Deep Supervised Hashing leveraging Quadratic Spherical Mutual Information for Content-based Image Retrieval

Nikolaos Passalis and Anastasios Tefas

Abstract

Several deep supervised hashing techniques have been proposed to allow for efficiently querying large image databases. However, deep supervised image hashing techniques are developed, to a great extent, heuristically often leading to suboptimal results. Contrary to this, we propose an efficient deep supervised hashing algorithm that optimizes the learned codes using an information-theoretic measure, the Quadratic Mutual Information (QMI). The proposed method is adapted to the needs of large-scale hashing and information retrieval leading to a novel information-theoretic measure, the Quadratic Spherical Mutual Information (QSMI). Apart from demonstrating the effectiveness of the proposed method under different scenarios and outperforming existing state-of-the-art image hashing techniques, this paper provides a structured way to model the process of information retrieval and develop novel methods adapted to the needs of each application.

1 Introduction

The vast amount of data available nowadays, combined with the need to efficiently provide quick answers to users’ queries led to the development of several hashing techniques [49, 17, 12, 45, 42, 48, 7, 23, 26]. Hashing provides a way to represent objects using compact codes, that allow for performing fast and efficient queries in large object databases. Early hashing methods, e.g., Locality Sensitive Hashing (LSH) [5], focused on extracting generic codes that could, in principle, describe every possible object and information need. However, it was later established that supervised hashing, that learns hash codes that are tailored to the task at hand, can significantly improve the retrieval precision. In this way, it is possible to learn even smaller hashing codes, since the extracted code must only encode the information needs for which the users are actually interested in. However, note that the extracted hash codes must also encode part of the semantic relationships between the encoded objects, to allow for providing a meaningful ranking of the retrieved results.

Many supervised and semi-supervised hashing methods have been proposed [15, 40, 16, 47, 50, 44, 37]. However, these methods were, to a great extent, heuristically developed, without a solid theory regarding the actual retrieval process. For example, many methods employ the pairwise distances between the images [15, 40, 50], or are based on sampling triplets that must satisfy specific relationships according to the given ground truth [16, 47], without a proper theoretical motivation for these choices. On the other hand, information-theoretic measures, such as entropy and mutual information [32], have been proven to provide robust solutions to many machine learning problems, e.g., classification [32]. However, very few steps towards using these measures for supervised hashing tasks have been made so far.

In this paper, we provide a connection between an information-theoretic measure, the Mutual Information (MI) [32], and the process of information retrieval. More specifically, we argue that mutual information can naturally model the process of information retrieval, providing a solid framework to develop retrieval-oriented supervised hashing techniques. Even though MI provides a sound theoretical
formulation for the problem of information retrieval, applying it in real scenarios is usually intractable, since there is no efficient way to calculate the actual probability densities, that are involved in the calculation of MI. The great amount of data as well as their high dimensionality further complicate the practical application of such measures.

The main contribution of this paper is the proposal of an efficient deep supervised hashing algorithm that optimizes the learned codes using a novel extension of an information-theoretic measure, the Quadratic Mutual Information (QMI) [36]. The architecture of the proposed method is shown in Fig. 1.

To derive a practical algorithm that can efficiently scale to large datasets:

1. We adapt QMI to the needs of supervised hashing by employing a similarity measure that is closer to the actual distance used for the retrieval process, i.e., the Hamming distance. This gives rise to the proposed Quadratic Spherical Mutual Information (QSMI). It is also experimentally demonstrated that the proposed QSMI is more robust compared to the classical Gaussian-based Kernel Density Estimation used in QMI [36], while it does not require carefully tuning of any hyper-parameters.

2. We propose using a more smooth optimization objective employing a novel square clamping approach. This allows for significantly improving the stability of the optimization, while reducing the risk of converging to bad local minima.

3. We adapt the proposed approach to work in batch-based setting by employing a method that dynamically estimates the prior probabilities, as they are observed within each batch. In this way, the proposed method can efficiently scale to larger datasets.

4. We demonstrate that the proposed method can be readily extended to efficiently handle different scenarios, e.g., retrieval of unseen classes [34].

The proposed method is extensively evaluated using three image datasets, including the two standard datasets used for evaluating supervised hashing methods, the CIFAR10 [14] and NUS-WIDE [4] datasets, and it is demonstrated that it outperforms the existing state-of-the-art techniques. Following the suggestions of [34], we also evaluate the proposed method in a different evaluation setup, where the learned hash codes are evaluated using unseen information needs. A PyTorch-based implementation of the proposed method is available at https://github.com/passalis/qsmi, allowing any researcher to easily use the proposed method and readily reproduce the experimental results.

The rest of the paper is structured as follows. The related work is discussed in Section 2. The proposed method is presented in detail in Section 3, while the experimental evaluation is provided in Section 4. Finally, Section 5 concludes the paper.

2 Related Work

The increasing interest for learning compact hash codes, together with the great learning capacity of recent deep learning models, led to the development of several deep supervised hashing techniques. Deep supervised hashing techniques involve: a) a deep neural network, that is used to extract a representation from the data, b) a supervised loss function, that is used to train the network, and c) a hashing mechanism, e.g., an appropriate non-linearity [18] or regularizer [50], that ensures that the output of the network can be readily transformed into a hash code. Most of the proposed methods fall into one of the following two categories according to the loss function employed for learning the supervised codes: a) pairwise-based hashing methods [15, 14, 50, 18, 21, 10, 19, 35] and b) triplet-based hashing methods [10, 47, 39].

1 The code will be available shortly after the review process.
Figure 1: Pipeline of the proposed method: A deep convolutional neural network (CNN) is used to extract a representation that can be used to directly obtain a binary hash code. The network is optimized using the proposed Quadratic Spherical Mutual Information loss that is adapted towards the needs of large-scale image hashing.

Pairwise-based methods work by learning hash codes that minimize / maximize the pairwise distance / log-likelihood between similar / dissimilar pairs, e.g., Convolutional Neural Network (CNN)-based hashing [40], network in network hashing [15], deep hashing network [50], deep pairwise-supervised hashing [19], deep hashing network [50], and deep supervised discrete hashing [15]. More advanced pairwise methods employ margins that allow for learning more regularized representations, e.g., deep supervised hashing [21], use asymmetric hashing schemes, e.g., deep asymmetric pairwise hashing [55], and asymmetric deep supervised hashing [10], or use more advanced techniques to obtain the binary codes, e.g., hashing by continuation [8]. Even though these techniques have been applied with great success, they were largely developed heuristically. On the other hand, the proposed method works by maximizing the mutual information between the learned hash codes and the ground truth.

Triplet-based methods work by sampling an anchor point along with a positive and a negative example [46, 47, 39, 6]. Then, they learn codes that increase the similarity between the anchor and the positive example, while reducing the similarity between the anchor and the negative example. However, triplet-based methods are significantly more computationally expensive than pairwise-based methods, requiring a huge number of triplets to be generated (many of which convey no information, since they are already satisfied by the code learned by the network), limiting their practical application. Also note that many non-deep supervised hashing methods have also been proposed, e.g., [11, 20, 22], but an extensive review of them is out of the scope of this paper. The interested reader is referred to [38] for an extensive literature review on hashing.

To the best of our knowledge, this is the first work that employs a quadratic spherical mutual information loss fully adapted to the needs of deep supervised hashing. Apart from deriving a practical algorithm and demonstrating its ability to outperform existing state-of-the-art methods, the proposed method provides a complete framework that can be used to model the process of information retrieval. This formulation is fully differentiable allowing for the end-to-end optimization of deep neural networks for any retrieval-related task, ranging from learning retrieval-oriented representations and compact
hash codes to fine-tuning the extracted representations using relevance feedback.

3 Proposed Method

The proposed method is presented in detail in this Section. First, the links between mutual information and information retrieval are provided. Then, the quadratic mutual information is introduced, the proposed quadratic spherical mutual information is derived and several aspects of the proposed method are discussed.

3.1 Information Retrieval and Mutual Information

Let \( Y = \{y_1, y_2, ..., y_N\} \) be a collection of \( N \) images, where \( y_i \in \mathbb{R}^n \) is the representation of the \( i \)-th image extracted using an appropriate feature extractor, e.g., a deep neural network. Each image \( y_i \) fulfills a set of information needs. For example, an image that depicts a “red car near a beach” fulfills at least the following information needs: “car”, “red car”, “beach”, “car near beach”. Note that the information needs that an image actually fulfills depend on both its content and the needs of the users, since, depending on the actual application, the interests of the users are usually focused on a specific area. For example, an image of a man entering a bank represents different information needs for a forensics database used by the police to identify suspects and for a generic web search engine. The problem of information retrieval can be then defined as follows: Given an information need \( q \) retrieve the images of the collection \( Y \) that fulfill this information need and rank them according to their relevance to the given information need. This work focuses on content-based information retrieval \cite{16}, where the information need \( q \) is expressed through a query image \( q \in \mathbb{R}^n \), that is usually not part of the collection \( Y \).

To be able to measure how well an information retrieval system works, a ground truth set that contains a set of information needs and the corresponding images that fulfill these information needs is usually employed. Let \( M \) be the number of information needs \( Q = \{q_1, q_2, ..., q_M\} \). Then, for each information need \( q_i \), a set of images \( Q_i = \{y_1^{(i)}, y_2^{(i)}, ..., y_N^{(i)}\} \), where \( y_j^{(i)} \in \mathbb{R}^n \) is the representation of the \( j \)-th image that fulfills the \( i \)-th information need, is given. Note that \( |Q_i| = N_i \). Since all these images fulfill the same information need, they can be all used as queries to express this information need. However, there are also other images, which are usually not known beforehand, that also express the same information need and they can be also used to query the database. The distribution of the images that fulfill the \( i \)-th information need can be modeled using the conditional probability density function \( p(y|q_i) \).

Let \( Y \) be a random vector that represents the images and \( Q \) be a random variable that represents the information needs. The Shannon’s entropy of the information needs, that expresses the uncertainty regarding the information need that a randomly sampled image fulfills, is defined as \( H(Q) \):

\[
H(Q) = -\sum q P(q) \log(P(q)),
\]

(1)

where \( P(q) \) is the prior probability of the information need \( q \), i.e., the probability that a random image of the collection fulfills the information need \( q \). Note that above definition implicitly assumes that the information needs are mutually exclusive, i.e., \( \sum q P(q) = 1 \), or equivalently, that each image satisfies only one information need. This is without loss of generality, since it is straightforward to extend this definition to the general case, where each image can satisfy multiple information needs, simply by measuring the entropy of each information need separately: \( H(Q) = -\sum q \left(P(q) \log(P(q)) + (1 - P(q)) \log(1 - P(q))\right) \).

To simplify the presentation of the proposed method, we assume that the information needs are mutually exclusive. Nonetheless, the proposed approach can be still used with minimal modifications, as we also experimentally demonstrate in Section \( \$4 \) even when this assumption does not hold.
When the query vector is known, then the uncertainty of the information need that it fulfills can be expressed by the conditional entropy:

$$H(\mathcal{D}|\mathcal{Y}) = -\int_{\mathcal{Y}} p(\mathcal{y}) \left( \sum_{\mathcal{q}} p(q|\mathcal{y}) \log(p(q|\mathcal{y})) \right) d\mathcal{y}. \tag{2}$$

Mutual information is defined as the amount by which the uncertainty for the information needs is reduced after observing the query vector:

$$I(\mathcal{D}, \mathcal{Y}) = H(\mathcal{D}) - H(\mathcal{D}|\mathcal{Y}) = \sum_{\mathcal{q}} \int_{\mathcal{Y}} p(\mathcal{q}, \mathcal{y}) \log \left( \frac{p(\mathcal{q}, \mathcal{y})}{p(\mathcal{q})p(\mathcal{y})} \right) d\mathcal{y}. \tag{3}$$

It is easy to see that MI can be interpreted as the Kullback-Leibler divergence between $p(\mathcal{q}, \mathcal{y})$ and $P(\mathcal{q})p(\mathcal{y})$. It is desired to maximize the MI between the representation of the images $\mathcal{Y}$ and the information needs $\mathcal{D}$, since this ensures that the uncertainty regarding the information need, that a query image expresses, is minimized. Also, note that MI models the intrinsic uncertainty regarding the query vectors, since it employs the conditional probability density between the information needs and the images, instead of just a limited collection of images.

On the other hand, it is usually intractable to directly calculate the required probability density $p(\mathcal{y}|q_i)$ and the corresponding integral in $(3)$, limiting the practical applications of MI. However, as it is demonstrated later, it is possible to efficiently estimate the aforementioned probability density and derive a practical algorithm that maximizes the MI between a representation and a set of information needs.

### 3.2 Quadratic Mutual Information

When the aim is not to calculate the exact value of MI, but to optimize a distribution that maximizes the MI, then a quadratic divergence metric, instead of the Kullback-Leibler divergence, can be used. In this way, the Quadratic Mutual Information (QMI) is defined as [33]:

$$I_T(\mathcal{D}, \mathcal{Y}) = \sum_{\mathcal{q}} \int_{\mathcal{Y}} (p(\mathcal{q}, \mathcal{y}) - P(\mathcal{q})p(\mathcal{y}))^2 d\mathcal{y}. \tag{4}$$

By expanding $(4)$, QMI can be expressed as the sum of three information potentials as $I_T(\mathcal{D}, \mathcal{Y}) = V_{IN}(\mathcal{D}, \mathcal{Y}) + V_{ALL}(\mathcal{D}, \mathcal{Y}) - 2V_{BTW}(\mathcal{D}, \mathcal{Y})$, where: $V_{IN}(\mathcal{D}, \mathcal{Y}) = \sum_{\mathcal{q}} \int_{\mathcal{Y}} p(\mathcal{q}, \mathcal{y})^2 d\mathcal{y}$, $V_{ALL}(\mathcal{D}, \mathcal{Y}) = \sum_{\mathcal{q}} \int_{\mathcal{Y}} p(\mathcal{q})^2 p(\mathcal{y})^2 d\mathcal{y}$, and $V_{BTW}(\mathcal{D}, \mathcal{Y}) = \sum_{\mathcal{q}} \int_{\mathcal{Y}} p(\mathcal{q}, \mathcal{y}) P(\mathcal{q})p(\mathcal{y}) d\mathcal{y}$.

To calculate these quantities, the probability $P(\mathcal{q})$ and the densities $p(\mathcal{y})$ and $p(\mathcal{q}, \mathcal{y})$ must be estimated. The prior probabilities depend only on the distribution of the information needs in the collection of images. Therefore, for the $i$-th information need: $P(q_i) = \frac{N_i}{N}$, where $N_i$ is the number of images that fulfill the $i$-th information need. The conditional density of the images that fulfill the $i$-th information need can be estimated using the Parzen window estimation method [27]:

$$p(\mathcal{y}|q_i) = \frac{1}{N_i} \sum_{j=1}^{N_i} K(\mathcal{y} - \mathcal{y}_j^{(i)}; \sigma^2), \tag{5}$$

where $K(\mathcal{y}; \sigma^2)$ is a Gaussian kernel (in an $n$-dimensional space) with width $\sigma$ defined as:

$$K(\mathcal{y}; \sigma) = \frac{1}{(2\pi)^{n/2}\sqrt{\sigma}} \exp(-\frac{\mathcal{y}^T\mathcal{y}}{2\sigma}). \tag{6}$$

Then, the joint probability density can be estimated as:

$$p(q_i, \mathcal{y}) = p(\mathcal{y}|q_i)p(q_i) = \frac{1}{N} \sum_{j=1}^{N_i} K(\mathcal{y} - \mathcal{y}_j^{(i)}; \sigma^2), \tag{7}$$
while the density of all the images as:

$$p(y) = \frac{1}{N} \sum_{j=1}^{N} K(y - y_j^{(i)}; \sigma^2).$$  \hspace{1cm} (8)$$

By substituting these estimations into the definitions of the information potentials, the following quantities are obtained:

$$V_{IN}(\mathcal{Q}, \mathcal{Y}) = \frac{1}{N^2} \sum_{k=1}^{M} \left( \frac{N_k}{N} \right)^2 \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} K(y_i^{(k)} - y_j^{(k)}, 2\sigma^2),$$  \hspace{1cm} (9)

$$V_{ALL}(\mathcal{Q}, \mathcal{Y}) = \frac{1}{N^2} \left( \sum_{k=1}^{M} \left( \frac{N_k}{N} \right)^2 \right) \sum_{i=1}^{N} \sum_{j=1}^{N} K(y_i - y_j, 2\sigma^2),$$  \hspace{1cm} (10)

and

$$V_{BTW}(\mathcal{Q}, \mathcal{Y}) = \frac{1}{N^2} \sum_{k=1}^{M} \left( \frac{N_k}{N} \right) \sum_{i=1}^{N} \sum_{j=1}^{N} K(y_i^{(k)} - y_j, 2\sigma^2),$$  \hspace{1cm} (11)

where the following property regarding the convolution between two Gaussian kernels was used: $$\int_{\mathbb{R}} K(y - y_i; \sigma^2) K(y - y_j; \sigma^2) dy = K(y_i - y_j, 2\sigma^2).$$ The information potential $$V_{IN}$$ expresses the interactions between the images that fulfill the same information need, the information potential $$V_{ALL}$$ the interactions between all the images of the collection, while the potential $$V_{BTW}$$ models the interactions of the images that fulfill a specific information need against all the other images. Therefore, the QMI formulation allows for the efficient calculation of MI, since the MI is expressed as a weighted sum over the pairwise interactions of the images of the collection.

Using Parzen window estimation with a Gaussian kernel for estimating the probability density leads to the implicit assumption that the similarity between two images is expressed through their Euclidean distance. Thus, the images that fulfill an information need expressed by a query vector $$q$$ can be retrieved simply using nearest-neighbor search.

### 3.3 Quadratic Spherical Mutual Information Optimization

Even though QMI allows for more efficient optimization of distributions, it suffers from several limitations: a) QMI involves the calculation of the pairwise similarity matrix between all the images of a collection. This quickly becomes intractable as the size of the collection increases. b) Selecting the appropriate width for the Gaussian kernels is not always straightforward, as a non-optimal choice can distort the feature space and slow down the optimization. c) The discrepancy between the distance metric used for QMI (Euclidean distance) and the distance used for the actual retrieval of the hashed images (Hamming distance) can negatively affect the retrieval accuracy. Finally, d) it was experimentally observed that directly optimizing the QMI is prone to bad local minima, due to the linear behavior of the loss function that fails to distinguish between the pairs of images that cause high error and those which have a smaller overall effect on the learned representation (more details are given later in this Section).

To overcome the limitations (b) and (c), we propose the Quadratic Spherical Mutual Information (QSMI). The proposed QSMI method replaces the Gaussian kernel in (6), used for calculating the similarity between two images in the information potentials in (9), (10), and (11), with the cosine similarity:

$$S_{\cos}(y_1, y_2) = \frac{1}{2} \frac{y_1^T y_2}{\|y_1\|_2 \|y_2\|_2} + 1,$$  \hspace{1cm} (12)

where $$\|\cdot\|_2$$ is the $$l^2$$ norm of a vector. In this way, we maintain the computationally efficient QMI formulation and avoid the need for manually tuning the width parameter of the Gaussian kernel, while
adopting a formulation that is closer to the Hamming distance that is actually used for the retrieval process. Indeed, the cosine similarity can be interpreted as a normalized Hamming-based similarity measure \([22]\). To understand this, consider that the Hamming “similarity” (number of bits that are equal between two binary vectors) can be calculated as \(y_1^T y_2\), for two binary \(n\)-dimensional vectors \(y_1, y_2 \in \{0, 1\}^n\). Therefore, when binary vectors are used, the cosine similarity can be interpreted as a bounded normalized version of the Hamming similarity.

Therefore, QSMI is defined as:

\[
I_T^{\cos}(\mathcal{D}, \mathcal{Y}) = V_{IN}^{\cos}(\mathcal{D}, \mathcal{Y}) + V_{ALL}^{\cos}(\mathcal{D}, \mathcal{Y}) - 2V_{BTW}^{\cos}(\mathcal{D}, \mathcal{Y}),
\]

where

\[
V_{IN}^{\cos}(\mathcal{D}, \mathcal{Y}) = \frac{1}{N^2} \sum_{k=1}^M \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} S_{\cos}(y_i^{(k)}, y_j^{(k)}),
\]

\[
V_{ALL}^{\cos}(\mathcal{D}, \mathcal{Y}) = \frac{1}{N^2} \left( \sum_{k=1}^M \left( \frac{N_k}{N} \right)^2 \right) \sum_{i=1}^N \sum_{j=1}^N S_{\cos}(y_i, y_j),
\]

and

\[
V_{BTW}^{\cos}(\mathcal{D}, \mathcal{Y}) = \frac{1}{N^2} \sum_{k=1}^M \left( \frac{N_k}{N} \right) \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} S_{\cos}(y_i^{(k)}, y_j^{(k)}).
\]

Note that when the information needs are equiprobable, i.e., \(P(q) = \frac{1}{M}\), then QSMI can be simplified as \(I_T^{\cos}(\mathcal{D}, \mathcal{Y}) = V_{IN}^{\cos}(\mathcal{D}, \mathcal{Y}) - V_{BTW}^{\cos}(\mathcal{D}, \mathcal{Y})\). Therefore, when this assumption holds, QSMI can be easily implemented just by defining the similarity matrix \(S \in \mathbb{R}^{N \times N}\), where \([S]_{ij} = S_{\cos}(y_i, y_j)\) and the notation \([S]_{ij}\) is used to refer to the \(i\)-th row and \(j\)-th column of matrix \(S\). Then, QSMI can be calculated as:

\[
I_T^{\cos} = \frac{1}{N^2} 1_N^T \left( \Delta \odot S - \frac{1}{M} S \right) 1_N,
\]

where the indicator matrix is defined as:

\[
[S]_{ij} = \begin{cases} 1, & \text{if the } i\text{-th and the } j\text{-th documents are similar} \\ 0, & \text{otherwise}. \end{cases}
\]

The notation \(1_N \in \mathbb{R}^N\) is used to refer an \(N\)-dimensional vector of 1s, while the operator \(\odot\) denotes the Hadamard product between two matrices. Please refer to the Appendix A for a more detailed derivation. This formulation also allows for directly handling information needs that are not mutually exclusive. In that case, the values of the indicator matrix are appropriately set to 1, if two images share at least one information need.

Instead of directly optimizing the QSMI, we propose using a “square clamp” around the similarity matrix \(S\), smoothing the optimization surface. Therefore, given that the values of \(S\) range in the unit interval, the loss function is re-derived as:

\[
L_{QSMI} = \frac{1}{N^2} 1_N^T \left( \Delta \odot (S - 1) \odot (S - 1) + \frac{1}{M} (S \odot S) \right) 1_N.
\]

As shown in Figure \(23\) this formulation penalizes the pairs with larger error more heavily than those with smaller error, allowing for discovering more robust solutions. This modification effectively addresses the limitation (d), as we also experimentally demonstrate in the ablation study given in Section 4.

To allow for scaling to larger datasets, batch-based optimization is used. That allows for reducing the complexity of QMI from \(O(N^2)\), which is intractable for larger datasets, to just \(O(N_B^2)\), where \(N_B\) is the used batch size that typically ranges from 64 to 256. However, this implies that each
batch will contain images only from a subsample of the available information needs. This in turn means that the observed in-batch prior probability \( P(q) \) will not match the collection-level prior, leading to underestimating the influence of the potential \( V_{\text{ALL}} \) to the optimization. To account for this discrepancy, we propose a simple heuristic to estimate the in-batch prior, i.e., the value of \( M \) in (27). \( M \) is estimated as:

\[
M = \frac{N_B^2}{(1_T N_B \Delta N_B)},
\]

where \( N_B \) is the batch size. To understand the motivation behind this, consider that if the whole collection was used for the optimization, then the number of 1s in \( \Delta \) would be:

\[
M(N_M)^2 = 1_T N_B \Delta 1_N,
\]

Solving this equation for \( M \) yields the value used for approximating \( M \). Note that the value of \( M \) is not constant and depends on the distribution of the samples in each batch. It was experimentally verified that this approach indeed improves the performance of the proposed method over using a constant value for \( M \).

### 3.4 Deep Supervised Hashing using QSMI

The proposed QSMI is used to train a deep neural network to extract short binary hash codes, as shown in Fig. 1. Let \( x \) be the raw representation of an image (e.g., the pixels of an image) and let

\[
y = f_W(x) \in \mathbb{R}^n
\]

be the output of a neural network \( f_W(\cdot) \), where \( W \) denotes the matrix of the parameters of the network and \( n \) is the length of the hash code. Apart from learning a representation that minimizes the \( J_{\text{QSMI}} \) loss, the network must generate an output that can be easily translated into a binary hash code. Several techniques have been proposed to this end, e.g., using the \( \tanh \) function [38]. In this work, the output of the network is required to be close to two possible values, either 1 or -1. Therefore, the used hashing regularizer is defined, following the recent deep supervised hashing approaches [21], as:

\[
\mathcal{L}_{\text{hash}} = \sum_{i=1}^{N} |||y_i| - 1_n||_1,
\]

where \( \cdot \) denotes the absolute value operator and \( ||\cdot||_1 \) denotes the \( l^1 \) norm. The final loss function is defined as:

\[
\mathcal{L} = \mathcal{L}_{\text{QSMI}} + \alpha \mathcal{L}_{\text{hash}},
\]

where \( \alpha \) is the weight of the hashing regularizer. The network \( f_W(\cdot) \) can be then trained using gradient descent, i.e., \( \Delta W = -\eta \frac{\partial J}{\partial W} \), where \( \eta \) is the used learning rate. Please refer to Appendix A on details regarding the derivation of \( \frac{\partial J}{\partial W} \). After training the network, the hash codes can be readily obtained using the \( \text{sign}(y) \) function.

Even though learning highly discriminative hash codes is desired for retrieving data that belong to the training domain, it can negatively affect the retrieval for previously unseen information needs [34]. The proposed method can be also easily modified to optimize the hash codes toward other unsupervised information needs. Even though any source of information can be used, in this work the information needs are discovered by clustering the training data. This allows for discovering information needs dictated by the structure of the data. Let \( \mathcal{L}_{\text{QSMI} + U} \) denote the loss induced by applying the QMSI loss function on these information needs. Then, the final loss function for this semi-supervised variant is defined as:

\[
\mathcal{L} = \mathcal{L}_{\text{QSMI} + U} + \alpha \mathcal{L}_{\text{hash}} + \beta \mathcal{L}_{\text{QSMI} + U}.
\]

This variant is used for the experiments conducted in Section 4.4.

### 4 Experimental Evaluation

The proposed method is extensively evaluated in this Section, using both an ablation study and comparing it to other state-of-the-art methods. First, the used datasets and the employed evaluation setup are briefly described. The hyper-parameters and the network architectures used for the evaluation are provided in Appendix B. Finally, an ablation study is provided and the proposed method is evaluated using three different datasets.
Table 1: Ablation study using the Fashion MNIST dataset (the mAP is reported)

| Clamped | Spherical | mAP       | precision (< 2bits) |
|---------|-----------|-----------|---------------------|
| No      | No        | 0.727 ± 0.008 | 0.674 ± 0.021       |
| Yes     | No        | 0.816 ± 0.009 | 0.864 ± 0.010       |
| Yes     | Yes       | **0.861 ± 0.004** | **0.876 ± 0.004**   |

4.1 Datasets and Evaluation Metrics

Three image datasets are used to evaluate the proposed method in this paper: The Fashion MNIST dataset, the CIFAR10 dataset and the NUS-WIDE dataset. All images were preprocessed to have zero mean and unit variance, according to the statistics of the dataset used for the training.

The Fashion MNIST dataset is composed of 60,000 training images and 10,000 test images [41]. The size of each image is 28 × 28 pixels (gray-scale images) and there is a total of 10 different classes (each one expresses a different information need). The whole training set was used to train the networks and build the database, while the test set was used to query the database and evaluate the performance of the methods.

The CIFAR10 dataset is composed of 50,000 training images and 10,000 test images [14]. The size of each image is 32 × 32 pixels (color images) and there is a total of 10 different classes (information needs). Again, the whole training set was used to train the networks and build the database, while the test set was used to query the database and evaluate the performance of the methods.

The NUS-WIDE is a large-scale dataset that contains 269,648 images that belong to 81 different concepts [4]. The images were resized to 224 × 224 pixels before feeding them to the network. Following [19], only images that belong to the 21 most frequent concepts, i.e., 195,834 images, were used for training/evaluating the methods. Each image might belong to multiple different concepts, i.e., the information needs are not mutually exclusive. For evaluating the methods, two images were considered relevant if they share at least one common concept, which is the standard protocol used for this dataset [19]. Similarly to the other two datasets, the whole training set (193,734 randomly sampled images) was used to train the networks and build the database, while 2,100 randomly sampled queries (100 from each category) were employed to evaluate the methods.

To evaluate the proposed methods, the following four metrics were used: precision, recall, mean average precision (mAP), and precision within hamming radius of 2. Nearest neighbor search using the Hamming distance was used to retrieve the relevant documents [24]. Following [24], precision is defined as $Pr(q,k) = \frac{rel(q,k)}{k}$, where $k$ is the number of retrieved objects and $rel(q,k)$ is the number of retrieved objects that fulfill the same information need as the query $q$, while recall is defined as $Rec(q,k) = \frac{rel(q,k)}{n_{total}(q)}$, where $n_{total}(q)$ is the total number of database objects that fulfill the same information need as $q$. The average precision (AP) is calculated at eleven equally spaced recall points (0, 0.1, ..., 0.9, 1) and the mean average precision is calculated as the mean of the APs for all queries. The precision within hamming radius of 2 is defined as: $Pr_{H2}(q) = \frac{rel(q,k)}{n_{total}(q)}$, where $rel_{H2}(q)$ is the number of relevant documents withing hamming distance 2 from the query, while $total_{H2}(q)$ is the total number of documents withing hamming distance 2 from the query.

4.2 Ablation Study

A smaller dataset, the Fashion MNIST dataset [41], was used to perform an ablation study. The effect of various design choices, i.e., using or not the proposed clamped loss and spherical formulation, is evaluated in Table 1. The Mean Average Precision (mAP) is averaged over 5 runs, while the code length was set to 48 bits for these experiments. Several conclusions can be drawn from the results reported in Table 1. First, employing the proposed clamped loss, instead of directly optimizing the QMI ($\sigma = 10$), improves the hashing precision, confirming our hypothesis regarding the benefits of
using the proposed clamped loss (as also described in the previous Section and shown in Fig. 2a). This is also confirmed in the learning curve shown in Fig. 2b, where both the proposed clamped loss and MI are monitored during the optimization. Optimizing the proposed clamped loss is directly correlated with the QMI and the proposed QSMI, both of which steadily increase during the 50 training epochs. When the spherical formulation is used (QSMI method), then the mAP further increase to 86.1% from 72.7% (standard QMI formulation). The effect of the regularization parameter $\alpha$ is also examined in Fig. 2c. The best performance is obtained for $\alpha = 0.01$.

The proposed method was compared to two other state-of-the-art techniques, the Deep Supervised Hashing (DSH) method [21] and the Deep Pairwise Supervised Hashing (DPSH) method [19]. We carefully implemented these methods in a batch-based setting and we tuned their hyper-parameters to obtain the best performance (please refer to Appendix B). The evaluation results are shown in Table 2. The proposed method is abbreviated as “QSMIH” and significantly outperforms the other two state-of-the-art pairwise hashing techniques, highlighting the importance of using theoretically-sound objectives for learning deep supervised hash codes. Recall that a deep CNN, that was trained from scratch, was employed for the conducted experiments. The precision within Hamming radius of 2 is also shown in Figure 3a. Again, the proposed method outperforms all the other methods for all the evaluated hash code lengths.

4.3 Supervised Hashing Evaluation

The proposed method was also evaluated using the CIFAR10 [14] and NUS-WIDE [4] datasets. For the CIFAR10, a DenseNet [9], pretrained on the same dataset, was employed, while for the NUS-WIDE dataset, a DenseNet pretrained on the ImageNet dataset was used [33].

The evaluation results for the CIFAR10 dataset are reported in Table 3. The proposed method is
Table 3: CIFAR10 Evaluation (the mAP for different hash code lengths is reported)

| Method    | 8 bits | 12 bits | 24 bits | 36 bits | 48 bits |
|-----------|--------|---------|---------|---------|---------|
| MIHash*   | N/A    | 0.929   | 0.933   | 0.938***| 0.942   |
| DSH**     | 0.936  | 0.958   | 0.967   | 0.970   | 0.970   |
| DPSH**    | 0.776  | 0.933   | 0.971   | 0.971   | 0.971   |
| QSMIH     | 0.962  | 0.970   | 0.971   | 0.971   | 0.971   |

*Results as reported in [2] (using a slightly different setup), **Results using our implementation of DSH [21] and DPSH [19], ***Results for 32 bits (as reported in [2]).

The proposed method outperforms all the other techniques by a large margin for small code lengths, i.e., 8 and 12 bits. For larger hash codes, the proposed method performs equally well with the DSH and DPSH methods. However, the proposed method is capable of achieving almost the same performance as the DSH and DPSH methods using less than half the bits, highlighting the expressive power of the proposed technique. Also, to the best of our knowledge, this is the best result reported in the literature for the CIFAR10 dataset (regardless the employed hashing technique). The Hamming precision within radius 2 is shown in Figure 3b. Again, the proposed method performs significantly better for smaller code lengths, i.e., 8 and 12 bits, while matching the precision of the DPSH method for hash codes larger than 24 bits.

The proposed QSMIH method was also evaluated using the larger-scale NUS-WIDE dataset that contains 269,648 images that belong to 81 different concepts. Following [19], we used the images that belong to the 21 most frequent concepts, i.e., 195,834 images. Note that an image might belong to more than one concept, i.e., fulfill multiple information needs. Furthermore, instead of using a subsample of the training set, we used the whole training set (193,734 randomly sampled images) to learn the hash codes (all the methods were used in a batch setting), and a test set of 2,100 randomly sampled queries was employed to evaluate the methods. Since there are many differences in the evaluation protocol used by different papers for this dataset, we compared the proposed method to the DSH and DPSH methods using the same network and evaluation setup. The evaluation results are shown in Table 4. Again, the proposed method outperforms the rest of the evaluated methods for any code length. The same behavior is also observed in the precision results illustrated in Fig. 3c. Note that a similar behavior for the DSH method, i.e., the precision is reduced as the code length increases after a certain point, is also reported by the authors of the DSH method [21].

Figure 3: Precision within hamming radius 2 for three different datasets
Table 4: NUS-WIDE Evaluation (the mAP for different hash code lengths is reported)

| Method | 8 bits | 12 bits | 24 bits | 36 bits | 48 bits |
|--------|--------|---------|---------|---------|---------|
| DSH    | 0.660  | 0.659   | 0.671   | 0.689   | 0.694   |
| DPSH   | 0.735  | 0.748   | 0.759   | 0.758   | 0.755   |
| QSMIH  | 0.746  | 0.753   | 0.766   | 0.764   | 0.763   |

Table 5: CIFAR10 Evaluation - Retrieval of Unseen Information Needs (the mAP for different hash code lengths is reported)

| Method   | 12 bits     | 24 bits     | 36 bits     |
|----------|-------------|-------------|-------------|
| DSH      | 0.615 ± 0.065| 0.689 ± 0.063| 0.674 ± 0.055|
| DPSH     | 0.568 ± 0.082| 0.635 ± 0.078| 0.658 ± 0.048|
| QSMIH+U  | 0.691 ± 0.093| 0.795 ± 0.061| 0.682 ± 0.144|

4.4 Retrieval of Unseen Information Needs

Finally, the proposed method was evaluated using the evaluation setup proposed in [34], i.e., the 75% of the classes were used to train the models and the rest 25% were used to evaluate the models. The process was repeated 5 times using different class/information needs splits and the mean and standard deviation are reported. The evaluation results are shown in Table 5. The proposed method also employed 5 unsupervised information needs (discovered using the k-means algorithm on the 75% of the training data). The weight of the unsupervised loss in the optimization was set to $\beta = 0.5$. The proposed variant, denoted by “QSMIH+U” leads to significantly more regularized representations, that do not collapse outside the training domain, increasing the mAP for unseen classes from 0.689 to 0.795, demonstrating the flexibility of the proposed approach as well as its effectiveness in this setup.

5 Conclusions

A deep supervised hashing algorithm, adapted to the needs of large-scale hashing, that optimizes the learned codes using an novel information-theoretic measure, the Quadratic Spherical Mutual Information, was proposed. The proposed method was evaluated using three different datasets and evaluation setups and compared to other state-of-the-art supervised hashing techniques. The proposed method outperformed all the other evaluated methods regardless the size of the used dataset and training setup, exhibiting a significantly more stable behavior than the rest of the evaluated methods. More specifically, when used with a randomly initialized network, the proposed QSMIH method managed to outperform the rest of the methods by a large margin. On the other hand, when combined with powerful pre-trained networks, again it yielded the best results regardless the length of the used hash code. Also, the proposed method provides theoretical justification for several existing deep supervised hashing techniques, while also paves the way for developing more advanced representation learning techniques for information retrieval using the proposed information-theoretic formulation, e.g., handling cross-modal retrieval tasks [43].

Appendix A - Implementation Details

To simplify the implementation of the proposed method, we assume that all the information needs are equiprobable: $P(q) = \frac{1}{M}$. Then, the information potentials $V_{ALL}^{\cos}(\mathcal{D}, \mathcal{Y})$ and $V_{BTW}^{\cos}(\mathcal{D}, \mathcal{Y})$ can be
calculated as:

\[ V_{\text{ALL}}^{\cos}(\mathcal{Q}, \mathcal{Y}) = \frac{1}{N^2} \left( \sum_{k=1}^{M} \left( \frac{N_k}{N} \right)^2 \right) \sum_{i=1}^{N} \sum_{j=1}^{N} S_{\cos}(y_i, y_j) \]
\[ = \frac{1}{N^2} \left( \sum_{k=1}^{M} \left( \frac{1}{M} \right)^2 \right) \sum_{i=1}^{N} \sum_{j=1}^{N} S_{\cos}(y_i, y_j) = \frac{1}{N^2} \frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{N} S_{\cos}(y_i, y_j), \]  

(22)

and

\[ V_{\text{BTW}}^{\cos}(\mathcal{Q}, \mathcal{Y}) = \frac{1}{N^2} \sum_{k=1}^{M} \left( \frac{N_k}{N} \right) \sum_{i=1}^{N} \sum_{j=1}^{N} S_{\cos}(y_i^{(k)}, y_j) \]
\[ = \frac{1}{N^2} \frac{1}{M} \sum_{k=1}^{M} \sum_{i=1}^{N} \sum_{j=1}^{N} S_{\cos}(y_i^{(k)}, y_j) \]
\[ = \frac{1}{N^2} \frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{N} S_{\cos}(y_i, y_j) = V_{\text{ALL}}^{\cos}(\mathcal{Q}, \mathcal{Y}), \]  

(23)

where we assumed that \( N_k = \frac{N}{M} \), since \( P(q) = \frac{1}{M} \). Also, the \( V_{\text{IN}}^{\cos}(\mathcal{Q}, \mathcal{Y}) \) information potential can be expressed using the indicator matrix \( \Delta \):

\[ V_{\text{IN}}^{\cos}(\mathcal{Q}, \mathcal{Y}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} |\Delta|_{ij} S_{\cos}(y_i, y_j). \]  

(24)

Therefore, the QSMI can be simplified as:

\[ I_T^{\cos}(\mathcal{Q}, \mathcal{Y}) = V_{\text{IN}}^{\cos}(\mathcal{Q}, \mathcal{Y}) - V_{\text{BTW}}^{\cos}(\mathcal{Q}, \mathcal{Y}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left( |\Delta|_{ij} S_{\cos}(y_i, y_j) - \frac{1}{M} S_{\cos}(y_i, y_j) \right) \]  

(25)

Finally, using the proposed clamping method, the final loss function is obtained as:

\[ \mathcal{L}_{\text{QSMI}} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left( |\Delta|_{ij} \left( S_{\cos}(y_i, y_j) - 1 \right) \right)^2 + \frac{1}{M} \left( S_{\cos}(y_i, y_j) \right)^2 \]  

(26)

since we aim to maximize \( I_T^{\cos}(\mathcal{Q}, \mathcal{Y}) \). Also, note that \( (25) \) can be equivalently expressed as:

\[ \mathcal{L}_{\text{QSMI}} = \frac{1}{N^2} 1_N \left( \Delta \odot (S - 1) \odot (S - 1) + \frac{1}{M} (S \odot S) \right) 1_N \]  

(27)

allowing for efficiently implementing the proposed method.

To implement the gradient descent algorithm, the derivative \( \frac{\partial \mathcal{L}_{\text{QSMI}}}{\partial \mathbf{W}} \), where \( \mathbf{W} \) are the parameters of the employed neural network, must be calculated. This derivative is calculated as:

\[ \frac{\partial \mathcal{L}_{\text{QSMI}}}{\partial \mathbf{W}} = \sum_{i=1}^{N} \left( \frac{\partial \mathcal{L}_{\text{QSMI}}}{\partial y_i} \right)^T \frac{\partial y_i}{\partial \mathbf{W}}, \]  

(28)

where \( y_i \) is the output of the used neural network for the \( i \)-th document. The derivative \( \frac{\partial \mathbf{W}}{\partial \mathbf{W}} \) depends on the employed architecture, while the derivative of the proposed loss with respect to the hash code \( \frac{\partial \mathcal{L}_{\text{QSMI}}}{\partial y_i} \) can be calculated as:

\[ \frac{\partial \mathcal{L}_{\text{QSMI}}}{\partial y_i} = \frac{1}{N^2} \sum_{j=1, i \neq j}^{N} \left( |\Delta|_{ij} \frac{\partial (S_{\cos}(y_i, y_j) - 1)}{\partial y_i} - \frac{1}{M} \frac{\partial (S_{\cos}(y_i, y_j))}{\partial y_i} \right), \]  

(29)
Table 6: Parameters used for the conducted experiments

| Parameter | Method | F. MNIST | CIFAR10 | NUS-WIDE |
|-----------|--------|----------|---------|----------|
| Learning rate | all | 0.001 | 0.001 | 0.001 |
| Batch size | all | 128 | 128 | 128 |
| Epochs | all | 50 | 5 | 50 |
| $\alpha$ | DSH | $10^{-5}$ | $10^{-5}$ | $10^{-5}$ |
| $\alpha$ | QSMIH | $10^{-2}$ | $10^{-2}$ | $10^{-1}$ |
| $\eta_{DPSH}$ | DPSH | 5 | 3 | 5 |

(3 for 36-48 bits)

where

$$\frac{\partial (S_{\cos}(y_i, y_j) - 1)^2}{\partial y_i} = 2 (S_{\cos}(y_i, y_j) - 1) \frac{\partial S_{\cos}(y_i, y_j)}{\partial y_i}$$

(30)

and

$$\frac{\partial (S_{\cos}(y_i, y_j))^2}{\partial y_i} = 2 S_{\cos}(y_i, y_j) \frac{\partial S_{\cos}(y_i, y_j)}{\partial y_i}$$

(31)

Finally, the derivative of the cosine similarity, needed for calculating (30) and (31), can be computed as:

$$\frac{\partial S_{\cos}(y_i, y_j)}{\partial y_i} = \frac{1}{2} \left( \frac{|y_j||y_j|_2 - y_i^T y_j}{|y_i||y_j|_2} \right).$$

(32)

The loss and the corresponding derivatives can be similarly calculated when the information needs are not equiprobable.

Appendix B - Hyper-parameters and Network Architectures

The selected hyper-parameters are shown in Table 6. We selected the best parameters for the two other evaluated methods, i.e., DSH and DPSH, by performing line search for each parameter. The Adam optimizer [13], with the default hyper-parameters, was used for the optimization. The experiments were repeated 5 times and the mean value of each of the evaluated metrics is reported, except otherwise stated.

For the experiments conducted on the Fashion MNIST dataset a relatively simple Convolutional Neural Network (CNN) architecture was employed, as shown in Table 7. The network was initialized using the default PyTorch initialization scheme [31], and it was trained from scratch for all the conducted experiments.

For the CIFAR10 dataset, a DenseNet-BC-190 (growth rate 40 and compression rate 2) [9], that was pretrained on the CIFAR dataset, was used. For the NUS-WIDE dataset, a DenseNet-201 (growth rate 32 and compression rate 2), that was pretrained on the Imagenet dataset [33], was also employed. The feature representation was extracted from the last average pooling layers of the networks. Then, two fully connected layers were used: one with $N_H$ neurons and rectifier activation functions [25], and one with as many neurons as the desired code length (no activation function was used for the output layer). The size of hidden layer was set to $N_H = 64$ for the CIFAR10 dataset and to $N_H = 2048$ for the NUS-WIDE dataset. To speedup the training process, we back-propagated the gradients only to the last two layers of the network, which were trained to perform supervised hashing.
Table 7: Network architecture used for the Fashion MNIST dataset

| Layer         | Kernel Size | Filters / Neurons | Activation |
|---------------|-------------|-------------------|------------|
| Convolution   | 5 × 5       | 32                | ReLU [25]  |
| Max Pooling   | 2 × 2       | -                 | -          |
| Convolution   | 5 × 5       | 64                | ReLU [25]  |
| Max Pooling   | 2 × 2       | -                 | -          |
| Dense         | -           | # bits            | -          |

References

[1] Mohannad Almasri, Catherine Berrut, and Jean-Pierre Chevallet. A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information. In Proceedings of the European Conference on Information Retrieval, pages 709–715, 2016.

[2] Fatih Cakir, Kun He, Sarah Adel Bargal, and Stan Sclaroff. Mihash: Online hashing with mutual information. In Proceedings of the IEEE International Conference on Computer Vision, 2017.

[3] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. Hashnet: Deep learning to hash by continuation. In Proceedings of the IEEE International Conference on Computer Vision, 2017.

[4] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao Zheng. NUS-WIDE: A real-world web image database from national university of singapore. In Proceedings of the of ACM Conference on Image and Video Retrieval, 2009.

[5] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In Proceedings of the Annual Symposium on Computational Geometry, pages 253–262, 2004.

[6] Cheng Deng, Zhaojia Chen, Xianglong Liu, Xinbo Gao, and Dacheng Tao. Triplet-based deep hashing network for cross-modal retrieval. IEEE Transactions on Image Processing, 27(8):3893–3903, 2018.

[7] Guiguang Ding, Jile Zhou, Yuchen Guo, Zijia Lin, Sicheng Zhao, and Jungong Han. Large-scale image retrieval with sparse embedded hashing. Neurocomputing, 257:24–36, 2017.

[8] Jiani Hu, Weihong Deng, and Jun Guo. Improving retrieval performance by global analysis. In Proceedings of the International Conference on Pattern Recognition, volume 2, pages 703–706, 2006.

[9] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4700–4708, 2017.

[10] Qing-Yuan Jiang and Wu-Jun Li. Asymmetric deep supervised hashing. In Proceedings of the International Joint Conference on Artificial Intelligence, 2018.

[11] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou. Column sampling based discrete supervised hashing. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 1230–1236, 2016.

[12] Vandana D Kaushik, J Umarani, Amit K Gupta, Aman K Gupta, and Phalgunti Gupta. An efficient indexing scheme for face database using modified geometric hashing. Neurocomputing, 116:208–221, 2013.
[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), 2015.

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical Report, 2009.

[15] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[16] Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. ACM Transactions on Multimedia Computing, Communications, and Applications, 2(1):1–19, 2006.

[17] Peng Li, Jian Cheng, and Hanqing Lu. Hashing with dual complementary projection learning for fast image retrieval. Neurocomputing, 120:83–89, 2013.

[18] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. In Proceedings of the Advances in Neural Information Processing Systems, pages 2479–2488, 2017.

[19] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pages 1711–1717, 2016.

[20] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton Van den Hengel, and David Suter. Fast supervised hashing with decision trees for high-dimensional data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1971–1978, 2016.

[21] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2064–2072, 2016.

[22] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2074–2081, 2012.

[23] Lei Ma, Hongliang Li, Fanman Meng, Qingbo Wu, and King Ng Ngan. Global and local semantics-preserving based deep hashing for cross-modal retrieval. Neurocomputing, 2018.

[24] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. Introduction to information retrieval, volume 1. Cambridge University Press, 2008.

[25] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of the International Conference on Machine Learning, pages 807–814, 2010.

[26] Shubham Pachori, Ameya Deshpande, and Shanmuganathan Raman. Hashing in the zero shot framework with domain adaptation. Neurocomputing, 275:2137–2149, 2018.

[27] Emanuel Parzen. On estimation of a probability density function and mode. The annals of mathematical statistics, 33(3):1065–1076, 1962.

[28] Nikolaos Passalis and Anastasios Tefas. Entropy optimized feature-based bag-of-words representation for information retrieval. IEEE Transactions on Knowledge & Data Engineering, (7):1664–1677, 2016.

[29] Nikolaos Passalis and Anastasios Tefas. Learning neural bag-of-features for large-scale image retrieval. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(10):2641–2652, 2017.
[30] Nikolaos Passalis and Anastasios Tefas. Learning bag-of-embedded-words representations for textual information retrieval. *Pattern Recognition*, 81:254–267, 2018.

[31] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[32] Jose C Principe. *Information theoretic learning: Renyi’s entropy and kernel perspectives*. Springer Science & Business Media, 2010.

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[34] Alexandre Sablayrolles, Matthijs Douze, Nicolas Usunier, and Hervé Jégou. How should we evaluate supervised hashing? In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1732–1736, 2017.

[35] Fumin Shen, Xin Gao, Li Liu, Yang Yang, and Heng Tao Shen. Deep asymmetric pairwise hashing. In *Proceedings of the ACM on Multimedia Conference*, pages 1522–1530, 2017.

[36] Kari Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, 2003.

[37] Di Wang, Xinbo Gao, and Xiumei Wang. Semi-supervised constraints preserving hashing. *Neurocomputing*, 167:230–242, 2015.

[38] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):769–790, 2018.

[39] Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep supervised hashing with triplet labels. In *Proceedings of the Asian Conference on Computer Vision*, pages 70–84, 2016.

[40] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2156–2162, 2014.

[41] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[42] Xing Xu, Li He, Atsushi Shimada, Rin-ichiro Taniguchi, and Huimin Lu. Learning unified binary codes for cross-modal retrieval via latent semantic hashing. *Neurocomputing*, 213:191–203, 2016.

[43] Xing Xu, Fumin Shen, Yang Yang, Heng Tao Shen, and Xuelong Li. Learning discriminative binary codes for large-scale cross-modal retrieval. *IEEE Transactions on Image Processing*, 26(5):2494–2507, 2017.

[44] Chengwei Yao, Jiajun Bu, Chenxia Wu, and Gencai Chen. Semi-supervised spectral hashing for fast similarity search. *Neurocomputing*, 101:52–58, 2013.

[45] Tao Yao, Xiangwei Kong, Haiyan Fu, and Qi Tian. Semantic consistency hashing for cross-modal retrieval. *Neurocomputing*, 193:250–259, 2016.

[46] Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, and Lei Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Transactions on Image Processing*, 24(12):4766–4779, 2015.
[47] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1556–1564. IEEE, 2015.

[48] Wanqing Zhao, Hangzai Luo, Jinye Peng, and Jianping Fan. Spatial pyramid deep hashing for large-scale image retrieval. Neurocomputing, 243:166–173, 2017.

[49] Liang Zheng, Shengjin Wang, and Qi Tian. Coupled binary embedding for large-scale image retrieval. IEEE Transactions on Image Processing, 23(8):3368–3380, 2014.

[50] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In Proceedings of the International Joint Conference on Artificial Intelligence, pages 2415–2421, 2016.