Revisiting Kernelized Locality-sensitive Hashing for Improved Large-scale Image Retrieval

Ke Jiang, Qichao Que, Brian Kulis
Department of Computer Science and Engineering, Ohio State University.

We present a simple but powerful reinterpretation of kernelized locality-sensitive hashing (KLSH) [2], a general and popular method developed in the vision community for performing approximate nearest-neighbor searches in an arbitrary reproducing kernel Hilbert space (RKHS). Our new perspective is based on viewing the steps of the KLSH algorithm in an appropriately projected space, and has several key theoretical and practical benefits. This new perspective gracefully resolves the "infinite Gaussian" issue and provides us with the first explicit performance bounds to clearly demonstrate tradeoffs between runtime and retrieval accuracy. Crucially, this tradeoff also reveals two potential techniques which boost the empirical performance of vanilla KLSH. In particular, we show how to modify KLSH to obtain improvements in recall performance of at least 12%, and sometimes much higher, on all benchmarks examined.

Given a set of n database samples $\{\boldsymbol{x}_1,\dots,\boldsymbol{x}_n\} \in \mathbb{R}^d$, query $\boldsymbol{q} \in \mathbb{R}^d$ and a normalized kernel function $\kappa(\cdot,\cdot) = \langle\Phi(\cdot),\Phi(\cdot)\rangle \in [0,1]$ with the feature map $\Phi : \mathbb{R}^d \to \mathcal{H}$, where $\mathcal{H}$ is the RKHS, we are interested in finding the most similar item in the database to the query $\boldsymbol{q}$ with respect to $\kappa(\cdot,\cdot)$, i.e., $\mathrm{argmax}_i \kappa(\boldsymbol{q},\boldsymbol{x}_i)$. Considering $\{\Phi(\boldsymbol{x}_1),\dots,\Phi(\boldsymbol{x}_t)\}$ as t realizations of random variable $\Phi(X)$ with known mean μ and covariance operator C, the classical CLT ensures that the random vector $C^{-1/2}\tilde{z}_t = C^{-1/2}[\sqrt{t}(\frac{1}{t}\sum_{i=1}^{t}\Phi(\boldsymbol{x}_i) - \mu)]$ converges to a standard Gaussian random vector as $t \to \infty$. Therefore, the LSH hash family can be approximated by [2]:

$$h(\Phi(x)) = \begin{cases} 1, & \text{if } \Phi(x)^T C^{-1/2}\tilde{z}_t \geq 0 \\ 0, & \text{otherwise} \end{cases}. \tag{1}$$

Unfortunately, there is no such canonical Gaussian distribution in an infinite-dimensional RKHS, as given by the following lemma.

Lemma 1. *[3] A Gaussian distribution with covariance operator C in a Hilbert space exists if and only if, in an appropriate base, C has a diagonal form with non-negative eigenvalues and the sum of these eigenvalues is finite.*

New Interpretation. Utilizing the eigen-decomposition of the covariance C, we can write

$$g(\Phi(\boldsymbol{x})) = \sum_{i=1}^{d_\Phi} \frac{1}{\sqrt{\lambda_i}}(\boldsymbol{v}_i^T\Phi(\boldsymbol{x}))\cdot(\boldsymbol{v}_i^T\tilde{z}_t) = \sum_{i=1}^{d_\Phi}(\boldsymbol{v}_i^T\Phi(\boldsymbol{x}))\cdot\left(\frac{1}{\sqrt{\lambda_i}}\boldsymbol{v}_i^T\tilde{z}_t\right), \tag{2}$$

where $h(\Phi(\boldsymbol{x})) = \mathrm{sign}[g(\Phi(\boldsymbol{x}))]$, $\lambda_1 \geq \dots \geq \lambda_m \geq \dots \geq 0$ are the eigenvalues of C with $\boldsymbol{v}_i$ the corresponding eigenvectors. In many situations, the dimension d_Φ of Φ is infinite. If we perform a truncation at k, we obtain a finite-dimensional representation for Φ. The resulting sum in (2) after this truncation can be viewed as an inner product between two k-dimensional vectors: the first vector is the data point $\Phi(\boldsymbol{x})$ after projecting via KPCA, and the second vector is a Gaussian random vector. We can therefore see that KLSH performs LSH after projecting to an (m-1)-dimensional space via principal component analysis in $\mathcal{H}$.

Thus, we can present a theoretical analysis of KLSH via the "KPCA+LSH" perspective. For simplicity, we assume truly random Gaussian vectors are used.

Theorem 2. *Consider an n-sample database $S = \{\boldsymbol{x}_1,\dots,\boldsymbol{x}_n\}$ and a query point $\boldsymbol{q}$. For any $\varepsilon,\xi > 0$, with success probability at least $(1-e^{-\xi})/2$ and query time dominated by $O(n^{\frac{1}{1+\varepsilon}})$ kernel evaluations, the KLSH algorithm retrieves a nearest neighbor $\hat{\boldsymbol{y}}_{\boldsymbol{q},k}$ with corresponding bound*

$$\kappa(\boldsymbol{q},\hat{\boldsymbol{y}}_{q,k}) \geq (1+\varepsilon)(1-\sqrt{\lambda_k}-\eta)\kappa(\boldsymbol{q},\boldsymbol{y}_q^*) - \varepsilon - (2+\varepsilon)\left(\sqrt{\lambda_k}+\eta\right)^2, \tag{3}$$

if we only keep those points with $N(\boldsymbol{x}) > 1-\sqrt{\lambda_k}-\eta$ for consideration, where $\eta = \frac{2}{\delta_k\sqrt{m}}\left(1+\sqrt{\frac{\xi}{2}}\right)$ and $0 < \eta < 1-\sqrt{\lambda_k}$.

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.

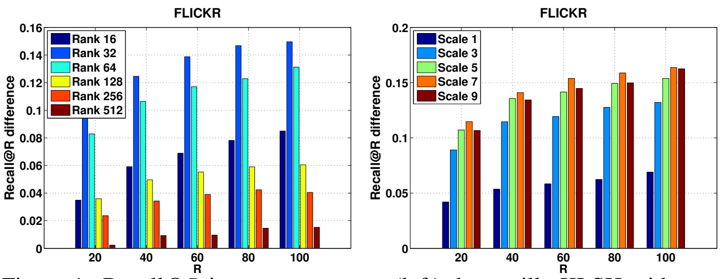

Figure 1: Recall@R improvement over (left) the vanilla KLSH with $m = 1000$ for rank $\in \{16,32,64,128,256,512\}$, (right) the rank-32 KLSH with $m = 1000$ for scale $\in \{1,3,5,7,9\}$ with 256-bit hash code.

Recall@100	Dataset	KLSH	LR	LR+T	IM
χ^2	Flickr	0.3629	0.5125	0.6762	**+0.3133**
	SIFT1M	0.6942	0.7642	0.8213	**+0.1271**
	GIST1M	0.2252	0.3360	0.5260	**+0.3008**
Intersection	Flickr	0.4182	0.5738	0.6529	**+0.2347**
	SIFT1M	0.6397	0.7468	0.7844	**+0.1447**
	GIST1M	0.2746	0.4014	0.4913	**+0.2167**

Table 1: Summary of absolute improvement for Recall@100.

Thus, as k (the number of chosen PCs) and m (the number of points for estimating the eigenspace) become larger at appropriate rates, both $\sqrt{\lambda_k}$ and η will go to zero. Also, with a fixed k, increasing m will always improve the bound, but the $\sqrt{\lambda_k}$ term will be non-zero and will likely yield retrieval errors. This has been empirically shown in [1], namely that the performance of KPCA+LSH saturates when m is large enough, usually in the thousands.

Low-Rank Extension. With a fixed m, there is a trade-off between decreasing λ_k and increasing $1/\delta_k$, similar to the classic bias-variance trade-off in statistics; we expect $1/\delta_k$ to increase as k increases, whereas λ_k will decrease for larger k. In light of this, we introduce a *low-rank extension of KLSH*: we may actually achieve better results by only projecting into a smaller r-dimensional subspace, which can be seen in Figure 1.

Extension via Monotone Transformation. Another relevant factor is the decaying property of the eigenvalues of the covariance operator C, which not only affects the λ_k and δ_k, but also the constant which corresponds to the estimation error. Since monotone transformation of the kernel function will not change the relative ranking, we can explore these transformations that can reward us better performance. For example, the exponential transformation with scale parameter $s > 0$,

$$\kappa_s(\boldsymbol{x},\boldsymbol{y}) = \exp\left(s * (\kappa(\boldsymbol{x},\boldsymbol{y}) - 1)\right). \tag{4}$$

We can see that the ranking of the nearest neighbors stays the same no matter what value we choose for s as long as $s > 0$. However, changing the scaling impacts the eigenvalues of the covariance operator. In particular, it often slows down the decay with $s > 1$ and will eliminate the decay entirely when $s \to \infty$. The effect of the monotone transformation can be seen in Figure 1.

Table 1 summaries the total absolute improvement combining the two techniques together. We can see that the retrieval improvement is at least 12%, sometimes much higher, among all benchmarks.This validates the merit of our analysis regarding the interesting trade-offs shown in our performance bound (3) and demonstrates the power of these simple techniques.

[1] A. Bourrier, F. Perronnin, R. Gribonval, P. Perez, and H. Jegou. Explicit embeddings for nearest neighbor search with Mercer kernels. *Journal of Mathematical Imaging and Vision*, 2015.

[2] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.

[3] M. Lifshits. *Lectures on Gaussian Processes*. Springer, 2012.