CRH: A Simple Benchmark Approach to Continuous Hashing

Miao Cheng
College of Information Engineering
Qingdao University
Qingdao, China
E-mail: mcheng@qdu.edu.cn

Ah Chung Tsoi
Faculty of Information Technology
Macau University of Science and Technology
Macau S.A.R., China
E-mail: actsoi@must.edu.mo

Abstract

In recent years, the distinctive advancement of handling huge data promotes the evolution of ubiquitous computing and analysis technologies. With the constantly upward system burden and computational complexity, adaptive coding has been a fascinating topic for pattern analysis, with outstanding performance. In this work, a continuous hashing method, termed continuous random hashing (CRH), is proposed to encode sequential data stream, while ignorance of previously hashing knowledge is possible. Instead, a random selection idea is adopted to adaptively approximate the differential encoding patterns of data stream, e.g., streaming media, and iteration is avoided for stepwise learning. Experimental results demonstrate our method is able to provide outstanding performance, as a benchmark approach to continuous hashing.

Index Terms

Data handling; ubiquitous computing; adaptive coding; streaming media.

I. Introduction and Related Work

Similarity search, as the basic issue of most information systems, addresses the problem of learning the similar data from buckets with respect to a given query data point [1][2]. It is also known as *Approximate Nearest Neighbour* (ANN) search [3] in large data sets with an identical demand. Generally, it is required to learn the compact binary codes of data, and then retrieval and recall of information can be implemented with codes matching [4].

The basic idea has resorted to use several hash functions mapping a bridge between input features and data codes. Then, the obtained short binary strings act as an index to directly access elements with comparison. Typical examples include Locality-Sensitive Hashing (LSH) [5], which works by projecting the data points to hyper planes and then obtaining the hash codes by certain bounding of thresholds, e.g.,

$$h_k\left(f_k\left(x\right)\right) = \begin{cases} 1, & if\ \ f_k\left(x\right) \geq \varepsilon \\ 0, & otherwise. \end{cases} \tag{1}$$

where ε indicates the given threshold for binary coding. Furthermore, the similarities corresponding to different data points, e.g., x_i and x_j, are expected to be preserved in the hashing codes $h\left(f_k\left(x_i\right)\right)$ and $h\left(f_k\left(x_j\right)\right)$, and ANN indexing is available for extended feedback search. And a data mapping function $f\left(x_i\right)$ is usually considered, which refers to linear transformations $w^T x_i + b$. Here, w is a projection direction and b is the mapping offset. The similar idea has been widely adopted to design variants of the hash functions with simple projections, while keeping the main patterns absorbed into binary hash codes. The outstanding advantage over others is their learning are mostly data-independent, and good compatibility is able to be provided for wide applications [6][7][8].

Another quite typical attempt is to define an objective function, usually a hash function as optimized encoder [4][9], so that binary codes can be obtained with continuous relaxation. In light of Laplacian eigenmaps, Spectral Hashing (SH) [10] computes approximate solutions by relaxing the binary constraints. Based on this idea, some variations use the codes from spectral hashing to define the hash function as labels of a binary classifier. To hash the similar data among a large data set, a common approach is used to query data patterns with similarity preserving. It has been a common view that iterative optimization is applied to find out the best solution by defining objective functions [11][4]. Nevertheless, it requires more calculations to meet the local optimum usually. And it would be ideal for complicated applications if few steps are adopted in encoder of binary hashing.

Until now, many outlets seek for data patterns, of which pair-distances are to be preserved in reduced binary codes as possible, inspired by theory below,

Theorem 1: [12] The distance $D\left(x_i, x_j\right)$ in L_2 norm in the full dimensional space between two vectors x_i and x_j is completely preserved in the reduced space obtained from the thin SVD. That is, $D\left(x_i, x_j\right) = D\left(\widetilde{x_i}, \widetilde{x_j}\right)$, where $\widetilde{x_i} = \widetilde{U}^T x_i$ and $\widetilde{x_j} = \widetilde{U}^T x_j$, and the thin SVD of X is $\widetilde{U}\widetilde{\Sigma}\widetilde{V}^T$.

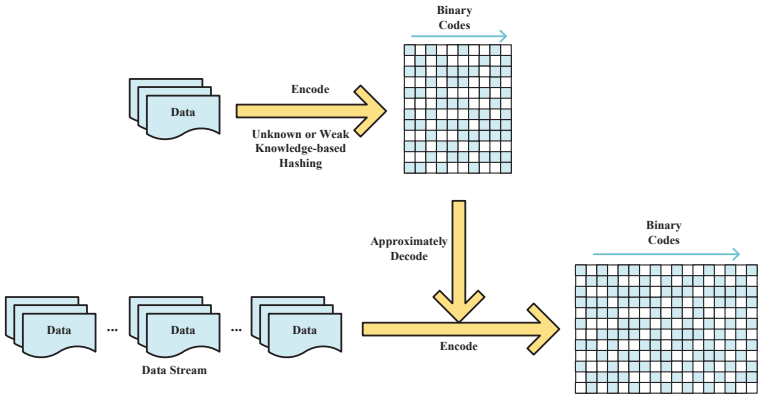


Fig. 1. The basic idea of continuous random hashing.

Furthermore, there are some other methods focusing on the supervised / semi-supervised extensions of hashing mechanism [13], while the training information is utilised to improved the binary coding ability. For example, PCA-like hashing algorithms apply the well-known principle component analysis (PCA) to find the projection functions in an iterative approach [9][14][11], where most solutions adopt an iteration approach to hash data into a suitable rotation of projections. But they may be in loss of adaptive controlling in such repeated steps, and difficult to handle data stream applications, e.g., online learning [15]. Furthermore, there are several issues should be highlighted if data stream is conducted for continuous hashing:

- As the increasing amount of appetitive set from data stream, the pre-learned hashing may fail to handle new data. And even worse, there could be few available knowledge connecting new hashing codes with previous ones.
- For persist and continuous computation, it is unreasonable and almost impossible to explicitly learn hashing function every once coming of new data, while the hashing results of previous data are desired to be absorbed.

Though acceptable results can be obtained, iteration is necessary to ensure the efficiency of stepwise improvement. And most of such kind of methods require the calculation of differences among whole data to determine the similarity or unsimilarity between different data. Recently, some novel solutions attempt to tackle the large amounts of data emerge as random hashing [16][17], and low calculation of data handling is preferred for further applications.

In this work, such problem is conducted to encode coming data as well as a set of available hashing results, but the pre-defined hashing mechanism is unknown for further handling as illustrated in Fig. 1. Or explained alternatively, the proposed method does not depend on the inherited hashing of labeled data for encoding of new data as no more knowledge could be explicitly learned. Similar to some existing methods, e.g., LSH, the calculation of differences among whole data is unnecessary to find and update nearest neighbours.

II. Continuous Random Hashing

Given a data set $X_0 = [x_1, x_2, \cdots, x_{n_0}] \in \mathbb{R}^{d \times n_0}$, while the associated hashing codes may be available at hand, the continuous random hashing (CRH) aims to encode the coming data stream $X_1 \in \mathbb{R}^{d \times n_1}, X_2 \in \mathbb{R}^{d \times n_2}, \cdots, X_k \in \mathbb{R}^{d \times n_k}$ in sequence. In the literature, the most matched approaches to the similar problem have been considered as such kind of scalable learning. However, they have proposed the basic hashing procedure with strictly required conditions with supervised or semi-supervised setting, as well as previous labeled information. Though these considerations are able to conduct scalable hashing sharing with the common idea of incremental learning, may fail to make further development on more complicated sources except for concrete assumptions.

The most popular assumption on such scenario is to construct the objective function with knowledge on nearest neighbors (NNs) of each data beforehand, then supervised or semi-supervised handling procedure can be carried out as an optimized hashing with neighborhood information. Distinguishingly, the searching for NNs are not required in CRH to enhance the binary discrimination ability, and further no serve condition is referred for sequential learning of presently coming data. In details, the data sets $X_1, X_2, \cdots, X_{n_k}$ can be afforded by CRH with different sample sizes, and few updates are resorted for handling of each data sets.

Without loss of generic strategy, it is assumed that there is an available mean centered data set $X \in \mathbb{R}^{d \times p}$ with binary codes $A = \left[t_1, t_2, \cdots, t_p\right] \in \mathbb{R}^{m \times p}$ of ideal hashing, where m usually holds a small size than dimensionality of original data d. Then, the open problem could be described as such that there comes another new data set $Y = \left[y_1, y_2, \cdots, y_q\right] \in \mathbb{R}^{d \times q}$ ($q \neq p$ *is possible*),

it is required to learn the binary hashing codes $S = \left[s_1, s_2, \cdots, s_q\right] \in \mathbb{R}^{m\times q}$ of each new data efficiently. As it is designed to preserve data similarity in reduced codes, it is straightforward to optimize the following objective function:

$$\begin{aligned} Obj(s) &= \arg\min_{s\in\{-1,1\}} \sum_{i=1}^{q}\sum_{j=1}^{p} \left\|H(s_i, t_j) - D(y_i, x_j)\right\|_2 \\ &= \arg\min_{s\in\{-1,1\}} \sum_{i=1}^{q}\sum_{j=1}^{p} \left\|\frac{1}{m}D(s_i, t_j) - D(y_i, x_j)\right\|_2 \end{aligned} \tag{2}$$

Here, $H(\cdot,\cdot)$ denotes the hamming distance between two binary codes, affiliated with corresponding Euclidean distance $D(\cdot,\cdot)$ of original data. Nevertheless, it is different to handle the objective with high-order optimization, and almost hardly to be applied if large amounts of data stream are to be sequentially encoded.

To address these limitations, the original data are all normalized ${x_i}^T x_i = 1$ as done in previous work [18]. By defining another function $g\left(x_i, x_j\right) = {x_i}^T x_j$, then the formalism can be further compressed as

$$\begin{aligned} Obj(s) &= \arg\min_{s\in\{-1,1\}} \sum_{i=1}^{q}\sum_{j=1}^{p} \left\|\frac{1}{m}{s_i}^T t_j - {y_i}^T x_j\right\|_2 \\ &= \arg\min_{s\in\{-1,1\}} \sum_{i=1}^{q}\sum_{j=1}^{p} \left\|\frac{1}{m}g\left(s_i, t_j\right) - g\left(y_i, x_j\right)\right\|_2 \end{aligned} \tag{3}$$

And it could be optionally solved as a standard linear equation. Nevertheless, there are several vital limitations suspended for further applications, e.g.,

- If large amounts of data are met, system would suffer from ordeal of computational cost on data similarities.
- The solution requires the inverse of coded data set as a necessary step, which is difficult to get if large data are involved.
- As the intrinsic differences between original and binary data, hashing codes are hardly to ideally approximate the original data with simple handling, while features are always reduced to short ones.

For adaptive considerations, the problem is to be addressed in a random approach to decoding, and is realistically approximate to objective in practice. Here, the following theorem has been referred.

Theorem 2: For a given data set $X \in \mathbb{R}^{d\times m}$ and another data set $Y \in \mathbb{R}^{d\times n}$ as described in context, the similarities between them G can be approximated by rank-k matrix $\Pi U_k {U_k}^T \Pi$, where Π is formed of X and Y as $\Pi = [X\ Y]$, U_k is the matrix consisting of the left singular vectors of C_Π. And C_Π is a submatrix whose columns consisting of randomly selected from Π. Proof: The proof is given in Appendix.

In terms of this consideration, it is feasible to use a subset of original data $\widehat{X}$ associated with hashing codes $\widehat{A}$ to meet the demand of original problem (3). Thereafter, the calculation could be reduced to a *random* approach with lightweight cost, and piecewise learning is feasible for sequential data.

More specifically, it is reliable to randomly pick a subset data $\widehat{X} \in \mathbb{R}^{d\times r}$ from X to form data set Π with whole coming data Y, and then its decomposition is obtained as

$$\Pi = [\ \widehat{X} \quad Y\ ] = U_m \Lambda_m {V_m}^T \tag{4}$$

Thus, the inner product of data can be approximated as $\Pi^T U_m {U_m}^T \Pi$. And the whole procedure of CRH could be summarized as:

- Randomly select a subset data set $\widehat{X}$ from normalized data set X, and form data $\Pi = [\ \widehat{X} \quad Y\ ]$ with coming data Y.
- Calculate the low-rank approximate decomposition $\Pi = [\ \widehat{X} \quad Y\ ] = U_m \Lambda_m {V_m}^T$.
- Solve the optimization problem (3) with approximate data, and obtain the result s.

III. Experiments

In this section, the performance of the proposed method is compared with several benchmark hashing methods, including LSH [2], SH [10], ITQ [11], and RMMH [16]. As the continuous hashing is considered, the encoding of testing data is hardly to refer to training stage of previous hashing. As a result, all hashing methods encode query or new data separating from pre-defined hashing functions, and encoders independently learn binary codes of data sets sequentially. Two datasets, CIFAR-10 image dataset [19] and MNIST digit database [20], are used to evaluate the hashing results.

For images in CIFAR-10 dataset, a set of 512 dimensional GIST descriptors [21] and 128 dimensional SIFT descriptors [22] are learned from every tiny image, so that each data is represented as a 640 dimensional sample vector. In the experiments, the different batches in CIFAR-10 dataset is combined into one by ignoring their groups and several subsets are randomly selected to form the aforehand hashing data, query data and sequential data. For each data in MNIST, 784 dimensional gray features are used to describe its visual handwritten patterns. Similarly, the whole data are combined while the original order of data is disordered, and then lots of subsets are randomly selected to be involved into experiments. Furthermore, the performance is

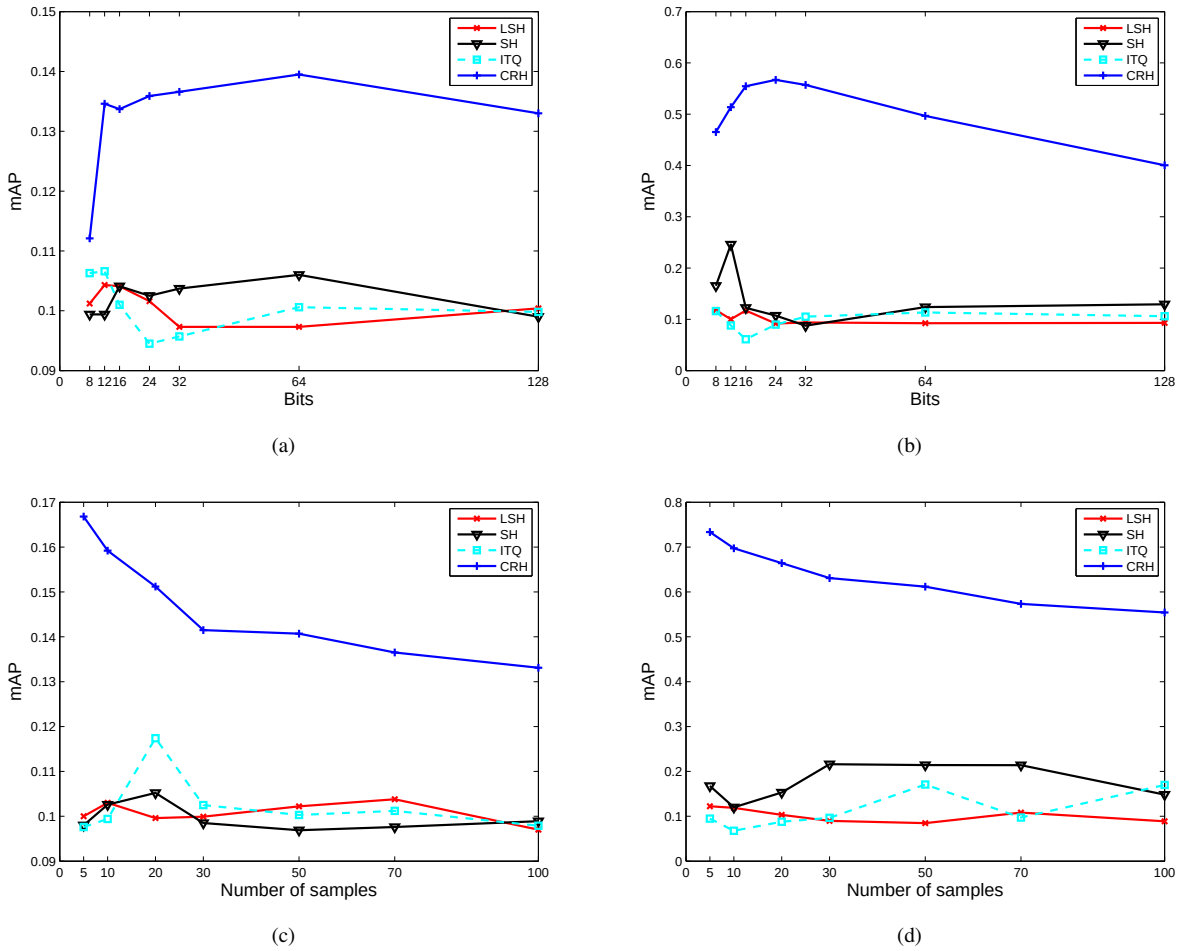


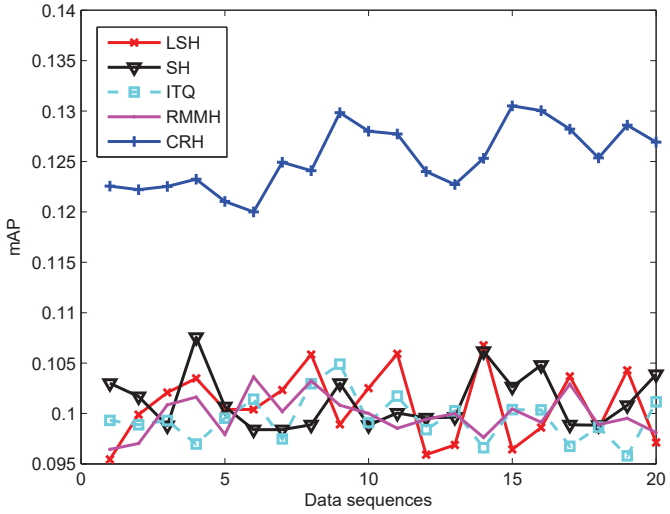
Fig. 2. The search results from CIFAR-10 and MNIST datasets. (a) and (b) the obtained results with changing hashing bits; (c) and (d) the obtained results with different number of involving nearest neighbors in mAP.

measured by the mean average precision (mAP) defining as the average area under the recall-precision curves, and the ground truths are computed based on k nearest neighbors.

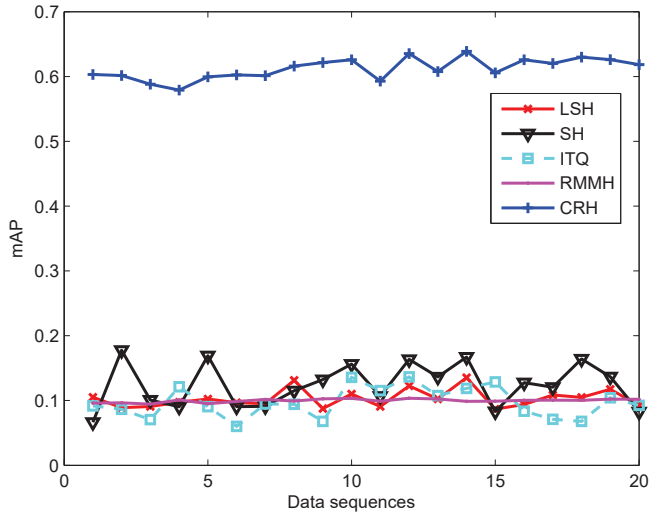
In this first experiment, the search performance of encoders are evaluated with two databases. Among all samples of each dataset, 10000 data are randomly selected and encoded with binary hashing, and then 500 data are randomly selected to push forward matching query. For each hashing algorithms, the two data sets are encoded separately, and no supervised / semi-supervised corporation between them is involved. With respect to random column selection of CRH, it is found large quantity of involved columns are to provide convincible performance. However, the selected columns are set to be around 8% - 10% of existing data, which is enough to give acceptable performance. The search results with different hashing bits are shown in Fig. 2 (a) and (b), while the obtained results against different number of samples used in mAP matching are shown in Fig. 2 (c) and (d). The experiments are proceeded while fixing the length of hashing bits and involved neighbors in mAP by turns. In details, 100 neighbors are used to measure mAP, and 16 fixed bits are used to generate results with different neighbors.

As illustrated, the results obtained from CRH outperforms other methods at an optimal ability of encoding, the learning binary codes are able to provide distinctive patterns for query. However, it seems proposed random hashing works better on MNIST, for concrete pattern concision. Among other benchmark hashing approaches, SH gives much better performance compared with other unsupervised ones, with robust self-adaptive encoding manner. ITQ also provides acceptable results if enough search information, e.g., hashing bits or nearest neighbors, can be given in searches. This may because it mainly depends on iterative rotation for improvement on projection directions, as an adjustment of learning error. Furthermore, most methods reach the common level of query performance, providing a similar performance of search matching of separate hashing.

In the second experiment, the continuous hashing ability of different methods are evaluated with a sequence of data stream. At first, 10000 samples are randomly selected from each database, and their binary codes are learned based on different hashing algorithms. Then a series of data sets are absorbed into the encoders by turns, and each data set has 500 data samples for



(a)



(b)

Fig. 3. The sequential encoding results from CIFAR-10 and MNIST datasets. (a) Sequential encoding results from CIFAR-10; (b) Sequential encoding results from MNIST.

encoding. Another recently proposed random hashing method, namely RMMH [16], is joined for comparison of continuous hashing with randomly selected dispersive data sets. Noticeably, no supervised / semi-supervised welfare of previously hashing data involve into encoding of currently coming data. Similarly, the binary codes of originally stored data and the appended data are learned separately, while each hashing methods handle the new data as their coming immediately. The encoding results against data stream sequences are shown in Fig. 3 (a) and (b), of which 16 binary bits and 50 NNs are used to measure performance learned from two dispersive groups with 100 randomly selected data.

For continuous hashing of the coming data, most algorithms give the similar results as standard hashing, while the hashing function is suspended for the labeled codes of available data. As the results shown, there is always dynamic fluctuation over decoded matching results as sequential data coming. Though the results are changed over different crests and troughs, the fluctuate area is strictly bounded to attain a stable performance for sequential learning of data stream. Nevertheless, SH is possible to give better results compared with other methods, benefitting from spectral preserving idea. Obviously, CRH is able to overcome the difficulties in sequentially adaptive learning, and provide the useful binary codes with random selected samples.

IV. Conclusions

As an adaptive encoder learning solution, a continuous hashing method is proposed to learn the binary patterns of multi-source data. The motivation is based on the fact that encoder may be unavailable for binary hashing of further sequential data, e.g., data stream. This pushes forward much attentions on designing efficient solutions to continuous hashing. Different from existing methods, the proposed benchmark approach is able to learn the binary codes without previously hashing knowledge involved, and iterative optimization can be avoided in an efficient manner. Furthermore, adaptive hashing is component to continuous learning by benefitting from a random selection of coded data. As experimental results shown, the proposed method is able to provide outstanding performance for sequential hashing compared with other benchmark methods.

V. Proof of Theorem 2

Lemma 1: [23] Given the matrix $G \in \mathbb{R}^{n\times n}$ with $G_{ij} = g(x_i, x_j)$ and $p_i = G_{ii}^2 / \sum_{i=1}^{n} G_{ii}^2 \left(\sum_{i=1}^{n} p_i = 1\right)$, the original matrix could be approximated as $\widetilde{G} = CW_k{}^{+}C^T$, where C is the selected columns of G according to the sampling probabilities p_i, and W_k is the best rank-k approximation to W, the matrix formed by the intersection between those c columns of G and the corresponding c rows of G.

Proof: Following the above lemma, the original matrix is able to be approximated by randomly selected C from G with uniform probability. By introducing the sampling matrix $S \in \mathbb{R}^{n\times c}$ as a zero-one matrix where $S_{ij} = 1$ if the i-th column of X is chosen and $S_{ij} = 0$ otherwise, the selected columns can be described as $C = GSD \in \mathbb{R}^{d\times c}$ and D is the rescaling diagonal matrix $D \in \mathbb{R}^{c\times c}$ with $D_{tt} = 1\Big/\sqrt{cp_{i_t}}$. Then, it is clear that $C_\Pi = \Pi SD$ and $W = (SD)^T GSD$. Let the rank-k decomposition of C_Π be $C_\Pi = \widetilde{U}_k \widetilde{\Sigma}_k \widetilde{V}_k{}^T$, the approximate matrix can be calculated as $\Pi^T \widetilde{U}_k \widetilde{U_k}{}^T \Pi$.

Acknowledgment

This work was supported by research committee of Macau University of Science and Technology. The corresponding author of this work is Dr. Miao Cheng.

References

[1] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the ACM Symposium on Computational Geometry*, 2004.

[2] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Communications of The ACM*, vol. 51, no. 1, pp. 117–122, 2008.

[3] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[4] M. Norouzi and D. Fleet, "Minimal loss hashing for compact binary codes," in *International Conference on Machine Learning*, 2011.

[5] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *STOC*, 2002, pp. 380–388.

[6] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[7] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[8] C. Strecha, A. A. Bronstein, M. M. Bronstein, and P. Fua, "Ldahash: Improved matching with smaller descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 66–78, 2012.

[9] M. A. Carreira-Perpinan and R. Raziperchikolaei, "Hashing with binary autoencoders," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 557–566.

[10] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Neural Information Processing Systems*, vol. 21, 2009, pp. 1753–1760.

[11] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2011.

[12] H. Kim, H. Park, and H. Zha, "Distance preserving dimension reduction for manifold learning," in *Proceedings of the SIAM International Conference on Data Mining*, 2007.

[13] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, and N. Yu, "Complementary hashing for approximate nearest neighbor search," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011.

[14] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2957–2964.

[15] F. R. Bach, "Exploring large feature spaces with hierarchical multiple kernel learning," in *Neural Information Processing Systems*, 2008.

[16] A. Joly and O. Buisson, "Random maximum margin hashing," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2011.

[17] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Neural Information Processing Systems*, 2007.

[18] M. Cheng, C.-M. Pun, and Y. Y. Tang, "Nonnegative class-specific entropy component analysis with adaptive step search criterion," *Pattern Analysis and Applications*, vol. 17, no. 1, pp. 113–127, 2014.

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. report, 2009.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, 1998, pp. 2278–2324.

[21] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[23] P. Drineas and M. W. Mahoney, "On the nystrom method for approximating a gram matrix for improved kernel-based learning," *Journal of Machine Learning Research*, vol. 6, pp. 2153–2175, 2005.