Hashing with Mutual Information

Fatih Cakir*, *Member, IEEE,* Kun He*, *Student Member, IEEE,* Sarah Adel Bargal, *Student Member, IEEE* and Stan Sclaroff, *Fellow, IEEE*

Abstract—Binary vector embeddings enable fast nearest neighbor retrieval in large databases of high-dimensional objects, and play an important role in many practical applications, such as image and video retrieval. We study the problem of learning binary vector embeddings under a supervised setting, also known as hashing. We propose a novel supervised hashing method based on optimizing an information-theoretic quantity, *mutual information*. We show that optimizing mutual information can reduce ambiguity in the induced neighborhood structure in the learned Hamming space, which is essential in obtaining high retrieval performance. To this end, we optimize mutual information in deep neural networks with minibatch stochastic gradient descent, with a formulation that maximally and efficiently utilizes available supervision. Experiments on four image retrieval benchmarks, including ImageNet, confirm the effectiveness of our method in learning high-quality binary embeddings for nearest neighbor retrieval.

Index Terms—Hashing, Deep learning, Nearest neighbor retrieval, Mutual information.

1 Introduction

In computer vision and many other application areas, there is typically an abundance of data with high-dimensional raw representations, such as mega-pixel images and high-definition videos. Besides obvious storage challenges, high-dimensional data pose additional challenges for semantic-level processing and understanding. One prominent example that we focus on in this paper is nearest neighbor retrieval. In applications such as image and video search, person and object recognition in photo collections, and action detection and classification in surveillance video, it is often necessary to map high-dimensional data objects to low-dimensional vector representations to allow for efficient retrieval of similar instances in large databases. In addition, the desired semantic similarity can vary from task to task, often prescribed by available supervision, *e.g.* class labels. Therefore, the mapping process is also responsible for leveraging supervised learning to encode task-specific similarities, such that objects that are semantically similar are mapped to close neighbors in the resulting vector space.

In this paper, we consider the problem of learning low-dimensional *binary* vector embeddings of high-dimensional data, also known as hashing. Binary embeddings enjoy a small memory footprint and permit fast search mechanisms, as Hamming distance computation between binary vectors can be implemented very efficiently in modern CPUs. As a result, across a variety of domains, hashing approaches have been widely utilized in applications requiring fast (approximate) nearest neighbor retrieval. Examples include: image annotation [58], visual tracking [31], 3D reconstruction [7], video segmentation [40], object detection [11], audio search [54], multimedia retrieval [16], [46], [47], and large-scale clustering [20], [21], [22]. Our goal is to learn hashing functions that can result in optimal nearest neighbor retrieval performance. In particular, as motivated above, we approach hashing as a supervised learning problem, such that the learned binary embeddings encode task-specific semantic similarity.

Supervised hashing is a well-studied problem. Although many different formulations exist, all supervised hashing formulations essentially constrain the learned Hamming distance to agree with the given supervision. Such supervision can be specified as pairwise affinity labels: pairs of objects are annotated with binary labels indicating their pairwise relationships as either "similar" or "dissimilar." In this case, a common learning strategy is *affinity matching*: the learned binary embedding should evaluate to low Hamming distances between similar pairs, and high Hamming distances between dissimilar pairs. Alternatively, supervision can also be given in terms of local relative distance comparisons, most notably three-tuples of examples, or "triplets", where one example is constrained to have a smaller distance to the second example than the third. This can be termed *local ranking*. Typically, for ease of optimization, these formulations define and optimize loss functions that match the form of supervision, *i.e.*, defined on training pairs or triplets. However, loss functions used in affinity matching and local ranking methods are usually only indirectly related to retrieval performance, and in order to optimize overall retrieval performance, it is often necessary to introduce additional regularization terms, or parameters such as margins, thresholds, and scaling factors.

We argue that approaches such as affinity matching and local ranking are insufficient to achieve optimal nearest neighbor retrieval performance. Instead, we view supervised hashing through an information-theoretic lens, and propose a novel solution tailored for the task of nearest neighbor retrieval. Our key observation is that a good binary embedding should well separate neighbors and non-neighbors in the Hamming space, or, achieve low *neighborhood ambiguity*. An alternative viewpoint is that the learned Hamming embedding should carry a high amount of information regarding the desired neighborhood structure. To quantify neighborhood ambiguity, we use a well-known quantity from information theory, *mutual information*, and show that it has direct and strong correlations with standard ranking-based retrieval performance metrics. An appealing property of the mutual information objective is that it is free of tuning parameters,

- *F. Cakir was with the Department of Computer Science, Boston University, Boston, MA, 02215.*
  E-mail: fcakir@bu.edu
- *K. He, S. A. Bargal, and S. Sclaroff are with the Department of Computer Science, Boston University, Boston, MA, 02215.*
  E-mails: hekun, sbargal, sclaroff@bu.edu

* *The first two authors contributed equally.*

unlike others that may require thresholds, margins, and so on. Finally, to optimize mutual information, we relax the original NP-hard discrete optimization problem, and develop a gradient-based optimization framework that can be efficiently applied with minibatch stochastic gradient descent in deep neural networks.

To briefly summarize our contributions, we propose a novel supervised hashing method that is based on quantifying and minimizing neighborhood ambiguity in the learned Hamming space, using mutual information as the learning objective. An end-to-end gradient-based optimization framework is developed, with an efficient minibatch-based implementation. Our proposed hashing method is named MIHash.[1] We conduct image retrieval experiments on four standard benchmarks: CIFAR-10, NUSWIDE, LabelMe, and ImageNet. MIHash achieves state-of-the-art retrieval performance across all datasets.

This paper builds upon the formulations introduced in [2] which primarily addressed *online hashing*, a separate and distinct task within the hashing for approximate nearest neighbor retrieval problem domain. In this paper, we expand and improve the formulation introduced in [2], and propose a hashing method for the *batch learning* setting. Notably, our proposed formulation is amenable for deep learning, and we propose an efficient formulation for utilizing supervision in minibatches during stochastic optimization. We further conduct extensive experiments in the batch learning setup, and provide detailed empirical analysis for our proposed approach.

The rest of this paper is organized as follows. First, the relevant literature is reviewed in Section 2. We propose and analyze mutual information as a learning objective for hashing in Section 3, and then discuss its optimization using stochastic gradient descent and deep neural networks in Section 4. Section 5 presents experimental results and empirical analysis of the proposed algorithm's behavior. Finally, Section 6 presents the conclusions.

2 Related Work

Many hashing methods have been introduced over the years. While providing a precise taxonomy of the literature is difficult, a rough grouping can be made as data-independent and dependent techniques. Data-independent techniques do not exploit the data distribution during hashing. Instead, similarity, as induced by a particular metric, is often preserved. This is achieved by maximizing the probability of "collision" when hashing similar items. Notable earlier examples include Locality Sensitive Hashing methods [10], [18], [28] where distance functions such as the Euclidean, Jaccard, and Cosine distances are approximated. These methods usually have theoretical guarantees on the approximation quality and conform with sub-linear retrieval mechanisms. However, they are confined to certain metrics, and they ignore the data distribution and accompanying supervision.

In contrast to data-independent techniques, recent approaches are data-dependent such that hash mappings are learned from the training set. While empirical evidence for the superiority of these methods over their data-independent counterparts is plentiful in the literature, a recent study has also theoretically validated their performance advantage [1]. These methods can be considered as binary embeddings that map the data into Hamming space while preserving a specific neighborhood structure. Such a neighborhood structure can be derived from meta-data (*e.g.*, labels), or can be completely determined by the user (*e.g.*, via similarity-dissimilarity indicators of data pairs). With the binary embeddings, distances can be very efficiently computed, thereby allowing even a linear search to be done efficiently for large-scale corpora. These data-dependent methods can be grouped as follows: similarity preserving techniques [17], [27], [33], [39], [45], [48], [56], [61], [64], quantization methods [19], [23], [24] and recently, deep learning based methods [5], [15], [29], [35], [38], [62], [66], [67], [69]. We now review a few of the prominent techniques in each category. For a more comprehensive survey of the hashing literature, we refer readers to [57].

Quantization methods are the first group of data-dependent hashing methods. Such techniques do not assume the availability of supervision, and generally optimize objectives involving a reconstruction error. Among these, Semi-Supervised Hashing [56] learns the hash mapping by maximizing the empirical accuracy on labeled data and also the entropy of the generated hash functions on unlabeled data. This is shown to be very similar to doing a PCA analysis where the hash functions are the eigenvectors of a covariance matrix. Other noteworthy work includes PCA-inspired methods where the principal components are taken as the hash functions. If "groups" that are suitable for clustering exist within the data, then further refining the principal components for better binarization has shown to be beneficial, as in Iterative Quantization [19] and K-means Hashing [23].

Unsupervised quantization can also be approached as a special case of generative modeling. Semantic Hashing [43] is one early example that is based on the autoencoding principle. It learns a generative model to encode data, in the form of stacked Restricted Boltzmann Machines (RBMs). Carreira-Perpinan and Raziperchikolaei [5] propose Binary Autoencoders, and construct autoencoders with a binary latent layer. They argue that finding the hash mapping without relaxing the binary constraints will yield better solutions, while in relaxation approaches that are more common in the literature, quantization errors can degrade the quality of learned hash functions. More recently, Stochastic Generative Hashing [9] learns a generative hashing model based on the minimum description length principle, and uses stochastic distributional gradient descent to optimize the associated variational objective and handle the difficulty in having binary stochastic neurons.

Similarity preserving methods, on the other hand, aim to construct binary embeddings that optimize loss functions induced from the supervision provided. Both the affinity matching and local ranking methods mentioned in Section 1 belong to this group. Among such techniques, Minimal Loss Hashing [41] considers minimizing a hinge-like loss function motivated by Structural SVMs [50]. In Binary Reconstructive Embeddings [27], a kernel-based solution is proposed where the goal is to construct hash functions by minimizing an empirical loss between the input and Hamming space distances via a coordinate descent algorithm. Supervised Hashing with Kernels [39] also proposes a kernel-based solution; but, instead of preserving the equivalence of the input and Hamming space distances, the kernel function weights are learned by minimizing an objective function based on binary code inner products. Spectral Hashing [61] and Self-Taught Hashing [64] are other notable lines of work where the similarity of the instances is preserved by solving a graph Laplacian problem. Rank alignment methods [13], [44] that learn a hash mapping to preserve rankings in the data can also be considered in this group.

Lately, several "two-stage" similarity preserving techniques

1. Our MATLAB implementation of MIHash is publicly available at `https://github.com/fcakir/mihash`

have also been proposed, where the learning is decomposed into two steps: binary inference and hash function learning. The binary inference step yields hash codes that best preserve the similarity. These hash codes are used as target vectors in the subsequent hash function learning step, for example, by learning binary classifiers to produce each bit. Notable two-stage methods include Fast Hashing with Decision Trees [33], [34], Structured Learning of Binary Codes [32] and Supervised Discrete Hashing [45]. All of these similarity preserving methods assume some type of supervision, such as labels or similarity indicators. Thus, in the literature, such techniques are also regarded as supervised hashing solutions.

Deep hashing methods have recently gained significant prominence following the success of deep neural networks in related tasks such as image classification. Although hashing methods that employ deep learning can be based on either unsupervised quantization or supervised learning, most existing deep hashing methods are supervised. A deep hashing study typically involves a novel architecture, a loss function or a binary inference formulation. Among such methods, Lai *et al.* [29] jointly learn the hash mapping and image features with a triplet loss formulation. This triplet loss ensures that an image is more similar to the second image than to a third one with respect to their binary codes. In [36], a network-in-network (NIN) deep net architecture is used, with a divide-conquer module that is shown to reduce redundancy in the hash bits. In [66], the authors follow the work of [45] and [5]. Similar to [45] they propose learning the hash mapping by optimizing a classification objective. Differently, they consider using a deep net consisting only of fully-connected layers, and use auxiliary variables, as in [5], to circumvent the vanishing gradient problem. Deep learning based hashing studies have also proposed sampling pairs or triplets of data instances to learn the hash mapping. Notable examples include [30] and [60], which optimize a likelihood function which ensures that similar (non-similar) pairs or triplets are mapped to nearby (distant) binary embeddings.

As the ultimate goal of hashing is to preserve a neighborhood structure in the Hamming space, we propose an information-theoretic solution and directly quantify the neighborhood ambiguity of the generated binary embeddings using a mutual information based criterion. Information-theoretic measures have also been considered in past hashing studies. Notably, in [37] an affinity matching formulation is proposed with a pairwise cross-entropy loss to penalize the discrepancy between pairwise Hamming similarities and the ground truth affinities. A similar cross-entropy loss is adopted by Zhu *et al.* [68] in convolutional neural networks. To permit gradient based optimization, the binary embeddings are relaxed to continuous values and a quantization loss is added. Venkateswara *et al.* [53] also consider the cross-entropy loss in an unsupervised domain adaptation setting. These simple affinity matching methods are different from our proposed solution, where we employ an information-theoretic measure to directly minimize neighborhood ambiguity: separating distance distributions between queries and their neighbor and non-neighbor sets. Our proposed mutual information objective is efficient to compute, amenable to batch learning, and leads to state-of-the-art results in standard retrieval benchmarks.

We utilize a recent study, [51], when optimizing our mutual information based objective, and use their differentiable histogram binning technique as a foundation in deriving gradient-based optimization rules. Note that both our problem setup and objective function are quite different from [51].

3 Hashing with Mutual Information

3.1 Preliminaries

Let $\mathcal{X} \subset \mathbb{R}^N$ be the feature space and $\mathcal{H}^b$ be the b-dimensional Hamming space, *i.e.* , $\mathcal{H}^b = \{-1, 1\}^b$. The goal of hashing is to learn an embedding function $\Phi : \mathcal{X} \to \mathcal{H}^b$, which induces a Hamming distance $d_\Phi : \mathcal{X} \times \mathcal{X} \to \{0, 1, \ldots, b\}$ that is equal to the number of bit differences between embedded vectors.

We consider a supervised learning setup. For some example $\hat{x} \in \mathcal{X}$, we assume that we have access to a set $\oplus_{\hat{x}} \subset \mathcal{X}$ containing examples that are labeled as similar to $\hat{x}$ (neighbors), and a set $\ominus_{\hat{x}}$ of dissimilar examples (non-neighbors). We also assume that this similar/dissimilar relationship is symmetric: $x \in \oplus_{\hat{x}} \Leftrightarrow \hat{x} \in \oplus_x$, and $x \in \ominus_{\hat{x}} \Leftrightarrow \hat{x} \in \ominus_x$. We call $\hat{x}$ an anchor example, and refer to $(\oplus_{\hat{x}}, \ominus_{\hat{x}})$ as its *neighborhood structure*. Then, we can cast the problem of learning the Hamming embedding Φ as one of preserving the neighborhood structure: the neighbors of $\hat{x}$ should be mapped to the close vicinity of $\hat{x}$ in the Hamming space, while the non-neighbors should be mapped farther away. Ideally, we would like to satisfy the following constraint:

$$d_\Phi(\hat{x}, x_p) < d_\Phi(\hat{x}, x_n), \;\; \forall x_p \in \oplus_{\hat{x}}, \forall x_n \in \ominus_{\hat{x}}. \qquad (1)$$

If the learned Φ successfully satisfies this constraint, then the neighborhood structure of $\hat{x}$ can be exactly recovered by thresholding the Hamming distance $d_\Phi(\hat{x}, \cdot)$. Generally, the neighborhood structure can be constructed by repeatedly querying a pairwise similarity oracle S, *e.g.* $S(x, \hat{x}) > 0$ iff $x \in \oplus_{\hat{x}}$, and $S(x, \hat{x}) < 0$ iff $x \in \ominus_{\hat{x}}$. In practice, such an oracle can be derived from agreement/disagreement of class labels, or from thresholding a distance metric (*e.g.* Euclidean distance) in the original feature space $\mathcal{X}$. We give concrete examples in Section 5.

In this work, we parameterize the functional embeddings using deep neural networks (DNNs), as DNNs have recently shown to have superior learning capabilities when coupled with appropriate hardware acceleration. Also, in order to take advantage of end-to-end training by backpropagation, we use gradient-based optimization, and adopt an equivalent formulation of the Hamming distance that is amenable to differentiation:

$$d_\Phi(x, x') = \frac{1}{2}\left(b - \Phi(x)^\top \Phi(x')\right), \qquad (2)$$

$$\Phi(x) = (\phi_1(x), \ldots, \phi_b(x)), \qquad (3)$$

$$\phi_i(x) = \text{sgn}(f_i(x; w)) \in \{-1, +1\}, \forall i, \qquad (4)$$

where $f_i, \forall i$ are the activations produced by a feed-forward neural network, with learnable parameters w.

3.2 Minimizing Neighborhood Ambiguity

We now discuss a formulation for learning the Hamming embedding Φ. As mentioned above, given $\hat{x}$ and its neighborhood structure, we would like to satisfy Equation 1 as much as possible, or, minimize the amount of violation. Indeed, many existing supervised hashing formulations are based on the idea of minimizing violations. For instance, affinity matching methods, mentioned in Section 1, typically enforce the following constraints through their loss functions:

$$d_\Phi(\hat{x}, x_p) < t_1, d_\Phi(\hat{x}, x_n) > t_2, \;\forall x_p \in \oplus_{\hat{x}}, \forall x_n \in \ominus_{\hat{x}}, \quad (5)$$

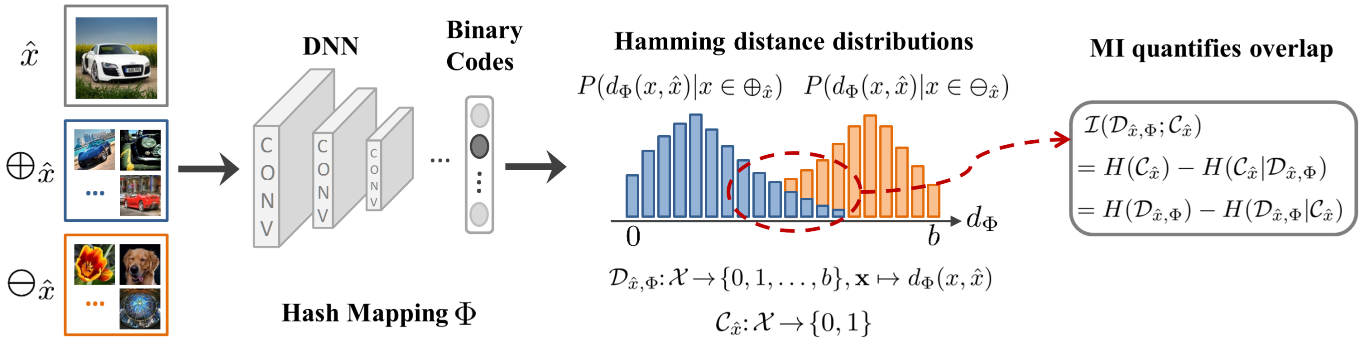

$\hat{x}$
$\oplus_{\hat{x}}$
$\ominus_{\hat{x}}$
DNN
CONV
Binary Codes
Hash Mapping Φ
Hamming distance distributions
$P(d_\Phi(x,\hat{x})|x \in \oplus_{\hat{x}})$
$P(d_\Phi(x,\hat{x})|x \in \ominus_{\hat{x}})$
0
b
d_Φ
$\mathcal{D}_{\hat{x},\Phi}\colon \mathcal{X} \to \{0,1,\dots,b\}, \mathbf{x} \mapsto d_\Phi(x,\hat{x})$
$\mathcal{C}_{\hat{x}}\colon \mathcal{X} \to \{0,1\}$
MI quantifies overlap
$\mathcal{I}(\mathcal{D}_{\hat{x},\Phi};\mathcal{C}_{\hat{x}})$
$= H(\mathcal{C}_{\hat{x}}) - H(\mathcal{C}_{\hat{x}}|\mathcal{D}_{\hat{x},\Phi})$
$= H(\mathcal{D}_{\hat{x},\Phi}) - H(\mathcal{D}_{\hat{x},\Phi}|\mathcal{C}_{\hat{x}})$

Fig. 1: Overview of the proposed hashing method. We use a deep neural network to compute b-bit binary codes for: a (1) query image $\hat{x}$, (2) its neighbors in $\oplus_{\hat{x}}$, and (3) its non-neighbors in $\ominus_{\hat{x}}$. The binary codes are obtained via thresholding the activations in the last layer of the neural network. Computing hamming distances between the binary code of the query and the binary codes of neighbors and non-neighbors yields two distributions of Hamming distances. The information-theoretic quantity, Mutual Information, can be used to capture the separability between these two distributions, which gives a good quality indicator and learning objective.

where $t_1 \le t_2$ are threshold parameters. This indirectly enforces Equation 1 by constraining the absolute values of individual Hamming distances. Alternatively, local ranking methods based on triplet supervision encourage the following:

$$d_\Phi(\hat{x},x_p) + \eta \le d_\Phi(\hat{x},x_n),\ \forall x_p \in \oplus_{\hat{x}}, \forall x_n \in \ominus_{\hat{x}}, \quad (6)$$

where η is a margin parameter. We note that both formulations are inflexible, since the same threshold or margin parameters are applied for all anchors $\hat{x}$. Also, it is often observed in practice that these parameters are nontrivial to tune.

Instead, we propose a novel formulation based on the idea of minimizing *neighborhood ambiguity*, which is more directly related to the quality of nearest neighbor retrieval. We say that Φ induces neighborhood ambiguity if the mapped image of some $x_n \in \ominus_{\hat{x}}$ is closer to that of $\hat{x}$ than some $x_p \in \oplus_{\hat{x}}$ in the Hamming space. When this happens, it is no longer possible to exactly recover the neighborhood structure by thresholding d_Φ. Consequently, when Φ is used to perform retrieval, the retrieved "nearest neighbors" of $\hat{x}$ would be contaminated by non-neighbors. Therefore, we conclude that a high-quality embedding should minimize neighborhood ambiguity.

To concretely formulate the idea, we define random variable $\mathcal{D}_{\hat{x},\Phi}\colon \mathcal{X} \to \{0,1,\dots,b\}, x \mapsto d_\Phi(x,\hat{x})$, and let $\mathcal{C}_{\hat{x}}\colon \mathcal{X} \to \{0,1\}$ be the membership indicator for the set $\oplus_{\hat{x}}$. Then, we naturally have two conditional distributions of the Hamming distance: $P(\mathcal{D}_{\hat{x},\Phi}|\mathcal{C}_{\hat{x}}=1)$ and $P(\mathcal{D}_{\hat{x},\Phi}|\mathcal{C}_{\hat{x}}=0)$. Note that the constraint in Equation 1 can be re-expressed as having no overlap between these two conditional distributions, and that minimizing neighborhood ambiguity amounts to minimizing the overlap. Please see Figure 1 for an illustration.

We use the *mutual information* between random variables $\mathcal{D}_{\hat{x},\Phi}$ and $\mathcal{C}_{\hat{x}}$ to capture the amount of overlap between conditional Hamming distance distributions. The mutual information is defined as

$$\mathcal{I}(\mathcal{D}_{\hat{x},\Phi};\mathcal{C}_{\hat{x}}) = H(\mathcal{C}_{\hat{x}}) - H(\mathcal{C}_{\hat{x}}|\mathcal{D}_{\hat{x},\Phi}) \quad (7)$$
$$= H(\mathcal{D}_{\hat{x},\Phi}) - H(\mathcal{D}_{\hat{x},\Phi}|\mathcal{C}_{\hat{x}}) \quad (8)$$

where H denotes (conditional) entropy. In the following, for brevity we will drop subscripts Φ and $\hat{x}$, and denote the two conditional distributions and the marginal $P(\mathcal{D}_{\hat{x},\Phi})$ as $p_\mathcal{D}^+$, $p_\mathcal{D}^-$, and $p_\mathcal{D}$, respectively.

By definition, $\mathcal{I}(\mathcal{D};\mathcal{C})$ measures the decrease in uncertainty in the neighborhood information $\mathcal{C}$ when observing the Hamming distances $\mathcal{D}$. If $\mathcal{I}(\mathcal{D};\mathcal{C})$ attains a high value, which means $\mathcal{C}$ can be determined with low uncertainty by observing $\mathcal{D}$, then Φ must have achieved good separation (*i.e.* low overlap) between $p_\mathcal{D}^+$ and $p_\mathcal{D}^-$. $\mathcal{I}$ is maximized when there is no overlap, and minimized when $p_\mathcal{D}^+$ and $p_\mathcal{D}^-$ are exactly identical. As $H(\mathcal{C}_{\hat{x}})$ is typically constant, maximizing mutual information corresponds to minimizing the conditional entropy $H(\mathcal{C}_{\hat{x}}|\mathcal{D}_{\hat{x},\Phi})$. Note that this conditional entropy directly corresponds to the neighborhood ambiguity when $\mathcal{D}$ is observed. Mutual information is also related to the *Kullback-Leibler* divergence measure D_{KL}, specifically as

$$\mathcal{I}(\mathcal{D};\mathcal{C}) = \mathbb{E}_\mathcal{D}[\![D_{KL}(P(\mathcal{C}|\mathcal{D})||P(\mathcal{C}))]\!], \quad (9)$$

corresponding to the expected divergence between the distributions $P(\mathcal{C}|\mathcal{D})$ and $P(\mathcal{C})$. Intuitively, if $\mathcal{D}$ were to be informative, these two Bernoulli distributions should differ. Indeed, maximizing the D_{KL} divergence maximizes the difference of the two distributions.

Next, for any hash mapping Φ, we can integrate $\mathcal{I}$ over the feature space to give a quality measure:

$$\mathcal{O}(\Phi) = \int_\mathcal{X} \mathcal{I}(\mathcal{D}_{\hat{x},\Phi};\mathcal{C}_{\hat{x}})p(\hat{x})d\hat{x}. \quad (10)$$

An appealing property of this mutual information objective is that it is *parameter-free*: the objective encourages distributions $p_\mathcal{D}^+$ and $p_\mathcal{D}^-$ to be separated, but does not include parameters dictating the distance threshold at which separation occurs, or the absolute amount of separation. The absence of such fixed parameters also increases flexibility, since the separation could occur at different distance thresholds depending on the anchor $\hat{x}$.

4 Optimizing Mutual Information

Having shown that mutual information is a suitable measure of hashing quality, we consider its use as a learning objective.

Clearly, optimizing $\mathcal{O}(\Phi)$, as defined in Equation 10, is intractable. As is usually the case in supervised learning, we optimize the parameters of Φ over a finite training set $\mathcal{T}$ of i.i.d. samples from $p(\hat{x})$. Our learning problem is then formulated as

$$\max_{\Phi} \frac{1}{|\mathcal{T}|} \sum_{x \in \mathcal{T}} \mathcal{I}(\mathcal{D}_{x,\Phi}; \mathcal{C}_x). \tag{11}$$

It is worth noting that for each $x \in \mathcal{T}$, elements of $\oplus_x$ and $\ominus_x$ are now restricted to be within $\mathcal{T}$. Inspired by recent advances in stochastic optimization, we will use stochastic gradient descent to solve the above problem.

We start by deriving the gradients of $\mathcal{I}$ with respect to the output of the hash mapping, $\Phi(x)$. First, note that with b-bit Hamming distances, the discrete distributions $p_{\mathcal{D}}^{+}$ and $p_{\mathcal{D}}^{-}$ can be modeled using normalized histograms over $\{0, \dots, b\}$. Specifically, let $p_{\mathcal{D},l}^{+}$ be the l-th element of $p_{\mathcal{D}}^{+}$, which is estimated by performing hard assignments on Hamming distances into histogram bins:

$$p_{\mathcal{D},l}^{+} = \frac{1}{|\oplus_{\hat{x}}|} \sum_{x \in \oplus_{\hat{x}}} \mathbf{1}[d_{\Phi}(\hat{x}, x) = l],\ l = 0, \dots, b, \tag{12}$$

where $\mathbf{1}[\cdot]$ denotes the binary indicator.

The mutual information $\mathcal{I}$ is continuously differentiable, and using the chain rule we can write

$$\frac{\partial \mathcal{I}}{\partial \Phi(x)} = \sum_{l=0}^{b} \left[\frac{\partial \mathcal{I}}{\partial p_{\mathcal{D},l}^{+}} \frac{\partial p_{\mathcal{D},l}^{+}}{\partial \Phi(x)} + \frac{\partial \mathcal{I}}{\partial p_{\mathcal{D},l}^{-}} \frac{\partial p_{\mathcal{D},l}^{-}}{\partial \Phi(x)} \right]. \tag{13}$$

Due to symmetry, we next only focus on terms involving $p_{\mathcal{D}}^{+}$. Let p^{+} and p^{-} be shorthands for the priors $P(\mathcal{C} = 1)$ and $P(\mathcal{C} = 0)$. For $l = 0, \dots, b$, we have

$$\frac{\partial \mathcal{I}}{\partial p_{\mathcal{D},l}^{+}} = -\frac{\partial H(\mathcal{D}|\mathcal{C})}{\partial p_{\mathcal{D},l}^{+}} + \frac{\partial H(\mathcal{D})}{\partial p_{\mathcal{D},l}^{+}} \tag{14}$$

$$= p^{+}(\log p_{\mathcal{D},l}^{+} + 1) - (\log p_{\mathcal{D},l} + 1)\frac{\partial p_{\mathcal{D},l}}{\partial p_{\mathcal{D},l}^{+}} \tag{15}$$

$$= p^{+}(\log p_{\mathcal{D},l}^{+} - \log p_{\mathcal{D},l}). \tag{16}$$

Note that for Equation 16, we used the fact that

$$p_{\mathcal{D},l} = p^{+} p_{\mathcal{D},l}^{+} + p^{-} p_{\mathcal{D},l}^{-}. \tag{17}$$

4.1 Continuous Relaxation

To complete the chain rule, we need to further derive the term $\partial p_{\mathcal{D},l}^{+} / \partial \Phi(x)$ in Equation 13. However, the hash mapping Φ is discrete by nature, precluding the use of continuous optimization. While it is possible to maintain such constraints and resort to discrete optimization, the resulting optimization problems are NP-hard.

Instead, in order to apply gradient-based continuous optimization, we take the relaxation approach to sidestep the NP-hard problems. Correspondingly, we need to perform a continuously differentiable relaxation to Φ. Recall from Equation 4 that each element in Φ is obtained by thresholding neural network activations with the sign function. We relax Φ into a real-valued vector $\hat{\Phi}$ by adopting a standard technique in hashing [3], [30], [39], where the discontinuous sign function is approximated with the sigmoid function $\sigma : \mathbb{R} \to (0, 1)$:

$$\hat{\Phi}(x) = (\hat{\phi}_1(x), \dots, \hat{\phi}_b(x)), \tag{18}$$

$$\hat{\phi}_i(x) = 2\sigma(\gamma f_i(x; w)) - 1 \in (-1, 1). \tag{19}$$

We include a tuning parameter γ, used to control the "steepness" of the sigmoid approximation. Typically, we choose $\gamma \geq 1$ so as to reduce the error introduced by the continuous relaxation. Although with $\gamma \to \infty$ the sigmoid approximation approaches the sign function, a large γ can make gradients vanish due to the saturation of the sigmoid function. We empirically evaluate the choice of γ with an ablation study in Section 5, and find that MIHash is quite robust with respect to the continuous relaxation. Other alternative relaxation strategies include using a quantization error term [30], [60] and applying the continuation method [4].

With the continuous relaxation in place, we move on to the partial differentiation of $p_{\mathcal{D}}^{+}$ and $p_{\mathcal{D}}^{-}$ with respect to $\hat{\Phi}(x)$. As mentioned before, these discrete distributions can be estimated via histogram binning (Equation 12); however, histogram binning is a non-differentiable operation, due to the use of the binary indicator function. In the following, we describe a differentiable approximation to the discrete histogram binning process, thereby enabling end-to-end backpropagation.

4.2 End-to-End Optimization

Without the continuous relaxation, Equation 12 performs histogram binning by assigning $d_{\Phi}(\hat{x}, x)$, which is an integer, into a specific bin. With the continuous relaxation developed above, we note that d_{Φ} in is no longer integer-valued, but is also continuously relaxed into

$$\hat{d}_{\Phi}(\hat{x}, x) = \frac{b - \hat{\Phi}(\hat{x})^{\top} \hat{\Phi}(x)}{2}. \tag{20}$$

When d_{Φ} is relaxed into $\hat{d}_{\Phi}$, we need to replace the hard assignment with soft assignment. The key is to approximate the binary indicator $\mathbf{1}[\cdot]$ with a differentiable function. For this purpose, we employ a technique from [51]. Specifically, the binary indicator is replaced by a triangular kernel function δ with slope parameter $\Delta > 0$, centered on the histogram bin center, which linearly interpolates the real-valued $\hat{d}_{\Phi}(\hat{x}, x)$ into the l-th bin:

$$\delta(d, l) = \max\left\{0, 1 - \frac{|d - l|}{\Delta}\right\}. \tag{21}$$

It is easy to see that this triangular approximation approaches the original binary indicator as $\Delta \to 0$. Also, this soft assignment admits simple subgradients:

$$\delta_l'(d) \triangleq \frac{\partial \delta(d, l)}{\partial d} = \begin{cases} 1/\Delta, & d \in [l - \Delta, l], \\ -1/\Delta, & d \in [l, l + \Delta], \\ 0, & \text{otherwise}. \end{cases} \tag{22}$$

We are now ready to tackle the term $\partial p_{\mathcal{D},l}^{+} / \partial \hat{\Phi}(x)$. From the definition of $p_{\mathcal{D},l}^{+}$ in Equation 12, we have, for $x = \hat{x}$:

$$\frac{\partial p_{\mathcal{D},l}^{+}}{\partial \hat{\Phi}(\hat{x})} = \frac{1}{|\oplus_{\hat{x}}|} \sum_{x \in \oplus_{\hat{x}}} \frac{\partial \delta(\hat{d}_{\Phi}(\hat{x}, x), l)}{\partial \hat{\Phi}(\hat{x})} \tag{23}$$

$$= \frac{1}{|\oplus_{\hat{x}}|} \sum_{x \in \oplus_{\hat{x}}} \frac{\partial \delta(\hat{d}_{\Phi}(\hat{x}, x), l)}{\partial \hat{d}_{\Phi}(\hat{x}, x)} \frac{\partial \hat{d}_{\Phi}(\hat{x}, x)}{\partial \hat{\Phi}(\hat{x})} \tag{24}$$

$$= \frac{-1}{2|\oplus_{\hat{x}}|} \sum_{x \in \oplus_{\hat{x}}} \delta_l'(\hat{d}_{\Phi}(\hat{x}, x)) \hat{\Phi}(x). \tag{25}$$

For the last step, we used the definition of $\hat{d}_{\Phi}$ in Equation 20. Next, for $x \neq \hat{x}$:

$$\frac{\partial p_{\mathcal{D},l}^{+}}{\partial \hat{\Phi}(x)} = \frac{1}{|\oplus_{\hat{x}}|} \mathbf{1}[x \in \oplus_{\hat{x}}] \frac{\partial \delta(\hat{d}_{\Phi}(\hat{x}, x), l)}{\partial \hat{d}_{\Phi}(\hat{x}, x)} \frac{\partial \hat{d}_{\Phi}(\hat{x}, x)}{\partial \hat{\Phi}(x)} \tag{26}$$

$$= \frac{-1}{2|\oplus_{\hat{x}}|} \mathbf{1}[x \in \oplus_{\hat{x}}] \delta_l'(\hat{d}_{\Phi}(\hat{x}, x)) \hat{\Phi}(\hat{x}). \tag{27}$$

Lastly, to back-propagate gradients to $\hat{\Phi}$'s inputs, and ultimately to the parameters of the underlying deep neural network, we only need to further differentiate the sigmoid approximation employed in $\hat{\Phi}$ (Equation 19). The derivative of the sigmoid function has a closed form expression, and is omitted here.

4.3 Efficient Minibatch Backpropagation

So far, our derivations of mutual information and its gradients have assumed a single anchor example $\hat{x}$. In information retrieval terminology, the current derivations are for a single query and a fixed database. However, the optimization objective in Equation 11 is the average of mutual information values over all anchors in a finite training set $\mathcal{T}$. We now address this mismatch.

We face two challenges when working with a (potentially large) training set $\mathcal{T}$. First, we need to perform the optimization in the stochastic/minibatch setting, since deep neural networks are typically trained by minibatch stochastic gradient descent (SGD), where it can be infeasible to access the entire database all at once. The second challenge is that, differently from traditional information retrieval, in many computer vision tasks (*e.g.* image retrieval), there is usually no clear split of a given training set into a set of queries and a database. This is due to the symmetry that an image can either be a query or a database item. Consequently, even if we were to create such a split, it can be arbitrary and does not fully utilize available supervision.

Here, we describe a way to efficiently utilize all the available supervision during minibatch SGD training, simultaneously addressing both challenges. Our reasoning is that, within a minibatch with M examples, a retrieval problem can be defined by retrieving one example (the query) against the other $M-1$ examples (the database). Further, considering the symmetry mentioned above, retrieval can be repeated M times, each time using a different example as the query. Then, the overall objective value for the minibatch is the average over the M individual retrieval problems. This way, the available supervision in the minibatch is utilized maximally. As we shall see next, the backpropagation in this case can be efficiently implemented using matrix multiplications.

Now consider a minibatch of size M, $\mathcal{B} = \{x_1, \ldots, x_M\}$. Since we only operate within the minibatch, for each example $x_i \in \mathcal{B}, 1 \le i \le M$, when used as the query, its neighborhood structure is now defined within $\mathcal{B}$: we take $\oplus_i = \oplus_{x_i} \cup \mathcal{B}$, and $\ominus_i = \ominus_{x_i} \cup \mathcal{B}$. Also, let $\mathcal{I}_i$ be a shorthand for $\mathcal{I}(\mathcal{D}_{x_i,\Phi}, C_{x_i})$. We group the relaxed hash mapping output for the entire minibatch into the following $b \times M$ matrix,

$$\hat{\mathbf{\Phi}} = \left[\hat{\Phi}(x_1)\ \hat{\Phi}(x_2)\ \cdots\ \hat{\Phi}(x_M)\right] \in \mathbb{R}^{b\times M}. \tag{28}$$

Similar to Equation 13, we can write the Jacobian matrix of the minibatch objective $\mathcal{O}_\mathcal{B}$ with respect to $\hat{\mathbf{\Phi}}$ as

$$\begin{aligned}\frac{\partial \mathcal{O}_\mathcal{B}}{\partial \hat{\mathbf{\Phi}}} &= \frac{1}{M}\sum_{i=1}^{M}\frac{\partial \mathcal{I}_i}{\partial \hat{\mathbf{\Phi}}} &(29)\\ &= \frac{1}{M}\left[\sum_{i=1}^{M}\sum_{l=0}^{b}\frac{\partial \mathcal{I}_i}{\partial p_{i,l}^+}\frac{\partial p_{i,l}^+}{\partial \hat{\mathbf{\Phi}}} + \sum_{i=1}^{M}\sum_{l=0}^{b}\frac{\partial \mathcal{I}_i}{\partial p_{i,l}^-}\frac{\partial p_{i,l}^-}{\partial \hat{\mathbf{\Phi}}}\right] &(30)\end{aligned}$$

where $p_{i,l}^+$ ($p_{i,l}^-$) denotes the l-th element of $p_\mathcal{D}^+$ ($p_\mathcal{D}^-$) when the query is x_i.

Again, we have covered the derivation of the partial derivative $\partial \mathcal{I}_i / \partial p_{i,l}^+$, and now the main issue is evaluating the Jacobian $\partial p_{i,l}^+ / \partial \hat{\mathbf{\Phi}}$. We do so by examining each column of the Jacobian. First, for $\forall j \neq i$,

$$\begin{aligned}\frac{\partial p_{i,l}^+}{\partial \hat{\Phi}(x_j)} &= \frac{\partial p_{i,l}^+}{\partial \hat{d}_\Phi(x_i,x_j)}\frac{\partial \hat{d}_\Phi(x_i,x_j)}{\partial \hat{\Phi}(x_j)} &(31)\\ &= \frac{\mathbf{1}[x_j \in \oplus_i]}{|\oplus_i|}\delta_l'(\hat{d}_\Phi(x_i,x_j))\frac{-\hat{\Phi}(x_i)}{2} &(32)\\ &\triangleq \frac{\beta_l^+(i,j)}{N_i^+}\frac{-\hat{\Phi}(x_i)}{2}, &(33)\end{aligned}$$

where we have made the following substitutions:

$$N_i^+ = |\oplus_i|, \tag{34}$$

$$\beta_l^+(i,j) = \mathbf{1}[x_j \in \oplus_i]\delta_l'(\hat{d}_\Phi(x_i,x_j)). \tag{35}$$

Next, for $j = i$,

$$\begin{aligned}\frac{\partial p_{i,l}^+}{\partial \hat{\Phi}(x_i)} &= \sum_{k\neq i}\frac{\partial p_{i,l}^+}{\partial \hat{d}_\Phi(x_i,x_k)}\frac{\partial \hat{d}_\Phi(x_i,x_k)}{\partial \hat{\Phi}(x_i)} &(36)\\ &= \sum_{k\neq i}\frac{\beta_l^+(i,k)}{N_i^+}\frac{-\hat{\Phi}(x_k)}{2}. &(37)\end{aligned}$$

By defining that $\beta_l^+(i,i) \equiv 0, \forall i$, we can further unify these two cases as:

$$\frac{\partial p_{i,l}^+}{\partial \hat{\Phi}(x_j)} = -\frac{\beta_l^+(i,j)\hat{\Phi}(x_i)}{2N_i^+} - \mathbf{1}[j=i]\sum_{k=1}^{M}\frac{\beta_l^+(i,k)\hat{\Phi}(x_k)}{2N_i^+}. \tag{38}$$

Having derived all the columns, we now write down the matrix form of $\partial p_{i,l}^+ / \partial \hat{\mathbf{\Phi}}$. First, we define $\boldsymbol{\beta}_{l,i}^+ = (\beta_l^+(i,1), \ldots, \beta_l^+(i,M)) \in \mathbb{R}^M$, and let $\boldsymbol{e}_i$ be the i-th standard basis vector in $\mathbb{R}^M$ (*i.e.*, the i-th element is 1 and other elements are 0). The matrix form can be compactly written as

$$\begin{aligned}\frac{\partial p_{i,l}^+}{\partial \hat{\mathbf{\Phi}}} &= -\frac{\hat{\Phi}(x_i)(\boldsymbol{\beta}_{l,i}^+)^\top}{2N_i^+} - \left[\sum_{k=1}^{M}\frac{\beta_l^+(i,k)\hat{\Phi}(x_k)}{2N_i^+}\right]\boldsymbol{e}_i^\top &(39)\\ &= -\frac{1}{2N_i^+}\left[\hat{\Phi}(x_i)(\boldsymbol{\beta}_{l,i}^+)^\top + \hat{\mathbf{\Phi}}\boldsymbol{\beta}_{l,i}^+\boldsymbol{e}_i^\top\right]. &(40)\end{aligned}$$

We will next complete Equation 30. First, we define a shorthand, which can be easily evaluated using the result in Equation 16:

$$\alpha_{l,i}^+ = \frac{1}{N_i^+}\frac{\partial \mathcal{I}_i}{\partial p_{i,l}^+}. \tag{41}$$

Using symmetry, we only consider the first term involving p_i^+ in Equation 30, and we omit the $1/M$ scaling factor for now:

$$\begin{aligned}&\sum_{i=1}^{M}\sum_{l=0}^{b}\frac{\partial \mathcal{I}_i}{\partial p_{i,l}^+}\frac{\partial p_{i,l}^+}{\partial \hat{\mathbf{\Phi}}} &(42)\\ =&\sum_{l=0}^{b}\sum_{i=1}^{M} -\frac{1}{2N_i^+}\frac{\partial \mathcal{I}_i}{\partial p_{i,l}^+}\left[\hat{\Phi}(x_i)(\boldsymbol{\beta}_{l,i}^+)^\top + \hat{\mathbf{\Phi}}\boldsymbol{\beta}_{l,i}^+\boldsymbol{e}_i^\top\right] &(43)\\ =& -\frac{1}{2}\sum_{l=0}^{b}\alpha_{l,i}^+\left[\sum_{i=1}^{M}\hat{\Phi}(x_i)(\boldsymbol{\beta}_{l,i}^+)^\top + \hat{\mathbf{\Phi}}\sum_{i=1}^{M}\boldsymbol{\beta}_{l,i}^+\boldsymbol{e}_i^\top\right] &(44)\\ =& -\frac{1}{2}\sum_{l=0}^{b}\left[\sum_{i=1}^{M}\alpha_{l,i}^+\hat{\Phi}(x_i)(\boldsymbol{\beta}_{l,i}^+)^\top + \hat{\mathbf{\Phi}}\sum_{i=1}^{M}\alpha_{l,i}^+\boldsymbol{\beta}_{l,i}^+\boldsymbol{e}_i^\top\right]. &(45)\end{aligned}$$

Define

$$A_l^+ = \operatorname{diag}(\alpha_{l,1}^+, \ldots, \alpha_{l,M}^+) \in \mathbb{R}^{M\times M}, \tag{46}$$

$$B_l^+ = \left[\boldsymbol{\beta}_{l,1}^+ \cdots \boldsymbol{\beta}_{l,M}^+\right] \in \mathbb{R}^{M\times M}, \tag{47}$$

then we can simplify Equation 45 as

$$-\frac{1}{2}\sum_{l=0}^{b}\left[\hat{\boldsymbol{\Phi}} A_l^+ (B_l^+)^\top + \hat{\boldsymbol{\Phi}} B_l^+ A_l^+\right] \tag{48}$$

$$= -\frac{1}{2}\hat{\boldsymbol{\Phi}}\sum_{l=0}^{b}\left(A_l^+ B_l^+ + B_l^+ A_l^+\right). \tag{49}$$

The last step is true since B_l^+ is symmetric: it can be seen from the definition of β^+ in Equation 35 that $\beta_l^+(i,j) = \beta_l^+(j,i)$, since both the neighbor relationship and the Hamming distance are symmetric.

Now, if we define A_l^- and B_l^- for the non-neighbor distance distribution $p_{\mathcal{D}}^-$, analogously as in Equations 46 and 47 (details are very similar and omitted), then the full Jacobian matrix in Equation 30 can be evaluated as

$$-\frac{\hat{\boldsymbol{\Phi}}}{2M}\sum_{l=0}^{b}(A_l^+ B_l^+ + B_l^+ A_l^+ + A_l^- B_l^- + B_l^- A_l^-). \tag{50}$$

Since only matrix multiplications and additions are involved, this operation can be implemented efficiently. In particular, note that A_l^+ (A_l^-) is a diagonal matrix, therefore multiplying with B_l^+ (B_l^-) effectively scales its rows or columns, which has $O(M^2)$ time complexity, as opposed to general matrix multiplication which is $O(M^3)$. We then conclude that the overall time complexity for computing Equation 50 is $O(bM^2)$.

Recently, Triantafillou *et al.* [49] also propose a minibatch-based learning formulation that is inspired by information retrieval, which attempts to maximize the utilization of supervision by treating each example in the minibatch as a query. However, we note that [49] tackles the problem of few-shot learning by learning real-valued embeddings, and it uses very different optimization machinery to approximately optimize mean Average Precision in a structured prediction framework. Nevertheless, it would be interesting to explore the use of hashing and the mutual information objective for that problem in future work.

5 Experiments

5.1 Datasets and Evaluation Setup

We conduct experiments on widely used image retrieval benchmarks: CIFAR-10 [25], NUSWIDE [8], 22K LabelMe [42] and ImageNet100 [12]. Each dataset is split into a test set and retrieval set, and instances from the retrieval set are used in training. We follow a standard information retrieval setup: at test time, queries from the test set are used to rank instances from the retrieval set using Hamming distances, and the performance metric is averaged over the queries.

- **CIFAR-10** is a dataset for image classification and retrieval, containing 60K images from 10 different categories. We follow the setup of [29], [30], [60], [69]. This setup corresponds to two distinct partitions of the dataset. In the first case (*cifar-1*), we sample 500 images per category, resulting in 5,000 training examples to learn the hash mapping. The test set contains 100 images per category (1000 in total). The remaining images are then used to populate the hash table. In the second case (*cifar-2*), we sample 1000 images per category to construct the test set (10,000 in total). The remaining items are both used to learn the hash mapping and populate the hash table. Two images are considered neighbors if they belong to the same class.
- **NUSWIDE** is a dataset containing 269K images from Flickr. Each image can be associated with multiple labels, corresponding with 81 ground truth concepts. For NUSWIDE experiments, following the setup in [29], [30], [60], [69], we only consider images annotated with the 21 most frequent labels. In total, this corresponds to 195,834 images. The experimental setup also has two distinct partitionings: *nus-1* and *nus-2*. For both cases, a test set is constructed by randomly sampling 100 images per label (2,100 images in total). To learn the hash mapping, 500 images per label are randomly sampled in *nus-1* (10,500 in total). The remaining images are then used to populate the hash table. In the second case, *nus-2*, all the images excluding the test set are used in learning and populating the hash table. Following standard practice, two images are considered as neighbors if they share at least one label.
- **22K LabelMe** consists of 22K images, each represented with a 512-dimensionality GIST descriptor. Following [3], [27], we randomly partition the dataset into a retrieval and a test set, consisting of 20K and 2K instances, respectively. A 5K subset of the retrieval set is used in learning the hash mapping. As this dataset is unsupervised, we use the Euclidean distance between GIST features in determining the neighborhood structure. Two examples that have a Euclidean distance below the 5% distance percentile are considered neighbors.
- **ImageNet100** is a subset of ImageNet [12] containing 130K images from 100 classes. We follow the setup in [4], and randomly sample 100 images per class for training. All images in the selected classes from the ILSVRC 2012 validation set are used as the test set. Two images are considered neighbors if they belong to the same class.

As for performance metric, we use the standard mean Average Precision (mAP), or its variants. We compare MIHash against both classical and recent state-of-the-art hashing methods. These methods include: Spectral Hashing (**SH**) [61], Iterative Quantization (**ITQ**) [19], Sequential Projection Learning for Hashing (**SPLH**) [55], Supervised Hashing with Kernels (**SHK**) [39], Fast Supervised Hashing with Decision Trees (**FastHash**) [33], Structured Hashing (**StructHash**) [32], Supervised Discrete Hashing (**SDH**) [45], Efficient Training of Very Deep Neural Networks (**VDSH**) [66], Deep Supervised Hashing with Pairwise Labels (**DPSH**) [30], Deep Supervised Hashing with Triplet Labels (**DTSH**) [60], and Hashing by Continuation (**HashNet**) [4]. These competing methods have been shown to outperform earlier and other works such as [27], [29], [41], [56], [62], [67].

We finetune deep Convolutional Neural Network models that are pretrained on the ImageNet dataset, by replacing the final softmax classification layer with a new fully-connected layer that produces the binary bits. The new fully-connected layer is randomly initialized. For CIFAR-10 and NUSWIDE experiments, we finetune a VGG-F [6] architecture, as in [30], [60]. For ImageNet100 experiments, following the protocol of HashNet [4], we finetune the AlexNet [26] architecture, and scale down the learning rate for pretrained layers by a factor of 0.01, since the model is finetuned on the same dataset for a different task. For *non-deep* methods, we use the output of the penultimate layer ($fc7$) of both architectures as input features, which are 4096-dimensional. For the 22K LabelMe benchmark, all methods learn shallow models on top of precomputed 512-dimensional GIST descriptors. For gradient-based hashing methods, this corresponds to learning a single fully connected layer.

We use SGD with momentum 0.9 and weight decay of 5×10^{-4}, and reduce the learning rate periodically by a predetermined factor (0.5 in most cases), which is standard practice. During training, the minibatches are randomly sampled from the training set.

5.2 Results

Table 1 gives results for *cifar-1* and *nus-1* experimental settings in which mAP and mAP@5K (mAP evaluated on the top 5,000 retrievals) are reported for the CIFAR-10 and NUSWIDE datasets, respectively. Deep learning based hashing methods such as DPSH and DTSH outperform most non-deep hashing solutions. This is not surprising as the hash mapping is learned simultaneously with feature learning. Non-deep solutions such as FastHash and SDH also perform competitively, especially in NUSWIDE experiments. Our proposed method, MIHash, surpasses all competing methods in the majority of the experiments. For example, with 32 and 48-bit binary embeddings MIHash surpasses the nearest competitor, DTSH, by 3%-4% in CIFAR-10. For NUSWIDE, MIHash achieves state-of-the-art performances in all experiments excluding with 12 bits.

The performance improvement of MIHash is much more significant in the *cifar-2* and *nus-2* settings, where more training data is available. In these settings, a VGG-F network pretrained on ImageNet is again finetuned. Following standard practice, mAP and mAP@50K metrics are used to evaluate the retrieval performance. Table 2 gives the results. As can be observed, in both CIFAR-10 and NUSWIDE, MIHash achieves state-of-the-art results in nearly all code lengths. For instance, MIHash consistently outperforms DTSH, the closest competitor, by a large margin in all embedding sizes.

Retrieval results for ImageNet100 are given in Table 3. In these experiments, we compare against DTSH, the overall best competing method in past experiments and another recently introduced deep learning based hashing method, HashNet [4]. Note that, HashNet also outperforms shallow methods such as [39] and [45] with deep features on ImageNet100, as reported in [4]. The evaluation metric is taken to be mAP@1K for consistency with the setup in [4]. In this benchmark, MIHash significantly outperforms both DTSH and HashNet for all embedding sizes. Notably, MIHash achieves 6% improvement over HashNet with 16-bit codes, indicating its superiority in learning high-quality compact binary codes.

To further emphasize the merits of MIHash, we consider shallow model experiments on the 22K LabelMe dataset. In this benchmark, we only consider the overall best non-deep and deep learning methods in past experiments. Also, to solely put emphasis on comparing the hash mapping learning objectives, all deep learning methods use a one-layer embedding on top of the GIST descriptor. The GIST descriptor is prominently used even in many recent hashing studies (*e.g.*, as in [33] and [32]). Its usage nullifies the feature learning aspect in deep hashing techniques enabling a more direct comparison to non-deep hashing methods. Still, some non-deep methods employ non-linear hash functions, such as FastHash and StructHash that use boosted decision trees. Table 4 gives the results, and we can see that non-deep methods FastHash and StructHash outperform deep learning methods DPSH and DTSH on this benchmark. This indicates that the prowess of DPSH and DTSH might come primarily through feature learning. On the other hand, MIHash is the best performing method across all code lengths, despite using a simpler one-layer embedding function compared to FastHash and StructHash. This further validates the effectiveness of our mutual information based objective in capturing the neighborhood structure.

5.3 Empirical Analysis and Ablation Studies

5.3.1 Mutual Information and Ranking Metrics

To evaluate the performance of hashing algorithms for retrieval tasks, it is common to use ranking-based metrics, and the most notable example is mean Average Precision (mAP). We note that there exists strong correlations between our mutual information criterion and mAP. Figure 2 provides an empirical analysis on the CIFAR-10 benchmark. The left plot displays the training objective value as computed from Equation 11 and the mAP score with respect to the epoch. These results are obtained from the *cifar-1* experiment with 32-bit codes, as specified in the previous section. Notice that both the mutual information objective and the mAP value show similar behavior, *i.e.* , exhibit strong correlation. While the mAP score increases from 0.40 to 0.80, the mutual information objective increases from 0.15 to above 0.40. In the middle figure, we apply min-max normalization in order to scale both measures to the same range.

To further analyze the correlation between mutual information and mAP, we also conducted an additional experiment in which 100 instances are selected as the query set, and the rest are used to populate the hash table. The hash mapping parameters are randomly sampled from a Gaussian distribution, similar to LSH [18], and each experiment is conducted 50 times. The right figure provides the scatter plot of mAP and the mutual information objective value. We can see that the relationship is almost linear, which is also validated by the Pearson Correlation Coefficient score of 0.98.

We give an intuitive explanation to the strong correlation. Given a query, the AP is optimized when all of its neighbors are ranked above all of its non-neighbors in the database. On the other hand, mutual information is optimized when the distribution of neighbor distances has no overlap with the distribution of non-neighbor distances. Therefore, we can see that AP and mutual information are simultaneously optimized by the same optimal solution. Conversely, AP is suboptimal when neighbors and non-neighbors are interleaved in the ranking, so is mutual information when the distance distributions have nonzero overlap. Although a theoretical analysis of the correlation is beyond the scope of this work, empirically we find that mutual information serves as a general-purpose surrogate metric for ranking.

An advantage of mutual information, as we have demonstrated, is that it is suitable for direct, gradient-based optimization. In contrast, optimizing mAP is much more challenging as it is non-differentiable, and previous works usually resort to approximation and bounding techniques [32], [59], [63].

VGG − F	CIFAR − 10				NUSWIDE			
Method	mAP				mAP@5K			
	12 Bits	24 Bits	32 Bits	48 Bits	12 Bits	24 Bits	32 Bits	48 Bits
SH [61]	0.183	0.164	0.161	0.161	0.621	0.616	0.615	0.612
ITQ [19]	0.237	0.246	0.255	0.261	0.719	0.739	0.747	0.756
SPLH [55]	0.299	0.33	0.335	0.33	0.753	0.775	0.783	0.786
SHK [39]	0.488	0.539	0.548	0.563	0.768	0.804	0.815	0.824
SDH [45]	0.478	0.557	0.584	0.592	**0.780**	0.804	0.816	0.824
FastHash [33]	0.553	0.607	0.619	0.636	0.779	0.807	0.816	0.825
StructHash [32]	0.664	0.693	0.691	0.700	0.748	0.772	0.790	0.801
VDSH [66]	0.538	0.541	0.545	0.548	0.769	0.796	0.803	0.807
DPSH [30]	0.713	0.727	0.744	0.757	0.758	0.793	0.818	0.830
DTSH [60]	0.710	0.750	0.765	0.774	0.773	0.813	0.820	0.838
MIHash	**0.738**	**0.775**	**0.791**	**0.816**	0.773	**0.820**	**0.831**	**0.843**

TABLE 1: Results on CIFAR-10 and NUSWIDE datasets with *cifar-1* and *nus-1* partitionings. The underlying deep learning architecture is VGG-F. MIHash outperforms competing methods on CIFAR-10, and shows improvements, especially with lengthier codes, on NUSWIDE.

VGG − F	CIFAR − 10				NUSWIDE			
Method	mAP				mAP@50K			
	16 Bits	24 Bits	32 Bits	48 Bits	16 Bits	24 Bits	32 Bits	48 Bits
DRSH [67]	0.608	0.611	0.617	0.618	0.609	0.618	0.621	0.631
DRSCH [65]	0.615	0.622	0.629	0.631	0.715	0.722	0.736	0.741
DPSH [30]	0.903	0.885	0.915	0.911	0.715	0.722	0.736	0.741
DTSH [60]	0.915	0.923	0.925	0.926	0.756	0.776	0.785	0.799
MIHash	**0.927**	**0.938**	**0.942**	**0.943**	**0.798**	**0.814**	**0.819**	**0.820**

TABLE 2: Results on CIFAR-10 and NUSWIDE datasets with *cifar-2* and *nus-2* partitionings (with VGG-F architecture). MIHash achieves new state-of-the-art performance, consistently improving over competing methods.

AlexNet	ImageNet100			
Method	mAP@1K			
	16 Bits	32 Bits	48 Bits	64 Bits
DTSH [60]	0.458	0.566	0.611	0.644
HashNet [4]	0.506	0.630	0.663	0.683
MIHash	**0.569**	**0.661**	**0.685**	**0.694**

TABLE 3: mAP@1K values on ImageNet100 using AlexNet. MIHash outperforms HashNet, a state-of-the-art deep hashing formulation using continuation methods [4].

GIST	22K LabelMe			
Method	mAP			
	16 Bits	32 Bits	48 Bits	64 Bits
DPSH [30]	0.295	0.346	0.391	0.427
DTSH [60]	0.304	0.342	0.361	0.378
FastHash [33]	0.324	0.394	0.433	0.456
StructHash [32]	0.369	0.474	0.538	0.582
MIHash	**0.384**	**0.496**	**0.554**	**0.598**

TABLE 4: 22K LabelMe results with GIST features. MIHash significantly improves over the state-of-the-art methods.

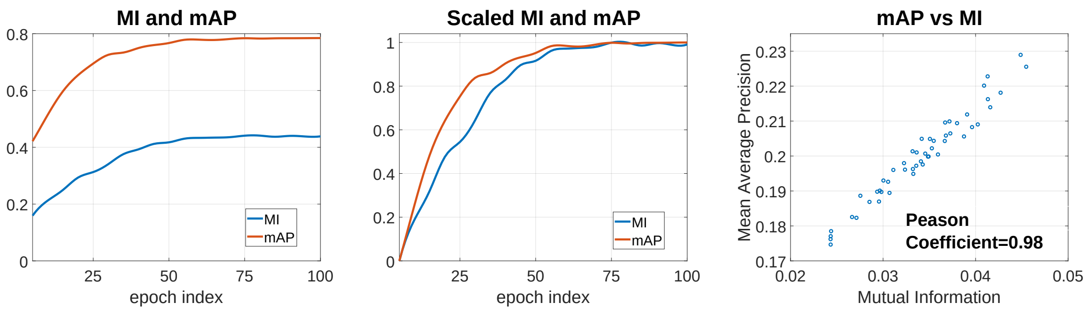


Fig. 2: (Left) We plot the training objective value (Equation 11) and the mAP score from the 32-bit *cifar-1* experiment. Notice that both the mutual information objective and the mAP value show similar behavior, *i.e.* , exhibit strong correlation. (Middle) We apply min-max normalization in order to scale both measures to the same range. (Right) We conduct an additional set of experiments in which 100 instances are selected as the query set, and the rest is used to populate the hash table. The hash mapping parameters are randomly sampled from a Gaussian, similar to LSH [18]. Each experiment is conducted 50 times. There exists strong correlation between MI and mAP as validated by the Pearson Correlation Coefficient score of 0.98.

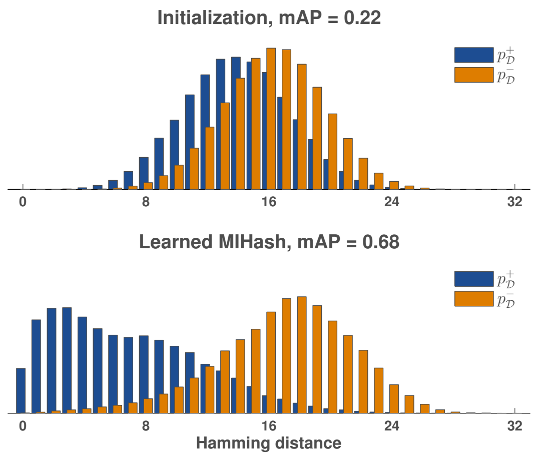


Fig. 3: We plot the distributions $p^+_\mathcal{D}$ and $p^-_\mathcal{D}$, averaged on the CIFAR-10 test set, before and after learning MIHash with a single-layer model and 20K training examples. Optimizing the mutual information objective substantially reduces the overlap between them, resulting in high mAP.

5.3.2 Distribution Separating Effect

To demonstrate that MIHash indeed separates neighbor and non-neighbor distance distributions, we consider a simple experiment. Specifically, we learn a single-layer model on top of precomputed $fc7$-layer features. The learning is done in an online fashion, which means that each training example is processed only once. We train such an MIHash model on the CIFAR-10 dataset with 20K training examples.

In Figure 3, we plot the distributions $p^+_\mathcal{D}$ and $p^-_\mathcal{D}$, averaged on the CIFAR-10 test set, before and after learning MIHash with 20K training examples. The hash mapping parameters are initialized using LSH, and lead to high overlap between the distributions, although they do not totally overlap due to the use of strong pretrained features. After learning, the overlap is significantly reduced, with $p^+_\mathcal{D}$ pushed towards zero hamming distances. Consequently, the mAP value increases to 0.68 from 0.22.

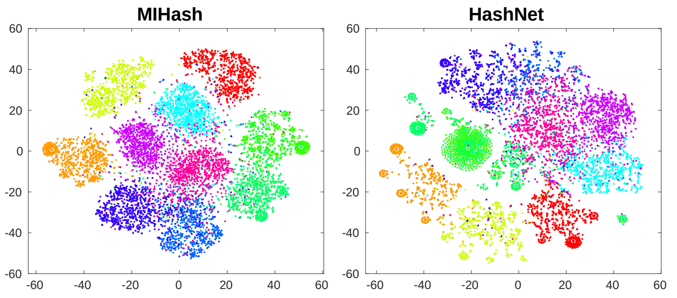


Fig. 4: t-SNE [52] visualization of the 48-bit binary codes produced by MIHash and HashNet on ImageNet100, for a random subset of 10 different color-coded classes in the test set. MIHash yields well-separated codes with distinct structures, opposed to HashNet, in which the binary codes have a higher overlap.

5.3.3 t-SNE Visualization of the Binary Embeddings

We also visualize the learned embeddings using t-SNE [52]. In Figure 4, we plot the visualization for 48-bit binary embeddings produced by MIHash and the top competing method, HashNet, on ImageNet100. For ease of visualization, we randomly sample 10 classes from the test set.

MIHash produces binary embeddings that separate different classes well into separate clusters. This is in fact predictable from the formulation of MIHash, in which the class overlap is quantified via mutual information and minimized. On the other hand, binary codes generated by HashNet have higher overlap between classes. This is also consistent with the fact that HashNet does not specifically optimize for a criterion related to class overlap, but belongs to the simpler "affinity matching" family of approaches.

5.3.4 Steepness parameter γ and Batch Size M

We provide an ablation study on the steepness parameter γ in Equation 19 and training minibatch size M. The experiments are conducted on the *cifar-1* benchmark with 32 bit codes.

Generally, continuous relaxation of the binary codes introduces discrepancies between the training and testing scenarios, and is thus prone to degrading the test-time retrieval performance. In deep hashing studies, this issue is often mitigated by a quantization loss (*e.g.* [30], [60]), or continuation methods [4], or by

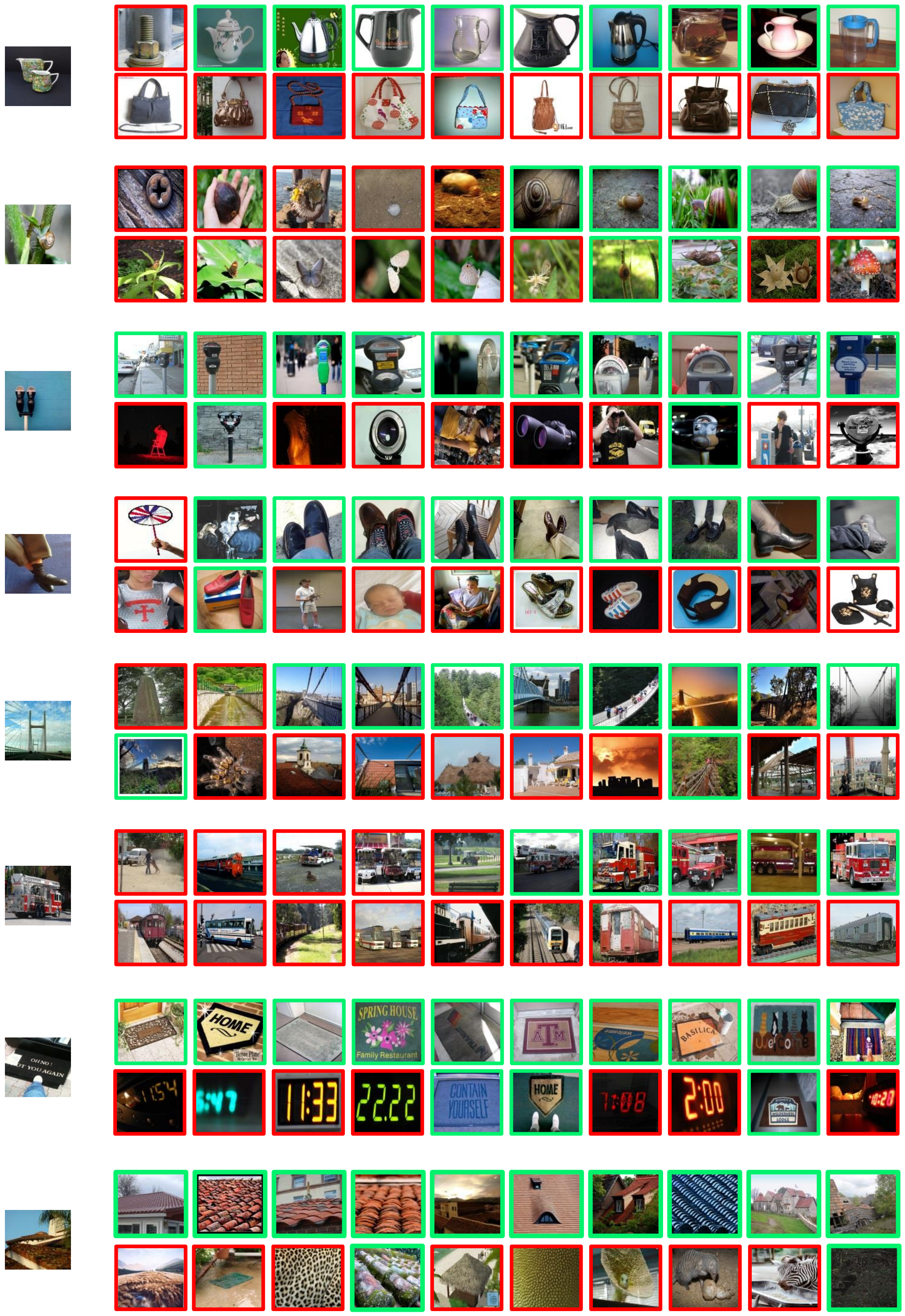


Fig. 5: We show sample retrieval results from the ImageNet100 dataset. Left: query images, right: top 10 retrieved images from MIHash (top row) and from HashNet (bottom row). Retrieved images marked with a green border belong to the same class as the query image, while ones marked with a red border do not belong to the same class as the query image.

VGG − F	CIFAR − 10 (mAP)			
Steepness γ	1	5	10	20
	0.791	0.768	0.749	0.776
Batch Size M	64	128	256	512
	0.771	0.765	0.791	0.783

TABLE 5: Ablation study for the steepness parameter γ and mini-batch size M on the *cifar-1* benchmark with 32 bit codes.

simply keeping the binary constraints [14]. However, we observe the MIHash model to be robust to the continuous relaxation: performance values are largely unaffected by variations in the γ parameter. This is also true for the minibatch size parameter M. This ablation study highlights the robustness of our MIHash formulation to the hyper-parameters.

5.3.5 Example Retrieval Results

In Figure 5, we present example retrieval results for MIHash and HashNet for several image queries from the ImageNet100 dataset. The top 10 retrievals of eight query images from eight distinct categories are presented. Correct retrievals (*i.e.* , having the same class label as the query) are marked in green, and incorrect retrievals are in red. In these examples, many of the retrieved images appear visually similar to the query, even if not sharing the same class label. Nevertheless, MIHash retrieves fewer incorrect images compared to HashNet. For example, HashNet returns bag images for the first query (image of cups), and digital-clock images for the second-to-last query (image of doormat).

6 Conclusion

We take an information-theoretic approach to hashing and propose a novel hashing method, called MIHash, in this work. It is based on minimizing neighborhood ambiguity in the learned Hamming space, which is crucial in maintaining high performance in nearest neighbor retrieval. We adopt the well-studied mutual information measure from information theory to quantify neighborhood ambiguity, and show that this measure has strong correlations with standard ranking-based retrieval performance metrics. Then, to optimize mutual information, we take advantage of recent advances in deep learning and stochastic optimization, and parameterize our embedding functions with deep neural networks. We perform a continuous relaxation on the NP-hard optimization problem, and use stochastic gradient descent to optimize the networks. In particular, our formulation maximally utilizes available supervision within each minibatch, and can be efficiently implemented. Our implementation is publicly available.

When evaluated on four standard image retrieval benchmarks, MIHash is shown to learn high-quality compact binary codes, and it achieves superior nearest neighbor retrieval performance compared to existing supervised hashing techniques. We believe that the mutual information based formulation is also potentially relevant for learning real-valued embeddings, and for other applications besides image retrieval, such as few-shot learning.

Acknowledgments

This research was supported in part by a BU IGNITION award, US NSF grant 1029430, and gifts from NVIDIA.

References

[1] A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proc. ACM Symposium on Theory of Computing (STOC)*, 2015.
[2] F. Cakir, K. He, S. Adel Bargal, and S. Sclaroff. Mihash: Online hashing with mutual information. In *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2017.
[3] F. Cakir and S. Sclaroff. Adaptive hashing for fast similarity search. In *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2015.
[4] Z. Cao, M. Long, J. Wang, and P. S. Yu. HashNet: Deep learning to hash by continuation. In *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2017.
[5] M. A. Carreira-Perpinan and R. Raziperchikolaei. Hashing with binary autoencoders. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference (BMVC)*, 2014.
[7] J. Cheng, C. Leng, J. Wu, H. Cui, and H. Lu. Fast and accurate image matching with cascade hashing for 3d reconstruction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.
[8] T. S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. T. Zheng. NUS-WIDE: A real-world web image database from National University of Singapore. In *Proc. ACM CIVR*, 2009.
[9] B. Dai, R. Guo, S. Kumar, N. He, and L. Song. Stochastic generative hashing. In *Proc. International Conf. on Machine Learning (ICML)*, 2017.
[10] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. on Computational geometry (SCG)*, 2004.
[11] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013.
[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
[13] K. Ding, C. Huo, B. Fan, and C. Pan. Knn hashing with factorized neighborhood representation. In *Proc. IEEE International Conf. on Computer Vision (ICCV)*, December 2015.
[14] T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to hash with binary deep neural network. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 219–234. Springer, 2016.
[15] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2475–2483, 2015.
[16] L. Gao, J. Song, F. Zou, D. Zhang, and J. Shao. Scalable multimedia retrieval by deep learning hashing with relative similarity learning. In *Proceedings of the 23rd ACM International Conference on Multimedia*, 2015.
[17] K. Ge, Tiezhengand He and J. Sun. Graph cuts for supervised binary coding. In *Proc. European Conf. on Computer Vision (ECCV)*, 2014.
[18] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. International Conf. on Very Large Data Bases (VLDB)*, 1999.
[19] Y. Gong and S. Lazebnik. Iterative quantization: A Procrustean approach to learning binary codes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011.
[20] Y. Gong, M. Pawlowski, F. Yang, L. Brandy, L. Boundev, and R. Fergus. Web scale photo hash clustering on a single machine. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.
[21] J. Han, Xin Jin. Locality sensitive hashing based clustering. In *Encyclopedia of Machine Learning*. Springer US, 2010.
[22] T. H. Haveliwala. Scalable techniques for clustering the web. In *Proc. of the WebDB Workshop*, 2000.
[23] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013.
[24] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2011.
[25] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. In *University of Toronto Technical Report*, 2009.
[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2012.
[27] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2009.
[28] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for

scalable image search. In *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2009.

[29] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[30] W. J. Li, S. Wang, and W. C. Kang. Feature learning based deep supervised hashing with pairwise labels. In *Proc. International Joint Conf. on Artificial Intelligence (IJCAI)*, 2016.

[31] X. Li, C. Shen, A. Dick, and A. van den Hengel. Learning compact binary codes for visual tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[32] G. Lin, F. Liu, C. Shen, J. Wu, and H. T. Shen. Structured learning of binary codes with column generation for optimizing ranking measures. *International Journal of Computer Vision (IJCV)*, 2016.

[33] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[34] G. Lin, C. Shen, D. Suter, and A. van den Hengel. A general two-step approach to learning-based hashing. In *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2013.

[35] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2015.

[36] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[37] R. S. Lin, D. A. Ross, and J. Yagnik. Spec hashing: Similarity preserving algorithm for entropy-based coding. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[38] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[39] J. W. Liu, Wei and, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[40] X. Liu, D. Tao, M. Song, Y. Ruan, C. Chen, and J. Bu. Weakly supervised multiclass video segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[41] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *Proc. International Conf. on Machine Learning (ICML)*, 2011.

[42] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. In *International Journal of Computer Vision (IJCV)*, 2008.

[43] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[44] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter sensitive hashing. In *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2003.

[45] F. Shen, C. S. Wei, L. Heng, and T. Shen. Supervised discrete hashing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[46] J. Song, L. Gao, Y. Yan, D. Zhang, and N. Sebe. Supervised hashing with pseudo labels for scalable multimedia retrieval. In *Proceedings of the 23rd ACM International Conference on Multimedia*, 2015.

[47] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo. Effective multiple feature hashing for large-scale near-duplicate video retrieval. *IEEE Transactions on Multimedia*, 2013.

[48] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. LDAHash: Improved matching with smaller descriptors. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2012.

[49] E. Triantafillou, R. Zemel, and R. Urtasun. Few-shot learning through an information retrieval lens. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 2252–2262, 2017.

[50] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, December 2005.

[51] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 4170–4178, 2016.

[52] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research (JMLR)*, 2008.

[53] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[54] A. L. C. Wang. An industrial-strength audio search algorithm. In *Proceedings of the 4th International Conference on Music Information Retrieval*, 2003.

[55] J. Wang, S. Kumar, and S. F. Chang. Sequential projection learning for hashing with compact codes. In *Proc. International Conf. on Machine Learning (ICML)*, 2010.

[56] J. Wang, S. Kumar, and S. F. Chang. Semi-supervised hashing for large-scale search. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2012.

[57] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. A survey on learning to hash. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2018.

[58] Q. Wang, B. Shen, S. Wang, L. Li, and L. Si. Binary codes embedding for fast image tagging with incomplete labels. In *Proc. European Conf. on Computer Vision (ECCV)*, 2014.

[59] Q. Wang, Z. Zhang, and L. Si. Ranking preserving hashing for fast similarity search. In *Proc. International Joint Conf. on Artificial Intelligence (IJCAI)*, 2015.

[60] Y. Wang, Xiaofang Shi and K. M. Kitani. Deep supervised hashing with triplet labels. In *Proc. Asian Conf. on Computer Vision (ACCV)*, 2016.

[61] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2008.

[62] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *Proc. AAAI Conf. on Artificial Intelligence (AAAI)*, volume 1, page 2, 2014.

[63] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proc. ACM Conf. on Research & Development in Information Retrieval (SIGIR)*, 2007.

[64] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proc. ACM Conf. on Research & Development in Information Retrieval (SIGIR)*, 2010.

[65] 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. *IEEE Trans. on Image Processing*, 2015.

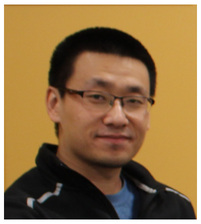
[66] Z. Zhang, Y. Chen, and V. Saligrama. Efficient training of very deep neural networks for supervised hashing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[67] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[68] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *Proc. AAAI Conf. on Artificial Intelligence (AAAI)*, 2016.

[69] B. Zhuang, G. Lin, C. Shen, and I. Reid. Fast training of triplet-based deep binary embedding networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Fatih Cakir is a Data Scientist at FirstFuel Software. He was previously a member at the Image and Video Computing Group at Boston University working with Professor Stan Sclaroff as his Ph.D. advisor. His research interests are in the fields of Computer Vision and Machine Learning.



Kun He is a Ph.D. candidate in Computer Science and a member of the Image and Video Computing group at Boston University, advised by Professor Stan Sclaroff. He obtained his M.Sc. degree in Computer Science from Boston University in 2013, and his B.Sc. degree (with honors) in Computer Science and Technology from Zhejiang University, Hangzhou, China, in 2010. He is a student member of the IEEE.

Sarah Adel Bargal is a Ph.D. candidate in the Image and Video Computing group in the Boston University Department of Computer Science. She received her M.Sc. from the American University in Cairo. Her research interests are in the areas of computer vision and deep learning. She is an IBM PhD Fellow and a Hariri Graduate Fellow.

Stan Sclaroff is a Professor in the Boston University Department of Computer Science. He received his Ph.D. from MIT in 1995. His research interests are in computer vision, pattern recognition, and machine learning. He is a Fellow of the IAPR and Fellow of the IEEE.