Deep Sketch Hashing: Fast Free-hand Sketch-Based Image Retrieval CVPR ‘17

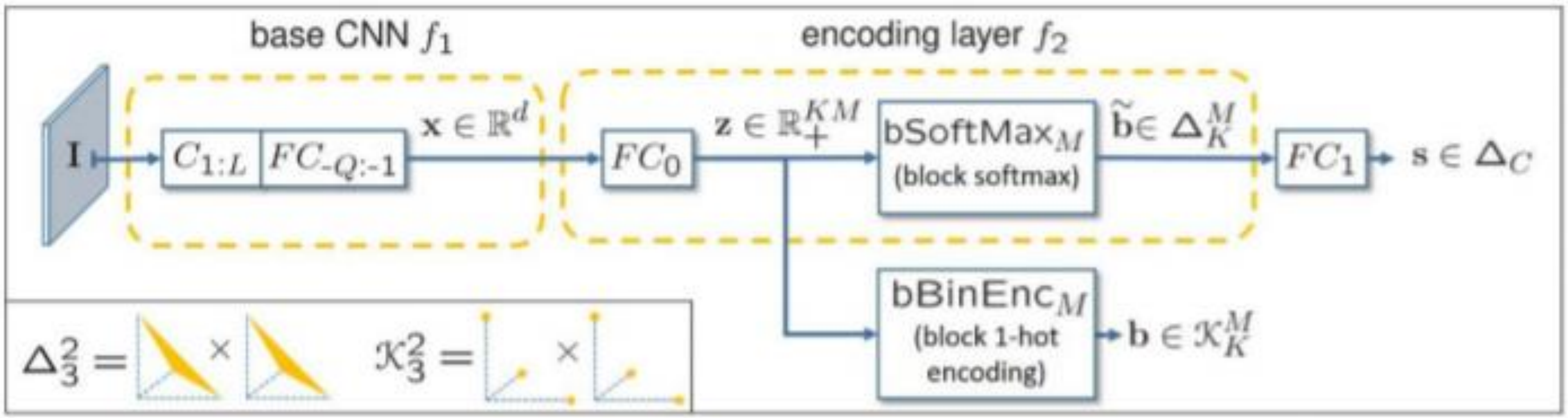
Paper presentation
2018. 11. 01.
Taeun Hwang (황태운)

CS688: Web-Scale image Retrieval

KAIST

Review

- SuBiC: A supervised, structured binary code for image search[ICCV 2017] presented by Huisu Yun



  - Very long Raw feature vectors → binary code
  - Code length in the SuBiC : **KM**
    - actual storage can be easily reduce to **M $\log_2$K**
  - **One hot code block → M** additions for distance computing

Contents

- Introduction
  - Main Idea
- Method
- Experiment & result

Introduction

Introduction

- Sketch-Based Image Retrieval
  - Image retrieval given freehand sketches

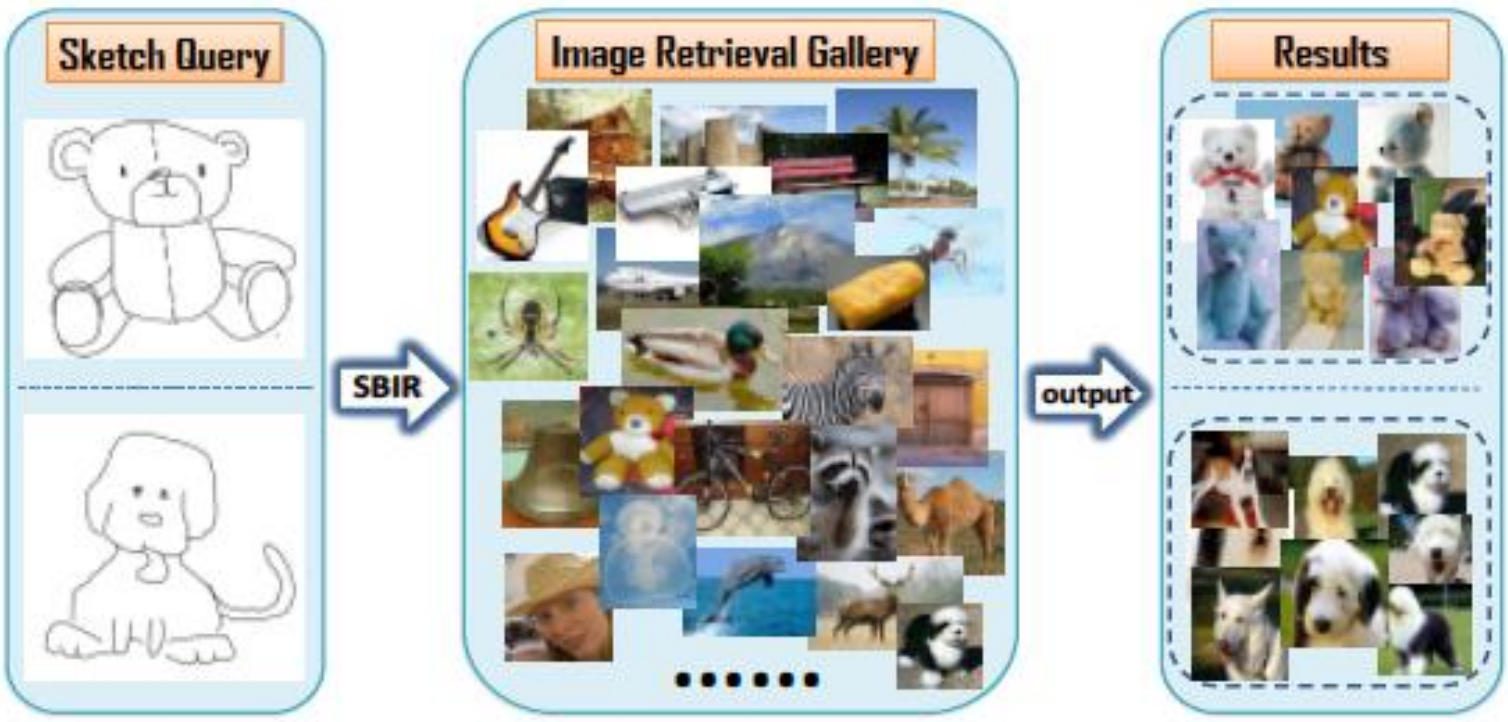


illustration of the SBIR

Challenges in SBIR

- **Geometric distortion between Sketch and Natural image**
  - IE) backgrounds, various viewpoints…

sketch

natural image

- **Searching efficiency of SBIR**
  - Most SBIR tech are based on applying NN
  - Computational complexity **O(Nd)**
  - Inappropriate for Large-scale SBIR

Main Idea

- **Geometric distortion**
  - diminish the geometric distortion using **"sketch-tokens"**

- **Speeds up SBIR** by embedding sketches and natural images into two sets of compact **binary codes**
  - In Large-scale SBIR, heavy continuous-valued distance computation is decrease

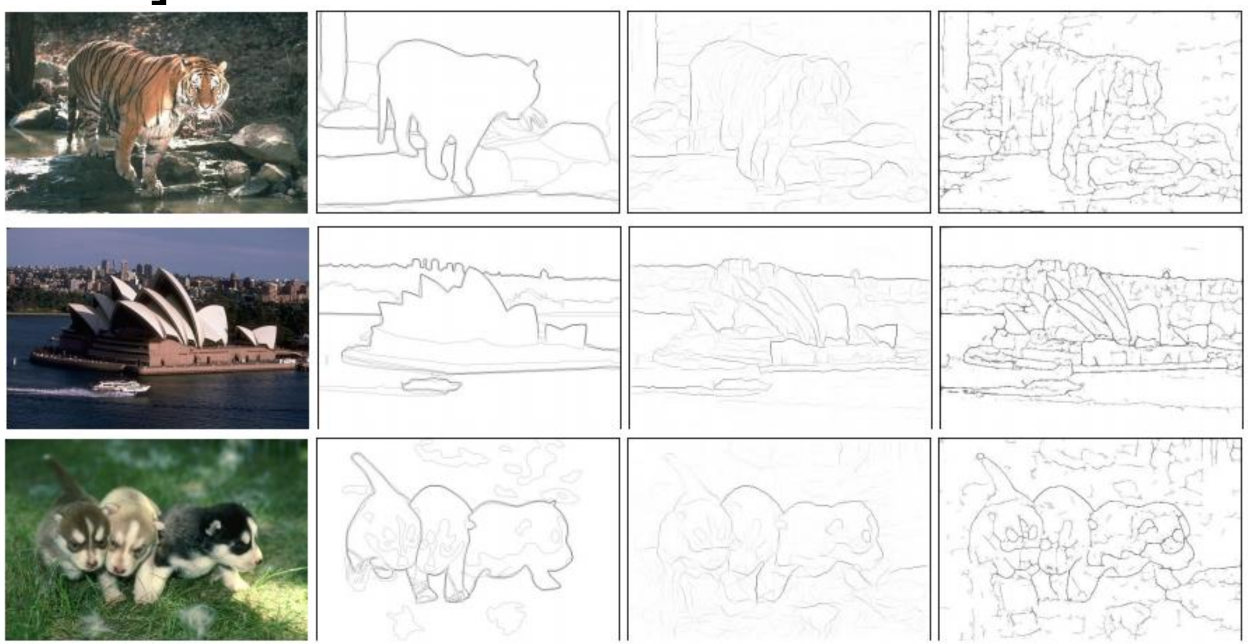
DSH: Method

Deep Sketch Hashing(DSH):

Fast Free-hand Sketch-Based Image Retrieval

Sketch token: background

- Sketch tokens: A learned mid-level representation for contour and object detection [JJ Lim et al., *CVPR*'13]

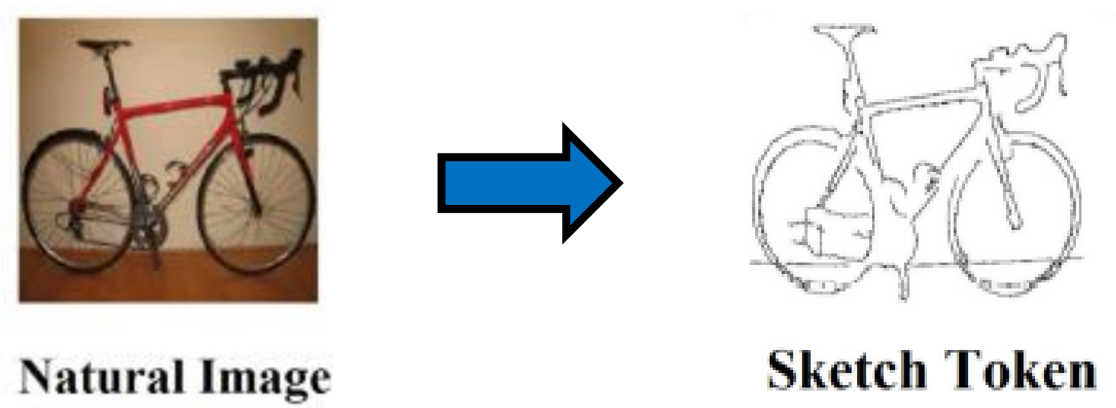


(a) Original image (b) Ground truth (c) SCG [24] (d) Sketch Tokens

- Sketch-token : Hand-drawn **contours** in images

Sketch token: background

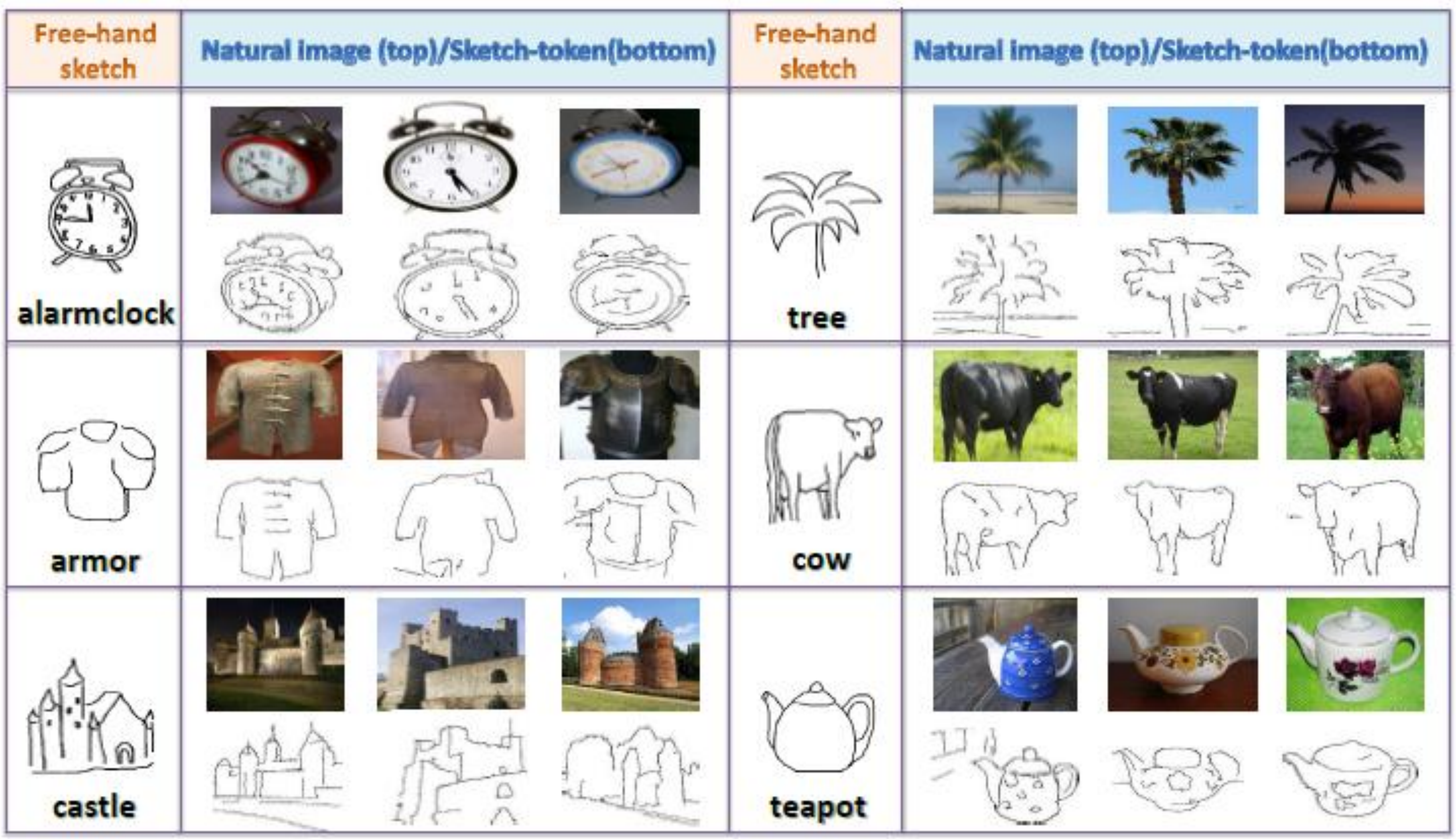
- Sketch-tokens have similar stroke patterns and appearance to free-hand sketches



- Reflect only **essential edges of natural images** without detailed texture information
- In this work : used for diminish geometric distortion between sketch and real image

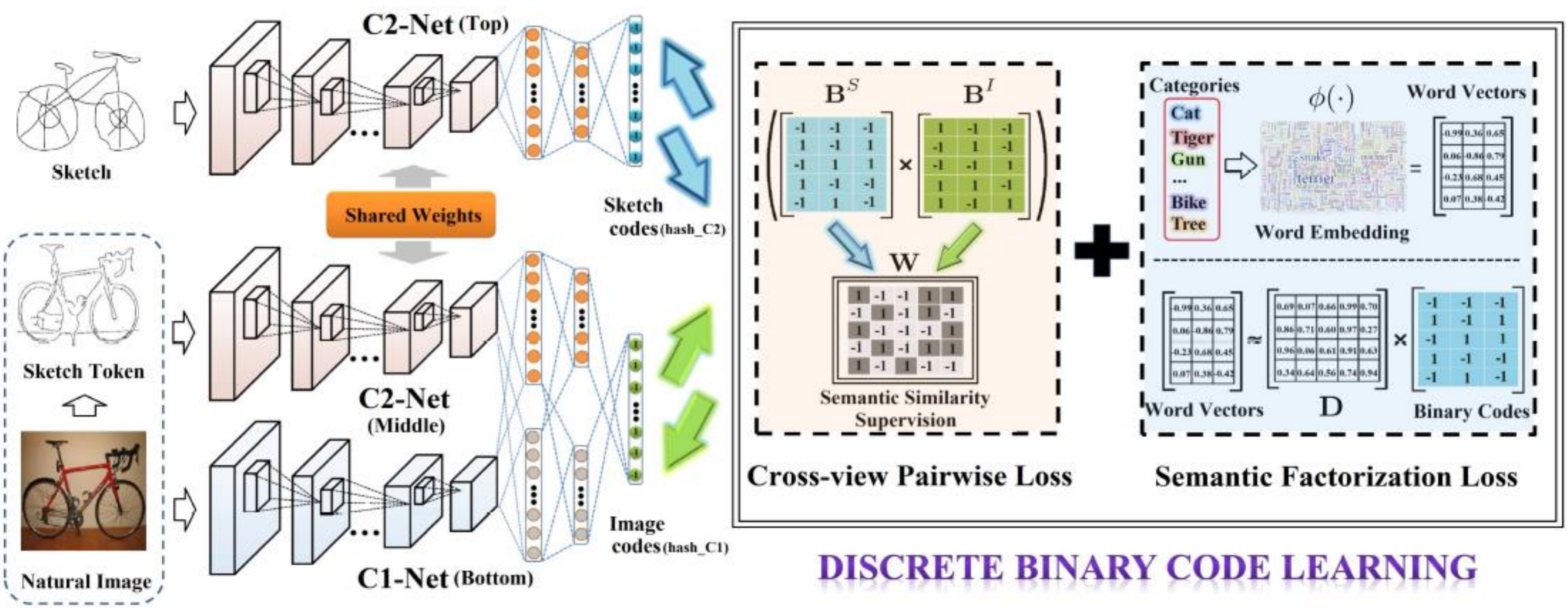
Network structure

- Inputs of DSH



Network structure

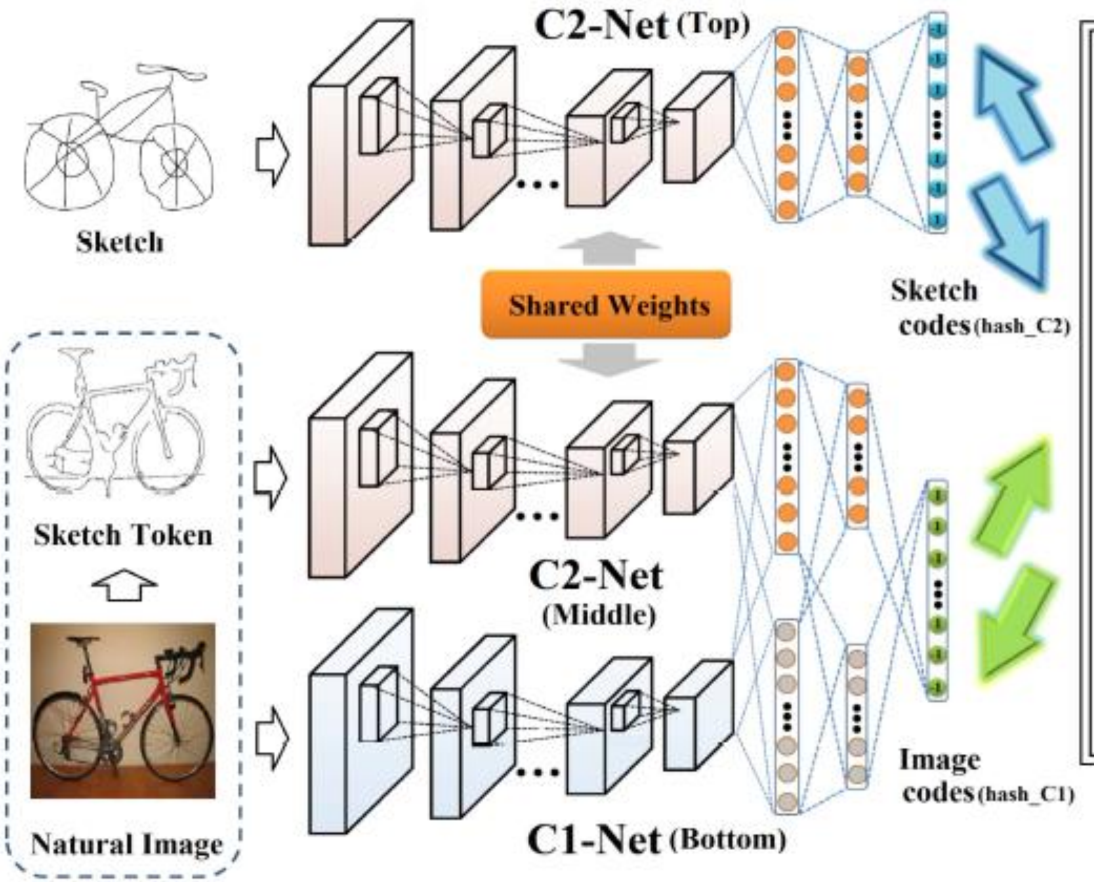
- Semi-heterogeneous Deep Architecture
- Discrete binary code learning



Semi-heterogeneous Deep Architecture

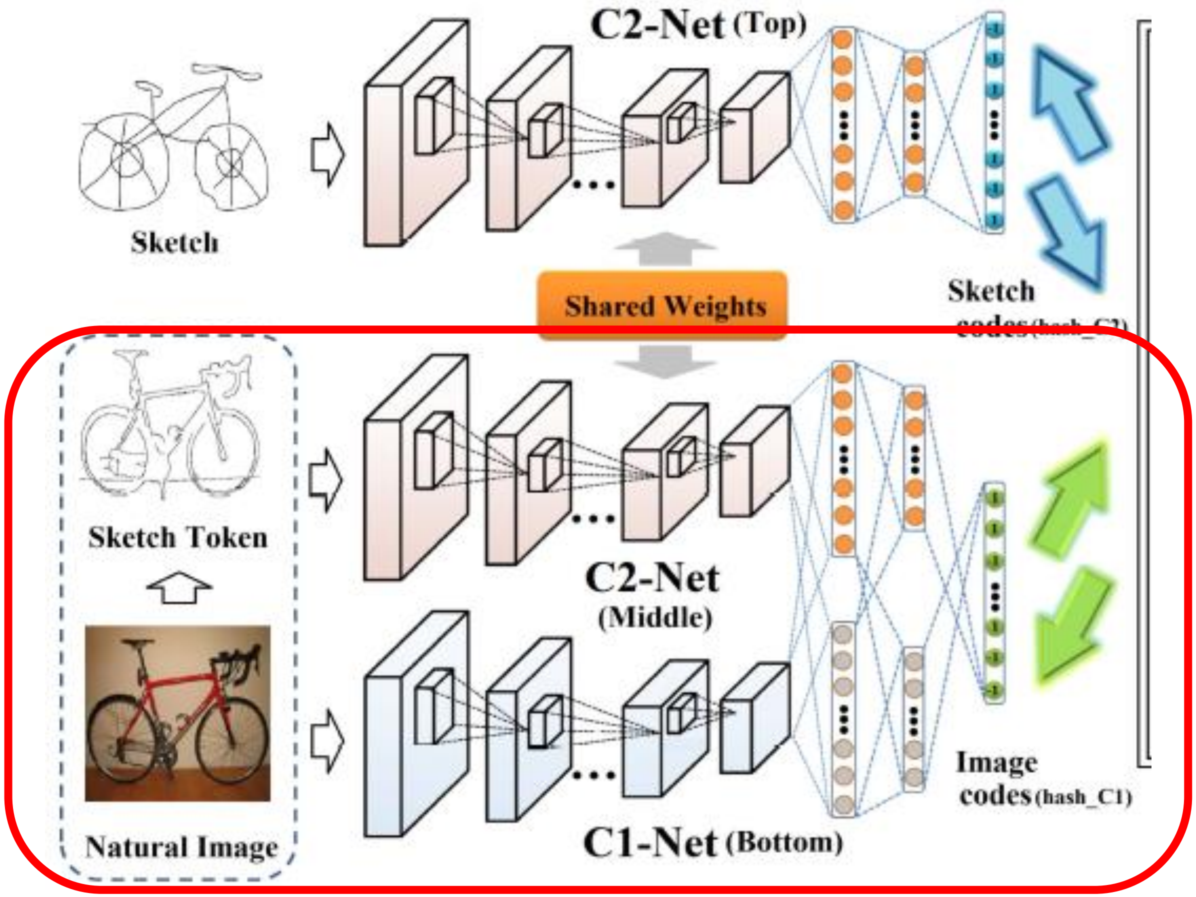
Network structure

- **C1-Net**(CNN) **for Natural image**
- **C2-Net**(CNN) **for sketch and sketch-token**



Semi-heterogeneous Deep Architecture

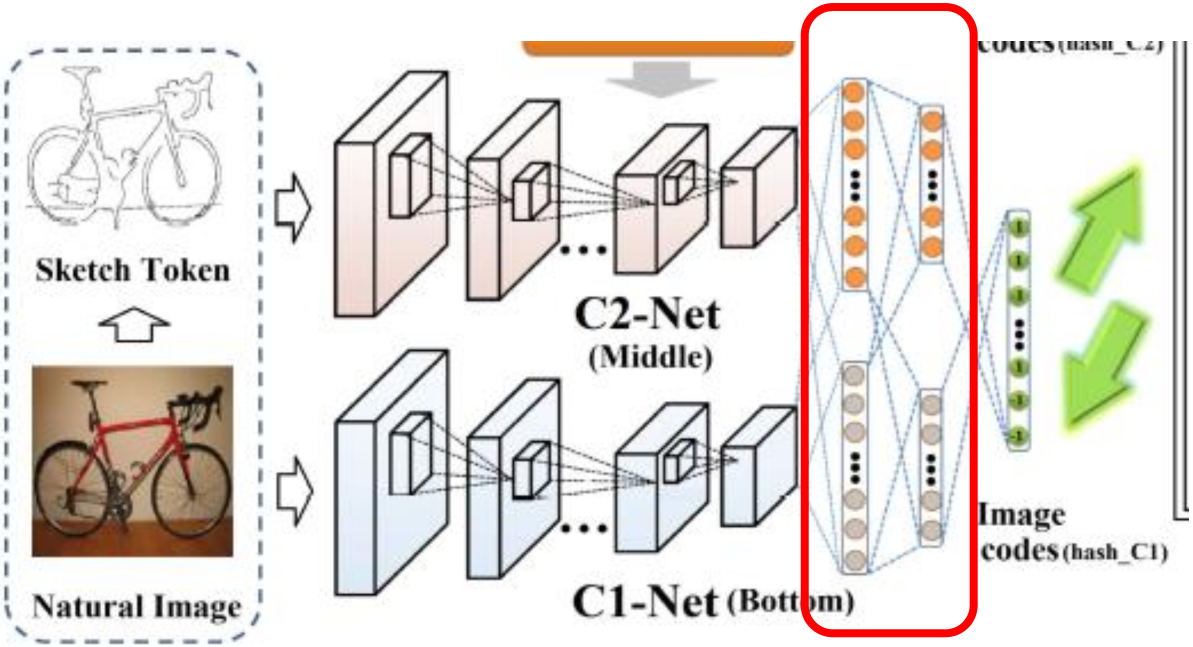
- **Cross-weight Late-fusion Net**



Semi-heterogeneous Deep Architecture

- **Cross-weight Late-fusion Net**

Connect the last pooling and fc layer
with **Cross-weight** [S Rastegar et al., CVPR'16]

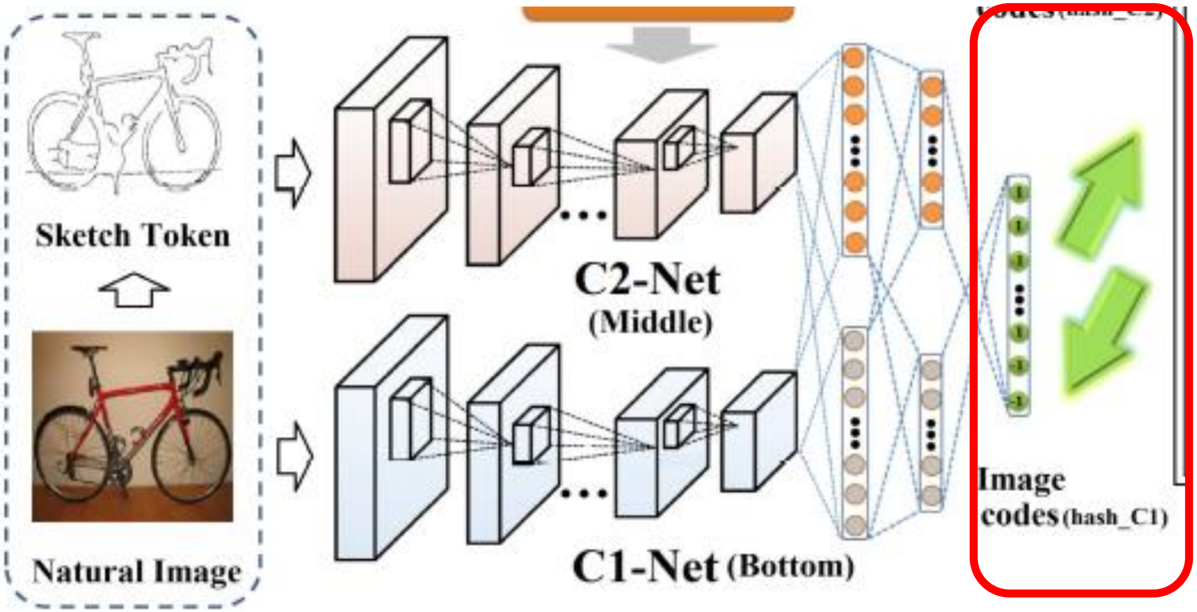


Maximize the mutual inform across both modalities,
while the information from each individual net is also preserved

Semi-heterogeneous Deep Architecture

- **Cross-weight Late-fusion Net**

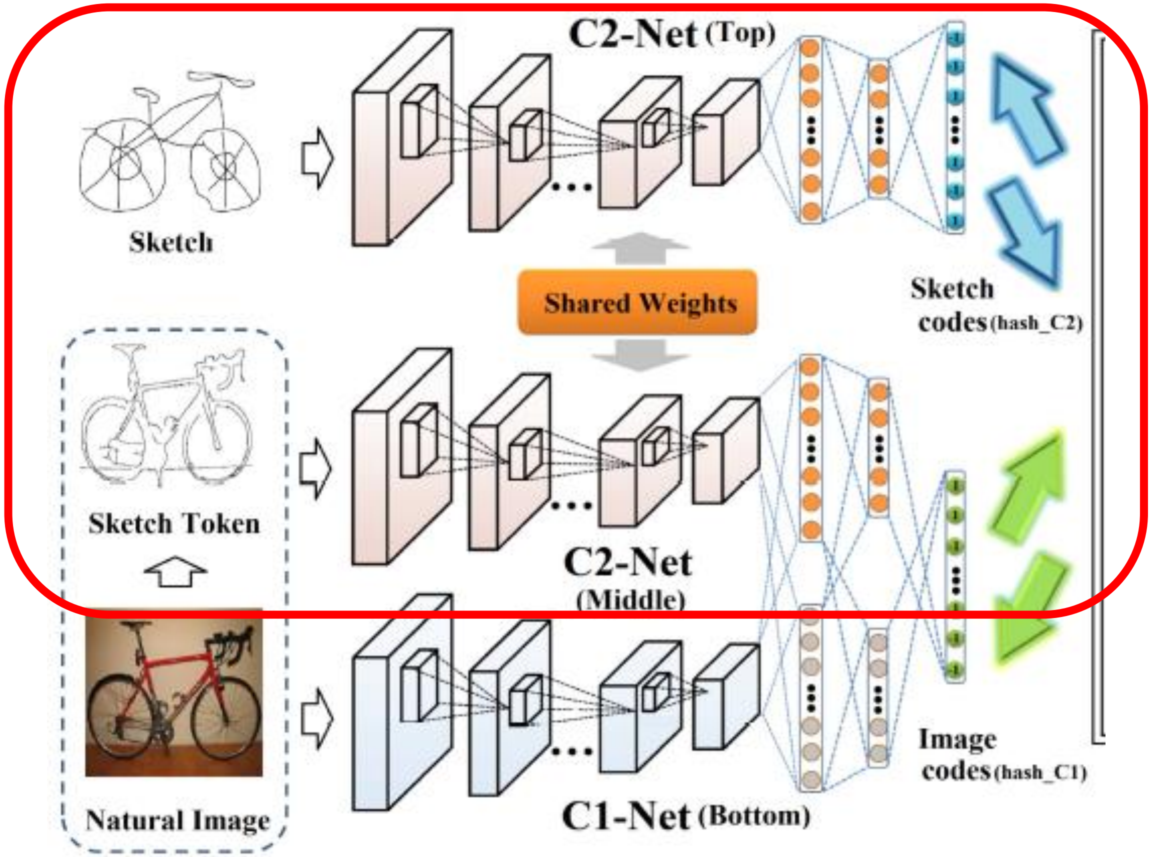
Late-fuse C1-Net and C2-Net
into a unified **binary coding layer hash_C1**



the learned codes can fully benefit from both natural images and their corresponding sketch-tokens

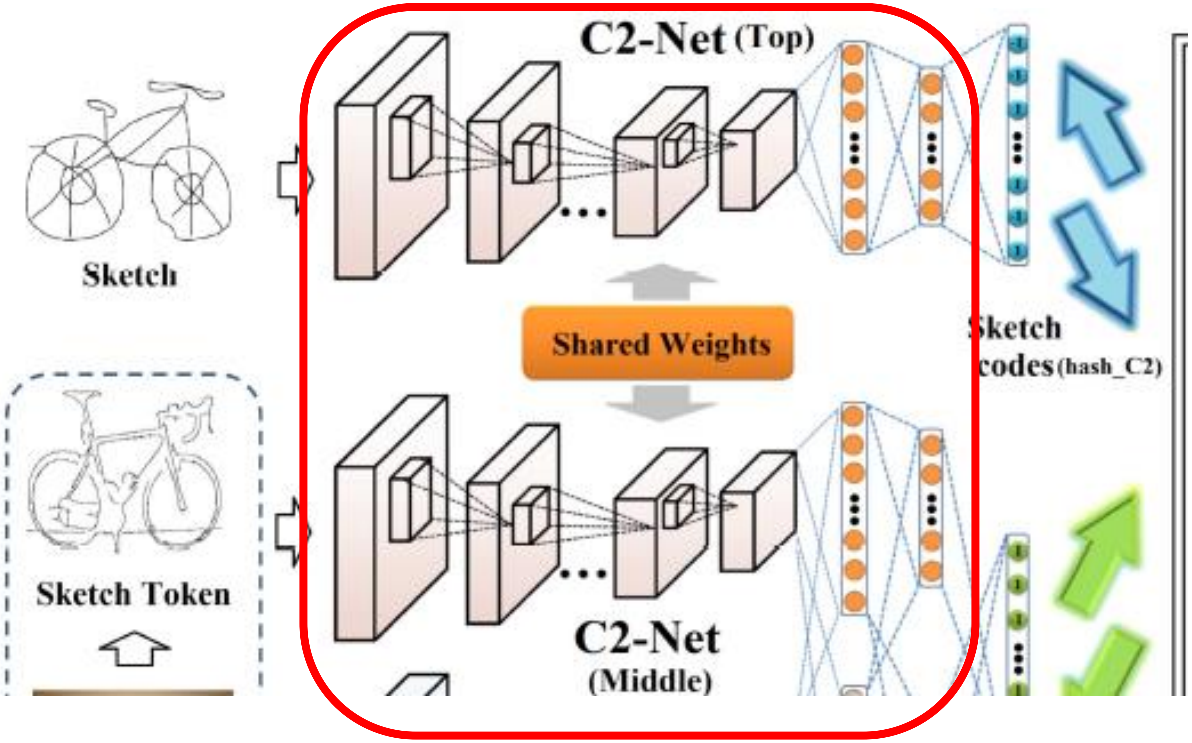
Semi-heterogeneous Deep Architecture

- **Shared-weight Sketch Net**



Semi-heterogeneous Deep Architecture

- **Shared-weight Sketch Net**



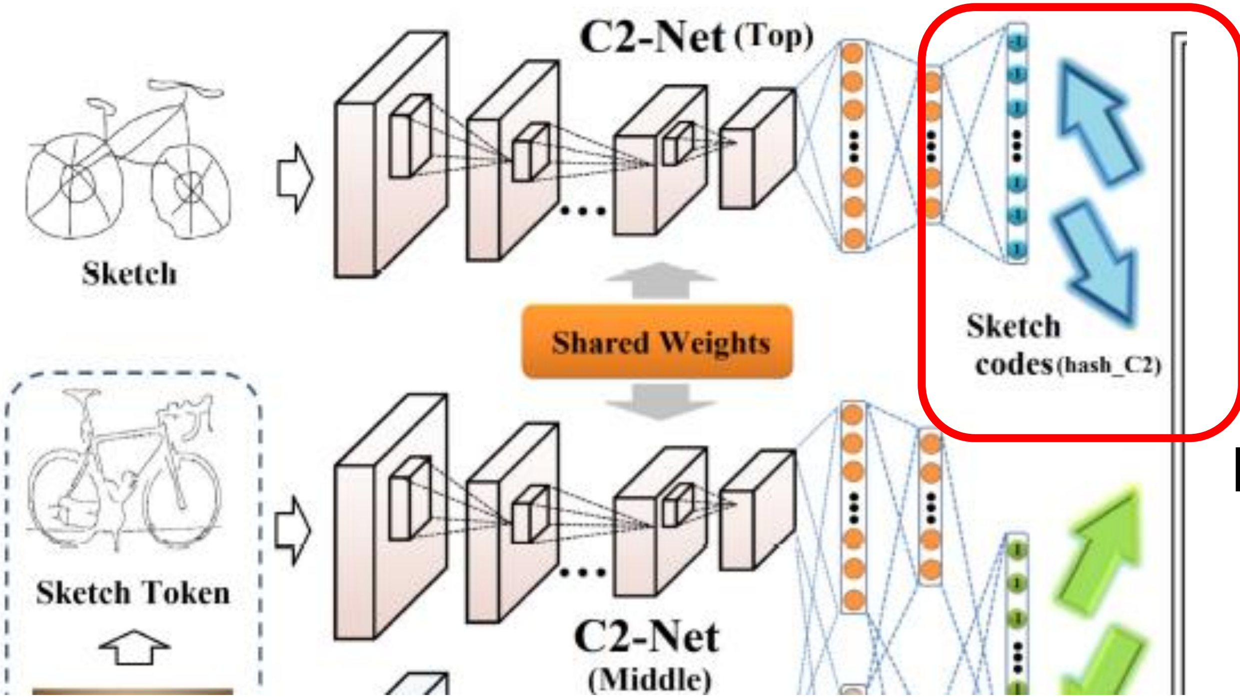
Siamese architecture

for C2-Net(Top) and C2-Net(Middle)

consider the **similar characteristics** and **implicit correlations** existing between sketch-tokens and free-hand sketches

Semi-heterogeneous Deep Architecture

- **Shared-weight Sketch Net**

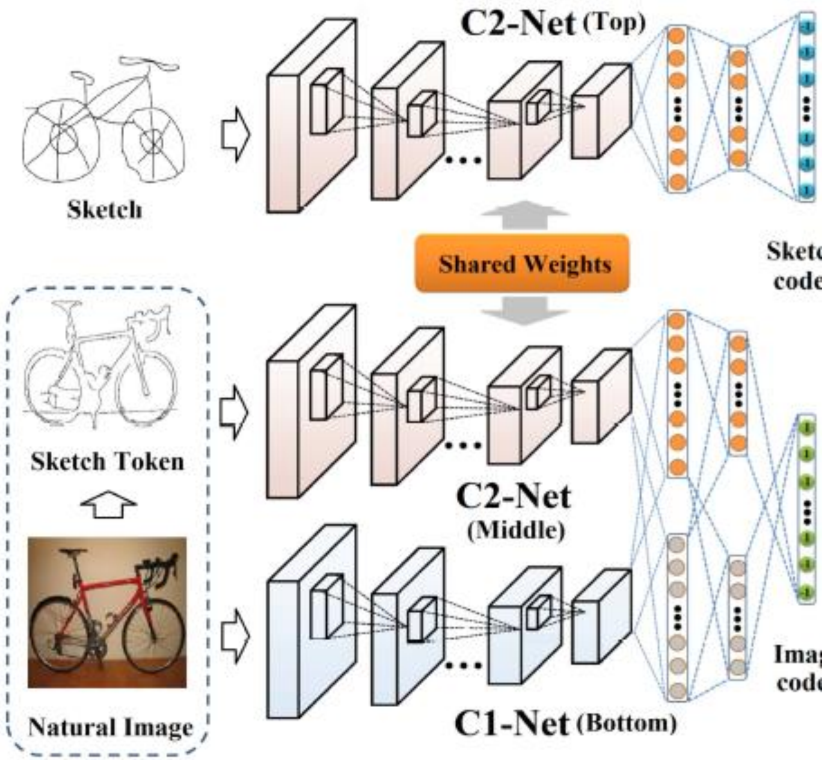


Binary coding layer
hash_C2

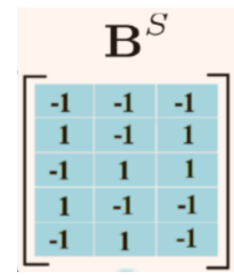
hash codes of free-hand sketches learned
shared-weight net will decrease the geometric difference
between images and sketches during SBIR.

Semi-heterogeneous Deep Architecture

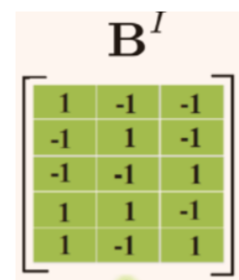
- Result : Deep hash function **B**



$B^S = \mathrm{sign}(F_2(A))$



$B^I = \mathrm{sign}(F_1(B, C))$

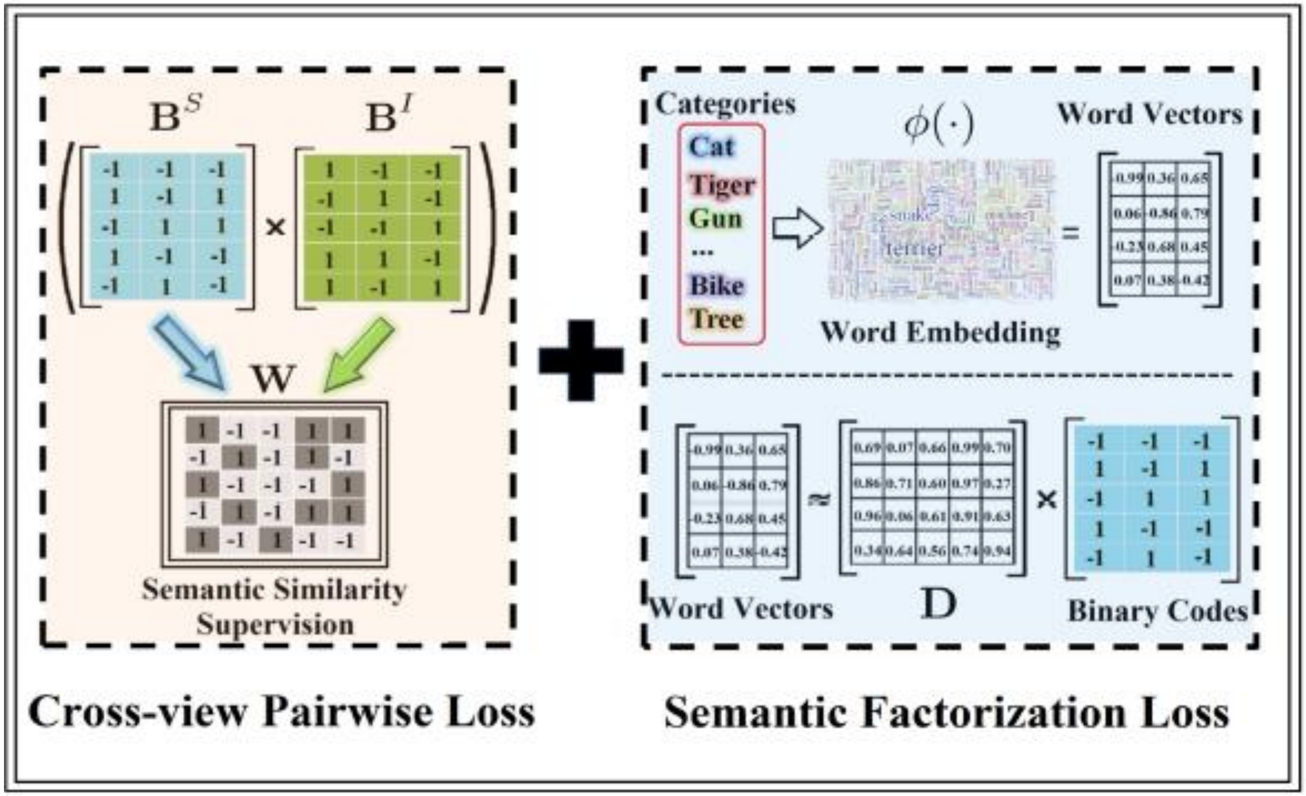


A = weights of C2(Top) : Sketch

B, C = weights of C2(Middle),C1 : Sketch-token, natural image

Discrete binary code learning

- **There are two loss function**
  - **Cross-view Pairwise Loss**
  - **Semantic Factorization Loss**

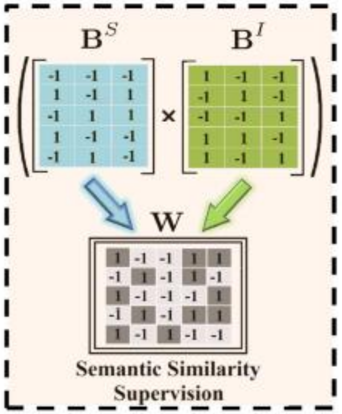


KAIST

Discrete binary code learning

- **Cross-view Pairwise Loss**
  - denotes the cross-view similarity between sketch and natural image

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



Cross-view Pairwise Loss

  - The binary codes of natural images and sketches from **the same category will be pulled** as close as possible (pushed far away otherwise)

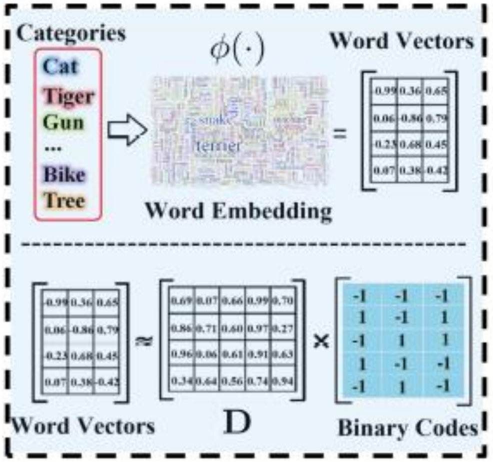
Discrete binary code learning

- **Semantic Factorization Loss**

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

$\phi(\cdot)$: Word embedding model

Y : label matrix



Semantic Factorization Loss

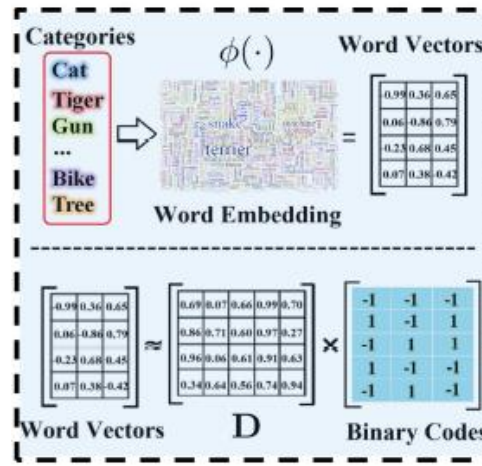
- Consider preserving the intra-set semantic relationships for both the image set and the sketch set
- Using Word2Vector, consider distance of label's semantic

Discrete binary code learning

- **Semantic Factorization Loss**

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

$\phi(\cdot)$: Word embedding model
Y : label matrix



Semantic Factorization Loss

- The semantic embedding of **"cheetah" will be closer to "tiger" but further from "dolphin"**

Discrete binary code learning

- **Final Objective Function**
  - Cross-view Pairwise Loss + Semantic Factorization Loss

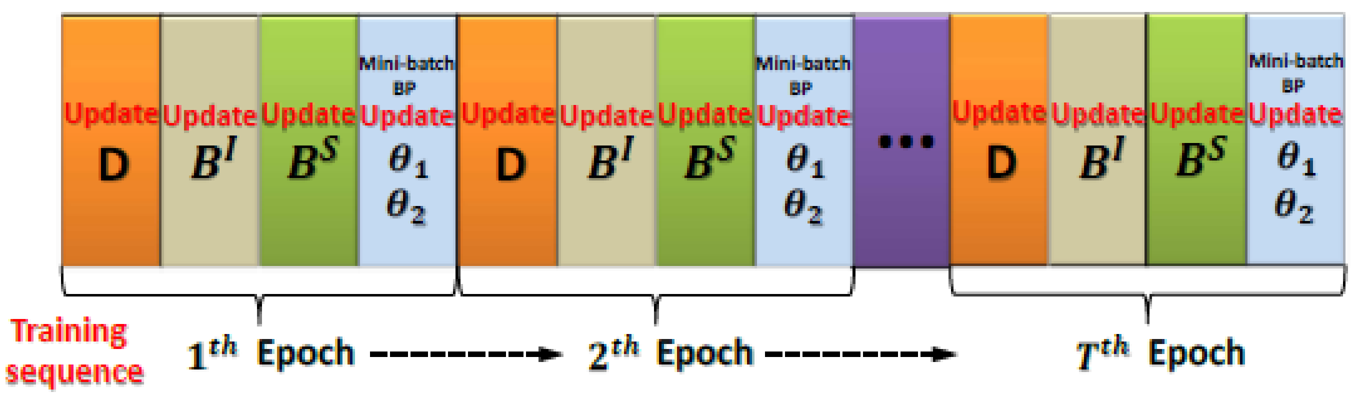
$$\min_{\mathbf{B}^I,\mathbf{B}^S,\mathbf{D}^I,\mathbf{D}^S,\Theta_1,\Theta_2} \mathcal{J} := ||\mathbf{W} \odot m - \mathbf{B}^{I^\top}\mathbf{B}^S||^2$$
$$+ \lambda(||\phi(\mathbf{Y}^I) - \mathbf{D}\mathbf{B}^I||^2 + ||\phi(\mathbf{Y}^S) - \mathbf{D}\mathbf{B}^S||^2)$$
$$+ \gamma(||\mathbf{F}_1(\mathcal{O}_1;\Theta_1,\Theta_2)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

Here, $\lambda > 0$ and $\gamma > 0$ are the balance parameters. The last two regularization terms aim to minimize the quantization loss between binary codes $\mathbf{B}^I$, $\mathbf{B}^S$ and deep hash functions $\mathbf{F}_1(\mathcal{O}_1;\Theta_1,\Theta_2)$, $\mathbf{F}_2(\mathcal{O}_2;\Theta_2)$. Similar regularization terms are also used in [50, 36] for effective hash code learning. Next, we will elaborate on how to optimize problem (3).

Optimization (training)

- The objective function is non-convex and non-smooth, which is in general an NP-hard problem **due to the binary constraints**

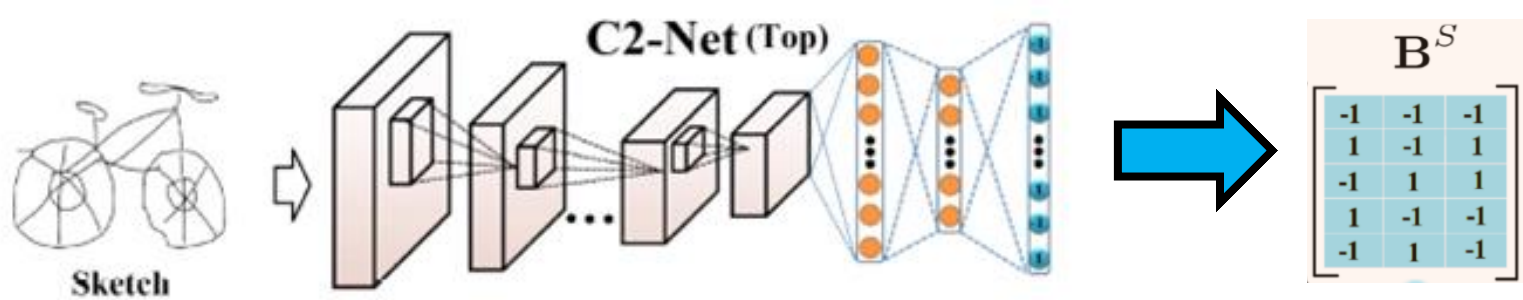
- **Solution : sequentially update parameters**
  - param : D, BI, BS and deep hash functions F1, F2



The illustration of DSH alternating optimization scheme

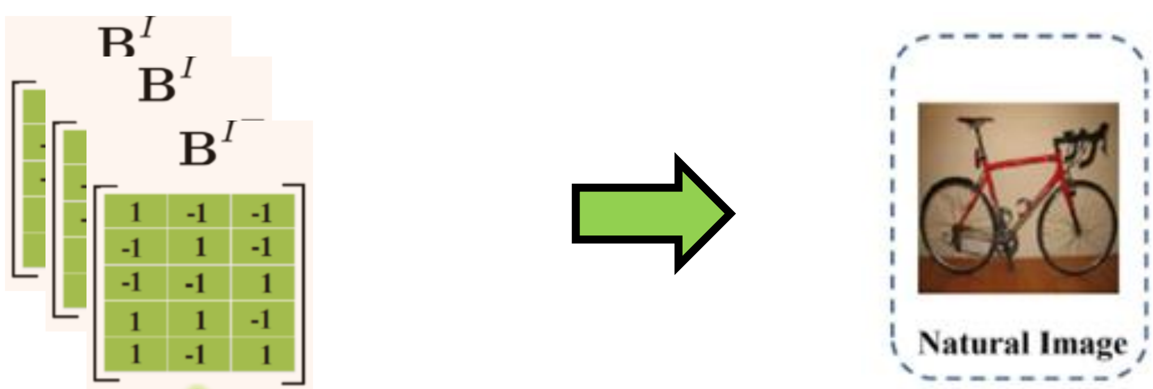
Test

- Given sketch query

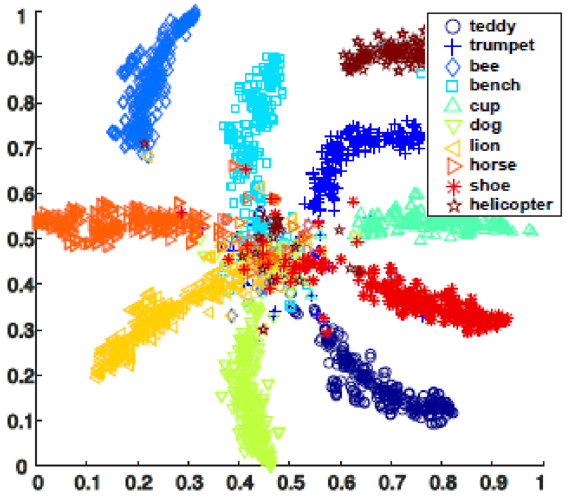


$B^S = \mathrm{sign}(F_2(A))$

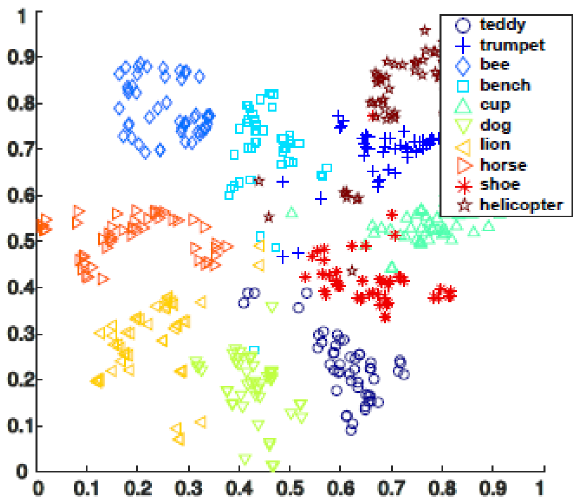
- Compare the distance with B^I's in retrieval database



Result



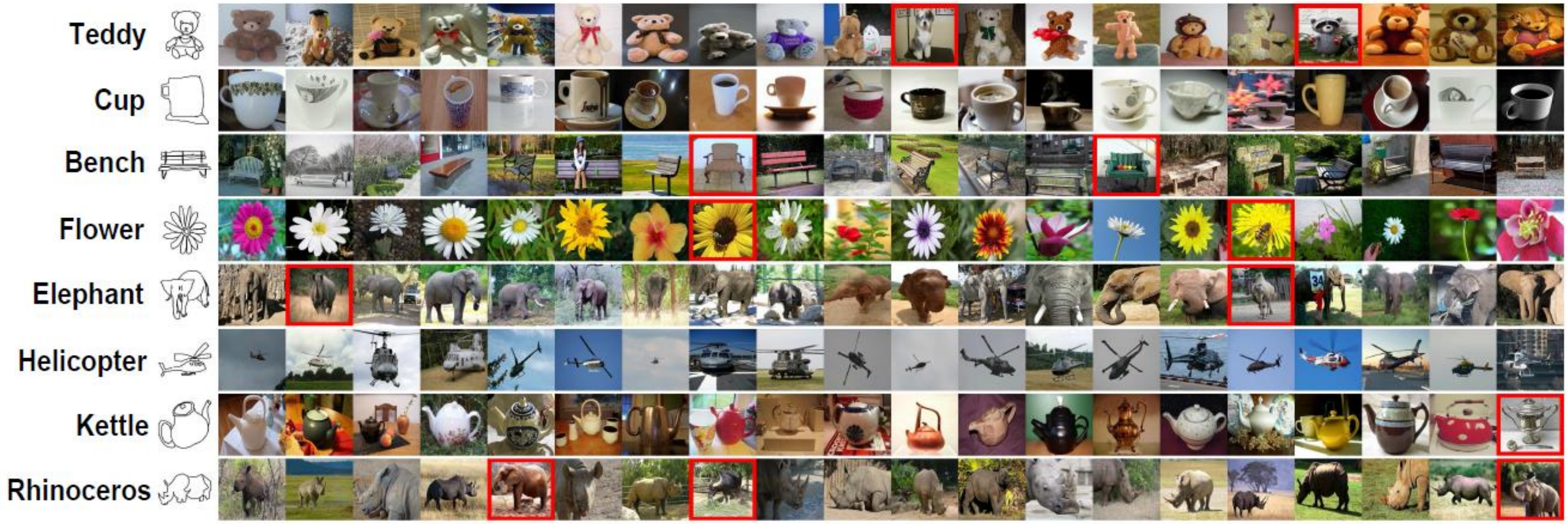
(a) Image retrieval gallery



(b) Test sketch queries

Experiments

- Data set
  - TU-Berlin Extension, Sketchy
  - All image has relatively complex backgrounds
- **Top-20 retrieval results (Red box : false positive)**



KAIST

Result

- **Comparison on other SBIR methods**

Methods	Dimension	TU-Berlin Extension MAP	Precision @200	Retrieval time per query (s)	Memory load(MB) (204,489 gallery images)
HOG [8]	1296	0.091	0.120	1.43	2.02×10^3
GF-HOG [18]	3500	0.119	0.148	4.13	5.46×10^3
SHELO [49]	1296	0.123	0.155	1.44	2.02×10^3
LKS [50]	1350	0.157	0.204	1.51	2.11×10^3
Siamese CNN [46]	64	0.322	0.447	7.70×10^{-2}	99.8
SaN [67]	512	0.154	0.225	0.53	7.98×10^2
GN Triplet* [52]	1024	0.187	0.301	1.02	1.60×10^3
3D shape* [61]	64	0.054	0.072	7.53×10^{-2}	99.8 MB
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 (Proposed)	32 (bits)	0.358	0.486	5.57×10^{-4}	0.78
	64 (bits)	0.521	0.655	7.03×10^{-4}	1.56
	128 (bits)	**0.570**	**0.694**	1.05×10^{-3}	3.12

End