DOUBLE-BIT QUANTIZATION AND WEIGHTING FOR NEAREST NEIGHBOR SEARCH

Han Deng[1], Hongtao Xie[1], Wei Ma[1], Zhendong Mao[1], and Chuan Zhou[2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, National Engineering Laboratory for Information Security Technologies, Beijing, China, 100093

[2] University of Chinese Academy of Sciences, Beijing, China, 100049

xiehongtao@iie.ac.cn

ABSTRACT

Binary embedding is an effective way for nearest neighbor (NN) search as binary code is storage efficient and fast to compute. It tries to convert real-value signatures into binary codes while preserving similarity of the original data. However, it greatly decreases the discriminability of original signatures due to the huge loss of information. In this paper, we propose a novel method double-bit quantization and weighting (DBQW) to solve the problem by mapping each dimension to double-bit binary code and assigning different weights according to their spatial relationship. The proposed method is applicable to a wide variety of embedding techniques, such as SH, PCA-ITQ and PCA-RR. Experimental comparisons on two datasets show that DBQW for NN search can achieve remarkable improvements in query accuracy compared to original binary embedding methods.

Index Terms—Double-bit quantization; weighted hamming distance; binary embedding; nearest neighbor search

1. INTRODUCTION

Nearest neighbor (NN) search has been one of the key problems of visual applications including image retrieval [1], object recognition [2] and copy detection [3]. NN search consists in finding the closest matches of a given query signature in large amounts of reference signatures. When searching similar signatures in a large-scale database composed of floating-point values, it usually computes the Euclidean distance between the query and all the reference signatures, which is quite costly. As a consequence, handling these large quantities of data has become a challenge on its own.

When coping with massive amounts of data, there exist two most influential factors: one is the computational cost, and the other is memory usage. Binary codes can exactly handle the two problems. On one hand, the calculation of hamming distance between two binary codes is extremely efficient and requires just a small number of machine instructions. On the other hand, the memory cost of binary codes is much less than real-valued signatures. These considerations directly lead to the growing interests in embedding real-valued signatures in compact binary codes. Despite of its remarkable advantages, the drawback is also obvious: it greatly reduces the distinctiveness between different signatures. For example, the possibilities of Euclidean distance between 32-dimensional real-valued signatures are endless. However, the hamming distance between two 32-bit binary codes only has limited 33 kinds of possibilities from 0 to 32. So presenting real-valued signatures in binary codes undoubtedly results in lower accuracy.

In general, several binary embedding algorithms, such as locality sensitive hashing (LSH) [4], PCA embedding (PCAE) [5], spectral hashing (SH) [6], PCA with random rotations (PCA-RR), and PCA with iterative quantization (PCA-ITQ) [7] can be decomposed into two steps: 1) The signatures are first embedded in an intermediate space and 2) Thresholding is performed in this space to obtain binary outputs. Certainly, the higher dimensionality of the signature, the higher retrieval accuracy we can get. However, we find in almost all of binary embedding algorithms, the average discriminability per dimension decreases gradually with the increase of dimensionality. Therefore, when the dimensionality is small, we can get more discriminability per dimension. Among the mentioned binary embedding methods, the step 2 usually maps each dimension in intermediate space to 1-bit binary code in hamming space. If we assign 2-bit binary code to each k dimension, in theory, the discriminability per dimension is twice of 1-bit quantization, in which the retrieval accuracy is higher than that of $2k$-dimension.

Building on the previous analysis, we propose a double-bit quantization and weighting (DBQW) algorithm for NN search which consists of two aspects:

- Double-bit quantization (DBQ). We first map each dimension of intermediated data to double-bit rather than one for higher retrieval accuracy.
- Weighted hamming distance (WHD). The hamming distances between binary codes based on DBQ are assigned different weights according to their spatial relationship.

In this way, we can preserve more discriminability of the original signatures while take advantages of binary codes to store and compare signatures efficiently.

In our experiments, we show that the DBQW can be applied very broadly, such as LSH [4], PCAE [5], SH [6], PCA-RR, and PCA-ITQ [7]. We evaluate our approach on two data sets BIGANN [9] and Calteach101 [10]. It demonstrates that the proposed DBQW consistently and significantly improves the retrieval accuracy of binary embedding methods over the traditional ones. In some cases, DBQW can provide precision on the order of 12%~48% against original methods.

The rest of the paper is organized as follows. In Section 2, we precisely describe the novel DBQW algorithm. Section 3 presents the experimental results on BIGANN and Calteach101 for NN search. Finally, conclusions are given in Section 4.

2. THE PROPOSED METHOD

In this section, we propose DBQW algorithm to improve retrieval accuracy for NN search. Firstly, we will briefly introduce binary embedding. Next, the novel DBQ method is given. Then, we specifically present the notion of WHD and its computation.

A. Double-Bit Quantization

1) Binary Embedding Algorithm

The prominent advantages of binary codes lead to the explosion in binary embedding techniques. Several successful binary embedding methods have been proposed, such as LSH, SH, PCAE, PCA-RR and PCA-ITQ. Binary embedding aims to transform real-value signatures into binary codes, while it guarantees that similar signatures are mapped into the same binary codes with a high probability.

In order to express the meaning of binary embedding clearly, we introduce a set of notations. Let s be an image signature with K dimensions in space Ω and let h_k be a binary embedding function, i.e., $h_k: \Omega \to \{0,1\}$. A set $H = \{h_k,\ k = 1 \dots K\}$ of K functions define a multidimensional embedding function $h: \Omega \to \{0,1\}^K$ with $h(s) = [h_1(s)\ h_K(s)]'$. Note that real-value signatures are not directly converted into binary codes via binary embedding. For LSH, SH, PCAE and PCAE-ITQ, binary embedding function h_k can be decomposed as follows:

$$h_k(s) = q_k[g_k(s)], \tag{1}$$

where $g_k(s): \Omega \to \mathcal{R}$(the intermediated space) is projection function and $q_k(s): \mathcal{R} \to \{0,1\}$ is quantization function. That is, binary embedding firstly projects image signature s to real-valued multidimensional vector $g(s) = [g_k(s), k = 1 \dots K]'$, which is an extremely good approximation to the original signature. Next, the real-valued data will be quantized into binary codes by thresholding (0 is often set as the threshold). That is, if $g_i(s) > 0$, s_i is mapped to 1. Otherwise, s_i will be mapped to 0. Thus, traditional quantization function just roughly divides each dimension into two parts decoded as 0 or 1, which greatly reduces the discriminability [4]. To alleviate this problem, we propose to retain more information of the original by assigning double-bit to each dimension of the intermediated data.

2) Double-Bit Quantization

In the previous section, we elaborate that binary embedding methods greatly reduce the discriminability of original signatures. To achieve higher accuracy, a lot of related works mainly concentrate on improving the performance of projection functions g_k. Instead, we propose a DBQ function to assign double-bit to each dimension of the intermediated data. The steps of DBQ are summarized below:

a) Signature projection and normalization. For a given signature x with K dimensions, we firstly use a multidimensional projection function $g(s) = [g_k(s), k = 1 \dots K]'$ to map original signature x to real-value vector $g(x)$ (the intermediated data). Next, to enhance the efficiency of comparisons, the vector is normalized by its $l_1$norms $\mathcal{N}$ for each dimension and the normalized intermediated data $l(x) = [l_k(s),$ $k = 1 \dots K]'$ are obtained with

$$l_i(s) = \mathcal{N}[g_i(s)]. \quad i = 1 \dots K \tag{2}$$

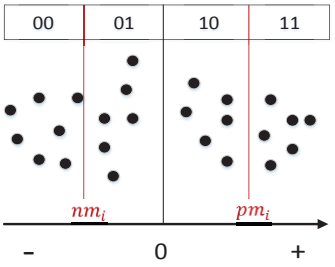


Figure 1 Encoding signature. Each dimension is divided into four parts by the sign and two medians

b) Data partition. We divide the intermediate data for each dimension into two categories according to the sign of the corresponding element, after which we get the medians of both two categories for all the dimensions. The medians of the negative and positive parts in dimension i are represented symbolically by nm_i and pm_i respectively. Based on the sign and two medians, each element of the intermediated vectors can be divided into four categories, shown as Fig. 1. Despite its crude nature, we will see that the partition scheme leads to competitive results on a variety of binary embedding methods.

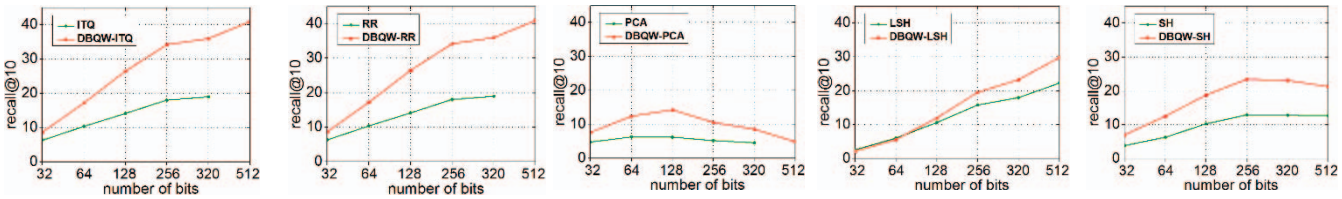
c) Binary quantization. After the partition, a novel quantization function needs to be raised, as positional relations of elements in each dimension via original quantization function have only two cases: either on the same side or the opposite side. In order to handle the new partition scheme, we quantize each dimension into double-bit. By this way, the quantization method may adapt well to the four relations of the elements in each dimension, as showed in Fig.1. For i_{th} dimension, DBQ function is defined as:

$$DBQ_i(s) = \begin{cases} 11, & if\ \ l_i(s) \geq pm_i \\ 10, & if\ \ l_i(s) \geq 0\ and\ l_i(s) < pm_i \\ 01, & if\ \ l_i(s) < 0\ and\ l_i(s) > nm_i \\ 00, & if\ \ l_i(s) \leq nm_i \end{cases} \tag{3}$$

Since intermediate data preserves good approximation to the similarity of original signatures, it has a high probability to map $g_i(x)$ and $g_i(y)$ to the same category if x is the nearest neighbor of y. Conversely, if signature x and y are far from each other, $g_i(x)$ and $g_i(y)$ are more likely to be mapped far apart. Thus, the quantization scheme can naturally preserve the similarity between two signatures.

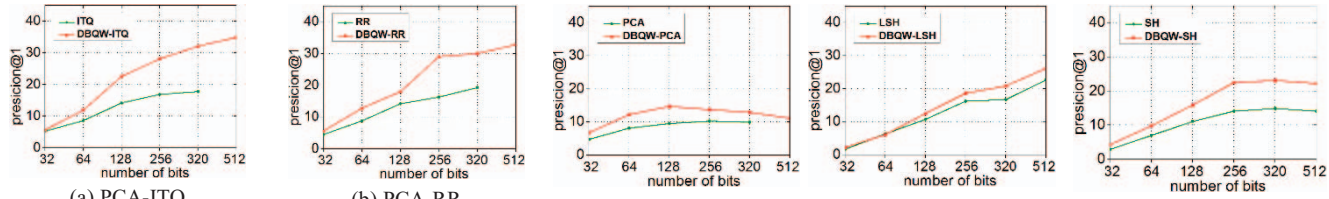
B. Weighted Hamming Distance

For binary codes generated from DBQ, we can't directly calculate the hamming distance between them. There are two reasons. First, DBQ partitions each dimension of intermediate data into four parts. Thus, the quantized signatures have four kinds of spatial relationship which can be represented by 4 kinds of distances (0, 1, 2 and 3). However, the hamming distance between 2-bit binary codes only has three possible values (0, 1 and 2). Second, XOR operation can't describe the distance between 2-bit codes accurately in our quantization method. For example, the dis-



(a) PCA-ITQ (b) PCA-RR (c) PCA (d) LSH (e) SH

Figure 2 Influence to recall of the weighted hamming distance on the Caltech101 dataset using 320-d gist descriptors.



(a) PCA-ITQ (b) PCA-RR (c) PCA (d) LSH (e) SH

Figure 3 Influence to precision of the weighted hamming distance on the Caltech101 dataset using 320-d gist descriptors.

tance of 01 and 10 is 1, but the hamming distance between them is 2. Likewise, the distance between 00 and 11 is 3, while the hamming distance is 2. Thus, we propose a novel WHD to satisfy the distance of DBQ binary codes. For a given query, we first build several look-up tables, and then the WHD is calculated by looking up the precomputed look-up tables. Notes that one dimension refers to 2-bit binary code in the following for clear expression.

Assume *x, q* are the intermediate data of reference signature and query signature respectively. To calculate the weighted hamming distance between x and q online, we should sum up the distance of each dimension composed of 2-bit. Let $d_k(DQ_k(x), DQ_k(q))$ represent the weighted hamming distance between k-th dimension of x and q which is demonstrated as follows:

	00	01	10	11
00	0	1	2	3
01	1	0	1	2
10	2	1	0	1
11	3	2	1	0

Table 1. Weighted hamming distance between each dimension

Online, for a given query q, we precompute and store in look-up tables the following query-dependent values

$$\beta_k^{x,q} = d_k(DQ_k(x), DQ_k(q)) \tag{4}$$

Assume d_{WH} stands for the weighted hamming distance between x and q. By definition, we have

$$d_{WH}(x,q) = \sum_k \beta_k^{x,q} \tag{5}$$

The cost of computing the values β is negligible with respect to that of computing $d_{WH}(x,q)$ for a large number of reference signatures x. The sum (5) can be computed very efficiently by grouping the dimensions. In our implementation, we subdivide a vector into blocks of four dimensions (8-bit). Assuming that the number of dimensions k is a multiple of 4, to simplify the notation, we get

$$d_{WH}(x,q) = \sum_{k=0}^{k/4-1} \sum_{j=1}^{j=4} \beta_{4k+j}^{x,q} \tag{6}$$

Because binary subvector $[DQ_{4k}(x),\ DQ_{4k+1}(x), DQ_{4k+2}(x), DQ_{4k+3}(x)]$ fits in 1 byte, each sum $\sum_{j=1}^{j=4} \beta_{4k+j}^{x,q}$ can only take 256 possible values. We can precompute these 256 values and store them in a look-up table. In total, we need to calculate $k/4$ tables for the computation.

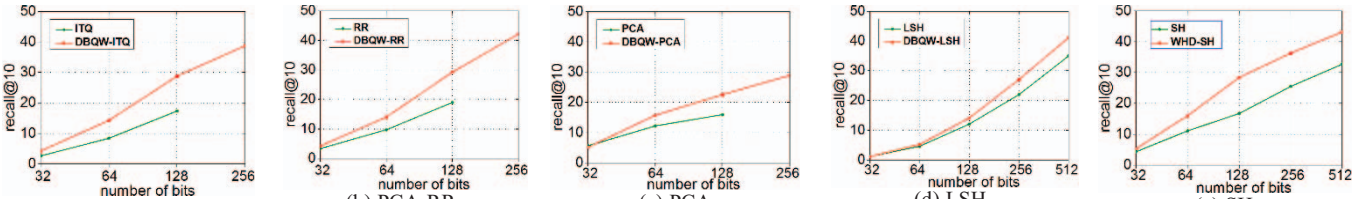
Generally, the $k/4$ look-up tables will be cached since the tables only require little computer storage. Therefore, the weighted hamming distance can be simply calculated by adding values from each look-up table together rather than computing hamming distance for each reference binary code via XOR operation. Undoubtedly, the inquiry speed will be significantly increased in our method. For the consistency of experiments, the comparative experiments we conduct also use look-up tables to compute the traditional hamming distance.

3. EXPERIMENTS

In this section, we show the benefits of the DBQW. We first introduce the two data sets and evaluation metrics used in the experiments. Then intensive comparisons between methods applying DBQW and traditional binary embedding methods are given.

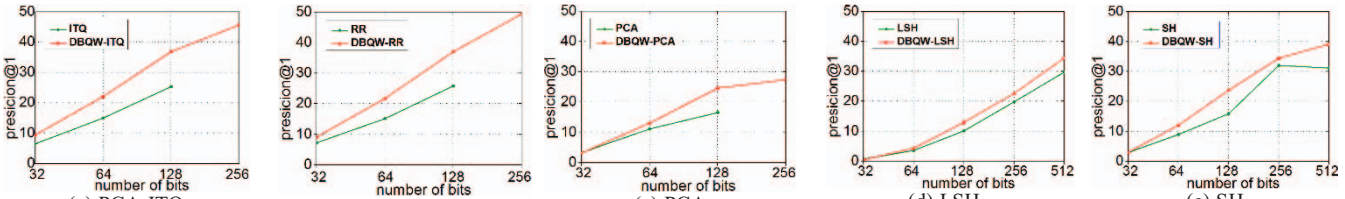
A. Dataset and Evaluation Metrics

We use two different datasets Caltech101 [10] and BIGANN [9] to evaluate our method. The first dataset Caltech101 contains approximately 60,000 images grouped in 101 classes. Through our experiments, we discard the notion of class and extract 320-d GIST for each image in the dataset. Then, we split the data set into three different sets. We randomly select 1,000 GIST to serve as queries, and 5,000 random GIST to serve as unsupervised training data. The remaining signatures are used as database. The next dataset BIGANN contains 1M SIFT [9] signatures which consists three vector subsets: a 100K training set, a 1M database set and a 10K query set. Each SIFT signatures is a 128D real-value vector. On both datasets, we evaluate the retrieval of traditional embedding algorithms and those applying DBQW, using Euclidean neighbors as ground truth. Note that there is no difference in time cost between



(a) PCA-ITQ (b) PCA-RR (c) PCA (d) LSH (e) SH

Figure 4 Influence to recall of the weighted hamming distance on the BIGANN dataset using 128-d sift descriptors.



(a) PCA-ITQ (b) PCA-RR (c) PCA (d) LSH (e) SH

Figure 5 Influence to precision of the weighted hamming distance on the BIGANN dataset using 128-d sift descriptors.

traditional binary embedding methods and those adopting DBQW, as all the methods execute exhaustive search in binary codes of the same dimensionality via look-up tables. In our experiments, the precision and recall with respect to the number of bits are used to evaluate our method. On Caltech101, we first compare the recall@10 and precision@1 of several binary embedding methods, such as PCA, LSH, SH, PCA-ITQ and those applying DBQW. Then we conduct the same experiments as Caltech101 on BIGANN.

B. Results

Each experiment has 1000 queries, in which precision and recall are used as metrics. For the different binary embedding methods in two different datasets, the results using DBQW are better than original binary embedding methods.

To get the double-bit binary codes, the signatures in training set are mapped to the intermediate data, and the medians are obtained according to the sign of each dimension. Then we transform both reference signatures and query signatures to intermediate data in the same way as training set. Next, the data are quantized into double-bit binary codes through DBQW. Finally, the weighted hamming distance is calculated for each binary code.

It show the comparison results on two datasets in Fig 2, 3, 4 and 5 of original binary embedding algorithms and the algorithms with our proposed DBQW. The results demonstrate that DBQW consistently improves the retrieval accuracy over the original binary embedding algorithms and is independent of the datasets, the descriptors, and the binary embedding methods. Here are two examples: On Caltech101 (Fig. 1(a)), we observe a relative improvement on recall of 80.2% at 256-bit when using RR-PCA. On BIGANN, when SH is employed, we can observe a relative improvement of precision of 42.3% (Fig. 4(e)) at 128 bits. The comparison results show that all methods make a great progress with DBQW. This can be explained by two primary factors:

On one hand, double-bit quantization preserves more information of original signatures. The average discrimination per bit declines as the dimensionality increases with one-bit quantization. Here is an example, For ITQ, when the dimensionality k is 128, the precision is 14.2% on Caltech101. Given the same circumstances except for the dimension being $2k$, the precision is 16.9% which only has 19% relative improvement. Thus, the discrimination per bit is stronger when k is smaller. In theory, $2k$-bit binary code using DBQW doubles the retrieval accuracy of k-bit binary code using one-bit quantization. That is to say, when the dimension of binary codes is both $2k$, the discrimination of DBQW significantly outperforms one-bit quantization.

On the other hand, weighted hamming distance has more distinguishing ability. Weighted hamming distance has a wider range than original method. For example, assume the number of bit is 64, the range of hamming distance is 0 to 64. However, the range of weighted hamming distance is 0 to 128, which is twice of the former. As a consequence, DBQW obtains stronger discrimination than traditional binary embedding methods.

4. CONCLUSIONS

In this paper, we propose the novel double-bit quantization and weighting algorithm for nearest neighbor search. To improve retrieval accuracy, each dimension of intermediate data in the dataset is quantized into double-bit binary codes offline. Then look-up tables are precomputed to calculate the weighted hamming distance to get final results. Experimental results show DBQW can achieve about up to 12% absolute enhancements on precision@1 and 16% on recall@10 compared to the original methods. It indicates that DBQW performs competitive results by assigning more weight to signatures close to query in hamming space. We believe the proposed DBQW can improve the accuracy of many nearest neighbor search applications.

5. ACKNOWLEDGEMENTS

This work is supported by the National Nature Science Foundation of China (61303171, 61502477, 61502479), the "Strategic Priority Research Program" of the Chinese Academy of Sciences (XDA06031000, XDA06010703).

6. REFERENCES

[1] Xie H, Gao K, Zhang Y, et al. Pairwise weak geometric consistency for large scale image search[C]// Proceedings of the 1st ACM ICMR. ACM, 2011:1-8.

[2] Zhang L, Zhang Y, Tang J, et al. Topology preserving hashing for similarity search[C]// ACM International Conference on Multimedia. 2013:123-132.

[3] Ding Duo, et al. Beyond audio and video retrieval: towards multimedia summarization. ACM ICMR, 2012.

[4] M. Raginsky and S. Lazebnik, Locality-Sensitive Binary Codes from Shift-Invariant Kernels, Proc. NIPS, pp. 1509-1517, 2009.

[5] A. Gordo and F. Perronnin, "Asymmetric Distances for Binary Embeddings," Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 729-736, 2011.

[6] Y. Weiss, A. Torralba, and R. Fergus, Spectral Hashing,Proc. NIPS, 2008.

[7] Y. Gong and S. Lazebnik, Iterative Quantization: A Procrustean Approach to Learning Binary Codes, Proc. IEEE Conf. CVPR, pp. 817-824, 2011.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 60(2):91–110, 2004

[9] H. Jegou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: re-rank with source coding. In IEEE ICASSP, pages 861–864. IEEE, 2011.

[10] L. Fei-Fei, R. Fergus and P. Perona.Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. IEEE. CVPR 2004