*19th Computer Vision Winter Workshop*
*Zuzana Kúkelová and Jan Heller (eds.)*
*Křtiny, Czech Republic, February 3–5, 2014*

# Using discriminative analysis for improving hierarchical compositional models

Domen Tabernik[1], Matej Kristan[1], Marko Boben[1], and Aleš Leonardis[1,2]

[1]Faculty of Computer and Information Science, University of Ljubljana, Slovenia
[2]CN-CR Centre, School of Computer Science, University of Birmingham

{domen.tabernik,matej.kristan,marko.boben,ales.leonardis}@fri.uni-lj.si

**Abstract** *In this paper we propose a method to extract discriminative information from a generative model produced by a compositional hierarchical approach. We present discriminative information as a score computed from a weighted summation of the activation vector. We base the activation vector on individual activations of features from a parse tree of the detection. We utilize the score to reduce false positive detections by removing generative models with poor discriminative information from the vocabulary and by thresholding the detections with low discriminative score. We evaluate our approach on the ETHZ Shape Classes database where we show a reduction in the number of false positives and a decrees in detection time without reducing the detection rate.*

## 1 Introduction

In the field of the visual object class detection, many complex descriptors have been proposed that produce excellent results, e.g. [12], but such descriptors are computationally expensive. This problem becomes even more apparent in the combination with sliding windows therefore additional preprocessing steps are required to eliminate as many irrelevant windows as possible before applying complex descriptors. Alexe et. al. have addressed this issue to some extent but there are other alternatives such as hierarchical methods [13, 9, 4, 5, 8] with their generative models that do not require any preprocessing at all. In this case, each visual category is represented by multiple object models that cover visual variability within a category. An object model is a composition, consisting of increasingly complex features, that captures the complexity of objects through hierarchical arrangement of features. Due to hierarchical nature of object models each detection is a parse tree of features. One benefit of hierarchical arrangement is shareability of simple features across different visual categories. Common and simpler features are formed in the lower layers of the hierarchy and can then be used by the higher layers as needed for specific categories. This allows for more compact representation of objects on the one hand and on the other hand brings efficient scalability when introducing more categories, since detected simple features for one category can immediately be re-used by any other category.

Having shearability is one important benefit that in some hierarchical approaches originates from generativeness of models. However, the generative nature of construction leads to models that tend to overgeneralize and may not capitalize on discriminative information. For instance, a generative model of a horse must capture different variability of a horse seen from different viewpoints. We may be able to capture this by using multiple object models for different viewpoints, but models will still have to capture variability among different configurations of head and legs positions, while at the same time encompassing the variability between different breeds of horses. Generative models will be able to capture such variations but they will also capture representation of many other categories, especially visually similar ones. This can quickly lead to over-generalization. In the case of a horse the model will frequently misidentify a cow for a horse as both categories are similar. Additionally, we have also observed that some object models frequently hallucinate on the background objects that have low visual similarities with the detected object. The effect is most prominent on highly textured backgrounds. This allows a generative model to find enough features for a (false) positive match but at the same time the model does not capitalize on the discriminative features present in the representation that could point to an absence of detected object. Not considering such discriminative information in the end introduces noise into the detection process and can significantly reduce performance of the detector.

This problem was partially addressed in our previous work. In [10], we introduced a global HoC descriptor and included the discrimination of categories through non-linear support vector machine. As shown in [11], using such an approach as an additional hypothesis verification step applied after the detection stage, produces excellent results in the *ETHZ Shape Classes* dataset but adding non-linear decision boundary breaks the hierarchical approach and introduces computational complexity due to expensive computation of non-linear support vectors.

In [6], we eliminated the need for a non-linear SVM which allowed for principled integration into the concept of hierarchies. Using a sparse logistic regression we searched for weights to differentiate between two categories and used a score from weights to re-score the final detection. But the method still relied upon a global descriptor computed from histograms of activations of parse within detected bounding box. This constrained us to a discriminative information of a whole category and prevented us from obtaining discriminative information for specific object model. At the same

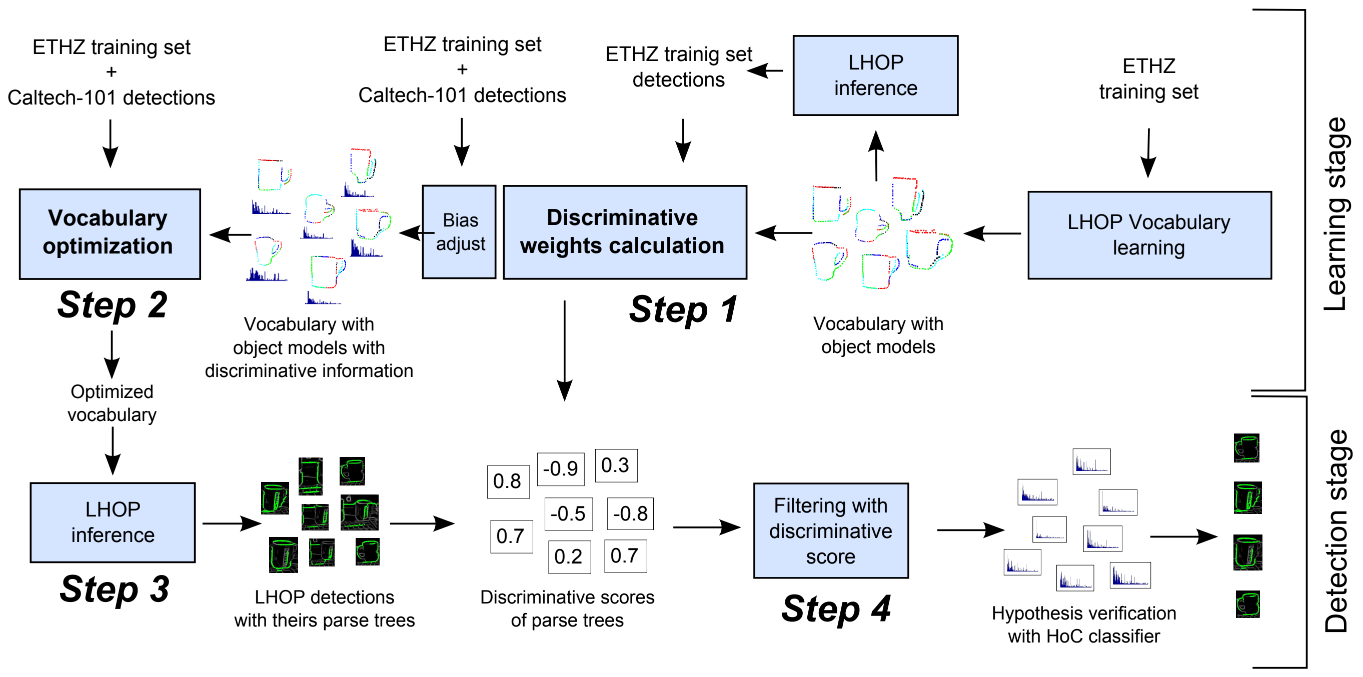

ETHZ training set + Caltech-101 detections
Vocabulary optimization
Step 2
Optimized vocabulary
Vocabulary with object models with discriminative information
ETHZ training set + Caltech-101 detections
Bias adjust
ETHZ trainig set detections
LHOP inference
Discriminative weights calculation
Step 1
ETHZ training set
LHOP Vocabulary learning
Vocabulary with object models
Learning stage
LHOP inference
Step 3
LHOP detections with theirs parse trees
0.8
-0.9
0.3
-0.5
-0.8
0.7
0.2
0.7
Discriminative scores of parse trees
Filtering with discriminative score
Step 4
Hypothesis verification with HoC classifier
Detection stage

Figure 1: Overview of our method in learning and detection stage. In step 1 we extract discriminative information in form of weights, in step 2 we optimize vocabulary by removing object parts with poor discriminative information, in step 3 we produce detections with optimized vocabulary and in step 4 we filter out detections with low discriminative score.

time global descriptor may have also removed parts of spatial information important for the discrimination.

In this paper, we focus on finding discriminative information within a category by analyzing each object model. Our contribution can be divided into two parts: (i) introduction of discriminative information extracted from activation vectors of detection parse tree and (ii) vocabulary optimization of the object models. We introduce discriminative information in form of linear combination of weights applied to the activation vector of the parse tree. Weights are computed for each object model separately which slows to fully capture spatial information of the activated features of the parse tree and get better precision in identifying features that are responsible for the discrimination. With the vocabulary optimization we eliminate object models that have poor discriminative capabilities and often activate on false positive detections. As a last step we also perform hypothesis verification with the HoC descriptor to re-score detections.

The benefits of our approach are two-fold. We eliminate many false positive detections through vocabulary optimization by removing non-discriminative object models. False positives are also removed by thresholding the weighted combination of parse tree response values and removing detections with low discriminative score. The second benefit is detection speedup that comes with the reduced number of detections. This translates into less computation time spent in the hypothesis verification step with the non-linear HoC classifier and less computation time in the inference stage. Additionally, having less positive detections also slightly increases the final detection rate. As in both [11] and [6] we apply our solution to the learnt-hierarchy-of-parts (LHOP) model [4].

The remainder of the paper is structured as follows. In Section 2 we introduce discriminative node of whole parse tree and provide further details of how we integrate discriminative information into vocabulary optimization. In Section 3 we present our evaluation procedure with final results and we conclude with the discussion in Section 4.

## 2 LHOP with discriminative information

We introduce discriminative information and vocabulary optimization into the learnt-hierarchy-of-parts process as shown in Figure 1. In the learning stage we start with the learned vocabulary containing object models and extract detections of object models in a form of parse tree. For each object model we learn linear classifier on positive and negative activation vectors of the detections and obtain weights used in the filtering stage of the detection. This process is detailed in Section 2.2. The optimization process, detailed in Section 2.3, is then applied to the vocabulary containing object models with their corresponding weights to obtain a smaller vocabulary with removed object models that produce too many false detections.

The detection stage starts with inference of the image with the optimized vocabulary. This produces a set of detections for each image together with its parse tree containing activation parts. Based on detected object model we apply appropriate weights to the activation vector of each detection and calculate its discriminative score. We perform filtering based on the score and remove as many false positive detections as possible. Finally, we calculate HoC descriptor from each detection and compute its final score by applying HoC classifier to it.

## 2.1 Learnt hierarchy of parts

We first provide a notation for LHOP model and refer the reader to [4] for further details. In the following we will denote the vocabulary of hierarchical parts trained for up to $L$ layers as a set of $N$ compositions $\mathcal{L} = \{P_i^l\}_{i=1:N}$ , where $P_i^l$ is an identifier of $i$-th composition and belongs to the $l$-th layer of the vocabulary. At the last layer $L$ each composition directly identifies one trained category, i.e. for each category we have only one corresponding composition on the $L$-th layer. We also define a set of $Links(P_i^l)$ which holds a list of linked sub-compositions on $l-1$ layer for the $P_i^l$ part:

$$Links(P_i^l) = \{(ind_j, P_j^{l-1})\}_{j=1..num\_subparts(P_i^l)},$$

where $P_j^{l-1}$ is the linked sub-composition type and $ind_j$ is the local index number for this sub-compositional ($ind_j$ goes through 0 and $num\_subparts(P_i^l) - 1$). For category layer parts ($L$-th layer), this subset holds object parts $(ind_o, P_o^{L-1}) \in Links(P_i^L)$ where each object part models a different view or an object of a category associated with the $P_i^L$ part.

Applying the vocabulary $\mathcal{L}$ on a given image $\mathcal{I}$, the algorithm of hierarchical models infers a set of $K$ detected parts, $\mathcal{C}(\mathcal{I}, \mathcal{L})$,

$$\mathcal{C}(\mathcal{I}, \mathcal{L}) = \{\pi_k^l\}_{k=1:K},$$

where the $k$-th detected part on the $l$-th layer $\pi_k^l = [P_k, \mathbf{c}_{\pi_k}, \lambda_k]$ is defined by its vocabulary identifier $P_k^l$, its location $\mathbf{c}_{\pi_k}$ in the image and its response values $\lambda_k$. All the inferred parts of the last layer $L$ directly correspond to detected objects in the image:

$$\mathcal{D}(\mathcal{I}, \mathcal{L}) = \{\pi_j^L\}_{j=1:J},$$

where $\mathcal{D}(\mathcal{I}, \mathcal{L})$ is a set of $J$ detected objects in the image $\mathcal{I}$ processed with the vocabulary $\mathcal{L}$. We also define a set of links $\Lambda(\pi_k^l)$ with a list of subparts for $\pi_k^l$ defined on previous layer $l-1$:

$$\Lambda(\pi_k^l) = \{(off_p, ind_p, \pi_p^{l-1}))\}_{P=1..num\_subparts(\pi_k^l)},$$

where $\pi_p^{l-1}$ is the linked subpart, $off_p$ is offset location $(x, y)$ relative to $\pi_k^l$ and $ind_p$ is the index matching to the corresponding sub-composition in $Links(P_p^l)$. For a detected category this always corresponds to one subpart representing detected object part $(off_p, ind_p, \pi_p^{L-1}) = \Lambda(\pi_k^L)$. Note, that we can have multiple detections of the same category in an image but they will have different location $\mathbf{c}_{\pi_k}$.

While each detected category part is defined the same as detected part at $L$-th layer $\pi_j^L = [P_j^L, \mathbf{c}_{\pi_j}, \lambda_j]$, we can also add a category information since a vocabulary identifier $P_j^L$ from the $L$-th layer always directly matches to one learning category. We can use $\Lambda(\pi_k^L)$ to recursively obtain list of all subparts for $\pi_k^L$ on all layers. List of all subparts obtained this way is called *an inferred parse tree*, while using the $Links(P_i^l)$ in the same recursive process would yield a list of all recursively traced compositions of vocabulary called *a vocabulary parse tree.* Minimal and maximal locations of all traced sub-parts in inferred parse tree define a bounding box of detected image. We can therefore define a set of detected objects from a given image $\mathcal{I}$ as:

$$\mathcal{D} = \{(\pi_j^L, c_j, r_j)\}_{j=1:J},$$

where $c_j$ is detected category and $r_j = (x, y, w, h)$ is a detection bounding box.

## 2.2 Discriminative interpretation of parse tree

We perform analysis on a set of training images $\mathcal{I} \in training\_set$ inferred with the initial vocabulary $\mathcal{L}$. This gives us a set of detections $\mathcal{D}(\mathcal{I}, \mathcal{L})$ with their corresponding parse trees for each image $\mathcal{I}$. We focus on analyzing each object model $P_o^{L-1}$ individually and extract only detections for that specific model:

$$\mathcal{D}_{P_o^{L-1}} = \{\ (\pi_j^L, c_j, r_j) \quad | \quad [P_o^{L-1}, \mathbf{c}_{\pi_o}, \lambda_o] = \Lambda(\pi_j^L)\ \}.$$

From each parse tree of detection $\pi_j^L \in \mathcal{D}_{P_o^{L-1}}$ we construct a discriminative descriptor $h_j$. We define $h_j$ as activation vector where components of the vector correspond to the activated parts in the parse tree. We collect all activation parts from an inferred parse tree and create activation vector using parts response values. Additionally, we also include relative offset location of each activated part relative to its parent into the activation vector. Based on discriminative descriptor $h_j$ of size $G$ we calculate a discriminative score using a weighted summation:

$$f(h_j; \Theta_o) = \sum_{g=1}^{G} \theta_o^{(g)} h_j^{(g)} + \theta_o^{(0)},$$

where $\Theta_o = [\theta_o^{(0)}, ..., \theta_o^{(G)}]$ is a vector of weights defining linear hyperplane between model object $P_o^{L-1}$ and a background clutter or any other model. Calculated discriminative score $f(h_j; \Theta_o)$ is later used in filtering process during the detection stage to retain only detections with high score.

We estimate the weights $\Theta_o$ via a linear Support Vector Machine from positive and negative detections of object model $P_o^{L-1}$. Additionally, we adjust bias by changing $\theta_o^{(0)}$ to eliminate as many potential false detections in the filtering stage as possible.

## 2.3 Vocabulary optimization

We start the vocabulary optimization process by assessing how well each object model $P_o^{L-1}$ performs on our training dataset. Using discriminative descriptor $h_j$ we obtain a discriminative scoring from a parse tree of all detections of object model $P_o^{L-1}$ and evaluate it according to its groundtruth data. We remove an object model if it does not contribute to the detection rate and produce mostly false positive detections. The optimization process tries to minimize the number of object models while retaining as high score as possible. The optimization is performed using a greedy approach with the following procedure: we start with an empty vocabulary and simulate individually adding every object model $P_o^{L-1}$ to the current vocabulary separately and calculate a performance score with added candidate model. Performance score is measured as an area under the ROC curve

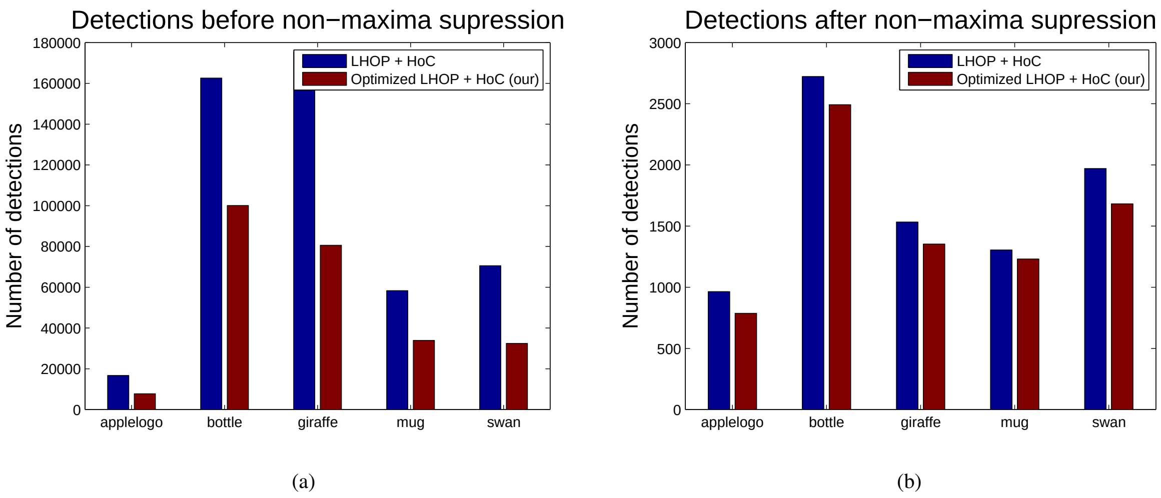


Figure 2: Number of detections shown for **original vocabulary (blue)** and **our method (red)** with left figure (2a) showing detections before any non-maxima suppression and right figure (2b) showing detections after non-maxima suppression. We used HoC scoring in non-maxima suppression.

score (AUC) over all training images but only counting detections from current vocabulary objects plus the current candidate object model. Out of all possible candidates we add only the model which improves performance score the most and repeat the procedure until performance cannot be improved any more. Using this iterative process we produce optimized vocabulary $\mathcal{L}_{opt}$. With current optimization we only remove parts from the $L-1$ layer and fix $L$-th layer to account for now missing parts, while we leave parts on other layers unchanged.

### 2.4 Hypothesis verification with HoC

Using optimized vocabulary we perform another inference of training images and calculate the HoC descriptor for every detected object $\mathcal{D}_{opt}(\mathcal{I}, \mathcal{L}_{opt})$. From detection we obtain category information $c_j$ and detected bounding box $r_j$ and within this bounding box we calculate a HoC descriptor $\mathcal{H}_j$ from all activated parts of second and third layer. We calculate HoC using second and third layer of the vocabulary. All computed descriptors $\mathcal{H}_j$ are then re-scored by a non-linear Support Vector Machine for a category model $c_j$. We used libSVM [1] with an RBF kernel and $\chi^2$ distance function. As the final step we perform a non-maxima suppression.

## 3 Experiments and results

We evaluated our method on the *ETHZ Shape Classes [2]* dataset with the same procedure as in [11]. Each category was evaluated independently and experiments were repeated five times to account for randomness in selecting training examples. Single experiment for each category was performed as follows. As training set we randomly selected half of the images that contained evaluated category and used as testing set used the other half combined with other images that did not contain any objects of the evaluated category. Similarly as in [11], we trained LHOP vocabulary $\mathcal{L}$ up to sixth layer, where 6th layer parts represent a category models and 5th layer parts represent an object models. We then ran an inference process with the LHOP vocabulary $\mathcal{L}$ on training images only and produced a set of initial training detections. Before the inference we also resized all the images by a factor of 1.2 and then started scaling them by a factor of $\sqrt{2}$ to produce around 4-7 scales per image.

### 3.1 Learning discriminative weights

From *ETHZ Shape Classes* training set we learn weights $\Theta_o$ for each object model $P_o^5$ from initial vocabulary. We first produce detections with the initial vocabulary $\mathcal{L}$ on all training images. All detections with the overlap (using PASCAL intersection/union function) of less than 0.3 are classified as negative while all detections with overlap over 0.7 are classified as positives. Any other detection that falls in between is classified as undefined and we avoid using them. From positive and negative detections we create activation vector $h_j$ and use linear libSVM [1] to obtain final weights $\Theta_o$. We additionally adjust bias for discriminative weights by performing detections on independent set of negative images. We randomly select 20% of images from Caltech-101 as independent set of images. In the detection stage weights are used to calculate discriminative score $f(h_j; \Theta_o)$ of each detection and any detection with score below threshold value of zero is filtered out.

### 3.2 Vocabulary optimization

In the vocabulary optimization step we used initial vocabulary $\mathcal{L}$ augmented with the discriminative weights for each object model and ran optimization process to produce optimized vocabulary $\mathcal{L}_{opt}$. As training set we used detections from all training images of *ETHZ Shape Classes* dataset for the specific category plus additional hard-negative detections from an independent set of images. The same criteria of 0.3 and 0.7 overlap threshold is used to select positive and negative detections. Similarly as in bias adjustment we also randomly selected 20% of images from Caltech-101 dataset. The object parts of both optimized and unoptimized vocabulary are show in Figure 6.

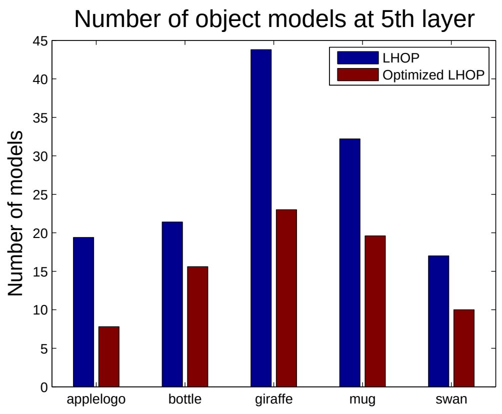


Figure 3: Number of object parts in 5th layer of our **optimized vocabulary (red)** compared to **original vocabulary (blue).**

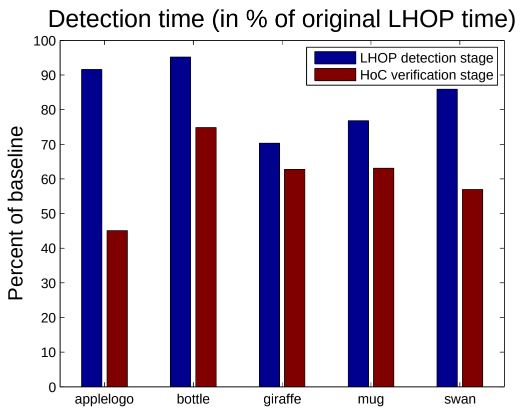


Figure 4: Computation time speedup. We report time required to complete **LHOP detection stage (blue)** and **HoC verification stage (red)** when using optimized vocabulary expressed in percent of the time taken by the unoptimized vocabulary.

### 3.3 Learning HoC classifier

In this process we used optimized vocabulary $\mathcal{L}_{opt}$ to produce hard-negative examples on *ETHZ Shape Classes* training set. Similarly as before we used detections overlapping less than 0.3 with the groundtruth as negative and examples overlapping more than 0.7 as positive. Together with the extracted original positive examples (regions were scaled to 120 pixels before inference process) we trained HoC classifier for each category.

### 3.4 Evaluation and results

In Figures 2 we measured the absolute number of all detections for each category separately. Figure 2a shows the number of detections *before* the non-maxima suppression and Figure 2b shows detections *after* the non-maxima suppression process. By looking at the detections before non-maxima suppression we see that the our method produces approximately 50% of all detections compared to the original vocabulary. The reductions is highest in the *apple logo*, *giraffe* and *swan* categories with only 45% of original detections while in categories *bottle* and *mug* there we retained 60% of all original detections. After the non-maxima suppression step this difference becomes less significant but we still produce by approximately 10% less detections.

We also measured a number of object models retained in the vocabulary after the optimization process. Figure 3 shows the absolute number of parts in 5th layer for each category compared to the number of parts from the original vocabulary. We can see that in each category we were able to significantly reduce number of parts. In particular the category *apple logo* has retained only 40% of original parts while category *giraffe* has around 52% of original parts. Slightly less parts were removed for other categories with 73% of original parts for *bottle*, 61% for *mug* and 58% of original parts for category *swan*.

Our optimizations also improved processing time in the detection stage. In Figure 4 we report for each category the time required for LHOP stage and for HoC verification stage in percent of the time required by the original vocabulary. Focusing on HoC verification stage (red bar), the biggest speedup has been achieved for the *apple logo* category with 45% of original time. With other categories the improvements are: 75% for *bottle*, 62% for *giraffe*, 63% for *mug* and 57% for category *swan*. We can also notice slight improvements in the LHOP inference stage (blue bar) with the averaged time of 83% of original time. Note, that we have optimized only 5th layer while other layers remained unchanged with the same number of parts. Considering that only 1/5th of the vocabulary has been optimized, the 17% speedup is quite significant. Overall we still need to take into account that LHOP inference stage takes most of the time and therefore overall speedup combined with HoC verification time is still around 17%.

We also evaluate overall detection rate to ensure we did not remove any information that is crucial for the detection. Looking at the result of Table 1 we see similar detection rate of our method and original method of [11]. In some cases our method performed by a percent or two better, while in other cases it performed slightly worse. In particular, our method achieved higher scores in categories *apple logo* and *mug*, while original method performed better in categories *bottle, giraffe* and *swan*. On average our method performed by less then half a percent worse, but still by almost 3% better then LHOP without any HoC verification step. We also noticed that both methods produced high variance, particularly in categories *bottle* and *swan*. The variations between different iterations are observable from Figure 5, where we see detection rate of category *bottle* varied between 70% and 90% and detection rate of category *swan* varied almost between 60% and 90%. Low performance is particularly noticeable in our method where at least two iterations in *swan* category produced detection rate of only 60%. This also explains a significant drop in overall performance of this category compared to the original method. Note, however overall performance is still at least at the same level as the results from [11].

We also report detection rate at 1.0 FPPI in Table 2 to compare our approach to the discriminative nodes of Kris-

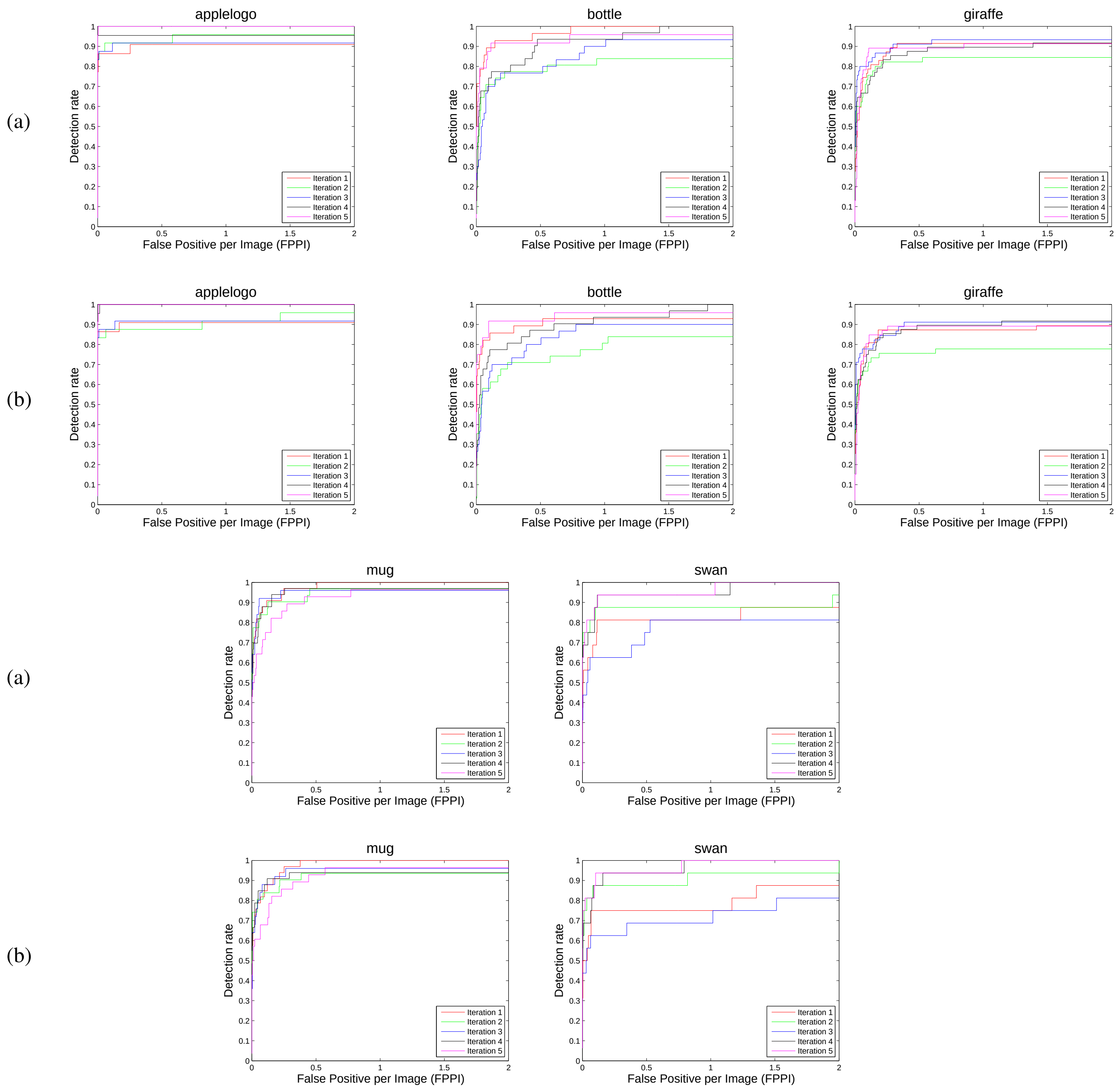


Figure 5: Detection rate over different FPPI rates (false positives per image) for each category from the *ETHZ Shape Classes* dataset. We ran five iterations and sampled examples randomly in each iteration to obtain different training/testing splits. Results are separated into row **(a) for unoptimized vocabulary** and row **(b) for optimized vocabulary (our).** Note, we used identical training/testing split for evaluation of both libraries.

| | Apple logo | Bottle | Giraffe | Mug | Swan | *Average* |
|---|---|---|---|---|---|---|
| our method | **94.0 (5.6)** | 83.1 (8.2) | 86.1 (6.1) | **94.5 (3,9)** | 83.7 (11.4) | 88.3 |
| LHOP + HoC verification [11] | 93.9 (3.8) | 84.5 (4.4) | **87.9 (4.0)** | 93.9 (3.8) | **85.0 (10.5)** | ***89.0*** |
| LHOP only [3] | 88.2 (3.4) | **87.6 (1.5)** | 83.5 (1.1) | 86.1 (2.0) | 80.0 (3.5) | *85.1* |
| Maji et al. [7] | 95.0 | 96.4 | 89.6 | 96.7 | 88.2 | 93.2 |

Table 1: Evaluation result on *ETHZ Shape Classes* with reported detection-rate (%) at **0.4 FPPI** averaged over five iterations (standard deviation values are shown in parentheses)

| | [6] | our method |
|---|---|---|
| Apple logo | 92.5 | **95.7** |
| Bottle | 85.4 | **86.6** |
| Giraffe | 82.3 | **87.0** |
| Mug | 86.5 | **96.0** |
| Swan | 70.5 | **87.5** |
| *Average* | *83.4* | **90.4** |

Table 2: Performance comparison to LHOP with discriminative node from [6] on *ETHZ Shape Classes* with reported detection-rate (%) at **1 FPPI** averaged over five iterations.

tan et al. [6]. We notice considerable improvements in the performance across all categories, with our average detection rate of 90.4% compared to detection rate of 83.4% in [6].

## 4 Discussion and conclusion

In this paper we introduced filtering with the discriminative information and vocabulary optimization to reduce number of false positive detections produced by hierarchical method learnt-hierarchy-of-parts (LHOP) [4]. We achieve this through vocabulary optimization to eliminate object models with poor discriminative information and through additional detection filtering step with the discriminative information. In contrast to [6] we introduce a discriminative information based on activation vector computed from response values and spatial information of activations from a detection's parse tree.

Using the *ETHZ Shape Classes [2]* database we have demonstrated that our method does not reduce performance in terms of detection rate but considerably reduces number of false positive detections while retaining only smaller set of object models in the vocabulary. In some cases, such as *apple logo* category, we are able to even increase the detection rate while we reduce false positives and the vocabulary of 5th layer by more then 60%. While detections before non-maxima suppression get reduced considerably, detections after non-maxima suppression are reduced only slightly. This indicates that in original method the false positive detections were handled mostly by good scoring of HoC descriptor and proper non-maxima suppression, while in our method they are now reduced even before non-maxima suppression. Additionally, we showed that our method also positively affect computation time. We were able to decrease computation time of LHOP inference stage by around 17% and by around 40% for the HoC verification stage. Lowered computation time of the HoC verification comes directly from the lower number of detections as there are less detections required to compute theirs HoC score with the non-linear kernel of support vectors. In the LHOP inference stage, saved computation time comes directly from the reduced number of object models in the optimized vocabulary. But since we optimize only 5th layer of the vocabulary we are only able to achieve 17% improvements in speedup.

In the future work we plan to reduce vocabulary parts in lower layers as well, in particular, we could remove parts connected to the 5th layer object parts that have been removed by our optimization method. We also plan to incorporate optimization step directly into learning of object layers to immediately select and reduce to the most optimal set of vocabulary parts.

**Acknowledgments.** This work was supported in part by ARRS research program P2-0214 and ARRS research projects J2-4284, J2-3607 and J2-2221.

## References

[1] Chih Chung Chang and Chih Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[2] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Object detection by contour segment networks. In *Proceeding of the European Conference on Computer Vision*, volume 3953 of *LNCS*, pages 14–28. Elsevier, June 2006.

[3] Sanja Fidler, Marko Boben, and Aleš Leonardis. A coarse-to-fine taxonomy of constellations for fast multi-class object detection. In *ICCV*, pages 687–700, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] Sanja Fidler and Ales Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *CVPR*. IEEE Computer Society, 2007.

[5] Iasonas Kokkinos and Alan Yuille. Inference and learning with hierarchical shape models. *Int. J. Comput. Vision*, 93(2):201–225, June 2011.

[6] Matej Kristan, Marko Boben, Domen Tabernik, and Aleš Leonardis. Adding discriminative power to hierarchical compositional models for object class detection. In *18th Scandinavian Conference on Image Analysis*, Jun 2013.

[7] Subhransu Maji and Jitendra Malik. Object detection using a max-margin hough transform. In *CVPR*, pages 1038–1045. IEEE, 2009.

[8] Björn Ommer and Joachim M. Buhmann. Learning the compositional nature of visual objects. In *Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

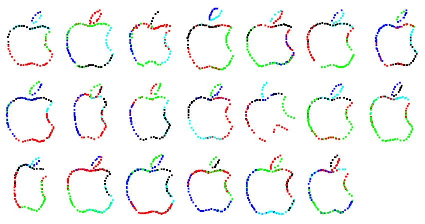
[9] Marc'Aurelio Ranzato, Fu J. Huang, Y. Lan Boureau, and Yann LeCun. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

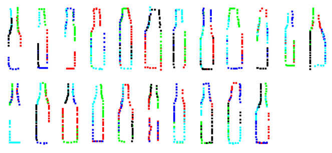
[10] D. Tabernik, M. Kristan, M. Boben, and A. Leonardis. Learning statistically relevant edge structure improves low-level visual descriptors. In *International Conference on Pattern Recognition*, 2012.

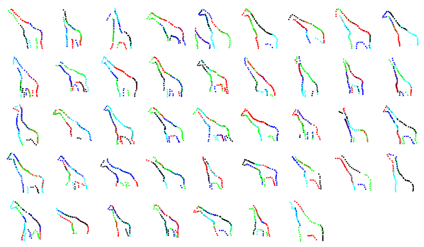
[11] Domen Tabernik, Matej Kristan, Marko Boben, and Aleš Leonardis. Hypothesis verification with histogram of compositions improves object detection of hierarchical models. In *Proceedings of the 21th International Electrotechnical and Computer Science Conference, ERK*, 2013.

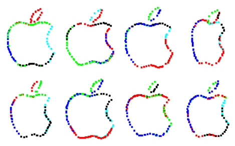
[12] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

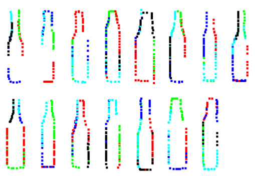
[13] Long Zhu, Yuanhao Chen, A. Torralba, W. Freeman, and A. Yuille. Part and appearance sharing: Recursive compositional models for multi-view. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1919 –1926, june 2010.

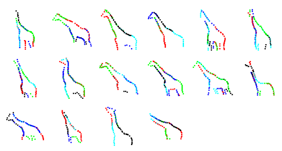

(a) Applelogo - unoptimized vocabulary

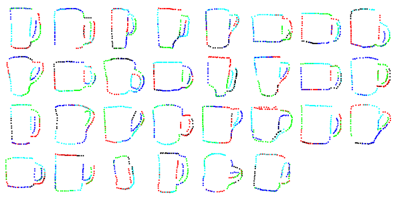

(b) Bottle - unoptimized vocabulary

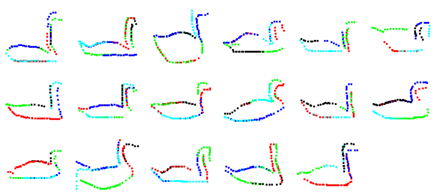

(c) Giraffe - unoptimized vocabulary

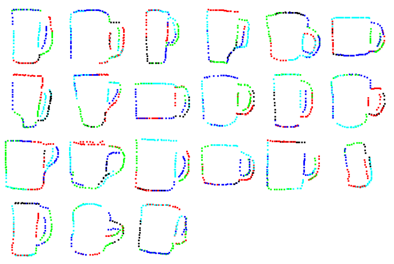

(d) Applelogo - optimized vocabulary (our)


(e) Bottle - optimized vocabulary (our)


(f) Giraffe - optimized vocabulary (our)


(g) Mug - unoptimized vocabulary


(h) Swan - unoptimized vocabulary


(i) Mug - optimized vocabulary (our)

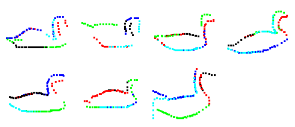

(j) Swan - optimized vocabulary (our)

Figure 6: Examples of 5th layer object parts for each category. In the first and third row are parts of the original vocabulary while in the second and fourth row are parts of our optimized vocabulary.