Fisher vectors over random density forests for object recognition

Claudio Baecchi, Francesco Turchini, Lorenzo Seidenari, Andrew D. Bagdanov, Alberto Del Bimbo
Media Integration and Communication Center
Università degli Studi di Firenze
Firenze, Italy
{claudio.baecchi,francesco.turchini}@unifi.it, bagdanov@dsi.unifi.it,
{lorenzo.seidenari, delbimbo}@unifi.it

***Abstract*—In this paper we describe a Fisher vector encoding of images over Random Density Forests. Random Density Forests (RDFs) are an unsupervised variation of Random Decision Forests for density estimation. In this work we train RDFs by splitting at each node in order to minimize the Gaussian differential entropy of each split. We use this as generative model of image patch features and derive the Fisher vector representation using the RDF as the underlying model. Our approach is computationally efficient, reducing the amount of Gaussian derivatives to compute, and allows more flexibility in the feature density modelling. We evaluate our approach on the PASCAL VOC 2007 dataset showing that our approach, that only uses linear classifiers, improves over bag of visual words and is comparable to the traditional Fisher vector encoding over Gaussian Mixture Models for density estimation.**

I. Introduction

Image classification, that is predicting the category of objects present in an image, is a fundamental problem in computer vision. Much progress has been made in recent years, and most of the improvements in the state-of-the-art have been obtained with the Bag-of-Words (BoW) approach [1]. A BoW pipeline usually consists of the following steps: feature sampling, dictionary learning, feature encoding and pooling and classification. In a BoW pipeline the dictionary is usually created with unsupervised learning techniques such as K-means or Gaussian Mixture Models. The dictionary allows mapping of local features from a set of any cardinality onto a fixed number of words. With this encoding step the final image signature has a fixed dimensionality and is easily used in supervised classifiers like the Support Vector Machine (SVM).

Local image patches are represented with invariant descriptors like SIFT [2]. Once descriptors are extracted, the final image signature is computed by first encoding each descriptor according to the learned dictionary and then by aggregating high-dimensional descriptor codes into a global descriptor. We refer to the first step as *encoding* and to the second step as *pooling*. Various encoding techniques can be used; among them, the most common are Histogram Encoding [1], Kernel Encoding [3], Super Vector Encoding [4], Locality-Constrained Linear Encoding [5] and Fisher Encoding [6]. The most common pooling operations are max-pooling and average-pooling. Max pooling is normally used with sparse-coding approaches [7]. Average pooling is used for Histogram encoding and for soft-assignment methods like [3] and Fisher coding [6]. The pooling operation allows encoding of spatial information in the final representation by aggregating descriptors depending on their locations. The first work to propose this approach is [8].

Fisher Encoding is emerging as a very effective way to encode visual descriptors. The main idea is to first use a generative model of visual word formation. Then each patch descriptor is encoded by taking into account the first and second order statistics of the contribution of all the words in the vocabulary when representing a point in the feature space, resulting in a more powerful representation than histogram encoding derived from K-means clustering. This approach represents the current state-of-the-art in image classification, but with the major drawback that the final image descriptor can become enormous. This is a limitation in terms of memory and computational efficiency. Therefore, in this paper we address this issue by using compact and effective Fisher Encoding representation that offers a tradeoff between dimensionality and precision. Recently deep convolutional neural network [9] shown state-of-the-art results with the only drawback of requiring massive amounts of labeled data to pre-train. Our main contribution is the use of Random Density Forests to build the generative model for the codebook in place of the Gaussian Mixture Model. Random Decision Forests have already been used for clustering in a few image classification works, like [10], but until now have not been used as generative models in conjunction with Fisher Encoding. Random Density Forests are a computationally efficient way to partition feature space.

In the next section we briefly summarize existing works on Random Decision Forests and Fisher vectors in the literature. In section III we describe our framework for using RDFs for Fisher Encoding. In section IV we analyze a series of experiments we performed to evaluate the potential of our approach, and we conclude in section V with a discussion of our contribution.

II. Related Work

In this section we review some recent results on Fisher vector encoding for image recognition and Random Decision Forests for classification and density estimation.

Fisher vectors for image recognition Assuming an image is represented as a set of local patch descriptors, Fisher vectors characterize the input features by taking the gradient of a generative model. In other words, they describe image samples by their deviation from an universal generative model. In most cases the generative model is a Gaussian Mixture Model, but other generative models have been considered in the same framework [11].

The Fisher vector image encoding technique is mature and has been used for large scale experiments of classification and retrieval [6] obtaining higher accuracy with respect to Bag-of-Words on most datasets. Moreover, classification using Fisher vectors is efficient as simple linear classifiers are used.

The major drawback of Fisher vectors is their density and size, resulting in a large memory footprint. Large scale experiments, especially on retrieval problems, demand small footprints and therefore some kind of compression techniques must be used. Fisher vectors can be effectively compressed without significantly affecting their performance [6]. A detailed comparison of different encoding techniques highlights that Fisher vectors achieve the best performances [12].

Random Decision Forests Random Decision Forests were first proposed by Leo Breiman as a combination of independent tree predictors, which have been shown to be very robust compared to other prediction techniques [13]. Though they are not widely used for image classification, RDFs have proven to be very good for tracking and pose estimation [14]. More broadly, they are a very flexible model which can be used for classification, regression, density estimation, as well as manifold learning, semi-supervised learning and active learning [15].

Random Decision Forests are a very flexible model which can address various kinds of problems. They have been used in a structured output learning in semantic segmentation and obtain state-of-the-art results [16]. Recently they have also been proposed for edge detection. In this case RDFs take advantage of the structure present in local image patches to learn both an accurate and computationally efficient edge detector [17]. In action recognition RDFs can be adapted to perform a generalized Hough transform in an efficient way. Their flexibility permits extensions of the Hough transform to new domains such as object tracking and action recognition [18].

A variant of Random Decision Forests, known as Random *Density* Forests, can be used as a generative model and can be considered as a generalization of GMM that create different partitionings of the feature space. In a Density Forest every point in the space can be characterized by multiple clusters, in contrast to the linear Gaussian Mixture Model. Using Random Decision Trees as clustering technique, it is also possible to create discriminative visual vocabularies [19].

III. Fisher vectors over random density forests

Our work is based on a combination of RDF as generative model with Fisher vector encoding. We will also show that we can use the structure of trees in the Random Density Forest in order to reduce dimensionality of the final descriptor while maintaining high mean average precision in the final classifier. We begin by describing the standard Fisher vector image encoding over Gaussian Mixture Models.

A. Fisher vector encoding for image recognition

Assume that we have learned a GMM representing the distribution of descriptors in feature space for a set of image patches sampled from a training set. The log-likelihood of a single point x from this GMM is:

$$\mathcal{L}(x) = \log \sum_{i=1}^{K} \omega_i \mathcal{N}(x; \mu_i, \Sigma_i), \tag{1}$$

where K is the number of Gaussian mixture components, ω_i is the weight of the i-th component, and μ_i and Σ_i are the parameters of the i-th Gaussian component. It is standard to assume diagonal covariance matrices, and thus the number of partial derivatives in the gradient is $2KD$, for x of dimensionality D. Assuming that descriptors in an image are independent, the log-likelihood of a set X of M descriptors is then:

$$\sum_{m=1}^{M} \log \left(\sum_{i=1}^{K} \omega_i p_i(x_m) \right) \tag{2}$$

The Fisher vector encoding works by calculating the gradient of this log-likelihood with respect to all model parameters. That is, the gradient with respect to all component means and covariances: We can differentiate the log-likelihood with respect to means and variances of the Gaussian functions in the mixture, for the entire set of input points X:

$$\nabla \mathcal{L}(X) = \sum_{m=1}^{M} \nabla \log \left(\sum_{i=1}^{K} \omega_i \mathcal{N}(x; \mu_i, \Sigma_i), \right) \tag{3}$$

For the Gaussian model these gradients reduce to:

$$\mathcal{G}_n^{\mu}(X) = \frac{1}{\sqrt{\omega_n}} \sum_{m=1}^{M} \gamma_{x_m}^{(n)} \left(\frac{x_m - \mu_n}{\sigma_n^2} \right) \tag{4}$$

$$\mathcal{G}_n^{\sigma}(X) = \frac{1}{\sqrt{2\omega_n}} \sum_{m=1}^{M} \gamma_{x_m}^{(n)} \left(\frac{(x_m - \mu_n)^2}{\sigma_n^2} - 1 \right), \tag{5}$$

where $\gamma_{x_m}^{(n)}$ is the probability that the point x_m has been generated by the n-th Gaussian component.

$$\gamma_{x_m}^{(n)} = \frac{\omega_n p_n(x_m)}{\sum_{j=1}^{N} \omega_j p_j(x_m)} \tag{6}$$

The final Fisher vector for the set of descriptors X is then obtained as a concatenation of $\mathcal{G}_n^{\mu}(X)$ and $\mathcal{G}_n^{\sigma}(X)$:

$$\mathcal{G}(X) = \left[\mathcal{G}_1^{\mu}(X), \mathcal{G}_1^{\sigma}(X), \quad \ldots \quad \mathcal{G}_K^{\mu}(X), \mathcal{G}_K^{\sigma}(X) \right]. \tag{7}$$

The dimensionality problem If we apply Fisher vector encoding to a Gaussian Mixture Model with K components and features of dimensionality D, we obtain descriptors of length $2 \times K \times D$ ($K \times D$ for the partial derivatives with respect to mean and $K \times D$ for the partial derivatives with respect to

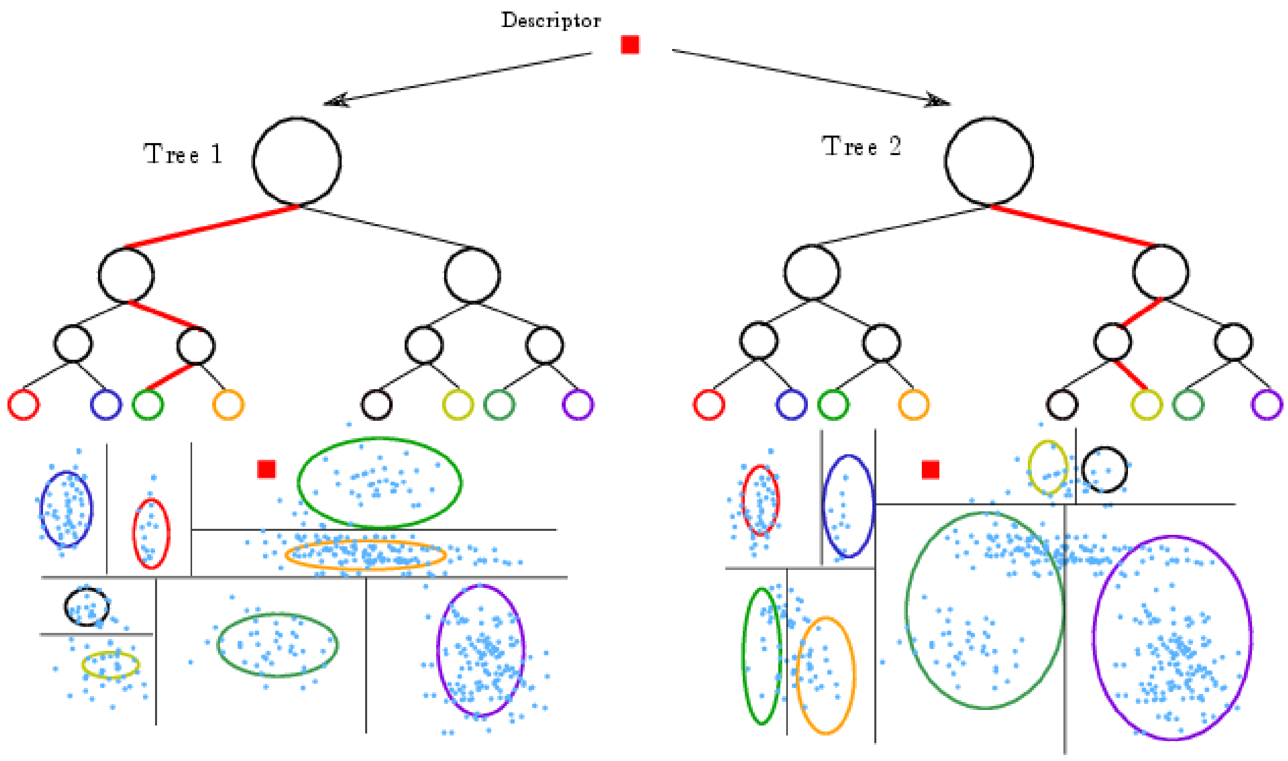


Figure 1: Visualization of a random density forest for a 2D toy space, with two trees of depth three. Descriptor path along trees is highlighted in red. Each Gaussian is colored as the respective leaf in the tree.

variance parameters in the diagonal covariance matrices). For a GMM with 128 components and features of 128 dimensions, this yields descriptors of approximately 32,000 dimensions.

Principal Component Analysis (PCA) can be used to partially address this problem by projecting the descriptors onto a subspace of lower dimensionality. However, this strategy is not completely effective since if a very small number of dimensions are retained this can significantly reduce the information carried by the features and their effectiveness for classification.

B. Random density forests for density estimation

In this section we describe how Random Density Forests can be used to learn a generative model of the distribution of patch descriptors in feature space. As with GMMs, RDFs result in a linear combination of Gaussian densities The main difference is that to describe a point in the feature space, a GMM takes in account all Gaussians of the mixture, while in an RDF we consider only the Gaussians at the leaves to which a point is assigned in each tree. An RDF is build by recursively splitting a set of input feature descriptors in a way that maximizes the information gain, or equivalently minimizing the differential entropy of a Gaussian fit to the descriptors, in the resulting split sets.

In Fig. 1 we show a visualization of how the likelihood of a descriptor is be computed with from a trained random density forest. In this toy example, the descriptor (red square) is explained by the weighted likelihood of the two leaf Gaussians (one for each tree). In this example the descriptor is better explained by the leaf in Tree 2. The final likelihood is computed as in Eq. (10) by averaging all the leaves at the end of each path in each tree of the forest.

Splitting functions Given a set of input points $S = \{s_1, s_2, \cdots, s_P\}$ of dimensionality d ($d = 128$ for SIFT descriptors, for example) we select the best split parameter θ_j^* for the j-th node by maximizing an information gain function $\theta_j^* = \arg\max_{\theta_j \in T_j} I_j$, where I_j is the generic information gain for splitting set $\mathcal{S}_j$ into two subsets $\mathcal{S}_j^L$ and $\mathcal{S}_j^R$:

$$I_j = |H(\mathcal{S}_j)| - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} \log |H(\mathcal{S}_j^i)|, \tag{8}$$

and $H(\mathcal{S})$ is the Shannon entropy of set $\mathcal{S}$. Since we work with multivariate Gaussian densities, we use the differential entropy of the d-dimensional Gaussian:

$$H(\mathcal{S}_j) = \frac{1}{2} \log \left[(2\pi e)^d |\Lambda(\mathcal{S}_j)| \right], \tag{9}$$

where $\Lambda(\mathcal{S}_j)$ is the $d \times d$ covariance matrix for the j-th Gaussian, and $|\Lambda(\mathcal{S}_j)|$ is the determinant of $\Lambda(\mathcal{S}_j)$.

The split used at each node of each tree is determined by a random sampling process. We randomly generate a fixed number of candidate splits and choose the one that maximizes Eq. 8. Splitting the set of descriptors using this rule guarantees that with each progressive level of the tree we can better explain the resulting splits with Gaussians. Note that this split candidate generation process is extremely flexible. It

Algorithm 1: `trainTree`($\mathcal{S}$)

```
Data: Training set $\mathcal{S} = \{x_0, \ldots x_n\}$, where $x_i \in \mathbb{R}^d$
      Depth $D$ of subtree being constructed
Result: The (sub)tree for the training point set $\mathcal{S}$
1  foreach $t \in \{1, \ldots, \mathrm{maxTries}\}$ do
2      $\hat{d}_t \leftarrow \{d | d \sim \mathcal{U}_{\{1,\ldots,d\}}\}$
3      $\hat{v}_t \leftarrow \{v | v \sim \mathcal{U}_{[\min(x^{\hat{d}_t}), \max(x^{\hat{d}_t})]}\}$
4      $\mathcal{S}_t^L \leftarrow \{x \in S | x^d \leq v\}$
5      $\mathcal{S}_t^R \leftarrow \{x \in S | x^d > v\}$
6      $I_t \leftarrow |H(\mathcal{S})| - \sum_{i \in \{L,R\}} \left(|\mathcal{S}_t^i| / |\mathcal{S}|\right) \log |H(\mathcal{S}_t^i)|$
7      $P_t \leftarrow \min(|\mathcal{S}_t^L|, |\mathcal{S}_t^R|)$
8  $t^* \leftarrow \arg\max I_t$
9  if $D < D_{\max} \wedge P_{t^*} > P_{\max} \wedge I_{t^*} > I_{\min}$ then
10     $T_L \leftarrow$ trainTree($\mathcal{S}_{t^*}^L$)
11     $T_R \leftarrow$ trainTree($\mathcal{S}_{t^*}^R$)
12 else
13     $C \leftarrow \mathbf{E}\left[(\mathcal{S} - \mathbf{E}[\mathcal{S}])(\mathcal{S} - \mathbf{E}[\mathcal{S}])^T\right]$
14     $V \leftarrow$ the first $R$ principal Eigenvectors of $C$
15     $S_V \leftarrow (\mathcal{S} - \mathbf{E}[\mathcal{S}])^T V$
16     $\mu_\ell \leftarrow \mathbf{E}[\mathcal{S}_\ell]$
17     $T_L \leftarrow \emptyset$
18     $T_R \leftarrow \emptyset$
19     $\Sigma_\ell \leftarrow \mathbf{E}\left[(\mathcal{S}_\ell - \mathbf{E}[\mathcal{S}_\ell])(\mathcal{S}_\ell - \mathbf{E}[\mathcal{S}_\ell])^T\right]$
20 return $(V_r, \mu_\ell, \Sigma_\ell, T_L, T_R)$
```

can be used to split on multiple features, for example, or to explicitly favor splits in particular dimensions. In combination with the definition of information gain, it can also be used to incorporate discriminative information into the feature space partitioning model.

Stopping criterion To avoid obtaining very deep trees, which could overfit the data, we define a criterion to stop the splitting process. We use a combination of several criteria:

- **Minimum Threshold on Gain value:** if the information gain in a node drops below a specified value $I_{\min}$, we do not perform node splitting as it would not improve precision.
- **Maximum Tree Depth:** to avoid a high number of leaves and subsequent overfitting, we halt splitting if the depth exceeds a maximum depth $D_{\max}$.
- **Minimum Split Cardinality at Leaves:** in order to balance the number of points assigned to each leaf we also stop splitting if the minimum cardinality of child split sets drops below a threshold $P_{\max}$. If a split is unbalanced, it is possible that the feature space is already well partitioned.

We combined the above stop criteria. These conditions must be respected at the same time, so we prevent node splitting if either information gain or number of points in a leaf are under the threshold value or if tree depth grows over the maximum value after the split evaluation.

The log-likelihood function for RDFs Given a point x and a tree t, we denote the leaf that x reaches by passing it through the sequence of splitting functions as $\ell_t(x)$. Consequently, given a forest F of trees, the likelihood of a point x according to the RDF generative model is:

$$\mathcal{L}(x|\mathcal{F}) = \log \frac{1}{|\mathcal{F}|} \sum_{t \in \mathcal{F}} \pi_{t,\ell(x)} \mathcal{N}\left(x; \mu_{t,\ell(x)}, \Sigma_{t,\ell(x)}\right) \quad (10)$$

Note the similarity between Eq. (10) and Eq. (1). The main difference is that in an RDF, exactly one Gaussian contributes to the final likelihood (the one fit to the training examples landing in leaf $\ell_t(x)$).

C. Fisher vectors encoding from an RDF

We now show how to compute the Fisher vector from the log-likelihood derived from an RDF model of descriptor distribution. As described in the previous section, a leaf of a trained RDF will contain mean and diagonal covariance of the training points which are assigned to it. Using the information contained in the leaves, the Fisher vectors for a single descriptor depends on the parameters of a single leaf Gaussian from each tree.

Recall that $\ell_t(x)$ is the leaf where feature x arrives in tree t. To calculate the Fisher vector of a set of descriptors we group the descriptors into sets according to how they fall into the leaves of each tree:

$$X_l^t = \{x \in X | \ell_t(x) = l\}, \quad (11)$$

so that X_l^t is the set of all descriptors from X that fall into leaf l in tree t. For a given tree t, the Fisher vector components for leaf l are:

$$\mathcal{G}_l^\mu(X_l^t) = \omega_{\ell_t(x)}^{-\frac{1}{2}} \sum_{x \in X_l^t} \gamma_{x_m}^{\ell_t(x)} \left(\frac{x_m - \mu_{\ell_t(x)}}{\sigma_{\ell_t(x)}^2}\right) \quad (12)$$

$$\mathcal{G}_l^\sigma(X_l^t) = (2\omega_{\ell_t(x)})^{-\frac{1}{2}} \sum_{x \in X_l^t} \gamma_{x_m}^{\ell_t(x)} \left(\frac{(x_m - \mu_{\ell_t(x)})^2}{\sigma_{\ell_t(x)}^2} - 1\right) \quad (13)$$

The partial Fisher vector corresponding to leaf l of tree t is then:

$$\mathcal{G}(X_l^t) = \left[\mathcal{G}_l^\mu(X_l^t), \mathcal{G}_l^\sigma(X_l^t)\right] \quad (14)$$

Iterating over all leaves and trees we concatenate each of these partial Fisher vectors to obtain the Fisher vector of the whole forest for a given image X:

$$\mathcal{G}(X) = \left[\mathcal{G}(X_1^1), \ldots, \mathcal{G}(X_l^t), \ldots, \mathcal{G}(X_L^T)\right] \quad (15)$$

Making compact Fisher vectors over RDFs As with the standard Fisher vector over GMMs, we can apply PCA to the set training points to reduce the dimensionality of the final descriptor (at the cost of descriptive power). However, RDFs are based on a hard partitioning of training data into the leaves of each tree, and since the splits at nodes are based on entropy minimization, we can fit a smaller PCA model to each set of *leaf node descriptors* instead of to the entire input set. At

Algorithm 2: computeFisherVector($\mathcal{F}$, $\mathcal{I}$)

```
Data: Previously trained Random Decision Forest $\mathcal{F}$,
      Test patches set $\mathcal{I} = \{x_1, \cdots, x_n\}$
Result: Fisher vector of $\mathcal{I}$ over $\mathcal{F}$
1 foreach $t$ in $\mathcal{F}$ do
2     foreach $\ell$ in $L_t$ do
3         $\mathcal{G}(X_f^t) \leftarrow [\mathcal{G}_l^{\mu}(X_l^t), \mathcal{G}_l^{\sigma}(X_l^t)]$          (Eq. (14))
4 $\mathcal{G}(\mathcal{I}) \leftarrow [\mathcal{G}(X_1^1), \ldots, \mathcal{G}(X_l^t), \ldots, \mathcal{G}(X_L^T)]$          (Eq. (15))
5 return $\mathcal{G}(\mathcal{I})$
```

each leaf node, before we fit the Gaussian density to the set of points landing in it, we project all points onto the first R principal Eigenvectors of their covariance matrix.

As with the standard Fisher vector encoding over GMMs, the dimensionality of Fisher vectors over RDFs encoded using Eq. (15) can be similarly excessive. The dimensionality is $2 \times T \times R \times 2^d$, where T is the number of trees, d is the depth each tree, and R is the number of principal eigenvectors retained at leaves (and thus the final dimensionality of descriptors to which each Gaussian is fit).

Putting it all together In algorithm 1 we give an algorithm for construction of trees in our Random Density Forests. The algorithm takes as input a set of training points $\mathcal{S}$, generates random splits of $\mathcal{S}$, and depending on whether or not the stopping criteria are met it recursively subdivides $\mathcal{S}$ into smaller subtrees or fits a PCA model with R components and a Gaussian of R dimensions to the set $\mathcal{S}$.

Algorithm 2, on the other hand, details the construction of a Fisher vector over a trained RDF. As described in section III-C we use the RDF to split the descriptors $\mathcal{I}$ from and image into the leaves of all trees. The partial Fisher vectors are computed at all leaves from each tree, and these are finally concatenated into the Fisher vector representation for the image.

IV. Experimental results

We performed a series of image classification experiments on the PASCAL VOC 2007 dataset to evaluate the potential of RDFs as generative models for Fisher vector encoding.

A. Datasets and experimental protocol

In all experiments we extract patches densely with a stride of 4 pixels at four scales: 16x16, 24x24, 32x32 and 40x40. After calculating Fisher Vectors, we apply power normalization with α parameter set to 0.5 and after we perform ℓ_2 normalization as suggested in [6].

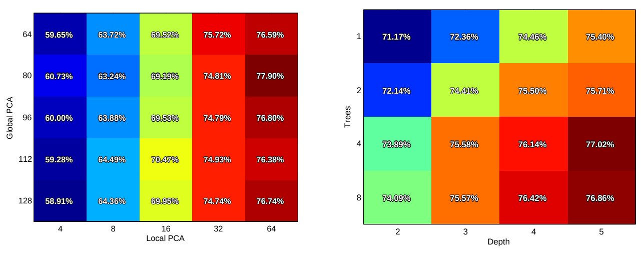
We then build one-versus-all linear SVM classifiers for all classes. The C parameter is selected by crossvalidation the mAP on the validation set and the final result is reported as the mAP on test images with training performed on the merged training and validation sets. We do not use a spatial pyramid for our final image representation.

To initially evaluate method during development, we reduced the original dataset to a subset of six classes (bottle, bus, cat, motorbike, pottedplant, sheep).

Vocabulary	Encoding	mAP	Trees	Depth	DIM
GMM	Fisher vector	56.25	-	-	16,384
RDF	Fisher vector	54.12	4	5	16,384
K-means	Hard BoW	47.63	-	-	16,384
RDF	Hard BoW	46.68	32	9	16,384
RDF	Hard BoW	44.85	32	7	4,096

Table I: Mean average precision and final descriptor dimensionality on entire PASCAL VOC 2007 for Fisher vectors over RDF, Fisher vectors over GMM and BoW with hard assignment. All experiments are without spatial pyramid.

B. Baseline evaluation



(a) (b)

Figure 2: Mean average precision on our reduced development set varying global/local PCA (a) and number of trees/depth (b).

We performed some preliminary tests to gain insight on the behavior of our algorithm. To initially evaluate method during development, we performed a first test on a reduced set containing six the original twenty PASCAL VOC classes: bottle, bus, cat, motorbike, pottedplant, sheep. This test is performed with forest size set to 4 trees and $D_{\max}$ set to 5 levels (2^5 leaves if the tree is perfectly balanced). We focused on the effects of PCA, both globally applied on the whole set of descriptors and locally applied on the leaves. We see in Fig. 2a that global PCA (GPCA) reduction has little or no effect on the quality of the learned dictionary while the local PCA (LPCA) influences both the final descriptor size and quality of the Gaussians estimated at each leaf.

In this second test we fixed the GPCA and LPCA to 128 and 64, respectively, to analyze the behavior of our algorithm varying the number of trees and their depth. This and all subsequent tests were performed on the entire set of twenty PASCAL VOC classes. In Fig. 2b we observe that these two parameters are quite complementary. Note that adding a tree to the forest will double the final descriptor size, while adding a level to each tree will increase by a factor of $2 \times T$. Since trees are trained in a randomized way, adding trees is like adding novel views of the same data. On the other hand, increasing depth makes a more detailed partitioning of the feature space. Excessively increasing in both forest size and tree depth may lead to overfitting. We can also hypothesize that a bigger forest may need a smaller depth of its trees to be effective, as having many different partitionings can compensate for their coarseness.

We performed another set of experiments to evaluate our

Trees	Depth	GPC	LPC	DIM	mAP
4	**6**	**128**	**64**	**32,768**	**55.26**
8	5	128	64	32,768	55.14
4	5	128	64	16,384	54.12
4	5	128	32	8,192	50.59

Table II: Mean average precision on entire PASCAL VOC 2007 for different configuration of our algorithm.

Method	mAP	kernel	SPM	DIM
Our approach	55.26	linear	-	32k
Our approach	54.12	linear	-	16k
IFV [20]	55.30	linear	-	32k
LLC [12]	53.79	linear	✓	32k
BOW [12]	53.47	χ^2	✓	32k
KCB [12]	54.60	χ^2	✓	32k

Table III: Comparison with the state-of-the art on the entire PASCAL VOC 2007 dataset.

approach compared to standard Fisher vector encoding over GMMs and BoW encoding with hard assignment. In order to use RDFs for BoW encoding, we modified our framework to build histograms of points hard assigned to tree leaves, using them as histogram bins and not as containers of Gaussian parameters. In Table I we report the mean average precision of our approach in various configurations. Note how Fisher vectors over RDFs are comparable to the standard Fisher vector encoding, with BoW encoding using K-means or RDFs for vocabulary construction is significantly worse at comparable dimensionality.

A final baseline comparison was performed to understand the best configuration of RDF along with global and local PCA using all the 20 classes of PASCAL VOC2007. In table II we report results over varying parameter settings.

C. Comparison with the state-of-the-art

Finally we compared our approach to state-of-the-art BoW techniques for image classification on PASCAL VOC2007. In Table III we report a comparison between our approach and techniques from the literature using only SIFT features and no spatial pyramid. Our approach performs similarly to the Improved Fisher Vector [20] at the same dimensionality, while at half the dimensionality of the Improved Fisher Vector we lose only about 1% in mean average precision.

V. Discussion

In this paper we described an approach to Fisher vector encoding using Random Density Forests as an underlying generative model of descriptor distribution in feature space. With respect to the Gaussian Mixture Models used in the standard Fisher vector encoding for image recognition, Random Density Forests offer a number of advantages. The hard assignment of training descriptors to leaves allows us to fit leaf-specific PCA models to the descriptors in each leaf node. Since the random trees are learned such that entropy in leaves is minimized, we can use smaller leaf-local PCA subspaces than can be used for global PCA as in the standard Fisher vector approach.

Experiments on the PASCAL VOC 2007 dataset demonstrate the potential of our approach. Fisher vectors over RDFs perform comparably to the standard and improved Fisher vector encoding over GMMs. Furthermore, at half the dimensionality obtained through local PCA, our approach only performs about 1% below the state-of-the-art. Our ongoing work is focused on better understanding how to guarantee diversity in trees in order to obtain richer information from the final Fisher vector encoding.

References

[1] J. Sivic and A. Zisserman, "Efficient visual search cast as text retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 591–606, 2009.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] J. C. van Gemert, C. J. Veenman, A. W. Smeulders, and J.-M. Geusebroek, "Visual word ambiguity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1271–1283, 2010.

[4] X. Zhou, K. Yu, T. Zhang, T. S. Huang, and T. S. Huang, "Image classification using super-vector coding of local image descriptors." in *ECCV*, 2010, pp. 141–154.

[5] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. of CVPR*, 2010.

[6] F. Perronnin, J. Sanchez, T. Mensink, and J. Verbeek, "Image classication with the fisher vector: Theory and practice," in *International Journal of Computer Vision*, 2012.

[7] Y.-L. Boureau, J. Ponce, and Y. Lecun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. of ICML*, 2010.

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

[9] I. L. J. S. Maxime Oquab1, Leon Bottou, "Learning and transferring mid-level image representations using convolutional neural networks."

[10] F. Moosmann, E. Nowak, and F. Jurie, "Randomized clustering forests for image classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 9, pp. 1632–1646, sept. 2008.

[11] R. G. Cinbis, J. Verbeek, and C. Schmid, "Image categorization using fisher kernels of non-iid image models," in *Proc. of CVPR*, 2012.

[12] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," in *Proc. of BMVC*, 2011.

[13] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[14] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from a single depth image," in *Proceedings of CVPR*, 2011.

[15] A. Criminisi, J. Shotton, and E. Konukoglu, *Decision Forests for Classification, Regression, Density Estimation,Manifold Learning and Semi-Supervised Learning*.

[16] P. Kontschieder, S. R. Bulò, H. Bischof, and M. Pelillo, "Structured class-labels in random forests for semantic image labelling." in *ICCV*, D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, Eds. IEEE, 2011, pp. 2190–2197.

[17] P. Dollár and C. L. Zitnick, "Structured forests for fast edge detection," in *ICCV*, 2013.

[18] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2188–2202, Nov. 2011.

[19] F. Moosmann, B. Triggs, and F. Jurie, "Fast discriminative visual codebooks using randomized clustering forests," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. Cambridge, MA: MIT Press, 2006, pp. 985–992.

[20] F. Perronnin, J. Sanchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proc. of ECCV*, 2010.