + n_2, \ldots, n_1 + n_2 + \cdots + n_K$. By the definition of expected value, the expected number of features can be calculated as follows:

$$E(n) = \sum_{1 \leqslant k \leqslant K} \Big(\sum_{1 \leqslant i \leqslant k} n_i \Big) \left(p \left(\prod_{1 \leqslant i < k} d_i \right) (1 - d_k)^{[k<K]} + (1-p) \left(\prod_{1 \leqslant i < k} a_i \right) (1 - a_k)^{[k<K]} \right), \tag{5}$$

where $[\cdot]$ is an indicator function.

2.3.2 Incremental formula and its approximation

By grouping the terms in (5) with respect to n_k the following alternative formula can be derived:

$$E(n) = \sum_{1 \leqslant k \leqslant K} n_k \left(p \prod_{1 \leqslant i < k} d_i + (1-p) \prod_{1 \leqslant i < k} a_i \right). \tag{6}$$

Obviously, in practical applications the true probability distribution underlying the data is unknown. Since the probability p of the positive class is very small (as already said, typically $p < 10^{-4}$), the expected value can be accurately approximated using only the summands related to the negative class as follows:

$$\widehat{E}(n) = \sum_{1 \leqslant k \leqslant K} n_k \prod_{1 \leqslant i < k} a_i \approx E(n). \tag{7}$$

It is also interesting to remark that in the original Viola and Jones' paper [23] the authors proposed an incorrect formula to estimate the expected number of features, namely:

$$E_{\text{VJ}}(n) = \sum_{k=1}^{K} n_k \prod_{i=1}^{k-1} r_i, \tag{8}$$

where r_i represents the "positive rate" of i-th stage. This is equivalent to

$$E_{\text{VJ}}(n) = \sum_{k=1}^{K} n_k \prod_{i=1}^{k-1} (pd_i + (1-p)a_i). \tag{9}$$

Please note that by multiplying positive rates of stages, one obtains mixed terms of form $d_i \cdot a_j$ that do not have any probabilistic sense. For example for $k = 3$ the product under summation becomes

$$(pd_1 + (1-p)a_1)\,(pd_2 + (1-p)a_2)\,,$$

with the terms $d_1 a_2$ and $a_1 d_2$ having no sense, because a fixed data point does not change its class label while traveling along the cascade.

3 Motivation theorem

The following theorem points out the possibility of *relaxation* of per-stage requirements and thereby constitutes our base motivation to propose new variations of cascade training algorithms.

Theorem 1 *The presence of slack between constant per-stage requirements* (a_{max}, d_{min}) *and actual rates* (a_k, d_k), $k = 1, \ldots, K$, *observed during cascade training —*

$$a_k = (1 - \epsilon_k) a_{max}, \qquad d_k = (1 + \delta_k) d_{min}, \tag{10}$$

where ϵ_k, δ_k *represent slack variables denoting small numbers — allows to introduce new relaxed requirements for each successive stage and carry out a training procedure that still satisfies the final requirements* (A, D) *for the whole cascade. In particular, when the* k*-th stage is done, the following two pairs of relaxed bounds (uniform and greedy) can be applied for the* $(k+1)$*-th stage:*

$$a_{k+1} \leqslant \frac{a_{max}}{(1 - \epsilon_{\leqslant k})^{1/(K-k)}}, \quad d_{k+1} \geqslant \frac{d_{min}}{(1 + \delta_{\leqslant k})^{1/(K-k)}}, \tag{11}$$

or

$$a_{k+1} \leqslant \frac{a_{max}}{1 - \epsilon_{\leqslant k}}, \qquad d_{k+1} \geqslant \frac{d_{min}}{1 + \delta_{\leqslant k}}, \tag{12}$$

where $1 - \epsilon_{\leqslant k} = \prod_{1 \leqslant i \leqslant k} (1 - \epsilon_i)$ *and* $1 + \delta_{\leqslant k} = \prod_{1 \leqslant i \leqslant k} (1 + \delta_i)$.

Before going into the proof, the following remarks should be made. The theorem states that slack variables ϵ_k, δ_k are small numbers, but it purposely does not specify their sign. Intuitively it seems that $\epsilon_k, \delta_k \geqslant 0$. But as a matter of fact, when relaxed bounds are applied throughout the training procedure, some of ϵ_k, δ_k can become negative too. To see this, consider for example the first two variables ϵ_1, ϵ_2 and the bound (12). It implies that $a_2 = (1 - \epsilon_2) a_{\text{max}} \leqslant a_{\text{max}}/(1 - \epsilon_1)$ and therefore:

$$\epsilon_2 \geqslant \frac{-\epsilon_1}{1 - \epsilon_1}. \tag{13}$$

Simultaneously note that $\epsilon_1 \geqslant 0$ since $a_1 = (1 - \epsilon_1) a_{\text{max}}$ must not exceed the original bound a_{max}. Therefore, (13) informs that ϵ_2

may, in particular, be negative. What is important though, is that the 'effective' slack variables $\epsilon_{\leqslant k}, \delta_{\leqslant k}$ (resulting from the product rules) should be guaranteed to be non-negative, so that the relaxation in bounds (11), (12) indeed takes place. The following lemma states this fact and is further used to prove the main theorem.

Lemma 1 *$\epsilon_{\leqslant k} \geqslant 0$ and $\delta_{\leqslant k} \geqslant 0$ for all $k = 1, \ldots, K$ and regardless of the bound applied: uniform* (11) *or greedy* (12).

Proof. It suffices to prove the lemma only for $\epsilon_{\leqslant k}$ variables, the arguments are analogical for $\delta_{\leqslant k}$ with direction of all inequalities reversed. We first look at the uniform bound (11) and prove the lemma by induction. Base: $\epsilon_{\leqslant 1} = \epsilon_1 \geqslant 0$ follows from the bound on $a_1 = (1-\epsilon_1)a_{\max} \leqslant a_{\max}/\prod_{1\leqslant i\leqslant 0}(1-\epsilon_i)^{1/(K-0)}$ since the empty product in the denominator yields 1. Inductive step: suppose the lemma is true for all indexes up to k, i.e. $\epsilon_{\leqslant k} \geqslant 0$. Then, to see that $\epsilon_{\leqslant k+1} \geqslant 0$, it suffices to show that $1 - \epsilon_{\leqslant k+1} = (1-\epsilon_{\leqslant k})(1-\epsilon_{k+1}) \leqslant 1$. From bound (11) we have that:

$$\begin{aligned} a_{k+1} &\leqslant \frac{a_{\max}}{(1-\epsilon_{\leqslant k})^{1/(K-k)}} \\ (1-\epsilon_{k+1})a_{\max} &\leqslant \frac{a_{\max}}{(1-\epsilon_{\leqslant k})^{1/(K-k)}} \\ (1-\epsilon_{k+1})(1-\epsilon_{\leqslant k})^{1/(K-k)} &\leqslant 1 \\ (1-\epsilon_{k+1})(1-\epsilon_{\leqslant k}) &\leqslant 1 \\ 1-\epsilon_{\leqslant k+1} &\leqslant 1. \end{aligned} \tag{14}$$

The third pass is true because the inductive assumption $\epsilon_{\leqslant k} \geqslant 0$ implies that expression $(1-\epsilon_{\leqslant k})^{1/(K-k)}$ is a fraction raised to a fractional power. Hence, omitting the power lowers the left-hand-side. This proves the lemma correctness for the uniform bound. It is easy to check that the inductive step for the greedy bound (12) leads directly to inequality: $(1-\epsilon_{k+1})(1-\epsilon_{\leqslant k}) \leqslant 1$ and the base is satisfied as well. ☐

Proof of Theorem 1. By virtue of lemma 1, note that when $\epsilon_{\leqslant k}, \delta_{\leqslant k} > 0$ the denominators in both (11) and (12) cause that new per-stage requirements can be met more easily than original ones. We now limit the considerations only to the sequence of false alarm rates $a_1, \ldots, a_K$ (the arguments are analogical for detection rates). Suppose k training stages are already done. To satisfy the final requirement the following inequality must hold

$$\begin{aligned} a_1\cdots a_k \cdot a_{k+1}\cdots a_K &\leqslant A \\ \underbrace{(1-\epsilon_1)a_{\max}\cdots(1-\epsilon_k)a_{\max}}_{k \text{ initial stages}} \cdot a_{k+1}\cdots a_K &\leqslant a_{\max}^K \\ a_{k+1}\cdots a_K &\leqslant \frac{a_{\max}^{K-k}}{\prod\limits_{1\leqslant i\leqslant k}(1-\epsilon_i)}. \end{aligned} \tag{15}$$

Now, it is possible to see the two approaches to bound the remaining rates $a_{k+1}, \ldots, a_K$. The first is to let all of them consume uniformly the slack 'cumulated' so far $\prod_{1\leqslant i\leqslant k}(1-\epsilon_i) = 1-\epsilon_{\leqslant k}$. To do so, think of a mean factor a_* such that $a_*^{K-k} = a_{k+1}\cdots a_K$ in (15). Now, the root of order $K-k$ taken sidewise yields formula (11) — the uniform bound. This approach can be interpreted as an updated geometric mean on remaining false alarm rates. The second approach is to let the very next rate a_{k+1} consume all the slack and assume pessimistically the subsequent rates $a_{k+2}, \ldots, a_K$ to be equal to the original bound $a_{\max}$. This leads to inequality $a_{k+1}a_{\max}^{K-k-1} \leqslant a_{\max}^{K-k}/(1-\epsilon_{\leqslant k})$ and yields formula (12) — the greedy bound. ☐

As regards the second approach, its greediness can be also well understood through the following consequence of relaxed bounds (12).

Corollary 1 *If $(k+1)$-th stage becomes trained according to the greedy approach and the observed resulting a_{k+1} is not less but exactly equal to the bound $a_{max}/(1-\epsilon_{\leqslant k})$, then the next stage requirement for a_{k+2} tightens back to the original bound a_{max}.*

Proof. The result follows directly from (15) by: isolating out a_{k+2}, inserting the right hand side of (12) into a_{k+1} and setting $a_i = a_{\max}$ for all $i \geqslant k+3$. ☐

The same argument is true for any two consecutive sensitivities d_{k+1}, d_{k+2}.

Can relaxed per-stage requirements reduce the expected number of extracted features?

Since relaxed per-stage requirements can be satisfied more easily — with fewer features — one might be tempted to think that this leads directly to a reduction of $\widehat{E}(n)$. Unfortunately, this is not true in general. Even though n_k numbers in (7) can in fact be decreased in many cases, note that the cost paid for that is an increase of false alarm rates a_k that are also present in formula (7). Moreover, note that each increased a_k contributes in a multiplicative manner to *all* subsequent summands in the expectation. Therefore, we remark that the relaxation provides the necessary (but not sufficient) condition to reduce the expected value.

4 Relaxed cascade training

4.1 Relaxed per-stage requirements expressed without slack variables

At implementation level there is no need to explicitly calculate the slack variables. The new per-stage requirements can be expressed in terms of A, D constants and a_i, d_i rates observed so far, that is for $i \leqslant k$. It is easy to check that the following pairs of formulas are equivalent counterparts of the right-hand-sides of (11) and (12), respectively.

$$a_{\max,k+1} = \left(\frac{A}{\prod\limits_{1\leqslant i\leqslant k} a_i}\right)^{\frac{1}{K-k}}, \quad d_{\min,k+1} = \left(\frac{D}{\prod\limits_{1\leqslant i\leqslant k} d_i}\right)^{\frac{1}{K-k}}. \tag{16}$$

$$a_{\max,k+1} = \frac{A^{\frac{k+1}{K}}}{\prod\limits_{1\leqslant i\leqslant k} a_i}, \qquad d_{\min,k+1} = \frac{D^{\frac{k+1}{K}}}{\prod\limits_{1\leqslant i\leqslant k} d_i}. \tag{17}$$

4.2 Training algorithm

Formulas (16) and (17) can be directly applied to form variations of the classical VJ cascade training procedure (Algorithm 1). It suffices to use them as replacements of constant requirements $(a_{\max}, d_{\min})$ in every iteration of the main loop. Algorithm 2 demonstrates the 'relaxed' cascade training procedure.

In the experimental section, we whall refer to Algorithm 2 coupled with formula (16) by the name **UGM** (standing for: Updated Geometric Mean), whereas for Algorithm 2 coupled with (17) we shall use the name **UGM-G** (Updated Geometric Mean – Greedy).

Algorithm 2 Cascade of classifiers training algorithm with relaxed per-stage requirements

procedure TrainRelaxedCascade($\mathcal{D}$, A, D, K, $\mathcal{V}$)
  From $\mathcal{D}$ take subset $\mathcal{P}$ with all positive examples,
  and subset $\mathcal{N}$ with all negative examples.
  $F := ()$ ▷ initial cascade — empty sequence
  $A_0 := 1$, $D_0 := 1$, $k := 0$.
  while $A_k > A$ **do**
    F_{k+1} :=TrainStage($\mathcal{P}$, $\mathcal{N}$, K, k, $\mathcal{V}$, F).
    $F := F \| F_{k+1}$.
    if $A_{k+1} > A$ **then**
      $\mathcal{N} := \emptyset$.
      Use current cascade F to populate set $\mathcal{N}$ with
      false detections sampled from non-face images.
    $k := k + 1$
  return $F = (F_1, F_2, \ldots, F_k)$.

procedure TrainStage($\mathcal{P}$, $\mathcal{N}$, K, k, $\mathcal{V}$, F)
  $n_{k+1} := 0$, $F_{k+1} := 0$, $A_{k+1} := A_k$.
  Calculate relaxed per-stage requirements
  $(a_{\max,k+1}, d_{\min,k+1})$ using (16) or (17).
  $a_{k+1} := A_{k+1}/A_k$.
  while $a_{k+1} > a_{\max,k+1}$ **do**
    $n_{k+1} := n_{k+1} + 1$.
    Train new weak classifier f using $\mathcal{P}$ and $\mathcal{N}$.
    $F_{k+1} := F_{k+1} + f$.
    Adjust decision threshold θ_{k+1} for F_{k+1}
    to satisfy $d_{\min,k+1}$ requirement.
    Use cascade $F \| F_{k+1}$ on validation
    set $\mathcal{V}$ to measure A_{k+1} and D_{k+1}.
    $a_{k+1} := A_{k+1}/A_k$.
  return F_k.

5 Experiments

5.1 Learning algorithm and general settings

In all experiments we apply *RealBoost+bins* [14] as the main learning algorithm, producing ensembles of weak classifiers as successive cascade stages. Each weak classifier is based on a single selected feature. The response of such a classifier is real-valued, calculated according to the logit transform with a binning mechanism.

Experiments on two collections of images are carried out. Firstly, we test the proposed approaches in face detection task, using Haar-like features as the input information. Secondly, we experiment with synthetic images representing letters (computer fonts originally prepared by T.E. de Campos et al. [7]). The letters are placed randomly on a set of backgrounds and we treat the ‘A’ letter as our target object. In that second group of experiments we expect to detect our targets regardless of their rotation (rather than in upright position). To do so, we apply rotationally invariant features based on Zernike moments (ZMs) as the input information [3]. In both cases, feature extraction is backed with integral images (complex-valued for ZMs).

In all experiments we used a machine with Intel Core i7-4790K 4/8 cores/threads, 8MB cache. For clear interpretation of time measurements, we report detection times using only a single thread [ST]. The software has been programmed in C#, with key computational procedures implemented in C++ as a dll library.

5.2 “Face detection” — Haar-like features

Training faces were cropped from 3 000 images, looked up using *Google Images* search engine. The training set contained 7 258 face examples. We preset our features space to contain 14 406 Haar-like features[3]. The test set contained 3 014 faces from *Essex* facial images collection [20, 2] and validation sets contained 1 000 face examples. The number of negatives in the test set was constant and equal to 1 000 000. In order to reduce training time, the number of negatives in training and validation sets was gradually reduced for successive cascade stages, as described in Table 1. Detection times of different cascades were determined as averages from 1000 executions.

Table 1. “Face detection”: experimental setup.

train data		
quantity / parameter	value	additional information
no. of positive examples	7 258	windows with faces
no. of negative examples	139 373/42 742/27 742	on 1st stage / 2nd stage / rest
train set size	146 631/50 000/35 000	examples in total on 1st stage / 2nd stage / rest
validation data		
no. of positive examples	1 000	windows with faces
no. of negative examples	$40\,000/25\,000 - \mathcal{P}$	on 1st stage / rest
test set size	41 000/25 000	examples in total on 1st stage / rest
test data		
no. of positive examples	3 014	windows with faces different scales, skin tones, glasses [20]
no. of negative examples	1 000 000	
test set size	1 001 000	examples in total
detection procedure (scanning with a sliding window)		
no. of repetitions	1000	
image resolution	600×480	
no. of detection scales	5	images scanned with 5 different sizes of window
window growing coefficient	1.2	window widths and heights increase by $\approx 20\%$ per scale
smallest window size	48×48	
largest window size	100×100	
window jumping coefficient	0.05	window jumps equal to ≈5% of its width and height

Fig. 1 shows visual examples of detection outcomes obtained by two best detectors (in terms of the expected number of features), trained to satisfy $A = 10^{-4}$ and $A = 10^{-5}$ FAR requirements.

Tables 3, 4 constitute a detailed comparison of all cascades obtained in face detection experiments, respectively for $K = 5$ and $K = 10$ stages. The final FAR requirements (A) were imposed to be either: 10^{-3}, 10^{-4} or 10^{-5} (that last setting only for cascades with 10 stages). Every row in the tables corresponds to some cascade, represented by two sequences: a sequence of feature counts n_k (top), and a sequence of false alarm rates a_k (bottom). Apart from accuracy measures, we report for each cascade its theoretical expected value $\widehat{E}(n)$ calculated according to formula (7). This value can be compared against an avarage observed on the test set — column avg(n).

Let us comment now on results for cascades with $K = 5$ stages. First of all it is worth noting that all trained cascades satisfied the imposed final requirements. As Table 3 shows, the relaxed greedy bound — UGM-G — produced the best cascades (marked with dark gray), having the smallest expectations: 14.7220 and 23.9587 respectively for $A = 10^{-3}$ and $A = 10^{-4}$. The second-best approach (light gray) was not consistent — for $A = 10^{-3}$, it was the classical VJ cascade yielding the expectation 15.3571, whereas for $A = 10^{-4}$ it was UGM yielding 24.9455.

As regards cascades with $K = 10$ stages, results reported in Table 4 indicate similar tendencies. Again, the smallest expectations were achieved by the UGM-G cascades trained using the relaxed

[3] generated for 6 wavelet templates, 7^2 scales and 7^2 anchoring points

Figure 1. "Face detection": detection examples. False alarms marked in yellow.

greedy bound, regardless of the FAR requirements. The UGM cascade placed itself second in all experiments.

5.3 "Synthetic A letters" — Zernike moments

We remind that a synthetic data set containing capital letters from the modern English alphabet [7] was prepared for this experiment. Fig. 2 gives an overview on the source graphical material, whereas Tab. 2 lists details of the experimental setup.

Table 2. "Synthetic A letters": experimental setup.

quantity / parameter	value	additional information
train data		
no. of positive examples	20 384	windows with letter 'A'
no. of negative examples	50 546	windows with letters other than 'A' plus random samples of backgrounds on each stage after resampling
train set size	70 930	examples in total
validation data		
no. of positive examples	1 000	windows with letter 'A'
no. of negative examples	10 000	windows with letters other than 'A' plus random samples of backgrounds on each stage after resampling
test set size	11 000	examples in total
test data		
no. of positive examples	20 000	windows with letter 'A'
no. of negative examples	1 000 000	windows with letters other than 'A' plus random samples of backgrounds
test set size	1 020 000	examples in total
detection procedure (scanning with a sliding window)		
no. of repetitions	1000	
image resolution	600×480	imposed resolution
no. of detection scales	5	images scanned with 5 different sizes of window
window growing coefficient	1.2	window widths and heights increase by $\approx 20\%$ per scale
smallest window size	100×100	
largest window size	208×208	
window jumping coefficient	0.05	window jumps equal to $\approx 5\%$ of its width and height

Our goal was to detect targets ('A' letters) regardless of their rotation. In training images, the letters were allowed to rotate randomly within a limited range of $\pm 45^\circ$. In test images, the letters were allowed to rotate randomly within the full range of 360°. As features, we applied 540 modules of Zernike moments (see [3] for details).

Figure 2. Objects and backgrounds used in "Synthetic A letters" data. Positives: letters A (a), negatives: other letters (b) + backgrounds (c).

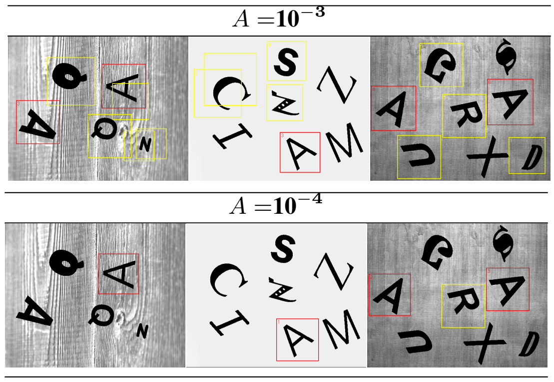

Figure 3. "Synthetic A letters": detection examples.

Fig. 3 presents examples of detection outcomes obtained by best detectors trained to satisfy 10^{-3} and 10^{-4} FAR requirements.

Table 5 presents a comparison of obtained cascades. Despite the small number of features (comparing to the previous experiment), the proposed methods still allow to reduce the expected number of features. FAR and sensitivity measures obtained on the validation set, satisfy the final requirements regardless of the training method. Accuracy measures observed on the test set are slightly worse. The best expected values of the number of feature were again obtained by the UGM-G approach: 2.5682 for $A = 10^{-3}$, and 3.7318 for $A = 10^{-4}$. UGM approach took second places.

5.4 Parallelization

We remarked that for clear interpretation of results, time measurements are reported for single threaded executions [ST], even though we normally apply multiple threads. Time gains that we report for the smallest expectations, may seem very small in ST mode. It should be explained that an improvement of e.g. a 10 ms [ST] constitutes a 6.5% difference per thread and allows for approximately 1 frame more. With 4 threads we observe approximately 46 ms–46 ms reductions, implying 2 FPS more. Common 8-threaded machines or GPUs with even more threads scale the gain further.

6 Conclusion

In our opinion, training a cascade of classifiers should be always carried out with the primary focus on the *expected number of extracted features*, because this quantity reflects directly how fast a cascade operates. In this paper we have demonstrated that the expected value can be improved by relaxation of per-stage requirements. This is achieved by taking advantage of the slack present between the worst-case constant requirements and the actual rates (detection / FAR)

Table 3. “Face detection” — cascades with $K = 5$ stages for different training approaches.

Training alghorithm	Cascade	Expected value	Validation FAR	Validation sensitivity	Test FAR	Test sensitivity	avg(n)	Detection time image [ST] [ms]	Detection time window [ST] [μs]
			Requirements: **0.001000**	**0.9500**				(windows per image: 130 971)	
VJ	9 18 26 30 38 0.2468, 0.2214, 0.2299, 0.2468, 0.2370	15.3571	0.000735	0.9520	0.000711	0.9572	16.21	89	0.680
UGM	9 17 32 29 29 0.2468, 0.2516, 0.2450, 0.2303, 0.2798	15.7243	0.000980	0.9510	0.000980	0.9555	16.58	90	0.687
UGM-G	9 16 21 22 39 0.2468, 0.2548, 0.2234, 0.2635, 0.2606	14.7220	0.000964	0.9510	0.000966	0.9575	15.50	88	0.675
			Requirements: **0.000100**	**0.9500**				(windows per image: 130 971)	
VJ	18 38 40 67 82 0.1516, 0.1582, 0.1452, 0.1533, 0.1509	24.9947	0.000081	0.9520	0.000051	0.9456	26.85	128	0.975
UGM	18 38 40 52 81 0.1516, 0.1582, 0.1452, 0.1656, 0.1694	24.9455	0.000098	0.9510	0.000079	0.9466	26.80	126	0.963
UGM-G	18 32 33 60 100 0.1516, 0.1647, 0.1487, 0.1685, 0.1843	23.9587	0.000115	0.9510	0.000081	0.9509	25.50	126	0.959

Table 4. “Face detection” — cascades with $K = 10$ stages for different training approaches.

Training alghorithm	Cascade	Expected value	Validation FAR	Validation sensitivity	Test FAR	Test sensitivity	avg(n)	Detection time image [ST] [ms]	Detection time window [ST] [μs]
			Requirements: **0.001000**	**0.9500**				(windows per image: 130 971)	
VJ	4 6 16 13 11 17 14 21 24 45 0.4521, 0.4491, 0.4849, 0.4204, 0.4966, 0.4722, 0.4522, 0.4933, 0.4990, 0.4679	12.3741	0.000505	0.9560	0.000512	0.9552	13.91	73	0.561
UGM	4 6 14 10 13 10 13 18 29 17 0.4521, 0.4491, 0.5040, 0.5093, 0.5151, 0.5061, 0.4726, 0.5298, 0.5161, 0.5586	11.9447	0.000980	0.9500	0.001009	0.9575	13.34	72	0.551
UGM-G	4 6 5 14 8 12 17 16 14 34 0.4521, 0.4435, 0.5892, 0.5058, 0.4905, 0.5225, 0.5042, 0.4746, 0.5431, 0.4516	10.7018	0.000899	0.9510	0.000902	0.9522	11.51	67	0.512
			Requirements: **0.000100**	**0.9500**				(windows per image: 130 971)	
VJ	6 13 24 16 34 43 38 46 33 41 0.3546, 0.3914, 0.3939, 0.3945, 0.3256, 0.3615, 0.3907, 0.3977, 0.3865, 0.3940	16.0108	0.000060	0.9550	0.000057	0.9492	17.38	89	0.678
UGM	6 13 24 16 34 40 34 32 42 47 0.3546, 0.3914, 0.3939, 0.3945, 0.3256, 0.4000, 0.3926, 0.4397, 0.4533, 0.3927	15.9898	0.000086	0.9510	0.000076	0.9542	17.35	88	0.671
UGM-G	6 12 15 21 34 19 30 47 29 40 0.3546, 0.4135, 0.4104, 0.4010, 0.4093, 0.3991, 0.3940, 0.3970, 0.4057, 0.3941	14.9437	0.000099	0.9510	0.000105	0.9545	16.26	82	0.625
			Requirements: **0.000010**	**0.9500**				(windows per image: 130 971)	
VJ	9 20 32 40 37 79 61 95 192 132 0.2768, 0.2924, 0.2911, 0.2904, 0.2996, 0.2995, 0.2909, 0.2851, 0.3066, 0.3137	18.5492	0.000005	0.9550	0.000005	0.9393	20.35	95	0.728
UGM	9 20 32 38 41 44 64 78 63 81 0.2768, 0.2924, 0.2911, 0.3241, 0.3031, 0.3339, 0.3311, 0.3197, 0.3254, 0.2975	18.5122	0.000008	0.9510	0.000016	0.9403	20.12	91	0.694
UGM-G	9 19 25 35 29 46 56 46 60 83 0.2768, 0.3529, 0.3200, 0.3005, 0.3202, 0.3053, 0.3032, 0.3581, 0.3103, 0.2965	18.2789	0.000009	0.9510	0.000007	0.9382	19.52	90	0.689

Table 5. "Synthetic A letters" — cascades with $K = 5$ stages for different training approaches.

Training alghorithm	Cascade	Expected value	Validation FAR	Validation sensitivity	Test FAR	Test sensitivity	avg(n)	Detection time image [ST] [ms]	Detection time window [ST] [μs]
			Requirements: **0.001000**	**0.9500**				(windows per image: 18 752)	
VJ	2 2 3 5 7 0.2115, 0.2128, 0.1893, 0.2180, 0.2098	2.6136	0.000389	0.9540	0.000869	0.9313	2.56	127	6.77
UGM	2 2 3 4 5 0.2115, 0.2128, 0.1893, 0.2926, 0.3910	2.5911	0.000975	0.9500	0.002340	0.9244	2.50	125	6.67
UGM-G	2 2 2 3 5 0.2115, 0.2128, 0.2954, 0.2313, 0.2812	2.5682	0.000865	0.9500	0.002016	0.9351	2.50	124	6.61
			Requirements: **0.000100**	**0.9500**				(windows per image: 18 752)	
VJ	3 4 5 8 10 0.1497, 0.1460, 0.1444, 0.1368, 0.1206	3.7393	0.000052	0.9530	0.000163	0.9411	3.84	157	8.37
UGM	3 4 5 7 10 0.1497, 0.1460, 0.1444, 0.1546, 0.1939	3.7350	0.000094	0.9510	0.000295	0.9413	3.78	153	8.16
UGM-G	3 4 5 6 8 0.1497, 0.1460, 0.1444, 0.1881, 0.1525	3.7318	0.000091	0.9510	0.000212	0.9427	3.73	147	7.84

observed during training. Both the mathematical considerations and practical experiments confirmed that this idea can lead to faster classifiers satisfying the same final requirements.

One of the plans for our future research is to incorporate the presented idea into a "branch-and-bound" search technique and to train cascades via both relaxation and searching.

ACKNOWLEDGEMENTS

This work was financed by the National Science Centre, Poland. Research project no.: 2016/21/B/ST6/01495.

REFERENCES

[1] S. S. A. Abbas, P. O. Jayaprakash, M. Anitha, and X. V. Jaini, 'Crowd detection and management using cascade classifier on ARMv8 and OpenCV-Python', in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pp. 1–6, (March 2017).

[2] Faizan Ahmad, Aaima Najam, and Zeeshan Ahmed, 'Image-based Face Detection and Recognition: "State of the Art"', *IJCSI International Journal of Computer Science Issues*, **9**, (2013).

[3] A. Bera, P. Klęsk, and D. Sychel, 'Constant-Time Calculation of Zernike Moments for Detection with Rotational Invariance', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **41**(3), 537–551, (2019).

[4] L. Bourdev and J. Brandt, 'Robust Object Detection via Soft Cascade', in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pp. 236–243. IEEE Computer Society, (2005).

[5] R. A. Maulana Budiman, B. Achmad, Faridah, A. Arif, Nopriadi, and L. Zharif, 'Localization of white blood cell images using Haar cascade classifiers', in *2016 1st International Conference on Biomedical Engineering (IBIOMED)*, pp. 1–5, (Oct 2016).

[6] L. Cuimei, Q. Zhiliang, J. Nan, and W. Jianhua, 'Human face detection algorithm via Haar cascade classifier combined with three additional classifiers', in *2017 13th IEEE International Conference on Electronic Measurement Instruments (ICEMI)*, pp. 483–487, (Oct 2017).

[7] T. E. de Campos et al., 'Character recognition in natural images', in *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, pp. 273–280, (2009).

[8] N. L. Fitriyani, C. Yang, and M. Syafrudin, 'Real-time eye state detection system using Haar cascade classifier and circular Hough transform', in *2016 IEEE 5th Global Conference on Consumer Electronics*, pp. 1–3, (Oct 2016).

[9] J. Gama and P. Brazdil, 'Cascade Generalization', *Machine Learning*, **41**(3), 315–343, (2000).

[10] J. Li and Y. Zhang, 'Learning SURF Cascade for Fast and Accurate Object Detection', in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pp. 3468–3475. IEEE Computer Society, (2013).

[11] Y. Li, X. Xu, N. Mu, and L. Chen, 'Eye-gaze tracking system by Haar cascade classifier', in *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 564–567, (June 2016).

[12] J. Lu, X. Xu, X. Li, L. Li, C. Chang, X. Feng, and S. Zhang, 'Detection of bird's nest in high power lines in the vicinity of remote campus based on combination features and cascade classifier', *IEEE Access*, **6**, 39063–39071, (2018).

[13] M. Pham and T. Cham, 'Fast training and selection of Haar features using statistics in boosting-based face detection', in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–7, (2007).

[14] B. Rasolzadeh et al., 'Response Binning: Improved Weak Classifiers for Boosting', in *IEEE Intelligent Vehicles Symposium*, pp. 344–349, (2006).

[15] Mohammad Saberian and Nuno Vasconcelos, 'Boosting Algorithms for Detector Cascade Learning', *Journal of Machine Learning Research*, **15**, 2569–2605, (2014).

[16] C. H. Setjo, B. Achmad, and Faridah, 'Thermal image human detection using Haar-cascade classifier', in *2017 7th International Annual Engineering Seminar (InAES)*, pp. 1–6, (Aug 2017).

[17] Chunhua Shen, Peng Wang, Sakrapee Paisitkriangkrai, and Anton van den Hengel, 'Training Effective Node Classifiers for Cascade Classification', *International Journal of Computer Vision*, **103**(3), 326–347, (2013).

[18] Chunhua Shen, Peng Wang, and Anton van den Hengel, 'Optimally Training a Cascade Classifier', *CoRR*, **abs/1008.3742**, (2010).

[19] D. Sychel, P. Klęsk, and A. Bera, 'Notes on Expected Computational Cost of Classifiers Cascade: A Geometric View', in *Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM*. SciTePress, (2018).

[20] University of Essex. Face Recognition Data. `https://cswww.essex.ac.uk/mv/allfaces/faces96.html`, 1997. [Online; accessed 11-May-2019].

[21] N. Vallez, O. Deniz, and G. Bueno, 'Sample selection for training cascade detectors', *PLos ONE*, **10**, (2015).

[22] P. Viola and M. Jones, 'Rapid Object Detection using a Boosted Cascade of Simple Features', in *Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pp. 511–518. IEEE, (2001).

[23] P. Viola and M. Jones, 'Robust Real-time Face Detection', *International Journal of Computer Vision*, **57**(2), 137–154, (2004).