Few-Shot Hash Learning for Image Retrieval

Yu-Xiong Wang Liangke Gui Martial Hebert
School of Computer Science, Carnegie Mellon University
{yuxiongw,liangkeg,hebert}@cs.cmu.edu

Abstract

Current approaches to hash based semantic image retrieval assume a set of pre-defined categories and rely on supervised learning from a large number of annotated samples. The need for labeled samples limits their applicability in scenarios in which a user provides at query time a small set of training images defining a customized novel category. This paper addresses the problem of few-shot hash learning, in the spirit of one-shot learning in image recognition and classification and early work on locality sensitive hashing. More precisely, our approach is based on the insight that universal hash functions can be learned off-line from unlabeled data because of the information implicit in the density structure of a discriminative feature space. We can then select a task-specific combination of hash codes for a novel category from a few labeled samples. The resulting unsupervised generic hashing (UGH) significantly outperforms current supervised and unsupervised hashing approaches on image retrieval tasks with small training samples.

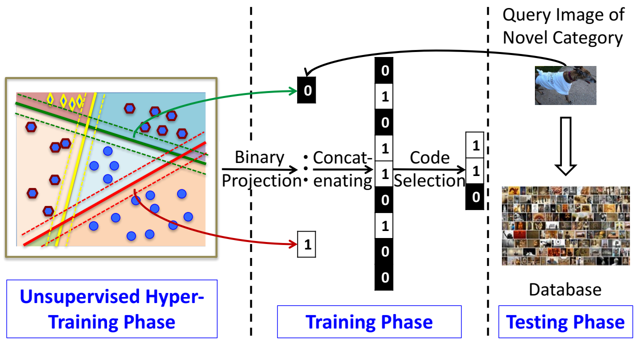


Figure 1: Few-shot hash learning for the retrieval of novel categories from few examples. During the unsupervised hyper-training phase, a large library of hash functions informative across categories is generated. For a new target task or category with limited samples, task-specific hash functions are selected from this library to form its compact binary representation during the training phase. Finally, the image retrieval of novel object categories is accomplished by nearest neighbor search in the Hamming space.

1. Motivation

With the rapidly increasing amount of visual data on the web, binary hashing, due to its computational and storage efficiency, has attracted considerable attention for representation and retrieval in large-scale image databases [51, 25, 60, 77, 76, 19, 20, 53]. To encode high-dimensional image data into compact and *semantic similarity* preserving binary codes in a Hamming space, current hashing approaches focus on scenarios that assume a set of pre-defined categories and rely on large annotated data. In practical applications, however, a user might define customized query categories *on-the-fly* by supplying only a *small set* of specific examples, and requires the entire learning and retrieval procedure to be manageable in real time [13, 22, 40, 65, 12, 11, 27].

Such *few-shot hash learning* scenarios pose a significant challenge for the existing techniques, since they are usually category/dataset specific and cannot generalize well from few examples or generalize to novel categories. In this paper, we attempt to design a system to facilitate the hash learning for novel object categories from few examples. While one/few-shot learning [21], as a fundamental problem, has been extensively discussed in the context of image recognition and classification [71, 66], very little work has addressed this issue for hash learning and image retrieval.

Specifically, the state-of-the-art hashing approaches are data-dependent and directly learn hash functions from the target dataset in either an unsupervised or supervised manner. The unsupervised hashing [72, 32, 25] aims to propagate neighborhood relation of samples from a certain metric space into the Hamming space. However, distance metrics (*e.g.*, Euclidean distance or angular distance) typically cannot measure well the semantic similarity that is essential for image retrieval. By leveraging supervisory information in form of class labels, supervised hashing [67, 51, 58, 60] preserves semantic structure of the data. Unfortunately, with limited data, these approaches are prone to over-fitting, leading to degenerated performance.

Interestingly, the early research on data-independent hashing has learned generic binary codes that are independent of the data and categories. The flagship representatives, the locality sensitive hashing (LSH) and its variants [23, 10, 31, 33, 55], use simple random projections to construct hash functions without exploring the data distribution. However, LSH usually requires long binary codes to achieve satisfactory retrieval accuracies, leading to large storage space and low recall performance [51]. Due to its pure data-independence, LSH still lacks the ability to preserve the desired semantic similarity.

In the spirit of learning classifier-based representations, some approaches, including Classemes [64], PiCoDes [6], Meta-Class [4], and predictable discriminative binary code (DBC) [58], leverage an *auxiliary labeled* dataset and generate codes either from pre-defined categories or learned super-classes of these categories. Unfortunately, to identify properties shared by many categories, these approaches rely on a large corpus of annotated auxiliary data samples and expensive training iterations. The generalization ability is still tied to this particular set of categories due to its supervised nature. In particular, the code length is usually constrained by the dimension of the original feature descriptors or number of categories [5].

To address these limitations, we propose a basic framework for generating *unsupervised generic hashing* (UGH), which frees the hash learning from ties to a specific set of categories and which can be transferable to those categories unknown in advance and with few samples. Distinguished from previous work, we design a *three-phase procedure* in this approach, as illustrated in Fig. 1. First, given a large corpus of *unlabeled* data samples, we produce a large library of "expressive" hash functions, each of which converts original features to a one-bit binary code. This *off-line* phase is called "hyper-training" by analogy with training hyper-parameters as in [68, 26]. Now, given a new target task with few samples, task/category-specific codes are selected from the large pool to form a compact representation of the novel category. Finally, the retrieval is accomplished by nearest neighbor search in the Hamming space.

Intuitively, to make our library of unsupervised hash codes generalizable across categories, a discriminative feature space is required in the first place. We use the activations of convolutional neural networks (pre-trained CNNs on the labeled ImageNet dataset) [38] as the feature space. Because they are learned from visual data, the CNN features carry information about the semantic structure of the feature space. More importantly, in the spirit of LSH, a single code should be informative by itself while the entire library of codes should have a good coverage of the feature space [23, 33, 55]. Our key observation here is that such expressive codes could be generated without supervision because of the information implicit in the *density structure* of the feature space. To this end, our unsupervised hyper-training uses a large corpus of unlabeled images as a much less biased sampling in the feature space. We then introduce a series of simple sampling procedures, and iteratively estimate diverse, salient *pseudo-categories* that are surrogates for plausible categories and produce hash functions that traverse across the corresponding low-density regions in the unlabeled data.

After producing a large library of codes during hyper-training, later in the training phase, we infer what is shared and discriminative between categories by selecting the most informative bits for novel categories and tasks. Separating unsupervised code generation and task-specific code selection also makes the code length adaptive for different categories/tasks and makes the use of long codes feasible, which facilitates the practical usage of the hash codes [75].

Our contributions are three-fold. (1) We show how such a three-phase setup can be operationalized for learning hash functions across datasets and categories. To the best of our knowledge, it is the first time to explore the few-shot hash learning problem for image retrieval. (2) We show how a universal library of hash codes (UGH), based on combining pre-trained CNN features and *unsupervised* hash learning, is generated without bias to a particular set of categories. (3) We show how informative codes are selected on novel categories with few examples and how image retrieval tasks can be achieved efficiently. In particular, our UGH outperforms state-of-the-art hashing baselines by significant margins for small samples when they are learned in the same CNN feature space.

2. Related Work

The existing hashing approaches can be divided into two categories: data-independent and data-dependent. The data-independent hashing approaches, such as a family of methods known as locality sensitive hashing (LSH) [23, 10, 31, 18, 57] and Min-Hash [8, 15], randomly generate a set of hash functions without any training and use these scattered codes to achieve asymptotically guaranteed performance. The data-dependent (or learning-based) hashing approaches learn similarity-preserving hash functions with compact codes in an unsupervised or (semi-)supervised manner. The unsupervised methods, such as spectral hashing (SH) [72], anchor graph hashing (AGH) [52], spherical hashing (SPH) [30], K-means hashing (KMH) [29], iterative quantization (ITQ) [25], locally linear hash (LLH) [32], discrete graph hashing (DGH) [50], and binary autoencoder (BA) [9], learn hash functions using unlabeled data to preserve some metric distance neighbors.

The semi-supervised and supervised hashing methods, such as binary reconstructive embedding (BRE) [39], minimal loss hashing (MLH) [54], semi-supervised hashing (SSH) [67], supervised hashing with kernels (KSH) [51], LDAHash [62], DBC [58], CGHash [44], ITQ with canonical correlation analysis (CCA-ITQ) [25], FastHash [46], supervised discrete hashing (SDH) [60], and kernel-based SDH (KSDH) [61], leverage supervised information in forms of category labels or pairwise similarity to preserve semantic structure in the Hamming space. Despite their promise, these approaches directly learn on the target task with large amounts of training data, and cannot generalize well from few examples or to novel categories. There has been little work on few-shot hash learning as ours, which is crucial for on-the-fly image retrieval in practice.

Instead of using hand-crafted features, most recent work [35, 73, 77, 41, 48, 75, 49, 76, 19] focuses on jointly learning image representations and hash codes with deep CNN models to preserve complex semantic similarity. Again, such supervised hashing with deep models relies on a large corpus of annotated data and cannot address our few-shot learning problem. Our approach benefits from a pre-trained, semantic CNN feature space, but learns a library of universal hash functions in an *unsupervised* manner. Our approach also scales to large code size. In principle, we could generate in parallel as many codes as desired, and then select the most informative ones for a novel category.

Another relevant line of work is learning intermediate representation as the (binarized) outputs of classifiers trained for related tasks. This line of work, including Object Bank [43], Classemes [64], PiCoDes [6], and Meta-Class [4], assumes that pre-trained classifiers on a large number of annotated basis classes would suffice to capture generic categorical properties, and could be re-purposed for novel categories/tasks encountered at test time. On the contrary, we demonstrate here that a universal library of hash codes could be discovered by directly leveraging the regularities of general visual data in an unsupervised manner.

In a broad sense, our few-shot hash learning problem is related to one/few-shot learning[21, 36, 68, 42, 59, 69, 71, 45, 7, 70, 66, 28] and transfer learning [1, 69] for image recognition and classification. Our approach is inspired by those in few-shot learning and multi-task learning but tailored to hash learning, such as generating off-line models that are recommended for object detection [68] or transferred for domain adaptation [1, 69].

3. Unsupervised Generic Hashing

Given a large collection of N *unlabeled* images in certain feature space, denoted as $\mathcal{G} = \{x_1, \ldots, x_N\}$, where $x_i \in \mathbb{R}^d$, we produce a large library of hash functions, each of which maps a d-dimensional input to a binary code, *i.e.*, $h(x) : \mathbb{R}^d \to \{0, 1\}$. Our goal is to generate the hash functions that are semantically generalizable across categories and tasks. That is, similar images should be encoded to similar binary codes in the corresponding Hamming space, and vice versa. Our key insight is that the information implicit in the density structure of a pre-trained, discriminative feature space is informative enough to enable us to generate such codes *without supervision*.

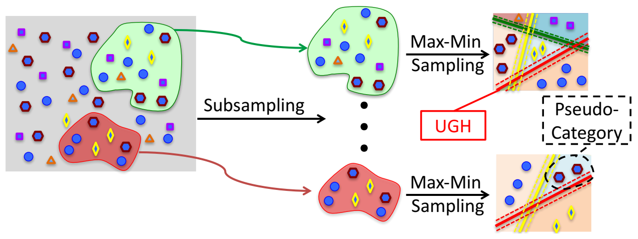


Figure 2: Illustration of generating unsupervised generic hashing (UGH) by obtaining pseudo-categories that are surrogate for plausible, latent categories and learning hash functions that traverse across the low-density regions.

To this end, we use CNN features, which affords us two advantages. First, CNN features have been shown in recent work to produce better results in recognition and classification tasks and so would presumably be more effective than hand-crafted features in our case as well. But, more importantly, because they are themselves learned from visual data, the CNN features carry information about the semantic structure of the feature space so that simple linear hash functions can be used to generate useful codes.

During the hyper-training phase, this large pool of unlabeled images $\mathcal{G}$ serve as a much less biased sampling in the feature space, with high-density regions corresponding to potential latent categories. The scale of the image set matters more than its sources: they could be existing large-scale image databases, *e.g.*, Yahoo/Flickr 100-million dataset [63], or random Internet images. In this section we first describe how to generate a diverse and expressive hash library in this unsupervised scenario by iteratively obtaining pseudo-labeled data and learning unsupervised generic hash (UGH) functions that traverse across the low-density regions, as shown in Fig. 2. We then explain how to select and use the codes for novel tasks with few examples.

3.1. Hash Learning during Unsupervised Hyper-Training

To obtain the hash functions that traverse across the low-density regions, predictable discriminative binary code (DBC) [58] is a good candidate, which seeks hyperplanes that separate categories with large margins. However, DBC relies on label information. Inspired by recent work on ensemble projection [16] and unlabeled data selection by attributes [14], we introduce a series of sampling procedures to obtain pseudo-categories and modify DBC to be estimated in an unsupervised manner, leading to our UGH.

3.1.1 Estimating Diversified Pseudo-Labeled Data

Since we have no supervised annotations, we first need to generate pseudo-labels that are surrogate for plausible, latent categories. More precisely, we want to satisfy both the within-pseudo-category constraint (*i.e.*, samples with the same pseudo-labels should be similar in the feature space) and the between-pseudo-category constraint (*i.e.*, samples with different pseudo-labels should be very dissimilar). To this end, from the image pool $\mathcal{G}$ we first draw an M-subset $\mathcal{U}_{\mathcal{S}}$ by random subsampling. Within $\mathcal{U}_{\mathcal{S}}$, we create *pseudo-labeled* prototype sets $\mathcal{V}_{\mathcal{PL}}$ by a two-step sampling procedure, Max-Min sampling [16], to satisfy the constraints.

At the Max-step, an initial skeleton of the prototype set is created by selecting data samples that are spread out. This is achieved by first randomly sampling m subsets, each of which consists of C random seed points. Each seed point can be viewed as the seed of a pseudo-category. The subset having the largest mutual distance between its member points is then chosen. At the Min-step, each seed point is augmented to a pseudo-category with T pseudo-labeled samples by adding its $T-1$ nearest neighbors. We have then generated $\mathcal{V}_{\mathcal{PL}} = \{(x_i, y_i)\}$, where $i = 1, \ldots, TC$, and $y_i \in \{1, \ldots, C\}$ indicates the pseudo-category labels.

3.1.2 Learning UGH

$\mathcal{V}_{\mathcal{PL}}$ now becomes the pseudo-labeled part for each subset $\mathcal{U}_{\mathcal{S}}$, and the remaining ones are still unlabeled, denoted as $\mathcal{W}_{\mathcal{UL}}$. We use a coarse-to-fine procedure that learns K hash functions of DBC in three steps: initialization, expansion, and calibration, which is inspired by the approach to learning unsupervised sources for transfer learning in [69] but with different expansion and additional calibration steps.

Initialization. In this first step, we generate a set of K initial hash functions, represented by K weight vectors w^k by using the max-margin formulation introduced in DBC [58]. w^k is generated by the label l^k while enforcing the codes similar in-class and dissimilar out-of-class in the pseudo-categories. Importantly, we use our pseudo-labeled samples from our C pseudo-categories instead of the labeled samples normally used in the original supervised DBC:

$$\begin{aligned}
\min_{w,\xi,L,H} \frac{1}{2} \sum_{c=1}^{C} \sum_{m,n\in c} dist\left(H_m, H_n\right) + \gamma \sum_{k=1}^{K} \left\|w^k\right\|^2 \\
+ \lambda_1 \sum_{k=1}^{K} \sum_{i=1}^{TC} \max\left\{1 - l_i^k \left({w^k}^T x_i\right), 0\right\} \\
- \frac{\lambda_2}{2} \sum_{\substack{c'=1 \\ a\in c'}}^{C} \sum_{\substack{c''=1 \\ b\in c'', c'\neq c''}}^{C} dist\left(H_a, H_b\right) \qquad (1)
\end{aligned}$$

where for the kth hash function, w^k is the weight vector, h_i^k and l_i^k are the binary hash code and training label of x_i, respectively. $h_i^k = \left(1 + \mathrm{sign}\left({w^k}^T x_i\right)\right)/2$ and $H_i = [h_i^1, \ldots, h_i^K]$. $dist$ is the Hamming distance.

Expansion. Now that we have an initial estimate of hash codes consistent with the C pseudo-categories, we can (pseudo-)label more samples to get a richer pseudo-labeled set, which we can use for refining the hash functions. This is similar to the semi-supervised learning approach in [14] for expanding the coverage of small labeled training sets by adding unlabeled samples to each category based on their attributes. In our case, we select and add F samples to each pseudo-category from the unlabeled pool $\mathcal{W}_{\mathcal{UL}}$. We select the samples that are similar in both the original feature space and the Hamming space (which is viewed as an attribute space), and the similarity is measured by the responses of max-margin classifiers generated in both spaces. This is different from using the actual attributes learned from auxiliary labeled data as in [14]. We use 90% of the new samples along with the original pseudo-labeled data to form an augmented dataset $\mathcal{V}_{\mathcal{AUG}} = \{(x_i, y_i)\}$, where $i = 1, \ldots, (T + 0.9F)C$, and $y_i \in \{1, \ldots, C\}$. We retrain a new set of K hash functions on $\mathcal{V}_{\mathcal{AUG}}$ by using Eqn. 1.

Calibration. Finally, using the 10% held-out samples as the validation set, we calibrate the hash functions by standard Platt's scaling to fit a sigmoid function with parameters α and β, leading to probabilistic outputs as in [56]

$$f\left(x|w^k, \alpha^k, \beta^k\right) = \frac{1}{1 + e^{-\alpha^k\left({w^k}^T x - \beta^k\right)}}. \qquad (2)$$

As noted in other related work, this sigmoid normalization is crucial when using the outputs of classifiers as descriptors. This calibration step can be interpreted as locally non-linear warping of feature space, with a simple rescaling and shifting of the decision boundary. Each bit is then obtained by thresholding the calibrated outputs at 0.5. To ensure diversity and coverage of UGH across the feature space, we repeat the subsampling procedure in Section 3.1.1 D times, and generate KD hash functions in total.

3.2. Code Selection and Usage for Novel Categories

The unsupervised hyper-training phase provides a large library of hash codes, which can be viewed as an over-complete representation. Given a new target task, *e.g.*, a novel category, at query time with a small set of training images, we could simply use all of these codes as descriptors for image retrieval. Motivated by the power of sparse representations, an alternative is to select the most informative bits so as to infer what is shared with this specific input category. To achieve this, during the training phase as shown in Fig. 1, using all the codes as features and the small training samples from the target task, we first learn an $L1$-regularized model, *e.g.*, $L1$-regularized SVM, and pick the active bits according to the desired code length, which correspond to the weights of larger absolute value [58]. We thus obtain category specific codes as a compact representation. Using these codes, we then perform nearest neighbor search for retrieval purpose and evaluate the performance.

4. Experimental Evaluation

In this section, we present experimental results evaluating our UGH on standard image retrieval benchmarks, comparing with state-of-the-art supervised and unsupervised hashing methods for small-sample learning, and validating across tasks and categories the generality of UGH.

4.1. Implementation Details

We begin by describing the setup used to generate our unsupervised generic hashing (UGH). Our approach is independent of the specific CNN designs. Here we use the Caffe AlexNet CNN pre-trained on ILSVRC 2012 [38, 34]. All the CNN weights are frozen to those learned on ILSVRC without any fine-tuning. For each resized image, we extract features on the standard 10 crops (4 corners plus one center and their flips) and average them as the final feature. It is a $d = 4{,}096$-D feature vector $fc6$ taking from the penultimate hidden layer of the network. All the data samples are normalized to have unit length.

As mentioned before, there is no restriction on the corpus of unlabeled data. Here, we randomly select a 2M subset of the Yahoo/Flickr 100-million Dataset [63], which was collected entirely automatically. Note that the generated UGH is insensitive to the choice of dataset. Experimentally, the results reported below are similar if using other unlabeled large-scale dataset. We then have $N = 2\text{M}$ unlabeled images $\mathcal{G}$. For each iteration, we subsample $M = 20{,}000$ data to form $\mathcal{U}_S$. For the Max-Min sampling procedure, we use the same setup and hyper-parameters reported in [16, 17]: after sampling $m = 50$ subsets, we obtain $\mathcal{V}_{\mathcal{PL}}$ consisting of $C = 30$ pseudo-categories with $T = 6$ samples per pseudo-category. Within $\mathcal{V}_{\mathcal{PL}}$, we generate $K = 10$ prototype hash functions. We use the same setup and default hyper-parameters as in [58] without tuning, where λ_1, γ are set to 1 and λ_2 is set to normalize for the size of pseudo-categories. We then select $F = 50$ samples [14] per pseudo-category as expansion, resulting in an augmented dataset $\mathcal{V}_{\mathcal{AUG}}$ with $(45+6) \times 30$ training samples and 5×30 validation samples. Within $\mathcal{V}_{\mathcal{AUG}}$ we retrain and refine a new set of $K = 10$ hash functions. Repeating $D = 2{,}000$ subsampling, we have hyper-trained 20,000 hash functions in total. Despite its large number, each set of the hash functions can be learned independently, allowing for easy parallelization.

Note that we used the same setup and hyper-parameters reported in [16, 17, 58, 14], and we already significantly outperformed the baselines. By tuning them, we could achieve even better performance. For fair comparisons, our UGH and all the following supervised and *unsupervised* baselines learn hash codes over the pre-trained AlexNet features. For all the baselines, we ran codes provided by the authors and used the suggested or optimized parameters in all experiments.

Here similarity labels are defined by semantic-level labels. Images from the same category are considered semantically similar, and vice verse. Following the standard evaluation protocols [24, 9, 51, 47], each dataset is split into a large retrieval database and a small query set. In the previous work, a large amount of random samples from the retrieval database are used to train the hashing models [51, 46, 60, 77]. Since we are interested in the few-shot learning scenarios, we randomly sample small size training data to select (for our UGH) or learn (for baselines) hash codes. The number of training examples per category varies from 1 to 100 and the length of the hash codes varies from 8 to 32 bits. The retrieval performance on the query set is evaluated using mean average precision (mAP) and precision within Hamming radius 2 (Precision@2). To reduce the influence of random selection, all experiments are repeated ten times and the average mAP and precision are reported.

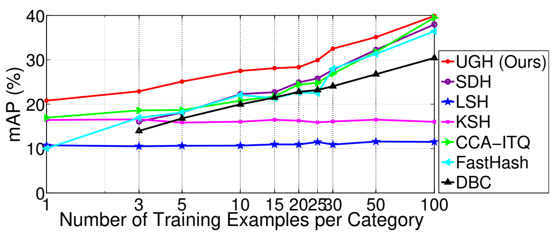
4.2. Comparisons with Supervised Hashing

Naturally, the first and most important question to answer is whether our UGH learned by unsupervised hyper-training indeed facilitates generalization to novel categories with few samples, compared with the state-of-the-art supervised hashing methods. We answer this question on the CIFAR10 benchmark [37]. This dataset consists of 60,000 images from 10 object classes, with 6,000 images per class. Following the standard practice [19], 50,000 images are used as the retrieval database and 10,000 images are used as the query set. This dataset is selected specifically for extensive evaluation and analysis. We include evaluation on more changeling datasets in the later sections.

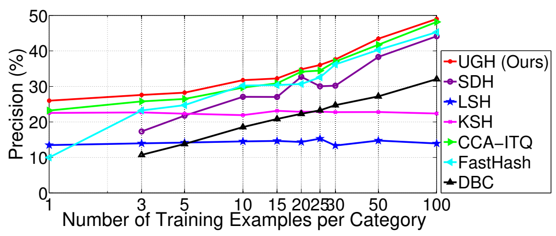
Baselines. We compare against the state-of-the-art supervised hashing approaches, including CCA-ITQ [24], FastHash [46], SDH [60], KSH [51], and DBC [58]. DBC is the original supervised version of our approach. We also include the data-independent LSH [2] as reference. These approaches can be viewed as *online binary codes*, as the hash functions are directly learned from the target dataset. In our preliminary experiments, we also tested the recent supervised deep hashing via training neural networks [77, 76]. These approaches typically require using the entire large-scale retrieval database for hash learning. With limited training data in our case, their performance is significantly inferior to that of other baselines which we reported. We thus did not include their results here.

4.2.1 Influence of Training Set Size

First we evaluate the performance as a function of the number of training examples per category. The code length for

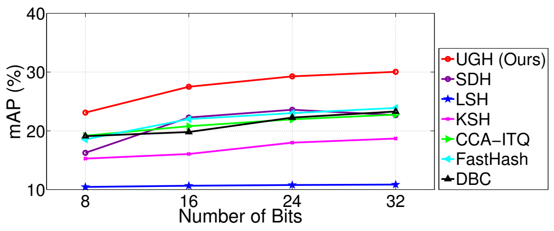


(a)

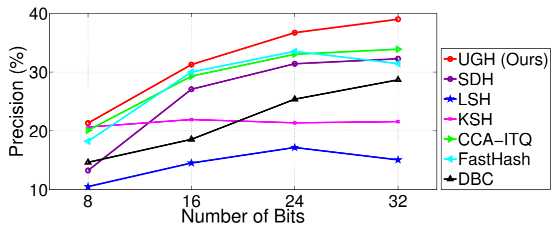


(b)

Figure 3: Performance comparisons between UGH and competing supervised hashing approaches for few-shot hash learning and image retrieval on the CIFAR10 dataset. X-axis: number of training examples per category. Y-axis: mean average precision (mAP) (Fig. 3a) and precision@2 (Fig. 3b). With the same code length 16, our UGH significantly outperforms these baselines for learning with few samples.



(a)



(b)

Figure 4: Performance comparisons between UGH and competing supervised hashing approaches for few-shot hash learning and image retrieval on the CIFAR10 dataset. X-axis: code length. Y-axis: mean average precision (mAP) (Fig. 4a) and precision@2 (Fig. 4b). With the same 10 training examples per category, our UGH consistently outperforms these baselines by large margins across different code lengths.

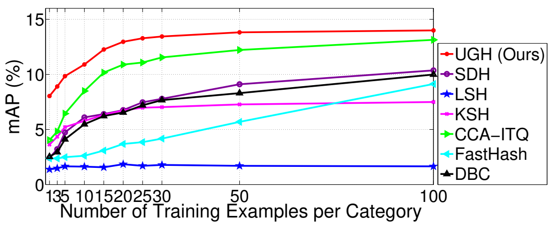
all approaches is fixed as 16: our UGH selects 16 category-specific codes from the 20,000 hash library and the baselines directly learn the codes at length 16. Due to lack of public protocols for few-shot learning, we randomly sample $1, 3, 5, 10, 15, 20, 25, 30, 50$ and 100 images per category from the retrieval database as the training set. Fig. 3 summarizes the average mAP and precision@2.

As shown in Fig. 3, our UGH consistently outperforms all the other supervised hashing for small-sample learning. While the vanilla hashing approaches are over-fitting in this scenario, our universal binary representation, by leveraging large-scale unlabeled data, is effectively learned and transferable to novel categories. In addition to the unsupervised aspect, our code selection phase leads to both compact and discriminative codes for the target task, making it significantly different from LSH which typically requires long binary codes. To verify this, we tested random selection of the codes. While it is still better than the baselines, the performance drops. *e.g.*, in the one-shot case, the random selection achieved 18.42% mAP, which is better than 16.96% of CCA-ITQ (the best performing baseline) and is worse than 20.80% of our UGH with discriminative code selection.

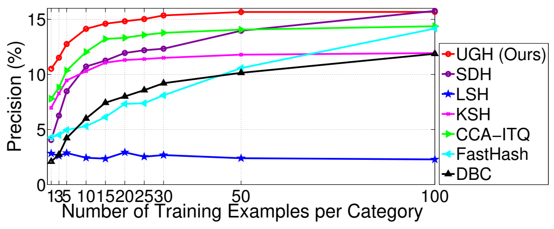
An obvious advantage of UGH over its supervised counterpart DBC is that UGH makes it feasible to generate a large collection of hash codes based on unlabeled data. Another promising finding, based on Fig. 3, is that UGH demonstrates more *expressive* and *universal* capability for novel categories with few samples compared to DBC. This verifies our assumption that information across categories is actually intrinsic in the data even without any supervision. One explanation is that by using another large-scale dataset (*e.g.*, Flicker-2M) apart from where the CNN feature is learned (ILSVRC), UGH would potentially prevent over-fitting and provide more generalization ability. This is similar to the case in which models are trained on the training dataset and their parameters are tuned on another validation dataset. More importantly, we introduce a series of sampling procedures in producing UGH to generate a diverse partition of the feature space in contrast to DBC (and other supervised hashing baselines). This leads to distributed representations, which is a crucial ingredient for generalization to new cases [3].

4.2.2 Influence of Code Length

We also investigate the influence of the code length (the number of selected codes as the final descriptor) on the CIFAR10 dataset. 10 images per category are randomly se-



(a)



(b)

Figure 5: Performance comparisons between UGH and competing supervised hashing approaches for few-shot hash learning and image retrieval on the CIFAR100 dataset. X-axis: number of training examples per category. Y-axis: mean average precision (mAP) (Fig. 5a) and precision@2 (Fig. 5b). With the same code length 16, our UGH significantly outperforms these baselines for large-scale image retrieval tasks with few training samples.

lected as the training set; the retrieval database and query set remain as before. Fig. 4 shows the mAP and precisioin@2 achieved by UGH and the supervised hashing baselines. These baselines directly learn the code at the desired length.

Fig. 4 shows that UGH is more robust than other hashing competitors by maintaining very stable performance across increasing code lengths. This indicates the effectiveness of selecting category specific codes. FastHash tends to have good mAP performance; its precision@2, however, drops with longer hash codes 32, which shows its inability to form compact clusters in the hash code space. Moreover, these conventional hashing approaches are restrictive in the sense that, for different code lengths, they need to re-learn the entire hash codes. On the contrary, the unsupervised, off-line and parallel aspects of our code generation mechanism makes it orders of magnitude faster and could be re-purposed for tasks with different desired code length. Even with additional code selection stage, training is efficient since the implementation of SVMs with binary codes could be greatly simplified and sped up by using a logical AND and a sparse summation for dot-products instead of floating-point calculations [4]. This favors such an approach to be used in ultra-large-scale scenarios.

4.3. Large-Scale Comparisons

We now move on to evaluate our UGH on the large-scale CIFAR100 dataset [37], which contains 100 categories with 600 images per category. Following the standard practice [53], we randomly select 100 images per category as the query set and use the remaining images as the retrieval database. Similar to the experimental setup in Section 4.2.1, we focus on the influence of the training set size. Fig. 5 summarizes the comparisons with the supervised baselines.

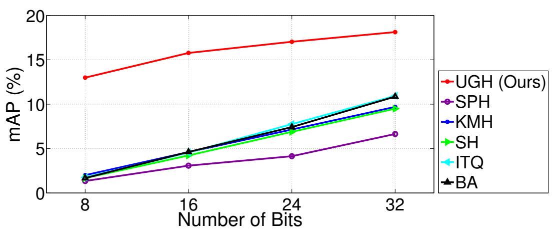
As shown in Fig. 5, our UGH outperforms all the baselines by large margins in this large-scale scenario. The low performance of LSH suggests that CIFAR100 is more challenging than CIFAR10. In particular, Fig. 5a shows that there is nearly 50% relative mAP performance boost in the one-shot learning case. Although SDH achieves comparable precision as our UGH when the number of samples is 100, it has a much lower mAP (3.63% smaller) than ours. The improvements of UGH are more significant in the small sample size regime (*e.g.*, 1, 3 and 5), which is consistent with the observation on CIFAR10. While the state-of-the-art hash codes are learned separately for different target tasks, our UGH is inferred *once off-line* without knowing any target dataset, and generalizes well to the novel task without requiring additional, extensive hash training.

4.4. Comparisons with Unsupervised Hashing

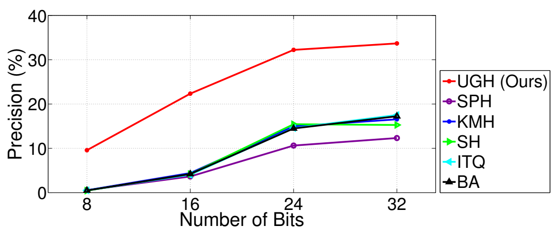
Our approach estimates diverse pseudo-categories and learns hash functions that traverse across the low-density regions. Rather than simply due to additional unsupervised data, this hash learning scheme is crucial for its generalization across categories. To show this point, we further evaluate our UGH and unsupervised hashing on the SUN397 dataset [74]. As a challenging benchmark, SUN397 contains 108,754 images of 397 scene categories. Following the standard practice [20], we use a subset which includes 42 categories with more than 500 images per category, leading to 35K images in total. The query set contains 4,200 images with 100 images per category randomly sampled from the dataset. The remaining images are used as the retrieval database. We focus on the influence of the code length.

Baselines. We compare against several state-of-the-art unsupervised hashing approaches, including PCA-ITQ [24], binary autoencoder (BA) [9], spectral hashing (SH) [72], spherical hashing (SPH) [30], and K-means hashing (KMH) [29], which are learned over the AlexNet features. Similar to our use case, they are now used in an *offline* manner, in which the codes are learned on Flickr-2M and then tested on the target SUN397.

Fig. 6 shows that our UGH consistently achieves the best performance across different code lengths. This verifies that our generalization ability to novel tasks and categories comes not only from the generic CNN features, but also from the code generation mechanism. In addition to the initial Max-Min sampling that enforces diversity, our UGH



(a)



(b)

Figure 6: Performance comparisons between UGH and competing unsupervised hashing approaches on the SUN397 dataset. X-axis: code length. Y-axis: mean average precision (mAP) (Fig. 6a) and precision@2 (Fig. 6b). Both UGH and the baselines learn unsupervised hash codes over the pre-trained AlexNet features. The codes are learned on Flickr-2M and then tested on SUN397. Our UGH consistently generalizes better than these baselines by large margins across different code lengths.

introduces an additional expansion step to augment pseudo-categories with more data in a bootstrap manner, yielding more accurate sampling of the feature space structure. We further group the pseudo-categories into a set of abstract classes, leading to more generic hash codes. On the contrary, the existing unsupervised hashing approaches are proposed mainly for compression; with semantic information only coming from the input CNN features, their generalization ability is significantly limited.

4.5. Qualitative Visualization

To understand our hash codes, we show qualitative visualization of representative codes in Fig. 7. These codes are randomly selected from those that could be visually interpretable for better analysis. The visualization shows that in the unsupervised scenario our UGH learns semantics that are informative across categories, which thus explains its generalization ability to novel categories and tasks.

5. Conclusions

We proposed an approach to few-shot hash learning in which hash functions learned from unlabeled data are used for generating binary codes of a new object category from a few samples. The hash functions are learned in a feature space constructed from high-dimensional expressive features so that computationally efficient linear classifiers can be used instead of the more expensive kernel-based classifiers. Although generated without supervision, these codes carry significant information about the structure of the visual space so that, given a novel category, a subset of the codes can be selected to form a good representation, even with very few samples. The crucial observation is that the codes are not tied to any specific category and therefore encode the visual space in an unbiased manner. The resulting hash codes are accurate in image retrieval performance and efficient in terms of computation, storage, and size of training data. By analogy with CNN terminology, our approach uses not only pre-trained features (the CNN

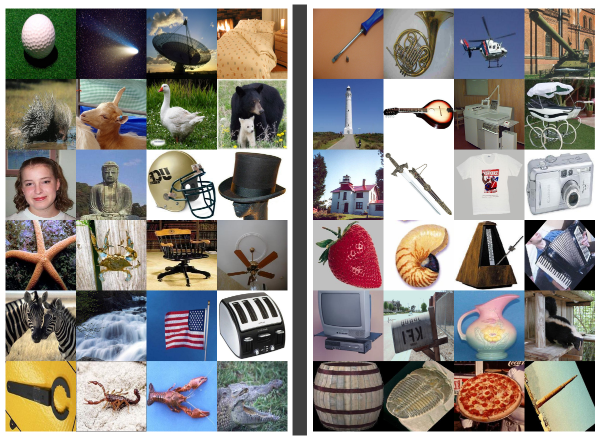


Figure 7: Hash code visualization. Each row of images illustrates a particular bit in our UGH, which groups the data based on unknown notions of similarity. For six representative bits (top to bottom), we show on two sides of the black bar 4 positive and negative images, respectively (left to right). One can find that the corresponding hash functions learn semantics that are informative across categories, *i.e.*, repeated pattern, fur/feather, head, star-shaped, stripes, pincer-shape (top to bottom). This behavior shows the generality of our codes for novel categories and tasks.

features trained on ILSVRC) but also pre-trained classifiers (on unlabeled data). Future work involves exercising this approach on other tasks, such as detection, expanding it to larger scale problems, and more sophisticated code selection mechanisms.

Acknowledgments: This work was supported in part by ONR MURI N000141612007 and U.S. Army Research Laboratory (ARL) under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016. We also thank NVIDIA for donating GPUs and AWS Cloud Credits for Research program.

References

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, 2005.

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

[3] Y. Bengio. Deep learning: Progress in theory and attention mechanisms. In *Deep Vision CVPR Workshop*, 2015.

[4] A. Bergamo and L. Torresani. Meta-class features for large-scale object categorization on a budget. In *CVPR*, 2012.

[5] A. Bergamo and L. Torresani. Classemes and other classifier-based features for efficient object categorization. *TPAMI*, 36(10):1988–2001, 2014.

[6] A. Bergamo, L. Torresani, and A. W. Fitzgibbon. PiCoDes: Learning a compact code for novel-category recognition. In *NIPS*, 2011.

[7] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *NIPS*, 2016.

[8] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences*, 1997.

[9] M. A. Carreira-Perpinán and R. Raziperchikolaei. Hashing with binary autoencoders. In *CVPR*, 2015.

[10] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002.

[11] K. Chatfield, R. Arandjelović, O. Parkhi, and A. Zisserman. On-the-fly learning for visual search of large-scale image and video datasets. *International journal of multimedia information retrieval*, 4(2):75–93, 2015.

[12] K. Chatfield, K. Simonyan, and A. Zisserman. Efficient on-the-fly category retrieval using convnets and GPUs. In *ACCV*, 2014.

[13] K. Chatfield and A. Zisserman. VISOR: Towards on-the-fly large-scale object category retrieval. In *ACCV*, 2012.

[14] J. Choi, M. Rastegari, A. Farhadi, and L. S. Davis. Adding unlabeled samples to categories by learned attributes. In *CVPR*, 2013.

[15] O. Chum and J. Matas. Large-scale discovery of spatially related images. *TPAMI*, 32(2):371–377, 2010.

[16] D. Dai and L. V. Gool. Ensemble projection for semi-supervised image classification. In *ICCV*, 2013.

[17] D. Dai and L. Van Gool. Ensemble projection with CNN features for semi-supervised image classification and image clustering. Technical report, ETH Zurich, May 2015.

[18] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, 2004.

[19] T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to hash with binary deep neural network. In *ECCV*, 2016.

[20] T.-T. Do, A.-D. Doan, D.-T. Nguyen, and N.-M. Cheung. Binary hashing with semidefinite relaxation and augmented Lagrangian. In *ECCV*, 2016.

[21] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *TPAMI*, 28(4):594–611, 2006.

[22] B. Fernando and T. Tuytelaars. Mining multiple queries for image retrieval: On-the-fly learning of an object-specific mid-level representation. In *ICCV*, 2013.

[23] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[24] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[25] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI*, 35(12):2916–2929, 2013.

[26] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. In *ICLR*, 2017.

[27] X. Han, B. Singh, V. I. Morariu, and L. S. Davis. VRFP: On-the-fly video retrieval using web images and fast fisher vector products. *TMM*, 2017.

[28] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, 2017.

[29] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, 2013.

[30] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *CVPR*, 2012.

[31] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998.

[32] G. Irie, Z. Li, X.-M. Wu, and S.-F. Chang. Locally linear hashing for extracting non-linear manifolds. In *CVPR*, 2014.

[33] H. Jégou, L. Amsaleg, C. Schmid, and P. Gros. Query adaptative locality sensitive hashing. In *ICASSP*, 2008.

[34] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014.

[35] Y. Kang, S. Kim, and S. Choi. Deep learning to hash with multiple representations. In *ICDM*, 2012.

[36] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Workshops*, 2015.

[37] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[39] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009.

[40] P. Kulkarni, G. Sharma, J. Zepeda, and L. Chevallier. Transfer learning via attributes for improved on-the-fly classification. In *WACV*, 2014.

[41] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, 2015.

[42] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[43] L.-J. Li, H. Su, E. P. Xing, and F.-F. Li. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010.

[44] X. Li, G. Lin, C. Shen, A. Van Den Hengel, and A. R. Dick. Learning hash functions using column generation. In *ICML*, 2013.

[45] Z. Li and D. Hoiem. Learning without forgetting. In *ECCV*, 2016.

[46] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, 2014.

[47] G. Lin, C. Shen, D. Suter, and A. van den Hengel. A general two-step approach to learning-based hashing. In *ICCV*, 2013.

[48] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *CVPRW*.

[49] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval.

[50] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *NIPS*, 2014.

[51] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernel. In *CVPR*, 2012.

[52] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, 2011.

[53] L. Mukherjee, J. Peng, T. Sigmund, and V. Singh. Network flow formulations for learning binary hashing. In *ECCV*, 2016.

[54] M. Norouzi and D. M. Blei. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[55] L. Paulevé, H. Jégou, and L. Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010.

[56] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, 1999.

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

[58] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV*, 2012.

[59] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. In *ICML*, 2016.

[60] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *CVPR*, 2015.

[61] X. Shi, F. Xing, J. Cai, Z. Zhang, Y. Xie, and L. Yang. Kernel-based supervised discrete hashing for image retrieval. In *ECCV*, 2016.

[62] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua. LDAHash: Improved matching with smaller descriptors. *TPAMI*, 34(1):66–78, 2012.

[63] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[64] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010.

[65] L. Torresani, M. Szummer, and A. Fitzgibbon. Classemes: A compact image descriptor for efficient novel-class recognition and search. In *Registration and Recognition in Images and Videos*, pages 95–111. Springer, 2014.

[66] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.

[67] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *TPAMI*, 34(12):2393–2406, 2012.

[68] Y.-X. Wang and M. Hebert. Model recommendation: Generating object detectors from few samples. In *CVPR*, 2015.

[69] Y.-X. Wang and M. Hebert. Learning by transferring from unsupervised universal sources. In *AAAI*, 2016.

[70] Y.-X. Wang and M. Hebert. Learning from small sample sets by combining unsupervised meta-training with CNNs. In *NIPS*, 2016.

[71] Y.-X. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016.

[72] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2009.

[73] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, 2014.

[74] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[75] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *TIP*, 24(12):4766–4779, 2015.

[76] Z. Zhang, Y. Chen, and V. Saligrama. Efficient training of very deep neural networks for supervised hashing. In *CVPR*, 2016.

[77] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, 2015.