Learning Tree-structured Descriptor Quantizers for Image Categorization

Josip Krapac[1]
http://lear.inrialpes.fr/~krapac
Jakob Verbeek[1]
http://lear.inrialpes.fr/~verbeek
Frédéric Jurie[2]
http://users.info.unicaen.fr/~jurie/

[1] LEAR Team, INRIA Grenoble, Montbonnot St-Martin, France
[2] Université de Caen Basse-Normandie, Caen, France

Abstract

Current state-of-the-art image categorization systems rely on bag-of-words representations that model image content as a histogram of quantization indices that code local image appearance. In this context, randomized tree-structured quantizers have been shown to be both computationally efficient and yielding discriminative visual words for a given categorization task. This paper presents a new algorithm that builds tree-structured quantizers not to optimize patch classification – as it is done by approaches such as [18] – but to directly optimize the image classification performance. This approach is experimentally validated on several challenging data sets for which it outperforms other patch quantizers such as standard decision trees or k-means.

1 Introduction

The goal of image categorization is to assign labels to an image that relate to the image content at a semantically meaningful level. For example, the labels can indicate the presence of object categories, such as *cars*, or *bicycles*, or they can refer to scene types such as *indoor*, *city*, *beach*, *etc*. Over the last decade significant progress has been made in this area. Most of the current state-of-the-art systems are based on the bag-of-words image representation, a processing pipeline tracing back to [3] which consists of the following steps. (i) Local image regions are sampled, using interest points or from a regular grid. (ii) The appearance of regions is described using features with some degree of photometric invariance, *e.g*. using SIFT [15]. (iii) Features are quantized into a finite vocabulary, *e.g*. obtained by k-means clustering and each feature is coded by its quantization index. (iv) The overall image content is described by aggregating the quantization indices of all regions into a normalized frequency histogram. (v) The image histograms are used to train a classifier, *e.g*. a SVM.

The various stages of this pipeline have been intensively studied, and we refer to Section 2 for pointers to representative work in this area. However, it is important to note that generally processing steps (i)-(iv) of the pipeline are completely unsupervised; only in step (v) the image labels are used to train the classifiers. Although in some cases it might be advantageous to have a single image representation that can be used to address different tasks,

BMVC 2011 http://dx.doi.org/10.5244/C.25.47

the fact that the representation is not optimized for the task means that it is suboptimal in the sense that it less discriminatively and/or less succinctly captures the relevant image content.

Recently, effort has been put into optimizing quantizers for categorization tasks, see *e.g.* [1, 10, 14, 18, 30, 31, 33]. Following this line of research, we build on the tree-structured quantizers explored earlier in [18]. Despite the computational efficiency of the approach, one of big limitations of [18] is that the trees are learned to classify local image regions, with labels inherited from the image. Therefore the image region classification is used as a proxy for the true objective of image categorization. In this paper, contrarily to this previous approach, we use decision trees to perform quantization of local appearances, but train them in a manner to directly optimize categorization performance of the whole image.

In Section 3 we describe how we learn tree-structured quantizers using a randomized greedy forward-selection process: in each step we enlarge the decision-tree by splitting one of the existing nodes in a way that maximally improves the categorization performance. We train the model to perform ordinal regression: defined on image pairs, the loss is more expressive than the loss defined on images, and therefore is more suited to the incremental manner in which we build the clusterers, as opposed to [10, 31] that adapt an initial k-means quantizer to be more discriminative by minimizing image classification loss.

In Section 4 we present experimental results on two challenging public image categorization benchmarks, we evaluate our approach and compare to unsupervised quantizers and the method of [18]. The results show that our approach outperforms these alternatives, while producing more compact image representations. We also find that our approach benefits from using ensembles of trees, or "forests", and more so than using k-means or the method of [18]. We summarize our conclusions in Section 5.

2 Related work

As pointed out in the introduction, most current state-of-the-art systems for image classification follow the bag-of-words framework [3] and rely on the processing pipeline described above. For the stages (i), (ii), and (v), a consensus seems to be reached, *i.e.* the use of dense sampling [20], the representation of local features by SIFT like descriptors [32], and the combination of different non-linear kernels [28], possibly using explicit embeddings [17]. Two intermediate stages (iii) quantization, and (iv) aggregation – the key ingredients of a good image representation – have been the topic of active research within the last two years.

The construction of the visual vocabulary is often considered as pure vector quantization, done with k-means [3, 13, 26] or in a soft manner with Gaussian mixture models [24, 27]. Some authors have also noticed that the way the quantization is done can influence the performance of the classification algorithm. In [7] it was observed that the clusters are strongly unbalanced, which makes k-means focus on clustering the most frequent descriptors, that are in general less discriminative. An algorithm similar to mean-shift was proposed to overcome this limitation. For similar reasons, [12] suggests to use agglomerative clustering which gives more compact clusters, while being more robust to outliers. Hierarchical clustering is also very efficient in terms of the computational cost to assign features to visual words [19]. Finally, the results shown in [7, 12] demonstrate that in terms of classification performance the k-means quantizer is clearly not optimal.

More recently, methods have started to appear that design vocabularies in order to optimize classification performance, typically by merging visual words from an existing visual vocabulary [4, 29]. The drawbacks of these methods are that they rely on a large initial vo-

cabulary, which itself is not optimal with respect to the criterion to be optimized, and they still require computationally costly assignment of features to a large number of visual words.

One of the first papers to propose optimizing the discriminative power of the dictionary is [22], which combines general and class-specific dictionaries. While [22] and [29] optimize the vocabulary to ensure that the image representations as a whole are discriminative, other recent approaches optimized the visual words independently [10, 16, 18], usually by maximizing mutual information between category labels and the visual words. A drawback of these methods is that they aim to find visual words that can discriminate the class labels of local features (inherited from the images from which they are sampled), while the true objective is to be able to discriminate images (as a whole) based on their distribution of visual word assignments of the patches sampled from them.

A recent group of promising methods tries to directly optimize the visual vocabulary to minimize the image classification loss [1, 14, 30, 31, 33]. One way to do it is to use boosting-like approaches as in [31], which unifies quantizer generation with classifier training. Each image feature is encoded by a sequence of "visual bits" that are optimized iteratively for each category, based on the classifiers' performance using previous visual bits on the training data. Similarly, [33] proposes a framework for learning multiple non-redundant vocabularies, each being learned in sequence to extract the discriminative information not captured by preceding ones, and their corresponding classifiers. In [1, 14, 30] the dictionary and the classification model are learned jointly. In their formulation these problems are not jointly convex, but each sub-problem is, so they alternate between updating the classification model and updating the dictionary. These approaches differ in the way the feature is represented, quantization vs. sparse coding, in the aggregation method, averaging vs. max-pooling, and the loss used for image classification: (squared) hinge vs. logistic. All these methods, however, initialize their dictionary based on the reconstruction errors of the patch descriptors, and converge to local optima close to the initialization.

Optimizing over the space of tree-structured quantizers that we employ in our work is hard because of the non-differentiable relationship between tree parameters and the used loss. However, our sampling strategy shows to be very effective in practice. Like [1, 14, 30], our approach alternates between learning classifiers and updating the quantizer. However, we incrementally grow trees to quantize the feature space — the leafs of the tree representing the visual words. By optimizing the quantizer in a coarse-to-fine manner we are less sensitive to a specific initialization of the quantizer.

3 Learning tree-structured quantizers

Below we present the learning algorithm for our quantizers in Section 3.1, the split selection criterion in Section 3.2, and how to leverage ensembles of quantizers in Section 3.3. We use $X_i = \{x_{ij}\}_{j=1}^{N_i}$ to denote the set of N_i feature vectors x_{ij} extracted from the regions of the i-th image.

3.1 Randomized greedy tree construction

To quantize the feature space we use binary decision trees. Each non-leaf node n has an associated split criterion $f(x;\theta_n,\tau_n)$ that determines for a given feature vector x whether it will proceed to the left child if $f(x;\theta_n,\tau_n) < 0$, or to the right child otherwise. In particular, we use axis-aligned splits that threshold one element of the feature vector, *i.e.* we have

$f(x;\theta_n,\tau_n) = x^\top \theta_n - \tau_n$, and θ_n is all-zero except one entry which equals 1. Clearly, other types of split functions are possible, *e.g.* linear functions without restrictions on θ, or generalized linear functions. Each node of the tree corresponds to a part of feature space: the root is associated with the complete feature space, child nodes being associated with a subset of the space associated with the parent. The set of leaf-nodes forms a partitioning of the complete feature space into non-overlapping subsets. Given a quantization tree with L leaf-nodes, an image X_i is then represented by an l_1 normalized histogram $h_i \in [0,1]^L$ that codes in dimension d the fraction of the N_i image regions associated with the d-th leaf.

We learn the tree-structured quantizers by following randomized greedy forward-selection procedure. We start with a trivial tree with just the root-node. We then expand the existing tree by splitting the leaf-nodes. By expanding a leaf-node with two child-nodes we refine the current partitioning of the feature space, therefore expanding the current image representation. At each tree expansion step, we sample T candidate splits, by uniformly selecting one of the existing leafs, and determine θ_n by sampling uniformly a feature dimension. Given these, we sample a feature vector from the set of vectors associated with selected leaf and use its value on the selected dimension as the threshold τ_n. We evaluate each of the tentative tree expansions and select the best one, using a criterion described below. We iteratively continue splitting the leaf-nodes until we reach the specified number of leaves L.

3.2 Evaluating splits for image categorization

The procedure outlined above is essentially the same as the one used in decision tree learning, where the leafs of the tree are used to classify the feature vectors that describe image regions. In that case, the branches of the tree can be grown independently since the design of the tree under one node will not affect classification performance of feature vectors that are assigned to its sibling node. Splits are typically evaluated using the information-gain criterion [5]. Remember that our goal is not to use the leafs of the tree to classify individual feature vectors x_{ij}, but rather to classify images represented by sets of feature vectors $X_i = \{x_{ij}\}_{j=1}^{N_i}$ using a histogram h_i of leaf-assignments over the set. Therefore, in our case the branches of the tree can not be grown independently, and we should use a criterion that evaluates a split directly by its impact on the image categorization performance.

For each sampled split, corresponding to candidate expansion of the current tree, we update the image representations h_i. The histogram bin corresponding to the parent node being split is replaced by two new histogram bins corresponding to the two new child nodes. We then learn a classifier on a set of training images, using this enlarged image representation, and evaluate the split based on the classification performance on a validation set of images. We use linear classifiers, as we have to evaluate many tentative tree expansions: we sample and evaluate T splits for each of L expansions of the tree, so construction of one tree requires learning of $T \times L$ classifiers.

Like much recent work on image categorization, we evaluate performance in terms of average precision (AP), rather than the classification rate for a given threshold on the classifier score. Since we aim to optimize ranking performance, rather than classification performance, we learn a linear score function for ordinal regression where the goal is to ensure that for each pair of a positive and a negative image the score of the positive one is larger than the score of the negative one by some margin. If the score difference between a positive image h_i^+ and a negative image h_j^- is smaller than 1 we suffer the loss ξ_{ij}, otherwise $\xi_{ij} = 0$. In addition we use an ℓ_2 regularization term, which then leads to the following optimization

problem [6]:

$$\min_{w,\xi_{ij}\geq 0} \quad \frac{1}{2} w^\top w + C \sum_{i,j} \xi_{ij}, \tag{1}$$

$$\text{s.t.} \quad \forall_{i,j}: \quad w^\top (h_i^+ - h_j^-) \geq 1 - \xi_{ij}, \tag{2}$$

where i ranges over indices of positive examples, and j over indices of negative examples.

The number of constraints and slack variables is quadratic in number of images, but only a small fraction of all constraints is actually active — up to 15% in our experiments. The solution of the optimization problem can be obtained by cutting-plane methods [8] that minimize the cost function subject to a subset of the constraints, and iteratively add constraints that are violated by the current solution. In practice, we initially minimize the cost function subject to a set of 1,000 randomly selected constraints, and the sequentially add all violated constraints and re-solve. Typically, all constraints are satisfied within three iterations.

3.3 Learning ensemble of quantization trees

Due to the random and greedy nature of the tree construction algorithm, there are no guarantees that the above construction algorithm will find the optimal tree. Even when exhaustively considering all possible splits, which would be computationally infeasible, the split that currently improves the model most might not be optimal in combination with subsequent refinement of the quantization.

To address this problem, we have experimented with expanding nodes by more than two children, *i.e.* with a small tree, and evaluating the current split based on the future improvement that could be obtained by incorporating this split. Similar procedures have been used in decision tree learning before [25]. However, we did not observe significant improvements when employing this procedure

Instead, we have found that using a "forest" of several trees does give significant improvements, as has also been observed before for randomized decision trees [2]. Here we learn each tree independently, and then concatenate the histogram representations obtained using each tree to obtain our final image representation. Combining K trees of L leafs each we obtain a final representation of size $K \times L$. In addition to improving the results, using ensembles also significantly reduces the variance of performance.

4 Experimental evaluation

In this section we evaluate our approach and compare it to k-means quantization, and the quantizers of [18], which also have tree structure, but where the quantizer is constructed with a goal of classifying image regions, rather then images. In Section 4.1 we describe the data sets we use in the experiments, feature extraction, and implementation details. We present results for the Graz-02 data set in Section 4.2, and those for the 15-scenes data set in Section 4.3.

4.1 Data sets, features, and implementation details

The Graz-02 data set [21] contains 1476 images displaying instances of three objects classes: bicycles, cars, and people. In addition background images are included, which contain none

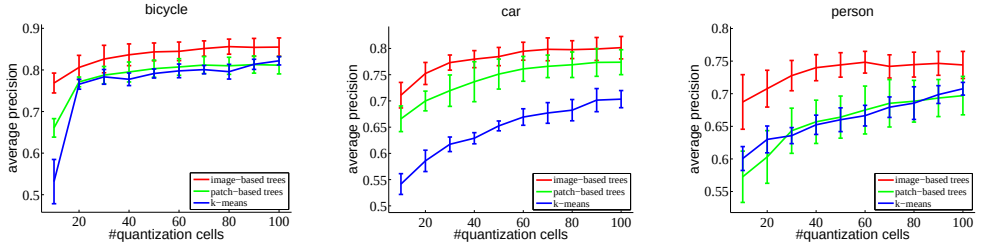

Figure 1: Performance for the classes in Graz-02, using one quantizer with a varying number of cells L using our image-based trees (red), the method of [18] (green), and k-means (blue).

of these categories but show scenes similar to the ones that display the objects. Strong intra-class variations due to different category instances, *e.g*. different car models, are represented in the data set. In addition, large variations in pose and scale makes recognition challenging. As in [4], we use odd numbered images from each class for learning the quantizers and classification models, and we evaluate our model on the remaining images.

The 15-scenes data set [11] contains 4485 images of various scene categories. Small inter-class variations between several indoor scenes are a major challenge of this dataset, *e.g*. *bedroom* vs. *livingroom*. As in [11] we use 100 randomly selected images per class for training, and use the remaining ones for evaluation. We repeat the procedure 10 times and report mean and standard deviation of average precision.

In all experiments we use the same image features. We sample regions of 20×20 pixels from a regular grid, spaced by 10 pixels. This is done at the original image scale, but also at four down-sampled versions of the image, rescaling it each time by a factor 1.2. For each patch we compute a SIFT descriptor [15] of 128 dimensions. When training the linear score functions, we use element-wise square-rooted visual word histograms instead of the original ones. In [23] it was shown that this leads to significant performance increases for linear classifiers over bag-of-word histograms generated by k-means, while maintaining the efficiency of linear classification. Experimentally we found that the same holds for histograms generated by tree structured quantizers (comparative results not included due to space limitations).

When learning our quantization trees, we sample $T = 100$ splits at each iteration. Since we evaluate performance in terms of average precision, we determine the regularization parameter C in Eq. (1) by 5-fold cross-validation to maximize average precision. Recall that we also use validation to prevent overfitting when growing the trees: for each tree we split the training data into two parts. We train the ordinal regression models on the first part, but select the split which minimizes the loss on the second part.

In our implementation of patch-based tree quantizers of [18], we fix the desired number of leafs L in advance. When constructing the tree, we keep a priority queue of nodes to expand, ordered by the information gain that can be obtained by expanding them, and grow the tree in a best-first manner.

4.2 Results on the Graz-02 dataset

In our first set of experiments we consider performance as a function of the number of quantization cells L, ranging from 10 up to 100. Since k-means depends on its initialization, and the tree-based methods on the random selection of splits, we show the average and standard deviation of performance over 10 experiments.

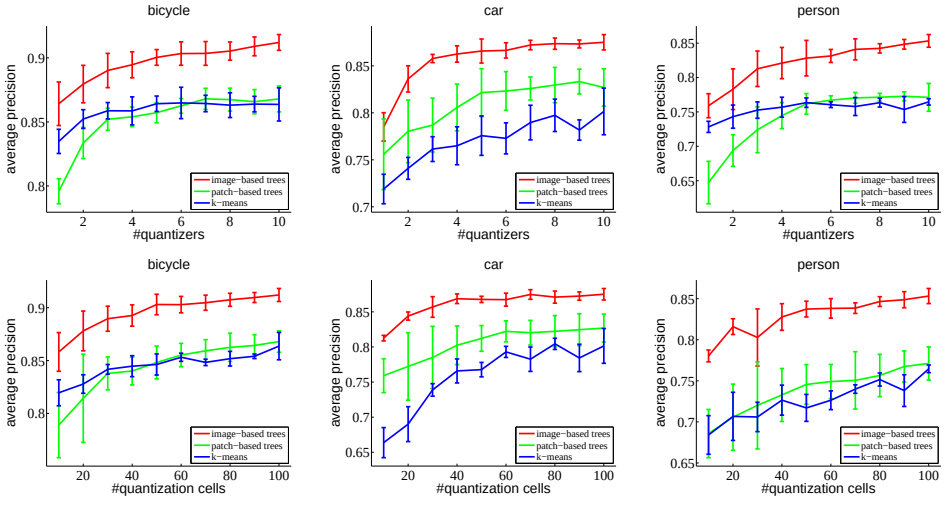

Figure 2: Performance for Graz-02. Top: using $L = 100$ cells, and varying the number of quantizers K. Bottom: ensembles of $K = 10$ quantizers while varying the number of cells L.

AP×100	K	L	bicycle	car	person
image-based tree (ours)	10	100	**91.2 ± 0.6**	**87.5 ± 0.8**	**85.3 ± 0.9**
patch-based tree [18]	10	100	86.8 ± 1.0	82.7 ± 2.0	77.1 ± 2.0
k-means	1	1000	88.3 ± 1.9	81.1 ± 0.8	83.1 ± 0.7
k-means	1	2000	90.0 ± 0.3	83.1 ± 0.6	85.6 ± 0.4
k-means	1	4000	**90.7 ± 0.3**	**84.8 ± 1.2**	**87.3 ± 0.4**

Table 1: Comparison to larger k-means vocabularies on Graz-02.

From results in Figure 1 we can see that our image-based trees give best results for all classes. The performance of k-means and patch-based trees are comparable to each other on the classes *bicycle* and *person*. To achieve similar performance our method needs much fewer quantization cells: on all classes using only 30-dimensional histograms our method outperforms the others when using up to 100-dimensional histograms.

In the second set of results in the top row of Figure 2 we look at performance as a function of the number of quantizers K that is combined in an ensemble, in each case using $L = 100$ cells. We see that the performance of tree-structured quantizers is improved on all three classes. For k-means vocabularies ensembles are less effective, in particular for the class bicycle. For this class, k-means and our image-based trees perform similar with a single quantizer, but our method outperforms k-means by 7% AP when using $K = 10$ quantizers.

In the third set of results in the bottom row of Figure 2 we again consider performance as a function of the number of quantization cells L, this time for ensembles of $K = 10$ quantizers. The results are similar to those of Figure 1: our method achieves very good performance using few quantization cells. Using only $L = 10$ cells our method outperforms k-means with $L = 100$ cells by a large margin. Our quantizers also significantly outperform those of [18].

Finally, in Table 1, we compare ensembles of shallow patch-based and image-based trees with results obtained with much larger k-means vocabularies. Only for one class and using vocabularies of 4000 cells, the k-means results improve over the image-based trees.

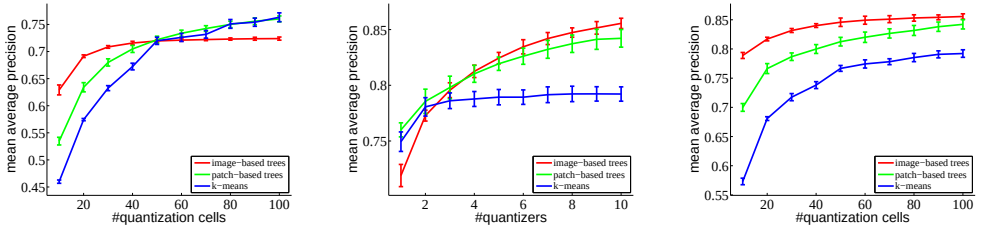

Figure 3: Performance in mAP for the 15-scenes data set: (left) using a single quantizer, and varying the number of cells L, (middle) using $L = 100$, and varying the number of quantizers K, (right) using $K = 10$ quantizers in the ensemble, and varying the number of cells L.

4.3 Results on the 15-scenes dataset

In Figure 3 we present graphs similar to those presented before, in this case showing the mean average precision over all 15 classes. The variance is measured over the mAP values obtained using ten random training sets. Here, using a single quantizer (left panel) our image-based trees give better performance than the other quantizers using $L < 50$ cells, and slightly worse using $L > 50$ cells. For this data set we also observe (middle panel) that the k-means vocabularies benefit least from ensembles: using more than two qunatizers has a marginal impact on performance. The tree-based quantizers, on the other hand, continue to improve with the number of quantizers in the ensembles. Using ensembles of $K = 10$ quantizers, and varying the number of cells L (right panel), we see that the k-means quantizer quickly improves with the number of cells, but its performance clearly remains behind as compared to that of the tree-based quantizers.

The difference between patch-based trees of [18] and our image-based trees on this dataset are smaller than on the Graz-02 dataset. This might be explained by the fact that in Graz-02 the different object classes appear against similar backgrounds, where in the 15-scenes data set the complete image contains discriminative information. Therefore, on Graz-02 when learning patch-based decision trees there will be many background patches, *e.g.* of buildings, that appear in all classes. Hence the decision tree will spend a large part of its capacity to separate buildings appearing as background of cars from buildings that appear as background of bicycles. The image-based trees do not suffer from this problem, since they are optimized for image classification, and it suffices to isolate a few informative features of the object class to obtain good classification results. On the 15-scenes data set, however, the class specific features are not hidden in a large pool of background features, and the patch-based trees also learn effective quantizers, without modeling the various backgrounds.

In order to compare our results to others that report multi-class classification accuracies we trained multi-class logistic regression models as follows. First, using one half of the training data, we learn a forest of K trees for each of the $C = 15$ classes in a one-versus-all manner as before. Then, we concatenate the K histograms of size L of all classes, yielding a representation of size $C \times K \times L$. We then learn a multi-class logistic discriminant classifier over this representation using all training data. Notice that this way we simply combine already learned class-specific quantizers to perform multi-class classification. Alternatively, we could optimize the quantizer directly for multi-class classification.

In Table 2 we report for various configurations of k-means, patch-based trees, and our image-based trees the recognition rates, and include mAP values for reference. Due to the

	K	L	accuraccy	mAP
k-means	10	10	59.7 ± 0.4	57.3 ± 0.6
patch-based tree	10	10	79.0 ± 0.9	70.0 ± 0.7
image-based tree	10	10	$\mathbf{80.0 \pm 0.9}$	$\mathbf{78.9 \pm 0.5}$
k-means	10	100	73.7 ± 0.8	79.2 ± 0.7
patch-based tree	10	100	$\mathbf{83.9 \pm 0.6}$	84.2 ± 0.8
image-based tree	10	100	$\mathbf{83.6 \pm 0.6}$	$\mathbf{85.6 \pm 0.5}$
k-means	1	1000	80.5 ± 0.7	84.0 ± 0.8
Lian *et al*. [14]	105	50	78.1 ± 0.7	–

Table 2: Summary of the results on 15-scenes dataset.

quite regular spatial layout of the 15-scenes classes, using features that capture spatial layout information may improve performance, see *e.g*. [1, 2] which report 85.6% and 88.1% respectively. For sake of clarity, however, we did not use the spatial layout information here. When using $K = 10, L = 10$ we obtain results comparable to k-means with $L = 1000$ cells, and using $K = 10$ and $L = 100$ we improve our accuracy by 3.6% to $83.6\% \pm 0.6$, This is clearly better than using k-means, and also better than the result of 81.4 ± 0.5 reported in [11] which captures the spatial layout information using spatial pyramids over k-means histograms. We compare our results to method of [14] which for each one-vs-one classification problem adapts the initial vocabulary of fixed size $L = 50$ via gradients of vocabulary parameters w.r.t. the image classification loss and combines the $K = 105$ classifiers to perform multi-class classification. Using ensembles of very shallow trees, $K = 10$ and $L = 10$, our results are already better than theirs.

Finally, we point out that even when an equal number of histogram cells is used, the histogram generation process requires several orders of magnitude less operations in our method as compared to k-means. For k-means, we need to compute distances to the $K \times L$ centers, each computed by D operations for a D dimensional descriptor; for SIFT descriptors with $D = 128$, $K = 1$ and $L = 1000$ this adds up to 128,000 operations per image patch. Using our tree-structured quantizers, we only need to compute approximately $\log L$ thresholds for each of the K trees, one threshold on each level of the tree. For $K = 10$ and $L = 100$ this results in applying only 70 thresholds.

5 Conclusion

We have introduced a method to learn tree-structured quantizers using a randomized greedy forward selection procedure, aimed at maximizing image classification performance. We found that combining several such trees in an ensemble leads to significant performance increases. The assignment of descriptors to histogram entries is much faster in the tree-structured quantizers than in k-means, since each node of the tree only applies a threshold on a single dimension of the descriptor. Our experimental results on two data sets show that our method improves over k-means quantization, and trees learned in a patch-based manner.

In future work we want to address the design of vocabularies that are shared across different classification tasks, both in multi-class settings as in settings with multiple binary labels. Secondly, we want to explore an approach where a forest of several trees is grown concurrently, so that they are trained to be mutually complementary.

Acknowledgements. This work was partly funded by the EU project AXES and the OESO and ANR project Quaero.

References

[1] Y-L. Boureau, F. Bach, Y. Le Cun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

[4] B. Fulkerson, A. Vedaldi, and S. Soatto. Localizing objects with smart dictionaries. In *ECCV*, 2008.

[5] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[6] T. Joachims. A support vector method for multivariate performance measures. In *ICML*, 2005.

[7] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, pages 604–610, 2005.

[8] J. Kelley. The cutting-plane method for solving convex programs. *SIAM Journal on Control and Optimization*, 1960.

[9] J. Krapac, J. Verbeek, and F. Jurie. Modeling spatial layout with Fisher vectors for image categorization. In *ICCV*, 2011.

[10] S. Lazebnik and M. Raginsky. Supervised learning of quantizer codebooks by information loss minimization. *Pattern Analysis and Machine Intelligence*, 31(7):1294–1309, July 2009.

[11] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[12] B. Leibe, K. Mikolajczyk, and B. Schiele. Efficient clustering and matching for object class recognition. In *BMVC*, 2006.

[13] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43(1):29–44, 2001.

[14] X. Lian, Z. Li, B. Lu, and L. Zhang. Max-margin dictionary learning for multiclass image categorization. In *ECCV*, 2010.

[15] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[16] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *NIPS*, 2008.

[17] S. Maji and A. Berg. Max-margin additive models for detection. In *ICCV*, 2009.

[18] F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification. *Pattern Analysis and Machine Intelligence*, 30(9):1632–1646, 2008.

[19] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[20] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *ECCV*, 2006.

[21] A. Opelt and A. Pinz. Object localization with boosting and weak supervision for generic object recognition. In *SCIA*, 2005.

[22] F. Perronnin. Universal and adapted vocabularies for generic visual categorization. *Pattern Analysis and Machine Intelligence*, 30(7):1243–1256, July 2008.

[23] F. Perronnin, J. Sánchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In *CVPR*, 2010.

[24] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[25] M. Roizman and M. Last. Look-ahead mechanism integration in decision tree induction models. In *Advances in Web Intelligence and Data Mining*, 2006.

[26] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[27] J. van Gemert, C. Veenman, A. Smeulders, and J.-M. Geusebroek. Visual word ambiguity. *Pattern Analysis and Machine Intelligence*, 32(7):1271–1283, 2010.

[28] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, 2007.

[29] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005.

[30] J. Yang, K. Yu, and T. Huang. Supervised translation-invariant sparse coding. In *CVPR*, 2010.

[31] L. Yang, R. Jin, R. Sukthankar, and F. Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. In *CVPR*, 2008.

[32] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73(2):213–238, 2007.

[33] W. Zhang, A. Surve, X. Fern, and T. Dietterich. Learning non-redundant codebooks for classifying complex objects. In *ICML*, 2009.