THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Efficient Bootsrapping and Query Adaptive Ranking for Image Search

A. A. R. Senthilkumar
Head of the Department, Department of Master of Computer Application
PGP College of Engineering and Technology, Namakkal
P. Mayuri
Department of Computer Science and Engineering
PGP College of Engineering and Technology, Namakkal

Abstract:
Scalable image search based on visual similarity has been an active topic of research in recent years. State-of-the-art solutions often use hashing methods to embed high-dimensional image features into Hamming space, where search can be performed in real-time based on Hamming distance of compact hash codes. Unlike traditional metrics (e.g., Euclidean) that offer continuous distances, the Hamming distances are discrete integer values. As a consequence, there are often a large number of images sharing equal Hamming distances to a query, which largely hurts search results where fine-grained ranking is very important.I introduces an approach that enables query-adaptive ranking of the returned images with equal Hamming distances to the queries. This is achieved by firstly offline learning bitwise weights of the hash codes for a diverse set of predefined semantic concept classes. We formulate the weight learning process as a quadratic programming problem that minimizes intra-class distance while preserving inter-class relationship captured by original raw image features. Query-adaptive weights are then computed online by evaluating the proximity between a query and the semantic concept classes.With the query-adaptive bitwise weights, returned images can be easily ordered by weighted Hamming distance at a finer-grained hash code level rather than the original Hamming distance level. Experiments on a Flickr image dataset show clear improvements from our proposed approach.
.
Key words: *Query-adaptive image search, scalability, hash codes, weighted Hamming distance*

1. Introduction

The explosion of images on the Internet, there is a strong need to develop techniques for efficient and scalable image search. While traditional image search engines heavily rely on textual words associated to the images, scalable content-based search is receiving increasing attention. Apart from providing better image search experience for ordinary Web users, large-scale similar image search has also been demonstrated to be very helpful for solving a number of very hard problems in computer vision and multimedia such as image categorization.Generally a large-scale image search system consists of two key components—an effective image feature representation and an efficient search mechanism. It is well known that the quality of search results relies heavily on the representation power of image features. The latter, an efficient search mechanism, is critical since existing image features are mostly of high dimensions and current image databases are huge, on top of which exhaustively comparing a query with every database sample is computationally prohibitive.

- **Bag-of Visual-Words(BOW):** I represent images using the popular bag-of-visual-words (BoW) framework where local invariant image descriptors (e.g., SIFT) are extracted and quantized based on a set of visual words. The BoW features are then embedded into compact hash codes for efficient search. For this, we consider state-of-the-art techniques including semi-supervised hashing and semantic hashing with deep belief networks.
- **Hashing :** The main contribution of this paper is the proposal of a novel approach that computes *query-adaptive weights* for each bit of the hash codes, which has two main advantages.
- **Ranking :** Images can be ranked on a finer-grained hash code level since—with the bitwise weights—each hash code is expected to have a unique similarity to the queries. In other words, we can push the resolution of ranking from (traditional Hamming distance level) up to (hash code level1).
- **Hamming Distance :** Contrary to using a single set ofweights for all the queries, our approach tailors a different and more suitable set of weights for each query.

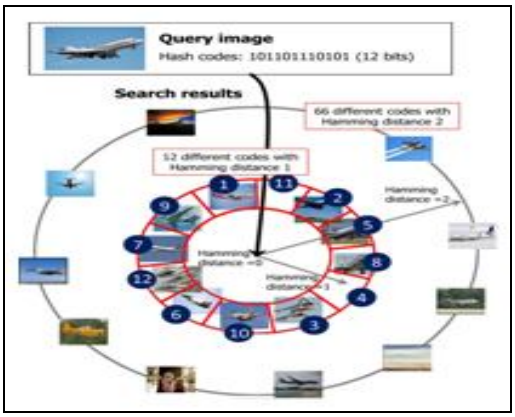

Figure 1

2. Related Work

There are very good surveys on general image retrieval task. See Smeulders *et al*. [11] for works from the 1990s and Datta *et al*. [12] for those from the past decade. Many people adopted simple features such as color and texture in systems developed in the early years [13], while more effective features such as GIST [14] and SIFT [3] have been popular recently [2], [15]. In this work, we choose the popular bag-of-visual-words (BoW) representation grounded on the local invariant SIFT features. The effectiveness of this feature representation has been verified in numerous applications. Since the work in this paper is more related to efficient search, this section mainly reviews existing works on efficient search mechanisms, which are roughly divided into three categories: inverted file, tree-based indexing, and hashing. Inverted index was initially proposed and is still very popular for document retrieval in the informational retrieval community [16]. It was introduced to the field of image retrieval as recent image feature representations such as BoW are very analogous to the bag-of-words representation of textual documents. In this structure, a list of references to each document (image) for each text (visual) word is created so that relevant documents (images) can be quickly located given a query with several words.

A key difference of document retrieval from visual search, however, is that the textual queries usually contain very few words. For instance, on average there are merely 4 words per query in Google web search.2 While in the BoW representation, a single imagemay contain hundreds of visual words, resulting in a large number of candidate images (from the inverted lists) that need further verification—a process that is usually based on similarities of the original BoW features. This largely limits the application of inverted files for large scale image search. While increasing visual vocabulary size in BoW can reduce the number of candidates, it will also significantly increase memory usage Fig. 2. Search result lists in a Flickr image dataset, using a "sunset scene" query image (left). Top and bottom rows respectively show the most similar images based on traditional Hamming distance and our proposed query-adaptive weighted Hamming distance. It can be clearly seen that our method produces more relevant result by ranking images at a finer resolution. Note that the proposed method does not permute images with exactly the same code to the query (three images in total for this query), i.e., . This figure is best viewed in color.

For example, indexing 1 million BoW features of 10 000 dimensions will need 1 GB memory with a compressed version of the inverted file. In contrast, for the binary representation in hashing methods, as will be discussed later, the memory consumption is much lower (e.g., 48 MB for 1 million 48-bit hash codes). Indexing with tree-like structures [7], [18], [15] has been frequently applied to fast visual search. Nister and Stewenius [15] used a visual vocabulary tree to achieve real-time object retrieval in 40 000 images. Muja and Lowe [18] adopted multiple randomized d-trees [7] for SIFT feature matching in image applications.

One drawback of the classical tree-based methods is that they normally do not work well with high-dimensional feature. For example, let the dimensionality be and the number of samples be , one general rule is in order to have d-tree working more efficiently than exhaustive search [19]. There are also several works focusing on improving tree-based approaches for large-scale search [20], [21], where promising image search performance has been reported. Compared with these methods, hashing has a major advantage in speed since it allows constant-time search. In view of the limitations of both inverted file and tree-based indexing, embedding high-dimensional image features into hash codes has become very popular recently. Hashing satisfies both query time and memory requirements as the binary hash codes are compact in memory and efficient in search via hash table lookup or bitwise operations. Hashing methods normally use a group of projections to divide an input space into multiple buckets such that similar images are likely to be mapped into the same bucket.

Most of the existing hashing techniques are *unsupervised*. Among them, one of the most well-known hashing methods is Locality Sensitive Hashing (LSH) [22]. Recently, Kulis and Grauman [23] extended LSH to work in arbitrary kernel space, and Chum *et al*. [24] proposed min-Hashing to extend LSH for sets of features. Since these LSH-based methods use random projections,when the dimension of the input space is high,many more bits (random projections) are needed to achieve satisfactory performance. Motivated by this mainly, Weiss *et al*. [9] proposed a spectral hashing (SH) method that hashes the input space based on data distribution. SH also ensures that the projections are orthogonal and sample number is balanced across different buckets. Although SH can achieve similar or even better performance than LSH with a fewer number of bits, it is important to underline that these unsupervised hashing techniques are not robust enough for similar image search. This is due to the fact that similarity in image search is not simply equivalent to the proximity of low-level visual features, but is more related to high-level image semantics (e.g., objects and scenes). Under this circumstance, it is helpful to use some machine learning techniques to partition the low level feature space according to training labels on semantic level. Several *supervised* methods have been proposed recently to learn

good hash functions [25], [26], [4], [27]–[30], [6]. In [25], Kulis and Darrell proposed a method to learn hash functions by minimizing reconstruction error between original feature distances and Hamming distances of hash odes. By replacing the original feature distances with semantic similarities, this method can be applied for supervised learning of hash functions. In [26], Lin *et al.* proposed to learn hash functions based on semantic similarities of objects in images. In [4], Wang *et al.* proposed a semi-supervised hashing algorithm that learns hash functions based on image labels. The advantage of this algorithm is that it not only utilizes given labels, but also exploits unlabeled data when learning the hash functions. It is especially suitable for the cases where only a limited number of labels are available. The idea of using a few pairwise data labels for hash function learning was also investigated in [27], using an algorithm called label-regularized max-margin partition. In [28], a scalable graph-based method was proposed to exploit unlabeled data in large datasets for learning hash codes. In [29], Bronstein *et al.* used boosting algorithms for supervised hashing in shape retrieval. Boosting was also used by Xu *et al.* [30] who proposed to learn multiple hash tables. In [6], Salakhutdinov and Hinton proposed a method called semantic hashing, which uses deep belief networks [5] to learn hash codes. Similar to the other supervised hashing methods, the deep belief network also requires image labels in order to learn a good mapping. In the network, multiple Restricted Boltzmann Machines (RBMs) are stacked and trained to gradually map image features at the bottom layer to binary codes at the top (deepest) layer. Several recent works have successfully applied this method for scalable image search [8], [31]. All these hashingmethods, either unsupervised or supervised, share one limitation when applied to image search. As discussed in the introduction, the Hamming distance of hash codes cannot offer fine-grained ranking of search results,

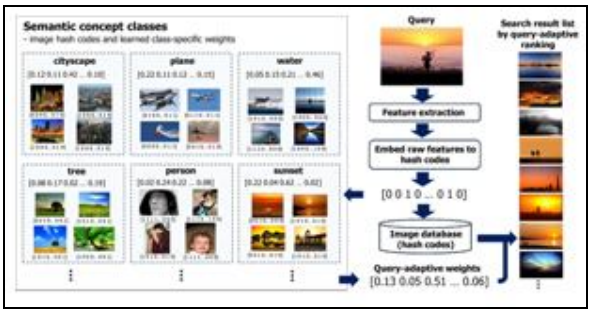

Figure 2

which is very important in practice. This paper proposes a means to rank images at a finer resolution. Note that we will not propose new hashing methods—our objective is to alleviate one weakness that all the hashing methods share particularly in the context of image search.

3. Implementation

As stated earlier, given all , the quadratic program in (10) is convex. Solving it with global minimization w.r.t. will always reduce the value of the energy function in (4), which will definitely not be greater than the energy value derived from a previous iteration. In addition, it is obvious that the energy function is lower-bounded since both terms of the function are non-negative (recall the definitions in (1) and (3)). Therefore, the iterative optimization process given in Algorithm 1 is a Block Coordinate which gradually reduces the energy and leads to convergence at a non-negative value. In practice, to avoid long time convergence procedure, we define an empirical stop condition (convergence) when the energy difference of two successive states , where and is set as a small value (in our experiments). Having such a stop condition can help avoid unneeded deep optimization that leads to almost invisible changes to the bitwise weights.

Connections to Existing Algorithms: We briefly discuss the differences and connections of our proposed algorithm to some well-known machine learning methods such as LDA [39] and distance metric learning, although ours is particularly designed for this specific application. LDA is a well-knownmethod that linearly projects data into a low-dimensional space where the sample proximity is reshaped to maximize class separability. While LDA also tries to condense samples from the same class, it learns a single uniform transformation matrix to map all original –dimensional features to a lower -dimensional space. In contrast, we learn a -dimensional weight vector separately for each class Distance metric learning tries to find a metric such that samples of different classes in the learned metric space can be better separated. Typically a single Mahalanobis distance metric is learned for the entire input space [40]. There are also a few exceptions where multiple metrics were considered, e.g., a metric was trained for each category in [41]. These methods are relevant in the sense that they also deal with multiple categories separately. Nevertheless, they cannot be directly applied to our problem, because the class-specific bitwise weights are in a very different form from that of distance metric matrices.

Computing Query-Adaptive Bitwise Weights As described in Fig. 3, images and the learned weights of the predefined semantic concept classes form a *semantic database* for rapid computation of the query-adaptive bitwise weights.

Given a query image and its hash codes , the objective here is to adaptively determine a suitable weight vector , such that weighted Hamming distance (WHD) can be applied to compare with each hash code in the target database: (14) To compute , we query against the semantic database based on typical Hamming distance. Semantic labels of the top- most similar images are collected to predict query semantics, which is then utilized to compute the query-adaptive weights. Specifically, denote as the set of the most relevant semantic classes to the query , and as the number of images (within the top list) from the th class in . The query adaptive weights are computed as (15) where is the precomputed weight vector of the corresponding class. We expect that,

although top results from typical Hamming distance lack in good ranking, query semantics may be inferred by collectively using a large pool of samples. We empirically set as 3 throughout the experiments. The right side of (14) can be rewritten as , where means the XOR of the two hash codes. While the weighting can now be achieved by an inner-product operation, we can avoid a significant portion of the computation in practice. The idea is very simple—since a common way of using hashing in a real system is to retrieve images locating only within a certain Hamming radius to a query, we can first find a subset of the hash codes with non-weighted Hamming distance smaller than a certain value to the query (e.g.,). Then, each of the retrieved hash code in the thresholded subset is re-ranked by computing the weighted Hamming Distance to the query. In other words, we do not need to order all the hash codes in practice; only the small subset of hash codes with a few bits different from the query is sufficient for pooling the top search results of a query image. In this way, it's possible that some of the hash codes are incorrectly excluded from the initial subset. But the true top matched hash codes with the shortest weighted distances are preserved.

Extension to Query-Adaptive Hash Code Selection

The above approach is designed to use a single set of general hash codes for image search. In this subsection, we further extend our approach to the case where multiple sets of hash codes are available. Since there are multiple semantic concept classes, it is very easy to train a set of hash codes for each class by extensively using images containing the corresponding concept. The class-specific hash codes are expected to bemore discriminative for a particular class of images. The weight learning algorithm introduced in Section I can be applied to each set of class-specific hash codes to produce a separate group of bitwise weights. During online search, the semantic database is used not only for computing query-adaptive weights, but also to select a suitable set of class-specific hash codes for each query. Fig. 4 depicts the extended framework. Given a query image, we first compute its *general* (globally trained) hash code and query against the semantic database. Like the query-adaptive weight computation process introduced earlier, we use the same set of images to predict query semantics. Hash codes trained for the dominant concept class in this selected set will be chosen. The query is then embedded into the selected class-specific hash code using hash functions trained for the corresponding concept class. Finally, the new hash code is concatenated with the general are sorted at binary code level based on bitwise weights predicted from a similar procedure as described in SectionV.B. This new framework provides ameans to adaptively select both hash codes and bitwise weights. Additional computation is required in this extended framework as the query needs to be hashed twice and more bits are used in search. Fortunately, since both testing process (hash code computation) of the hashing algorithms and logical XOR operations are very efficient, this added computational cost is acceptable in general.

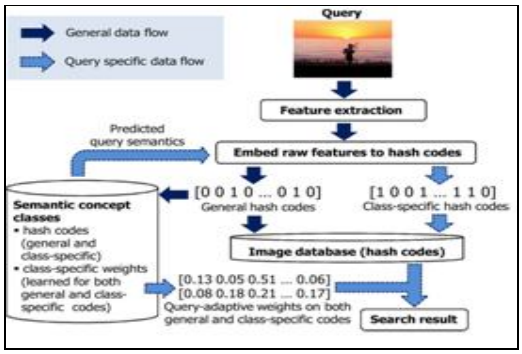

Figure 3

Figure 4

4. Conclusion

A novel framework for query-adaptive image search with hash codes. By harnessing a large set of predefined semantic concept classes, our approach is able to predict query-adaptive bitwise weights of hash codes in real-time, with which search results can be rapidly ranked by weighted Hamming distance at finer-grained hash code level. This capability largely alleviates the effect of a coarse ranking problem that is common in hashing-based image search. Experimental results on a widely adopted Flicker image dataset confirmed the effectiveness of our proposal. To answer the question of "how much performance gain can class-specific hash codes offer?" Query-adaptive hash code selection. Our findings indicate that the class-specific codes can further improve search performance significantly. One drawback, nevertheless, is that nontrivial extra memory is required by the use of additional class-specific codes, and therefore we recommend careful examination of the actual application needs and hardware environment in order to decide whether this extension could be adopted.

5. References

1. A. Torralba, R. Fergus, and W. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," IEEE Trans. Pattern Anal.Mach. Intell., vol. 30, no. 11, pp. 1958–1970, Nov. 2008.
2. J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in Proc. IEEE Int. Conf. Computer Vision, 2003.
3. D. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vision, vol. 60, no. 2, pp. 91–110, 2004.
4. J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2010.
5. G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, no. 5786, pp. 504–507, 2006

6. R. Salakhutdinov and G. Hinton, "Semantic hashing," in Proc. Workshop of ACM SIGIR Conf. Research and Development in Information Retrieval, 2007.
7. J. L. Bentley, "Multidimensional binary search trees used for associative searching," Commun. ACM, vol. 18, no. 9, pp. 509–517, 1975.
8. A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2008.
9. Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in Adv. Neural Inf. Process. Syst., 2008.
10. J. Wang, S. Kumar, and S.-F. Chang, "Sequential projection learning for hashing with compact codes," in Proc. Int. Conf.Machine Learning, 2010.
11. A. W. Smeulders,M.Worring, S. Santini, A. Gupta, and R. Jain, "Content- based image retrieval at the end of the early years," IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 12, pp. 1349–1380, Dec. 2000.
12. R. Datta, D. Joshi, J. Li, and J. Z.Wang, "Image retrieval: Ideas, influences, and trends of the new age," ACM Comput. Surveys, vol. 40, no. 2, 2008.
13. J. R. Smith and S.-F. Chang, "VisualSEEk: A fully automated contentbased image query system," in Proc. ACM Int. Conf.Multimedia, 1996.
14. A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," Int. J. Comput. Vision, vol. 42, pp. 145–175, 2001.
15. D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2006.
16. J. Zobel and A. Moffat, "Inverted files for text search engines," ACM Comput. Surveys, vol. 38, no. 2, 2006.
17. H. Jegou, M. Douze, and C. Schmid, "Packing bag-of-features," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.
18. M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in Proc. Int. Conf. Computer Vision Theory and Applications, 2009, pp. 331–340.
19. P. Indyk, , J. E. Goodman and J. O'Rourke, Eds., "Nearest neighbours in high-dimensional spaces," in Handbook of Discrete and Computational Geometry. Boca Raton, FL: CRC, 2004, ch. 39