Hardware Compliant Approximate Image Codes

Da Kuang
Georgia Institute of Technology
Atlanta, GA 30332
da.kuang@cc.gatech.edu

Alex Gittens
University of California, Berkeley
Berkeley, CA 94720
gittens@alumni.caltech.edu

Raffay Hamid
DigitalGlobe Inc
Seattle, WA 98103
raffay@cc.gatech.edu

Abstract

In recent years, several feature encoding schemes for the bags-of-visual-words model have been proposed. While most of these schemes produce impressive results, they all share an important limitation: their high computational complexity makes it challenging to use them for large-scale problems. In this work, we propose an approximate locality-constrained encoding scheme that offers significantly better computational efficiency ($\sim 40\times$) than its exact counterpart, with comparable classification accuracy. Using the perturbation analysis of least-squares problems, we present a formal approximation error analysis of our approach, which helps distill the intuition behind the robustness of our method. We present a thorough set of empirical analyses on multiple standard data-sets, to assess the capability of our encoding scheme for its representational as well as discriminative accuracy.

1. Introduction

Image classification frameworks generally consist of (a) extracting local features (*e.g.*, SIFT [19]), (b) transforming them into more informative codes, and (c) using these codes for classification (*e.g.*, by using linear SVM [5]). Over the years, several different image encoding techniques have been proposed [3] [16] [8] [28] [22] [31] [29] [30]. As reported in [2], given all things equal, most of these encoding schemes tend to produce impressive yet comparable classification accuracies. At the same time however, they can be computationally expensive [2]. Particularly during the testing phase, their complexity can be a significant proportion of image classification pipeline (see Table 1). This limitation often makes it challenging to use these encoding schemes for large-scale learning problems.

	Extract	Assign	Encode	Pool	Test
% Times	6.77%	37.76%	**42.50**%	7.01%	5.93%

Table 1: %-times taken by different steps during testing for LLC [28]. Here D = 128, M = 1024, and K = 10 (§ 3.1).

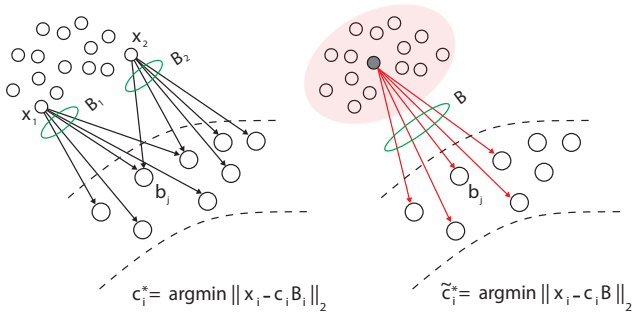


Figure 1: (Left) Locality-constrained encoding [28] finds different sets of bases nearest to each feature to exactly construct its locally-constrained codes. (Right) In contrast, we approximately encode clusters of points simultaneously by using shared sets of bases nearest to the cluster-centroid.

In this work, we propose an approximate locality-constrained [28] encoding scheme which offers significantly better efficiency than its exact counterpart, with comparable classification accuracy.

Our key insight is that for locality-constrained encodings, the set of bases used to encode a point $\mathbf{x}$, can be used equally effectively to encode a group of points similar to $\mathbf{x}$. This observation enables us to approximately encode similar groups of points simultaneously by using shared sets of bases, as opposed to exactly encoding points individually each using their own bases (see Figure 1). This difference improves our encoding efficiency in two important ways:

- It significantly reduces the number of locality related matrices from number of points ($\mathcal{O}$(millions)) to number of point-clusters ($\mathcal{O}$(thousands)).
- It lets us view the encoding problem of each point-group as a linear system with a shared left hand side. Solving such a system can be posed as matrix-matrix multiplication that can fully exploit the cache-efficient modern hardware architecture.

These efficiency advantages enable our approximate scheme to achieve a significant speed-up ($\sim 40\times$) over its exact counterpart, while maintaining comparable accuracy.

The main factor effecting the robustness of our approximation scheme is the cohesiveness of point-clusters sharing sets of bases among their respective members, *i.e.*, more cohesive point-clusters incur less approximation error, and vice versa. We empirically show that for a wide range of cluster granularity, the reconstruction error of our approximation scheme is almost identical to its exact counterpart. Therefore, the efficiency gains of our method come at no extra cost of requiring a more granular feature clustering.

Moreover, we present theocratical bounds on the approximation error of our scheme using perturbation analysis of least-squares problems. This analysis further helps distill the intuition behind the robustness of our technique with respect to the granularity of feature-clusters.

To summarize, the **main contributions** of our work are:

- A simple yet effective approximate encoding scheme with significant performance gains and similar classification accuracy compared to its exact counterpart.
- A formal approximation analysis of our approach using perturbation analysis of least-square problems.
- A thorough set of empirical analyses to assess the capability of our encoding scheme both from a representational as well as a discriminative perspective.

In the following, we begin by going over relevant previous work, followed by presenting the details of our scheme in § 3, and its approximation error analysis in § 4. We show its empirical performance in § 5, and conclude our work in § 6.

2. Related Work

Image-feature codes have so far been mainly looked at from three different perspectives. The basic encoding method introduced by [3] [16] [10] [25] uses vector-quantization to assign each local feature to the most similar of the precomputed feature-clusters (or *visual words*). The hard-quantization adopted by these techniques however suffers from high reconstruction error, resulting in modest classification accuracy when used with linear classifiers.

The second class of encoding methods express features as combinations of *multiple* visual words in order to reduce the reconstruction error and therefore potentially boost the classification accuracy. Examples of this class of methods include soft quantization [8], and locally linear encodings [28] [30]. These techniques tend to produce non-linear spaces, and can therefore produce impressive classification accuracies even when used with linear classifiers.

The third class of encoding schemes uses the *differences* between features and the visual words, as opposed to representing features in terms of the visual words themselves. Example methods in this class include Fisher [24] [22], and super-vector encoding [31], among other approaches.

An important challenge faced by the recent encoding schemes is that they can be computationally expensive [2], which often makes it challenging to use them for large-scale learning problems. Although there has been substantial amount of work to minimize the memory-footprint of feature codes [14] [1] [21] [13], not enough attention has been given towards improving the efficiency of code extraction by a large factor. Our work is geared towards addressing this limitation.

3. Approximate Image Encoding

3.1. Preliminaries

Let $\mathbf{X}$ be a set of D-dimensional local image descriptors, *i.e.*, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_\mathrm{N}] \in \mathbb{R}^{\mathrm{D}\times\mathrm{N}}$. Also, let B define a codebook with M entries, *i.e.*, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_\mathrm{M}] \in \mathbb{R}^{\mathrm{D}\times\mathrm{M}}$. Generally, $\mathbf{B}$ is found by clustering a subset of image descriptors. The main idea behind image-encoding is to use $\mathbf{B}$ as a set of basis vectors to find a more informative representation of $\mathbf{X}$. Let us denote this new representation of the descriptor-set as $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_\mathrm{N}]$.

3.2. The Efficiency-Accuracy Tradeoff

The way in which an ecoding scheme uses $\mathbf{B}$ determines its tradeoff between the efficiency with which it can be computed and the representational accuracy it offers. For instance, vector quantization (VQ), traditionally used in Spatial Pyramid Matching (SPM) [16], optimizes the following cost function to represent $\mathbf{X}$ in terms of $\mathbf{B}$:

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{\mathrm{N}} ||\mathbf{x}_i - \mathbf{B}\mathbf{c}_i||^2 \qquad (1)$$
$$\text{such that, } ||\mathbf{c}_i||_{l^0} = 1, ||\mathbf{c}_i||_{l^1} = 1, \text{and, } \mathbf{c}_i \succeq \mathbf{0}, \forall i$$

This constrained least-squares problem results in each $\mathbf{c}_i$ with only one non-zero element, which in practice is found by searching for the nearest neighbor of $\mathbf{x}_i$ in $\mathbf{B}$.

While efficient to compute, VQ offers high reconstruction error due to its unit cardinality constraint ($||\mathbf{c}_i||_{l^0} = 1$). One way to address this limitation is by using locally linear coding (LLC) [28] that uses a locality constraint instead:

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{\mathrm{N}} ||\mathbf{x}_i - \mathbf{B}_i\mathbf{c}_i||^2 \qquad (2)$$

where $\mathbf{B}_i \in \mathbb{R}^{\mathrm{D}\times\mathrm{K}}$ are the K nearest bases to $\mathbf{x}_i$.

While LLC generates codes with quite high reconstruction accuracy and has emerged as one of the leading image encoding schemes, its accuracy benefits come at the cost of relatively low computational efficiency [2]. We further expand on this point below.

	Level	Memory I/O	Flops	Flop:Byte ratio
Matrix-vector multiplication	BLAS-2	$4n^2$ (Bytes)	$2n^2$	$1/2$
Matrix-matrix multiplication	BLAS-3	$12n^2$ (Bytes)	$2n^3$	$n/6$

Table 2: BLAS-2 versus BLAS-3 operations for matrices of size $n \times n$, single-precision.

3.3. Efficiency Challenges for LLC Encoding

To better understand the efficiency of LLC encoding, let us re-write Equation 2 as the solution of the normal equation:

$$\mathbf{c}_i = (\mathbf{B}_i^{\mathrm{T}}\mathbf{B}_i)^{-1}\mathbf{B}_i^{\mathrm{T}}\mathbf{x}_i, \tag{3}$$

which is computed numerically by solving a linear system with $\mathbf{B}_i^{\mathrm{T}}\mathbf{B}_i$ on the left and $\mathbf{B}_i^{\mathrm{T}}\mathbf{x}_i$ on the right-hand side. Note that Equation 3 poses two main efficiency challenges:

1- Algorithmic Bottleneck: Equation 3 requires computing $\mathbf{B}_i^{\mathrm{T}}\mathbf{B}_i$ and its corresponding Cholesky factorization for each $\mathbf{c}_i$. This adds a cost of $\mathcal{O}(\mathrm{NDK}^2 + \mathrm{NK}^3)$, which can be significant for larger values of N and especially for larger numbers of nearest neighbors K[1].

2- Hardware Bottleneck: Equation 3 requires solving a different linear system for each $\mathbf{c}_i$. From a hardware perspective, it means bringing in a different matrix $\mathbf{B}_i^{\mathrm{T}}\mathbf{B}_i$ and a different vector $\mathbf{B}_i^{\mathrm{T}}\mathbf{x}_i$ to the CPU cache for each $\mathbf{c}_i$. Once in cache, a matrix-vector multiplication is performed to compute $\mathbf{B}_i^{\mathrm{T}}\mathbf{x}_i$ and then finally a linear solver is invoked to compute the vector $\mathbf{c}_i$. These matrix-vector operations are generally not cache efficient, and therefore cannot fully exploit the modern hardware architecture.

3.4. Significance of Cache Efficiency

Note that the gap between processor and main-memory (DRAM) speeds has been consistently increasing [20]. This *memory wall* prevents algorithms with a low flop-byte ratio to fully exploit the compute-power of the modern CPUs.

Matrix-vector operations (called Level-2 BLAS – Basic Linear Algebra Subprograms) have a low flop-byte ratio. However, matrix-matrix operations (called Level-3 BLAS) have their flop-byte ratio increase proportional to the matrix size [9] (Table 2). Matrix-matrix multiplication therefore offers much higher compute intensity by pre-loading matrix blocks into the CPU cache, keeping the compute-units maximally busy. Leveraging BLAS-3 for LLC encoding could therefore greatly improve its efficiency.

[1]Consider *e.g.*, D = 128, K = 10, and N = 1 Billion. The number of flops required to find $\mathbf{B}_i^{\mathrm{T}}\mathbf{B}_i \ \forall\ i$ and their Cholesky factors are $\sim$ 14 Tera-flops. On an 8 core 2.4 GHz machine, this would take $\sim$ 10 hours.

Exact LLC	Proposed scheme
Forming $\mathbf{B}_i^{\mathrm{T}}\mathbf{B}_i, \forall i$ $\mathcal{O}(\mathrm{NDK}^2)$	Forming $\mathbf{B}^{m\mathrm{T}}\mathbf{B}^m\ \forall m$ $\mathcal{O}(\mathrm{MDK}^2)$
Cholesky on $\mathbf{B}_i^{\mathrm{T}}\mathbf{B}_i, \forall i$ $\mathcal{O}(\mathrm{NK}^3)$	Cholesky on $\mathbf{B}^{m\mathrm{T}}\mathbf{B}^m\ \forall m$ $\mathcal{O}(\mathrm{MK}^3)$
Forming $\mathbf{B}_i^{\mathrm{T}}\mathbf{x}_i, \forall i$ $\mathcal{O}(\mathrm{NDK})$ BLAS-2	Forming $\mathbf{B}^{m\mathrm{T}}\mathbf{X}^m\ \forall m$ $\mathcal{O}(\mathrm{NDK})$ BLAS-3
Solving $\mathbf{c}_i$ in Equation 3 $\mathcal{O}(\mathrm{NK}^2)$ BLAS-2	Solving $\mathbf{c}_i$ in Equation 5 $\mathcal{O}(\mathrm{NK}^2)$ BLAS-3

Table 3: Comparison between the original LLC and our proposed scheme. Note that N >> M $\geq$ K.

3.5. An Approximate Coding Scheme

To address the algorithmic complexity and hardware challenges in LLC encoding, we present an approximate coding scheme that is not only algorithmically more efficient, but also more compliant to modern hardware architecture.

Recall that both the aforementioned efficiency challenges with Equation 3 stem from the fact that it uses a *different* $\mathbf{B}_i$ for each $\mathbf{x}_i$. If instead we used the same set of bases for a cluster of similar $\mathbf{x}_i$, we could significantly increase our encoding efficiency. This intuition can be concretized in terms of the following objective function:

$$\arg\min_{\widetilde{\mathbf{C}}} \sum_{i=1}^{\mathrm{N}^m} ||\mathbf{x}_i^m - \mathbf{B}^m\widetilde{\mathbf{c}}_i||^2 \tag{4}$$

Here $m \in \mathrm{M}$ represents the cluster index, N^m the size of the m-th cluster, and $\widetilde{\mathbf{c}}_i$ the i-th approximate code. Equation 4 can be written in the normal equation form as:

$$\widetilde{\mathbf{c}}_i = (\mathbf{B}^{m\mathrm{T}}\mathbf{B}^m)^{-1}\mathbf{B}^{m\mathrm{T}}\mathbf{X}^m \tag{5}$$

where $\mathbf{X}^m$ is a $\mathrm{D} \times \mathrm{N}^m$ matrix of descriptors of m^{th} cluster.

Unlike Equation 3, Equation 5 requires computing $(\mathbf{B}^{m\mathrm{T}}\mathbf{B}^m)^{-1}\mathbf{B}^{m\mathrm{T}}$ only once for all descriptors in $\mathbf{X}^m$, reducing the total sub-space computation cost from $\mathcal{O}(\mathrm{NDK}^2 + \mathrm{NK}^3 + \mathrm{NDK} + \mathrm{NK}^2)$ for LLC (Table 3), to $\mathcal{O}(\mathrm{MDK}^2 + \mathrm{MK}^3 + \mathrm{NDK} + \mathrm{NK}^2)$ for our proposed scheme. This eliminates the potentially very large number N from the first two terms. Furthermore, forming the right-hand side $\mathbf{B}^{m\mathrm{T}}\mathbf{X}^m$ and solving $\mathbf{c}_i$ in Equation 5 (corresponding to the last two terms) can be treated as matrix-matrix operations rather than matrix-vector operations, and can therefore fully exploit the modern hardware architecture and achieve peak CPU flops using BLAS-3 [9] (Table 3).

Our proposed scheme is summarized in Algorithm 1. Note that we find the descriptors belonging to each cluster before going through all the clusters for encoding (line 3), in order to visit the cluster assignment array of length N for just $\mathcal{O}(1)$ times rather than M times. We found that this strategy significantly increased the empirical efficiency of

Algorithm 1 Hardware Compliant Approximate Encoding

Input: Image descriptors $\mathbf{X} \in \mathbb{R}^{\mathrm{D}\times\mathrm{N}}$, codebook $\mathbf{B} \in \mathbb{R}^{\mathrm{D}\times\mathrm{M}}$, cluster assignment for all the descriptors
Output: Approximate image codes $\widetilde{\mathbf{C}} \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$
Form $\mathbf{X}^1, \mathbf{X}^2, \cdots, \mathbf{X}^{\mathrm{M}}$ by gathering the descriptors belonging to each individual cluster
for $m = 1$ to M **do**
  Determine $\mathbf{B}^m \in \mathbb{R}^{\mathrm{D}\times\mathrm{K}}$
  Compute left-hand side $\mathbf{B}^{m\mathrm{T}}\mathbf{B}^m \equiv \mathbf{W}$
  Perform Cholesky factorization: $\mathbf{W} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$
  Compute right-hand side $\mathbf{B}^{m\mathrm{T}}\mathbf{X}^m \equiv \mathbf{Y}$
  Solve $\mathbf{L}\mathbf{Z} = \mathbf{Y}$ for $\mathbf{Z}$, and $\mathbf{L}^{\mathrm{T}}\widetilde{\mathbf{C}}^m = \mathbf{Z}$ for $\widetilde{\mathbf{C}}^m$
end for
Normalize each column of $\widetilde{\mathbf{C}}$ to unit L_2 norm

our algorithm besides its lower algorithmic complexity and better compliance to the hardware architecture. In addition, because the learned image codes are typically sparse, *i.e.* only K out of M entries in the code are nonzero, we use a sparse matrix to represent the codes in our implementation to save memory space, as opposed to a dense matrix in previous LLC implementations [28, 2]. Note that we adopted the same implementation optimizations for both our scheme and its exact counterpart while reporting any results.

4. Approximation Error Analysis

We now present a formal treatment of our approximation error bound, and use this analysis to distill the intuition behind the robustness of our method. We then present empirical evidence to validate these intuitions.

4.1. Approximation Error Bounds

Consider the solution $\mathbf{c}_i^\star$ to the problem

$$\arg\min_{\mathbf{c}} \|\mathbf{x}_i - \mathbf{B}_i\mathbf{c}\|_2 \tag{6}$$

and $\tilde{\mathbf{c}}_i^\star$, the solution to the problem

$$\arg\min_{\mathbf{c}} \|\mathbf{x}_i - \mathbf{B}^m\mathbf{c}\|_2. \tag{7}$$

Intuitively, the difference between $\mathbf{c}_i^\star$ and $\tilde{\mathbf{c}}_i^\star$ will be small when the following two conditions are satisfied:

1. Small perturbations to $\mathbf{B}_i$ result in small perturbation to $\mathbf{c}_i^\star$—this is a property of the dictionary.
2. $\mathbf{B}^m$ is a small perturbation of $\mathbf{B}_i$—this depends on the interaction of a specific $\mathbf{x}_i$ with the dictionary.

The first condition depends on the degree of linear independence amongst the columns of $\mathbf{B}_i$, *i.e.*, the more linearly independent the columns of $\mathbf{B}_i$ are, the more stable the coordinates of $\mathbf{x}_i$ will become to a given perturbation.

The second condition is quantified by the approximation error $\|\mathbf{B}^m - \mathbf{B}_i\|_2$. Moreover, we expect that the distance between $\mathbf{c}_i^\star$ and $\tilde{\mathbf{c}}_i^\star$ should increase as the norm of $\mathbf{x}_i$ does. These conditions lead us to anticipate that $\mathbf{c}_i^\star$ and $\tilde{\mathbf{c}}_i^\star$ satisfy a relationship of the form:

$$\|\tilde{\mathbf{c}}_i^\star - \mathbf{c}_i^\star\|_2 \lesssim f\big(\|\mathbf{B}_i^\dagger\|_2, \|\mathbf{B}^m - \mathbf{B}_i\|_2, \|\mathbf{x}_i\|_2\big),$$

where f is an increasing function of its arguments, and $\mathbf{B}_i^\dagger$ denotes the pseudo-inverse of $\mathbf{B}_i$.

The following theorem establishes an upper bound on the approximation error that verifies this intuition and is sharper in our application than the classic perturbation bounds [9, Theorem 5.3.1]. We note in particular that although the bound is stated in an asymptotic form for clarity, the proof can be trivially modified to yield an exact inequality.

Theorem 1. *Let* $\mathbf{E} = \mathbf{B}^m - \mathbf{B}_i$. *When*

$$\|\mathbf{E}\|_2 \leq \frac{\|\mathbf{B}_i^\dagger\|_2^{-2}}{4\|\mathbf{B}_i\|_2},$$

the coding error satisfies

$$\|\mathbf{c}_i^\star - \tilde{\mathbf{c}}_i^\star\|_2 \leq \frac{1+\sqrt{5}}{2}\|\mathbf{B}_i^\dagger\|_2^2\|\mathbf{E}\|_2\|\mathbf{x}_i\|_2 + \mathrm{O}(\|\mathbf{E}\|_2^2\|\mathbf{x}_i\|_2).$$

Proof. First we note that the solutions to the two least-squares problems are

$$\mathbf{c}_i^\star = \mathbf{B}_i^\dagger\mathbf{x}_i \text{ and } \tilde{\mathbf{c}}_i^\star = (\mathbf{B}^m)^\dagger\,\mathbf{x}_i,$$

It follows that

$$\|\mathbf{c}_i^\star - \tilde{\mathbf{c}}_i^\star\|_2 \leq \big\|\mathbf{B_i}^\dagger - (\mathbf{B}^m)^\dagger\big\|_2\|\mathbf{x}_i\|_2.$$

To continue, we use the fact [26, Theorem 3.3], that for conformal matrices $\mathbf{A}$ and $\mathbf{M}$,

$$\|\mathbf{A}^\dagger - \mathbf{M}^\dagger\|_2 \leq \frac{1+\sqrt{5}}{2}\max\{\|\mathbf{A}^\dagger\|_2^2, \|\mathbf{M}^\dagger\|_2^2\}\|\mathbf{A}-\mathbf{M}\|_2.$$

From this, it follows that

$$\|\mathbf{c}_i^\star - \tilde{\mathbf{c}}_i^\star\|_2 \leq \frac{1+\sqrt{5}}{2}\max\left\{\|\mathbf{B}_i^\dagger\|_2^2, \big\|(\mathbf{B}^m)^\dagger\big\|_2^2\right\}\|\mathbf{E}\|_2\|\mathbf{x}_i\|_2. \tag{8}$$

As $\mathbf{B}^m$ is sufficiently close to $\mathbf{B}_i$, we can replace $\big\|(\mathbf{B}^m)^\dagger\big\|_2$ in this estimate with $\|\mathbf{B}_i^\dagger\|_2$.
To do so, we use the facts that:

$$\|\mathbf{A}^\dagger\|_2^2 = \lambda_{\min}(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}$$

and

$$\lambda_{\min}(\mathbf{A}^{\mathrm{T}}\mathbf{A} + \mathbf{S}) \geq \lambda_{\min}(\mathbf{A}^{\mathrm{T}}\mathbf{A}) - \|\mathbf{S}\|_2$$

when $\mathbf{S}$ is a symmetric matrix [12, Theorem 4.3.1]. Indeed, applying these facts, we find that

$$\begin{aligned}\left\|\left(\mathbf{B}^m\right)^\dagger\right\|_2^2 &= \lambda_{\min}((\mathbf{B}_i+\mathbf{E})^{\mathrm{T}}(\mathbf{B}_i+\mathbf{E}))^{-1} \\ &\leq \left(\lambda_{\min}(\mathbf{B}_i^{\mathrm{T}}\mathbf{B}_i) - \|\mathbf{E}^{\mathrm{T}}\mathbf{B}_i + \mathbf{B}_i^{\mathrm{T}}\mathbf{E} + \mathbf{E}^{\mathrm{T}}\mathbf{E}\|_2\right)^{-1} \\ &\leq \left(\|\mathbf{B}_i^\dagger\|_2^{-2} - 2\|\mathbf{E}\|_2\|\mathbf{B}_i\|_2 + \|\mathbf{E}\|_2^2\right)^{-1} \\ &= \|\mathbf{B}_i^\dagger\|_2^2 + \mathrm{O}(\|\mathbf{E}\|_2).\end{aligned}$$

The final equality follows from Taylor series expansion of $1/u$ about $u = \|\mathbf{B}_i^\dagger\|_2^{-2}$ and the readily verified fact that the denominator is strictly positive when, as we assumed:

$$\|\mathbf{E}\|_2 \leq \frac{\|\mathbf{B}_i^\dagger\|_2^{-2}}{4\|\mathbf{B}_i\|_2}.$$

Using the fact that $\left\|\left(\mathbf{B}^m\right)^\dagger\right\|_2^2 \leq \|\mathbf{B}_i^\dagger\|_2^2 + \mathrm{O}(\|\mathbf{E}\|_2)$ in the estimate (8), we conclude that

$$\|\mathbf{c}_i^\star - \tilde{\mathbf{c}}_i^\star\|_2 \leq \frac{1+\sqrt{5}}{2}\|\mathbf{B}_i^\dagger\|_2^2\|\mathbf{E}\|_2\|\mathbf{x}_i\|_2 + \mathrm{O}(\|\mathbf{E}\|_2^2\|\mathbf{x}_i\|_2).$$

□

The two main quantities involved in Theorem 1, $\|\mathbf{B}_i^\dagger\|_2$ and $\|\mathbf{E}\|_2$, depend respectively on the geometry of the codewords and the interaction of the given point $\mathbf{x}_i$ with the codewords. In the best case, the points in $\mathbf{B}$ are in general position, *i.e.* no subsets of $\mathrm{K} < \mathrm{D}$ codewords from the dictionary are linearly dependent; in this situation the columns of $\mathbf{B}_i$ can be modeled by independent D-dimensional Gaussian vectors rescaled by $1/\sqrt{\mathrm{D}}$ to lie around the unit sphere. It then follows from standard results [27, Corollary 5.35], that $\|\mathbf{B}_i^\dagger\|_2 \leq \left(1 - \sqrt{\frac{2\mathrm{K}}{\mathrm{D}}}\right)^{-1}$ with high probability.

To understand the behavior of the remaining term $\|\mathbf{E}\|_2$, note that the norms of all columns of $\mathbf{E}$ are bounded by the maximum distance ($d_{\max}$) between any two codewords in the dictionary. Moreover, if there is a sufficiently large increase in the distance from $\mathbf{x}_i$ to the corresponding codeword as we move from its j-th neighbor to its $(j+1)$-th neighbor: then at least j columns of $\mathbf{E}$ are identically zero.

Lemma 1. *Given $\mathbf{x}_i$ order the codewords in $\mathbf{B}$ so that $c_j(\mathbf{x}_i)$ denotes the jth nearest codeword to $\mathbf{x}_i$. If*

$$\|\mathbf{x}_i - c_j(\mathbf{x}_i)\|_2 \leq \|\mathbf{x}_i - c_{j+1}(\mathbf{x}_i)\|_2 - 2\|\mathbf{x}_i - c_1(\mathbf{x}_i)\|_2 \quad (9)$$

then at least j columns of $\mathbf{E}$ are zero.

Proof. Recall that $\mathbf{E} = \mathbf{B}^m - \mathbf{B}_i$, where $\mathbf{B}^m$ contains the k nearest codewords to $c_1(\mathbf{x}_i)$. It therefore suffices to establish that (9) implies that the jth nearest codewords to $\mathbf{x}_i$ are also the jth nearest codewords to $c_1(\mathbf{x}_i)$.

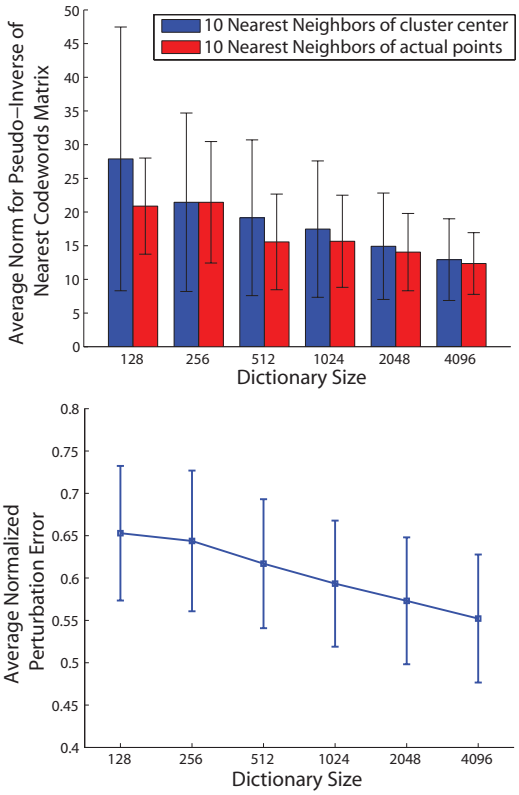


Figure 2: (Top) Average norm of the pseudo-inverse for the nearest codeword-neighbor matrices of individual points (blue) and for cluster centroids (red). (Bottom) Average L_2 distance between point's nearest codewords (α) and the codewords nearest to the point's cluster centroid (β). Here these distances were normalized by $||\beta||_2$.

To establish this, observe that for $\ell = 1, \ldots, j$,

$$\begin{aligned}\|c_1(\mathbf{x}_i) - c_\ell(\mathbf{x}_i)\|_2 &\leq \|\mathbf{x}_i - c_\ell(\mathbf{x}_i)\|_2 + \|\mathbf{x}_i - c_1(\mathbf{x}_i)\|_2 \\ &\leq \|\mathbf{x}_i - c_j(\mathbf{x}_i)\|_2 + \|\mathbf{x}_i - c_1(\mathbf{x}_i)\|_2 \\ &\leq \|\mathbf{x}_i - c_{j+1}(\mathbf{x}_i)\|_2 - \|\mathbf{x}_i - c_1(\mathbf{x}_i)\|_2,\end{aligned}$$

where the last inequality holds because of (9).

At the same time, for any codeword $\mathbf{c}$ in $\mathbf{B}$ that is not in the set of j nearest codewords to $\mathbf{x}_i$,

$$\begin{aligned}\|c_1(\mathbf{x}_i) - \mathbf{c}\|_2 &\geq \|\mathbf{x}_i - \mathbf{c}\|_2 - \|\mathbf{x}_i - c_1(\mathbf{x}_i)\|_2 \\ &\geq \|\mathbf{x}_i - c_{j+1}(\mathbf{x}_i)\|_2 - \|\mathbf{x}_i - c_1(\mathbf{x}_i)\|_2.\end{aligned}$$

It follows that the j nearest codewords to $c_1(\mathbf{x}_i)$ are exactly the j nearest codewords to $\mathbf{x}_i$. □

This lemma implies that if the gap between $\|\mathbf{x}_i - c_j(\mathbf{x}_i)\|_2$ and $\|\mathbf{x}_i - c_{j+1}(\mathbf{x}_i)\|_2$ is sufficiently small, then

$$\|\mathbf{E}\|_2 \leq \|\mathbf{E}\|_{\mathrm{F}} \leq d_{\max}\sqrt{k-j}.$$

4.2. Empirical Trends in Approximation Error

We now show how the factors identified by our theoretical analysis (§ 4.1) effecting our approximation error vary in practice on real data as a function of dictionary-size. These results are presented on Caltech-101 [6], and similar trends hold for other data-sets as well.

4.2.1 Codeword Matrix Stability

In Figure 2-a, we plot the average pseudo-inverse of $[\mathbf{B}_i]_{i=1}^{\mathrm{N}}$ (blue) and $[\mathbf{B}^m]_{m=1}^{\mathrm{M}}$ (red) as a function of the dictionary size. There are two important things to note here.

First, both the pseudo-inverses decrease monotonically with the dictionary size, implying a monotonic increase in the stability of the nearest codeword matrix.

Second, the average pseudo inverse of $[\mathbf{B}^m]_{m=1}^{\mathrm{M}}$ is strictly greater than that of $[\mathbf{B}_i]_{i=1}^{\mathrm{N}}$. Recall that our theoretical analysis predicted this trend. This plot empirically validates that our approximation does not decrease the stability of the nearest codewords matrices.

4.2.2 Perturbation Behavior

In Figure 2-b, we plot the average normalized Euclidean distance between the nearest codewords to a point, and the codewords nearest to its cluster centroid. Recall that this quantity encodes the amount of perturbation in the nearest codewords matrices brought about by our approximation. The plot monotonically decreases as a function of the dictionary size, indicating that our scheme becomes more robust to perturbations with increase in the dictionary size.

5. Experiments and Results

5.1. Representational Accuracy

5.1.1 Normalized Relative Reconstruction Error

First, to get an absolute sense of how much reconstruction error spatially-constrained exact image codes incur, we present in Figure 3-a the histogram of reconstruction errors using LLC [28] codes for Caltech-101 [6] . Here the values of M and K were set equal to 1024 and 10 respectively.

Figure 3-b, shows the histogram of normalized reconstruction error ($\mathbf{e}$) of our proposed approximate codes relative to the error ($\mathbf{e}_0$) incurred by their exact LLC counterpart. More precisely, it plots the histogram of $\mathrm{abs}(||\mathbf{e}||_2 - ||\mathbf{e}_0||_2)/||\mathbf{e}_0||_2$. As before, here $\mathrm{M} = 1024$ and $\mathrm{K} = 10$.

Note that the majority of the probability-mass in Figure 3-b is close to zero, which empirically demonstrates that our approximation scheme increases the reconstruction error of its exact counterpart only by a marginal amount.

Test Accuracy	LLC	Proposed
Caltech-101	$\mathbf{72.16} \pm 0.7$	$\mathbf{71.35} \pm 0.8$
Caltech-256	$\mathbf{37.04} \pm 0.3$	$\mathbf{35.69} \pm 0.2$
Pascal VOC'07	**51.95**	**52.90**
MIT Scenes	**38.30**	**39.91**

Table 4: Percent classification results for different data-sets using the exact LLC [28] and the proposed approximate encoding. The classification accuracy and standard deviation over 10 runs are shown for Caltech-101 and Caltech-256. The mean average precision (MAP) is shown for Pascal VOC 2007 and MIT Scenes data-sets without standard deviation since we used the fixed standard train-test splits.

5.1.2 Reconstruction Error versus Codebook Size

Figure 3-c shows the average reconstruction error incurred by both our proposed approximate encoding scheme and its exact LLC [28] counterpart. As before, for both of these cases $\mathrm{M} = 1024$ and $\mathrm{K} = 10$.

It can be seen from Figure 3-c that our approximate encoding scheme traces the reconstruction error of its exact counterpart quite closely over a wide range of dictionary sizes. Note that the maximum average reconstruction error incurred by our approximate scheme for any value of the considered dictionary size is only $\sim 5\%$ more than that incurred by its exact counterpart.

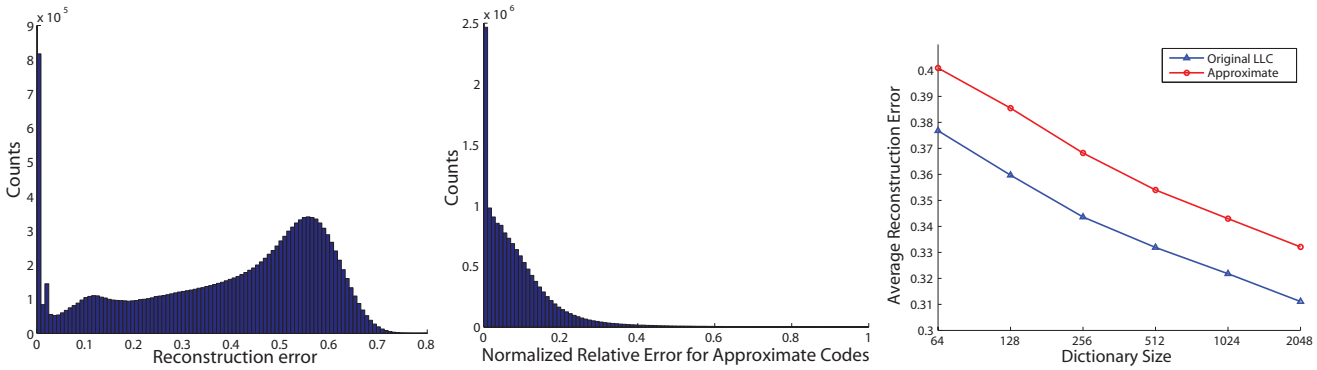
5.2. Discriminative Accuracy

5.2.1 Average Classification Results

The average classification accuracy over multiple data-sets for our approximate approach and its exact LLC counterpart is given in Table 4. Here we consider 4 data-sets, namely Caltech-101 [6], Caltech-256 [11], Pascal VOC 2007 [4], and MIT Scenes [23]. For Caltech-101 and Caltech-256 we provide test classification accuracy with 30 images per-class for training and the rest for testing. For Pascal VOC 2007 and MIT Scenes we provide mean average precision (MAP). The train-test split used for Pascal was the same as the standard defined by the challenge. For MIT Scenes data-set, we used the same train-test split as in [23], with 80 images for training and 20 for testing per-class. For all experiments, we only used SIFT [19] with linear-SVM [5].

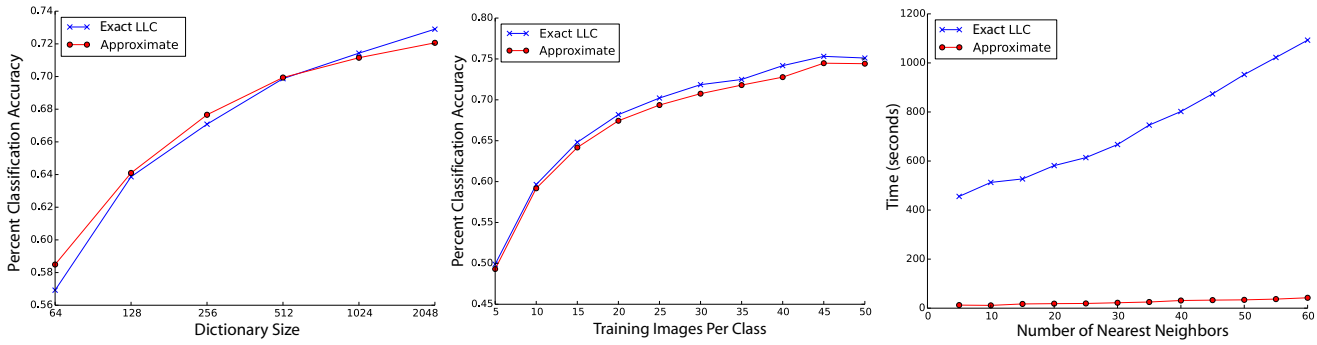
Our codebook sizes (M) were equal to $1024, 2048, 4000$ and 4096 for each of the four data-set respectively. Note that we did not incorporate any coodbook optimizations. Moreover, other than Pascal VOC 2007, we did not incorporate cross-validation to fine-tune SVM parameters. Furthermore, we set $\mathrm{K} = 10$ in all these experiments.

As can be observed from Table 4, the classification accuracy of our approximate scheme is quite close to that of the



(a) (b) (c)

Figure 3: (a) Histogram of reconstruction errors for the exact LLC [28] codes on Caltech-101 [6]. (b) Average normalized reconstruction error for the proposed approximate codes $\mathbf{e}$, relative to the exact reconstruction error $\mathbf{e}_0$, *i.e.* $\text{abs}(||\mathbf{e}||_2 - ||\mathbf{e}_0||_2)/||\mathbf{e}_0||_2$. (c) Average reconstruction error on Caltech-101 [6] for exact LLC [28] and the proposed scheme.



(a) (b) (c)

Figure 4: Average classification accuracy for exact LLC [28] and our scheme on Caltech-101 [6] for (a) different dictionary sizes, and (b) different training images per-class. (c) Run-time on Caltech-101 [6] for exact LLC [28] and the proposed approximate encoding scheme with different numbers of nearest neighbors.

exact LLC codes, with the average loss in accuracy of 1% for Caltech-101 and Caltech-256. For Pascal VOC 2007 and MIT Scene data-sets, our accuracy is in fact slightly higher than that of exact LLC.

5.2.2 Effect of Codebook and Training Size

Figure 4-a plots the average accuracy our proposed approximate codes and their exact LLC counterpart for different codebook sizes. Note that the rate of increase for both techniques is quite similar over a wide range of dictionary sizes. The plot in Figure 4-a was computed for Caltech-101 data-set, and similar trends for classification accuracy increase hold for other data-sets as well.

Figure 4-b presents a similar analysis for the effect of training data size on our proposed approximate scheme in comparison to its exact LLC counterpart. Again, both these plots are quite similar to each other, demonstrating that our scheme is robust across a wide training size spectrum.

Encoding Time	LLC	Proposed	Speed-Up
Caltech-101	512.8 sec	12.9 sec	**39.8**×
Caltech-256	1779 sec	47.3 sec	**37.6**×
Pascal VOC'07	208.0 min	4.86 min	**42.8**×
MIT Scenes	473.3 sec	12.8 sec	**37.0**×

Table 5: Encoding times on four different data-sets using the exact LLC [28] and the proposed approximate encoding.

5.3. Run-Time Efficiency

The run-time results for our approximate approach and its exact LLC counterpart is given in Table 5. We used the same data-sets and settings as in Section 5.2. For each ex-

periment, we used an 8-core 2.66 GHz machine with 32-bit floats for processing. Our proposed approach achieved $\sim 40\times$ speed-up over the exact LLC on all our data-sets.

We plot the run-time of our proposed encoding scheme on Caltech-101 as a function of K in Figure 4-c for more detailed analysis. When $K \leq 50$, the increasing trend of the run-time is sublinear with K, suggesting that using more nearest neighbors for encoding only adds a fractional amount of computational burden. For example, the run-time for $K = 30$ is less than twice of that for $K = 5$, and the run-time for $K = 50$ is less than three times of that for $K = 5$. This efficiency advantage directly results from the cache blocking scheme of BLAS-3 operations [9]. In contrast, the run-time of exact LLC increases linearly with K.

This is particularly important because the accuracy of our approximate scheme can in fact increase for larger values of K. This is specially true when using larger codebook sizes. In Caltech-256 for example, our test accuracy went from 35.69 ± 0.2 to 36.1 ± 0.2 as we increased K from 10 to 30. We can therefore exploit these accuracy gains with minimal extra computational cost. For instance, using $K = 30$ in our scheme for Caltech-256 is still $30\times$ faster than exact LLC using $K = 10$.

5.4. Comparison with State-of-the-Art Methods

We further compare our proposed approach with state-of-the-art image coding methods. Because not all the image coding methods are based on nearest neighbor search, we include both the "Assign" and "Encode" stages (ref. Table 1) to account for the run-time of our approach. For the "Assign" stage that generates cluster assignment for all the descriptors, we employed the *kd*-tree based approximate nearest neighbor search, rather than using exact search.

The comparisons were performed on the Caltech-101 [6] data-set with Product Sparse Coding (PSC) [7], Approximate Locality-constrained Soft Assignment (LcSA) [18], Soft Assignment (SA) [8], and Sparse Coding (SC) [17]. LcSA employs the *Fast Hierarchical Nearest Neighbor Search* (FHNNS) scheme proposed in [15] for approximate assignment for the descriptors. The same experiment settings as in [2] were used with the numbers of training images per category set to 15 or 30.

Table 6 shows the classification accuracy and run-time for coding per image on Caltech-101 [6] using a codebook of size 4096[2]. Compared to PSC, our proposed approach is nearly twice as fast while producing comparable accuracy. Our encoding approach with approximate nearest neighor based on *kd*-tree produces higher classification accuracy than LcSA with FHNNS and SA, and is 13 times faster than SA and more than 20 times faster than SC.

[2]**Errata**: Note that the times for LcSA, SC, and SA given in Table 6 in the camera-ready version were mistakenly reported incorrectly. This online version contains the corrected results for these methods.

	Run-time	Accuracy
PSC$_{n=30}$	0.45 sec	76.71%
Proposed$_{n=30}$	0.258 sec	76.43%
Proposed$_{n=15}$	0.258 sec	72.54%
LcSA$_{n=15}$	0.115 sec	71.90%
SA$_{n=15}$	3.51 sec	71.60%
SC$_{n=15}$	5.61 sec	74.60%

Table 6: Classification accuracy and run-time per image for various state-of-the-art image coding methods on Caltech-101 [6]. The parameters n in the subscript indicates the number of training images per category.

6. Conclusions and Future Work

Descriptor encoding is one of the main efficiency bottlenecks in image classification. Particularly during the testing phase, the complexity of descriptor encoding can be a significant proportion of classification pipeline. In this work, we proposed an approximate locality-constrained encoding scheme that offers significantly better computational efficiency compared to its exact counterpart, with similar classification accuracy. Using the perturbation analysis of least-squares problems, we presented a formal error analysis of our approach to identify the factors affecting the approximation error of our scheme. We empirically verified that in practice these factors indeed behave as predicted by our theoretical analysis, and introduce only a nominal error compared to their exact conuterparts. Finally, we presented a thorough set of comparative experimental results on multiple data-sets to assess both representational and discriminative capabilities of our scheme.

So far in our comparison, we considered two important factors to judge the effectiveness of various encoding schemes, *i.e.*, (i) encoding speed, and (ii) classification accuracy. An important comparative factor we did not consider in this work is (iii) the memory footprint of an encoding scheme. In our future work, we plan to consider all these three dimensions to do a more comprehensive comparative analysis of how our approximate encoding scheme fits in the overall encoding landscape. In particular, including memory footprint as one of the comparative factors would allow us to compare our approach against schemes like Fisher Vectors [22], which although are quite efficient and accurate, but have a substantially large memory footprint.

We also plan to explore if applying our intuition of representing similar feature-groups with shared set of bases would make sense to approximate other encoding techniques as well. Similarly, we plan on exploring the effectiveness of our approximation for more general-purpose locally linear subspace learning.

References

[1] R. Arandjelovic and A. Zisserman. All about VLAD. In *Proc. of the 2013 IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR '13, pages 1578–1585, 2013. 2

[2] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. pages 76.1–76.12, 2011. 1, 2, 4, 8

[3] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2, 2004. 1, 2

[4] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal visual object classes (VOC) challenge. *Int. J. Comput. Vision*, 88(2):303–338, 2010. 6

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2008. 1, 6

[6] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Comput. Vis. Image Underst.*, 106(1):59–70, 2007. 6, 7, 8

[7] T. Ge, K. He, and J. Sun. Product sparse coding. In *Proc. of the 2014 IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR '14, pages 939–946, 2014. 8

[8] J. C. Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders. Kernel codebooks for scene categorization. In *Proc. of the 10th European Conf. on Computer Vision: Part III*, ECCV '08, pages 696–709. 2008. 1, 2, 8

[9] G. H. Golub and C. F. Van Loan. *Matrix Computations, 4th edition*. The Johns Hopkins University Press, 2013. 3, 4, 8

[10] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proc. of the 10th IEEE Int. Conf. on Computer Vision - Volume 2*, ICCV '05, pages 1458–1465, 2005. 2

[11] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007. 6

[12] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985. 5

[13] H. Jégou and O. Chum. Negative evidences and cooccurences in image retrieval: The benefit of PCA and whitening. In *Proc. of the 12th European Conf. on Computer Vision - Volume Part II*, ECCV '12, pages 774–787, 2012. 2

[14] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proc. of the 2010 IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR '10, pages 3304–3311, 2010. 2

[15] P. Koniusz, F. Yan, and K. Mikolajczyk. Comparison of mid-level feature coding approaches and pooling strategies in visual concept detection. *Comput. Vis. Image Underst.*, 117(5):479–492, 2013. 8

[16] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. of the 2006 IEEE Conf. on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2169–2178, 2006. 1, 2

[17] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems 19*, NIPS '07, pages 801–808, 2007. 8

[18] L. Liu, L. Wang, and X. Liu. In defense of soft-assignment coding. In *Proc. of the 2011 Int. Conf. on Computer Vision*, ICCV '11, pages 2486–2493, 2011. 8

[19] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the Int. Conf. on Computer Vision - Volume 2*, ICCV '99, pages 1150–1157, 1999. 1, 6

[20] S. A. McKee. Reflections on the memory wall. In *Proc. of the 1st Conf. on Computing Frontiers*, pages 162–167, 2004. 3

[21] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed Fisher vectors. In *Proc. of the 2010 IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR '10, pages 3384–3391, 2010. 2

[22] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *Proc. of the 11th European Conf. on Computer Vision: Part IV*, ECCV '10, pages 143–156, 2010. 1, 2, 8

[23] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Proc. of the 2009 IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR '09, pages 413–420, 2009. 6

[24] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *Int. J. Computer Vision*, 105(3):222–245, 2013. 2

[25] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proc. of the 9th IEEE Int. Conf. on Computer Vision - Volume 2*, ICCV '03, pages 1470–1477, 2003. 2

[26] G. W. Stewart. On the Perturbation of Pseudo-Inverses, Projections and Linear Least Squares Problems. *SIAM Review*, 19:634–662, 1977. 4

[27] R. Vershynin. *Compressed Sensing: Theory and Applications*, chapter Introduction to the non-asymptotic analysis of random matrices, pages 210–268. Cambridge University Press, 2012. 5

[28] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proc. of the 2010 IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR '10, pages 3360–3367, 2010. 1, 2, 4, 6, 7

[29] J. Yang, K. Yu, and T. Huang. Supervised translation-invariant sparse coding. In *Proc. of the 2010 IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR '10, pages 3517–3524, 2010. 1

[30] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *Advances in Neural Information Processing Systems 22*, NIPS '09, pages 2223–2231, 2009. 1, 2

[31] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *Proc. of the 11th European Conf. on Computer Vision: Part V*, ECCV '10, pages 141–154, 2010. 1, 2