Deep supervised hashing using quadratic spherical mutual information for efficient image retrieval

Nikolaos Passalis, Anastasios Tefas

Deep Supervised Hashing using Quadratic Spherical Mutual Information for Efficient Image Retrieval

Nikolaos Passalis, Anastasios Tefas

Department of Informatics, Aristotle University of Thessaloniki
Thessaloniki 54124, Greece Tel,Fax: +30-2310996304

Abstract

Several deep supervised hashing techniques have been proposed to allow for extracting compact and efficient neural network representations for various tasks. However, many deep supervised hashing techniques ignore several information-theoretic aspects of the process of information retrieval, often leading to sub-optimal results. In this paper, we propose an efficient deep supervised hashing algorithm that optimizes the learned compact codes using an information-theoretic measure, the Quadratic Mutual Information (QMI). The proposed method is adapted to the needs of efficient image 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 different applications.

Keywords: Deep Supervised Hashing, Compact Representations, Quadratic Mutual Information, Image Hashing

1. Introduction

The vast amount of data available nowadays, combined with the need to efficiently and promptly provide answers to users' queries led to the development of several *hashing* techniques [1, 2]. Hashing provides a way to represent objects using compact codes, that allow for performing fast and efficient queries in large object databases, lowering the computational requirements and accelerating many information retrieval-based applications. The increasing need to perform inference on the edge using embedded devices with limited processing power [3, 4], along with the vast amount of multimedia data collected from various sensors [5, 6], further stress the need for developing efficient hashing methods.

Early hashing methods, e.g., Locality Sensitive Hashing (LSH) [7], 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 compact hash codes that are tailored to the task at hand, can significantly improve the retrieval precision, as well as reduce the size of the extracted codes. 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 [8, 9, 10]. However, many deep supervised hashing techniques ignore several information-theoretic aspects of the process of information retrieval, often leading to sub-optimal results. For example, many methods employ the pairwise distances between the images [8, 9, 11], or are based on sampling *triplets* that must satisfy specific relationships according to the given ground truth [10, 12]. On the other hand, *information-theoretic* measures, such as mutual information [13], have been proven to provide robust solutions to many machine learning problems, e.g., classification [13]. 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) [13], 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 efficient retrieval-oriented supervised hashing techniques. Even though MI provides a well-defined 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

Email addresses: passalis@csd.auth.gr (Nikolaos Passalis), tefas@csd.auth.gr (Anastasios Tefas)

efficient deep supervised hashing algorithm that is capable of extracting short and efficient codes using a novel extension of an information-theoretic measure, the Quadratic Mutual Information (QMI) [14]. 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 close 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 [14], while it does not require careful 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 [15].

The proposed method is extensively evaluated using five image datasets, including two standard datasets used for evaluating supervised hashing methods, the CIFAR10 [16] and NUS-WIDE [17] datasets, and it is demonstrated that it outperforms several existing approaches. Following the suggestions of [15], we also evaluate the proposed method in a different evaluation setup, where the learned hash codes are evaluated using unseen information needs.

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 [11, 18], along with semi-supervised approaches [19, 20] and sophisticated unsupervised ones [21, 22]. Deep supervised hashing techniques involve: a) a deep neural network, that is used to extract a representation from the data, b) a (semi)-supervised loss function, that is used to train the network, and c) a hashing mechanism, e.g., an appropriate non-linearity [18] or regularizer [11], that ensures that the output of the network can be readily transformed into a compact 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 [2, 8, 9, 11, 18, 23, 24, 25] and b) triplet-based hashing methods [10, 12, 26].

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 [9], network in network hashing [8], deep hashing network [11], deep pairwise-supervised hashing [24], deep hashing network [11], and deep supervised discrete hashing [18]. More advanced pairwise methods employ margins that allow for learning more regularized representations, e.g., deep supervised hashing [2], use asymmetric hashing schemes, e.g., deep asymmetric pairwise hashing [25], asymmetric deep supervised hashing [23], discriminative deep metric learning for asymmetric discrete hashing [27], or use more advanced techniques to obtain the binary codes, e.g., hashing by continuation [28], or focus on cross-modal retrieval [29].

Triplet-based methods work by sampling an anchor point along with a positive and a negative example [10, 12, 26, 30]. 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., [31, 32, 33], but an extensive review of them is out of the scope of this paper. The interested reader is referred to [34] for an extensive literature review on hashing.

More recent works on deep supervised hashing employ objectives based on class-wise loss [35], semantic cluster-based unary loss [36], multi task-based loss [37], list-wise loss [38], or using anchor graphs for defining the loss function and further improving the hashing performance [39]. Furthermore, an end-to-end supervised product quantization approach for information retrieval was proposed in [40], while deep discrete hashing approaches [18, 41], incremental hashing methods [42] and correlation filtering-based fine-grained hashing approaches [43] have also been utilized to the same end.

The use of MI has also been investigated to aid various aspects of the retrieval process. In [44, 45] MI is employed to provide relevance feedback, in [46, 47], MI is used to provide a powerful deep hashing formulation, while in [48] MI is employed for performing locality sensitive hashing. The Shannon's definition for MI is used in [46] and [47], leading to employing a Monte Carlo sampling scheme to approximate MI, together with a differentiable histogram

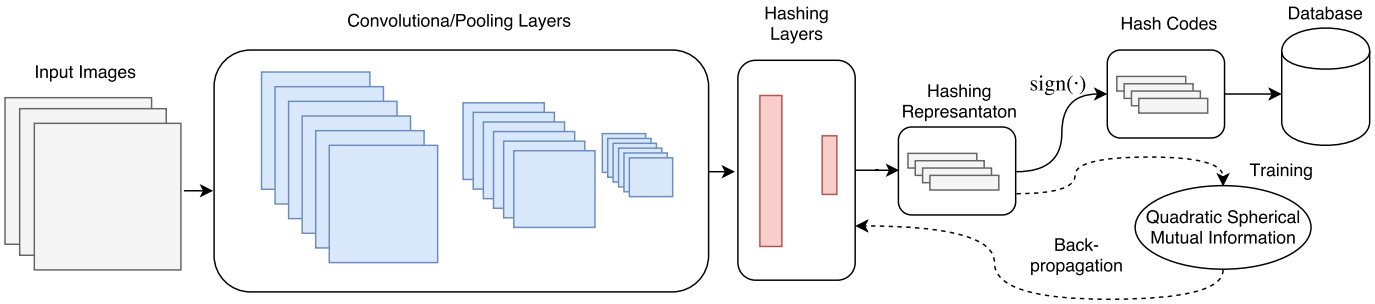


Figure 1: Pipeline of the proposed method: A deep convolutional neural network (CNN) is used to extract a representation that can be used to directly obtain a compact binary hash code. The network is optimized using the proposed Quadratic Spherical Mutual Information loss that is adapted towards the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

binning technique. It should be noted that our approach is vastly different, since instead of approximating MI through random sampling, we analytically derive computationally tractable solutions for calculating MI through a closed-form solution obtained through a Quadratic Mutual Information formulation. Therefore, even though both approaches begin with the same objective, i.e., maximizing mutual information between the hash codes and the information needs, a significantly different approach is employed for tackling the intractable problem of mutual information estimation and optimization in high dimensional spaces.

To the best of our knowledge, this is one of the first works 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 $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$ be a collection of N images, where $\mathbf{y}_i \in \mathbb{R}^n$ is the representation of the i-th image extracted using an appropriate feature extractor, e.g., a deep neural network. Each image $\mathbf{y}_i$ 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 $\mathcal{Y}$ 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 [49], where the information need q is expressed through a *query image* $\mathbf{q} \in \mathbb{R}^n$, that is usually not part of the collection $\mathcal{Y}$.

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 $\mathcal{Q} = \{q_1, q_2, ..., q_M\}$. Then, for each information need q_i, a set of images $\mathcal{Q}_i = \{\mathbf{y}_1^{(i)}, \mathbf{y}_2^{(i)}, ..., \mathbf{y}_{N_i}^{(i)}\}$, where $\mathbf{y}_j^{(i)} \in \mathbb{R}^n$ is the representation of the j-th image that fulfills the i-th information need, is given. Note that $|\mathcal{Q}_i| = N_i$. 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 $p(\mathbf{y}|q_i)$.

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 [13]:

$$H(Q) = -\sum_q P(q) log(P(q)), \quad (1)$$

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: $H(Q) = -\sum_q \Big(P(q)log(P(q)) + (1-P(q))log(1-P(q))\Big)$.

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:

$$H(Q|Y) = -\int_{\mathbf{y}} p(\mathbf{y}) \left(\sum_q p(q|\mathbf{y}) log(p(q|\mathbf{y})) \right) d\mathbf{y}. \quad (2)$$

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

$$\begin{aligned} I(Q, Y) &= H(Q) - H(Q|Y) \\ &= \sum_q \int_{\mathbf{y}} p(q, \mathbf{y}) log \left(\frac{p(q, \mathbf{y})}{P(q)p(\mathbf{y})} \right) d\mathbf{y}. \end{aligned} \quad (3)$$

It is easy to see that MI can be interpreted as the Kullback-Leibler divergence between $p(q, \mathbf{y})$ and $P(q)p(\mathbf{y})$. It is desired to maximize the MI between the representation of the images Y and the information needs Q, 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. On the other hand, it is usually intractable to directly calculate the required probability density $p(\mathbf{y}|q_i)$ 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 quadric divergence metric, instead of the Kullback-Leibler divergence, can be used. In this way, the *Quadratic Mutual Information* (QMI) is defined as [14]:

$$I_T(Q, Y) = \sum_q \int_{\mathbf{y}} \left(p(q, \mathbf{y}) - P(q)p(\mathbf{y})\right)^2 d\mathbf{y}. \quad (4)$$

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

To calculate these quantities, the probability $P(q)$ and the densities $p(\mathbf{y})$ and $p(q, \mathbf{y})$ 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 [50]:

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

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

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

Then, the joint probability density can be estimated as:

$$p(q_i, \mathbf{y}) = p(\mathbf{y}|q_i)p(q_i) = \frac{1}{N} \sum_{j=1}^{N_i} K(\mathbf{y} - \mathbf{y}_j^{(i)}; \sigma^2), \quad (7)$$

while the density of all the images as:

$$p(\mathbf{y}) = \frac{1}{N} \sum_{j=1}^{N} K(\mathbf{y} - \mathbf{y}_j^{(i)}; \sigma^2). \quad (8)$$

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

$$V_{IN}(Q, Y) = \frac{1}{N^2} \sum_{k=1}^{M} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} K(\mathbf{y}_i^{(k)} - \mathbf{y}_j^{(k)}, 2\sigma^2), \quad (9)$$

$$V_{ALL}(Q, Y) = \frac{1}{N^2} \left(\sum_{k=1}^{M} \left(\frac{N_k}{N} \right)^2 \right) \sum_{i=1}^{N} \sum_{j=1}^{N} K(\mathbf{y}_i - \mathbf{y}_j, 2\sigma^2), \quad (10)$$

and

$$V_{BTW}(Q, Y) = \frac{1}{N^2} \sum_{k=1}^{M} \left(\left(\frac{N_k}{N} \right) \sum_{i=1}^{N_k} \sum_{j=1}^{N} K(\mathbf{y}_i^{(k)} - \mathbf{y}_j, 2\sigma^2) \right), \quad (11)$$

where the following property regarding the convolution between two Gaussian kernels was used: $\int_{\mathbf{y}} K(\mathbf{y} - \mathbf{y}_i; \sigma^2) K(\mathbf{y} - \mathbf{y}_j; \sigma^2) d\mathbf{y} = K(\mathbf{y}_i - \mathbf{y}_j, 2\sigma^2)$. 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 $\mathbf{q}$ 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:

$$S_{cos}(\mathbf{y_1}, \mathbf{y_2}) = \frac{1}{2}\left(\frac{\mathbf{y}_1^T\mathbf{y}_2}{\|\mathbf{y}_1\|_2\|\mathbf{y}_2\|_2} + 1\right), \tag{12}$$

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 close to the Hamming distance that is actually used for the retrieval process [33].

Therefore, QSMI is defined as:

$$I_T^{cos}(Q, Y) = V_{IN}^{cos}(Q, Y) + V_{ALL}^{cos}(Q, Y) - 2V_{BTW}^{cos}(Q, Y), \tag{13}$$

where

$$V_{IN}^{cos}(Q, Y) = \frac{1}{N^2}\sum_{k=1}^{M}\sum_{i=1}^{N_k}\sum_{j=1}^{N_k} S_{cos}(\mathbf{y}_i^{(k)}, \mathbf{y}_j^{(k)}), \tag{14}$$

$$V_{ALL}^{cos}(Q, Y) = \frac{1}{N^2}\left(\sum_{k=1}^{M}\left(\frac{N_k}{N}\right)^2\right)\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i, \mathbf{y}_j), \tag{15}$$

and

$$V_{BTW}^{cos}(Q, Y) = \frac{1}{N^2}\sum_{k=1}^{M}\left(\left(\frac{N_k}{N}\right)\sum_{i=1}^{N_k}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i^{(k)}, \mathbf{y}_j)\right). \tag{16}$$

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

$$I_T^{cos} = \frac{1}{N^2}\mathbf{1}_N^T\left(\mathbf{\Delta}\odot\mathbf{S} - \frac{1}{M}\mathbf{S}\right)\mathbf{1}_N, \tag{17}$$

where the indicator matrix is defined as:

$$[\mathbf{\Delta}]_{ij} = \begin{cases} 1, & \text{if the } i\text{-th and the } j\text{-th documents are similar} \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

The notation $\mathbf{1}_N \in \mathbb{R}^N$ 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 this 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 $\mathbf{S}$, smoothing the optimization surface. Therefore, given that the values of $\mathbf{S}$ range in the unit interval, the loss function is re-derived as:

$$\mathcal{L}_{QSMI} = \frac{1}{N^2}\mathbf{1}_N^T\left(\mathbf{\Delta}\odot(\mathbf{S}-1)\odot(\mathbf{S}-1) + \frac{1}{M}(\mathbf{S}\odot\mathbf{S})\right)\mathbf{1}_N. \tag{19}$$

As shown in Figure 2a 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.

The complexity for calculating QSMI is quadratic, since calculating V_{IN}^{cos}, V_{ALL}^{cos} and V_{BTW}^{cos} require a quadratic number of similarity calculations, i.e., $O(N^2)$. To allow for scaling to larger datasets, batch-based optimization is used. This allows for reducing the complexity of QMI from $O(N^2)$ to just $O(N_B^2)$ for one optimization step, where N_B is the used batch size that typically ranges from 64 to 256. Therefore, the total complexity for completing one training epoch is reduced from $O(N^2)$ to $O(NN_B)$. Note that during inference, the proposed method does not require any method-specific components, and, as a result, the complexity only depends on the used network architecture and the length of the hash codes. However, also

note that 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_{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 (28): M is estimated as $M = N_B^2/(\mathbf{1}_{N_B}^T \boldsymbol{\Delta} \mathbf{1}_{N_B})$, 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 $\boldsymbol{\Delta}$ would be: $M(\frac{N}{M})^2 = \mathbf{1}_N^T \boldsymbol{\Delta} \mathbf{1}_N$. 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 $\mathbf{x}$ be the raw representation of an image (e.g., the pixels of an image) and let $\mathbf{y} = f_{\mathbf{W}}(\mathbf{x}) \in \mathbb{R}^n$ be the output of a neural network $f_{\mathbf{W}}(\cdot)$, where $\mathbf{W}$ 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_{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 [34]. 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 [2], as:

$$\mathcal{L}_{hash} = \sum_{i=1}^{N} \| |\mathbf{y}_i| - \mathbf{1}_n \|_1, \tag{20}$$

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

$$\mathcal{L} = \mathcal{L}_{QSMI} + \alpha \mathcal{L}_{hash}, \tag{21}$$

where α is the weight of the hashing regularizer. The network $f_{\mathbf{W}}(\cdot)$ can be then trained using gradient descent, i.e., $\Delta\mathbf{W} = -\eta \frac{\partial J}{\partial \mathbf{W}}$, where η is the used learning rate. Please refer to Appendix A on details regarding the derivation of $\frac{\partial J}{\partial \mathbf{W}}$. After training the network, the hash codes can be readily obtained using the $sign(\mathbf{y})$ 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 [15]. 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 $\mathcal{L}_{QSMI+U}$ 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: $\mathcal{L} = \mathcal{L}_{QSMI} + \alpha \mathcal{L}_{hash} + \beta \mathcal{L}_{QSMI+U}$. 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 five different datasets.

4.1. Datasets and Evaluation Metrics

Five image datasets are used to evaluate the proposed method in this paper: The Fashion MNIST dataset, the CIFAR10 dataset, the NUS-WIDE dataset, the MS COCO dataset, as well as the ILSVRC dataset.

The Fashion MNIST dataset is composed of 60,000 training images and 10,000 test images [51]. 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 [16]. The size of each image is 32×32 pixels (color images) and there is a total of 10 different classes (information needs). The NUS-WIDE is a large-scale dataset that contains 269,648 images that belong to 81 different concepts [17]. The images were resized to 224×224 pixels before feeding them to the network. Following [24], 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 [24]. 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. The MS COCO dataset [52] is another large-scale multi-label dataset that contain 80 different categories. For this dataset we followed a similar standardized setup, as reported in [53]. The ILSVRC dataset [54] is a large-scale image dataset that contains more than one million images that belong to 1,000 different categories. All training images were used for optimizing the models and building the

database, while the validation images were used for querying the database and evaluating the quality of the learned hash codes. Note that a 20-way classification setup is employed, i.e., for each experiment 20 different classes were randomly selected to build the database. All experiments were repeated 20 times and the mean and standard deviation is reported for all the conducted experiments.

To evaluate the proposed method, 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 [55]. Following [55], 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 $Rec(q,k) = \frac{rel(q,k)}{ntotal(q)}$, where $ntotal(q)$ is the total number of database objects that fulfill the same information as q. Precision within hamming radius of 2 is defined as: $Pr_{H2}(q) = \frac{rel_{H2}(q)}{total_{H2}(q)}$, where $rel_{H2}(q)$ is the number of relevant documents within hamming distance 2 from the query, while $total_{H2}(q)$ is the total number of documents within hamming distance 2 from the query. Furthermore, we use the notation (AP_α) to refer to the average precision over all queries at the α-th recall level. For all the experiments conduncted in this work we report the mean Average Precision at eleven equally spaced recall points (0, 0.1, ..., 0.9, 1) calculated as:

$$mAP = \frac{1}{11} \sum_{\alpha=0,\ 0.1,\ ...,\ 0.9,\ 1} AP_\alpha. \tag{22}$$

For multi-label datasets, such as the NUS-WIDE dataset, we calculated precision based on the agreement on the labels of the query, e.g., retrieving objects that carry only half of the labels of the query would lead to a 50% precision for each of them, while retrieving an object that is annotated with all the labels of the query (and possibly more) would lead to a precision of 100% for the specific object. Then, using this definition, mean Average Precision can be similarly calculated for multi-label datasets.

Finally, note that the proposed method was implemented using the PyTorch framework [56] (version 1.0), while the experimental evaluation was conducted on an 8-core workstation that was equipped with an RTX 2060 Graphics Processing Unit (GPU). An open-source implementation of the proposed method is available at `https://github.com/passalis/qsmi`.

4.2. Ablation Study

First, the Fashion MNIST dataset [51], is used to perform an ablation study. The effect of various design choices, i.e., using 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 α for three datasets (Fashion MNIST, CIFAR-10 and NUS-WIDE) is evaluated in Fig. 3. The proposed method is quite stable and consistently achieve the best performance for $\alpha = 0.01$. However, note that this parameter can have a significant effect on the learned hash codes, since a sub-optimal choice can significantly reduce the retrieval precision, as demonstrated in Fig. 3, especially for the NUS-WIDE dataset.

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	$\mathbf{0.861 \pm 0.004}$	$\mathbf{0.876 \pm 0.004}$

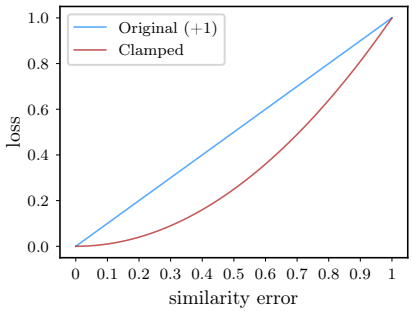
Table 2: Fashion MNIST Evaluation (the mAP for different hash code lengths is reported)

Method	**12 bits**	**24 bits**	**36 bits**	**48 bits**
DSH	0.761 ± 0.02	0.792 ± 0.01	0.809 ± 0.01	0.819 ± 0.01
DPSH	0.767 ± 0.02	0.773 ± 0.01	0.774 ± 0.01	0.759 ± 0.01
Proposed	$\mathbf{0.842 \pm 0.01}$	$\mathbf{0.857 \pm 0.01}$	$\mathbf{0.858 \pm 0.01}$	$\mathbf{0.861 \pm 0.01}$

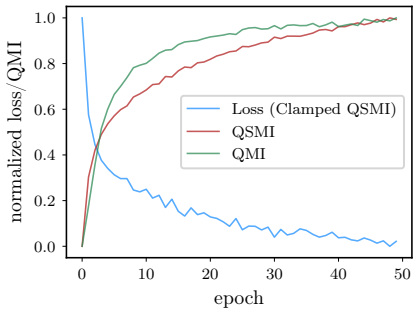
The proposed method was compared to two other state-of-the-art techniques, the Deep Supervised Hashing (DSH) method [2] and the Deep Pairwise Supervised Hashing (DPSH) method [24]. We carefully implemented these methods in a batch-based setting and we tuned their hyper-parameters to obtain the best performance (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 competitive pairwise hashing techniques. Recall that a deep CNN, that was trained from scratch, was employed for the conducted experiments. Again, the proposed method outperforms all the other methods for all the evaluated hash code lengths.

4.3. Supervised Hashing Evaluation

The evaluation results for the CIFAR10 dataset are reported in Table 3. The proposed method outperforms all the other techniques by a large margin for small code

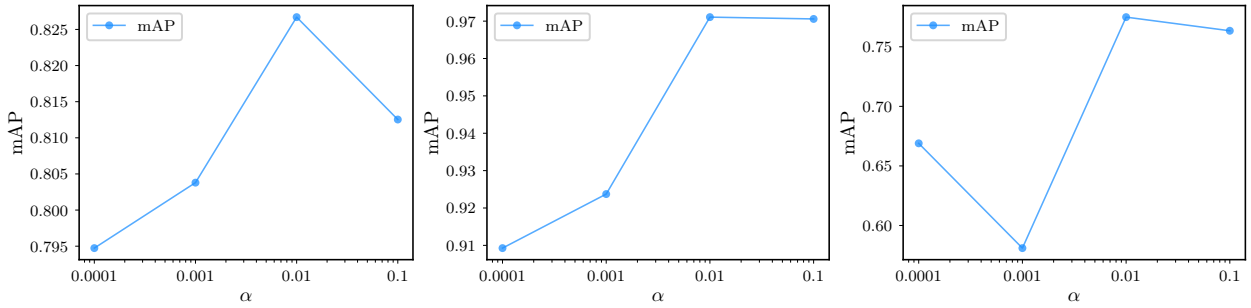


(a) Comparing the behavior of different loss functions for various errors (the original QMI loss is shifted by 1 to allow for easily comparing the plots)



(b) The proposed Clamped QSMI loss and the QSMI and QMI measures during the optimization (the plots have been normalized to the unit interval)

Figure 2: Ablation Study: Studying the differences between the original loss and the proposed hamming loss (Fig. 2a), and the effect of the optimization on the QMI and QSMI measures (Fig. 2b).



(a) Fashion MNIST (b) CIFAR-10 (c) NUS-WIDE

Figure 3: Sensitivity Analysis: Studying effect of the regularization parameter α

Table 3: CIFAR10 Evaluation (the mAP for different hash code lengths is reported)

Method	8 bits	12 bits	24 bits	36 bits	48 bits
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**

Table 4: Training and inference time evaluation (time per batch is reported)

Method	Feed-forward (inference)	Back-propagation (training)
DSH	2.9 ms	0.7 ms
DPSH	4.1 ms	0.7ms
Proposed	3.4 ms	0.7ms

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 of the bits, highlighting the expressive power of the proposed technique.

The time required for training and performing inference using three different methods is reported in Table 4. More specifically, the time needed for performing one weight update in the last layer, i.e., calculating the loss and back-propagating the gradients in the last layer, is 3.4 ms for the proposed method, 2.9 ms for the DSH method and 4.1 ms for the DPSH method. Note that the actual overhead of the proposed approach during training is small (less than 20% compared to DSH method), while there is no overhead during inference, since the same architecture is used (0.7 ms is required per batch for feed-forwarding through the hashing layer of the network). For all these cases, the batch size was set to 128. Furthermore, the hash codes learned using the proposed method were plotted using the t-SNE algorithm [57] in Fig. 4. First, note the generalization abilities of the learned hash codes, since the structure of the space remains the same for both the database and query codes, while the representation of different classes remains quite disentangled, both for the train and test sets. There are some entanglements between different classes on the query/test split, but this only concerns a relatively small number of samples. Also, note that the intra-class similarities seem to be maintained, since the samples that belong to the same class do not collapse into a single point, but instead scatter around each class, both for the train and test sets, leveraging the additional bits to encode these similarities.

The proposed method was also compared to other competitive deep supervised methods in Table 5. The proposed method is compared to DNNH [58], DSH [2], DPSH [24], HashNet [59], MIHash [46, 47], HashGAN [60] and PGDH [61] methods using the same evaluation protocol, i.e., 1,000 test images are sampled, 5,000 images

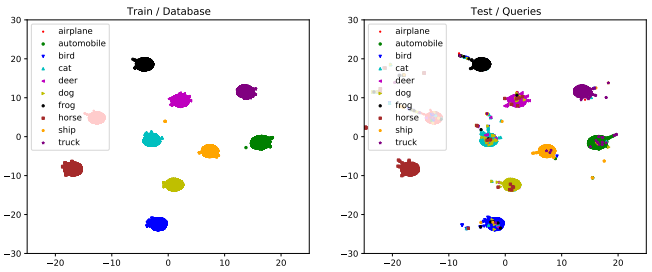


Figure 4: Visualization of the hash codes (48 bits) learned using the proposed method on the CIFAR-10 dataset (30,000 training / database samples are plotted on the left plot, while 10,000 testing samples / queries are plotted on the right plot). The t-SNE [57] algorithm was used for the visualization, while the perplexity of the algorithm was set to 2.

Table 5: CIFAR10 Evaluation: Comparison with other state-of-the-art approaches (the mAP for different hash code lengths is reported)

Method	16 bits	32 bits	64 bits
DNNH [58]	0.555	0.558	0.623
DSH [2]	0.689	0.691	0.716
DPSH [24]	0.646	0.661	0.686
HashNet [59]	0.703	0.711	0.739
HashGAN [60]	0.668	0.731	0.749
PGDH [61]	0.736	0.741	0.762
MIHash [46, 47]	0.760	**0.776**	0.761
Proposed	**0.762**	**0.776**	**0.780**

For all the evaluated methods, apart from the proposed one, the results are as reported in the corresponding literature, i.e., [61], [60] and [62]. The same setup and neural network architecture are used for the evaluated methods.

are used for training the models and the rest of them are used to form the database. Note that the results for the competitive methods are as reported in the corresponding literature [60, 61, 62], while we also used the same neural network architecture (AlexNet [54]) for the experiments conducted using the proposed method. The proposed method significantly outperforms most of the evaluated methods, including the recently proposed HashGAN [60] and PGDH [61] methods. It also achieves comparable performance with the MIHash approach for 16 and 32 bits. However, for longer hash codes (64 bits), it further increases the mAP to 0.78 from 0.76 (next best performing method).

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 [24], 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 6. Again, the proposed method outperforms the rest of the evaluated methods for any code length.

Table 6: 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
Proposed	**0.746**	**0.753**	**0.766**	**0.764**	**0.763**

Table 7: ILSVRC Evaluation (20-way experiments, the mAP for different hash code lengths is reported)

Method	16 bits	32 bits	48 bits
DSH	0.922 ± 0.006	0.941 ± 0.004	0.945 ± 0.003
DPSH	0.782 ± 0.036	0.909 ± 0.049	0.939 ± 0.020
DCH	0.944 ± 0.012	0.911 ± 0.016	0.902 ± 0.005
Proposed	$\mathbf{0.945 \pm 0.004}$	$\mathbf{0.951 \pm 0.004}$	$\mathbf{0.953 \pm 0.003}$

The proposed method was also evaluated using the ILSVRC dataset. A DenseNet backbone pre-trained on the same dataset was used [63], while the same hashing architecture and experimental setup as the one used for the NUS-WIDE dataset were employed. Note that a 20-way retrieval setup, that was repeated 20 times, was used for the evaluation. The experimental results are reported in Table 7. Again, the proposed method outperforms the other evaluated method, i.e., DSH [2], DPSH [24] and DCH [64], leading to both larger mAP, as well as smaller standard deviation among the different evaluation runs.

Finally, we also evaluated the proposed method with a recent state-of-the-art approach for deep quantization, the Central Similarity Quantization (CSQ) approach [53], as well as against several other hashing approaches, demonstrating that further improvements that can be obtained when the proposed method is employed. We followed the same setup as in [53] and evaluated the proposed method on both the NUS-WIDE and MS COCO datasets using a ResNet-50 architecture. The evaluation results are provided in Table 8. Note that again the proposed method led to the overall best results for all the evaluated datasets and hash code lengths.

Table 8: MS COCO and NUS-WIDE Evaluation using a ResNet-50 architecture (the mAP for the first 5000 results for different hash code lengths is reported). The results for the competitive methods are reported from [53].

Method	MS COCO		NUS-WIDE	
	32 bits	64 bits	32 bits	64 bits
CNNH [9]	0.617	0.620	0.659	0.647
DNNH [58]	0.651	0.647	0.738	0.754
DHN [11]	0.731	0.745	0.759	0.771
HashNet [59]	0.773	0.788	0.775	0.790
DCH [64]	0.801	0.825	0.795	0.818
CSQ [53]	0.838	0.861	0.825	0.839
Proposed	**0.854**	**0.883**	**0.830**	**0.842**

Table 9: 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
Proposed	$\mathbf{0.691 \pm 0.093}$	$\mathbf{0.795 \pm 0.061}$	$\mathbf{0.682 \pm 0.144}$

4.4. Retrieval of Unseen Information Needs

Finally, the proposed method was evaluated using the evaluation setup proposed in [15], 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 9. 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 efficient image retrieval, that optimizes the learned codes using an novel information-theoretic measure, the Quadratic Spherical Mutual Information, was proposed. The proposed method was evaluated using five 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 [65].

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 $V_{ALL}^{cos}(Q, Y)$ and $V_{BTW}^{cos}(Q, Y)$ can be calculated as:

$$\begin{aligned} V_{ALL}^{cos}(Q, Y) &= \frac{1}{N^2}\left(\sum_{k=1}^{M}\left(\frac{N_k}{N}\right)^2\right)\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \\ &= \frac{1}{N^2}\left(\sum_{k=1}^{M}\left(\frac{1}{M}\right)^2\right)\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \\ &= \frac{1}{N^2}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i, \mathbf{y}_j), \end{aligned} \tag{23}$$

and

$$\begin{aligned} V_{BTW}^{cos}(Q, Y) &= \frac{1}{N^2}\sum_{k=1}^{M}\left(\left(\frac{N_k}{N}\right)\sum_{i=1}^{N_k}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i^{(k)}, \mathbf{y}_j)\right) \\ &= \frac{1}{N^2}\frac{1}{M}\sum_{k=1}^{M}\sum_{i=1}^{N_k}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i^{(k)}, \mathbf{y}_j) \\ &= \frac{1}{N^2}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \\ &= V_{ALL}^{cos}(Q, Y), \end{aligned} \tag{24}$$

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

$$V_{IN}^{cos}(Q, Y) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N_k}[\mathbf{\Delta}]_{ij} S_{cos}(\mathbf{y}_i, \mathbf{y}_j). \tag{25}$$

Therefore, the QSMI can be simplified as:

$$\begin{aligned} I_T^{cos}(Q, Y) &= V_{IN}^{cos}(Q, Y) - V_{BTW}^{cos}(Q, Y) \\ &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left([\mathbf{\Delta}]_{ij} S_{cos}(\mathbf{y}_i, \mathbf{y}_j) - \frac{1}{M} S_{cos}(\mathbf{y}_i, \mathbf{y}_j)\right) \end{aligned} \tag{26}$$

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

$$\begin{aligned} \mathcal{L}_{QSMI} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\Bigg([\mathbf{\Delta}]_{ij}\left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) - 1\right)^2 \\ + \frac{1}{M}\left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j)\right)^2\Bigg) \end{aligned} \tag{27}$$

since we aim to maximize (26). Also, note that (27) can be equivalently expressed as:

$$\mathcal{L}_{QSMI} = \frac{1}{N^2}\mathbf{1}_N^T\left(\mathbf{\Delta} \odot (\mathbf{S} - 1) \odot (\mathbf{S} - 1) + \frac{1}{M}(\mathbf{S} \odot \mathbf{S})\right)\mathbf{1}_N \tag{28}$$

allowing for efficiently implementing the proposed method.

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

$$\frac{\partial \mathcal{L}_{QSMI}}{\partial \mathbf{W}} = \sum_{i=1}^{N}\left(\frac{\partial \mathcal{L}_{QSMI}}{\partial \mathbf{y}_i}\right)^T \frac{\partial \mathbf{y}_i}{\partial \mathbf{W}}, \tag{29}$$

Table 10: Parameters used for the conducted experiments

Param.	Method	F. MNIST	CIFAR10	NUS-WIDE
Learn. rate	all	0.001	0.001	0.001
Batch size	all	128	128	128
Epochs	all	50	5	50
α	DSH	10^{-5}	10^{-5}	10^{-5}
α	QSMIH	10^{-2}	10^{-2}	10^{-1}
η_{DPSH}	DPSH	5/3* (*36-48 bits)	3	5

Table 11: Network architecture used for the Fashion MNIST dataset

Layer	Kernel	Filters / Neurons	Activation
Convolution	5×5	32	ReLU
Max Pooling	2×2	-	-
Convolution	5×5	64	ReLU
Max Pooling	2×2	-	-
Dense	-	# bits	-

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

$$\frac{\partial \mathcal{L}_{QSMI}}{\partial \mathbf{y}_i} = \frac{1}{N^2} \sum_{j=1, i \neq j}^{N} \left([\mathbf{\Delta}]_{ij} \frac{\partial \left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) - 1 \right)^2}{\partial \mathbf{y}_i} - \frac{1}{M} \frac{\partial \left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \right)^2}{\partial \mathbf{y}_i} \right), \tag{30}$$

where

$$\frac{\partial \left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) - 1 \right)^2}{\partial \mathbf{y}_i} = 2 \left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) - 1 \right) \frac{\partial S_{cos}(\mathbf{y}_i, \mathbf{y}_j)}{\partial \mathbf{y}_i}, \tag{31}$$

and

$$\frac{\partial \left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \right)^2}{\partial \mathbf{y}_i} = 2 S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \frac{\partial S_{cos}(\mathbf{y}_i, \mathbf{y}_j)}{\partial \mathbf{y}_i}. \tag{32}$$

Finally, the derivative of the cosine similarity, needed for calculating (31) and (32), can be computed as:

$$\frac{\partial S_{cos}(\mathbf{y}_i, \mathbf{y}_j)}{\partial [\mathbf{y}_i]_l} = \frac{1}{2} \left(\frac{[\mathbf{y}_j]_l}{\|\mathbf{y}_i\|_2 \|\mathbf{y}_j\|_2} - \frac{\mathbf{y}_i^T \mathbf{y}_j}{\|\mathbf{y}_i\|_2 \|\mathbf{y}_j\|_2} \frac{[\mathbf{y}_i]_l}{\|\mathbf{y}_i\|_2^2} \right). \tag{33}$$

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 10. 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 [66], 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 11. The network was initialized using the default PyTorch initialization scheme [56], 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) [63], 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 [67], 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, 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. For the MS COCO and NUS-WIDE evaluation using the CSQ approach (reported in Table 8), we used an Imagenet-pretrained ResNet-50 architecture, while the batch size was set to 32. We also combined the CSQ loss ($\mathcal{L}_{hash}$) with the proposed one, after weighting the proposed one with $a_{csq} = 0.001$ to ensure that the gradients back-propagated from the different losses are on the same magnitude. The optimization ran for 20 epochs for the MS COCO dataset and 10 epochs for the NUS-WIDE dataset.

Acknowledgement

This work was supported by the European Union's Horizon2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] K. E. Dungan and L. C. Potter, "Classifying vehicles in wide-angle radar using pyramid match hashing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 3, pp. 577–591, 2011.

[2] 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*, 2016, pp. 2064–2072.

[3] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar, "Squeezing deep learning into mobile and embedded devices," *IEEE Pervasive Computing*, vol. 16, no. 3, pp. 82–88, 2017.

[4] N. Passalis and A. Tefas, "Training lightweight deep convolutional neural networks using bag-of-features pooling," *IEEE Trans. on Neural Networks and Learning Systems*, 2018.

[5] J. Plata-Chaves, A. Bertrand, M. Moonen, S. Theodoridis, and A. M. Zoubir, "Heterogeneous and multitask wireless sensor networks - algorithms, applications, and challenges," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 3, pp. 450–465, 2017.

[6] S.-H. Ou, C.-H. Lee, V. S. Somayazulu, Y.-K. Chen, and S.-Y. Chien, "On-line multi-view video summarization for wireless video sensor network," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 1, pp. 165–179, 2015.

[7] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. Annual Symposium on Computational Geometry*, 2004, pp. 253–262.

[8] 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*, 2015.

[9] 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*, 2014, pp. 2156–2162.

[10] 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*, vol. 24, no. 12, pp. 4766–4779, 2015.

[11] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proc. Int. Joint Conf. on Artificial Intelligence*, 2016, pp. 2415–2421.

[12] 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*. IEEE, 2015, pp. 1556–1564.

[13] J. C. Principe, *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science & Business Media, 2010.

[14] K. Torkkola, "Feature extraction by non-parametric mutual information maximization," *Journal of Machine Learning Research*, vol. 3, pp. 1415–1438, 2003.

[15] A. Sablayrolles, M. Douze, N. Usunier, and H. Jégou, "How should we evaluate supervised hashing?" in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2017, pp. 1732–1736.

[16] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical Report*, 2009.

[17] 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. of ACM Conf. on Image and Video Retrieval*, 2009.

[18] Q. Li, Z. Sun, R. He, and T. Tan, "Deep supervised discrete hashing," in *Advances in neural information processing systems*, 2017, pp. 2482–2491.

[19] T. Song, J. Cai, T. Zhang, C. Gao, F. Meng, and Q. Wu, "Semi-supervised manifold-embedded hashing with joint feature representation and classifier learning," *Pattern Recognition*, vol. 68, pp. 99–110, 2017.

[20] C. Zhang and W.-S. Zheng, "Semi-supervised multi-view discrete hashing for fast image search," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2604–2617, 2017.

[21] L. Ma, H. Li, F. Meng, Q. Wu, and L. Xu, "Manifold-ranking embedded order preserving hashing for image semantic retrieval," *Journal of Visual Communication and Image Representation*, vol. 44, pp. 29–39, 2017.

[22] L. Ma, H. Li, F. Meng, Q. Wu, and K. N. Ngan, "Learning efficient binary codes from high-level feature representations for multilabel image retrieval," *IEEE Transactions on Multimedia*, vol. 19, no. 11, pp. 2545–2560, 2017.

[23] Q.-Y. Jiang and W.-J. Li, "Asymmetric deep supervised hashing," in *Proc. Int. Joint Conf. on Artificial Intelligence*, 2018.

[24] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. Twenty-Fifth Int. Joint Conf. on Artificial Intelligence*, 2016, pp. 1711–1717.

[25] F. Shen, X. Gao, L. Liu, Y. Yang, and H. T. Shen, "Deep asymmetric pairwise hashing," in *Proc. ACM on Multimedia Conf.*, 2017, pp. 1522–1530.

[26] X. Wang, Y. Shi, and K. M. Kitani, "Deep supervised hashing with triplet labels," in *Proc. Asian Conf. on Computer Vision*, 2016, pp. 70–84.

[27] L. Ma, H. Li, F. Meng, Q. Wu, and K. N. Ngan, "Discriminative deep metric learning for asymmetric discrete hashing," *Neurocomputing*, vol. 380, pp. 115–124, 2020.

[28] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *Proc. IEEE Int. Conf. on Computer Vision*, 2017.

[29] L. Ma, H. Li, F. Meng, Q. Wu, and K. N. Ngan, "Global and local semantics-preserving based deep hashing for cross-modal retrieval," *Neurocomputing*, vol. 312, pp. 49–62, 2018.

[30] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao, "Triplet-based deep hashing network for cross-modal retrieval," *IEEE Trans. on Image Processing*, vol. 27, no. 8, pp. 3893–3903, 2018.

[31] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," in *Proc. AAAI Conf. on Artificial Intelligence*, 2016, pp. 1230–1236.

[32] 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*, 2014, pp. 1971–1978.

[33] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2012, pp. 2074–2081.

[34] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, "A survey on learning to hash," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769–790, 2018.

[35] X. Zhe, S. Chen, and H. Yan, "Deep class-wise hashing: Semantics-preserving hashing via class-wise loss," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1681–1695, 2019.

[36] S. Zhang, J. Li, and B. Zhang, "Semantic cluster unary loss for efficient deep hashing," *IEEE Transactions on Image Processing*, vol. 28, no. 6, pp. 2908–2920, 2019.

[37] L. Ma, H. Li, Q. Wu, C. Shang, and K. Ngan, "Multi-task learning for deep semantic hashing," in *Proc. IEEE Visual Communications and Image Processing*, 2018, pp. 1–4.

[38] J. Revaud, J. Almazán, R. S. Rezende, and C. R. d. Souza, "Learning with average precision: Training image retrieval with a listwise loss," in *Proc. IEEE International Conference on Computer Vision*, 2019, pp. 5107–5116.

[39] Y. Chen, Z. Lai, Y. Ding, K. Lin, and W. K. Wong, "Deep supervised hashing with anchor graph," in *Proc. of the IEEE International Conference on Computer Vision*, 2019, pp. 9796–9804.

[40] B. Klein and L. Wolf, "End-to-end supervised product quantization for image search and retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5041–5050.

[41] Q.-Y. Jiang, X. Cui, and W.-J. Li, "Deep discrete supervised hashing," *IEEE Transactions on Image Processing*, vol. 27, no. 12, pp. 5996–6009, 2018.

[42] D. Wu, Q. Dai, J. Liu, B. Li, and W. Wang, "Deep incremental hashing network for efficient image retrieval," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9069–9077.

[43] L. Ma, X. Li, Y. Shi, J. Wu, and Y. Zhang, "Correlation filtering-based hashing for fine-grained image retrieval," *IEEE Signal Processing Letters*, 2020.

[44] M. Almasri, C. Berrut, and J.-P. Chevallet, "A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information," in *Proc. European Conf. on Information Retrieval*, 2016, pp. 709–715.

[45] J. Hu, W. Deng, and J. Guo, "Improving retrieval performance by global analysis," in *Proc. Int. Conf. on Pattern Recognition*, vol. 2, 2006, pp. 703–706.

[46] F. Cakir, K. He, S. Adel Bargal, and S. Sclaroff, "Mihash: Online hashing with mutual information," in *Proc. IEEE Int.*

Conf. on Computer Vision, 2017.

[47] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff, "Hashing with mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2424–2437, 2019.

[48] L. Chen, H. Esfandiari, G. Fu, and V. Mirrokni, "Locality-sensitive hashing for f-divergences: Mutual information loss and beyond," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 044–10 054.

[49] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-based multimedia information retrieval: State of the art and challenges," *ACM Trans. on Multimedia Computing, Communications, and Applications*, vol. 2, no. 1, pp. 1–19, 2006.

[50] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[51] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[52] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 740–755.

[53] L. Yuan, T. Wang, X. Zhang, F. E. Tay, Z. Jie, W. Liu, and J. Feng, "Central similarity quantization for efficient image and video retrieval," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3083–3092.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[55] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge University Press, 2008, vol. 1, no. 1.

[56] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[57] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[58] 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*, 2015, pp. 3270–3278.

[59] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *Proc. IEEE Int. Conf. on Computer Vision*, 2017, pp. 5608–5617.

[60] Y. Cao, B. Liu, M. Long, and J. Wang, "Hashgan: Deep learning to hash with pair conditional wasserstein gan," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 1287–1296.

[61] X. Yuan, L. Ren, J. Lu, and J. Zhou, "Relaxation-free deep hashing via policy gradient," in *Proc. European Conf. on Computer Vision*, 2018, pp. 134–150.

[62] Y. Shen, J. Qin, J. Chen, L. Liu, and F. Zhu, "Embarrassingly simple binary representation learning," in *Proc. Int. Conf. in Computer Vision - Compact and Efficient Feature Representation and Learning in Computer Vision*, 2019.

[63] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[64] Y. Cao, M. Long, B. Liu, and J. Wang, "Deep cauchy hashing for hamming space retrieval," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1229–1237.

[65] X. Xu, F. Shen, Y. Yang, H. T. Shen, and X. Li, "Learning discriminative binary codes for large-scale cross-modal retrieval," *IEEE Trans. on Image Processing*, vol. 26, no. 5, pp. 2494–2507, 2017.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2015.

[67] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *Int. Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.