SignRFF: Sign Random Fourier Features

Xiaoyun Li, Ping Li

LinkedIn Corporation
700 Bellevue WA NE, Bellevue, WA 98004, USA
{lixiaoyun996,pingli98}@gmail.com

Abstract

The method of Random Fourier Feature (RFF) has been popular for large-scale learning, which generates non-linear random features of the data. It has also been used to construct Locality-Sensitive Hashing (LSH) codes via stochastic quantization for efficient information retrieval. In this paper, we revisit binary hashing from RFF, and study SignRFF, a simple strategy to extract RFF-based binary codes. Particularly, we propose a new analytical measure called *ranking efficiency*, which provides a systematic and unified framework for comparing different LSH methods in practice. It suggests that the non-linear kernel based methods (e.g., SignRFF) should be preferred over the simple sign random projection method in high similarity region. Experiments are conducted to show that SignRFF is consistently better than the previous RFF-based method, and also outperforms other data-dependent and deep learning based hashing methods with sufficient number of hash bits. Moreover, the proposed ranking efficiency is a strong indicator of the empirical search performance.

1 Introduction

We study the search problem with data points in $\mathbb{R}^d$. Let $\boldsymbol{X}$ denote the database consisting of n data points. Given a query data point $\boldsymbol{q}$, the task is to find the a similar data point in $\boldsymbol{X}$. For the purpose of discussion, we denote two data points in $\boldsymbol{X}$ by $\boldsymbol{x}$ and $\boldsymbol{y}$. For simplicity, we consider the data points are normalized, i.e., $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = 1$. Therefore, the "cosine" similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$ is simply $\rho = \cos(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^d x_i y_i$. We also denote the Gaussian kernel between $\boldsymbol{x}$ and $\boldsymbol{y}$ to be $k(\rho) \equiv k(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma^2(1-\rho)}$, where $\gamma > 0$ is a tuning parameter. In this paper, we study binary coding algorithms for efficiently finding near neighbors based on similarities related to ρ and $k(\rho)$. In particular, our methods are based on random projections and (quantized) random Fourier features.

1.1 Random Projections (RP) and Sign Random Projections (SignRP)

Denote by $\boldsymbol{w}$ a d-dim vector of random i.i.d. entries, i.e., $w_i \sim N(0,1)$, $i = 1$ to d. The idea of random projections (RP) is to compute the inner product $\boldsymbol{w}^T\boldsymbol{x}$ between $\boldsymbol{w}$ and the data vector $\boldsymbol{x}$, and the same for $\boldsymbol{w}^T\boldsymbol{y}$. It is easy to see that $\mathbb{E}[(\boldsymbol{w}^T\boldsymbol{x})(\boldsymbol{w}^T\boldsymbol{y})] = \cos(\boldsymbol{x}, \boldsymbol{y}) = \rho$. The idea of sign random projections (SignRP) is to keep only the signs of the projected values:

$$\textbf{SignRP:} \quad h_{RP}(\boldsymbol{x}) = sign(\boldsymbol{w}^T\boldsymbol{x}), \qquad h_{RP}(\boldsymbol{y}) = sign(\boldsymbol{w}^T\boldsymbol{y}) \tag{1}$$

There is a chance that the two hash values, i.e., the two signs $h_{RP}(\boldsymbol{x})$ and $h_{RP}(\boldsymbol{y})$, will be equal [14]:

$$P(h_{RP}(\boldsymbol{x}) = h_{RP}(\boldsymbol{y})) = 1 - \frac{\cos^{-1}(\rho)}{\pi}. \tag{2}$$

Interestingly, if w_i is sampled from a Cauchy distribution (instead of Gaussian), then the collision probability is closely related to the popular Chi-square similarity instead the cosine [35, 29].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

1.2 Binary Code and Approximate Near Neighbor (ANN) Search

The method of sign random projections (SignRP) and the collision probability (2) have been widely used [4] as a standard hashing method for locallility sensitive hashing (LSH) in the context of approximate near neighbor (ANN) search [19]. Basically, we can generate b independent hashes to form a hash table of size 2^b (e.g., $b = 20$ means a hash table with 1 million buckets). For a query $\boldsymbol{q}$, one can compute b hash values $h_{RP}(\boldsymbol{q})$ to form a b-bit code and retrieve all data points form $\boldsymbol{X}$ which fall into the bucket corresponding to the b-bit code. To improve the retrieval accuracy, typically multiple hash tables are used to return the union of retrieved results from all the tables.

SignRP is one example of a broad class of method based on binary coding for ANN. That is, for each data vector $\boldsymbol{x} \in \mathbb{R}^d$, we hash it into a length-b binary 0/1 vector $h(\boldsymbol{x}) \in \{0, 1\}^b$, where the geometry of the data should be well preserved in the hamming space. Searching with binary codes has been widely applied in many applications, such as large-scale image retrieval [62, 15, 23, 18, 43, 44, 45, 58, 69, 65]. Besides the benefit of storage saving (especially for high-dimensional data), compressing data to binary can also significantly speedup the retrieval process. In this paper, our analysis and evaluation will focus on the hamming ranking approach which has been widely applied in image/video retrieval [23, 15, 47, 48, 3, 54, 67, 16, 46, 50], where we conduct exhaustive search in the hamming space. In fact, SignRP is also used by other ANN systems such as graph-based ANN as a crucial component to save the storage and speedup distance computations [72].

1.3 Random Fourier Features (RFF) ans SignRFF (Our Contribution)

Nonlinear kernels have been proven to be effective in many machine learning tasks [55, 2, 17]. See [31] for a study which empirically compared specially designed nonlinear kernels (i.e., "tunable GMM kernels") with deep nets and boosted trees. In this study, we focus on the Guassian kernel:

$$k(\rho) \equiv k(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma^2(1-\rho)}.$$

One can also utilize random projections to approximate the Gaussian kernel via the random Fourier features (RFF) [53, 52]. For data vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, the RFF for the Gaussian kernel is defined as

$$\textbf{RFF:} \quad F(\boldsymbol{x}) = \cos(\boldsymbol{w}^T\boldsymbol{x} + \tau), \qquad F(\boldsymbol{y}) = \cos(\boldsymbol{w}^T\boldsymbol{y} + \tau), \tag{3}$$

where $\boldsymbol{w} \sim N(0, \gamma^2\boldsymbol{I}_d)$ and $\tau \sim Unif(0, 2\pi)$ where $\boldsymbol{I}_d$ is the identity matrix. It holds that $\mathbb{E}[F(\boldsymbol{x})F(\boldsymbol{y})] = k(\boldsymbol{x}, \boldsymbol{y})/2$, i.e., the non-linear kernel can be preserved in expectation by the linear inner product of RFF. To approximate the kernel, we generate b independent RFFs for each data point using i.i.d. $\boldsymbol{w_1}, ..., \boldsymbol{w_b}$ and $\tau_1, ..., \tau_b$, which can be used for subsequent learning tasks. Additionally, it was shown in [30] that the normalized RFFs (NRFF) can substantially reduce the variance of RFF.

This method has lead to many applications in large-scale learning where one trains linear models on RFF to approximate non-linear kernel machines [68, 8, 60, 1, 59, 61, 37]. To extend RFF to retrieval tasks, in this paper, we propose SignRFF by keeping only the signs of the random Fourier features:

$$\textbf{SignRFF:} \quad h_{sign}(\boldsymbol{x}) = sign(F(x)) = sign(\cos(\boldsymbol{w}^T\boldsymbol{x} + \tau)). \tag{4}$$

We skip the same step for $h_{sign}(y)$. Note that, this is different from the earlier work called "SQ-RFF" [51] which proposed to construct binary codes from RFFs using stochastic binary quantization:

$$\textbf{SQ-RFF:} \quad h_{SQ}(\boldsymbol{x}) = sign(F(\boldsymbol{x}) + \xi) = sign(\cos(\boldsymbol{w}^T\boldsymbol{x} + \tau) + \xi), \tag{5}$$

where $\xi \sim Unif(-1, 1)$ is a random perturbation. To an extent, our idea was inspired by [34], which showed that the stochastic perturbation in [10] for quantized random projections was not needed. In this study, we will show theoretically and empirically that SignRFF is better than SQ-RFF. In particular, we propose a new evaluation metric named "ranking efficiency" (RE) to serve as a unified measure to analytically compare the search performance of different LSH methods. Under this metric, SignRFF consistently outperforms various other LSH schemes. Our experimental study also confirms that RE is a strong predictor of the empirical search performances.

1.4 Other Related Methods

Quantizations methods for random projections have been extensively studied in the literature, e.g., [14, 4, 10, 12, 34, 32, 39, 38]. Quantized methods for random Fourier features have also been heavily-studied [51, 71, 41, 41, 40]. In the meanwhile, there have been many works which have focused

on learning *data-dependent* binary hash codes, through different objective functions. Examples include Iterative Quantization (ITQ) [15], Spectral Hashing (SpecH) [62] and Binary Reconstruction Embedding (BRE) [23]. Recently, some unsupervised deep learning based methods have been proposed, many of which are, to some extent, "task-specific" for cross-modal/video/image retrieval, implemented based on some deep models like the autoencoder and VGG-16 [44, 11, 7, 36, 66, 16, 46, 50], showing promising performance in image retrieval tasks by taking advantage of the complicated model structures (e.g., CNN layers) [66, 50].

2 Background: Locality-Sensitive Hashing (LSH)

As mentioned earlier, in large-scale information retrieval, exhaustive search of the exact nearest neighbors is usually too expensive. A common relaxation in this setting is the Approximate Nearest Neighbor (ANN) search [19], where we return a "good" neighborhood of a query with high probability. In this paper, we consider the search problem with data points in $\mathbb{R}^d$. $\boldsymbol{X}$ denotes the database consisting of n data points, and $\boldsymbol{q}$ is a query point. Recall that $\boldsymbol{x}, \boldsymbol{y}$ are two data points with $\rho = \cos(\boldsymbol{x}, \boldsymbol{y})$.

Definition 1 ($\tilde{S}$-neighbor)**.** *For a similarity measure $S : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$, the $\tilde{S}$-neighbor set of $\boldsymbol{q}$ is defined as $\{\boldsymbol{x} \in \boldsymbol{X} : S(\boldsymbol{x}, \boldsymbol{q}) > \tilde{S}\}$.*

Definition 2 ($(c, \tilde{S})$-ANN)**.** *An algorithm $\mathbb{A}$ is a $(c, \tilde{S})$-ANN method provided the following: with probability at least $1 - \delta$, for $0 < c < 1$, if there exists an $\tilde{S}$-neighbor of $\boldsymbol{q}$ in $\boldsymbol{X}$, $\mathbb{A}$ returns a $c\tilde{S}$-neighbor of $\boldsymbol{q}$, where $\delta > 0$ is a parameter.*

One popular family of hash functions satisfying Definition 2 is the Locality-Sensitive Hashing (LSH), whose general definition is provided below.

Definition 3 ([19])**.** *A family of hash functions $\mathcal{H}$ is called $(\tilde{S}, c\tilde{S}, p_1, p_2)$-locality-sensitive for similarity measure S and $0 < c < 1$, if for $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ and hash function h uniformly chosen from $\mathcal{H}$, it holds that: 1) If $S(\boldsymbol{x}, \boldsymbol{y}) \geq \tilde{S}$, then $P(h(\boldsymbol{x}) = h(\boldsymbol{y})) \geq p_1$; 2) If $S(\boldsymbol{x}, \boldsymbol{y}) \leq c\tilde{S}$, then $P(h(\boldsymbol{x}) = h(\boldsymbol{y})) \leq p_2$, with $p_2 < p_1$.*

A key intuition of LSH is that, similar data points will have a higher chance of hash collision in the Hamming space. One example of LSH method, associated with the cosine similarity, is given by the SignRP approach as in (1). Note that the hash collision probability (2) is increasing in ρ, which, by Definition 3, is the key to ensure the locality sensitivity of SignRP.

Compared with the data-dependent methods (e.g., as mentioned in Section 1.4), LSH has seve advantages. Firstly, although data-dependent procedures can provide improved performance with fewer binary codes, a known weakness of many of these mechanisms is the performance bottleneck when we increase the code length b [51, 21]. On the other hand, the search performance of the data-independent LSH would keep boosting with larger b. In many scenarios where short codes (e.g., ≤ 128 bits) cannot achieve a desirable level of search accuracy for practical purposes, using longer LSH codes could be more favorable. Moreover, LSH is very simple to implement (only with random projections), while data-dependent methods require additional optimization/training and longer inference time (e.g., for deep networks). Lastly, LSH enjoys rigorous theoretical guarantees on the retrieval performance. Therefore, LSH has been a popular hashing method with many practical applications for decades [56, 28, 49, 13, 6, 33, 70, 25, 27, 9, 5, 63, 64, 42].

Kernelized LSH (KLSH). The SignRP (1) approach is based on linear random projections. Recall the Gaussian kernel function defined for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ as $k(\rho) \equiv k(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma^2(1-\rho)}$, where γ is a hyper-parameter. Let $\Psi : \mathbb{R}^d \mapsto \mathcal{F}$ be the feature map to the kernel induced feature space $\mathcal{F}$. To incorporate non-linearity in the hash codes, [24] proposed Kernelized Locality-Sensitive Hashing (KLSH) by conceptually applying SignRP (1) in the kernel induced feature space $\mathcal{F}$, i.e., $h(\boldsymbol{x}) = sign(\boldsymbol{w}^T \Psi(\boldsymbol{x}))$. As in many cases (e.g., for the Gaussian kernel) the map Ψ cannot be explicitly identified, KLSH proposes to approximate the random Gaussian distribution using data through the Central Limit Theorem (CLT) in the Reproducing Kernel Hilbert Space. Specifically, we first sample m data points from $\boldsymbol{X}$ to form a kernel matrix $\boldsymbol{K}$, then uniformly pick t points from $[1, ..., m]$ at random to approximate the Gaussian distribution. After some algebra, the hash code for

$\boldsymbol{x} \in \mathbb{R}^d$ is finally computed as

$$\textbf{KLSH:} \quad h_{KLSH}(\boldsymbol{x}) = sign(\sum_{i=1}^{m} \boldsymbol{r}(i) k(\boldsymbol{x}, \boldsymbol{x}_i)), \tag{6}$$

where $\boldsymbol{r} = \boldsymbol{K}^{-1/2}\boldsymbol{e}_t$, and $\boldsymbol{e}_t \in \{0,1\}^m$ has ones at the indices of the t selected points. Since KLSH uses a pool of data samples to approximate the Gaussian distribution, the hash codes are in fact dependent in implementation. Thus, a performance bottleneck has also been observed for KLSH as b increases [21], similar to many data-dependent methods. See Appendix D for more explanation.

3 Locality-Sensitive Hashing From Random Fourier Features

Random Fourier feature (RFF) was proposed in [52] to alleviate the computational bottleneck of standard kernel methods. For a data vector $\boldsymbol{x} \in \mathbb{R}^d$, recall (3) the RFF for the Gaussian kernel as

$$\textbf{RFF:} \quad F(\boldsymbol{x}) = \cos(\boldsymbol{w}^T \boldsymbol{x} + \tau),$$

where $\boldsymbol{w} \sim N(0, \gamma^2 \boldsymbol{I}_d)$ and $\tau \sim Unif(0, 2\pi)$ where $\boldsymbol{I}_d$ is the identity matrix. It holds that $\mathbb{E}[F(\boldsymbol{x})F(\boldsymbol{y})] = k(\boldsymbol{x}, \boldsymbol{y})/2$. The probability distribution of RFF is given as follows.

Lemma 1 ([41]). *For two normalized data points $\boldsymbol{x}, \boldsymbol{y}$ with cosine ρ, let $F(\cdot)$ be the RFF defined as (3). The joint distribution of $z_x = F(x)$ and $z_y = F(y)$ is*

$$f(z_x, z_y|\rho) = \frac{\sum_{k=-\infty}^{\infty} \left[\phi_\sigma(a_x^* - a_y^* + 2k\pi) + \phi_\sigma(a_x^* + a_y^* + 2k\pi)\right]}{\pi\sqrt{1 - z_x^2}\sqrt{1 - z_y^2}},$$

where $a_x^* = \cos^{-1}(z_x), a_y^* = \cos^{-1}(z_y)$, and $\phi_\sigma(\cdot)$ is the p.d.f. of $N(0, \sigma^2)$ with $\sigma = \sqrt{2(1-\rho)}\gamma$. Furthermore, $\mathbb{E}[sign(F(\boldsymbol{x}))sign(F(\boldsymbol{y}))]$ is an increasing function of ρ.

3.1 SQ-RFF with Stochastic Quantization

To extend RFF to efficient search algorithms, [51] designed a mapping $[-1, 1] \mapsto \{0, 1\}$ to construct binary codes from RFF. For $\boldsymbol{x} \in \mathbb{R}^d$, the code is produced by stochastic quantization (SQ):

$$\textbf{SQ-RFF:} \quad h_{SQ}(\boldsymbol{x}) = sign(F(\boldsymbol{x}) + \xi) = sign(\cos(\boldsymbol{w}^T \boldsymbol{x} + \tau) + \xi),$$

where ξ is a random perturbation from $Unif(-1, 1)$. Effectively, the so-called SQ-RFF applies a sampling procedure where we first compute the RFF $z = \cos(\boldsymbol{w}^T \boldsymbol{x} + \tau)$, and then assign it to 1 (otherwise -1) with probability $\frac{|1+z|}{2}$. This approach has been used in [71, 41] for large-scale low-precision kernel training. The collision probability of "SQ-RFF" is given below.

Theorem 1 ([51]). *For SQ-RFF (5), for normalized $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ with $\cos(\boldsymbol{x}, \boldsymbol{y}) = \rho$, it holds that*

$$P_{SQ}(\rho) := P(h_{SQ}(\boldsymbol{x}) = h_{SQ}(\boldsymbol{y})) = 1 - \frac{8}{\pi^2} \sum_{s=1}^{\infty} \frac{1 - e^{-\gamma^2 s^2 (1-\rho)}}{4s^2 - 1}. \tag{7}$$

Proposition 1. *The SQ-RFF in (5) is $(\tilde{k}, c\tilde{k}, P_{SQ}(\rho_1), P_{SQ}(\rho_2))$-locality sensitive w.r.t. similarity measure $k(\cdot)$, where $\rho_1 = \log(\tilde{k})/\gamma^2 + 1$ and $\rho_2 = \log(c\tilde{k})/\gamma^2 + 1$.*

Proof. According to Definition 3, SQ-RFF is locality-sensitive w.r.t. ρ because (7) is an increasing function of ρ. Hence, it is also locality-sensitive w.r.t. the kernel $k(\rho)$ by the monotonicity of $k(\rho)$. The ρ_1 and ρ_2 are derived by inserting $\tilde{k}$ into the inverse map of the kernel function. □

3.2 SignRFF: Why Not Drop the Noise?

The SQ-RFF has been a standard approach for constructing binary codes from RFF for over a decade. Yet, is the additional perturbation added to the RFF before binarization really necessary? Motivated by

this question, we propose a simpler coding strategy to directly take the sign of RFF (i.e., deterministic quantization) and remove the noise ξ. Formally, the SignRFF approach is defined by

$$\textbf{SignRFF:} \quad h_{sign}(\boldsymbol{x}) = sign(F(x)) = sign(\cos(\boldsymbol{w}^T\boldsymbol{x} + \tau)).$$

Operationally, SignRFF is extremely convenient. At the first glance, it may appear a bit surprising that this simple scheme has not been studied in literature. We believe one of the reasons might be that the theoretical correctness of SignRFF was hard to justify. Based on the joint density function of RFF, we show that SignRFF indeed belongs to the LSH family.

Proposition 2. *The SignRFF (4) is $(\tilde{k}, c\tilde{k}, P_{sign}(\rho_1), P_{sign}(\rho_2))$-locality sensitive w.r.t. to $k(\cdot)$, with $\rho_1 = \log(\tilde{k})/\gamma^2 + 1$ and $\rho_2 = \log(c\tilde{k})/\gamma^2 + 1$, with collision probability*

$$P_{sign}(\rho) := P(h_{sign}(\boldsymbol{x}) = h_{sign}(\boldsymbol{y})) = 2\int_0^1\int_0^1 f(z_x, z_y|\rho)dz_x dz_y, \tag{8}$$

where $f(z_x, z_y|\rho)$ is the density function of RFF given by Lemma 1.

Proof. By Definition 3 and the monotonicity of $k(\rho)$, it suffices to show that the collision probability, $P_{sign}(\rho)$, is increasing in ρ. This immediately follows from Lemma 1 that

$$\mathbb{E}[sign(F(\boldsymbol{x}))sign(F(\boldsymbol{y}))] = P_{sign}(\rho) - (1 - P_{sign}(\rho)) = 2P_{sign}(\rho) - 1$$

is an increasing function in ρ. □

Compared with SQ-RFF, the proposed SignRFF exhibits less "variation" brought by the stochastic sampling procedure. In fact, in the kernel approximation problem, it has been shown that stochastic rounding has higher variance due to the noise ξ which hurts the kernel estimation accuracy [41]. While such comparison is not immediately obvious for the task of nearest neighbor search, next we will show that dropping the extra noise in the binary codes indeed leads to improved search performance, measured by a new metric specifically for the binary hash codes from LSH.

4 Ranking Efficiency: A New Measure for Search Performance

In this section, we systematically compare the above LSH methods (SignRP, SQ-RFF and SignRFF) from a analytical point of view. Before we start, we first provide some analysis of KLSH introduced in Section 2 which will also be included in our comparison.

Collision probability of KLSH. In our theoretical analysis, we assume for KLSH [24] that the Gaussian projection in the kernel induced feature space $\mathcal{F}$ is truly random (recall in practice we use data dependent approximations). That is, KLSH "ideally" performs SignRP in the kernel space: $h_{KLSH}(\boldsymbol{x}) = sign(\boldsymbol{w}^T\Psi(\boldsymbol{x}))$ where $\boldsymbol{w}$ is a random Gaussian vector with proper dimensionality. Since $\Psi(\boldsymbol{x})^T\Psi(\boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y})$, applying (2) in $\mathcal{F}$ we obtain the collision probability as

$$P_{KLSH}(\rho) := P(h_{KLSH}(\boldsymbol{x}) = h_{KLSH}(\boldsymbol{y})) = 1 - \frac{\cos^{-1}(e^{-\gamma^2(1-\rho)})}{\pi}. \tag{9}$$

Remark 1. *Again, we emphasize that due to the dependence among KLSH hash codes resulting from its implementation, in practice the collision probability of KLSH would be different from (9).*

Proposition 3. *KLSH is $(\tilde{k}, c\tilde{k}, 1 - \frac{\cos^{-1}(\tilde{k})}{\pi}, 1 - \frac{\cos^{-1}(c\tilde{k})}{\pi})$-locality sensitive w.r.t. $k(\cdot)$.*

In Figure 1, we plot the theoretical hash collision probabilities of the four hashing algorithms as in (2), (7), (8) and (9). We see that SQ-RFF has highest collision probability. Recall Definition 3 of the $(\tilde{k}, c\tilde{k}, p_1, p_2)$-LSH. It is known [19] that one can construct an LSH data structure with the worst case query time $\mathcal{O}(n^R)$, where $R := \frac{\log p_1}{\log p_2}$ is called the *LSH efficiency*, which has been used in literature, e.g., to compare SimHash vs. MinHash [56]. However, since the LSH efficiency is only considers the collision probability (i.e., first moemnt) and based on the a worst case analytical bound, it may not well explain/predict the practical search performance. See Appendix B for related discussion.

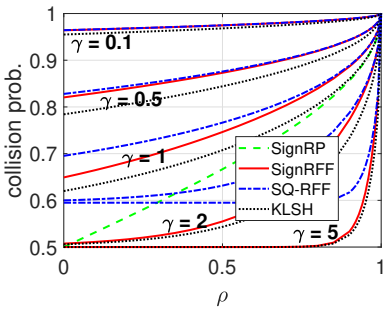

Figure 1: Hash collision probabilities.

4.1 Ranking Efficiency

We now introduce the concept of *ranking efficiency*, which allows us to better compare different LSH methods, theoretically, in terms of search performance. As we will see, the new metric shows the superiority of SignRFF over SQ-RFF, and provides insight on comparing (linear) SignRP with non-linear LSH methods (e.g., when is SignRFF advantageous?). The definition is motivated by the fact that the effectiveness of LSH binary codes in nearest neighbor retrieval is essentially determined by how well it can preserve the ranking of the true similarities in the Hamming space. Suppose $\boldsymbol{x}$ and $\boldsymbol{y}$ are two data points in the database, and $\boldsymbol{q}$ is a query with $\rho_x = \cos(\boldsymbol{q}, \boldsymbol{x})$, $\rho_y = \cos(\boldsymbol{q}, \boldsymbol{y})$. Assume $\boldsymbol{x}$ is the true nearest neighbor of $\boldsymbol{q}$, implying $\rho_x > \rho_y$. By the property of LSH, we know that the hash collision probability $p_x > p_y$. For an LSH hash function h, define the corresponding collision probability estimators as

$$\hat{p}_x = \frac{1}{b}\sum_{i=1}^{b} \mathbb{1}\{h_i(\boldsymbol{x}) = h_i(\boldsymbol{q})\}, \quad \hat{p}_y = \frac{1}{b}\sum_{i=1}^{b} \mathbb{1}\{h_i(\boldsymbol{y}) = h_i(\boldsymbol{q})\}.$$

Now, the problem becomes comparing $\hat{p}_x$ and $\hat{p}_y$ to estimate the true ranking of p_x and p_y. We consider the event of obtaining a wrong similarity comparison from our estimation, i.e. $\hat{p}_x \leq \hat{p}_y$. Obviously, a higher probability implies worse search accuracy, as we are more likely to retrieve the wrong nearest neighbor $\boldsymbol{y}$. Denote $E_x = \mathbb{E}[\mathbb{1}\{h(\boldsymbol{x}) = h(\boldsymbol{q})\}]$, $E_y = \mathbb{E}[\mathbb{1}\{h(\boldsymbol{y}) = h(\boldsymbol{q})\}]$, and $Cov(\mathbb{1}\{h(\boldsymbol{x}) = h(\boldsymbol{q})\}, \mathbb{1}\{h(\boldsymbol{y}) = h(\boldsymbol{q})\}) = \Sigma = \begin{pmatrix} V_x & V_{xy} \\ V_{xy} & V_y \end{pmatrix}$. By the Central Limit Theorem, as $b \to \infty$ asymptotically, we have that $\begin{pmatrix} \hat{p}_x \\ \hat{p}_y \end{pmatrix} \sim N(\begin{pmatrix} E_x \\ E_y \end{pmatrix}, \Sigma/b)$. This approximation would be good with a sufficiently large b, e.g. $b \geq 30$. In this regime, we have

$$P(\hat{p}_x \leq \hat{p}_y) = P(\hat{p}_x - \hat{p}_y \leq 0) = 1 - \Phi\Big(\frac{\sqrt{b}(E_x - E_y)}{\sqrt{V_x + V_y - 2V_{xy}}}\Big), \tag{10}$$

where $\Phi(\cdot)$ is the c.d.f. of standard normal distribution. Based on this characterization, we formally define the *ranking effectiveness* as follows.

Definition 4 ((ρ, c)-Ranking Efficiency (RE))**.** *For a LSH method, let the hash collision probability at cosine ρ and $c\rho$ be E and E_c, respectively, with $0 \leq c < 1$. Let $V = E(1-E)$, $V_c = E_c(1-E_c)$. The (ρ, c)-ranking efficiency (RE) is defined as*

$$\textbf{\textit{Ranking Efficiency:}} \qquad RE = \frac{E - E_c}{\sqrt{V + V_c}}. \tag{11}$$

Remark 2. *In most cases, we may care more about large c values (e.g., $c = 0.95$) which corresponds to similar data points with higher chance of reversed ranking.*

Remark 3. *The covariance V_{xy} in (10) is in general intractable which depends on the specific data $\boldsymbol{x}$ and $\boldsymbol{y}$. We assume that it has same relative impact on the RE values and drop it from the definition for simplicity. The RE, when multiplied by $\sqrt{b}$, is equivalent to the z-score in a two-sample z-test.*

Higher RE implies smaller probability of (10), which is more favorable. For all the LSH methods, the E_x and E_y are concretely the collision probabilities given in (2), (9), (7) and (8), and $V_x = E_x(1-E_x)$, $V_y = E_y(1-E_y)$ from the binomial distribution, respectively.

4.2 Analytical Comparison of LSH Methods Through Ranking Efficiency

Next, we leverage Definition 4 to theoretically compare different hashing methods. In Figure 2, we provide the RE of (linear) SignRP, KLSH, SQ-RFF and SignRFF at different ρ level, which covers the cases in our empirical study (Section 5). We present the plots for $c = 0.95$, and the conclusions are the same for other c values. We observe that:

- **Comparing Non-linear LSHs.** Firstly, compared with the previous RFF-based approach SQ-RFF, the proposed SignRFF is uniformly more efficient at all ρ, verifying its superiority. Compared with KLSH, SignRFF is more efficient when γ is large (e.g., $\gamma > 2$) in all plots, while KLSH tends to be more efficient only with small γ.

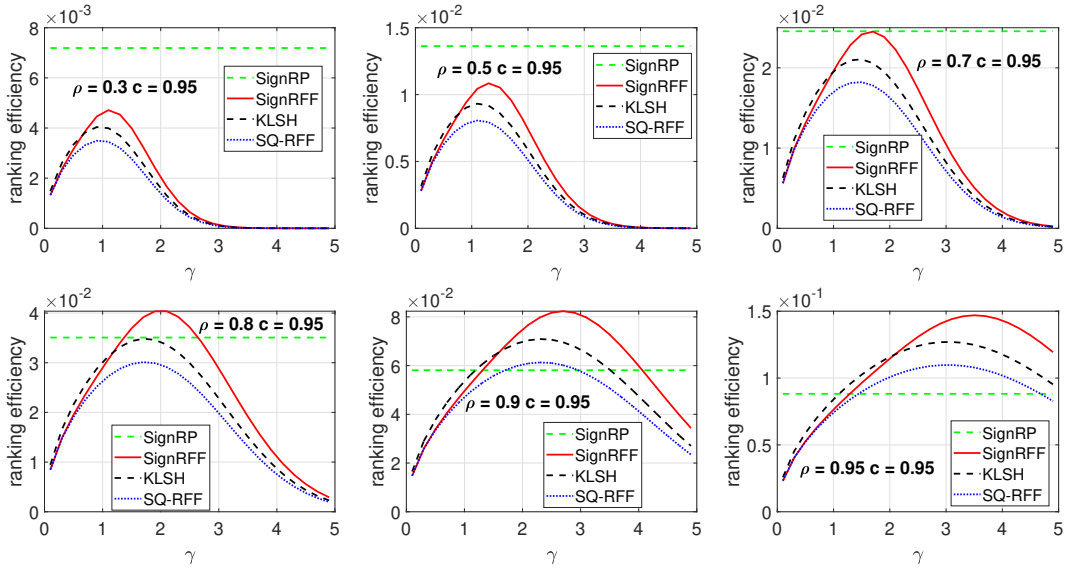

Figure 2: Ranking efficiency (Definition 4) of different LSH methods with various ρ, at $c = 0.95$.

- **When should we prefer non-linear LSH?** We see that when ρ is high (e.g., $\rho > 0.8$), with proper tuning, kernel methods (e.g., SignRFF) can be more efficient than SignRP. However, if the target ρ is small (e.g., $\rho < 0.7$), SignRP becomes more favorable, even for best tuned non-linear LSH. In other words, SignRFF is better than SignRP on datasets where the near neighbors are close to each other, with high similarity/short distance.

The ranking efficiency measures the search accuracy for a given ρ level. In practice, a dataset contains many sample pairs with different ρ. Our experiments in the next section show that the performance is largely consistent with the prediction at the "average ρ" level. That said, ranking efficiency may provide a convenient way to choose the proper LSH method based on the data of interest.

5 Experiments

We conduct experiments to demonstrate the effectiveness of our approach and justify that ranking efficiency indeed provides reliable prediction of the empirical search accuracy.

Datasets. We use three popular benchmark datasets for image retrieval. The `SIFT` dataset [20] contains 1M 128-dimensional SIFT image features, and 1000 query samples. The `MNIST` dataset [26] contains 60000 hand-written digits. The `CIFAR` dataset [22] contains 50000 natural images and we use the gray-scale images in our experiments. For these two datasets, we randomly choose 1000 samples from the test set as queries. In addition, when comparing with VGG-16 [57] based deep methods, following prior literature (e.g.,[66, 50]), we use the 4096-d features from the last fully connected layer of the pre-trained VGG-16 network as the input data for shallow methods for fairness. This dataset is called "`CIFAR-VGG`". On all datasets, the data points are normalized to have unit norm. In Figure 3, we report the average cosine between queries to their N-th nearest neighbor, which can be approximately regarded as the "target ρ level" when we compare the ranking efficiency (e.g., 0.95 for `CIFAR`).

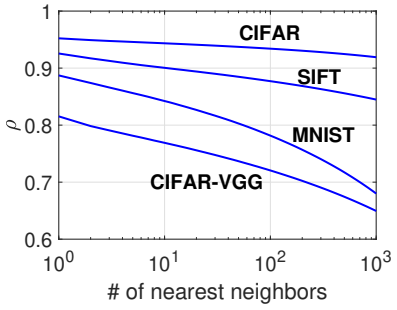

Figure 3: Average ρ to the N-th neighbor on four datasets.

Methods and Evaluation. We compare the following unsupervised hashing methods: 1) **SignRP** [4], defined by (1) with random Gaussian projections; 2) **Iterative Quantization (ITQ)** [15], which finds rotations to minimize the quantization error of mapping the randomly projected data to binary; 3)

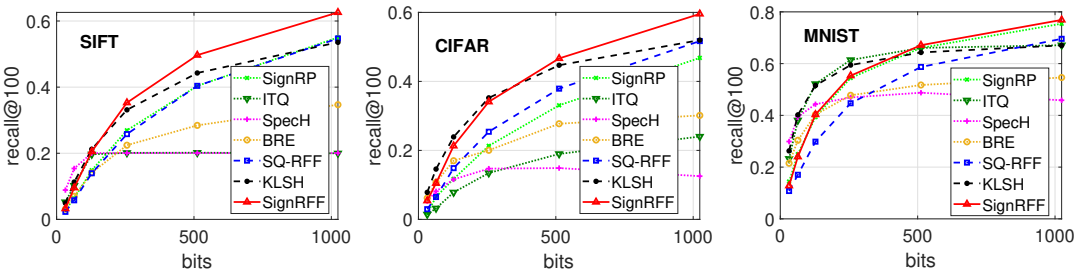

Figure 4: Recall@100 vs. b. Note that in our case, recall@100 is equivalent to precision@100.

Spectral Hashing (SpecH) [62], which is based on quantizing the values of analytical eigenfunctions computed along principal component directions of the data; 4) **Binary Reconstruction Embedding (BRE)** [23], which explicitly minimizes the reconstruction error between the original distances and the Hamming distances. We use 1000 random samples for model training as suggested by [23]; 5) **KLSH** [24] as in (6), with $m = 500$ random samples for formulating the kernel matrix and $t = 50$ samples for the CLT Gaussian approximation, more accurate than $(300, 30)$ recommended in [24]; 6) **SQ-RFF** [51] given in (5), binary codes from stochastically quantized RFF; 7) Our proposed **SignRFF** method (4) with deterministic quantization from RFF. For methods (5) - (7) involving Gaussian kernel, we tune γ on a fine grid over $0.1 \sim 5$ and report the best result.

For each tested method, we generate binary codes and find neighbors based on Hamming distances. After running each algorithm, the Hamming ranking returns R estimated nearest neighbors to each query. Define N as the number of ground truth neighbors. For each query point, the ground truth nearest neighbors are set by ranking the top $N = 100$ smallest Euclidean distance (top-100 largest cosine similarity). We report the average recall or precision over 1000 queries. The search recall and precision (the higher the better) are defined as recall@R $= \frac{\text{\# true neighbors in } R \text{ retrieved points}}{N}$ and precision@R $= \frac{\text{\# true neighbors in } R \text{ retrieved points}}{R}$. Note that recall@N is equivalent to precision@N.

5.1 Results

In Figure 4, we report the recall@100 (precision@100) against the number of binary codes b:

- On `SIFT` and `MNIST`, data-dependent methods (ITQ, SpecH, BRE) perform well with low bits, but the recall does not improve much after $b > 100 \sim 200$. Yet, their recall level is too low (e.g. < 0.3 on `SIFT` and `CIFAR`) for real-world tasks, which is a known limitation of these methods. When $b \geq 256$, LSH-type methods start to dominate. On `CIFAR`, SignRFF and KLSH outperform all the data-dependent methods even with low bits.
- On all three datasets, SignRFF is substantially better than SQ-RFF with all b. Due to dependence, KLSH has higher recall than SignRFF when b is small (e.g., ≤ 256), but is beaten by SignRFF with more bits. The gap is significant and consistent when b is as large as 512, where SignRFF achieves the highest recall on all three datasets.

In Figure 5, we present the recall versus the number of retrieved neighbors, with $b = 512$ and $b = 1024$. In general, SignRFF performs the best on all three datasets with a sufficient number of bits. Due to space limitation, we report search precision results with consistent conclusions in Appendix A, which also includes discussion on the practical implementation and efficiency.

Comparison with Deep Hashing. We provide experiments on the `CIFAR-VGG` dataset using a recent unsupervised deep hashing method, the Contrastive Information Bottleneck (CIB) [50], which uses VGG-16 pre-trained model as the backbone. We apply the same training setting as in [50]. Note that, CIB is actually not designed to find the most similar data points. Instead, by using techniques like cropping and rotation in CNN, CIB aims at finding data points with the same label as the query, which does not necessarily imply high similarity. Hence, to favor CIB, in this experiment we expand the range of true neighbors of a query to the top-1000 most similar points in the database as in [50] (see Appendix C for detailed discussion). We observe:

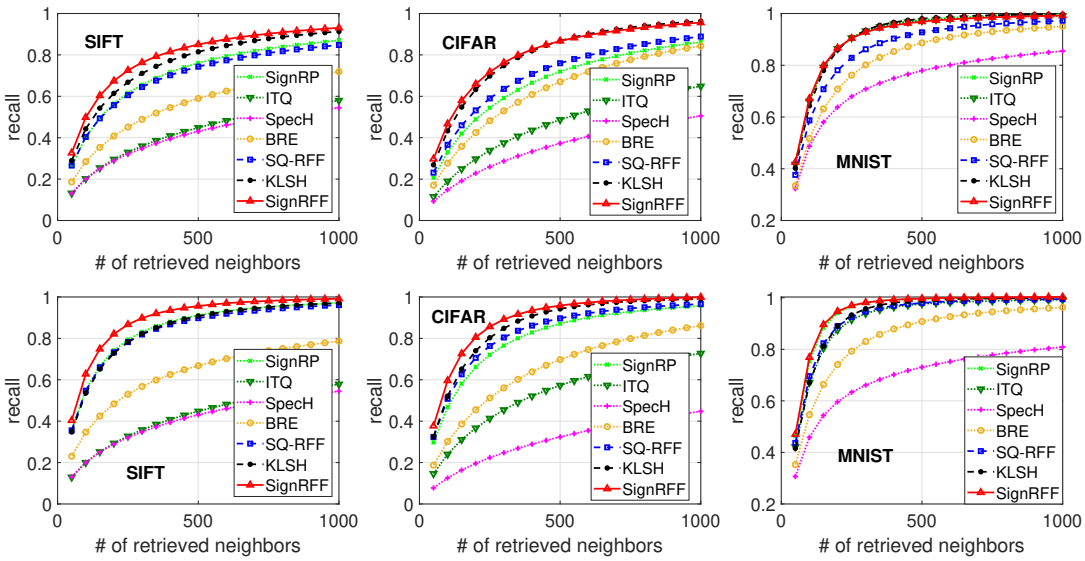

Figure 5: Recall@R vs. # of retrieved neighbors R. **1st row:** $b = 512$. **2nd row:** $b = 1024$.

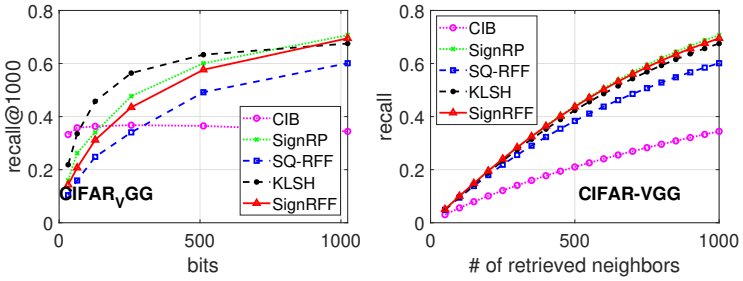

Figure 6: **Left:** Recall@1000 (precision@1000) vs. b. **Right:** Recall vs. # retrieved points, $b = 1024$.

- From Figure 6, CIB performs the best when $b \leq 64$, illustrating the benefit of deep hashing with short codes. Yet, the recall is only 0.3∼0.4, and does not improve with more bits. When $b \geq 256$, SignRP, SignRFF and KLSH provide much higher recall than CIB.

- Due to the dependence among codes, KLSH performs the best on this dataset with short codes. SignRFF again consistently improves SQ-RFF, and surpasses KLSH with $b = 1024$.

5.2 Ranking Efficiency is a Predictive Measure of Search Accuracy

Firstly, as shown in Figure 3, the average ρ between each query and its near neighbors is around 0.7, 0.8, 0.9 and 0.95 for `CIFAR-VGG`, `MNIST`, `SIFT` and `CIFAR`, respectively. The theoretical RE in Figure 2 suggests that compared with SQ-RFF (blue), SignRP (light green) would perform better on `MNIST` and `CIFAR-VGG`, similarly on `SIFT`, and worse on `CIFAR`. This aligns very well with the recall curves in Figure 4 and Figure 5. Here we use SQ-RFF vs. SignRP as an example; similar alignment holds for comparing other methods. To make more detailed justification, in Figure 7, we additionally provide the recall against the Gaussian parameter γ. In Figure 2, the RE curves predict that KLSH should perform better with small γ, while SignRFF is more powerful with larger γ. This matches the empirical evidence, e.g., 1 for KLSH vs. 2.5 for SignRFF on `SIFT` to reach highest recall.

Our results verify that: 1) the ranking efficiency can be used as an informative and effective measure to compare different LSH methods in practice; 2) kernel based LSH (e.g., the proposed SignRFF) is more favorable than the linear SignRP method in high similarity region.

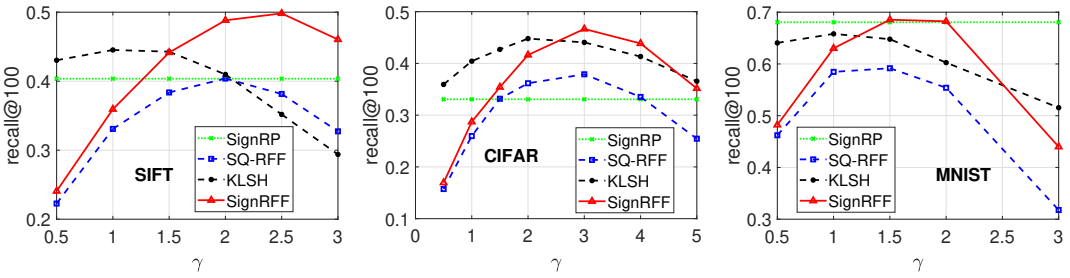

Figure 7: Recall@100 (precision@100) vs. γ, with $b = 512$.

6 Conclusion

Random Fourier Feature (RFF) is a popular method in large-scale kernel learning, which has also been used for constructing binary codes for fast nearest neighbor retrieval [51]. In this work, we revisit RFF-based binary hashing, and propose a simple yet more powerful strategy named SignRFF with proven locality-sensitivity. On a higher level, we design a new unified theoretical framework to compare different hashing methods, based on a measure called *ranking efficiency*. This metric provides an interesting comparison that, one should prefer non-linear kernel based LSH methods (such as SignRFF) over the classical linear SignRP in the high similarity region when data points are close to each other. Our extensive experiments on image retrieval tasks validate that SignRFF is consistently better than the previous RFF-based approach, SQ-RFF, and outperforms data-dependent and deep hashing methods when the number of hash bits is moderately large. Moreover, our empirical results match well with the theoretical comparison of ranking efficiency, suggesting that it could serve as a promising tool for comparing different LSH methods and choosing the proper one in practice.

Acknowledgement: Xiaoyun Li's work was conducted as a PhD student in the Department of Statistics at Rutgers University.

References

[1] Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. Random fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 253–262, Sydney, Australia, 2017.

[2] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[3] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. Deep cauchy hashing for hamming space retrieval. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1229–1237, Salt Lake City, UT, 2018.

[4] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing (STOC)*, pages 380–388, Montreal, Canada, 2002.

[5] Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Ré. MONGOOSE: A learnable LSH framework for efficient neural network training. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, Austria, 2021.

[6] Lin Chen, Hossein Esfandiari, Gang Fu, and Vahab S. Mirrokni. Locality-sensitive hashing for f-divergences: Mutual information loss and beyond. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10044–10054, Vancouver, Canada, 2019.

[7] Zhixiang Chen, Xin Yuan, Jiwen Lu, Qi Tian, and Jie Zhou. Deep hashing via discrepancy minimization. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6838–6847, Salt Lake City, UT, 2018.

[8] Kacper Chwialkowski, Aaditya Ramdas, Dino Sejdinovic, and Arthur Gretton. Fast two-sample testing with analytic representations of probability measures. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1981–1989, Montreal, Canada, 2015.

[9] Shabnam Daghaghi, Tharun Medini, Nicholas Meisburger, Beidi Chen, Mengnan Zhao, and Anshumali Shrivastava. A tale of two efficient and informative negative sampling distributions. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 2319–2329, Virtual Event, 2021.

[10] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry (SCG)*, pages 253–262, Brooklyn, NY, 2004.

[11] Thanh-Toan Do, Dang-Khoa Le Tan, Trung T. Pham, and Ngai-Man Cheung. Simultaneous feature aggregating and hashing for large-scale image search. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4217–4226, Honolulu, HI, 2017.

[12] Wei Dong, Moses Charikar, and Kai Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 123–130, Singapore, 2008.

[13] Anne Driemel and Francesco Silvestri. Locality-sensitive hashing of curves. *arXiv preprint arXiv:1703.04040*, 2017.

[14] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[15] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 817–824, Colorado Springs, CO, 2011.

[16] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. Unsupervised semantic hashing with pairwise reconstruction. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (SIGIR)*, pages 2009–2012, Virtual Event, China, 2020.

[17] Trevor J. Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2 edition, 2009.

[18] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2938–2945, 2013.

[19] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 604–613, Dallas, TX, 1998.

[20] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011.

[21] Alexis Joly and Olivier Buisson. Random maximum margin hashing. In *CVPR 2011*, pages 873–880. IEEE, 2011.

[22] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[23] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1042–1050, Vancouver, Canada, 2009.

[24] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *proceedings of the IEEE 12th International Conference on Computer Vision (ICCV)*, pages 2130–2137, Kyoto, Japan, 2009.

[25] Dung D. Le and Hady W. Lauw. Stochastically robust personalized ranking for LSH recommendation retrieval. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pages 4594–4601, New York, NY, 2020.

[26] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[27] Yifan Lei, Qiang Huang, Mohan S. Kankanhalli, and Anthony K. H. Tung. Locality-sensitive hashing scheme based on longest circular co-substring. In *Proceedings of the 2020 International Conference on Management of Data (SIGMOD)*, pages 2589–2599, Online conference [Portland, OR, USA], 2020.

[28] Jinfeng Li, James Cheng, Fan Yang, Yuzhen Huang, Yunjian Zhao, Xiao Yan, and Ruihao Zhao. LoSHa: A general framework for scalable locality sensitive hashing. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 635–644, Shinjuku, Tokyo, Japan, 2017.

[29] Ping Li. Sign stable random projections for large-scale learning. *arXiv preprint arXiv:1504.07235*, 2015.

[30] Ping Li. Linearized GMM kernels and normalized random Fourier features. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 315–324, Halifax, Canada, 2017.

[31] Ping Li. Tunable GMM kernels. *arXiv preprint arXiv:1701.02046*, 2017.

[32] Ping Li. Sign-full random projections. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pages 4205–4212, Honolulu, HI, 2019.

[33] Ping Li, Xiaoyun Li, and Cun-Hui Zhang. Re-randomized densification for one permutation hashing and bin-wise consistent weighted sampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15900–15910, Vancouver, Canada, 2019.

[34] Ping Li, Michael Mitzenmacher, and Anshumali Shrivastava. Coding for random projections. In *Proceedings of the 31th International Conference on Machine Learning (ICML)*, pages 676–684, Beijing, China, 2014.

[35] Ping Li, Gennady Samorodnitsky, and John E. Hopcroft. Sign cauchy projections and chi-square kernel. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2571–2579, Lake Tahoe, NV, 2013.

[36] Shuyan Li, Zhixiang Chen, Jiwen Lu, Xiu Li, and Jie Zhou. Neighborhood preserving hashing for scalable video retrieval. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8211–8220, Seoul, Korea, 2019.

[37] Xiaoyun Li, Jie Gui, and Ping Li. Randomized kernel multi-view discriminant analysis. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI)*, pages 1276–1284, Santiago de Compostela, Spain, 2020.

[38] Xiaoyun Li and Ping Li. Generalization error analysis of quantized compressive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15124–15134, Vancouver, Canada, 2019.

[39] Xiaoyun Li and Ping Li. Random projections with asymmetric quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10857–10866, Vancouver, Canada, 2019.

[40] Xiaoyun Li and Ping Li. One-sketch-for-all: Non-linear random features from compressed linear measurements. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2647–2655, Virtual Event, 2021.

[41] Xiaoyun Li and Ping Li. Quantization algorithms for random fourier features. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 6369–6380, Virtual Event, 2021.

[42] Xiaoyun Li and Ping Li. C-MinHash: Improving minwise hashing with circulant permutation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 12857–12887, Baltimore, MD, 2022.

[43] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen. Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 27–35, 2015.

[44] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2064–2072, Las Vegas, NV, 2016.

[45] Hong Liu, Rongrong Ji, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Cross-modality binary code learning via fusion similarity hashing. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6345–6353, Honolulu, HI, 2017.

[46] Song Liu, Shengsheng Qian, Yang Guan, Jiawei Zhan, and Long Ying. Joint-modal distribution-based similarity hashing for large-scale unsupervised deep cross-modal retrieval. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (SIGIR)*, pages 1379–1388, Virtual Event, China, 2020.

[47] Mohammad Norouzi, Ali Punjani, and David J. Fleet. Fast search in hamming space with multi-index hashing. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3108–3115, Providence, RI, 2012.

[48] Eng-Jon Ong and Miroslaw Bober. Improved hamming distance search using variable length hashing. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2000–2008, Las Vegas, NV, 2016.

[49] Lianyong Qi, Xuyun Zhang, Wanchun Dou, and Qiang Ni. A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data. *IEEE J. Sel. Areas Commun.*, 35(11):2616–2624, 2017.

[50] Zexuan Qiu, Qinliang Su, Zijing Ou, Jianxing Yu, and Changyou Chen. Unsupervised hashing with contrastive information bottleneck. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 959–965, Virtual Event / Montreal, Canada, 2021.

[51] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1509–1517, Vancouver, Canada, 2009.

[52] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1184, Vancouver, Canada, 2007.

[53] Walter Rudin. *Fourier Analysis on Groups*. John Wiley & Sons, New York, NY, 1990.

[54] Dominik Schlegel and Giorgio Grisetti. HBST: A hamming distance embedding binary search tree for feature-based visual place recognition. *IEEE Robotics Autom. Lett.*, 3(4):3741–3748, 2018.

[55] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[56] Anshumali Shrivastava and Ping Li. In defense of minhash over simhash. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 886–894, Reykjavik, Iceland, 2014.

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, 2015.

[58] Jingkuan Song, Tao He, Lianli Gao, Xing Xu, Alan Hanjalic, and Heng Tao Shen. Binary generative adversarial networks for image retrieval. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAAI)*, pages 394–401, New Orleans, LA, 2018.

[59] Yitong Sun, Anna C. Gilbert, and Ambuj Tewari. But how does it work in theory? linear SVM with random features. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3383–3392, Montréal, Canada, 2018.

[60] Danica J. Sutherland and Jeff G. Schneider. On the error of random fourier features. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 862–871, Amsterdam, The Netherlands, 2015.

[61] Anthony Tompkins and Fabio Ramos. Fourier feature approximations for periodic kernels in time-series modelling. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, pages 4155–4162, New Orleans, LA, 2018.

[62] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1753–1760, Vancouver, Canada, 2008.

[63] Zhaozhuo Xu, Beidi Chen, Chaojian Li, Weiyang Liu, Le Song, Yingyan Lin, and Anshumali Shrivastava. Locality sensitive teaching. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 18049–18062, virtual, 2021.

[64] Zhaozhuo Xu, Zhao Song, and Anshumali Shrivastava. Breaking the linear iteration cost barrier for some well-known conditional gradient methods using maxip data-structures. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5576–5589, virtual, 2021.

[65] Chenggang Yan, Biao Gong, Yuxuan Wei, and Yue Gao. Deep multi-view enhancement hashing for image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(4):1445–1451, 2021.

[66] Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, and Dacheng Tao. DistillHash: Unsupervised deep hashing by distilling data pairs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2946–2955, Long Beach, CA, 2019.

[67] Erkun Yang, Tongliang Liu, Cheng Deng, and Dacheng Tao. Adversarial examples for hamming space search. *IEEE Trans. Cybern.*, 50(4):1473–1484, 2020.

[68] Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in Neural Information Processing Systems (NIPS)*, pages 485–493, Lake Tahoe, NV, 2012.

[69] Laihang Yu, Lin Feng, Huibing Wang, Li Li, Yang Liu, and Shenglan Liu. Multi-trend binary code descriptor: a novel local texture feature descriptor for image retrieval. *Signal Image Video Process.*, 12(2):247–254, 2018.

[70] Amir Zandieh, Navid Nouri, Ameya Velingker, Michael Kapralov, and Ilya P. Razenshteyn. Scaling up kernel ridge regression via locality sensitive hashing. In *Proceedings of th 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 4088–4097, Online [Palermo, Sicily, Italy], 2020.

[71] Jian Zhang, Avner May, Tri Dao, and Christopher Ré. Low-precision random fourier features for memory-constrained kernel approximation. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1264–1274, Naha, Okinawa, Japan, 2019.

[72] Weijie Zhao, Shulong Tan, and Ping Li. SONG: approximate nearest neighbor search on GPU. In *Proceedings of the 36th IEEE International Conference on Data Engineering (ICDE)*, pages 1033–1044, Dallas, TX, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results? [Yes] Public datasets.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All results are averaged over 5 runs.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Our experiments are performed on a single core 2.0GHz CPU.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A More Experiments

A.1 Precision

In Figure 8, we report the Precision@10 against the number of bits (1st row) and precision@R against the number of retrieved points R (2nd row). Basically, we get the same conclusions as those from the recall curves. SignRFF performs the best on all datasets after $b \geq 256$.

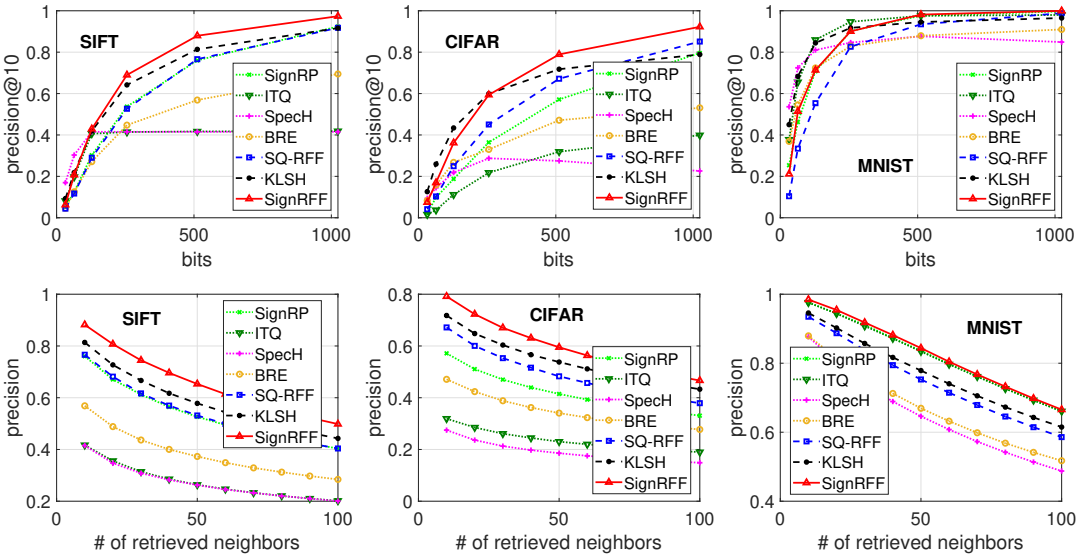

Figure 8: **1st row:** Precision@10 against b. **2nd row:** Precision vs. # of retrieved points, $b = 512$.

A.2 Practical Considerations and Efficiency

In practical retrieval systems, the query time is an important consideration which consists of the processing time (to generate query binary codes), and the search time (to compare the query codes to the database). We briefly discuss the two aspects below.

Processing time. Noticeably, another benefit of SignRFF is its simplicity in implementation. This can be reflected in the data processing time. In Figure 9, we plot the comparison of processing time (for 1000 queries). We observe that SignRFF and LSH are the most efficient methods (mostly only a random projection). KLSH is notably slower than SignRFF. The two data-dependent methods, SpecH and BRE, are significantly slower than SignRFF. This reveals a potential advantage of the simplicity of SignRFF in practical retrieval systems.

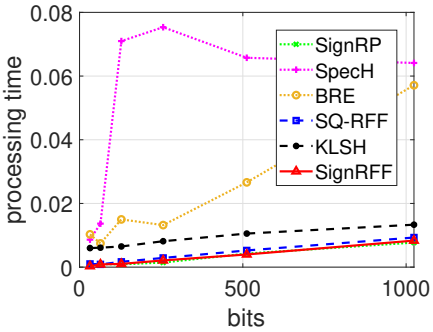

Figure 9: Data processing time (1000 queries).

Search time. In our experiments, we adopt the standard exact Hamming search by linear scan. On a single core 2.0GHz CPU compiled with C++, searching over 1M samples on `SIFT` takes

approximately 0.15s per query with $b = 512$. Note that linear scan is a naive strategy. While the efficiency of Hamming search algorithms is not the focus of this work, we point out that in practice, we can use multi-index hashing [47, 48] to perform exact Hamming search with a substantial $10^2 \sim 10^3$ times acceleration. Additionally, the method in [48] is particularly effective for long codes, the regime where SignRFF possesses its best advantage. As such, searching with SignRFF can also be very efficient in practice.

B More Analytical Figures

B.1 LSH Efficiency is Not Enough to Predict Search Accuracy

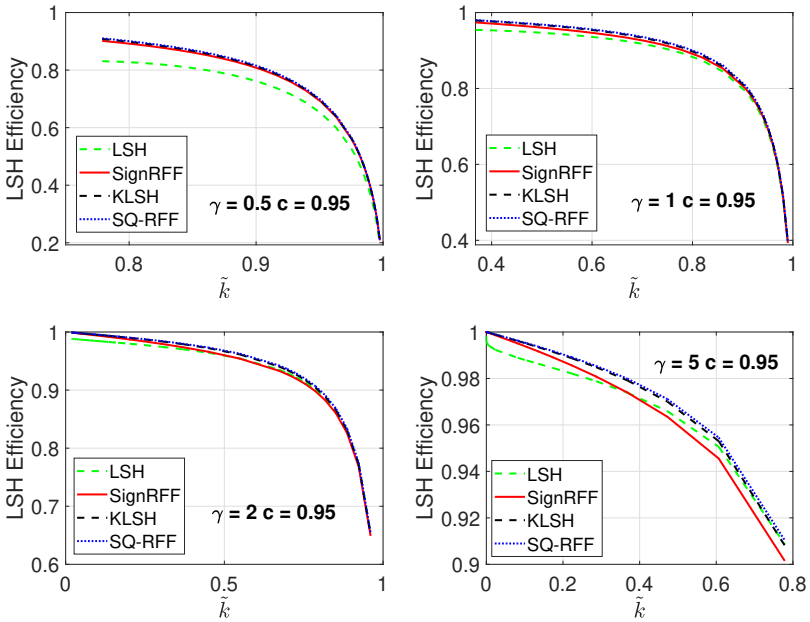

Figure 10: LSH efficiency of different LSH methods with various γ, $c = 0.95$. The x-axis $\tilde{k}$ is the Gaussian kernel value in alignment with Definition 3, Proposition 1, 1 and 3.

Recall Definition 3 of the $(\tilde{k}, c\tilde{k}, p_1, p_2)$-LSH. It is known [19] that one can construct an LSH data structure with the worst case query time $\mathcal{O}(n^R)$, where $R := \log p_1 / \log p_2$ is called the *LSH efficiency*, which has been used in literature to theoretically compare different LSH methods, e.g. SimHash vs. MinHash [56]. However, we found that in our case, the LSH efficiency does not provide much informative comparison of different LSH methods of interest. In Figure 10, we provide the LSH efficiency at multiple γ. Firstly, we see that the differences among the curves are very small. Basically, the figures tells us that SignRFF is always better than KLSH and SQ-RFF, but do not provide the comparison of SignRFF and KLSH regarding γ, as the ranking efficiency does. Also, the figures seem to suggest that SignRFF could (roughly) be better than LSH with large γ and large ρ, but it does not give a good threshold at around $\rho = 0.7$ (validated by the experiments) as suggested by the ranking efficiency. Thus, LSH efficiency is insufficient to well predict the practical Hamming search performance.

B.2 Ranking Efficiency: More c and ρ Values

we provide more theoretical comparisons on the ranking efficiency at more ρ and c values. The observation (relative comparison) is similar to the results presented in the main paper. For convenience, we plot the ratio of ranking efficiency of KLSH and SQ-RFF over SignRFF. In Figure 11, Figure 12 and Figure 13, we plot the ratio at a wide range of $\rho = 0.01 \sim 0.95$ and $c = 0.95, 0.7, 0.5$. We see that the ratio SignRFF/SQ-RFF is always larger than 1, i.e., SignRFF is always more efficient than SQ-RFF. At small γ, KLSH is more efficient (ratio SignRFF/KLSH less than 1), while for larger γ, SignRFF is better. This is consistent with the result presented in the main paper.

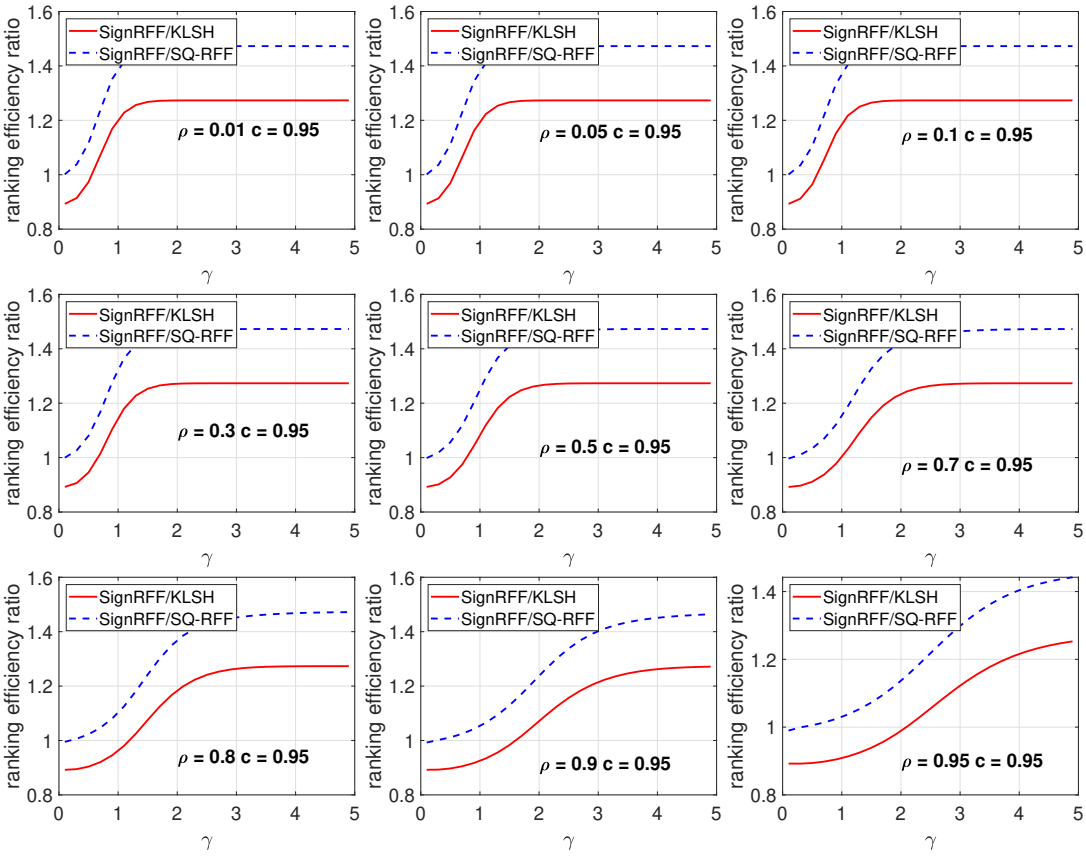

Figure 11: Ranking efficiency ratio against Gaussian kernel parameter γ at various ρ. $c = 0.95$.

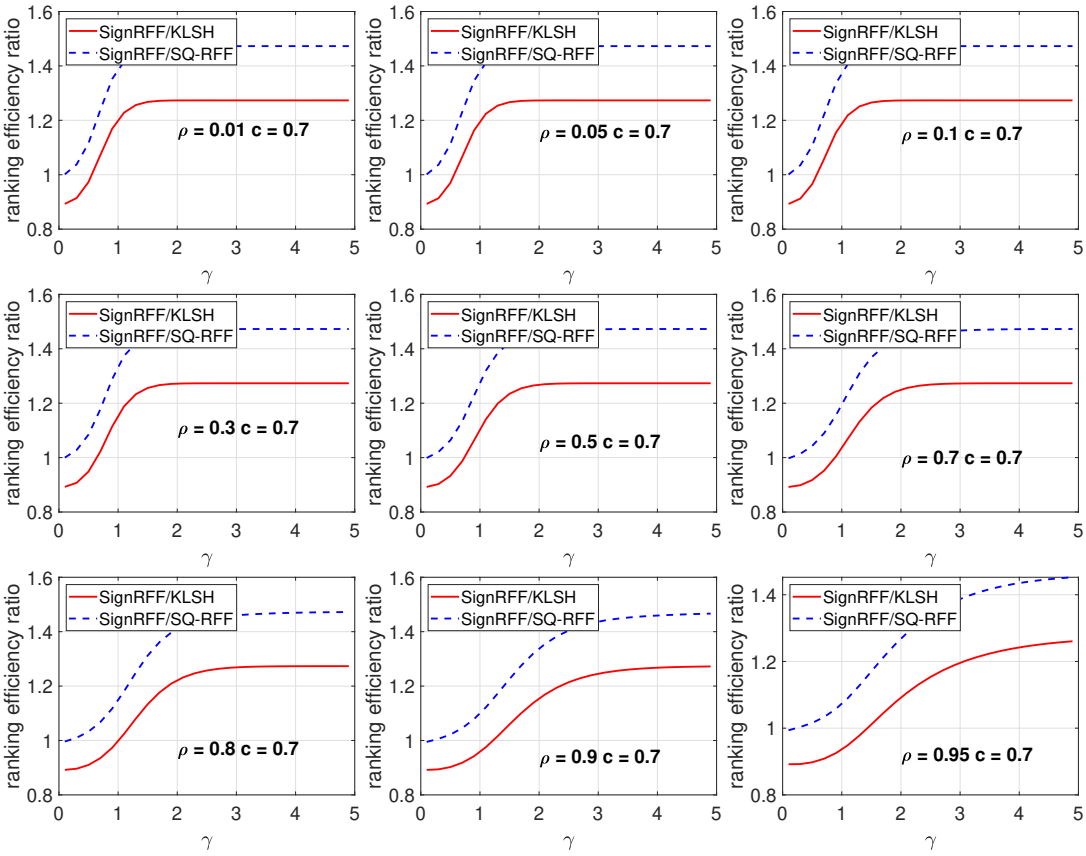

Figure 12: Ranking efficiency ratio against Gaussian kernel parameter γ at various ρ. $c = 0.7$.

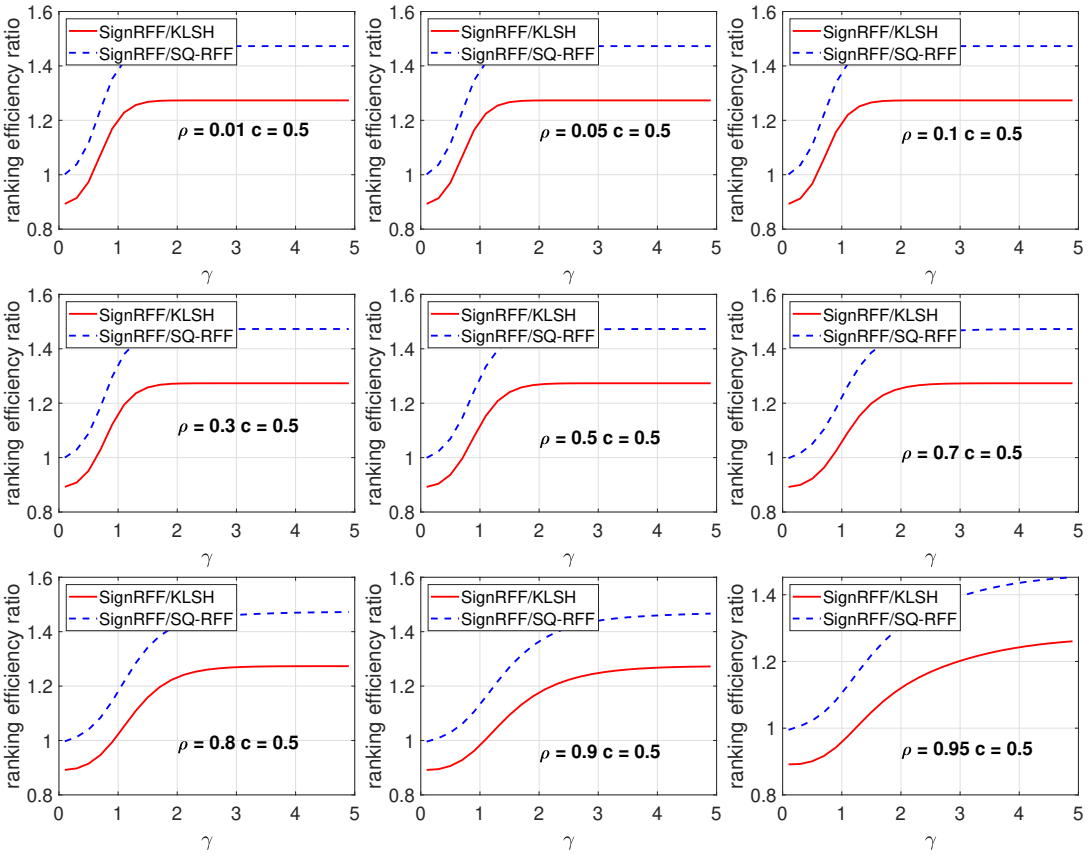

Figure 13: Ranking efficiency ratio against Gaussian kernel parameter γ at various ρ. $c = 0.5$.

C More Image Retrieval Results on CIFAR-VGG

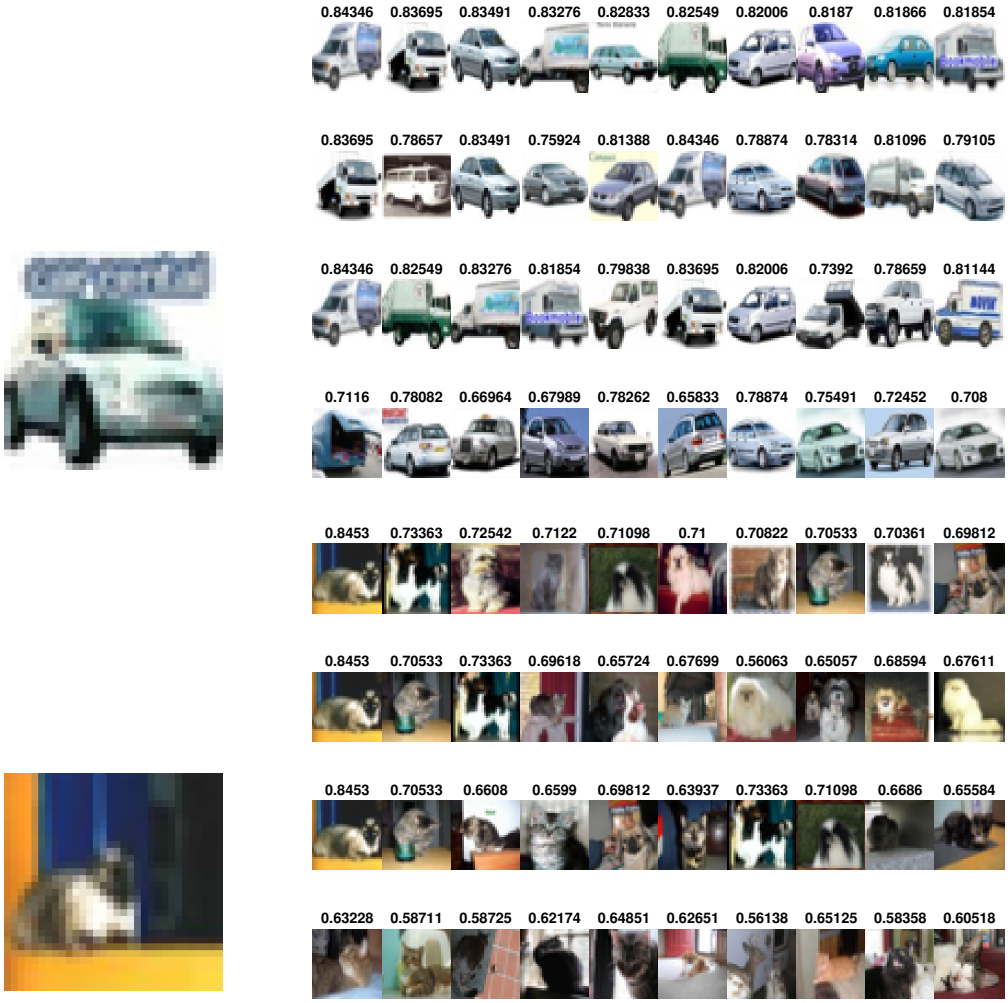

Figure 14: CIFAR-VGG Top-10 retrieved images (right) for two example query images (left, automobile and cat) with $b = 512$. **1st row:** true nearest neighbors in terms of cosine similarity of the features extract from the last fc layer of VGG-16. **2nd row:** SignRFF. **3rd row:** KLSH. **4th row:** CIB. The number on each retrieved image is the cosine similarity of its VGG-feature to the VGG-feature of the query image.

In Section 5.1, we compare the locality-sensitive hashing methods with a deep learning based method, the Contrastive Information Bottleneck (CIB) [50]. CIB is built upon the pre-trained VGG-16 CNN model, whose objective is to generate binary representations such that images from the same class are close. Thus, same as many other deep hashing methods, the objective of CIB is slightly different from our setting (we find the most similar data points, without any label information), as being in the same class does not necessarily implies high similarity. In fact, in the empirical evaluation of many of these papers (e.g., [66, 50]), the true neighbors are simply set as those data points with same label as the query. In our setting, we strictly follow the ranking to find the true neighbors with highest similarities. To better illustrate the difference, we present top-10 retrieval result of two queries (automobile and cat) in Figure 14. For SignRFF and KLSH, the retrieved images mostly have high similarity with the query, but may include some data points from other classes (e.g., truck vs. automobile, dog vs. cat). On the contrary, the retrieved images of CIB clearly have lower similarity with the query, but contain fewer other classes. This is largely because the VGG-16 is pre-trained by classification. Hence, in

this experiment we set the true neighbors as the top-1000 similar data points, following the original paper [50], where CIB could be evaluated properly since most images from the same class would, at least, have not too low similarity.

D Impact of the Dependence in KLSH

In practical implementation, the hash codes of KLSH are dependent. We give an intuitive explanation on how this dependence affect the search performance. Firstly, let us review the reason why data-independent methods with independent codes (LSH, SQ-RFF, SignRFF) can boost search accuracy with increasing b. Essentially, same as the intuition of the rank efficiency, comparing Hamming distance is equivalent to searching for the data points with highest estimated hash collision probability using b codes. Let $\boldsymbol{x}, \boldsymbol{y}$ be two database points and $\boldsymbol{q}$ be the query, with $\rho_x > \rho_y$. This means the hash collision probability $p_x > p_y$. We estimate the probability by averaging the collision indicators:

$$\hat{p}_x = \frac{1}{b}\sum_{i=1}^{b} \mathbb{1}\{h_i(\boldsymbol{x}) = h_i(\boldsymbol{q})\}, \qquad \hat{p}_y = \frac{1}{b}\sum_{i=1}^{b} \mathbb{1}\{h_i(\boldsymbol{y}) = h_i(\boldsymbol{q})\}. \tag{12}$$

For data-dependent methods, with sufficient b, (10) gives the probability of wrong ranking (i.e., $\hat{p}_x < \hat{p}_y$), strictly decreasing with b.

Recall the steps of KLSH. We first sample m data points from database $\boldsymbol{X}$, denoted as $\tilde{\boldsymbol{X}}$, to form a kernel matrix $\boldsymbol{K}$. Then we uniformly pick t points, denoted as $\tilde{\boldsymbol{X}}'$, from $[1, ..., m]$ at random to approximate the Gaussian distribution. After some algebra, the hash code has the form

$$\textbf{KLSH:} \qquad h(\boldsymbol{x}) = sign(\sum_{i=1}^{m} \boldsymbol{w}(i) k(\boldsymbol{x}, \boldsymbol{x}_i)), \tag{13}$$

where $\boldsymbol{w} = \boldsymbol{K}^{-1/2}\boldsymbol{e}_t$, and $\boldsymbol{e}_t \in \{0,1\}^m$ has ones in the entries with indices of the t selected points. To generate multiple codes, we use the same pool of points $\tilde{\boldsymbol{X}}$, but with different choice of e_t, i.e., $\tilde{\boldsymbol{X}}'$ to approximate the Gaussian distribution.

Boosted performance with small b. For two hash codes h_k and h_{k-1} of $\boldsymbol{x}$, the $k(\boldsymbol{x}, \boldsymbol{x}_i)$ terms in (13) are the same. The $\boldsymbol{w}$'s are dependent since the points used for Gaussian approximation (i.e. $\tilde{\boldsymbol{X}}'$) may overlap. Thus, the conditional hash collision probability is usually larger than the unconditional one,

$$P(h_k(\boldsymbol{x}) = h_k(\boldsymbol{y}) | h_{k-1}(\boldsymbol{x}) = h_{k-1}(\boldsymbol{y})) > P(h_k(\boldsymbol{x}) = h_k(\boldsymbol{y})),$$

and it may increase as k get larger (intuitively, by dependence, more previous collisions implies higher chance of collision later on). Similarly,

$$P(h_k(\boldsymbol{x}) \neq h_k(\boldsymbol{y}) | h_{k-1}(\boldsymbol{x}) \neq h_{k-1}(\boldsymbol{y})) > P(h_k(\boldsymbol{x}) \neq h_k(\boldsymbol{y})).$$

Thus, at the beginning, the more similar (dissimilar) the data pair is, the more the estimator $\hat{p}$ is upward (downward) biased. In other words, similar points would have further increased hash collision probability, while dissimilar points would have even lower chance of collision. This is the main reason that the empirical performance of KLSH is higher than theoretical prediction (where we assume independence) with short binary codes.

Slow improvement with large b. However, this is not the whole story. As more bits are used, there is less and less marginal information left. That is, the later generated codes would be more and more dependent on the previous ones. In an extreme case, at the point when all $\binom{m}{t}$ combinations of the t points in $\tilde{\boldsymbol{X}}'$ have been used, the hash codes produced afterwards would all be the same as some previously generated ones, which would hardly improve the distance estimation anymore. This is why for KLSH, the recall curve becomes flat as b increases.