Multiclass Adaboost and Coupled Classifiers for Object Detection

Rodrigo Verschae and Javier Ruiz-del-Solar*

Universidad de Chile, Department of Electrical Engineering
{rverscha,jruizd}@ing.uchile.cl

Abstract. Building robust and fast multiclass object detection systems is a important goal of computer vision. In the present paper we extend the well-known work of Viola and Jones on boosted cascade classifiers to the multiclass case with the goal of building multiclass and multiview object detectors. We propose to use nested cascades of multiclass boosted classifiers and we introduce the concept of coupled components in multiclass classifiers. We evaluate the system by building several multiview face detectors, each one built to detect a different number of classes. Thus, we present results showing how well the system scales. Promising results are obtained in the BioID database, showing the potentiality of the proposed methods for building object detectors.

Keywords: Multiclass Adaboost, Coupled Components, Object Detection.

1 Introduction

The development of robust and realtime object detection systems is an important goal of the computer-vision community. The particular case of (multiview) face detection is still an unsolved problem, and at the same time it is an interesting application to test object-detection systems. Moreover, the development of robust face-detection systems is very important from the point of view of many applications, from robotics to multimedia retrieval and security applications.

Most research groups have focused on building one-class object detection systems, mainly because it is still unclear how to extend binary classifiers to the multiclass case. In addition, multiclass systems should be scalable with the number of classes in terms of computational efficiency, training time and robustness. Taking these requirements into consideration, we take the well known work of Viola and Jones [8] on cascade object detection, designed for the one-class case, and extend it to the multiclass case.

Our work builds on the ideas of [9][7] on one-class nested cascade classifiers, and on the ideas of [2][6] on multiclass extentions of Adaboost. The proposed detector corresponds to a *multiclass* cascade that has a *nested* structure, and is trained using a generalized version of Adaboost [2]. We also introduce the use of *coupled* components to train multiclass weak classifiers, a method that can avoid overfitting and produce accurate detections. The main differences with previous works are: *a*) the proposed system is based

* This research was funded by the Millennium Nucleus Center for Web Research, Grant P04-067-F, Chile.

on a multiclass classifier (unlike [9] and [3]), *b*) the classifiers correspond to nested multiclass cascades (unlike [2]), *c*) the training procedure scales well (unlike [6] where the training time grows exponentially with the number of classes), and *d*) no a-priori partition of the classes is used (unlike [2]). In addition, unlike [10], our system is meant to build not only multiview detection systems, but also multiclass detection systems.

The structure of the paper is as follows. In section 2 some related work is presented. The proposed multiclass nested cascade, together with the training algorithms, and three variants to train the multiclass weak classifiers, are presented in section 3. In section 4 results are presented and in section 5 some conclusions of this work are given.

2 Related Work

Cascade classifiers are an efficient way for tackling the problem of object detection. This is due to the natural asymmetry of the classification problem: in the images under analysis most windows (image patches) to be analyzed correspond to non-object windows (background). Thus, to achieve an efficient detection, less time should be spent on non-object windows than on object windows. This can be achieved using cascade classifiers [8], which consists on a sequence of layers/stages of increasing complexity, each one designed to reject non-object windows, and to let object windows pass.

In [8] is proposed the use of Adaboost [5] to train each layer of a cascade. Adaboost builds a model of the form $H(x) = \sum_{t=0}^{T} \hat{h}_t(x)$ by iteratively minimizing an upper bound of the training error: $E[\exp(-yH(x)]$ [1] (with $y \in \{-1, 1\}$), and focusing on wrongly classified samples at each iteration. Thanks to the additivity of the model, it is possible to control the computational complexity of each stage of the cascade.

In [8] a particular feature $f_t(x) \in \mathbf{F}$ is associated to each weak classifier: $\hat{h}_t(x) = h_t(f_t(x)), h_t \in \mathbf{H}$. Each feature ($f_t$) is selected from a family of features ($\mathbf{F}$) of Haar-like wavelets that are evaluated efficiently using the so-called integral image. Examples of other features that have been used in similar systems are edgelets [10], modified Local Binary Patterns (mLBP) [7], and granular features [2]. For the weak classifiers h_t, families of functions $\mathbf{H}$ that have been used include decision stumps (binary [8] and real outputs [6]) , domain partitioning classifiers [9][7], and CART classifiers.

An important improvement over [8] was proposed in [9], where nested cascades are built for reusing information of one stage in the following ones. This is done, again, thanks to the additivity of the model by letting the classifier used at stage k be $H_k(x) = H_{k-1}(x) + \sum_{t=0}^{T} h_{t,k}(f_{t,k}(x))$ with $H_{-1}(x) = 0$, producing a faster and more robust cascade than in the non-nested case. The training procedure of a nested cascade is not trivial and different procedures have been proposed [1][7].

During the last years, several works have tried to deal with the problem of detecting multiple objects classes (e.g. [9][3][4]), but they are based on one-class (binary) classifiers. Their main problems are that they do not exploit possible similarities among classes or they are not scalable with the number of classes. Others (e.g [10][6][2]) have developed multiclass detections systems, trying to overcome some of these problems.

In [10] a tree of boosted classifiers for multiview object (pedestrian and face) detection problems is proposed. Although the core of the training algorithm works on binary

[1] Note that a classification is correct if the margin is positive: $yH(x) \geq 0$.

classification problems, a multiclass tree classifier is built in a recursive fashion, and a subclass partition is learnt. This is done by "splitting", the weak classifiers of the nodes of tree, as well as the training set, when they do not achieve a predefied performance.

In [6] is proposed an algorithm called Jointboost, which is used to build multiclass boosted classifiers and it is tested using up to 32 classes, including frontal faces, cars, keyboards, etc. A vectorized classifier that shares features and weak classifiers among classes is built. It presents interesting characteristics, including: *a*) the number of weak classifiers grows logarithmically with the number of classes, and *b*) the number of training examples per class does not need to be large. However its training time grows exponentially with the number of classes, and a large boosted classifier is built.

In [2] a multiclass version of Adaboost, called Vectorboost, is used to train nodes of a classifier tree. At each node, a different subset of face-views is tested (up to 5 views). The basic idea of Vectorboost is to assign to each class a target region, defined as the intersection of half spaces, in the objective space. This algorithm has some interesting characteristics, including: *a*) the target regions for the classes may overlap (the classes do not "compete"), *b*) one class may not share a target region with any of the other classes (i.e. it is a "negative" class).

3 Proposed Multiclass Nested Cascade

The multiclass classifier used at each layer of the cascade has a vectorized form: $\boldsymbol{H}(x) = \sum_{t=0}^{T} \boldsymbol{h}_t(\boldsymbol{f}_t(x))$ (in the following bold, cursive letters are used for vectors). The training of each layer is performed using an algorithm similar to the one proposed in [2], but that considers the nested structure of the cascade. The basic idea behind this algorithm is to assign to each training example x_i an objective region in a vector space. The objective region is defined as the intersection of a set of half spaces, with each half space defined by a vector $\boldsymbol{a}$, and a set of regions defined by a vector set $\mathbf{A}$. Then a sample x, belonging to class Y_q and represented by a vector set $\mathbf{A}_q$, is classified correctly if and only if $\forall \boldsymbol{a} \in \mathbf{A}_q, \langle \boldsymbol{a}, \boldsymbol{H}(x) \rangle \geq 0$. For simplicity, in the following we consider C as the number of classes we want to detect and the vectors $\boldsymbol{a} \in \{\mathbf{e} \cup -\mathbf{e}\}$, with $\mathbf{e}$ the canonical base.

Given that we have a nested cascade, the actual form of the classifier at layer k is: $\boldsymbol{H}_k(x) = \boldsymbol{H}_{k-1}(x) + \sum_{t=0}^{T} \boldsymbol{h}_{t,k}(\boldsymbol{f}_{t,k}(x))$ with $\boldsymbol{H}_{-1}(x) = 0$, which is trained using the algorithm TRAINLAYER (see Algorithm 1). As already mentioned, the advantages of using nested cascades are that the obtained cascades are more compact and more robust than in the non-nested case. The training of the complete cascade system is based on the TRAINCASCADE algorithm (see Algorithm 3), which iteratively collects "negative" training examples (using a multiclass extention of the BOOTSTRAPP algorithm; see Algorithm 2), and then trains and adds the layers to the cascade. To the best of our knowledge, *multiclass* nested cascades have not been used previously.

To speed up the classification process and to simplify the learning process, we assign a single feature to each weak classifier: $\boldsymbol{h}_t(\boldsymbol{f}_t(x)) = \boldsymbol{h}_t(f_t(x))$. Therefore, at step 4 of Algorithm 1, not only is a classifier, $\boldsymbol{h}_t \in \mathbf{H}$, selected, but also a feature $f_t \in \mathbf{F}$. In the following, to simplify the notation, we will use $h_t(f_t(x), c)$ to refer to the component c of a weak classifier $\boldsymbol{h}_t(f_t(x))$.

Domain partitioning weak classifiers [5] were introduced in the context of face detection by Wu et al. [9]. Basically, given a partition of the feature space, the (weak) classifier assigns a constant output to each region of the partition, i.e. the weak classifier belongs to a family $\mathbf{H}$ of piece-wise functions. In the multiclass case, given a partition $F_1, \ldots, F_J$ of the feature space, we have that $\forall f(x_1), f(x_2) \in F_j, h(f(x_1), c) = h(f(x_2), c) = \alpha_{j,c}$, with $\alpha_{c,j}$ a constant. This kind of classifier has two important advantages: *a*) if the partition is defined as equally sized intervals, the classifier can be evaluated in $O(1)$ thanks to the use of look-up-tables, and *b*) it can be used even when the features do not lie on a Euclidean space [7] (like in the case of mLBP features).

In the present work we considered three different ways of selecting the functions $h(f(x), c)$. In the first one, that we call *independent* components, the components $\{h(f(x), c)\}_{c=1,\ldots,C}$ of $\boldsymbol{h}(f(x))$ are chosen independently of each other (like in[2]). In the second case, *joint* components, the components are chosen jointly (like in [6]), and the same classifier is used for different components/classes, and the remaining components output a zero value: $h_t(f_t(x), c) = \beta_t^c h_t(f_t(x)), \beta_t^c \in \{0, 1\}$. Finally, in the third case, we introduce the concept of *coupled* components, where components share a common function, but for each component this function is multiplied by a different value: $h_t(f_t(x), c) = \gamma_t^c h_t(f_t(x)), \gamma_t^c \in \mathbb{R}$. This last case, *coupled* components, resembles the case of joint components, but it presents several advantages: its training time is much faster (and unlike [6] do not need an heuristics search), it is scalable with the number of classes, and as we will see later, it is much more accurate. *Coupled* components also presents advantages over independent components, e.g. less parameters need to be estimated, which can help to avoid overfitting, in particular for small training sets. Table 1 presents a summary of the optimization problems that have to be solved in each case (given that Domain Partitioning classifiers are used), as well as the number of parameters that have to be estimated, and the order, $O()$, of the training algorithm. Note that for *joint* components the training time grows exponentially with the number of classes, while for *coupled* components grows only quadratically.

Table 1. Comparison of weak classifiers training methods. Domain partitioning classifiers are used. J: number of partitions (bins); M: number of classes; N: number of training examples.

Weak classifier training method	Minimization Problem	Variables $_{j=1,\ldots,J;m=1,\ldots,M}$	Solution	Order $O()$
Jointly trained	$\sum_{j,m} w_+^{j,m} e^{-b_m c_j} + w_-^{j,m} e^{b_m c_j}$	$c_j, \beta_m \in \{0,1\}$	Analytic	$N + J2^M$
Independent	$\sum_{m,j} w_+^{j,m} e^{-c_{j,m}} + w_-^{j,m} e^{c_{j,m}}$	$c_{j,m}$	Analytic	$N + JM$
Coupled	$\sum_{j,m} w_+^{j,m} e^{-\gamma_m c_j} + w_-^{j,m} e^{\gamma_m c_j}$	$c_j, \gamma_m \in \mathbb{R}$	Newton	$N + (J+M)^2$

4 Experimental Results

To evaluate the proposed learning algorithms, we took the problem of multiview face detection, and considered different views as different classes by taking frontal faces and rotating them on multiples of 45 degrees (in-plane rotation) to define new classes.

Table 2 shows a summary of the 8 defined classes. In order to evaluate how well the system scales with the number of classes, we considered 4 problems, each one having

Table 2. Problem setup and training parameters. TPR(FPR): true(false) positive rate per class. ETPR: Expected TPR for a cascade of 14 layers; under this configuration a FPR of 0.410^{-6} is expected for a complete cascade.

Problem	1 class	2 classes	4 classes	8 classes
Classes [degrees]	0	0, 180	0, 90, 180 and 270	0, 45, 90, 135, 180, 225, 270, 315
Min TPR per layer	0.999	0.998	0.996	0.992
Max FPR per layer	0.35	0.35	0.35	0.35
ETPR	0.986	0.972	0.945	0.89

a different number of classes (1, 2, 4 and 8 classes). In all cases the number of training examples per class (face views) was 5000. The number of training examples for the negative class, selected during bootstrapping, was $5000C$, with C the number of object classes. In this way we got an equal number of "positive" and "negative" examples.

When increasing the number of classes for a given detection rate, the number of false positives was expected to be larger. Therefore we set a smaller minimum detection rate per layer when larger number of classes were used (see details in Table 2).

To speed up the training, we used feature sampling and we consider rectangular features in the first two layers, and mLBP in the subsequent layers, as done in [7]. Note that using these feature types imposes an important difference between the problems of 1, 2 and 4 classes with the one of 8 classes (see Table 2), because the problems with 8 classes considers also diagonal rotations (45, 135, 225 and 315 degrees), which are more difficult to detect using the employed features. For the training of the multiclass weak classifiers, we considered *independent*, *joint* and *coupled* weak classifier components.

The obtained results are presented as ROC curves (Fig. 1), Detection Rates (DR), False Positives (FP) and processing times (Table 3). The evaluation was done using the BioID database, which consists of 1521 images (of 384x286 pixels), each one containing one frontal face; the images were rotated to the eight already mentioned rotations.

Figure 1 presents the obtained detection results for class 1 using the multiclass classifiers, omitting the output of other classes (i.e. false positives given for other classes are not considered). Results for other classes are very similar. In the one-class case (Fig. 1, top-left) the three weak learning algorithms gave very similar results, with coupled components working slightly better. In the cases of the two-class (Fig. 1, top-right), and four-class (Fig. 1, bottom-left) classifiers, best results are obtained by coupled components, followed by independent and joint components. In the case of the 8-class classifier (Fig. 1, bottom-left), best results are obtained by independent components, followed by the coupled ones. Table 3 presents a summary of this results in the case of 5 FP.

An interesting result is that for coupled components, the four-class and two-class classifier worked better than the independent one and the joint one, even when the classifier was trained to detect half of the classes (see Table 3). However, for the eight-class case, best results are obtained by independent components, although they are not very good (60% DR for 5 FP). We think that this happens because coupled components are better suited when the features are good discriminants for all classes, while independent components still can give reasonable results when the features are not so good at discriminating the classes, thanks to the independent selection of the parameters.

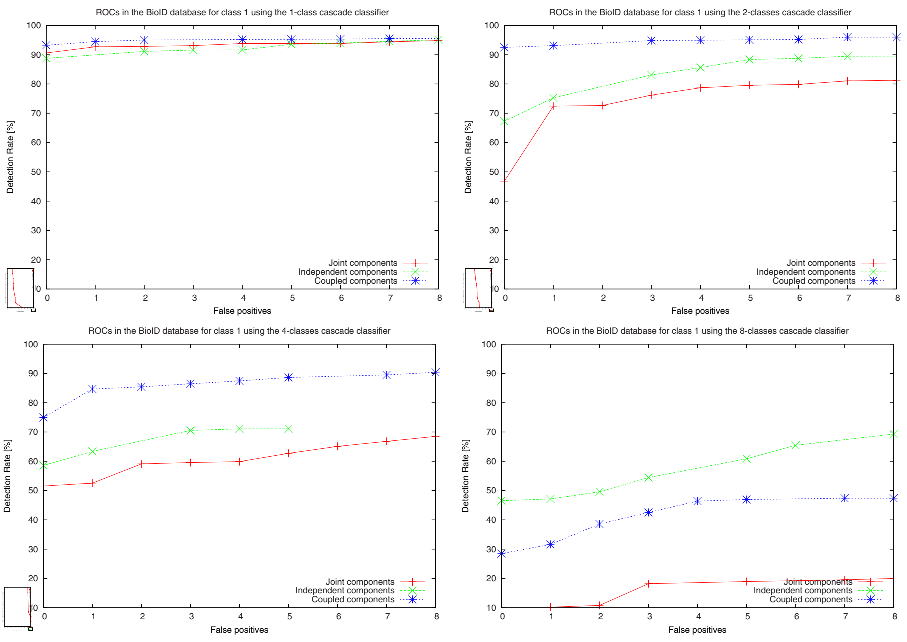

Fig. 1. ROCs for the 1-class (top-left), 2-classes (top-right), 4-classes (bottom-left) and 8-classes (bottom-right) cascades used for detecting faces of class 1

Table 3. Detection rate [%] for 5 false positives; Average processing time [sec] for a 384x286 pixels image; Number of weak classifiers at the first layer of the cascade (NWC)

Weak classifier's training method	Detection Rate				Processing Time				NWC			
	Number of classes											
	1	2	4	8	1	2	4	8	1	2	4	8
Independent	93.56	88.36	71.07	**60.88**	0.37	0.40	0.44	4.46	12	10	8	28
Jointly trained	93.82	79.55	62.72	18.94	0.37	0.54	1.49	0.79	12	9	7	9
Coupled	**95.27**	**95.07**	**88.63**	46.94	0.35	1.0	4.29	4.99	9	11	17	24

Table 3 also summarises some results regarding the processing time and the size of the boosted classifiers. Note that the number of weak classifiers selected at the first layer does not grow faster with the number of classes: in the case of jointly trained components it did not grow, in the case of coupled components it grows logarithmically, while in the case of independent components, the number of weak classifiers remains almost constant for 1, 2 and 4 classes, but in the case of 8 classes it jumps to 28 weak classifiers. In terms of the training time, all methods took about two hours for the one-class case. In the eight-class case it took 51, 56 and 109 hours for the coupled, independent and the joint components respectively (in a 3 GHz Pentium 4 computer). In the 8-class case, the training time of coupled components and independent components is twice as fast as for joint components; this result reflects the analysis of Table 1. Regarding the processing time, jointly trained components is the fastest one, growing very slowly with the number of classes. In the case of coupled components, the processing time grows almost linearly with the number of classes. In the case of independent components the processing speed remains very fast for 2 and 4 classes (about 0.4 sec.), but it goes up to 4.4 seconds (12 times slower than for the 4 class case).

5 Conclusions

In the present paper a multiclass object detection system was proposed. We presented a multiclass nested cascade and proposed training the multiple weak classifiers using *coupled* components. Finally we have built a nested cascade based on the multiclass algorithms already mentioned. We evaluated the system by building multiview face detection systems, considering different numbers of views. *Coupled* components and two other variants were used to build multiclass nested cascade classifiers; good performance and reasonable processing times were obtained. Also, under the best working variants, *coupled* components, the training time did not grow exponentially. As future work we plan to add other feature types, to evaluate the system with other types of objects, and to build a nested tree structure instead of a cascade.

Algorithm 1. TRAINLAYER$(S, F, D, \boldsymbol{H}_{\text{init}})$

```
Input: Training Set $S = \{(x_i, \boldsymbol{a}_i)\}_{i=1,\ldots,m}$
1: Init: $t \leftarrow 0$, $\boldsymbol{H}_0 \leftarrow \boldsymbol{H}_{init}$, $F_0 \leftarrow 1$, $w_0(i) \leftarrow \exp(-\boldsymbol{a}_i \cdot \boldsymbol{H}_0(x_i))$
2: while $F_t > F$ do
3:     Normalize $\{w_t(i)\}_{i=1,\ldots,m_t}$ s.t they add to one
4:     Select $\boldsymbol{h}_t \in \mathbf{H}$, $f_t \in \mathbf{F}$ that minimizes $Z_t = \sum_i w_t(i) \exp(-\boldsymbol{a}_i \cdot \boldsymbol{h}_t(x_i))$
5:     Update weights: $w_{t+1}(i) = w_t(i) \exp(-\boldsymbol{a}_i \cdot \boldsymbol{h}_t(f_t(x_i)))$
6:     Set $\boldsymbol{H_t} \leftarrow \boldsymbol{H}_{t-1} + \boldsymbol{h}_t$ and search threshold for $\boldsymbol{H_t}$ such that for each class $D_t > D$
       and evaluate current false positive rate $F_t$
7:     $t \leftarrow t + 1$
8: end while
Output: Trained layer $\boldsymbol{H}(\cdot) = \sum_{t=0}^{T} \boldsymbol{h}_t(\cdot)$
```

Algorithm 2. BOOTSTRAPP$(S_B, H, M, \mathbf{A})$

```
Input: Boostrapp set $S_B$; Maximum number of negative examples $M$; Classifier $H$; Vector set $\mathbf{A}$
1: $S = \emptyset, i = 0$
2: while $i < M$ do
3:     $x^* \leftarrow$ sample $S_B$
4:     for all $\boldsymbol{a}_j \in \mathbf{A}$ do
5:         if $\langle \boldsymbol{a}_j, \boldsymbol{H}(x) \rangle > 0$ then
6:             $S \leftarrow S \cup \{(x^*, -\boldsymbol{a}_j)\}$
7:             $i \leftarrow i + 1$
8:         end if
9:     end for
10: end while
Output: $S$
```

Algorithm 3. TRAINCASCADE(S, B, F, F_C, D)

```
Input: Set of positive examples $S = \{(x_i, \boldsymbol{a}_i)\}_{i=1,\ldots,m}$; Bootstrapp set $B$
Input: Minimum detection rate per node $D$; Maximum false positive rate per node $F$ and global $F_C$
1: $k = 1, F_k = 1, H \leftarrow 0, S_{B_0} = B, \mathbf{A} = \cup_{i=1}^{m}\{\mathbf{a}_i\}$
2: while $F_k > F_C$ do
3:     $S_{B_k} = $ BOOTSTRAPP $(S_{B_{k-1}}, \boldsymbol{H}, \mathbf{A})$
4:     $\boldsymbol{H}_k = $ TRAINLAYER $(S \cup S_{B_k}, D, F, \boldsymbol{H})$
5:     $\boldsymbol{H} \leftarrow \boldsymbol{H} \cup \boldsymbol{H}_k$
6:     $F_k \leftarrow$ Evaluate current cascade $\boldsymbol{H}$
7:     $k \leftarrow k + 1$
8: end while
Output: Trained Cascade $\boldsymbol{H}$
```

References

1. Bourdev, L., Brandt, J.: Robust object detection via soft cascade. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 236–243 (2005)
2. Huang, C., Ai, H., Li, Y., Lao, S.: High-performance rotation invariant multiview face detection. IEEE Trans. on Pattern Analysis and Machine Intelligence 29(4), 671–686 (2007)
3. Jones, M., Viola, P.: Fast multi-view face detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2003)
4. Li, S.Z., Zhang, Z.: Floatboost learning and statistical face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(9), 1112–1123 (2004)
5. Schapire, R., Singer, Y.: Improved boosting using confidence-rated predictions. Machine Learning 37(3), 297–336 (1999)
6. Torralba, A., Murphy, K.P., Freeman, W.T.: Sharing visual features for multiclass and multiview object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(5), 854–869 (2007)
7. Verschae, R., Ruiz-del-Solar, J., Correa, M.: A unified learning framework for object detection and classification using nested cascades of boosted classifiers. Machine Vision Applications 19(2), 85–103 (2008)
8. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, pp. 511–518 (2001)
9. Wu, B., Ai, H., Huang, C., Lao, S.: Fast rotation invariant multi-view face detection based on real adaboost. In: Proc. of the 6th Int. Conf. on Face and Gesture Recognition, pp. 79–84 (2004)
10. Wu, B., Nevatia, R.: Cluster boosted tree classifier for multi-view, multi-pose object detection. In: Proc. of the Eleventh IEEE Int. Conf. on Computer Vision (ICCV 2007) (2007)