Inductive Hashing on Manifolds

Fumin Shen[‡†*], Chunhua Shen[†], Qinfeng Shi[†], Anton van den Hengel[†], Zhenmin Tang[‡]
[†] The University of Adelaide, Australia [‡] Nanjing University of Science and Technology, China

Abstract

Learning based hashing methods have attracted considerable attention due to their ability to greatly increase the scale at which existing algorithms may operate. Most of these methods are designed to generate binary codes that preserve the Euclidean distance in the original space. Manifold learning techniques, in contrast, are better able to model the intrinsic structure embedded in the original high-dimensional data. The complexity of these models, and the problems with out-of-sample data, have previously rendered them unsuitable for application to large-scale embedding, however.

In this work, we consider how to learn compact binary embeddings on their intrinsic manifolds. In order to address the above-mentioned difficulties, we describe an efficient, inductive solution to the out-of-sample data problem, and a process by which non-parametric manifold learning may be used as the basis of a hashing method. Our proposed approach thus allows the development of a range of new hashing techniques exploiting the flexibility of the wide variety of manifold learning approaches available. We particularly show that hashing on the basis of t-SNE [29] outperforms state-of-the-art hashing methods on large-scale benchmark datasets, and is very effective for image classification with very short code lengths.

1. Introduction

One of many challenges emerging from the current explosion in the volume of image-based media available is how to index and organize the data accurately, but also efficiently. Various hashing techniques have attracted considerable attention in computer vision, information retrieval and machine learning [8, 9, 19, 31, 33], and seem to offer great promise towards this goal. Hashing methods aim to encode documents or images as a set of short binary codes, while maintaining aspects of the structure of the original data. The advantage of these compact binary representations is that pairwise comparisons may be carried out extremely efficiently. This means that many algorithms which are based on such pairwise comparisons can be made more efficient, and applied to much larger datasets.

Locality sensitive hashing (LSH) [8] is one of the most well-known *data-independent* hashing methods, and generates hash codes based on random projections. With the success of LSH, random hash functions have been extended to several similarity measures, including p-norm distances [6], the Mahalanobis metric [17], and kernel similarity [16, 24]. However, the methods belonging to the LSH family normally require relatively long hash codes and several hash tables to achieve both high precision and recall. This leads to a larger storage cost than would otherwise be necessary, and thus limits the sale at which the algorithm may be applied.

Data-dependent or learning-based hashing methods have been developed with the goal of learning more compact hash codes. Directly learning binary embeddings typically results in an optimization problem which is very difficult to solve, however. Relaxation is often used to simplify the optimization (*e.g.*, [3, 31]. As in LSH, the methods aim to identify a set of hyperplanes, but now these hyperplanes are learned, rather than randomly selected. For example, PCAH [31], SSH [31], and ITQ [9] generate linear hash functions through simple PCA projections, while LDAhash [3] is based on LDA. Extending this idea, there are also methods which learn hash functions in a kernel space, such as reconstructive embeddings (BRE) [15], random maximum margin hashing (RMMH) [14] and kernel-based supervised hashing (KSH) [20]. In a departure from such methods, however, spectral hashing (SH) [33], one of the most popular learning-based methods, generates hash codes by solving the relaxed mathematical program that is similar to the one in Laplacian eigenmaps [1].

Embedding the original data into a low dimensional space while simultaneously preserving the inherent neighborhood structure is critical for learning compact, effective hash codes. In general, nonlinear manifold learning methods are more powerful than linear dimensionality reduction techniques, as they are able to more effectively preserve the *local* structure of the input data without assuming *global* linearity [26]. The geodesic distance on a manifold has been shown to outperform the Euclidean distance in the high-dimensional space for image retrieval [10], for

*F. Shen's contribution was made when he was visiting The University of Adelaide.

(a) Queries **(b)** ℓ_2 dist. on 784D **(c)** LSH with 128-bits **(d)** ℓ_2 dist. on embeded 48D **(e)** Hamming dist. with 48-bits

Figure 1: Top 10 retrieved digits for 4 queries (a) on a subset of MNIST with 300 samples. Search is conducted in the original feature space (b, c) and embedding space by t-SNE [29] (d, e) using Euclidean distance (b, d) and hamming distance (c, e).

example. Figure 1 demonstrates that searching using either the Euclidean or Hamming distance after nonlinear embedding results in more semantically accurate neighbors than the same search in the original feature space, and thus that low-dimensional embedding may actually improve retrieval or classification performance. However, the only widely used nonlinear embedding method for hashing is Laplacian eigenmaps (LE) (*e.g.*, in [21, 33, 35]). Other effective manifold learning approaches (*e.g.*, LLE [25], elastic embedding [4] or t-SNE [29]) have rarely been explored for hashing.

One problem hindering the use of manifold learning for hashing is that these methods do not directly scale to large datasets. For example, to construct the neighborhood graph (or pairwise similarity matrix) in these algorithms for n data points is $O(n^2)$ in time, which is intractable for large datasets. The second problem is that they are typically non-parametric and thus cannot efficiently solve the critical out-of-sample extension problem. This fundamentally limits their application to hashing, as generating codes for new samples is an essential part of the problem. One of the widely used solutions for the methods involving spectral decomposition (*e.g.*, LLE, LE and ISOMap [27]) is the Nyström extension [2], which solves the problem by learning eigenfunctions of a kernel matrix. As mentioned in [33], however, this is impractical for large-scale hashing since the Nyström extension is as expensive as doing exhaustive nearest neighbor search ($O(n)$). A more significant problem, however, is the fact that the Nyström extension cannot be directly applied to *non-spectral* manifold learning methods such as t-SNE.

In order to address the out-of-sample extension problem, we propose a new non-parametric regression approach which is both efficient and effective. This method allows rapid assignment of new codes to previously unseen data in a manner which preserves the underlying structure of the manifold. Having solved the out-of-sample extension problem, we develop a method by which a learned manifold may be used as the basis for a binary encoding. This method is designed so as to generate encodings which reflect the geodesic distances along such manifolds. On this basis we develop a range of new embedding approaches based on a variety of manifold learning methods. The best performing of these is based on manifolds identified through t-SNE, which has been shown to be effective in discovering semantic manifolds amongst the set of all images [29].

Given the computational complexity of many manifold learning methods, we show that it is possible to learn the manifold on the basis of a small subset of the data $\mathbf{B}$ (with size m), and subsequently to inductively insert the remainder of the data, and any out-of-sample data, into the embedding in $O(m)$ time per point. This process leads to an embedding method we label Inductive Manifold-Hashing (IMH) which we show to outperform state-of-the-art methods on several large scale datasets both quantitatively and qualitatively.

Related work *Spectral Hashing* Weiss et al. [33] formulated the spectral hashing (SH) problem as

$$\min_{\mathbf{Y}} \sum_{\mathbf{x}_i,\mathbf{x}_j \in \mathbf{X}} \mathrm{w}(\mathbf{x}_i,\mathbf{x}_j)\|\mathbf{y}_i - \mathbf{y}_j\|^2 \quad (1)$$
$$\text{s.t. } \mathbf{Y} \in \{-1,1\}^{n\times r},\ \mathbf{Y}^\top\mathbf{Y} = n\mathbf{I},\ \mathbf{Y}^\top\mathbf{1} = 0.$$

Here $\mathbf{y}_i \in \{-1,1\}^r$, the ith row in $\mathbf{Y}$, is the hash code we want to learn for $\mathbf{x}_i \in \mathbf{R}^d$, which is one of the n data points in the training data set $\mathbf{X}$. $\mathbf{W} \in \mathbf{R}^{n\times n}$ with $\mathbf{W}_{ij} = \mathrm{w}(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2)$ is the graph affinity matrix, where σ is the bandwidth parameter. $\mathbf{I}$ is the identity matrix. The last two constraints force the learned hash bits to be uncorrelated and balanced, respectively. By removing the first constraint (*i.e.*, *spectral relaxation* [33]), $\mathbf{Y}$ can be easily obtained by spectral decomposition on the Laplcaian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D} = \mathrm{diag}(\mathbf{W1})$ and $\mathbf{1}$ is the vector with all ones. However, constructing $\mathbf{W}$ is $O(dn^2)$ (in time) and calculating the Nyström extension for a new point is $O(rn)$, which are both intractable for large datasets. It is assumed in SH [33], therefore, that the data are sampled from a uniform distribution, which leads to a simple analytical eigenfunction solution of 1-D Laplacians. However, this strong assumption is not true in practice and the manifold structure of the original data are thus destroyed [21].

Anchor Graph Hashing To efficiently solve problem (1), anchor graph hashing (AGH) [21] approximated the affinity matrix $\mathbf{W}$ by the low-rank matrix $\hat{\mathbf{W}} = \mathbf{Z}\Lambda^{-1}\mathbf{Z}$, where $\mathbf{Z} \in \mathbf{R}^{n\times m}$ is the normalized affinity matrix (with k nonzeros in each row) between the training samples and m *anchors* (generated by K-means), and Λ^{-1} normalizes $\hat{\mathbf{W}}$ to be doubly stochastic. Then the desired hash functions may be efficiently identified by binarizing the Nyström eigenfunctions [2] with the approximated affinity matrix $\hat{\mathbf{W}}$. AGH is thus efficient, in that it has linear training time and constant search time, but as is the case for SH [33], the

generalized eigenfunction is derived only for the Laplacian eigenmaps embedding.

Self-Taught Hashing Self-taught hashing (STH) [35] addressed the out-of-sample problem by a novel way: hash functions are obtained by training an SVM classifier for each bit using the pre-learned binary codes as class labels. The binary codes were learned by directly solving (1) with a cosine similarity function. This process has prohibitive computational and memory costs, however, and training the SVM can be very time consuming for dense data.

2. The proposed method

2.1. Inductive learning for hashing

Assuming that we have the manifold-based embedding $\mathbf{Y} := \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n\}$ for the entire training data $\mathbf{X} := \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$. Given a new data point $\mathbf{x}_q$, we aim to generate an embedding $\mathbf{y}_q$ which preserves the local neighborhood relationships among its neighbors $\mathcal{N}_k(\mathbf{x}_q)$ in $\mathbf{X}$. We choose to minimize the following simple objective:

$$\mathcal{C}(\mathbf{y}_q) = \sum_{i=1}^{n} \mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)\|\mathbf{y}_q - \mathbf{y}_i\|^2. \tag{2}$$

Here we define

$$\mathrm{w}(\mathbf{x}_q, \mathbf{x}_i) = \begin{cases} \exp(-\|\mathbf{x}_q - \mathbf{x}_i\|^2/\sigma^2), & \text{if } \mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_q), \\ 0 & \text{otherwise.} \end{cases}$$

Minimizing (2) naturally uncovers an embedding for the new point on the basis of its nearest neighbors on the low-dimensional manifold initially learned on the base set. That is, in the low-dimensional space, the new embedded location for the point should be close to those of the points close to it in the original space.

Differentiating $\mathcal{C}(\mathbf{y}_q)$ with respect to $\mathbf{y}_q$, we have

$$\left.\frac{\partial \mathcal{C}(\mathbf{y}_q)}{\mathbf{y}_q}\right|_{\mathbf{y}_q=\mathbf{y}_q^\star} = 2\sum_{i=1}^{n} \mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)(\mathbf{y}_q^\star - \mathbf{y}_i) = 0, \tag{3}$$

which leads to the optimal solution

$$\mathbf{y}_q^\star = \frac{\sum_{i=1}^{n} \mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)\mathbf{y}_i}{\sum_{i=1}^{n} \mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)}. \tag{4}$$

Equation (4) provides a simple inductive formulation for the embedding: produce the embedding for a new data point by a (sparse) linear combination of the base embeddings.

The proposed approach here is inspired by Delalleau et al. [7], where they have focused on non-parametric graph-based learning in semi-supervised classification. Our aim here is completely different: We try to scale up the manifold learning process for hashing in an unsupervised manner.

The resulting solution (4) is consistent with the basic smoothness assumption in manifold learning, that close-by data points lie on or close to a locally linear manifold [1, 25, 27]. This local-linearity assumption has also been widely used in semi-supervised learning [7, 34], image coding [32], and similar. In this paper, we propose to apply this assumption to hash function learning.

However, as aforementioned, (4) does not scale well for both computing $\mathbf{Y}$ ($O(n^2)$ *e.g.*, for LE) and out-of-sample extension ($O(n)$), which is intractable for large scale tasks. Next, we show that the following prototype algorithm is able to approximate $\mathbf{y}_q$ using only a small base set well. This prototype algorithm is based on entropy numbers defined below.

Definition 1 (Entropy numbers [12])**.** *Given any $\mathbf{Y} \subseteq \mathbb{R}^r$ and $p \in \mathbb{N}$, the m-th entropy number $\epsilon_m(\mathbf{Y})$ of Y is defined as*

$$\epsilon_m(Y) := \inf\{\epsilon > 0 | \, \mathcal{N}(\epsilon, Y, \|\cdot - \cdot\|) \leq m\},$$

where $\mathcal{N}$ is the covering number. Then $\epsilon_m(Y)$ is the smallest radius that Y can be covered by less or equal to m balls.

2.1.1 The prototype algorithm

Inspired by Theorem 27 of [12], we construct a prototype algorithm below. We use m clusters to cover $\mathbf{Y}$. Let $\alpha_i = \frac{\mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)}{\sum_{j=1}^{n} \mathrm{w}(\mathbf{x}_q, \mathbf{x}_j)}$ and $C_j = \sum_{i \in I_j} \alpha_i$. For each cluster index set I_j, we randomly draw $\ell_j = \lfloor mC_j + 1 \rfloor$ many indices from I_j proportional to their weight α_i. That is, for $\mu \in \{1, \cdots, \ell_j\}$, the μ-th randomly drawn index $u_{j,\mu}$ $\Pr(u_{j,\mu} = i) = \frac{\alpha_i}{C_j}, \forall j \in \{1, \cdots, m\}$. We then construct $\hat{\mathbf{y}}_q$ as

$$\hat{\mathbf{y}}_q = \sum_{j=1}^{m} \frac{C_j}{\ell_j} \sum_{\mu=1}^{\ell_j} \mathbf{y}_{u_{j,\mu}}. \tag{5}$$

Theorem 2. *For any even number $n' \leq n$. If Prototype Algorithm uses n' many non-zero $\mathbf{y} \in \mathbf{Y}$ to express $\hat{\mathbf{y}}_q$, then*

$$\Pr[\|\hat{\mathbf{y}}_q - \mathbf{y}_q\| \geq t] < \frac{2(\epsilon_{\frac{n'}{2}}(\mathbf{Y}))^2}{n't^2}. \tag{6}$$

Corollary 3. *For an even number n', any $\epsilon > \epsilon_{\frac{n'}{2}}(\mathbf{Y})$, any $\delta \in (0, 1)$ and any $t > 0$, if $n' \geq \frac{2\epsilon^2}{\delta t^2}$, then with probability at least $1 - \delta$,*

$$\|\hat{\mathbf{y}}_q - \mathbf{y}_q\| < t.$$

Refer to the supplementary material for the proofs of the theorem and corollary. The quality of the approximation depends on $\epsilon_{\frac{n'}{2}}(\mathbf{Y})$ and n'. If data exhibit strong clustering patterns, *i.e.*, data within each cluster are very close to cluster center, we will have small $\epsilon_{\frac{n'}{2}}(\mathbf{Y})$, hence better approximation. Likewise, the bigger n' is, the better approximation is.

2.1.2 Approximation of the prototype algorithm

The clusters can be obtained via clustering algorithm such as K-means. Since the n could be potentially massive, it is impractical to compute α_i within all clusters. Let $\alpha_i(\mathbf{x}_q) = \frac{\mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)}{\sum_{j=1}^{n} \mathrm{w}(\mathbf{x}_q, \mathbf{x}_j)}$. Ideally, for each cluster, we want to select the $\mathbf{y}_i$ that has high overall weight $O_i =$

$\sum_{\mathbf{x}_q \in X} \alpha_i(\mathbf{x}_q)$. For large scale $\mathbf{X}$, we only have limited information available such as cluster centers $\{\mathbf{c}_j, j = 1, \cdots, m\}$ and $\mathrm{w}(\mathbf{c}_j, \mathbf{x}), \mathbf{x} \in \mathbf{X}$. Fortunately, the clustering result gives useful information about O_i. The cluster centers have the largest overall weight w.r.t the points from their own cluster, *i.e.* $\sum_{i \in I_j} \mathrm{w}(\mathbf{c}_j, \mathbf{x}_i)$. This suggests we should select all cluster centers to express $\hat{\mathbf{y}}_q$.

Following many methods in the area (*e.g.*, [21, 33]), we obtain our general inductive hash function by binarizing the low-dimensional embedding

$$h(\mathbf{x}) = \mathrm{sgn}\left(\frac{\sum_{j=1}^{m} \mathrm{w}(\mathbf{x}, \mathbf{c}_j)\mathbf{y}_j}{\sum_{j=1}^{m} \mathrm{w}(\mathbf{x}, \mathbf{c}_j)}\right), \tag{7}$$

where $\mathrm{sgn}(\cdot)$ is the sign function and $\mathbf{Y_B} := \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_m\}$ is the embedding for the base set $\mathbf{B} := \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_m\}$, which is the cluster centers obtained by K-means. Here we assume that the embeddings $\mathbf{y}_i$ are centered on the origin. We term our hashing method *Inductive Manifold-Hashing* (IMH). The inductive hash function provides a natural means for generalization to new data, which has a constant $O(dm + rk)$ time. With this, the embedding for the training data becomes

$$\mathbf{Y} = \bar{\mathbf{W}}_{\mathbf{XB}} \mathbf{Y}_{\mathbf{B}}, \tag{8}$$

where $\bar{\mathbf{W}}_{\mathbf{XB}}$ is defined such that $\bar{\mathbf{W}}_{ij} = \frac{\mathrm{w}(\mathbf{x}_i, \mathbf{c}_j)}{\sum_{i=1}^{m} \mathrm{w}(\mathbf{x}_i, \mathbf{c}_j)}$, for $\mathbf{x}_i \in \mathbf{X}$, $\mathbf{c}_j \in \mathbf{B}$.

Although the objective function (2) is formally related to LE, it is general in preserving local similarity. The embeddings $\mathbf{Y_B}$ can be learned by any appropriate manifold learning method which preserves the similarity of interest in the low dimensional space. We empirically evaluate several other embedding methods in Section 2.4. Actually, as we show, some manifold learning methods (*e.g.*, t-SNE described in Section 2.2) can be better choices for learning binary codes, although LE has been widely used. We will discuss two methods for learning $\mathbf{Y_B}$ in the sequel.

We summarize the Inductive Manifold-Hashing framework in Algorithm 1. Note that the computational cost is dominated by K-means in the first step, which is $O(dmnl)$ in time (with l the number of iterations). Considering that m (normally a few hundreds) is much less than n, and is a function of manifold complexity rather than the volume of data, the total training time is linear in the size of training set. If the embedding method is LE, for example, then using IMH to compute $\mathbf{Y_B}$ requires constructing the small affinity matrix $\mathbf{W_B}$ and solving r eigenvectors of the $m \times m$ Laplacian matrix $\mathbf{L_B}$ which is $O(dm^2 + rm)$. Note that in step 3, to compute $\bar{\mathbf{W}}_{\mathbf{XB}}$, one needs to compute the distance matrix between $\mathbf{B}$ and $\mathbf{X}$, which is a natural output of K-means, or can be computed additionally in $O(dmn)$ time. The training process on a dataset of 70K items with 784 dimensions can thus be achieved in a few seconds on a standard desktop PC.

Algorithm 1 Inductive Manifold-Hashing (IMH)

Input: Training data $\mathbf{X} := \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, code length r, base set size m, neighborhood size k

Output: Binary codes $\mathbf{Y} := \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\} \in \mathbb{R}^{n \times r}$

1) Generate the base set $\mathbf{B}$ by random sampling or clustering (*e.g.* K-means).

2) Embed $\mathbf{B}$ into the low dimensional space by (9), (12) or any other appropriate manifold leaning method.

3) Obtain the low dimensional embedding $\mathbf{Y}$ for the whole dataset inductively by Equation (8).

4) Threshold $\mathbf{Y}$ at zero.

Connection to the Nyström method As Equation (4), the Nyström eigenfunction by Bengio et al. [2] also generalizes to a new point by a linear combination of a set of low dimensional embeddings:

$$\phi(\mathbf{x}) = \sqrt{n} \sum_{j=1}^{n} \tilde{\mathrm{K}}(\mathbf{x}, \mathbf{x}_j) \mathbf{V}_r^j \mathbf{\Sigma}_r^{-1}.$$

For LE, $\mathbf{V}_r$ and $\mathbf{\Sigma}_r$ correspond to the top r eigenvectors and eigenvalues of a normalized kernel matrix $\tilde{K}$ with $\tilde{K}_{ij} = \tilde{\mathrm{K}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{n} \frac{\mathrm{w}(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{E_{\mathbf{x}}[\mathrm{w}(\mathbf{x}_i, \mathbf{x})] E_{\mathbf{x}}[\mathrm{w}(\mathbf{x}, \mathbf{x}_j)]}}$. In AGH [21], the formulated hash function was proved to be the corresponding Nyström eigenfunction with the approximate low-rank affinity matrix. LELVM [5] also formulate out-of-sample mappings for LE in a manner similar to (4) by combining latent variable models. Both of these methods, and ours, can thus be seen as applications of the Nyström method. *Note, however, that our method differs in that it is not restricted to spectral methods such as LE, and that we aim to learn binary hash functions for similarity-based search rather than dimensionality reduction.* LELVM [5] cannot be applied to other embedding methods other than LE.

2.2. Stochastic neighborhood preserving hashing

In order to demonstrate our approach we now derive a hashing method based on t-SNE [29], which is a non-spectral embedding method. t-SNE is a modification of stochastic neighborhood embedding (SNE) [13] which aims to overcome the tendency of that method to crowd points together in one location. t-SNE provides an effective technique for visualizing data and dimensionality reduction, which is capable of preserving local structures in the high dimensional data while retaining some global structures [29]. These properties make t-SNE a good choice for nearest neighbor search. Moreover, as stated in [30], the cost function of t-SNE in fact maximizes the *smoothed recall* [30] of query points and their neighbors.

The original t-SNE does not scale well, as it has a time complexity which is quadratic in n. More significantly, however, it has a non-parametric form, which means that there is no simple function which may be applied to out-of-sample data in order to calculate their coordinates in the embedded space. As was proposed in the previous subsec-

tion, we first apply t-SNE to the base set $\mathbf{B}$ [29],

$$\min_{\mathbf{Y_B}} = \sum_{\mathbf{x}_i \in \mathbf{B}} \sum_{\mathbf{x}_j \in \mathbf{B}} p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right). \quad (9)$$

Here p_{ij} is the symmetrized conditional probability in the high dimensional space, and q_{ij} is the joint probability defined using the t-distribution in the low dimensional embedding space. The optimization problem (9) is easily solved by a gradient descent procedure. After we get embeddings $\mathbf{Y_B}$ of samples $\mathbf{x}_i \in \mathbf{B}$, the hash codes for the entire dataset can be easily computed using (7). It is this method which we label IMH-tSNE.

2.3. Hashing with relaxed similarity preservation

As in the last subsection, we can compute $\mathbf{Y_B}$ considering local smoothness only within $\mathbf{B}$. Based on equation (4), in this subsection, we alternatively compute $\mathbf{Y_B}$ by considering the smoothness both within $\mathbf{B}$ and between $\mathbf{B}$ and $\mathbf{X}$. As in [7], the objective can be easily obtained by modifying (1) as:

$$\begin{aligned} \mathcal{C}(\mathbf{Y_B}) = & \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathbf{B}} \mathrm{w}(\mathbf{x}_i, \mathbf{x}_j) \|\mathbf{y}_i - \mathbf{y}_j\|^2 \quad (\mathcal{C}_{\mathbf{BB}}) \\ & + \lambda \sum_{\mathbf{x}_i \in \mathbf{B}, \mathbf{x}_j \in \mathbf{X}} \mathrm{w}(\mathbf{x}_i, \mathbf{x}_j) \|\mathbf{y}_i - \mathbf{y}_j\|^2 \quad (\mathcal{C}_{\mathbf{BX}}) \end{aligned} \quad (10)$$

where λ is the trade-off parameter. $\mathcal{C}_{\mathbf{BB}}$ enforces smoothness of the learned embeddings within $\mathbf{B}$ while $\mathcal{C}_{\mathbf{BX}}$ ensures the smoothness between $\mathbf{B}$ and $\mathbf{X}$. This formulation is actually a relaxation of (1), by discarding the part which minimizes the dissimilarity within $\mathbf{X}$ (denoted as $\mathcal{C}_{\mathbf{XX}}$). $\mathcal{C}_{\mathbf{XX}}$ is ignored since computing the similarity matrix within $\mathbf{X}$ costs $O(n^2)$ time. The smoothness between points in $\mathbf{X}$ is implicitly ensured by (8).

Applying equation (8) for $\mathbf{y}_j, j \in \mathbf{X}$ to (10), we obtain the following problem

$$\begin{aligned} & \min \mathrm{trace}(\mathbf{Y_B^\top}(\mathbf{D_B} - \mathbf{W_B})\mathbf{Y_B}) \quad (11) \\ & + \lambda \, \mathrm{trace}(\mathbf{Y_B^\top}(\mathbf{D_{BX}} - \bar{\mathbf{W}}_{\mathbf{XB}}^\top \mathbf{W_{XB}})\mathbf{Y_B}), \end{aligned}$$

where $\mathbf{D_B} = \mathrm{diag}(\mathbf{W_B 1})$ and $\mathbf{D_{BX}} = \mathrm{diag}(\mathbf{W_{BX} 1})$ are both $m \times m$ diagonal matrices. Taking the constraint in (1), we obtain

$$\begin{aligned} & \min_{\mathbf{Y_B}} \mathrm{trace}(\mathbf{Y_B^\top}(\mathbf{M} + \lambda \mathbf{T})\mathbf{Y_B}) \quad (12) \\ & \text{s.t. } \mathbf{Y_B^\top Y_B} = m\mathbf{I} \end{aligned}$$

where $\mathbf{M} = \mathbf{D_B} - \mathbf{W_B}$, $\mathbf{T} = \mathbf{D_{BX}} - \bar{\mathbf{W}}_{\mathbf{XB}}^\top \mathbf{W_{XB}}$. The optimal solution $\mathbf{Y_B}$ of the above problem is easily obtained by identifying the r eigenvectors of $\mathbf{M} + \lambda \mathbf{T}$ corresponding to the smallest eigenvalues (excluding the eigenvalue 0 with respect to the trivial eigenvector $\mathbf{1}$)[1]. We name this method IMH-LE in the following text.

[1] We set λ to 2 in all experiments.

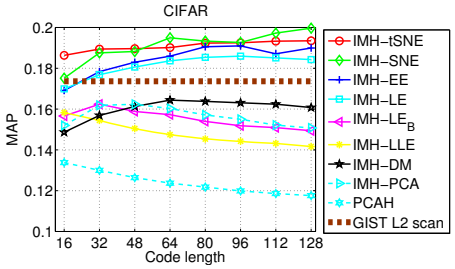


Figure 2: Comparison among different manifold learning methods within our IMH hashing framework on CIFAR-10. IMH with the linear PCA (IMH-PCA) and PCAH [31] are also evaluated for comparison. For clarity, forIMH-LE in Section 2.3, we term IMH with the original LE algorithm on the base set $\mathbf{B}$ as IMH-LE$_\mathbf{B}$. IMH-DM is IMH with the diffusion maps of [18].

2.4. Manifold learning methods for hashing

In this section, we compare different manifold learning methods for hashing within our IMH framework. The comparison results are reported in Figure 2. For comparison, we also evaluate the linear PCA within the framework (IMH-PCA in the figure). We can clearly see that IMH-tSNE, IMH-SNE and IMH-EE (with Elastic Embedding (EE) [4]) perform slightly better than IMH-LE (Section 2.3). This is mainly because these three methods are able to preserve local neighborhood structure while, to some extent, preventing data points from crowding together. It is promising that all of these methods perform better than an exhaustive ℓ_2 scan using the uncompressed GIST features.

Figure 2 shows that LE (IMH-LE$_\mathbf{B}$ in the figure), the most widely used embedding method in hashing, does not perform as well as a variety of other methods (including t-SNE), and in fact performs worse than PCA, which is a linear technique. This is not surprising because LE (and similarly LLE) tends to collapse large portions of the data (and not only nearby samples in the original space) close together in the low-dimensional space. The results are consistent with the analysis in [4, 29]. Based on the above observations, we argue that manifold learning methods (*e.g.* t-SNE, EE), which not only preserve local similarity but also force dissimilar data apart in the low-dimensional space, are more effective than the popular LE for hashing.

It is interesting to see that IMH-PCA outperforms PCAH [31] by a large margin, despite the fact that PCAH is performed on the whole training data set. This shows that the generalization capability of IMH based on a very small set of data points also works for linear dimensionality methods.

3. Experimental results

We evaluate IMH on four large scale image datasets: CIFAR-10[2], MNIST , SIFT1M [31] and GIST1M[3]. The MNIST dataset consists of $70,000$ images, each of 784 dimensions, of handwritten digits from '0' to '9'. As a subset of the well-known 80M tiny image collection [28], CIFAR-10 consists of 60,000 images which are manually labelled

[2] http://www.cs.toronto.edu/~kriz/cifar.html
[3] http://corpus-texmex.irisa.fr/

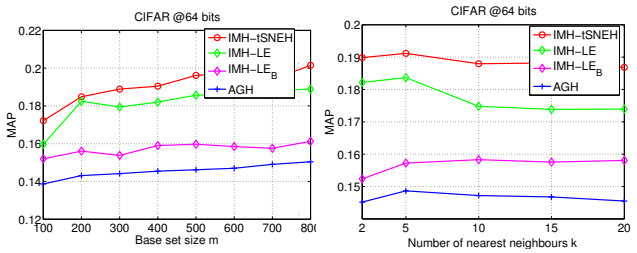


Figure 3: MAP results versus varying base set size m (left, fixing $k = 5$) and number of nearest base points k (right, fixing $m = 400$) for the proposed methods and AGH. The comparison is conducted on the CIFAR-10 dataset using 64-bits .

bits		IMH-LE$_\mathbf{B}$	IMH-LE	IMH-tSNE	AGH
32	Random	14.07	16.20	17.26	-
	K-means	**16.05**	**17.48**	**18.38**	15.76
64	Random	14.64	16.98	16.93	-
	K-means	**15.90**	**18.20**	**19.0**4	14.55
96	Random	14.76	17.02	17.21	-
	K-means	**15.46**	**18.56**	**19.41**	13.98

Table 1: MAP (%) evaluation of different base generating methods: random sampling vs. K-means. The comparison is performed on the CIFAR-10 dataset with code lengths from 32 to 96 and base set size 400.

as 10 classes with $6,000$ samples for each class. We represent each image in this dataset by a GIST feature vector [23] of dimension 512. For MNIST and CIFAR-10, the whole dataset is split into a test set with $1,000$ samples and a training set with all remaining samples.

We compare nine hashing algorithms including the proposed IMH-tSNE, IMH-LE and seven other unsupervised state-of-the-art methods: PCAH [31], SH [33], AGH [21] and STH [35], BRE [15], ITQ [9], Spherical Hashing (SpH) [11]. We use the provided codes and suggested parameters according to the authors of these methods. Because our methods are fully unsupervised we did not consider supervised methods in our experiments. Due to the high computational cost of BRE and high memory cost of STH, we sample $1,000$ and $5,000$ training points for these two methods respectively. We measure performance by mean of average precision (MAP) or precision and recall curves for *hamming ranking* using 16 to 128 hash bits. We also report the results for *hash lookup* using a Hamming radius within 2 by F1 score [22]: $F_1 = 2(precision \cdot recall)/(precision + recall)$. Ground truths are defined by the category information for the labeled datasets MNIST and CIFAR-10, and by Euclidean neighbors for SIFT1M and GIST1M.

Base selection In this section, we take the CIFAR-10 dataset for example to compare different base generation methods and different base sizes for the proposed methods. AGH is also evaluated here for comparison. Table 1 compares two methods for generating base point sets: random sampling and K-means on the training data. Not surprisingly, we see that the performance of our methods using K-means is better at all code lengths than that using random sampling. Also we can see that, even with base set by random sampling, the proposed methods outperform AGH in all cases but one. Due to the superior results and high efficiency in practice, we generate the base set by K-means in the following experiments.

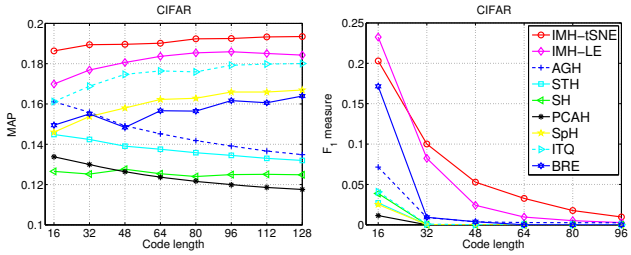


Figure 4: Comparison of different methods on CIFAR-10 based on MAP (left) and F_1 (right) for varying code lengths.

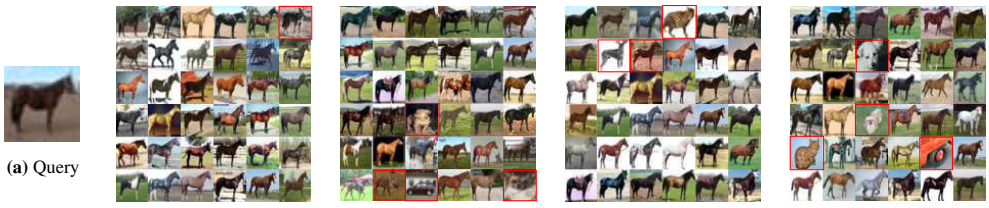
From Figure 3, we see that the performance of the proposed methods and AGH do not change significantly with both the base set size m and the number of nearest base points k. Based on this observation, for the remainder of this paper, we set $m = 400$ and $k = 5$ for our methods, unless otherwise specified. Also it is clear that IMH-LE$_\mathbf{B}$, which only enforces smoothness in the base set, does not perform as well as IMH-LE, which also enforces smoothness between the base set and training set. Note, however, that IMH-LE$_\mathbf{B}$ is still better than AGH on this dataset.

Results on CIFAR-10 dataset We report the comparative results based on MAP for hamming ranking with code lengths from 16 to 128 bits in Figure 4. We see that the proposed IMH-LE and IMH-tSNE perform best in all cases. Among the proposed algorithms, the LE based IMH-LE is inferior to the t-SNE based IMH-tSNE. IMH-LE is still much better than AGH and STH, however. ITQ performs better than SpH and BRE on this dataset, but is still inferior to IMH. SH and PCAH perform worst in this case, because SH relies upon its uniform data assumption while PCAH simply generates the hash hyperplanes by PCA directions, which does not explicitly capture the similarity information. The results are consistent with the complete precision and recall curves shown in the supplementary material. We also report the F_1 results for hash lookup with Hamming radius 2 It is can be seen that IMH-LE and IMH-tSNE also outperform all other methods by large margins. BRE and AGH obtain better results than the remaining methods, although the performance of all methods drop as code length grows.

Figure 5 shows the precision and recall curves of hamming ranking for the compared methods. We see that STH and AGH obtain relatively high precisions when a small number of samples are returned, however precision drops significantly as the number of retrieved samples increases. In contrast, IMH-tSNE, IMH-LE and ITQ achieve higher precisions with relatively larger numbers of retrieved points.

We also show qualitative results of IMH and related methods on a sample query in Figure 6. As can be seen, IMH-tSNEH achieves the best search quality in term of visual relevance.

Results on MNIST dataset The MAP and F_1 scores for these compared methods are reported in Figure 7. As in Figure 4, IMH-tSNE achieves the best results. On this dataset



(a) Query **(b)** IMH-tSNE **(c)** SH **(d)** AGH **(e)** STH

Figure 6: The query image (a) and the query results returned by various methods with 32 hash bits. False positive returns are marked with red borders.

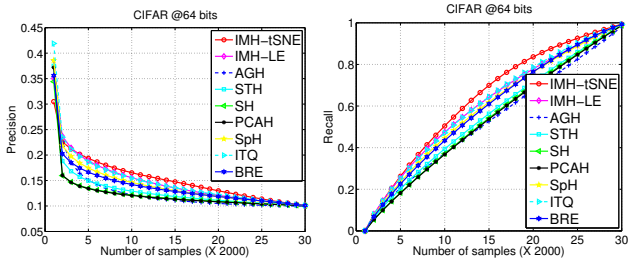


Figure 5: Comparison of different methods on CIFAR-10 based on precision (left) and recall (right) using 64-bits. Please refer to the complementary for complete results for other code lengths.

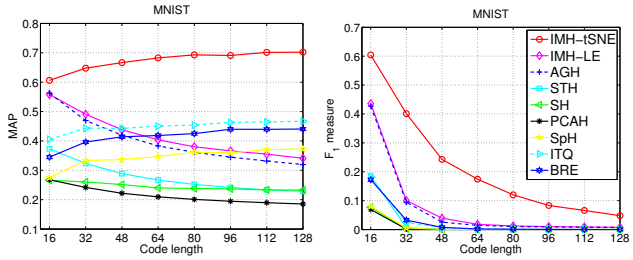


Figure 7: Comparison of different methods on the MNIST dataset using MAP (left) and F_1 (right) for varying code lengths.

we can clearly see that IMH-tSNE outperforms IMH-LE by a large margin, which increases as code length increases. This further demonstrates the advantage of t-SNE as a tool for hashing by embedding high dimensional data into a low dimensional space. The dimensionality reduction procedure not only preserves the local neighborhood structure, but also reveals important global structure (such as clusters) [29]. Among the four LE-based methods, while IMH-LE shows a small advantage over AGH, both methods achieve much better results than STH and SH. ITQ and BRE obtain high MAPs with longer bit lengths, but they still perform less well for the hash look up F_1. PCAH performs worst in terms of both MAP and the F_1 measure. Refer to the supplementary material for the complete precision and recall curves which validate the observations here.

Method	Train time		Test time	
	64-bits	128-bits	64-bits	128-bits
IMH-LE	9.9	9.9	5.1×10^{-5}	3.8×10^{-5}
IMH-tSNE	16.7	20.2	2.8×10^{-5}	3.1×10^{-5}
SH	6.8	16.2	5.8×10^{-5}	1.8×10^{-4}
STH	266.1	485.4	1.8×10^{-3}	3.6×10^{-3}
AGH	9.5	9.5	4.7×10^{-5}	5.5×10^{-5}
PCAH	3.8	4.1	5.7×10^{-6}	1.2×10^{-5}
SpH	19.7	41.0	1.3×10^{-5}	2.0×10^{-5}
ITQ	10.4	20.3	6.9×10^{-6}	1.1×10^{-5}
BRE	418.9	1731.9	1.2×10^{-5}	2.4×10^{-5}

Table 2: Comparison of training and testing times (in seconds) on MNIST with 70K 784D feature points. K-means dominates the cost of AGH and IMH (8.9 seconds), which can be conducted in advance in practice. The experiments are based on a desktop PC with a 4-core 3.07GHZ CPU and 8G RAM.

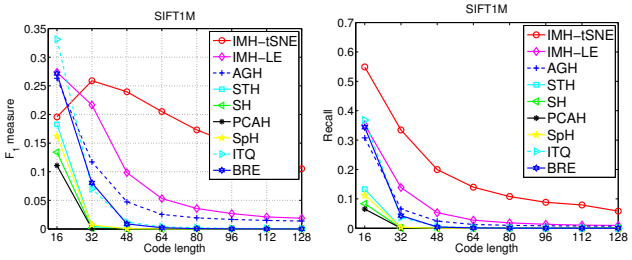


Figure 8: Comparative results results on SIFT1M for F_1 (left) and recall (right) with hamming radius 2. Ground truth is defined to be the closest 2 percent of points as measured by the Euclidean distance.

Efficiency Table 2 shows training and testing time on the MNIST dataset for various methods, and shows that the linear method, PCAH, is fastest. IMH-tSNE is slower than IMH-LE, AGH and SH in terms of training time, however all of these methods have relatively low execution times and are much faster than STH and BRE. In terms of test time, both IMH algorithms are comparable to other methods, except STH which takes much more time to predict the binary codes by SVM on this non-sparse dataset.

Results on SIFT1M and GIST1M SIFT1M contains one million local SIFT descriptors extracted from a large set of images [31], each of which is represented by a 128D vector of histograms of gradient orientations. GIST1M contains one million GIST features and each feature is represented by a 960D vector. For both of these datasets, one million samples are used as training set and additional 10K are used for testing. As in [31], ground truth is defined as the closest 2 percent of points as measured by the Euclidean distance. For these two large datasets, we generate $1,000$ points by K-means and set $k = 2$ for both IMH and AGH. The comparative results on SIFT1M and GIST1M are summarized in Figure 8 and Figure 9, respectively. Again, IMH consistently achieves superior results in terms of both F_1 score and recall with hamming radius 2. We see that the performance of most of these methods decreases dramatically with increasing code length as the hamming spaces become more sparse, which makes the hash lookup fail more often. However IMH-tSNE still achieves relatively high scores with large code lengths. If we look at Figure 8 (left), ITQ obtains the highest F_1 with 16-bits, however it decreases to near zero at 64-bits. In contrast, IMH-tSNE still manages an F_1 of 0.2. Similar results are observed in

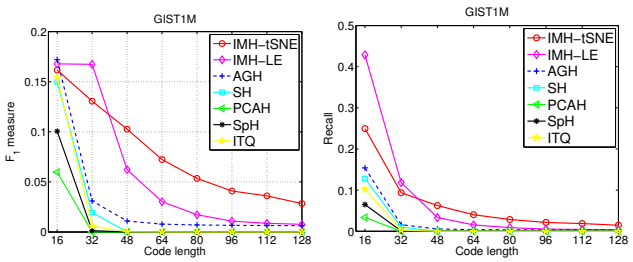


Figure 9: Comparative results results on GIST1M by F_1 (left) and recall (right) with hamming radius 2. Ground truth is defined to be the closest 2 percent of points as measured by the Euclidean distance.

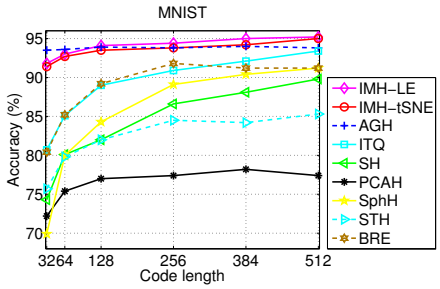


Figure 10: Classification accuracy (%) on MNIST with binary codes of various hashing methods by linear SVM.

the recall curves.

Classification on binary codes In order to demonstrate classification performance we have trained a linear SVM on the binary codes generate by IMH for the MNIST data set. In order to learn codes with higher bit lengths for IMH and AGH, we set the size of the base set to $1,000$. Accuracies of different binary encodings are shown in Figure 10. Both IMH and AGH achieve high accuracies on this dataset, although IMH performs better with higher code lengths. In contrast, the best results of all other methods, obtained by ITQ, are consistently worse than those for IMH, especially for short code lengths. Note that even with only 128-bit binary features IMH obtains a high 94.1%. Interestingly, we get the same classification rate of 94.1% applying the linear SVM to the uncompressed 784D features, which occupy several hundreds times as much space as the learned hash codes.

Conclusion We have proposed a simple yet effective hashing framework which provides a practical connection between manifold learning methods (typically non-parametric and with high computational cost) and hash function learning (requiring high efficiency). By preserving the underlying manifold structure with several non-parametric dimensionality reduction methods, the proposed hashing methods outperform several state-of-the-art methods in terms of both hash lookup and hamming ranking on several large-scale retrieval-datasets. The proposed inductive formulation of the hash function sees the proposed methods require only linear time ($O(n)$) for indexing all of the training data and a constant search time for a novel query. The learned hash codes were also shown to have promising results on a classification problem even with very short code lengths.

This work was in part supported by ARC Future Fellowship FT120100969.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proc. Adv. Neural Inf. Process. Syst.*, 2001.

[2] Y. Bengio, O. Delalleau, N. Roux, J. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Comput.*, 16(10):2197–2219, 2004.

[3] M. M. Bronstein and P. Fua. LDAHash: Improved matching with smaller descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2012.

[4] M. Carreira-Perpinán. The elastic embedding algorithm for dimensionality reduction. In *Proc. Int. Conf. Mach. Learn.*, 2010.

[5] M. Carreira-Perpinán and Z. Lu. The laplacian eigenmaps latent variable model. *Proc. Int. Conf. Artif. Intell. Stat.*, 2007.

[6] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Ann. Symp. Comput. Geometry*, 2004.

[7] O. Delalleau, Y. Bengio, and N. Le Roux. Efficient non-parametric function induction in semi-supervised learning. In *Proc. Int. Workshop Artif. Intelli. Stat.*, pages 96–103, 2005.

[8] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. Int. Conf. Very Large Databases*, 1999.

[9] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2011.

[10] X. He, W.-Y. Ma, and H.-J. Zhang. Learning an image manifold for retrieval. In *Proc. ACM Multimedia*, 2004.

[11] J. Heo, Y. Lee, J. He, S. Chang, and S. Yoon. Spherical hashing. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012.

[12] R. Herbrich and R. C. Williamson. Algorithmic luckiness. *J. Mach. Learn. Res.*, 3:175–212, 2002.

[13] G. Hinton and S. Roweis. Stochastic neighbor embedding. In *Proc. Adv. Neural Inf. Process. Syst.*, 2002.

[14] A. Joly and O. Buisson. Random maximum margin hashing. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2011.

[15] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Proc. Adv. Neural Inf. Process. Syst.*, 2009.

[16] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2009.

[17] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 2143–2157, 2009.

[18] S. Lafon and A. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(9):1393–1403, 2006.

[19] X. Li, G. Lin, C. Shen, A. van den Hengel, and A. Dick. Learning hash functions using column generation. In *Proc. Int. Conf. Mach. Learn.*, 2013.

[20] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang. Supervised hashing with kernels. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012.

[21] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *Proc. Int. Conf. Mach. Learn.*, 2011.

[22] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[23] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comp. Vis.*, 2001.

[24] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Proc. Adv. Neural Inf. Process. Syst.*, 2009.

[25] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, pages 2323–2326, 2000.

[26] A. Talwalkar, S. Kumar, and H. Rowley. Large-scale manifold learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2008.

[27] J. Tenenbaum, V. De Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, pages 2319–2323, 2000.

[28] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1958–1970, 2008.

[29] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *J. Mach. Learn. Res.*, 9:2579–2605, 2008.

[30] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *J. Mach. Learn. Res.*, 11:451–490, 2010.

[31] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393 –2406, 2012.

[32] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2010.

[33] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proc. Adv. Neural Inf. Process. Syst.*, 2008.

[34] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *Proc. Adv. Neural Inf. Process. Syst.*, 2009.

[35] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proc. ACM SIGIR Conf.*, 2010.