Coarse-To-Fine Multiclass Nested Cascades for Object Detection

Rodrigo Verschae*,+
*Network Design Research Center
Kyushu Institute of Tecnhology (Kyutech)
Fukuoka, Japan
rodrigo@verschae.org

Javier Ruiz-del-Solar+
+Dept. Electrical Engineering - AMTC Center
Universidad de Chile
Santiago, Chile
jruizd@ing.uchile.cl

***Abstract*—Building robust and fast object detection systems is an important goal of computer vision. A problem arises when several object types are to be detected, because the computational burden of running several specific classifiers in parallel becomes a problem. In addition the accuracy and the training time can be greatly affected. Seeking to provide a solution to these problems, we extend cascade classifiers to the multiclass case by proposing the use of multiclass coarse-to-fine (CTF) nested cascades. The presented results show that the proposed system scales well with the number of classes, both at training and running time.**

***Keywords*-Object detection, multiclass cascade, coarse-to-fine, adaboost**

I. Introduction

The development of robust and real-time object detection systems is an important goal of the computer-vision community. Some examples of applications that could make use of multiclass detection system include human computer interfaces (e.g. face and hands detection), autonomous drivings system (e.g. car, pedestrian and sign detection), and security applications.

For detecting all instances of an object class, the sliding window approach [1] [2] requires to perform an exhaustive search over the window patches at different scales and positions of the image. The classification of all possible windows requires a high computational power. 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 [3]. In the case of a multiclass object detection problems, the most simple solution would be using several cascades working in parallel, with one cascade for each object class. However, a problem arises when several object types are to be detected, case where both, the processing time and the false positive rates, increase.

Seeking to provide a solution to these problems, we propose an improvement of our previous work on multiclass nested bootesd cascades [4] by adding a coarse-to-fine (CTF) search in the object target-space. The CTF search allows to: (1) reduce the processing time, (2) reduce the training time, and (3) improve the accuracy of the classifier, making the classifier capable of scaling well with many classes.

II. Related work

In [3], the use of Adaboost [5] to train each layer of a cascade is proposed. Adaboost builds an additive model of the form

$$H(x) = \sum_{t=1}^{T} \hat{h}_t(x) \tag{1}$$

by iteratively adding terms to the sum for minimizing an upper bound of the training error, $E[\exp(-yH(x))]$, with $y \in \{-1, 1\}$, by focusing on wrongly classified samples at each iteration. Thanks to the additivity of the model, when training a coarse-to-fine classifiers (e.g. a cascade), it is possible to control the computational complexity of each stage of the classifier. This can be done, for example, by selecting the number of terms (T) and/or the complexity of the weak classifiers $\hat{h}_t(\cdot)$.

In [3] a particular feature $f_t(x) \in \mathbf{F}$ is associated to each weak classifier:

$$\hat{h}_t(x) = h_t(f_t(x)), h_t \in \mathbf{H} \tag{2}$$

where each feature, $f_t(\cdot)$, is selected from a family of features $\mathbf{F}$ consisting of Haar-like wavelets that can be evaluated very efficiently using the so-called integral image. Examples of other features that have been used in similar systems include edgelets [6] (car and pedestrian detection), modified Local Binary Patterns (mLBP) [7] [8] (face detection), granular features [9] (multiview face detecion), anisotropic Gaussian filters [10] (frontal face detection) and object-part correlation [11] (many objects classes). Regarding the weak classifiers h_t, previously used functions families $\mathbf{H}$ include decision stumps (with binary [3] [10] and real outputs [12] [11]), domain-partitioning classifiers [13] [6] [7], and CART classifiers [14].

An important improvement over the work on cascades classifiers by [3] was proposed in [13] , 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 of the cascade be

$$H_k(x) = H_{k-1}(x) + \sum_{t=1}^{T_k} h_{t,k}(f_{t,k}(x)) \tag{3}$$

with $H_0(x) = 0$ and $k = \{1, \ldots, K\}$, producing faster and more robust cascades compared to the non-nested case. The training of a nested cascade is non-trivial. and different procedures have been proposed (see for example [15] [7]). The procedure used here is an extension of [7], where the number of weak classifiers, T_k, is used to control the trade-off between the speed and accuracy of layer k.

III. Multiclass Adaboost

The basic idea behind the used [9] multiclass boosting 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 a set of half spaces, with each half-space defined by a vector, a, and a set of half-spaces defined by a set $\mathbf{R}$ of parameters. Under this setting, a sample x, belonging to class Y_q and represented by a parameter set $\mathbf{R}_q$, is classified correctly if and only if:

$$\forall \vec{a} \in \mathbf{R}_q, \left\langle \vec{a}, \vec{H}(x) \right\rangle \geq 0 \tag{4}$$

Therefore a class represented by $\mathbf{R}_q$ will be assigned to a new sample x, if this inequalities are fulfilled.

Following [9], the multiclass classifier used at each layer of the cascade has a vectorized form:

$$\vec{H}(x) = \sum_{t=0}^{T} \vec{h}_t(f_t(x)) \tag{5}$$

where in the simplest case, each component can represent a class or view. The training of each layer is performed using an algorithm similar to the one proposed in [9], however there are several differences that are presented in the following sections (some of them in more detail in [4]).

A. Multiclass Weak Classifiers

The weak classifiers are designed after the *domain-partitioning weak hypotheses* paradigm [5]. Each feature domain $\mathbb{F}$ is partitioned into disjoint blocks $F_1, \ldots, F_J$, and a weak classifier $h(f(x), m)$ will have an output for each partition block of its associated feature. The weak classifiers $\{h(f(x), m)\}_m$ can have different relations with each other [4]. In the present work we consider that they are independent of each other as in [9][4].

As part of the optimization problem, the values $\{W_l^{j,m}\}_{j=\{1,\ldots,J\},m=\{1,\ldots M\},l=\{-1,+1\}}$ need to be evaluated, where $W_l^{j,m}$ represents the bin j of a weighted histogram (of the feature values) for the component m for "positive" ($l = 1$) or "negative" ($l = -1$) examples:

$$W_l^{j,m} = \sum_{i,j: f(x_i) \in F_j \wedge (\vec{a}_i)_m = l} w_{i,j}, \tag{6}$$

with $(\vec{a}_i)_m$ representing the value of the component m of the vector $\vec{a}_i$. The evaluation of $\{W_l^{j,m}\}_{m,j,l}$ takes order O(N), with N the number of training examples, and does not depend on J (number of partitions) nor on M (number of classes). This linear dependency, on N only, is important to keep a low computational complexity of the training algorithm.

In contrast to the one-class case, the output of the domain partitioning weak classifier also depends on the component of the multiclass classifier, therefore the output of the classifier, for a feature f, is: $h(f(x), m) = c_{j,m} \in \mathbb{R}$ such that $f(x) \in F_j$.

For each weak classifier, the value associated to each partition block ($c_{j,m}$) is selected to minimize Z_t:

$$\min_{c_{j,m} \in \mathbb{R}} Z_t = \min_{c_{j,m} \in \mathbb{R}} \sum_{j,m} W_{+1}^{j,m} e^{-c_{j,m}} + W_{-1}^{j,m} e^{c_{j,m}} \tag{7}$$

and the value of $c_{j,m}$ depends on the kind of weak classifier being used. Given that we are considering independent components, this minimization problem has an analytic solution:

$$c_{j,m} = \frac{1}{2} \ln \left(\frac{W_{+1}^{j,m} + \epsilon_m}{W_{-1}^{j,m} + \epsilon_m} \right) \tag{8}$$

with ϵ_m a regularization parameter.

In [5], in a two class classification problem, it is shown that it is appropriate to use ϵ on the order of $1/n$, with n the number of training examples. In our multiclass setting, the value of ϵ_m should be of the order of $1/n_m$, with n_m the number of training examples for class m (this can be derived from Eq. (11) on [5]). This corresponds to smoothing the weighted histograms taking into account the number of training samples used to evaluate it.

B. Multiclass Nested Cascades

The nested structure of the multiclass cascade [4] is achieved by reusing information of a layer in the following ones. This is done by letting the classifier at layer k of the nested cascades be:

$$\vec{H}_k(x) = \vec{H}_{k-1}(x) + \sum_{t=0}^{T} \vec{h}_{t,k}(f_{t,k}(x)) - \vec{b_k} \tag{9}$$

$$\text{with } \vec{H}_0(x) = 0.$$

The training of a layer of the multiclass nested cascades is presented in Algorithm 3 and the training of the complete nested cascade is presented in Algorithm 2. The main advantages of using nested cascades are that the obtained cascades are more compact and more robust than in the non-nested case [7] [13]. In algorithm 2 (line 5) the multiclass bootstrapp proposed in [4] is used.

IV. Multiclass CTF Nested Cascades

The main contribution of this paper corresponds to the use of a coarse-to-fine (CTF) search in the object target space. This is done in order to reduce the processing time and the training time, and at the same time to increase the accuracy of the cascade classifier in cases when several classes are used. For this we make use of a function that has a binary output and that represents a subsets of active components (classes) at that layer. At layer k of the cascade, this function is written as $\vec{\mathbf{A}}_k(\cdot)$, we refer to it as *active mask* of layer k, and it is defined component-wise by:

$$\mathbf{A}_k(x,m) = u\left(H_k(x,m)\right)\prod_{i=0}^{k-1}\mathbf{A}_i(x,m) \qquad (10)$$

with $u(x)$ the unit step function (equal to 1 if $x \geq 0$ and 0 otherwise). Note that if $\mathbf{A}_k(x,m)$ is 0, then $\mathbf{A}_j(x,m)$ is 0 for all $j \geq k$. Using the *active mask*, we now redefine the output of a layer, k, of the cascade as:

$$\vec{H}_k(x) = \left[\vec{H}_{k-1}(x) + \sum_{t=1}^{T_k}\vec{h}_{k,t}(f_{k,t}(x)) - \vec{b}_k\right] \odot \vec{\mathbf{A}}_{k-1}(x) \qquad (11)$$

with, $\vec{H}_0(x) = \vec{0}$, $\vec{\mathbf{A}}_0(x) = \vec{1}$, and with $\odot$ the point-wise product between two vectors. Eq. 11 can be interpreted as verifying the condition of the input belonging to a particular class (Eq. 4) at each layer of the cascade in a per component manner, and that this is done only for the subset of hypothesis that were already verified at the previous layers of the cascade. This also allows to use the classifier to detect a subset of classes by setting the corresponding components in $\vec{\mathbf{A}}_0(x)$ to one and the remaining ones to zero.

It must be noticed in this definition that in $\vec{H}_k(\cdot)$, only non-zero components of $\vec{\mathbf{A}}_{k-1}(x)$ need to be evaluated at layer k. These non-zero components represent a subset of classes with positive output at the current layer (and potentially a positive output in the cascade). In this way, as a sample moves through the cascade, the output goes from a coarse output in the target space, to a finner one. This is why we called it a coarse-to-fine cascade in the target object space. Also, as in a standard cascade classifier, there is also a coarse-to-fine search on the non-object space (at each layer a sample can be discarded or passed to the next layer). In the non-CTF cascade (Eq. 9) the decision about the positive classes is done at the end of the cascade, and at each layer only if all components are zero, while here is done component-wise at each layer. Note that the CTF search has to be taken into account during the training of the cascade.

V. 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. In order to evaluate how well the system scales with the number of classes, we considered 4 problems, each one having an increasing 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 $5000M$, with M the number of object classes. In this way we got an equal number of "positive" and "negative" examples. This is the same setting used in [4]. We consider rectangular features in the first two layers, and mLBP in the subsequent layers [7][4].

Figure 1 presents the obtained ROC curves obtained on the BioID database when non-CTF cascades, and CTF cascades are used to detect frontal faces (only the output for one class is considered). As it can be observed, in the case of the non-CTF cascades (Fig. 1 (a)), the performances decreases considerably when increasing the number of classes, while in the CTF cascade (Fig. 1 (b)) the performance only decreases by a small factor when increasing the number of classes, going from 92% (1-class cascade) to 88% (8-classes cascade) for 5 FP when increasing the number of classes.

In terms of the processing time, we observed a logarithmic grow, with the 1-class, 2-classes, 4-classes, and 8-classes detectors taking 0.22ms, 0.33ms, 0.60ms and 0.67ms respectively (the 8-classes case being only three times slower than the 1-class case). On the contrary, in the non-CTF cascades the processing time increased 12 times [4] when going from 1 to 8 classes. In addition, the training time of the CTF cascades was only 13 times longer when going from 1 to 8 classes (recall that the training set is 8 times larger).

VI. Conclusions

We propose the use of CTF multiclass nested cascades on multiclass object detection problems. Results show that the proposed systems scales well with the number of classes, both at training and running time. The processing time grows only logarithmically with the number of classes and the accuracy only decreases by a small factor (4%) when going from 1 to 8 classes, result that is much better than the 25% decrease when a non-CTF multiclass cascade is used.This results show that the proposed methods are suitable for building object detection systems than can scale up with the number of classes.

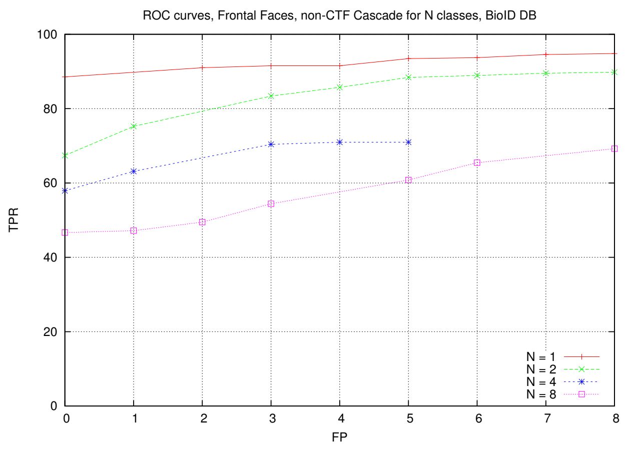
References

[1] E. Hjelmås and B. K. Low, "Face detection: A survey," *Computer Vision and Image Understanding*, vol. 83, no. 3, pp. 236–274, Sep. 2001. [Online]. Available: http://dx.doi.org/10.1006/cviu.2001.0921

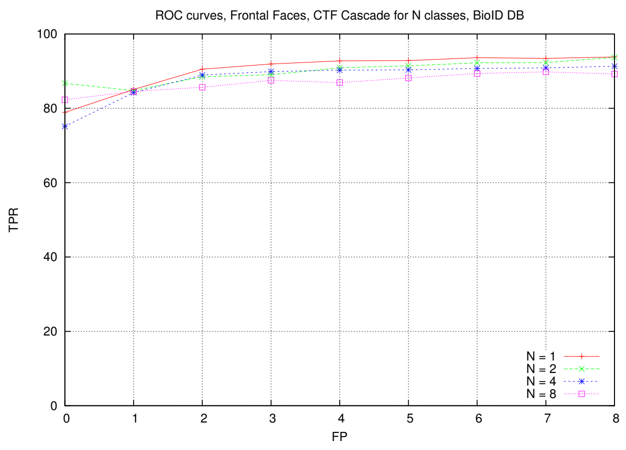
[2] M.-H. Yang, D. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Transactions on PAMI*, vol. 24, no. 1, pp. 34–58, 2002.

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2001, pp. 511–518.

[4] R. Verschae and J. Ruiz-del-Solar, "Multiclass adaboost and coupled classifiers for object detection." in *CIARP*, ser. LNCS, vol. 5197. Springer, 2008, pp. 560–567.



(a) Non-CTF search



(b) CTF seach

Figure 1: CTF and Non-CTF searh in multiclass nested cascade versus number of classes

```
**Input:** Positive examples $S = \{(x_i, \vec{a}_i)\}_{i=1,\ldots,n}$
    Bootstrapp set $B$
    Minimum detection rate per node $D$
    Maximum false positive rate per node $F$
    Global false positive rate $F_C$
 1: $k = 1,\ F_k = 1, \vec{H} = \vec{0}$
 2: $S_{B_0} = B$
 3: $\mathbf{R} = \cup_{i=1}^{n} \{\vec{a}_i\}$
 4: **while** $F_k > F_C$ **do**
 5:     $S_{B_k} =$ BOOTSTRAPPMC$(S_{B_{k-1}}, \vec{H}, \mathbf{R})$
 6:     $\vec{H}_k =$ TRAINLAYERNESTEDMC()
 7:     $\vec{H} \leftarrow \vec{H} \cup \vec{H}_k$
 8:     $F_k \leftarrow$ Evaluate current cascade $\vec{H}$
 9:     $k \leftarrow k + 1$
10: **end while**
**Output:** Trained Cascade $\vec{H}$
```

Figure 2: TRAINCASCADEMC()

```
**Input:** Training Set $S = \{(x_i, \vec{a}_i)\}_{i=1,\ldots,n}$
 1: $t \leftarrow 0,\ \vec{H}_0(\cdot) \leftarrow \vec{0}, w_0(i) \leftarrow 1/n\ ,\ i = 1, \ldots, n$
 2: **for** $T = 0, \ldots, T_{max}$ **do**
 3:     Normalize $\{w_t(i)\}_{i=1,\ldots,n}$ s.t they add to one
 4:     Select $\vec{h}_t \in \mathbf{H},\ f_t \in \mathbf{F}$ that minimizes
        $Z_t = \sum_i w_t(i) \exp\left(-\left\langle \vec{a}_i, \vec{h}_t(f_t(x_i)) \right\rangle\right)$
 5:     Update weights:
        $w_{t+1}(i) = w_t(i) \exp\left(-\left\langle \vec{a}_i, \vec{h}_t(f_t(x_i)) \right\rangle\right)$
 6:     $t \leftarrow t + 1$
 7: **end for**
**Output:** Trained layer $\vec{H}(\cdot) = \sum_{t=1}^{T} \vec{h}_t(\cdot) - \vec{b}_k$
```

Figure 3: TRAINLAYERNESTEDMC()

[5] R. Schapire and Y. Singer, “Improved boosting using confidence-rated predictions,” *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999. [Online]. Available: citeseer.ist.psu.edu/schapire99improved.html

[6] B. Wu and R. Nevatia, “Cluster boosted tree classifier for multi-view, multi-pose object detection,” in *Proc. 11th IEEE Inter. Conf. on Computer Vision (ICCV)*, 2007.

[7] R. Verschae, J. Ruiz-del-Solar, and M. Correa, “A unified learning framework for object detection and classification using nested cascades of boosted classifiers,” *Machine Vision Applications*, vol. 19, no. 2, pp. 85–103, 2008.

[8] B. Fröba and A. Ernst, “Face detection with the modified census transform,” in *Proc. 6th Int. Conf. on Face and Gesture Recognition*, 2004, pp. 1–96.

[9] C. Huang, H. Ai, Y. Li, and S. Lao, “High-performance rotation invariant multiview face detection,” *IEEE Trans. on PAMI. Intell.*, vol. 29, no. 4, pp. 671–686, 2007.

[10] J. Meynet, V. Popovici, and J.-P. Thiran, “Face detection with boosted gaussian features,” *Pattern Recognition*, vol. 40, no. 8, pp. 2283–2291, 2007.

[11] A. Torralba, K. P. Murphy, and W. T. Freeman, “Sharing visual features for multiclass and multiview object detection,” *IEEE Trans. on PAMI*, vol. 29, no. 5, pp. 854–869, 2007.

[12] P. Viola and M. Jones, “Fast and robust classification using asymmetric adaboost and a detector cascade,” in *Advances in Neural Information Processing System 14*. MIT Press, 2002.

[13] B. Wu, H. AI, C. Huang, and S. Lao, “Fast rotation invariant multi-view face detection based on real adaboost,” in *Proc. of the 6th Int. Conf. on Face and Gesture Recognition*, 2004, pp. 79–84.

[14] S. C. Brubaker, M. D. Mullin, and J. M. Rehg, “Towards optimal training of cascaded detectors,” in *9th European Conf. on Computer Vision (ECCV 2006), LNCS 3951*, vol. 2, 2006, pp. 325–337.

[15] L. Bourdev and J. Brandt, “Robust object detection via soft cascade,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) - Vol 2*, 2005, pp. 236–243.