# Asymmetric sparse kernel approximations for large-scale visual search

Damek Davis
University of California
Los Angeles, CA 90095
UCLA
damek@math.ucla.edu

Jonathan Balzer
University of California
Los Angeles, CA 90095
UCLA
balzer@cs.ucla.edu

Stefano Soatto
University of California
Los Angeles, CA 90095
UCLA
soatto@cs.ucla.edu

## Abstract

*We introduce an asymmetric sparse approximate embedding optimized for fast kernel comparison operations arising in large-scale visual search. In contrast to other methods that perform an explicit approximate embedding using kernel PCA followed by a distance compression technique in $\mathbb{R}^d$, which loses information at both steps, our method utilizes the implicit kernel representation directly. In addition, we empirically demonstrate that our method needs no* explicit *training step and can operate with a dictionary of random exemplars from the dataset. We evaluate our method on three benchmark image retrieval datasets: SIFT1M, ImageNet, and 80M-TinyImages.*

## 1. Introduction

Image comparison for the purpose of detection, recognition, and localization, often reduces to computing statistics, or *features*, from observe data and comparing them in some high-dimensional embedding space, where the Euclidean distance is rarely meaningful. Thus, many kernels have been developed to emphasize different aspects of similarity, such as the intersection of two high-dimensional histograms. Kernel-based similarity is usually more costly to evaluate than the Euclidean distance and becomes impractical as database size and dimensionality grow, which often encourages designers of classification schemes to perform generic dimensionality-reduction as a pre-processing step before discrimination takes place. Such dimensionality-reduction, unless specifically tied to the task at hand [21], usually reduces discriminative power. Therefore, we focus on making kernel comparisons efficient by compressing the database as a whole, rather than datapoints.

Similarity search in high-dimensional spaces is usually divided into two steps: (i) The dataset is *filtered* to rule out a large percentage that is unlikely to be similar to the test datum, leaving a relatively small *candidate set*, where (ii) exact comparison with the test datum is performed via *reranking*. Filtering (i) is commonly tackled by hashing each object into a short code and performing lookups in the resulting hash table [1, 6, 8, 11, 12, 19, 24]. Reranking (ii) is usually the bottleneck when the data space has dimension in the hundreds to thousands. Depending on the similarity function used, the complexity of each similarity computation is at least directly proportional to this dimension. Additionally, filtering (i) must return a large candidate set of similar objects to achieve high recall rates.

(a) Query (b) Top five hits

Figure 1. This paper presents an algorithm capable of determining the top 100 nearest neighbors in a database of approximately 79 million 384-dimensional GIST descriptors in under 2.5 $s$ on a single core processor.

Therefore, in this manuscript we address (ii) by proposing a method to perform approximate nearest neighbor (ANN) computations in high-dimensional space, relative to an arbitrary kernel, with an asymmetric similarity score. This is done for the purpose of comparing high-dimensional feature descriptors that do not live in linear spaces, so the use of kernels is often necessary. In addition, our method can be applied to accelerate sparse coding [15], and any other application where frequent computation of similarities in high-dimensional spaces are required and the use of kernels is beneficial.

The computation of the asymmetric kernel approximation is described in Sect. 2.3, after we place our contribution in the context of existing work (Sect. 1.1). We compare the performance of our approach to the state-of-the-art in an ANN task on benchmark datasets in Sect. 3.

### 1.1. Related work and our contributions

The theory of metric embeddings [18] provides methods to embed certain metric spaces into computationally tractable spaces, while achieving low distortions. For instance, the Johnson-Lindenstrauss lemma [10] exploits the concentration of measure phenomenon of the Gaussian distribution to produce random projections that map $n$ points in $\ell_2^d$ to $\ell_2^{\Omega(8\ln(n)/\varepsilon^2)}$, with distortion $\varepsilon$. These mappings are fast to evaluate, but the resulting approximation is often outperformed by data-dependent methods that can capture the data's statistics. This is especially the case for natural images that are concentrated on a "small volume" of the embedding space of all possible images. Furthermore, the resulting embeddings are symmetric, so a comparison of two points must happen in the compressed domain, which can introduce additional approximation error.

Many statistics of interest are normalized. For instance, histograms of local gradient orientations such as SIFT, SURF, HOG, CHOG, PHOG and other image descriptors are elements of the sphere $S^{d-1}$ and are compared with the cosine similarity function. For the cosine similarity, Product Quantization (PQ) [9] provides an asymmetric embedding of data points into indices of centroids obtained from $k$-means clustering. PQ breaks a vector into $\alpha$ groups of components, performs $k$-means on each group, and then, at approximation time, it estimates the cosine similarity between a novel point and a point in the database asymmetrically by performing $\alpha$ lookups, one for each of the closest centroids to the data point. This method can provide a good approximation to the cosine similarity, but it requires an extremely costly $k$-means "training" step. In addition, its representative power is limited because (iii) it only constructs a *piecewise constant* approximation of the dataset, (iv) it only takes advantage of the *marginal* distribution of the data and (v) it cannot directly be applied to *arbitrary kernels.* To partially alleviate (v), the authors of [3] first construct an explicit embedding of the dataset into $\ell_2$ by performing Kernel Principal Component Analysis (KPCA) before PQ. As we have argued, generic dimensionality-reduction typically yields complexity benefits at the expense of discriminative power [5], which we wish to avoid.

For general kernels, kernelized locality sensitive hashing (KLSH) [11] combines the central limit theorem and locality sensitive hashing (LSH) [8] to generate approximately Gaussian-distributed vectors in feature space. Then, it uses the sign of the dot product with these vectors to construct an embedding into the Hamming cube $\{0,1\}^d$, where $d \gg 0$. The Hamming distance between the image of two points under this embedding is asymptotically proportional to their angle similarity under the chosen kernel. Sparse kernel approximations were recently used to provide significant speed up in image-based classification and detection tasks in [23], where it is shown that PQ can be cast as a sparse kernel approximation.

We wish to overcome the limitations described above in the efficient computation of ANNs by (iii) computing a *piecewise affine* approximation of the dataset (iv) that takes into account the *joint distribution* of the data and is, therefore, adapted to the task and that (v) can be applied with an *arbitrary kernel.* Our method does not require learning (*e.g.* clustering, as in [9]). It can be applied without pre-processing the dataset to reduce its dimensionality[1] and, therefore, captures the natural statistics of the data. We emphasize that our work addresses (ii) and is, therefore, complementary to any indexing structure, such as KLSH, because these filtering algorithms (i) all require an exact kernel computation on a returned list of candidates.

We describe our algorithm in Sect. 2.3, after introducing some background notions. Our asymmetric approximate kernel is in Eq. (12), which is followed by a description of its features compared to existing methods: It provides an efficient kernel approximation for data in high-dimensional spaces and yields a natural compression scheme through sparse embeddings. It also naturally takes advantage of the joint distribution of the data without requiring learning, clustering, or pre-processing via dimensionality reduction. In particular, we show that our algorithm outperforms PQ with a dictionary of random datapoints. Our algorithm also applies to arbitrary reproducing kernel Hilbert spaces, without need for intermediate approximations, such as KPCA. Because of asymmetric nature of our approximation, only the dataset, not the query set, needs to be compressed, and the achieved compression is significant: We show that equivalent ANN performance can be achieved using a significantly smaller memory footprint, and the compression ratio does not depend on the size of the dictionary. In addition, the speed of our algorithm is essentially constant across different kernels.

### 1.2. Background: kernel embeddings

A kernel embedding [18] is any map between kernel spaces that preserves similarities between pairs of points. Kernel embeddings can be used to convert a computationally challenging problem into a tractable one. We focus on kernel embeddings that convert computationally expensive kernel evaluations in one space into computationally tractable kernel evaluations in another.

It is not always possible to find an embedding between two kernel spaces. Thus, we relax the definition to include approximate embeddings with an allowable degree of distortion. Given two kernel spaces $(M, K)$ and $(M', K')$, an *$\varepsilon$-embedding* of $(M, K)$ into $(M, K')$ is a map $f : M \to$

[1] Of course, our method can *also* be applied after generic dimensionality reduction if one so wishes; we discuss this issue in Sect. 3.

$M'$ for which

$$K(x,y) - \varepsilon \leq K'(f(x), f(y)) \leq K(x,y) + \varepsilon, \quad (1)$$

for all $x, y \in M$. We call $\varepsilon$ the *distortion* of the embedding. Approximate metric embeddings were introduced to quantify the distortion of an embedding.

In addition to the symmetric $\varepsilon$-embedding in equation (1), we introduce the following asymmetric variant: Suppose $g$ and $h$ are maps between kernel spaces $M$ and $M'$. We call the relaxation of equation (1) to

$$K(x,y) - \varepsilon \leq K'(h(x), g(y)) \leq K(x,y) + \varepsilon, \quad (2)$$

for all $x, y \in M$, an *asymmetric $\varepsilon$-embedding*. This definition is useful when we have a subset, $Y \subseteq M$, that we would like to search, but computing $g$ is costly, while computing $h$ and $K'(h(x), g(y))$ is not. Thus, to efficiently search $Y$ we compute and store the image, $g(y)$, of each data point and compute comparisons in the range, $M'$. Ideally, $g(M)$ has a smaller memory footprint than $M$. Note that PQ is an asymmetric embedding with $h : \mathbb{R}^d \to \mathbb{R}^d$ given by the identity, while $g$ is the concatenation of several subquantizers.

## 2. Sparse kernel approximations

Suppose our data lives on the unit sphere, $S^{d-1} \subseteq \mathbb{R}^d$, and we have a dictionary matrix, $D = [z_1, \cdots, z_m] \in \mathbb{R}^{d \times m}$. If $Y$ is our dataset, our aim is to map each $y \in Y$ to an $\alpha$-sparse point, $\Phi^D(y) \in \mathbb{R}^m$, such that $y \approx D\Phi^D(y)$. Because our goal is to approximate the inner product, $\langle x, y \rangle$, for any $x \in S^{d-1}$, Cauchy-Schwartz implies that we should minimize $\|y - D\Phi^D(y)\|$:

$$\begin{aligned} |\langle x, y \rangle - \langle x, D\Phi^D(y) \rangle| &= |\langle x, y - D\Phi^D(y) \rangle| \quad (3) \\ &\leq \|y - D\Phi^D(y)\|. \quad (4) \end{aligned}$$

Now, if the residual $\|y - D\Phi^D(y)\|$ is small, the following simple observation is key to our paper:

$$\langle x, y \rangle \approx \langle x, D\Phi^D(y) \rangle = \langle D^T x, \Phi^D(y) \rangle. \quad (5)$$

This produces an asymmetric $\sup_{y \in Y}\{\|y - D\Phi^D(y)\|\}$-embedding with $g(y) = \Phi^D(y)$ and $h(x) = D^T x$. If we precompute $D^T x$, we reduce the complexity of computing the dot product in $\mathbb{R}^d$ to $\alpha \ll d$ additions. Despite its simplicity and the computational benefits it affords, to the best of our knowledge, this fact has not been exploited in the literature. In the following sections, we develop this observation in the context of sparse kernel approximations.

### 2.1. Reproducing kernel Hilbert spaces (RKHS)

Given a set, $X$, a reproducing kernel on $X$ is a map, $K : X \times X \to \mathbb{R}$, such that $K(x,y) = \langle \Phi(x), \Phi(y) \rangle_{\mathcal{H}}$ for some $\Phi : X \to \mathcal{H}$, to a Hilbert space, $\mathcal{H}$. We call $\mathcal{H}$ a *feature space*. If $K$ is positive definite, the existence of $\Phi$ is guaranteed by Moore-Aronszajn's theorem [2]. Now, suppose that $K$ is a normalized kernel function on $X$, i.e. $K(x,x) = 1$ for $x \in X$. We consider distances on $X$ induced by $K : X \times X \to \mathbb{R}$:

$$\begin{aligned} d(x,y)^2 &= K(x,x) + K(y,y) - 2K(x,y) \quad (6) \\ &= 2(1 - K(x,y)). \quad (7) \end{aligned}$$

Because $K$ implicitly represents an inner product in $\mathcal{H}$, $d$ is a true metric on $X$. Thus, any approximation of the metric, $d$, naturally induces an approximation of $K$ and vice versa.

From now on, we work in the general setting: let $Y \subseteq X$ and $x \in X$. From (7), we see that $d(x,y)$ is minimal when $K(x,y)$ is maximal. Thus, a *nearest neighbor* of $x$ in $Y$ is any point $y \in Y$ such that $K(x,y)$ is maximal.

### 2.2. Symmetric kernel embeddings

Suppose we are given a dictionary set, $D = \{z_1, \cdots, z_m\} \subseteq X$, and a kernel, $K$. One way to provide an approximation for arbitrary kernels is to approximate the mapping, $\Phi$. The Hilbert space, $\mathcal{H}$, associated to $K$ may be infinite-dimensional. Thus, $\Phi$ is infeasible to compute explicitly. In this case, we opt to approximate $\mathcal{H} = \text{span}(\Phi(X))$ with a finite-dimensional subspace, $\mathcal{H}_D \subseteq \mathcal{H}$:

$$\mathcal{H}_D = \text{span}\{\Phi(z_1), \cdots, \Phi(z_m)\}. \quad (8)$$

Thus, we seek a map, $\Phi^D : X \to \mathcal{H}_D$, so that $K(x,y) \approx K_D(x,y) = \langle \Phi^D(x), \Phi^D(y) \rangle_{\mathcal{H}_D}$.

Sparse kernel approximations [23] provide a sparse approximate embedding into feature space by solving the optimization:

$$\Phi^D(x) = \arg \min_{\substack{\Phi' \in \mathbb{R}^m \\ \|\Phi'\|_0 \leq \alpha}} \left\| \Phi(x) - \sum_{i=1}^{m} \Phi(z_i)\Phi'_i \right\|_{\mathcal{H}}^2. \quad (9)$$

Approximate solutions can be generated using sparse recovery algorithms, such as orthogonal matching pursuit (OMP) [17], that only need to compute inner products in $\mathcal{H}$. The vectors, $\Phi^D(x)$, produced by this embedding are sparse and high-dimensional. If we increase the size of the dictionary, $D$, we can only increase the approximation quality of the resulting vectors. In addition, a larger $D$ requires no extra storage for fixed sparsity $\Phi^D(x)$. Unfortunately, these large sparse vectors are not directly comparable in terms of inner products. Instead, if $D_G \in \mathbb{R}^{m \times m}$ is the Gram matrix of $D$, $\Phi^D(x)^T D_G \Phi^D(y) \approx K(x,y)$.

KPCA [20] performs a dense $m' \leq m$ dimensional embedding of the dataset into $\mathcal{H}_D$ by choosing the $m'$ dimensional subspace, $\mathcal{H}'_D$, that best approximates $\mathcal{H}_D$ in the least-squares sense. Note that KPCA aims to find a single

subspace that best explains all of the data, whereas sparse kernel approximations choose the $\alpha$-dimensional subspace that best approximates each data point.

### 2.3. Asymmetric sparse kernel approximations (ASKA)

If the sparse vectors, $\Phi^D(x)$, were directly comparable, we could compute inner products in, at most, $\alpha$ additions. However, note that, in general, we do not wish to compare points within the (training) dataset but want to compare a test datum (query) to each element of the dataset. In addition, we do not enforce that two similar points $x$ and $y$ even have similar encodings: The inner product between two high-dimensional sparse vectors could easily be zero.

In the beginning of Section 2, we showed that the sparse vectors, $\Phi^D(x)$, were directly comparable with the images of query points under the adjoint map, $D^T$. In the case of kernels, the adjoint map takes the form

$$\begin{aligned} x &\mapsto K(x, D) && (10)\\ &= (K(x, z_1), \cdots, K(x, z_m)). && (11)\end{aligned}$$

Thus, if $Y \subseteq X$ is a dataset and $x \in X$ a query, we can compute $K(x, y)$, for $y \in Y$ via

$$K(x, y) \approx \langle K(x, D), \Phi^D(y)\rangle_{\mathcal{H}_D}. \quad (12)$$

This follows because

$$\begin{aligned} K(x, y) &\approx \langle \Phi(x), \sum_{i=1}^{m} \Phi(z_i)\Phi_i^D(y)\rangle_{\mathcal{H}_D} && (13)\\ &= \sum_{i=1}^{m} \Phi_i^D(y)K(x, z_i) && (14)\\ &= \langle K(x, D), \Phi^D(y)\rangle_{\mathcal{H}_D}. && (15)\end{aligned}$$

By using sparse kernel approximations, we can save considerably when $|Y| \gg m$. In addition to providing a fast kernel approximation, if $\alpha$ is sufficiently small, the sparse embedding, $\Phi^D(x)$, produces a natural compression scheme. The advantages of ASKA (12) can be summarized as follows:

1. The search time is essentially constant across different kernels. Indeed, after computing $K(x, D)$ ($m$ kernel operations, where $m \ll |Y|$), we can approximate $\langle x, y\rangle$ with an $\alpha$-sparse Euclidean dot product, $\langle K(x, D), \Phi^D(y)\rangle_{\mathcal{H}_D}$.

2. It takes advantage of the joint distribution of the data through the dictionary, $D$.

3. It requires no prior learning, clustering, or dimensionality reduction: In Sect. 3 we show that it outperforms PQ with a dictionary of random exemplars from the dataset. In addition, as previously noted by [4], for the purposes of classification, dictionaries built from random exemplars perform just as well as dictionaries learned from the data.

4. It applies to arbitrary RKHS kernels without an intermediate approximation step, such as KPCA.

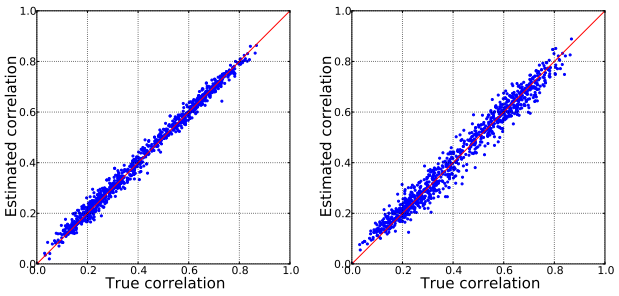
5. It does not compress query points, leading to faster search and better approximation.

6. It can greatly reduce the memory usage of the original dataset by a factor of $O(\frac{d}{\alpha})$. In addition, the compression ratio does not depend on the size of the dictionary, $D$. Thus, one can achieve better approximation, while maintaining the same compression ratio.

7. It is a piecewise linear model, so it is necessarily more expressive than a piecewise constant model, such as PQ. Indeed, for PQ, one may need to greatly increase $k$ (see Table 1) in order to place a subquantizer near enough to each projected data point.

8. The encoding of any point can be computed quickly with inexpensive algorithms, such as OMP [17], while achieving a good approximation.

The main disadvantage of the sparse kernel approximation is that it does not map objects to a discrete set. Instead, it maps objects to the list of non-zero components and coefficient values of the sparse representation. This still achieves high compression with less accuracy loss as shown in Tab. 2 and Fig. 2.




(a) ASKA (b) PQ

Figure 2. SIFT1M comparison of (a) ASKA with $K(x, y) = \langle x, y\rangle$, $m = 1024$ and $\alpha = 8$, and (b) PQ with 8 subquantizers. The mean square errors are $3.8 \cdot 10^{-6}$ and $1.2 \cdot 10^{-5}$ respectively.

## 3. Experiments

We developed our sparse kernel approximation in C++ using the SPAMS-toolbox [14]. Of the many sparse approximation algorithms, we chose OMP [17] because it is fast ($O(\alpha f(K)m)$ with $f(K)$ the cost of a kernel evaluation), easy to implement, and provides a good approximation with low sparsity levels.

| Cost | ASKA | PQ | KLSH | JL |
|---|---|---|---|---|
| Preprocessing | 0 | $k$-means($\|Y'\|$) | 0 | 0 |
| Data encoding | $O(\alpha dm)$ | $O(\alpha dk)$ | $O(db)$ | $O(dp)$ |
| Memory (bytes) | $O(4\alpha)$ | $O(\alpha \log_2(k)/8)$ | $O(b/8)$ | $O(4p)$ |
| Search (flops) | $O(dm + \alpha\|Y\|)$ | $O(dk + \alpha\|Y\|)$ | $O(db + \frac{b}{32}\|Y\|)$ | $O(dp + p\|Y\|)$ |

Table 1. Various costs associated to each algorithm. Notation: $Y'$ is a training set for $k$-means, $b$ is the number of bits in the KLSH hash code, $p$ is the dimension of the JL projection. For simplicity, we only show the costs for the cosine kernel, $\langle \cdot, \cdot \rangle$. Note that the data encoding cost of ASKA is dependent on the sparse coding algorithm. We used OMP for all of our experiments [17].

### 3.1. Datasets

We utilized three benchmark datasets common in large-scale image search: a medium-scale, SIFT1M [9], consisting of 1 million 128-dimensional SIFT descriptors [13] and 10000 queries; a larger ImageNet-based set of 1.26 million 1000-dimensional bag-of-words (BOW) vectors and their associated class labels from the large scale recognition challenge [7]. Our query set consists of 50000, 1000-dimensional BOW vectors and their associated class labels. Finally, the TinyImages [22] dataset consists of 80 million, 384-dimensional Gist descriptors [16] extracted from $28 \times 28$ images crawled on the web. The TinyImages dataset contains about 1 million constant color images, resulting in many identically zero Gist descriptors. These were removed from the database.

### 3.2. State of the art and evaluation protocol

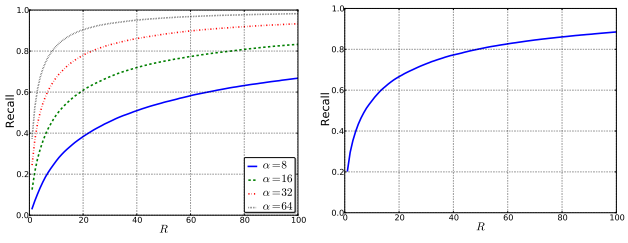
We compared our method with KLSH [11], KPCA followed by Product Quantization [9] (PQ), and a baseline method consisting of KPCA followed by a random projection in the style of the Johnson-Lindenstrauss lemma (JL) [10]. We only compared against methods that generate an approximation of the underlying kernel. Thus, we did not compare against hashing methods which approximate a non-kernel (*e.g.* semantic) similarity. We could compare at either similar memory, or similar computation costs. We chose the latter because comparing at equal storage is unfair to PQ: the computation cost would blow up at the query ($2^{O(\#\text{bytes})}$ flops) and preprocessing ($2^{O(\#\text{bytes})}$-means) stages as shown in Table 1. Throughout the comparisons we set these methods to the default parameters found in the respective papers: KLSH generates a 300-bit hash code, PQ and JL perform kernel PCA with a dictionary of size $m = 8192$ and approximate subspace $\mathcal{H}'_D$ of dimension 128. Then, PQ performs an asymmetric approximate embedding with 8 vector subquantizers. Similarly, JL randomly projects to a 32-dimensional vector space spanned by the columns of a random Gaussian matrix. Note that we chose a 32-dimensional projection so that JL could compete with the other methods. After KPCA, we randomly permuted the components of the database and query set as in [3]. This does not affect the JL projection, but is empirically known to improve the result of PQ [3]. Note that computing the Hamming distance between two 300-bit hash codes is slightly more costly than simply adding 8 floating point numbers.

We evaluated our method on three datasets with four kernels: cosine, chi-square, intersection, and Hellinger. We performed exhaustive nearest neighbors search and evaluated the quality using two scores: Recall@$R$ and recognitionRate@$R$. Recall@$R$ measures the proportion of queries for which the first approximate nearest neighbor appears in the top $R$ true nearest neighbors. The recognitionRate@$R$ metric is computed by performing $k = 5$ nearest neighbor classification and calculating whether or not the correct class label appears among the top $R$ most frequently appearing class labels.

We reiterate that ASKA is designed to address (ii), so it is complementary to any indexing structure that addresses (i). Thus, we we do not attempt to evaluate it in conjunction with such a scheme.

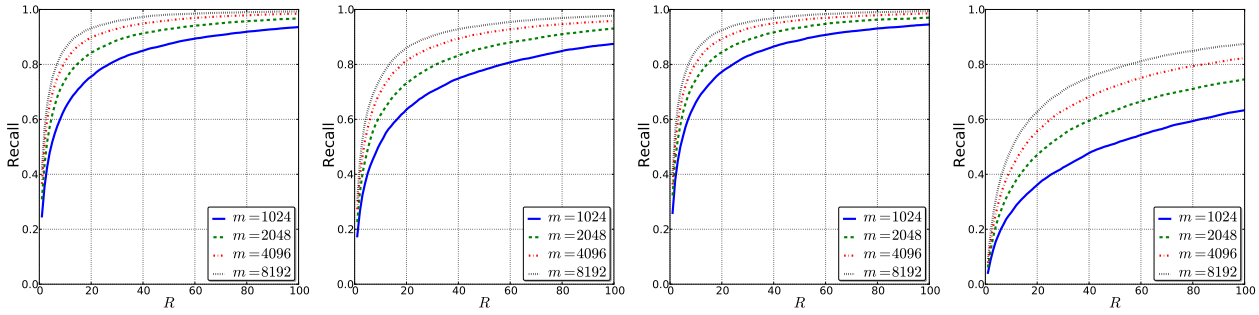
### 3.3. Nearest neighbors search: effect of dictionary size and sparsity

Fig. 4 shows the effect of the dictionary size $m$: All but the intersection kernel performed at an acceptable level with $m = 1024$. This is probably due to the substantial non-linearity present in the "min" operation. We confirmed the statement of [3] that performance of KPCA + PQ saturates




(a) $\chi^2$-kernel (b) $\chi^2$-kernel

Figure 3. ImageNet: (a) Recall@R for ImageNet with $m = 8192$ and $\alpha \in \{8, 16, 32, 64\}$. (b) Recall@R for 128-dimensional KPCA.




(a) Cosine kernel (b) $\chi^2$-kernel (c) Hellinger kernel (d) Intersection kernel

Figure 4. SIFT1M recall@R for $m \in \{1024, 2048, 4096, 8192\}$ and $\alpha = 8$.

beyond $m = 1024$. This makes sense because KPCA forces a single 128-dimensional subspace to explain all of the data. In contrast, ASKA achieved better approximation with increasing dictionary size, while maintaining the same compression ratio.

In Fig. 3(a), we experimented with different sparsity levels and a fixed dictionary size on ImageNet. As expected, the sparsity level dramatically affects the approximation quality: Fig. 3(b) shows that we outperformed 128-dimensional KPCA with sparsity $\alpha = 32$.

### 3.4. Comparisons with other methods

Figs. 5 and 6 compare our performance with other methods. In all of our experiments we fixed a dictionary with $m = 8192$ columns randomly chosen from the database and a sparsity level $\alpha = 8$. Tab. 2 shows that we achieved 60-fold compression of ImageNet, while still outperforming each of the other methods. At query time, we performed the exact same number of operations as KPCA + PQ, less operations than KPCA + JL, and slightly less operations than KLSH to compute the kernel (although KLSH does not require computing $K(x, D)$ for each query).

Fig. 7 plots the recognition rates we obtained using different kernels. Sometimes, performing KPCA on the data prior to classification improves accuracy. We showed that applying KPCA with kernel $K$, followed by applying ASKA with the cosine kernel, achieves higher recognition rates than all other methods. It is interesting to note that the cosine nearest neighbors classifier performs worse than the other kernels that are specifically tailored to histograms.

| | SIFT1M | ImageNet | Tiny Images |
|---|---|---|---|
| Original file size | 516 MB | 5.1 GB | 121.7 GB |
| Compressed file size | 67.8 MB | 85.8 MB | 5.31 GB |
| Compression ratio | 7.61 | 60.86 | 22.9 |

Table 2. File compressions achieved by sparse coding, while outperforming the other methods.

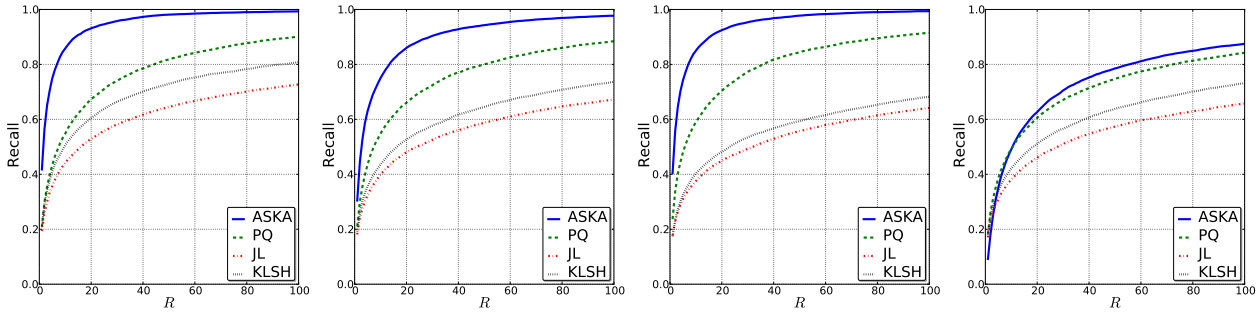
### 3.5. TinyImages

Fig. 8 shows the results we obtained by exhaustive search with a dictionary of size $m = 8192$ and sparsity $\alpha = 8$. Because there is no ground truth set for this database, we displayed the top 5 search results obtained and highlight the instances for which we retrieved the correct nearest neighbor in the database with a green box. The search took $\approx 2.5$ seconds per query to retrieve the 100 nearest neighbors on a single core processor. The encoding of the 79 million, 384-dimensional GIST features took between 2 to 4 hours, per kernel, on a 4-core Intel CPU @ 3.4 GHz, with 12 GB of memory. Note that this time is essentially independent of the kernel used: After computing $K(x, D)$ for a query $x$ ($m = 8192$ kernel evaluations), we can compute the nearest neighbors among the dataset with sparse Euclidean inner products ($\alpha = 8$). In addition, the 22.9-fold compression (Tab. 2) allows storing the database in main memory.

## 4. Conclusion

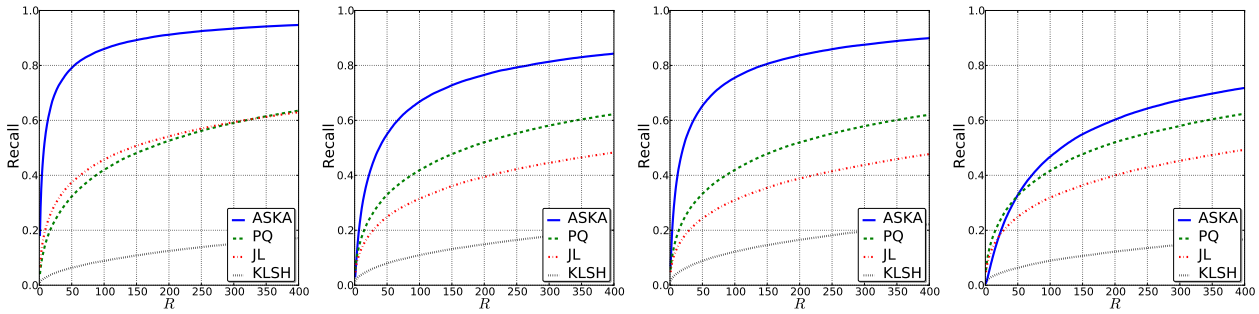
We introduced asymmetric sparse kernel approximations in the context of nearest neighbors search. Our algorithm achieved satisfactory approximations with no training, heavy compression of the original database, and fast query time while outperforming the state of the art. This makes our method particularly suited to image-based image retrieval, where a large number of features living in high-dimensional non-linear spaces must be evaluated at query time. Asymmetric sparse kernel approximations enjoy wide applicability beyond nearest neighbors search. Indeed, many learning algorithms require the asymmetric evaluation of kernels against large collections of fixed database points. Such applications will be the subject of future research. Our code is available at the following URL: http://www.math.ucla.edu/∼damek.

**Acknowledgments:** Research sponsored by ONR N00014-13-1-0563, FA8650-11-1-7156 and NGA HM02101310004. Damek Davis was also sponsored by the NSF Graduate Research Fellowship DGE-0707424.




(a) Cosine kernel (b) $\chi^2$-kernel (c) Hellinger kernel (d) Intersection kernel

Figure 5. SIFT1M recall@$R$ with $m = 8192$ and $\alpha = 8$. The average search time per query was $\approx 21ms$. The average encoding time across kernels was $1.5ms$ per database element. Both timings were achieved on a single core Intel CPU @ 3.4 GHz with 16 GB of memory




(a) Cosine kernel (b) $\chi^2$-kernel (c) Hellinger kernel (d) Intersection kernel

Figure 6. ImageNet recall@$R$ with $m = 8192$ and $\alpha = 8$. The average search time per query was $\approx 40\ ms$. The average encoding time across kernels was $6.2ms$ per database element. Both timings were achieved on a single core Intel CPU @ 3.4 GHz with 16 GB of memory

## References


[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, 2006. 1

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950. 3

[3] A. Bourrier, F. Perronnin, R. Gribonval, P. Pérez, and H. Jégou. Nearest neighbor search for arbitrary kernels with explicit embeddings. Research Report RR-8040, INRIA, Aug. 2012. 2, 5

[4] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning (ICML)*, volume 8, page 10, 2011. 4

[5] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2012. 2

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *ACM Annual Symposium on Computational Geometry*, pages 253–262, 2004. 1

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 5

[8] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing*, pages 604–613, 1998. 1, 2

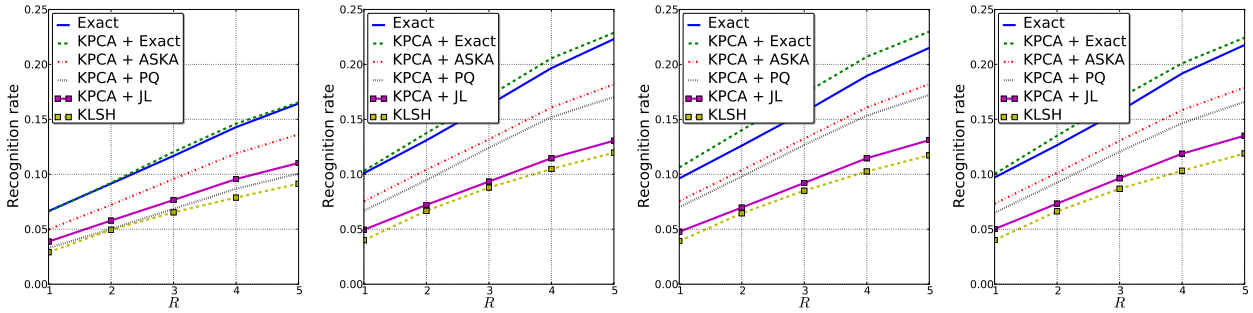
[9] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. 2, 5

[10] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26(1):189–206, 1984. 2, 5

[11] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2130–2137, 2009. 1, 2, 5

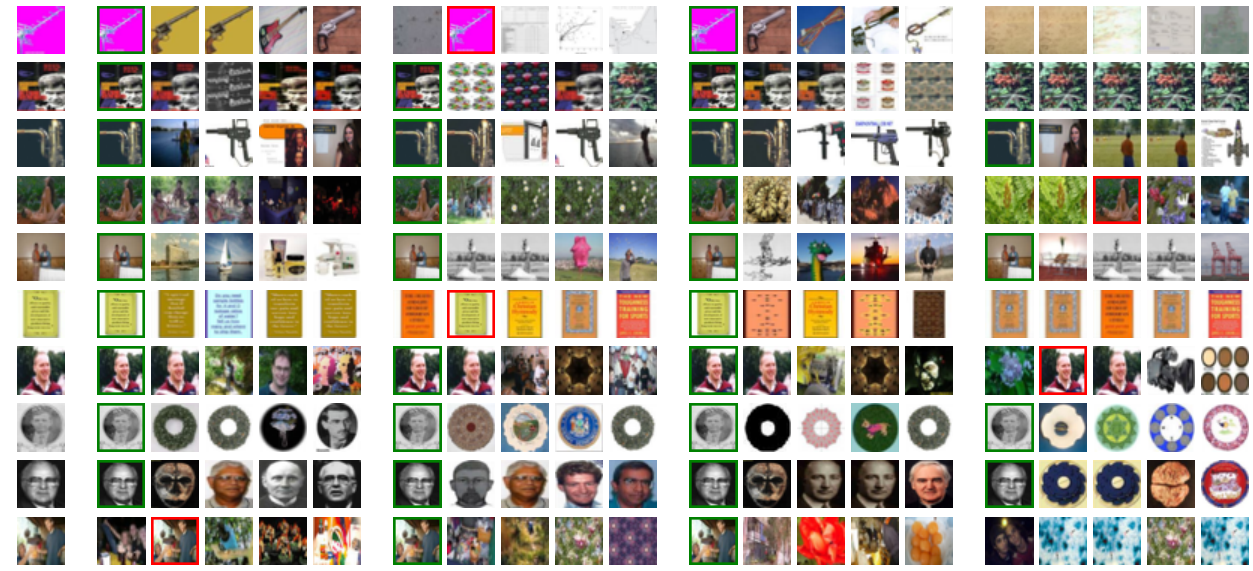
[12] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000. 1

[13] D. G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157, 1999. 5




(a) Cosine kernel (b) $\chi^2$-kernel (c) Hellinger kernel (d) Intersection kernel

Figure 7. ImageNet recognition@$R$ with $m = 8192$ and $\alpha = 8$.



(a) Query (b) Cosine kernel (c) $\chi^2$-kernel (d) Hellinger kernel (e) Intersection kernel

Figure 8. Sparse linear scan with $m = 8192$ and $\alpha = 8$.


[14] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010. 4

[15] Y. Mu, J. Wright, and S.-F. Chang. Accelerated large scale optimization by concomitant hashing. In *European Conference on Computer Vision (ECCV)*, pages 414–427, 2012. 1

[16] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001. 5

[17] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *IEEE Conference on Signals, Systems and Computers (ASILOMAR)*, pages 40–44, 1993. 3, 4, 5

[18] J.-R. Sack and J. Urrutia. *Handbook of Computational Geometry*. North Holland, 1999. 2

[19] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009. 1

[20] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998. 3

[21] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Conference on Communication and Computation*, pages 368–377, 1999. 1

[22] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008. 5

[23] A. Vedaldi and A. Zisserman. Sparse kernel approximations for efficient classification and detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2320–2327, 2012. 2, 3

[24] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS)*, 2008. 1