Semantic Structure-based Unsupervised Deep Hashing

Erkun Yang[1], **Cheng Deng**[1*], **Tongliang Liu**[2], **Wei Liu**[3], **Dacheng Tao**[2]
[1] School of Electronic Engineering, Xidian University, Xi'an 710071, China
[2] UBTECH Sydney AI Centre, SIT, FEIT, University of Sydney, Australia
[3] Tencent AI Lab, Shenzhen, China
ekyang@stu.xidian.edu.cn, chdeng.xd@gmail.com, tongliang.liu@sydney.edu.au,
wliu@ee.columbia.edu, dacheng.tao@sydney.edu.au

Abstract

Hashing is becoming increasingly popular for approximate nearest neighbor searching in massive databases due to its storage and search efficiency. Recent supervised hashing methods, which usually construct semantic similarity matrices to guide hash code learning using label information, have shown promising results. However, it is relatively difficult to capture and utilize the semantic relationships between points in unsupervised settings. To address this problem, we propose a novel unsupervised deep framework called Semantic Structure-based unsupervised Deep Hashing (SSDH). We first empirically study the deep feature statistics, and find that the distribution of the cosine distance for point pairs can be estimated by two half Gaussian distributions. Based on this observation, we construct the semantic structure by considering points with distances obviously smaller than the others as semantically similar and points with distances obviously larger than the others as semantically dissimilar. We then design a deep architecture and a pair-wise loss function to preserve this semantic structure in Hamming space. Extensive experiments show that SSDH significantly outperforms current state-of-the-art methods.

1 Introduction

The explosive growth of visual data (e.g., photos and videos) has led to renewed interest in efficient indexing and searching algorithms [Liu *et al.*, 2016; Andoni and Indyk, 2006; Weiss *et al.*, 2009; Gong *et al.*, 2013; Dai *et al.*, 2017; Ma *et al.*, 2017; You *et al.*, 2017a; Deng *et al.*, 2018; Li *et al.*, 2018]. However, since exact nearest neighbor searching is typically time-consuming and sometimes infeasible for big data applications, approximate nearest neighbor (ANN) searching [Andoni and Indyk, 2006], which balances retrieval performance and computational cost, has recently become more popular. Representative ANN solutions include tree-based [Guttman, 1984] and hash-based methods [Andoni and Indyk, 2006; Gui *et al.*, 2017; Liu *et al.*, 2014; Weiss *et al.*, 2009; Liu *et al.*, 2012]. In real applications, visual data are usually represented by high-dimensional features, e.g., SIFT-based bag-of-words features [Lowe, 2004] and deep features. Traditional tree-based methods are known to suffer from high feature space dimensionality, and their performance has been theoretically shown to degrade to that of the linear scan in many cases[Toth *et al.*, 2004]. Therefore, focus has shifted to hash-based methods for large-scale ANN searching.

Based on whether the hash learning processes are aware of the data distributions, hashing methods can be divided into data-independent hashing, also known as locality-sensitive hashing (LSH), and data-dependent hashing, also called learn-to-hash (L2H). The seminal LSH in [Andoni and Indyk, 2006] formulates a paradigm for the locality-sensitive hashing technique and guarantees that similar data points will be mapped to similar hash codes in Hamming space. However, since the LSH hash functions are usually generated independently, long hash bits are often needed to achieve a specific retrieval performance. Conversely, data-dependent methods [Weiss *et al.*, 2009; Liu *et al.*, 2011; 2011; Gong *et al.*, 2013; Dai *et al.*, 2017], which learn hash functions from data distributions, usually perform well with shorter binary codes, increasing their value.

For data-dependent methods, several studies [Lin *et al.*, 2016; Yang *et al.*, 2018] have focused on learning hash codes under supervised settings in which data labels (e.g., instance labels, pair-wise labels, or triplet labels) are assumed to be available. In these methods, the hash codes are usually learned to be consistent with these high-level supervised signals. However, manually labeling data is often time-consuming, laborious, and expensive, hindering the applications of these methods in practice. To make better use of widely-available unlabeled data, efforts have been made to exploit unsupervised hashing.

Most traditional unsupervised hashing methods are based on shallow architectures that consider feature learning and hash code learning as two separate processes. Representative unsupervised shallow methods include spectral hashing (SH) [Weiss *et al.*, 2009], anchor graph hashing (AGH) [Liu

[*]Corresponding author

et al., 2011], iterative quantization (ITQ) [Gong *et al.*, 2013], and stochastic generative hashing (SGH) [Dai *et al.*, 2017]. SH utilizes spectral graph partitioning to interpret the hash code learning and then relax the original problem by using a spectral method that can be solved efficiently. AGH attempts to approximate the data structure by constructing anchor graphs, which can also accelerate the computation of graph Laplacian eigenvectors. ITQ tries to map the data to the vertices of a zero-centered binary hypercube by finding an orthogonal rotation matrix that minimizes the quantization error. SGH adopts a generative approach to compress the dataset into hash codes using the minimum description length principle. The learned hash codes can also be used to regenerate the inputs. Though these methods have progressed the field, they usually depend on pre-defined features and cannot simultaneously optimize the feature and hash code learning processes, thus missing an opportunity to learn more effective hash codes.

Recently, deep learning has revolutionized computer vision, machine learning, and other related areas [You *et al.*, 2017b]. deep unsupervised hashing methods have also been proposed, which adopt deep architectures to extract features and perform hash mapping. Representative unsupervised deep hashing methods include semantic hashing [Salakhutdinov and Hinton, 2009], deep auto-encoder hashing [Krizhevsky and Hinton, 2011], and deep binary descriptors (DeepBit) [Lin *et al.*, 2016]. Semantic hashing uses pre-trained restricted Boltzmann machines (RBMs) to construct an auto-encoder network, which is then used to generate efficient hash codes and reconstruct the original inputs. Deep auto-encoder hashing designs very deep auto-encoders to map inputs to binary codes, and a reconstruction loss is also used to guide hash code learning. DeepBit considers original images and the corresponding rotated images as similar pairs and tries to learn hash codes to preserve this similarity. By integrating the feature and hash code learning processes, deep unsupervised hashing methods usually produce better results.

However, most existing unsupervised deep hashing methods such as semantic hashing, deep auto-encoder hashing, and DeepBit, usually learn hash codes based on the reconstruction loss or by preserving the similarity between rotated images. They cannot capture and utilize high-level semantic relationships between different data points. In reality, in supervised settings, most deep methods [Yang *et al.*, 2017] leverage the supervised signals to construct a semantic similarity matrix and learn hash codes to preserve this semantic relationship. However, these methods are unsuitable for unsupervised settings.

Here we ask: can we learn semantic structures without labels? Our approach is inspired by recent studies [Girshick *et al.*, 2014], which show that features extracted from pre-trained deep architectures contain rich semantic information. We empirically analyze the deep feature statistics for the NUSWIDE and FLICKR25K datasets and find that the distribution of the cosine distance for different data points can be estimated by two half Gaussian distributions. Since deep features contain rich semantic information, we assume that data points with distances obviously smaller than the others are semantically similar and data points with distances obviously larger than the others are semantically dissimilar. Based on the above observation and assumptions, we construct a semantic structure that denotes whether data point pairs are similar. We then design a pair-wise loss function to preserve this semantic information. To integrate the feature and hash code learning processes, we design a unified, end-to-end trainable deep framework. Extensive experiments demonstrate that our proposed method significantly outperforms the current state-of-the-art.

2 Notation and Problem Definition

In this paper, boldface uppercase letters (such as $\boldsymbol{A}$) are used to denote matrices. $\boldsymbol{A}_{*i}$ denotes the i-th column of $\boldsymbol{A}$. A_{ij} denotes the (i,j)-th element of $\boldsymbol{A}$. Further, $\|\boldsymbol{A}\|_F$ and $\boldsymbol{A}^\top$ are used to denote the Frobenius norm and the transpose of matrix $\boldsymbol{A}$, respectively. The $\odot$ symbol is used to denote the Hadamard product and sign$(\cdot)$ represents the element-wise signum function as:

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ -1 & \text{if } x < 0. \end{cases} \quad (1)$$

Assume that we have m database points denoted $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^m$. The goal of unsupervised hashing is to learn binary hash codes for database points and a hash function that can be used to generate hash codes for query data points. We use $\boldsymbol{B} = \{\boldsymbol{b}_i\}_{i=1}^m$ to denote the hash codes for $\boldsymbol{X}$, and $\boldsymbol{b}_i \in \{-1,+1\}^K$ corresponds to the hash code for point $\boldsymbol{x}_i$, where K denotes the hash code length.

3 Model Formulation

In this section, we introduce our Semantic Structure-based unsupervised Deep Hashing (SSDH) approach in detail, including its network architecture, semantic structure learning, and hash code learning.

3.1 Network Architecture

We apply the VGG-F model from [Simonyan and Zisserman, 2014] to perform feature and hash code learning. This model has been used in many other hashing methods [Lin *et al.*, 2016; Yang *et al.*, 2018]. The original VGG-F model contains five convolutional layers (*conv1 - conv5*) and three fully-connected layers (*fc6-fc8*), the details of which can be found in [Simonyan and Zisserman, 2014]. We modify the original VGG-F model by replacing the last fully-connected layer with a fully-connected layer with k hidden units to incorporate the hash code learning process in this deep learning framework. Note that we adopt VGG-F network for illustration purposes only, and any other network can easily be integrated into our framework.

3.2 Semantic Structure Learning

Since hashing methods usually aim to map the original points into Hamming space where the semantic relationship across different data points can be preserved, constructing a semantic similarity matrix using label information in supervised settings is natural. However, for unsupervised hashing learning, it is difficult to capture the semantic structure without

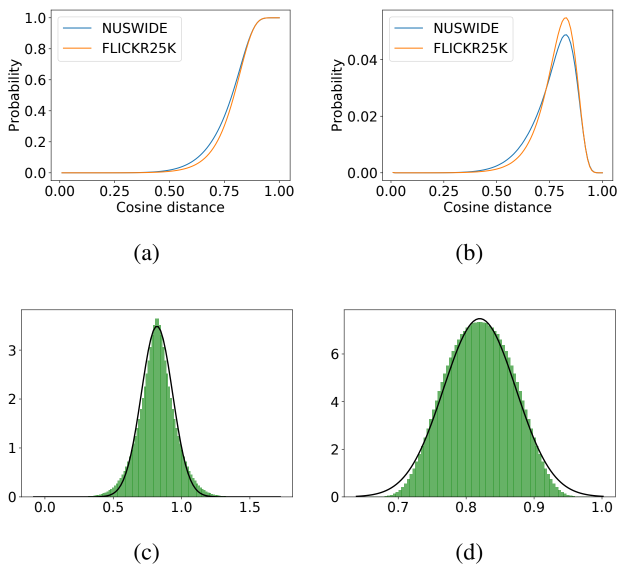


Figure 1: Statistics of the cosine distance. (a) Cumulative distribution. (b) Corresponding histogram distribution. (c) Gaussian estimation for the histogram distribution of the left part. (d) Gaussian estimation for the histogram distribution of the right part. For better visualization, we complete (c) and (d) to make them symmetrical.

any supervised information. To address this problem, and inspired by recent studies [Girshick *et al.*, 2014] showing that features extracted from pre-trained deep architectures contain rich semantic information, we first extract features from a pre-trained deep architecture and analyze their statistics. Based on this analysis, we can then learn a semantic structure that explicitly captures the semantic relationships across different data points.

Specifically, we first randomly sample 10,000 data points from the FLICKR and NUSWIDE datasets and then extract their fc7-layer features from the VGG-F network pre-trained with ImageNet [Deng *et al.*, 2009]. To analyze the semantic relationships between these data points, we calculate the cosine distance for each pair of points based on the extracted deep features. Figure 1(a) shows the distance cumulative histogram, and Figure 1(b) is the corresponding distance histogram over all data point pairs. It can be seen that most data point pairs have relatively large distances, and each distance histogram is similar to two half Gaussian distributions. Since deep features contain rich semantic information, we assume that data points with distances obviously smaller than the others are semantically similar and data points with distances obviously larger than the others are semantically dissimilar.

Based on the above observation and assumptions, we split the distance histogram into two parts based on the maximum value and fit each part with a half Gaussian distribution. The results for the NUSWIDE dataset are shown in Figures 1(c) and 1(d), which clearly show that the distance histogram can be approximately estimated by two half Gaussian distributions. We use m_l and σ_l to denote the mean and standard deviation of the Gaussian distribution for the left part and m_r and σ_r for the right part. To obtain the semantically similar

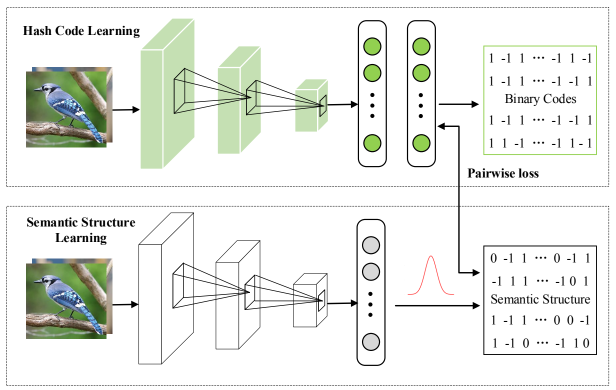


Figure 2: The framework of SSDH.

data points, we set a distance threshold $d_s = (m_l - \alpha\sigma_l)$, and consider data points with distance smaller than d_s as semantically similar data points. α is a hyper-parameter to control the value of the distance threshold d_s, and it also dicates the percentage of similar points from all data point pairs. For example, if we only consider points from the left part and choose α as 2, then from the properties of Gaussian distributions we know that only about 2.2% of the distances are smaller than $(m_l - 2\sigma_l)$. To obtain semantically dissimilar data points, we set a distance threshold as $d_d = (m_r + \beta\sigma_r)$, and consider data points with distance larger than d_s as semantically dissimilar. β is a hyper-parameter to control the value of the distance threshold d_d, which also dictates the percentage of dissimilar points.

Based on the above two thresholds, we construct a semantic structure $\boldsymbol{S}$ as:

$$S_{ij} = \begin{cases} 1, & \text{if } d(i,j) \leq d_s, \\ 0, & \text{if } d_s < d(i,j) < d_d, \\ -1, & \text{if } d(i,j) \geq d_d, \end{cases} \tag{2}$$

where $d(i,j)$ represents the cosine distance of the deep representations for points x_i and x_j, and S_{ij} is set to 1 if x_i and x_j are semantically similar, and -1 if they are semantically dissimilar. If we cannot decide whether they are semantically similar or dissimilar, we set S_{ij} to 0.

3.3 Hash Code Learning

In order to preserve the learned semantic structure, we try to map semantically similar data points into similar hash codes, semantically dissimilar data points into dissimilar hash codes, and discard pairs of data points if we cannot decide whether they are semantically similar or dissimilar. We first construct a similarity structure $\boldsymbol{H}$ by using the inner product of different hash codes:

$$H_{ij} = \frac{1}{K}\boldsymbol{b}_i \top \boldsymbol{b}_j, \quad \boldsymbol{b}_i \in \{-1, +1\}^K, \tag{3}$$

where $\boldsymbol{b}_i$ is the hash code for data point $\boldsymbol{x}_i$. To explicitly enforce semantic structure-preserving criteria, we minimize the L_2 loss between this similarity structure and the learned semantic structure, which can be formulated as:

$$\min_{\boldsymbol{B}} \mathcal{J}(\boldsymbol{B}) = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} |S_{ij}| \left(H_{ij} - S_{ij}\right)^2. \qquad (4)$$

We set $\boldsymbol{b}_i = \text{sign}(F(\boldsymbol{x}_i;\Theta))$, where $F(\boldsymbol{x}_i;\Theta)$ denotes the output of the neural network, and Θ is the parameters. Therefore we can integrate the above loss function into the deep architecture. However, binary values make standard back-propagation for the deep neural network infeasible, which is known as the *ill-posed gradient* problem. So, here we relax the binary constraint, use $\tanh(\cdot)$ to approximate the $\text{sign}(\cdot)$ function, and adopt the following objective function:

$$\begin{aligned} \min_{\Theta} \mathcal{J}(\Theta) &= \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} |S_{ij}| \left(H_{ij} - S_{ij}\right)^2, \\ s.t. \quad H_{ij} &= \frac{1}{K} \boldsymbol{v}_i \top \boldsymbol{v}_j, \quad \boldsymbol{v}_i = \tanh(F(\boldsymbol{x}_i;\Theta)), \end{aligned} \qquad (5)$$

where we use $\boldsymbol{v}_i$ to denote the relaxed binary representation.

4 Optimization

To optimize the problem in Equation 5, we first extract 10,000 samples from the dataset before constructing the semantic structure using Equation 2. Then, we minimize Equation 5 using the mini-batch stochastic gradient descent (SGD) method. The whole learning procedure is summarized in Algorithm 1.

When S is given, we use the mini-batch SGD method to learn the neural network parameter Θ. Specifically, we first sample a mini-batch of the training data points and then update parameter Θ based on this mini-batch data. For the sake of simplicity, we define $\boldsymbol{z}_i = F(\boldsymbol{x}_i;\Theta)$. Then we calculate the gradient of the loss function with regard to $\boldsymbol{z}_i$ as:

$$\begin{aligned} \frac{\partial \mathcal{J}(\Theta)}{\partial \boldsymbol{z}_i} = & \frac{2}{Km^2} \sum_{i=1}^{m} (1 - \boldsymbol{v}_i^2) \odot (\sum_{j=1}^{m} |S_{ij}| (\frac{1}{K} \boldsymbol{v}_i \top \boldsymbol{v}_j - S_{ij}) \\ & \cdot \boldsymbol{v}_j + (\frac{1}{K} \boldsymbol{v}_i \top \boldsymbol{v}_i - 1) \boldsymbol{v}_i). \end{aligned} \qquad (6)$$

The gradients for loss function $\mathcal{J}(\Theta)$ with regard to other parameters can be readily computed using the chain rule, and all the parameters can be updated using mini-batch SGD.

We performed experiments to update the semantic structure based on updating image features in each epoch. Constructing this semantic structure was time-consuming, and the update strategy did not significantly impact performance. So, in this work, we first obtain the semantic structure and then fix it when updating the the neural network parameters.

5 Out of Sample Extension

For any point $\boldsymbol{q}_i$ not in the training set, we can obtain its hash code $\boldsymbol{b}_i$ by directly forward propagating it through the learned neural network as follows:

$$\boldsymbol{b}_i = \text{sign}(F(\boldsymbol{q}_i;\Theta)). \qquad (7)$$

Algorithm 1: SSDH

```
Training Stage
Input: Training images $\mathbf{X}$, code length $K$, parameters $\alpha$ and $\beta$, mini-batch size $N$.
Output: Parameters $\Theta$ for the neural network and hash codes $\boldsymbol{B}$ for training images.
Procedure:
1. Initialize parameters for the VGG-F network $\Theta$.
2. Extract their fc7-layer features based on the VGG-F network.
3. Construct the semantic structure $S$ by using Equation 2.
repeat
    3.1 Randomly sample $N$ points from $\boldsymbol{X}$ to construct a mini-batch.
    3.2 Calculate the outputs by forward-propagating through the VGG-F network.
    3.3 Update parameters of the VGG-F network by (6).
until convergence;
Testing Stage
Input: Image query $\mathbf{q}_i$, parameters for the VGG-F network.
Output: Hash codes for the query.
Procedure:
1. Calculate the output of the neural network by directly forward-propagating the input images.
2. Calculate the hash codes by using Equation (7).
```

6 Experiments

We evaluate our method on two popular benchmark datasets, **NUSWIDE** and **FLICKR25K** . We provide extensive evaluations of the proposed hash codes and demonstrate their performance. We first introduce the datasets and then present and discuss our experimental results together with comparative evaluations with current state-of-the-art methods.

6.1 Datasets

- **NUSWIDE** contains 269,648 images, with each data point annotated with multiple labels based on 81 concepts. The subset belonging to the 10 most popular concepts are used here. We randomly select 5,000 images as a test set. The remaining images are used as a retrieval set, from which we randomly select 5,000 images as a training set.
- **FLICKR25K** contains 25,000 images collected from the Flickr website. Each image is manually annotated with at least one of 24 unique provided labels. We randomly select 2,000 images as the test set. The remaining images are used as the retrieval set, from which we randomly select 10,000 images as the training set.

For our proposed approach and deep learning-based comparator methods, we use raw pixels as inputs. For traditional shallow methods, we extract 4096-dimensional feature vectors from the *fc7* layer using the VGG-F architecture pre-trained on ImageNet for fair comparison.

Method	FLICKR25K			
	16 bits	32 bits	64 bits	128 bits
ITQ	0.6492	0.6518	0.6546	0.6577
SH	0.6091	0.6105	0.6033	0.6014
DSH	0.6452	0.6547	0.6551	0.6557
SpH	0.6119	0.6315	0.6381	0.6451
SGH	0.6362	0.6283	0.6253	0.6206
DeepBit	0.5934	0.5933	0.6199	0.6349
Ours	**0.7240**	**0.7276**	**0.7377**	**0.7343**

Table 1: Comparison with baselines in terms of MAP. The highest accuracy is shown in bold.

Method	NUSWIDE			
	16 bits	32 bits	64 bits	128 bits
ITQ	0.5270	0.5241	0.5334	0.5398
SH	0.4350	0.4129	0.4062	0.4100
DSH	0.5123	0.5118	0.5110	0.5267
SpH	0.4458	0.4537	0.4926	0.5000
SGH	0.4994	0.4869	0.4851	0.4945
DeepBit	0.3844	0.4341	0.4461	0.4917
Ours	**0.6374**	**0.6768**	**0.6829**	**0.6831**

Table 2: Comparison with baselines in terms of MAP. The highest accuracy is shown in bold.

6.2 Implementation Details

We implement our approach using the open source TensorFlow [Abadi *et al.*, 2016], and optimize our model by mini-batch SGD with momentum. The mini-batch size is set to 24 and momentum to 0.9. Training images are resized to 224×224 as the inputs. The first seven layers of our neural network are fine-tuned from the model pre-trained with ImageNet, and the last fully-connected layer is learnt from scratch. The learning rate is fixed at 0.001.

6.3 Protocols and Baseline Methods

Three evaluation criteria are adopted to evaluate our method's performance, namely mean average precision (MAP), precision-recall, and TopN-precision.

MAP is one of the most widely-used criteria to evaluate retrieval accuracy. Given a query and a list of R ranked retrieval results, the average precision (AP) for this query is defined as

$$AP = \frac{1}{N}\sum_{r=1}^{R} P(r)\delta(r), \tag{8}$$

where N is the number of ground-truth relevant instances in the database for the query, and $P(r)$ represents the precision for the top r retrieved instances. $\delta(r) = 1$ when the r-th retrieval instance is relevant to the query, otherwise $\delta(r) = 0$. MAP is defined as the average of APs for all queries. R is set to 5,000 in our experiments. The ground-truth relevant instances for a query are defined as those sharing at least one label with the query. Precision-recall reports recall and the corresponding precision values. The TopN-precision denotes the precision at different numbers of retrieved instances.

We compare SSDH with several other unsupervised hashing methods including ITQ [Gong *et al.*, 2013], SH [Weiss *et al.*, 2009], DSH [Jin *et al.*, 2014], SpH [Heo *et al.*, 2012], DeepBit [Lin *et al.*, 2016], and SGH [Dai *et al.*, 2017], where ITQ, SH, DSH, SpH and SGH are shallow architecture-based methods, and DeepBit is a deep architecture-based method. All codes of these methods are kindly provided by the authors, and configurations for all methods are set according to the original papers.

6.4 Results and Discussion

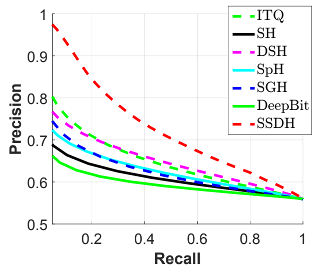
As a global evaluation, we first present the MAP values for SSDH and all comparative methods across different hash code lengths on the two datasets. We then draw precision-recall and TopN-precision curves for SSDH and all baseline methods with 32 and 64 hash code lengths as a more comprehensive comparison.

Table 1 shows the MAP results for SSDH and all comparative methods on FLICKR25K with code lengths varying from 16 to 128. It can be seen that the proposed method consistently obtains the best results across different hash bit lengths. Specifically,the SSDH MAP obtains a relative improvement over the next-best baseline methods of 7.48%, 7.29%, 8.26%, and 7.66% for 16, 32, 64, and 128 bit lengths, respectively. Table 2 shows the MAP results for all methods on NUSWIDE. Again, our proposed method outperforms all other baseline methods in all cases. Specifically, the MAP of our proposed method obtains a relative improvement over the next-best baseline methods of 11.04%, 15.27%, 14.95%, and 14.33% for 16, 32, 64, and 128 bit lengths, respectively. The results presented in Table 1 and 2 well demonstrate the superiority of our proposed method.

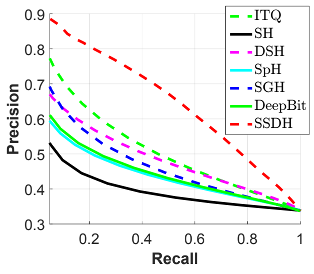
Figures 3(c), 3(d), 4(c), and 4(d) show TopN-precision curves of SSDH and all comparative methods on FLICKR25K and NUSWIDE, respectively. It can clearly be seen that SSDH always achieves the highest precision. Since MAP values and TopN-precision curves are all based on Hamming ranking, taken together the above analysis shows that SSDH can achieve superior performance for Hamming ranking-based evaluations. To illustrate hash lookup results, we plot the recall-precision curves for SSDH and other comparative methods, as shown in Figures 3(a), 3(b), 4(a), and 4(b), where Figures 3(a) and 3(b) present the 32 bit length results and Figures 4(a) and 4(b) presents the 64 bit length results. It can again be seen that SSDH always achieves the best performance by a large margin, further demonstrating the superiority of our proposed method.

6.5 Parameter Sensitivity

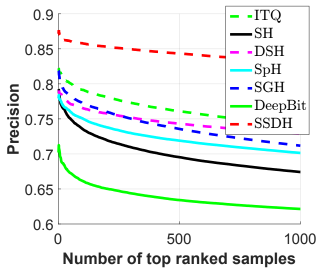
We next investigate the sensitivity of hyper-parameters α and β. Figure 5 shows the effect of these two hyper-parameters in SSDH on NUSWIDE dataset with hash code lengths of 32 and 64 bits. We first fix β to 1 and evaluate the MAP values by varying α between 0 and 3, the results are presented in Figure 5(a). SSDH performance first increases and then decreases as α varies, showing a desirable bell-shaped curve. The best result is obtained when α is 2, so we fix α to 2 in our other experiments. According to a Gaussian distribution, only about 2.2% points are smaller than the threshold $(m_l - 2\delta_l)$, consistent setting data points with distances obviously smaller than the others as semantically similar points.



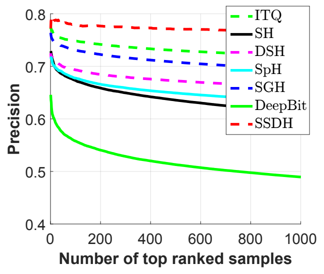
(a) PR curves on FLICKR25K



(b) PR curves on NUSWIDE

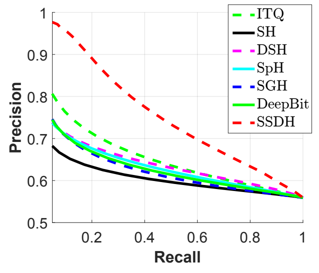


(c) TopN on FLICKR25K

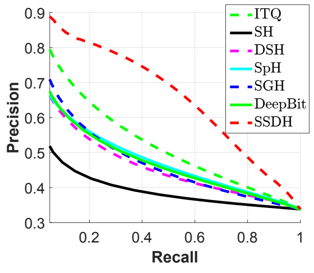


(d) TopN on NUSWIDE

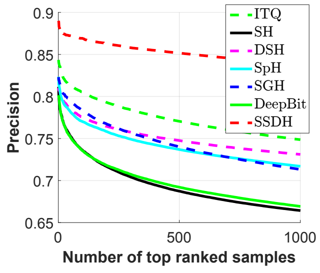
Figure 3: Precision-recall curves and topN-precision with code length 32.



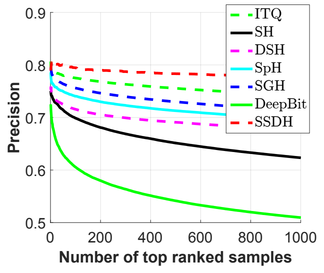
(a) PR curves on FLICKR25K



(b) PR curves on NUSWIDE



(c) TopN on FLICKR25K



(d) TopN on NUSWIDE

Figure 4: Precision-recall curves and topN-precision with code length 64.

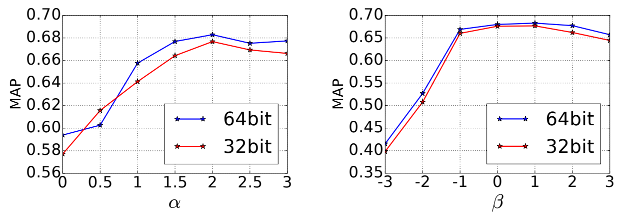


Figure 5: Hyper-parameters on NUSWIDE dataset.

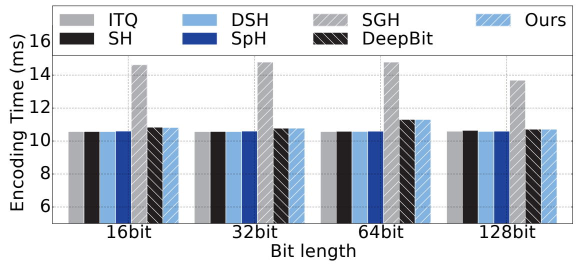


Figure 6: Encoding time on NUSWIDE with code length 32.

Figure 5(b) shows the MAP values when varying β between -3 and 3, The performance of SSDH first increases and then is kept at a relatively high level. The result is not sensitive to β in the range of $0 \leq \beta \leq 3$. For other experiments in this paper, we select β as 1.

6.6 Encoding Time

In this subsection, we compare the encoding time. For shallow methods, the overall time includes the time used for feature extraction. The results are shown in Figure 6, from which we can see that SSDH takes almost the same amount of time as DeepBit and both these two methods have comparative encoding time with ITQ, SH, DSH, and SpH. From the results, we can get that deep learning based methods enable as efficient encoding as most shallow methods.

7 Conclusion

Here we present a novel deep unsupervised hashing method called semantic structure-based unsupervised deep hashing (SSDH). Our empirical study of the deep feature statistics shows that the cosine distance histogram distributions can be estimated by two half Gaussian distributions. Based on this observation, we construct a semantic structure to capture the semantic relationships and design a pair-wise loss function to preserve the relationships in Hamming space. Comparative studies on two bench-mark datasets show that SSDH significantly outperforms current state-of-the-art methods.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61572388 and Grant 61703327, in part by the Key R&D Program-The Key Industry Innovation Chain of Shaanxi under Grant 2017ZDCXL-GY-05-04-02 and Grant 2017ZDCXL-GY-05-04-02, and in part by ARC projects: FL-170100117, DP-140102164, and LP-150100671.

References

[Abadi *et al.*, 2016] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[Andoni and Indyk, 2006] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.

[Dai *et al.*, 2017] Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. Stochastic generative hashing. *arXiv preprint arXiv:1701.02815*, 2017.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[Deng *et al.*, 2018] Cheng Deng, Zhaojia Chen, Xianglong Liu, Xinbo Gao, and Dacheng Tao. Triplet-based deep hashing network for cross-modal retrieval. *IEEE Transactions on Image Processing*, 2018.

[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.

[Gong *et al.*, 2013] 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, 2013.

[Gui *et al.*, 2017] Jie Gui, Tongliang Liu, Zhenan Sun, Dacheng Tao, and Tieniu Tan. Fast supervised discrete hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[Guttman, 1984] Antonin Guttman. *R-trees: A dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[Heo *et al.*, 2012] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2957–2964. IEEE, 2012.

[Jin *et al.*, 2014] Zhongming Jin, Cheng Li, Yue Lin, and Deng Cai. Density sensitive hashing. *IEEE transactions on cybernetics*, 44(8):1362–1371, 2014.

[Krizhevsky and Hinton, 2011] Alex Krizhevsky and Geoffrey E Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011.

[Li *et al.*, 2018] Chao Li, Cheng Deng, Ning Li, Wei Liu, Xinbo Gao, and Dacheng Tao. Self-supervised adversarial hashing networks for cross-modal retrieval. *arXiv preprint arXiv:1804.01223*, 2018.

[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.

[Liu *et al.*, 2011] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1–8, 2011.

[Liu *et al.*, 2012] W. Liu, J. Wang, R. Ji, Y. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

[Liu *et al.*, 2014] Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. Collaborative hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2139–2146, 2014.

[Liu *et al.*, 2016] Xianglong Liu, Cheng Deng, Bo Lang, Dacheng Tao, and Xuelong Li. Query-adaptive reciprocal hash tables for nearest neighbor search. *IEEE Transactions on Image Processing*, 25(2):907–919, 2016.

[Lowe, 2004] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[Ma *et al.*, 2017] Chao Ma, Chen Gong, Yun Gu, Jie Yang, and Deying Feng. Shiss: Supervised hashing with informative set selection. *Pattern Recognition Letters*, 2017.

[Salakhutdinov and Hinton, 2009] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

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

[Toth *et al.*, 2004] Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. CRC press, 2004.

[Weiss *et al.*, 2009] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[Yang *et al.*, 2017] Erkun Yang, Cheng Deng, Wei Liu, Xianglong Liu, Dacheng Tao, and Xinbo Gao. Pairwise relationship guided deep hashing for cross-modal retrieval. In *AAAI*, 2017.

[Yang *et al.*, 2018] Erkun Yang, Cheng Deng, Chao Li, Wei Liu, Jie Li, and Dacheng Tao. Shared predictive cross-modal deep quantization. *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

[You *et al.*, 2017a] Shan You, Chang Xu, Yunhe Wang, Chao Xu, and Dacheng Tao. Privileged multi-label learning. *arXiv preprint arXiv:1701.07194*, 2017.

[You *et al.*, 2017b] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1285–1294. ACM, 2017.