electronics

MDPI

Article

Unsupervised Deep Pairwise Hashing

Ye Ma [1], Qin Li [2,*], Xiaoshuang Shi [3] and Zhenhua Guo [1]

[1] Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China; my17@tsinghua.org.cn (Y.M.); zhenhua.guo@sz.tsinghua.edu.cn (Z.G.)
[2] School of Software Engineering, Shenzhen Institute of Information Technology, Shenzhen 518172, China
[3] School of Computer Science and Engineering, Univeristy of Electronic Science and Technology of China, Chengdu 611731, China; xsshi2013@gmail.com
* Correspondence: liqin@sziit.edu.cn

Abstract: Although unsupervised deep hashing is potentially very useful for tackling many large-scale tasks, its performance is still far below satisfactory. Additionally, its performance might be significantly improved by effectively exploiting the pair similarity relationship among training data, but the attained similarity matrix usually contains noisy information, which often largely decreases the model performance. To alleviate this issue, in this paper, we propose a novel unsupervised deep pairwise hashing method to effectively and robustly exploit the similarity information between training samples and multiple anchors. We first create an ensemble anchor-based pairwise similarity matrix to enhance the robustness of similarity and dissimilarity relations between training samples and anchors. Afterwards, we propose a novel loss function to directly and robustly take advantage of the similarity and dissimilarity information via a weighted cross-entropy loss, and make use of a square loss to reduce the gap between latent binary vectors and binary codes, and another square loss to form consensus predictions of latent binary vectors. Extensive experiments on large-scale benchmark databases demonstrate the effectiveness of the proposed method, which outperforms recent state-of-the-art unsupervised hashing methods with significantly better ranking performance.

Keywords: deep hashing; anchor based; unsupervised

Citation: Ma, Y.; Li, Q.; Shi, X.; Guo, Z. Unsupervised Deep Pairwise Hashing. *Electronics* **2022**, *11*, 744. https://doi.org/10.3390/electronics11050744

Academic Editor: George A. Papakostas

Received: 19 January 2022
Accepted: 21 February 2022
Published: 28 February 2022

1. Introduction

Hashing has attracted considerable attention for tackling large-scale tasks because it can encode originally high-dimensional data into short binary codes while maintaining the similarity of neighbors, thereby leading to significant gains in computation and storage costs [1,2]. Hashing can be roughly categorized into two main classes, supervised and unsupervised, based on whether semantic labels are used. Supervised hashing [3,4] usually requires a large amount of labels to achieve satisfactory performance; however, label annotation is usually time-consuming and expensive. By contrast, unsupervised hashing [5] does not need semantic labels and aims to discover and, meanwhile, encode the significant intrinsic patterns or structures hidden in data into binary codes. Thus, unsupervised hashing has the potential for large-scale applications.

Early efforts focus on data-independent hashing methods [1,6], which utilize random projections or permutations to construct hash functions, and they usually require long bits to attain high precision per hash table and multiple tables to improve the recall. Data-dependent hashing usually produces more compact binary codes with higher precision and recall. Although numerous data-dependent hashing methods have been proposed and achieved promising performance on various similarity measures, such as Euclidean distance and ℓ_1-norm distance [7], they are still far from being satisfactory for many tasks via the semantic similarity measure. Most of them [5,8,9] learn hash functions using hand-crafted features, which might not be able to represent the image content [10] optimally.

Recently, because convolutional neural networks (CNNs) exhibit the powerful capability of automatically learning feature representations, several unsupervised deep hashing

methods [11–15] adopt CNNs to learn image representations and hash functions. Most of these methods [11–13] utilize either the quantization loss or data reconstruction error to train models without considering the similarity relationship among data, thereby decreasing their retrieval performance on some applications. To address this problem, similarity-adaptive deep hashing (SADH) [15] takes into account the pair similarity among training data and alternately proceeds over three major modules: training deep hash models, updating a similarity graph, and learning binary codes. Although SADH achieves better performance than most previous hashing algorithms, its performance might still be restricted by the noisy information in the similarity graph, i.e., positive values are often given to some dissimilar pairs. In addition, anchor-based models have made significant advances in semi-supervised deep hashing [16] and a scalable optimization mechanism has been proposed [17].

Motivated by the observations mentioned earlier, in this paper, we propose a novel, robust, yet straightforward method, unsupervised deep pairwise hashing (UDPH), to effectively and robustly utilize the pair similarity between training data and unlabeled anchors. The framework of the proposed UDPH is presented in Figure 1. The major contributions of this paper are listed as follows:

- Different from existing anchor-based methods, we enhance the robustness of similarity relations between training data and unlabeled anchors by creating an anchor-based pairwise similarity matrix to preserve their similarity and build a robust ensemble matrix with weighted average;
- We propose a novel loss function composed of three terms: a weighted cross-entropy loss to exploit the similarity information between training data and multiple anchors, a mean square loss to reduce the gap between latent binary vectors and desired codes, and another mean square loss to form consensus predictions of latent binary vectors;
- Extensive experiments on large-scale benchmark databases illustrate the superior performance of UDPH over recent state-of-the-art methods. Additionally, ablation experiments also demonstrate the effectiveness of the three terms in the proposed loss function.

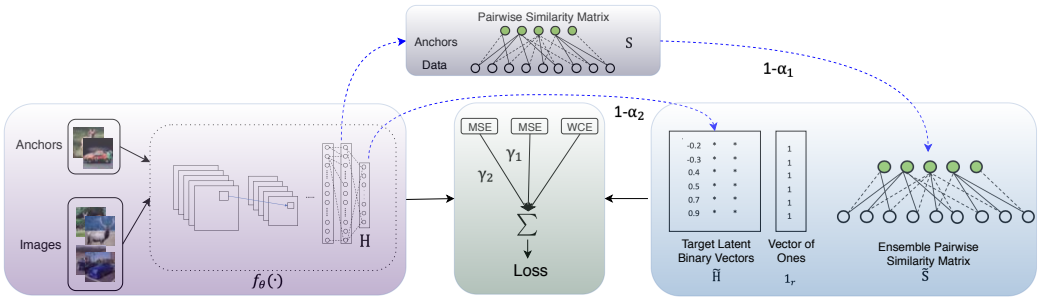


Figure 1. The framework of the proposed method, UDPH, which utilizes the VGG-16 model as our backbone network. MSE denotes the mean square error/loss, and WCE means the weighted cross-entropy loss. WCE utilizes the similarity obtained from the ensemble pairwise similarity matrix as the weight, MSE with γ_1 is to calculate the loss between absolute values of latent binary vectors and vectors of ones, and MSE with γ_2 is to calculate the loss between latent binary vectors and target latent binary vectors. $f_\theta(\cdot)$ represents a convolutional neural network and $\mathbf{H}$ denotes latent binary vectors. α_1 and α_2 are non-negative values to adjust the weight of $\mathbf{S}$ and $\mathbf{H}$, respectively.

2. Related Work

In this section, we briefly review some popular unsupervised non-deep and deep hashing algorithms, and introduce their differences to the proposed method, UDPH.

Unsupervised non-deep hashing usually adopts hand-crafted features to learn hash functions. The popular hashing algorithms [5,8,18,19] learn binary codes via the strategy of "relaxation + thresholding", which might degrade their performance due to the accumulated quantization error between binary codes and its relaxed matrix. To alleviate this issue,

numerous discrete hashing algorithms [9,20–23] have been proposed to directly learn binary codes.

Unsupervised deep hashing usually utilizes CNNs to learn image features and hash functions. Deep hashing (DH) [11], DeepBit [13], and unsupervised deep binary descriptors (UDBD) [24] mainly adopt the quantization loss to learn image representations and hash functions. Unsupervised hashing with a binary deep neural network (UH-BDNN) [12] utilizes the reconstruction loss to encourage the similarity among samples. Discriminative attributes representations (DAR) [25] firstly trains a CNN coupled with unsupervised discriminative clustering and then utilizes the cluster membership as a soft supervision to learn hash functions. Unsupervised triplet hashing (UTH) [26] exploits an unsupervised triplet loss to minimize the distance between an anchor image and its rotated version, while maximize the distance between the anchor image and a random image. HashGAN [14] adopts three networks including a generator, a discriminator, and an encoder to learn hash functions. Similarity-adaptive deep hashing (SADH) [15] constructs a similarity graph using the pair similarity between real training data and anchors, and learns hash functions via alternately proceeding over three major modules: training deep hash models, updating a similarity graph, and learning binary codes. Because of its effectively exploration of similarity information among training data, SADH has achieved state-of-the-art retrieval performance on CIFAR-10 [27] and NUS-WIDE [28] databases. However, its performance might be still restricted by the noisy information in the similarity graph. Compared to SADH, UDPH takes advantage of more robust similarity information by creating a strong ensemble anchor-based pairwise similarity matrix. Additionally, unlike SADH, which approximately solves an NP-hard problem to learn binary codes for model training, the proposed UDPH directly utilizes a novel loss function to robustly train models for exploring the semantic similarity information among training data and anchors.

3. Methodology

In this section, we firstly define an anchor-based pairwise similarity matrix and its ensemble version, and then propose a novel loss function to effectively and robustly exploit the similarity information between training samples and anchors for model training.

3.1. Anchor-Based Pairwise Similarity Matrix

Given training data $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$ and an L-layer deep hashing network $f_\theta(\cdot)$ (please see Figure 1), n is the number of samples and θ denotes the network parameters. Note that we utilize $f_\theta^l(\mathbf{x}_i)$ $(1 \leq l \leq L)$ to represent the output of the l-th layer for the sample $\mathbf{x}_i$. Because n is usually very large, it is inefficient or even impractical to calculate the pair similarity of any two training samples. Fortunately, the similarity relationship among training data can be propagated through multiple anchors [8,29]. We randomly select m $(m << n)$ samples from $\mathbf{X}$ as anchors $\mathbf{A} = \{\mathbf{a}_j\}_{j=1}^{m}$ $(\mathbf{A} \subset \mathbf{X})$ to construct a scalable anchor-based pairwise similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times m}$. Additionally, because samples that are close in the feature space should be close in the output space (*local consistency*) [30], we select the $p(t)$ closest neighbors of each training sample from anchors and then calculate their similarities, where t is the current number of training epochs and $p(t)$ is a piecewise linear function dependent on t to gradually exploit more useful information. However, when only utilizing the similarity information to train models, the features will easily collapse together. To avoid this issue and, meanwhile, exploit more useful information, we select the $p(t)$ farthest anchors of each training sample as non-neighbors and then calculate their dissimilarities.

Specifically, let $\mathbf{x}^f = f_\theta^{(L-1)}(\mathbf{x}) \in \mathbb{R}^d$ and $\mathbf{A}^f = f_\theta^{(L-1)}(\mathbf{A}) \in \mathbb{R}^{m \times d}$ denote feature representations of one training sample $\mathbf{x}$ and anchors $\mathbf{A}$ at the $(L-1)$-th layer, respectively. By [9,15], their similarities can be calculated by leveraging a nonlinear data-to-anchor mapping $(\mathbb{R}^d \to \mathbb{R}^m)$:

$$s_s(\mathbf{x}) = \left[\delta_{s1}e^{-\frac{D^2(\mathbf{x}^f,\mathbf{a}_1^f)}{\varrho_s}}, \delta_{s2}e^{-\frac{D^2(\mathbf{x}^f,\mathbf{a}_2^f)}{\varrho_s}}, \cdots, \delta_{sm}e^{-\frac{D^2(\mathbf{x}^f,\mathbf{a}_m^f)}{\varrho_s}}\right]^T / M_s,$$

where $\delta_{sj} \in \{0,1\}$, and $\delta_{sj} = 1$ if $\mathbf{a}_j$ is one of the $p(t)$ closest anchors of $\mathbf{x}$ based on the distance function $D(\cdot)$ (e.g., Euclidean distance), ϱ_s is a bandwidth parameter for similarity calculation, and $M_s = \sum_{j=1}^{m} \delta_{sj}e^{-\frac{D^2(\mathbf{x},\mathbf{a}_j)}{\varrho_s}}$ so that $\|s_s(\mathbf{x})\|_1 = 1$. Similarly, we calculate their dissimilarities by:

$$s_d(\mathbf{x}) = \left[\delta_{d1}e^{-\frac{D^2(\mathbf{x}^f,\mathbf{a}_1^f)}{\varrho_d}}, \delta_{d2}e^{-\frac{D^2(\mathbf{x}^f,\mathbf{a}_2^f)}{\varrho_d}}, \cdots, \delta_{dm}e^{-\frac{D^2(\mathbf{x}^f,\mathbf{a}_m^f)}{\varrho_d}}\right]^T / M_d,$$

where $\delta_{dj} \in \{0,1\}$ and $\delta_{dj} = 1$ if $\mathbf{a}_j$ is one of the $p(t)$ farthest anchors of $\mathbf{x}$ according to the distance function $D(\cdot)$, ϱ_d is a bandwidth parameter for dissimilarity calculation, and $M_d = \sum_{j=1}^{m} \delta_{dj}e^{-\frac{D^2(\mathbf{x},\mathbf{a}_j)}{\varrho_d}}$ leads to $\|s_d(\mathbf{x})\|_1 = 1$. Then, we can obtain the anchor-based pairwise similarity matrix $\mathbf{S} = [s(\mathbf{x}_1), s(\mathbf{x}_2), \cdots, s(\mathbf{x}_n)]^T \in \mathbb{R}^{n\times m}$, where $s(\mathbf{x}) = s_s(\mathbf{x}) - s_d(\mathbf{x})$. Let $\mathcal{M}$ represent the neighbor set and $\mathcal{C}$ denote the non-neighbor set. For clarity, $\mathbf{S}$ can be calculated by:

$$s_{ij} = \begin{cases} \frac{e^{-\frac{D^2(\mathbf{x}_i^f,\mathbf{a}_j^f)}{\varrho_s}}}{M_s} & (\mathbf{x}_i, \mathbf{a}_j) \in \mathcal{M} \\ -\frac{e^{-\frac{D^2(\mathbf{x}_i^f,\mathbf{a}_j^f)}{\varrho_d}}}{M_d} & (\mathbf{x}_i, \mathbf{a}_j) \in \mathcal{C} \\ 0 & otherwise. \end{cases} \tag{1}$$

To attain a robust relationship between training data and anchors, we create a strong ensemble anchor-based pairwise similarity matrix $\tilde{\mathbf{S}}$ by applying a weight average to $\mathbf{S}$ within multiple previous training epochs, e.g., $\tilde{\mathbf{S}} = \alpha_1\tilde{\mathbf{S}} + (1-\alpha_1)\mathbf{S}$, where $\alpha_1 \in (0,1)$ is a momentum term to determine how far the ensemble reaches into the training history.

3.2. Formulation and Procedure

Hashing is to project original data from a high-dimensional space into a low-dimensional binary space while preserving their similarity relations. Specifically, given the L-layer hashing network $f_\theta(\cdot)$, for any sample $\mathbf{x}$, suppose $f_\theta^L(\mathbf{x}) \in \mathbb{R}^r$ to be the output of the L-th layer in the network, where r is the number of hash bits. Its hash function is defined as: $h(\mathbf{x}) = sgn(f_\theta^L(\mathbf{x}))$, where $sgn(\cdot)$ is a non-linear function with the definition $sgn(z) = 1$ for $z > 0$, otherwise $sgn(z) = -1$.

For one training sample $\mathbf{x}_i$ and one anchor $\mathbf{a}_j$, there exists $-r \le h(\mathbf{x}_i) \circ h(\mathbf{a}_j) \le r$, where $\circ$ denotes the inner product. In order to make $h(\mathbf{x}_i) \circ h(\mathbf{a}_j) > 0$ if $(\mathbf{x}_i, \mathbf{a}_j) \in \mathcal{M}$ and $h(\mathbf{x}_i) \circ h(\mathbf{a}_j) < 0$ when $(\mathbf{x}_i, \mathbf{a}_j) \in \mathcal{C}$, we define $u_{ij} = \sigma(\lambda h(\mathbf{x}_i) \circ h(\mathbf{a}_j)) = \frac{1}{1+e^{-\lambda h(\mathbf{x}_i)\circ h(\mathbf{a}_j)}}$ to represent the similarity between $\mathbf{x}_i$ and $\mathbf{a}_j$ in the low-dimensional binary space, and $u_{ij} \to 1$ if $(\mathbf{x}_i, \mathbf{a}_j) \in \mathcal{M}$, and $u_{ij} \to 0$ when $(\mathbf{x}_i, \mathbf{a}_j) \in \mathcal{C}$, where $\sigma(\cdot)$ represents the sigmoid function and $\lambda > 0$ is a positive constant to regularize the value of $h(\mathbf{x}_i) \circ h(\mathbf{a}_j)$.

Because the function $sgn(\cdot)$ is non-differential, it is usually replaced by a relaxed differential function for model training. There are many choices for relaxed differential functions; for simplicity, here, we choose a differential hyperbolic tangent function $tanh(\cdot)$ in the hashing network. Then, we can obtain a latent binary vector $\mathbf{h} = tanh(f_\theta^L(\mathbf{x})) \in (-1,1)^r$ of $\mathbf{x}$. Let B and B^a be the index set of mini-batch data randomly selected from training data $\mathbf{X}$ and anchors $\mathbf{A}$, respectively. To exploit the similarity information contained in the ensemble pairwise similarity matrix $\tilde{\mathbf{S}}$, one common strategy is to first learn binary codes by solving a non-differential optimization problem and to then utilize them for model training. However, it is usually difficult and time-consuming to solve the non-differential

problem. To avoid this issue, we propose a novel strategy by using a weighted cross-entropy loss function to directly train networks. Because $\tilde{\mathbf{S}}$ denotes the similar weight of pairs, we introduce a matrix $\mathbf{W}$ for the cross-entropy loss function in order to directly represent whether the pair is similar, where w_{ij} is dependent on $\tilde{s}_{ij}$ to determine whether $\mathbf{x}_i$ and $\mathbf{a}_j$ are similar, i.e., $w_{ij} = 1$ when $\tilde{s}_{ij} > 0$ and $w_{ij} = 0$ when $\tilde{s}_{ij} < 0$. Then, the weighted cross-entropy loss is:

$$\begin{aligned} J_{wce} &= \frac{1}{\sum_{i\in B, j\in B^a} |\tilde{s}_{ij}|} \sum_{i\in B, j\in B^a} \tilde{s}_{ij}(-w_{ij} log\, u_{ij} + (1 - w_{ij}) log\,(1 - u_{ij})) \\ s.t.\ \mathbf{h}_i &= tanh(f_\theta^L(\mathbf{x}_i)), \mathbf{h}_j = tanh(f_\theta^L(\mathbf{a}_j)), u_{ij} = \frac{1}{1+e^{-\lambda \mathbf{h}_i \mathbf{h}_j^T}}, \end{aligned} \tag{2}$$

where $|\cdot|$ is an absolute value function. Note that when $\tilde{s}_{ij} \neq 0$ ($j \in B^a$), we do not set $\tilde{s}_{ij} = 1$ or -1 in order to decrease the effect of noisy similarity information.

However, there exists a gap between the latent binary vector $\mathbf{h}$ and desired binary codes $h(\mathbf{x})$, thereby potentially decreasing the model performance [15]. To reduce this gap, we utilize a square loss as follows:

$$J_{hmse} = \| |\mathbf{h}_i| - \mathbf{1}_r \|_2^2, \tag{3}$$

where $\mathbf{1}_r \in \mathbb{R}^r$ is a row vector with all entries being ones.

Recent semi-supervised methods [31–33] illustrate that forming consensus predictions under different configurations (such as training epochs, dropout, and augmentation conditions) for each training sample can improve the model performance when exploring the semantic information of unlabeled data. Inspired by these methods, we aim to form a consensus prediction of the latent binary vector for each training sample in order to boost the model performance. Specifically, similar to [32], we create a target latent binary vector $\tilde{\mathbf{h}}_i$ for $\mathbf{x}_i$ by applying exponential moving average (EMA) to $\mathbf{h}_i$ of multiple previous training epochs, i.e., we first accumulate $\mathbf{h}_i$ into an ensemble vector $\mathbf{h}_i^e$ by $\mathbf{h}_i^e = \alpha_2 \mathbf{h}_i^e + (1 - \alpha_2)\mathbf{h}_i$, and then calculate $\tilde{\mathbf{h}}_i = \mathbf{h}_i^e / (1 - \alpha_2^t)$, where $\alpha_2 \in (0, 1)$ is a momentum term to determine how much of $\mathbf{h}_i^e$ is affected by previous training epochs, $1 - \alpha_2^t$ is to correct the startup bias, and t is the current number of training epochs. Then, we minimize the difference between $\mathbf{h}_i$ and $\tilde{\mathbf{h}}_i$ with the following square loss:

$$J_{smse} = \left\| \mathbf{h}_i - \tilde{\mathbf{h}}_i \right\|_2^2. \tag{4}$$

To simultaneously exploit the pairwise similarity information, reduce the gap between $\mathbf{h}$ and $h(\mathbf{x})$, and form consensus prediction for each training sample, we integrate the three terms (Equations (2)–(4)) to learn the model parameters θ as follows:

$$\begin{aligned} J(\theta) &= J_{wce} + \frac{1}{l_B r} \sum_{i\in B} (\gamma_1 J_{hmse} + \gamma_2 J_{smse}) \\ &= \frac{1}{\sum_{i\in B, j\in L} |\tilde{s}_{ij}|} \sum_{i\in B, j\in L} \tilde{s}_{ij}(-w_{ij} log\, u_{ij} + (1 - w_{ij}) log\,(1 - u_{ij})) \\ &\quad + \frac{1}{l_B r} \sum_{i\in B} (\gamma_1 \| |\mathbf{h}_i| - \mathbf{1}_r \|_2^2 + \gamma_2 \left\| \mathbf{h}_i - \tilde{\mathbf{h}}_i \right\|_2^2) \\ s.t.\ \mathbf{h}_i &= tanh(f_\theta^L(\mathbf{x}_i)), \mathbf{h}_j = tanh(f_\theta^L(\mathbf{a}_j)), u_{ij} = \frac{1}{1+e^{-\lambda \mathbf{h}_i \mathbf{h}_j^T}}, \end{aligned} \tag{5}$$

where l_B is the length of B, and $\gamma_1 \geq 0$ and $\gamma_2 \geq 0$ are to weight the corresponding two regularization terms, respectively.

Based on Equation (5), we can adopt any optimizer, e.g., Adam [34], to learn the model parameters θ. For clarity, we present the detailed procedure of solving Equation (5) to learn θ in Algorithm 1: UDPH. Note that, to boost the model performance, training samples are usually augmented; we utilize $g(\cdot)$ to denote the augmentation function. Additionally, $f_\theta(g(\mathbf{x}_{i\in B}), g(\mathbf{a}_{j\in B^a}))$ denotes the output of the L-th layer followed by the function $tanh(\cdot)$ for the sample $\mathbf{x}_i$ and anchor $\mathbf{a}_j$. After obtaining θ, we can attain binary codes of each training or query data $\mathbf{x}$ by: $h(\mathbf{x}) = sgn(f_\theta^L(\mathbf{x}))$.

Algorithm 1: UDPH

Input: Data $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$, anchors $\mathbf{A} = \{\mathbf{a}_j\}_{j=1}^{m}$, bit number r, parameters λ, γ_1, and γ_2, piecewise linear function $p(t)$, ensembling momentums α_1, and α_2, training epoch number T, network with parameters θ: $f_\theta(\cdot)$, stochastic input augmentation function: $g(\cdot)$

Output: Parameters θ

Initialization:
  Initialize parameters θ by the pre-trained VGG-16 model on ImageNet;
  Construct $\mathbf{S}$ by Equation (1);
  $\tilde{\mathbf{S}} \leftarrow \mathbf{S}$, ▷ ensemble pairwise similarity matrix;
  $\mathbf{H}^e \leftarrow \mathbf{0}_{n\times m}$, ▷ ensemble latent binary vectors;
  $\tilde{\mathbf{H}} \leftarrow \mathbf{0}_{n\times m}$, ▷ target latent binary vectors;
for t **in** $(1, T)$ **do**:
  for each mini-batch B and B^a **do**:
    $\mathbf{h}_{i\in B}, \mathbf{h}_{j\in B^a} \leftarrow f_\theta(g(\mathbf{x}_{i\in B}), g(\mathbf{a}_{j\in B^a}))$;
    loss $\leftarrow$ Equation (5);
    updating θ using optimizers, e.g., Adam [34];
  end for;
  Construct $\mathbf{S}$ by Equation (1);
  $\tilde{\mathbf{S}} \leftarrow \alpha_1\tilde{\mathbf{S}} + (1-\alpha_1)\mathbf{S}$;
  $\mathbf{H}^e \leftarrow \alpha_2\mathbf{H}^e + (1-\alpha_2)\mathbf{H}$;
  $\tilde{\mathbf{H}} \leftarrow \mathbf{H}^e/(1-\alpha_2^t)$;
end for.

4. Experiments

To evaluate the proposed UDPH, we conduct extensive experiments on two large-scale benchmark databases: CIFAR-10 [27] and NUS-WIDE [28]. CIFAR-10 has 60,000 color 32×32 images belonging to 10 classes on average. These images are split into one training set with 50,000 images and one testing set containing 10,000 images. NUS-WIDE is composed of 269,648 images collected from Flickr [28]. There are, totally, 81 semantic concepts, with each image containing multiple labels. Similar to [8,15], we select the 21 most frequent labels for evaluation and, in total, obtain around 195,834 color images. Following [15], for these two databases, we randomly select 100 images from each class to construct the query set and use the rest as a training/gallery set.

4.1. Experimental Settings

We compare UDPH against six popular non-deep unsupervised hashing algorithms (LSH [1], SH [5], AGH [8], ITQ [21], SpH [18], and SGH [19]) and eight state-of-the-art unsupervised deep hashing algorithms (DH [11], UAR [25], UH-BDNN [12], DeepBit [13], HashGAN [14], UTH [26], UDBD [24], and SADH [15]). For the non-deep hashing algorithms, we evaluate them by using hand-crafted features. Specifically, each image in CIFAR-10 is represented by a 512-dimensional GIST vector [35], and each one in NUS-WIDE is represented as a 500-dimensional bag of words (BoW) feature vector. Additionally, we also show their performance on deep features extracted from the $fc7$ layer of the VGG-16 model pre-trained on the ImageNet database. Among the eight deep hashing algorithms, DeepBit, UTH, UDBD, and SADH utilize the VGG-16 model as their backbone networks. For the proposed UDPH, we empirically set the function and adopt the parameters $\gamma_1 = 0.01$, $\gamma_2 = 0.1$, $\alpha_1 = 0.9$, $\alpha_2 = 0.6$, $\lambda = 0.8$ on both CIFAR-10 and NUS-WIDE, except $\lambda = 1.6$ on CIFAR-10 at 16-bit. We randomly select 500 images from the training data of the two databases as anchors, i.e., $m = 500$.

Following [12,15], we adopt semantic similarity as the ground truth in our experiments. For NUS-WIDE, two images are neighbors if they share at least one common label. We evaluate the performance of the aforementioned hashing algorithms by using mean average precision (MAP), MAP@1000, Precision@5000, and Precision@1000. Here, MAP denotes the mean of the average precision of query images over all images in the gallery set; MAP@1000 is the MAP calculated over the top 1000 returned images from the gallery set. Precision@5000 means the rate of correctly retrieved samples from the top 5000 ranked images. A similar definition is applied to Precision@1000. We run the experiments five times and report the average results.

4.2. Experimental Results and Analysis

Table 1 presents retrieval results of the non-deep and deep hashing algorithms at 16-, 32- and 64-bit, selecting 100 query images per class and using the remaining ones as a training/gallery set from CIFAR-10 and NUS-WIDE databases, respectively. Note that we do not present the results of DH, DAR, Hash-GAN, UTH, and UDPD on NUS-WIDE, because there is no publicly reported results. As we can see from Table 1, non-deep hashing algorithms with deep features extracted from the pre-trained VGG-16 model achieve better performance than that using hand-crafted features, probably because the data distribution of ImageNet is similar to that of CIFAR-10 and NUS-WIDE. Additionally, SADH outperforms the other non-deep and deep hashing algorithms except UDPH, because it effectively exploits the adaptive pairwise similarity among data. Moreover, UDPH obtains superior performance over all the non-deep and deep hashing algorithms, especially in terms of the metric MAP. Specifically, on CIFAR-10, the gain of UDPH in MAP is from a relative 2.87% to a relative 14.78% over the best competitor, SADH; on NUS-WIDE, the gain of UDPH in MAP ranges from 1.01% to 12.46%, relatively, over the best competitor; it also obtains better performance in terms of MAP@1000, Precision@5000, and Precision@1000 on the two databases except the Precision@5000 at 16-bit on NUS-WIDE. These results demonstrate the effectiveness and strength of the proposed UDPH. Note that SADH usually achieves its best MAP with short binary codes, e.g., 16-bit, while UDPH obtains better performance with an increasing number of bits. This might be because SADH can effectively preserve the similarity information of the low-rank graph matrix by using short binary codes, but UDPH with longer binary codes can better preserve the similarity relationship between training data and anchors.

Table 1. Retrieval results (%) in terms of MAP, MAP@1000, Precision@5000, and Precision@1000 on CIFAR-10 and NUS-WIDE. † denotes the implemented results based on the provided codes, ∗ means the results copied from Shen et al., 2018 [15], and the other results are directly copied from the corresponding publications. The best accuracy are in bold and the second-best results of each database are underlined.

Method	MAP			MAP@1000			Precision@5000			Precision@1000		
	16-bit	**32-bit**	**64-bit**	**16-bit**	**32-bit**	**64-bit**	**16-bit**	**32-bit**	**64-bit**	**16-bit**	**32-bit**	**64-bit**
CIFAR-10												
LSH †	13.18	14.00	14.92	19.05	21.10	23.83	14.30	15.82	17.07	15.82	18.21	20.53
SH †	12.85	12.65	12.51	20.69	21.03	20.52	14.24	14.06	14.12	16.66	16.98	16.97
AGH †	14.31	13.52	13.44	22.74	22.12	23.54	16.11	15.57	15.58	19.02	19.30	20.48
ITQ †	15.52	15.94	16.49	24.13	26.00	27.56	17.41	18.17	18.89	20.06	22.12	23.39
SpH †	14.28	14.53	15.27	21.68	23.31	26.38	15.94	16.80	17.92	18.34	20.02	22.55
SGH †	14.51	15.05	15.37	22.97	24.92	26.38	15.82	17.67	17.86	19.79	21.34	22.34
LSH+VGG †	13.71	15.81	19.54	20.45	26.13	34.03	15.09	18.37	23.09	17.64	22.45	29.54
SH+VGG †	22.14	19.65	18.18	40.26	38.89	38.48	25.84	23.41	22.07	34.05	32.58	31.76
AGH+VGG †	31.43	28.26	26.55	45.05	47.24	48.79	34.02	32.21	30.79	42.77	43.75	44.90
ITQ+VGG †	31.93	32.21	33.76	45.58	50.53	53.86	35.24	35.75	37.17	42.31	45.94	48.93
SpH+VGG †	19.84	24.23	26.00	33.48	41.34	45.02	22.88	28.13	30.09	28.72	36.78	40.26
SGH+VGG †	23.93	24.30	27.15	42.01	44.12	48.48	27.40	28.52	31.48	36.49	38.74	43.14
DH	16.17	16.62	16.96	-	-	-	-	-	-	23.79	26.00	27.70
DAR	16.82	17.01	17.21	-	-	-	-	-	-	24.54	26.62	28.06
UH-BDNN ∗	30.10	30.89	31.18	-	-	-	33.97	34.48	35.00	-	-	-
HashGAN	29.94	31.47	32.53	44.65	46.34	48.12	-	-	-	41.76	43.62	45.51
DeepBit*	15.95	19.16	20.96	-	-	-	18.02	22.27	24.36	-	-	-
UTH	-	-	-	28.66	30.66	32.41	-	-	-	-	-	-
UDBD	21.70	20.64	23.07	26.36	27.92	34.05	-	-	-	-	-	-
SADH ∗	38.70	38.49	37.68	-	-	-	41.80	41.56	41.15	-	-	-
UDPH	**39.81**	**40.68**	**43.25**	**46.11**	**52.52**	**58.17**	**42.08**	**43.06**	**44.31**	**42.95**	**49.72**	**54.23**

Table 1. *Cont.*

Method	MAP			MAP@1000			Precision@5000			Precision@1000		
	16-bit	**32-bit**	**64-bit**	**16-bit**	**32-bit**	**64-bit**	**16-bit**	**32-bit**	**64-bit**	**16-bit**	**32-bit**	**64-bit**
NUS-WIDE												
LSH †	36.06	36.19	37.16	39.95	40.98	43.01	39.17	39.63	41.43	39.48	40.32	42.28
SH †	34.47	34.96	34.97	41.78	43.00	41.94	37.80	38.77	38.07	40.22	41.40	40.41
AGH †	35.74	35.87	35.75	42.30	43.26	44.41	40.29	41.44	41.71	41.56	42.36	43.46
ITQ †	38.25	38.64	38.75	45.35	46.36	47.05	42.97	43.85	44.20	44.25	45.23	45.71
SpH †	37.26	37.42	37.83	43.58	44.82	46.30	41.82	42.69	43.77	42.76	43.93	45.25
SGH †	37.39	37.33	37.41	45.66	45.79	45.82	43.11	42.91	43.02	44.60	44.56	44.62
LSH+VGG †	38.95	39.56	44.53	46.55	50.57	62.82	43.42	45.64	55.69	45.28	48.65	60.32
SH+VGG †	44.74	42.59	41.54	67.21	66.40	66.69	58.65	55.52	53.93	64.84	63.01	62.54
AGH+VGG †	49.91	49.77	48.38	70.59	72.67	73.94	66.48	67.79	67.64	69.65	71.49	72.63
ITQ+VGG †	54.76	55.20	55.55	70.21	74.14	76.32	68.92	70.43	71.55	70.22	74.01	74.69
SpH+VGG †	47.40	50.44	51.33	64.28	70.18	72.94	58.97	63.83	66.10	62.64	68.24	70.90
SGH+VGG †	46.98	47.71	50.01	69.43	71.49	74.89	61.81	63.07	66.67	67.30	69.22	72.70
UH-BDNN *	39.22	40.32	42.06	-	-	-	45.54	51.34	57.72	-	-	-
DeepBit *	54.26	51.72	54.74	-	-	-	70.18	69.60	72.74	-	-	-
SADH *	60.14	57.99	56.33	-	-	-	**71.45**	73.88	75.04	-	-	-
UDPH	**60.75**	**61.29**	**63.35**	**71.89**	**76.60**	**77.87**	70.49	**74.52**	**75.56**	**71.31**	**75.65**	**76.88**

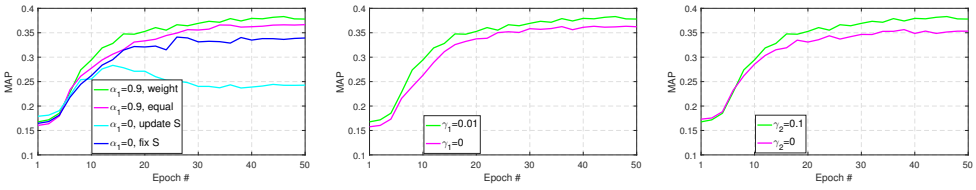
On CIFAR-10, some popular methods, including DBD-MQ [36] and GraphBit [37], utilize 50,000 images as a training/gallery set and 10,000 images as a query set, and they adopt relatively shallow CNNs. Following the experimental protocols in DBD-MQ and GraphBit, to better illustrate the strength of the proposed UDPH, we adopt a shallow network AlexNet [38] as the backbone network, which is pre-trained on the ImageNet database. Additionally, UDPH with AlexNet adopts the same parameter settings as that with VGG-16. Table 2 presents their raking performance in terms of MAP@1000. This further demonstrates the superior performance of UDPH. Specifically, its gain is 17.40%, 18.84%, and 18.35%, relatively, over the best competitor GraphBit at 16-, 32-, and 64-bit, respectively.

Table 2. Retrieval results (%) in terms of MAP@1000 on CIFAR-10 with AlexNet as the backbone. The results of DBD-MQ and GraphBit are directly copied from the publications. The best accuracy results are in bold and the second-best results are underlined.

Method	MAP@1000		
	16-bit	32-bit	64-bit
DBD-MQ	21.53	26.50	31.85
GraphBit	<u>32.15</u>	<u>36.74</u>	<u>39.90</u>
UDPH	**37.81**	**43.66**	**47.22**

4.3. Ablation Studies

Here, we analyze the influence of the ensemble anchor-based pairwise similarity matrix $\tilde{\mathbf{S}}$ and the two terms, J_{hmse} and J_{smse}, on UDPH using VGG-16 as the backbone network. We randomly select 100 images per class from the CIFAR-10 database to construct a query set, and 500 images per class from the remaining ones to construct a set for training and retrieval. Figure 2a shows the MAP of UDPH using four different settings of $\tilde{\mathbf{S}}$. Specifically, (i) "$\alpha_1 = 0.9$, weight" means using the entries in $\tilde{\mathbf{S}}$ as the weight of J_{wce}; (ii) "$\alpha_1 = 0.9$, equal" means $\tilde{s}_{ij} = 1$ if $\tilde{s}_{ij} > 0$ and $\tilde{s}_{ij} = -1$ when $\tilde{s}_{ij} < 0$; (iii) "$\alpha_1 = 0$, update $\mathbf{S}$" denotes $\tilde{\mathbf{S}} = \mathbf{S}$ with updating $\mathbf{S}$ every training epoch; (iv) "$\alpha_1 = 0$, fix $\mathbf{S}$" represents $\tilde{\mathbf{S}} = \mathbf{S}$ without updating $\mathbf{S}$ during the training process. Figure 2a suggests that $\tilde{\mathbf{S}}$ can boost the model performance and smooth the training process, i.e., the best or sub-best MAP is achieved at the last several training epochs. Figure 2b presents the MAP of UDPH without J_{hmse}, i.e., $\gamma_1 = 0$. It illustrates the effectiveness J_{hmse} and the significance of reducing the gap between latent binary vector and binary codes. Figure 2c displays the result of UDPH without J_{smse}, i.e., $\gamma_2 = 0$. It demonstrates that forming consensus predictions of latent binary vectors can also improve model performance.



(**a**) @$\tilde{\mathbf{S}}$ (**b**) @J_{hmse} (**c**) @J_{smse}

Figure 2. Ranking performance in terms of MAP of UDPH with different settings on different training epochs at 32-bit on CIFAR-10 using VGG-16 as the backbone: (**a**) different settings of $\tilde{\mathbf{S}}$, (**b**) with/without the term J_{hmse} in Equation (5), (**c**) with/without the term J_{smse} in Equation (5).

5. Conclusions

In this paper, we propose a novel unsupervised deep pairwise hashing method, which effectively and robustly takes advantage of the similarity information between training samples and anchors. We first construct an anchor-based pairwise similarity matrix, upon

which we create a strong and robust ensemble pairwise similarity matrix to preserve their similarity and dissimilarity relations. Then, we propose a novel loss function consisting of a weighted cross-entropy loss, which utilizes the similarity and dissimilarity between training samples and anchors as the weight to explore their semantic similarity relationship, a square loss to reduce the gap between latent binary vectors and binary codes, and another square loss to form consensus predictions of latent binary vectors for boosting model performance. Experiments on benchmark databases demonstrate the strength of the proposed method and the effectiveness of each term in the proposed loss function. In the future, it is very promising to apply the robust ensemble pairwise similarity matrix and the weighted cross-entropy loss on unsupervised or semi-supervised deep methods because they can effectively explore the semantic similarity information hidden in unlabeled data. Exploring advanced backbones, such as ResNet, to further improve performance is another research direction.

Author Contributions: Conceptualization, X.S.; methodology, X.S. and Y.M.; writing—original draft preparation, Y.M.; writing—review and editing, X.S. and Z.G.; supervision, Q.L. and Z.G.; funding acquisition, Q.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Intelligent perception and computing innovation platform of the Shenzhen Institute of Information Technology (No. PT2019E001), and the Guangdong v2x data security key technology and the expanded application R&D Industry Education Integration Innovation Platform (No. PT2021C002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: CIFAR-10, https://www.cs.toronto.edu/~kriz/cifar.html; NUS-WIDE, https://lms.comp.nus.edu.sg/wp-content/uploads/2019/research/nuswide/NUS-WIDE.html (accessed on 18 January 2022).

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Gionis, A.; Indyk, P.; Motwani, R. Similarity Search in High Dimensions via Hashing. In Proceedings of the VLDB '99 Proceedings of the 25th International Conference on Very Large Data Bases, Edinburgh, Scotland, UK, 7–10 September 1999; pp. 518–529.
2. Torralba, A.; Fergus, R.; Weiss, Y. Small codes and large image databases for recognition. In Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
3. Liu, W.; Wang, J.; Ji, R.; Jiang, Y.G.; Chang, S.F. Supervised hashing with kernels. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 2074–2081.
4. Xia, R.; Pan, Y.; Lai, H.; Liu, C.; Yan, S. Supervised hashing for image retrieval via image representation learning. In Proceedings of the Twenty-eighth AAAI Conference on Artificial Intelligence, Quebec City, QC, Canada, 27–31 July 2014.
5. Weiss, Y.; Torralba, A.; Fergus, R. Spectral hashing. In Proceedings of the 21st International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–10 December 2008; pp. 1753–1760.
6. Broder, A.Z.; Charikar, M.; Frieze, A.M.; Mitzenmacher, M. Min-wise independent permutations. *J. Comput. Syst. Sci.* **2000**, *60*, 630–659. [CrossRef]
7. Shen, F.; Liu, W.; Zhang, S.; Yang, Y.; Shen, H.T. Learning Binary Codes for Maximum Inner Product Search. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 11–18 December 2015; pp. 4148–4156.
8. Liu, W.; Wang, J.; Kumar, S.; Chang, S.F. Hashing with Graphs. In Proceedings of the 28th International Conference on Machine Learning, Washington, DC, USA, 28 June–2 July 2011; pp. 1–8.
9. Liu, W.; Mu, C.; Kumar, S.; Chang, S.F. Discrete Graph Hashing. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 3419–3427.
10. Li, W.J.; Wang, S.; Kang, W.C. Feature learning based deep supervised hashing with pairwise labels. In Proceedings of the IJCAI'16 Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, New York, NY, USA, 9–15 July 2016; pp. 1711–1717.
11. Liong, V.E.; Lu, J.; Wang, G.; Moulin, P.; Zhou, J. Deep hashing for compact binary codes learning. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 2475–2483.
12. Do, T.T.; Doan, A.D.; Cheung, N.M. Learning to Hash with Binary Deep Neural Network. In *European Conference on Computer Vision 2016*; Springer: Cham, Switzerland, 2016; pp. 219–234.

13. Lin, K.; Lu, J.; Chen, C.S.; Zhou, J. Learning Compact Binary Descriptors with Unsupervised Deep Neural Networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 1183–1192.
14. Ghasedi Dizaji, K.; Zheng, F.; Sadoughi, N.; Yang, Y.; Deng, C.; Huang, H. Unsupervised deep generative adversarial hashing network. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 3664–3673.
15. Shen, F.; Xu, Y.; Liu, L.; Yang, Y.; Huang, Z.; Shen, H.T. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 3034–3044. [CrossRef] [PubMed]
16. Shi, X.; Guo, Z.; Xing, F.; Liang, Y.; Yang, L. Anchor-Based Self-Ensembling for Semi-Supervised Deep Pairwise Hashing. *Int. J. Comput. Vis.* **2020**, *128*, 2307–2324. [CrossRef]
17. Shi, X.; Xing, Z.; Zhang, Z.; Sapkota, M.; Guo, Z.; Yang, L. A Scalable Optimization Mechanism for Pairwise Based Discrete Hashing. *IEEE Trans. Image Process.* **2021**, *30*, 1130–1142. [CrossRef] [PubMed]
18. Heo, J.P.; Lee, Y.; He, J.; Chang, S.F.; Yoon, S.E. Spherical hashing. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 2957–2964.
19. Jiang, Q.Y.; Li, W.J. Scalable graph hashing with feature transformation. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 25–31 July 2015.
20. Kulis, B.; Darrell, T. Learning to hash with binary reconstructive embeddings. In Proceedings of the 22nd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 7–10 December 2009; pp. 1042–1050.
21. Gong, Y.; Lazebnik, S.; Gordo, A.; Perronnin, F. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 2916–2929. [CrossRef] [PubMed]
22. Li, X.; Hu, D.; Nie, F. Large graph hashing with spectral rotation. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
23. Do, T.T.; Le Tan, D.K.; Pham, T.T.; Cheung, N.M. Simultaneous feature aggregating and hashing for large-scale image search. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6618–6627.
24. Lin, K.; Lu, J.; Chen, C.S.; Zhou, J.; Sun, M.T. Unsupervised deep learning of compact binary descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 1501–1514. [CrossRef] [PubMed]
25. Huang, C.; Change Loy, C.; Tang, X. Unsupervised learning of discriminative attributes and visual representations. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 5175–5184.
26. Huang, S.; Xiong, Y.; Zhang, Y.; Wang, J. Unsupervised Triplet Hashing for Fast Image Retrieval. In Proceedings of the on Thematic Workshops of ACM Multimedia 2017, Mountain View, CA, USA, 23–27 October 2017; pp. 84–92.
27. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; Citeseer: Pennsylvania, PA, USA, 2009.
28. Chua, T.S.; Tang, J.; Hong, R.; Li, H.; Luo, Z.; Zheng, Y. Nus-wide: A real-world web image database from national university of singapore. In Proceedings of the ACM International Conference on Image and Video Retrieval, Santorini Island, Greece, 8–10 July 2009; pp. 1–9.
29. Shi, X.; Xing, F.; Xu, K.; Sapkota, M.; Yang, L. Asymmetric discrete graph hashing. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
30. Zhou, D.; Bousquet, O.; Lal, T.; Weston, J.; Schölkopf, B. Learning with local and global consistency. *Adv. Neural Inf. Process. Syst.* **2003**, *16*, 321–328.
31. Rasmus, A.; Berglund, M.; Honkala, M.; Valpola, H.; Raiko, T. Semi-supervised Learning with Ladder Networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 3546–3554.
32. Laine, S.; Aila, T. Temporal ensembling for semi-supervised learning. *arXiv* **2016**, arXiv:1610.02242.
33. Tarvainen, A.; Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv* **2017**, arXiv:1703.01780
34. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
35. Oliva, A.; Torralba, A. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **2001**, *42*, 145–175. [CrossRef]
36. Duan, Y.; Lu, J.; Wang, Z.; Feng, J.; Zhou, J. Learning deep binary descriptor with multi-quantization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1183–1192.
37. Duan, Y.; Wang, Z.; Lu, J.; Lin, X.; Zhou, J. GraphBit: Bitwise interaction mining via deep reinforcement learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8270–8279.
38. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2012; pp. 1097–1105. Available online: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html (accessed on 22 February 2022).