Multimed Tools Appl
DOI 10.1007/s11042-017-4794-7

Indexing quantized ensembles of exemplar-SVMs with rejecting taxonomies

Federico Becattini[1] · **Lorenzo Seidenari**[1] · **Alberto Del Bimbo**[1]

Received: 30 September 2016 / Revised: 21 March 2017 / Accepted: 2 May 2017

Abstract Ensembles of Exemplar-SVMs have been introduced as a framework for Object Detection but have rapidly found a large interest in a wide variety of computer vision applications such as mid-level feature learning, tracking and segmentation. What makes this technique so attractive is the possibility of associating to instance specific classifiers one or more semantic labels that can be transferred at test time. To guarantee its effectiveness though, a large collection of classifiers has to be used. This directly translates in a high computational footprint, which could make the evaluation step prohibitive. To overcome this issue we organize Exemplar-SVMs into a taxonomy, exploiting the joint distribution of Exemplar scores. This permits to index the classifiers at a logarithmic cost, while maintaining the label transfer capabilities of the method almost unaffected. We propose different formulations of the taxonomy in order to maximize the speed gain. In particular we propose a highly efficient Vector Quantized Rejecting Taxonomy to discard unpromising image regions during evaluation, performing computations in a quantized domain. This allow us to obtain ramarkable speed gains, with an improvement up to more than two orders of magnitude. To verify the robustness of our indexing data structure with reference to a standard Exemplar-SVM ensemble, we experiment with the Pascal VOC 2007 benchmark on the Object Detection competition and on a simple segmentation task.

Keywords Media indexing · Object detection · Efficient data structures · Label transfer · Exemplar SVM · Vector quantization · Taxonomy learning

✉ Federico Becattini
federico.becattini@unifi.it

Lorenzo Seidenari
lorenzo.seidenari@unifi.it

Alberto Del Bimbo
alberto.delbimbo@unifi.it

[1] University of Florence, Viale Morgagni, 65, Firenze, Firenze 50134, Italy

1 Introduction

The exponential growth of user produced media has made large scale indexing a central task in computer vision and multimedia analysis. This huge amount of data can be leveraged in training complex computer vision algorithms in order to boost performance in tasks such as image classification, object detection or, more broadly, scene understanding.

Indexing algorithms are not just meant to store and retrieve data, but can be a key component in the process of scaling a computer vision technique to data sizes of today's scenarios. Data therefore requires specific indexing structures in order to be handled properly and be accessed as fast as possible. Several approaches have been developed, mostly based on hashing and taxonomy learning techniques. Such structures are built with a process that is guided by similarity in some visual feature space, possibly exploiting partial matching of images to improve robustness. The task at hand, namely image retrieval, classification or object detection, usually determines the choice of structure and similarity function. In this paper we propose an object based indexing strategy which adopts a metric learning approach to improve with respect to nearest neighbor methods.

Our method, instead of indexing image patches, organizes a set of instance specific classifiers into a taxonomy, exploiting as a key component the Exemplar-SVM framework (ESVM) [26]. Exemplar-SVM learns a single classifier for each available training object, meaning that the positive set will contain a single point, opposite to a wide collection of negative patches. ESVM is considered a semi-parametric approach, where object similarity metrics are trained discriminatively. Highly specific templates are learned in this process improving the quality of nearest neighbours. What makes the Exemplar framework even more appealing is the possibility of maintaining the properties of a nearest neighbour technique, namely the ability of establishing correspondences between train and test samples.

All the annotation available for training instances, such as segmentation masks, 3D information or user tags can be transferred to unlabelled data thanks to this direct correspondence that can only be established with non-parametric approaches. This property has found strong interest in the computer vision community in a broad range of tasks, thanks to being simple and intuitive.

The main drawback of Exemplar-SVMs, which limits the practical use of the method, is related to computational efficiency since thousands of classifiers have to be evaluated. This both affects training and testing time complexity. Training is a less relevant issue since it is usually performed once and for all. Moreover it has been shown that given enough background patches linear classifiers can be trained efficiently [21]. In this paper we present a novel method to learn a taxonomy on exemplars, that can be exploited as an indexing structure. Furthermore we also show how taxonomy nodes can be processed using Vector Quantization obtaining even higher performance.

The goal is to introduce a logarithmic factor instead of a linear one regarding the number of classifiers that need to be evaluated at test time. To further reduce the amount of classifier evaluations we learn node specific thresholds in order to reject patches that are unlikely to be classified as positive by the subsequent nodes. The choice of exploiting a taxonomy is also justified by the fact that it allows us to cluster similar Exemplars, adding implicit information about visual interclass similarity, which is useful for tasks as detection or fine grained classification.

The presented work extends a previous version of the method published as a conference paper [3]. The main contribution of this paper is a novel Vector Quantization strategy, similar to the one of Sadeghi et al. [35, 36], applied to our indexing data structure. We quantize clusters of Exemplar-SVMs and treat them as separate and highly efficient classifiers. This approach differs from Fast Template Vector Quantization (FTVQ) [35] since we quantize cluster centroids instead of proper ESVM classifiers. This permits to evaluate the ensemble in the quantized domain, using these centroids to guide the taxonomy traversal with a reduced computational footprint. We find that exploiting a combination of our speed-up strategies we are able to obtain an improvement of a couple of orders of magnitude in speed. An in depth qualitative and quantitative evaluation of our method is reported, showing the impact of the indexing data structure on an object detection task and a segmentation task.

The paper is organized as follows. In Section 2 an overview on previous related work is presented. After providing background theory notions about Exemplar-SVM ensembles in Section 3, the proposed method is explained in Section 4 and Section 5, focusing on the learning and evaluation parts. Experimental results are discussed in Section 6 and conclusions are drawn in Section 7.

2 Related work

Visual taxonomies are often exploited to stress both computational efficiency and to impose a structure on the data. Regarding efficiency, a logarithmic dependency is created on the number of elements that have to be accessed. On the other hand, the topology that this kind of data structure imposes on the data is useful for classification tasks, since samples are organized ranging from coarse to fine as the tree is traversed from the root to a leaf. Both these aspects, which are directly tied to effectiveness and efficiency, depend of how well data is distributed and how it has been clustered together. In fact the branching factor and the overall balance have an impact on both elements. With this in mind, efforts to learn an optimal taxonomy have been done.

Gao and Koller [17] propose to learn a relaxed hierarchy in which a subset of confusing class labels can be ignored in the upper levels of the tree. This method is derived from [6] where a set of binary classifiers is organized in a tree or a DAG structure. A tree data structure where each node groups a set of category labels more and more specific as one gets closer to the leaves has been proposed in [27].

In [4] spectral clustering is used recursively to optimize an overall tree loss, extending the work of Deng et al. [11] who jointly learn a hierarchical structure and a set of classifiers used at each node. An optimization problem is designed to maximize efficiency given a constraint on accuracy. A similar approach is used in [18] where the confusion matrix of class similarities is exploited to build a label hierarchy.

Liu et al. [25] instead have proposed a probabilistic approach for learning the label tree parameters using maximum likelihood estimation. Similarly, random forests have been exploited to build fast taxonomies for classification [33] and fine-grained categorization [43].

A part from speed issues, taxonomies are also useful to organize data semantically, since objects are often naturally organizable in hierarchies. In [10] a hierarchy is built to represent composition and exclusion relations among classes. Instead of learning a taxonomy in [24] and [20] the ImageNet [9] hierarchy is exploited to transfer annotations between similar classes. Similarly in [34] WordNet [28] is used to propagate knowledge from known to novel categories. In [31] the WordNet taxonomy is used in combination with spatial information to establish scene configurations. Coarse-to-fine taxonomies have been used to classify plants by Fan et al. [15] using a hierarchical multi-task structural learning algorithm.

On a related note, other general methods for speeding up classifiers for object detection tasks have been proposed in literature. Cascade classifiers [40] have found large use in the past years, following the intuition according to which a series of weak classifiers could outperform a single strong classifier. An evolution of this architecture has been proposed introducing soft cascades [5], where the speed-accuracy trade-off is weighed exploring ROC surfaces. With the introduction of Felzenszwalb's Deformable Part Models [16], techniques for speeding up the evaluation have been proposed exploiting Fast Fourier Transform [12], Vector Quantization [35] or combining various strategies to deal with bottlenecks at different stages [36]. Other kinds of optimized data structures, such as hash tables, can also be used in object detection. For instance, Dean et al. [8] exploit Local Sensitive Hashing to replace dot products and effectively detect up to 100.000 classes in less than 20 seconds.

In this paper we exploit taxonomies for indexing ensembles of Exemplar-SVMs (E-SVM) [26], which can benefit from a hierarchical structure to reduce the computational complexity during evaluation. Other efforts to improve the efficiency of ensembles of Exemplar-SVMs have been done in literature. In [21] the problem of training thousands of classifiers has been reconducted to a trivial matrix inversion with Linear Discriminant Analysis. This approach differs from ours since it only takles training efficiency while we focus on the evaluation phase. Therefore the two methods provide benefits that are orthogonal and could be combined.

Sadeghi and Forsyth [35] have proposed a Vector Quantization strategy to speed-up dot products using a lookup table, whereas Context Forests [29] have been used to efficiently predict properties of objects exploiting their global appearance. We adopt [35] as part of our speed-up pipeline (achieving up to two orders of magnitude gain in evaluation speed). As for [29], our method is significantly different since we are able to speed-up the evaluation with a taxonomy using the whole ensemble, whereas they use forests to obtain a retrieval set of relevant exemplars. Moreover they base their forests on global image features while we directly index the Exemplar classifiers.

New formulations of the framework have also been proposed. A joint calibration algorithm for optimizing the ensemble in its entirety is used in [30] to learn specific per-exemplar thresholds; recursive E-SVM is defined by [44], where exemplars are used as visual features encoders and in [23] three different viewpoints for interpreting E-SVM are proposed. All this interest towards Exemplar-SVMs is justified by the wide range of possible label transfer applications that can be paired with object detection: segmentation [39], 3D model and viewpoint estimation [1], part level regularization [2], GPS transfer [19], scene classification [37] amongst others.

3 Exemplar-SVM ensembles

Recently, Exemplar-SVMs [26] have been proposed to perform label transfer among objects in images. Labels can be tags or categorical variables that identify the object (as in an object detection task), or more structured meta-data such as segmentation masks or 3D models. Ensembles of Exemplar-SVMs leverage the intuition according to which a pool of simple classifiers, one for each training sample, can outperform a single and complex one. Moreover, behind this choice lays the desire to be able to exploit in object detection tasks the explicit correspondence typical of a nearest-neighbor method inside a discriminative learning framework, such as Support Vector Machines.

An optimization problem is defined for each exemplar $\mathbf{x}_E$, separating it from the negative windows by learning a weight vector $\mathbf{w}_E$ and a bias b_E which identify the optimal separating hyperplane $\mathbf{w}_E^T\mathbf{x} + b_E = 0$. The optimization problem has the following convex objective function

$$\Omega_E(\mathbf{w}, b) = \|\mathbf{w}\|^2 + C_1 h(\mathbf{w}^T\mathbf{x}_E + b) + C_2 \sum_{\mathbf{x}\in\mathcal{N}_E} (-\mathbf{w}^T\mathbf{x} - b) \tag{1}$$

where h is the hinge loss function $h(x) = \max(0, 1-x)$ and C_1 and C_2 are set separately to moderate the unbalance in the data set. We follow the approach of Malisiewicz et al. [26] which report $C_1 = 0.5$ and $C_2 = .01$.

The evaluation phase of the ESVM framework requires that each classifier is independently tested in a sliding window fashion, generating different detections according to the classifier scores. To perform the step faster one must either reduce the amount of windows to be evaluated or the amount of exemplars. In the following we show how to deal with both issues. First we show how to learn a taxonomy of exemplars per class in order to reduce the number of classifiers. Then we introduce different strategies for reducing the number of test windows such as a rejecting taxonomies and flipped Exemplars. Moreover the use of Vector Quantization tackles the overall speed of the method.

4 Learning an ESVM taxonomy

We propose to build a highly balanced tree using spectral clustering hierarchically (Fig. 1). Different clustering techniques for learning the taxonomy (such as hierarchical k-means and agglomerative clustering with different cluster aggregation criteria) have been tested in a set of preliminary experiments. All but the divisive spectral clustering resulted in poor taxonomies with unbalanced trees which are detrimental to performance; moreover, as

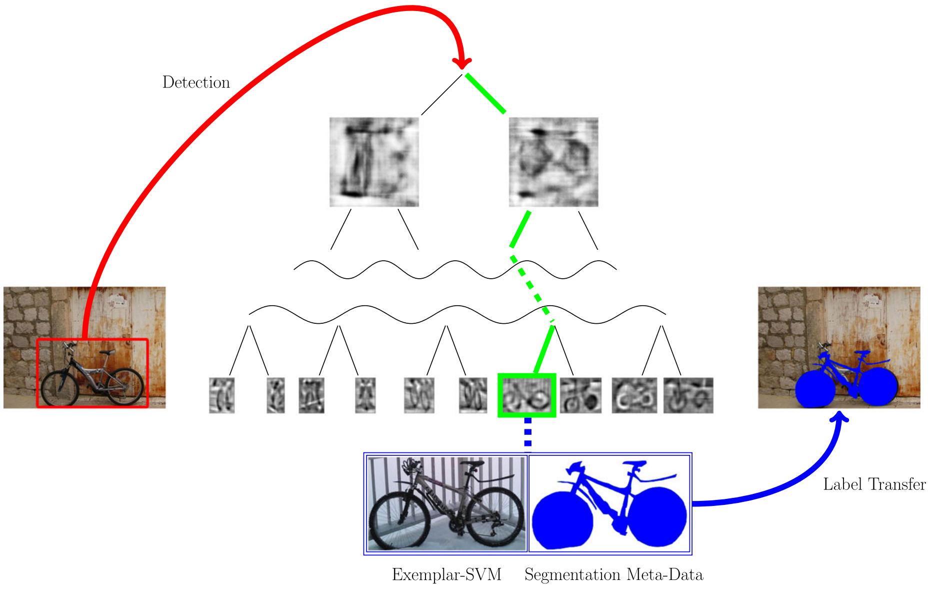

Fig. 1 Overview of our approach. Vector Quantized Exemplars of each class are stored in a learned taxonomy which is traversed to locate the best matching one enabling detection and label transfer at a logarithmic cost

also shown in [22, 37], methods based on Euclidean metrics do not perform well in high-dimensional spaces, due to the well known curse of dimensionality. This is due to the fact that spectral clustering can be seen as a relaxation of a graph partitioning problem [32, 38, 41], which directly optimizes cluster balance. As a quantitative confirmation of this good behavior we perform an experiment to measure the depth and balance of our trees. We use as baseline K-means and compare it to our approach based on spectral clustering.

Table 1 reports a comparative analysis between spectral clustering and K-means, highlighting the benefits of the former in creating a balanced tree. We show statistics for different trees, created using the 20 classes of the Pascal VOC 2007 dataset [14]. Balanced trees have a depth equal to the $log_2(X)$, where X is the data stored in the tree. Therefore comparing tree depth with data cardinality is a good cue of tree balancing. Moreover tree depth also affects efficiency, since in the worst case the amount of comparison can not be larger than the tree depth. Spectral clustering highly outperforms K-means, building trees with depths close to the optimal value. To stress this we also report a balance value, calculated as the average ratio of cluster cardinalities at each split in the tree

$$\texttt{Balance}(N) = \frac{1}{|N|} \sum_{i \in N} \frac{\min(|L_i|, |R_i|)}{\max(L_i|, |R_i)}, \tag{2}$$

where L_i and R_i are the sets of exemplar obtained splitting node i. A perfectly balanced tree would have `Balance`=1, which is obtained when all node splits have the same cardinality. Considering these results we adopted Spectral Clustering for building our taxonomies.

Spectral clustering is a technique that reformulates clustering as a graph partitioning problem, where connected components of the graph are associated with different clusters. In spectral clustering the attention is focused on a tool called graph Laplacian matrix, which establishes the similarity of nearby vertices. In literature various formulations of the Lapla-

Table 1 Spectral Clustering (SC) vs. K-means Clustering (KM)

log_2(Num)	8.26	8.46	8.92	8.18	8.98	7.84	10.29	8.55	9.64	8.02
Depth SC	13	13	13	12	13	12	14	12	14	11
Depth KM	25	37	34	39	47	32	46	31	45	27
Balance SC	0.76	0.77	0.75	0.77	0.76	0.78	0.76	0.76	0.76	0.77
Balance KM	0.49	0.49	0.45	0.42	0.47	0.47	0.47	0.46	0.46	0.46

log_2(Num)	7.75	8.99	8.5	8.41	12.2	9.01	8.01	7.95	8.21	8.34
Depth SC	11	13	13	12	17	13	12	13	13	12
Depth KM	37	30	39	28	59	43	31	31	27	36
Balance SC	0.74	0.75	0.74	0.76	0.76	0.76	0.77	0.73	0.77	0.76
Balance KM	0.43	0.47	0.47	0.51	0.47	0.44	0.48	0.49	0.5	0.47

Statistics on the trees created for each class of the Pascal VOC 2007 dataset [14] are shown. `Num` is the number of Exemplars for each class and its binary logarithm provides a lower bound on the tree depth (perfectly balanced tree). `Depth SC` and `Depth KM` are the depths obtained with Spectral Clustering and K-means, respectively (lower is better). `Balance` indicates the average ratio of cluster sizes at each split in the tree (higher is better, perfect balance is obtained with `Balance` $= 1$)

cian matrix have been proposed. In this paper we follow the approach that refers to the normalized version of [32]:

$$\mathbf{L}_n = \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2} \tag{3}$$

where $\mathbf{S}$ is the affinity matrix and $\mathbf{D}$ is a diagonal matrix where each element $D_{ii} = \sum_{j=1}^{N} S_{ij}$.

We first define the matrix $\mathbf{A}$ that captures the likelihood of two exemplars $\mathbf{w}_i$ and $\mathbf{w}_j$ firing together on the same sample. This matrix represents the compatibility of exemplars. So given the matrix obtained by concatenating all the raw features $\mathbf{H} = [\mathbf{h_1} \ldots \mathbf{h_N}]$ and the matrix of the respective learnt exemplar hyperplanes $\mathbf{W} = [\mathbf{w_1} \ldots \mathbf{w_N}]$ our affinity matrix is defined as:

$$\mathbf{A} = \mathbf{W}^T\mathbf{H}. \tag{4}$$

Therefore element A_{ij} represents the score of exemplar $\mathbf{w}_i$ on the feature $\mathbf{h}_j$ on which exemplar $\mathbf{w}_j$ has been trained and vice versa.

Since the matrix $\mathbf{A}$ is not guaranteed to be symmetric we apply the same strategy as in [4] and define

$$\mathbf{S} = \frac{1}{2}\left(\mathbf{A}^T + \mathbf{A}\right) \tag{5}$$

that is symmetric.

Recursively applying spectral clustering we create our taxonomy as shown in Algorithm 1. Note that for each split we set the representative of each node as

$$\hat{\mathbf{w}}_N = \frac{1}{|\mathcal{W}_N|}\sum_{i \in \mathcal{W}_N} \mathbf{w}_i. \tag{6}$$

An example of the representatives for the first split of the bicycle and bus classes is shown in Fig. 2, highlighting how the dominant views of the objects, frontal and sideways, are captured. Even though the proposed approach is feature independent in our experiments we employed Histogram of Oriented Gradients features (HOG) [7], as in the original Exemplar-SVM formulation. The representations of the centroids in Fig. 2 are done showing the HOG glyphs and the inverse Hoggle representation of [42].

Algorithm 1 Taxonomy Learning. We recursively apply spectral clustering obtaining a binary tree and keeping as representative for node n the mean hyperplane $\hat{\mathbf{w}}_n$.

```
FUNCTION splitSet($\mathcal{W}$)
Data: $\mathcal{W} = \{\mathbf{w}_i \ldots \mathbf{w}_M\}; S$
Result: subsets $\mathcal{W}_L, \mathcal{W}_R : \mathcal{W}_L \cap \mathcal{W}_R = \emptyset$;
        $\mathcal{W} = \mathcal{W}_L \cup \mathcal{W}_R$;
        subset representatives: $\hat{\mathbf{w}}_L$, $\hat{\mathbf{w}}_R$
if $|\mathcal{W}| > 1$ then
    $[\mathcal{W}_{\mathcal{L}}, \mathcal{W}_{\mathcal{R}}] \leftarrow$ spectralClustering($\mathbf{S}, 2$) ;
    $\hat{\mathbf{w}}_\mathbf{L} \leftarrow \frac{1}{|\mathcal{W}_L|} \sum_{i \in \mathcal{W}_L} \mathbf{w}_i$;
    $\hat{\mathbf{w}}_\mathbf{R} \leftarrow \frac{1}{|\mathcal{W}_R|} \sum_{i \in \mathcal{W}_R} \mathbf{w}_i$;
    splitSet($\mathcal{W}_L$);
    splitSet($\mathcal{W}_R$);
else
    leaf node reached;
end
```

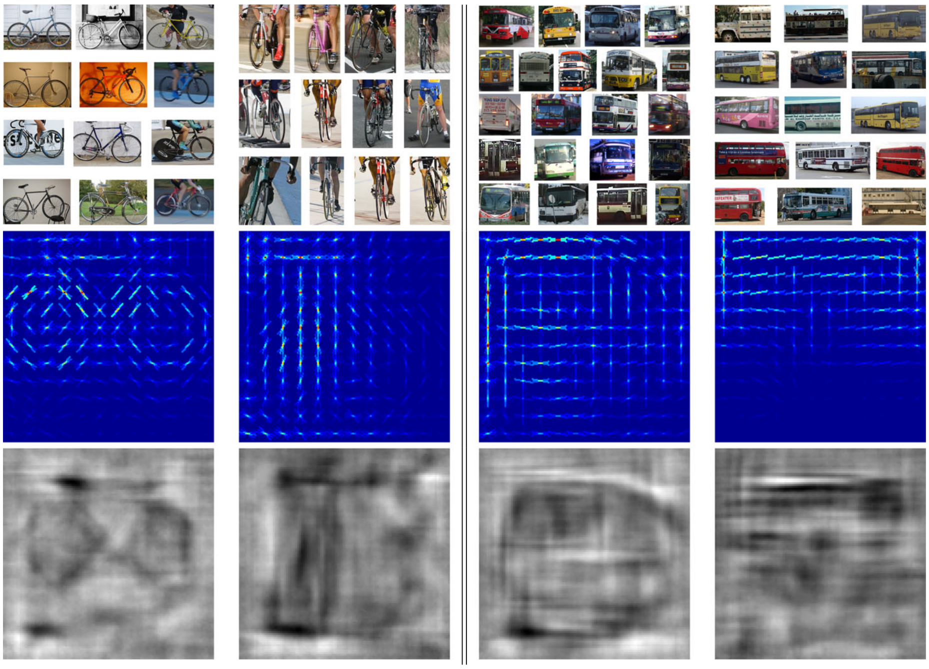

Fig. 2 Visualization of samples from the first two splits for the classes `bicycle` and `bus` of Pascal VOC 2007 [13] (*top*). From the HOG (*center*) and inverted HOG [42] (*bottom*) representations of their centroids can be clearly seen how the exemplars are indexed based on their viewpoint, clustering frontal and lateral views of the objects. Better viewed on computer

5 Accelerated ESVM evaluation

An ensemble of exemplars can be easily associated with a set $\mathcal{W}$ of exemplar hyperplanes $\mathbf{w}_i$ of the same class. Each image window feature vector $\mathbf{h}$ can be evaluated selecting the best exemplar using:

$$\arg\max_{i \in \mathcal{W}} \mathbf{w}_i^T \mathbf{h} \tag{7}$$

In order to do this each exemplar has to be tested against every test window. Thanks to our learned hierarchy we can reduce this operation to a tree traversal. The computation is performed by iteratively selecting from the current node, the child with the highest scoring representative:

$$\texttt{next\_node} = \arg\max_{i \in \{L,R\}} \hat{\mathbf{w}}_i^T \mathbf{h} \tag{8}$$

where L and R are the left and right children of the current node, respectively.

The scores

$$\hat{\mathbf{w}}_i^T \mathbf{h} = \frac{1}{|\mathcal{W}_N|} \sum_{i \in \mathcal{W}_N} \mathbf{w}_i^T \mathbf{h} \tag{9}$$

represent a lower bound on the score obtained by the best exemplar present in each sub-tree therefore we greedily pursue the path that maximizes this bound. This of course does not guarantee to select the leaf with the actual maximum. A schematic pipeline of our system is shown in Fig. 1.

5.1 Flipped exemplars

To improve the efficiency of the taxonomy, we also propose a small variant adding flipped copies of the exemplars to the ensemble. In fact a common trick to enhance detection accuracy, which is also used in the Exemplar-SVM formulation, is to evaluate both test images and their horizontally flipped copies. In this way the classifiers become more expressive, gaining a certain degree of invariance to viewpoint. This, in the case of a sliding window based detector, comes at the cost of doubling the number of windows that have to be evaluated.

With this in mind we evaluate windows only from the unflipped image and instead we double the number the classifiers, adding flipped exemplars. In the case of the standard ensemble there is no substantial difference between testing flipped windows or evaluating flipped models, but thanks to the logarithmic factor introduced by the tree, we are able to halve the number of windows without almost any additional cost. In fact if the tree is well balanced, doubling the number of exemplars only increases the depth of the tree by one.

To obtain the flipped version of each classifier we simply swap its components according to the orientations of the gradients in the correspondent HOG feature vector without any additional training. Figure 3 shows an example of flipped HOG template.

5.2 Early rejection

For each query image tens of thousands of windows have to be evaluated, making the evaluation phase very expensive. Inspired by the similarities with cascade classifiers by Bourdev and Brandt [5], we inserted in our framework a rejecting strategy by learning early rejection thresholds. The aim is to discard most of the windows at the higher levels of the tree if they do not look promising. This is done learning a threshold on the output score of each node, which makes the evaluation faster by drastically reducing the number of comparisons for each image. As shown in Fig. 4, after a few nodes most of the background windows are removed while windows that reach the leaves are densely clustered around the object to be detected.

We introduce an approach which is close to the philosophy of soft-cascades proposed in [5] to learn the threshold values. Soft-cascades are based on the usage of a rejection distribution vector $\mathbf{v} = (v_1, \dots v_T)$ where $v_t \geq 0$ is the minimum fraction of objects that we are allowed to miss at the t-th stage of the cascade. In our setting, instead of a cascade with T classifiers we have a tree with M leaves, i.e. $2M - 1$ classifiers. The aim is to learn a

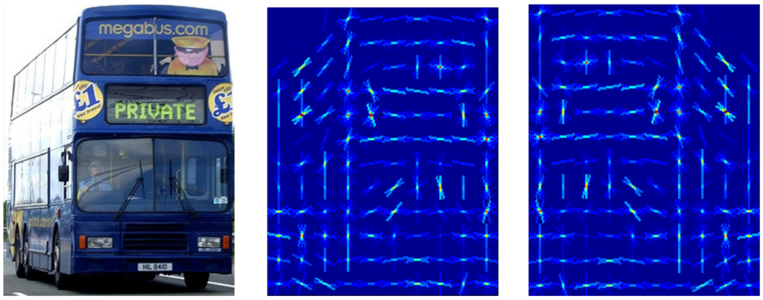

Fig. 3 Horizontal flipped HOG: From the original image (*left*) an exemplar HOG template is learnt (*center*) and then the flipped copy is obtained reorganizing the bins of the histogram (*right*)

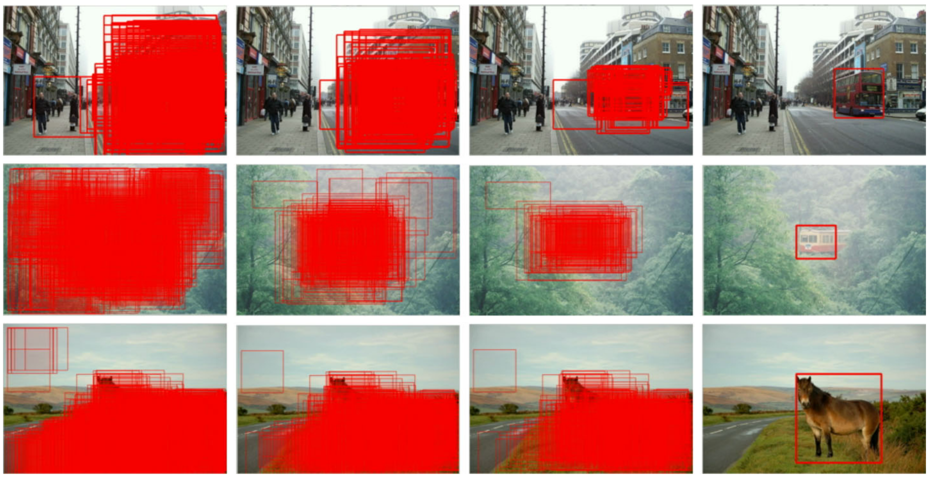
Fig. 4 Windows retained during tree traversal using rejection thresholds. The amount of windows decreases significantly from the first tree node (*leftmost*) to the last (*rightmost*)

rejection threshold for each node and to do so we group together nodes at the same depth, in order to establish the rejection distribution vector. Each path down the tree is treated as a soft-cascade. For a tree of depth D we employ a rejection distribution vector $\mathbf{v}$ with D values, one for each tree level, defined as

$$v_d = \exp \frac{1}{2} \left(\frac{d}{D} - 1 \right). \tag{10}$$

In our implementation this function defines the percentage of windows we want to keep at each level of our tree and has the property to saturate towards 1 descending towards a leaf. With this strategy we reject most of the least promising windows in the first levels, increasing the amount of kept windows at each stage.

Thresholds t_n for each node n are learned by evaluating approximately 2M windows drawn randomly from a validation set. Considering a set of windows H that reach node n we select the value that at this node allows $v_t \cdot |H|$ windows to be retained.

5.3 Quantized nodes

In literature the task of speeding-up the evaluation of an ensemble of Exemplar-SVMs has been tackled by Sadeghi et al. [35] exploiting Vector Quantization. We show that their approach is orthogonal to ours and we propose a vector quantized indexing taxonomy.

In our framework, image patches are represented as HOG feature vectors [7]. The feature vector is built by dividing the patch in $N_w \times N_h$ cells. For each cell a histogram of oriented gradients is computed over N_b bins. The resulting feature vector is composed by $N_w \times N_h$ arrays of N_b bins (we use $N_b = 31$ as in [16, 26, 35]). When searching for an object in an image we compare a template of such object category which in our case is a learned exemplar-SVM. Therefore, a comparison between a learned HOG template (e.g. an Exemplar-SVM) and a HOG feature vector will require $N_w \times N_h \times N_b$ comparisons.

To approximate this computation, Vector Quantization requires to learn a dictionary in order to represent HOG feature vectors. This can be done simply by clustering random HOG cells and collecting the resulting centroids. This allows us to represent each cell as a scalar instead of a 31-dimensional array by taking the index of the closest centroid.

A convenient and efficient strategy to accelerate computation is to pre-compute values and store them in a lookup table. For each classifier a lookup table is then built by computing partial scores between the cells of the template and all the centroids. These lookup tables are data structures which can be indexed with a centroid index and return the precomputed partial score with a given template cell. In this way, at test time the full dot product between a template and a detection window is approximated as a sum of partial scores by looking them up in the table.

We can integrate this technique in our framework by replacing the representative of each node in the taxonomy with a lookup table. In this way we are able to exploit the fast evaluation of single templates offered by Vector Quantization and to maintain the speed-up offered by the taxonomy.

Representing the tree as a lookup table $\mathbf{T}^{C \times M \times N}$, where C is the number of cells in a template, M the number of centroids used to quantize the models and N the number of nodes in the taxonomy, we can reformulate the node selection for traversing the tree from (8) as:

$$\texttt{next\_node} = \arg \max_{i \in \{L,R\}} \sum_{c=1}^{C} \mathbf{T}(c, \mathbf{h}(c), i) \tag{11}$$

where $\mathbf{h}(c)$ is the c–th cell of the current window $\mathbf{h}$ and L and R are the left and right children of the current node, respectively.

To ensure a fast evaluation of the lookup table we employ fixed point arithmetic for computing scores, as suggested in [35]. To do so we map the values in the lookup table as 16 bit integers in the range $\left[-\frac{(2^{15}-1)}{C}; \frac{(2^{15}-1)}{C}\right]$ in order to be able to sum up to C signed values without overflow or underflow. To reverse to the correct score after evaluation, we simply rescale the value to the original range.

The complexity of evaluating a template on a single window hence amounts just to C table lookups and C sums between 16 bit integers instead of a dot product between two high dimensional floating point vectors. As stated in [35], the dot product could be faster than a table lookup, depending on the CPU architecture and on how data is stored in memory. We therefore ensure that at training time the lookup table is packed in a contiguous block of memory, permitting the usage of fast SIMD instructions sets such as SSE (Streaming SIMD Extensions) and AVX (Advanced Vector Extension). Moreover, a feature quantization step has to be added to the evaluation pipeline. In fact each HOG cell in the feature vectors extracted from the test windows have to be associated with a centroid. However the cost of this operation is negligible compared to the time needed to evaluate the ensemble.

All the proposed variants of our method are compatible with the Vector Quantized Taxonomy. In particular, for the rejecting taxonomy of Section 5.2 we simply compute the rejection thresholds directly in the quantized domain.

5.4 Rescoring

After evaluating the hierachy we rescore the best 5% of the test windows with the whole ensemble, in order to gather more detections. This step is required since the ESVM framework boosts detected bounding boxes using an exemplar co-occurrence matrix. This boosting procedure is less effective with a reduced number of detections, as in the case of our taxonomy which associates only one detection for bounding box. The rescoring step allows us to overcome this problem by evaluating the entire ensemble only on a restricted subset of windows, which requires a negligible additional cost. Furthermore this helps to mitigate the approximation effect of Vector Quantization. In fact a similar approach is used in [35].

5.5 Complexity analysis of ESVM ensembles

To evaluate and ensemble of ESVMs, a set of N windows taken from an image has to be tested against a pool of M Exemplar classifiers. Therefore, the complexity of evaluating an ensemble is $O(NM)$. Our proposed algorithm instead, in the case of a balanced binary tree has a complexity of $O(N \log_2(M))$ since each window only has to traverse it from the root to a leaf.

The rescoring step (Section 5.4) has an impact on this cost which depends on the windows to be rescored in an image and the number of exemplars, so it is $O(N'M)$ where N' is the amount of windows to be rescored. The cost will increase linearly in the amount of rescored windows, since the number of exemplar remains fixed. Our method has a trade-off which is found when $\log_2(M)N + N'M \geq NM$, which is not dependent on N since $N' = pN$, with p the fraction of windows to rescore. Hence, we obtain that the rescoring becomes expensive when $\log_2(M) \geq (1 - p)M$. In case window rejection is applied a factor furtherly reducing the rescoring cost has to be considered. The above trade-off depends both on the amount of exemplars and on the percentage of rescored windows. It easily seen that for $p < 1$ there is always a value of M for which our method is faster than the ensemble. In practice our approach becomes inconvenient for small M (e.g less than 50) and high p (more than .15). Moreover the use of Vector Quantized Exemplars, effectively reduces the computational time required for each comparison.

6 Experimental results

We evaluate the proposed algorithm on the Pascal VOC 2007 object detection benchmark [13], comparing the results to the baselines given by the original ESVM framework [26] and its Vector Quantized version [35]. Besides accuracy we focus on the speed-up gain in the evaluation step. We also present results for a segmentation task based on label transfer, showing how our approach is able to provide comparable high quality masks with the monolithic ensemble formulation.

6.1 Object detection

Object detection accuracy is important in order to establish how the taxonomy is able to approximate the capabilities of a standard ESVM ensemble. To build the hierarchical ensembles we use the pre-trained exemplars provided by [26] for each of the Pascal `trainval` objects (20 categories, 12608 exemplars). Each class is evaluated using a different taxonomy, specific to its class.

We report the results obtained using four different approaches: the standard hierarchy built through plain spectral clustering (`Tree`), an augmented version created using both exemplars and their horizontally flipped copies (`Tree-flip`, see Section 5.1) and the rejecting version of the two previous strategies (`Tree-rej` and `Tree-flip-rej`, respectively. See Section 5.2).

The same experiments are then repeated combining these techniques with the Fast Template Vector Quantization method of [35], which to the best of our knowledge is the fastest Exemplar-SVM formulation in literature (see Section 5.3).

Table 2 summarizes the results for all of the proposed methods, along with the ESVM ensemble baseline (`ESVM`) and the Fast Template Vector Quantization [35] (`FTVQ`) method applied to Exemplar-SVM. To provide these baselines we used the Exemplar-SVM framework of [26] available online and our implementation of FTVQ. All experiments have been performed on an Intel Core i7-2600K, 4 x 3.40Ghz to provide a fair comparison of timings.

Detection accuracy is reported in terms of mean Average Precision (mAP) and the evaluation time represents the average time to evaluate an image with a single exemplar. This is a cost which is normalized both with the number of images to evaluate and with the number of classifiers in the ensemble. Average Precision is calculated by numerically integrating the area under the Precision-Recall curve for each class. This is the standard evaluation metric for object detection since Pascal VOC 2010 [14], which differs from the 11-point AP used previously and in particular in [26].

The `Tree` method obtains a mAP of 18.65, which is comparable to the ESVM ensemble baseline and is on average 10 times faster. The use of the flipped exemplars strategy instead, at almost no additional cost in mAP, lowers by 60% the overall evaluation time since we have to evaluate only the original unflipped windows, obtaining a 18× speed-up with respect to the ESVM baseline.

Employing rejecting thresholds we are able to reach a 25× and 38× speed up for the `Tree-rej` and the `Tree-rej-flip` methods, respectively. These two methods leave

Table 2 Results on the Pascal VOC 2007 dataset

Method	✈️	🚲	🐦	⛵	🍾	🚌	🚗	🐈	🪑	🐄	┳┳
ESVM [26]	**19.0**	**47.0**	3.0	**11.0**	9.0	39.0	40.0	2.0	6.0	15.0	**7.0**
ELDA[21]	18.4	39.9	**9.6**	10.0	**11.3**	39.6	**42.1**	**10.7**	6.1	12.1	3.0
FTVQ [35]	18.0	**47.0**	3.0	**11.0**	9.0	39.0	40.0	2.0	6.0	15.0	**7.0**
Tree-flip-rej	18.0	45.5	2.2	9.0	9.4	39.0	37.4	1.6	**6.2**	13.5	6.1
Tree-rej	13.1	46.6	1.9	10.0	8.0	39.7	37.8	1.2	5.9	**15.5**	5.1
Tree-flip	17.6	45.5	2.3	9.2	7.7	39.0	38.0	1.5	**6.2**	13.6	6.1
Tree	13.0	46.4	2.1	10.5	7.2	**40.0**	38.5	1.3	5.9	**15.5**	**7.0**

	🐕	🐎	🏍️	🚶	🪴	🐑	🛋️	🚆	📺	mAP	Time
ESVM [26]	2.0	**44.0**	**38.0**	13.0	**5.0**	**20.0**	**12.0**	**36.0**	28.0	**19.8**	6.48 ms
ELDA[21]	**10.6**	38.1	30.7	**18.2**	1.4	12.2	11.1	27.6	**30.2**	19.1	6.48 ms
FTVQ [35]	2.0	**44.0**	**38.0**	13.0	**5.0**	**20.0**	**12.0**	**36.0**	28.0	19.7	1.06 ms
Tree-flip-rej	1.2	41.9	37.8	8.8	3.0	17.1	10.5	31.1	27.2	18.3	**0.17** ms
Tree-rej	1.6	41.6	36.0	9.7	3.4	16.8	11.2	35.0	26.8	18.4	0.26 ms
Tree-flip	1.2	42.0	37.6	10.6	3.0	17.2	10.8	30.4	27.1	18.3	0.35 ms
Tree	1.5	42.0	37.5	11.7	3.3	17.7	10.9	34.0	27.0	18.7	0.59 ms

Comparison of our method with the Exemplar-SVM baseline [26]. The results obtained using Fast Template Vector Quantization [35] are also given as a term of comparison. The four variants of our approach (standard tree, tree with flip augmentation, rejecting tree, rejecting tree with flip augmentation) are reported, showing detection accuracy and timings. Running times refer to the mean time required for evaluating each exemplar on an image. Best results are reported in bold the mAP of the system almost unchanged with respect to the standard `Tree` algorithm, meaning that we are able to prune windows that do not contain any detectable object. All of the proposed variants of the method are faster than FTVQ.

To analyse the benefit of our approach in relation with the ensemble size, in Fig. 5 we report a comparison of timings between our methods and the `ESVM` baseline. Each point in the plot compares a timing obtained with the same number of exemplars with both methods. To perform these experiments, large ensemble have been syntetically generated in order to gather timings. The plot shows a clear logarithmic relationship between the timings of the ensemble and the `Tree` and `Tree-flip` methods. Using the rejecting taxonomy, the logarithmic trend is less evident since the number of comparisons is reduced, depending on the image content.

It is important to note that the speed-up increases as more models are used. For example, considering the `Tree` method, we have a 10× speed-up with 5k exemplars but we are able to obtain a 28× speed-up when 20k exemplars must be evaluated. The same behaviour is registered for the other variants reaching a 65× gain with `Tree-rej-flip` at 20k exemplars, going from more than 2 minutes to less than 2 seconds. In the experiments reported in Table 2 this speed-up is not appreciable since most of the classes in Pascal have

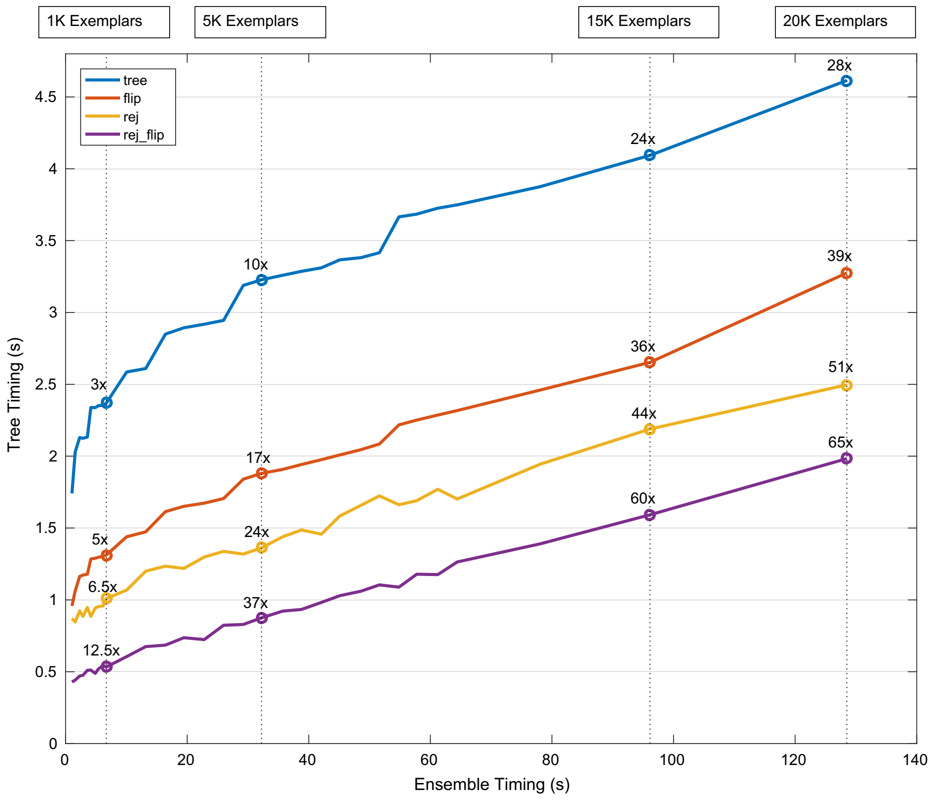

Fig. 5 Plot of timings varying the number of exemplars. The average execution time for evaluating an image of the four variants of our method (*Tree Timing*) is compared with the time required by the monolithic ensemble [26] (*Ensemble Timing*). A logarithmic trend is observed in relation to the number of exemplars. Speed-up values are reported, at the correspondent mark in the plot, for 1k, 5k, 15k and 20k exemplars

a small number of exemplars (200-300 on average) and we are averaging timings on all classes.

We perform the same analysis on our Vector Quantized variants of the taxonomies, comparing them to the fairer baseline of FTVQ [35]. In Table 3 the Object Detection results on Pascal 2007 are shown. Overall, the mAP of the method is slightly lower than the respective unquantized versions but we are able to improve the mean execution time to less than 0.1ms per exemplar per image for each variant. Note that on average all quantized tree methods have almost the same speed. This is caused by the feature quantization step which dominates over the evaluation phase. This reflects on the trends of the plots in Fig. 6. Here the speed of the vector quantized hierarchies is compared to the speed of FTVQ. Whereas all these techniques are much faster than `ESVM`, it can be seen how the benefit brought by the taxonomy is even bigger. With 20k exemplars we obtain from 39x speed-up with `VQ-Tree` to 95.5x with `VQ-Tree-rej-flip`. As stated before, the time required by the rejecting and flipped versions of the quantized tree are almost the same for a relatively small amount of exemplars due to a fixed time needed to quantize the features from the test windows.

To provide a final overview, in Fig. 7 we show the execution time varying the number of exemplars for all the proposed methods. As expected, the vector quantized taxonomies perform much better than their unquantized counterparts and the usage of rejection thresholds and flipped exemplars is crucial in ensuring an extremely fast evaluation of the method. On the plot are underlined the speed-ups obtained by the most performing variant of the algorithm: `VQ-Tree-rej-flip`. Evaluation speed with a large number of classifiers

Table 3 Results on the Pascal VOC 2007 dataset (quantized version)

Method											
ESVM [26]	**19.0**	**47.0**	3.0	**11.0**	9.0	39.0	40.0	2.0	6.0	15.0	**7.0**
ELDA [21]	18.4	39.9	**9.6**	10.0	**11.3**	39.6	**42.1**	**10.7**	6.1	12.1	3.0
FTVQ [35]	18.0	**47.0**	3.0	**11.0**	9.0	39.0	40.0	2.0	6.0	15.0	**7.0**
VQ-Flip-rej	17.7	45.0	2.2	9.0	7.3	34.9	36.7	1.2	5.3	14.8	4.6
VQ-Tree-Rej	14.8	46.3	1.8	10.0	7.9	**39.2**	37.5	1.6	5.5	15.3	3.6
VQ-Tree-Flip	17.1	44.8	2.2	9.1	7.1	34.7	37.0	1.3	5.2	14.5	4.4
VQ-Tree	15.0	46.0	2.0	9.7	8.0	**39.2**	37.4	1.7	5.6	**15.4**	4.7

										mAP	Time
ESVM [26]	2.0	**44.0**	**38.0**	13.0	**5.0**	**20.0**	**12.0**	**36.0**	28.0	**19.8**	6.48 ms
ELDA [21]	**10.6**	38.1	30.7	**18.2**	1.4	12.2	11.1	27.6	**30.2**	19.1	6.48 ms
FTVQ [35]	2.0	**44.0**	**38.0**	13.0	**5.0**	**20.0**	**12.0**	**36.0**	28.0	19.7	1.06 ms
VQ-Flip-rej	1.2	40.8	36.8	10.5	2.7	16.8	10.9	30.0	26.8	17.8	**0.04** ms
VQ-Tree-Rej	1.5	41.5	36.9	10.6	3.4	16.6	11.2	34.5	27.8	18.4	0.05 ms
VQ-Tree-Flip	1.3	41.1	36.6	10.4	2.7	16.7	10.2	29.9	26.9	17.7	0.05 ms
VQ-Tree	1.6	41.3	36.6	11.0	3.3	16.5	11.5	34.9	**28.0**	18.5	0.08 ms

Comparison of the quantized version of our method with the Exemplar-SVM baseline [26]. The results obtained using the Fast Template Vector Quantization [35] are also given as a term of comparison. The four variants of our Vector Quantized Taxonomies (standard tree, tree with flip augmentation, rejecting tree, rejecting tree with flip augmentation) are reported, showing detection accuracy and timings. Running times refer to the mean time required for evaluating each exemplar on an image. Best results are reported in bold

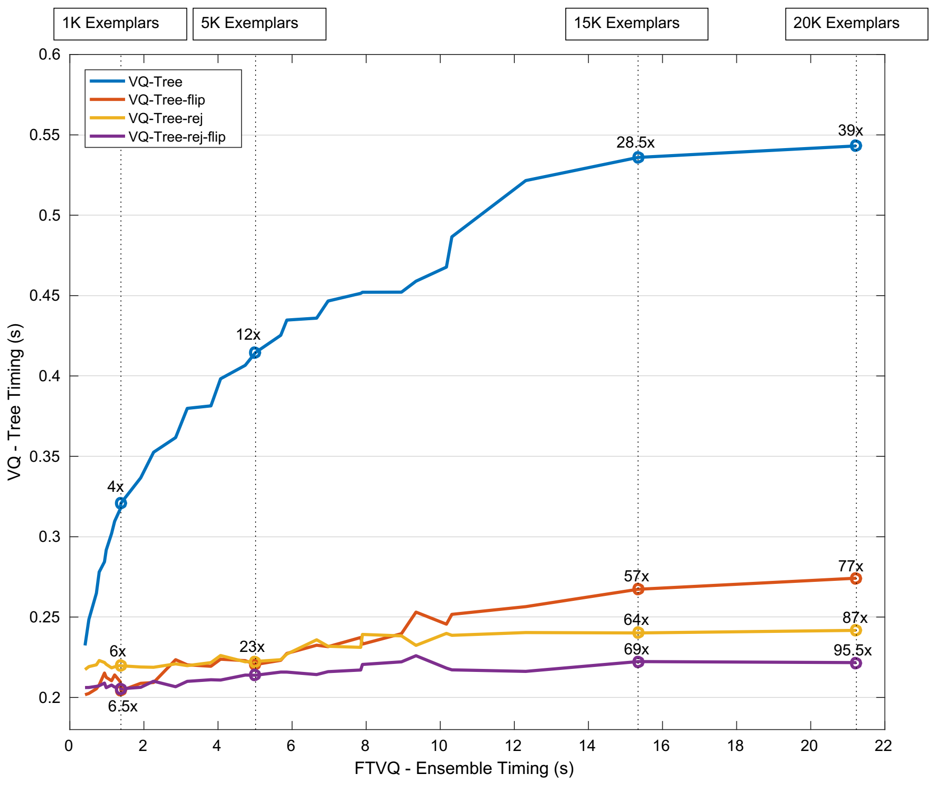

Fig. 6 Plot of timings varying the number of exemplars applying Vector Quantization. The execution time of the four variants of our Vector Quantized Taxonomies are compared with the time for evaluating the monolithic FTVQ ensemble [35]. Speed-up values, at the correspondent mark in the plot, are reported for 1k, 5k, 15k and 20k exemplars (overlapping points are omitted). A logarithmic trend is observed in relation to the number of exemplars with the `VQ-Tree` variant while the others are domniated by the linear quantization cost

is a couple of orders of magnitude faster than `ESVM`, lowering the required time from approximately 2 minutes to half a second for 20k exemplars with more than a 500-fold gain.

We analyze the rescoring effect for the `Tree` method varying the percentage of rescored windows. As can be seen in Fig. 8 using less than 5% of the windows affects the mAP by almost 2 points. Increasing the fraction of windows to be rescored with the whole ensemble the mAP obviously saturates to the baseline value. Note that rescoring the 5% of the windows has a cost of 0.1 ms per exemplar. This operation becomes expensive above 15% of the windows.

6.2 Label transfer segmentation

Along with object detection accuracy we evaluated the label transfer capabilities of the system performing a segmentation experiment. In order to do so, we manually annotated the Pascal VOC *Bus* class (229 exemplars and 213 test objects for a total of 442 buses) with segmentation masks. Given a set of detection stemmed from a bank of exemplars the following procedure can be applied to generate segmentation masks. First we weight all transferred

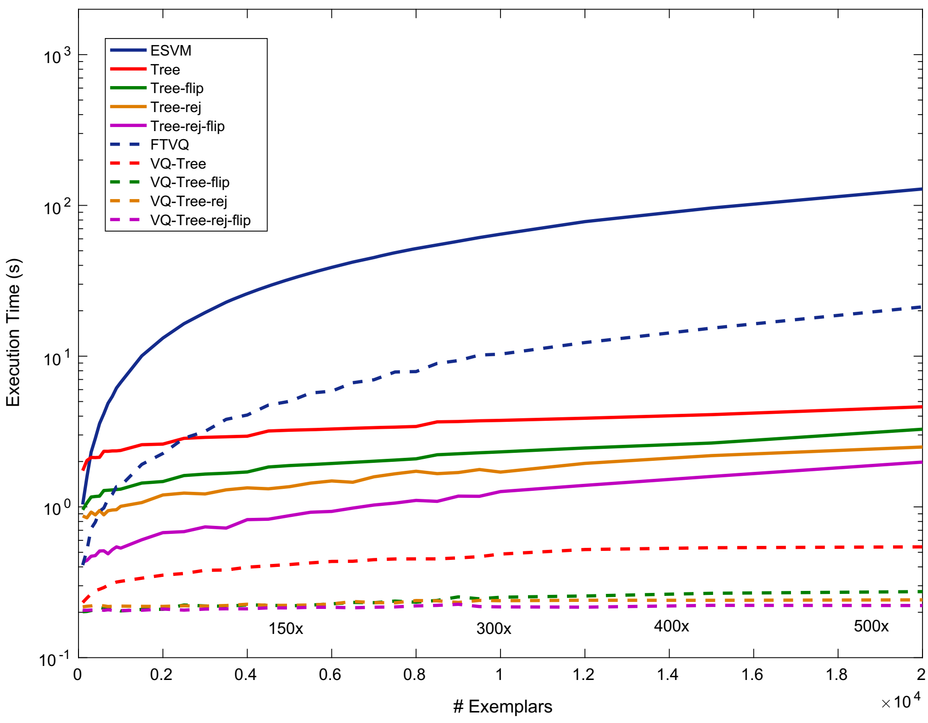

Fig. 7 Comparison of execution times for all the presented methods against the ESVM [26] and FTVQ [35] baselines varying the number of exemplars in the ensemble. Vector Quantized taxonomies are reported as dashed lines while continuous lines are the respective unquantized formulations. Speed-ups are reported for our fastest method (`VQ-Tree-rej-flip`)

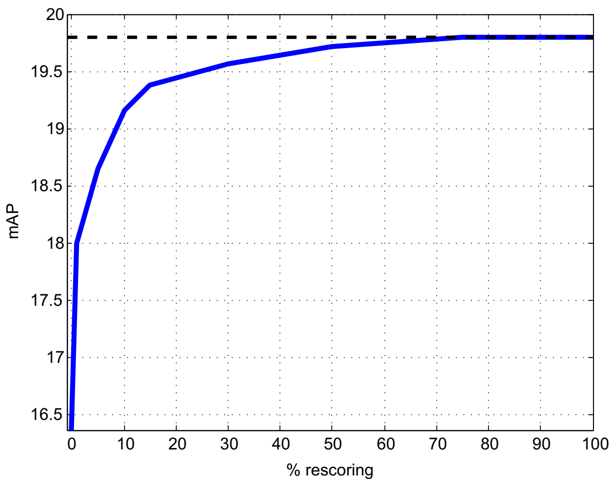

Fig. 8 Mean average precision varying the percentage of rescored windows. Rescoring the whole window set is equivalent to evalute the full Ensemble. We report mAP without rescoring as 0%

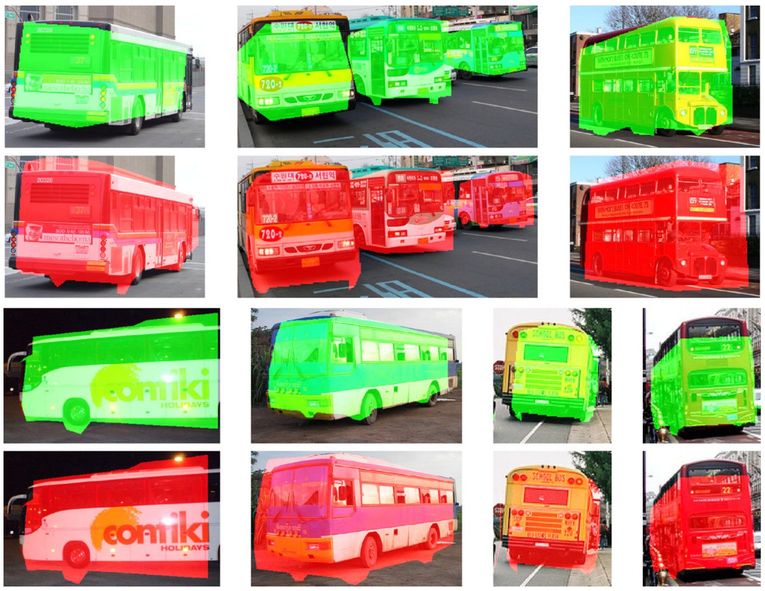

Fig. 9 Segmentation masks produced by Exemplar-SVM label transfer. *Green masks* in the upper row are produced using the standard ensemble [26], *red masks* in the lower row are generated using our hierarchical ensemble

masks according to the exemplar detection confidence and accumulate such weighted masks into a segmentation map. To remove noise, we rescale the mask into $[-1, 1]$ and threshold it. Mask rescaling is necessary on one hand to avoid misses in case few exemplars have positive score, on the other hand to avoid over segmentations when many exemplars are matched. Threshold is cross-validated on the validation set. As baseline we use the standard monolithic ensemble [26] and compare it to `VQ-Tree-rej-flip`. The segmentation masks have been evaluated for both the *Bus* class and the *Background* class, measuring a pixel-wise accuracy

$$Acc = \frac{tp}{tp + fp + fn} \tag{12}$$

as specified in the Pascal guidelines for segmentation. Using the hierarchical structure we obtained a 48.07% accuracy for the *Bus* class and 76.30% for the *Background* against a baseline of 49.59% and 77.14%, respectively. Some qualitative results are reported in Fig. 9 where the segmentation masks generated by our method and by the baseline are compared. These results show that with our indexing strategy we are still able to retrieve objects which maintain a good alignment with test objects, leaving the label transfer capabilities of the framework almost unaffected.

7 Conclusion

In this work we have proposed a technique to overcome the main drawback of the Exemplar-SVM framework, i.e. the high computational burdain at test time. In fact, even if it has found large interest for many computer vision tasks, the real applicability of this technique is limited due to the linear dependency in the number of examples. We solve this issue by building a quantized indexing taxonomy to access data at a logarithmic cost and just a small loss in detection performance. We have shown how combining different strategies to speed-up the evaluation we are able to provide a reliable approximation of a ESVM ensemble with a very small computational footprint, which is orders of magnitude faster than the original formulation. Moreover we have shown how our indexing impacts the label transfer property of the method, applying the technique to a segmentation task. Confirming the previous experiments, the results are satisfactory providing high quality masks compared to the baseline.

Acknowledgements Federico Becattini and Lorenzo Seidenari are partially supported by "THE SOCIAL MUSEUM AND SMART TOURISM", MIUR project no. CTN01_00034_23154_SMST.

References

1. Aubry M, Maturana D, Efros AA, Russel BC, Sivic J (2014) Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In: Proc. of CVPR
2. Aytar Y, Zisserman A (2012) Enhancing exemplar svms using part level transfer regularization. In: BMVC, pp 1–11
3. Becattini F, Seidenari L, Del Bimbo A (2016) Indexing ensembles of exemplar-svms with rejecting taxonomies. In: 2016 14th International workshop on content-based multimedia indexing (CBMI). IEEE, pp 1–6
4. Bengio S, Weston J, Grangier D (2010) Label embedding trees for large multi-class tasks. In: Proc. of NIPS
5. Bourdev L, Brandt J (2005) Robust object detection via soft cascade. In: Proc. of CVPR
6. Chen Y, MC, Ghosh J (2004) Integrating support vector machines in a hierarchical output space decomposition framework. In: Proc. of IGARSS
7. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: Proc. of CVPR, pp 886–893
8. Dean T, Ruzon M, Segal M, Shlens J, Vijayanarasimhan S, Yagnik J (2013) Fast, accurate detection of 100,000 object classes on a single machine. In: Proc. of CVPR. Washington, DC
9. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: IEEE Conference on computer vision and pattern recognition, 2009. CVPR 2009. IEEE, pp 248–255
10. Deng J, Ding N, Jia Y, Frome A, Murphy K, Bengio S, Li Y, Neven H, Adam H (2014) Large-scale object classification using label relation graphs. In: European conference on computer vision. Springer, pp 48–64
11. Deng J, Satheesh S, Berg AC, Fei-Fei L (2011) Fast and balanced: efficient label tree learning for large scale object recognition. In: Proc. of NIPS
12. Dubout C, Fleuret F (2012) Exact acceleration of linear object detectors. In: Proc. of ECCV, pp 301–311
13. Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. Int J Comput Vis 88(2):303–338
14. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A (2010) The PASCAL visual object classes challenge (VOC2010) results
15. Fan J, Zhou N, Peng J, Gao L (2015) Hierarchical learning of tree classifiers for large-scale plant species identification. IEEE Trans Image Process 24(11):4172–4184
16. Felzenszwalb PF, Girshick RB, McAllester D, Ramanan D (2010) Object detection with discriminatively trained part-based models. IEEE Pattern Anal Mach Intell. 32(9)
17. Gao T, Koller D (2011) Discriminative learning of relaxed hierarchy for large-scale visual recognition. In: Proc. of ICCV
18. Griffin G, Perona P (2008) Learning and using taxonomies for fast visual categorization. In: IEEE Conference on computer vision and pattern recognition, 2008. CVPR 2008. IEEE, pp 1–8
19. Gronat P, Obozinski G, Sivic J, Pajdla T (2013) Learning and calibrating per-location classifiers for visual place recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 907–914

20. Guillaumin M, Ferrari V (2012) Large-scale knowledge transfer for object localization in imagenet. In: 2012 IEEE Conference on computer vision and pattern recognition (CVPR). IEEE, pp 3202–3209
21. Hariharan B, Malik J, Ramanan D (2012) Discriminative decorrelation for clustering and classification. In: Computer vision–ECCV 2012. Springer, pp 459–472
22. Jain A, Gupta A, Rodriguez M, Davis LS (2013) Representing videos using mid-level discriminative patches. In: Proc. of CVPR
23. Kobayashi T (2015) Three viewpoints toward exemplar svm. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2765–2773
24. Kuettel D, Guillaumin M, Ferrari V (2012) Segmentation propagation in imagenet. In: European conference on computer vision. Springer, pp 459–473
25. Liu B, Sadeghi F, Tappen M, Shamir O, Liu C (2013) Probabilistic label trees for efficient large scale image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 843–850
26. Malisiewicz T, Gupta A, Efros AA (2011) Ensemble of exemplar-svms for object detection and beyond. In: Proc. of ICCV
27. Marszałek M, Schmid C (2008) Constructing category hierarchies for visual recognition. In: Proc. of ECCV. Springer
28. Miller GA (1995) Wordnet: a lexical database for english. Commun ACM 38(11):39–41
29. Modolo D, Vezhnevets A, Ferrari V (2015) Context forest for object class detection. Arxiv preprint
30. Modolo D, Vezhnevets A, Russakovsky O, Ferrari V (2015) Joint calibration of ensemble of exemplar svms. In: 2015 IEEE Conference on computer vision and pattern recognition (CVPR). IEEE, pp 3955–3963
31. Mrowca D, Rohrbach M, Hoffman J, Hu R, Saenko K, Darrell T (2015) Spatial semantic regularisation for large scale object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2003–2011
32. Ng AY, Jordan MI, Weiss Y (2002) On spectral clustering: analysis and an algorithm. In: Proc. of NIPS
33. Ristin M, Guillaumin M, Gall J, Gool L (2014) Incremental learning of ncm forests for large-scale image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3654–3661
34. Rohrbach M, Ebert S, Schiele B (2013) Transfer learning in a transductive setting. In: Advances in neural information processing systems, pp 46–54
35. Sadeghi MA, Forsyth D (2013) Fast template evaluation with vector quantization. In: Proc. of NIPS, pp 2949–2957
36. Sadeghi M, Forsyth D (2014) 30hz object detection with dpm v5. In: Proc. of ECCV, lecture notes in computer science, vol 8689. Springer International Publishing, pp 65–79
37. Singh S, Gupta A, Efros AA (2012) Unsupervised discovery of mid-level discriminative patches. In: Proc. of ECCV
38. Spielmat DA, Teng SH (1996) Spectral partitioning works: planar graphs and finite element meshes. In: 37th Annual symposium on foundations of computer science, 1996. Proceedings. IEEE, pp 96–105
39. Tighe J, Lazebnik S (2013) Finding things: image parsing with regions and per-exemplar detectors. In: Proc. of CVPR
40. Viola P, Jones MJ (2004) Robust real-time face detection. Int J Comput Vis 57(2):137–154
41. Von Luxburg U (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416
42. Vondrick C, Khosla A, Malisiewicz T, Torralba A (2013) Hoggles: visualizing object detection features. In: Proc. of ICCV
43. Yao B, Khosla A, Fei-Fei L (2011) Combining randomization and discrimination for fine-grained image categorization. In: 2011 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 1577–1584
44. Zepeda J, Perez P (2015) Exemplar svms as visual feature encoders. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3052–3060

Federico Becattini is a Ph.D. student at the Dipartimento di Ingegneria dell'Informazione at the University of Florence. He received the MS degree in computer engineering from the University of Florence, Italy, in 2014 with a thesis titled "Accelerating Exemplar-SVM label transfer". Currently he is working as a PhD student at the Media Integration and Communication Center (MICC) at the University of Florence, under the supervision of Prof. Alberto Del Bimbo, Ing. Lorenzo Seidenari and Prof. Marco Bertini. His main research interest is on efficient machine learning techniques for object detection and semantic annotation, focusing on instance based object detectors.

Lorenzo Seidenari is a Postdoctoral researcher at the Media Integration and Communication Center of the University of Florence. He received his Ph.D. degree in computer engineering in 2012 from the University of Florence. His research focuses on object and action recognition in video and images. On this topics he addressed RGB-D activity recognition, embedding learning for multimodal-fusion, anomaly detection in video and people behavior profiling. He was a visiting scholar at the University of Michigan in 2013. He organized and gave a tutorial at ICPR 2012 on image categorization. He is author of 9 journal papers and more than 20 peer-reviewed conference papers.

Alberto Del Bimbo is a Full Professor of Computer Engineering, and the Director of the Media Integration and Communication Center with the University of Florence. His scientific interests are multimedia information retrieval, pattern recognition, image and video analysis, and natural human–computer interaction. From 1996 to 2000, he was the President of the IAPR Italian Chapter and the Member-at-Large of the IEEE Publication Board from 1998 to 2000. He was the General Co-Chair of ACM Multimedia 2010 and the European Conference on Computer Vision in 2012. He was nominated as ACM Distinguished Scientist in 2016. He received the SIGMM Technical Achievement Award for Outstanding Technical Contributions to Multimedia Computing, Communications and Applications. He is an IAPR Fellow, and an Associate Editor of Multimedia Tools and Applications, Pattern Analysis and Applications, the Journal of Visual Languages and Computing, and the International Journal of Image and Video Processing, and was an Associate Editor of Pattern Recognition, the IEEE Transactions on Multimedia, and the IEEE Transactions on Pattern Analysis and Machine Intelligence. He serves as the Editor-in-Chief of the ACM Transactions on Multimedia Computing, Communications, and Applications.