Nearest Neighbors Using Compact Sparse Codes

Anoop Cherian

Nearest Neighbors Using Compact Sparse Codes

Anoop Cherian ANOOP.CHERIAN@INRIA.FR
INRIA, LEAR Project-team, Grenoble, France

Abstract

In this paper, we propose a novel scheme for approximate nearest neighbor (ANN) retrieval based on dictionary learning and sparse coding. Our key innovation is to build compact codes, dubbed *SpANN codes*, using the active set of sparse coded data. These codes are then used to index an inverted file table for fast retrieval. The active sets are often found to be sensitive to small differences among data points, resulting in only near duplicate retrieval. We show that this sensitivity is related to the coherence of the dictionary; small coherence resulting in better retrieval. To this end, we propose a novel dictionary learning formulation with incoherence constraints and an efficient method to solve it. Experiments are conducted on two state-of-the-art computer vision datasets with 1M data points and show an order of magnitude improvement in retrieval accuracy without sacrificing memory and query time compared to the state-of-the-art methods.

1. Introduction

In the past few years, there has been a huge increase in the amount of user generated data in the web. On the one hand, availability of such big data offers tremendous opportunities for creating new applications. On the other hand, organizing such enormous data, so that it is easily and promptly accessible, is a challenge. In this respect, a core component in several algorithms that work in accessing such big data is that of approximate nearest neighbors (ANN), in which the goal is to retrieve a subset of a dataset that is most similar to a query. A few examples in computer vision are image retrieval (Fergus et al., 2009), and object recognition (Nister & Stewenius, 2006). Often these applications require accessing billions of data points (Jégou et al., 2011), which is difficult without efficient ANN schemes.

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

In this paper, we introduce a new ANN retrieval algorithm based on the idea of *sparse data representations*. We assume that our high dimensional data inhabit a low dimensional structure which can be learned in a data-driven way. Towards this end, we use the idea of *Dictionary Learning* (DL) (Aharon et al., 2006). DL learns an overcomplete dictionary from the data space such that each data point is k-sparse in this dictionary. The main idea put forth in this paper is to design compact codes, dubbed *Sparse ANN codes* (SpANN) using these k-sparse vectors. In our new representation, each data vector is encoded as an integer set corresponding to the indices of the support of its sparse code. Our scheme is related to Min-Hash (Broder, 1997) and can be used for fast data access using an inverted index (Chum et al., 2007; 2008). At the same time, sparsity helps to approximately represent data in very low dimensions (about one-fourth to one-ninth of the original data size in our experiments), and thus our storage requirement is low.

Sparsity has been suggested as a promising method for ANN in the recent past (Cheng et al., 2012; Zepeda et al., 2010; Cherian et al., 2012). A major difficulty affecting the performance of these methods, as well as our scheme proposed above, is that the SpANN codes generated by dictionaries learned using traditional DL algorithms are often found to be sensitive to small differences in the data points. Recall that DL is a data partitioning technique that partitions dense regions of the data space into multiple non-orthogonal subspaces. DL methods (such as K-SVD (Aharon et al., 2006)) do not impose any constraints on the coherence between the dictionary atoms; large coherence allow neighboring data points to use different active sets in their sparse codes, resulting in the retrieval of only near duplicates. We analyze this issue theoretically and propose a novel DL algorithm, dubbed *incoherent DL* (IDL), that uses additional incoherence constraints between atoms. We show that our new optimization objective can be minimized efficiently. Our experimental results on benchmark datasets demonstrate that our IDL formulation makes the sparse codes more robust and improve ANN retrieval accuracy rivaling the state of the art.

To set the context for our later discussions, we will review

in the next section, prior literature on the problem of ANN retrieval. Before we proceed, let us first introduce our notations.

Notations: We use $\mathcal{I}_p = \{1, 2, \cdots, p\}$, that is, the first p natural numbers. The notation $\widehat{\boldsymbol{a}, \boldsymbol{b}}$ stands for the angle between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. We will use $\mathbf{I}_n$ for the $n \times n$ identity matrix.

2. Related Work

It is well-known that when the dimensionality of data increases as high as 20, the performance of traditional ANN algorithms such as k-d trees, R-trees, etc. deteriorates (Beyer et al., 1999). Of the several schemes to tackle this difficulty, one particularly effective method is locality sensitive hashing (LSH) (Indyk & Motwani, 1998) that uses specialized hash functions to map the data to compact binary strings. Typically, the hash family selected for LSH is independent of the structural properties of the data; knowledge of which might help generate more compact hash codes. To this end, Spectral Hashing (Weiss et al., 2009) suggests minimizing the sum of Hamming distances between pairs of data points weighted by a Gaussian kernel. In (Liu et al., 2011; Raginsky & Lazebnik, 2009), extensions to SH are provided showing better retrieval performance especially for increasing code lengths. More recently, there have been several machine learning approaches to suggested to address ANN via LSH (Kulis & Grauman, 2009; Liu et al., 2012; Norouzi & Fleet, 2011; Strecha et al., 2012). Similar to these methods, ours is also a data dependent method that learns a non-uniform tiling of the data space via learning an overcomplete dictionary. In contrast to these methods, which embeds the data into a binary space, we project the data into a set of integers, resulting in two important benefits, (i) while the representational power of LSH based techniques is 2^m for an m-bit binary code, it can be shown that using our compositional model, the same representational power can be attained for k-sparse codes using a dictionary with only $2^{\mathcal{O}(m/k)}$ atoms, and (ii) the actual number of hash bits can be made relatively independent to the length of the active set (assuming $k << m$) by using integer hashing techniques such as Bloom filters (Bloom, 1970) without loss in accuracy.

A different direction in which the ANN problem has been tackled is using vector quantization. In these methods, data points are approximated by their nearest cluster centroids, typically learned using k-means on the training data (Tuytelaars & Schmid, 2007; Winder et al., 2009). The main advantage of such methods is that the centroids can be precomputed and queried very fast using inverted files. A drawback is that often a large number of centroids are required for high recall. To circumvent this problem, (Jegou et al., 2011) proposes product quantization (PQ) in which the dimensions of the data are represented as a Cartesian product of independent subspaces, each subspace relatively low dimensional. In (Norouzi & Fleet, 2013), orthogonal subspaces are learned which are then combined to reconstruct a query point. In (Babenko & Lempitsky, 2012), an enhancement to PQ is suggested by using only two subspaces, but using an efficient algorithm for computing query distances from the subspaces. While, almost all these methods assume that either the centroids or the subspaces are uncorrelated, we allow the basis to be correlated (via overcompleteness), thereby taking advantage of the redundancy between subspaces to generate compact codes. Further, we need to store only the sparse coefficients (instead of the centroids) which is significantly compressed.

Building similarity metrics on sparse codes has been investigated several times in the recent past (Klenk & Heidemann, 2009; Cheng et al., 2012). Unfortunately, the adequacy of these metrics for retrieval problems is not thoroughly investigated. Sparse coding for image retrieval has been suggested in (Yang et al., 2009; Wang et al., 2010; Boix et al., 2012). The goal of these methods is visual understanding, while we target retrieval at the descriptor level. To take an example, (Boix et al., 2012) proposes a two level binary encoding scheme for image retrieval, but their final representation does not preserve any relation to the euclidean distance between the image descriptors. A sparse coding framework that approximates the inner-product distance between data points for retrieval is presented in (Zepeda et al., 2010). Their method is hindered by the instability of sparse codes, for which they provide heuristic solutions. More recently, (Cherian et al., 2012) proposes a hashing framework similar to our approach, but requires solving a robust optimization problem for preserving the locality of sparse codes. In (Zhu et al., 2013), a graph Laplacian, obtained from a training set, is used to enforce locality, followed by canonical correlation analysis for producing stable and generalizable hash codes. In contrast to these methods, our experiments demonstrate that our incoherent dictionary learning formulation implicitly encourages generalizability in a much simpler setting.

3. Sparse Coding and ANN Retrieval

To set the stage for further discussions, we will review the basics of dictionary learning and sparse coding in this section. Later, we will establish a connection between the euclidean distances in the original data space and the sparse representations. First, let us formally define data *sparsity*.

Definition 1 (k-sparse)**.** *Given a data point* $\mathbf{y} \in \mathbb{R}^d$*, a mapping* $S : \mathbb{R}^d \to \mathbb{R}^n$ *for* $n \gg d$*, and a loss function*

$\mathcal{L} : \mathbb{R}^d \times \mathbb{R}^n \rightarrow \mathbb{R}$, we define $S(\mathbf{y})$ to be k-sparse, if $|\operatorname{Supp}(S(\mathbf{y}))| = k, (k \ll d)$ and $\mathcal{L}(\mathbf{y}, S(\mathbf{y})) \leq \delta$, $(\delta > 0)$ where Supp *represents the support function and $|\,.\,|$ defines the set cardinality. We will denote a k-sparse representation of a data point $\mathbf{y}$ as $S_k(\mathbf{y})$.*

Let $\mathbf{y} \in \mathbb{R}^d$ be a data point and given a dictionary $\mathbf{B} \in \mathbb{R}^{d \times n}$ with $\mathbf{b}_i, i \in \mathcal{I}_n$ as its columns such that $\mathbf{x} = S_k(\mathbf{y})$ in $\mathbf{B}$. Then, finding $\mathbf{x}$ can be cast as the following optimization problem,

$$S_k(\mathbf{y}) := \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{B}\mathbf{x}\|_2^2 \text{ subject to } \|\mathbf{x}\|_0 = k. \quad (1)$$

It is well-known that solving (1) is NP-hard, but a sub-optimal solution can be obtained using greedy schemes such as *orthogonal matching pursuit* (OMP) (Tropp & Gilbert, 2007). Although, (1) can be solved more optimally using its closest convex relaxation via the ℓ_1 norm (Efron et al., 2004), most of these techniques are computationally expensive, which is a primary concern for problems such as ANN.

In (1), we assumed that $\mathbf{B}$ is known, which is seldom true in general problems. To address this difficulty, DL techniques have been suggested, in which one learns a dictionary from the data itself by solving a variant of (1), but with $\mathbf{B}$ also as an unknown. Given a training dataset $v_i, i \in \mathcal{I}_N$, the DL problem is cast as follows:

$$\min_{\mathbf{B}, \mathbf{x}_i, i \in \mathcal{I}_N} \sum_{i=1}^{N} \|\mathbf{y}_i - \sum_j \mathbf{x}_i^j \mathbf{b}_j\|_2^2 + \lambda \|\mathbf{x}_i\|_1, \quad (2)$$

where $\mathbf{x}_i^j$ denotes the j-th dimension of the coefficient vector $\mathbf{x}_i$ and λ is a suitable regularization. The atoms are generally constrained to have unit norm to avoid scaling issues. Although (2) is non-convex, it is convex in either $\mathbf{B}$ or $\mathbf{x}_i$'s (thanks to the ℓ_1 regularization, instead of ℓ_0) and thus can be solved efficiently to a local minimum using block coordinate descent (Aharon et al., 2006).

One property of the dictionary that is important to our scheme is *coherence*, μ, defined as:

$$\mu = \max_{i \neq j} |\mathbf{b}_i^T \mathbf{b}_j|, \ \forall i, j \in \mathcal{I}_n. \quad (3)$$

Now, we have all the ingredients necessary to establish a connection between ANN retrieval and sparse coding.

Theorem 1 (Distances)**.** *Let $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^d$ be two zero-mean data points, $\mathbf{B} \in \mathbb{R}^{d \times n}$ be a dictionary with coherence μ, and let $\|\mathbf{y}_i - \mathbf{B}\mathbf{x}_i\|_2^2 \leq \delta_i^2$, for k-sparse vectors $\mathbf{x}_i$, $i \in \{1, 2\}$. Then,*

$$\|\mathbf{y}_1 - \mathbf{y}_2\|_2 \leq \delta_1 + \delta_2 + \sqrt{1 + (n-1)\mu} \|\mathbf{x}_1 - \mathbf{x}_2\|_2. \quad (4)$$

Proof. Assuming spherical balls with centers $\mathbf{t}_i = \mathbf{B}\mathbf{x}_i$, with locus $\mathbf{y}_i$, and radii δ, it can be shown that

$$\|\mathbf{y}_1 - \mathbf{y}_2\|_2 \leq \max_{\|\mathbf{r}_1\|_2 = \|\mathbf{r}_2\|_2 = 1} \|\mathbf{t}_1 - \delta_1 \mathbf{r}_1 - \mathbf{t}_2 - \delta_2 \mathbf{r}_2\|_2, \quad (5)$$

$$\leq \delta_1 + \delta_2 + \|\mathbf{B}\|_2 \|\mathbf{x}_1 - \mathbf{x}_2\|_2, \quad (6)$$

where the last result is obtained from the definition of $\mathbf{t}_i$'s, followed by applying the triangle inequality and the Cauchy-Schwartz inequality. From Gershgorin's theorem, we have $\|\mathbf{B}\|_2^2 \leq \|\mathbf{B}^T\mathbf{B}\|_\infty$ which can be upper-bounded by $1 + (n-1)\mu$ using the definition of μ. Substituting this upper-bound instead of $\|\mathbf{B}\|_2$ in (6), we have the desired result. □

Building on Theorem 1, we will now introduce our compact data representation for NN retrieval.

4. Sparse ANN Codes

As is clear from our previous discussions, each atom in the dictionary represents a non-orthogonal subspace. Sparse coding seeks a new subspace that is a sparse linear combination of the atoms that hosts a given data vector. If the data vector is exactly k-sparse in $\mathbf{B}$, then there is a unique combination of k subspaces from the dictionary for this data vector as formally stated in the following proposition which can be proved directly by invoking a contradiction.

Proposition 1. *If $\mathbf{y} = \sum_{i=1}^{k} \mathbf{x}^j \mathbf{b}_{\mathcal{A}_j}$, where $\mathcal{A}_j$ denotes the j-th atom in a basis active set $\mathcal{A}$ and if $\mu < 1$, then OMP will find this active set uniquely.*

This implies that the k atoms can be used as *representatives* for a subset of the data, thus producing a data partitioning as shown in Figure 1. Our main idea is to build compact ANN codes using these atoms, dubbed *Sparse ANN codes* (SpANN).

Definition 2 (SpANN codes)**.** *Suppose for a data point $\mathbf{y}$, let the k-sparse code be $S_k(\mathbf{y})$. Then, the* SpANN code $\operatorname{SpANN}_k(\mathbf{y})$ *is defined as the set of k indices corresponding to the support of $S_k(\mathbf{y})$.*

To make our representation clear, let us take a simple example.

Example 1. *Assume $\mathbf{y} \in \mathbb{R}^{100}$ and let $\mathbf{x} = S_3(\mathbf{y})$ where $\mathbf{x} \in \mathbb{R}^{1000}$. Let $\operatorname{Supp}(\mathbf{x}) = \{\mathbf{x}^{25}, \mathbf{x}^{49}, \mathbf{x}^{972}\}$ correspond to the non-zero dimensions in $\mathbf{x}$. Then, we have $\operatorname{SpANN}_3(\mathbf{y}) = \{25, 49, 972\}$.*

There exists several efficient methods for encoding integer sets into binary bits (such as Bloom filters (Bloom, 1970)), enabling efficient access using an inverted file. A drawback our representation is that SpANN codes are invariant to data scaling, as formally stated below.

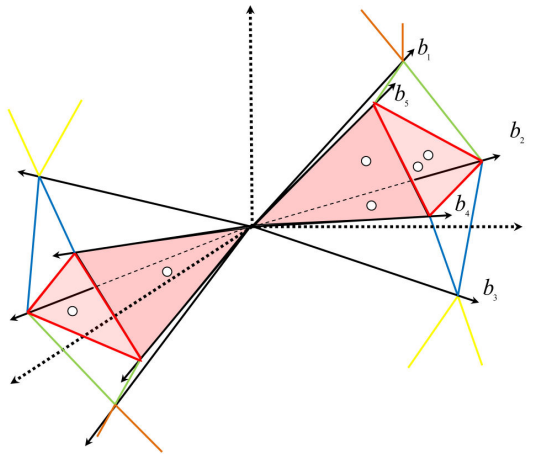


Figure 1. A schematic representation of the non-uniform tiling of the data space introduced by SpANN codes for 3-sparse case. The dotted lines indicate the data space axis, while the solid lines indicate the basis. The figure highlights the region in which a 3-sparse code will use only basis $\mathbf{b}_2, \mathbf{b}_4, \mathbf{b}_5$ in its SpANN representation.

Proposition 2. *For* $\mathbf{y} \in \mathbb{R}^d$, $\text{SpANN}(\mathbf{y}) = \text{SpANN}(w\mathbf{y})$, *for* $w \in \mathbb{R}\backslash\{0\}$.

However, by storing the sparse codes associated with all the data points having the same SpANN code in an inverted file bucket, the ANN can be found using a linear scan on these sparse codes using Theorem 1. As observed in (Gordo & Perronnin, 2011), using an asymmetric distance in which the query point is not sparse coded, is empirically seen to provide better retrieval performance. It is assumed that each such inverted file bucket will contain only a few data points mapped into it. Note that, we need to store only the sparse codes instead of the original data points, thus enabling compressed storage.

The following theorem establishes a connection between the SpANN codes and the angle between the corresponding data vectors. Since our codes are invariant to data scaling, we assume the data vectors are unit normalized.

Theorem 2. *For any two zero-mean unit-norm data vectors* $\mathbf{y}_1$ *and* $\mathbf{y}_2$, *let* $\mathbf{x}_i = S_k(\mathbf{y}_i)$ *and* $T_i = \text{SpANN}_k(\mathbf{y}_i)$ *for* $i \in \{1, 2\}$. *Suppose* $\|\mathbf{y}_1 - \mathbf{y}_2\|_2 \leq \delta_1 + \delta_2$ *where* $\|\mathbf{y}_i - \mathbf{B}\mathbf{x}_i\|_2 \leq \delta_i$. *Then, for a probability defined on the volume of overlap of two hypershperes with centroids* $\mathbf{y}_i$, *locus* $\mathbf{t}_i = \mathbf{B}\mathbf{x}_i$, *and radii* δ, $Prob\left(|T_1 \cap T_2| = k\right)$ monotonically increases *with decreasing* $\widehat{\mathbf{y}_1, \mathbf{y}_2}$.

Proof. The proof is a direct extension of (Gromov, 1987)[Theorem 1.A]. □

Theorem 2 says that as the angle between the data points decreases, there is an increasing probability that their SpANN codes match exactly. From another perspective, the theorem suggests that by using $|T_1 \cap T_2| \geq \eta$, for some $\eta < k$–that is, to look at neighboring subspaces of $\mathbf{y}_1$ that overlaps with at least η subspaces of $\mathbf{y}_2$– will have a non-zero probability of finding the ANN. Using this intuition, we will rank the SpANN codes of the database points according to their basis overlap with the SpANN codes of a query point, and will explore the inverted file buckets of only those codes that overlaps by more than θ indices. Towards this end, we can in fact use the more standard *Jaccard distance* (Jaccard, 1901), given by $|T_1 \cap T_2|/|T_1 \cup T_2|$, for comparing the SpANN codes, for which efficient hashing schemes exist for fast lookup (Chum & Matas, 2010). Algorithm 1 summarizes the steps involved for populating the inverted file and query retrieval using SpANN codes.

Algorithm 1 SpANN Indexing and Retrieval

Input: $\mathbf{B}$, Y, k, θ, H {an inverted index}
Output: $\mathbf{y}^*$
for all $\mathbf{y}_i \in Y$ **do**
  Compute $\mathbf{x}_i = S_k(\mathbf{y}_i)$ and $T_i = \text{SpANN}_k(\mathbf{y}_i)$.
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_i\}$
  $H(T_i) \leftarrow H(T_i) \cup \mathbf{x}_i$ {assign a location to $\mathbf{x}_i$ in the hash bucket $H(T_i)$}
end for

Given a query q, compute $T_q = \text{SpANN}_k(q)$,
$\hat{\mathcal{T}} \leftarrow \text{Jaccard}(T_q, h) \geq \eta, \forall h \in \mathcal{T}$
$\mathbf{x}^* \leftarrow \text{argmin}_{\forall \mathbf{x} \in H(\hat{\mathcal{T}})} \|q - \mathbf{B}\mathbf{x}\|_2$.
$\mathbf{y}^*$ associated with $\mathbf{x}^*$.

The next theorem connects the subspace overlap with the coherence of the dictionary.

Theorem 3. *Let* $\mathbf{y}_1, \mathbf{y}_2$ *be two zero-mean unit norm data points and let* $\mathbf{B}$ *be a dictionary with coherence* μ. *Let* $T_i = \text{SpANN}_k(\mathbf{y}_i)$ *for* $i \in \mathcal{I}_2$. *If* $T_1 \neq T_2$, *then there exists a* k', $1 \leq k' < k$ *and* $T_i' = \text{SpANN}_{k'}(\mathbf{y}_i)$, *such that* $T_1' = T_2'$ if and only if *there exists* $\mathbf{b}_j$ *for which* $|\mathbf{y}^T\mathbf{b}_j| > \sqrt{\frac{1}{2}(1+\mu)}$.

Proof. Let $cos\theta = \mu$ so that $\sqrt{\frac{1}{2}(1+\mu)} = cos\frac{\theta}{2}$. To prove the *if* part: suppose $\exists \mathbf{b}_i$ such that $|\mathbf{b}_i^T\mathbf{y}| > \sqrt{\frac{1}{2}(1+\mu)}$, then $\widehat{\mathbf{b}_i, \mathbf{y}} < \frac{\theta}{2}$. This means $\mathbf{b}_i$ is the most correlated dictionary atom to $\mathbf{y}$ and thus will be selected by OMP in the first step. If not, there must exist another basis $\mathbf{b}_j$ such that this condition is satisfied. In such a case, $\mathbf{b}_i^T\mathbf{b}_j < \mu$, which contradicts the condition that the coherence of the dictionary is μ. To prove the *only if* part: assume $\exists k' : 1 \leq k' < k$ such that $T_1 = T_2 = T = \{i_1, i_2, \cdots, i_{k'}\}$. Without loss of generality, let $\mathbf{b}_{i_1}$ be the most correlated atom to $\mathbf{y}$ and thus will be selected by OMP in its first step, from which the result follows. □

Theorem 3 shows that the sensitivity of SpANN codes is related to the coherence of the dictionary. Lower coherence implies larger apex angles for the atoms. As a result, their chance to be included in active sets of neighboring data points is higher. Traditional DL schemes do not allow any control on the coherence (empirically, they are seen to cohere by more than 90% for the data used in our experiments). This motivates us to build a novel formulation of (2) that incorporates incoherence constraints.

5. Incoherent Dictionary Learning

The problem of learning incoherent dictionaries has been dealt with through heuristic approximations (Yaghoobi et al., 2010) using parametric models, or in a theoretical setting with stochastic assumptions on the dictionary (Arora et al., 2013). Another class of methods incorporate additional incoherence regularizations of the form $\|\mathbf{B}^T\mathbf{B} - I\|_{\mathrm{F}}^2$ into the DL objective (Lin et al., 2012; Ramirez et al., 2010). However, these methods have two shortcomings, namely (i) they do not offer any explicit control over coherence and (ii) empirically, it is often found to be difficult to reduce the coherence to arbitrarily low values. To circumvent these problems, in this paper, we propose a novel DL formulation that incorporates incoherence explicitly. Before showcasing our formulation, we review a result on the minimum coherence achievable by any overcomplete dictionary.

Proposition 3. *For any dictionary* $\mathbf{B} \in \mathbb{R}^{d\times n}$ *such that* $n > d$*, assuming* $d = dim(span(\mathbf{B}))$*, the minimum coherence* μ_{min} *is given by:*

$$\mu_{min} = \mu(\mathbf{B}) \geq \sqrt{\frac{n-d}{d(n-1)}}. \tag{7}$$

Proof. See (Benedetto & Kolesar, 2006)[Theorem IV.2]. □

Now, let us reformulate (2) with constraints on the maximum coherence between the atoms:

$$\begin{aligned} \min_{\mathbf{B},X}\ & \frac{1}{2}\sum_{i=1}^{N}\|\mathbf{y}_i - \sum_{j=1}^{n}\mathbf{b}_j\mathbf{x}_i^j\|_2^2 \ +\ \boldsymbol{\lambda}\|\mathbf{x}_i\|_1 \\ \text{subject to}\quad & \|\mathbf{B}_{\sim k}^T\mathbf{b}_k\|_\infty \leq \gamma; \forall k \in \mathcal{I}_n, \end{aligned} \tag{8}$$

where $\mathbf{B}_{\sim k}$ is the dictionary $\mathbf{B}$ with the k-th atom removed, and γ is a constant; $\mu_{min} \leq \gamma \leq 1$ and controls the allowed dictionary coherence. Recall that $\|\ v\ \|_\infty$ corresponds to the maximum of the absolute values of entries in an input vector v. Compared to the traditional DL formulation, the only non-trivial part in (8) is to solve for $\mathbf{B}$ which we consider in details next.

We can rewrite the DL part of (8) as:

$$\begin{aligned} \min_{\mathbf{B}}\ & \frac{1}{2}\sum_{i=1}^{N}\|\mathbf{y}_i - \sum_{j=1}^{n}\mathbf{b}_j\mathbf{x}_i^j\|_2^2 \ +\ \beta\|\mathbf{x}_i\|_1 \\ \text{subject to}\ & \|\mathbf{B}_{\sim k}^T\mathbf{b}_k\|_\infty \leq \gamma; \forall k \in \mathcal{I}_n \end{aligned} \tag{9}$$

As is clear, this DL problem is non-convex, but interestingly, is convex in each dictionary atom while keeping the rest of the atoms fixed. This suggests an additional block coordinate descent scheme for each atom separately. The subproblem for solving the atom $\mathbf{b}_k$ has the following form:

$$\begin{aligned} \min_{\mathbf{b}_k}\ & \frac{1}{2}\sum_{i=1}^{N}\|\mathbf{y}_i - \mathbf{B}_{\sim k}\mathbf{x}_i^{\sim k} - \mathbf{b}_k\mathbf{x}_i^k\|_2^2 \\ & \|\mathbf{B}_{\sim k}^T\mathbf{b}_k\|_\infty \leq \gamma. \end{aligned} \tag{10}$$

With a slight abuse of notation to avoid writing too many $\sim$, let $\tilde{\mathbf{B}} = \mathbf{B}_{\sim k}$ and $\tilde{\mathbf{z}}_i = \mathbf{x}_i^{\sim k}$. Further, let us generically refer the dictionary atom under consideration by dropping the subscript k, then we concisely rewrite (10) as:

$$\begin{aligned} \min_{\mathbf{b}}\ & \frac{1}{2}\sum_{i=1}^{N}\|\mathbf{y}_i - \tilde{\mathbf{B}}\tilde{\mathbf{z}}_i - \mathbf{b}z_i\|_2^2 \\ \text{subject to}\ & \tilde{\mathbf{B}}^T\mathbf{b} \leq \gamma\mathbf{1}, -\tilde{\mathbf{B}}^T\mathbf{b} \leq \gamma\mathbf{1}, \end{aligned} \tag{11}$$

where the scalar $z_i = \mathbf{x}_i^k$. Further, let $\tilde{\mathbf{y}}_i = \mathbf{y}_i - \tilde{\mathbf{B}}\tilde{\mathbf{z}}_i$, then taking the Lagrangian of (11) using dual vectors $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \in \mathbb{R}^n \geq 0$, followed by setting its derivative w.r.t. $\mathbf{b}$ to zero, we have $\mathbf{b}$ in terms of the dual variables as:

$$\mathbf{b} = \left(\sum_{i=1}^{N} z_i^2\right)^{-1}\left\{\sum_{i=1}^{N}\tilde{\mathbf{y}}_i\mathbf{z}_i + \tilde{\mathbf{B}}^T(\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_1)\right\}. \tag{12}$$

Substituting $\mathbf{b}$ in the Lagrangian, we have the following dual for (11):

$$\min_{\boldsymbol{\lambda}\geq 0}\ \frac{1}{2}a\boldsymbol{\lambda}^T\boldsymbol{\Lambda}^T\tilde{\mathbf{B}}\tilde{\mathbf{B}}^T\boldsymbol{\Lambda}\boldsymbol{\lambda} + a\boldsymbol{\lambda}^T\boldsymbol{\Lambda}^T\tilde{\mathbf{B}}\boldsymbol{\beta} + \gamma\|\boldsymbol{\lambda}\|_1. \tag{13}$$

where, $a = \left(\sum_i z_i^2\right)^{-1}$, $\boldsymbol{\beta} = \sum_i \tilde{\mathbf{y}}_i\mathbf{z}_i$, the matrix $\boldsymbol{\Lambda} = [\mathbf{I}_n, -\mathbf{I}_n]$, and $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_1^T, \boldsymbol{\lambda}_2^T]^T$. Introducing $\mathbf{A} = \tilde{\mathbf{B}}^T\boldsymbol{\Lambda}$, we can rewrite (13) more concisely as:

$$\min_{\boldsymbol{\lambda}\geq 0}\ \frac{1}{2}a\|\mathbf{A}^T\boldsymbol{\lambda}\|_2^2 + a\boldsymbol{\lambda}^T\mathbf{A}^T\boldsymbol{\beta} + \gamma\|\boldsymbol{\lambda}\|_1. \tag{14}$$

The form in (14) is a convex ℓ_1 regularized non-negative least squares problem and can be solved efficiently using standard toolboxes (Kim et al., 2012). The various steps of the IDL algorithm are summarized in Algorithm 2. Due to the coupling of the dictionary atoms with each other through the incoherence constraints, the dictionary learning

sub-problem is non-convex and thus convergence to a local minima (8) is not guaranteed (similar to other approaches such as (Ramirez et al., 2010)). Empirically, it is seen to converge in less than 20 iterations for the inner loop and in about 100 iterations for the outer loop in all our experimental datasets.

Algorithm 2 IDL Algorithm

Input: Y, $k = 0$, $\mathbf{B}_k$\{ initial dictionary\}
Output: $\mathbf{B}_{out}$
repeat
  $X \leftarrow \min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{B}_k\mathbf{x}_i\|_2^2 + \beta\|\mathbf{x}_i\|_1, \forall i \in \mathcal{I}_N$
  $p \leftarrow 0$, $\hat{\mathbf{B}}_p = \mathbf{B}_k$
  repeat
    for all $j \in \mathcal{I}_n$ **do**
      $\mathbf{b}^j \leftarrow$ from (14) and (12)
      $\hat{\mathbf{B}}_p^j \leftarrow \frac{\mathbf{b}^j}{\|\mathbf{b}^j\|_2}$ {update the j-th atom }
    end for
    $p \leftarrow p + 1$
  until $\|\hat{\mathbf{B}}_p - \hat{\mathbf{B}}_{p-1}\|_{\mathrm{F}} \leq \epsilon$
  $\mathbf{B}_{k+1} \leftarrow \hat{\mathbf{B}}_p$
  $k \leftarrow k + 1$
until *convergence*
$\mathbf{B}_{out} \leftarrow \mathbf{B}_k$

6. Experiments

In this section, we will first introduce our evaluation datasets and benchmark metrics, preceding which, we will be furnishing our results comparing against several state-of-the-art ANN methods.

Datasets and Evaluation Metric: Our experiments are mainly based on the evaluation protocol of (Jegou et al., 2011) using two publicly available ANN datasets: (i) 1M SIFT and (ii) 1M GIST descriptors. The first dataset is split into a training set with 100K 128-dimensional SIFT descriptors, a base set of 1M descriptors to be queried, and 10K query descriptors. Of the 100K training set, we use a random sample of 90K descriptors for learning the dictionary and 10K for validation. The GIST dataset consists of 960-dimensional descriptors and a training, database, and query step split of 500K, 1M, and 1K respectively. Of the training set, we use 400K descriptors for DL and 100K for validation. We also use the 1B BIGANN dataset (Jégou et al., 2011) consisting of one billion SIFT descriptors, which we use to evaluate our query time performance. We will use Recall@K for evaluating our algorithms (as in (Jegou et al., 2011)), which is defined as the proportion of the queries for which the ground truth neighbor is found in the first K retrieved points. For $K = 1$, this measure corresponds to the standard precision measure.

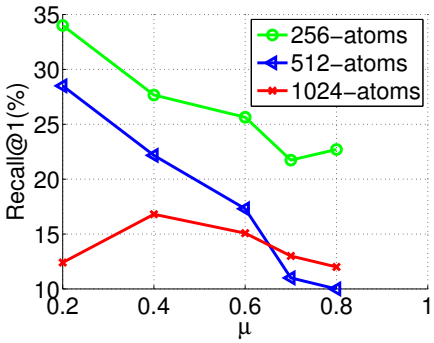


Figure 2. Recall@1 on SIFT using SpANN codes against increasingly coherent dictionaries learned using IDL for various sizes. We used 64 bit SpANN codes for all plots.

There are two parameters in our ANN scheme, namely (i) the number of dictionary atoms, and (ii) the dictionary incoherence, which are important for the performance of our method. In the following, we empirically investigate the sensitivity of these parameters against ANN accuracy. We use the SIFT validation set for these experiments, and use separate 1K descriptors from the validation set as queries. For all the experiments, we used a fixed Jaccard threshold of $\eta = 0.33$.

Evaluation of IDL: In Figure 2, we plot the Recall@1 against an increasing dictionary size and varying maximum coherences with IDL. The plot shows the accuracy for dictionary sizes 256, 512, and 1024, and maximum coherence ranging from 0.2 to 0.8 at steps of 0.1. As is clear from the plot, increasing coherence results in lower ANN accuracy. We found $\mu = 0.2$ gave the best performance for dictionaries of sizes 256 and 512, while $\mu = 0.3$ performed best for 1024 atoms. We use these coherences for the respective dictionaries in the experiments to follow. For highly overcomplete dictionaries (such as for SIFT with 1024 atoms), the IDL algorithm was found to converge slowly for incoherences closer to the theoretical minimum. This is not unexpected as finding optimal basis directions respecting the incoherences in such settings is difficult. We also found that for alternative incoherence models such as (Ramirez et al., 2010), the incoherence could not be decreased below 0.6 (for SIFT descriptors) even for large regularizations.

Increasing Active Set Size: Figure 3 plots the Recall@1 for increasing active set size and varying dictionary sizes on our SIFT validation set. If n is the number of atoms in a dictionary, then we use $\lceil \log_2(n) \rceil$ bits to encode each integer in the active set to access the inverted file table. For all data points having the same active set, we store M floating point sparse active coefficients for each point, in the disk for the linear scan (note that we do not need to store the

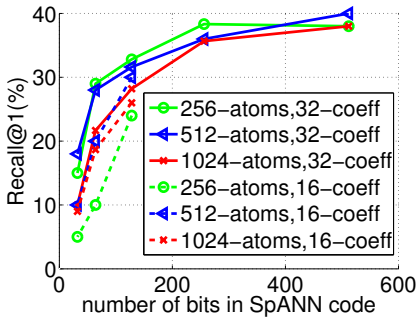


Figure 3. Recall@1 against increasing number of bits for SIFT (i.e., increasing active-set size), for different dictionary sizes and number of sparse coefficients stored on the disk (for linear scan).

original descriptors). In the figure, we plot the recall for $M = 16$ and $M = 32$ corresponding to the largest M coefficients in the sparse codes for the database descriptors. As expected, the plot shows that the recall increases with an increasing SpANN code length and larger values of M. We use $M = 32$ for our SIFT experiments and $M = 64$ for GIST.

Recall against K for SIFT: We now present comparisons against the state-of-the-art methods. We compare our method to (i) Spectral Hashing (SH) (Weiss et al., 2009), (ii) Product Quantization using 16 cells (PQ-IVFADC@16) (Jegou et al., 2011), (iii) PQ with asymmetric linear scan (PQ-ADC), (iv) Shift Invariant Kernel Hashing (SIKH) (Raginsky & Lazebnik, 2009), (v) Iterative Quantization (ITQ) (Gong & Lazebnik, 2011), and (vi) Cartesian K-Means (CKMeans-AD). The results for these methods are taken from their respective publications (except SIKH, which is our implementation). Figure 4(a) compares the Recall@K of SpANN codes generated using 256 atoms against these methods. All the methods (including ours) used 64 bits for hashing. As is clear from the plot, SpANN codes outperform the next best method (PQ-IVFADC@16) by about 12% at Recall@1, while performing competitively for higher values of K. Note that, methods such as PQ-ADC (that show better Recall@1000) uses an exhaustive search on the entire dataset to achieve high recall, while our method uses only about 2% (20,061 out of 1M points) to achieve 83.1% recall, which we believe is a significant gain. In Figure 4(b), we compare the Recall@K against other sparse ANN methods such as (i) (Cherian et al., 2012), (ii) using a dictionary learned with the classical DL formulation, and (iii) using an incoherent dictionary learned using (Ramirez et al., 2010). All these other methods used 256-atom dictionaries and 64-bit codes. We also compare to other sizes of the dictionaries, demonstrating performance significantly better than other sparse ANN methods.

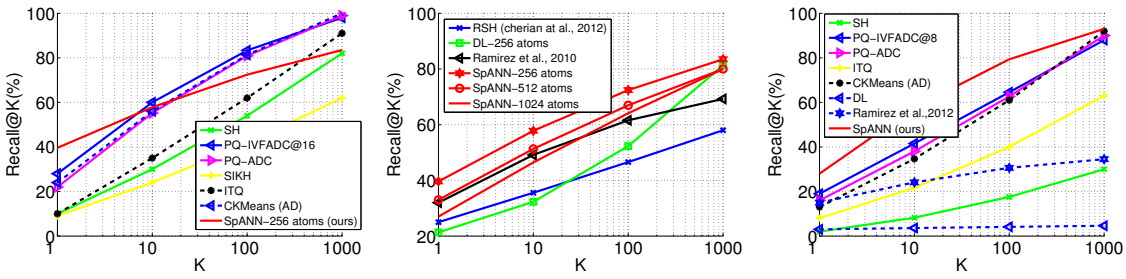
Dataset	# atoms	# (R@1)	# (R@100)
1M SIFT	256	3,769	9,760
1M GIST	1024	10,354	84,234

Table 1. Avg. number of code comparisons in the inverted table for Recall@K (R@K) on 1M SIFT and 1M GIST datasets with 64-bit SpANN codes.

Recall against K for GIST: In Figure 4(c), we show the Recall@K performance for 960D GIST descriptors. Through the validation set, we found a dictionary of size 1024 atoms demonstrated an optimum trade off between sparse coding time and recall accuracy. In the figure, we plot the Recall@K for K varying from 1 to 1000, and compare it with other state-of-the-art methods listed above. We also show the retrieval performance of (i) coherent dictionary (DL) learned using the classical DL, and (ii) incoherent DL learned using (Ramirez et al., 2010). All the methods used 64-bit codes, while the DL methods used the same number of atoms. We find that our SpANN codes outperform the next best method (PQ-IVFADC@8) by about 9% at Recall@1 and by about 21% at Recall@100. We point out that the performance of classical DL was found to be poor (recall@K less than 4%) for all K on this dataset. Plots for all comparison methods are taken from their published works.

Retrieval Time: We used the BIGANN dataset for this experiment which consists of one-billion SIFT descriptors. For improving query performance on this huge dataset, we initially used K-Means with 1K centroids to partition the dataset, followed by using separate hash tables for each partition to index the SpANN codes. The codes are binary encoded using Bloom filters so that computing Jaccard overlap is reduced to finding Hamming distances. While querying, we first find the K-Means centroid nearest to a given query, later querying the inverted indices in the corresponding data partition. For K-NN retrieval, we might need to look at multiple such partitions. To further improve the efficiency, we used a query expansion scheme loosely based on the method of (Chum & Matas, 2010) in which we query indices in the order of their probability of being in the active set (found using the training set).

Our timing comparisons are based on a single core 2.7 GHz AMD processor with 32GB memory. We used the SPAMS toolbox (Mairal et al., 2010) for sparse coding. It took a few microseconds on average to sparse code each SIFT descriptor for all our dictionary sizes, while it took less than a millisecond for each GIST descriptor. In Figure 5, we show the average query time for SpANN codes ($\sim$64 bits) when the database size increases from 1M to 1000M. Our implementation was primarily in MATLAB with optimized C++ routines for accessing the inverted file table.



(a) 1M SIFT-64 bits (b) 1M SIFT-64 bits (c) 1M GIST-64 bits

Figure 4. (a) Recall@K for 1M SIFT descriptors against state-of-the-art ANN methods, (b) Recall@K for 1M SIFT descriptors against other sparse ANN methods, and (c) Recall@K for 1M GIST descriptors against state-of-the-art ANN methods and using dictionary learned using other methods.

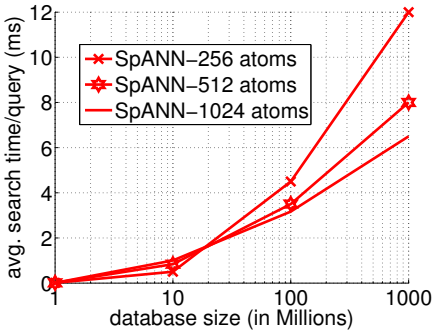


Figure 5. Avg. query time for 64-bit SpANN codes on the 1B SIFT dataset. Increasing the dictionary size spreads the SpANN codes in the inverted list; as a result, the query time improves.

The plot shows that our method offers competitive query time when compared to results reported in earlier works such as (Jegou et al., 2011) on the same dataset. In Table 1, we show the average number of data points accessed from the inverted table for the two datasets.

7. Conclusions

This paper introduced SpANN codes for efficient approximate nearest neighbors using sparse coding. After exploring several theoretical properties of ANN using SpANN codes, we showed that their robustness is tied to the incoherence of the sparse coding dictionary. Using this observation, we proposed a novel incoherent dictionary learning formulation and an efficient method to solve it. Our experiments demonstrated the adequacy of our scheme for efficient ANN retrieval. An issue that remains to be addressed is when data is very sparsely distributed in space; as a result, there might not be sufficient overlap between SpANN codes of nearby data points. One way to tackle this problem is to use hierarchical dictionary learning, investigations into such a scheme is an interesting future work.

Acknowledgements

The author would like to thank Dr. Julien Mairal, Dr. Vassilios Morellas, and several anonymous reviewers for useful feedback.

References

Aharon, M., Elad, M., and Bruckstein, A. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *TSP*, 54(11):4311–4322, 2006.

Arora, S., Ge, R., and Moitra, A. New algorithms for learning incoherent and overcomplete dictionaries. *arXiv preprint arXiv:1308.6273*, 2013.

Babenko, A. and Lempitsky, V. The inverted multi-index. In *CVPR*, 2012.

Benedetto, J. J. and Kolesar, J. D. Geometric properties of grassmannian frames for r 2 and r 3. *Journal on Advances in Signal Processing*, 2006, 2006.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. When is nearest neighbor meaningful? *Database Theory*, pp. 217–235, 1999.

Bloom, B. H. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7): 422–426, 1970.

Boix, X., Roig, G., Leistner, C., and Van Gool, L. Nested sparse quantization for efficient feature coding. In *ECCV*, pp. 744–758. Springer, 2012.

Broder, A. Z. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*. IEEE, 1997.

Cheng, H., Liu, Z., Hou, L., and Yang, J. Sparsity induced similarity measure and its applications. In *TCSVT*, 2012.

Cherian, A., Morellas, V., and Papanikolopoulos, N. Robust sparse hashing. In *ICIP*. IEEE, 2012.

Chum, O. and Matas, J. Large-scale discovery of spatially related images. *PAMI*, 32(2):371–377, 2010.

Chum, O., Philbin, J., Isard, M., and Zisserman, A. Scalable near identical image and shot detection. In *CIVR*. ACM, 2007.

Chum, O., Philbin, J., and Zisserman, A. Near duplicate image detection: Min-Hash and TF-IDF weighting. In *BMVC*, 2008.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. Least angle regression. *Annals of Statistics*, 32(2):407–451, 2004.

Fergus, R., Weiss, Y., and Torralba, A. Semi-supervised learning in gigantic image collections. In *NIPS*, 2009.

Gong, Y. and Lazebnik, S. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

Gordo, A. and Perronnin, F. Asymmetric distances for binary embeddings. In *CVPR*, 2011.

Gromov, M. Monotonicity of the volume of intersection of balls. *Geometrical Aspects of Functional Analysis*, pp. 1–4, 1987.

Indyk, P. and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. *Theory of Computing*, pp. 604–613, 1998.

Jaccard, P. Etude comparative de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Socit vaudoise des Sciences Naturelles*, 37:547–579, 1901.

Jegou, H, Douze, M, and Schmid, C. Product quantization for nearest neighbor search. *PAMI*, 33(1):117–128, 2011.

Jégou, H., Tavenard, R., Douze, M., and Amsaleg, L. Searching in one billion vectors: re-rank with source coding. In *ICASSP*, pp. 861–864, 2011.

Kim, D., Sra, S., and Dhillon, I. S. A non-monotonic method for large-scale non-negative least squares. *Optimization Methods and Software*, 2012.

Klenk, S. and Heidemann, G. A sparse coding based similarity measure. In *DMIN*, 2009.

Kulis, B. and Grauman, K. Kernelized locality-sensitive hashing for scalable image search. In *CVPR*, 2009.

Lin, T., Liu, S., and Zha, H. Incoherent dictionary learning for sparse representation. In *ICPR*, 2012.

Liu, W., Wang, J., Kumar, S., and Chang, S. Hashing with graphs. In *ICML*, 2011.

Liu, W., Wang, J., Ji, R., Jiang, Y., and Chang, S. Supervised hashing with kernels. In *CVPR*, 2012.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. Online learning for matrix factorization and sparse coding. *JMLR*, 11:19–60, 2010.

Nister, D. and Stewenius, H. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

Norouzi, M. and Fleet, D. J. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

Norouzi, M. and Fleet, D. J. Cartesian k-means. In *CVPR*, 2013.

Raginsky, M. and Lazebnik, S. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, 2009.

Ramirez, I., Sprechmann, P., and Sapiro, G. Classification and clustering via dictionary learning with structured incoherence and shared features. In *CVPR*, 2010.

Strecha, C., Bronstein, A., Bronstein, M., and Fua, P. LDAHash: Improved matching with smaller descriptors. *PAMI*, 34(1):66–78, 2012.

Tropp, J. and Gilbert, A. Signal recovery from random measurements via orthogonal matching pursuit. *TIT*, 53 (12):4655–4666, 2007.

Tuytelaars, T. and Schmid, C. Vector quantizing feature space with a regular lattice. In *ICCV*, 2007.

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

Weiss, Y., Torralba, A., and Fergus, R. Spectral hashing. In *NIPS*, 2009.

Winder, S., Hua, G., and Brown, M. Picking the best daisy. In *CVPR*, 2009.

Yaghoobi, M., Daudet, L., and Davies, M. Structured and incoherent parametric dictionary design. In *ICASSP*, 2010.

Yang, J., Yu, K., Gong, Y., and Huang, T. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*. IEEE, 2009.

Zepeda, J., Kijak, E., and Guillemot, C. Approximate nearest neighbors using sparse representations. In *ICASSP*, 2010.

Zhu, X., Huang, Z., Cheng, H., Cui, J., and Shen, H. Sparse hashing for fast multimedia search. *ACM TIS*, 31(2):9, 2013.