Sketch-Based Image Retrieval via Compact Binary Codes Learning

Xinhui Wu(✉) and Shuangjiu Xiao

School of Software, Shanghai Jiao Tong University, Shanghai, China
kexuan@sjtu.edu.cn, xsjiu99@cs.sjtu.edu.cn

Abstract. With the exploding number of images on the Internet and the convenience of free-hand sketch drawing, sketch-based image retrieval (SBIR) has attracted much attention in recent years. Due to the ambiguity and sparsity of sketches, SBIR is more challenging to cope with than conventional content-based problem. Existing approaches usually adopt high-dimensional features which require high-computational cost. Furthermore, they often use edge detection and parameter-sharing networks which may lose important information in training. In this study, we propose a compact binary codes learning strategy using deep architecture. By leveraging well-designed prototype hash codes, we embed different domains input (sketch and photo) into a common comparable feature space. Besides, we present two separate networks specific to sketches and real photos which can learn very compact features in Hamming space. Our method achieves state-of-the-art results in accuracy, retrieval time and memory cost on two standard large-scale datasets.

Keywords: Deep learning · Hashing · Sketch-based image retrieval

1 Introduction

Sketches are highly abstract representations which express sufficient stories. Different from natural images, they are formed of a few hand-drawn strokes. Humans can draw simple sketches quickly without any reference, at the same time conveying information precisely. With such interesting characteristics, there exists much research dealing with sketch-based image retrieval [6,17,20], sketch-based 3D model retrieval [23] and sketch recognition [28].

In this paper, our research direction focuses on sketch-based image retrieval (SBIR). It aims at retrieving most similar results in image gallery collection by a query free-hand sketch. Figure 1 gives an example of retrieval flow. SBIR can solve the situation when it is hard to describe an object in words or query image is not available. In this situation, text-based image retrieval (TBIR) and content-based image retrieval (CBIR) [5,25,26] fail. Essentially, SBIR has two main advantages: (i) Science proves that people are sensitive to outlines [11,29]. Free-hand sketches can show enough key query points without noisy background. (ii) With the appearance of touch-screen mobile devices in recent years, drawing

L. Cheng et al. (Eds.): ICONIP 2018, LNCS 11306, pp. 294–306, 2018.
https://doi.org/10.1007/978-3-030-04224-0_25

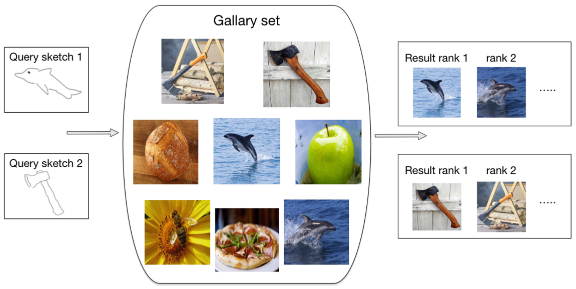

Fig. 1. An illustration of sketch-based image retrieval.

sketches becomes quite convenient. Even non-artists could draw sketches within few seconds.

However, SBIR confronts several challenges. First, sketches lack texture and color information. The feature of sparse lines is totally different with traditional images. Second, since people depict query sketch without reference as aforesaid, sketches usually exhibit large intra-class variations.

Over the past 25 years, the study on SBIR has developed rapidly [20]. Among them, the bulk of methods exploit traditional hand-crafted pipeline [6,18,19]. They usually first transform real images to detected edgemap photos in order to narrow the semantic gap between the sketch and the image. Then, hand-crafted features of both sketches and edgemap photos are extracted and fed into bag-of-words architecture. Whereas, their shallow features cannot handle large internal variations well. Recently, convolutional neural networks (CNN) [10] has shown great power on deep feature representation. By means of robust end-to-end deep frameworks, the deep methods [17,20] are superior to hand-crafted ones typically. Actually, the deep learning methods of SBIR come out much later than those in CBIR due to the lack of available fine-grained sketch dataset. Since 2016, the appearance of Sketchy [20] dataset boosts the development of SBIR. Though deep methods have achieved progress, they mainly calculate feature distances in Euclidean space with high complexity. It is not feasible when dealing with large-scale retrieval task. Hence, we introduce a deep hashing architecture to perform fast retrieval in Hamming space with low memory cost.

SBIR problem is a typical cross-domain retrieval case. To address the issue of aforementioned sketch challenges, previous works generally translate the real image to the approximate sketch in advance. However, salient structural information may be lost during edge extraction process. In addition, most of the existing methods adopt shared Siamese network. However, learning parameters individually will perform better if possible. Besides, previous works generally compare features in high-dimensional space requiring high-computational cost and long retrieval time.

In this paper, we present a novel deep hashing framework to solve sketch-based image retrieval. The main contributions of our work include: (i) Encoding supervised information into a semantic-preserving set of prototype hash codes to achieve better guide for deep training process; (ii) Presenting a deep hashing framework with two separate networks for sketches and photos, which is more suitable for cross-domain challenge, capturing the internal semantic relationship and cross-domain similarities at the same time; (iii) Achieving state-of-the-art results in accuracy, retrieval time and memory cost compared with existing methods on two large datasets.

2 Related Work

Sketch-Based Image Retrieval. Prior hand-crafted methods first use edge detectors like Canny [1] to generate edge or contour maps from real images. After that, they extract features of both sketches and generated edgemap photos, such as SIFT [15], HOG [2], SSIM [21], Gradient Field HOG [6] etc. Then, the bag-of-words framework is used to learn discriminative semantic representation. With the help of CNN, deep methods achieve better performance recently on category-level [13,17] and fine-grained SBIR [27]. Wang et al. [23] use a Siamese network to retrieve 3D models by a query sketch. Qi et al. [17] also adopt a similar Siamese strategy to solve category-level SBIR based on a small dataset Flickr15K [6], which is the first attempt in deep SBIR technique. Yu et al. [27] achieve a nice result in fine-grained search with triplet loss. To our best knowledge, DSH [13] is the only existing work that employs deep hash learning in SBIR. It uses a relatively complex semi-heterogeneous hashing framework, achieving a good performance in large-scale dataset. Despite that, our work surpasses DSH in evaluation with rather compact hash code learning.

Hashing Learning. Hashing is an effective method for fast image retrieval. It projects high-dimensional features to compact semantic-preserving binary codes, which are called hash codes. The mapping strategy is the crucial hashing function. Early unsupervised hashing methods include LSH [3], SH [24] and ITQ [5]. With the help of label information, supervised hashing can deal with more complicated semantics than unsupervised hashing. The representative ones are BRE [8], MLH [16] and KSH [14]. In recent years, deep hashing methods have shown promising power. CNNH [26], DPSH [12] and NINH [9] are representative methods. They leverage pair-wise or triplet-wise approaches to learn semantic similarity, while large storage of pair or triplet samples is required. Moreover, above-mentioned hashing methods are devoted to CBIR, which have not been specially designed for SBIR yet.

3 Methodology

3.1 Problem Formulation

We let $\mathcal{P} = \{p_i\}_{i=1}^{n_1}$ be the set of all real image photos and $\mathcal{S} = \{s_i\}_{i=1}^{n_2}$ be the set of all sketches, where n_1 and n_2 are the sample number of set $\mathcal{P}$ and

$\mathcal{S}$ respectively. The corresponding label set of real photos is denoted as $\mathcal{L}^{\mathcal{P}} = \{l_i^{\mathcal{P}}\}_{i=1}^{n_1}$, $l_i^{\mathcal{P}} \in \{1, 2, ..., C\}$. Each photo p_i is connected to one label tag $l_i^{\mathcal{P}}$ coming from total C classes. Similarly, we have the label set of sketches $\mathcal{L}^{\mathcal{S}} = \{l_i^{\mathcal{S}}\}_{i=1}^{n_2}$, $l_i^{\mathcal{S}} \in \{1, 2, ..., C\}$.

Our target is to learn a semantic-preserving hashing function mapping original photos $\mathcal{P}$ and sketches $\mathcal{S}$ to compact hash codes $\mathcal{B}^{\mathcal{P}} = \{b_i^{\mathcal{P}}\}_{i=1}^{n_1}$ and $\mathcal{B}^{\mathcal{S}} = \{b_i^{\mathcal{S}}\}_{i=1}^{n_2}$. These hash codes are d-bit binary representation $b_i^{\mathcal{P}}, b_i^{\mathcal{S}} \in \{0, 1\}^d$ that well preserving intrinsic semantics. For our particular task, we need to bridge the domain gap between real photos and sketches, in the meantime we should maintain the similarity relationship in original feature space for both domains themselves. More specifically, given two images, no matter which domain they belong to (perhaps a sketch or a natural photo): (i) If their corresponding labels are the same, they should be semantically similar all the way. In other words, the hamming distance of their hash codes has to be quite small. (ii) Otherwise, the distance between their codes should be pushed away as far as possible.

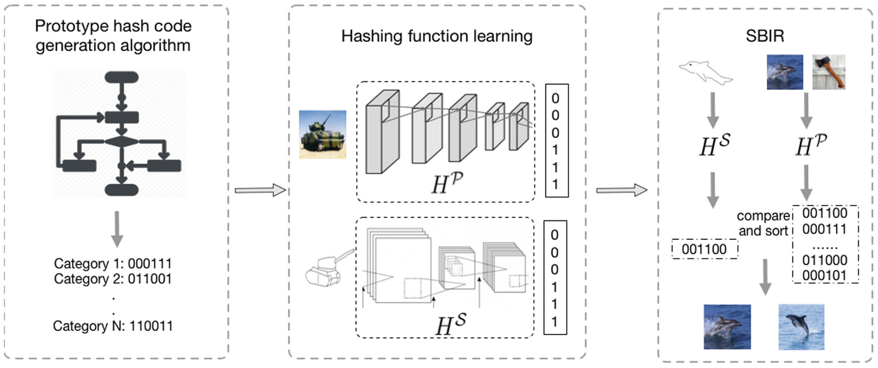

Fig. 2. Pipeline of our proposed idea. The first step is the prototype hash code generation algorithm. The next step is the hashing function learning procedure using prototype codes. The last step is the sketch-based image retrieval process.

To achieve our goal mentioned above, we design an efficient pipeline as shown in Fig. 2. It consists of three parts. Firstly, we encode a set of prototype hash codes fully utilizing the label semantic information. The next step is the hashing function learning procedure. We propose a novel deep hashing architecture that is specific to sketches and natural photos respectively with the help of generated prototype codes. Finally, we conduct sketch-based image retrieval process.

3.2 Prototype Hash Code

To mend the semantic gap in cross-domain situation such as our problem, we call for a comparable feature space applying to both sketches and real images. Hence, we specially design a common prototype binary encoding for both domains, which we call it prototype hash code. Since label information indicates the inherent semantic content for useful hashing learning aforesaid, a straight thinking is to generate a series of prototype hash codes based on label supervision. Given C label classes and d bit length of hash code, we denote the prototype set of C distinct hash codes as $\mathcal{B}^o = \{b_i^o\}_{i=1}^C$, $b_i^o \in \{0,1\}^d$. Among set $\mathcal{B}^o$, every code element b_i^o is matched with a class label. Then, during the subsequent hashing learning procedure, these prototype codes provide a firmly supervised support for efficient training. The target is to train a network which can output a hash code close to its nearest prototype code as much as possible.

To achieve our desired result, our generated prototype codes should meet some requirements. The hamming distance between every code pairs should be maximized, in order to capture more discriminative intrinstic structure and reduce the error rate in retrieval. That is to say, we need to enlarge the minimum hamming distance of this set to the greatest extent:

$$\begin{gathered}\max_f \left\{ d_{min} = \min_{i,j} \left\| b_i^o - b_j^o \right\|_H \right\}, f : \mathcal{L} \rightarrow \mathcal{B}^0 \\ s.t. \mathcal{B}^0 \in \{0,1\}^{d \times C}, b_i^o, b_j^o \in \mathcal{B}^0, i \neq j\end{gathered} \tag{1}$$

where d_{min} is the crucial minimum hamming distance, $\| \cdot \|_H$ is the Hamming distance, $\mathcal{L}$ is the whole label set, f is our prototype encoding algorithm. In practice, our generation problem has no general sole solution in mathematics. Here, we search the feasible solution through controlling d_{min} to grow up increasingly. Starting with a relatively small d_{min}, we can easily find out a candidate set with more than C codewords satisfying the hamming distance between any codes is larger than d_{min}. Afterwards, we increase d_{min} by 1 repeatedly and follow the same searching strategy. Along with the increased d_{min}, the number of possible available codes of a candidate set will be reduced. We stop when the candidate set has less than the lower bound C codes. The last candidate set satisfying the C bound limitation is our final set resource. We randomly select C codes within it as optimal prototype code set $\mathcal{B}^o$. Consequently, we have our specially designed prototype result well maintaining discriminative essence. It is worth noting that with our generation algorithm, even very short hash codes can provide a large minimum hamming distance which is quite efficient for SBIR.

3.3 Deep Hashing Architecture

In this part, we propose a novel deep hashing architecture with two networks for sketches and photos individually as shown in Fig. 3. Such deep network can be seen as the hashing function which is a decisive factor in hashing learning. We denote hashing function as $H^{\mathcal{S}}$ for the sketch and $H^{\mathcal{P}}$ for the photo. As previously mentioned, sketches have a quite different appearance to real photos. It is unable to directly imitate mature network technique from CBIR here. If we let sketches and real images share the same network, the learned model will add extra noise on sketches and ignore detail structure of natural images, causing a bad effect on both domains. In addition, if we follow previous works to convert

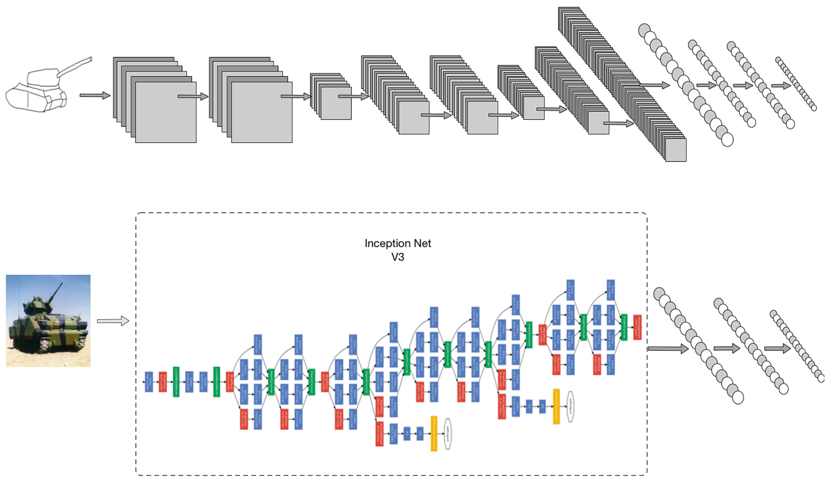

Fig. 3. An illustration of our proposed deep hashing retrieval architecture. We adopt two separate networks for two domains. The upper part is the network for sketch, and the lower part is the network for real photo.

natural images to approximate sketches by edge extraction in advance, it still has some defects. Actually, a sketch is different from a simple tracing of image boundary. People often draw sketches with geometry distortion and simply rely on their vague memory. We once test this edge extraction strategy and it turned out to be less effective.

Benefiting from prototype hash code introduced before, we adopt two separate networks for sketches and photos. The networks will learn a shared embedding in Hamming space. Prototype code set is a common binary embedding for sketches and photos both, guiding hashing learning at the last layer. Thus, two separate networks have the same mission that samples have to be grouped around the targeted prototype code.

For sketches, we adopt a carefully designed network containing 6 convolutional layers and 2 fully connected layers illustrated in the upper part of Fig. 3. The last layer is a fully connected layer with d nodes, relying on the hash code bit length. We also call it hash layer since it will encode the high-dimensional feature into binary-like one y_i via a following sigmoid activation. Sigmoid function has proved to be effective for hashing methods in that it could regulate features within a $(0, 1)$ real-valued range, becoming hash-like features. To train the sketch network end-to-end, we exploit Mean Squared Logarithmic Error (MSLE) with the supervision of prototype hash code:

$$\mathcal{L} = \frac{1}{n_1} \sum_{i=1}^{n_1} \left\| \log(b_i^o + 1) - \log(y_i + 1) \right\|^2 \tag{2}$$

where b_i^o is the prototype hash code that y_i referred to. As MSLE is more robust to overfitting than mean squared error, we choose MSLE as our learning objective. The detailed network configuration is illustrated in Table 1. The reason why we employ such kind of shallow network is based on sketch trait itself. The sketch is a grayscale image with rather sparse lines. Due to lack of abundant structure information and the unbalanced zero-one amount in sketches, a very deep network will fall into overfitting. Besides, we employ data augmentation procedure to further avoid overfitting situation and make limited training samples more robust. Each sketch sample runs through the augmentation preprocessing and then be fed into the network. A Random rotation, shear, zoom and translation will be applied.

Table 1. Detailed configuration of sketch network.

Layer	Filter Size	Filter Num	Stride	Pad	Activation	Output
Input	-	-	-	-	-	$1 \times 128 \times 128$
Conv	3×3	32	1	1	ReLU	$32 \times 128 \times 128$
Conv	3×3	32	1	1	ReLU	$32 \times 128 \times 128$
Dropout(0.25)	-	-	-	-	-	$32 \times 128 \times 128$
MaxPool	3×3	-	3	0	-	$32 \times 42 \times 42$
Conv	3×3	64	1	1	ReLU	$64 \times 42 \times 42$
Conv	3×3	64	1	1	ReLU	$64 \times 42 \times 42$
Dropout(0.25)	-	-	-	-	-	$64 \times 42 \times 42$
MaxPool	2×2	-	2	0	-	$64 \times 21 \times 21$
Conv	3×3	128	1	1	ReLU	$128 \times 21 \times 21$
Conv	3×3	2048	1	1	ReLU	$2048 \times 21 \times 21$
Dropout(0.25)	-	-	-	-	-	$2048 \times 21 \times 21$
GlobalAvgPool	21×21	-	21	0	-	2048
FC	1×1	256	-	-	ReLU	256
FC	1×1	256	-	-	ReLU	256
Dropout(0.5)	-	-	-	-	-	256
FC(hash)	1×1	d	-	-	Sigmoid	d

For real photos, we employ a revised successful deep network Inception-v3 [22] as shown in the lower figure of Fig. 3. Because the photo retrieval task is similar to traditional image-based task, we directly use a well-performing standard configuration from ILSVRC competition as our basic framework. Inception net is the winner of ILSVRC 2014 proposed by Google. Real photo input is downsampled to 140×140. Then we replace the last fully connected layers and softmax layer of original Inception net by a fully connected layer with 1024 nodes right after the global average pooling. At the end, a hash layer is applied exactly as sketch network works. We still use MSLE as our loss function.

3.4 Retrieval Process

The output of training network is real-valued feature y_i. To obtain the final hash code b_i, we binarize the activation output with a threshold:

$$\begin{gathered} b_i = sgn(y_i - 0.5), \\ sgn(x) = \begin{cases} 0 & x \leq 0, \\ 1 & x > 0 \end{cases} \end{gathered} \tag{3}$$

Next, we carry out retrieval process. For SBIR, hashing retrieval process is a little more complicated than CBIR. (i) With our fine trained photo hashing function $H^{\mathcal{P}}$, all the real images in gallery set are transformed to compact d-bit hash codes. We denote the candidate hash pool as $\mathcal{P}$. (ii) Given a query sketch s^q, it will go through the trained sketch model $H^{\mathcal{S}}$ and output its sketch code b^q. (iii) We compare each candidate photo code in $\mathcal{P}$ with a query code b^q by calculating hamming distance. The hamming distance has a positive relation to similarity. Hence, it forms a rank of retrieval results in ascending order of distance.

4 Experiments

In this section, we demonstrate the effectiveness of our proposed method on sketch-based image retrieval. We conduct extensive experiments on two public datasets and our method is compared with several state-of-the-art methods. At last, we evaluate our method and verify its good performance.

4.1 Datasets

So far, the largest datasets in SBIR are *TU-Berlin Extension* and *Sketchy extension*. *TU-Berlin* benchmark [4] is aimed at sketch recognition and classification. It consists of 20,000 sketch images evenly belonging to 250 categories covering daily objects like teapot, car and horse. The extended TU-Berlin [30] dataset adds 204,489 real images in total as gallery set for sketch-based image retrieval. *Sketchy* [20] is the latest released dataset specifically collected for retrieval. It contains 75,471 sketches of 12,500 natural objects from 125 categories. The extended sketchy [13] provides another 60,502 natural images and merges original natural images into the retrieval gallery pool. Both two collections are convincing evaluation datasets for large-scale SBIR task.

For fair comparison to previous works, we follow the same experimental setting as DSH [13]. We randomly select 2,500 sketches (10 sketches per category) for TU-Berlin and 6,250 sketches (50 sketches per category) for Sketchy as test query sketches. And we use the remaining natural images and the rest of sketches for training.

4.2 Implementation Details

We implement our experiments on single GTX1060 GPU with 6GB memory. For sketch model, data augmentation is applied before entering the network. We perform random rotation in the range of 20 degrees, horizontal and vertical translation up to 25 pixels, zoom from 0.8 to 1.2 times and a 0.2 shear intensity to reduce overfitting. During training, batch size is set to 40 and the initial learning rate is 0.001. The model is trained for 200 epochs with Adam [7] optimizer. For photo model, we do training for 40 epochs with batch size of 128. And we still use Adam optimizer and the learning rate is 0.001.

Table 2. Performance comparison in SBIR with state-of-the-arts via mAP, Precision@200, Retrieval time per query and Memory load on TU-Berlin Extension.

Method	Dimension	TU-Berlin Extension			
		mAP	P@200	Retrieval time per query(s)	Memory load(MB)
HOG	1296	0.091	0.120	1.43	2.02×10^3
GF-HOG	3500	0.119	0.148	4.13	5.46×10^3
SHELO	1296	0.123	0.155	1.44	2.02×10^3
LKS	1350	0.157	0.204	1.51	2.11×19^3
Siamese CNN	64	0.322	0.447	7.70×10^{-2}	99.8
SaN	512	0.154	0.225	0.53	7.98×10^2
GN Triplet	1024	0.187	0.301	1.02	1.60×10^3
3D Shape	64	0.054	0.072	7.53×10^{-2}	99.8
Siamese-AlexNet	4096	0.367	0.476	5.35	6.39×10^3
Triplet-AlexNet	4096	0.448	0.552	5.35	6.39×10^3
DSH-32	32	0.358	0.486	5.57×10^{-4}	0.78
DSH-64	64	0.521	0.655	7.03×10^{-4}	1.56
DSH-128	128	0.570	0.694	1.05×10^{-3}	3.12
Our-12	12	0.550	0.622	3.04×10^{-4}	0.29
Our-24	24	0.561	0.634	4.48×10^{-4}	0.59
Our-32	32	0.573	0.650	5.43×10^{-4}	0.78
Our-64	64	0.591	0.668	6.99×10^{-4}	1.56
Our-128	128	0.613	0.693	9.72×10^{-4}	3.12

4.3 Results and Analysis

Our results are evaluated within the whole gallery set on extended TU-Berlin and Sketchy respectively. We compare our method with several state-of-the-art deep SBIR approaches including Siamese CNN [17], sketch-a-net (SaN) [28], GN

Table 3. Performance comparison in SBIR with state-of-the-arts via mAP, Precision@200, Retrieval time per query and Memory load on Sketchy Extension.

Method	Dimension	Sketchy Extension			
		mAP	P@200	Retrieval time per query(s)	Memory load(MB)
HOG	1296	0.115	0.159	0.53	7.22×10^2
GF-HOG	3500	0.157	0.177	1.41	1.95×10^3
SHELO	1296	0.161	0.182	0.50	7.22×10^2
LKS	1350	0.190	0.230	0.56	7.52×10^2
Siamese CNN	64	0.481	0.612	2.76×10^{-2}	35.4
SaN	512	0.208	0.292	0.21	2.85×10^2
GN Triplet	1024	0.529	0.716	0.41	5.70×10^2
3D Shape	64	0.084	0.079	2.64×10^{-2}	35.6
Siamese-AlexNet	4096	0.518	0.690	1.68	2.28×10^3
Triplet-AlexNet	4096	0.573	0.761	1.68	2.28×10^3
DSH-32	32	0.653	0.797	2.55×10^{-4}	0.28
DSH-64	64	0.711	0.858	2.82×10^{-4}	0.56
DSH-128	128	0.783	0.866	3.53×10^{-4}	1.11
Our-12	12	0.762	0.839	2.21×10^{-4}	0.11
Our-24	24	0.772	0.850	2.43×10^{-4}	0.21
Our-32	32	0.789	0.867	2.57×10^{-4}	0.28
Our-64	64	0.796	0.876	2.81×10^{-4}	0.56
Our-128	128	0.810	0.890	3.57×10^{-4}	1.11

Triplet [20], 3D Shape [23], as well as Siamese-AlexNet and Triplet-AlexNet [13]. Traditional hand-crafted methods HOG [2], GF-HOG [6]. SHELO [18] and LKS [19] are also included. To better demonstrate our outstanding performance, we conduct our experiments with 12, 24, 32, 64 and 128 bits hash code. This is the same setting compared with deep hashing method DSH [13]. During the comparison to SBIR baselines, we use a ranking based criterion mean Average Precision (mAP) and precision at top 200 (P@200) to evaluate the retrieval quality. Higher mAP and P@200 indicate a higher retrieval level in the ranking list. The memory load over the whole gallery images and retrieval time per query are also listed. Public results data of previous works are derived from DSH. The comparison results on two datasets are illustrated in Tables 2 and 3.

From our extensive comparison results, we have the following findings: (i) Our method outperforms all the baseline methods on both large-scale datasets. We increase around 4% and 3% mAP over TU-Berlin Extension and Sketchy Extension respectively. (ii) It is noteworthy that results on TU-Berlin are inferior to Sketchy due to much more categories and larger gallery size. (iii) Deep methods strongly beat traditional hand-crafted ones, proving the powerful ability of deep features. Additionally, with the help of hashing, our method and DSH can save memory load and retrieval time by almost four orders of magnitude. (iv) In comparison to the only deep hashing competitor DSH, our method has a better performance under several metrics. Especially, our method can achieve a good behavior even with quite compact hash code. For instance, our 12-bit hashing result surpasses 64-bit DSH in mAP. It demonstrates that our unified prototype hash code set is suitable for fast large-scale retrieval task. Moreover, our method utilizes a point-to-point training rather than pairwise loss in DSH, avoid tedious sample building step and weak training guidance.

5 Conclusion

In this paper, we introduce a novel deep hashing method for sketch-based image retrieval. Our method adopts a prototype hash code set for constraining feature representation. A deep hashing architecture is specially designed for two different domains, sketches and natural photos respectively. By means of mapping different domains into a common hamming space, our method achieves good performances with very compact binary codes. Extensive experiments across large-scale retrieval benchmarks demonstrate that our method outperforms all non-deep and deep methods under several metrics. In general, our method exhibits promising result in fast and efficient retrieval via compact binary codes learning.

References

1. Canny, J.: A computational approach to edge detection. In: Readings in Computer Vision, pp. 184–203. Elsevier (1987)
2. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2005. CVPR 2005, vol. 1, pp. 886–893. IEEE (2005)
3. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry, pp. 253–262. ACM (2004)
4. Eitz, M., Hays, J., Alexa, M.: How do humans sketch objects? ACM Trans. Graph. **31**(4), 44–1 (2012)
5. Gong, Y., Lazebnik, S.: Iterative quantization: a procrustean approach to learning binary codes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2011, pp. 817–824. IEEE (2011)
6. Hu, R., Collomosse, J.: A performance evaluation of gradient field hog descriptor for sketch based image retrieval. Comput. Vis. Image Underst. **117**(7), 790–806 (2013)
7. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
8. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings. In: Advances in Neural Information Processing Systems, pp. 1042–1050 (2009)

9. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3270–3278 (2015)
10. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. Handb. Brain Theory Neural Netw. **3361**(10), 1995 (1995)
11. Li, G., Liu, J., Jiang, C., Zhang, L., Lin, M., Tang, K.: Relief R-CNN: utilizing convolutional features for fast object detection. In: Cong, F., Leung, A., Wei, Q. (eds.) ISNN 2017. LNCS, vol. 10261, pp. 386–394. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59072-1_46
12. Li, W.J., Wang, S., Kang, W.C.: Feature learning based deep supervised hashing with pairwise labels. arXiv preprint arXiv:1511.03855 (2015)
13. Liu, L., Shen, F., Shen, Y., Liu, X., Shao, L.: Deep sketch hashing: Fast free-hand sketch-based image retrieval. In: Proceedings of CVPR, pp. 2862–2871 (2017)
14. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2012, pp. 2074–2081. IEEE (2012)
15. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
16. Norouzi, M., Blei, D.M.: Minimal loss hashing for compact binary codes. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 353–360 (2011)
17. Qi, Y., Song, Y.Z., Zhang, H., Liu, J.: Sketch-based image retrieval via siamese convolutional neural network. In: IEEE International Conference on Image Processing (ICIP) 2016, pp. 2460–2464. IEEE (2016)
18. Saavedra, J.M.: Sketch based image retrieval using a soft computation of the histogram of edge local orientations (s-helo). In: IEEE International Conference on Image Processing (ICIP) 2014, pp. 2998–3002. IEEE (2014)
19. Saavedra, J.M., Barrios, J.M., Orand, S.: Sketch based image retrieval using learned keyshapes (LKS). In: BMVC, vol. 1, p. 7 (2015)
20. Sangkloy, P., Burnell, N., Ham, C., Hays, J.: The sketchy database: learning to retrieve badly drawn bunnies. ACM Trans. Graph. (TOG) **35**(4), 119 (2016)
21. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: IEEE Conference on Computer Vision and Pattern Recognition 2007. CVPR 2007, pp. 1–8. IEEE (2007)
22. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
23. Wang, F., Kang, L., Li, Y.: Sketch-based 3D shape retrieval using convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015, pp. 1875–1883. IEEE (2015)
24. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in Neural Information Processing Systems, pp. 1753–1760 (2009)
25. Wu, X., Kamata, S.i., Ma, L.: Supervised two-step hash learning for efficient image retrieval. In: 2017 4th Asian Conference on Pattern Recognition. IEEE (2017)
26. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: AAAI, vol. 1, p. 2 (2014)
27. Yu, Q., Liu, F., Song, Y.Z., Xiang, T., Hospedales, T.M., Loy, C.C.: Sketch me that shoe. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016, pp. 799–807. IEEE (2016)
28. Yu, Q., Yang, Y., Song, Y.Z., Xiang, T., Hospedales, T.: Sketch-a-net that beats humans. arXiv preprint arXiv:1501.07873 (2015)

29. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
30. Zhang, H., Liu, S., Zhang, C., Ren, W., Wang, R., Cao, X.: Sketchnet: Sketch classification with web images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1105–1113 (2016)