Beyond "Near-Duplicates": Learning Hash Codes for Efficient Similar-Image Retrieval

Shumeet Baluja and Michele Covell
Google Research, Google Inc., 1600 Amphitheatre Parkway, Mountain View CA 94043
shumeet@google.com covell@google.com

Abstract

Finding similar images in a large database is an important, but often computationally expensive, task. In this paper, we present a two-tier similar-image retrieval system with the efficiency characteristics found in simpler systems designed to recognize near-duplicates. We compare the efficiency of lookups based on random projections and learned hashes to 100-times-more-frequent exemplar sampling. Both approaches significantly improve on the results from exemplar sampling, despite having significantly lower computational costs. Learned-hash keys provide the best result, in terms of both recall and efficiency.

1. Introduction

The ability to efficiently retrieve similar images from a large database can be useful in a wide variety of applications, including search-by-example on large image sets, example-based image compression and enhancement, and texture synthesis. In all cases, we want to retrieve the most similar image from a large, distributed database of example images. Extensive research has been conducted on nearest-neighbor retrieval [1], with a large focus on tree-based space carving, either spherical or rectilinear.

One simple, but expensive, type of space carving is to use a fixed subset of exemplars from the image set in a Voronoi tessellation. With this system, the exemplar set must be large enough to provide good matches across the full image space. This type of space carving is attractive for its guaranteed limits on the probe distance to the retrieved result, compared to that of the actual nearest-neighbor, but it is impractical for a production system due to the large number of exemplar-probe distances that must be computed if a suitable covering of the expected images is used. However, because of the simplicity of the approach and since this approach provides a set of retrieval guarantees, it is used as a baseline.

We compare this baseline to two different two-tier systems [2]. In a two-tier system, the first tier selects a very small subset of the full database to request for further examination. This small subset is then retrieved from the distributed database and a second tier of examination is done (on the smaller subset), using the same, potentially expensive, thumbnail comparisons as used in a fixed-exemplar system.

Unlike the fixed-exemplar approach, a two-tier system has the opportunity to return any entry from the full database. To permit this, however, the first-tier decisions must be efficiently made, since they are on the full database size. Additionally, for deployment in real systems, the first-tier entry representation must be compact and its subsequent pruning must be both quick and accurate, since the number of examples examined in the second tier must be minimal. These requirements lead us to use compact signatures with Locality Sensitive Hash (LSH) table entries as our only representation for the full database in the first tier.

LSH tables are used extensively for near-duplicate retrievals [2][3]. Each database entry (images, for this application) is processed to obtain a compact and robust signature vector. The signature vector is then projected into multiple subspaces, with each projection creating an LSH key. The subspace projections often are simply disjoint subsets (or "subbands") of the signature dimensions. Retrieval uses the same banding on the probe signature and collects the results across bands. The collected result list is often long but is guaranteed to contain "near" neighbors – all entries with at least one matching signature subband.

For efficiency, we use LSH tables for similar-image retrieval. However, because the task of finding similar images – images visually similar to the probe image – is more difficult than finding near-duplicates (in which the goal is commonly to find the *same* image, with small amounts of noise, scaling and encoding artifacts), we use more subbands, each based on smaller subband keys, and employ a learning approach to create the hash keys. To avoid requesting too many candidates from the distributed database, each candidate is ranked by the number of matching LSH subbands it shares with the probe. This provides a

rapid approximation of the Hamming-distance rank of the candidate list, without database image (or even signature) examination, and is solely based on the hash tables in the distributed first-tier.

Section 2 provides details about our experimental set up, including the evaluation methods. Sections 3 and 4 present alternative approaches to generating the image signatures, for use in the first tier of the two-tier system. In Section 3, we consider random-projection-based signatures and, in Section 4, we describe learned-category-based signatures. In Section 5, we introduce two improvements, the first improves recall and the second controls the LSH retrieval size.

2. Experimental Setup

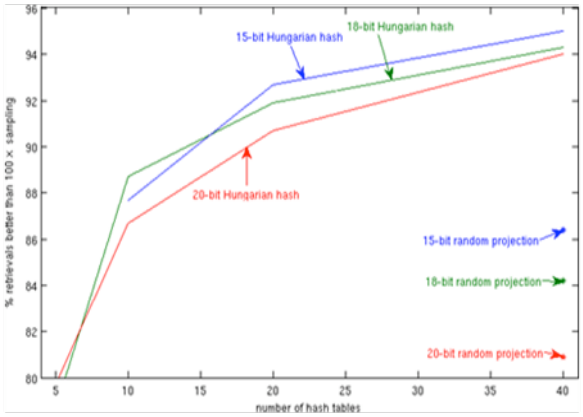
Throughout this paper, similar image retrieval is tested using a database of 2,400,000 images and collecting statistics across 1000 probe images. The database images are collected from product searches (800,000) and randomly from Google's image search (1,600,000). For those experiments that require training data, we use training images that are disjoint from our testing sets. For unlabeled training data, we simply use 450,000 image-search results. For training data with similar-image labels, we use 20,000 from that training set as "probe images" and mark the 10 nearest neighbors to each of these probe images accordingly (allowing overlaps in the labels).

For evaluation, similarity is based on the distance between the closest retrieved image and the probe image. We measure this distance by combining distances across RGB low-resolution (24x24 pixel) space and (grayscale) wavelet space. Our distance is the L_2 difference between these 24x24x4 thumbnails (R,G,B & maximally decimated Haar wavelet). Many distance metrics could be used; here, we employ a simple, easily reproducible one. The quality of the returned result is based on the *single closest image returned by each method.* For our use, this is an important measurement since often only the first search result is used for image modeling or is examined by users of web-based image search. For recall, we report how often, across our 1000 probe set, the two-tier approach returned a closer image result than an approach based on fixed exemplars. We allow *100*× more sampling of the fixed exemplars (10,000 exemplars total) than with the two learning approaches (100 retrieved exemplars). The restriction that the second tier of the two-tier system considers only 1% of the number allowed for the fixed exemplar images, makes this a difficult test, and easily compensates for the overhead of hash-signature computation.

3. Random-projection-based Hash Keys

The first approach is projection onto randomly oriented planes. This method is widely used in nearest-neighbor retrieval due to its average-case retrieval complexity. Each projection is taken onto an independent 24x24x4 random vector, where each vector coefficient is independently sampled from a zero-mean, unit-variance Gaussian distribution. The median of each projection is found on our training set. That median value is used as a threshold, giving a balanced bit per projection. This process is repeated to give a signature for each image. The results are shown in Figure 1 as "random projection." The signature is then divided into sub-bands, and LSH used for retrieval. The results are better than the fixed-exemplar approach for 81% to 87% of the probe tests, depending on the key length. As the number of bits increases, the performance decreases: with more bits, too close a match is required and "similar" (but non-duplicate) images do not match enough bits/subbands. However, with a smaller number of bits, note that the number of candidates before tier-2 examination is much larger because many entries share short sections of the image signature with the probe.

Using a two-tier system with hash-keys created from random projections does significantly better than the fixed-exemplar approach. This result is a significant achievement, since the total computation used to find the random-projection result is less than 10% of that used for the fixed-exemplar approach,



Average retrieved list length (per hash table)	15 bits	18 bits	20 bits
random projection	5279	2153	1247
Hungarian hash	2451	1038	611

Figure 1: Two-tier hash retrieval compared to *100*× fixed-exemplar retrieval. X-axis: number of hash sets. Y-axis: % of retrievals with better (more similar) results than *100*× sampling using fixed exemplars. Table: average retrieved-lists length.

SUBROUTINE LearnGrouping
  ASSIGN 2^8 categories to 2^8 unique labels, at random
LOOP until iteration limit:
  FOREACH bit in 8-bit label code
    LEARN assigned bit value on training data, selecting 1024 boosted weak classifiers using Adaboost
  END foreach bit
  EVALUATE the association strength of each 2^8 categories with all 2^8 labels, using weighted average bitwise Hamming similarity between 8 strong-Adaboost classifier outputs and 8-bit codes, weighted by classifier margin
  REASSIGN all 2^8 categories to 2^8 labels, using Hungarian (stable bi-partite) matching
END loop
END subroutine

FOR 250 iterations:
  COLLECT 8 1-bit classifiers from LearnGrouping
END for 250 iterations
FOR number of bits needed
  MOVE 1 bit from available pool to hash key, based on current bit sampling criteria
END for number of bits

Figure 2: Forgiving-hash learning

even with the first-stage probe-image signature computation. In the fixed exemplar case, each of the 10,000 fixed exemplars requires an L_2 distance computation across (24x24x4)=2304 dimensions. Even with the **first-stage** overhead of the signature generation (600-800 projections * 2304 dimensions), the reduced number of L_2 distance computations in the **second-stage** from 10,000 to 100 (# of examples fully examined in the second tier) is significant. This savings, combined with better retrieval results, provides strong support for these two-tier methods.

4. Learned-category Hash Keys

Our second two-tier retrieval system builds on previous work in learning hash keys, termed Forgiving Hashing [4]. The approach approximates an underlying data/label-manifold using a combination of balanced decision boundaries where each boundary is placed according to a subset of the total training data. We learn binary decision boundaries to separate images into nearest-neighbor categories, while maintaining the entropy properties described below.

We employ a large number of nearest-neighbor categories (20,000) and small amount of images per category (10). The small number of examples in each class could lead to poor decision boundaries. To avoid this, we use the approach outlined in Figure 2. We repeatedly (M=250) take a random subset of 2^N training categories (N=8), so that the decision boundaries across this training subset is specified by N binary classifiers. To start, we make a random assignment of the 2^N training groups to the 2^N N-bit codewords. Since we learn our decision boundaries at a single bit level, this yields $10*2^{N-1}$ training examples per binary category.

To learn the codewords, each single-bit of the codeword is a separate 'strong' Adaboost classifier. Each strong classifier is comprised of 1024 weak classifiers. The population of weak classifiers considered for inclusion into the set of 1024 is based on a pair of averages taken from distinct rectangular regions within the 24x24x4 thumbnail data. Each weak classifier is defined by the corner locations for these two rectangles, by an operator used to combine these average values, and by a threshold value for changing the combined value into a single bit. The Adaboost approach of [7] is used to set these thresholds to minimize that weighted error distribution at the time of evaluation. We consider 3 combination operators: ratio, sum, and absolute difference. Together this provides a pool of more than a million possible weak classifiers. Since this pool is too large to evaluate at each of the 1024*N*M Adaboost steps, we evaluate only a tiny fraction of them: 500 randomly selected weak classifiers, with new random selection at each stage. In each iteration, out of the 500 evaluated, the weak classifier that best reduces the residual error is selected for the strong classifier being trained. This sparse sampling approach has been shown to work effectively in vision-based classification problems [5].

Even though the actual learning occurs at a single-bit level, our requirement that the selected training categories be evenly split across 2^N codewords means the assignment of targets occurs within an N-bit context (and is repeated M=250 times, using different category combinations). We start to learn a specific (initially random) assignment of 2^N categories to 2^N N-bit codewords. Since we have no *a priori* way to create an assignment learnable in N binary decision boundaries, we allow the *reassignment* of labels.

Reassignment is shown in the learn-reassign loop in the *LearnGrouping* subroutine in Figure 2. In this loop, we first try to learn our current assignment. After training, we measure the error for each category. Specifically, we evaluate how well each of the 2^N categories associate with the 2^N N-bit codewords, using the similarity between the N learned outputs and the current N-bit codeword, averaged over the 10 training examples and over the N bits and weighted by the decision margin for each of the example output bit. We then create a new set of 2^N category-codeword assignments, based on this "affinity" between the selected categories and the codewords. In the reassignment, all labels are again used and each

category is assigned a unique label. This ensures maximal entropy of each bit (1.0), in the target labels, across the 2^N categories. Since each example within a category is assigned the same label, the target within-class entropy is minimal (0.0).

In previous work [4], a greedy approach was used for reassignment. We improve on that approach by using a stable-matching algorithm for bi-partite graph labeling, also known as the Hungarian algorithm [6], to find the best match in (at most) m^2 proposed category-codeword pairings, where $m=2^N$. In the Hungarian algorithm, at each step, an un-coded category proposes itself to the codeword to which it has the strongest affinity and has not yet proposed. If that codeword is already used for another category, to which its affinity is stronger, the codeword refuses the new proposal. Otherwise, the codeword breaks off any previous assignment (with the jilted category returning to the un-coded state) and the codeword re-associates itself with the current proposing category.

After the bits are learned (in 250 groups of 8 bits each), we select from this combined pool of N*M (N=8, M=250) bits to create our hash keys. For the results reported in this section, we select the bits that will be used for the hash key randomly (without replacement). The results from this approach are shown in Figure 1 as "Hungarian hash".

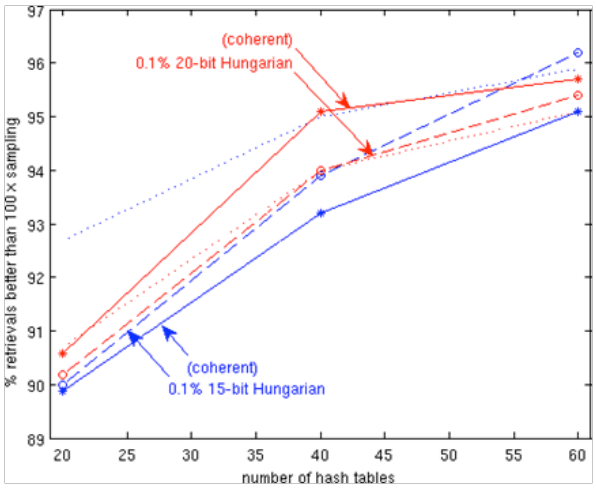
These results all significantly improve over the fixed-exemplar retrieval, with closer retrieval results for more than 90% of our probes, as long as at least 20 hash tables are used. The approach achieves these superior results at a much lower computational cost than the fixed-exemplar approach, since the weak classifiers can be computed efficiently, using a shared integral-image representation [7]. Furthermore, the results are all significantly better than those obtained through random projections (and at a lower cost). We consistently match or exceed the performance of random projection, using $1/4^{th}$ of the memory that was needed for that alternative: it takes ¼ the number of hash tables (and ¼ of the lookups) for Hungarian hash to achieve the performance of random projection. Due to the performance improvements in memory and recall rates, the remainder of the discussion focuses on Hungarian hashes, although the same improvements are applicable to random-projection keys.

5. Improvements to Two-Tier Retrieval

To this point, we have demonstrated the strength of two-tier approaches, in particular the Hungarian hash learning procedure – strong retrieval result without the need for keeping an explicit signature for any of the database entries. No signature is required since the evidence for each entry in the database is measured by the number of matching subbands in the first-tier. In this section, we examine two improvements to the basic two-tier approach: first, to control the size of the hash retrievals and second, to improve the within-category coherence within the hash tables.

5.1 Soft limiting retrieval size

When we examine our retrieval results, the sometimes long likely-candidate lists are due to a small number of table lookups in which the bins are disproportionately highly populated. This occurs because the image population clumps together within the individual hash tables. That clumping occurs in different parts of the image population within the separate tables, so only a small number of the hash lookups for any one probe image will hit a clump. We can take advantage of this incoherence to reduce the likely-candidate list length. We first retrieve the bin sizes for all of the lookups needed for a probe. We then test the bin occupancy against some threshold occupancy (*e.g.,* 0.1% of the full database) and only count the below-threshold retrievals. If too few of the bins are below our threshold (*e.g.,* fewer than ten), then we dynamically adjust the threshold upward, until at least 10 bins are counted. These results are shown for



Average retrieved list length (per hash table)	15 bits	18 bits	20 bits
original Hungarian hash	2451	1038	611
using coherent bits	7758	3460	2177
0.1% Hungarian hash	550	424	325
0.1 & using coherent bits	670	489	421

Figure 3: Retrieval limits and coherent-bit bias in Hungarian hash retrieval (compared to 100x random retrieval). The x- and y-axes are as in Figure 1 (over different ranges). Dashed lines: randomly selected hash keys with a 0.1% bin-size limit. Solid lines: coherency-selected hash keys with the same bin-size limit. Dotted lines: original Hungarian hashing results from Figure 1. Random projection results are below shown range (not shown).

15- and 20-bit hashes with dashed line plots in Figure 3 with the labels "0.1% X-bit Hungarian".

Comparing these results to the original approach (Figure 1 or dotted lines in Figure 3), we lose some recall by limiting the retrieval size: the loss is largest for the short hash keys and for the smaller numbers of hash tables; this is where the limits have a greater impact on the ranking the likely candidates. Nonetheless, the advantage of employing a hash-bin size limit is that it significantly reduces likely-candidate list lengths, down by 50% to 75% of those without using limits. This behavior (a small recall loss for an extreme reduction in the number of likely candidates listed for voting) suggests that the areas in the hash tables where the worst clumping occurs are providing little information about image similarity.

5.2 Selecting for within-category coherence

The second improvement explicitly improves the stability of hash keys on similar images by using only those bits that best keep the *within-category* training images together. Each bit has been trained to minimize within-class entropy while maintaining between-class entropy, but only for a small subset of the training set (just 256 classes out of 20,000). With a total of 2000 bits in the trained pool, none of our tests will use all of the available bits. We can bias our selection of hash bits according to their performance over the *full* training set. To do this, we evaluated the *within*-category entropy of each bit across the full training set (not just the subset of examples it was trained on) and then selected bits for the hash keys according to this ranking. Only bits with the lowest within-class entropy, averaged over all training classes, were used.

When this heuristic was used without retrieval limits, this coherent-bit-selection bias resulted in improved recall: there was a consistent 1-2% absolute improvement in the recall across all of the approaches using 20 or more hash tables and 15- or more bits per hash key. However, this improvement in recall incurred long likely-candidate lists: the list lengths grew by 200% to 300% from those shown in Figure 3.

When the two modifications (retrieval-size limits and coherence-selected bits) are combined, we get our best results. As shown by the "(coherent)" label modifiers in Figure 3, our recall performance for 18- and 20-bit Hungarian hash keys improves significantly, most obviously for the 40 hash table set up. The most interesting effect is that recall does not degrade (and even slightly *improves*) for the longer hash keys, even though we have many fewer likely candidates under consideration. At 40 hash tables, the best results from this combined approach (using 20-bit hashes) achieves the same recall performance as the best results from the original approach (which used 15-bit hashes), while examining approximately $1/6^{th}$ of the number of likely candidates needed for the original best approach (utilizing 15 bits).

6. Larger Tests and Conclusions

In this paper, we examined efficient techniques for similar-image retrieval over large, distributed databases. Our approaches balanced bandwidth requirements, computational costs, and local-memory usage. In these tests on a diverse 2.4-million image database, we improved on fixed-exemplar space carving in 95% of our probe tests, while using only 1% of the computational cost of that approach. As a final test, to ensure that we did not over-engineer this system specifically for the set of 2.4 million images we used for the experiments presented, we repeated the tests by increasing our database size to 5.8 million images. This strengthened our findings: the learned-hash key approach is even more effective (improving on fixed exemplars in 97% of our probes), as well as computationally efficient (maintaining the same 1% computational ratio).

Our future investigations will first focus on reducing the local memory usage for the learned-hash approach. We currently keep the fully populated hash tables, even when using retrieval limits, due to our requirement to retrieve a fixed minimum number of bins. We would like to reduce the retained size of these over-full bins. One interesting approach to this is to use within-bin clustering, based on the full signatures of the referred-to elements. Further work is needed to be able to do this clustering without adding significant memory overhead. Second, for these experiments, an L_2-based measure of similarity was chosen for reproducibility and simplicity. It is just one of many similarity measures that can be incorporated into the learning approaches; the use of higher-level features, such as local features, as inputs into the learning procedures is currently under investigation.

References

[1] P. Yianilos. "Data structures and alg. for nearest-neighbor search in general metric spaces" *ACM-SIAM Symp. Disc. Alg.* 1993.

[2] M. Covell, S. Baluja. "Known-Audio Detection using Waveprint: Spectrogram Fingerprinting by Wavelet Hashing", *ICASSP*, 2007.

[3] Y. Ke, R. Sukthankar, L. Hustonet "Efficient Near-duplicate Detection and Sub-Image Retrieval", *ACM Int. Conf. on Multimedia*, 2004.

[4] S. Baluja, M. Covell. "Learning to Hash: Forgiving Hash Applications", *Data Mining and Knowledge Discovery*, 2008.

[5] S. Baluja, M. Covell. "Finding Images and Line Drawings in Document-Scanning Systems", *ICDAR*, 2009.

[6] H.W. Kuhn, "Variants of the Hungarian method for assignment problems", *Naval Research Logistics Quarterly*, **3**: 253–258, 1956.

[7] P. Viola, M. Jones. "Robust real-time object detection", *Workshop of Statistical & Computational Theories of Vision*, 2001.