Nuances in visual recognition

Gavves, E.

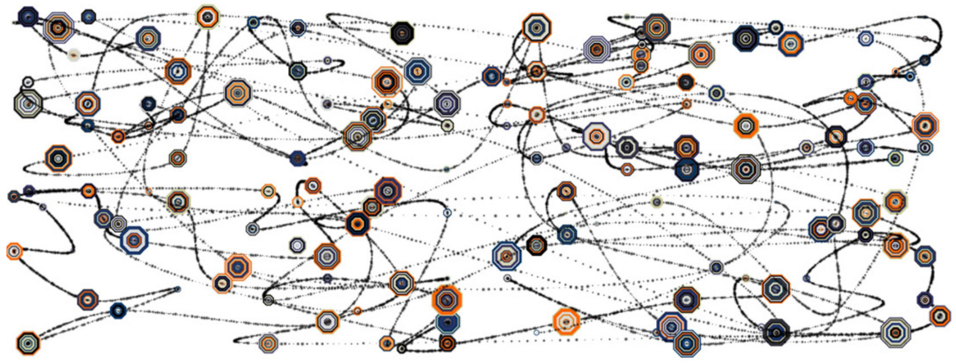
Publication date
2014

Link to publication

Citation for published version (APA):
Gavves, E. (2014). *Nuances in visual recognition*. [Thesis, fully internal, Universiteit van Amsterdam].

4

CONVEX REDUCTION OF HIGH-DIMENSIONAL KERNELS FOR VISUAL CLASSIFICATION



INVISIBLE HILBERT WITH CATMULL-ROM SPLINE INTERPOLATION
Minimal Abstract Geometric Art
Picture courtesy of Don Relyea, http://donrelyea.com/

"He who seeks for methods without having a definite problem in mind seeks in the most part in vain."

– David Hilbert (1862-1943)

Having detected nuances nuances that are able to describe the objects of interest accurately, we do not necessarily know what is their relation to the rest of the information that was deemed less relevevant. Furthermore, alleviating the dependency on some prior knowledge on the distribution of images would allow for a more practical setting.

4.1 INTRODUCTION

In classifying pictures in object and scene classes, the issue of compactness of data emerges on the agenda. In fact, the essential information content emerges due to the enormous progress that has been made, so much so that very large number of classes are now within reach.

From the TRECvid [125] and PASCAL experiments [39] it has become clear that large sets of features are necessary to capture the internal complexity of an arbitrary type of object in its distinction from the rest of the objects in the world. The feature set has to be capable of registering all internal and external variations. Therefore it is essential to start from a big set of features. For object classification this set has now grown up to 250K and 500K per images [63, 104] for the two ImageNet Large Scale Visual Recognition Challenge winners of this year.

However, large sets of features tend to obscure the understanding of the classification result, *i.e.* which ones of the features deliver most of the distinction. What is needed is a small selected set of features specific for each class. From a complexity point of view, both for storage and computation, selection of features is to be preferred over compression. Effective and semantically meaningful reduction of feature complexity is a necessary step before making the jump to routinely classifying anything on very large numbers of visual object classes.

In this chapter we discuss fast and effective classifiers by selection of a small subset of visual features from an initially very large set of features. In general, classifiers are dependent on the number of images analyzed *and* on the dimensionality of the image representation. To cope with the increasing number of images, efficient linear SVM kernels are preferred. Yet, linear SVM kernels have repeatedly been shown [103, 144] to underperform in visual classification tasks. Two current-day extensions improve the performance of linear SVM kernels. The first one relies on image feature transformations mimicking non-linear kernel maps [82, 103, 144]. The second relies on richer image representations, such as Fisher vectors [102], known to be effective in combination with linear SVM kernels. While both solutions are effective in terms of classification accuracy, they both result in high-dimensional image representations and, hence, in inefficiency. We aim to reduce the complexity of the image representation *a posteriori* by feature selection when building the kernel, while maintaining the accuracy advantage of both non-linear kernel maps and Fisher vectors.

Dimensionality reduction for visual classification has been a topic of constant interest, since it allows for more compact and thus more efficient image representations. Recent approaches [71,85] focus on supervised vocabulary reduction by optimizing statistical and information theory criteria. Both [85] and [71] use class labels from ground truth to select vocabulary words that maximize statistical inter-class criteria [61]. However, large data sets are also accompanied by the presence of large number of classes, *e.g* ImageNet currently counts approximately 22K classes [36]. In the presence of so many, and often visually similar, classes optimizing such inter-class criteria is a futile task. In contrast, unsupervised data dimensionality reduction, such as PCA [13] or UKDR [150] overcomes this hurdle. We consider unsupervised kernel reduction.

We propose to perform kernel reduction before classification. Where standard kernels calculate distances between features dimension by dimension, before summing them, we propose to calculate the distance on *selected and weighed* feature dimensions. We pose selection and

Published in the *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2012* [46].

weighing of kernel dimensions as a convex optimization problem, searching for those dimensions that cooperatively approximate the original kernel value set. Such a treatment allows for class-independent feature dimensionality reduction. Experiments on the Scenes [72] and PASCAL VOC 2007 [39] datasets show that our reduction performs as accurate as PCA yet much faster. Moreover, in contrast to existing dimensionality reduction methods, our method can be applied independent of *and* additional to, both non-linear embedding and Fisher vectors. The method does not use any labeling of images, nevertheless we find that the selected dimensions tend to correspond to meaningful patches in the image.

4.2 RELATED WORK

4.2.1 *Enhancing linear kernels*

Simple linear SVM kernels have repeatedly been shown [103, 144] to underperform in visual classification. However, linear SVM efficiency benefits are crucial when moving to large data and feature sets. Several approaches have been proposed that maintain the efficiency of linear SVM kernels, while making them more accurate.

Kernel theory dictates that for every non-linear kernel map $K(\mathbf{p}, \mathbf{q})$, there exists a feature mapping $\phi(\,\cdot\,)$, such that $K(\mathbf{p}, \mathbf{q}) = \phi(\mathbf{q})^T \phi(\mathbf{p})$. This is a simple dot product, thus equivalent to a linear kernel calculated using $\phi(\mathbf{p})$ instead of $\mathbf{p}$. Several methods have been proposed that approximate $\phi(\cdot)$.

In their pioneering work, Maji and Berg [82] propose to approximate $\phi(\,\cdot\,)$ for the histogram intersection kernel with a modified version of Heaviside (step) functions. Generalizing on the Maji and Berg embedding, Vedaldi and Zisserman [144] describe the theoretical formulations for explicit feature maps of the χ^2, histogram intersection, Hellinger's and Jensen-Shannon kernel maps, calculated on their corresponding Fourier transform space. They furthermore propose a periodic approximation of these explicit feature maps, that allows for a concise, yet accurate enough representation for $\phi(\,\cdot\,)$. Rahimi and Recht [107] and Li et al. [77] also propose exploitation of the Fourier space of kernel maps, using random sampling instead of the exact Fourier formulation. Starting from the data point of view, Williams et al. [152] proposed the kernel PCA. The kernel PCA learns the non-linear embeddings from the data distribution. To improve the efficiency of kernel PCA when learning the non-linear embedding, Perronnin et al. propose the additive kernel PCA [103] that uses additivity to speed up the process. Both kernel PCA and additive kernel PCA are characterized of high computational complexity during embedding of new samples, although this complexity can be decreased in exchange of larger image feature dimensionality.

Rather than focusing on the kernel, other approaches emphasize enriching the image representation. In [102], Perronnin et al. propose to encode images using the Fisher vectors. Instead of mapping every local feature to a single element of a visual vocabulary, the difference between the local features and all the elements in the visual vocabulary is computed. In a similar fashion, Zhou et al. [169] propose the Super vector encoding, which also captures the difference between local features and vocabulary elements. Both methods have shown to perform exceptionally well in visual classification benchmarks like PASCAL VOC 2007 [28].

4.2.2 *Reducing linear kernels*

Data dimensionality reduction is a classical problem in the field of machine learning and statistics. The most popular method for unsupervised data reduction is PCA [13], which has been shown to work in a variety of contexts. Recently, Perronnin et al. [102] used PCA to reduce the dimensionality of SIFT features, so that their proposed Fisher vectors have a more manageable size. In [108], Raina et al. use PCA as a competitor to their LASSO optimization, which was applied on a transfer learning problem. Other methods for reducing the kernel dimensionality in an unsupervised manner include the work of Wang et al. [150]. Their desideratum is to mimic the autoencoders from the neural network literature, and like PCA, learn a transformation map which would reproduce the original kernel when applied inversely. They obtain low classification errors, especially for very low dimensions (less than 5). For larger dimensionality problems, which is the standard in visual classification, they achieve at best on par with PCA.

4.2.3 *Contribution and novelty*

State-of-the-art visual classification approaches such as the approximated feature maps [144] and Fisher vectors [102] enhance linear SVM kernels at the expense of image features with high dimensionality. Since both approaches build on the linear SVM kernel, reducing the linear kernel will result in substantial efficiency benefits. Methods for unsupervised kernel dimensionality reduction exist [13, 150], but it is unclear whether their reduced kernels benefit from methods that enhance linear SVM models.

We propose to perform linear kernel reduction before classification. We pose selection and weighing of kernel dimensions as a convex optimization problem, searching for those dimensions that cooperatively approximate the original kernel value set. Such a treatment allows for class-independent feature dimensionality reduction. Our main contributions are that *i)* our reduction performs as accurate as PCA yet much faster, *ii)* our reduction is complementary to non-linear embeddings where others are not, and *iii)* combining our reduced kernel with Fisher vectors and approximate feature maps yields robust accuracy.

4.3 CONVEX REDUCTION OF KERNELS

Let $K(\mathbf{p},\mathbf{q})$ denote the abstract manifestation of a kernel, such as $K(\mathbf{p},\mathbf{q}) = \sum_i \min\{p_i, q_i\}$ for the histogram intersection kernel. With $N(\mathbf{p},\mathbf{q})$ we shall denote the convex reduced kernels we propose. With $\mathbf{x} = [x_1, ..., x_D]^T, \mathbf{y} = [y_1, ..., y_D]^T$ we denote the image feature composed of D dimensions. Given $\mathbf{x}, \mathbf{y}$, let $K_{\mathbf{x},\mathbf{y}}$ denote the actual distance value calculated between the two images, that is $K_{\mathbf{x},\mathbf{y}} = K(\mathbf{p},\mathbf{q})|^{\mathbf{p}=\mathbf{x}}_{\mathbf{q}=\mathbf{y}}$. Finally, we shall denote the 1-d kernel distance values calculated per dimension i of $\mathbf{x}, \mathbf{y}$ with $k^i_{\mathbf{x},\mathbf{y}}$, that is $k^i_{\mathbf{x},\mathbf{y}} = K(p_i, q_i)|^{p_i=x_i}_{q_i=y_i}$.

4.3.1 *Theory*

The most popular kernels applied for visual classification are the χ^2, histogram intersection and Hellinger's (or Bhattacharya) kernels. These kernels share two basic properties, additivity and homogeneity. A kernel is additive, when

$$K(\mathbf{p},\mathbf{q}) = \sum_i^D K(p_i, q_i), \tag{4.1}$$

This property is especially convenient, since performing 1-d, non-linear operations and adding them is always more efficient than performing a single multi-dimensional, non-linear operation. A kernel $K(\mathbf{p},\mathbf{q})$ is γ-homogeneous, if

$$K(c\,\mathbf{p}, c\,\mathbf{q}) = c^\gamma K(\mathbf{p},\mathbf{q}),\ \forall c \geq 0. \tag{4.2}$$

$$\begin{array}{cccccccc} \begin{array}{l} \langle x = 32, y = 234\rangle \\ \langle x = 142, y = 91\rangle \\ \langle x = 332, y = 54\rangle \\ \quad\dots \end{array} & \begin{bmatrix}0.6\\0.3\\0.6\\\dots\end{bmatrix} & \approx & \begin{bmatrix}0.55\\0.3\\0.65\\\dots\end{bmatrix} & - & \begin{bmatrix}0 & 0.15 & 0 & 0.3 & 0.10\\0.05 & 0.1 & 0.05 & 0.1 & 0\\0.1 & 0.2 & 0.1 & 0.2 & 0.05\\\dots & \dots & \dots & \dots & \dots\end{bmatrix} & \cdot & \begin{bmatrix}0\\2\\0\\1\\0\end{bmatrix} \\ & N_{x,y} & & K_{x,y} & & k^{1,\dots,5}_{x,y} & & c \end{array}$$

Figure 24: ***Simple example of a convex reduced kernel using eq.*** (4.5). *The new kernel distances $N_{\mathbf{x},\,\mathbf{y}}$ approximate $K_{\mathbf{x},\,\mathbf{y}}$ for various pairs of images $< \mathbf{x}, \mathbf{y} >$, by using only 2 out of 5 of the 1-d kernel distances.*

Homogeneity implies that scaling the kernel values by a constant c^γ has the same effect as scaling the corresponding image feature dimensions by the same constant c. The simplest kernel having both properties is the linear kernel, *i.e.* $K(\mathbf{p}, \mathbf{q}) = \mathbf{q}^T\mathbf{p}$.

Given an additive kernel $K(\mathbf{p}, \mathbf{q})$ and two image feature representations, $\mathbf{x}, \mathbf{y}$, their kernel distance may be written as a sum of the kernel distances of the individual dimensions, that is

$$K_{\mathbf{x},\mathbf{y}} = k^1_{\mathbf{x},\mathbf{y}} + \dots + k^D_{\mathbf{x},\mathbf{y}}. \tag{4.3}$$

Interestingly, eq. (4.3) implicitly assumes that when calculating the distance between two images, the 1-*d* distances between the individual dimensions have equal unit weight and thus equal importance. However, when two images are similar, it is mainly because some specific words contribute the most by having large values for both images. For example, when we have two images of boats, we expect that the distance between “water”visual words will contribute more to their similarity than the distance between “car wheel” visual words [139]. Therefore, we propose a different scaling factor c_i in front of each 1-*d* kernel distance and form a new kernel distance $N_{\mathbf{x},\mathbf{y}}$, that is

$$\begin{aligned} N_{\mathbf{x},\mathbf{y}} &= c_1 k^1_{\mathbf{x},\mathbf{y}} + \dots + c_D k^D_{\mathbf{x},\mathbf{y}} \\ &= \mathbf{c}^T \mathbf{k}_{\mathbf{x},\,\mathbf{y}}, \end{aligned} \tag{4.4}$$

where $\mathbf{k}_{\mathbf{x},\,\mathbf{y}} = [k^1_{\mathbf{x},\mathbf{y}}, \dots, k^D_{\mathbf{x},\mathbf{y}}]^T$. Eq. (4.4) implies that in order to find the similarity between two images, we may as well measure the distances $k^i_{\mathbf{x},\mathbf{y}}$ between each individual dimension and multiply them with the corresponding scaling factor c_i. By doing so, we place more importance to the distances between *certain* dimensions. Note that $N(\mathbf{p}, \mathbf{q})$ is an additive kernel map as well.

4.3.2 *Convex reduction*

Starting from eq. (4.4), we want to compact $N(\mathbf{p}, \mathbf{q})$. Therefore, we first separate the 1-*d* kernel dimensions contributing more to the final kernel distance values $N_{\mathbf{x},\mathbf{y}}$ from the less important dimensions. In order to distinguish the two, we denote with Δ the number of important dimensions, with c_i^+ the scaling factors of the important dimensions and with c_i^- the scaling factors of the unimportant dimensions. Compacting the kernel, or ignoring the unimportant dimensions, is equivalent to setting their scaling factors to zero, that is $c_i^- = 0$.

Intuitively, in order to obtain a successful compact approximation $N_{\mathbf{x},\mathbf{y}}$, we want two conditions to hold: *i)* for all possible image pairs $\mathbf{x}, \mathbf{y}$ to have $N_{\mathbf{x},\mathbf{y}} \approx K_{\mathbf{x},\mathbf{y}}$ and *ii)* $\Delta \ll D$. The first condition is typically resolved in the literature by minimizing the squared difference between the target value $K_{\mathbf{x},\mathbf{y}}$ and the regressed value $N_{\mathbf{x},\mathbf{y}}$. The second condition is mathematically equivalent to minimizing the ℓ_1 norm of vector $\mathbf{c}$, that is driving as many elements in $\mathbf{c}$ to zero as possible. Taking into account eq. (4.4), we formulate the following optimization problem

$$\arg\min_{\mathbf{c}} \left\| K_{\mathbf{x},\mathbf{y}} - \mathbf{c}^T \mathbf{k}_{\mathbf{x},\mathbf{y}} \right\|^2 + \lambda \left\| c \right\|_{\ell_1} . \tag{4.5}$$

Eq. (4.5) is a regularized least squares problem, also known as LASSO convex optimization problem. A LASSO problem may be solved efficiently using non-negative quadratic programming, such as the feature-sign algorithm [74]. In order to phrase our optimization problem as a non-negative quadratic one, we assume the values of $\mathbf{c}$ to be positive. In order to efficiently solve eq. (4.5), we use 1-d kernel distances as training data to learn from the kernel data distribution at hand. We collect training data by measuring the kernel distances between random image pairs $\mathbf{x}, \mathbf{y}$ to form the column vector $\mathbf{K}_{\mathbf{x},\mathbf{y}}$. At the same time, we store the individual 1-d kernel distances as computed from each one of the kernel dimensions $k^i_{\mathbf{x},\mathbf{y}}$, see Fig. 24. While all possible image pairs $\mathbf{x}, \mathbf{y}$ can be used, we observe in our experiments that randomly sampling and using 5% of them is adequate enough, see Table 9. Parameter λ controls the sparsity of our solution, effectively affecting the coarseness of our approximation. The smaller the λ, the more dimensions are activated and therefore the better an approximation is obtained. An example is shown in Fig. 25.

After solving the optimization of eq. (4.5), we obtain the scale vector $\mathbf{c}$ that approximates the original kernel distances $K_{\mathbf{x},\mathbf{y}}$ with a small number of selected and weighed 1-d kernel dimensions $k^i_{\mathbf{x},\mathbf{y}}$. In order to be able to apply the optimization solution directly on the image feature representations, we take advantage of γ-homogeneity, see eq. (4.2). More specifically

$$\begin{aligned} N(\mathbf{x},\mathbf{y}) &= \sum_i^{\Delta} c_i^{\gamma\,+} K(x_i, y_i) \\ &= \sum_i^{\Delta} K(c_i^+ x_i, c_i^+ y_i). \end{aligned} \tag{4.6}$$

Based on eq. (4.6) we only need to consider the dimensions, which correspond to the non-zero scale factors c_i^+. We then multiply the respective dimensions of the image features $\mathbf{x}, \mathbf{y}$ with $\sqrt[\gamma]{c_i^+}$ and ignore the rest of the dimensions. When considering the 2-homogeneous linear kernel for example we multiply each image's feature dimension i with the square root of the corresponding scaling factor, *i.e.* $\sqrt{c_i}$.

The kernel $N(\mathbf{p}, \mathbf{q})$ is valid if it meets two conditions, that is: *i)* being symmetric and *ii)* being semi-positive definite. Based on eq. (4.4) and given that $K(\mathbf{p}, \mathbf{q})$ is a valid kernel, we derive that the first condition is met, since the image features in the 1- d kernel distances are interchangeable, *i.e.* $k^i_{\mathbf{x},\mathbf{y}} = k^i_{\mathbf{y},\mathbf{x}}$. Moreover, since $c_i \geq 0$ for all elements in vector $\mathbf{c}$ and $K(\mathbf{p}, \mathbf{q})$ is positive semi-definite, then $N(\mathbf{p}, \mathbf{q})$ is also positive semi-definite. We therefore conclude that the new kernel is valid.

An index of measuring the quality of our convex reduced kernel is the root mean square error $\epsilon(\mathbf{c})$ between $N_{\mathbf{x},\mathbf{y}}$ and $K_{\mathbf{x},\mathbf{y}}$. If the error is small, then our reduced kernel properly approximates the original kernel distances. In practice, we require that the root mean square error given vector $\mathbf{c}$ over the average kernel distance value is below a threshold t, that is

$$\frac{\epsilon(\mathbf{c})}{\hat{K}_{\mathbf{x},\mathbf{y}}} < t \tag{4.7}$$

We may use eq. (4.7) to define the optimal Δ for keeping a balance between performance loss and reduction ratio.

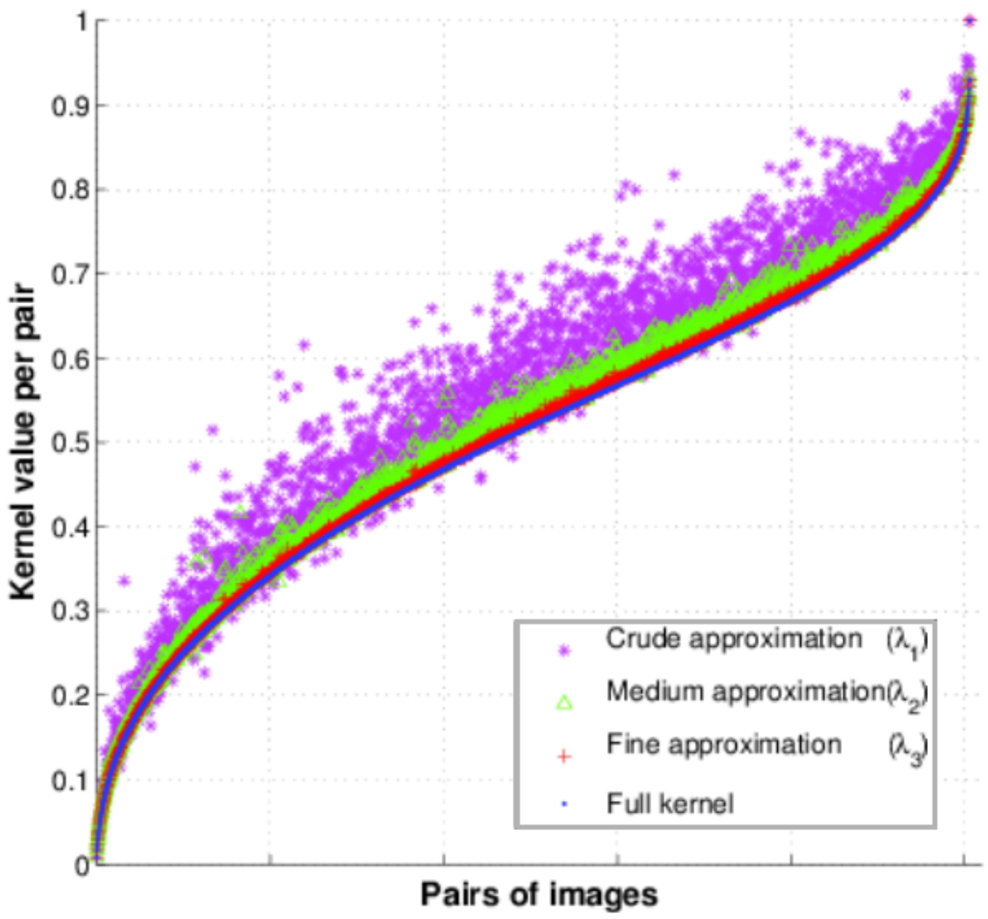


*Figure 25: **Kernel value distribution for various image pairs** (best viewed in color). In blue we have the original kernel distance calculated. With the magenta, green and red we visualize convex reduced versions, that is $\lambda_1 > \lambda_2 > \lambda_3$. The λ_1 approximations uses very few (i.e. 66) dimensions, therefore accuracy of the kernel reconstruction is hurt.*

All the formulations above did not specify the kernel type. Although every kernel may be reduced, we observed that with the current linear model of eq. (4.4), we obtain the most accurate reconstruction when reducing the linear kernel. Moreover, by reducing the original linear kernel, we experimentally observe to have the additional benefit of being complementary to non-linear embeddings, such as [82, 103, 144]. In the remaining text we consider the linear kernel, that is $K(\mathbf{p}, \mathbf{q}) = \mathbf{q}^T \mathbf{p}$.

4.3.3 *Complexity*

We identify four stages that need to be evaluated with respect to complexity. We separate these four stages into two groups, the *offline* and the *online* computations. The offline computations are fixed and independent of the dataset size; whether having $1,000$ or $1,000,000$ images, the offline optimization will spend the same time. Yet, the online computation timings will be severly affected when having $1,000,000$ instead of $1,000$ images. Since datasets become larger, we consider them separate for fairness.

1. Optimization time *(offline)*. In the first stage we optimize eq. (4.5). We first calculate 1-*d* kernel distances between pairs of images in the training set, with a $O(D^2 \cdot Q)$ for the linear kernel, where Q the number of pairs of images used to extract kernel distances. Then we solve eq. (4.5), with the complexity depending on the solver used. The optimization is performed in an offline manner once. Hence it does not affect learning and classification.

2. Preparation time *(online)*. The second stage refers to the complexity for obtaining the new image representations. We first reduce the original image features according to the new reduced kernel. During reduction, we require only scaling of the selected dimensions with an appropriate constant. Thus we have a $O(\Delta)$ complexity per image. Applying the non-linear embeddings [82, 103, 144] on top of our reduced kernel is optional. The complexity depends on the embedding. However, having fewer dimensions always results in faster embeddings. Therefore, embedding our convex reduced kernel is more efficient than embedding the full kernel.

3. Learning time *(online)*. The third stage involves the run-time complexity. After obtaining the reduced kernel, a linear SVM is applied for learning and classification. Linear SVM has learning complexity that depends on the algorithm used. However, state-of-the-art algorithms like [118] use sparse matrix algebra, hence complexity depends on the number of non-zero elements per feature. When using convex reduced kernel the dimensionality becomes smaller and the number of non-zero elements per feature drops, therefore learning becomes faster.

4. Classification time *(online)*. The forth stage refers to the classification. We multiply the obtained weight vector $\mathbf{w}$ with the respective image features (and perform one summation for the bias term). If we have M images, classification is a vector-matrix multiplication characterized by $O(D \cdot M)$ complexity. When applying our convex reduced kernel, this complexity drops to $O(\Delta \cdot M)$, which is a noticeable speedup, especially for large data and feature sets.

4.4 EXPERIMENTS

4.4.1 *Experimental setup*

Datasets. For the experiments we use two popular datasets. The first one is the *Scenes* dataset introduced by Lazebnik *et al.* [72]. The *Scenes* dataset contains 4,485 medium size images of 15 indoor and outdoor scenes, such as “kitchen” or “forest”. The second dataset is the *VOC 2007* dataset [39], which contains 9,963 images and is composed of 20 different objects. The *VOC 2007* dataset is a particularly challenging dataset, since one image may contain several of the objects. What is more, the instances of an object in a picture may exhibit large variation in appearance, size, context, etc. For both datasets we use the common training and test set divisions. Note that our convex reduced kernel is optimized on the training sets only.

Evaluation criteria. We study the algorithms with respect to their average precision accuracy, which is equivalent to the area under the precision-recall curve. We report the mean of average precision (mAP) over all scenes or objects. We also study the algorithms with respect to their computational efficiency. We focus on the *online* timings required by all the algorithms, that is the preparation timings, the learning timings and the classification timings, as discussed in Subsection 4.3.3. We repeat the experiments 10 times and report the mean and the standard deviation. The linear SVM solver we use, pegasos [118], returns approximate solutions, so we report the mean and standard deviation of mean average precision for the 10 runs. For completeness, we also report the *offline* optimization timings. All timings were computed on a standard Xeon machine at 2.93GHz.

Implementation. For all images we extract dense SIFT [81] features every 2 pixels on multiple scales, *i.e.* 4, 6, 8 and 10 pixels for bin size. We then construct a visual vocabulary of 4,000 visual words using approximate k-means [105]. Spatial pyramids [72] or other descriptors may be used to further improve accuracy, but we do not include them in our current experiments. In order to solve the convex-optimization problem of eq. (4.5), we use the feature-sign algorithm [74]. We learn the linear SVM using pegasos [118]. Prior to learning and classification, we perform

ℓ_2 normalization. For the non-linear embeddings, we use the code provided [142] for the approximated feature maps [144]. We set its periods extension to 3. Preliminary experiments have shown similar results also for kernel-PCA [152]. Finally, we use an approximate version of Fisher vectors, which does not include the variance term. We calculate the mixture parameters of the gaussian mixture model (GMM) from the approximate k-means clusters. Hence, our mixture model is equivalent to a GMM after one round.

4.4.2 *Experiment 1: Linear kernel reduction*

In the first experiment we compare our proposed convex reduced kernel with PCA [108]. We examine how well both approximate the full linear kernel, which uses a dimensionality of 4,000 words. The results are summarized in Table 7.

For both the *Scenes* and *VOC 2007* datasets, our kernel and PCA perform on par in terms of classification accuracy. Moreover, both approximate the full kernel well. We observe similar behavior for *Caltech-101* (data not shown). In terms of efficiency for offline optimization, our method needs approximately 3 minutes to harvest the 1-d kernel distances from $50K$ pairs of images, and it takes 4, 36 and 248 seconds for the 600, 1,200 and 2,200 dimensional kernels respectively. In the online stages of visual classification, the efficiency of our kernel becomes prevalent. Especially in the kernel preparation stage where PCA requires a matrix-matrix multiplication with a $O(\Delta \cdot D \cdot M)$ complexity, and ours is linear with $O(\Delta)$. Hence, we are 10x faster as PCA. Since the size of the reduced kernel is equal for learning and classification, our method and PCA are equally efficient for these stages. Depending on the reduction rate, both our method and PCA can be up to 10x faster than using the full linear kernel. In terms of total online classification time our method is most efficient.

gforge.inria.fr/projects/yael

Table 7: ***Experiment 1: Linear kernel reduction.*** *Our proposed convex reduced kernels are as accurate as PCA when approximating a full linear kernel, yet faster for online classification. We report rounded codebook sizes.*

		Scenes				***VOC 2007***			
Method	Size	mAP	Preparation(s)	Learning(s)	Classification(s)	mAP	Preparation(s)	Learning(s)	Classification(s)
Full	4,000	0.767±0.001	0.18±0.01	10.39±0.05	0.13±0.00	0.388±0.002	0.36±0.01	20.89±0.14	0.32±0.01
PCA	600	0.763±0.001	1.44±0.01	1.14±0.01	0.04±0.00	0.381±0.001	3.10±0.06	3.59±0.01	0.08±0.00
PCA	1,200	0.766±0.001	2.74±0.08	2.71±0.15	0.05±0.00	0.387±0.002	6.17±0.15	6.70±0.02	0.12±0.00
PCA	2,200	0.766±0.001	4.88±0.01	4.14±0.01	0.08±0.00	0.388±0.002	10.33±0.35	11.14±0.04	0.18±0.00
Our kernel	600	0.747±0.001	0.11±0.00	1.14±0.01	0.03±0.00	0.364±0.002	0.20±0.00	3.58±0.01	0.08±0.00
Our kernel	1,200	0.757±0.002	0.19±0.01	2.88±0.12	0.06±0.00	0.376±0.002	0.42±0.01	6.72±0.01	0.12±0.00
Our kernel	2,200	0.761±0.000	0.41±0.01	4.42±0.16	0.08±0.00	0.379±0.002	0.80±0.01	11.13±0.02	0.18±0.00

4.4.3 *Experiment 2: Non-linear kernel embedding*

We test in the second experiment the complementarity of both our reduced kernel and PCA with the non-linear embedding using approximated feature maps [144]. The results are summarized in Table 8.

In contrast to PCA, our kernel reduction is complementary to approximated feature maps. Hence, our visual classification results are much more accurate for both the *Scenes* and *VOC 2007* datasets. We approach the accuracy of an embedding using a full kernel with approximately half of the number of dimensions. Regarding the efficiency, we observe substantial gains from using our reduction over the full kernel. Data preparation is up to 80% faster than using the full kernel for both *Scenes* and *VOC 2007* datasets. Learning becomes 50-85% faster for both the *Scenes* dataset and the *VOC 2007* dataset. Classification becomes 55-85% faster for the *Scenes* dataset and 45-85% faster for the *VOC 2007*dataset. Again, PCA is considerably slower in the data preparation time. The offline optimization is the same as in Section 4.4.2. Performing the reduction first on the linear kernel is theoretically inferior, yet has the apparent benefit that from that moment on we no longer need to apply subsequent optimizations on irrelevant kernel dimensions. For example, applying the approximate feature maps on the 600 most relevant kernel dimensions results in a 4,500-*d* kernel. When we compare it with the approximately similar sized regular 4,000-*d* kernel, we obtain a relative increase of 7% on *Scenes* and 11% on *VOC 2007* (see Table 8).

We show some classification examples for the *VOC 2007* dataset in Fig. 27. In the pictures the 10 dimensions with the largest c_i values are visualized. Although images are densely sampled, we find that features mapped on these dimensions tend to be located on salient locations, often on objects. It is interesting that no labeled data are used to find these features lying on salient locations. We conclude that our convex reduced kernel is complementary to non-linear embeddings, and should be used together with these embeddings, such as the approximated features maps, to obtain an accurate and efficient visual classification.

Table 8: ***Experiment 2: Non-linear kernel embedding.*** *In contrast to PCA, our kernel reduction is complementary to non-linear embeddings like approximated feature maps with χ^2 kernel [144]. Our kernel maintains the accuracy advantages of non-linear embeddings, but is more efficient for online classification, during all stages, when compared to non-linear embedding of the full 4,000 dimensional kernel.*

		Scenes				***VOC 2007***			
Method	Size	mAP	Preparation(s)	Learning(s)	Classification(s)	mAP	Preparation(s)	Learning(s)	Classification(s)
Full	4,000	0.767±0.001	0.18±0.01	10.39±0.05	0.13±0.00	0.388±0.002	0.36±0.01	20.89±0.14	0.32±0.01
AFM(*Full*)	28,000	0.855±0.000	2.84±0.45	61.32±9.51	0.97±0.09	0.475±0.001	4.77±0.04	143.25±0.20	2.05±0.01
AFM(*PCA*-600)	4,500	0.746±0.001	1.88±0.10	7.55±0.11	0.13±0.0	0.351±0.001	4.11±0.16	23.21±3.08	0.36±0.00
AFM(*PCA*-1200)	8,700	0.753±0.001	3.72±0.03	14.65±0.15	0.24±0.01	0.346±0.001	7.55±0.45	42.37±0.07	0.65±0.03
AFM(*PCA*-2200)	15,300	0.753±0.001	4.07±0.72	23.38±0.08	0.39±0.01	0.345±0.001	13.46±0.25	70.54±0.11	1.11±0.00
AFM(*Our kernel*-600)	4,500	0.819±0.000	0.47±0.02	8.77±0.63	0.15±0.01	0.431±0.001	0.98±0.00	22.98±0.05	0.35±0.00
AFM(*Our kernel*-1200)	8,700	0.832±0.000	0.89±0.02	16.88±2.82	0.26±0.01	0.449±0.000	1.86±0.01	43.83±0.04	0.66±0.00
AFM(*Our kernel*-2200)	15,300	0.841±0.000	1.63±0.05	29.76±0.22	0.47±0.01	0.458±0.001	3.29±0.01	74.04±0.15	1.11±0.01

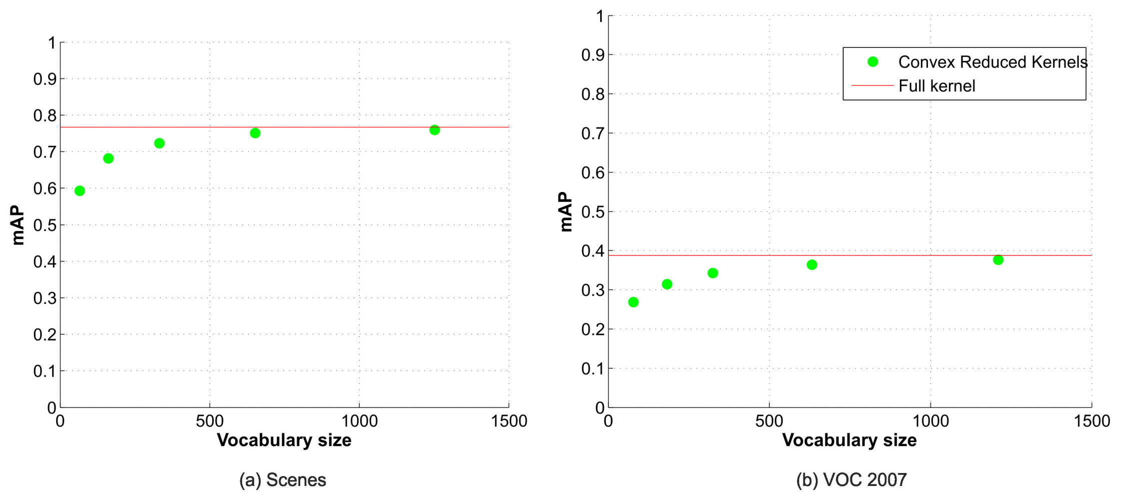


Figure 26: ***Experiment 3: Convex reduction parameters.*** *Mean average precision given λ.*

4.4.4 *Experiment 3: Convex reduction parameters*

In experiment 3 we first compare the performance of our reduced kernels with respect to the image feature dimensionality, which is controlled by parameter λ of eq. (4.5). We show results in Fig. 26. For the *Scenes* dataset we observe in Fig. 26(a) that a reduction to only 60 dimensions leads to a crude approximation. As a result the error ratio, see eq. (4.7), is high; around 14%. Increasing the number of dimensions results in smaller error ratios and better classification accuracy. The same holds for the *VOC 2007* dataset, see Fig. 26(b).

We also test the influence of the number of samples used for solving the optimization of eq. (4.5). We sample the 1-d kernel distances geometrically in the interval $[10K, 200K]$ for the *Scenes* dataset. The results are shown in Table 9. We observe that using the kernel distances between 50K to 100K data samples, which corresponds to 2-5% of the number of the 2.5M possible pairs of images in the training set, is adequate for a good approximation. Similar results were obtained for the *VOC 2007* dataset (data not shown).

4.4.5 *Experiment 4: Non-linear Fisher kernel*

In our fourth experiment, we take advantage of the convex reduced kernels to combine them with the popular Fisher vectors [102]. Extracting Fisher vectors using the full kernel is computationally too expensive (our full kernel is composed of 4,000 dimensions, contrast to the 256 in [102]). Using the Fisher representation for the reduced kernels results in image feature of large, yet

Table 9: ***Experiment 3: Convex reduction parameters.*** *Performance of the kernel approximation with respect to the sample size used for training in eq.* (4.5). *The results are averaged over 3 runs including standard deviation. Using* 50K *pairs of images, that is 2% of the number of possible pairs in the training set of the Scenes dataset is adequate to solve eq.* (4.5). *Similar results are obtained for the VOC 2007 dataset (data not shown).*

Method	**10K**	**20K**	**50K**	**100K**
Our kernel	0.740±0.002	0.744±0.000	0.748±0.000	0.750±0.001
AFM(*Our kernel*)	0.816±0.001	0.818±0.001	0.819±0.001	0.819±0.000

Table 10: ***Experiment 4: Non-linear Fisher kernel.*** *Using our 600-d kernel as a basis, we can apply additional optimizations like Fisher vectors and approximate feature maps, whose computation using the Full 4,000-d kernel is prohibitive. Combining our kernel with Fisher vectors and approximate feature maps is fruitful (best results bold).*

	Size		***Scenes***		***VOC 2007***	
Method	Full	Ours	Full	Ours	Full	Ours
Kernel	4K	600	0.767	0.747	0.388	0.364
AFM(Kernel)	28K	4.5K	0.855	0.819	0.475	0.431
Fisher(Kernel)	256K	40K	N/A	0.848	N/A	0.457
AFM(Fisher(Kernel))	1.8M	290K	N/A	**0.883**	N/A	**0.529**

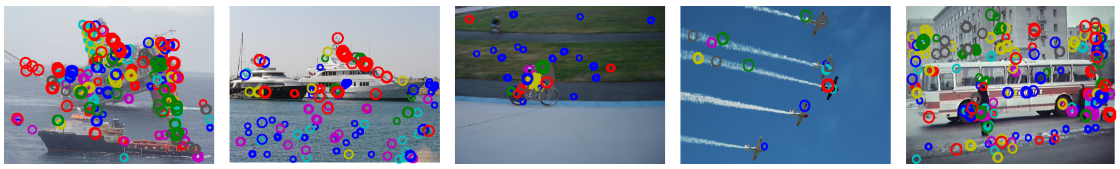


Figure 27: ***Ten kernel dimensions with the largest weights*** c_i ***after reduction for VOC 2007 images*** *(best viewed in color). Note that dimensions were found based on pairwise 1-d kernel distances between pairs of images only. No individual bag-of-words image features, nor labeled examples were used.*

manageable dimensionality. The reduced feature size allows us to embed the reduced Fisher vectors on the approximated feature maps [144], so that it benefits from non-linear SVMs. Using the full kernel of 4,000 dimensions, combined with Fisher vectors and approximated feature maps would require prohibitively expensive computations, resulting in vectors of 1.8M dimensions, too large to handle. We show results in Table 10. For completeness, we also include results from experiments 1 and 2.

We observe an explosion of dimensions when using Fisher vectors, resulting in image features of 40K dimensions. For the *Scenes* dataset the increase comes with an accuracy boost from 0.747 to 0.848. When we apply the approximated feature maps on the Fisher vectors and our kernel reduction we improve even further, scoring 0.883 in *mAP*. Since for the *Scenes* dataset also precision is used for evaluation, we report our current best result of 0.840, not yet including spatial pyramids and their newest variants. For the *VOC 2007* dataset the absolute improvement of embedding the reduced Fisher vectors on the approximated feature maps is even larger, going from 0.364 in mAP to 0.529, a serious improvement. We conclude that our convex reduced kernel is complementary to Fisher kernels. What is more, using our convex kernel to reduce Fisher vectors makes non-linear embeddings computationally feasible, and results in robust classification accuracy.

4.5 CONCLUSION

We propose to perform linear kernel reduction before classification. We pose selection and weighting of kernel dimensions as a convex optimization problem by searching for those dimensions that cooperatively approximate the original kernel value set. Our proposed algorithm performs the reduction in an unsupervised manner, making it independent of the number of classes in the classification. We show on both the Scenes and VOC 2007 datasets that our reduced linear kernels are more efficient and as accurate as existing reduction methods. In contrast to existing reduction methods, our reduced kernels benefit from the state-of-the-art in visual classification, such as the approximated feature maps [144] and Fisher vectors [102]. We advocate to use our convex reduced kernel together with Fisher kernels and approximate feature maps to obtain the highest accuracy most efficiently.