Hindawi Publishing Corporation
The Scientific World Journal
Volume 2015, Article ID 350676, 9 pages
http://dx.doi.org/10.1155/2015/350676

Research Article

Fast Image Search with Locality-Sensitive Hashing and Homogeneous Kernels Map

Jun-yi Li[1,2] and Jian-hua Li[1]

[1] *School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*
[2] *Electrical and Computer Engineering Department, National University of Singapore, Singapore 119077*

Correspondence should be addressed to Jun-yi Li; leejy2006@163.com

Received 30 December 2013; Accepted 19 February 2014

Academic Editors: Y.-B. Yuan and S. Zhao

Fast image search with efficient additive kernels and kernel locality-sensitive hashing has been proposed. As to hold the kernel functions, recent work has probed methods to create locality-sensitive hashing, which guarantee our approach's linear time; however existing methods still do not solve the problem of locality-sensitive hashing (LSH) algorithm and indirectly sacrifice the loss in accuracy of search results in order to allow fast queries. To improve the search accuracy, we show how to apply explicit feature maps into the homogeneous kernels, which help in feature transformation and combine it with kernel locality-sensitive hashing. We prove our method on several large datasets and illustrate that it improves the accuracy relative to commonly used methods and make the task of object classification and, content-based retrieval more fast and accurate.

1. Introduction

In Web 2.0 applications era, we are experiencing the growth of information and confronted with the large amounts of user-based content from internet. As each one can publish and upload their information to the internet, it is urgent for us to handle the information brought by these people from internet. In order to organize and be close to these vision data from Internet, it has caused considerable concern of people. Therefore, the task of fast search and index for large video or image databases is very important and urgent for multimedia information retrieval such as vision search especially now the big data in some certain domains such as travel photo data from the website and social network image data or other image archives.

With the growth of vision data, we focus on two important aspects of problem including nearest neighbor search and similarity metric learning. For metric learning, many of the researchers have proposed some algorithms such as Information-Theoretic metric learning [1]. As for nearest neighbors search, the most common situation and task for us is to locate the most similar image from an image database. Among all the methods, given the similarity of example and query item, the most common method is to find all the vision data among the vision database and then sort them. However time complexity of this algorithm is too large and also impractical. When we handle image or video data, especially, this complexity will not be calculated, because it is very difficult for us to compute the distance of two items in higher dimensional space and also vision datum is sparse, so we cannot complete it by limited time.

Many researchers believe that linear scanning can solve this problem; although we believe it is a common approach and not suitable for computing in large-scale datasets, it promotes the development of ANN. LSH was used in ANN algorithms. To get fast query response for high-dimensional space input vectors [1–5], when using LSH, we will sacrifice the accuracy. To assure a high probability of collision for similar objects, randomized hash function must be computed; this is also referred to in many notable locality-sensitive hashing algorithms [6, 7].

Although, in object similarity search task, the LSH has played an important role, some other issues and problems have been neglected. In image retrieval, recognition, and search tasks, we find that they are very common:

(A) in the sample feature space, traditionally LSH approaches can only let us get a relatively high collision probability for items nearby. As a lot of vision datasets contained much rich information, we can find that the category tags are attached to YouTube and Flickr data and the class labels are attached to Caltech-101 images. However the low-level and high-level of vision samples have great gap, which means that the gap low-level features and high-level semantic information exist. To solve this problem, we intend to utilize the side additional information for constructing hash table;

(B) As to manipulate nonlinear data which is linear inseparable, we commonly use kernel method in vision task because of its popularity. For instance, in vision model, objects are often modeled as BOF and kernel trick is an important approach in classifying these data from low-dimension space to high-dimension space. However, how to create hash table in kernel spaces is a tough problem for us.

To verify our idea, we did several experiments in object search task. For example, we show our results on the Caltech-101 [8] dataset and demonstrate that our approach is superior to the existing hashing methods as our proposed algorithm.

In order to test our algorithm performance on dataset, we design some experiments on certain visual task such as Caltech-101 [8] and demonstrate that the performance of algorithm in our paper is beyond the traditional LSH approaches on the dataset, as hash functions can be calculated beyond many kernels. Arbitrary kernel in ANN is suitable in our scheme; actually we can find that a lot of similarity hashing functions can be accessed in the task of vision search tasks based on content retrieval.

2. Homogeneous Kernel

In our paper, we mainly focus on some similar kernels like intersection, Jensen-Shannon, Hellinger's, and χ^2 kernels. In the fields of machine learning and vision search, we often use these kernels as learning kernels. These kernels have two common attributes: being homogeneous and additive. The idea of kernel signature has been smoothly connected to these kernels in this section. Meanwhile we can use pure functions to represent these kernels. Also these attributes will be applied in Section 3 to obtain kernel feature maps. Through the kernel feature map, we can get their approximate expression.

Homogeneous Kernels. A kernel $k_l : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}$ is γ-homogenous if

$$\forall m \geq 0 : k_l(ma, mb) = m^\gamma k_l(a, b). \tag{1}$$

When $\gamma = 1$, we believe that $k_l(ma, mb)$ is homogeneous. Let $m = 1/\sqrt{ab}$; we can obtain a γ-homogeneous kernel and we can also write the formula as

$$\begin{aligned} k_l(a, b) &= m^{-\gamma} k_l(ma, mb) = (ab)^{\gamma/2} k_l\left(\sqrt{\frac{b}{a}}, \sqrt{\frac{a}{b}}\right) \\ &= (ab)^{\gamma/2} \kappa(\log b - \log a). \end{aligned} \tag{2}$$

Here the pure function

$$\kappa(\lambda) = k_l\left(e^{\lambda/2}, e^{-\lambda/2}\right), \quad \lambda \in \mathbb{R} \tag{3}$$

is called the kernel signature.

Stationary Kernels. A kernel k_s: $\mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}$ is called stationary kernels if

$$\forall l \in \mathbb{R} : k_s(l + a, l + b) = k_s(a, b). \tag{4}$$

Let $l = -(a + b)/2$; the $k_s(a, b)$ is represented as

$$\begin{aligned} k_s(a, b) &= k_s(l + a, l + b) \\ &= k_s\left(\frac{b - a}{2}, \frac{a - b}{2}\right) = \kappa(b - a), \end{aligned} \tag{5}$$

$$\kappa(\lambda) = k_s\left(\frac{\lambda}{2}, -\frac{\lambda}{2}\right), \quad \lambda \in \mathbb{R}. \tag{6}$$

Here we call formula (6) kernel feature.

In the field of machine learning or computer vision, most of the homogeneous kernels are composed of the Jensen-Shannon, intersection, χ^2, and Hellinger's kernels. So we can also view them as additive kernels. In the next section, we will focus on these kernels and their kernel maps. Table 1 shows the details [9].

χ^2 Kernel. We define $k(a, b) = 2ab/(a + b)$ as the χ^2 kernel [10, 11]. Here the χ^2 distance is then defined as $D^2(a, b) = \chi^2(a, b)$.

Jensen-Shannon (JS) Kernel. We define $k(a, b) = (a/2)\log_2(a + b)/a + (b/2)\log_2(a + b)/b$ as the JS kernel. Here the JS kernel distance $D^2(a, b)$ can be obtained by $D^2(a, b) = KL(a \mid (a + b)/2) + KL(b \mid (a + b)/2)$, where we import the concept of Kullback-Leibler divergence computed by $KL(a \mid b) = \sum_{l=1}^{d} a_l \log_2(a_l/b_l)$.

Intersection Kernel. We defined $k(a, b) = \min\{a, b\}$ as the intersection kernel [12]. The distance metric $D^2(a, b) = \|a - b\|_1$ is l^1 distance between variants a and b.

Hellinger's Kernel. We defined $k(a, b) = \sqrt{ab}$ as the Hellinger's kernel and specified distance metric $D^2(a, b) = \|\sqrt{a} - \sqrt{b}\|_2^2$ as Hellinger's distance between variants a and b. The function expression $\kappa(\lambda) = 1$ is the signature of the kernel, which is constant.

γ-Homogeneous Parameters. In previous research paper, we can see that the homogeneous kernels are used by parameters $\gamma = 1$ and $\gamma = 2$. When $\gamma = 2$, the kernel becomes $k(a, b) = ab$. Now, in our paper, we can derive the γ-homogeneous kernel by formula (2).

3. Homogeneous Kennel Map

When handling low-dimensional data which is inseparable, we should create kernel feature map $\psi(x)$ for the kernel

Table 1: Common kernels, signature, and their feature maps.

Kernel	$k(a,b)$	Signature $\kappa(\theta)$	$\kappa(\omega)$	Feature $\psi_\omega(a)$
Hellinger's	$\sqrt{ab}$	1	$\delta(\omega)$	$\sqrt{a}$
χ^2	$\frac{2ab}{a+b}$	$\operatorname{sech}\left(\frac{\theta}{2}\right)$	$\operatorname{sech}(\pi\omega)$	$e^{i\omega\log a}\sqrt{a\operatorname{sech}(\pi\omega)}$
Intersection	$\min\{a,b\}$	$e^{-\lvert\theta\rvert/2}$	$\frac{2}{\pi}\frac{1}{1+4\omega^2}$	$e^{i\omega\log a}\sqrt{\frac{2a}{\pi}\frac{1}{1+4\omega^2}}$
JS	$\frac{a}{2}\log_2\frac{a+b}{a}+\frac{b}{2}\log_2\frac{a+b}{b}$	$\frac{e^{\theta/2}}{2}\log_2\left(1+e^{-\theta}\right)+\frac{e^{-\theta/2}}{2}\log_2\left(1+e^{\theta}\right)$	$\frac{2}{\log 4}\frac{\operatorname{sech}(\pi\omega)}{1+4\omega^2}$	$e^{i\omega\log a}\sqrt{\frac{2}{\log 4}\frac{\operatorname{sech}(\pi\omega)}{1+4\omega^2}}$

so that we can map our input data information in low-dimensional space to relatively high-dimensional (Hilbert) information space with $\langle\cdot,\cdot\rangle$:

$$\forall a,b\in R^D : K(a,b)=\langle\psi(a),\psi(b)\rangle. \tag{7}$$

In order to compute the feature maps and get approximate kernel feature maps expression for the homogeneous kernels, we should use Bochner's theorem by expanding the configuration of γ-homogeneous expression. Here we notice that if a homogeneous kernel is Positive Definite [13], its signature will also be Positive Definite expression. The assumption condition is suitable for a stationary kernel. So, depending on formulae (2) and Bochner's theorem (9), we can derive the $k(a,b)$ and closed feature map.

We can compute the kernel density and feature map closed form [9] for most machine learning kernels. Table 1 illustrates the results. Consider

$$\begin{aligned} k(a,b) &= (ab)^{\theta/2}\int_{-\infty}^{+\infty} e^{-i\omega\lambda}\kappa(\omega)\,d\omega, \quad \theta=\log\frac{b}{a} \\ &= \int_{-\infty}^{+\infty}\left(e^{-i\omega\log a}\sqrt{a^\theta\kappa(\omega)}\right)^*\left(e^{-i\omega\log b}\sqrt{b^\theta\kappa(\omega)}\right)d\omega, \end{aligned} \tag{8}$$

$$\psi_w(a)=e^{-i\omega\log a}\sqrt{a^\gamma\kappa(\omega)}. \tag{9}$$

4. Kernelized Locality-Sensitive Hashing

To create and conduct the data association, we take the approach of Kernelized LSH [14] which is also a hash table-based algorithm. KLSH is proposed based on LSH algorithm, which is more efficient and accurate for query search and matching. When searching the input query, the KLSH approach can quickly locate the possible similar and nearest neighbor items in the hash table and match it. In addition, KLSH has another characteristic: traditional LSH methods can only find a part of hashes in the kernel space, while KLSH can locate all the possible hash tables in kernel space. Moreover KLSH has been applied in the vision search tasks by large scale datasets such as Tiny Image and other datasets [14].

Similar to LSH, constructing the hash functions for KLSH has been the key problem for us. That means if we intend to compute the collision probabilities of input query and the database points, we should compute the extent of similarity between them in the database as proposed by [15].

KLSH Principle. Any locality-sensitive hashing algorithm is based on the probability of distribution of hash function clusters. So we should compute the collision probability of a bundle of points, for example, m and n:

$$P_r(h(m)=h(n))=\operatorname{sim}(m,n). \tag{10}$$

We can also view the problem as the issue of computing the similarity of objects between m and n. Here $\operatorname{sim}(m,n)$ in the algorithm is the measure function of calculating the similarity, while $h(m)$ and $h(n)$ are randomly selected from the hash function cluster H. The instinct beyond this is that we find the fact that m and n will collide in the same hash bucket. So those objects which are significantly similar will be more possible to be memorized in the hash table and this eventually results in confliction [1].

We can derive the similarity function expression according to the vector inner product:

$$\operatorname{sim}(m,n)=m^T n. \tag{11}$$

In [15, 16], the definition of LSH function has been extended from formula (10) as

$$h_{\vec{r}}(m)=\begin{cases}1, & \text{if } \vec{r}^T m\geq 0,\\ 0, & \text{else.}\end{cases} \tag{12}$$

Here we create a random hyper plane vector $\vec{r}$. The distribution of $\vec{r}$ fit has a zero-mean multi-Gaussian $N(0,\Sigma_p)$ distribution. The dimensionality of $\vec{r}$ is the same with the input vector m. This demonstrates that the statistical characteristic of input vector is uniquely matched with each hash function.

Meanwhile this verification has been detailedly reported in the LSH attribute in [17]. When we project on a point m, actually the sigh function we obtain in this process is a hash function and then we repeat it k times; a couple of hashes can be created. We can also call this couple of hashes hash bucket. The hash bucket can be formed as

$$g(m) = \langle h_1(m), \ldots, h_t(m), \ldots, h_k(m) \rangle. \quad (13)$$

From (13), we can see that, after repeating k times, we can get one column of hash bucket (14); then repeating b times, we can finally obtain the hash bucket $g(m)$:

$$g_j(m) = \langle h_1(m), \ldots, h_t(m), \ldots, h_k(m) \rangle \quad (1 < j < b). \quad (14)$$

When given the value of b, we can get all the the hash functions located in the bucket; we can see the following:

$$g(m) = \begin{Bmatrix} h_{1,1\vec{r}}(m) & h_{2,1\vec{r}}(m) & \cdots & h_{s,1\vec{r}}(m) & \cdots & h_{t,1\vec{r}}(m) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{1,j\vec{r}}(p) & h_{2,j\vec{r}}(m) & \cdots & h_{s,j\vec{r}}(m) & \cdots & h_{t,j\vec{r}}(m) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{1,b\vec{r}}(m) & h_{2,b\vec{r}}(m) & \cdots & h_{s,b\vec{r}}(m) & \cdots & h_{t,b\vec{r}}(m) \end{Bmatrix}, \quad (1 < j < b; 1 < s < t). \quad (15)$$

Due to the fact that we compute the similarity measure function in high-dimensional kernel space, the similarity function can also be extended and written as

$$\mathrm{sim}\left(\left(m_i, m_j\right)\right) = \kappa\left(m_i, m_j\right) = \phi\left(m_i\right)^T \phi\left(m_j\right). \quad (16)$$

In formula (16), we use kernel function $\phi(m)$ to construct $\kappa(m_i, m_j)$ to complete the kernel mapping for the points of m_i and m_j. And $\phi(m_i)^T\phi(m_j)$ is a product of projection on hash function from the $\mathfrak{R}$ space. The problem is that nothing is known about the data while in kernel space to generate $\vec{r}$ from $N(0, \Sigma_p)$. Therefore, in order to construct the hash function, $\vec{r}$ needs to be created so that we can quickly compute the $\vec{r}^T\phi(m)$ function based on the kernel. Similar to normal $\vec{r}^T$, we could use only the kernel of $\phi(m)$ to approximately compute the function of $\vec{r}^T\phi(m)$. We should select a subset of database to construct $\vec{r}$. By the large number of central limit theory, if we intend to choose parts of database items from the whole database to form the dataset S, the sample of kernel data must be satisfied by the distribution with mean μ and variance Σ. The variable z_a can be written as

$$z_a = \frac{1}{k}\sum_{i \in s} \phi(m_i). \quad (17)$$

With the growth of variable a, the theory tells us that the vector $\tilde{z}_a = \sqrt{t}(z_a - \mu)$ has also been satisfied by the distribution of normal Gaussian.

We used the whitening transform to obtain $\vec{r}$:

$$\vec{r} = \Sigma^{-1/2}\tilde{z}_a. \quad (18)$$

The LSH function has been yielded:

$$h(\phi(m)) = \begin{cases} 1, & \text{if } \phi(m)^T \Sigma^{-1/2}\tilde{z}_a \geq 0, \\ 0, & \text{else.} \end{cases} \quad (19)$$

As analyzed above, we use kernel function to represent the database data; then the statistical data like variance and mean are uncertain. If we intend to estimate and calculate μ and Σ, we could sample the data from the database by KPCA and eigen decomposition in [18] and we let $\Sigma = V\Lambda V^T$ and $\Sigma^{-1/2} = V\Lambda^{-1/2}V^T$; therefore we can obtain the hash function $h(\phi(m))$:

$$h(\phi(m)) = \mathrm{sign}\left(\phi(m)^T V\Lambda^{-1/2}V^T\tilde{z}_a\right). \quad (20)$$

From the above, we can see how to construct the hash function for the kernel matrix input vectors. In this case, we let the kernel matrix input be $K = U\Omega U^T$ by decomposing the K matrix. Here Ω and Λ have the same nonzero eigenvalue; it is also viewed as another form of kernel matrix input. From [18], we compute the projection

$$v_t^T\phi(m) = \sum_{i=1}^{n} \frac{1}{\sqrt{\theta_t}} u_t(i)\, \phi\left(m_i\right)^T \phi(m). \quad (21)$$

Here u_t and v_t are, respectively, the tth eigenvector of the kernel matrix and its covariance matrix.

As mentioned before, we choose n data points from the database to form $\phi(m_i)$; traversing all the t eigenvectors and conducting the computation yields

$$\begin{aligned} h(\phi(m)) &= \phi(m)^T V\Lambda^{-1/2}V^T\tilde{z}_a \\ &= \sum_{t=1}^{n} \sqrt{\theta_t} v_t^T \phi(m)^T v_t^T \tilde{z}_a. \end{aligned} \quad (22)$$

Substituting (21) into (22) yields

$$\sum_{t=1}^{n} \sqrt{\theta_t} \left(\sum_{i=1}^{n} \frac{1}{\sqrt{\theta_t}} u_t(i)\, \phi\left(m_i\right)^T \phi(m) \right) \cdot \left(\sum_{i=1}^{n} \frac{1}{\sqrt{\theta_t}} u_t(i)\, \phi\left(m_i\right)^T \tilde{z}_a \right). \quad (23)$$

(1) Process Data
(a) Obtain K matrix from database throughout the n points.
(b) Obtain e_s by randomly sampling a subset from the $\{1, 2, \ldots, n\}$
(c) Project on ath subset to obtain $h_a(\phi(m))$.
(d) Obtain $w = K^{1/2} \cdot e_s/a$
(e) Project $w(i)$ onto the points in kernel space
(f) Obtain hash bucket $g_j(m)$
(2) Query Processing
(a) Obtain the same hash bucket in (29) from the database
(b) Use Ann search for query matching.

Algorithm 1: KLSH algorithm.

Simplifying (23) yields

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\phi\left(m_i\right)^T\phi\left(m\right)\right) \cdot\left(\phi\left(m_j\right)^T\tilde{z}_a\right)\left(\sum_{t=1}^{n}\frac{1}{\sqrt{\theta_t}}u_t\left(i\right)u_t\left(j\right)\right). \tag{24}$$

Since $K_{ij}^{-1/2} = \sum_{k=1}^{n}(1/\sqrt{\theta_k})u_k(i)u_k(j)$, we further simplify the (24) yields

$$h\left(\phi\left(m\right)\right) = \sum_{i=1}^{n} w\left(i\right)\left(\phi\left(m_i\right)^T\phi\left(m\right)\right), \tag{25}$$

where $w(i) = \sum_{j=1}^{n} K_{ij}^{-1/2}\phi(x_j)^T\tilde{z}_a$.

Through the above derived formula $w(i)$ we can obtain $\vec{r} = \sum_{i=1}^{n} w(i)\phi(x_i)$ which obeys random Gaussian distribution, then we can substitute (17) into $w(i)$:

$$w\left(i\right) = \frac{1}{a}\sum_{j=1}^{n}\sum_{l\in s}K_{ij}^{-1/2}K_{jl}. \tag{26}$$

We neglect the term of $\sqrt{a}$, and finally the simplified $w(i)$ yields (27). e_s represents the unit vector for S.

And therefore hash function for kernel input will finally be

$$w = K^{1/2}\cdot\frac{e_s}{k}, \tag{27}$$

$$h\left(\phi\left(m\right)\right) = \operatorname{sign}\left(\sum_{i=1}^{n} w\left(i\right)\kappa\left(m, m_i\right)\right). \tag{28}$$

κ is the kernel mapping matrix for points m and m_i in space. After several iterations, the hash function will form a hash bucket.

In order to get the suitable parameters in this process, we implement the query matching for several iterations. The detailed algorithm is illustrated finally in Algorithm 1. Consider

$$g_j\left(m\right) = \left[h_1\left(\phi\left(m\right)\right), h_{2,j}\left(\phi\left(m\right)\right), \ldots, h_{t,j}\left(\phi\left(m\right)\right), \ldots, h_{k,j}\left(\phi\left(m\right)\right)\right], \quad \left(1 < l < t\right), \quad \left(1 < j < b\right). \tag{29}$$

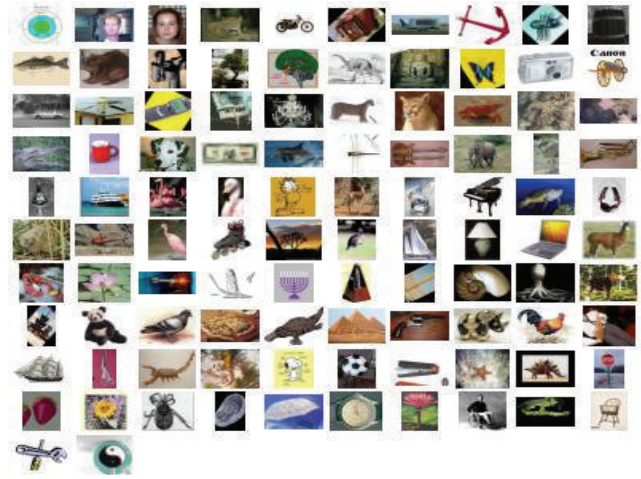

Figure 1: Datasets: Caltech-101 Example.

5. Experimental Result

In the experiment, we proposed the homogenous kernel-hashing algorithm and verified the high efficiency on the dataset. In our scheme, homogenous kernel-KLSH method makes it possible to get the unknown feature embeddings. We use these features to conduct vision search task to locate the most similar items in the database, and the neighbors we find in the task will give their scores on the tags. The method proved to be more effective and accurate than the linear scan search.

In this part, we design and verify our algorithm on the Caltech-101 dataset in Figure 1. Caltech-101 dataset is a benchmark on image recognition and classification, which has 101 categories objects and each category has about 100 images, so 10000 images totally. In recent years, many researchers have done useful research on this dataset such us proposing some important and useful image represent kernels [19]. Also there are many published papers that focused on this dataset, some of which are very valuable and significantly historic. For example, papers [20–22], respectively, state their contribution to the dataset. The author of [21] proposed the matching method for pyramid kernel of images histograms, while Berg [20] proposed and created the CORR kernel of

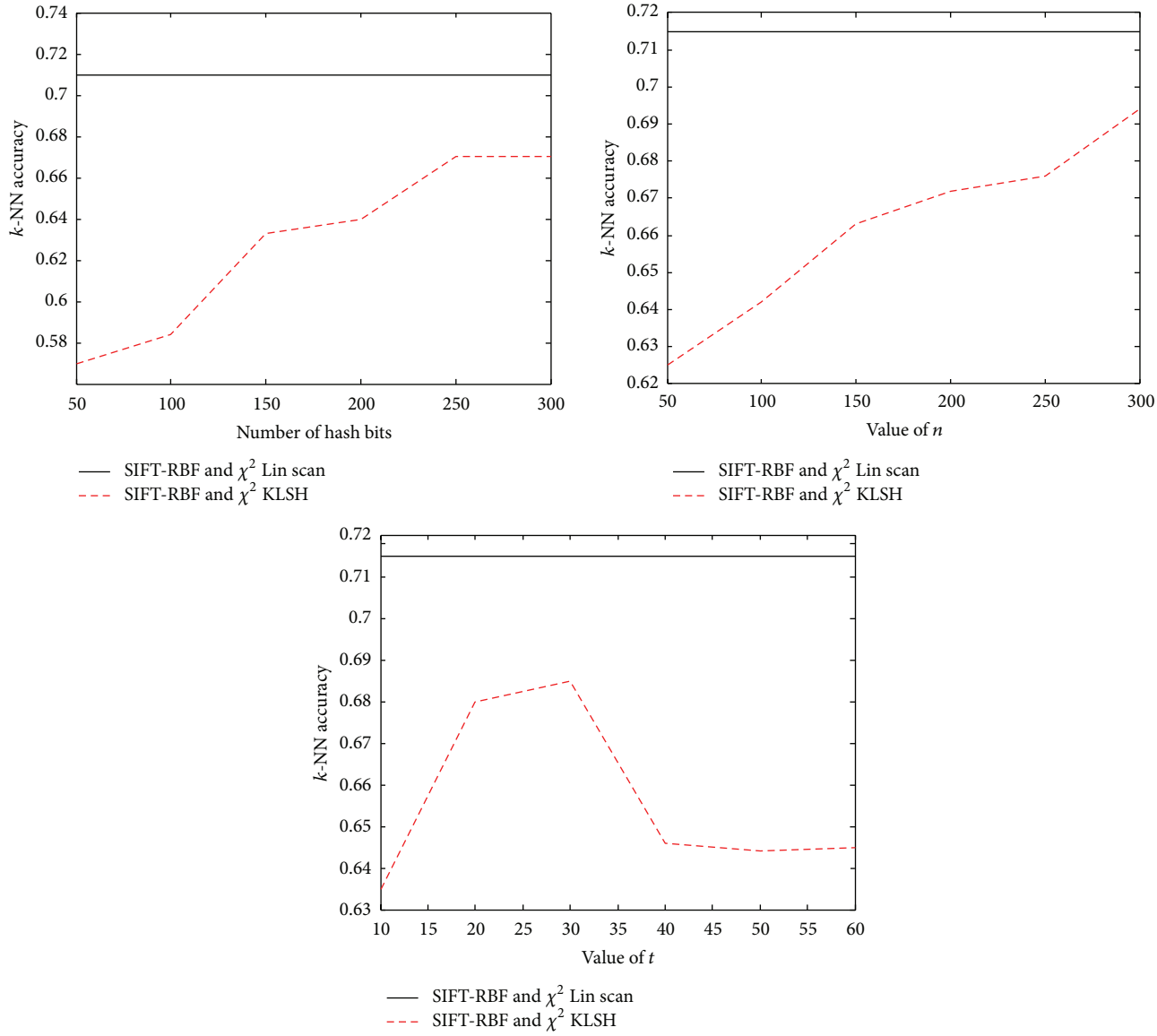

FIGURE 2: Hashing using a RBF-χ^2 kernel for SIFT based on homogenous kernels χ^2 ($\gamma = 1/2$). We choose $t = 30$, $n = 300$, and $b = 300$ in our experiment.

image local feature using geometric blur for matching local image similarity.

In our paper, we apply our algorithm to complete the vision classification and similar search task. The platform of our benchmark is based on Intel 4 core 3.6 GHZ CPU and 16 GB of memory and 2 TB hard disk.

We used χ^2 kernel for γ-homogeneous kernel maps ($\gamma = 1/2$) and applied the nonlinear RBF-χ^2 kernel designed in [19, 23] to the SIFT-based local feature. Meanwhile we applied and learnt the homogenous kernel map beyond it. Compared with the nonlearnt kernel, our learnt kernel has been more accurate. And we use KNN classifier, respectively, for KLSH and linear scan to compute the accuracy of classification. We also compare it with CORR [24] and the result proves to be better than them, here we use 15 images per class for training task.

From Figure 2 we can see that the growth of parameters is closely related with accuracy. As is seen, the accuracy increased with the increase of n, while it has little relationship with the number of t and b. The value of (n, t, b) is chosen as $n = 300$, $b = 300$, $t = 30$ as the best parameters through a series of experiments.

We find that the combination of these parameters can result in better performance than the large-scale dataset. Meanwhile it can be seen that our approach with homogenous kernel map has higher accuracy than CORR-KLSH with metric learning [25].

Figure 3 illustrates that our method is superior to other existing approaches [25–28] tested on this dataset. Comparing with other kernel classifiers, our classifier with RBF-χ^2 kernel for local features performs better. In Table 2 we can see that the result of ours has higher accuracy with $T = 15$

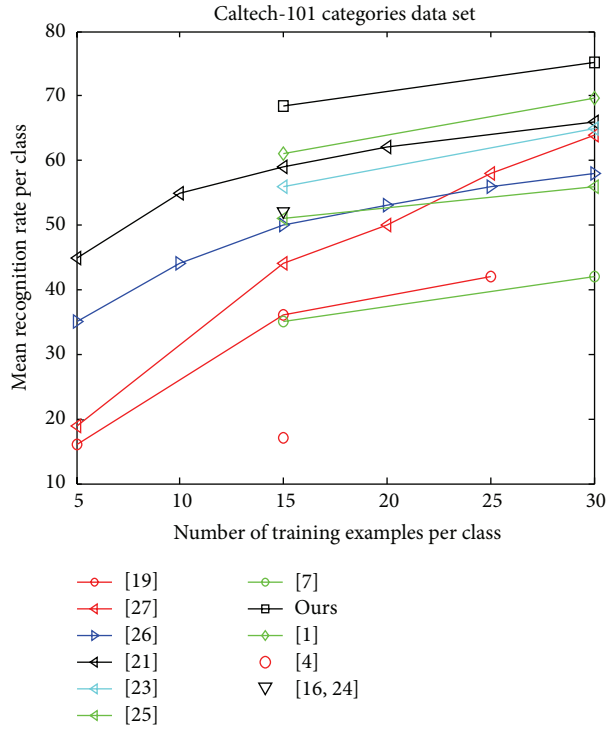

Figure 3: Comparison against existing techniques on the Caltech-101.

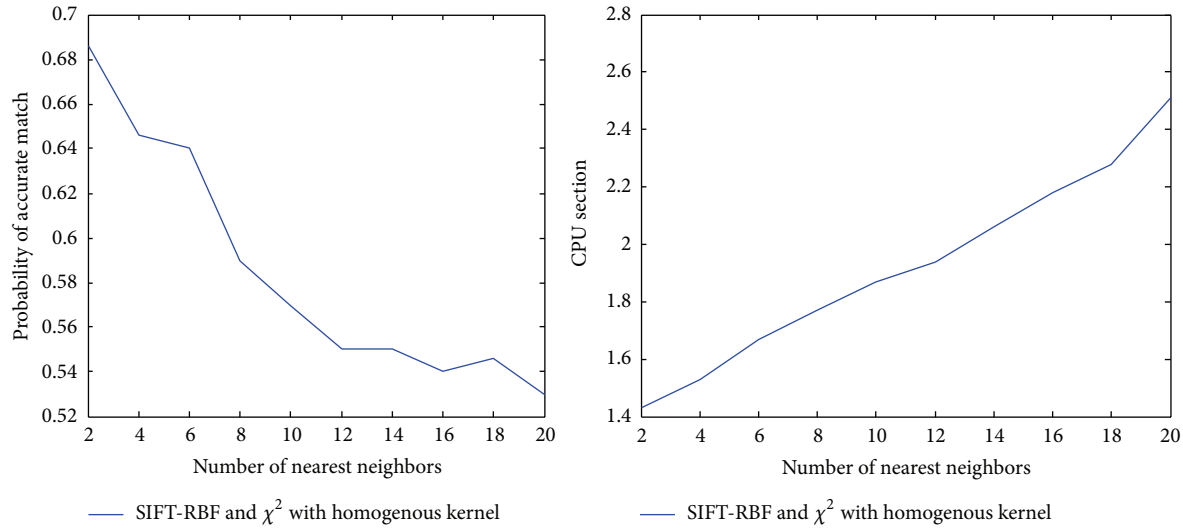

Figure 4: Classification beyond CPU load performance.

and $T = 30$ than other papers' results including better than [24] which obtains the result by 61% for $T = 15$ and 69.6% for $T = 30$. More clearly, it has improved the result by 16% several years ago.

In order to find the best parameters in our experiment for NN search for our scheme, we should take into account the balance between performance and CPU time. Therefore here we conducted to analyze the performance and CPU time of different of k ($k = 2, 3, \ldots, 20$) for NN search. Figure 4 illustrates the accuracy and CPU time by each k in our dataset.

The author of [26] proposed the method by combining KPCA and normal LSH. That means computing hashing beyond the KPCA. However this method has apparent disadvantage because KPCA will bring on the loss of input information although it can reduce the dimensionality in the processing, while KLSH can solve this problem to assure the integrity of input information to compute the LSH. Therefore

Table 2: Accuracy of Caltech-101.

#train	Ours	[18]	[29]	[30]	[31]	[32]	[33]
15	68.5	59.05	56.4	52	51	49.52	44
30	75.2	66.23	64.6	N/A	56	58.23	63

we found that our method has high accuracy and better performance than the algorithm in [26].

6. Conclusions

In our paper, we properly use the concept of homogeneous kernel maps to help us to solve the problem of approximation of those kernels, including those commonly used in machine learning such as χ^2, JS, Hellinger's, and intersection kernels. Combining with the KLSH scheme, it enables us to have access to any kernel function for hashing functions. Although our approach is inferior to linear scan search in time but it can guarantee that the search accuracy will not be affected. Moreover we do not need to consider the distribution of input data; to some extent, it can be applicable for many other databases as Flicker and Tiny Image. Experimental results demonstrate that it is superior to standard KLSH algorithm.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank Brian Kulis from UC Berkeley University for helpful comments on experiments and also they should thank Professor Yan Shuicheng for advice on their paper. The work was supported in part by the National Program on Key Basic Research Project (973 Program) 2010CB731403/2010CB731406.

References

[1] B. Kulis, P. Jain, and K. Grauman, "Fast similarity search for learned metrics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2143–2157, 2009.

[2] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV '03)*, pp. 750–757, Nice, France, October 2003.

[3] G. Shakhnarovich, T. Darrell, and P. Indyk, Eds., *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, The MIT Press, Cambridge, Mass, USA, 2006.

[4] A. Bardera, J. Rigau, I. Boada, M. Feixas, and M. Sbert, "Image segmentation using information bottleneck method," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1601–1612, 2009.

[5] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-Hash and tf-idf weighting," in *Proceeding of the British Machine Vision Conference (BMVC '08)*, September 2008.

[6] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pp. 380–388, Montreal, Canada, May 2002.

[7] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*, pp. 518–529, 1999.

[8] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object cateories," in *Proceedings of the Workshop on Generative Model Based Vision*, 2004.

[9] M. Hein and O. Bousquet, "Hilbertian metrics and positive definite kernels on probability measures," in *Proceedings of the 10th Workshop on Artificial Intelligence and Statistics (AISTAT '05)*, January 2005.

[10] J. Puzicha, J. M. Buhmann, Y. Rubner, and C. Tomasi, "Empirical evaluation of dissimilarity measures for color and texture," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV '99)*, pp. 1165–1172, September 1999.

[11] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[12] A. Barla, F. Odone, and A. Verri, "Histogram intersection kernel for image classification," in *Proceedings of the International Conference on Image Processing (ICIP '03)*, pp. 513–516, September 2003.

[13] B. Scholkopf and A. J. Smola, *Learning with Kernels*, The MIT Press, 2002.

[14] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proceeding of the 12th International Conference on Computer Vision (ICCV '09)*, pp. 2130–2137, Kyoto, Japan, October 2009.

[15] Z. Chen, J. Samarabandu, and R. Rodrigo, "Recent advances in simultaneous localization and map-building using computer vision," *Advanced Robotics*, vol. 21, no. 3-4, pp. 233–265, 2007.

[16] M. Weems, *Kernelized locality sensitive hashing for fast landmark association [M.S. thesis]*, 2011.

[17] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the Association for Computing Machinery*, vol. 42, no. 6, pp. 1115–1145, 1995.

[18] B. Schölkopf, A. Smola, and K. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[19] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple Kernels for object detection," in *Proceedings of the 12th International Conference on Computer Vision (ICCV '09)*, pp. 606–613, October 2009.

[20] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, 33, p. 26, San Diego, Calif, USA, June 2005.

[21] K. Grauman and T. Darrell, "Discriminative classification with sets of image features," in *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV '05)*, pp. 1458–1465, October 2005.

[22] A. D. Holub, M. Welling, and P. Perona, "Combining generative models and fisher kernels for object recognition," in *Proceedings*

of the 10th IEEE International Conference on Computer Vision (ICCV '05), pp. 136–143, October 2005.

[23] M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off," in *Proceedings of the IEEE 11th International Conference on Computer Vision (ICCV '07)*, October 2007.

[24] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "SVM-KNN: discriminative nearest neighbor classification for visual category recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp. 2126–2136, June 2006.

[25] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, 2008.

[26] J. Yang, X. Gao, and D. Zhang, "Kernel ICA: an alternative formulation and its application to face recognition," *Pattern Recognition*, vol. 38, no. 10, pp. 1784–1787, 2005.

[27] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, pp. 994–1000, June 2005.

[28] N. Rasiwasia and N. Vasconcelos, "Latent dirichlet allocation models for image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2665–2679, 2013.

[29] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: spatial pyramid matching for recognizing natural scene categories," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp. 2169–2178, June 2006.

[30] A. C. Berg, *Shape matching and object recognition [Ph.D. thesis]*, Computer Science Division, University of California, Berkeley, Calif, USA, 2005.

[31] J. Mutch and D. Lowe, "Multiclass object recognition using sparse, localized features," in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR '06)*, 2006.

[32] K. Grauman and T. Darrell, "Pyramid match kernels: discriminative classication with sets of image features," Tech. Rep. CSAIL-TR-2006-020, MIT, 2006.

[33] G. Wang, Y. Zhang, and F.-F. Li, "Using dependent regions for object categorization in a generative framework," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp. 1597–1604, New York, NY, USA, June 2006.