Bilinear Random Projections for Locality-Sensitive Binary Codes

Saehoon Kim and Seungjin Choi
Department of Computer Science and Engineering
Pohang University of Science and Technology, Korea
{kshkawa,seungjin}@postech.ac.kr

Abstract

Locality-sensitive hashing (LSH) is a popular data-independent indexing method for approximate similarity search, where random projections followed by quantization hash the points from the database so as to ensure that the probability of collision is much higher for objects that are close to each other than for those that are far apart. Most of high-dimensional visual descriptors for images exhibit a natural matrix structure. When visual descriptors are represented by high-dimensional feature vectors and long binary codes are assigned, a random projection matrix requires expensive complexities in both space and time. In this paper we analyze a bilinear random projection method where feature matrices are transformed to binary codes by two smaller random projection matrices. We base our theoretical analysis on extending Raginsky and Lazebnik's result where random Fourier features are composed with random binary quantizers to form locality sensitive binary codes. To this end, we answer the following two questions: (1) whether a bilinear random projection also yields similarity-preserving binary codes; (2) whether a bilinear random projection yields performance gain or loss, compared to a large linear projection. Regarding the first question, we present upper and lower bounds on the expected Hamming distance between binary codes produced by bilinear random projections. In regards to the second question, we analyze the upper and lower bounds on covariance between two bits of binary codes, showing that the correlation between two bits is small. Numerical experiments on MNIST and Flickr45K datasets confirm the validity of our method.

1. Introduction

Nearest neighbor search, the goal of which is to find most relevant items to a query given a pre-defined distance metric, is a core problem in various applications such as classification [14], object matching [9], retrieval [10], and so on. A naive solution to nearest neighbor search is linear scan where all items in database are sorted according to their similarity to the query, in order to find relevant items, requiring linear complexity. In practical applications, however, linear scan is not scalable due to the size of examples in database. Approximate nearest neighbor search, which trades accuracy for scalability, becomes more important than ever. Earlier work [4, 1] is a tree-based approach that exploits spatial partitions of data space via various tree structures to speed up search. While tree-based methods are successful for low-dimensional data, their performance is not satisfactory for high-dimensional data and does not guarantee faster search compared to linear scan [5].

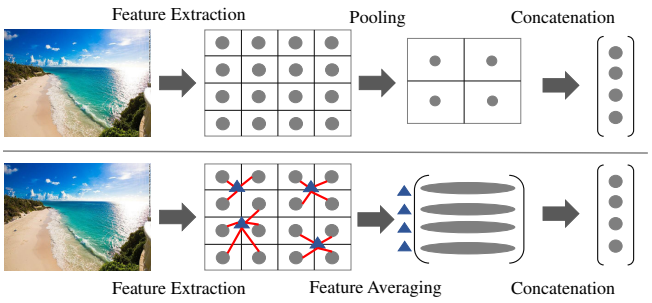


Figure 1. Two exemplary visual descriptors, which have a natural matrix structure, are often converted to long vectors. The above illustration describes LLC [15], where spatial information of initial descriptors is summarized into a final concatenated feature with spatial pyramid structure. The bottom shows VLAD [7], where the residual between initial descriptors and their nearest visual vocabulary (marked as a triangle) is encoded in a matrix form.

For high-dimensional data, a promising approach is approximate similarity search via hashing. Locality-sensitive hashing (LSH) is a notable data-independent hashing method, where randomly generates binary codes such that two similar items in database are hashed to have high probability of collision [5, 2, 11]. Different similarity metric leads to various LSH, including angle preservation [2], ℓ_p norm ($p \in (0, 2]$) [3], and shift-invariant kernels [11]. Since LSH is a pure data-independent approach, it

needs multiple hash tables or long code, requiring high memory footprint. To remedy high memory consumption, data-dependent hashing [13, 17, 16] has been introduced to learn similarity-preserving binary codes from data such that embedding into a binary space preserves similarity between data points in the original space. In general, data-dependent hashing generates compact binary codes, compared to LSH. However, LSH still works well, compared to data-dependent hashing methods for a very large code size.

Most of existing hashing algorithms does not take into account a natural matrix structure frequently observed in image descriptors [7, 15], as shown in Fig. 1. When a matrix descriptor is re-organized as a high-dimensional vector, most of hashing methods suffer from high storage and time complexity due to a single large projection matrix. Given a d-dimensional vector, the space and time complexities to generate a code of size k are both $O(dk)$. In the case of $100,000$-dimensional data, 40GB[1] is required to store a projection matrix to generate a binary code of length $100,000$, which is not desirable in constructing a large-scale vision system.

Bilinear projection, which consists of left and right projections, is a promising approach to handling data with a matrix structure. It has been successfully applied to two-dimensional principal component analysis (2D-PCA) [18] and 2D canonical correlation analysis (2D-CCA)[8], demonstrating that the time and space complexities are reduced while retaining performance, compared to a single large projection method. Recently, bilinear projections are adopted to the angle-preserving LSH [6], where the space and time complexities are $O(\sqrt{dk})$ and $O(d\sqrt{k})$, to generate binary codes of size k for a $\sqrt{d}$ by $\sqrt{d}$ matrix data. Note that when such matrix data is re-organized as d-dimensional vector, the space and time complexities for LSH are both $O(dk)$. While promising results for hashing with bilinear projection are reported in [6], its theoretical analysis is not available yet.

In this paper we present a bilinear extension of LSH from shift-invariant kernels (LSH-SIK) [11] and attempt to the following two questions on whether:

- randomized bilinear projections also yield similarity-preserving binary codes;
- there is performance gain or loss when randomized bilinear projections are adopted instead of a large single linear projection.

Our analysis shows that LSH-SIK with bilinear projections generates similarity-preserving binary codes and that the performance is not much degraded compared to LSH-SIK with a large single linear projection.

[1]If we use single precision to represent a floating-point, the projection matrix needs $100,000 \times 100,000 \times 4$ bytes $\approx$ 40GB.

2. Related Work

In this section we briefly review LSH algorithms for preserving angle [2] or shift-invariant kernels [11]. We also review an existing bilinear hashing method [6].

2.1. LSH: Angle Preservation

Given a vector $\boldsymbol{x}$ in $\mathbb{R}^d$, a hash function $h_a(\boldsymbol{x})$ returns a value 1 or 0, i.e., $h_a(\cdot) : \mathbb{R}^d \mapsto \{0, 1\}$. We assume that data vectors are centered, i.e., $\sum_{i=1}^{N} \boldsymbol{x}_i = 0$, where N is the number of samples. The random hyperplane-based hash function involves a random projection followed by a binary quantization, taking the form:

$$h_a(\boldsymbol{x}) \triangleq \frac{1}{2}\Big\{1 + \text{sgn}\left(\boldsymbol{w}^\top \boldsymbol{x}\right)\Big\}, \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is a random vector sampled on a unit d-sphere and $\text{sgn}(\cdot)$ is the sign function which returns 1 whenever the input is nonnegative and 0 otherwise. It was shown in [2] that the random hyperplane method naturally gives a family of hash functions for vectors in $\mathbb{R}^d$ such that

$$\mathbb{P}\Big[h_a(\boldsymbol{x}) = h_a(\boldsymbol{y})\Big] \;\; = \;\; 1 - \frac{\theta_{\boldsymbol{x},\boldsymbol{y}}}{\pi}, \tag{2}$$

where $\theta_{\boldsymbol{x},\boldsymbol{y}}$ denotes the angle between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. This technique, referred to as LSH-angle, works well for preserving an angle, but it does not preserve other types of similarities defined by a kernel between two vectors.

2.2. LSH: Shift-Invariant Kernels

Locality-sensitive hashing from shift-invariant kernels [11], referred to as LSH-SIK, is a random projection-based encoding scheme, such that the expected Hamming distance between the binary codes of two vectors is related to the value of shift-invariant kernel between two vectors. Random Fourier feature (RFF) is defined by

$$\phi_w(\boldsymbol{x}) \triangleq \sqrt{2}\cos(\boldsymbol{w}^\top \boldsymbol{x} + b), \tag{3}$$

where $\boldsymbol{w}$ is drawn from a distribution corresponding to an underlying shift-invariant kernel, i.e., $\boldsymbol{w} \sim p_\kappa$, b is drawn from a uniform distribution over $[0, 2\pi]$, i.e., $b \sim \text{Unif}[0, 2\pi]$. If the kernel κ is properly scaled, Bochner's theorem guarantees $\mathbb{E}_{w,b}\big[\phi_w(\boldsymbol{x})\phi_w(\boldsymbol{y})\big] = \kappa(\boldsymbol{x}-\boldsymbol{y})$, where $\mathbb{E}$ is the statistical expectation and $\kappa(\cdot)$ represents a shift-invariant kernel [12].

LSH-SIK builds a hash function, $h(\cdot) : \mathbb{R}^d \mapsto \{0, 1\}$, composing RFFs with a random binary quantization

$$h(\boldsymbol{x}) \triangleq \frac{1}{2}\left\{1 + \text{sgn}\left(\frac{1}{\sqrt{2}}\phi_w(\boldsymbol{x}) + t\right)\right\}, \tag{4}$$

where $t \sim \text{Unif}[-1, 1]$. The most appealing property of LSH-SIK provides upper- and lower-bounds on the expected Hamming distance between any two embedded points, which is summarized in Theorem 1.

Theorem 1. *[11] Define the functions*

$$g_1(\zeta) \triangleq \frac{4}{\pi^2}(1-\zeta)$$
$$g_2(\zeta) \triangleq \min\left\{\frac{1}{2}\sqrt{1-\zeta}, \frac{4}{\pi^2}\left(1-\frac{2}{3}\zeta\right)\right\},$$

where $\zeta \in [0,1]$, *and* $g_1(0) = g_2(0) = \frac{4}{\pi^2}$, $g_1(1) = g_2(1) = 0$. *Mercer kernel* κ *is shift-invariant, normalized, and satisfies* $\kappa(\alpha\boldsymbol{x} - \alpha\boldsymbol{y}) \leq \kappa(\boldsymbol{x}-\boldsymbol{y})$ *for any* $\alpha \geq 1$. *Then the expected Hamming distance between any two embedded points satisfies*

$$g_1(\kappa(\boldsymbol{x}-\boldsymbol{y})) \leq \mathbb{E}\Big[\mathcal{I}\left[h(\boldsymbol{x}) \neq h(\boldsymbol{y})\right]\Big] \leq g_2(\kappa(\boldsymbol{x}-\boldsymbol{y})), \quad (5)$$

where $\mathcal{I}[\cdot]$ *is the indicator function which equals 1 if its argument is true and 0 otherwise.*

The bounds in Theorem 1 indicate that binary codes determined by LSH-SIK well preserve the similarity defined by the underlying shift-invariant kernel.

2.3. Hashing with Bilinear Projections

Most of high-dimensional descriptors for image, including HOG, Fisher Vector (FV), and VLAD, exhibit a natural matrix structure. Suppose that $\boldsymbol{X} \in \mathbb{R}^{d_w \times d_v}$ is a descriptor matrix. The matrix is reorganized into a vector $\boldsymbol{x} = \text{vec}(\boldsymbol{X}) \in \mathbb{R}^d$ where $d = d_w d_v$, and then a binary code of size k is determined by k independent use of the hash function (4). This scheme requires $O(dk)$ in space and time.

A bilinear projection-based method [6] constructs a hash function $H_a(\cdot) : \mathbb{R}^{d_w \times d_v} \mapsto \{0,1\}^{k_w k_v}$ that is of the form

$$H_a(\boldsymbol{X}) \triangleq \frac{1}{2}\left\{1 + \text{sgn}\Big(\text{vec}\left(\boldsymbol{W}^\top \boldsymbol{X}\boldsymbol{V}\right)\Big)\right\}, \quad (6)$$

where $\boldsymbol{W} \in \mathbb{R}^{d_w \times k_w}$ and $\boldsymbol{V} \in \mathbb{R}^{d_v \times k_v}$, to produce a binary code of size $k = k_w k_v$. This scheme reduces space and time complexity to $O(d_w k_w + d_v k_v)$ and $O(d_w^2 k_w + d_v^2 k_v)$, respectively, while a single large linear projection requires $O(d_w d_v k_w k_v)$ in space and time.[2] Empirical results in [6] indicate that a random bilinear projection produces comparable performance compared to a single large projection. However, its theoretical behavior is not fully investigated. In the next section, we consider a bilinear extension of LSH-SIK (4) and present our theoretical analysis.

[2]Recently, [19] proposes a circulant embedding, which is implemented by discrete Fourier transform, to reduce the space and time complexities to $O(d)$ and $O(d \log d)$ when $(d = d_w \times d_v)$ and the code length is d. Even though the circulant embedding is faster than bilinear projections, we believe that it is worth analyzing hashing with bilinear projections, because the implementation is simpler than [19].

3. Analysis of Bilinear Random Projections

In this section we present the main contribution that is an theoretical analysis of a bilinear extension of LSH-SIK. To this end, we consider a hash function $h(\cdot) : \mathbb{R}^{d_w \times d_v} \mapsto \{0,1\}$ that is of the form

$$h(\boldsymbol{X}) \triangleq \frac{1}{2}\Big\{1 + \text{sgn}\Big(\cos\left(\boldsymbol{w}^\top \boldsymbol{X}\boldsymbol{v} + b\right) + t\Big)\Big\}, \quad (7)$$

where $\boldsymbol{w}, \boldsymbol{v} \sim \mathcal{N}(0, \boldsymbol{I})$, $b \sim \text{Unif}[0, 2\pi]$, and $t \sim \text{Unif}[-1, 1]$. With the abuse of notation, we use $h(\cdot)$ for the case of randomized bilinear hashing, however, it can be distinguished from (4), depending on its input argument $\boldsymbol{x}$ or $\boldsymbol{X}$. To produce binary code of size $k = k_w k_v$, the hash function $H(\cdot) : \mathbb{R}^{d_w \times d_v} \mapsto \{0,1\}^k$ takes the form:

$$H(\boldsymbol{X}) \triangleq \frac{1}{2}\Big\{1 + \text{sgn}\Big(\cos\left(\text{vec}(\boldsymbol{W}^\top \boldsymbol{X}\boldsymbol{V}) + \boldsymbol{b}\right) + \boldsymbol{t}\Big)\Big\}, \quad (8)$$

where each column of $\boldsymbol{W}$ or of $\boldsymbol{V}$ is independently drawn from spherical Gaussian with zero mean and unit variance, each entry of $\boldsymbol{b} \in \mathbb{R}^k$ or of $\boldsymbol{t} \in \mathbb{R}^k$ is drawn uniformly from $[0, 2\pi]$ and $[-1, 1]$, respectively.

We attempt to answer two questions on whether: (1) bilinear random projections also yield similarity-preserving binary codes like the original LSH-SIK; (2) there is performance gain or degradation when bilinear random projections are adopted instead of a large linear projection.

To answer the first question, we compute the upper and lower bound on the expected Hamming distance $\mathbb{E}\Big[\mathcal{I}\big[h(\boldsymbol{X}) \neq h(\boldsymbol{Y})\big]\Big]$ between any two embedded points computed by bilinear LSH-SIK with Gaussian kernel. Compared to the the original upper and lower bounds for LSH-SIK [11] with a single linear projection (Theorem 1), our upper bound is the same and lower bound is slightly worse when the underlying kernel is Gaussian.

Regarding the second question, note that some of bits of binary codes computed by the hash function (8) share either left or right projection (column vector of $\boldsymbol{W}$ or $\boldsymbol{V}$), leading to correlations between two bits. We show that the covariance between two bits is not high by analyzing the upper and lower bounds on covariance between the two bits.

3.1. Random Fourier Features

We begin with investigating the properties of random Fourier features [12] in the case of bilinear projections, since BLSH-SIK (an abbreviation of bilinear LSH-SIK) bases its theoretical analysis on these properties. To this end, we consider bilinear RFF:

$$\phi_{w,v}(\boldsymbol{X}) \triangleq \sqrt{2}\cos(\boldsymbol{w}^\top \boldsymbol{X}\boldsymbol{v} + b), \quad (9)$$

where $\boldsymbol{w}, \boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $b \sim \text{Unif}[0, 2\pi]$.

In the case of randomized linear map where $\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{I})$, $\mathbb{E}[\phi_w(\boldsymbol{x})\phi_w(\boldsymbol{y})] = \kappa_g(\boldsymbol{x}-\boldsymbol{y})$, where $\kappa_g(\cdot)$ is

Gaussian kernel. Unfortunately, for the randomized bilinear map, $\mathbb{E}\big[\phi_{w,v}(\boldsymbol{X})\phi_{w,v}(\boldsymbol{Y})\big] \neq \kappa_g\big(\text{vec}(\boldsymbol{X}-\boldsymbol{Y})\big)$, where Gaussian kernel defined as

$$\kappa_g\big(\text{vec}(\boldsymbol{X}-\boldsymbol{Y})\big) \triangleq \exp\left\{-\frac{1}{2}\|\text{vec}(\boldsymbol{X}-\boldsymbol{Y})\|_2^2\right\} = \exp\left\{-\frac{1}{2}\text{tr}\Big[(\boldsymbol{X}-\boldsymbol{Y})(\boldsymbol{X}-\boldsymbol{Y})^\top\Big]\right\},$$

where $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{d_w \times d_v}$ and the scaling parameter of Gaussian kernel is set as 1. However, we show that $\mathbb{E}\big[\phi_{w,v}(\boldsymbol{X})\phi_{w,v}(\boldsymbol{Y})\big]$ is between $\kappa_g\big(\text{vec}(\boldsymbol{X}-\boldsymbol{Y})\big)$ and $\kappa_g\big(\text{vec}(\boldsymbol{X}-\boldsymbol{Y})\big)^{0.79}$, which is summarized in the following lemma.

Lemma 1. *Define $\Delta = (\boldsymbol{X}-\boldsymbol{Y})(\boldsymbol{X}-\boldsymbol{Y})^\top$. Denote by $\{\lambda_j\}$ leading eigenvalues of Δ. The inner product between RFFs is given by*

$$\mathbb{E}_{w,v,b}\big[\phi_{w,v}(\boldsymbol{X})\phi_{w,v}(\boldsymbol{Y})\big] = \prod_j (1+\lambda_j)^{-\frac{1}{2}} \triangleq \kappa_b(\boldsymbol{X}-\boldsymbol{Y}). \quad (10)$$

Then, $\kappa_b(\boldsymbol{X}-\boldsymbol{Y})$ is upper and lower bounded in terms of Gaussian kernel $\kappa_g(vec(\boldsymbol{X}-\boldsymbol{Y}))$:

$$\kappa_g\big(vec(\boldsymbol{X}-\boldsymbol{Y})\big) \leq \kappa_b\big(\boldsymbol{X}-\boldsymbol{Y}\big) \leq \kappa_g\big(vec(\boldsymbol{X}-\boldsymbol{Y})\big)^{0.79},$$

provided that the following assumptions are satisfied:

- $||\boldsymbol{X}||_F \leq 0.8$, *which can be easily satisfied by re-scaling the data.*
- $\lambda_1 \leq 0.28\sum_{i=2}^{d_w}\lambda_i$, *which can be easily satisfied for large d_w.*

Proof.

$$\begin{aligned}
&\mathbb{E}_{w,v,b}\Big[\phi_{w,v}(\boldsymbol{X})\phi_{w,v}(\boldsymbol{Y})\Big] \\
&= \int\int \cos\Big(\boldsymbol{w}^\top(\boldsymbol{X}-\boldsymbol{Y})\boldsymbol{v}\Big)p(\boldsymbol{w})p(\boldsymbol{v})\,d\boldsymbol{w}d\boldsymbol{v} \\
&= \int \kappa_g\Big((\boldsymbol{X}-\boldsymbol{Y})^\top\boldsymbol{w}\Big)p(\boldsymbol{w})\,d\boldsymbol{w} \\
&= (2\pi)^{-\frac{d_w}{2}}\int \exp\left\{-\frac{\boldsymbol{w}^\top(\boldsymbol{I}+\Delta)\boldsymbol{w}}{2}\right\}d\boldsymbol{w} \\
&= \big|\boldsymbol{I}+\Delta\big|^{-\frac{1}{2}},
\end{aligned}$$

where $|\cdot|$ denotes the determinant of a matrix. The eigen-decomposition of Δ is given by $\Delta = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, where $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$ are eigenvector and eigenvalue matrices, respectively. Then we have

$$\begin{aligned}
\big|\boldsymbol{I}+\Delta\big|^{-\frac{1}{2}} &= \big|\boldsymbol{U}(\boldsymbol{I}+\boldsymbol{\Lambda})\boldsymbol{U}^\top\big|^{-\frac{1}{2}} \\
&= \prod_j (1+\lambda_j)^{-\frac{1}{2}}.
\end{aligned}$$

Now we prove the following inequalities:

$$\kappa_g(\text{vec}(\boldsymbol{X}-\boldsymbol{Y})) \leq \kappa_b(\boldsymbol{X}-\boldsymbol{Y}) \leq \kappa_g(\text{vec}(\boldsymbol{X}-\boldsymbol{Y}))^{0.79}.$$

Lower bound: First, we can easily show the lower bound on $\kappa_b(\boldsymbol{X}-\boldsymbol{Y})$ with the following inequality: $\kappa_g(\text{vec}(\boldsymbol{X}-\boldsymbol{Y})) = \prod_j \exp(\lambda_j)^{-\frac{1}{2}} \leq \prod_j (1+\lambda_j)^{-\frac{1}{2}} \triangleq \kappa_b(\boldsymbol{X}-\boldsymbol{Y})$, because $1+\lambda_j \leq \exp(\lambda_j)$.
Upper bound: Second, assuming that $||\boldsymbol{X}||_F \leq 0.8$, we can derive the upper bound on $\kappa_b(\boldsymbol{X}-\boldsymbol{Y})$. Now, we can bound λ_j with the following logic.

$$\begin{aligned}
&\text{tr}\Big[(\boldsymbol{X}-\boldsymbol{Y})(\boldsymbol{X}-\boldsymbol{Y})^\top\Big] \leq (2*0.8)^2 = 2.56 \\
\Rightarrow\quad &\sum_k \lambda_k \leq 2.56 \\
\Rightarrow\quad &\lambda_1 \leq 0.56 \quad (\because \lambda_k \geq 0, \lambda_1 \leq 0.28\sum_{i=2}^{d_w}\lambda_i).
\end{aligned}$$

For $0 \leq \lambda_j \leq 0.56$, we know that $\exp(\lambda_j)^{0.79} \leq 1+\lambda_j$, leading to the upper bound on $\kappa_b(\boldsymbol{X}-\boldsymbol{Y})$, i.e., $\kappa_b(\boldsymbol{X}-\boldsymbol{Y}) \leq \kappa_g(\text{vec}(\boldsymbol{X}-\boldsymbol{Y}))^{0.79}$. □

Lemma 1 indicates that random Fourier features with bilinear projections are related to the one with single projection in case of Gaussian kernel. Due to this relation, we can conclude that random Fourier features with bilinear projections can generate similarity-preserving binary codes in the following section. Finally, we summarize some important properties of $\kappa_b(\boldsymbol{X}-\boldsymbol{Y})$, showing that $\kappa_b(\boldsymbol{X}-\boldsymbol{Y})$ shares the similar properties with $\kappa_g(\boldsymbol{X}-\boldsymbol{Y})$:

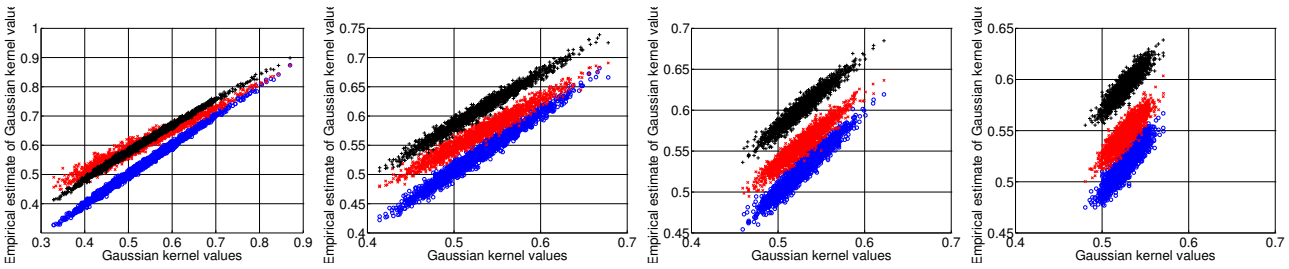
- **Property 1**: $0 \leq \kappa_b(\boldsymbol{X}-\boldsymbol{Y}) \leq 1$.
- **Property 2**: $\kappa_b(m\boldsymbol{X}-m\boldsymbol{Y}) \leq \kappa_b(\boldsymbol{X}-\boldsymbol{Y})$, where m is a positive integer.

Fig. 2 demonstrates that the inner product of two data points induced by bilinear RFF is upper and lower bounded with respect to Gaussian kernel as shown in Lemma 1. For the high d_w, the upper bound is satisfied in Fig. 2 (c-d), which is consistent with our intuition.

For Fig. 2, we generate the data from an uniform distribution with different dimensions, and re-scale the data to be $||\boldsymbol{X}||_F = 0.8$. To compute the estimates of bilinear RFF, we independently generate 10,000 triples $\{w_i, v_i, b_i\}$ and calculate the following sample average: $\sum_{i=1}^{k}\big[\phi_{w_i,v_i}(\boldsymbol{X})\phi_{w_i,v_i}(\boldsymbol{Y})\big]$, where $k = 10,000$. For the estimates of RFF, we calculate the sample average with 10,000 independently generated pairs $\{w_i, b_i\}$.

3.2. Bounds on Expected Hamming Distance

In this section, we derive the upper and lower bounds on the expected Hamming distance between binary codes computed by BLSH-SIK to show that BLSH-SIK can generate



(a) 4-by-4 matrix (b) 8-by-8 matrix (c) 15-by-15 matrix (d) 25-by-25 matrix

Figure 2. Estimates of bilinear RFF ($\kappa_b(\cdot)$) and its lower/upper bounds ($k_g(\cdot)$ and $k_g(\cdot)^{0.79}$) with respect to Gaussian kernel values. Red marks represent the inner products of two data points induced by bilinear RFF, and blue (black) marks represent its lower (upper) bounds.

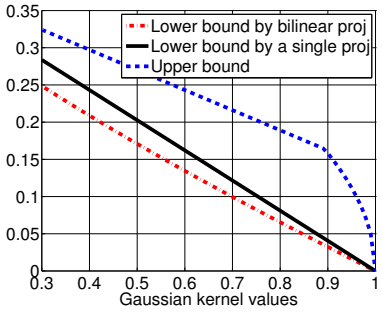


Figure 3. Upper and lower bounds on the expected Hamming distance between binary codes computed by BLSH-SIK and LSH-SIK.

similarity-preserving binary codes in the sense of Gaussian kernel. Lemma 2 is a slight modification of the expected Hamming distance by LSH-SIK with a single projection [11], indicating that the expected Hamming distance is analytically represented.

Lemma 2.

$$\mathbb{E}_{\boldsymbol{w},\boldsymbol{v},b,t}\Big[\mathcal{I}\big[h(\boldsymbol{X}) \neq h(\boldsymbol{Y})\big]\Big] = \frac{8}{\pi^2}\sum_{m=1}^{\infty}\frac{1-\kappa_b\big(m\boldsymbol{X}-m\boldsymbol{Y}\big)}{4m^2-1}.$$

Proof. This is a slight modification of the result (for a randomized linear map) in [11]. Since the proof is straightforward, it is placed in the supplementary material. □

Though the expected Hamming distance is analytically represented with respect to κ_b, its relationship with κ_g is not fully exploited. In order to figure out the similarity-preserving property of BLSH-SIK in a more clear way, Theorem 2 is described to show the upper and lower bounds on the expected Hamming distance for BLSH-SIK in terms of $\kappa_g\big(\mathrm{vec}(\boldsymbol{X}-\boldsymbol{Y})\big)$.

Theorem 2. *Define the functions*

$$g_1(\zeta) \triangleq \frac{4}{\pi^2}\left(1-\zeta^{0.79}\right),$$
$$g_2(\zeta) \triangleq \min\left\{\frac{1}{2}\sqrt{1-\zeta}, \frac{4}{\pi^2}\left(1-\frac{2}{3}\zeta\right)\right\},$$

where $\zeta \in [0,1]$ and $g_1(0) = g_2(0) = \frac{4}{\pi^2}$, $g_1(1) = g_2(1) = 0$. Gaussian kernel κ_g is shift-invariant, normalized, and satisfies $\kappa_g(\alpha\boldsymbol{x}-\alpha\boldsymbol{y}) \leq \kappa_g(\boldsymbol{x}-\boldsymbol{y})$ for any $\alpha \geq 1$. Then the expected Hamming distance between any two embedded points computed by bilinear LSH-SIK satisfies

$$g_1\Big(\kappa_g\big(\boldsymbol{\tau}\big)\Big) \leq \mathbb{E}\Big[\mathcal{I}\big[h(\boldsymbol{X}) \neq h(\boldsymbol{Y})\big]\Big] \leq g_2\Big(\kappa_g\big(\boldsymbol{\tau}\big)\Big), \quad (11)$$

where $\boldsymbol{\tau} = vec(\boldsymbol{X}-\boldsymbol{Y})$.

Proof. We prove the upper and lower bound one at a time, following the technique used in [11]. Note that the lower bound $g_1(\zeta)$ is slightly different from the one in Theorem 1, however the upper bound $g_2(\zeta)$ is the same as the one in Theorem 1.

Lower bound: It follows from Property 2 and Lemma 1 that we can easily find the lower bound as

$$\begin{aligned}\mathbb{E}\Big[\mathcal{I}\big[h(\boldsymbol{X}) \neq h(\boldsymbol{Y})\big]\Big] &\geq \frac{4}{\pi^2}\Big(1-\kappa_b\big(\mathrm{vec}(\boldsymbol{X}-\boldsymbol{Y})\big)\Big)\\ &\geq \frac{4}{\pi^2}\Big(1-\kappa_g\big(\mathrm{vec}(\boldsymbol{X}-\boldsymbol{Y})\big)^{0.79}\Big)\\ &\triangleq g_1\Big(\kappa_g\big(\mathrm{vec}(\boldsymbol{X}-\boldsymbol{Y})\big)\Big).\end{aligned}$$

Upper bound: By the proof of Lemma 2.3 [11], we can easily find the upper bound as

$$\mathbb{E}\Big[\mathcal{I}\big[h(\boldsymbol{X}) \neq h(\boldsymbol{Y})\big]\Big] \leq \min\left\{\frac{1}{2}\sqrt{1-\kappa_b\big(\boldsymbol{\tau}\big)}, \frac{4}{\pi^2}\left(1-\frac{2}{3}\kappa_b\big(\boldsymbol{\tau}\big)\right)\right\}.$$

Moreover, the inequality $\kappa_g\big(\boldsymbol{\tau}\big) \leq \kappa_b\big(\boldsymbol{\tau}\big)$ in Lemma 1

yields

$$\begin{aligned}\mathbb{E}\Big[\mathcal{I}\big[h(\boldsymbol{X}) \neq h(\boldsymbol{Y})\big]\Big] \\ \leq \quad & \min\left\{\frac{1}{2}\sqrt{1-\kappa_g(\boldsymbol{\tau})}, \frac{4}{\pi^2}\left(1-\frac{2}{3}\kappa_g(\boldsymbol{\tau})\right)\right\}, \\ \triangleq \quad & g_2\Big(\kappa_g(\boldsymbol{\tau})\Big).\end{aligned}$$

□

Theorem 2 shows that bilinear projections can generate similarity-preserving binary codes, where the expected Hamming distance is upper and lower bounded in terms of $\kappa_g\big(\text{vec}(\boldsymbol{X}-\boldsymbol{Y})\big)$. Compared with the original upper and lower bounds in case of a single projection shown in the Lemma 2.3 [11], we derive the same upper bound and slightly worse lower bound as depicted in Fig. 3.

3.3. Bounds on Covariance

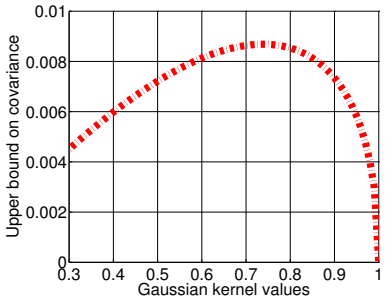


Figure 4. Upper bound on covariance between the two bits induced by BLSH-SIK. Horizontal axis suggests a Gaussian kernel value of two data points. Vertical axis shows an upper bound on covariance.

In this section, we analyze the covariance between two bits induced by BLSH-SIK to address how much the performance would be dropped compared with a single large projection matrix.

A hash function for multiple bits using bilinear projections (8) implies that there exists the bits which share one of the projection vectors. For example, assume that $h_i(\cdot)$ is given as

$$h_i(\boldsymbol{X}) = \text{sgn}\Big(\cos(\boldsymbol{w}_1^\top \boldsymbol{X}\boldsymbol{v}_i + b_i) + t_i\Big). \tag{12}$$

We can easily find the following $d_v - 1$ hash functions which shares $\boldsymbol{w}_1$ with $h_i(\cdot)$.

$$h_j(\boldsymbol{X}) = \text{sgn}\Big(\cos(\boldsymbol{w}_1^\top \boldsymbol{X}\boldsymbol{v}_j + b_j) + t_j\Big), \tag{13}$$

where $j \in \{1, \cdots, d_v\} \setminus \{i\}$.

If the two bits does not share any one of projection vectors, the bits should be independent which indicates a zero correlation. This phenomenon raises a natural question to ask that how much the two bits, which share one of projection vectors, are correlated. Intuitively, we expect that the highly correlated bits are not favorable, because such bits does contain redundant information to approximate $\mathbb{E}\Big[\mathcal{I}\big[h(\boldsymbol{X}) \neq h(\boldsymbol{Y})\big]\Big]$. Theorem 3 shows that the upper bound on covariance between two bits induced by bilinear projections is small, establishing the reason why BLSH-SIK performs well enough in case of a large number of bits.

Theorem 3. *Given the hash functions as Eq. (12-13), the upper bound on the covariance between the two bits is derived as*

$$\begin{aligned}cov(\cdot) \quad \leq \quad & \frac{64}{\pi^4}\left\{\left(\sum_{m=1}^{\infty}\frac{\kappa_g\big(vec(\boldsymbol{X}-\boldsymbol{Y})\big)^{0.79m^2}}{4m^2-1}\right)^2\right. \\ & \left. -\left(\sum_{m=1}^{\infty}\frac{\kappa_g\big(vec(\boldsymbol{X}-\boldsymbol{Y})\big)^{m^2}}{4m^2-1}\right)^2\right\},\end{aligned}$$

where $\kappa_g(\cdot)$ is the Gaussian kernel and $cov(\cdot)$ is the covariance between two bits defined as

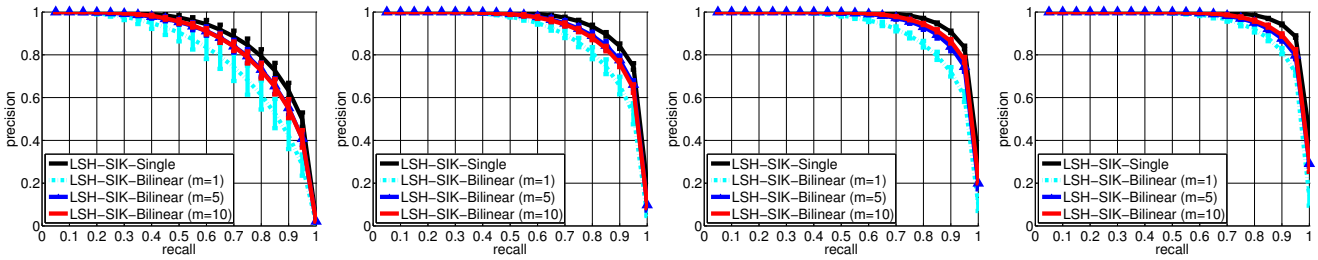
$$\begin{aligned}cov(\cdot) \quad = \quad & \mathbb{E}\Big[\mathcal{I}\big[h_i(\boldsymbol{X}) \neq h_i(\boldsymbol{Y})\big]\mathcal{I}\big[h_j(\boldsymbol{X}) \neq h_j(\boldsymbol{Y})\big]\Big] \\ - \quad & \mathbb{E}\Big[\mathcal{I}\big[h_i(\boldsymbol{X}) \neq h_i(\boldsymbol{Y})\big]\Big]\mathbb{E}\Big[\mathcal{I}\big[h_j(\boldsymbol{X}) \neq h_j(\boldsymbol{Y})\big]\Big].\end{aligned}$$

Proof. Since the proof is lengthy and tedious, the detailed proof and lower bound on the covariance can be found in the supplementary material. □

Fig. 4 depicts the upper bound on covariance between the two bits induced by BLSH-SIK with respect to Gaussian kernel value. We can easily see that the covariance between the two bits for the highly similar ($\kappa_g(\text{vec}(\boldsymbol{X}-\boldsymbol{Y})) \approx 1$) is nearly zero, indicating that there is no correlation between the two bits. Unfortunately, there exists unfavorable correlation for the data points which is not highly (dis)similar. To remedy such unfavorable correlation, a simple heuristic is proposed, in which $k \times m^2$ bits are first generated and randomly select the k bits when k is the desired number of bits and m is a free parameter for reducing the unfavorable correlation trading-off storage and computational costs. This simple heuristic reduces the correlation between the two bits without incurring too much computational and storage costs. Algorithm 1 summarizes the BLSH-SIK with the proposed heuristic.

4. Experiments

In this section, we represent the numerical experimental results to support the analysis presented in the previous sections, validating the practical usefulness of BLSH-SIK. For



(a) 400bits (b) 900bits (c) 1,600bits (d) 2,500bits

Figure 5. Precision-recall curves for LSH-SIK with a single projection (referred to as LSH-SIK-Single) and BLSH-SIK (referred to as LSH-SIK-Bilinear) on MNIST with respect to the different number of bits. In case of BLSH-SIK, the precision-recall curves are plotted for the different m, which is introduced to reduce the correlation in Algorithm 1.

Algorithm 1 LSH for Shift-invariant Kernels with Bilinear Projections (BLSH-SIK)

Input: A data point is $\boldsymbol{X} \in \mathbb{R}^{d_w \times d_v}$, k is the desired number of bits, m is the hyper-parameter to reduce the correlation, and I is a subset with k elements of $\{1, 2, \cdots, m^2 \times k\}$.

Output: A binary code of $\boldsymbol{X}$ with k bits.

1: $\boldsymbol{W} \in \mathbb{R}^{d_w \times m\sqrt{k}}$ and $\boldsymbol{V} \in \mathbb{R}^{d_v \times m\sqrt{k}}$ are element-wise drawn from the zero-mean Gaussian, $\mathcal{N}(0, 1)$.

2: $\boldsymbol{b} \in \mathbb{R}^{mk}$ and $\boldsymbol{t} \in \mathbb{R}^{mk}$ are element-wise drawn from uniform distributions, Unif$[0, 2\pi]$ and Unif$[-1, +1]$, respectively.

3: Generate a binary code whose the number of bit is $k \times m^2$: $\frac{1}{2}(1 + \text{sgn}(\cos(\text{vec}(\boldsymbol{W}^\top \boldsymbol{X} \boldsymbol{V}) + \boldsymbol{b}) + \boldsymbol{t}))$.

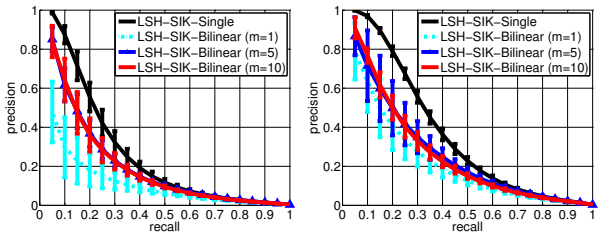
4: Select the k-bits from the binary code using the pre-defined subset I.

the numerical experiments, the two widely-used datasets, MNIST [3] and Flickr45K [4], are used to investigate the behaviors of BLSH-SIK from small- to high-dimensional data. MNIST consists of 70,000 handwritten digit images represented by a 28-by-28 matrix, where the raw images are used for the experiments. Flickr45K is constructed by randomly selecting 45,000 images from 1 million Flickr images used in [7]. VLAD [7] is used to represent an image with 500 cluster centers, resulting in a $500 \times 128 = 64,000$ dimensional vector normalized to the unit length with l_2 norm. For BLSH-SIK, we reshape an image into a 250-by-256 matrix.

The ground-truth neighbors should be carefully constructed for comparing the hashing algorithm in a fair manner. We adopt the same procedure to construct the ground-truth neighbors presented in [11]. First of all, we decide an appropriate threshold to judge which neighbors should be ground-truth neighbors, where the averaged Euclidean distance between the query and the 50th nearest neighbor

[3] http://yann.lecun.com/exdb/mnist/

[4] http://lear.inrialpes.fr/people/jegou/data.php

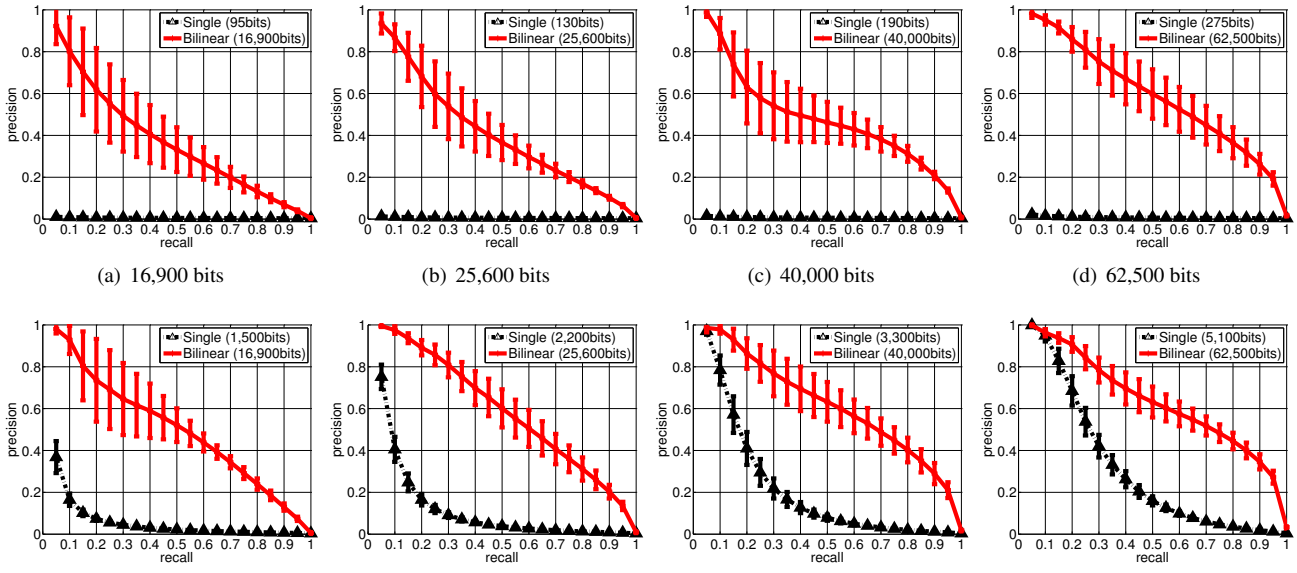


(a) 4,900bits (b) 6,400bits

Figure 6. Precision-recall curves for LSH-SIK with a single projection (referred to as LSH-SIK-Single) and BLSH-SIK (referred to as LSH-SIK-Bilinear) on Flickr45K with respect to the different number of bits, where the precision-recall curves for BLSH-SIK are plotted for the different hyper-parameter m.

is set to the appropriate threshold. Then, the ground-truth neighbor is decided if the distance between the query and the point is less than the threshold. Finally, we re-scale the dataset such that the threshold is one, leading that the scaling parameter for Gauassian kernel can be set to one. For both datasets, we randomly select 300 data points for queries, and the queries which has more than 5,000 ground-truth neighbors are excluded. To avoid any biased results, all precision-recall curves in this section are plotted by error bars with mean and one standard deviation over 5 times repetition.

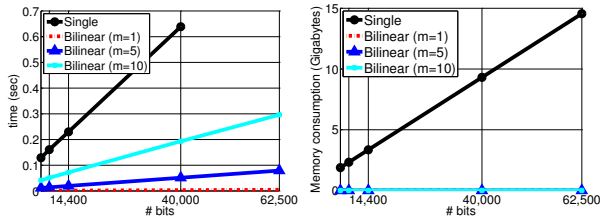
Fig. 5 and 6 represent precision-recall curves for LSH-SIK with a single projection and BLSH-SIK on MNIST and Flickr45K with respect to the different number of bits. In case of BLSH-SIK, the precision-recall curves are plotted for the different m, which is introduced to reduce the correlation in Algorithm 1. From the both figures, we observe that the larger m helps to reduce the correlation of the bits induced by BLSH-SIK. Even though BLSH-SIK cannot generate the same performance of LSH-SIK with a single projection, we argue that the performance is comparable. Moreover, the computational time and memory consumption for generating binary codes are significantly reduced as



(a) 16,900 bits (b) 25,600 bits (c) 40,000 bits (d) 62,500 bits

(e) 16,900 bits (f) 25,600 bits (g) 40,000 bits (h) 62,500 bits

Figure 8. Precision-recall curves for LSH-SIK with a single projection (referred to as Single) and BLSH-SIK (referred to as Bilinear) on Flickr45K when the same computational time for generating a binary code is required to LSH-SIK with a single projection and BLSH-SIK. The first (second) row shows the results when m of BLSH-SIK is one (five).



(a) Computational time (b) Memory consumption

Figure 7. Comparison between LSH-SIK with a single large projection (referred to as Single) and BLSH-SIK (referred to as Bilinear) in terms of the computational time and memory consumption on the Flickr45K dataset.

explained in the next paragraph.

Fig. 7 represents the comparison between LSH-SIK with a single large projection and BLSH-SIK in terms of the computational time [5] and memory consumption on the Flickr45K dataset. In case of BLSH-SIK, the time cost and memory consumption are reported with respect to the different m, which evidently shows that the computational time and memory consumption of BLSH-SIK are much smaller than LSH-SIK with a single projection. From Fig. 5, 6 and 7, we can conclude that $m = 5$ is a good choice for BLSH-SIK, because $m = 5$ performs well compared to $m = 10$ but it is much faster than $m = 10$.

Fig. 8 represents the precision-recall curves for LSH-SIK with a single projection and BLSH-SIK on Flickr45K with the same computational time limitation for generating a binary code. Therefore, fewer bits are used for LSH-SIK with a single projection compared to BLSH-SIK. For both $m = 1$ and $m = 5$, BLSH-SIK is superior to LSH-SIK with a single projection with the same computational time.

5. Conclusions

In this paper we have presented a bilinear extension of LSH-SIK [11], referred to as BLSH-SIK, where we proved that the expected Hamming distance between the binary codes of two vectors is related to the value of Gaussian kernel when column vectors of projection matrices are independently drawn from spherical Gaussian distribution. Our theoretical analysis have confirmed that: (1) randomized bilinear projection yields similarity-preserving binary codes; (2) the performance of BLSH-SIK is comparable to LSH-SIK, showing that the correlation between two bits of binary codes computed by BLSH-SIK is small. Numerical experiments on MNIST and Flickr45K datasets confirmed the validity of our method.

Acknowledgements: This work was supported by National Research Foundation (NRF) of Korea (NRF-2013R1A2A2A01067464) and the IT R&D Program of MSIP/IITP (B0101-15-0307, Machine Learning Center).

[5] To measure the computational time, a single thread is used with a Intel i7 3.60GHz machine (64GB main memory). Fig. 7 (a) does not include the computational time of LSH-SIK with a single projection for 62,500bits due to the high memory consumption.

References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45(6):891–923, 1998.

[2] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, 2002.

[3] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Annual ACM Symposium on Computational Geometry (SoCG)*, 2004.

[4] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Softwares*, 3(3):209–226, 1977.

[5] A. Gionis, P. Indyk, and R. Motawani. Similarity search in high dimensions via hashing. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1999.

[6] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Portland, Oregon, USA, 2013.

[7] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, 2010.

[8] S. H. Lee and S. Choi. Two-dimensional canonical correlation analysis. *IEEE Signal Processing Letters*, 14(10):753–758, 2007.

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[10] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, NY, USA, 2006.

[11] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22. MIT Press, 2009.

[12] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, 2008.

[13] R. Salakhutdinov and G. Hinton. Semantic hashing. In *Proceeding of the SIGIR Workshop on Information Retrieval and Applications of Graphical Models*, 2007.

[14] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, 2006.

[15] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, 2010.

[16] Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Firenze, Italy, 2012.

[17] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20. MIT Press, 2008.

[18] J. Yang, D. Zhang, A. F. Frangi, and J. Y. Yang. Two-dimensional PCA: A new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):131–137, 2004.

[19] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014.