Fast Approximate GMM Soft-Assign for Fine-Grained Image Classification with Large Fisher Vectors

Josip Krapac and Siniša Šegvić

Faculty of Electrical Engineering and Computing
University of Zagreb, Croatia

Abstract. We address two drawbacks of image classification with large Fisher vectors. The first drawback is the computational cost of assigning a large number of patch descriptors to a large number of GMM components. We propose to alleviate that by a generally applicable approximate soft-assignment procedure based on a balanced GMM tree. This approximation significantly reduces the computational complexity while only marginally affecting the fine-grained classification performance. The second drawback is a very high dimensionality of the image representation, which makes the classifier learning and inference computationally complex and prone to overtraining. We propose to alleviate that by regularizing the classification model with group Lasso. The resulting block-sparse models achieve better fine-grained classification performance in addition to memory savings and faster prediction. We demonstrate and evaluate our contributions on a standard fine-grained categorization benchmark.

1 Introduction

In this work we address fine-grained classification (FGC), a problem where the inter-class variance is small *w.r.t.* intra-class variance, *i.e.* the objects from different classes may be very similar. This is a challenging task since the differences between the categories can be subtle and may cover a very small part of the image. For example, a discriminating feature between two bird species can be a specific feather pattern around the eye. The more specialized the class, the less data for learning its class model we can expect, and the more challenging learning good class models becomes. In short, FGC relies on finding a few discriminative features in a very large feature pool using a small amount of images annotated only with class labels, which resembles searching for needle in a haystack.

Although large Fisher vectors have displayed state-of-the-art performance for several FGC tasks [7], their main drawback remains computational complexity. First, obtaining large Fisher vector representation involves costly assignment of a large number of patches to a large number of GMM components. Second, classifier learning and inference involves very high dimensional dense vectors. This may pose scalability problems when the number of classes is large.

Our contributions address both of these drawbacks. In practice each patch is assigned to a very small number of GMM components [18]. We exploit this to

build a hierarchy over GMM components by agglomerative clustering [16]. This enables us to discard a large number of components early. GMM tree yields significant soft-assign speed up while mostly retaining classification performance. The technique is general since it does not require any labeled data, so it can be used *e.g.* in image retrieval. Second, we propose group-sparse classification models which are fast to train and evaluate, and have a small memory footprint. In addition, these models display significantly improved classification performance, especially when coupled with Fisher vector intra-normalization. We demonstrate the value of our contributions experimentally on the 14 category subset ("Birdlets") [5] of the Caltech-UCSD Birds 200-2011 dataset [22].

2 Related Work

Goldberger and Roweis [6] consider grouping of GMM components by an iterative regroup-refit procedure similar to the k-means clustering algorithm. The main idea is to find the grouping of original components into a smaller mixture model such that the KL divergence between the obtained smaller model and the original one is minimized. However, there is no guarantee that the obtained higher-level GMM will indeed speedup the soft-assign, since many lower-level components can be merged into a single higher-level component.

Simonyan *et al.* [19] used hard assignment to the closest GMM component in order to speed up Fisher vector computation. Although this procedure speeds up the computation of Fisher vector once the soft-assigns have been computed, it does not reduce the complexity of the soft-assign computation which is the most intensive part of Fisher vector computation.

Verbeek *et al.* [21] speed up the EM algorithm for large datasets by first clustering the data with a kd-tree, and then performing EM steps on the clusters instead of individual points. The combination of this approach and the one proposed here is an interesting avenue for speeding the soft-assign for a group of descriptors, since our contribution is able to speed up the soft-assign for a single descriptor.

Gosselin *et al.* [7] speed up large Fisher vector construction by discarding the patches whose SIFT descriptor norm is below a threshold. They show that this does not influence significantly the classification performance, while it reduces the computational complexity of Fisher vector computation. For the remaining SIFT descriptors they still have to perform computationally intensive soft-assign.

Approximate nearest neighbor methods (ANN) like product quantization [9] or locality-sensitive hashing [8] can be used to quickly short-list the components most responsible for generating the data point. However, in our case the number of the descriptors in the image is an order of magnitude larger than the number of the GMM components. Therefore it is difficult to obtain a good trade-off between feature coding time and quality of the recall of the GMM components. Additionally, ANN methods usually assume L_2 distance, while the GMM soft-assign requires computing the likelihood of a normal distribution, which corresponds to Mahalanobis distance.

Group-sparse classification models have been previously used for general object classification with bag-of-visual-words histograms [13], but to the best of our knowledge we are the first to report results of group-sparse classification models with Fisher vectors. We have previously used sparse classification models [11], but have not constrained them to be group-sparse. Instead, we have selected a predefined number of top components considering the norm of the corresponding part of the model vector. In this work, due to group-sparse regularizers, we are able to directly control the trade-off between the classification performance and the number of GMM components selected by the model.

The intra-component variant of Fisher vector normalization we use here is related to intra-normalization used in VLAD descriptor [1]. As noted in [1], this normalization is beneficial for reducing the influence of bursty visual elements on the image representation.

3 Fisher Vector Image Representation

We represent images with a set of densely sampled patches at a fixed grid and multiple scales [4]. This enables a good description of image content and invariance to scale changes, but it also usually results in a fairly large number N of patches per image. Each patch is described by a D-dimensional descriptor $\mathbf{x} \in \mathbb{R}^D$ invariant to local photometric and geometric transformations [14] and coded using a generative model of patch descriptors, which is usually a Gaussian mixture model (GMM):

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k), \quad \pi_k = \frac{\exp(\alpha_k)}{\sum_{i=1}^{K} \exp(\alpha_i)} \,. \tag{1}$$

In the above equation, K is the number of components, while $\Theta = [\pi_k, \mu_k, \mathbf{\Sigma}_k]_{k=1}^{K}$ are GMM parameters. The parametrization of component weights $\boldsymbol{\pi} = \{\pi_k\}_{k=1}^{K}$ with $\boldsymbol{\alpha} = \{\alpha_k\}_{k=1}^{K}$ ensures that $\boldsymbol{\pi}$ sum to one. To reduce the number of GMM parameters we assume a diagonal $\boldsymbol{\Sigma}_k$ for all K components. The parameters are learned to maximize the training data likelihood using the EM algorithm. Each patch descriptor $\mathbf{x}$ is coded with a Fisher vector $\Phi(\mathbf{x})$ [18] that is a gradient of the GMM log-likelihood *w.r.t.* the GMM parameters Θ:

$$\begin{aligned} \Phi(\mathbf{x}) &= [\cdots \Phi_k(\mathbf{x}) \cdots] = [\cdots \nabla_{\alpha_k} \log p(\mathbf{x}|\Theta), \nabla_{\boldsymbol{\mu}_k} \log p(\mathbf{x}|\Theta), \nabla_{\boldsymbol{\Sigma}_k^{-1}} \log p(\mathbf{x}|\Theta), \cdots] \\ &= [\cdots \gamma_k(\mathbf{x}) - \pi_k, \gamma_k(\mathbf{x}) \mathbf{\Sigma}_k^{-1} (\mathbf{x} - \mu_k), \gamma_k(\mathbf{x}) (\mathbf{\Sigma}_k - (\mathbf{x} - \mu_k)^2), \cdots]. \end{aligned} \tag{2}$$

In the Eq. (2) $\gamma_k(\mathbf{x}) = p(k|\mathbf{x})$ corresponds to the responsibility of component k for generating the descriptor $\mathbf{x}$, also known as the soft-assign of the descriptor $\mathbf{x}$ to the Gaussian component k:

$$\gamma_k(\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k)}{p(\mathbf{x}|\Theta)}. \tag{3}$$

Assuming independence of the image patches, we obtain the Fisher vector for the whole image $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ as an average of patch Fisher vectors $\Phi(\mathbf{X}) = \sum_{i=1}^N \Phi(\mathbf{x}_i)/N$. The dimension of this representation is $(2D+1)K$. In the implementation we use the improved version of Fisher vector [17] which employs normalizations that significantly improve the classification performance.

To ensure that object's fine-grained class-specific parts remain identifiable even after aggregation of patch Fisher vectors into the image representation a large number of GMM components K is needed. This way a set of patch descriptors is embedded in a highly-dimensional vector space, in which a hyperplane defined by a linear classifier corresponds to a highly non-linear decision surface in the patch descriptor space. Therefore large Fisher vectors enable modeling of subtle image details, a requirement necessary for discrimination of very similar images.

4 Fast Approximated Soft-Assign Computation

Our main contribution is related to the fast computation of soft-assign (3). For each of N patches we need to compute the Mahalanobis distances to K components, required to compute the denominator of (3). Therefore the complexity of the soft-assign computation for N patches represented by D-dimensional vectors is $O(DKN)$. In practice only a small fraction of components is responsible for generation of a data point. This means that the Fisher vector encoding for each data point $\mathbf{x}$ will be block-sparse, since for many components $\Phi_k(\mathbf{x})$ will be zero.

We want to take advantage of this sparsity to speed up the soft-assign by discarding the components that are not likely to be responsible for generation of the data point. To this end we construct a hierarchy of GMM models by iteratively merging the components of the original flat GMM. This hierarchy enables us to concentrate, at each level of hierarchy, on a subset of top K_t most responsible components for generating the data point. Thus at each level of the GMM tree, the soft-assign computation for each data point requires $O(DK_t)$ operations.

GMM Tree Construction. Since our goal is to speed-up the soft-assign we concentrate on balanced binary trees. Clearly, this constraint produces a sub-optimal approximation of the considered GMM with a given number of components. However, it gives us theoretical guarantees in terms of expected speed-up, since the number of operations to compute soft-assign is the same for each data point, unlike [6]. We start from a large flat GMM whose soft-assign we want to speed up, and consider its components as the leaves of the tree. At each new level of the tree we create a new component (the parent node) by merging exactly two components at the lower level (children nodes). To determine the best two children nodes to merge, we first find the closest sibling of each child node in terms of KL divergence:

$$c(i) = \underset{j \in \{1 \ldots K\}/i}{\arg\min} \, d_{KL}(\theta_i, \theta_j) \tag{4}$$

$$d(i) = d_{KL}(\theta_i, \theta_{c(i)}). \tag{5}$$

There exists a closed-form solution for KL divergence between two normal distributions, so we can quickly determine the distances between the children nodes. We then greedily merge the closest children nodes, by first merging the child node i whose KL divergence $d(i)$ to the closest sibling $c(i)$ is the largest. Every time we merge the children nodes into the parent node we re-compute the closest siblings for the ones that are not yet merged. The main motivation for this procedure is to ensure that the newly created parent nodes do not overlap.

When merging the children nodes i and $j = c(i)$, we derive the parameters of the parent node m following [23]:

$$\pi_m = \pi_i + \pi_j \tag{6}$$

$$\pi_m \mu_m = \pi_i \mu_i + \pi_j \mu_j \tag{7}$$

$$\pi_m (\Sigma_m + \mu_m^\top \mu_m) = \pi_i (\Sigma_i + \mu_i^\top \mu_i) + \pi_j (\Sigma_j + \mu_j^\top \mu_j). \tag{8}$$

This construction does not produce optimal component pairings in the sense of minimizing the KL divergence between the GMMs at the two neighboring levels of binary tree. However, the procedure is very fast and gives very good results, as we demonstrate in the experimental section.

Fast Assignment. We use the GMM tree to perform the soft-assignment of patch descriptor $\mathbf{x}$ to the original GMM components that are the leaves of the GMM tree. Given a selected top number K_t of components we skip first $\log_2(K_t) + 1$ levels of the tree, since at these levels the number of nodes is smaller than K_t. At each level we expand the K_t nodes of interest into $2K_t$ nodes at the next level. We subsequently select the top K_t nodes by considering the value of numerator in (3). We continue this procedure until we reach the bottom of the tree. The denominator of the expression (3) is then approximated by considering only K_t components selected by the tree. This way the complexity for each patch is reduced from $O(DK)$ to $O(DK_t \log_2(\frac{K}{K_t}))$, which significantly speeds up the computation of Fisher vectors. This fast assignment is illustrated in Fig. 1. The choice of K_t determines the trade-off between soft-assign speed up and the quality of the approximation.

5 Group-Sparse Models

In order to perform FGC, we learn a linear classifier $\mathbf{w}$ operating on the Fisher vector representation:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\arg\min} \sum_{i=1}^{M} \ell(\mathbf{w}, \Phi(\mathbf{X}_i), y_i) + \lambda \mathcal{R}(\mathbf{w}), \tag{9}$$

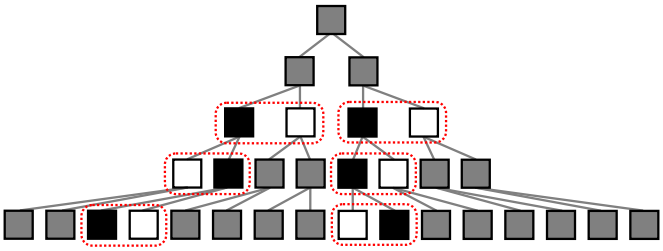

Fig. 1. The fast assignment procedure for $K = 16, K_t = 2$. The likelihood for the gray nodes is not evaluated. At each level we evaluate the likelihood of only $2K_t$ components (circled) and only top K_t components are expanded (black).

where $(\Phi(\mathbf{X}_i), y_i)_{i=1}^M$ is the training set of M image representations and class labels, ℓ is the loss function, $\mathcal{R}$ is the regularizer, while λ determines the compromise between the two. To enforce block sparsity, we employ group Lasso regularization $\mathcal{R}(\mathbf{w}) = \sum_{k=1}^{K} \|\mathbf{w}_k\|_2$, where $\mathbf{w}_k$ is the Fisher vector block of size $2D + 1$ corresponding to the k^{th} GMM component. We use the logistic loss $\ell_{\log}(\mathbf{w}, \mathbf{x}, y) = \log(1+\exp(-y\mathbf{w}^\top \mathbf{x}))$ which means that the optimization problem (9) is convex.

We assume that class-specific object parts correspond to small number of class-specific patches that lie in compact parts of the patch descriptor space. Fisher vector representation maps compact parts of the descriptor space into blocks of Fisher vector representation. The learned group-sparse discriminative model selects the GMM components corresponding to the class-specific features. Therefore these models are well suited for FGC, especially where scalability is needed, since the sparsity also enables fast model evaluation and small memory requirements.

6 Experiments

Dataset. We demonstrate our contributions on the 14-class subset [5] of Caltech-UCSD Birds 200-2011 birds dataset [22]. This subset consists of 7 classes of *Vireos* (Black-capped Vireo, Blue-headed Vireo, Philedelphia Vireo, Red-eyed Vireo, Warbling Vireo, White-eyed Vireo and Yellow-throated Vireo) and 7 classes of *Woodpeckers* (American Three-toed Woodpecker, Pileated Woodpecker, Red-bellied Woodpecker, Red-cockaded Woodpecker, Red-headed Woodpecker, Downy Woodpecker and Northern Flicker). There are 419 train and 398 test images, each split having around 30 images per class.

Parameters. We follow the standard Fisher vector classification pipeline: we use dense patch sampling at four scales following the procedure from [3] and describe the patches by SIFT [14] and color features of Perronnin et al. [17]. We used the VLFeat library [20] for computing SIFT descriptors. We reduce

both SIFT and color features to 64 dimensions by PCA. For each feature type we select a random subset of 500000 features from the images in the training set and learn one GMM per feature type. We use the learned GMMs to obtain Fisher vectors, again one per feature type. We normalize the Fisher vectors using the power and metric normalizations, as suggested in [17]. We do not use spatial layout coding, although it is likely that such coding (e.g. SPM [12]) could improve classification results, especially when training data is enlarged by using mirrored images (image flips) and random crops. All our classification models are one *vs.* all logistic regression, learned with 1000 iterations of FISTA algorithm [2] implemented in SPAMS [15].

Results. Fig. 2 shows the speedup and the quality of Fisher vectors with fast approximate soft-assign as we vary K_t. Increasing K_t gives a better approximation at the expense of a smaller speed-up. The speed-up is defined by the ratio of the time needed to compute the original Fisher vector and the time needed

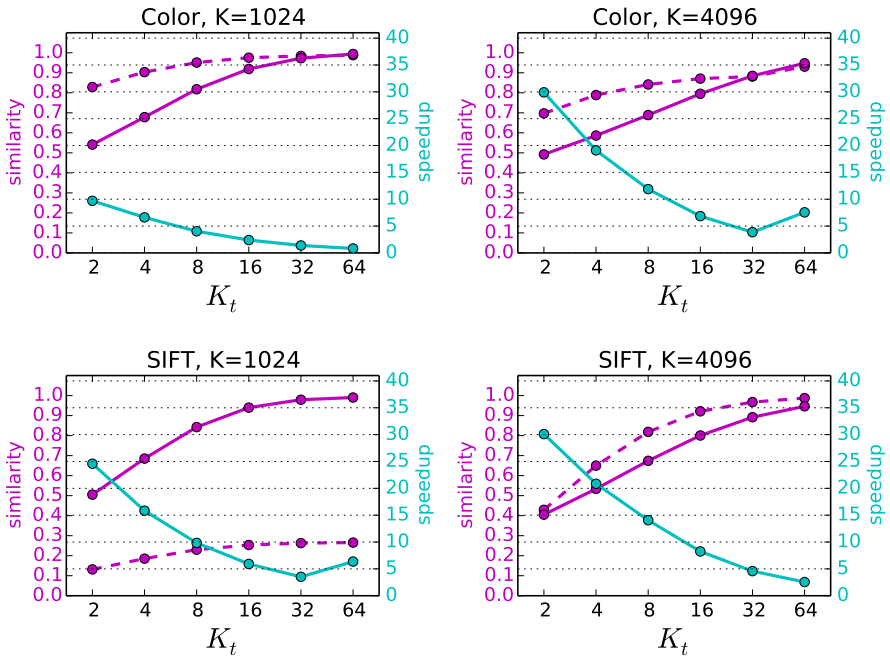

Fig. 2. Influence of the approximated soft-assign on the speedup and the approximation error measured as the dot product between the original Fisher vector and the approximated one. The magenta curves denote dot product between vectors (full: block-wise L_2 normalization, dotted: full L_2 normalization). The cyan curve displays the speedup achieved with top K_t components at each level of the GMM tree. Our approximation offers a trade-off between quality and the speed-up.

to compute its approximated version. Notice that the speedup can be very high for smaller K_t because of the caching patterns employed by the CPU: the components in the higher levels of the tree that are queried more often could be kept the L1 cache. The quality of approximation is measured by the dot product between the original and the approximated Fisher vector normalized with their L_2 norm. We consider the full Fisher vector normalization and the block-wise (or intra-component [1]) normalization where we normalize Fisher vector blocks corresponding to GMM components.

The approximated Fisher vector is different from the original vector due to the errors in soft-assign approximation. The Fisher vector of the patch descriptor can "blow up" if the GMM tree does not give a correct prediction of the soft-assign. A couple of mis-assigned descriptors can significantly influence the Fisher vector of the image (*c.f.* SIFT features with $K = 1024$). Block-wise normalization ensures that the approximation errors influence only the Fisher vector blocks corresponding to the components whose responsibility is mis-predicted. This is another benefit of using blockwise normalization, in addition to reducing the influence of bursty visual elements [1]. With $K_t = 64$, for all considered cases, we obtain the Fisher vectors that are very close to original ones. Although we concentrate on image classification, these results suggest that the approximations could also be useful for image retrieval, *e.g.* for construction of the VLAD descriptor [10].

Next we show the influence of the Fisher vector approximations on classification performance measured by mean average precision (mAP). In these experiments we fix $\lambda = 10^{-4}$ to obtain sparse models, and to ensure that only the approximations influence the performance. Tab. 1 shows that classification performance is not significantly influenced for a wide range of approximations. For our best performing feature and vocabulary size (color features with $K = 1024$), when using only $K_t = 2$ we lose only 5.5 points of mAP compared to using Fisher vectors computed without approximations, while achieving 10x speedup. In all following experiments we report results on the approximated Fisher vec-

Table 1. Influence of approximated soft-assign on the classification performance, with metric intra-normalization and group Lasso regularization.

		color				SIFT			
	K	1024		4096		1024		4096	
		mAP	speedup	mAP	speedup	mAP	speedup	mAP	speedup
K_t	2	51.08	9.70	46.08	29.90	37.52	24.58	32.99	30.09
	4	52.87	6.62	48.20	19.09	39.96	15.83	36.10	20.85
	8	54.50	4.03	51.25	11.88	42.60	9.83	39.28	14.06
	16	54.73	2.40	52.25	6.85	42.65	5.91	39.34	8.24
	32	55.00	1.39	54.32	3.86	43.30	3.52	40.15	4.58
	64	**56.00**	0.83	54.90	7.56	**43.44**	6.36	40.35	2.55
	full	55.59	1	54.71	1	42.58	1	41.59	1

tors using $K_t = 16$, since this setting offers a good trade-off between obtained speedup and classification performance.

We also compare our performance to the Fisher vector baseline and show how block-wise normalization, sparsity-inducing regularization and use of approximations influence the classification performance (Table 2). Here we determine the hyper-parameter λ by two-fold cross-validation in the range of values $\log_{10}(\lambda) = [-4, -7]$. Our first experiment uses Fisher vectors obtained without approximations, with overall metric normalization and L_2 regularization. This is the standard Fisher vector representation, used also in [7]. Intra-component L_2 normalization achieves remarkable effects: the classification performance is improved by almost 10 percent points with color descriptors and 7 percent points with SIFT descriptors. The group Lasso regularization gives additional 5 percent points with color descriptors, while only marginally improving the classification performance with SIFT descriptors. When we use our approximated Fisher vectors, we obtain almost the same performance as with original Fisher vectors: marginally worse with color descriptors and marginally better with SIFT. We also tried learning the classifier on concatenation of color and SIFT Fisher vectors, but this resulted in slightly worse performance compared to using color features alone (53.54% mAP).

Table 2. Influence of Fisher vector normalization, regularization and approximated soft-assign on the classification performance. Fast Fisher vectors were obtained with $K_t = 16$.

L_2 normalization	Regularization	Fast	color	SIFT
full	L_2	No	41.67	35.18
intra-component	L_2	No	50.09	42.09
intra-component	group Lasso	No	55.83	42.67
intra-component	group Lasso	Yes	55.00	43.37

Finally, Tab. 3 shows per-class performance with a fixed $\lambda = 10^{-4}$ and when λ is cross-validated. We first notice that our performance is worse for *Vireos*, and that for these 7 classes the cross-validating λ results in non-sparse models. The classification performance for *Woodpeckers* is much better and almost all cross-validated models are sparse. When we fix λ to give more weight to the regularizer we obtain sparse models without significant drop in performance: 55.00% mAP with cross-validated λ *vs.* 54.73% mAP with the fixed λ. This setting is especially interesting when learning models for a large number of classes, since the obtained classification performance is almost the same, while the used modes are 2–4 times smaller. In addition to memory savings, the sparsity enables faster model learning and evaluation. When coupled with our approximations for Fisher vector computation, the group-sparsity allows skipping the computation of Fisher vectors for the descriptors generated by the GMM components discarded by the classification model.

Table 3. Per-class analysis of influence of sparsity on classification performance when using color features and $K = 1024$. The performance is measured by average precision (AP), while NNZ denotes the number of GMM components selected by the group sparse classification model. Enforcing sparsity only slightly degrades classification performance, while it yields 2-4x more compact models *w.r.t.* to L_2 regularized models. The best results are obtained with group-sparse models (boldface).

	$\lambda = \lambda_{\text{cv}}$		$\lambda = 10^{-4}$	
Name	AP	NNZ	AP	NNZ
BC	47.02	1024	**48.17**	403
BH	25.83	1024	24.19	409
P	32.90	1024	29.06	357
RE	24.94	1020	**25.74**	374
W	21.79	1018	**22.43**	387
WE	50.51	374	**50.51**	374
YT	46.54	252	**46.54**	252

Vireos

	$\lambda = \lambda_{\text{cv}}$		$\lambda = 10^{-4}$	
Name	AP	NNZ	AP	NNZ
AT	72.77	327	**72.77**	327
PW	64.56	292	**64.56**	292
RB	88.89	242	**88.88**	242
RC	81.44	1015	80.54	311
RH	84.74	251	**84.74**	251
D	60.94	268	**60.94**	268
NF	67.15	294	**67.15**	294

Woodpeckers

7 Conclusion

We have proposed an approximate algorithm for fast soft-assignment of high-dimensional patterns to the components of a large GMM model. The proposed algorithm brings substantial speed-ups to the recovery of global image representations based on Fishers vectors, at the price of a tolerable (or even negligible) impact to the classification accuracy. Additionally, we have shown that a recent method for enforcing group sparsity may improve both the classification performance and the processing speed at the same time. Finally, we have shown that these sparse models achieve best results when the Fisher vectors are subjected to the metric intra-normalization, rather than the usual metric normalization across the whole vector.

These three contributions improve the classification performance in the fine-grained case, where only a small portion of the image allows to bring the decision about the image class. Experiments performed on the "Birdlets" dataset confirm substantial advantage over the baseline in terms of better performance classification performance and faster execution. The proposed method allows to choose a desired trade-off between the classification performance and the execution speed. Our best performing classification models achieved improvement of 14 points of mAP *w.r.t.* the baseline while offering 2x increase in the execution speed. The most interesting direction for the future work is application of the presented contribution to more diverse classification datasets such as PASCAL VOC.

Acknowledgement . This work has been fully supported by Croatian Science Foundation under the project I-2433-2014.

References

1. Arandjelović, R., Zisserman, A.: "All about VLAD". In: CVPR (2013)
2. Beck, A., Teboulle, M.: "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems". SIAM Journal of Imaging Sciences (2009)
3. Chatfield, K., Lempitsky, V., Vedaldi, A., Zisserman, A.: "The devil is in the details: an evaluation of recent feature encoding methods". In: BMVC (2011)
4. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: ECCV-WSLCV. pp. 1–22 (2004)
5. Farrell, R., Oza, O., Zhang, N., Morariu, V.I., Darrell, T., Davis, L.S.: Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In: ICCV. pp. 161–168 (2011)
6. Goldberger, J., Roweis, S.: "Hierarchical clustering of a mixture model". In: NIPS. pp. 505–512. MIT Press (2005)
7. Gosselin, P.H., Murray, N., Jégou, H., Perronnin, F.: "Inria+Xerox@FGcomp: Boosting the Fisher vector for fine-grained classification". Tech. rep., INRIA / XRCE (2013)
8. Indyk, P., Rajeev Motwani . In Proceedings of the (STOC 1998), pages, .: "Approximate nearest neighbors: towards removing the curse of dimensionality". In: 30th ACM Symposium on the Theory of Computing. pp. 604–613 (1998)
9. Jégou, H., Douze, M., Schmid, C.: "Product Quantization for Nearest Neighbor Search". PAMI 33(1), 117–128 (Jan 2011)
10. Jégou, H., Douze, M., Schmid, C., Pérez, P.: "Aggregating local descriptors into a compact image representation". In: CVPR (2010)
11. Krapac, J., Šegvić, S.: "Weakly Supervised Object Localization with Large Fisher Vectors". In: VISAPP (2015)
12. Lazebnik, S., Schmid, C., Ponce, J.: "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories". In: CVPR (2006)
13. Liu, Y.: "Image Classification with Group Fusion Sparse Representation". In: ICME. pp. 568–573 (2012)
14. Lowe, D.G.: "Distinctive Image Features from Scale-Invariant Keypoints". IJCV 60, 91–110 (2004)
15. Mairal, J., Jenatton, R., Bach, F.R., Obozinski, G.R.: "Network Flow Algorithms for Structured Sparsity". In: NIPS (2009)
16. Murtagh, F., Contreras, P.: "Algorithms for hierarchical clustering: an overview". Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery 2(1), 86–97 (2012)
17. Perronnin, F., Sánchez, J., Mensink, T.: "Improving the Fisher Kernel for Large-Scale Image Classification". In: ECCV (2010)
18. Sánchez, J., Perronnin, F., Mensink, T., Verbeek, J.J.: "Image Classification with the Fisher Vector: Theory and Practice". IJCV 105(3), 222–245 (2013)
19. Simonyan, K., Vedaldi, A., Zisserman, A.: "Deep Fisher Networks for Large-Scale Image Classification". In: NIPS. pp. 163–171 (2013)
20. Vedaldi, A., Fulkerson, B.: "VLFeat: An Open and Portable Library of Computer Vision Algorithms". http://www.vlfeat.org (2008)
21. Verbeek, J.J., Nunnink, J., Vlassis, N.: "Accelerated EM-based clustering of large data sets". Data Mining and Knowledge Discovery 13(3), 291–307 (2006)
22. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: "The Caltech-UCSD Birds-200-2011 Dataset". Tech. rep., California Institute of Technology (2011)
23. Zhang, Z., Chen, C., Sun, J., Chan, K.L.: "EM algorithms for Gaussian mixtures with split-and-merge operation". Pattern Recognition 36(9), 1973–1983 (2003)