What Makes A Good Code? A New Look At LSH From Random Fourier Features

Anonymous Author(s)
Affiliation
Address
email

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 ask: what makes a good LSH binary code? We answer the question by investigating a new 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 LSH 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 aligns well with the empirical search performance.

1 Introduction

Developing efficient machine learning algorithms in large-scale problems has been an important research topic to deal with massive data. In this paper, we focus on efficient retrieval/search methods, specifically, by designing similarity-preserving binary representations of the data. 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 [45, 11, 19, 13, 27, 28, 29, 41, 50, 46]. 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 [19, 11, 2, 37, 48, 12, 30, 34], where we conduct exhaustive search in the Hamming space. By the built-in CPU hardware instructions, this is much faster than computing the exact Euclidean distances. Moreover, techniques such as multi-index hashing [31, 32] can further accelerate exact Hamming search by orders of magnitude.

Review of Existing Methods. In general, binary hashing methods can be categorized into supervised and unsupervised approaches. In this work, we focus on the unsupervised setting. Locality-Sensitive Hashing (LSH) [14] is one classical and popular hashing scheme that includes "linear LSH" method [3] and "non-linear LSH" methods [20, 35, 16]. The linear LSH that targets at cosine similarity [3], also known as SimHash, generates binary hash codes using random hyperplanes by taking the sign of the *data-independent* random Gaussian projections. Later, many works have focused on learning *data-dependent* binary hash codes, through different objective functions. Examples include Iterative Quantization (ITQ) [11], Spectral Hashing (SpecH) [45] and Binary Reconstruction Embedding (BRE) [19]. Recently, some unsupervised deep learning based methods have been proposed,

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

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 [28, 9, 6, 25, 47, 12, 30, 34]. By taking advantage of the complicated model structures (e.g., CNN layers), these deep methods show promising performance in many image retrieval tasks [47, 34]. Compared with the data-dependent methods, LSH has three 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 degradation when we increase the code length b [35, 17]. On the other hand, the search performance of the data-independent LSH would keep boosting with larger b. In many scenarios, it is often the case that only using short codes (e.g., ≤ 128 bits) cannot achieve a desirable level of search accuracy for practical purposes. In these cases, using longer LSH codes could be more favorable. Secondly, 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, it is hard to analyze the error of data-dependent methods, while LSH enjoys rigorous guarantees on the retrieval performance. With the above merits, LSH has been a popular hashing method with many practical applications for decades [39, 24, 33, 10, 5, 51, 21, 23, 8, 4].

LSH from Random Fourier Features. Kernel methods have gained great success in machine learning [38, 53, 1, 42]. However, standard kernel methods require complexity at least quadratic in the sample size, which is very expensive on large-scale data. [36] proposed the method of random Fourier feature (RFF), which defines a feature map that approximates shift-invariant kernels by the linear inner products. This leads to many applications in large-scale learning where one trains linear models on RFF to approximate non-linear kernel machines [49, 7, 43, 1, 42, 44]. One natural step is to extend RFF to retrieval tasks. In an earlier work, [35] proposed to construct binary codes from RFF using stochastic binary quantization, which we call "SQ-RFF". This "non-linear LSH" is simple: after RFF is generated, we add a uniform noise and then binarize. The authors showed that SQ-RFF can achieve competitive search accuracy with sufficient code length b, e.g., $b \geq 512$.

The main **contributions** of this work are summarized as follows:

- **Simpler hashing from RFF.** Since its first proposal, SQ-RFF [35] has been regarded as the standard method of hashing with RFF (over 750 citations since 2010). However, digging into the code design, immediate questions arise: *is the random noise truly necessary?* In Section 3, we revisit the problem of generating binary codes from RFF, and propose the simpler strategy called "Sign-RFF" that removes the random noise from SQ-RFF. We show the locality-sensitiveness of Sign-RFF as a valid (non-linear kernel based) LSH method.
- **A new metric for LSH comparison.** The next question we ask is, *which LSH scheme is better?* More generally, *what makes a good LSH code?* In Section 4, we study a new measure called *ranking efficiency* as a general scheme for comparing LSH codes. Under this metric, the proposed SignRFF is uniformly better than SQ-RFF, justifying its advantage theoretically. Moreover, the ranking efficiency comparison suggests that, typically one should only prefer non-linear LSH over linear LSH when the near neighbors are close to each other, a result of independent interest. To our knowledge, this is the first such comparison in literature of linear vs. non-linear LSH, from a theoretical viewpoint.
- **Comprehensive empirical justification.** In Section 5 we conduct experiments on benchmark datasets. First, our results verify the superiority of SignRFF over SQ-RFF in search accuracy in all cases. Also, SignRFF outperforms various competing methods (including deep learning methods) with a moderately large number of hash bits, indicating its advantages in practical applications. Second, our experiments validate that, the proposed ranking efficiency metric aligns very well with the empirical search accuracy. Thus, it could serve as an informative measure for theoretically comparing different LSHs in practice.

2 Background

We first recap LSH and another non-linear kernel based method called KLSH.

2.1 Locality-Sensitive Hashing (LSH)

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 [14], 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. $\boldsymbol{x}, \boldsymbol{y}$ 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)**.** *Assume $\delta > 0$ is a parameter. 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 $\boldsymbol{q}$.*

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

Definition 3 ([14])**.** *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. For a data vector $\boldsymbol{x} \in \mathbb{R}^d$, the simplest linear LSH code associated with the cosine similarity [3] is given by

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

where $\boldsymbol{w}$ is a standard Gaussian random vector. We use b i.i.d. $\boldsymbol{w}_1, ..., \boldsymbol{w}_b$ to generate b LSH binary codes. The collision probability between the codes of two data points $\boldsymbol{x}, \boldsymbol{y}$ is

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

where $\rho = \cos(\boldsymbol{x}, \boldsymbol{y})$. Note that $P(h_{LSH}(\boldsymbol{x}) = h_{LSH}(\boldsymbol{y}))$ is increasing in ρ, which, by Definition 3, is the key to ensure the locality sensitivity.

2.2 Kernelized LSH (KLSH)

In this paper, we will consider the popular Gaussian kernel function defined for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ as $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\frac{-\gamma^2 \|\boldsymbol{x} - \boldsymbol{y}\|^2}{2}\right)$, 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 LSH, [20] proposed Kernelized Locality-Sensitive Hashing (KLSH) by applying LSH (1) in the kernel induced feature space $\mathcal{F}$, i.e., $h(\boldsymbol{x}) = sign(\boldsymbol{w}^T \Psi(\boldsymbol{x}))$. However, 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

$$\textbf{KLSH:} \quad h_{KLSH}(\boldsymbol{x}) = sign(\sum_{i=1}^{m} \boldsymbol{w}(i) k(\boldsymbol{x}, \boldsymbol{x}_i)), \tag{3}$$

where $\boldsymbol{w} = \boldsymbol{K}^{-1/2} \boldsymbol{e}_t$, and $\boldsymbol{e}_t \in \{0, 1\}^m$ has ones at the entries with 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 practical implementation. It is observed that the performance of KLSH also drops as b increases [17], similar to many data-dependent methods. See Appendix D for more explanation.

3 Locality-Sensitive Binary Codes From Random Fourier Features

Random Fourier feature (RFF) was proposed in [36] to alleviate the computational bottleneck of standard kernel methods. For a data vector $\boldsymbol{x} \in \mathbb{R}^d$, the RFF for the Gaussian kernel is defined as

$$\textbf{RFF:} \quad F(\boldsymbol{x}) = \cos(\boldsymbol{w}^T \boldsymbol{x} + \tau), \tag{4}$$

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. For good approximation, 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.

From now on, we will assume the data vectors are normalized to have unit l_2 norm: 1) this is a standard pre-processing step in practice; and 2) the theoretical guarantees of LSH (1) is only valid for the cosine similarity (i.e., normalized data). Nevertheless, our analysis and results can be easily extended to the general unnormalized case. With normalized data, the Gaussian kernel can be written as a function of $\rho = \cos(\boldsymbol{x}, \boldsymbol{y})$ as $k(\rho) \equiv k(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma^2(1-\rho)}$. In this setting, the probability distribution of RFF is given as follows.

Lemma 1 ([26]). *For two normalized data points $\boldsymbol{x}, \boldsymbol{y}$ with cosine ρ, let $F(\cdot)$ be the RFF defined as (4). The joint distribution of $z_x = F(x)$ and $z_y = F(y)$ is*

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

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, [35] 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):

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

where ξ is a random perturbation from $Unif(-1, 1)$. Effectively, the so-called SQ-RFF applies a sampling procedure: we first compute the RFF $z = \cos(\boldsymbol{w}^T\boldsymbol{x} + \tau)$, and then assign it to 1 (otherwise -1) with probability $\frac{|1+z|}{2}$. Similar stochastic quantization has also been used in [52, 26] for large-scale low-precision training. The collision probability of "SQ-RFF" is given below.

Theorem 1 ([35]). *For SQ-RFF (5), for normalized $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ with $\cos(\boldsymbol{x}, \boldsymbol{y}) = \rho$, it holds that*

$$P_{SQ}(\rho) := P(h_{SQ}(\boldsymbol{x}) = h_{SQ}(\boldsymbol{y})) = 1 - \frac{8}{\pi^2}\sum_{s=1}^{\infty}\frac{1 - e^{-\gamma^2 s^2(1-\rho)}}{4s^2 - 1}. \tag{6}$$

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 (6) 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 Sign-RFF: Why Not Drop the Noise?

The SQ-RFF has been a standard approach for constructing binary codes from RFF for over a decade. Yet, why is an additional perturbation added to the RFF before binarization? Is it 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

$$\textbf{SignRFF:} \quad h_{sign}(\boldsymbol{x}) = sign(F(x)) = sign(\cos(\boldsymbol{w}^T\boldsymbol{x} + \tau)). \tag{7}$$

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 (7) 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 [26]. 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 What Makes A Good LSH Code?

Next, we compare SignRFF with SQ-RFF by introducing a novel metric called the *ranking efficiency* which allows us to 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 LSH with non-linear LSH methods systematically (e.g., when is non-linear LSH advantageous?) which has not been studied in literature before. We first provide some analysis of KLSH introduced in Section 2.2 which will also be included in our comparison.

Collision probability of KLSH. In our theoretical analysis, we assume for KLSH [20] that the Gaussian projection in the kernel induced feature space $\mathcal{F}$ is truly random (recall in practice we use data dependent approximations in the kernel induced feature space). That is, KLSH "ideally" performs linear LSH 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.* $k(\cdot)$.

4.1 Ranking Efficiency

Recall Definition 3 of the $(\tilde{k}, c\tilde{k}, p_1, p_2)$-LSH. It is known [14] 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 [39]. However, since the LSH efficiency is only based on a worst case analytical bound, it may not well explain/predict the practical search performance. See Appendix B for related discussion.

We now introduce the concept of *ranking efficiency*, 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. Let $V = E(1-E)$, $V_c = E_c(1-E_c)$. The (ρ, c)-ranking efficiency (RE) is defined as $RE = \frac{E-E_c}{\sqrt{V+V_c}}$.*

Remark 2. *In most cases, we may care more about large c values (e.g., $c = 0.95$) which corresponds to similar data points. As shown in Appendix B.1, the covariance V_{xy} in (10) in general does not impact the relative RE comparison, so we drop it from the definition for simplicity and consistency.*

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), (6) and (8), and $V_x = E_x(1-E_x)$, $V_y = E_y(1-E_y)$ from the binomial distribution, respectively.

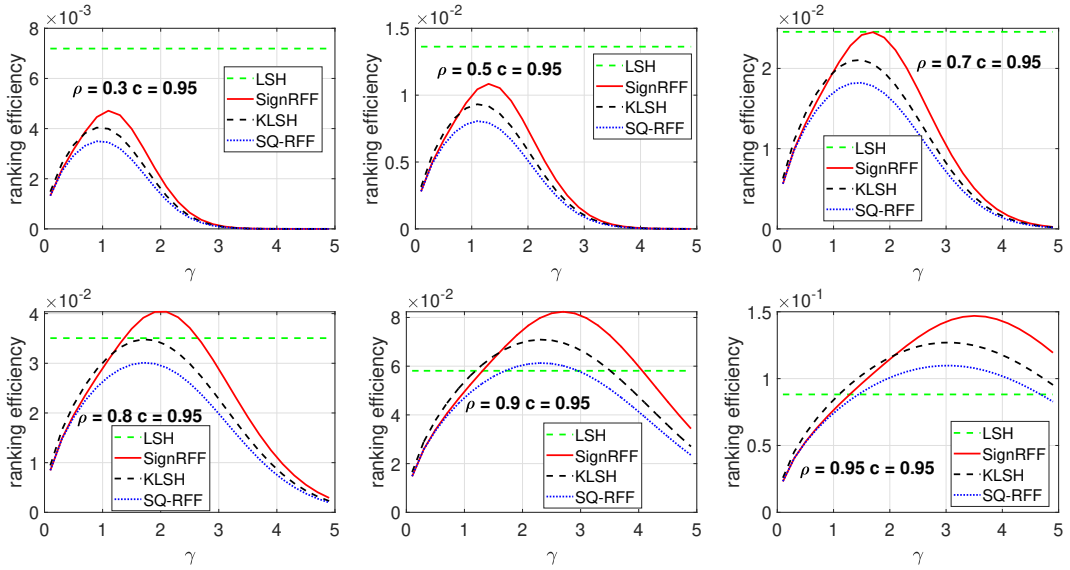

Figure 1: Ranking efficiency (Definition 4) of different LSH methods with various ρ, at $c = 0.95$.

4.2 Analytical Comparison of LSH Methods

Now, we leverage Definition 4 to theoretically compare different LSH methods. In Figure 1, we provide the RE of (linear) LSH, 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.

- **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 linear LSH. However, if the target ρ is small (e.g., $\rho < 0.7$), linear LSH becomes more favorable, even for best tuned non-linear LSH. In other words, SignRFF is better than LSH 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 SIFT1M dataset [15] contains 1M 128-dimensional SIFT image features, and 1000 query samples. The MNIST dataset [22] contains 60000 handwritten digits. The CIFAR dataset [18] 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 [40] based deep methods, following prior literature (e.g.,[47, 34]), 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 2, 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.

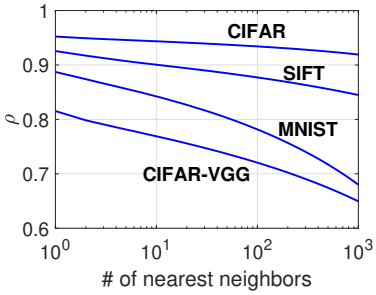

Figure 2: Average ρ to the N-th neighbor on four datasets.

Methods and Evaluation. We compare the following unsupervised hashing methods: 1) **LSH** [3], defined by (1) with random Gaussian projections; 2) **Iterative Quantization (ITQ)** [11], which finds rotations to minimize the quantization error of mapping the randomly projected data to binary; 3) **Spectral Hashing (SpecH)** [45], which is based on quantizing the values of analytical eigenfunctions computed along principal component directions of the data; 4) **Binary Reconstruction Embedding (BRE)** [19], 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 [19]; 5) **KLSH** [20] as in (3), 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 [20]; 6) **SQ-RFF** [35] given in (5), binary codes from stochastically quantized RFF; 7) Our proposed **SignRFF** method (7) with deterministic quantization from RFF. For LSH-family 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 3, 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

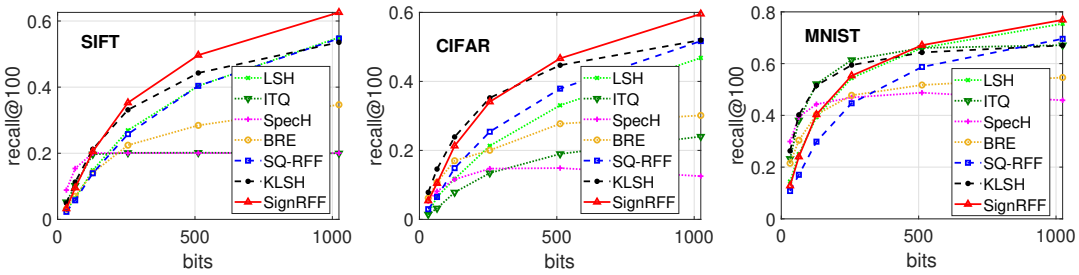

Figure 3: Recall@100 vs. b. Note that in our case, recall@100 is equivalent to precision@100.

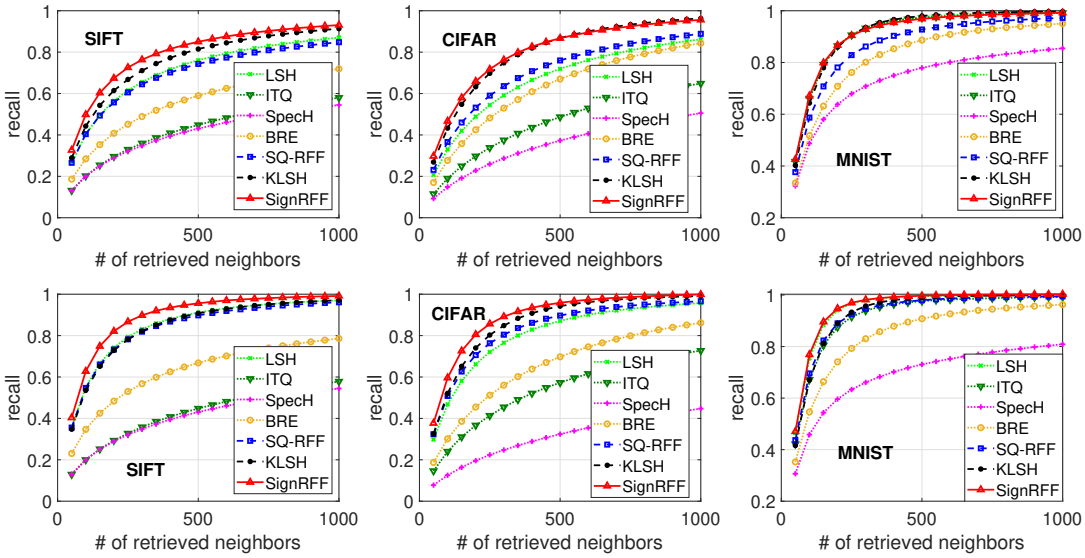

Figure 4: Recall@R vs. # of retrieved neighbors R. **1st row:** $b = 512$. **2nd row:** $b = 1024$.

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 4, 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) [34], which uses VGG-16 pre-trained model as the backbone. We apply the same training setting as in [34]. 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 [34] (see Appendix C for detailed discussion). We observe:

- From Figure 5, CIB performs the best when $b \leq 64$, illustrating the benefit of deep hashing with short codes. Yet, the recall is only 0.3$\sim$0.4, and does not improve with more bits. When $b \geq 256$, LSH, SignRFF and KLSH provide much higher recall than CIB.

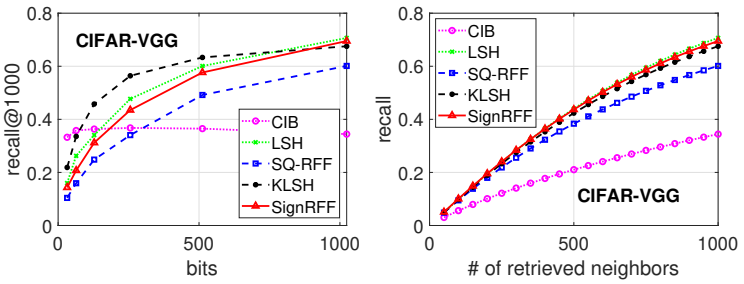

Figure 5: **Left:** Recall@1000 (precision@1000) vs. b. **Right:** Recall vs. # retrieved points, $b = 1024$.

- 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 2, 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 1 suggests that compared with SQ-RFF (blue), LSH (light green) would perform better on MNIST and CIFAR-VGG, similarly on SIFT, and worse on CIFAR. Here we use SQ-RFF vs. LSH as an example; similar comparison holds for other methods. This aligns very well with the recall curves in Figure 3 and Figure 4. To make a more detailed justification, in Figure 6, we additionally provide the recall against the Gaussian parameter γ. In Figure 1, 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 their 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 simple LSH in high similarity region.

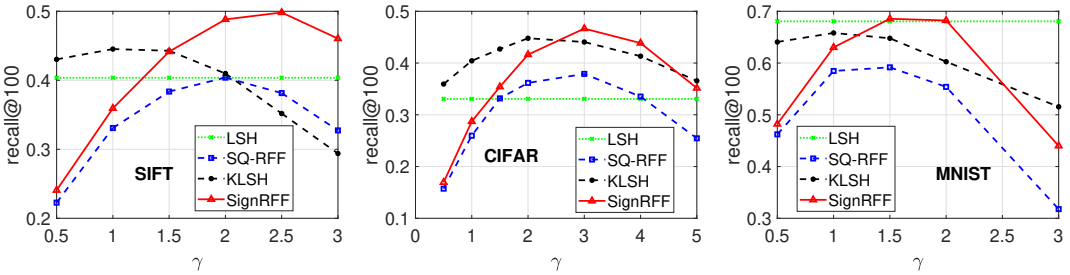

Figure 6: Recall@100 (precision@100) vs. γ, with $b = 512$.

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 [35]. 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 answer the question "what makes a good LSH code?" by designing a new unified theoretical framework to compare different LSH 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 simple LSH 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.

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] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. Deep cauchy hashing for hamming space retrieval. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1229–1237. Computer Vision Foundation / IEEE Computer Society, 2018.

[3] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 380–388, Montreal, Canada, 2002.

[4] 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 *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[5] 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 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 10044–10054, 2019.

[6] Zhixiang Chen, Xin Yuan, Jiwen Lu, Qi Tian, and Jie Zhou. Deep hashing via discrepancy minimization. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 6838–6847. Computer Vision Foundation / IEEE Computer Society, 2018.

[7] 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.

[8] 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 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2319–2329. PMLR, 2021.

[9] 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 *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4217–4226. IEEE Computer Society, 2017.

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

[11] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pages 817–824. IEEE Computer Society, 2011.

[12] 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 2020, Virtual Event, China, July 25-30, 2020*, pages 2009–2012. ACM, 2020.

[13] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 2938–2945. IEEE Computer Society, 2013.

[14] 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.

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

[16] Ke Jiang, Qichao Que, and Brian Kulis. Revisiting kernelized locality-sensitive hashing for improved large-scale image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4933–4941. IEEE Computer Society, 2015.

[17] Alexis Joly and Olivier Buisson. Random maximum margin hashing. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pages 873–880. IEEE Computer Society, 2011.

[18] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[19] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, pages 1042–1050. Curran Associates, Inc., 2009.

[20] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137, 2009.

[21] Dung D. Le and Hady W. Lauw. Stochastically robust personalized ranking for LSH recommendation retrieval. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 4594–4601. AAAI Press, 2020.

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

[23] 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.

[24] 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, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 635–644. ACM, 2017.

[25] Shuyan Li, Zhixiang Chen, Jiwen Lu, Xiu Li, and Jie Zhou. Neighborhood preserving hashing for scalable video retrieval. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 8211–8220. IEEE, 2019.

[26] Xiaoyun Li and Ping Li. Quantization algorithms for random fourier features. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6369–6380. PMLR, 2021.

[27] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen. Deep learning of binary hash codes for fast image retrieval. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2015, Boston, MA, USA, June 7-12, 2015*, pages 27–35. IEEE Computer Society, 2015.

[28] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2064–2072. IEEE Computer Society, 2016.

[29] Hong Liu, Rongrong Ji, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Cross-modality binary code learning via fusion similarity hashing. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6345–6353. IEEE Computer Society, 2017.

[30] 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 2020, Virtual Event, China, July 25-30, 2020*, pages 1379–1388. ACM, 2020.

[31] Mohammad Norouzi, Ali Punjani, and David J. Fleet. Fast search in hamming space with multi-index hashing. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 3108–3115. IEEE Computer Society, 2012.

[32] Eng-Jon Ong and Miroslaw Bober. Improved hamming distance search using variable length hashing. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2000–2008. IEEE Computer Society, 2016.

[33] 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.

[34] 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 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 959–965. ijcai.org, 2021.

[35] 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.

[36] 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.

[37] 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.

[38] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

[39] 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.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[41] 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, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 394–401. AAAI Press, 2018.

[42] 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.

[43] 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.

[44] 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.

[45] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *NIPS*, 2008.

[46] 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.

[47] 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.

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

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] Ji Zhu and Trevor Hastie. Kernel logistic regression and the import vector machine. In *NIPS*, pages 1081–1088, Vancouver, BC, Canada, 2001.

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]