mathematics

MDPI

Article

PLDH: Pseudo-Labels Based Deep Hashing

Huawen Liu [1,*], Minhao Yin [2], Zongda Wu [1], Liping Zhao [1], Qi Li [1], Xinzhong Zhu [3] and Zhonglong Zheng [3]

[1] Department of Computer Science, Shaoxing University, Shaoxing 312000, China
[2] School of Information Science and Technology, Northeast Normal University, Changchun 130024, China
[3] School of Computer Science and Technology, Zhejiang Normal University, Jinhua 311231, China
* Correspondence: liu@usx.edu.cn

Abstract: Deep hashing has received a great deal of attraction in large-scale data analysis, due to its high efficiency and effectiveness. The performance of deep hashing models heavily relies on label information, which is very expensive to obtain. In this work, a novel end-to-end deep hashing model based on pseudo-labels for large-scale data without labels is proposed. The proposed hashing model consists of two major stages, where the first stage aims to obtain pseudo-labels based on deep features extracted by a pre-training deep convolution neural network. The second stage generates hash codes with high quality by the same neural network in the previous stage, coupled with an end-to-end hash layer, whose purpose is to encode data into a binary representation. Additionally, a quantization loss is introduced and interwound within these two stages. Evaluation experiments were conducted on two frequently-used image collections, CIFAR-10 and NUS-WIDE, with eight popular shallow and deep hashing models. The experimental results show the superiority of the proposed method in image retrieval.

Keywords: learning to hash; image retrieval; deep learning; nearest neighbor search; unsupervised learning; pseudo-label

MSC: 68T07

Citation: Liu, H.; Yin, M.; Wu, Z.; Zhao, L.; Li, Q.; Zhu, X.; Zheng, Z. PLDH: Pseudo-Labels Based Deep Hashing. *Mathematics* **2023**, *11*, 2175. https://doi.org/10.3390/math11092175

Academic Editor: Catalin Stoean

Received: 5 April 2023
Revised: 28 April 2023
Accepted: 4 May 2023
Published: 5 May 2023

1. Introduction

Finding interesting objects from a given data collection is an essential task in information retrieval, data mining and image retrieval. When the given data collection is small and low-dimensional, precisely identifying exactly desired objects from the collection has been extensively studied and a great number of retrieval methods have been developed during the past decades [1]. Taking *k*NN (*k* nearest neighbors), which is the most classic and popular neighbor search technique, as an example, it is highly efficient and effective to pick exact neighbors out from a small-scale and low-dimensional data collection [2]. Along with the modern information technology emerging, the scale of data collected from a variety of domains becomes larger and larger. The large-scale property poses great challenges to traditional retrieval techniques. Even for *k*NN, its efficiency of finding exact neighbors in a large-scale data collection is very low, hampering its wide applications in practice greatly [3].

Approximate nearest neighbor search (ANNS) derives objects, which are similar or proximate to a given query, from a large-scale data collection [2]. Since it is highly efficient and scales to the large-scale property without degrading retrieval precision and recall slightly, ANNS has received significant attention. Hash learning and vector quantity are two representative approximate search techniques, where hash learning attracts more extraordinary attentions because of its extreme efficiency [4]. Hash learning encodes data objects into a binary representation by linear or non-linear projection functions [5]. With the binary representation, the search process can be turned to bit operations, e.g., `XOR` and `POPCNT`, which can be executed straightforwardly by CPU.

Roughly speaking, hash techniques can be grouped into two major categories, i.e., data-independent hashing and data-dependent hashing, according to whether the projection functions are learned from data [5]. Locality sensitive hashing (LSH) and its variants are typical examples of the former, whose projection functions are generated randomly [6]. Generally, LSH has relatively poor performance, because it is independent of data. On the contrary, the data-dependent hashing methods construct hashing functions by virtue of inherent properties of data, so that the structural information of data can be preserved. Emblematic data-dependent methods include spectral hashing (SH) [7], iterative quantization (ITQ) [8] and spherical hashing (SpH) [9]. Compared to the data-independent hashing, the data-dependent hashing is superior in retrieval performance.

A recent trend of hash learning is that hash functions are often learned by deep learning [10]. The underlying motivation is that deep learning can derive deep features from data by exploiting deep neural networks, such as AlexNet, CNN, VGG and ResNet [10]. Since the deep features embed high-order semantic information of data, deep hashing methods usually have competitive performance than the conventional ones. DeepBit [11], SGH [12] and DistillHash [13] are typical deep hashing algorithms. For instance, semantic structure-based deep hashing (SSDH) [14] extracts deep features with rich semantic information to construct data labels by using a pre-training convolution neural network, so that the hashing objective functions can preserve semantic similarities among data [15]. For deep hashing, notwithstanding its popularity, there are several limitations that require more effort to work on. Firstly, the early deep hashing methods usually generate binary codes from hand-crafted features (e.g., GIST and SIFT) [16]. As we know, the hand-crafted features embodies less semantic information, making the performance improvement limitedly. Additionally, most of deep hashing methods exploit label information to learn semantic features of data and further to derive binary codes. However, data labels are often unavailable in real-world applications and obtaining them is expensive and intractable.

With this motivation, in this work we leverage a novel end-to-end deep hashing method, called pseudo-labels-based deep hashing (PLDH), for image retrieval. It mainly consists of two stages: obtaining pseudo-labels and generating binary codes. To be specific, the proposed method first adopts a pre-training deep convolution neural network to obtain the similarity degree for each pair of data objects. Afterwards, the pseudo-labels of data objects are generated based on the similarity degrees. The second stage of PLDH covers three major components, i.e., feature learning, code transformation and loss function. Feature learning aims to extract semantic features from data by using a seven-layer convolution neural network, where the first layers are convolution ones, followed by two full-connection layers. Code transformation encodes the data objects into a binary representation by an end-to-end layer. The loss function is used to control the similarity preservation of data and the quality of binary representation. These components are interwound with each other and obtain feedback information alternately during the whole learning process. Owing to the end-to-end layer, the generated binary codes have higher quality and more powerful capabilities.

In a nutshell, the main contributions of this work are briefly summarized as follows:

- We exploit a pre-training deep convolution neural network to obtain the similarity degrees of data, so that the pseudo-labels of data can be further derived.
- The binary representation of data can be achieved by the end-to-end deep neural network, coupled with the pseudo-labels, where information loss between feature learning and code transformation is considered during the whole learning process.
- We conducted extensive experiments on public datasets, i.e., CIFAR-10 and NUS-WIDE. The experimental results show the superiority of PLDH to the state-of-the-art hashing algorithms.

2. Related Work

Due to its extreme efficiency, hash learning receives a great deal of attention and has now become one of the hot topics in image retrieval and big data analysis. To date, many

hash learning methods have been developed. More details can be found in good survey papers, e.g., [4,5], and references therein. Here, we only discuss several typical ones briefly.

As mentioned above, hash techniques include data-independent hashing and data-dependent hashing, according to whether the projection functions are learned from data [5]. Additionally, the hash techniques can also be categorized as supervised hashing and unsupervised hashing, if the label information of data is considered. The former constructs a hashing model with the label information, while the latter does not take data labels into account when constructing a hashing model. For example, ITQ [8] first adopts the technique of principal component analysis to transform data and then take the principal components to generate hash functions. Additionally, SpH [9] employs a hypersphere, instead of hyperplane, to partition data, so that those similar data may fall into closest adjacent regions. As a result, the generated hash functions are more determinative.

Typical supervised hashing algorithms include KSH (Supervised Hashing with Kernels) [17], FastH (Fast Supervised Hashing) [18] and FSDH (Fast Supervised Discrete Hashing) [19], where KSH considers the kernels of binary codes when design hash functions, so that it can effectively handle non-linear data. FastH [18] utilizes boosting trees to cope with the problem raised by the high-dimensional data, while FSDH [19] exploits the strategy of regression of the class labels to binary codes to accelerate hashing process. Note that the performance of supervised hashing is usually better than that of the unsupervised one, because the supervised hashing fully exploits the semantic information of data.

Recently, deep learning has also be widely used in hash learning. The underlying reason is that deep learning can effectively capture the semantic information of data, which may benefit analyzing the inherent properties of data. CNNH (Convolutional Neural Network Hashing) [20] is a representative deep supervised hashing method. It simultaneously derives a feature representation as well as hash functions by using a deep convolutional neural network within two stages. In a similar vein, DPSH (Deep Pairwise-Supervised Hashing) [21] makes use of pairwise labels to learn deep features and hash codes simultaneously. DSDH (Deep Supervised Discrete Hashing) [22] takes both the pairwise label and classification information into consideration within one schema when learning hash codes.

As we know, obtaining class labels for data is a tedious and very expensive thing, especial for large-scale data. Thus, some studies focus on unsupervised deep hashing techniques coupled with inherent properties of data or hash codes. For instance, DeepBit [11] adopts three strategies, i.e., training quantization loss, code even-distributions and bit invariance, to evaluate the quality of generated binary codes. To preserve semantic similarities, SSDH (Semantic Structure Deep Hashing) [14] constructs the semantic structure of data, where data are semantically similar if their distances is obviously smaller than others, to guide the generation of hash codes. Recently, UDPH (Unsupervised Deep Pairwise Hashing) [23] employs anchor-based pairwise similarities to enhance the robustness of binary codes. HashSIM (Hashing via Structural and Intrinsic siMilarity) [24] first constructs structural similarities on highly confident data, and then utilizes them to guide the generation of codes. SPL-UDH (Soft-Pseudo-Label-based Unsupervised Deep Hashing) [25] utilizes a deep auto-encoder network to generate soft pseudo-labels and local similarities of images, and then derive binary codes based on them via the Bayesian theory.

3. Materials and Methods

3.1. Problem Statement

Assume that $\mathbf{X} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n} \in \mathcal{R}^{n \times d}$ is a training data collection consisting of n data objects, each $\mathbf{x}_i$ ($i = 1, \ldots, n$) is the feature vector, represented as d dimensions, of the i-th data object. $\mathbf{y}_i \in \{0,1\}^m$ refers to the label information of $\mathbf{x}_i$. $\mathbf{X}$ is a multi-label data collection when each $\mathbf{y}_i$ ($i = 1, \ldots, n$) is a vector with p values of 0 or 1, i.e., $\mathbf{y}_i \in \{0,1\}^p$; otherwise, it is a normal data collection for supervised learning, where $\mathbf{y} \in \{0,1\}$. Contrastively, as the label information is unavailable, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n} \in \mathcal{R}^{n \times d}$ is used for unsupervised learning.

Since obtaining the label information is extremely expensive in reality, here we only discuss the unsupervised case of $\mathbf{X}$.

Learning to hash attempts to implicitly or explicitly derive a series of hash functions $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m] \in \mathcal{R}_d^m$, so that the data objects $\mathbf{X}$ can be encoded as a binary representation such as $\mathbf{B} \in \{0,1\}^{n \times m}$, where m is the length of binary code, i.e., the number of hash bits, and $m < d$ usually. Formally, each hash function $\mathbf{h}$ is defined as

$$\mathbf{h} : \mathbf{x} \mapsto b \in \{0,1\}, \tag{1}$$

that is, each function $\mathbf{h}$ represents the data object $\mathbf{x}$ as a binary value. If m hash functions $\mathbf{h}_i$ ($i = 1, \ldots, m$) are applied, $\mathbf{x}$ can be further denoted as a binary vector $\mathbf{b} = [b_1, b_2, \ldots, b_m]$ by assembling straightly the m binary values generated by $\mathbf{h}_i$ ($i = 1, \ldots, m$). With this context, the hash codes of $\mathbf{X}$ can be shown as the binary representation $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^n \in \{0,1\}^{n \times m}$, where $\mathbf{b}_i$ is the binary vector of $\mathbf{x}_i$. For the convenience of discussion, hereafter the binary values within $\mathbf{B}$ are denoted as -1 and 1, i.e., $\mathbf{B} \in \{-1,1\}^{n \times m}$.

How to learning efficient and effective hashing functions is still an open and challenging issue. A naive strategy is to generate the hashing functions randomly [6]. Since this kind of generating manner has not taken the property of data into consideration, the quality of generated hash codes is relatively poor. Some shallow hashing techniques exploit the property of data to learn the hashing functions. However, the original features of data are hand-crafted, e.g., GIST or SIFT features of images, and have less semantic information [16]. To address this problem, deep hashing has been introduced. It adopts deep features to construct the hash functions non-linearly. As a result, the generated hash codes are more compact and powerful. Although many deep hashing methods have been developed by now, they pay more attention on the data with label information and less on the data without label information. In this work, we consider the similarities between the data objects as pseudo-labels to address the problem of the deficiency of label information.

3.2. Hashing Model Framework

In this work, we leverage a novel end-to-end deep hashing algorithm, called PLDH (pseudo-labels based deep hashing), for data without label information. The model framework of PLDH is shown as Figure 1. It mainly includes two stages: generating pseudo-labels and deriving hash codes. The first stage first applies a pre-training deep convolution neural network to capture deep features of data, and then calculates the similarity degrees between the data. According to the pairwise similarity degrees, the pseudo-labels of data can be further obtained. The second stage transforms the data into the hash codes via an additional hash layer, with the help of deep features captured by the same deep network. These two stages are interwined to derive the codes and the pseudo-labels in terms of errors estimated by a loss function alternatively.

For the convolution network in PLDH, we take VGGNet (Visual Geometry Group Network) as the architecture of a deep neural network to capture the deep features of data [26]. The architecture of VGGNet was initially designed for large-scale image classification, e.g., ImageNet [27]. Due to its high efficiency and effectiveness, VGGNet has been widely applied in a variety of domains since it was introduced [28]. The VGGNet architecture here is comprised of five convolution layers with different sizes and quantities of sub-layers, and three fully-connected layers. For our purpose, only the first two fully-connected layers are considered and the output of the second layer is used to represent deep features during the first stage of PLDH. In the second stage, the third fully-connected layer, consisting of 1000 units, is taken as the hash layer, whose output is binary.

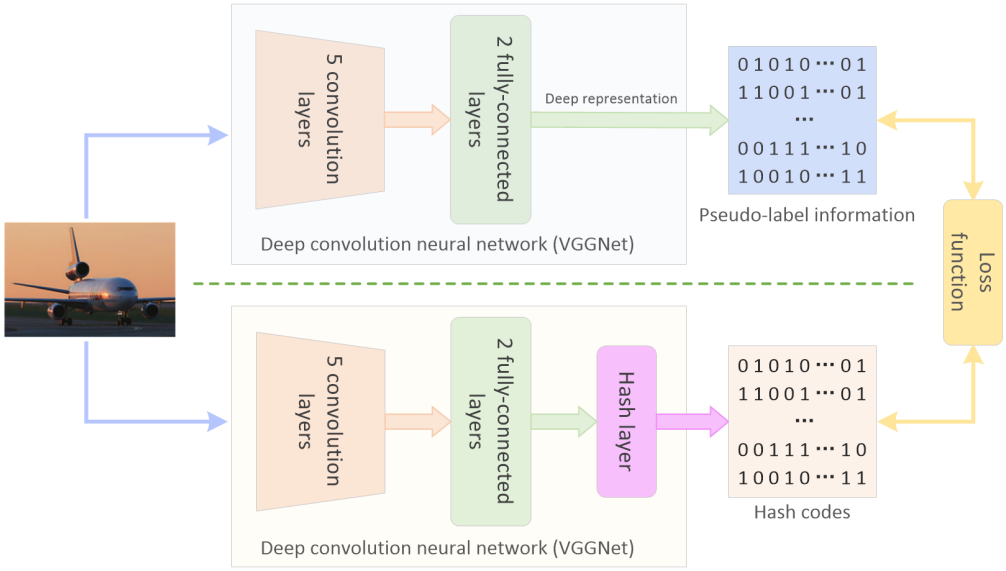


Figure 1. The model framework of PLDH.

3.2.1. Pseudo-Labels

As we know, deep learning is primitively devised for supervised learning, where data labels are available in advance. This implies that the predominant performance of deep learning is heavily dependent on the label information of data. Indeed, data labels are tagged by experienced experts in reality. Thus, they often embody some kind of semantic information and can help to make a decision in data analysis. Unfortunately, tagging data with label information is nontrivial and expensive in the real world, especially for large-scale data collections. To tackle the deficiency problem of label information, several supervised learning techniques utilize Euclidean distance or cosine distance between data objects to take the place of label information. However, the learning performance can be improved limitedly, because there is a large semantic gap between the true semantic of data and the made-up metric space with the hand-crafted features [16].

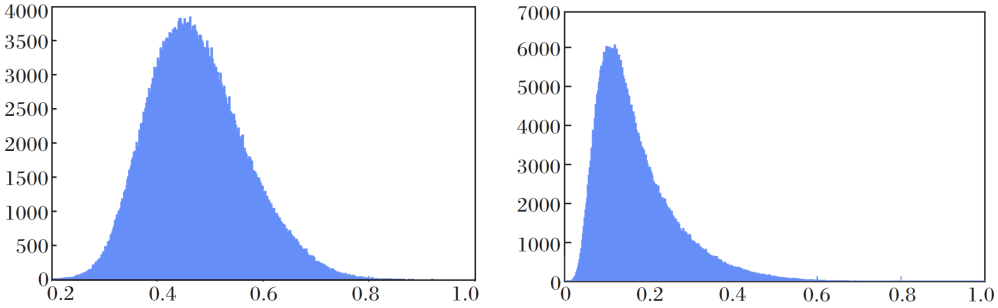
We exploit deep features of data to mimic the true semantic information, so that the semantic gap can be narrowed further. As a matter of fact, the deep features of graphs extracted by a pre-training deep convolution neural network embody rich semantic information [29]. To verify this statement, we conducted an experiment on two public image collections, i.e., CIFAR-10 and NUS-WIDE, to obtain the similarity degrees of images by a pre-training deep convolution neural network. Specifically, for each image collection, we first picked 1000 images out from the collection randomly, and then extracted deep features for each image by the VGG16 network trained on ImageNet in advance; that is,

$$\mathbf{F} = \{\boldsymbol{f}_i\}_{i=1}^{1000}, \tag{2}$$

where $\boldsymbol{f}_i$ is the deep features of the i-th image. Based on the deep features, the cosine similarity between two images can be estimated as follows.

$$sim(\boldsymbol{f}_i, \boldsymbol{f}_j) = \frac{\boldsymbol{f}_i \cdot \boldsymbol{f}_j}{\|\boldsymbol{f}_i\|_2 \cdot \|\boldsymbol{f}_j\|_2} \tag{3}$$

The frequency of cosine similarities of images is given in Figure 2. Observing the frequency, we can find that the higher the pairwise similarity of images, the lower the corresponding frequency. This property is inherent consistent with the structural information of image collections, where the similar images are relatively rarely and sparse. The quantity of dissimilar images is far more than that of similar images in each collection.



(**a**) CIFAR-10 (**b**) NUS-WIDE

Figure 2. The frequency histogram of cosine similarities of images.

With the observation above, we construct semantic pseudo-labels based on the pairwise similarities of images. Let $sim(\boldsymbol{f}_i, \boldsymbol{f}_j)$ be the similarity of the i-th image to the j-th image. The similarity label, s_{ij}, of $\boldsymbol{f}_i$ to $\boldsymbol{f}_j$ is defined as

$$s_{ij} = \begin{cases} 1, & sim(\boldsymbol{f}_i, \boldsymbol{f}_j) > \alpha \\ 0, & otherwise \end{cases} \tag{4}$$

where α is a threshold value between 0 and 1, i.e., $\alpha \in [0, 1]$. From this definition, an image is semantically similar to another one if their cosine similarity is larger than the given threshold; otherwise, they are not considered to be similar to each other. Thus, the semantic labels of images are represented as $\boldsymbol{S} = \{s_{ij}\}_{i,j=1}^{n}$. Since these semantic labels are not true labels, they are pseudo-labels of images.

3.2.2. Deep Hashing

Once the pseudo-labels $\boldsymbol{S}$ are available, the hash codes of data can be derived by the estimation of maximum posterior probability. Let $\boldsymbol{B}$ be the binary representation, i.e., hash codes, of data $\boldsymbol{X}$. The posterior probability of $\boldsymbol{B}$ with respect to $\boldsymbol{S}$ is

$$\begin{aligned} p(\boldsymbol{B} \mid \boldsymbol{S}) &\propto p(\boldsymbol{S} \mid \boldsymbol{B}) p(\boldsymbol{B}) \\ &= \textstyle\prod_{s_{ij} \in S} p(s_{ij} \mid \boldsymbol{b}_i, \boldsymbol{b}_j) p(\boldsymbol{b}_i) p(\boldsymbol{b}_j), \end{aligned} \tag{5}$$

where $p(\boldsymbol{S} \mid \boldsymbol{B})$ is the likelihood function of $\boldsymbol{S}$ under the context of $\boldsymbol{B}$ given, while $p(\boldsymbol{B})$ is the distribution of prior probability. For $p(s_{ij} \mid \boldsymbol{b}_i, \boldsymbol{b}_j)$, it refers to the conditional probability of the pseudo-label s_{ij}, after the hash codes, $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, of the i-th and the j-th data objects are derived. Assume that Φ_{ij} is the inner product of $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, i.e.,

$$\Phi_{ij} = \frac{1}{2} \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle = \frac{1}{2} \boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{b}_j. \tag{6}$$

The conditional probability function $p(s_{ij} \mid \boldsymbol{b}_i, \boldsymbol{b}_j)$ can be calculated as the following equation.

$$p(s_{ij} \mid \boldsymbol{b}_i, \boldsymbol{b}_j) = \begin{cases} \sigma(\Phi_{ij}), & s_{ij} = 1 \\ 1 - \sigma(\Phi_{ij}), & s_{ij} = 0 \end{cases} \tag{7}$$

where $\sigma(\cdot)$ is an active function. For convenience, here we take Sigmoid function as the active function, i.e.,

$$\sigma(\Phi_{ij}) = (1 + e^{-\Phi_{ij}})^{-1}. \tag{8}$$

Equation (7) is capable of representing the inherent property of data. Let $D_H(\boldsymbol{b}_i, \boldsymbol{b}_j)$ be the Hamming distance of binary codes, $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, i.e.,

$$dist_H(\boldsymbol{b}_i, \boldsymbol{b}_j) = \frac{1}{2}(k - \boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{b}_j), \tag{9}$$

where k is the length of binary codes. According to the definition of Equation (7), we know the smaller the Hamming distance $dist_H(\boldsymbol{b}_i, \boldsymbol{b}_j)$, the larger the inner product $\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{b}_j$, and the larger the conditional probability $p(1 \mid \boldsymbol{b}_i, \boldsymbol{b}_j)$. This indicates that the data objects corresponding to $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ are more similar to each other than others. On the other hand, a far distance $dist_H(\boldsymbol{b}_i, \boldsymbol{b}_j)$ between $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ implies that $p(0 \mid \boldsymbol{b}_i, \boldsymbol{b}_j)$ is large, resulting in them being highly dissimilar.

Owing to the semantic property of $\boldsymbol{S}$, the loss function of PLDH can be defined to maximize the likelihood function of the conditional probability. Formally, it is represented as

$$\begin{array}{ll} \min_{\boldsymbol{B}} & \boldsymbol{L} = -\frac{1}{|\boldsymbol{S}|}\Sigma_{s_{ij}\in\boldsymbol{S}} \ln p(s_{ij} \mid \boldsymbol{b}_i, \boldsymbol{b}_j) \\ & \quad = -\frac{1}{|\boldsymbol{S}|}\Sigma_{s_{ij}\in\boldsymbol{S}}(s_{ij}\Phi_{ij} - \ln(1+e^{\Phi_{ij}})) \\ s.t. & \boldsymbol{b}_i, \boldsymbol{b}_j \in \{-1,1\}^k, i,j = 1,2,\ldots,n. \end{array} \tag{10}$$

where $|\boldsymbol{S}|$ denotes the total number of pairwise similarities. Solving straightforwardly the optimization problem of Equation (10) is NP-hard, because its constraint condition is a discrete one; that is, $\boldsymbol{b}_i$ only involves binary values. To cope with this issue, a frequently-used strategy is to relax the discrete constraint into a continuous one. Specifically, we introduce an auxiliary variable $\boldsymbol{u}_i$ for the binary code $\boldsymbol{b}_i$, i.e., $\boldsymbol{u}_i = \boldsymbol{b}_i$. Hence, the optimization problem of Equation (10) can be transformed as the following formulation.

$$\begin{array}{ll} \min_{\boldsymbol{B},\boldsymbol{U}} & \boldsymbol{L} = -\frac{1}{|\boldsymbol{S}|}\Sigma_{s_{ij}\in\boldsymbol{S}}(s_{ij}\Phi_{ij} - \ln(1+e^{\Phi_{ij}})) + \eta\frac{1}{|\boldsymbol{B}|}\Sigma_{i=1}^{n}\|\boldsymbol{u}_i - \boldsymbol{b}_i\|_2^2 \\ s.t. & \Phi_{ij} = \frac{1}{2}\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{u}_j. \end{array} \tag{11}$$

where $\boldsymbol{u}_i \in \mathcal{R}^k$, η is a Lagrange operator, and $|\boldsymbol{B}|$ is the total number of binary codes. For the second penalty term in Equation (11), it aims to narrow the difference of the discrete space to its corresponding continuous one as much as possible.

3.3. Optimization Analysis

For the optimization problem above, it can be solved by an alternative way. Let Θ be the hyper-parameters of deep convolution neural network (i.e., VGGNet) and $\boldsymbol{F}(\boldsymbol{x}_i;\boldsymbol{\Theta})$ be the output of the last fully-connected layer of VGGNet. The output $\boldsymbol{u}_i$ of hash layer for $\boldsymbol{x}_i$ can be represented as

$$\boldsymbol{u}_i = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{F}(\boldsymbol{x}_i;\boldsymbol{\Theta}) + \boldsymbol{v}, \tag{12}$$

where $\boldsymbol{W} \in \mathcal{R}^{4096\times k}$ is the weighted matrix of hash layer, and $\boldsymbol{v} \in \mathcal{R}^{k\times 1}$ is a bias. Substituting $\boldsymbol{u}_i$ in Equation (11) with Equation (12), we have the following equivalent form of Equation (11).

$$\min_{\boldsymbol{B},\boldsymbol{\Theta},\boldsymbol{W},\boldsymbol{v}} \quad \boldsymbol{L} = -\frac{1}{|\boldsymbol{S}|}\Sigma_{s_{ij}\in\boldsymbol{S}}(s_{ij}\Phi_{ij} - \ln(1+e^{\Phi_{ij}})) + \eta\frac{1}{|\boldsymbol{B}|}\Sigma_{i=1}^{n}\|\boldsymbol{W}^{\mathrm{T}}\boldsymbol{F}(\boldsymbol{x}_i;\boldsymbol{\Theta}) + \boldsymbol{v} - \boldsymbol{b}_i\|_2^2. \tag{13}$$

We can iteratively derive the optimization values of $\boldsymbol{B}, \boldsymbol{\Theta}, \boldsymbol{W}$ and $\boldsymbol{v}$ alternatively; that is, solving one of them, while fixing others. To be specific, the iteration process of the optimization problem consists of the following steps.

- **Updating $\boldsymbol{B}$, as $\boldsymbol{\Theta}, \boldsymbol{W}$ and $\boldsymbol{v}$ fixed**. In this case, each hash code $\boldsymbol{b}_i \in \boldsymbol{B}$ can be easily obtained by a soft-threshold function. For example, if the sign function is adopted, $\boldsymbol{b}_i$ can be derived as follows.

$$\boldsymbol{b}_i = sgn(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{F}(\boldsymbol{x}_i;\boldsymbol{\Theta}) + \boldsymbol{v}), \tag{14}$$

  where $sgn(x)$ is the sign function. If $x > 0$, $sgn(x) = 1$; otherwise, $sgn(x) = 0$.
- **Updating $\boldsymbol{\Theta}, \boldsymbol{W}$ and $\boldsymbol{v}$, as $\boldsymbol{B}$ fixed**. We can apply the strategy of back-propagation to estimate the optimal values as $\boldsymbol{B}$ is fixed. Concretely, let the gradients of Equation (13) with respect to the parameters $\boldsymbol{\Theta}, \boldsymbol{W}$ and $\boldsymbol{v}$, respectively, be zero, and we have the following equations hold.

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{F}(\boldsymbol{x}_i;\boldsymbol{\Theta})} = \boldsymbol{W}\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{u}_i}, \tag{15}$$

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}} = \boldsymbol{F}(\boldsymbol{x}_i; \boldsymbol{\Theta})(\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{u}_i})^{\mathrm{T}}, \tag{16}$$

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{v}} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{u}_i}, \tag{17}$$

where

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{u}_i} = \frac{1}{2|\boldsymbol{S}|}\Sigma_{j:s_{ij}\in \boldsymbol{S}}(\sigma(\Phi_{ij}) - s_{ij})\boldsymbol{u}_j + \frac{1}{2|\boldsymbol{S}|}\Sigma_{j:s_{ji}\in \boldsymbol{S}}(\sigma(\Phi_{ji}) - s_{ji})\boldsymbol{u}_j + \eta\frac{2}{|\boldsymbol{B}|}(\boldsymbol{u}_i - \boldsymbol{b}_i). \tag{18}$$

Based on the aforementioned discussion, the implementation details of PLDH are given as follows (Algorithm 1).

Algorithm 1 PLDH: Pesudo-Labels based Deep Hashing

Input: A data collection $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^n$, the code length k, and the hyper-parameter η.
Output: The hash codes $\boldsymbol{B}$ of $\boldsymbol{X}$.
1: Take the output of the pre-training VGGNet as deep features $\boldsymbol{F} = \{\boldsymbol{f}_i\}_{i=1}^n$;
2: Obtain the pseudo-labels $\boldsymbol{S} = \{s_{ij}\}_{i,j=1}^n$ by virtue of Equations (3) and (4);
3: Initialize $\boldsymbol{\Theta}$ with the same parameters of the pre-trained VGGNet;
4: Initialize $\boldsymbol{W}$ and $\boldsymbol{v}$ with random values in $[-\alpha, \alpha]$, where $\alpha = 1/64$;
Repeat
5: Select a mini-batch training data $\{\boldsymbol{x}_i\}_{i=1}^t$ ($t \ll n$) from $\boldsymbol{X}$;
6: Obtain $\{\boldsymbol{F}(\boldsymbol{x}_i; \boldsymbol{\Theta})\}_{i=1}^t$ by the back-propagation strategy;
7: Calculate $\{\boldsymbol{u}_i\}_{i=1}^n$ via Equation (12);
8: Calculate $\{\boldsymbol{b}_i\}_{i=1}^n$ via Equation (14);
9: Update $\boldsymbol{W}$, $\boldsymbol{v}$ and $\boldsymbol{\Theta}$ via Equations (15)–(17), respectively;
Until iteration steps reach a given threshold
10:Return $\boldsymbol{B}$ as the hash codes of $\boldsymbol{X}$.

4. Results

To validate the effectiveness of PLDH, we conducted a series of comparison experiments with eight popular hashing algorithms on two public data collections. In this section, we will discuss the experimental results.

4.1. Experimental Settings

The comparison experiments were conducted on two frequently used benchmark data collections, i.e., CIFAR-10 and NUS-WIDE. The CIFAR data collection contains 60,000 colorful images, each with the size of 32×32 pixels. These images are tagged with ten class labels, such as airplane, truck, ship, car, horse, dog, cat, frog, deer and bird. Each class label involves 6000 colorful images. For each class, 1000 images were randomly picked as queries, and 500 images were taken as training data. NUS-WIDE comprises 269,648 images collected from Flickr, where each image was associated to multiple class labels. There are eighty-one class labels totally, including cars, dogs, airports, birds and earthquake. In the experiments, only the images associated with top-10 frequently-used class labels were considered, and 5000 images were randomly selected as the query, while 10,500 images were taken as training data.

To verify the competitive performance of PLDH, eight popular hashing algorithms were adopted in the comparison experiments. The baselines cover both shallow hashing, such as LSH, SH, ITQ and SpH; and deep hashing, such as DeepBit, SGH, SSDH and DistillHash. They stand for different hash learning techniques. To make a fair comparison, the shallow hashing algorithms were trained on deep features, which were the outputs of the last layer of VGGNet, rather than the original features. Following the routine in the literature, we treated the similarities of images calculated according to their class labels as the ground-truth; that is, two images were considered to be similar, if they were tagged with the same labels. Otherwise, they were dissimilar to each other.

The evaluation protocol used to compare the baselines in our experiments was mean average precision (mAP), which is a widely-used measurement to evaluate the retrieval performance of hashing techniques in information retrieval. Let $\boldsymbol{Q} = \{q_i\}_{i=1}^{t}$ be a set of query. For each query $q \in \boldsymbol{Q}$, its average precision in retrieval is represented as

$$AP(q) = \frac{\sum_{r=1}^{R} Pre_q(r)\delta(\ell_r = \ell_q)}{\sum_{r=1}^{R} \delta(\ell_r = \ell_q)}, \tag{19}$$

where R is the total number of the retrieval results for the query q, and $Pre_q(r)$ is the retrieval precision for the top r results. ℓ_i refers to the data label of the i-th retrieval result. $\delta(\cdot)$ is an indication function, where $\delta(\ell_r = \ell_q) = 1$ if the r-th retrieval result is truly similar to the query q; otherwise, $\delta(\ell_r = \ell_q) = 0$, that is, they have different class labels. Based on this definition, the mean average precision of the set $\boldsymbol{Q}$ of queries is

$$mAP(\boldsymbol{Q}) = \frac{1}{|\boldsymbol{Q}|} \sum_{q \in \boldsymbol{Q}} AP(q). \tag{20}$$

The proposed hashing algorithm is implemented under the framework of PyTorch. To be specific, the optimizer of PLDH is the mini-batch stochastic gradient descent. During the experiments, the size of batch, the momentum and the weight decay of the optimizer were set to 32, 0.9 and 0.0005, respectively. Additionally, the learning rate on CIFAR-10 and NUS-WIDE was fixed at 0.0003 and 0001, respectively, after the cross-validation manner was performed. All experiments were carried out under the platform of Ubuntu Server 16.04, with Intel i7 8700@3.20GHz CPU, Nvidia GTX 1060 GPU and 32GB main memory.

4.2. Experimental Results

mAP is a widely-used evaluation protocol in information retrieval. We also adopted this evaluation protocol to measure the retrieval performance of PLDH to the baselines. Table 1 provides the comparison results of mAP scores of the baselines with different quantities of hash bits, where the bold values are the best ones among the hashing techniques.

Table 1. The mAP comparison of PLDH to the baselines with different numbers of hash bits.

	CIFAR-10				**NUS-WIDE**			
	16 bits	**32 bits**	**64 bits**	**128 bits**	**16 bits**	**32 bits**	**64 bits**	**128 bits**
LSH [6]	0.132	0.158	0.167	0.179	0.432	0.441	0.443	0.482
SH [7]	0.161	0.158	0.151	0.154	0.446	0.454	0.493	0.500
SpH [9]	0.144	0.167	0.178	0.184	0.453	0.460	0.496	0.513
ITQ [8]	0.194	0.209	0.215	0.219	0.528	0.532	0.532	0.542
DeepBit [11]	0.220	0.241	0.252	0.253	0.454	0.463	0.476	0.492
SGH [12]	0.180	0.183	0.189	0.190	0.494	0.483	0.487	0.498
SSDH [14]	0.257	0.256	0.259	0.260	0.623	0.630	0.632	0.649
DistillHash [13]	0.284	0.285	0.287	0.290	0.667	0.675	0.677	0.675
PLDH	**0.459**	**0.495**	**0.509**	**0.500**	**0.685**	**0.701**	**0.702**	**0.703**

From the experimental results in Table 1, one can observe that PLDH has achieved competitive performance in comparing the baselines on these two benchmark image collections. For example, PLDH boosted 28.15% and 16.42% retrieval performance on CIFAR-10 and NUS-WIDE, respectively, in comparing to ITQ, which is the best shallow hashing algorithms. Contrastively, for DistillHash, the best deep hashing algorithms, the retrieval performance was improved 20.43% and 2.42% on CIFAR-10 and NUS-WIDE, respectively, by the proposed hashing method.

For the shallow hashing techniques, the data-dependent algorithms, i.e., ITQ, SH and SpH, achieved better retrieval precision than LSH, which is independent of training data. This fact, however, is consistent with our knowledge, because the data-dependent

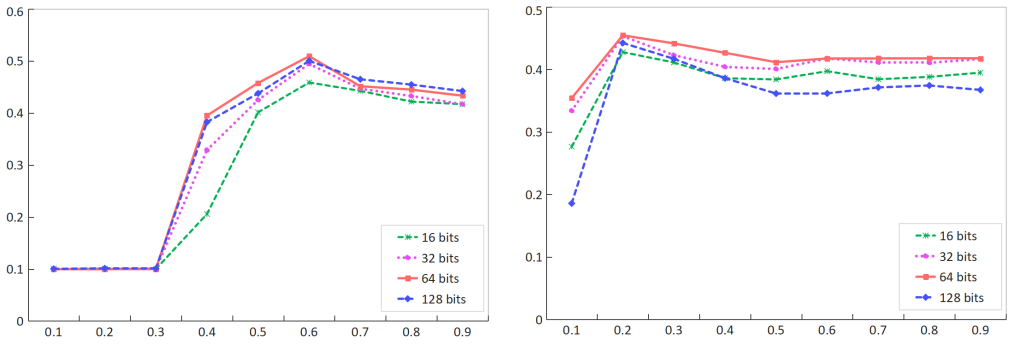
algorithms can effectively capture the inherent properties of data. It should be pointed out that the performance of the shallow hashing techniques was better than that of deep ones in some cases. For instance, ITQ had higher precision than SGH on these image collections. The underlying reason is that the shallow hashing models were constructed on deep features extracted by VGGNet. Additionally, the performance of deep hashing models heavily rely on label information. The absence of label information may degrade the retrieval precision of deep hashing models, notwithstanding the pseudo-labels that are available.

Generally, SSDH, DistillHash and PLDH achieved comparable performance in comparing the shallow hashing techniques, because they were not only constructed on deep features, but also with the help of semantic information, which is derived from the similarities of images. Note that SSDH also adopted the analogical strategy to derive semantic labels on the similarities of data. However, PLDH involved less hyper-parameters and did not require to calculate the cosine similarities, making it more robust and efficient. Additionally, their learning objective functions were also different, where the objective functions of SSDH and PLDH were the minimum of the mean square errors of similarity matrix and the maximum of the likelihood function of similarity matrix, respectively. Moreover, both SSDH and DistillHash did not take the quantization loss into account during the learning stage. In fact, the quantization loss can further improve the quality of hash codes.

4.3. Ablation Analysis

As discussed above, there are two hyper-parameters, i.e., α and η, for PLDH, where the threshold α is used to derive the pseudo-labels. The Lagrange operator η controls the quantization loss during the hash learning stage. To test how much effect they might have brought to PLDH, we carried out additional experiments with different values of α and η on the benchmark data collections.

Figure 3 illustrates the mAP scores of PLDH with different values of the threshold α, when η was fixed. According to the mAP scores in Figure 3, one can observe that the optimization threshold values of α were 0.6 and 0.2 for CIFAR-10 and NUS-WIDE, respectively, albeit the length of hash codes derived by PLDH was different. This is consistent with the distributions of data; that is, the quantities of the pairwise similarities of images are far less than those of dissimilar ones (see Figure 2).

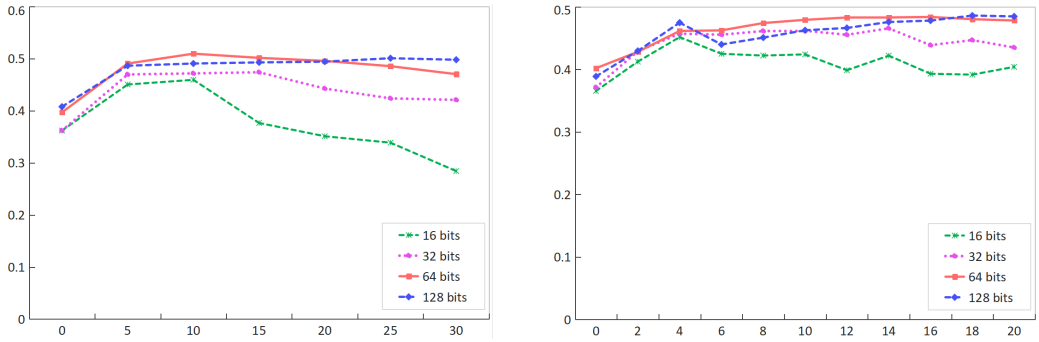


(**a**) CIFAR-10 (**b**) NUS-WIDE

Figure 3. The mAP scores of PLDH with different α values.

Figure 4 shows the mAP curves of PLDH with different values of the operator η, when the threshold α was fixed (α was set to 0.6 for CIFAR-10 and 0.2 for NUS-WIDE, respectively). From the experimental results, we know that as the length of hash codes increases, the optimal value of η also turns out to be large. For example, the optimal values of η were 5, 5, 10 and 25 on CIFAR-10, if the hash codes contained 16, 32, 64 and 128 hash bits, respectively. Similar cases can be found for the data collection of NUS-WIDE. Another observation is that the performance of the model was relatively poor if the quantization loss

was not considered, i.e., $\eta = 0$. This fact implies that the quantization loss may be beneficial to the performance of the deep model, if it is taken into account during the training stage.



(**a**) CIFAR-10 (**b**) NUS-WIDE

Figure 4. The mAP scores of PLDH with different η values.

5. Conclusions

In this work, we proposed a novel end-to-end deep hashing method, called PLDH, for large-scale data without label information. It constructs pseudo-labels for deep hashing by using the inherent semantic property of data, namely the pairwise similarities of data calculated from deep features, to remedy the absence of label information. Compared to unsupervised hashing models, PLDH takes the semantic information of data into account, so that the similarities of data can be preserved, making the generated hashing codes with higher quality. To validate the effectiveness of PLDH, we conducted a series of experiments on two public benchmark image collections. The experimental results show the superiority of PLDH in comparison to the state-of-the-art hashing models. Since deep features are vital to the pseudo-labels, we will adopt different deep learning architectures with different layers to obtain semantic features in our future work.

Before obtaining the pseudo-labels, the similarities among data should be calculated in advance. This requires a large amount of storage space, resulting in PLDH not being very friendly to particularly large-scale data. Thus, our future work will concentrate on addressing this issue by sampling techniques. Moreover, apart from VGG, there are many architectures, e.g., AlexNet, ResNet and Transformer, developed in deep learning, where the Transformer one has received extraordinary popularity recently. So, we will adopt the Transformer architecture as the backbone of PLDH to further improve its performance. Meanwhile, we will also testify the retrieval performance of PLDH on more large-scale image collections.

Author Contributions: Conceptualization, H.L., M.Y. and Z.Z.; methodology, Z.W. and X.Z.; software, Q.L.; validation, H.L., L.Z. and Z.Z.; formal analysis, H.L. and M.Y.; investigation, L.Z.; resources, Z.W.; data curation, Z.W.; writing—original draft preparation, H.L.; writing—review and editing, H.L. and M.Y.; visualization, Q.L.; supervision, X.Z.; project administration, Q.L.; funding acquisition, H.L., X.Z. and Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the Natural Science Foundation (NSF) of China (No. 61976195, 62271321, 62272419, 61976196, 62002226) and the Natural Science Foundation of Zhejiang Province (No. LZ23F020003, LR23F020001, LZ22F020010), Outstanding Talents of "Ten Thousand Talents Plan" in Zhejiang Province (No. 2018R51001), and the Science and Technology Plan Project in Basic Public Welfare class of Shaoxing city (No.2022A11002).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Manning, C.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008.
2. Liu, H.; Li, X.; Zhang S.; Tian, Q. Adaptive hashing with sparse matrix factorization. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4318–4329. [CrossRef] [PubMed]
3. Chen, J.; Zhu, X.; Liu, H. A mutual neighbor-based clustering method and its medical applications. *Comput. Biol. Med.* **2022**, *150*, 106184. [CrossRef]
4. Liu, H.; Zhou, W.; Zhang, H.; Li, G.; Zhang, S.; Li, X. Bit Reduction for Locality-Sensitive Hashing. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, 1–12. [CrossRef] [PubMed]
5. Wang, J.; Zhang, T.; Song, J.; Sebe N.; Shen, H.T. A survey on learning to hash. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 769–790. [CrossRef] [PubMed]
6. Jafari, O.; Maurya, P.; Nagarkar, P.; Islam, K.M.; Crushev, C. A survey on locality sensitive hashing algorithms and their applications. *arXiv* **2021**, arXiv:2102.08942.
7. Weiss, Y.; Torralba, A.; Fergus, R. Spectral hashing. In Proceedings of the Advances in Neural Information Processing Systems 21 (NIPS 2008), Vancouver, BC, Canada, 8–11 December 2008; pp. 1753–1760 .
8. Gong, Y.C.; Lazebnic, S. Iterative Quantization: A procrustean approach to learning binary codes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2011), Washington, DC, USA, 20–25 June 2011; pp. 817–824.
9. Heo, J.-P.; Lee, Y.; He, J.; Chang, S.-F.; Yoon, S.-E. Spherical hashing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2012), Washington, DC, USA, 16–21 June 2012; pp. 2957–2964.
10. Luo, X.; Wang, H.; Wu, D.; Chen, C.; Deng, M.; Huang, J.; Hua, X. A survey on deep hashing methods. *ACM Trans. Knowl. Discov. Data* **2023**, *17*, 50. [CrossRef]
11. Lin, K.; Lu, J.; Chen, C.-S.; Zhou, J.; Sun, M.-T. Unsupervised deep learning of compact binary descriptors. *IEEE Trans. Pattern Analy. Mach. Intell.* **2019**, *41*, 1501–1514. [CrossRef] [PubMed]
12. Dai, B.; Guo, R.; Kumar, S.; He, N.; Song, L. Stochastic generative hashing. In Proceedings of the International Conference om Machine Learning (ICML2017), Sydney, Australia, 6–11 August 2017; pp. 913–922.
13. Yang, E.; Liu, T.; Deng, C.; Liu, W.; Tao, D. DistillHash: Unsupervised deep hashing by distilling data pairs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019 (CVPR2019), Long Beach, CA, USA, 16–17 June 2019; pp. 2941–2950.
14. Yang, E.; Deng, C.; Liu, T.; Liu, W.; Tao, D. Semantic structure-based unsupervised deep hashing. In Proceedings of the 27th International Joint Conference on Artificial Intelligence 2018 (IJCAI2018), Stockholm, Sweden, 13–19 July 2018; pp. 1064–1070.
15. Zhou, Z.; Liu, H.; Lou, J.; Chen, X. Locality sensitive hashing with bit selection. *Appl. Intell.* **2022**, *52*, 14724–14738. [CrossRef]
16. Malik, S.; Amin, J.; Sharif, M.; Yasmin, M.; Kadry, S.; Anjum S. Fractured elbow classification using hand-crafted and deep feature fusion and selection based on whale optimization approach. *Mathematics* **2022**, *10*, 3291. [CrossRef]
17. Liu, W.; Wang, J.; Ji, R.; Jiang, Y.-G.; Chang, S.-F. Supervised hashing with kernels. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 2074–2081.
18. Lin, G.; Shen, C.; Shi Q.; Hengel, A.; Suter D. Fast supervised hashing with decision trees for high-dimensional data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014, Washington, DC, USA, 23–28 June 2014; pp. 1971–1978.
19. Gui, J.; Liu, T.; Sun, Z.; Tao, D.; Tan, T. Fast supervised discrete hashing. *IEEE Tran. Pattern Anal. Mach. Intell.* **2018**, *40*, 490–496. [CrossRef] [PubMed]
20. Xia, R.; Pan, Y.; Lai, H.; Liu, C.; Yan, S. Supervised hashing for image retrieval via image representation learning. In Proceedings of the 28th AAAI Conference Artificial Intelligence (AAA14), Quebec City, QC, Canada, 27–31 July 2014; pp. 2156–2162.
21. Li, W.-J.; Wang, S.; Kang, W.-C. Feature learning based deep supervised hashing with pairwise labels. In Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI16), New York, NY, USA, 9–15 July 2016; pp. 1711–1717.
22. Li, Q.; Sun, Z.; He, R.; Tan, T. Deep supervised discrete hashing. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 2479–2488.
23. Ma, Y.; Li, Q.; Shi, X.; Guo, Z. Unsupervised deep pairwise hashing. *Electronics* **2022**, *11*, 744. [CrossRef]
24. Luo, X.; Ma, Z.; Cheng, W.; Deng, M. Improve deep unsupervised hashing via structural and intrinsic similarity learning. *IEEE Signal Process. Lett.* **2022**, *29*, 602–606. [CrossRef]
25. Sun, Y.; Ye, Y.; Li, X.; Feng, S.; Zhang, B.; Kang, J.; Dai, K. Unsupervised deep hashing through learning soft pseudo label for remote sensing image retrieval. *Knowl.-Based Syst.* **2022**, *239*, 107807. [CrossRef]
26. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015; pp. 1–14.
27. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]

28. Tian, C.; Zhang, Y.; Zuo, W.; Lin, C.-W.; Zhang, D.; Yuan, Y. A Heterogeneous Group CNN for Image Super-Resolution. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *115*, 1–13. [CrossRef]
29. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR14), Columbus, OH, USA, 23–28 June 2014; pp. 580–587.