Adaptive Quantization for Hashing: An Information-Based Approach to Learning Binary Codes

Caiming Xiong* Wei Chen* Gang Chen* David Johnson* Jason J. Corso*

Abstract

Large-scale data mining and retrieval applications have increasingly turned to compact binary data representations as a way to achieve both fast queries and efficient data storage; many algorithms have been proposed for learning effective binary encodings. Most of these algorithms focus on learning a set of projection hyperplanes for the data and simply binarizing the result from each hyperplane, but this neglects the fact that informativeness may not be uniformly distributed across the projections. In this paper, we address this issue by proposing a novel adaptive quantization (AQ) strategy that adaptively assigns varying numbers of bits to different hyperplanes based on their information content. Our method provides an information-based schema that preserves the neighborhood structure of data points, and we jointly find the globally optimal bit-allocation for all hyperplanes. In our experiments, we compare with state-of-the-art methods on four large-scale datasets and find that our adaptive quantization approach significantly improves on traditional hashing methods.

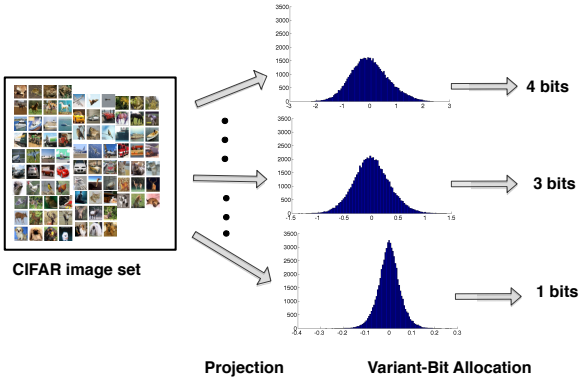


Figure 1: Example of our Adaptive Quantization (AQ) for binary codes learning. Note the varying distributions for each projected dimension (obtained via PCA hashing). Clearly, the informativeness of the different projections varies significantly. Based on AQ, some of the projections are allocated multiple bits while others are allocated fewer or none.

1 Introduction

In recent years, methods for learning similarity-preserving binary encodings have attracted increasing attention in large-scale data mining and information retrieval due to their potential to enable both fast query responses and low storage costs [1–4]. Computing *optimal* binary codes for a given data set is NP hard [5], so similarity-preserving hashing methods generally comprise two stages: first, learning the projections and second, quantizing the projected data into binary codes.

Most existing work has focused on improving the first step, attempting to find high-quality projections that preserve the neighborhood structure of the data (e.g., [6–15]). Locality Sensitive Hashing (LSH) [6] and its variants [8,10,16,17] are exemplar data-independent methods. These data-independent method produce more generalized encodings, but tend to need long codes because they are randomly selected and do not consider the distribution of the data. In contrast, data-dependent methods that consider the neighbor structure of the data points are able to obtain more compact binary codes (e.g., Restricted Boltzmann Machines (RBMs) [7], spectral hashing [5], PCA hashing [18], spherical hashing [11], kmeans-hashing [13], semi-supervised hashing [19, 20], and iterative quantization [21, 22]).

However, relatively little attention has been paid to the quantization stage, wherein the real-valued projection results are converted to binary. Existing methods typically use Single-Bit Quantization (SBQ), encoding each projection with a single-bit by setting a threshold. But quantization is a lossy transformation that reduces the cardinality of the representation, and the use of such a simple quantization method has a significant impact on the retrieval quality of the obtained binary codes [23, 24]. Recently, some researchers have responded to this limitation by proposing higher-bit quantizations, such as the hierarchical quantization method of Liu et al. [25], the double-bit quantization of Kong

*Department of Computer Science and Engineering, SUNY at Buffalo. {cxiong, wchen23, gangchen, davidjoh, jcorso}@buffalo.edu

et al. [23] (see Section 2 for a thorough discussion of similar methods).

Although these higher-bit quantizations report marked improvement over the classical SBQ method, they remain limited because they assume that each projection requires the same number of bits. To overcome these limitation, Moran et al. [26] first propose variable-bit quantization method with adaptive learning based on the score of the combination of F_1 score and regulation term, but the computational complexity is high when obtaining the optimal thresholds and the objective score in each dimension with variable bits. From an information theoretic view, the optimal quantization of each projection needs to consider the distribution of the projected data: projections with more information require more bits while projections with less require fewer.

To that end, we propose a novel quantization stage for learning binary codes that adaptively varies the number of bits allocated for a projection based on the informativeness of the projected data (see Fig. 1). In our method, called Adaptive Quantization (AQ), we use a variance criterion to measure the informativeness of the distribution along each projection. Based on this uncertainty/informativeness measurement, we determine the information gain for allocating bits to different projections. Then, given the allotted length of the hash code, we allocate bits to different projections so as to maximize the total information from all projections. We solve this combinatorial problem efficiently and optimally via dynamic programming.

In the paper, we fully develop this new idea with an effective objective function and dynamic programming-based optimization. Our experimental results indicate adaptive quantization universally outperforms fixed-bit quantization. For example, for the case of PCA-based hashing [18], it performs lowest when using fixed-bit quantization but performs highest, by a significant margin, when using adaptive quantization. The rest of the paper describes related work (Sec. 2), motivation (Sec. 3), the AQ method in detail (Sec. 4), and experimental results (Sec. 5).

2 Related Work

Some of previous works have explored more sophisticated multi-bit alternatives to SBQ. We discuss these methods here. Liu et al. [25] propose a hierarchical quantization (HQ) method for the AGH hashing method. Rather than using one bit for each projection, HQ allows each projection to have four states by dividing the projection into four regions and using two bits to encode each projection dimension.

Kong et al. [23] provide a different quantization strategy called DBQ that preserves the neighbor structure more effectively, but only quantizes each projection into three states via double bit encoding, rather than the four double bits can encode. Lee et al. [27] present a similar method that *can* utilize the four double bit states by adopting a specialized distance metric.

Kong et al. [28] present a more flexible quantization approach called MQ that is able to encode each projected dimension into multiple bits of natural binary code (NBC) and effectively preserve the neighborhood structure of the data under Manhattan distance in the encoded space. Moran et al. [24] also propose a similar way with F-measure criterion under Manhattan distance.

The above proposed quantization methods have all improved on standard SBQ, yielding significant performance improvements. However, all of these strategies share significant limitation that they adopt a fixed k-bit allowance for each projected dimension, with no allowance for varying information content across projections.

Moran et al. [26] propose variable-bit quantization method to address the limitation based on the score of the combination of F-measure score and regulation term. Since the computational complexity is high when obtaining the optimal thresholds and the objective score in each dimension with variable bits, they propose an approximation method, but without optimal guarantee.

Our proposed adaptive quantization technique addresses both of these limitations, proposing an effective and efficient information gain criterion that account for the number of bits allocated in each dimension and solve the allocation problem with dynamic programming.

3 Motivation

Most hash coding techniques, after obtaining the projections, quantize each projection with a single bit without considering the distribution of the dataset in each projection. The success of various multi-bit hashing methods (Section 2) tells us that this is insufficient, and information theory [29] suggests that standard multi-bit encodings that allocate the same number of bits to each projection are inefficient. The number of bits allocated to quantize each projection should depend on the informativeness of the data within that projection. Furthermore, by their very nature many hashing methods generate projections with varying levels of informativeness. For example, PCAH [18] obtains independent projections by SVD and LSH [6] randomly samples projections. Neither of these methods can guarantee a uniform distribution of informativeness across their projections. Indeed, particularly in the case of PCAH, a highly non-uniform distribution is to be expected (see the variance distributions in Figure 2, for example).

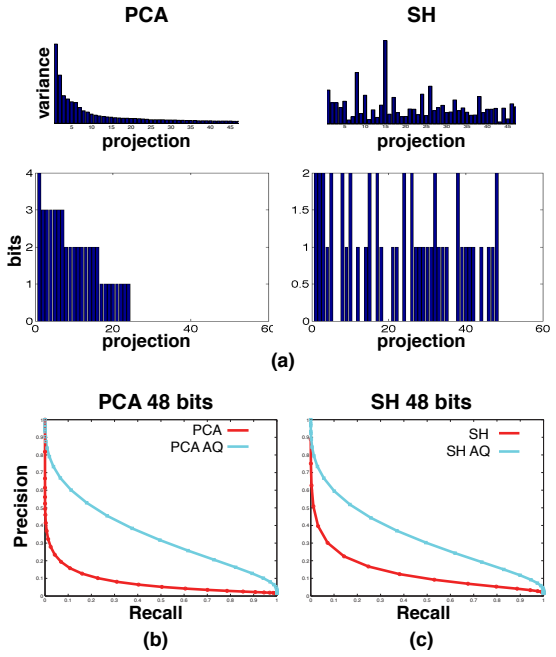


Figure 2: Illustrating the adaptive quantization process on real data for two different hashing methods. (a-c) is an example of projection informativeness/uncertainty, corresponding bit allocation and its Precision-Recall Curve for PCA hashing and SH with 48 bit length of binary code in NUS [30] image dataset. *Best viewed in color.*

To address this problem, we define a simple and novel informativeness measure for projections, from which we can derive the information gain for a given bit allocation. We also define a meaningful objective function based on this information gain, and optimize the function to maximize the total information gain from a given bit allocation.

Given the length of the hash code L, fixed k-bit quantizations require $\frac{L}{k}$ projections from the hashing methods, since each projection must be assigned k bits. However, with our variable bit allocation, the number of projections obtained from hashing methods can be any number from 1 to L or larger, since our method is as capable of allocating zero bits to uninformative projections as multiple bits to highly informative projections. Therefore, our AQ method also includes implicit projection selection. Figure 2 illustrates the ideas and outputs of our method, showing the informativeness and bit allocation for each projection on the NUS [30] image set for PCAH [18] and SH [5], as well as the resulting significant increase in retrieval accuracy.

4 Adaptive Quantization

Assume we are given m projections $\{f_1(\cdot), f_2(\cdot), \cdots, f_m(\cdot)\}$ from some hashing method, such as PCAH [18] or SH [5]. We first define a measure of informativeness/uncertainty for these projections. In information theory, variance and entropy are both well-known measurements of uncertainty/informativeness, and either can be a suitable choice as an uncertainty measurement for each projection. For simplicity and scalability, we choose variance as our informativeness measure. The informativeness of each projection $f_i(x)$ is then defined as:

$$E_i(X) = \frac{\sum_{j=1}^{n}(f_i(x_j) - \mu_i)^2}{n} \tag{4.1}$$

where μ_i is the center of the data distribution within projection $f_i(x)$. Generated projections are, by nature of the hashing problem, independent, so the total informativeness of all projections is calculated via $E_{\{1,2,\cdots,m\}} = \sum_i E_i(X)$.

Allocating k bits to projection $f_i(x)$ yields 2^k different binary codes for this projected dimension. Thus, there should be 2^k centroids in this dimension that partition it into 2^k intervals, or clusters. Each data point is assigned to one of these intervals and represented via its corresponding binary code. Given a set of 2^k clusters, we define the informativeness of the projected dimension as:

$$E_i^{'}(X,k) = \sum_{l=1}^{2^k} \frac{\sum_j Z_{jl}(f_i(x_j) - \mu_l)^2}{\sum_j Z_{jl}} \tag{4.2}$$

where $Z_{jl} \in \{0,1\}$ indicates whether data point x_j belongs to cluster l and μ_l is the center of cluster l.

$E_i(X)$ in Eq. 4.1 can be thought of as the informativeness of the projection when 0 bits have been allocated (i.e. when there is only one center). Therefore we can define $E_i^{'}(X,0) = E_i(X)$. Based on this definition of $E_i^{'}(X,k)$, we propose a measure of "information gain" from allocating k-bits to projection $f_i(x)$ which is the difference between $E_i^{'}(X,0)$ and $E_i^{'}(X,k)$:

$$G_i(X,k) = E_i^{'}(X,0) - E_i^{'}(X,k) \tag{4.3}$$

The larger $G_i(X,k)$, the better this quantization corresponds to the neighborhood structure of the data in this projection.

$E_i^{'}(X,0)$ is fixed, so maximizing $G_i(X,k)$ is same as choosing values of $\{Z_{jl}\}$ and $\{\mu_l\}$ that minimize $E_i^{'}(X,k)$. This problem formulation is identical to the objective function of single-dimensional $\mathcal{K}$-means, and can thus be solved efficiently using that algorithm. Therefore, when allocating k bits for a projection, we

can quickly find 2^k centers (and corresponding cluster and binary code assignments) that maximize our notion of information gain (Eq. 4.3). In our experiments, the maximum value of k is 4. One expects that, for a given data distribution, the information gain gradient will decrease exponentially with increasing k; hence the optimal k will typically be small.

4.1 Joint Optimization for AQ We propose an objective function to adaptively choose the number of bits for each projection based on resulting information gains. Assume there are m projections $\{f_1(x), f_2(x), \cdots, f_m(x)\}$ and corresponding information gain $G_i(X, k)$ for k bits. The goal of the objective function is to find an optimal bit allocation scheme that maximizes the total information gain from all projections for the whole data set. Our objective function can be formulated:

$$\begin{aligned} \{k_1^*, k_2^*, \cdots, k_m^*\} = & \underset{\{k_1, k_2, \cdots, k_m\}}{\operatorname{argmax}} \sum_{i=1}^{m} G_i^{'}(X, k_i) \\ \text{s.t.} \quad & \forall i \in \{1:m\}, k_i \in \{0, 1, \cdots, k_{max}\} \\ & \sum_{i=1}^{m} k_i = L \\ & G_i^{'}(X, k_i) = \max_{Z,\mu} G_i(X, k_i) \\ & \qquad = E_i^{'}(X, 0) - E_i^{'}(X, k_i), \end{aligned} \tag{4.4}$$

where k_i is the number of bits allocated to projection $f_i(x)$ and L is the total length of all binary codes. $\sum_{i=1}^{m} G_i^{'}(X, k_i)$ is the total information gain from all projections, and $G_i^{'}(X, k_i)$ is the corresponding maximal information gain $G_i(X, k_i)$ for k_i bits in projection $f_i(x)$ (easily computed via single-dimensional $\mathcal{K}$-means). Again, because the projections are independent, we can simply sum the information gain from each projection.

With m projections $\{f_1(x), f_2(x), \cdots, f_m(x)\}$ and corresponding information gains $G_i^{'}(X, k)$ for k bits, we can find the optimal bit allocation for each projection by solving Eq. 4.4, which maximizes total information gain from the L bits available. However, optimizing Eq. 4.4 is a combinatorial problem—the number of possible allocations is exponential, making a brute force search infeasible.

We thus propose an efficient dynamic-programming-based [31] algorithm to achieve the optimal bit allocation for our problem (k_{max} is a parameter controlling the maximum number of bits that can be allocated to a single projection). Given the binary hash code length L and m projections such that $L \leq m \cdot k_{max}$, denote total information gain with length L and m projections as $J_m(L) = \sum_{i=1}^{m} G_i^{'}(X, k_i)$ s.t. $\sum_{i=1}^{m} k_i = L$. We can then express our problem via a Bellman equation [31]:

$$J_m^*(L) = \max_{L - k_{max} \leq v_m \leq L} (J_{m-1}^*(v_m) + G_m^{'}(X, L - v_m)) \tag{4.5}$$

where $J_m^*(L)$ is the optimal cost (maximal total information gain) of the bit allocation that we seek. Based on this setup, each subproblem (to compute some value $J_i^*(v_m)$) is characterized fully by the values $1 \leq i \leq m$ and $0 \leq v_m \leq L$, leaving only $\mathbf{O}(mL)$ unique subproblems to compute. We can thus use dynamic programming, to quickly find the globally optimal bit allocation for the given code length and projections.

4.2 Adaptive Quantization Algorithm Given a training set, an existing projection method with m projections, a fixed hash code length L and a parameter k_{max}, our Adaptive Quantization (AQ) for hashing method can be summarized as follows:

1. Learn m projections via an existing projection method such as SH, PCAH.
2. For each projection $f_i(x)$ calculate the corresponding maximal information gain $G_i^{'}(X; k)$ for each possible k ($0 \leq k \leq k_{max}$).
3. Use dynamic programming to find the optimal bit allocation that maximizes total information gain (as formulated in Eq. 4.4).
4. Based on the optimized bit allocation and corresponding learned centers for each projection, quantize each projected dimension into binary space and concatenate them together into binary codes.

4.3 Complexity analysis During training, the method will run $\mathcal{K}$-means k_{max} times for each projection to acquire different information gain scores for different numbers of bits. The complexity cost of computing each projection's information gain is thus $\mathbf{O}(nk_{max})$. Given that there are m projections, the total cost of computing all of the $G_i^{'}(X, k_i)$ values is $\mathbf{O}(mnk_{max})$. These values are then used in dynamic programming to find the optimal bit allocation, costing $\mathbf{O}(mLk_{max})$ time. This yields a total complexity of $\mathbf{O}(mk_{max}(n + L))$, which is effectively equivalent to $\mathbf{O}(mnk_{max})$, since we assume $L << n$. Further, in typical cases $k_{max} << m$ (indeed, in our experiments we use $k_{max} = 4$), so it is reasonable to describe the complexity simply as $\mathbf{O}(mn)$.

Obviously, the most time-consuming part of this process is $\mathcal{K}$-means. We can significantly reduce the time needed for this part of the process by obtaining

$\mathcal{K}$-means results using only a subset of the data points. Indeed, in our experiments, we run $\mathcal{K}$-means on only 10,000 points in each dataset (note that this is only about 1% of the data on the Gist-1M-960 dataset). For each projection, we run $\mathcal{K}$-means four times (since we allocate at most four bits for each projection). For the 64-bit case, using typical PC hardware, it takes less than 9 minutes to compute all of our information gain values, and less than 2 seconds to assign bits using dynamic programming. Using a larger sample size may potentially increase performance, at the cost of commensurately longer run times, but our experiments (Section 5) show that a sample size of only 10,000 nonetheless yields significant performance increases, even on the larger datasets.

5 Experiments

5.1 Data We test our method and a number of existing state-of-the-art techniques on four datasets:

- CIFAR [22, 32]: a labeled subset of the 80 million tiny images dataset, containing 60,000 images, each described by 3072 features (a 32x32 RGB pixel image).
- NUS-WIDE [30]: composed of roughly 270,000 images, with 634 features for each point.
- Gist-1M-960 [1]: one million images described by 960 image gist features.
- 22K-Labelme [14, 33]: 22,019 images sampled from the large LabelMe data set. Each image is represented with 512-dimensional GIST descriptors as in [33].

5.2 Evaluation Metrics We adopt the common scheme used in many recent papers which sets the average distance to the 50th nearest neighbor of each point as a threshold that determines whether a point is a true "hit" or not for the queried point. For all experiments, we randomly select 1000 points as query points and the remaining points are used for training. The final results are the average of 10 such random training/query partitions. Based on the Euclidean ground-truth, we measure the performance of each hashing method via the precision-recall (PR) curve, the mean average precison (mAP) [34, 35] and the recall of the 10 ground-truth nearest neighbors for different numbers of retrieved points [1]. With respect to our binary encoding, we adopt the Manhattan distance and natural multi-bit binary encoding method suggested in [28].

5.3 Experimental Setup Here we introduce the current baseline and state-of-the-art hashing and quantization methods.

Baseline and state-of-the-art-hashing methods

- LSH [6]: obtains projections by randomly sampling from the Standard Gaussian function.
- SKLSH [9]: uses random projections approximating shift-invariant kernels.
- PCAH [18]: uses the principal directions of the data as projections.
- SH [5]: uses the eigendecomposition of the data's similarity matrix to generate projections.
- ITQ [21]: an iterative method to find an orthogonal rotation matrix that minimizes the quantization loss.
- Spherical Hashing (SPH) [11]: a hypersphere-based binary embedding technique for providing compact data representation.
- Kmeans-Hashing (KMH) [13]: a kmeans-based affinity-preserving binary compact encoding method.

In order to test the impact of our quantization strategy, we extract projections from the above hashing methods and feed them to different quantization methods.

Baseline and state-of-the-art quantization methods

In this paper, we compare our adaptive quantization method against three other quantization methods: SBQ, DBQ and 2-MQ:

- SBQ: single-bit quantization, the standard technique used by most hashing methods.
- DBQ [23]: double-bit quantization.
- 2-MQ [28]: double bit quantization with Manhattan distance.

We use 2-MQ as the baseline, because it generally performs the best out of the existing methods [28]. We test a number of combinations of hashing methods and quantization methods, denoting each 'XXX YYY', where XXX represents the hashing method and YYY is the quantization method. For example 'PCA DBQ' means PCA hashing (PCAH) [18] with DBQ [23] quantization.

5.4 Experimental Results To demonstrate the generality and effectiveness of our adaptive quantization method, we present two different kinds of results. The first is to apply our AQ and the three baseline quantization methods to projections learned via LSH, PCAH,

SH and ITQ and compare the resulting scores. The second set of results uses PCAH as an examplar hashing method (one of the simplest) and combines it with our AQ method; then compares it with current state-of-the-art hashing methods. We expect that by adding our adaptive quantization, the "PCA AQ" method will achieve performance comparable to or or even better than other state-of-the-art algorithms.

Comparison with state-of-the-art quantization methods

The mAP values are shown in Figures 3, 4 and 5 for the GIST-1M-960, NUS-WIDE and CIFAR image datasets, respectively. Each element in these three tables represents the mAP value for a given dataset, hash code length, hashing method and quantization technique. For *any* combination of dataset, code length and projection algorithm, our adaptive quantization method performs on-par-with or better-than the other quantization methods, and in most cases is *significantly* better than the next-best algorithm.

This pattern can also be seen in the PR curves shown in Figure 6, where once again our quantization method never underperforms, and usually displays significant improvement relative to all other methods. Due to space restrictions, we only included the PR curves for the CIFAR dataset, but we observed similarly strong PR curve performance on both of the other datasets.

Comparison with state of the art hashing methods According to typical experimental results [13, 21] PCAH generally performs worse than other state-of-the-art hashing methods such as ITQ, KMH and SPH.

To demonstrate the importance of the quantization stage, we add our AQ method to PCAH and compare the resulting "PCA AQ" method with other state-of-the-art techniques. Running the experiment on the 22K Labelme dataset, we evaluate performance using mean average precison (mAP) [34,35] and recall of the 10-NN for different numbers of retrieved points [1].

In Figure 7 (a) and (b), we show that, while the standard PCAH algorithm is consistently the worst-performing method, simply replacing its default quantization scheme with our AQ method produces results significantly better than any of the other state-of-the-art hashing methods.

6 Conclusion

Existing hashing methods generally neglect the importance of learning and adaptation in the quantization stage. In this paper we propose an adaptive learning to the quantization step produced hashing solutions that were uniformly superior to previous algorithms. This promises to yield immediate and significant benefits to existing hashing applications, and also suggests that quantization learning is a promising and largely unexplored research area, which may lead to many more improvements in data mining and information retrieval.

Acknowledgements This work was funded partially by NSF CAREER IIS-0845282 and DARPA CSSG D11AP00245 and D12AP00235.

References

[1] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33(1):117–128, 2011.

[2] Zeehasham Rasheed and Huzefa Rangwala. Mc-minh: Metagenome clustering using minwise based hashing. In *SIAM SDM*, 2013.

[3] Mingdong Ou, Peng Cui, Fei Wang, Jun Wang, Wenwu Zhu, and Shiqiang Yang. Comparing apples to oranges: a scalable solution with heterogeneous hashing. In *Proceedings of the 19th ACM SIGKDD*, pages 230–238. ACM, 2013.

[4] Yi Zhen and Dit-Yan Yeung. A probabilistic model for multimodal hash function learning. In *Proceedings of the 18th ACM SIGKDD*, pages 940–948. ACM, 2012.

[5] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. NIPS, 2008.

[6] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *IEEE FOCS 2006*.

[7] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[8] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *Proceedings of BMVC*, 2008.

[9] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. *NIPS*, 22, 2009.

[10] D. Gorisse, M. Cord, and F. Precioso. Locality-sensitive hashing scheme for chi2 distance. *IEEE TPAMI*, (99):1–1, 2012.

[11] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *Proceedings of CVPR*, pages 2957–2964. IEEE, 2012.

[12] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *Proceedings of CVPR*, pages 2074–2081. IEEE, 2012.

[13] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In *Proceedings of CVPR*. IEEE, 2013.

[14] Zhao Xu, Kristian Kersting, and Christian Bauckhage. Efficient learning for hashing proportional data. In *Proceedings of ICDM*, pages 735–744. IEEE, 2012.

[15] Junfeng He, Wei Liu, and Shih-Fu Chang. Scalable similarity search with optimized kernel hashing. In

#bits	32				48				64			
	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ
ITQ	0.2414	0.2391	0.2608	**0.2824**	0.2642	0.2866	0.3021	**0.3619**	0.2837	0.3216	0.3457	**0.3941**
LSH	0.1788	0.1651	0.1719	**0.1897**	0.2109	0.2011	0.2298	**0.2311**	0.2318	0.2295	**0.2663**	0.2558
PCA	0.1067	0.1829	0.1949	**0.2726**	0.1179	0.2151	0.2325	**0.3052**	0.1196	0.2292	0.2281	**0.3341**
SH	0.1135	0.1167	0.2158	**0.2554**	0.1473	0.1435	0.2473	**0.2842**	0.1648	0.1670	0.2725	**0.3278**
SKLSH	0.1494	0.1309	0.1420	**0.1671**	0.1718	0.1485	0.1974	**0.2025**	0.1911	0.1840	0.2202	**0.2284**
#bits	128				192				256			
	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ
ITQ	0.3130	0.4063	0.4701	**0.5165**	0.3245	0.4589	0.5074	**0.5900**	0.3340	0.4926	0.5700	**0.6321**
LSH	0.2791	0.2996	0.3064	**0.3270**	0.3030	0.3493	0.3625	**0.3846**	0.3191	0.3904	0.4232	**0.4314**
PCA	0.1209	0.2447	0.2408	**0.3769**	0.1183	0.2457	0.2650	**0.3815**	0.1168	0.2442	0.2621	**0.3673**
SH	0.2028	0.2258	0.3464	**0.3774**	0.2377	0.2622	0.3832	**0.4240**	0.2440	0.2743	0.3761	**0.4689**
SKLSH	0.2541	0.2359	0.2706	**0.2955**	0.3011	0.2797	**0.3536**	0.3480	0.3456	0.3033	0.3683	**0.3872**

Figure 3: mAP on Gist-1M-960 image dataset. The mAP of the best quantization method for each hashing method is shown in bold face.

#bits	32				48				64			
	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ
ITQ	0.1786	0.1939	0.2107	**0.2404**	0.2142	0.2378	0.2722	**0.3022**	0.2382	0.2903	0.3873	**0.3493**
LSH	0.1005	0.0976	0.1089	**0.1126**	0.1299	0.1293	**0.1558**	**0.1533**	0.1531	0.1606	**0.1867**	**0.1871**
PCA	0.1021	0.1534	0.1960	**0.2265**	0.1082	0.1816	0.2110	**0.2980**	0.1094	0.2036	0.2334	**0.3316**
SH	0.0815	0.1225	0.1823	**0.2173**	0.0963	0.1307	0.2371	**0.2621**	0.1101	0.1425	0.3314	**0.2989**
SKLSH	0.0689	0.0725	0.0734	**0.0911**	0.1066	0.0810	0.0997	**0.1211**	0.1282	0.0982	**0.1526**	0.1452
#bits	128				192				256			
	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ
ITQ	0.3010	0.3992	0.4530	**0.5175**	0.3270	0.4668	0.5788	**0.6260**	0.3444	0.5149	0.6158	**0.6774**
LSH	0.2259	0.2664	0.3103	**0.3099**	0.2705	0.3408	0.3741	**0.3979**	0.3028	0.4008	**0.4605**	**0.4602**
PCA	0.1003	0.2359	0.2842	**0.4242**	0.0965	0.2321	0.2837	**0.4485**	0.0938	0.2177	0.2673	**0.4547**
SH	0.1405	0.1978	0.3936	**0.4395**	0.1539	0.2216	0.3872	**0.4787**	0.1539	0.2456	0.3694	**0.5303**
SKLSH	0.1988	0.1821	**0.2673**	**0.2660**	0.2785	0.2421	0.3373	**0.3404**	0.3295	0.2809	**0.4080**	**0.4001**

Figure 4: mAP on NUS-WIDE image dataset. The mAP of the best quantization method for each hashing method is shown in bold face.

#bits	32				48				64			
	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ
ITQ	0.1591	0.2054	0.2346	**0.3051**	0.1821	0.2469	0.2884	**0.3907**	0.1984	0.2914	0.3374	**0.4437**
LSH	0.0985	0.1160	0.1221	**0.1429**	0.1273	0.1499	0.1721	**0.1882**	0.1485	0.1816	**0.2219**	**0.2239**
PCA	0.0638	0.1537	0.1508	**0.2557**	0.0672	0.1683	0.1822	**0.3043**	0.0654	0.1732	0.1833	**0.3316**
SH	0.1107	0.1718	0.2188	**0.2423**	0.1164	0.1860	0.2727	**0.2929**	0.1328	0.2345	0.3189	**0.3369**
SKLSH	0.1089	0.0852	0.1045	**0.1171**	0.1455	0.1134	0.1507	**0.1693**	0.1441	0.1493	0.1836	**0.1985**
#bits	128				192				256			
	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ	SBQ	DBQ	2-MQ	AQ
ITQ	0.2355	0.3841	0.4481	**0.5849**	0.2506	0.4263	0.5393	**0.6590**	0.2594	0.4612	0.5850	**0.7065**
LSH	0.2163	0.2893	0.3348	**0.3568**	0.2538	0.3697	0.4193	**0.4576**	0.2809	0.4269	0.4702	**0.5245**
PCA	0.0619	0.1691	0.1652	**0.3519**	0.0606	0.1584	0.1634	**0.3344**	0.0599	0.1493	0.1501	**0.3101**
SH	0.1797	0.2974	0.3844	**0.4442**	0.1942	0.3506	0.4381	**0.4747**	0.1951	0.3617	0.4334	**0.4587**
SKLSH	0.2527	0.2166	0.2781	**0.3272**	0.3606	0.2988	0.3917	**0.4228**	0.4059	0.3478	0.4652	**0.4887**

Figure 5: mAP on CIFAR image dataset. The mAP of the best quantization method for each hashing method is shown in bold face.

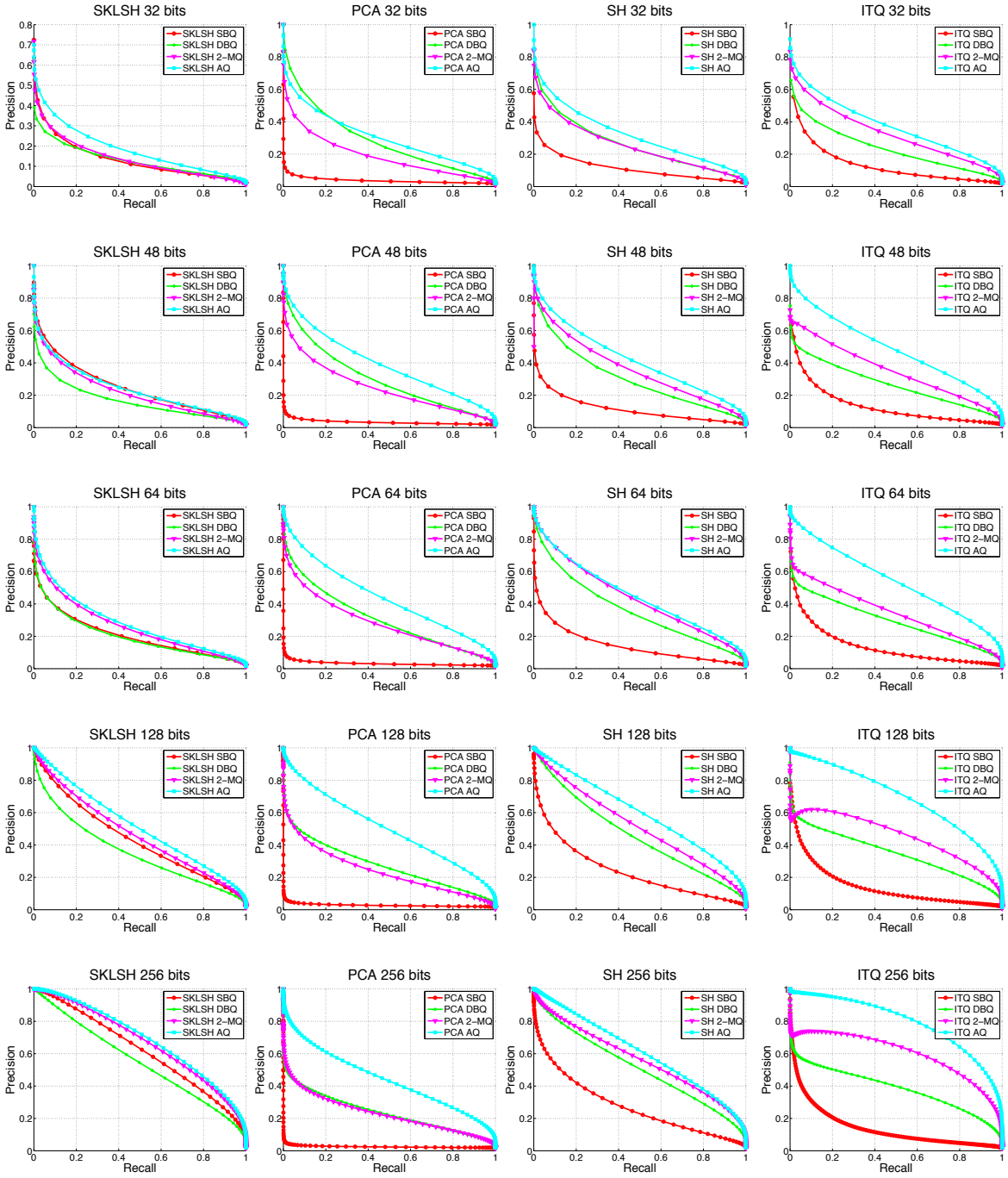
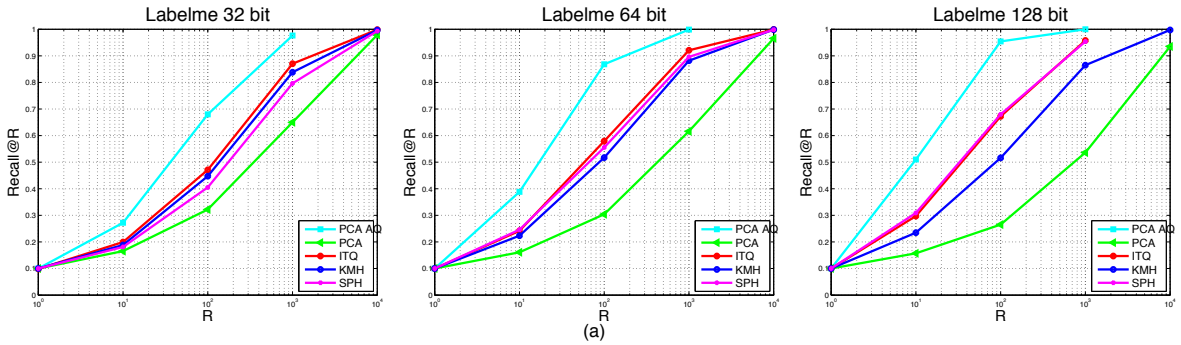


Figure 6: Precision-Recall curve results on CIFAR image dataset.



(a)

	PCAH	KMH	ITQ	SPH	PCA AQ
32 bit	0.0539	0.1438	0.2426	0.1496	**0.2700**
64 bit	0.0458	0.1417	0.2970	0.2248	**0.4615**
128 bit	0.0374	0.1132	0.3351	0.2837	**0.6241**

(b)

Figure 7: (a) Euclidean 10-NN recall@R (number of items retrieved) at different hash code lengths; (b) mAP on 22K Labelme image dataset at different hash code lengths.

Proceedings of the 16th ACM SIGKDD, pages 1129–1138. ACM, 2010.

[16] Kave Eshghi and Shyamsundar Rajaram. Locality sensitive hash functions based on concomitant rank order statistics. In *Proceedings of the 14th ACM SIGKDD*, pages 221–229. ACM, 2008.

[17] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. Fast locality-sensitive hashing. In *Proceedings of the 17th ACM SIGKDD*, 2011.

[18] Xin-Jing Wang, Lei Zhang, Feng Jing, and Wei-Ying Ma. Annosearch: Image auto-annotation by search. In *Proceedings of CVPR*, volume 2, pages 1483–1490. IEEE, 2006.

[19] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *Proceedings of CVPR*, pages 3424–3431. IEEE, 2010.

[20] Saehoon Kim and Seungjin Choi. Semi-supervised discriminant hashing. In *Proceedings of ICDM*, pages 1122–1127. IEEE, 2011.

[21] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824. IEEE, 2011.

[22] Jeong-Min Yun, Saehoon Kim, and Seungjin Choi. Hashing with generalized nyström approximation. In *Proceedings of ICDM*, pages 1188–1193. IEEE, 2012.

[23] Weihao Kong and Wu-Jun Li. Double-bit quantization for hashing. In *Proceedings of the Twenty-Sixth AAAI*, 2012.

[24] Sean Moran, Victor Lavrenko, and Miles Osborne. Neighbourhood preserving quantisation for LSH. In *36th Annual International ACM SIGIR*, 2013.

[25] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *Proceedings of the 28th ICML*, pages 1–8, 2011.

[26] Sean Moran, Victor Lavrenko, and Miles Osborne. Variable bit quantisation for LSH. In *Proceedings of ACL*, 2013.

[27] Youngwoon Lee, Jae-Pil Heo, and Sung-Eui Yoon. Quadra-embedding: Binary code embedding with low quantization error. In *Proceedings of ACCV*, 2012.

[28] Weihao Kong, Wu-Jun Li, and Minyi Guo. Manhattan hashing for large-scale image retrieval. In *Proceedings of the 35th international ACM SIGIR*, pages 45–54. ACM, 2012.

[29] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[30] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of ACM ICIVR*, page 48. ACM, 2009.

[31] Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716, 1952.

[32] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis*, 2009.

[33] Antonio Torralba, Robert Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *Proceedings of CVPR*, pages 1–8. IEEE, 2008.

[34] Albert Gordo and Florent Perronnin. Asymmetric distances for binary embeddings. In *Proceedings of CVPR*, pages 729–736. IEEE, 2011.

[35] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Proceedings of CVPR*, pages 3304–3311. IEEE, 2010.