Deep Sketch Hashing: Fast Free-hand Sketch-Based Image Retrieval

Li Liu[1], Fumin Shen[2], Yuming Shen[1], Xianglong Liu[3], and Ling Shao[1]

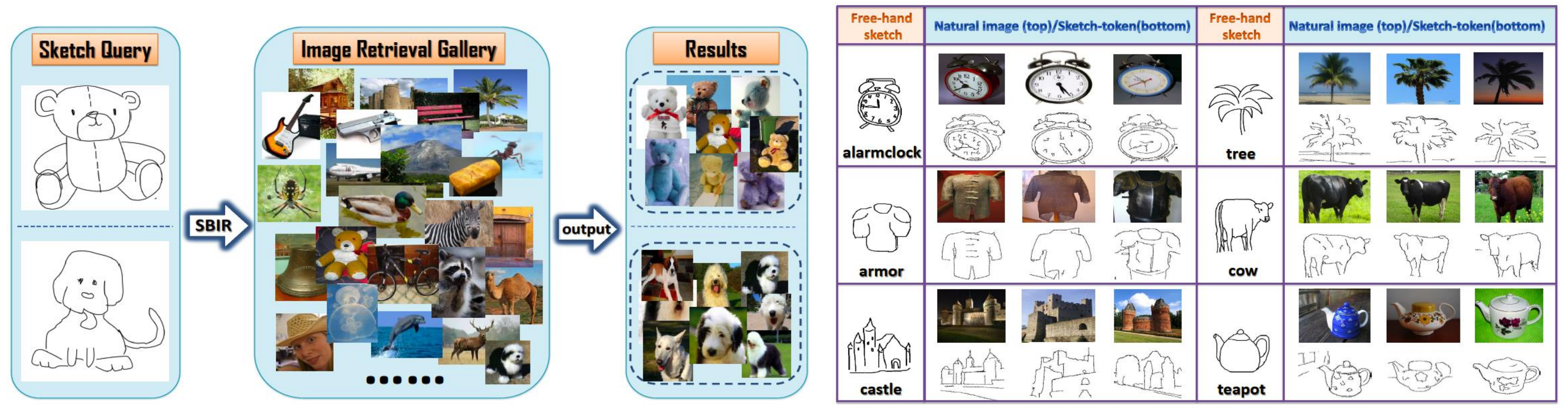
1. University of East Anglia, UK 2. University of Electronic Science and Technology of China, China 3. Beihang University, China

IEEE 2017 Conference on Computer Vision and Pattern Recognition

CVPR July 21-26 HONOLULU 2017

Motivation: Sketch-Based Image Retrieval (SBIR)

Given a free-hand sketch query, we aim to retrieve relevant natural images in the same category as the query from the gallery.

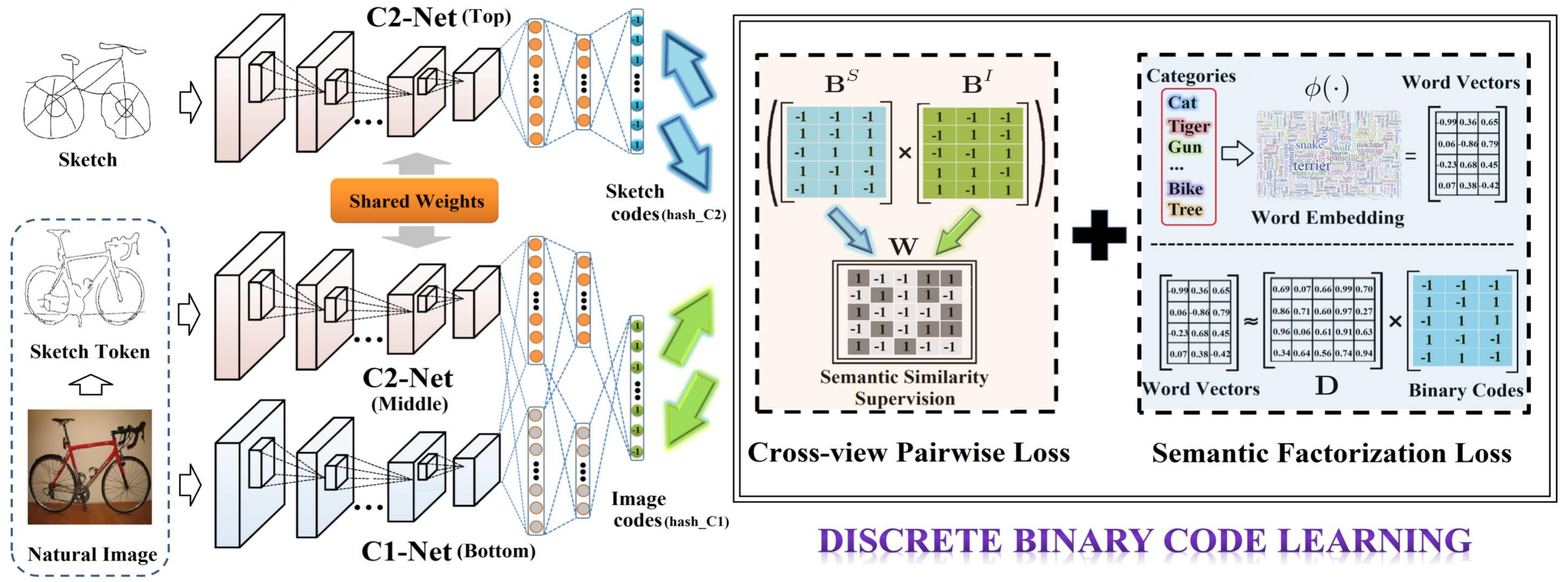


Contribution of This Work

- To the best of our knowledge, **DSH** is the first hashing work **specifically designed** for category-level SBIR.
- A novel **semi-heterogeneous deep architecture** is developed in **DSH**.
- The experiments consistently illustrate **superior performance** of **DSH** compared to the state-of-the-art methods.

Network Architecture

A **convolutional neural network** and **discrete binary code learning** are integrated into a unified end-to-end framework, optimized in an **alternating manner**.

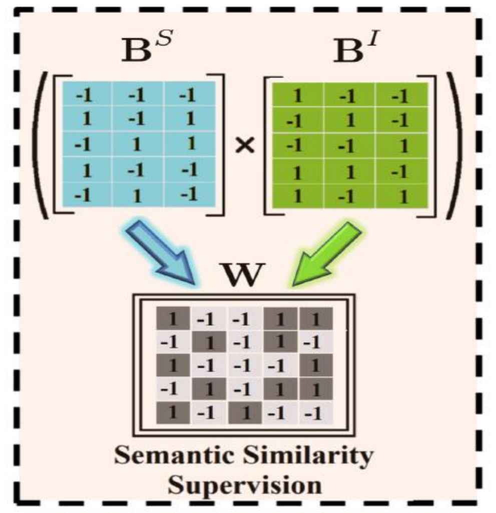


Learning Objectives

1. Cross-view pairwise loss. The produced binary codes of images and sketches need to be similar.

$$\min_{\mathbf{B}^I,\mathbf{B}^S} \mathcal{J}_1 := ||\mathbf{W} \odot m - {\mathbf{B}^I}^\top \mathbf{B}^S||^2,$$

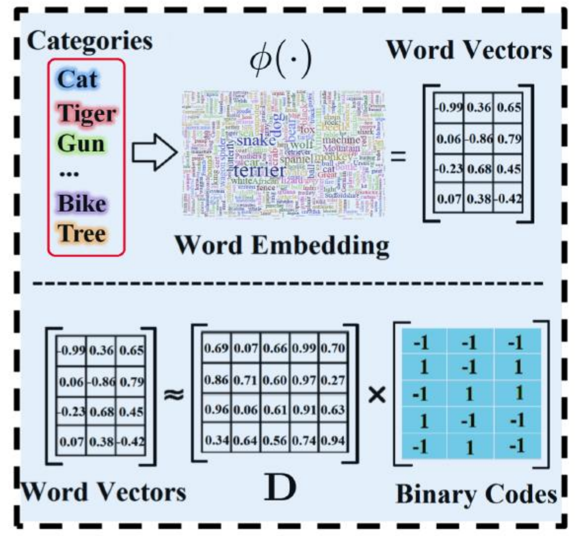
$$\text{s.t. } \mathbf{B}^I \in \{-1,+1\}^{m \times n_1},\ \mathbf{B}^S \in \{-1,+1\}^{m \times n_2}$$



2. Semantic factorization loss. The intra-set semantic relationships across different categories are also considered.

$$\min_{\mathbf{B}^I,\mathbf{B}^S} \mathcal{J}_2 := ||\phi(\mathbf{Y}^I) - \mathbf{D}\mathbf{B}^I||^2 + ||\phi(\mathbf{Y}^S) - \mathbf{D}\mathbf{B}^S||^2,$$

$$\text{s.t. } \mathbf{B}^I \in \{-1,+1\}^{m \times n_1},\ \mathbf{B}^S \in \{-1,+1\}^{m \times n_2},$$



The final loss is formulated by combining the learning objectives above together, resulting in a non-convex optimization problem:

$$\min_{\mathbf{B}^I,\mathbf{B}^S,\mathbf{D}^I,\mathbf{D}^S,\Theta_1,\Theta_2} \mathcal{J} := ||\mathbf{W} \odot m - {\mathbf{B}^I}^\top \mathbf{B}^S||^2$$
$$+ \lambda(||\phi(\mathbf{Y}^I) - \mathbf{D}\mathbf{B}^I||^2 + ||\phi(\mathbf{Y}^S) - \mathbf{D}\mathbf{B}^S||^2)$$
$$+ \gamma(||\mathbf{F}_1(\mathcal{O}_1;\Theta_1,\Theta_2) - \mathbf{B}^I||^2 + ||\mathbf{F}_2(\mathcal{O}_2;\Theta_2) - \mathbf{B}^S||^2)$$
$$\text{s.t. } \mathbf{B}^I \in \{-1,+1\}^{m \times n_1},\ \mathbf{B}^S \in \{-1,+1\}^{m \times n_2}.$$

Alternating Optimization

D Update Step. By fixing all variables except for $\mathbf{D}$, Eq.(3) shrinks to a classic quadratic regression problem

$$\min_{\mathbf{D}} ||\phi(\mathbf{Y}^I) - \mathbf{D}\mathbf{B}^I||^2 + ||\phi(\mathbf{Y}^S) - \mathbf{D}\mathbf{B}^S||^2$$

$$\mathbf{D} = (\phi(\mathbf{Y}^I){\mathbf{B}^I}^\top + \phi(\mathbf{Y}^S){\mathbf{B}^S}^\top)(\mathbf{B}^I{\mathbf{B}^I}^\top + \mathbf{B}^S{\mathbf{B}^S}^\top)^{-1}$$

$\mathbf{B}^I$ Update Step. By fixing all other variables, we optimize $\mathbf{B}^I$ by the following equation

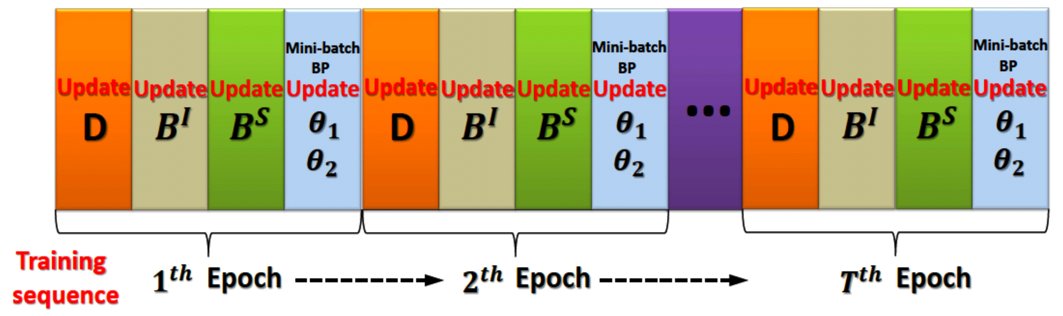
$$\min_{\mathbf{B}^I} ||\mathbf{W} \odot m - {\mathbf{B}^I}^\top \mathbf{B}^S||^2 + \lambda||\phi(\mathbf{Y}^I) - \mathbf{D}\mathbf{B}^I||^2$$
$$+ \gamma||\mathbf{F}_1(\mathcal{O}_1;\Theta_1,\Theta_2) - \mathbf{B}^I||^2,$$
$$\text{s.t. } \mathbf{B}^I \in \{-1,+1\}^{m \times n_1}.$$

$$\widehat{\mathbf{b}}^I_k = \text{sign}(\widehat{\mathbf{r}}_k - \widehat{\mathbf{b}}^S_k {\widehat{\mathbf{B}}^S_{\neg k}}^\top \widehat{\mathbf{B}}^I_{\neg k} - \lambda {\widehat{\mathbf{d}}_k}^\top \widehat{\mathbf{D}}_{\neg k} \widehat{\mathbf{B}}^I_{\neg k})$$

Θ_1 and Θ_2 Update Step. Once $\mathbf{B}^I$ and $\mathbf{B}^S$ are obtained, we update parameters Θ_1 and Θ_2 of C1-Net and C2-Net according to the following *Euclidean loss*:

$$\min_{\Theta_1,\Theta_2} \mathcal{L} := ||\mathbf{F}_1(\mathcal{O}_1;\Theta_1,\Theta_2) - \mathbf{B}^I||^2 + ||\mathbf{F}_2(\mathcal{O}_2;\Theta_2) - \mathbf{B}^S||^2.$$

By first computing the partial gradients $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_1(\Theta_1,\Theta_2)}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_2(\Theta_2)}$, we can obtian $\frac{\partial \mathcal{L}}{\partial(\Theta_1,\Theta_2)}$ by the chain rule. We then use the standard mini-batch back-propagation (BP) scheme to simultaneously update Θ_1 and Θ_2 for our entire deep architecture. In practice, the above procedure can be easily achieved by deep learning toolboxes



Experimental Results

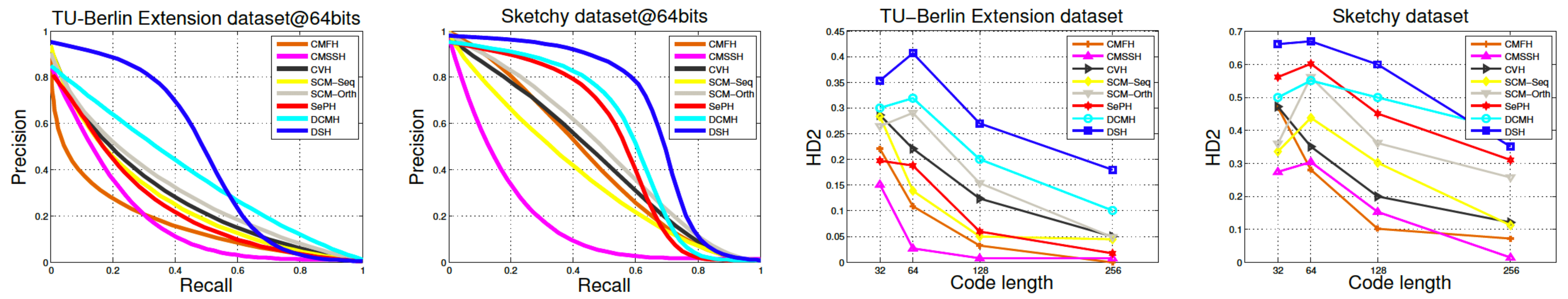
Retrieval performances of **DSH** on the two large-scale image-sketch datasets are shown below.

Methods	Dimension	TU-Berlin Extension MAP	TU-Berlin Extension Precision @200	TU-Berlin Extension Retrieval time per query (s)	TU-Berlin Extension Memory load(MB) (204,489 gallery images)	Sketchy MAP	Sketchy Precision @200	Sketchy Retrieval time per query (s)	Sketchy Memory load(MB) (73,002 gallery images)
HOG [8]	1296	0.091	0.120	1.43	2.02×10^3	0.115	0.159	0.53	7.22×10^2
GF-HOG [18]	3500	0.119	0.148	4.13	5.46×10^3	0.157	0.177	1.41	1.95×10^3
SHELO [49]	1296	0.123	0.155	1.44	2.02×10^3	0.161	0.182	0.50	7.22×10^2
LKS [50]	1350	0.157	0.204	1.51	2.11×10^3	0.190	0.230	0.56	7.52×10^2
Siamese CNN [46]	64	0.322	0.447	7.70×10^{-2}	99.8	0.481	0.612	2.76×10^{-2}	35.4
SaN [67]	512	0.154	0.225	0.53	7.98×10^2	0.208	0.292	0.21	2.85×10^2
GN Triplet* [52]	1024	0.187	0.301	1.02	1.60×10^3	0.529	0.716	0.41	5.70×10^2
3D shape* [61]	64	0.054	0.072	7.53×10^{-2}	99.8 MB	0.084	0.079	2.64×10^{-2}	35.6
Siamese-AlexNet	4096	0.367	0.476	5.35	6.39×10^3	0.518	0.690	1.68	2.28×10^3
Triplet-AlexNet	4096	0.448	0.552	5.35	6.39×10^3	0.573	0.761	1.68 s	2.28×10^3
DSH (Proposed)	32 (bits)	0.358	0.486	5.57×10^{-4}	0.78	0.653	0.797	2.55×10^{-4}	0.28
	64 (bits)	0.521	0.655	7.03×10^{-4}	1.56	0.711	0.858	2.82×10^{-4}	0.56
	128 (bits)	**0.570**	**0.694**	1.05×10^{-3}	3.12	**0.783**	**0.866**	3.53×10^{-4}	1.11

'*' denotes we directly use the public models provided by the original papers without any fine-tuning on TU-Berlin Extension or Sketchy datasets.

Method		TU-Berlin Extension MAP 32 bits	MAP 64 bits	MAP 128 bits	Precision@200 32 bits	Precision@200 64 bits	Precision@200 128 bits	Sketchy MAP 32 bits	MAP 64 bits	MAP 128 bits	Precision@200 32 bits	Precision@200 64 bits	Precision@200 128 bits
Cross-Modality Hashing Methods (binary codes)	CMFH [10]	0.149	0.202	0.180	0.168	0.282	0.241	0.320	0.490	0.190	0.489	0.657	0.286
	CMSSH [2]	0.121	0.183	0.175	0.143	0.261	0.233	0.206	0.211	0.211	0.371	0.376	0.375
	SCM-Seq [68]	0.211	0.276	0.332	0.298	0.372	0.454	0.306	0.417	0.671	0.442	0.529	0.758
	SCM-Orth [68]	0.217	0.301	0.263	0.312	0.420	0.470	0.346	0.536	0.616	0.467	0.650	0.776
	CVH [26]	0.214	0.294	0.318	0.305	0.411	0.449	0.325	0.525	0.624	0.459	0.641	0.773
	SePH [31]	0.198	0.270	0.282	0.307	0.380	0.398	0.534	0.607	0.640	0.694	0.741	0.768
	DCMH [23]	0.274	0.382	0.425	0.332	0.467	0.540	0.560	0.622	0.656	0.730	0.771	0.784
Proposed	**DSH**	**0.358**	**0.521**	**0.570**	**0.486**	**0.655**	**0.694**	**0.653**	**0.711**	**0.783**	**0.797**	**0.858**	**0.866**
Cross-View Feature Learning Methods (continuous-value vectors)	CCA [59]	0.276	0.366	0.365	0.333	0.482	0.536	0.361	0.555	0.705	0.379	0.610	0.775
	XQDA [28]	0.191	0.197	0.201	0.263	0.278	0.278	0.460	0.557	0.550	0.607	0.715	0.727
	PLSR [63]	0.141 (4096-d)			0.215 (4096-d)			0.462 (4096-d)			0.623 (4096-d)		
	CVFL [64]	0.289 (4096-d)			0.407 (4096-d)			0.675 (4096-d)			0.803 (4096-d)		

PLSR and CVFL are both based on reconstructing partial data to approximate full data, so the dimensions are fixed to 4096-d.



We also provide some empirical retrieval results. **DSH** successfully recognizes the hand-crafted sketch query and produces high-quality retrieval results.

https://github.com/ymcidence/DeepSketchHashing