Sparse Discriminative Fisher Vectors in Visual Classification

Vinay Garg[*] vinay.garg@research.iiit.ac.in
Siddhartha Chandra siddhartha.chandra@research.iiit.ac.in
C. V. Jawahar jawahar@iiit.ac.in
Center for Visual Information Technology, IIIT-Hyderabad

ABSTRACT

Constructing global image representations from local feature descriptors is a common step in most visual classification tasks. Traditionally, the Bag of Features (BoF) representations involving hard vector quantization have been used ubiquitously for such tasks. Recent works have demonstrated superior performance of soft assignments over hard assignments. Fisher vector representations have been shown to outperform other global representations on most benchmark datasets. Fisher vectors (i) use soft assignments, and (ii) reduce information loss due to quantization by capturing the deviations from the mean. However, the Fisher vector representations are huge and the representation size increases linearly with the vocabulary size. Recent findings report that the classification performance of Fisher vectors is proportional to the vocabulary size. Computational and storage requirements, however, discourage the use of arbitrarily large vocabularies. Also, Fisher vectors are not inherently discriminative. In this paper, we devise a novel strategy to compute sparse Fisher representations. This allows us to increase the vocabulary size with little computation and storage overhead and still attain the performance of a larger vocabulary. Further, we describe an approach to encode class-discriminative information in the Fisher vectors. We evaluate our method on four popular datasets. Empirical results show that our representations consistently outperform the traditional Fisher Vector representations and are comparable to the state of art approaches.

Keywords

Visual Classification, Fisher Vectors, Sparse Representations, Discriminative Features

1. INTRODUCTION

Recognition of visual artefacts is one of the most popular problems in the Computer Vision community. Common tasks such as visual object recognition, image retrieval and scene classification, all involve assigning class labels to images, either implicitly or explicitly. Researchers generally agree on a three fold approach (Figure 1) to tackle such problems [3]: (a) computing low level feature descriptors over small image parts, (b) pooling these local descriptors to arrive at a global image representation of some sort, and (c) using a classification method that learns the underlying distribution of the seen examples from their global representations and uses this knowledge to predict the class labels of the unseen examples. Over the years, much research has gone into improving the individual stages of the pipeline.

Feature learning has evolved steadily over the past few decades and is perhaps the most stable component of the pipeline. Dense SIFT has emerged as a popular local descriptor for most classification tasks [10]. Meanwhile, majority of the efforts in this field have been focused on improving the feature pooling stage [3, 2]. The Bag of Features approach, where local features in each image are vector quantized (using K-Means on the training set features) and counted to arrive at a histogram image representation has enjoyed reasonable success for years. Some recent approaches have used soft assignments, as opposed to the traditional hard quantization in K-Means by expressing features as linear combinations of visual words [7]. Other approaches have aimed at minimizing loss of information by encoding the quantization errors in the representation [14, 27]. Finally, classification methods have seen their share of innovations too [20].

Fisher Kernels, which were introduced in 1998 [8], have achieved huge success after their introduction to the image classification domain [14]. Fisher encoding assumes the data distribution is a Gaussian Mixture Model (GMM) and captures the discrepancy in the fit. GMM can be understood as a soft visual vocabulary. Most state of the art approaches on various datasets employ Fisher encodings [3]. However, this performance comes at a huge computational and storage price. A fisher vector for K Gaussian components has a final representation size of $K(2D+1)$, where D is the dimension of the local features, and requires computations of the same order. Remedies that involve compressing Fisher vectors [15] and methods such as Product Quantization [9, 18] have been suggested to cope up with the high memory requirements. While most of these methods compress Fisher vectors to reduce storage requirements and uncompress these on the fly during classification, [22] demonstrates that the Product Quantization code itself can be used for classification. Recent advances into this field have involved fine tuning the

[*]Corresponding author

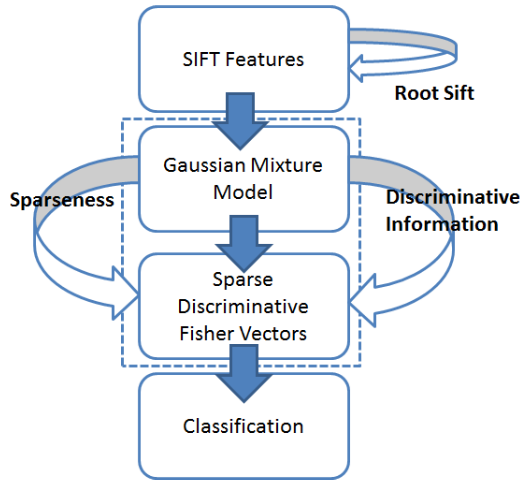


Figure 1: The Fisher vector computation pipeline. The loopy arrows mark our contributions.

model parameters to incorporate class-discriminative information [5].

Some of the recent literature [6, 2, 24] has emphasized the success of sparse representations in classification. Sparse representations are said to be more discriminative, better representative and since the introduction of certain data structures come at little computational and storage price.

In this paper, we devise a simple approach to introduce sparseness in the Fisher vector representation. This was motivated by two objectives (a) to reap the performance advantages of the sparse representation in classification, and (b) to reduce the representation size and computational cost of the feature encoding procedure. In section 3.2 we describe an implementation trick that helps us achieve objective (b). Our approach was inspired by [23], where locality constraints were enforced to compute a sparse image representation. We however use a different scheme to introduce sparsity in our representation. Another difference is that while [23] computes the best describing visual words for each low level feature of an image, we find the best representative visual words for an image or class. We also devise a novel strategy for class-discriminative encoding of Fisher vectors that identifies the Gaussian components most representative of each class and weights them appropriately to arrive at a discriminative representation. We demonstrate the superior performance of our approach over the traditional Fisher vector representations on several datasets. On all these datasets, we either outperform the state of the art or produce results that are comparable to the corresponding state of the art approaches. Figure 1 describes a block diagram of the Fisher vector computation procedure. The loopy arrows mark the modifications we have made to the pipeline. These modifications have been elaborated in section 3.

2. FISHER KERNEL

Pattern classification models can be broadly divided into *"Generative Models"* and *"Discriminative Models"*. While generative models randomly generate input data typically given some hidden variables, discriminative models predict models for the unseen samples based on the distribution learnt from the training examples. Fisher Kernel [8] combines the advantages of generative statistical models (like GMM, Hidden Markov Model) and those of discriminative methods (like SVM). Recently Fisher kernels have been extensively used for various computer vision and machine learning tasks like retrieval and classification [14]. We now describe the Fisher kernel formulation.

Let $P(x_n|\theta)$ denote the generative probability distribution model, θ being the model parameters. $X = \{x_n, n = 1, \dots, N\}$ represents the data item set and x_n is a data item. Let ∇_θ be the gradient function with respect to θ and $\log p(x_n|\theta)$ be the log-likelihood of x_n with respect to the model given the set of model parameters θ. Now for each data item we can define the Fisher score, F_{x_n} as the gradient of the log likelihood of x_n with respect to model paramters θ which is given by

$$F_{x_n} = \nabla_\theta \log p(x_n|\theta) \tag{1}$$

The Fisher score gives us the direction in which parameters should be modified to best fit the data. It transforms the variable length data into a N-dimensional feature space $\mathbb{R}^N$. Fisher kernel is defined by

$$K(x_i, x_j) = F_{x_i}^T I^{-1} F_{x_j} \tag{2}$$

where I is the Fisher information matrix, which is used for normalizing the gradient vectors. Because of its cost of computation and inversion, various approximations of I have been proposed in literature. We discuss some of these in section 2.1. Comparison of two data items is done by directly comparing their Fisher scores. If their Fisher scores are similar it implies that they would require similar adaptations to the model parameters and hence they are similar. Thus, we can directly classify these gradient vectors in place of the data items using any discriminative classifier such as SVM.

2.1 Fisher Vector Image Representation

Computing the gradient vector for each image involves computing a Gaussian Mixture model learnt over the low level features (eg. SIFT). $\theta = \{w_i, \mu_i, \Sigma_i, i = 1, \dots, K\}$ denotes the model parameters of GMM with K components, where w_i, μ_i, and Σ_i represents the weight, mean and covariance matrix respectively, corresponding to the i^{th} Gaussian component. The mixture weights capture the relative frequency of each component of the Gaussian, thus, $\sum w_i = 1$. Various types of covariance matrices can be used in this formulation. We use the diagonal covariance matrix for simplicity.

Let $X = \{x_n, n = 1, \dots, N\}$ denote the set of low level features for an image, each of dimension D, and $L(X|\theta) = \log p(X|\theta)$, where L is the log likelihood of the features with respect to the model. Since the features in the feature set X, are independent of each other, we can rewrite $L(X|\theta)$ as:

$$L(X|\theta) = \sum_{n=1}^{N} \log p(x_n|\theta) \tag{3}$$

The likelihood that a feature x_n is generated by GMM is given by:

$$p(x_n|\theta) = \sum_{i=1}^{K} w_i p(x_n|\theta_i) \tag{4}$$

where,

$$p(x_n|\theta_i) = \frac{\exp\{-\frac{1}{2}(x_n - \mu_i)'\Sigma_i^{-1}(x_n - \mu_i)\}}{(2\pi^{\frac{D}{2}})|\Sigma_i|^{\frac{1}{2}}} \tag{5}$$

Also let, q_{ni} be the probability that feature x_n is generated by i^{th} Gaussian component, which will be given by

$$q_{ni} = \frac{w_i p(x_n|\theta_i)}{\sum_{j=1}^{K} w_j p(x_n|\theta_j)} \tag{6}$$

For computing the Fisher vector representation for an image we calculate the gradient $\frac{\partial L(X|\theta)}{\partial \theta_i}$ for each parameter separately and then concatenate them to get the final gradient vector. Since the number of low level features can vary from image to image, to achieve invariance with respect to the number of features in the image, the final Fisher vector representation of an image is divided by the number of features. This is also referred to as average pooling in literature.

$$F_{Image} = \frac{1}{N}\frac{\partial L(X|\theta)}{\partial \theta_i} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial L(x_n|\theta)}{\partial \theta_i} \tag{7}$$

$$\frac{\partial L}{\partial w_i} = \sum_{n=1}^{N}[\frac{q_{ni}}{w_i} - \frac{q_{n1}}{w_1}], i \geq 2 \tag{8}$$

$$\frac{\partial L}{\partial \mu_i^d} = \sum_{n=1}^{N} q_{ni}[\frac{x_n^d - u_i^d}{\Sigma_i^d}] \tag{9}$$

$$\frac{\partial L}{\partial \Sigma_i^d} = \sum_{n=1}^{N} q_{ni}[\frac{(x_n^d - u_i^d)^2}{(\Sigma_i^d)^{3/2}} - \frac{1}{(\Sigma_i^d)^{1/2}}] \tag{10}$$

$$\frac{\partial L(x_n|\theta)}{\partial \theta_i} = [\frac{\partial L}{\partial w_i}, \frac{\partial L}{\partial \mu_i}, \frac{\partial L}{\partial \Sigma_i}] \tag{11}$$

$\frac{\partial L}{\partial w_i}$ is a K-dimensional vector of the 0^{th} order statistics, $\frac{\partial L}{\partial \mu_i}$ is a KD-dimensional vector of 1^{st} order statistics, and $\frac{\partial L}{\partial \Sigma_i}$ is a KD-dimensional vector of the 2^{nd} order statistics of the gradient for each x_n. Exact derivations of $\frac{\partial L(x_n|\theta)}{\partial \theta_i}$ can be found in the appendix section of [14].

Implementation Details: Essentially we used the same Fisher vector computation pipeline as described in [11]. We start by computing SIFT features at multiple scales, from which GMM components are computed using the *yael library*. Fisher vectors representations are then computed for all the images as described above. We use the Fisher vector representations to compute the kernel matrix for SVM learning using Eq. 2. As described in [8], the information matrix I can be safely ignored, thus we replace I in Eq. 2, with an identity matrix. An empirical approximation to Fisher information matrix is shown in [11], that uses a diagonal approximation for I. It is shown that performing whitening normalization on the Fisher vectors approximates the effect of I well. Whitening normalization ensures that the final gradient vectors have zero-mean and unit-variance. As described in [16], we performed three kinds of normalizations, (a) whitening normalization, (b) power normalization, (c) L2-normalization in the same order to arrive at the final image representation.

Discussion: Computing the Fisher representation of an image involves computing K 0^{th} order terms, $K \times D$ 1^{st} order terms, and $K \times D$ 2^{nd} order terms which are calculated using Eqns. 8, 9, 10 respectively. This gives us a final representation of size $K(2D+1)$, where D is the size of each feature. For SIFT features of size $D = 128$ and $K = 50$ Gaussian components, the final image representation size is 12850. The representation size increases linearly with the number of Gaussian components. In such high dimensional spaces, SVM classifier training is nearly intractable. Hence precomputing the kernel matrix (using Eq. 2) is indispensable. Computing the kernel matrix requires $O(n^2)$ dot products and $O(n^2 \times (K(2D+1))$ computations. Recent works [3], further use 8 spatial regions which increases the representation size and computations 8 fold. In section 4.3, we demonstrate this is unnecessary; we achieve comparable performance without using spatial pyramids, which reduces our computational and storage requirements.

3. ENHANCING FISHER VECTOR REPRESENTATIONS

Computing the Fisher vector representation of images involves several independent steps. The various steps in this pipeline can be enumerated as (i) computing low level features (SIFT) for images, (ii) building a generative pdf model (GMM), (iii) building Fisher vector representation for images using GMMs and low level features for respective images, (iv) SVM Classification. We intend to introduce changes to the traditional methods of implementing these steps. Improving any one or more of these results in better performance of the complete system.

Baseline: In this paper, we treat the traditional Fisher vector representation (as described in [11, 16]) as our baseline representation and describe several ways to improve it. Our baseline representation is described in section 2.1. The classification scheme described in section 4 remains unaffected. In the rest of this section, we describe modifications / additional steps that we introduce in the feature encoding procedure. Each of these supplements the classification performance of the baseline representation. We back this claim by producing superior results on four datasets in section 4. We argue that our final representations are both more compact and more discriminative, and come at little computational and storage price. Results on the Pascal VOC 2007 dataset using the baseline Fisher vector representations are shown in Fig. 2. We also show results using our improved Fisher representations (with compatible representation sizes) alongside. Our representations significantly outperform the traditional representations.

3.1 Root Sift

Recent works on classification [22] have endeavoured to find better distance metrics in the feature space. Inspired by [1], we experimented with various kernels in the SIFT feature space and found that the Intersection and Hellinger Kernels enhance the classification performance. This is in line with the findings of [19, 1, 22]. Table 1 shows the effect of variation of the distance metric / kernel on the classification performance. The dataset here is Scene 15 and the number of Gaussian components are 25 and 50 in columns 2 and 3 respectively. This result motivated us to use a different distance metric in the SIFT feature space. Both the Intersection and Hellinger kernels can be approximated by computing corresponding explicit feature maps on the SIFT features. However, while the intersection kernel feature map

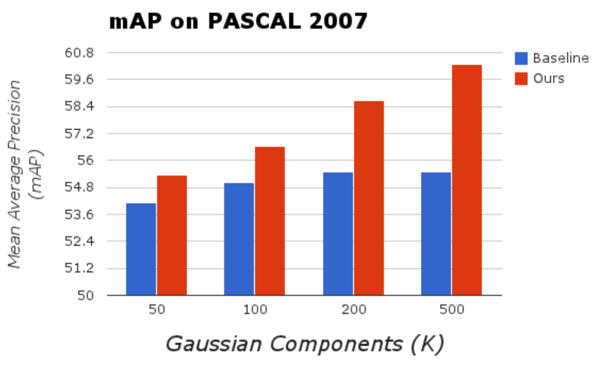


Figure 2: Baseline Results showing mAP on PASCAL VOC 2007 dataset [11]

increases the feature dimension 3 folds, the Hellinger kernel preserves the size of the input features. We therefore choose to map our SIFT features using the Hellinger feature map. As described in [22], this mapping enables us to use linear kernel to approximate the Hellinger kernel.

Table 1: Effects of different distance measures for SIFT Features on classification performance (Scene 15)

Distance Measure	$mAP(K = 25)$	$mAP(K = 50)$
Euclidean(Baseline)	85.03	86.30
χ^2	85.62	87.00
Jensen-Shannon	85.40	87.63
Intersection	86.04	87.51
Hellinger	85.96	88.18

As described in [1], this mapping consists of two simple steps: (i) L1 normalizing each SIFT feature, and (ii) Square rooting the individual values of 128 dimensional vector. Once this transformation is done, we can simply use this transformed SIFT feature namely **"RootSIFT"** [1] as it is in our pipeline.

3.2 Sparse Fisher Vector

We denote the N SIFT features in an image by the set $X = \{x_n, n = 1, \dots, N\}$ and the GMM model parameters by $\theta = \{w_i, \mu_i, \Sigma_i, i = 1, \dots, K\}$. Using Eq. 6, we compute a K dimensional feature assignment vector for each feature x_n.

$$\vec{q_n} = \{q_{n1}, q_{n2}, \dots, q_{nK}\} \tag{12}$$

The k^{th} entry in $\vec{q_n}$ (q_{nk}) represents the soft assignment value of feature x_n to the k^{th} Gaussian component. We define the image assignment vector $\vec{Q}$ as the sum of all feature assignment vectors in the image.

$$\vec{Q} = \sum_{n=1}^{N} \vec{q_n} \tag{13}$$

The k^{th} value in $\vec{Q}$ represents the sum of assignment values of all the SIFT features in that image to the k^{th} Gaussian component. Having obtained $\vec{Q}$, computing the sparse image representation involves picking k' out of the K Gaussian components. In this paper, the k' components we pick are the ones with the most population. This model assumes that the Gaussian components that occur the most in an image are the most representative of the image. This is implemented as picking the k' components that have the highest values in $\vec{Q}$. The sparseness is introduced by ignoring $K - k'$ components. In other words, we compute the Fisher vector representation for the chosen k' Gaussian components only and values corresponding to rest $K - k'$ Gaussians are ignored / set to zero. We argue that our final Fisher vector representation has a size of $k'(2D + 1)$.

Let G' be the set of these top k' Gaussians. Kindly note that we have to update our Gaussian weights w_i to ensure they add up to 1 after this modification (as described in section 2.1). $\sum_{i \in G'} w_i = 1$. This is achieved using Eq. 14

$$w_i = \frac{Q_i}{\sum_{j \in G'} Q_j} \tag{14}$$

In the next section, we describe an alternate strategy to pick the k' Gaussian components out of K. The alternate strategy allows us to encode class-discriminative information in the Fisher vectors and hence is more useful.

Implementation Details: We now describe an implementation trick that enables us to reduce the storage requirements and computations for our sparse representation. Having computed the G', we compute the Fisher vector representations using Eq. 11. From Eqns. 9 and 10, it is clear that if $q_{ni} = 0$, the gradient value for that particular Gaussian is also zero. Since all but the values corresponding to the top k' Gaussians are zero, we dont have to explicitly compute all of them. Instead, for each image we save the indices of Gaussians for which $q_{ni} \neq 0$. Now we compute the Fisher vectors for G'. Our final image representation is thus of size $2k'[D + 1]$ (the $k'[2D + 1]$ values for Fisher vector representation corresponding to k' Gaussian components and an additional k' for storing indices of non-zero q_{ni}). However, implicitly our representation is that corresponding to K.
By removing these least representative Gaussians for an image we gain two advantages:

1. Since our implicit representation for an image is sparse with only $k'(2D+1)$ non-zero values, while computing kernel $k(F_i, F_j)$, we will have to make lesser computations (only for non-zero values). This can be achieved with a little implementation trick, where we first compute the intersection of the non-zero indices of F_i, F_j and then compute the dot products of the intersecting components since the other products all result in zeros.

2. Though our computations are done only on the non-zero values, the implicit length of Fisher vector still corresponds to the original K. By choosing appropriate the value of k' judiciously, we can achieve performance comparable to K by using only k' Gaussian components.

3.3 Discriminative Fisher Vectors

In this subsection, we describe a simple approach that encodes class-discriminative information in the Fisher Vector representation of images. Our approach involves computing mutual information (M.I.) [1] between the class labels and the Gaussian components. Given C classes, we define the class assignment vector $(\phi_c, c = 1, \ldots, C)$ as the distribution of the gaussian components in images corresponding to the class. Let $\vec{Q_t}, t = 1, \ldots, T$ be the image assignment vector (computed using Eq. 13) corresponding to an image t. Let $\mathbf{t}_c$ be indices t of images belonging to class c. ϕ_c is computed as

$$\phi_c = \sum_{t \in \mathbf{t}_c} \vec{Q_t} \qquad (15)$$

Computing M.I. between class labels and the Gaussian components involves computing all $\phi_c, c = 1, \ldots, C$. Let ϕ_{ck} denote the entry in ϕ_c corresponding to the k^{th} Gaussian component. The M.I. I_{ck} between class c and the Gaussian component k is computed as:

$$I_{ck} = \phi_{ck} \log \left(\frac{\phi_{ck}}{\sum_{c=1}^{C} \phi_{ck} \sum_{k=1}^{K} \phi_{ck}} \right) \qquad (16)$$

We use the M.I. values as multiplicative coefficients to the weights of Gaussian components. More precisely, we update each w_k by multiplying it with the corresponding I_{ck} to build the GMM for class c. This class specific GMM is used for computing Fisher Vectors for all images. Thus, intuitively, by computing I_{ck}, we are trying to gauge which Gaussian components are most representative of a class. Multiplying I_{ck} with the weights of the Gaussian components ensures that we use more information from the most representative components. This gives us class specific Fisher vector representation of all images without having to compute class specific vocabularies.

Discriminative Sparse Fisher Vectors: This strategy can further be used to improve our method of picking the most representative k' Gaussian components per image (section 3.2). Instead of choosing the most populated k' Gaussian components in each image, we use the M.I. values to decide which components are most representative of a class. More specifically, we pick the k' Gaussian components per class which share most mutual information with the class. The sparcity induced using this method promotes class level discrimination as opposed to image level discrimination. Note that while we picked the most representative Gaussian components for each image in section 3.2, here we pick the most representative Gaussian components corresponding to each class. Intuitively, this strategy makes more sense in a classification problem setting. We back this claim with results in section 4.1. In table 2 , we study the effects of introducing sparsity and discriminative information in isolation. The dataset used in these experiments is Scene 15. We show the effects on mAP for four vocabulary sizes (number of Gaussian components). In this set of experiments, we chose $k' = K/2$ and for sparse representations, the values of k' are those in column 1. It can be observed that both sparseness and discriminative information enhance classification performance in isolation. These gains are more marked at lower vocabularies.

[1]Mutual Information between two random variables X (e.g. cluster labels from a clustering method) and Y (e.g. actual class labels) is defined as: $MI(X,Y) = \sum_{y \in Y} \sum_{x \in X} P(x,y) log(\frac{P(x,y)}{P(x)P(y)})$ where $P(x,y)$ is the joint probability of X and Y and $P(x)$ and $P(y)$ are the marginal probability distributions of X and Y respectively.

Table 2: Effects of introducing sparseness and discriminative information in Fisher vector representations on mAP (Scene 15).

Vocabulary	Baseline	Discriminative	Sparseness	Both
20	84.89	86.84	87.48	88.04
50	86.30	88.64	88.43	88.91
100	86.80	88.81	89.46	89.31
250	87.19	89.40	89.47	89.59
500	87.50	89.60	89.53	89.60

3.4 Discussion

We now discuss the implications of the three improvements we have introduced to the baseline approach. Root SIFT is a preprocessing step in our algorithm and does not affect the rest of the pipeline. Computing root sift involves little computational overhead, precisely $O(1)$ for each SIFT feature.

Introducing sparsity in Fisher vectors enables us to enjoy performance comparable to more Gaussian components while adhering to the storage and computational requirements of a smaller number of Gaussian components. The implementation strategy described in section 3.2 enables us to achieve this objective. Empirical results in section 4 indicate that we can achieve performance worth K components by choosing as less as $k' = K/2$ components. This is significant improvement and reduces the memory requirements and computations by half.

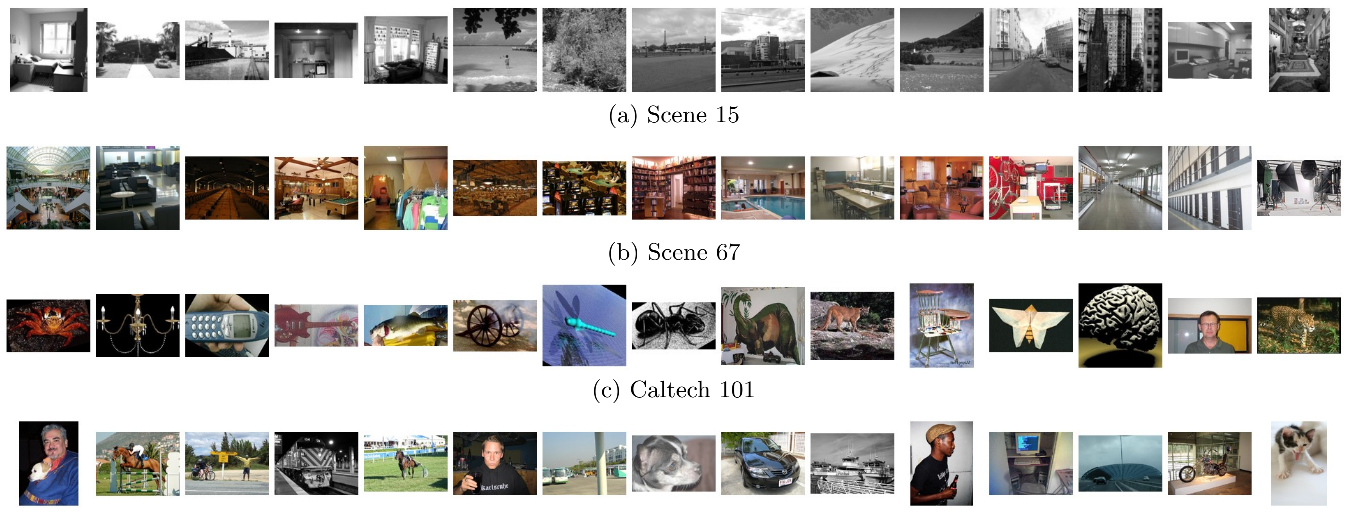
Computing discriminative Fisher vectors involves computing the multiplicative coefficients. Computing $\mathbf{I}_{ck}$ involves $n' \times K \times D$ computations where n' is the total number of training features. We compute a different Fisher vector representation per class which are used to train a classifier for this class.

It should be noted that all these modifications affect only the Fisher vector computation process. The other stages like GMM learning and classification remain unaffected.

4. EXPERIMENTS AND RESULTS

In this section, we evaluate our method on some standard visual classification datasets and provide empirical evidence to demonstrate the superior performance of our method over the traditional Fisher representation. Further, we also compare our performance with the state of the art classification approaches. We show results on 4 popular datasets (Figure 3): (a) two scene classification datasets namely Scene15 and Scene67 and (b) two object recognition datasets namely Pascal VOC 2007 and Calech 101. Table 3, describes the datasets we use in our evaluations.

Experimental Setup: All our SIFT features on Pascal VOC 2007 and Caltech 101 were computed on multiple scales of [4, 6, 8, 10] and a step size 3, as in [3]. For datasets Scene 15 and Scene 67, we computed SIFT on a single scale of 12 and a step size of 6. We used the vl_feat library [21] for SIFT computation. The number of Gaussian Components for building the Fisher Vectors were different and have been described in the following subsections. We used the *yael library* for GMM computation. For all the datasets, we trained a one-vs-rest svm [4] for each class. For Caltech101 and Scene 67, the test sample was assigned the label of the



(a) Scene 15

(b) Scene 67

(c) Caltech 101

(d) PASCAL VOC 2007

Figure 3: Random Images sampled from the 4 datasets: Scene 15, Scene 67, Caltech 101 and PASCAL 2007

Table 3: Specifications of the datasets we used.

Dataset	#Classes	#Images (Train/Test)	Evaluation
Scene15	15	4485 (1500/2985)	mAP
Scene67	67	6700 (5360/1340)	Accuracy
Caltech-101	101	5975 (3030/2945)	Accuracy
PASCAL VOC 2007	20	9963 (5011/4952)	mAP

Table 4: Comparison with state of the art approaches on the 4 Datasets

Dataset	State of the art	Our
Scene 15	88.18 [26]	89.60
Scene 67	43.1 [13]	49.87
Caltech101	77.78[3]	76.16
PASCAL VOC 2007	61.69 [3]	61.09

classifier that gave the maximum score and we report the classification accuracy while for the other datasets, we report the mean Average Precision of the retrieval task.

Experimental Details: Our baseline Fisher representation has been described in section 3. We use this baseline in all empiricial comparisions with our enhanced representations. In some of these experiments, we empirically compare the effect of varying k'. As described in section 3.2, by choosing the k' judiciously, we can achieve performance comparable to K with the storage and time complexity corresponding to k'. Since our final Fisher vector representation sizes depend linearly on K, this reduction in time and space complexity is also directly proportional to the difference between K and k'. For comparision with the state of the art approaches, we selected K and k' after extensive experimentation using a validation set.

4.1 Scene 15

Table 5: Comparison with the state of the art approaches on Scene 15

Method	mAP
SPM[19]	81.40
DSSIC [20]	85.50
Sun [26]	88.18
Sparse Discriminative FV	**89.60**

The Scene 15 dataset consists of 4485 images split over 15 different scene categories. As in [19], we randomly choose 100 images per category for training and the rest for testing. We repeated the experiments 5 times and report the mean Average Precision of the retrieval task.

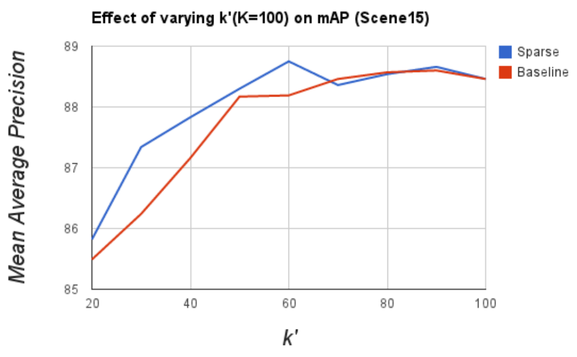


Figure 4: Effect of varying k' on the mAP on the Scene 15 dataset. K=100.

Figure 4 demonstrates the effect of varying k' on the mAP. In these experiments we gradually increase k' from $20 - 100$ using a step size of 10 by fixing $K = 100$. It can be seen that our sparse representation with comparable explicit implicit dimension almost always beats the baseline representation. Another observation is, at implicit size $k = 60$, our method even outperforms that of $K = 100$. This indicates that choosing the implicit representation size wisely can produce results better than those obtained using a higher representation size. Table 5 compares our method with the recent state of the art approaches. We beat all published results on this dataset. The previous best state of the art approach described in [26] uses 14 low level features. Our best performance was achieved at $K = 750$, $k' = 400$.

4.2 Scene 67

The Scene 67 dataset [17] has 67 Indoor categories, and a total of 15620 images. However, as in [17], we choose a fixed set of 80 images per class for training and 20 for testing.

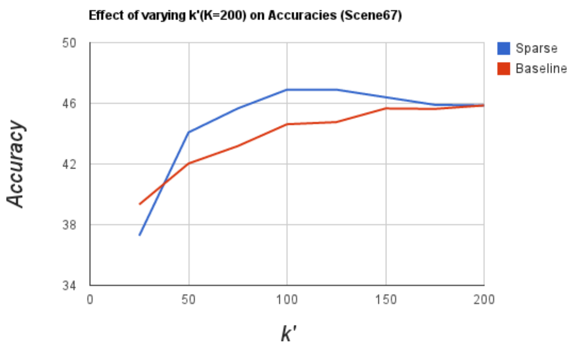


Figure 5: Effect of varying k' on the mAP on the Scene 15 dataset. K=100.

Figure 5, shows the effect of increasing k' on scene67 dataset. As compared to Scene15, the effect of varying k' is more significant on this dataset. The performance at $k' = 25$, is lower than that of the baseline representation. The gain in performance happens as we move beyond $k' = 50$. Further increasing k' beyond 100 (note that at this poing, $k' = K/2$), gives significant improvement in accuracy and as we approach $k' = K = 200$, the accuracies begin to converge to the one corresponding to $K = 200$. After extensive experimentation on Scene 67 and other datasets, our findings indicate that in general, we should choose $k' >= K/2$.

Table 6: Comparison with the state of the art approaches on Scene 67

Method	Accuracy
ROI+GIST [17]	26.5
DPM [13]	43.1
CENTRIST [25]	36.9
Object Bank [12]	37.6
Sparse Discriminative FV	**49.87**

Table 6 compares our method with the state of the art approaches. The parameters $k' = 650$ and $K = 1000$ give us our best performance quoted in the table.

4.3 Caltech 101

The Caltech 101 dataset has 9146 images coming from 101 distinct object categories. In our experiments, we randomly sample 30 images for training from each of the 101 categories, getting a total of 3030 training images; we test our method on the remaining images. We however limit the number of testing images per category to 50, as in [19]. We repeated the experiments on Caltech 101 with 5 such random subsamplings and report the mean classification accuracies over the five experiments.

Table 7: Comparison with the state of the art approaches on Caltech 101

Method	Accuracy
SPM [19]	64.6
LLC [23]	73.44
DD-FV [3]	**77.78**
Sparse Discriminative FV	76.16

Table 7 compares our method with the state of the art approaches. We achieve classification accuracy comparable to [3]. They have used the traditional Fisher kernel representation with 256 Gaussian components and spatial pyramids on top of the Fisher representation. It should be noted that while they used 8 spatial regions, we did not use spatial pyramids. Our best results were achieved at $k' = 200$ and $K = 250$. Hence the representation size used in [3] is more than 8 times that of ours.

4.4 Pascal VOC 2007

PASCAL VOC 2007 data has a total of 5011 training images and 4952 testing images in 20 classes. This is one of the hardest datasets used for visual classification as the objects in these images vary significantly in scale and orientation and some of the images have multiple objects present in them.

Table 8: Comparison with the state of the art approaches on Pascal

Method	mAP
KCB [7]	56.26
SV [27]	58.16
LLC [23]	59.74
DD-FV [3]	**61.69**
Sparse Discriminative FV	61.09

Table 8 reports the mAP achieved by the state of the art approaches on Pascal 2007. As with Caltech, our performance is comparable to [3]. Our best results were achieved at $k' = 175$ and $K = 250$. The results reported in [3] were achieved by using Fisher vectors with 256 Gaussian components and spatial pyramid with 8 spatial regions on top.

4.5 Discussion

Empirical evaluations on 4 datasets indicate the utility of our approach. We outperform all previously published results on the scene classification datasets. On Caltech 101 and Pascal 2007, not only do we manage to match the state of the art classification rates (which they report were achieved using the traditional Fisher vector representations with spatial pyramids), our explicit representation sizes are much less. These results are highlighted in table 4.

Our experiments on all the datasets revealed a common observation: if we desire a representation that uses k Gaussian components, it is always better to pick these k components from a larger vocabulary than compute a vocabulary of k Gaussian components; the former always outperforms the latter. This gain in performance can be attributed to the implicit representation size which is always bigger than the explicit representation size. Another general trend we observed is, the performance is best when $K/2 \leq k' \leq K$.

Our primary motivation behind introducing sparseness in Fisher vectors was to counter the huge representation size. Product Quantization [9, 18] has been described in literature to reduce the storage requirements. These approaches generally encode the high dimensional features and store the compressed code on the disk. However, these codes are decoded on the fly when the original features are for classification. A recent work [22] describes a strategy that uses the Product Quantization codes itself in classification.

5. CONCLUSIONS

Most practical solutions require us to understand and resolve the tradeoff between performance and resources. Fisher vectors demonstrate this concept well. The representative power of Fisher vectors gives excellent classification performance but comes at a storage and computational cost. While most recent works in this direction have attempted to compress Fisher representations to save disk space, we have endeavoured to address another important issue: reducing the representation size of Fisher vectors. We devised a novel sparse representation for Fisher vectors. Our sparse solution is desirable because it combines two elements: (a) the power of sparse representation, and (b) reduction in the explicit representation size. Further, we introduced a novel strategy for the discriminative computation of Fisher vectors. This further boosts our classification performance. In this paper, we described two approaches of picking the most representative / discriminative Gaussian components from a pool of Gaussian components to introduce sparcity. Developing better schemes for picking these Gaussian components is a research problem in its own right. We evaluated our method on four popular visual classification datasets and our method consistently outperformed the baseline Fisher representation. The performance of our model is at par with the state of the art approaches on these datasets.

6. REFERENCES

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

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

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

[4] C. chung Chang and C.-J. Lin. Libsvm: a library for support vector machines, 2001.

[5] L. V. der Maaten. Learning discriminative fisher kernels. In *ICML*, 2011.

[6] S. Gao, I. W.-H. Tsang, and L.-T. Chia. Kernel sparse representation for image classification and face recognition. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV, 2010.

[7] J. C. Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, 2008.

[8] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1998.

[9] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *PAMI*, 2011.

[10] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *CVPR*, 2005.

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

[12] E. P. X. Li-Jia Li, Hao Su and L. Fei-Fei. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010.

[13] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. *CVPR*, 2011.

[14] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[15] F. Perronnin, Y. Liu, J. SaÌĄ andnchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 2010.

[16] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[17] A. Quattoni and A. B. Torralba. Recognizing indoor scenes. In *CVPR*, 2009.

[18] J. Sanchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

[19] C. Schmid. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[20] G. Sharma, F. Jurie, and C. Schmid. Discriminative Spatial Saliency for Image Classification. In *CVPR*, 2012.

[21] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[22] A. Vedaldi and A. Zisserman. Sparse kernel approximations for efficient classification and detection. In *CVPR*, 2012.

[23] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[24] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 2010.

[25] J. Wu and J. M. Rehg. Centrist: A visual descriptor for scene categorization. *PAMI*, 2011.

[26] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. *CVPR*, 2010.

[27] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010.