Locally Linear Hashing for Extracting Non-Linear Manifolds

Go Irie
NTT Corporation
Kanagawa, Japan
irie.go@lab.ntt.co.jp

Zhenguo Li
Huawei Noah's Ark Lab
Hong Kong, China
li.zhenguo@huawei.com

Xiao-Ming Wu, Shih-Fu Chang
Columbia University
New York, NY, USA
{xmwu, sfchang}@ee.columbia.edu

Abstract

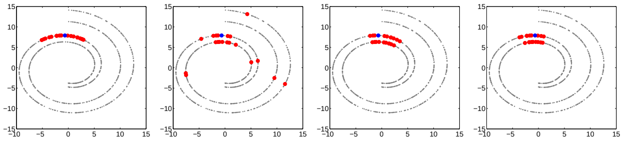
Previous efforts in hashing intend to preserve data variance or pairwise affinity, but neither is adequate in capturing the manifold structures hidden in most visual data. In this paper, we tackle this problem by reconstructing the locally linear structures of manifolds in the binary Hamming space, which can be learned by locality-sensitive sparse coding. We cast the problem as a joint minimization of reconstruction error and quantization loss, and show that, despite its NP-hardness, a local optimum can be obtained efficiently via alternative optimization. Our method distinguishes itself from existing methods in its remarkable ability to extract the nearest neighbors of the query from the same manifold, instead of from the ambient space. On extensive experiments on various image benchmarks, our results improve previous state-of-the-art by 28-74% typically, and 627% on the Yale face data.

1. Introduction

Hashing is a powerful technique for efficient large-scale retrieval. An effective hashing method is expected to efficiently learn compact and similarity-preserving binary codes for representing images or documents in a large database. How to improve hashing accuracy and efficiency is a key issue and has attracted considerable attention.

The classical kd-tree [18] allows logarithmic query time, but is known to scale poorly with data dimensionality. The seminal locality sensitive hashing (LSH) [8] enables sub-linear search time in high dimension, but usually requires long hash codes. To generate compact codes, it is realized that hash functions should be adapted to data distribution. This leads to a variety of data-dependent methods, including PCA-based hashing [20, 5], graph-based hashing [24, 13], (semi-)supervised hashing [20, 12, 10, 15], and others [11, 7, 6], to name a few.

Based on the widely adopted manifold assumption that semantically similar items tend to form a low-dimensional manifold [16, 22], two major approaches are adopted in data-dependent methods to reduce the dimensionality of data such that it can be represented by more compact codes. One is to perform principal component analysis (PCA) to preserve data variance, such as PCAH [20] and ITQ [5], which only works for linear manifolds. The other is to employ non-linear dimension reduction techniques to preserve pairwise affinity, such as spectral hashing (SH) [24], anchor graph hashing (AGH) [13], and inductive manifold hashing (IMH-tSNE) [19].



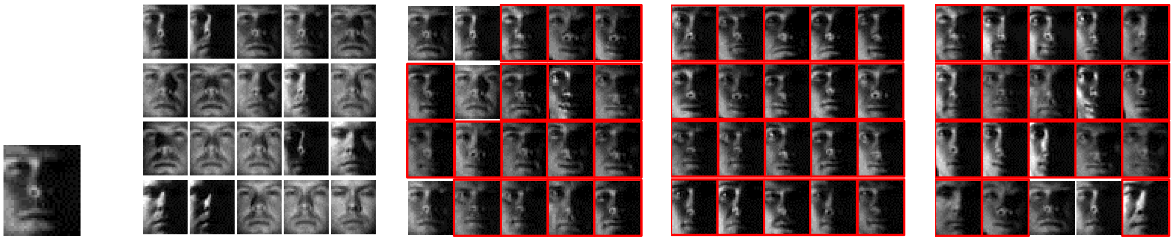
(a) Our LLH (b) SH [24] (c) AGH [13] (d) IMH-tSNE [19]

Figure 1. Retrieval on **Two-SwissRolls**. Top 2% retrieved points (red) with the query (blue) are shown. The proposed LLH is able to retrieve the closest items in the same manifold as the query.

While variance is a global property of data and pairwise affinity is a first-order structure, neither of them is adequate to capture the local geometric structures of manifolds (e.g., the locally linear structure), which, however, contain the most valuable information for nearest neighbor search. The feature space of a large-scale database of multiple categories is usually cluttered with manifolds at different scales and dimensionalities, which may not be well separated, and sometimes even overlap. Items which are semantically similar to the query, may not necessarily be the closest ones in the ambient feature space, but the ones lying on the same manifold. Preserving global properties or first-order structures while ignoring local, higher-order structures will destroy the intrinsic data structures and result in unsatisfactory performance (Fig. 1(b-d) & Fig. 2(c-e)).

In this paper, instead of preserving global properties, we propose to preserve local geometric structures. Our proposed method, called Locally Linear Hashing (LLH), is able to identify nearest neighbors of the query from the same manifold and demonstrates significantly better retrieval quality than other hashing methods (Fig. 1(a) & Fig. 2(b)). Our key contributions are:

- We capture the local linearity of manifolds using locality-sensitive sparse coding, which favors the closest items located in the same manifold as the query. In contrast, most previous manifold learning methods, such as Locally Linear Embedding (LLE) [17], simply



(a) Query (b) Our LLH (c) SH [24] (d) AGH [13] (e) IMH-tSNE [19]

Figure 2. Retrieval on **Yale** using 64-bit codes. The top 20 retrieved images are shown, with false positive highlighted in red. It demonstrates the superb benefit of modeling the locally linear structures of data.

adopt k-nearest neighbors, which may include points from other nearby manifolds.

- We preserve the learned locally linear structures in a low-dimensional Hamming space through a joint minimization of the reconstruction error and the quantization loss. Despite its NP-hardness, we show that a local optimum can be obtained efficiently by alternating optimization between optimal reconstruction and quantization.
- We design an out-of-sample extension which is effective in handling new points not seen in the training set. Our results improve the performances of the state-of-the-art methods by 28-74% typically, and 627% on the Yale face data.

We present our method LLH in Sec. 2. An extensive experimental evaluation of LLH is provided in Sec. 3.

2. Locally Linear Hashing

Denote by $\mathcal{F} \subset \mathbb{R}^D$ a feature space generated by an unknown probability distribution $\mathcal{P} : \mathbb{R}^D \to \mathbb{R}_+$. In practice, $\mathcal{F}$ may contain multiple non-linear manifolds with different intrinsic dimensionality, and the number of manifolds is unknown. Given a dataset $\mathcal{X} := \{\mathbf{x}_i\}_{i=1}^n$ sampled i.i.d. from $\mathcal{P}$, our goal is to learn a mapping from $\mathcal{F}$ to a low-dimensional Hamming space $\mathcal{H} := \{\pm 1\}^c$ such that the manifold structures of $\mathcal{F}$ is optimally preserved in $\mathcal{H}$ (c is manually specified).

Let $\mathcal{M} \subset \mathcal{F}$ be a non-linear manifold of intrinsic dimension $d \ll D$. By definition, $\mathcal{M}$ is a topological space which is only locally Euclidean, i.e., neighborhood of each feature point is homeomorphic to $\mathbb{R}^d$. We can then make the following observation.

Observation 1. *Any point $\mathbf{x} \in \mathcal{M}$ can be linearly spanned by $d+1$ points in general position in its tangent space.*

This simple observation tells us that a set of $d+1$ neighbor points of $\mathbf{x}$ on $\mathcal{M}$ is sufficient to capture the locally linear structures of $\mathcal{M}$ around $\mathbf{x}$. Therefore, to preserve the manifold structures of $\mathcal{M}$ is to preserve such locally linear structures. Based on this observation, our framework consists of three major steps:

1) Capturing the locally linear structures of data. For any $\mathbf{x} \in \mathcal{X}$, the locally linear structure at $\mathbf{x}$ is captured by a set of its nearest neighbor points on the same manifold that linearly span $\mathbf{x}$, with the reconstruction weights. We introduce a principled method to identify such neighbors and compute the reconstruction weights simultaneously (Sec. 2.1).

2) Preserving locally linear structures in $\mathcal{H}$. We propose a new formulation which jointly minimizes the embedding error and the quantization loss (Sec. 2.2).

3) Out-of-sample extension. We design an effective way to generate binary codes for any unseen point $\mathbf{q} \in \mathcal{F}$ but $\mathbf{q} \notin \mathcal{X}$ (Sec. 2.3).

2.1. Capturing Locally Linear Structures

Given a data point $\mathbf{x} \in \mathcal{X}$, denote by $\mathcal{M}$ the manifold where $\mathbf{x}$ lies on. Our goal is to find $\mathcal{N}_{\mathcal{M}}(\mathbf{x})$, a set of neighbor points of $\mathbf{x}$ from $\mathcal{M}$ that is able to linearly span $\mathbf{x}$. Let $\mathcal{N}_E(\mathbf{x})$ be a set of nearest neighbors of $\mathbf{x}$ in the ambient space that can linearly span $\mathbf{x}$. Normally $\mathcal{N}_E(\mathbf{x}) \neq \mathcal{N}_{\mathcal{M}}(\mathbf{x})$. Since in real-world data, manifolds may be close to each other or even overlap, $\mathcal{N}_E(\mathbf{x})$ is likely to contain data points from other manifolds. However, if $\mathcal{M}$ is sampled properly (under mild conditions), we can safely assume $\mathcal{N}_{\mathcal{M}}(\mathbf{x}) \subset \mathcal{N}_E(\mathbf{x})$. Then the problem becomes that given $\mathcal{N}_E(\mathbf{x})$, how to find its subset $\mathcal{N}_{\mathcal{M}}(\mathbf{x})$.

To this end, we introduce the Locally Linear Sparse Reconstruction (LLSR):

$$\text{(LLSR)} \quad \min_{\mathbf{w}_i} \ \lambda \|\mathbf{s}_i^\top \mathbf{w}_i\|_1 + \frac{1}{2}\|\mathbf{x}_i - \sum_{j \in \mathcal{N}_E(\mathbf{x}_i)} w_{ij}\mathbf{x}_j\|^2 \tag{1}$$

$$\text{s.t.: } \mathbf{w}_i^\top \mathbf{1} = 1, \tag{2}$$

where $\mathbf{w}_i = (w_{i1}, \ldots, w_{in})^\top$, and $w_{ij} = 0$ if $j \notin \mathcal{N}_E(\mathbf{x}_i)$. The first term is a sparse term that penalizes distant points with $\mathbf{s}_i = (s_{i1}, \ldots, s_{in})^\top$ such that s_{ij} is large if $\mathbf{x}_j$ is far from $\mathbf{x}_i$. Here we set $s_{ij} = \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sum_{j \in \mathcal{N}_E(\mathbf{x}_i)} \|\mathbf{x}_i - \mathbf{x}_j\|}$. The second term is a locally linear reconstruction term. Note that this is a small problem with $|\mathcal{N}_E(\mathbf{x}_i)|$ variables (the construction of $\mathcal{N}_E(\mathbf{x}_i)$ is discussed in Sec. 2.3), and can be solved with typical sparse coding algorithms. In this paper, we use a homotopy algorithm [2] since $\mathbf{w}_i$ is expected to be highly sparse.

The key idea of LLSR is to use ℓ_1-norm to impose sparsity, and use weighting to favor closer points, while minimizing the reconstruction cost. Based on Observation 1, the sparsest linear reconstruction (i.e., the lowest-dimensional reconstruction) of $\mathbf{x}_i$ is given by its $d+1$ neighbors from the same manifold. Therefore, by imposing the weighted sparsity constraint, the solution of LLSR is expected to find $\mathcal{N}_{\mathcal{M}}(\mathbf{x}_i)$ (corresponding to the non-zero elements in $\mathbf{w}_i$), i.e., neighbors on the same manifold rather than from the ambient space. This distinguishes LLSR from LLC [21]. We also note that a similar idea was used for spectral clustering in a recent work [3].

2.2. Preserving Locally Linear Structures

After learning the locally linear structures of data in the sparse matrix $W = [\mathbf{w}_1, \ldots, \mathbf{w}_n]$, our next goal is to optimally reconstruct W in the Hamming space $\mathcal{H}$. Specifically, denoting by $\mathbf{y}_i \in \mathcal{H}$ the binary code for $\mathbf{x}_i$, we consider the optimization problem:

$$\min_{\mathbf{y}_1,\ldots,\mathbf{y}_n \in \mathcal{H}} \sum_i \|\mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j\|^2 = \mathrm{tr}(Y^\top M Y), \quad (3)$$

where $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top$, and $M = (I_n - W)^\top (I_n - W)$ is a sparse matrix. Minimizing the objective function, i.e., the reconstruction (embedding) error, is supposed to preserve the locally linear structures optimally. Unfortunately, this optimization problem is NP-hard due to the binary constraints $\mathbf{y}_i \in \mathcal{H}$.

A natural idea to conquer this is to relax the binary constraints to find a continuous embedding, and then binarize it. For example, one can use LLE [17] to obtain the continuous embedding (with additional orthogonal constraints), and then get binary codes by simple thresholding or other quantization method [5]. Despite its simplicity, such a naíve two-stage solution is likely to incur larger errors and result in sub-optimal performance (shown later). Below we show that it is possible to jointly minimize the embedding error and the quantization loss.

Formally, we propose the optimization problem:

$$\min_{Y,R\in\mathbb{R}^{c\times c}, Z\in\mathbb{R}^{n\times c}} \mathrm{tr}(Z^\top M Z) + \eta\|Y - ZR\|_F^2 \quad (4)$$

$$\text{s.t.: } Y \in \{\pm 1\}^{n\times c}, R^\top R = I_c \quad (5)$$

where the first term in Eq. (4) is the embedding error (as in Eq. (3)), the second term is the quantization loss, and $\eta \geq 0$ is a balancing parameter. Here, Z serves as a continuous embedding, R is an orthogonal transformation that rotates Z to align with the hypercube $\{\pm 1\}^{n\times c}$ as close as possible, and Y is the desired binary embedding. Note that we do not need to impose orthogonality constraints on Z, as opposed to LLE[1]. This offers additional freedom to further minimize the reconstruction error.

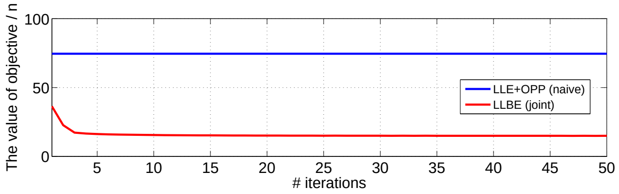
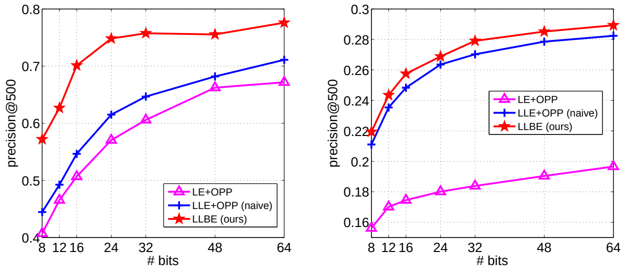


Figure 3. Behavior of the objective function Eq. (4) by the naíve LLE+OPP and our LLBE.



(a) (b)

Figure 4. Averaged precision vs. #bits on (a) **USPS** and (b) **CIFAR**.

Although the problem is still NP-hard, its sub-problem w.r.t. each of Y, R, and Z is convex. Therefore, we can minimize it in an alternating procedure.

Fixing Z, optimize Y and R. This reduces to the Orthogonal Procrustes Problem (OPP):

$$(\text{OPP}) \quad \min_{Y,R} \|Y - ZR\|_F^2 \quad (6)$$

$$\text{s.t.: } Y \in \{\pm 1\}^{n\times c}, R^\top R = I_c. \quad (7)$$

A local optimum of OPP can be obtained by alternating minimization between Y and R [25, 5].

Fixing Y and R, optimize Z. The second term can be rewritten as:

$$\|Y - ZR\|_F^2 = \|Y\|_F^2 + \|Z\|_F^2 - 2\mathrm{tr}(RY^\top Z) \quad (8)$$

$$= \mathrm{tr}(Z^\top Z - 2RY^\top Z) + const. \quad (9)$$

Thus, under fixed Y and R, the optimization problem w.r.t. Z becomes:

$$\min_Z \mathrm{tr}(Z^\top (M + \eta I_n) Z - 2\eta RY^\top Z) \quad (10)$$

$$= \mathrm{tr}(Z^\top A Z - 2B^\top Z) \quad (11)$$

where $A := M + \eta I_n$ is a positive-definite matrix and $B := \eta Y R^\top$. This is a simple convex quadratic program, and can be solved efficiently with the conjugate gradient method [4].

In our experiments, we initialize Z by LLE [17], and then alternate between updating Y, R, and Z for several iterations. The typical behavior of the objective value of Eq. (4) is shown in Fig. 3, where our embedding method is

[1] The orthogonality constraints in LLE and many other manifold learning methods are imposed primarily for computational tractability, which may impair hashing performance [20, 23].

denoted as Locally Linear Binary Embedding (LLBE) and the naíve two-stage algorithm is denoted as LLE+OPP. It is shown that our algorithm converges after only a few alternating iterations (we use 10 iterations for all the experiments in Sec. 3), and achieves significantly smaller error. Fig. 4 shows semantic retrieval accuracies on two datasets (USPS and CIFAR, details are given later in Sec. 3), where LLBE clearly outperforms the naive approach.

Remark. Some existing methods [24, 13] follow Laplacian Eigenmaps (LE) [1] to optimize:

$$\text{(LE)} \quad \min_{Z} \sum_{i,j} w_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|^2 \tag{12}$$

$$\text{s.t. } Z^\top Z = nI_c, \; \mathbf{1}^\top Z = 0, \tag{13}$$

where $Z = [\mathbf{z}_1, \dots, \mathbf{z}_n]^\top$. The objective is to preserve data affinity in $\mathcal{H}$. However, its significant drawback is that, no matter how accurately one can capture the locally linear structures, LE may not be able to recover them in $\mathcal{H}$. We also experimentally compare our LLBE to LE+OPP with the same W computed by LLSR. The results in Fig. 4 show that our LLBE, or even the naíve LLE+OPP, outperforms LE+OPP with significant gains.

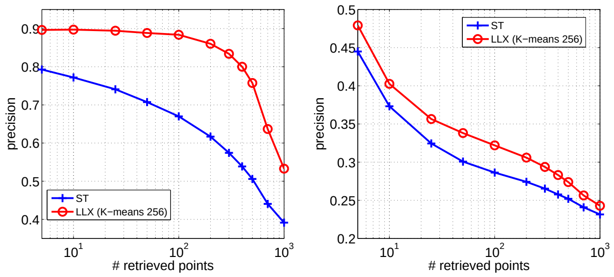
2.3. Out-of-Sample Extension

We have presented a model LLBE for computing binary codes Y for the training data $X := [\mathbf{x}_1, \dots, \mathbf{x}_n]$, the next question is how to generalize it for any query $\mathbf{q} \notin \mathcal{X}$. One idea is to train a set of binary classifiers as hash functions, by using (X, Y) as training data[2], as does in the Self-Taught (ST) method [26] where linear SVMs are trained. While being generic, it ignores the important information of data structures. Another idea is to follow AGH [13] and IMH-tSNE [19] to use a weighted linear combination of the binary codes from $\mathbf{q}$'s k-nearest anchor points. However, such anchor points may come from different manifolds and the weights used seem arbitrary. Here we propose an out-of-sample extension method tailored to our LLBE model.

Recall that the locally linear structure at $\mathbf{q}$ is captured by a set of its nearest neighbor points on the same manifold with the associated reconstruction weights $\mathbf{w_q}$, which can be obtained by solving LLSR. To optimally preserve such structures in the Hamming space, the binary code $\mathbf{y_q}$ for $\mathbf{q}$ should minimize the reconstruction error:

$$\min_{\mathbf{y_q} \in \mathcal{H}} \|\mathbf{y_q} - Y^\top \mathbf{w_q}\|^2. \tag{14}$$

Clearly its solution is $\mathbf{y_q} = \text{sign}(Y^\top \mathbf{w_q})$, which is actually a linear combination of codes of a small subset of the nearest neighbors learned by LLSR, weighted by $\mathbf{w_q}$. It takes only $O(ct)$ time on average to compute, where t is the number of non-zero elements in $\mathbf{w_q}$, and is supposed to be the number of intrinsic dimensionality of the manifold where $\mathbf{q}$ belongs. Typically $t \ll n$.



(a)

(b)

Figure 5. Out-of-sample extension on (a) **USPS** and (b) **CIFAR**. Averaged precision vs. # of retrieved points using 32-bit codes.

The question now becomes to identify $\mathcal{N}_E(\mathbf{q})$ so that we can apply LLSR to obtain $\mathbf{w_q}$. Here we use an efficient procedure to collect $\mathcal{N}_E(\mathbf{q})$. We first randomly sample $\mathcal{X}_m := \{\mathbf{x}_i\}_{i=1}^m$ $(m \le n)$ data points from $\mathcal{X}$, and then apply K-means to group $\mathcal{X}_m$ into K clusters. Note that this can be performed offline in the training stage. Given a query $\mathbf{q}$, we find its nearest cluster (i.e., the one $\mathbf{q}$ has the smallest distance to its centroid) and use all the points in the cluster as $\mathcal{N}_E(\mathbf{q})$, which takes only $O(KD)$ time. Then we solve LLSR with $\mathcal{N}_E(\mathbf{q})$ to compute $\mathbf{w_q}$, which takes $O(t^3 + |\mathcal{N}_E(\mathbf{q})|)$ time on average with the homotopy algorithm [2]. Using this procedure, $\mathbf{w_q}$ can be obtained in constant time, regardless of n. This idea can also be used to efficiently collect $\mathcal{N}_E(\mathbf{x})$ in the training stage, so K-means only needs to be run once for training and querying.

We compare our out-of-sample extension method, called Locally Linear Extension (LLX), with the ST method. As shown in Fig. 5, LLX performs significantly better than ST.

2.4. Complexity

Out-of-sample coding. As discussed in Sec. 2.3, its time complexity is $O(KD+ct+t^3+|\mathcal{N}_E(\mathbf{q})|) \approx O(m)$, with K, D, c, and t fixed (note that $|\mathcal{N}_E(\mathbf{q})| \approx m/K$), which is constant w.r.t. n. The space complexity is $O((c+D)|\mathcal{N}_E(\mathbf{q})| + KD) \approx O(m)$, which is also constant w.r.t. n.

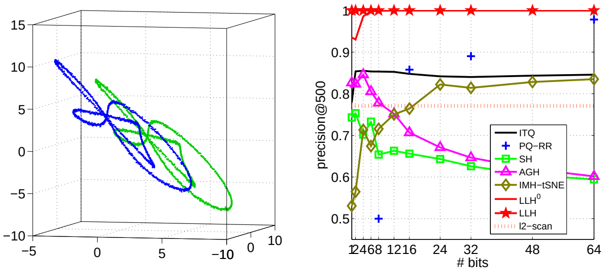
Training. For a large database of size n, we sample a subset of size m for training (as discussed in Sec. 2.3) and compute the binary codes for the remaining items using LLX. The time complexity of K-means is linear in m. The bottleneck for training is to solve LLBE which takes $O(m^2)$ for conjugate gradient method to update Z. The training time thus is quadratic[3] in m. To construct the entire database, we then apply LLX to the remaining $(n-m)$ data points which takes $O(nm)$ time. Note that training and database construction are performed offline, and can be readily parallelized using multiple CPUs. The space complexity for training is $O(m)$.

3. Experiments

In this section, we experimentally compare our LLH to the state-of-the-art hashing methods, including ITQ [5], SH

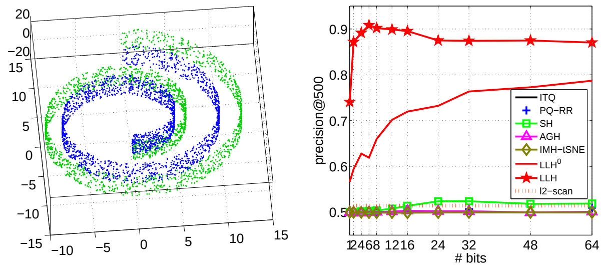
[2] Note each column of Y defines a binary classification problem of X.

[3] In our experiments, it takes about 30 minutes for training on MIRFLICKR-1M dataset.



(a) (b)

Figure 6. (a) **Two-TrefoilKnots**; (b) Averaged precision vs. #bits.



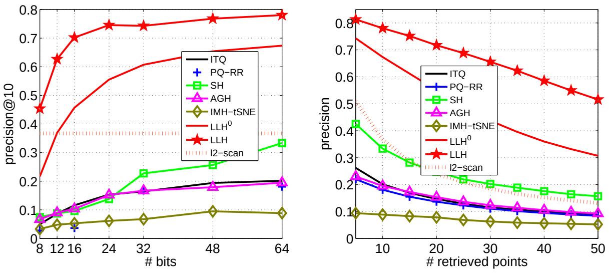
(a) (b)

Figure 7. (a) **Two-SwissRolls**; (b) Averaged precision vs. #bits.

[24], AGH [13], and IMH-tSNE [19], for semantic retrieval tasks[4]. We also compared LLH to Product Quantization with Random Rotation (PQ-RR) and ℓ_2-scan for approximate nearest neighbor search [9]. Further, we evaluate LLH using W trained by Eq. (1) without the sparse term, as in LLE [17]. We denote it as LLH^0.

We run previous methods using publicly available Matlab codes provided by the authors with suggested parameters (if given) in their papers. For AGH and IMH-tSNE, the numbers of anchor points and neighbor anchor points are selected from $[100, 500]$ and $\{2, 3, 5, 10, 20\}$ by cross-validation. For PQ-RR, the number of sub-spaces is set as the best in $\{4, 8\}$. For our LLH, we set the number of K-means clusters no larger than 256, and select λ and η from $\{0.01, 0.05, 0.1, 0.5, 1.0\}$. We set $m = 100$K for datasets larger than 100K, and $m = n$ otherwise.

Following evaluation protocols used in previous hashing methods (e.g., [5, 24]), we split each dataset into training and query sets. The training set is used to train binary codes and construct the database. We evaluate *Hamming ranking* performance, i.e., retrieved images are sorted according to their Hamming distances to the query. This is exhaustive but is known to be fast enough in practice (228 times faster than l^2-scan in our experiments, see Sec. 3.5). Following [5], we measure the retrieval performance using the averaged precision of first 500 ranked images for each query[5].



(a) (b)

Figure 8. Results on **Yale**; (a) Averaged precision vs. #bits; (b) Precision vs. # of top retrieved images using 64-bit codes.

3.1. Results on Synthetic Datasets

To illustrate the basic behavior of our LLH, we first report results on two synthetic datasets, Two-TrefoilKnots (Fig. 6(a)) and Two-SwissRolls (Fig. 7(a)), each of which consists of 4000 points, with 2000 in each manifold. In Two-TrefoilKnots, the two manifolds are partially overlapped. For 15% of the data points, their 20 nearest neighbors contain points from the other manifold. Two-SwissRolls is even more challenging with the two manifolds stuck together. For each dataset, we first generate 3-d points, and then embed them into $\mathbb{R}^{100}$ by adding 97-d small uniform noise. We randomly sample 1000 points as queries and use the remaining 3000 points for training.

Fig. 6(b) shows the results on Two-TrefoilKnots, where our LLH and LLH^0 clearly outperform others and achieve almost 100% precision for all the code lengths. The results on Two-SwissRolls are shown in Fig. 7(b). In this dataset, with high probability, the ambient neighborhood contains points from different manifolds, so the performances of previous methods are severely degraded. In sharp contrast, our LLH shows great performance and is significantly superior to others. While LLH^0 also outperforms others by a large margin, it is much poorer than LLH. This suggests the effectiveness of LLSR in capturing the local manifold structures, and the effectiveness of our LLBE in preserving these structures in the Hamming space.

3.2. Results on Face Images

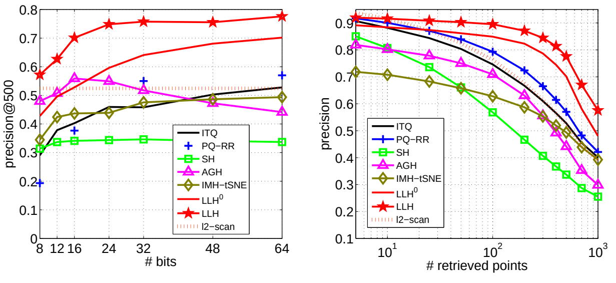
We next evaluate our LLH on the Yale[6] face image dataset, which is popular for manifold learning [27, 3]. Yale contains 2414 grayscale images of 38 subjects under 64 different illumination conditions. The number of images of each subject ranges from 59 to 64. We resize all the images to 36×30 and use their pixel values as feature vectors (thus $D = 1080$). We randomly sample 200 query images and use the remaining as the training set.

Fig. 8(a) shows the average precision on the top 10 retrieved images at code lengths from 8-bit to 64-bit. Our LLH is consistently better than all the other methods at each code length, and LLH^0 is the second best. The gain of LLH is huge, ranging from 199% to 627% over SH which

[4] Due to space limit, we put additional experimental results and further comparisons against other methods including KMH [6], MDSH [23], SPH [7], and KLSH [11] in the supplementary material.

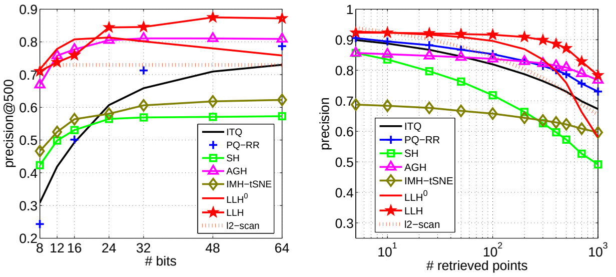
[5] The only exception is Yale which only has 50+ images in each class. We assign random rank to images having the same Hamming distance.

[6] http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html



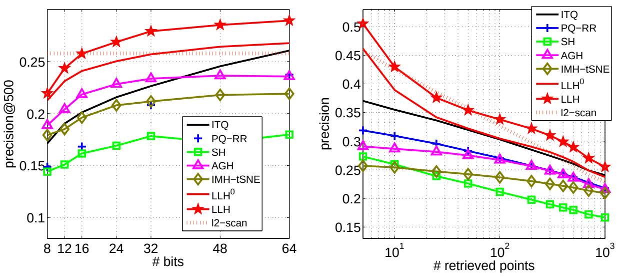
(a) (b)

Figure 9. Results on **USPS**; (a) Averaged precision vs. #bits; (b) Precision vs. # of top retrieved images using 64-bit codes.



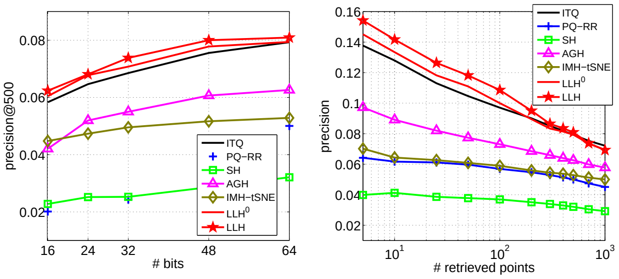
(a) (b)

Figure 10. Results on **MNIST**; (a) Averaged precision vs. #bits; (b) Precision vs. # of top retrieved images using 64-bit codes.



(a) (b)

Figure 11. Results on **CIFAR**; (a) Averaged precision vs. #bits; (b) Precision vs. # of top retrieved images using 64-bit codes.



(a) (b)

Figure 12. Results on **ImageNet-200K**; (a) Averaged precision vs. #bits; (b) Precision vs. # of top retrieved images using 64-bit codes.

is the best competitive one. These results demonstrate the remarkable ability of LLH in extracting non-linear image manifolds. Notably, our LLH with only 8-bit codes already achieves higher accuracy than exhaustive l^2-scan, and the gain gets larger with more bits (from 24% to 114%). In contrast, AGH, ITQ, IMH-tSNE, and PQ-RR with 64-bit codes remain inferior to l^2-scan. Fig. 8(b) shows the results with 64-bit codes on varied retrieval points. Again, LLH outperforms all the other methods in all cases by a large margin. The retrieval results can be visualized in Fig. 2.

3.3. Results on Handwritten Digits

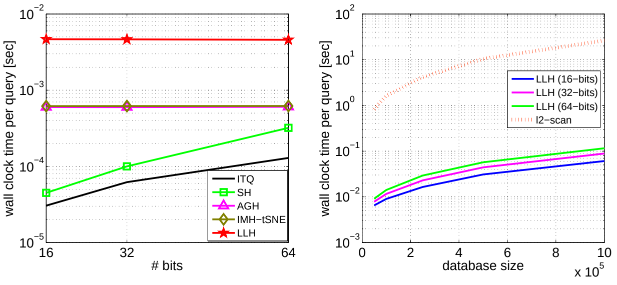
We next test on two popular handwritten digits datasets, USPS[7] and MNIST[8], which are frequently used to evaluate hashing methods [13, 14]. USPS contains $11K$ images of digits from "0" to "9" and MNIST has $70K$. On USPS, we randomly sample 1K queries and use the rest as the training set. For MNIST, we use the 10K test images as the query set and the other 60K as the training set. Each image in USPS has 16×16 pixels, and each has 28×28 pixels in MNIST, corresponding to $D = 256$ in USPS and $D = 784$ in MNIST.

Fig. 9(a) and Fig. 10(a) show the average precision for varied code lengths on USPS and MNIST, respectively. Our LLH is the best in most cases, and significantly better than l^2-scan (44% gain on USPS and 20% on MNIST with 64-bit codes). AGH is the best competitive method on these two datasets. Though AGH is slightly better than LLH on MNIST when using very short codes (12- or 16-bit), it is significantly worse with long codes. The performance of AGH on USPS even drops with long codes (48- or 64-bit) and is inferior to the other methods, while our LLH performs best at all code lengths.

3.4. Results on Natural Images

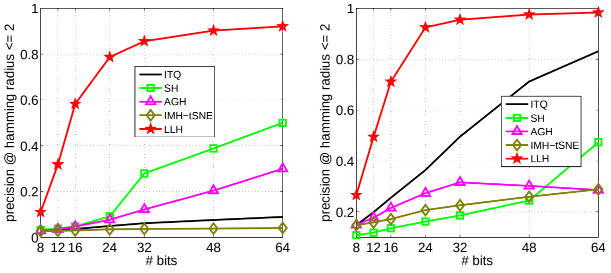
Lastly, we report results on three natural image datasets, CIFAR[9], ImageNet-200K[10], and MIRFLICKR-1M[11]. CIFAR contains 60K tiny images of 10 object classes. We randomly sample 2K query images and use the rest for the training set. Every image is represented by a 512-d GIST feature vector. ImageNet-200K is a 200K subest of the entire ImageNet dataset. We select the top 100 frequent classes (synsets) with total 191050 images of various kinds of objects such as animals, artifacts, and geological formation. We randomly sampled 20 images from each class to form the query set, which gives us 189050 training images and 2K query images. Following [9], each image is represented by a 512-d VLAD. We first extract a 4096-d VLAD and then use PCA to reduce the dimensionality to 512. MIRFLICKR-1M contains 1M images collected from Flickr, without ground-truth semantic labels. We use this dataset to evaluate qualitative results and computation time.

[7] http://www.cs.nyu.edu/~roweis/data/usps_all.mat
[8] http://yann.lecun.com/exdb/mnist/
[9] http://www.cs.toronto.edu/~kriz/cifar.html
[10] http://www.image-net.org/
[11] http://press.liacs.nl/mirflickr/



(a) (b)

Figure 14. Computation time on **MIRFLICKR-1M**; (a) Averaged binary code generation time per query vs. #bits; (b) Averaged query time vs. the database size (from 50K to 1M).



(a) (b)

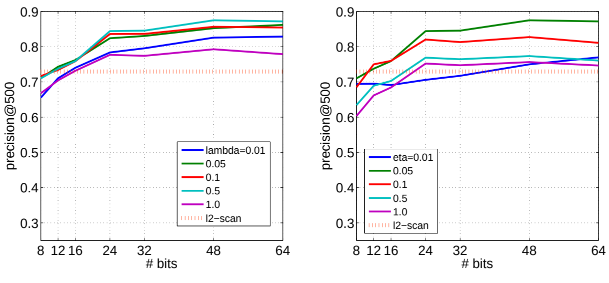
Figure 15. Averaged precision with the hash lookup protocol vs. #bits. (a) **Yale** and (b) **CIFAR**.

Here each image is represented by a 960-d GIST color feature vector.

Fig. 11 shows the results on CIFAR, where our LLH consistently outperforms all the other methods in all cases. The second best is ITQ or AGH. Fig. 12 shows the results on ImageNet-200K. Again, our LLH is the best, and LLH^0 and ITQ follow. With 64-bit codes, ITQ is highly competitive with LLH, but LLH yields higher precision values when the number of retrieved images is less than or equal to 500 (Fig. 12(b)). Fig. 13 shows some qualitative results on MIRFLICKR-1M, where LLH consistently retrieves more semantically similar images than the other methods.

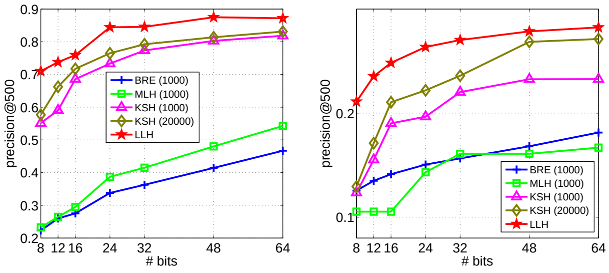
3.5. Analysis & Discussions

Computation Time. Fig. 14(a) shows the binary code generation time for an out-of-sample query, and Fig. 14(b) shows the entire query time (i.e., binary code generation time + Hamming ranking time) on MIRFLICKR-1M dataset. All results are obtained using MATLAB on a workstation with 2.53 GHz Intel Xeon CPU and 64GB RAM. Although our LLH is somewhat slower compared to the other methods, it is still fast enough in practice, with less than 5 msec per query (Fig. 14(a)), and orders of magnitude faster than l^2-scan (Fig. 14(b)). For example, it takes LLH only 0.12 sec to perform querying on one million database using 64-bit codes, which is about 228 times faster than l^2-scan (26.3 sec). In our experiments, we also find that the binary code generation time is independent of n and not sensitive to K (see Fig. 17 in supplementary material).



(a) (b)

Figure 16. Parameter sensitivity on **MNIST**. (a) λ and (b) η.



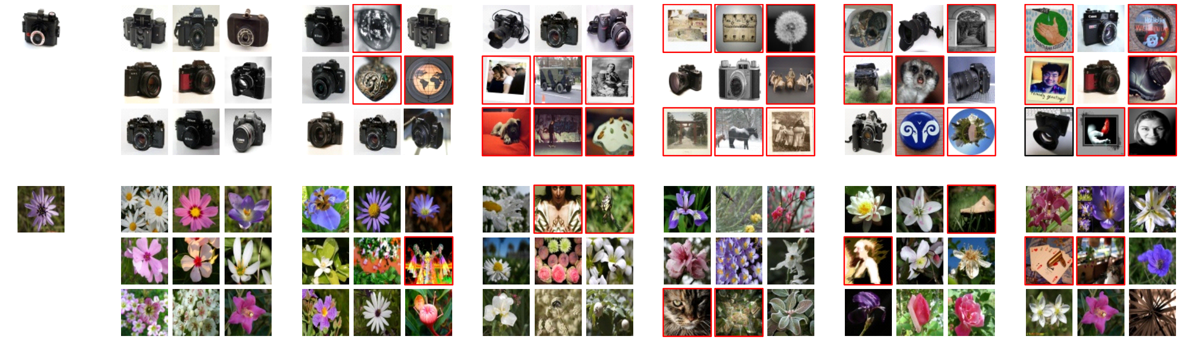
(a) (b)

Figure 17. LLH vs. supervised methods. Results on (a) **MNIST** and (b) **CIFAR**. The number within parentheses indicates # of semantic labels used to train binary codes in supervised methods.

Lookup Search. We have evaluated retrieval performances using *Hamming ranking*. Here we report results using *hash lookup*, which implements a hash lookup table and runs queries with it (e.g, [24, 13]). Fig. 15 shows the results on Yale and CIFAR. We can see that LLH still outperforms others by a very large margin on both datasets, which further confirms the remarkable effectiveness of LLH.

Parameter Study. Now we evaluate the performance of LLH with different settings of λ and η. The results on MNIST are shown in Fig. 16. While their performances are slightly different, the retrieval accuracies are still much better than l^2-scan in most cases. Also, LLH is quite stable w.r.t. K (see Fig. 17 in supplementary material). We remark that similar results are observed on other datasets.

Comparison with Supervised Methods. While ours LLH is unsupervised, there are quite a few supervised hashing methods like Binary Reconstructive Embedding (BRE) [10], Minimal Loss Hashing (MLH) [15], and Kernel-based Supervised Hashing (KSH) [12]. It is interesting to see how our unsupervised LLH compared to those supervised methods. The results on two datasets (MNIST and CIFAR) are shown in Fig. 17, where our LLH still performs best on both datasets. Among the three supervised methods, KSH is significantly better than the other two. However, even we increase the number of supervision labels for KSH from 1000 to 20000, KSH still cannot beat LLH at a single code length. These results indicate that preserving locally linear structures is highly effective for semantic retrieval, and our LLH successfully retains those structures in the Hamming space.



(a) query (b) LLH (c) ITQ (d) PQ-RR (e) SH (f) AGH (g) IMH-tSNE

Figure 13. Top retrieved images on **MIRFLICKR-1M** using 64-bit codes. Red border denotes false positive.

4. Conclusion

Finding items semantically similar to the query requires extracting its nearest neighbors lying on the same manifold, which is vital to retrieval but is a non-trivial problem. Our new hashing method, Locally Linear Hashing (LLH), is designed to tackle this problem through faithfully preserving the locally linear manifold structures of high-dimensional data in a low-dimensional Hamming space.

The locally linear manifold structures are first captured using locality-sensitive sparse coding, and then recovered in a low-dimensional Hamming space via a joint minimization of the embedding error and the quantization loss. A non-parametric method tailored to LLH is developed for out-of-sample extension.

LLH outperforms state-of-the-art hashing methods with very large performance gains on various types of visual benchmarks, demonstrating its remarkable effectiveness, scalability, and efficiency for large-scale retrieval.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. 4

[2] D. Donoho and Y. Tsaig. Fast solution of l_1-norm minimization problems when the solution may be sparse. *IEEE Trans. IT*, 54(11):4789–4812, 2008. 3, 4

[3] E. Elhamifar and R. Vidal. Sparse manifold clustering and embedding. In *NIPS*, 2011. 3, 5

[4] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012. 3

[5] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. PAMI*, 35(12):2916–2929, 2013. 1, 3, 4, 5

[6] K. He, F. Wen, and J. Sun. K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In *CVPR*, 2013. 1, 5

[7] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *CVPR*, 2012. 1, 5

[8] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, 1998. 1

[9] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. PAMI*, 33(1):117–128, 2011. 5, 6

[10] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009. 1, 7

[11] B. Kulis, P. Jain, and K. Grauman. Kernelized locality-sensitive hashing. *IEEE Trans. PAMI*, 34(6):1092–1104, 2012. 1, 5

[12] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, 2012. 1, 7

[13] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, 2011. 1, 2, 4, 5, 6, 7

[14] Y. Mu and S. Yan. Non-metric locality-sensitive hashing. In *AAAI*, 2010. 6

[15] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, 2011. 1, 7

[16] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000. 1

[17] L. Saul and S. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *JMLR*, 4:119–155, 2003. 1, 3, 5

[18] T. Sellis, N. Roussopoulos, and C. Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. *VLDB endowments*, 1987. 1

[19] F. Shen, C. Shen, Q. Shi, A. van den Hengel, and Z. Tang. Inductive hashing on manifolds. In *CVPR*, 2013. 1, 2, 4, 5

[20] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. PAMI*, 34:2393–2406, 2012. 1, 3

[21] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 3

[22] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *IJCV*, 70(1):77–90, 2006. 1

[23] Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *ECCV*, 2012. 3, 5

[24] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008. 1, 2, 4, 5, 7

[25] S. Yu and J. Shi. Multiclass spectral clustering. In *ICCV*, 2003. 3

[26] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *SIGIR*, 2010. 4

[27] Z. Zhang, J. Wang, and H. Zha. Adaptive manifold learning. *IEEE Trans. PAMI*, 34(2):253–265, 2012. 5