MLS^{3}RDUH: Deep Unsupervised Hashing via Manifold based Local Semantic Similarity Structure Reconstructing

Rong-Cheng Tu[1,2], **Xian-Ling Mao**[*1,3] and **Wei Wei**[4]
[1]Department of Computer Science and Technology, Beijing Institute of Technology, China
[2]CETC Big Data Research Institute Co., Ltd., Guiyang 55002, China
[3]Zhijiang Lab, Hangzhou, China
[4]School of Computer Science, Huazhong University of Science and Technology, China
{tu_rc, maoxl}@bit.edu.cn, weiw@hust.edu.cn

Abstract

Most of the unsupervised hashing methods usually map images into semantic similarity-preserving hash codes by constructing local semantic similarity structure as guiding information, i.e., treating each point similar to its k nearest neighbours. However, for an image, some of its k nearest neighbours may be dissimilar to it, i.e., they are noisy datapoints which will damage the retrieval performance. Thus, to tackle this problem, in this paper, we propose a novel deep unsupervised hashing method, called MLS^{3}RDUH, which can reduce the noisy datapoints to further enhance retrieval performance. Specifically, the proposed method first defines a novel similarity matrix by utilising the intrinsic manifold structure in feature space and the cosine similarity of datapoints to reconstruct the local semantic similarity structure. Then a novel log-cosh hashing loss function is used to optimize the hashing network to generate compact hash codes by incorporating the defined similarity as guiding information. Extensive experiments on three public datasets show that the proposed method outperforms the state-of-the-art baselines.

1 Introduction

With the unprecedented growth of image data, hashing based approximate nearest neighbour (ANN) searching have attracted more and more attention due to their high retrieval efficiency and low storage cost. The main idea of hashing methods is to project high dimensional data points into compact binary codes, meanwhile, preserve the semantic similarity of original datapoints.

Generally, hashing methods can be grouped into supervised and unsupervised categories. The supervised hashing [Li *et al.*, 2016; Wang *et al.*, 2018; Huang *et al.*, 2019] methods mainly utilize semantic labels as supervised information to train models to get remarkable performance. However, they extremely rely on vast labeled datapoints to train their models. Thus, it means supervised hashing methods are not suitable to the cases where there are no labeled training datapoints.

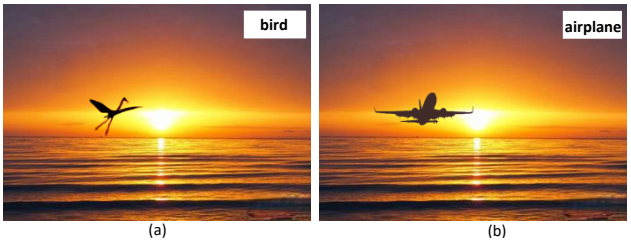

Figure 1: The two images belong to two different categories.

Unsupervised hashing [Gong *et al.*, 2012; Huang *et al.*, 2017; Tu *et al.*, 2019; Yang *et al.*, 2019] methods, which can be used in the cases, learn hashing functions with unlabeled datapoints. Most of the unsupervised hashing methods usually project datapoints into semantic similarity-preserving hash codes by constructing local semantic similarity structure as the guiding information, i.e., treating a datapoint similar to its k nearest neighbours. The k nearest neighbours are the top k datapoints ranked by the natural distance, such as the Euclidean distance and the cosine similarity of their features. However, among the k nearest neighbours of an image, some of them are dissimilar to the image, i.e., they are noisy datapoints which will damage the retrieval performance. For example, as shown in Figure 1, the two images are mostly the same except their core objects that the core object of Figure 1 (a) is a bird and the one of Figure 1 (b) is an airplane. It means the two images are semantic dissimilar, but their natural distance is small. Then, Figure 1 (a) is probably one of the k nearest neighbours of Figure 1 (b), and Figure 1 (a) will be misjudge as similar to Figure 1 (b) which will misguide the hashing model and damage the retrieval performance.

Intuitively, we can use the intrinsic manifold structure in the feature space of datapoints to reduce the noisy datapoints. For example, as shown in Figure 2, among the k nearest neighbours of the query red "circle" point which defined by the natural distance, the brown "triangle" points which are the noisy datapoints are on different manifolds with the red "circle" point. Thus the manifold similarities, defined on the manifold structure, between the red "circle" point and the brown "triangle" points are large. It means the noisy datapoints can be distinguished from the k nearest datapoints by

*The author is corresponding author

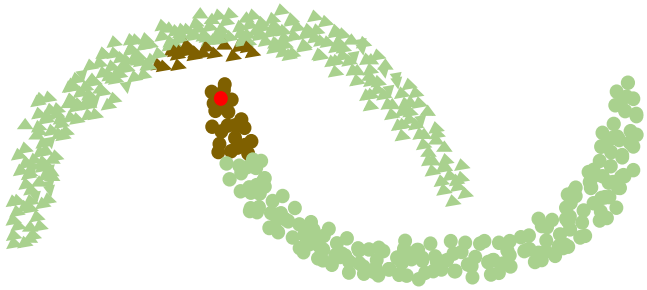

Figure 2: The "triangle" points and the "circle" points are belong to different manifolds, i.e., different categories "triangle" and "circle", respectively. The red "circle" point is a query points, and its k nearest neighbours are the brown points. The brown points are divided into two groups by the manifold structure. *Best viewed in color.*

incorporating the manifold structure. Specifically, given an image $\boldsymbol{x}$, its k nearest neighbours, denoted as $nn_k^c(\boldsymbol{x})$, are the top k points ranked by their cosine similarity with the image $\boldsymbol{x}$; and its o nearest points on manifold structure, denoted as $nn_o^m(\boldsymbol{x})$, are the top o points ranked by the manifold similarity which are measured by a random walk [Zhou *et al.*, 2004b] . Then, if a point in $nn_k^c(\boldsymbol{x})$ but not in $nn_o^m(\boldsymbol{x})$, it can be defined as noisy points, i.e., it is dissimilar to $\boldsymbol{x}$.

Thus, in this paper, we proposed a novel deep unsupervised hashing method, called MLS^{3}RDUH, which reduces the noisy points by incorporating manifold structure to reconstruct the local semantic similarity structure. Specifically, MLS^{3}RDUH deems an image $\boldsymbol{x}$ similar to the points in the set $nn_k^c(\boldsymbol{x}) \cap nn_o^m(\boldsymbol{x})$, and dissimilar to the other points in the set $nn_k^c(\boldsymbol{x})$. Furthermore, inspired by the recently work [Girshick *et al.*, 2014] that rich semantic informations are contained in the features extracted by a pre-trained CNN, MLS^{3}RDUH defines the semantic similarity between the image $\boldsymbol{x}$ and the datapoints that do not belong to $nn_k^c(\boldsymbol{x})$ by the corresponding cosine similarity of their features extracted by a pre-trained CNN. Finally, a novel log-cosh hashing loss function is used to optimize the hashing network to generate compact hash codes by using the defined similarity as guiding information.

To sum up, our contributions can be outlined as follows:

- MLS^{3}RDUH utilises the intrinsic manifold structure in the feature space and cosine similarity to reconstruct the local semantic similarity structure to define a novel similarity matrix.
- A novel log-cosh hashing loss is proposed to optimize the hashing network to improve the performance.
- Experiments on three public datasets show that the proposed method outperforms the state-of-the-art baselines.

2 Related Work

A variety of hashing methods have been proposed in recent years, and based on whether supervised information is needed in the training phase, they can be broadly categorized into supervised and unsupervised hashing methods.

Supervised hashing methods learn hashing functions by using not only the data representation but also the label information in the training phase. A mass of methods in this category have been proposed, such as Deep Supervised Hashing with Pairwise (DPSH) [Li *et al.*, 2016], HashNet [Cao *et al.*, 2017] and Deep Hashing With Gradient Attention (GAH) [Huang *et al.*, 2019]. HashNet alleviates data imbalance by adjusting the weights of semantic similarity matrix to learn discriminative hash codes. GAH utilizes a novel gradient attention mechanism to train deep hashing model.

The unsupervised hashing methods can be divided into traditional unsupervised hashing methods and deep unsupervised hashing methods. The traditional unsupervised hashing methods use hand-crafted features and shallow hash functions to obtain binary hash codes. Numerous algorithms in this category have been proposed, such as Spectral Hashing (SH) [Weiss *et al.*, 2009], and Circulant Binary Embedding (CBE) [Yu *et al.*, 2014]. However, limited by the hand-crafted features and shallow hash functions, it is hard for them to generate high-quality hash codes for complex and high dimensional real-world data. The deep unsupervised hashing methods utilize deep architecture to extract image features to learn hash code. For example, Deepbit [Lin *et al.*, 2016] get rotation invariant and balanced binary hash codes by defined a quantization loss. Semantic structure-based unsupervised deep hashing (SSDH) [Yang *et al.*, 2018] constructs semantic structures based on a Gaussian estimation to guide hashing network learning. DistillHash [Yang *et al.*, 2019] learns hashing models by distilling data pairs with confident semantic similarity relationships as training set.

Compared with these methods, MLS^{3}RDUH utilizes manifold structure to reduce the noisy datapoints when defining the similarity matrix, and a novel loss function is used to incorporate the defined similarity matrix into the training process to get a better performance.

3 Our Method

In this section, we first give a problem definition in section 3.1. The whole architecture of MLS^{3}RDUH will be introduced in section 3.2. Then we discuss the detail of similarity matrix definition and the object function in section 3.3 and section 3.4, respectively. Finally, we will introduce the learning of parameters in section 3.5.

3.1 Problem Definition

Suppose a dataset has n images $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^n$, and the i^{th} image is $\boldsymbol{x}_i$. The goal of unsupervised hashing is to learn a hashing model which maps an image $\boldsymbol{x}_i$ into a similarity-preserving hash code $\boldsymbol{b}_i \in \{-1, 1\}^l$ where l is the length of hash codes, such that an input image $\boldsymbol{x}_i$ will be encoded into a l bit binary code $\boldsymbol{b}_i$.

3.2 Design Overview

As shown in Figure 3, MLS^{3}RDUH consists of a similarity generating part and a hashing network part. In the similarity generating part a pre-trained AlexNet [Krizhevsky *et al.*, 2012] is used to extract features for training images to generate a similarity matrix. Then by using the generated similarity matrix as guiding information, the hashing network can

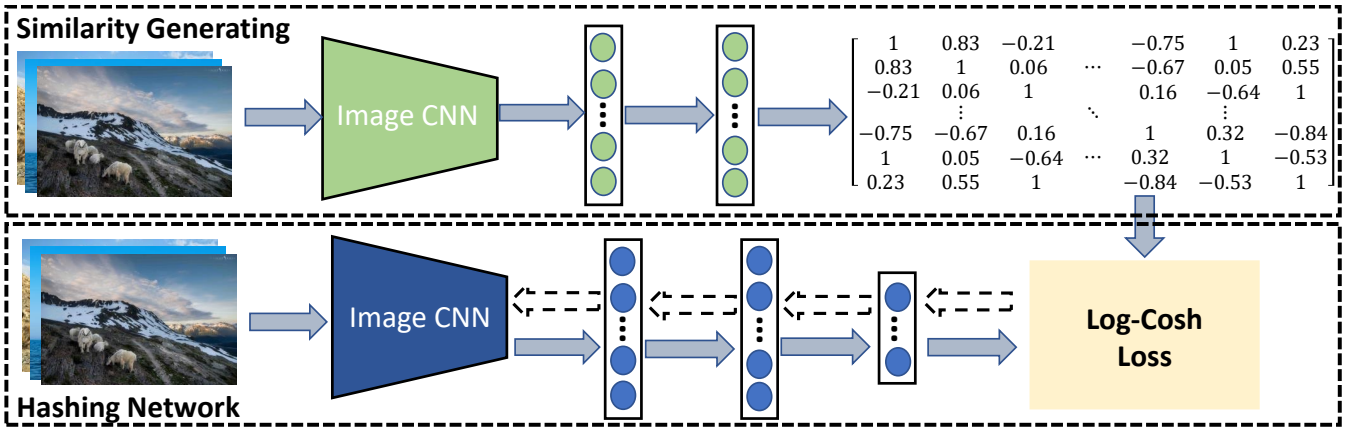

Figure 3: The architecture of MLS^{3}RDUH. The solid arrows indicate forward-propagation, and the dotted arrows indicate back-propagation.

be trained well to generate hash codes for images. Moreover, the hashing network contains five convolutional layers and three fully connected layers. The first seven layers are the same with the first seven layers of AlexNet, and the third fully-connected layer has l units.

3.3 Similarity Matrix Generating

For each image $\boldsymbol{x}_i$, we first select its k nearest neighbours $nn_k^c(\boldsymbol{x}_i)$ based on their cosine similarity. The cosine similarity is formulated as follows:

$$s_c(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{\boldsymbol{f}_i^T \boldsymbol{f}_l}{|\boldsymbol{f}_i| \cdot |\boldsymbol{f}_j|} \tag{1}$$

where $\boldsymbol{f}_i$ is the feature of image $\boldsymbol{x}$ which is extracted by a pre-trained CNN such as Alexnet [Krizhevsky *et al.*, 2012] and VGG [Simonyan and Zisserman, 2014]. $|\cdot|$ denotes the length of a vector.

Then, inspired by [Zhou *et al.*, 2004b], we measure the manifold similarity between datapoints by utilizing a random walk on the nearest neighbour graph. The nearest neighbour graph is undirected weighted which is constructed by using the n image $\boldsymbol{X}$ as nodes and can be represented by sparse symmetric adjacency matrix $\boldsymbol{G} \in \mathcal{R}^{n \times n}$ which is formulated as follows:

$$g_{ij} = \begin{cases} 0, & \boldsymbol{x}_i \notin nn_k^c(\boldsymbol{x}_j) \vee \boldsymbol{x}_j \notin nn_k^c(\boldsymbol{x}_i), \\ s_c(\boldsymbol{x}_i, \boldsymbol{x}_j), & otherwise. \end{cases} \tag{2}$$

Moreover, the diagonal elements of $\boldsymbol{G}$ are zero. With the nearest neighbour graph, for each node $\boldsymbol{x}_i$, the random walk follows the iteration:

$$\boldsymbol{r}_i^{(t)} = \alpha \hat{\boldsymbol{G}} \boldsymbol{r}_i^{(t-1)} + (1-\alpha)\boldsymbol{h}_i \tag{3}$$

where $\alpha \in [0, 1)$ is a hyper-parameter; $\hat{\boldsymbol{G}} = \boldsymbol{D}^{-1/2}\boldsymbol{G}\boldsymbol{D}^{-1/2}$ and $\boldsymbol{D} = diag(\boldsymbol{G}\boldsymbol{1})$ where $\boldsymbol{1}$ is a vector whose elements are 1; $\boldsymbol{r}_i^{(0)} \in \mathcal{R}^n$ is an arbitrary vector; $\boldsymbol{h}_i$ is a one-hot vector that only the i^{th} element of $\boldsymbol{h}_i$ equals to 1, and the others equal to 0. According to [Zhou *et al.*, 2004a], the sequence $\{\boldsymbol{r}_i^t\}_{t=0}^*$ can converges to the solution $\boldsymbol{r}_i^*$, then we have:

$$\boldsymbol{r}_i^* = \alpha \hat{\boldsymbol{G}} \boldsymbol{r}_i^* + (1-\alpha)\boldsymbol{h}_i \tag{4}$$

$$\boldsymbol{r}_i^* = (1-\alpha)(\boldsymbol{I} - \alpha\hat{\boldsymbol{G}})^{-1}\boldsymbol{h}_i \tag{5}$$

where $\boldsymbol{I}$ is an identity matrix. Then, we use $\boldsymbol{r}_{ij}^*$, the j^{th} element of $\boldsymbol{r}_i^*$, denotes the manifold similarity between image $\boldsymbol{x}_i$ and image $\boldsymbol{x}_j$. Finally, for each image $\boldsymbol{x}_i$, we rank the other points by the manifold similarity from large to small and $nn_o^m(\boldsymbol{x}_i)$ are the set of the top o datapoints.

Then, with the constructed $nn_o^m(\boldsymbol{x}_i)$, we can reconstruct $nn_k^c(\boldsymbol{x}_i)$ that divide the k nearest neighbours into two groups: the datapoints in one group are similar to image $\boldsymbol{x}_i$, and the datapoints in the other group are dissimilar to image $\boldsymbol{x}_i$. Specifically, for each datapoint in the $nn_k^c(\boldsymbol{x}_i)$, if it also belong to the $nn_o^m(\boldsymbol{x}_i)$, then it is similar to $\boldsymbol{x}_i$, otherwise, it is dissimilar to $\boldsymbol{x}_i$; for the other datapoints that are not in the $nn_k^c(\boldsymbol{x}_i)$, their similarity with the image $\boldsymbol{x}_i$ are fuzzy. Thus, we can define a similarity matrix $\hat{\boldsymbol{S}}$ as follows:

$$\hat{\boldsymbol{S}}_{ij} = \begin{cases} 1, & \boldsymbol{x}_j \in nn_k^c(\boldsymbol{x}_i) \wedge \boldsymbol{x}_j \in nn_o^m(\boldsymbol{x}_i), \\ -1, & \boldsymbol{x}_j \in nn_k^c(\boldsymbol{x}_i) \wedge \boldsymbol{x}_j \notin nn_o^m(\boldsymbol{x}_i), \\ 0, & otherwise. \end{cases} \tag{6}$$

where $\hat{\boldsymbol{S}}_{ij}$ is the i^{th} row j^{th} column of $\hat{\boldsymbol{S}}$. When $\hat{\boldsymbol{S}}_{ij} = 1$, it means image $\boldsymbol{x}_i$ is similar to image $\boldsymbol{x}_j$; when $\hat{\boldsymbol{S}}_{ij} = -1$, it means image $\boldsymbol{x}_i$ is dissimilar to image $\boldsymbol{x}_j$; when $\hat{\boldsymbol{S}}_{ij} = 0$, it means the similarity between image $\boldsymbol{x}_i$ and image $\boldsymbol{x}_j$ is fuzzy. The defined $\hat{\boldsymbol{S}}$ may be an asymmetric matrix, then to ensure the symmetry, we further update it as $\widehat{\boldsymbol{S}}$ following the rules: if $\hat{\boldsymbol{S}}_{ij} = 1$ or $\hat{\boldsymbol{S}}_{ji} = 1$, then $\widehat{\boldsymbol{S}}_{ij} = \widehat{\boldsymbol{S}}_{ji} = 1$; if $\hat{\boldsymbol{S}}_{ij} = 0$ and $\hat{\boldsymbol{S}}_{ji} = 0$, then $\widehat{\boldsymbol{S}}_{ij} = \widehat{\boldsymbol{S}}_{ji} = 0$; otherwise $\hat{\boldsymbol{S}}_{ij} = \hat{\boldsymbol{S}}_{ji} = -1$.

Furthermore, recently work [Girshick *et al.*, 2014] shows that rich semantic information is contained in the feature extracted by a pre-trained CNN. It means that some semantic similarity information can be mined from the feature of images. Thus, for the similarity fuzzy image pairs, i.e., $\widehat{\boldsymbol{S}}_{ij} = 0$, we further define their similarity by the cosine similarity of their features. Then, we can get the final similarity matrix $\boldsymbol{S}$ which can be formulated as follows:

$$\boldsymbol{S}_{ij} = \begin{cases} \widehat{\boldsymbol{S}}_{ij}, & \widehat{\boldsymbol{S}}_{ij} \neq 0, \\ 2s_c(\boldsymbol{x}_i, \boldsymbol{x}_j) - 1, & otherwise. \end{cases} \tag{7}$$

Thus, the similarity between two images can be divided into three types: completely similar $\boldsymbol{S}_{ij} = 1$, completely dissimilar $\boldsymbol{S}_{ij} = -1$, and partly similar $\boldsymbol{S}_{ij} \in (-1, 1)$.

3.4 Objective Function

The goal of hashing model is to map the images into hash codes which can preserving the constructed similarity $\boldsymbol{S}$, i.e., if image $\boldsymbol{x}_i$ and image $\boldsymbol{x}_j$ are similar, the Hamming distance $d(\boldsymbol{b}_i, \boldsymbol{b}_j) = \frac{1}{2}(l - \boldsymbol{b}_i^T \boldsymbol{b}_j)$ should be small otherwise should be large. To achieve this goal, we used a novel log-cosh hashing loss which can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{W}} \mathcal{L}_1 = & \sum_{i=1}^{n} \sum_{j=1}^{n} log(cosh(\frac{1}{l} \boldsymbol{b}_i^T \boldsymbol{b}_j - \boldsymbol{S}_{ij})) \\ & s.t. \quad \boldsymbol{b}_i = sign(\mathcal{F}(\boldsymbol{x}_i; \boldsymbol{W})). \end{aligned} \tag{8}$$

where $\mathcal{F}(\boldsymbol{x}_i; \boldsymbol{W})$ denotes the output of the hash network with image $\boldsymbol{x}_i$ as input, and $\boldsymbol{W}$ represents the set of parameters of hashing network; $cosh(a) = \frac{e^a + e^{-a}}{2}$; $sign(\cdot)$ is an element-wise sign function which returns 1 if the element is positive and returns -1 otherwise.

By minimizing Formula (8), its goal is to make $\frac{1}{l} \boldsymbol{b}_i^T \boldsymbol{b}_j = \boldsymbol{S}_{ij}$, i.e., it can make the hamming distance between two completely similar points as small as possible, and simultaneously make the hamming distance between two completely dissimilar points as large as possible. Meanwhile, it can make the partly similar images $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ have the suitable hamming distance complying with the similarity $\boldsymbol{S}_{ij}$.

However, the $sign(\cdot)$ function is in-differentiable at zero and the derivation of it will be zeros for a non-zero input. It means that the parameters of hashing model will not be updated with the back-propagation algorithm when minimizing the loss function $\mathcal{L}_1$. Thus, we directly discard the $sign(\cdot)$ function to ensure the parameters of our hashing model can be updated, and use $\tanh(\cdot)$ to approximate the $sign(\cdot)$ function to make each element of output of hashing network can be close to "+1" or "-1". Then the final objective function can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{W}} \mathcal{L} = & \sum_{i=1}^{n} \sum_{j=1}^{n} log(cosh(\frac{1}{l} \hat{\boldsymbol{b}}_i^T \hat{\boldsymbol{b}}_j - \boldsymbol{S}_{ij})) \\ & s.t. \quad \hat{\boldsymbol{b}}_i = \tanh(\mathcal{F}(\boldsymbol{x}_i; \boldsymbol{W})). \end{aligned} \tag{9}$$

3.5 Optimization

To optimize the proposed hashing model, we first construct the similarity matrix by using Formula (7), then minimize Formula (9) by using the mini-batch stochastic gradient descent (SGD) and update the parameter of hashing model with the back propagation (BP) algorithm. The details of the learning procedure are shown in Algorithm 1.

After the similarity matrix $\boldsymbol{S}$ is constructed, we use the mini-batch SGD method to learn the parameters $\boldsymbol{W}$ of hashing network. Specifically, we can calculate the gradient of the loss function $\mathcal{L}$ with regard to $\hat{\boldsymbol{b}}_i$ as follows:

$$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{b}}_i} = \frac{2}{l} \sum_{j=1}^{n} \tanh(\frac{1}{l} \hat{\boldsymbol{b}}_i^T \hat{\boldsymbol{b}}_j - \boldsymbol{S}_{ij})) \hat{\boldsymbol{b}}_j \tag{10}$$

where $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$.

Then by using the chain rule, the gradients of the loss function $\mathcal{L}$ with regard to the parameters $\boldsymbol{W}$ can be calculated as follows:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{b}}_i} \cdot \frac{\partial \hat{\boldsymbol{b}}_i}{\partial \boldsymbol{W}} \tag{11}$$

Algorithm 1 Learning algorithm for MLS^{3}RDUH

Input: Images $\boldsymbol{X}$, the length of hash codes k.
Output: Parameters of hashing network $\boldsymbol{W}$, hash codes $\boldsymbol{B}$.

```
Initialize parameters: W, α, k, o. learning rate: lr, iteration number: T, mini-batch size z (see Section 4.1).
Extract 4,096-dimensional deep features for images by Alexnet model which is pre-trained on ImageNet dataset.
Construct the semantic structure S by using Formula (7).
for i = 1 : T do
    for j = 1 : n/z do
        Randomly sample z image from database as a mini-batch.
        Generate b̂_i with image x_i as input by hash network.
        Update parameters of hash network W by Formula (11) with back propagation algorithm.
    end for
end for
Generate image hash codes B.
```

4 Experiments

In this section, we conduct extensive experiments on three commonly used image retrieval datasets to evaluate the proposed method against the state-of-the-art baselines.

4.1 Datasets and Settings

Three benchmark image retrieval datasets are used for evaluation, i.e., ***NUS-WIDE*** [Chua *et al.*, 2009], ***MS COCO*** [Lin *et al.*, 2014] and ***CIFAR10*** [Krizhevsky *et al.*, 2009], which are described below.

NUS-WIDE dataset contains 269,648 images crawled from Flickr. Each image is annotated with one or multiple labels from 81 concept labels. To ensure sufficient samples in each category, only 195,834 images that belong to the 21 most frequent concepts are selected for our experiment. We randomly sample 5,000 images as the test set and use the remaining images as the database, 10,500 images from the database as the training set.

MS COCO contains about 82,783 training images and 40,504 validation images, where each image is labeled by some of the 80 categories. After pruning images with no category information, we obtain 12,2218 images by combining the training and validation images. We randomly sample 5,000 images as the test set and use the other images as the database, 10,000 images from the database as the training set.

CIFAR10 is a popular image dataset which contains 60,000 images in 10 classes. We randomly sample 1,000 images as the test set and use the remaining images as the database, 5,000 images from the database as the training set.

Our proposed method is an unsupervised method, thus we compare our method with eight classical and state-of-the-art

Method	NUS-WIDE			MS COCO			CIFAR10		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
LSH	0.369	0.386	0.396	0.359	0.380	0.382	0.126	0.143	0.162
SH	0.412	0.402	0.418	0.377	0.381	0.383	0.173	0.178	0.182
PCAH	0.352	0.356	0.358	0.366	0.370	0.375	0.135	0.143	0.143
CBE	0.345	0.391	0.407	0.360	0.372	0.377	0.132	0.152	0.160
CNN+LSH	0.408	0.449	0.523	0.429	0.456	0.526	0.171	0.189	0.261
CNN+SH	0.571	0.551	0.565	0.487	0.510	0.535	0.280	0.284	0.295
CNN+PCAH	0.614	0.608	0.616	0.551	0.563	0.584	0.238	0.237	0.243
CNN+CBE	0.408	0.460	0.546	0.418	0.462	0.511	0.164	0.203	0.254
Deepbit	0.391	0.406	0.499	0.399	0.410	0.475	0.115	0.161	0.165
UTH	0.450	0.495	0.549	0.438	0.465	0.508	0.175	0.206	0.215
SSDH	0.580	0.593	0.610	0.540	0.562	0.586	0.262	0.271	0.280
DistillHash	0.627	0.656	0.671	0.546	0.566	0.593	0.285	0.294	0.308
MLS^{3}RDUH	**0.713**	**0.727**	**0.750**	**0.607**	**0.622**	**0.641**	**0.369**	**0.394**	**0.412**

Table 1: MAP of Hamming Ranking for Different Number of Bits on the Three Image Datasets.

unsupervised hashing methods: four traditional shallow unsupervised methods LSH [Gionis *et al.*, 1999], SH [Weiss *et al.*, 2009], PCAH [Wang *et al.*, 2010] and CBE [Yu *et al.*, 2014]; four deep unsupervised hashing methods: Deepbit [Lin *et al.*, 2016], UTH [Huang *et al.*, 2017], SSDH [Yang *et al.*, 2018] and DistillHash [Yang *et al.*, 2019]. The four traditional shallow unsupervised hashing methods use 512-dimensional GIST features of images as inputs for all the datasets, and the four deep hashing methods use the raw images as their inputs. For fair comparison, we adopt the AlexNet architecture [Krizhevsky *et al.*, 2012] for all the deep hashing methods. Moreover, we extract 4,096-dimensional deep features by Alexnet model which is pre-trained on ImageNet [Russakovsky *et al.*, 2015] dataset as the inputs of the four shallow hashing methods and denote them as LSH+CNN, SH+CNN, PCAH+CNN and CBE+CNN, respectively.

In our implementation of MLS^{3}RDUH, we utilize the AlexNet architecture [Krizhevsky *et al.*, 2012] and implement it based on Pytorch framework. The parameters in the first seven layers of hashing model are initialized with the parameters of the first seven layers of Alexnet which is pretrained on ImageNet, and the parameters in the eight layer of hashing model are initialized by Xavier initialization [Glorot and Bengio, 2010]. We use mini-batch stochastic gradient descent (SGD) with 0.9 momentum and the learning rate is fixed to 0.04. The iteration number is 150. We fix the mini-batch size of images as 128 and the weight decay parameter as 10^{-5}. We set k=0.06N, o=0.06N where N is the number of training datapoints, and following [Zhou *et al.*, 2004b], the hyper-parameters α is set as 0.99.

4.2 Evaluation Criterions

We evaluate the retrieval quality based on three evaluation metrics: Mean Average Precision (MAP), Precision curves with respect to the number of top returned results (P@N) and Precision-Recall curves (PR). The first two criterions are based on Hamming ranking which sorts the datapoints based the Hamming distance to the query datapoint. Specifically, MAP is one of the most widely-used criteria for evaluating retrieval accuracy. Given a query and a list of R ranked retrieval results, the average precision (AP) for this query can be calculated. MAP is the average APs of all queries. P@N is defined as the precision of the top N retrieved instances. In our experiments, following the settings in [Yang *et al.*, 2018], R is set to 5,000. and N is set to 1,000. PR curve is based on hash lookup which aims to return retrieval data in radius of a certain Hamming distance to the query datapoint.

Moreover, in our experiments, two images are considered as similar if they share share at least one common label, otherwise they are dissimilar.

4.3 Experimental Results

For Hamming ranking, the results of MAP for MLS^{3}RDUH and all the baselines on all the three datasets are shown in Table 1 with hash code numbers varying from 16 to 64, and P@N curves of the proposed method and all baselines over the three dataset on 64 bits are shown in Figure 4 (a), (c) and (e), respectively. In general, from Table 1 and Figure 4 (a), (c) and (e), three observations can be got: (1) Our proposed method outperforms all the baselines for different length of hash codes. For example, on NUS-WIDE dataset, comparing with the best traditional competitor PCAH+CNN on 64-bits, the MAP of MLS^{3}RDUH have a increases of 13.4%, and comparing with the best deep competitor DistillHash, MLS^{3}RDUH achieves a increase of 7.9%. Moreover, as shown in Figure 4, the P@N curves of the proposed method are better than all baselines on all the three datasets. (2) The performance of most shallow architecture methods with deep feature as input are better than the ones with hand-crafted features as input. For example, on MS COCO dataset, CNNH+SH which used deep features as input outperforms SH which whose inputs are hand-crafted features by 11% on 16 bits. (3) When there are not enough supervisory signals, deep hashing methods may not outperform the shallow architecture hashing methods with deep feature as input. For example, on CIFAR10 dataset, the MAP results of CNN+SH and CNN+PCAH which are shallow architecture hashing methods with deep feature as input are higher than the results of deep hashing methodes DeepBit and UTH.

For hash lookup, the PR curves of the proposed method and all baselines over the three dataset on 64 bits are shown in Figure 4 (b), (d) and (f). It can be found that the curves of MLS^{3}RDUH are higher than all the baselines' on the whole which demonstrates MLS^{3}RDUH outperforms all the base-

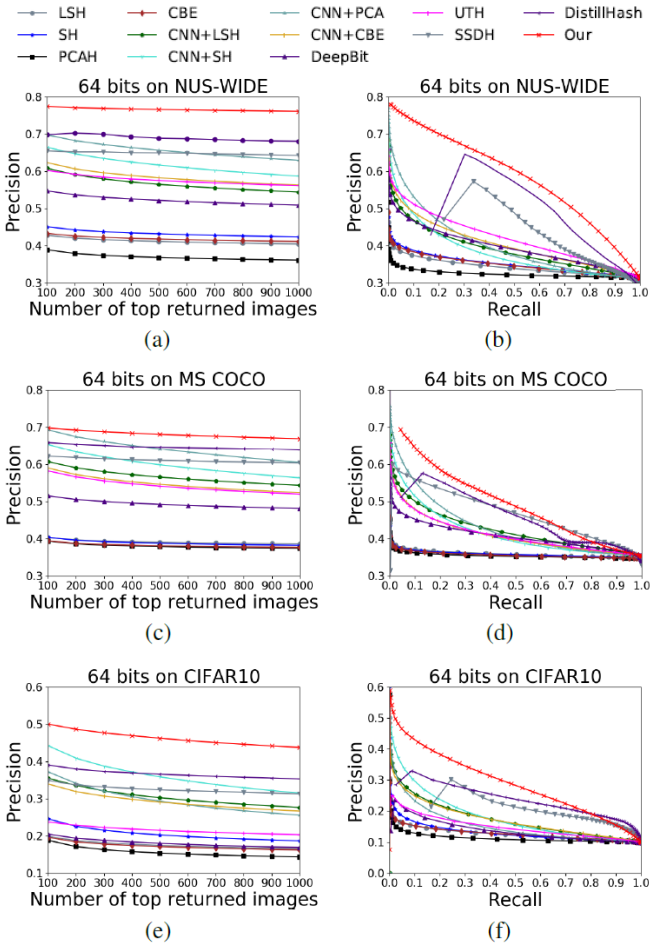

Figure 4: Precision-recall curves on the two datasets dataset

Method	NUS-WIDE	MS COCO	CIFAR10
MLS^{3}RDUH	**0.750**	**0.641**	**0.412**
MLS^{3}RDUH-1	0.691	0.596	0.349
MLS^{3}RDUH-2	0.687	0.573	0.328

Table 2: MAP Comparison of MLS^{3}RDUH and Its Variants for 64 bits on the Three Image Datasets.

lines in hash lookup.

4.4 Ablation Study

We investigate two variants of MLS^{3}RDUH: (1) MLS^{3}RDUH-1 is a variant of MLS^{3}RDUH that only used manifold based reconstructed local semantic similarity, i.e., $\widehat{\boldsymbol{S}}_{ij}$, as guiding information. (2) MLS^{3}RDUH-2 is a variant of MLS^{3}RDUH that only use the cosine similarity between datapoints, i.e., without the manifold based reconstructed local semantic similarity part as guiding information. The MAP results on 64 bits over three datasets are shown in Table 2. From the results, there are two points can be observed: (1) the manifold based reconstructed local semantic similarity can improve the retrieval performance. For example, MLS^{3}RDUH outperforms MLS^{3}RDUH-2 by 6.3%, 6.8% and 8.4% on NUS-WIDE dataset, MS COCO dataset and CIFAR10 dataset, respectively. (2) Deep features of images extracted by a pre-trained CNN contains semantic information which can be used to improve retrieval performance. For example, compared with MLS^{3}RDUH-1, MLS^{3}RDUH which use the cosine similarity of deep features as additional guiding information has a increases of 5.3%, 4.5% and 6.2% on NUS-WIDE dataset, MS COCO dataset and CIFAR10 dataset, respectively.

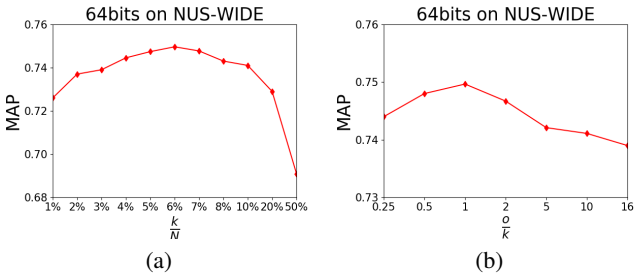

Figure 5: Precision-recall curves on the two datasets dataset

4.5 Sensitivity to Hyper-parameters

We investigate the influence of the hyper-parameters k and o. Figure 5 shows the effect of these two hyper-parameters over NUS-WIDE dataset on 64 bits. To investigate the influence of k, we fix $o = k$ and evaluate the MAP values of the proposed method by varying k from 0.01N to 0.5N where N is the number of training datapoints. The results are shown in Figure 5 (a). It can be found the performance first increases and then decreases as k varies, and can get a good performance in the range of [0.05N, 0.07N]. Then in order to investigate the influence of o, we set $k = 0.06N$ and vary the ratio between o and k from 0.25 to 16. The results are shown in 5 (b). It can be find when the ratio equals to 1, i.e., $k = o$, MLS^{3}RDUH can get the best performance. Then, for the proposed method, the parameters k and o are both set as $0.06N$.

5 Conclusion

In this paper, we have proposed a novel Deep Unsupervised Hashing via Manifold based Local Semantic Similarity Structure Reconstructing, called MLS^{3}RDUH. MLS^{3}RDUH first construct a novel similarity matrix by utilizing manifold and cosine similarity between datapoints to reconstruct semantic similarity structure, then a log-cosh loss is used to optimize the hashing model by incorporating the defined similarity matrix into the training process. Extensive experiments on three real-world public datasets have shown that the proposed MLS^{3}RDUH outperforms the state-of-the-art unsupervised baselines.

Acknowledgments

The work is supported by National Key R&D Plan (No. 2018YFB1005100), NSFC (No. 61772076, 61751201 and 61602197), NSFB (No. Z181100008918002), Major Project of Zhijiang Lab (No. 2019DH0ZX01), Open fund of BDAIGGCNEL, and CETC Big Data Research Institute Co., Ltd (No. w-2018018), and the funds of Beijing Advanced Innovation Center for Language Resources (No. TYZ19005).

References

[Cao *et al.*, 2017] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, pages 5608–5617, 2017.

[Chua *et al.*, 2009] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 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*, page 48. ACM, 2009.

[Gionis *et al.*, 1999] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[Girshick *et al.*, 2014] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[Gong *et al.*, 2012] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2012.

[Huang *et al.*, 2017] Shanshan Huang, Yichao Xiong, Ya Zhang, and Jia Wang. Unsupervised triplet hashing for fast image retrieval. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, pages 84–92. ACM, 2017.

[Huang *et al.*, 2019] Long-Kai Huang, Jianda Chen, and Sinno Jialin Pan. Accelerate learning of deep hashing with gradient attention. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5271–5280, 2019.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

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

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[Lin *et al.*, 2016] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1183–1192, 2016.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Tu *et al.*, 2019] Rong-Cheng Tu, Xian-Ling Mao, Bo-Si Feng, Bing-Bing Bian, and Yu-shu Ying. Object detection based deep unsupervised hashing. In *IJCAI*, pages 3606–3612, 2019.

[Wang *et al.*, 2010] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. 2010.

[Wang *et al.*, 2018] Dan Wang, Heyan Huang, Chi Lu, Bo-Si Feng, Liqiang Nie, Guihua Wen, and Xian-Ling Mao. Supervised deep hashing for hierarchical labeled data. In *AAAI*, pages 7388–7395, 2018.

[Weiss *et al.*, 2009] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2009.

[Yang *et al.*, 2018] Erkun Yang, Cheng Deng, Tongliang Liu, Wei Liu, and Dacheng Tao. Semantic structure-based unsupervised deep hashing. In *IJCAI*, pages 1064–1070, 2018.

[Yang *et al.*, 2019] Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, and Dacheng Tao. Distillhash: Unsupervised deep hashing by distilling data pairs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2946–2955, 2019.

[Yu *et al.*, 2014] Felix Yu, Sanjiv Kumar, Yunchao Gong, and Shih-Fu Chang. Circulant binary embedding. In *International conference on machine learning*, pages 946–954, 2014.

[Zhou *et al.*, 2004a] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*, pages 321–328, 2004.

[Zhou *et al.*, 2004b] Dengyong Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. Ranking on data manifolds. In *NIPS*, pages 169–176, 2004.