

Learning to Hash for Big Data: A Tutorial

李武军

LAMDA Group

南京大学计算机科学与技术系
软件新技术国家重点实验室

Nov 29, 2015

Outline

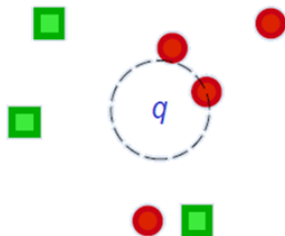
- 1 Introduction
- 2 Unsupervised Hashing
- 3 Supervised Hashing
- 4 Ranking-based Hashing
- 5 Multimodal Hashing
- 6 Deep Hashing
- 7 Quantization
- 8 Conclusion
- 9 Reference

Outline

- 1 Introduction
- 2 Unsupervised Hashing
- 3 Supervised Hashing
- 4 Ranking-based Hashing
- 5 Multimodal Hashing
- 6 Deep Hashing
- 7 Quantization
- 8 Conclusion
- 9 Reference

Nearest Neighbor Search (Retrieval)

- Given a query point q , return the points **closest** (similar) to q in the database (e.g., image database).



- Underlying many **machine learning, data mining, information retrieval** problems.

Big Data

Big data has attracted much attention from both academia and industry.

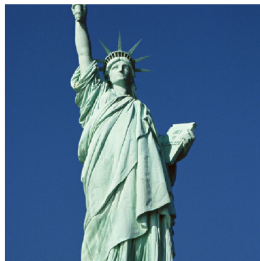
- Facebook: 750 million users
- Flickr: 6 billion photos
- Wal-Mart: 267 million items/day; 4PB data warehouse
- Sloan Digital Sky Survey: New Mexico telescope captures 200 GB image data/day

Nearest Neighbor Search (NNS) for Big Data

Challenge in big data applications:

- Curse of dimensionality
- Storage cost
- Query speed

Similarity Preserving Hashing



$h(\text{Statue of Liberty}) =$
10001010



$h(\text{Napoléon}) =$
01100001



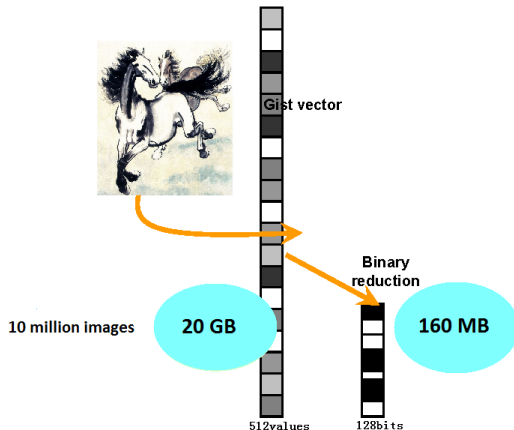
$h(\text{Napoléon}) =$
01100101

flipped bit

Should be very different

Should be similar

Reduce Dimensionality and Storage Cost



Fast Query Speed

- By using hash-code to construct **index**, we can achieve **constant** or **sub-linear** search time complexity.
- In some cases, **exhaustive search** with linear time complexity is also acceptable because the distance calculation cost is low with binary representation.

Two Stages of Hash Function Learning

Two main categories:

- **Category I:**

- Projection Stage (Dimension Reduction)
 - Projected with real-valued projection function
 - Given a point \mathbf{x} , each projected dimension i will be associated with a real-valued projection function $f_i(\mathbf{x})$ (e.g., $f_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x}$)
- Quantization Stage
 - Turn real into binary
 - Essential difference between metric learning and learning to hash

- **Category II:**

- Binary-Code Learning Stage
- Hash Function Learning Stage

Data-Independent Methods

The hash function family is defined **independently** of the training dataset.

- **Locality-sensitive hashing (LSH):** (Gionis et al., 1999; Andoni and Indyk, 2008) and its extensions (Datar et al., 2004; Kulis and Grauman, 2009; Kulis et al., 2009).
- **SIKH:** Shift invariant kernel hashing (SIKH) (Raginsky and Lazebnik, 2009).
- **MinHash** (Broder et al., 1998) and the extensions (Li and König, 2011)

Hash function: **random projections** or **manually constructed**.

They do not belong to **learning to hash** methods. Hence, this kind of methods will not be included in this tutorial.

Data-Dependent Methods

Hash functions are learned from a given training dataset.

- Compared with data-independent methods, data-dependent methods (also called **learning to hash** methods) can achieve comparable or even better accuracy with shorter binary codes.

Seminal papers: (Salakhutdinov and Hinton, 2007, 2009; Torralba et al., 2008; Weiss et al., 2008)

Two categories:

- Unimodal
 - Supervised methods
given some supervised (semantic) information, such as **pairwise labels** s_{ij} , **point-wise labels** y_i or **triplet labels** $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$
 - Unsupervised methods
- Multimodal
 - Supervised methods
 - Unsupervised methods

(Unimodal) Unsupervised Methods

No labels to denote the categories of the training points.

- **PCA**: principal component analysis.
- **SH**: (Weiss et al., 2008) eigenfunctions computed from the data similarity graph.
- **AGH**: anchor graph-based hashing (Liu et al., 2011).
- **ITQ**: (Gong and Lazebnik, 2011) orthogonal rotation matrix to refine the initial projection matrix learned by PCA.
- **IsoHash**: (Kong and Li, 2012b) orthogonal rotation matrix to make the variances of different directions isotropic (equal)
- **DGH**: (Liu et al., 2014) directly learn the discrete binary code
- **SGH**: (Jiang and Li, 2015) scalable graph hashing with feature transformation

(Unimodal) Supervised (semi-supervised) Methods

Class labels or pairwise constraints:

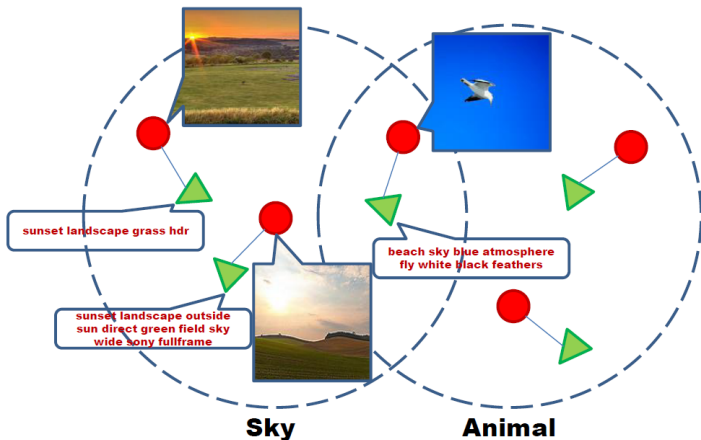
- **SSH (SPLH):** Semi-supervised hashing (Wang et al., 2010a,b) exploiting both labeled data and unlabeled data
- **MLH:** Minimal loss hashing (Norouzi and Fleet, 2011) based on the latent structural SVM framework
- **LDAHash:** Linear discriminant analysis based hashing (Strecha et al., 2012)
- **KSH:** Kernel-based supervised hashing (Liu et al., 2012)
- **LFH:** Latent factor models for supervised hashing (Zhang et al., 2014)
- **FastH:** (Lin et al., 2014) Supervised hashing using graph cut and decision trees
- **SDH:** (Shen et al., 2015) Supervised discrete hashing with point-wise labels
- **COSDISH:** (Kang et al., 2016) Scalable discrete hashing with pairwise supervision

Ranking-based Methods

The supervised information is ranking labels, such as triplets $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$.

- **HDML**: Hamming distance metric learning (Norouzi et al., 2012)
- **OPH**: Order preserving hashing for approximate nearest neighbor search (Wang et al., 2013b)
- **RSH**: Learning hash codes with listwise supervision (Wang et al., 2013a)
- **RPH**: Ranking preserving hashing for fast similarity search (Wang et al., 2015)

Multimodal Methods



- Multi-Source Hashing
- Cross-Modal Hashing

Multi-Source Hashing

- Aim at learning better codes than unimodal hashing by leveraging auxiliary views.
 - Assume that all the views are provided for a query.
-
- **MFH**: Multiple feature hashing (Song et al., 2011)
 - **CH**: Composite hashing (Zhang et al., 2011)

Cross-Modal Hashing

Given a query of either image or text, return images or texts similar to it.

- **CVH**: Cross view hashing (Kumar and Udupa, 2011)
- **MLBE**: Multimodal latent binary embedding (Zhen and Yeung, 2012a)
- **CRH**: Co-regularized hashing (Zhen and Yeung, 2012b)
- **IMH**: Inter-media hashing (Song et al., 2013)
- **RaHH**: Relation-aware heterogeneous hashing (Ou et al., 2013)
- **SCM**: Semantic correlation maximization (Zhang and Li, 2014)
- **CMFH**: Collective matrix factorization hashing (Ding et al., 2014)
- **QCH**: Quantized correlation hashing (Wu et al., 2015)
- **SePH**: Semantics-preserving hashing (Lin et al., 2015b)

Deep Hashing

Deep learning for hashing

- **CNNH**: Supervised hashing via image representation learning (Xia et al., 2014)
- **NINH**: Simultaneous feature learning and hash coding with deep neural networks (Lai et al., 2015)
- **DSRH**: Deep semantic ranking based hashing (Zhao et al., 2015)
- **DRSCH**: Bit-scalable deep hashing (Zhang et al., 2015)
- **DH**: Deep hashing for compact binary codes learning (Liong et al., 2015)
- Deep learning of binary hash codes (Lin et al., 2015a)
- **DPSH**: Feature learning based deep supervised hashing with pairwise labels (Li et al., 2015)

Quantization

The quantization stage is at least as important as the projection stage:

- **DBQ:** Double-bit quantization (Kong and Li, 2012a)
- **MQ:** Manhattan quantization (Kong et al., 2012)
- **VBQ:** Variable bit quantization (Moran et al., 2013)

Outline

- 1 Introduction
- 2 Unsupervised Hashing**
- 3 Supervised Hashing
- 4 Ranking-based Hashing
- 5 Multimodal Hashing
- 6 Deep Hashing
- 7 Quantization
- 8 Conclusion
- 9 Reference

Problem Definition

Input:

- Feature vectors: $\{\mathbf{x}_i\}$ (compact matrix form for all training points: \mathbf{X}).

Output:

- Binary codes: $\{\mathbf{b}_i\}$ (compact matrix form for all training points: \mathbf{B}).
Instances similar in the original feature space should have similar binary codes.

or

- When \mathbf{x}_i is close to \mathbf{x}_j , the Hamming distance between \mathbf{b}_i and \mathbf{b}_j should be low.
- When \mathbf{x}_i is far away from \mathbf{x}_j , the Hamming distance between \mathbf{b}_i and \mathbf{b}_j should be high.

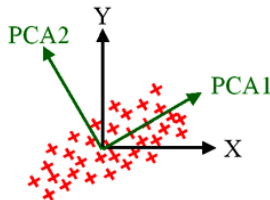
PCA Hashing (PCAH)

To generate a code of m bits, PCAH performs PCA on \mathbf{X} , and then use the top m eigenvectors of the matrix $\mathbf{X}\mathbf{X}^T$ as columns of the projection matrix $\mathbf{W} \in \mathbb{R}^{d \times m}$. Here, top m eigenvectors are those corresponding to the m largest eigenvalues $\{\lambda_k\}_{k=1}^m$, generally arranged with the non-increasing order $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m$. Let $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_m]^T$. Then

$$\Lambda = \mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W} = \text{diag}(\lambda)$$

Define hash function

$$h(\mathbf{x}) = \text{sgn}(\mathbf{W}^T \mathbf{x})$$



Spectral Hashing (Weiss et al., 2008)

$$\begin{aligned}
 & \min_{\{\mathbf{y}_i\}} \sum_{ij} W_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \\
 & \text{subject to : } \mathbf{y}_i \in \{-1, 1\}^k \\
 & \quad \sum_i \mathbf{y}_i = \mathbf{0} \\
 & \quad \frac{1}{n} \sum_i \mathbf{y}_i \mathbf{y}_i^T = \mathbf{I}
 \end{aligned}$$

where W_{ij} is the similarity between \mathbf{x}_i and \mathbf{x}_j , the constraint $\sum_i \mathbf{y}_i = \mathbf{0}$ requires each bit to be fire 50% of the time, and the constraint $\frac{1}{n} \sum_i \mathbf{y}_i \mathbf{y}_i^T = \mathbf{I}$ requires the bits to be uncorrelated.

NP hard problem!

Spectral Hashing (Weiss et al., 2008)

In matrix form, and by relaxation:

$$\begin{aligned} \min \operatorname{tr}(\mathbf{Y}^T \mathcal{L} \mathbf{Y}) \\ \text{subject to : } \mathbf{Y}^T \mathbf{1} = 0 \\ \frac{1}{n} \mathbf{Y}^T \mathbf{Y} = \mathbf{I} \end{aligned}$$

where \mathbf{Y} is a **real-valued** $n \times k$ matrix whose j th row is \mathbf{y}_j^T , $\mathcal{L} = \mathbf{D} - \mathbf{W}$ is the Laplacian matrix, \mathbf{D} is a diagonal matrix with $D(i, i) = \sum_j W(i, j)$.

Solution: simply the k eigenvectors of \mathcal{L} with minimal eigenvalues after excluding the trivial eigenvector $\mathbf{1}$ which has eigenvalue 0.

$\operatorname{sign}(\mathbf{Y})$ to get the binary codes (quantization stage).

Out-of-sample extension with eigenfunctions by simply fitting a multidimensional rectangle distribution to the data (by using PCA to align the axes, and then assuming a uniform distribution on each axis).

Spectral Hashing (Weiss et al., 2008)

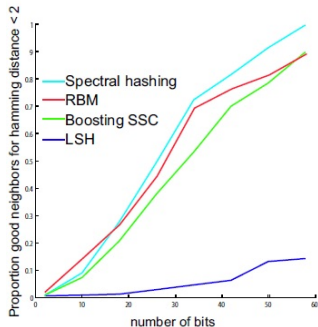


Figure 5: Performance of different binary codes on the LabelMe dataset described in [3]. The data is certainly not uniformly distributed, and yet spectral hashing gives better retrieval performance than boosting and LSH.

Anchor Graph Hashing (AGH) (Liu et al., 2011)

$$\min \frac{1}{2} \sum_{ij} \|Y_i - Y_j\|^2 A_{ij} = \text{tr}(Y^T L Y)$$

$$\text{subject to : } Y \in \{1, -1\}^{n \times r}, \mathbf{1}^T Y = 0, Y^T Y = n I_{r \times r}$$

where A_{ij} is the similarity between points i and j .

The same objective function as that in SH.

Problem: the time complexity of direct eigen-decomposition is $O(n^3)$.

Solution: K-means clustering to obtain m ($m \ll n$) cluster centers (**anchors**) $\mathcal{U} = \{\mathbf{u}_j \in \mathbb{R}^d\}_{j=1}^m$

$$Z_{ij} = \begin{cases} \frac{\exp(-\mathcal{D}^2(\mathbf{x}_i, \mathbf{u}_j)/t)}{\sum_{j' \in \langle i \rangle} \exp(-\mathcal{D}^2(\mathbf{x}_i, \mathbf{u}_{j'})/t)}, & \forall j \in \langle i \rangle \\ 0, & \text{otherwise} \end{cases}$$

where $\langle i \rangle \subset [1 : m]$ denotes the s ($s \ll m$) nearest anchors of \mathbf{x}_i .

Anchor Graph Hashing (AGH) (Liu et al., 2011)

Anchor graph: $\hat{A} = Z\Lambda Z^T$ where $\Lambda = \text{diag}(Z^T \mathbf{1})$.

The solution is the r graph Laplacian eigenvectors associated with the smallest eigenvalues, which are also eigenvectors of \hat{A} associated with the r largest eigenvalues (ignoring eigenvalue 1).

$$Y = \sqrt{n}Z\Lambda^{-1/2}V\Sigma^{-1/2} = ZW$$

where V and Σ are the eigenvectors and eigenvalues of the small $m \times m$ matrix $\Lambda^{-1/2}Z^T\Lambda^{-1/2}$.

The hash function: $h(x) = \text{sign}(W^T z(x))$

Time complexity: Decrease from $O(n^3)$ to $O(nm^2)$.

Anchor Graph Hashing (AGH) (Liu et al., 2011)

Hierarchical quantization:

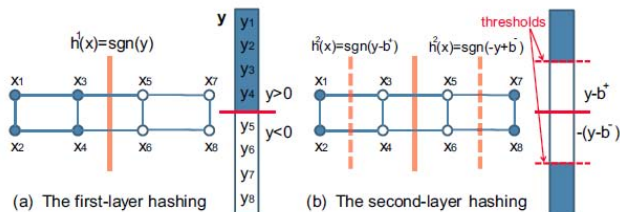


Figure 1. Hierarchical hashing on a data graph. x_1, \dots, x_8 are data points and y is a graph Laplacian eigenvector. The data points of filled circles take '1' hash bit and the others take '-1' hash bit. The entries with dark color in y are positive and the others are negative. (a) The first-layer hash function h^1 uses threshold 0 ; (b) the second-layer hash functions h^2 use thresholds b^+ and b^- .

Anchor Graph Hashing (AGH) (Liu et al., 2011)

Method	MNIST (70K)			
	MAP		Train Time	Test Time
	$r = 24$	$r = 48$	$r = 48$	$r = 48$
ℓ_2 Scan	0.4125		—	
SE ℓ_2 Scan	0.5269	0.3909	—	—
LSH	0.1613	0.2196	1.8	2.1×10^{-5}
PCAH	0.2596	0.2242	4.5	2.2×10^{-5}
USPLH	0.4699	0.4930	163.2	2.3×10^{-5}
SH	0.2699	0.2453	4.9	4.9×10^{-5}
KLSH	0.2555	0.3049	2.9	5.3×10^{-5}
SIKH	0.1947	0.1972	0.4	1.3×10^{-5}
1-AGH	0.4997	0.3971	22.9	5.3×10^{-5}
2-AGH	0.6738	0.6410	23.2	6.5×10^{-5}
BRE	0.2638	0.3090	57.9	6.7×10^{-5}

Iterative Quantization (ITQ) (Gong and Lazebnik, 2011)

$$\begin{aligned} \max \operatorname{tr}(W^T X X^T W) \\ \text{subject to : } W^T W = I \end{aligned}$$

It is just PCA. If W is a solution, then $\tilde{W} = WR$ is also a solution where R is an orthogonal rotation matrix.

Let $V = XW$ denote the projected data, and B denote the binary code. ITQ tries to minimize the following objective function:

$$\mathcal{Q}(B, R) = ||B - VR||$$

Learn by iterating the following two steps:

- Fix R and update B : $B = \operatorname{sign}(VR)$;
- Fix B and update R : first compute the SVD of $B^T V$ as $S\Omega\hat{S}^T$, then let $R = \hat{S}S^T$. It is the classic orthogonal Procrustes problem.

Iterative Quantization (ITQ) (Gong and Lazebnik, 2011)

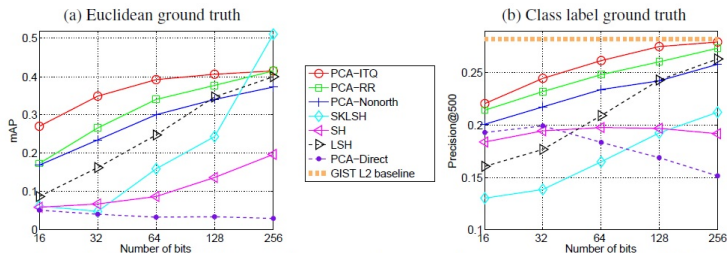
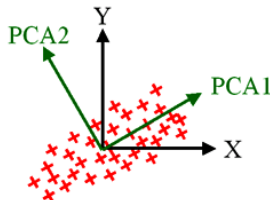


Figure 3. Comparative evaluation on CIFAR dataset. (a) Performance is measured by mean average precision (mAP) for retrieval using top 50 Euclidean neighbors of each query point as true positives. Refer to Figure 4 for the complete recall-precision curves for the state-of-the-art methods. (b) Performance is measured by the averaged precision of top p ranked images for each query where ground truth is defined by semantic class labels. Refer to Figure 5 for the complete class label precision curves for the state-of-the-art methods.

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

Problem:

All existing methods use the **same number of bits** for different projected dimensions with **different variances**.



Possible Solutions:

- Different number of bits for different dimensions (**Variable bit quantization** (Moran et al., 2013))
- Isotropic (equal) variances for all dimensions: **Isotropic hashing (IsoHash)** (Kong and Li, 2012b)

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

To generate a code of m bits, **PCA hashing (PCAH)** performs PCA on X , and then use the top m eigenvectors of the matrix XX^T as columns of the projection matrix $W \in \mathbb{R}^{d \times m}$. Here, top m eigenvectors are those corresponding to the m largest eigenvalues $\{\lambda_k\}_{k=1}^m$, generally arranged with the non-increasing order $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m$. Let $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_m]^T$. Then

$$\Lambda = W^T X X^T W = \text{diag}(\lambda)$$

Define hash function

$$h(\mathbf{x}) = \text{sgn}(W^T \mathbf{x})$$

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

Weakness of PCA Hash:

Using the **same number of bits** for different projected dimensions is **unreasonable** because larger-variance dimensions will carry more information.

Solve it by making variances equal (isotropic)!

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

Idea of IsoHash:

- Learn an **orthogonal** matrix $Q \in \mathbb{R}^{m \times m}$ which makes $Q^T W^T X X^T W Q$ become a matrix with **equal diagonal values**.
- **Effect of Q** : to make each projected dimension has the **same variance** while **keeping the Euclidean distances** between any two points unchanged.

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

Problem Definition:

$$\text{tr}(Q^T W^T X X^T W Q) = \text{tr}(W^T X X^T W) = \text{tr}(\Lambda) = \sum_{i=1}^m \lambda_i$$

$$\mathbf{a} = [a_1, a_2, \dots, a_m] \text{ with } a_i = a = \frac{\sum_{i=1}^m \lambda_i}{m},$$

and

$$\mathcal{T}(\mathbf{z}) = \{T \in \mathbb{R}^{m \times m} | \text{diag}(T) = \text{diag}(\mathbf{z})\},$$

Problem

*The **problem of IsoHash** is to find an orthogonal matrix Q making $Q^T W^T X X^T W Q \in \mathcal{T}(\mathbf{a})$.*

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

IsoHash Formulation:

Because $Q^T \Lambda Q = Q^T [W^T X X^T W] Q$, let

$$\mathcal{M}(\Lambda) = \{Q^T \Lambda Q | Q \in \mathcal{O}(m)\},$$

where $\mathcal{O}(m)$ is the set of all orthogonal matrices in $\mathbb{R}^{m \times m}$.

Then, the IsoHash problem is equivalent to:

$$\|T - Z\|_F = 0,$$

where $T \in \mathcal{T}(\mathbf{a})$, $Z \in \mathcal{M}(\Lambda)$, $\|\cdot\|_F$ denotes the Frobenius norm.

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

Lemma

[Schur-Horn Lemma (Horn, 1954)] Let $\mathbf{c} = \{c_i\} \in \mathbb{R}^m$ and $\mathbf{b} = \{b_i\} \in \mathbb{R}^m$ be real vectors in non-increasing order respectively, i.e., $c_1 \geq c_2 \geq \dots \geq c_m$, $b_1 \geq b_2 \geq \dots \geq b_m$. There exists a Hermitian matrix H with eigenvalues \mathbf{c} and diagonal values \mathbf{b} if and only if

$$\sum_{i=1}^k b_i \leq \sum_{i=1}^k c_i, \text{ for any } k = 1, 2, \dots, m,$$

$$\sum_{i=1}^m b_i = \sum_{i=1}^m c_i.$$

So we can prove :

There **exists a solution** to the IsoHash problem. And this solution is in the intersection of $\mathcal{T}(\mathbf{a})$ and $\mathcal{M}(\Lambda)$.

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

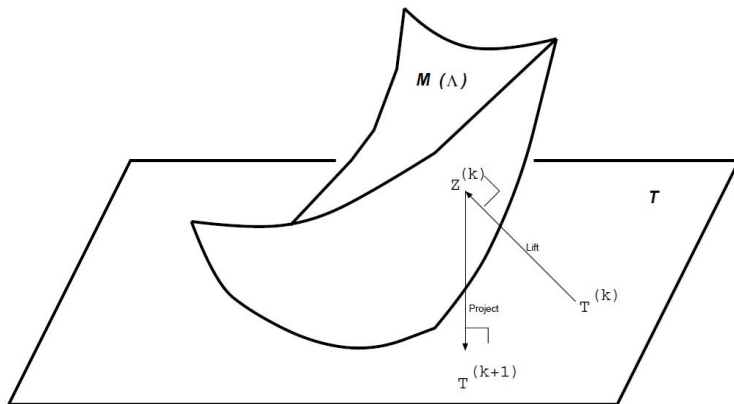
Learning Methods:

Two methods: (Chu, 1995)

- Lift and projection (LP)
- Gradient Flow (GF)

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

Learning Method: Lift and projection (LP)



Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

Learning Method: Gradient Flow

- Objective function:

$$\min_{Q \in \mathcal{O}(m)} F(Q) = \frac{1}{2} \|\text{diag}(Q^T \Lambda Q) - \text{diag}(\mathbf{a})\|_F^2.$$

- The gradient ∇F at Q :

$$\nabla F(Q) = 2\Lambda\beta(Q),$$

where $\beta(Q) = \text{diag}(Q^T \Lambda Q) - \text{diag}(\mathbf{a})$.

- The projection of $\nabla F(Q)$ onto $\mathcal{O}(m)$

$$g(Q) = Q[Q^T \Lambda Q, \beta(Q)]$$

where $[A, B] = AB - BA$ is the Lie bracket.

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

Learning Method: Gradient Flow

The vector field $\dot{Q} = -g(Q)$ defines a steepest descent flow on the manifold $\mathcal{O}(m)$ for function $F(Q)$. Letting $Z = Q^T \Lambda Q$ and $\alpha(Z) = \beta(Q)$, we get

$$\dot{Z} = [Z, [\alpha(Z), Z]],$$

where \dot{Z} is an isospectral flow that moves to reduce the objective function $F(Q)$.

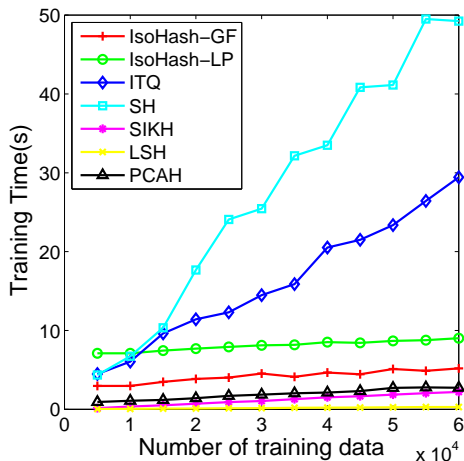
Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

Accuracy (mAP):

Method	CIFAR				
# bits	32	64	96	128	256
IsoHash	0.2249	0.2969	0.3256	0.3357	0.3651
PCAH	0.0319	0.0274	0.0241	0.0216	0.0168
ITQ	0.2490	0.3051	0.3238	0.3319	0.3436
SH	0.0510	0.0589	0.0802	0.1121	0.1535
SIKH	0.0353	0.0902	0.1245	0.1909	0.3614
LSH	0.1052	0.1907	0.2396	0.2776	0.3432

Isotropic Hashing (IsoHash) (Kong and Li, 2012b)

Training Time:



Discrete Graph Hashing (DGH) (Liu et al., 2014)

$$\min_{\mathbf{B}} \frac{1}{2} \sum_{ij}^n \|\mathbf{b}_i - \mathbf{b}_j\|^2 A_{ij}^o = \text{tr}(\mathbf{B}^T \mathbf{L}^o \mathbf{B})$$

$$\text{subject to : } \mathbf{B} \in \{1, -1\}^{n \times r}, \mathbf{1}^T \mathbf{B} = \mathbf{0}, \mathbf{B}^T \mathbf{B} = n \mathbf{I}_r$$

Problem

- Solving this problem in the discrete code space is NP hard.
- Relaxing the discrete constraints will lead to poor performance.

Solution

- Leverage the anchor graph \mathbf{A} like AGH.
- Soften the constraint $\mathbf{1}^T \mathbf{B} = \mathbf{0}$ and the constraint $\mathbf{B}^T \mathbf{B} = n \mathbf{I}_r$

Discrete Graph Hashing (DGH) (Liu et al., 2014)

Define a set $\Omega = \{\mathbf{Y} \in \mathbb{R}^{n \times r} | \mathbf{1}^T \mathbf{Y} = \mathbf{0}, \mathbf{Y}^T \mathbf{Y} = n \mathbf{I}_r\}$

$$\begin{aligned} \max_{\mathbf{B}} \quad & \text{tr}(\mathbf{B}^T \mathbf{A} \mathbf{B}) - \frac{\rho}{2} \text{dist}^2(\mathbf{B}, \Omega) \\ \text{subject to: } & \mathbf{B} \in \{1, -1\}^{n \times r} \end{aligned}$$

where $\text{dist}(\mathbf{B}, \Omega) = \min_{\mathbf{Y} \in \Omega} \|\mathbf{B} - \mathbf{Y}\|_F$

Learn by iterating the following two steps:

- B-subproblem:

$$\max_{\mathbf{B} \in \{-1, +1\}^{n \times r}} f(\mathbf{B}) := \text{tr}(\mathbf{B}^T \mathbf{A} \mathbf{B}) + \rho \text{tr}(\mathbf{Y}^T \mathbf{B})$$

- Y-subproblem:

$$\begin{aligned} \max_{\mathbf{Y} \in \mathbb{R}^{n \times r}} \quad & \text{tr}(\mathbf{B}^T \mathbf{Y}) \\ \text{subject to: } & \mathbf{1}^T \mathbf{Y} = \mathbf{0}, \mathbf{Y}^T \mathbf{Y} = n \mathbf{I}_r \end{aligned}$$

Discrete Graph Hashing (DGH) (Liu et al., 2014)

B-subproblem

- Use an iterative ascent procedure called *Signed Gradient Method*
Repeat $\mathbf{B}^{k+1} = \text{sgn}(\mathcal{C}(2\mathbf{A}\mathbf{B}^k + \rho\mathbf{Y}, \mathbf{B}^k))$ until \mathbf{B}^k converges.
Where

$$\mathcal{C}(x, y) = \begin{cases} x, & x \neq 0 \\ y, & x = 0 \end{cases}$$

Y-subproblem

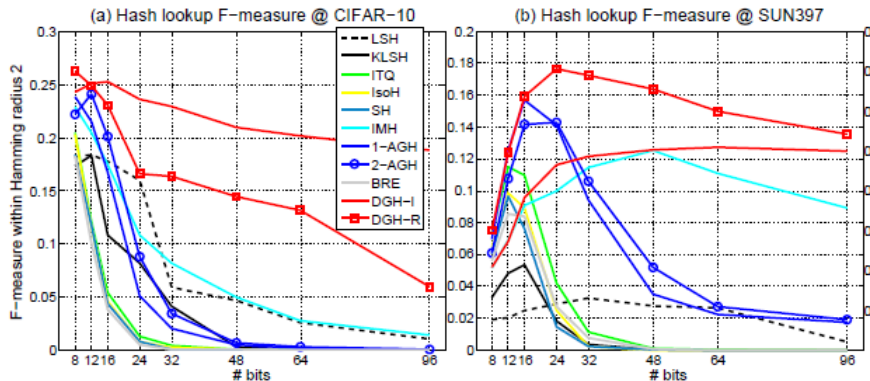
- Solve Y-subproblem by using a closed-form solution:
 $\mathbf{Y} = \sqrt{n}[\mathbf{U} \bar{\mathbf{U}}][\mathbf{V} \bar{\mathbf{V}}]$, where $\mathbf{U}\Sigma\mathbf{V}^T = (\mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{B}$

Discrete Graph Hashing (DGH) (Liu et al., 2014)

Table 1: Hamming ranking performance on **YouTube Faces** and **Tiny-1M**. r denotes the number of hash bits used in the hashing methods. All training and test times are in seconds.

Method	YouTube Faces					Tiny-1M				
	Mean Precision / Top-2K			TrainTime	TestTime	Mean Precision / Top-20K			TrainTime	TestTime
	$r = 48$	$r = 96$	$r = 128$	$r = 128$	$r = 128$	$r = 48$	$r = 96$	$r = 128$	$r = 128$	$r = 128$
ℓ_2 Scan	0.7591			-		1			-	
LSH	0.0830	0.1005	0.1061	6.4	1.8×10^{-5}	0.1155	0.1324	0.1766	6.1	1.0×10^{-5}
KLSH	0.3982	0.5210	0.5871	16.1	4.8×10^{-5}	0.3054	0.4105	0.4705	20.7	4.6×10^{-5}
ITQ	0.7017	0.7493	0.7562	169.0	1.8×10^{-5}	0.3925	0.4726	0.5052	297.3	1.0×10^{-5}
IsoH	0.6093	0.6962	0.7058	73.6	1.8×10^{-5}	0.3896	0.4816	0.5161	13.5	1.0×10^{-5}
SH	0.5897	0.6655	0.6736	108.9	2.0×10^{-4}	0.1857	0.1923	0.2079	61.4	1.6×10^{-4}
MDSH	0.6110	0.6752	0.6795	118.8	4.9×10^{-5}	0.3312	0.3878	0.3955	193.6	2.8×10^{-5}
IMH	0.3150	0.3641	0.3889	92.1	2.3×10^{-5}	0.2257	0.2497	0.2557	139.3	2.7×10^{-5}
1-AGH	0.7138	0.7571	0.7646	84.1	2.1×10^{-5}	0.4061	0.4117	0.4107	141.4	3.4×10^{-5}
2-AGH	0.6727	0.7377	0.7521	94.7	3.5×10^{-5}	0.3925	0.4099	0.4152	272.5	4.7×10^{-5}
BRE	0.5564	0.6238	0.6483	10372.0	9.0×10^{-5}	0.3943	0.4836	0.5218	8419.0	8.8×10^{-5}
DGH-I	0.7086	0.7644	0.7750	402.6	2.1×10^{-5}	0.4045	0.4865	0.5178	1769.4	3.3×10^{-5}
DGH-R	0.7245	0.7672	0.7805	408.9	2.1×10^{-5}	0.4208	0.5006	0.5358	2793.4	3.3×10^{-5}

Discrete Graph Hashing (DGH) (Liu et al., 2014)



Scalable Graph Hashing (SGH) (Jiang and Li, 2015)

Problem

- The memory cost and time complexity are at least $\mathcal{O}(n^2)$ for graph hashing if all pairwise similarities are explicitly computed.
- How to utilize the whole graph and avoid $\mathcal{O}(n^2)$ complexity?

Scalable Graph Hashing(SGH)

- A **feature transformation** (Shrivastava and Li, 2014) method to effectively approximate the whole graph without **explicitly** computing it.
- A sequential method for bit-wise complementary learning.
- Linear complexity.

Scalable Graph Hashing (SGH) (Jiang and Li, 2015)

Objective function

$$\min_{\mathbf{W}} \|\mathbf{c}\tilde{\mathbf{S}} - \text{sgn}(K(\mathbf{X})\mathbf{W}^T)\text{sgn}(K(\mathbf{X})\mathbf{W}^T)^T\|_F^2$$

$$s.t. \quad \mathbf{W}K(\mathbf{X})^TK(\mathbf{X})\mathbf{W}^T = \mathbf{I}$$

- $\forall \mathbf{x}$, define: $K(\mathbf{x}) = [\phi(\mathbf{x}, \mathbf{x}_1) - \sum_{i=1}^n \phi(\mathbf{x}_i, \mathbf{x}_1)/n, \dots, \phi(\mathbf{x}, \mathbf{x}_m) - \sum_{i=1}^n \phi(\mathbf{x}_i, \mathbf{x}_m)/n]$
- $\tilde{S}_{ij} = 2S_{ij} - 1 \in (-1, 1]$.

Notation

- $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}^T \in \mathbb{R}^{n \times d}$: n data points.
- Pairwise similarity metric defined as: $S_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|_F^2 / \rho} \in (0, 1]$

Scalable Graph Hashing (SGH) (Jiang and Li, 2015)

Feature transformation

- $\forall \mathbf{x}$, define $P(\mathbf{x})$ and $Q(\mathbf{x})$:

$$P(\mathbf{x}) = \left[\sqrt{\frac{2(e^2 - 1)}{e\rho}} e^{-\frac{\|\mathbf{x}\|_F^2}{\rho}} \mathbf{x}; \sqrt{\frac{e^2 + 1}{e}} e^{-\frac{\|\mathbf{x}\|_F^2}{\rho}}; 1 \right]$$

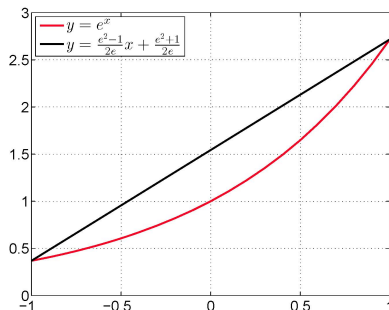
$$Q(\mathbf{x}) = \left[\sqrt{\frac{2(e^2 - 1)}{e\rho}} e^{-\frac{\|\mathbf{x}\|_F^2}{\rho}} \mathbf{x}; \sqrt{\frac{e^2 + 1}{e}} e^{-\frac{\|\mathbf{x}\|_F^2}{\rho}}; -1 \right]$$

- $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$

$$\begin{aligned} P(\mathbf{x}_i)^T Q(\mathbf{x}_j) &= 2 \left[\frac{e^2 - 1}{2e} \times \frac{2\mathbf{x}_i^T \mathbf{x}_j}{\rho} + \frac{e^2 + 1}{2e} \right] e^{-\frac{\|\mathbf{x}_i\|_F^2 + \|\mathbf{x}_j\|_F^2}{\rho}} - 1 \\ &\approx 2e^{-\frac{\|\mathbf{x}_i\|_F^2 - \|\mathbf{x}_j\|_F^2 + 2\mathbf{x}_i^T \mathbf{x}_j}{\rho}} - 1 = \tilde{S}_{ij} \end{aligned}$$

Scalable Graph Hashing (SGH) (Jiang and Li, 2015)

- Here, we use an approximation $\frac{e^2-1}{2e}x + \frac{e^2+1}{2e} \approx e^x$



- We assume $-1 \leq \frac{2}{\rho} \mathbf{x}_i^T \mathbf{x}_j \leq 1$. It is easy to prove that $\rho = 2 \max\{\|\mathbf{x}_i\|_F^2\}_{i=1}^n$ can make $-1 \leq \frac{2}{\rho} \mathbf{x}_i^T \mathbf{x}_j \leq 1$.
- Then we have $\tilde{\mathbf{S}} \approx P(\mathbf{X})^T Q(\mathbf{X})$

Scalable Graph Hashing (SGH) (Jiang and Li, 2015)

- **Direct relaxation** may lead to poor performance. We adopt a **sequential learning** strategy in a bit-wise complementary manner

- Residual definition:

$$\mathbf{R}_t = c\tilde{\mathbf{S}} - \sum_{i=1}^{t-1} \text{sgn}(K(\mathbf{X})\mathbf{w}_i)\text{sgn}(K(\mathbf{X})\mathbf{w}_i)^T$$

$$\mathbf{R}_1 = c\tilde{\mathbf{S}}$$

- Objective function:

$$\begin{aligned} \min_{\mathbf{w}_t} \quad & \|\mathbf{R}_t - \text{sgn}(K(\mathbf{X})\mathbf{w}_t)\text{sgn}(K(\mathbf{X})\mathbf{w}_t)^T\|_F^2 \\ \text{s.t.} \quad & \mathbf{w}_t^T K(\mathbf{X})^T K(\mathbf{X})\mathbf{w}_t = 1 \end{aligned}$$

By relaxation, we can get:

$$\begin{aligned} \max_{\mathbf{w}_t} \quad & \text{tr}(\mathbf{w}_t^T K(\mathbf{X})^T \mathbf{R}_t K(\mathbf{X})\mathbf{w}_t) \\ \text{s.t.} \quad & \mathbf{w}_t^T K(\mathbf{X})^T K(\mathbf{X})\mathbf{w}_t = 1 \end{aligned}$$

Scalable Graph Hashing (SGH) (Jiang and Li, 2015)

- Then we obtain a generalized eigenvalue problem:

$$K(\mathbf{X})^T \mathbf{R}_t K(\mathbf{X}) \mathbf{w}_t = \lambda K(\mathbf{X})^T K(\mathbf{X}) \mathbf{w}_t$$

- Key component:

$$\begin{aligned} cK(\mathbf{X})^T \tilde{\mathbf{S}} K(\mathbf{X}) &= cK(\mathbf{X})^T \underbrace{P(\mathbf{X})^T Q(\mathbf{X})}_{\tilde{\mathbf{S}}} K(\mathbf{X}) \\ &= c[K(\mathbf{X})^T P(\mathbf{X})^T][Q(\mathbf{X}) K(\mathbf{X})] \end{aligned}$$

- Adopting $\mathbf{R}_t = c\tilde{\mathbf{S}} - \sum_{i=1, i \neq t}^c \text{sgn}(K(\mathbf{X})\mathbf{w}_i) \text{sgn}(K(\mathbf{X})\mathbf{w}_t)^T$, we continue the sequential learning procedure.
- The information in $\tilde{\mathbf{S}}$ is fully used, but is not explicitly computed. Time complexity is decreased from $O(n^2)$ to $O(n)$.

Scalable Graph Hashing (SGH) (Jiang and Li, 2015)

Algorithm 1 Sequential learning algorithm for SGH

Input: Feature vectors $\mathbf{X} \in \mathbb{R}^{n \times d}$; code length c ; number of kernel bases m .

Output: Weight matrix $\mathbf{W} \in \mathbb{R}^{c \times m}$.

Procedure

Construct $P(\mathbf{X})$ and $Q(\mathbf{X})$;

Construct $K(\mathbf{X})$ based on the kernel bases, which are m points randomly selected from \mathbf{X} ;

$\mathbf{A}_0 = [K(\mathbf{X})^T P(\mathbf{X})^T][Q(\mathbf{X}) K(\mathbf{X})]$;

$\mathbf{A}_1 = c\mathbf{A}_0$;

$\mathbf{Z} = K(\mathbf{X})^T K(\mathbf{X}) + \gamma \mathbf{I}_d$;

for $t = 1 \rightarrow c$ **do**

 Solve the following generalized eigenvalue problem

$\mathbf{A}_t \mathbf{w}_t = \lambda \mathbf{Z} \mathbf{w}_t$;

$\mathbf{U} = [K(\mathbf{X})^T \text{sgn}(K(\mathbf{X}) \mathbf{w}_t)][K(\mathbf{X})^T \text{sgn}(K(\mathbf{X}) \mathbf{w}_t)]^T$;

$\mathbf{A}_{t+1} = \mathbf{A}_t - \mathbf{U}$;

end for

$\hat{\mathbf{A}}_0 = \mathbf{A}_{c+1}$

Randomly permute $\{1, 2, \dots, c\}$ to generate a random index set \mathcal{M} ;

for $t = 1 \rightarrow c$ **do**

$\hat{t} = \mathcal{M}(t)$;

$\hat{\mathbf{A}}_0 = \hat{\mathbf{A}}_0 + K(\mathbf{X})^T \text{sgn}(K(\mathbf{X}) \mathbf{w}_{\hat{t}}) \text{sgn}(K(\mathbf{X}) \mathbf{w}_{\hat{t}})^T K(\mathbf{X})$;

 Solve the following generalized eigenvalue problem

$\hat{\mathbf{A}}_0 \mathbf{v} = \lambda \mathbf{Z} \mathbf{v}$;

 Update $\mathbf{w}_{\hat{t}} \leftarrow \mathbf{v}$

$\hat{\mathbf{A}}_0 = \hat{\mathbf{A}}_0 - K(\mathbf{X})^T \text{sgn}(K(\mathbf{X}) \mathbf{w}_{\hat{t}}) \text{sgn}(K(\mathbf{X}) \mathbf{w}_{\hat{t}})^T K(\mathbf{X})$;

end for

Scalable Graph Hashing (SGH) (Jiang and Li, 2015)

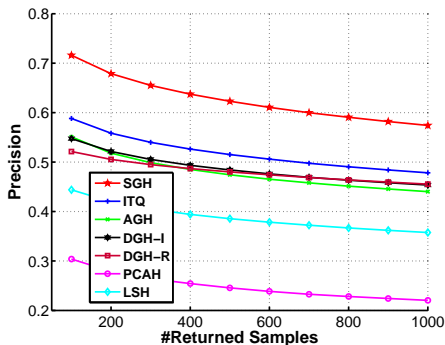
Top-1k precision @TINY-1M.

Method	32 bits	64 bits	96 bits	128 bits	256 bits
SGH	0.4697	0.5742	0.6299	0.6737	0.7357
ITQ	0.4289	0.4782	0.4947	0.4986	0.5003
AGH	0.3973	0.4402	0.4577	0.4654	0.4767
DGH-I	0.3974	0.4536	0.4737	0.4874	0.4969
DGH-R	0.3793	0.4554	0.4871	0.4989	0.5276
PCAH	0.2457	0.2203	0.2000	0.1836	0.1421
LSH	0.2507	0.3575	0.4122	0.4529	0.5212

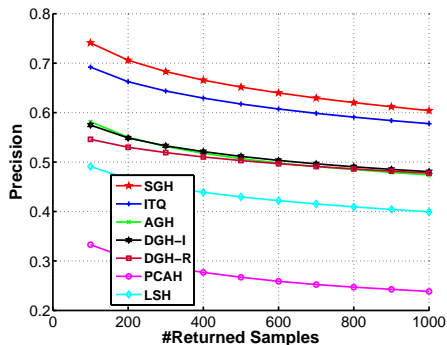
Top-1k precision @MIRFLICKR-1M.

Method	32 bits	64 bits	96 bits	128 bits	256 bits
SGH	0.4919	0.6041	0.6677	0.6985	0.7584
ITQ	0.5177	0.5776	0.5999	0.6096	0.6228
AGH	0.4299	0.4741	0.4911	0.4998	0.506
DGH-I	0.4299	0.4806	0.5001	0.5111	0.5253
DGH-R	0.4121	0.4776	0.5054	0.5196	0.5428
PCAH	0.2720	0.2384	0.2141	0.1950	0.1508
LSH	0.2597	0.3995	0.466	0.5160	0.6072

Scalable Graph Hashing (SGH) (Jiang and Li, 2015)



(a) 64 bits @TINY-1M



(b) 64 bits @MIRFLICKR-1M

Scalable Graph Hashing (SGH) (Jiang and Li, 2015)

Training time @TINY-1M. Here, $t_1 = 1438.60$

Method	32 bits	64 bits	96 bits	128 bits	256 bits
SGH	34.49	52.37	71.53	89.65	164.23
ITQ	31.72	60.62	89.01	149.18	322.06
AGH	$18.60 + t_1$	$19.40 + t_1$	$20.08 + t_1$	$22.48 + t_1$	$25.09 + t_1$
DGH-I	$187.57 + t_1$	$296.99 + t_1$	$518.57 + t_1$	$924.08 + t_1$	$1838.30 + t_1$
DGH-R	$217.06 + t_1$	$360.18 + t_1$	$615.74 + t_1$	$1089.10 + t_1$	$2300.10 + t_1$
PCAH	4.29	4.54	4.75	5.85	6.49
LSH	1.68	1.77	1.84	2.55	3.76

Training time @MIRFLICKR-1M. Here, $t_2 = 1564.86$

Method	32 bits	64 bits	96 bits	128 bits	256 bits
SGH	41.51	59.02	74.86	97.25	168.35
ITQ	36.17	64.61	89.50	132.71	285.10
AGH	$17.99 + t_2$	$18.80 + t_2$	$20.30 + t_2$	$19.87 + t_2$	$21.60 + t_2$
DGH-I	$85.81 + t_2$	$143.68 + t_2$	$215.41 + t_2$	$352.73 + t_2$	$739.56 + t_2$
DGH-R	$116.25 + t_2$	$206.24 + t_2$	$308.32 + t_2$	$517.97 + t_2$	$1199.44 + t_2$
PCAH	7.65	7.90	8.47	9.23	10.42
LSH	2.44	2.43	2.71	3.38	4.21

Comments on Unsupervised Methods

Although some methods are mainly for **unsupervised setting**, they can be adapted for supervised setting if the label information is available.

For example, we can also use the label information to construct the graph in SH and AGH to get the supervised counterparts.

However, these adapted unsupervised methods can not make effective use of **both \mathbf{X} and \mathbf{Y}** for training.

The **supervised methods** introduced later typically try to consider both \mathbf{X} and \mathbf{Y} for training.

Outline

- 1 Introduction
- 2 Unsupervised Hashing
- 3 Supervised Hashing**
- 4 Ranking-based Hashing
- 5 Multimodal Hashing
- 6 Deep Hashing
- 7 Quantization
- 8 Conclusion
- 9 Reference

Problem Definition

Input:

- Feature vectors: $\{\mathbf{x}_i\}$ (compact form: \mathbf{X}).
- **Class labels**: $\{\mathbf{y}_i\}$ (compact form: \mathbf{Y}).
 or **Pairwise constraints**: $\{s_{ij}\}$ (Compact form: $\mathcal{S} = \{s_{ij}\}$)
 - $s_{ij} = 1$ if points i and j belong to the same class. **must link**
 - $s_{ij} = 0$ if points i and j belong to different classes. **cannot link**

Output:

- Binary codes: $\{\mathbf{b}_i\}$ (compact form: \mathbf{B}).
 Instances with similar labels should have similar binary codes.
or
 - When $s_{ij} = 1$, the Hamming distance between \mathbf{b}_i and \mathbf{b}_j should be low.
 - When $s_{ij} = 0$, the Hamming distance between \mathbf{b}_i and \mathbf{b}_j should be high.

(Unimodal) Supervised (semi-supervised) Methods

Class labels or pairwise constraints:

- **SSH (SPLH)**: Semi-supervised hashing (Wang et al., 2010a,b) exploiting both labeled data and unlabeled data
- **MLH**: Minimal loss hashing (Norouzi and Fleet, 2011) based on the latent structural SVM framework
- **LDAHash**: Linear discriminant analysis based hashing (Strecha et al., 2012)
- **KSH**: Kernel-based supervised hashing (Liu et al., 2012)
- **LFH**: Latent factor models for supervised hashing (Zhang et al., 2014)
- **FastH**: (Lin et al., 2014) Supervised hashing using graph cut and decision trees
- **SDH**: (Shen et al., 2015) Supervised discrete hashing with point-wise labels
- **COSDISH**: (Kang et al., 2016) Scalable discrete hashing with pairwise supervision

Semi-Supervised Hashing (SSH) (Wang et al., 2010a,b)

$$\begin{aligned}
 J(\mathbf{W}) &= \frac{1}{2} \text{tr}\{\mathbf{W}^T \mathbf{X}_l \mathbf{S} \mathbf{X}_l^T \mathbf{W}\} + \frac{\eta}{2} \text{tr}[\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}] \\
 &= \frac{1}{2} \text{tr}\{\mathbf{W}^T [\mathbf{X}_l \mathbf{S} \mathbf{X}_l^T + \eta \mathbf{X} \mathbf{X}^T] \mathbf{W}\} \\
 &= \frac{1}{2} \text{tr}\{\mathbf{W}^T \mathbf{M} \mathbf{W}\}
 \end{aligned}$$

where \mathbf{S} contains the supervised information:

$$S_{ij} = \begin{cases} 1 : (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \\ -1 : (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \\ 0 : otherwise \end{cases}$$

Semi-Supervised Hashing (SSH) (Wang et al., 2010a,b)

Sequential projection learning algorithm:

Algorithm 1 Semi-supervised sequential projection learning for hashing (S3PLH)

Input: data \mathbf{X} , pairwise labeled data \mathbf{X}_l , initial pairwise labels \mathbf{S}_1 , length of hash codes K , constant α

for $k = 1$ **to** K **do**

 Compute adjusted covariance matrix:

$$\mathbf{M}_k = \mathbf{X}_l \mathbf{S}_k \mathbf{X}_l^\top + \eta \mathbf{X} \mathbf{X}^\top$$

 Extract the first eigenvector \mathbf{e} of \mathbf{M}_k and set:

$$\mathbf{w}_k = \mathbf{e}$$

 Update the labels from vector \mathbf{w}_k :

$$\mathbf{S}_{k+1} = \mathbf{S}_k - \alpha \mathbf{T}(\tilde{\mathbf{S}}^k, \mathbf{S}_k)$$

 Compute the residual:

$$\mathbf{X} = \mathbf{X} - \mathbf{w}_k \mathbf{w}_k^\top \mathbf{X}$$

end for

Semi-Supervised Hashing (SSH) (Wang et al., 2010a,b)

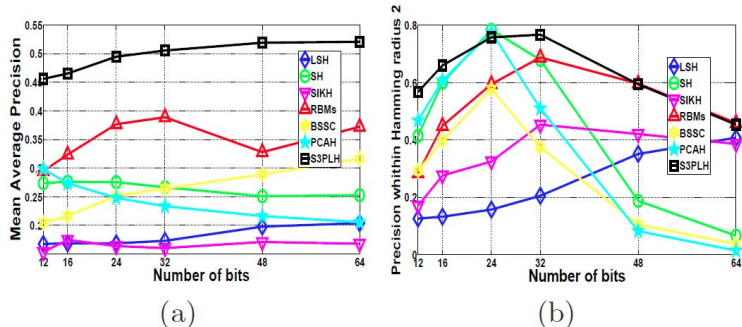


Figure 3. Results on MNIST dataset. a) MAP for different number of bits using Hamming ranking; b) Precision within Hamming radius 2 using hash lookup.

Minimal Loss Hashing (MLH) (Norouzi and Fleet, 2011)

$$b(\mathbf{x}, \mathbf{w}) = \text{sign}(W\mathbf{x})$$

$$\mathcal{L}(\mathbf{w}) = \sum_{(i,j) \in S} L(b(\mathbf{x}_i; \mathbf{w}), b(\mathbf{x}_j; \mathbf{w}), s_{ij})$$

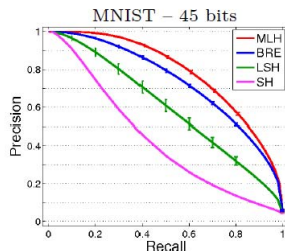
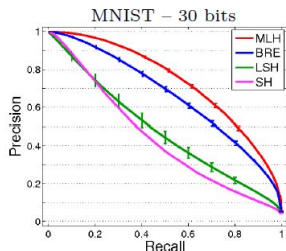
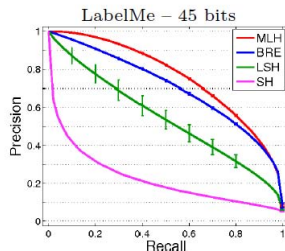
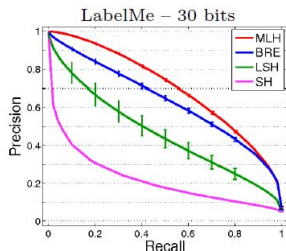
$$L(\mathbf{h}, \mathbf{g}, s) = \ell_\rho(\|\mathbf{h} - \mathbf{g}\|_H, s)$$

where $\ell_\rho()$ is a hinge-like loss function:

$$\ell_\rho(m, s) = \begin{cases} \max(m - \rho + 1, 0), & \text{for } s = 1 \\ \lambda \max(\rho - m + 1, 0), & \text{for } s = 0 \end{cases}$$

Formulated as a structural SVM problem.

Minimal Loss Hashing (MLH) (Norouzi and Fleet, 2011)



LDAHash (Strecha et al., 2012)

Given a positive-pair set $\mathcal{P} = \{(\mathbf{x}, \mathbf{x}')\}$ and a negative-pair set $\mathcal{N} = \{(\mathbf{x}, \mathbf{x}')\}$, learn the hash function parameterized by \mathbf{P} and \mathbf{t} :

$$\mathbf{y} = \text{sign}(\mathbf{P}\mathbf{x} + \mathbf{t})$$

$$\Sigma_{\mathcal{P}} = \mathbb{E} \{ (\mathbf{x} - \mathbf{x}')(\mathbf{x} - \mathbf{x}')^T | \mathcal{P} \}$$

$$\Sigma_{\mathcal{N}} = \mathbb{E} \{ (\mathbf{x} - \mathbf{x}')(\mathbf{x} - \mathbf{x}')^T | \mathcal{N} \}$$

Linear Discriminant Analysis (LDA)

$$\begin{aligned} \tilde{L} &\propto \text{tr} \{ \mathbf{P} \Sigma_{\mathcal{N}}^{-1/2} \Sigma_{\mathcal{P}} \Sigma_{\mathcal{N}}^{-T/2} \mathbf{P}^T \} \\ &= \text{tr} \{ \mathbf{P} \Sigma_{\mathcal{P}} \Sigma_{\mathcal{N}}^{-1} \mathbf{P}^T \} = \text{tr} \{ \mathbf{P} \Sigma_{\mathcal{R}} \mathbf{P}^T \}, \end{aligned}$$

Difference of Covariances (DIF)

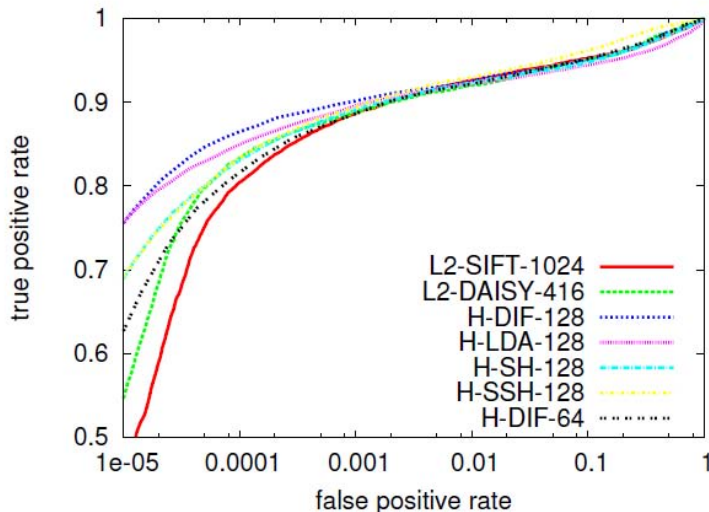
$$\tilde{L} = \text{tr} \{ \mathbf{P} \Sigma_{\mathcal{D}} \mathbf{P}^T \}$$

$$\Sigma_{\mathcal{D}} = \alpha \Sigma_{\mathcal{P}} - \Sigma_{\mathcal{N}}$$

$$\mathbf{P} = (\Sigma_{\mathcal{D}})^{-1/2}$$

$$\begin{aligned} \min_{t_i} \quad & \mathbb{E} \{ \text{sign}((\mathbf{p}_i^T \mathbf{x} + t_i)(\mathbf{p}_i^T \mathbf{x}' + t_i)) | \mathcal{N} \} \\ & - \alpha \mathbb{E} \{ \text{sign}((\mathbf{p}_i^T \mathbf{x} + t_i)(\mathbf{p}_i^T \mathbf{x}' + t_i)) | \mathcal{P} \} \end{aligned}$$

LDAHash (Strecha et al., 2012)

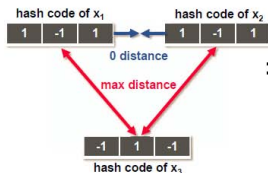


Supervised Hashing with Kernels (KSH) (Liu et al., 2012)

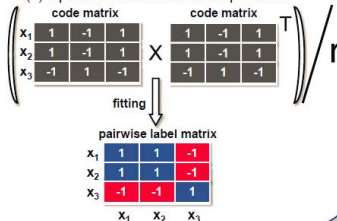
(a) The labeled data

supervised
hashing

(b) Optimization on Hamming distances



(c) Optimization on code inner products



Supervised Hashing with Kernels (KSH) (Liu et al., 2012)

$$f(\mathbf{x}) = \sum_{j=1}^m \left(\kappa(\mathbf{x}_{(j)}, \mathbf{x}) - \frac{1}{n} \sum_{i=1}^n \kappa(\mathbf{x}_{(j)}, \mathbf{x}_i) \right) \mathbf{a}_j \\ = \mathbf{a}^\top \bar{\mathbf{k}}(\mathbf{x}),$$

$$\bar{\mathbf{k}}(\mathbf{x}) = [\kappa(\mathbf{x}_{(1)}, \mathbf{x}) - \mu_1, \dots, \kappa(\mathbf{x}_{(m)}, \mathbf{x}) - \mu_m]^\top,$$

$$S_{ij} = \begin{cases} 1, & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \\ -1, & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \\ 0, & \text{otherwise.} \end{cases}$$

$$\min_{A \in \mathbb{R}^{m \times r}} \mathcal{Q}(A) = \left\| \frac{1}{r} \text{sgn}(\bar{K}_l A) (\text{sgn}(\bar{K}_l A))^\top - S \right\|_F^2$$

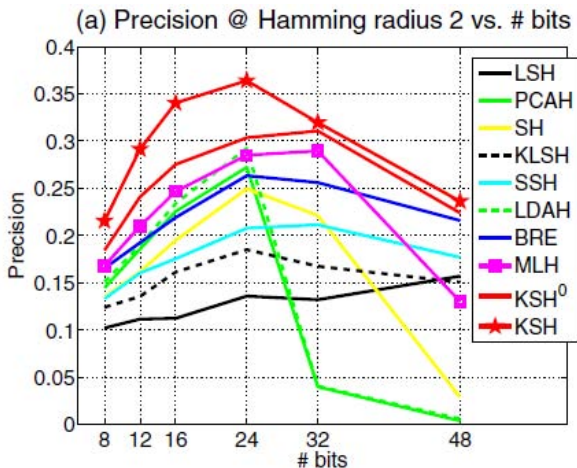
where $\bar{K}_l = [\bar{\mathbf{k}}(\mathbf{x}_1), \dots, \bar{\mathbf{k}}(\mathbf{x}_l)]^\top \in \mathbb{R}^{l \times m}$
and $A = [\mathbf{a}_1, \dots, \mathbf{a}_r] \in \mathbb{R}^{m \times r}$

$$\max_{\mathbf{a}_k} (\bar{K}_l \mathbf{a}_k)^\top R_{k-1} (\bar{K}_l \mathbf{a}_k) \\ \text{s.t. } (\bar{K}_l \mathbf{a}_k)^\top (\bar{K}_l \mathbf{a}_k) = l$$

$$R_{k-1} = rS - \sum_{t=1}^{k-1} \text{sgn}(\bar{K}_l \mathbf{a}_t^*) (\text{sgn}(\bar{K}_l \mathbf{a}_t^*))^\top \\ \left\| \text{sgn}(\bar{K}_l \mathbf{a}_k) (\text{sgn}(\bar{K}_l \mathbf{a}_k))^\top - R_{k-1} \right\|_F^2$$

$$\bar{K}_l^\top R_{k-1} \bar{K}_l \mathbf{a} = \lambda \bar{K}_l^\top \bar{K}_l \mathbf{a}$$

Supervised Hashing with Kernels (KSH) (Liu et al., 2012)



Latent Factor Hashing (LFH) (Zhang et al., 2014)

Motivation

Existing supervised methods:

- High training complexity
- Semantic information is poorly utilized

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Contribution

- A novel likelihood function based on latent factor models
- A learning algorithm with convergence guarantee
- A linear-time stochastic learning strategy for scalable training
- State-of-the-art accuracy

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Model

The likelihood on the observed similarity labels \mathcal{S} is defined as:

$$p(\mathcal{S} \mid \mathbf{B}) = \prod_{s_{ij} \in \mathcal{S}} p(s_{ij} \mid \mathbf{B})$$

$$p(s_{ij} \mid \mathbf{B}) = \begin{cases} a_{ij}, & s_{ij} = 1 \\ 1 - a_{ij}, & s_{ij} = 0 \end{cases}$$

a_{ij} is defined as $a_{ij} = \sigma(\Theta_{ij})$ with:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\Theta_{ij} = \frac{1}{2} \mathbf{b}_i^T \mathbf{b}_j$$

Relationship between the Hamming distance and the inner product:

$$\text{dist}_H(\mathbf{b}_i, \mathbf{b}_j) = \frac{1}{2}(Q - \mathbf{b}_i^T \mathbf{b}_j) = \frac{1}{2}(Q - 2\Theta_{ij})$$

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Hardness

The posteriori of \mathbf{B} can be computed as follows:

$$p(\mathbf{B} \mid \mathcal{S}) \sim p(\mathcal{S} \mid \mathbf{B})p(\mathbf{B})$$

We can learn the optimal \mathbf{B} through maximum a posteriori (MAP) estimation.

However, directly optimizing on \mathbf{B} is an **NP-hard** problem. Thus we optimize \mathbf{B} through two stages:

- 1 **Relax** \mathbf{B} to be a real valued matrix $\mathbf{U} \in \mathbb{R}^{N \times Q}$ (Latent factors).
Then learn an optimal \mathbf{U} under the same probabilistic framework as that for \mathbf{B} .
- 2 Get optimal \mathbf{B} from \mathbf{U} through some **rounding** techniques.

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Relaxation

Re-defined Θ_{ij} as:

$$\Theta_{ij} = \frac{1}{2} \mathbf{U}_{i*}^T \mathbf{U}_{j*}$$

$p(\mathcal{S} | \mathbf{B})$, $p(\mathbf{B})$, $p(\mathbf{B} | \mathcal{S})$ becomes $p(\mathcal{S} | \mathbf{U})$, $p(\mathbf{U})$, $p(\mathbf{U} | \mathcal{S})$.

Define a normal distribution of $p(\mathbf{U})$ as:

$$p(\mathbf{U}) = \prod_{d=1}^Q \mathcal{N}(\mathbf{U}_{*d} | \mathbf{0}, \beta \mathbf{I})$$

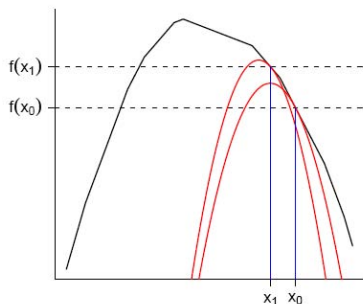
The log posteriori of \mathbf{U} can be derived as:

$$L = \log p(\mathbf{U} | \mathcal{S}) = \sum_{s_{ij} \in \mathcal{S}} (s_{ij} \Theta_{ij} - \log(1 + e^{\Theta_{ij}})) - \frac{1}{2\beta} \|\mathbf{U}\|_F^2 + c$$

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Surrogate algorithm

- Choose an initial point \mathbf{x}_0
- Construct a concave lower bound of $f(\mathbf{x})$ at \mathbf{x}_0
- Maximize this lower bound (at \mathbf{x}_1)
- Repeat



Latent Factor Hashing (LFH) (Zhang et al., 2014)

Alternating projection

Optimize each row of \mathbf{U} (denoted by \mathbf{U}_{i*}) at a time with other rows fixed using surrogate algorithm.

Gradient vector:

$$\frac{\partial L}{\partial \mathbf{U}_{i*}^T} = \frac{1}{2} \sum_{j:s_{ij} \in \mathcal{S}} (s_{ij} - a_{ij}) \mathbf{U}_{j*}^T + \frac{1}{2} \sum_{j:s_{ji} \in \mathcal{S}} (s_{ji} - a_{ji}) \mathbf{U}_{j*}^T - \frac{1}{\beta} \mathbf{U}_{i*}^T$$

Hessian matrix:

$$\begin{aligned} \frac{\partial^2 L}{\partial \mathbf{U}_{i*}^T \partial \mathbf{U}_{i*}} = & -\frac{1}{4} \sum_{j:s_{ij} \in \mathcal{S}} a_{ij}(1 - a_{ij}) \mathbf{U}_{j*}^T \mathbf{U}_{j*} \\ & -\frac{1}{4} \sum_{j:s_{ji} \in \mathcal{S}} a_{ji}(1 - a_{ji}) \mathbf{U}_{j*}^T \mathbf{U}_{j*} - \frac{1}{\beta} \mathbf{I} \end{aligned}$$

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Alternating projection

Define \mathbf{H}_i as:

$$\mathbf{H}_i = -\frac{1}{16} \sum_{j:s_{ij} \in \mathcal{S}} \mathbf{U}_{j*}^T \mathbf{U}_{j*} - \frac{1}{16} \sum_{j:s_{ji} \in \mathcal{S}} \mathbf{U}_{j*}^T \mathbf{U}_{j*} - \frac{1}{\beta} \mathbf{I}$$

We can prove that:

$$\frac{\partial^2 L}{\partial \mathbf{U}_{i*}^T \partial \mathbf{U}_{i*}} \succeq \mathbf{H}_i$$

$\mathbf{A} \succeq \mathbf{B}$ means $\mathbf{A} - \mathbf{B}$ is a positive semi-definite matrix.

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Surrogate function

Constructed lower bound of $L(\mathbf{U}_{i*})$:

$$\begin{aligned}\tilde{L}(\mathbf{U}_{i*}) = & L(\mathbf{U}_{i*}(t)) + (\mathbf{U}_{i*} - \mathbf{U}_{i*}(t)) \frac{\partial L}{\partial \mathbf{U}_{i*}^T}(t) \\ & + \frac{1}{2}(\mathbf{U}_{i*} - \mathbf{U}_{i*}(t)) \mathbf{H}_i(t) (\mathbf{U}_{i*} - \mathbf{U}_{i*}(t))^T\end{aligned}$$

Update rule for \mathbf{U}_{i*} : (by setting the gradient of $\tilde{L}(\mathbf{U}_{i*})$ with respect to \mathbf{U}_{i*} to 0)

$$\mathbf{U}_{i*}(t+1) = \mathbf{U}_{i*}(t) - \left(\frac{\partial L}{\partial \mathbf{U}_{i*}^T}(t) \right)^T \mathbf{H}_i(t)^{-1}$$

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Optimize \mathbf{U} (Cont'd)

Algorithm 2 Optimizing \mathbf{U} using surrogate algorithm

Input: $\mathbf{X} \in \mathbb{R}^{N \times D}$, $\mathcal{S} = \{s_{ij}\}$, $Q, T \in \mathbb{N}^+$, $\beta, \varepsilon \in \mathbb{R}^+$

Initializing \mathbf{U} by performing PCA on \mathbf{X} .

for $t = 1 \rightarrow T$ **do**

for $i = 1 \rightarrow N$ **do**

 Compute \mathbf{U}_{i*} following the update rule.

end for

 Compute L with the updated \mathbf{U} .

 Terminate the iterative process when the change of L is smaller than ε .

end for

Output: $\mathbf{U} \in \mathbb{R}^{N \times Q}$

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Rounding

The real values of \mathbf{U} are quantized into the binary codes of \mathbf{B} by taking their signs:

$$B_{ij} = \begin{cases} 1, & U_{ij} > 0 \\ -1, & \text{otherwise} \end{cases}$$

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Out-of-sample extension

Goal: Compute the binary code \mathbf{b} for a query \mathbf{x} not in the training set in order to perform ANN search.

Find $\mathbf{W} \in \mathbb{R}^{D \times Q}$ that maps \mathbf{x} to \mathbf{u} , then perform rounding on \mathbf{u} to obtain \mathbf{b} :

$$\mathbf{u} = \mathbf{W}^T \mathbf{x}$$

Solve \mathbf{W} using linear regression:

$$L_e = \|\mathbf{U} - \mathbf{XW}\|_F^2 + \lambda_e \|\mathbf{W}\|_F^2$$

$$\mathbf{W} = (\mathbf{X}^T \mathbf{X} + \lambda_e \mathbf{I})^{-1} \mathbf{X}^T \mathbf{U}$$

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Complexity analysis

Total time cost:

- Optimize \mathbf{U} : $\mathcal{O}(T(NQ^3 + |\mathcal{S}|Q^2))$ (T is the number of iterations)
- Rounding: $\mathcal{O}(NQ)$
- Out-of-sample extension: $\mathcal{O}(ND^2 + D^3 + NDQ)$ (reduced to $\mathcal{O}(ND^2)$ when $Q < D \ll N$)
- Query: $\mathcal{O}(DQ)$

Latent Factor Hashing (LFH) (Zhang et al., 2014)

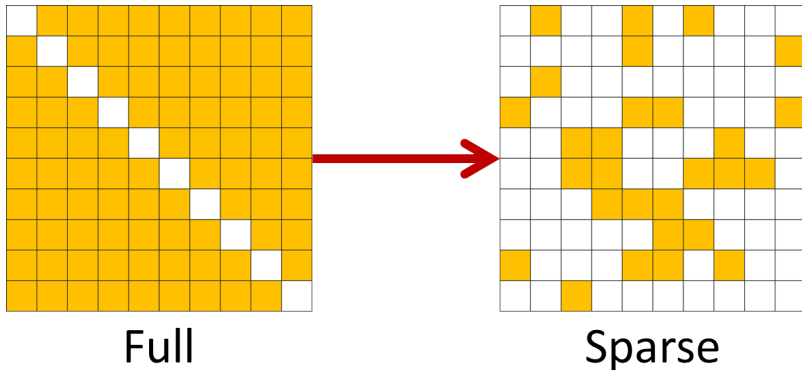
Stochastic learning

- $\mathcal{O}(N^2)$ for full \mathcal{S} : both time cost and the memory cost are unacceptable
- Choose only a subset of \mathcal{S} for updating inspired by the idea used in stochastic gradient descent method
- For storage efficiency, compute only the needed subset of \mathcal{S} during each iteration instead of pre-computing the full \mathcal{S}

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Stochastic learning

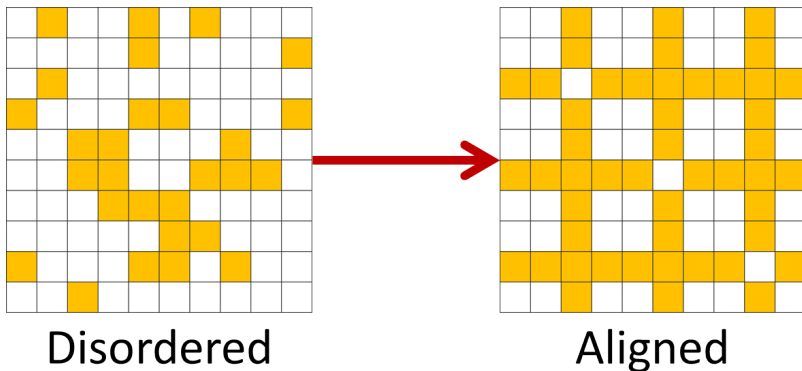
During each iteration, we randomly sample a subset of \mathcal{S} with $\mathcal{O}(NQ)$ entries. Then the time cost to update \mathbf{U} by one iteration is reduced to $\mathcal{O}(NQ^3)$.



Latent Factor Hashing (LFH) (Zhang et al., 2014)

Stochastic learning

Furthermore, if we choose the subset of \mathcal{S} by randomly selecting $\mathcal{O}(Q)$ of its columns and rows, we can further reduce the time cost to $\mathcal{O}(NQ^2)$ per iteration.



Latent Factor Hashing (LFH) (Zhang et al., 2014)

Datasets

- CIFAR-10

- 60,000 color images from 80M tiny image collection.
- 512-D GIST feature vector.
- manually classified into 10 mutually exclusive classes.
- ground-truth neighbors defined based on whether share the same class label.

- NUS-WIDE

- 269,648 images collected from Flickr.
- 1134-D low-level feature vector.
- manually assigned with some of the 81 concept tags.
- ground-truth neighbors defined based on whether share the same tags.

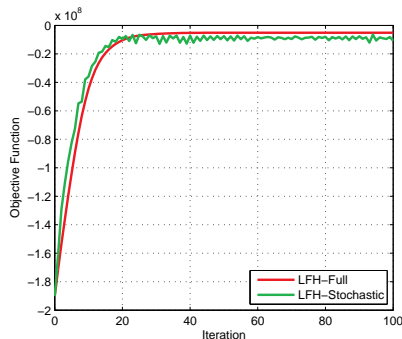
Latent Factor Hashing (LFH) (Zhang et al., 2014)

Baselines

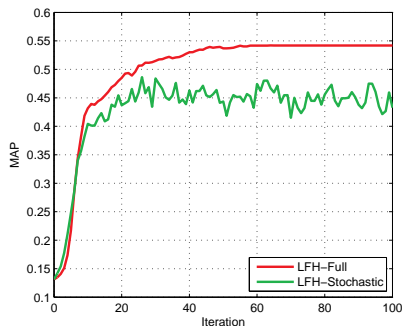
- Locality-sensitive hashing (LSH)
- Shift-invariant kernels hashing (SIKH)
- Spectral hashing (SH)
- Anchor graph hashing (AGH)
- Principal component analysis based hashing (PCAH)
- Iterative quantization (ITQ)
- Minimal loss hashing (MLH)
- Kernel-based supervised hashing (KSH)

Latent Factor Hashing (LFH) (Zhang et al., 2014)

Convergence of learning



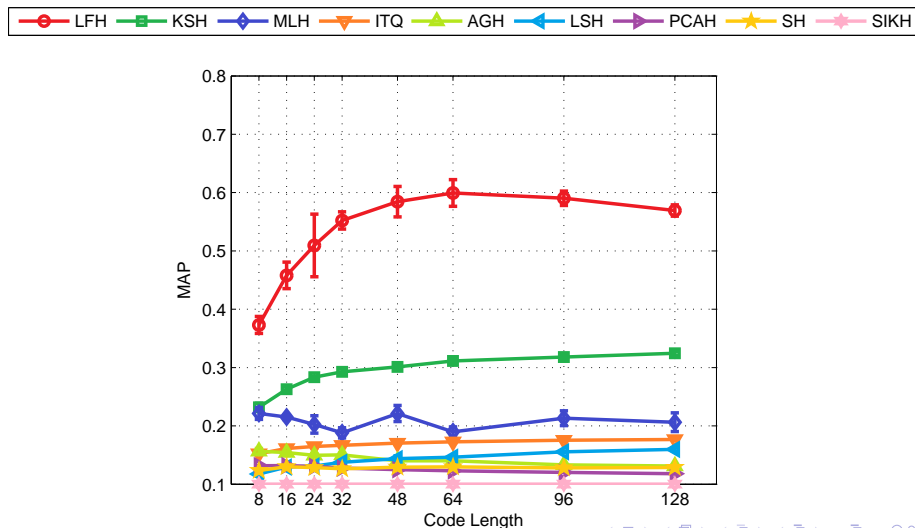
Objective Function



MAP

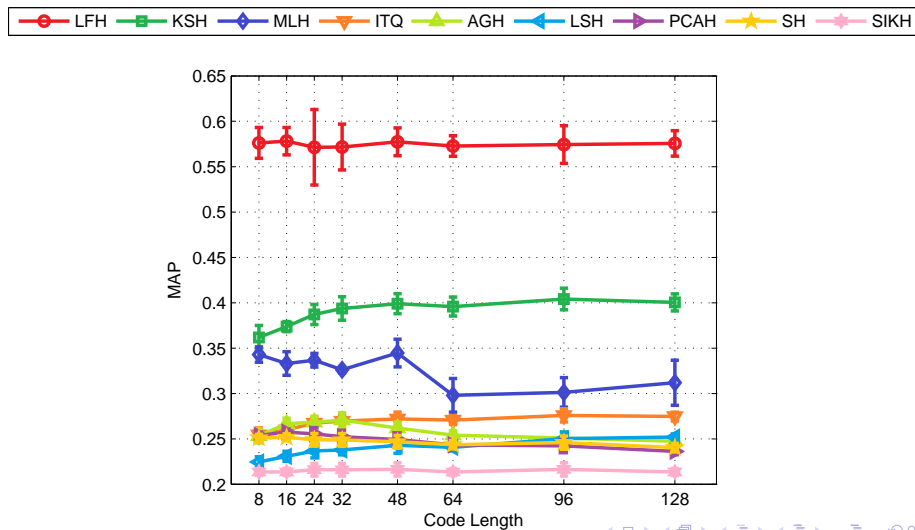
Latent Factor Hashing (LFH) (Zhang et al., 2014)

MAP (CIFAR-10)



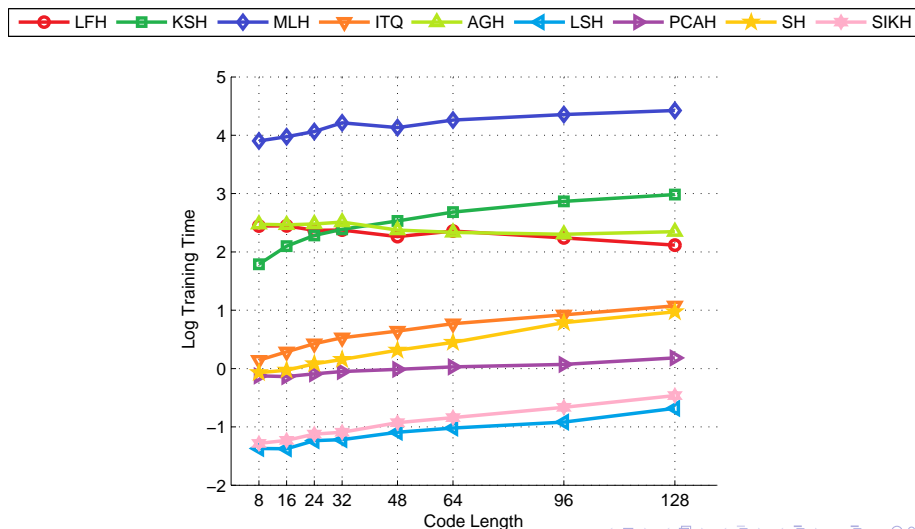
Latent Factor Hashing (LFH) (Zhang et al., 2014)

MAP (NUS-WIDE)



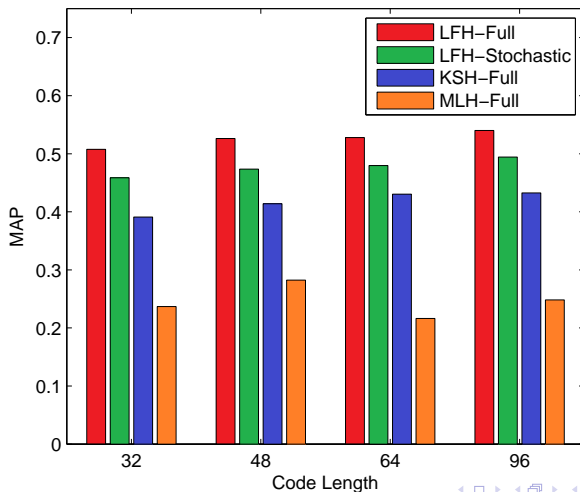
Latent Factor Hashing (LFH) (Zhang et al., 2014)

Training time (CIFAR-10)



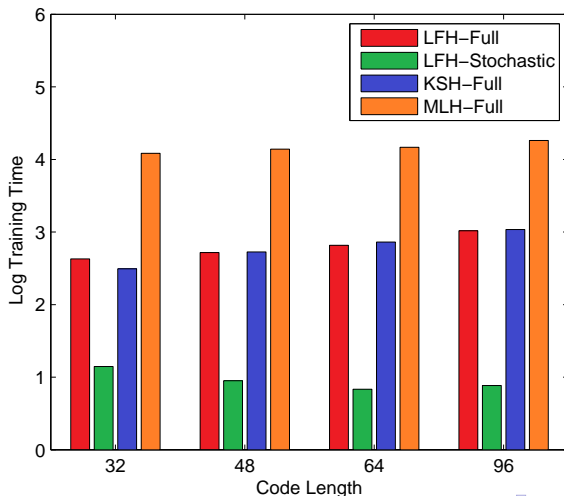
Latent Factor Hashing (LFH) (Zhang et al., 2014)

MAP (Full supervised information)



Latent Factor Hashing (LFH) (Zhang et al., 2014)

Training time (full supervised information)



FastH (Lin et al., 2014)

Overall objective

$$\min_{\mathbf{Z} \in \{-1,1\}^{m \times n}} \sum_{i=1}^n \sum_{j=1}^n |y_{ij}| \left(m y_{ij} - \sum_{k=1}^m z_{k,i} z_{k,j} \right)^2; \quad (3a)$$

$$\min_{\Phi(\cdot)} \sum_{k=1}^m \sum_{i=1}^n \delta(z_{k,j} = h_k(\mathbf{x}_i)). \quad (3b)$$

where $y_{ij} \in \{-1, 0, 1\}$ is the pairwise label between points i and j

Optimization on a block in one bit

$$\min_{\mathbf{z}_{k,\mathcal{B}} \in \{-1,1\}^{|\mathcal{B}|}} \sum_{i \in \mathcal{B}} u_i z_{k,i} + \sum_{i \in \mathcal{B}} \sum_{j \in \mathcal{B}} v_{ij} z_{k,i} z_{k,j}, \quad (8a)$$

$$\text{where, } v_{ij} = -|y_{ij}|(k y_{ij} - \sum_{p=1}^{k-1} z_{p,i}^* z_{p,j}^*), \quad (8b)$$

$$u_i = -2 \sum_{j \notin \mathcal{B}} \hat{z}_{k,j} |y_{ij}| (k y_{ij} - \sum_{p=1}^{k-1} z_{p,i}^* z_{p,j}^*). \quad (8c)$$

FastH (Lin et al., 2014)

Algorithm 3: FastHash

Input: Training data points: $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$; Affinity matrix: \mathbf{Y} ; bit length: m ; blocks: $\{\mathcal{B}_1, \mathcal{B}_2, \dots\}$.

Output: Hash functions: $\Phi = [h_1, \dots, h_m]$

- 1 **for** $k = 1, \dots, m$ **do**
 - 2 Step-1: call Algorithm 2 to obtain binary codes of k -th bit;
 - 3 Step-2: train trees in (9) to obtain hash function h_k ;
 - 4 update the binary codes of k -th bit by the output of h_k ;
-

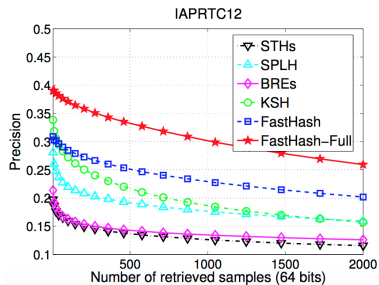
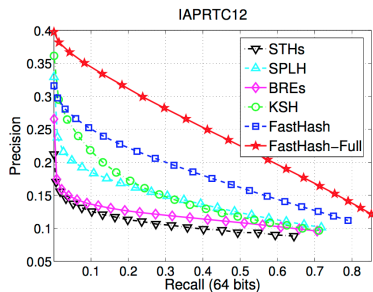
Algorithm 2: Step 1: Block GraphCut for binary code inference

Input: Affinity matrix: \mathbf{Y} ; bit length: k ; max inference iteration; blocks: $\{\mathcal{B}_1, \mathcal{B}_2, \dots\}$; binary codes: $\{\mathbf{z}_1, \dots, \mathbf{z}_{k-1}\}$.

Output: Binary codes of one bit: \mathbf{z}_k

- 1 **repeat**
 - 2 Randomly permute all blocks;
 - 3 **for each** \mathcal{B}_i **do**
 - 4 Solve the inference in (8a) on \mathcal{B}_i using GraphCut;
 - 5 **until** max iteration is reached ;
-

FastH (Lin et al., 2014)



Supervised Discrete Hashing (SDH) (Shen et al., 2015)

Objective

$$\begin{aligned} \min_{\mathbf{B}, \mathbf{W}, F} & \|\mathbf{Y} - \mathbf{W}^\top \mathbf{B}\|^2 + \lambda \|\mathbf{W}\|^2 + \nu \|\mathbf{B} - F(\mathbf{X})\|^2 \quad (7) \\ \text{s.t. } & \mathbf{B} \in \{-1, 1\}^{L \times n}. \end{aligned}$$

where $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^n \in \{0, 1\}^{C \times n}$ is the point-wise label matrix. ($y_{ki} = 1$ if x_i belongs to class k and $y_{ki} = 0$ otherwise)

Alternating optimization:

$$\mathbf{P} = (\phi(\mathbf{X})\phi(\mathbf{X})^\top)^{-1}\phi(\mathbf{X})\mathbf{B}^\top. \quad (5)$$

$$\mathbf{W} = (\mathbf{B}\mathbf{B}^\top + \lambda\mathbf{I})^{-1}\mathbf{B}\mathbf{Y}^\top. \quad (8)$$

$$\mathbf{z} = \text{sgn}(\mathbf{q} - \mathbf{B}^\top \mathbf{W}' \mathbf{v}). \quad (15)$$

$$\mathbf{b}_i = \text{sgn}(F(\mathbf{x}_i) + \frac{\delta}{2} \sum_{k=1}^C \mathbf{w}^{(ki)}). \quad (22)$$

Supervised Discrete Hashing (SDH) (Shen et al., 2015)

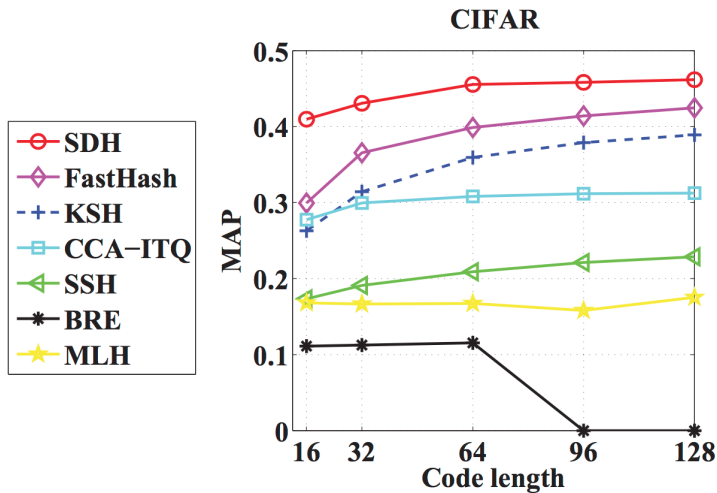
Algorithm 1 Supervised Discrete Hashing (SDH)

Input: Training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$; code length L ; number of anchor points m ; maximum iteration number t ; parameters λ and ν .

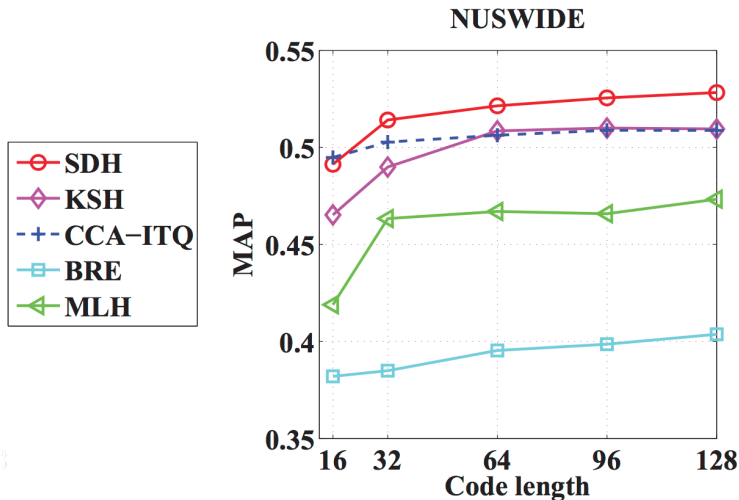
Output: Binary codes $\{\mathbf{b}_i\}_{i=1}^n \in \{-1, 1\}^{L \times n}$;
hash function $H(\mathbf{x}) = \text{sgn}(F(\mathbf{x}))$.

1. Randomly select m samples $\{\mathbf{a}_j\}_{j=1}^m$ from the training data and get the mapped training data $\phi(\mathbf{x})$ via the RBF kernel function.
 2. Initialize \mathbf{b}_i as a $\{-1, 1\}^L$ vector randomly, $\forall i$.
 3. Loop until converge or reach maximum iterations:
 - **G-Step:** Calculate \mathbf{W} using equation (8) or multi-class SVM.
 - **F-Step:** Compute \mathbf{P} using (5) to form $F(\mathbf{x})$.
 - **B-Step:** For the ℓ_2 loss, iteratively learn $\{\mathbf{b}_i\}_{i=1}^n$ bit by bit using the DCC method with equation (15); for the hinge loss, compute \mathbf{b}_i by (22).
-

Supervised Discrete Hashing (SDH) (Shen et al., 2015)



Supervised Discrete Hashing (SDH) (Shen et al., 2015)



Column Sampling based Discrete Supervised Hashing (COSDISH) (Kang et al., 2016)

Motivation

- Learning to hash is essentially a **discrete optimization problem**
- Most existing methods solve relaxed continuous problems
- FastH (Lin et al., 2014) illustrates better accuracy with discrete optimization, but it **cannot utilize all training points** due to high time complexity.

We propose a discrete supervised hashing method which can leverage all training points:

- **C**olumn **S**ampling based **D**iscrete **S**upervised Hashing (**COSDISH**).

Column Sampling based Discrete Supervised Hashing (COSDISH) (Kang et al., 2016)

KSH (Liu et al., 2012), TSH (Lin et al., 2013), FastH (Lin et al., 2014) all optimize the following objective function:

$$\min_{\mathbf{B} \in \{-1, 1\}^{n \times q}} \|\mathbf{qS} - \mathbf{B}\mathbf{B}^T\|_F^2 = \sum_{i=1}^n \sum_{j=1}^n (qS_{ij} - \mathbf{B}_{i*} \mathbf{B}_{j*}^T)^2$$

where $S_{ij} \in \{-1, 1\}$ denotes the supervised label.

The inner product reflects the opposite of the Hamming distance:

$$\text{dist}_{\mathcal{H}}(\mathbf{B}_{i*}, \mathbf{B}_{j*}) = (q - \mathbf{B}_{i*} \mathbf{B}_{j*}^T) / 2$$

Column Sampling based Discrete Supervised Hashing (COSDISH) (Kang et al., 2016)

Column Sampling

In each iteration, we randomly choose a subset Ω of $\mathcal{N} = \{1, 2, \dots, n\}$, and sample $|\Omega|$ columns of \mathbf{S} with the column numbers indexed by Ω .

We use $\tilde{\mathbf{S}} \in \{-1, 1\}^{n \times |\Omega|}$ to denote the **sampled sub-similarity matrix**.

Let $\Gamma = \mathcal{N} - \Omega$, we can split $\tilde{\mathbf{S}}$ and \mathbf{B} into two parts:

Ω part: $\tilde{\mathbf{S}}^\Omega \in \{-1, 1\}^{|\Omega| \times |\Omega|}$, and $\mathbf{B}^\Omega \in \{-1, 1\}^{|\Omega| \times q}$

Γ part: $\tilde{\mathbf{S}}^\Gamma \in \{-1, 1\}^{|\Gamma| \times |\Omega|}$, and $\mathbf{B}^\Gamma \in \{-1, 1\}^{|\Gamma| \times q}$

Based on sampled columns in each iteration, the objective can be reformulated as follows:

$$\min_{\mathbf{B}^\Omega, \mathbf{B}^\Gamma} \|q\tilde{\mathbf{S}}^\Gamma - \mathbf{B}^\Gamma[\mathbf{B}^\Omega]^T\|_F^2 + \|q\tilde{\mathbf{S}}^\Omega - \mathbf{B}^\Omega[\mathbf{B}^\Omega]^T\|_F^2.$$

Our learning strategy is to **alternatively optimize \mathbf{B}^Ω and \mathbf{B}^Γ**

Column Sampling based Discrete Supervised Hashing (COSDISH) (Kang et al., 2016)

Update \mathbf{B}^Γ with \mathbf{B}^Ω Fixed

$\mathbf{B}^\Gamma = \text{sgn}(\tilde{\mathbf{S}}^\Gamma \mathbf{B}^\Omega)$ reaches the minimum of $f_1(\cdot)$ which changes the F -norm to L_1 norm.

Theorem

Suppose that $f_1(\mathbf{F}_1^)$ and $f_2(\mathbf{F}_2^*)$ reach their minimum at the points \mathbf{F}_1^* and \mathbf{F}_2^* , respectively. We have $f_2(\mathbf{F}_1^*) \leq 2qf_2(\mathbf{F}_2^*)$.*

Column Sampling based Discrete Supervised Hashing (COSDISH) (Kang et al., 2016)

Update \mathbf{B}^Ω with \mathbf{B}^Γ Fixed

When \mathbf{B}^Γ is fixed, the sub-problem of \mathbf{B}^Ω is given by:

$$\min_{\mathbf{B}^\Omega \in \{-1,1\}^{|\Omega| \times q}} \|q\tilde{\mathbf{S}}^\Gamma - \mathbf{B}^\Gamma [\mathbf{B}^\Omega]^T\|_F^2 + \|q\tilde{\mathbf{S}}^\Omega - \mathbf{B}^\Omega [\mathbf{B}^\Omega]^T\|_F^2.$$

We can transform the above problem to q binary quadratic programming (BQP) problems. The optimization of the k th bit of \mathbf{B}^Ω is given by:

$$\min_{\mathbf{b}^k \in \{-1,1\}^{|\Omega|}} [\mathbf{b}^k]^T \mathbf{Q}^{(k)} \mathbf{b}^k + [\mathbf{b}^k]^T \mathbf{p}^{(k)}$$

where \mathbf{b}^k denotes the k th column of \mathbf{B}^Ω , and

$$Q_{i,j}^{(k)} = -2(q\tilde{S}_{i,j}^\Omega - \sum_{m=1}^{k-1} b_i^m b_j^m), Q_{i,i}^{(k)} = 0, \\ p_i^{(k)} = -2 \sum_{l=1}^{|\Gamma|} B_{l,k}^\Gamma (q\tilde{S}_{l,i}^\Gamma - \sum_{m=1}^{k-1} B_{l,m}^\Gamma B_{i,m}^\Omega)$$

Column Sampling based Discrete Supervised Hashing (COSDISH) (Kang et al., 2016)

Out-of-Sample Extension

We have only learned binary codes for training points. How to learn a hash function for query points?

- Train q binary classifiers as hash functions.

Classifiers:

Linear Classifier SPLH, LFH, COSDISH

Kernel Vector KSH, SDH

SVM TSH

CNN CNNH, NINH

Boosted Tree FastH, COSDISH_BT

Column Sampling based Discrete Supervised Hashing (COSDISH) (Kang et al., 2016)

Accuracy (MAP)

Method	CIFAR-10 (60K)			
	8-bits	16-bits	32-bits	64-bits
COSDISH	0.4986	0.5768	0.6191	0.6371
SDH	0.2642	0.3994	0.4145	0.4346
LFH	0.2908	0.4098	0.5446	0.6182
TSH	0.2365	0.3080	0.3455	0.3663
KSH	0.2334	0.2662	0.2923	0.3128
SPLH	0.1588	0.1635	0.1701	0.1730
COSDISH_BT	0.5856	0.6681	0.7079	0.7346
FastH	0.4230	0.5216	0.5970	0.6446

Column Sampling based Discrete Supervised Hashing (COSDISH) (Kang et al., 2016)

Training Time

Table 2: Training time (in second) on subsets of NUS-WIDE

Method	3K	10K	50K	100K	200K
COSDISH	5.6	8.0	33.7	67.7	162.2
SDH	3.9	11.8	66.2	126.9	248.2
LFH	14.3	16.3	27.8	40.8	85.9
TSH	922.2	27360	>50000	-	-
KSH	1104	4446	>50000	-	-
SPLH	25.3	185	-	-	-
COSDISH_BT	60.2	69.1	228.3	422.6	893.3
FastH	172.3	291.6	1451	3602	-

Outline

- 1 Introduction
- 2 Unsupervised Hashing
- 3 Supervised Hashing
- 4 Ranking-based Hashing**
- 5 Multimodal Hashing
- 6 Deep Hashing
- 7 Quantization
- 8 Conclusion
- 9 Reference

Ranking-based Methods

The supervised information is ranking labels, such as triplets $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$.

- **HDML**: Hamming distance metric learning (Norouzi et al., 2012)
- **OPH**: Order preserving hashing for approximate nearest neighbor search (Wang et al., 2013b)
- **RSH**: Learning hash codes with listwise supervision (Wang et al., 2013a)
- **RPH**: Ranking preserving hashing for fast similarity search (Wang et al., 2015)

Hamming Distance Metric Learning (Norouzi et al., 2012)

The objective is given by:

$$L(w) = \sum_{(x, x^+, x^-) \in D} \ell_{\text{triplet}}(b(x; w), b(x^+, w); b(x^-, w)) + \frac{\lambda}{2} \|w\|_2^2$$

discontinuous and non-convex !

Construct a continuous upper bound on the loss:

$$\begin{aligned} \ell_{\text{triplet}}(b(x; w), b(x^+, w); b(x^-, w)) \leq \\ \max_{g, g^+, g^-} \{ \ell_{\text{triplet}}(g, g^+, g^-) + g^T f(x; w) + g^{+T} f(x^+; w) + g^{-T} f(x^-; w) \} \\ - \max_h \{ h^T f(x; w) \} - \max_{h^+} \{ h^{+T} f(x^+; w) \} - \max_{h^-} \{ h^{-T} f(x^-; w) \} \end{aligned}$$

where,

$$\begin{aligned} b(x; w) &= \text{sign}(f(x; w)) \\ &= \arg \max_{h \in H} h^T f(x; w) \end{aligned}$$

Hamming Distance Metric Learning (Norouzi et al., 2012)

To use the upper bound, we need to find the binary codes given by:

$$(\hat{g}, \hat{g}^+, \hat{g}^-) = \arg \max_{(g, g^+, g^-)} \{ \ell_{\text{triplet}}(g, g^+, g^-) + g^T f(x) + g^{+T} f(x^+) + g^{-T} f(x^-) \}$$

Assume $d(g, g^+, g^-) = m$, m is an integer between $-q$ and q , q is code length. The problem is converted to

$$(\hat{g}, \hat{g}^+, \hat{g}^-) = \ell'(m) + \max_{g, g^+, g^-} \{ g^T f(x) + g^{+T} f(x^+) + g^{-T} f(x^-) \}$$

where $\ell'(\alpha) = [\alpha - 1]_+$

It is straightforward that the solution is the largest of those $2q + 1$ maxima.

Hamming Distance Metric Learning (Norouzi et al., 2012)

Perceptron-like learning

Randomly initialize $w^{(0)}$, then use the following procedure to update $w^{(t+1)}$:

1. Select a random triplet (x, x^+, x^-) from dataset D.
2. Compute $(\hat{h}, \hat{h}^+, \hat{h}^-) = (b(x; w^{(t)}), b(x^+; w^{(t)}), b(x^-; w^{(t)}))$
3. Compute $(\hat{g}, \hat{g}^+, \hat{g}^-)$
4. Update model parameters using

$$w^{(t+1)} = w^{(t)} + \eta \left[\frac{\partial f(x)}{\partial w} (\hat{h} - \hat{g}) + \frac{\partial f(x^+)}{\partial w} (\hat{h}^+ - \hat{g}^+) + \frac{\partial f(x^-)}{\partial w} (\hat{h}^- - \hat{g}^-) - \lambda w^{(t)} \right]$$

Hamming Distance Metric Learning (Norouzi et al., 2012)

Hashing, Loss	Distance	k NN	64 bits	128 bits	256 bits	512 bits
Linear, pairwise hinge [24]	H	7 NN	72.2	72.8	73.8	74.6
Linear, pairwise hinge	AH	8 NN	72.3	73.5	74.3	74.9
Linear, triplet ranking	H	2 NN	75.1	75.9	77.1	77.9
Linear, triplet ranking	AH	2 NN	75.7	76.8	77.5	78.0
Baseline			Accuracy			
One-vs-all linear SVM [6]			77.9			
Euclidean 3NN			59.3			

Table 2: Recognition accuracy on the CIFAR-10 test set (H \equiv Hamming, AH \equiv Asym. Hamming).

Order Preserving Hashing (Wang et al., 2013b)

Assuming the binary code length is m , OPH tries to align the following two kinds of categories:

- $(C_0^h, C_1^h, \dots, C_m^h)$: ordered categories according to the $m + 1$ Hamming distance values.
- $(C_0^e, C_1^e, \dots, C_m^e)$: target categories computed from the original space.

Two principles:

- **Category order alignment**: categories computed from the Hamming space and those from the original space are matched, i.e., $C_k^h = C_k^e$.
- **Bucket balance**: the points are uniformly distributed in all hash buckets.

Order Preserving Hashing (Wang et al., 2013b)

The objective function:

$$\begin{aligned} \min_{\mathbf{W}} \quad & \frac{1}{N} \sum_{n=1}^N \sum_{i=0}^{m-1} L_{ni}(\mathbf{W}) \\ \text{s.t.} \quad & \mathbf{W}_k^T \mathbf{W}_k = 1, \forall k = 1, \dots, m \end{aligned}$$

where,

$$L_{ni}(\mathbf{W}) = \sum_{\mathbf{x}' \in \mathcal{N}_{ni}^e} \phi_{\gamma_{i1}}(d_{RH}(\mathbf{x}_n, \mathbf{x}') - i) + \lambda \sum_{\mathbf{x}' \notin \mathcal{N}_{ni}^e} \phi_{\gamma_{i2}}(i + 1 - d_{RH}(\mathbf{x}_n, \mathbf{x}'))$$

$$d_{RH}(\mathbf{x}, \mathbf{x}') = \|\phi_{\beta}(\mathbf{W}^T \mathbf{x}) - \phi_{\beta}(\mathbf{W}^T \mathbf{x}')\|_2^2$$

$$\phi_{\alpha}(z) = \frac{1}{1 + e^{-\alpha z}}$$

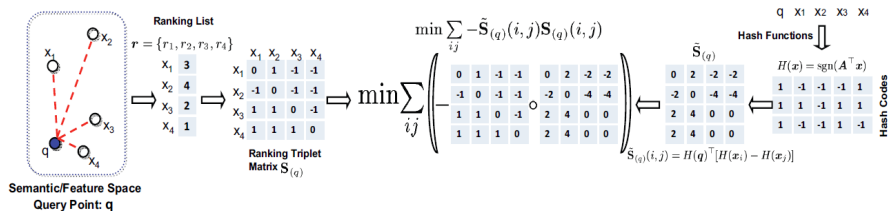
Solution: Quadratic penalty, mini-batch gradient descent and active set are utilized.

Ranking Preserving Hashing (Wang et al., 2013b)

Table 1: Comparison in terms of mean average precision (%). The best scores are in bold font. Our approach OPH obtains a significant gain on larger data sets (SIFT100K, SIFT1M, GIST1M) and comparable performance or superior on smaller data sets (Labelme, Peekaboom).

Data Set	Code Length	Approaches									
		OPH	MLH	BRE	ITQ	IsoHash	KSH	AGH	SH	USPLH	LSH
Labelme	32	21.11	19.91	16.45	20.36	18.51	16.72	10.90	9.28	9.62	8.87
	64	33.94	32.48	27.59	32.09	28.35	24.57	12.04	11.18	17.66	17.57
	128	44.36	45.22	40.59	44.66	42.34	31.45	12.35	13.73	9.67	32.52
	256	56.85	55.89	53.31	58.40	57.35	36.32	13.10	14.54	9.61	49.64
Peekaboom	32	13.00	12.59	9.42	12.10	10.63	9.40	6.01	5.25	6.22	4.47
	64	23.00	21.65	18.70	21.20	19.67	14.64	6.74	6.25	8.73	10.62
	128	34.14	33.08	30.42	32.75	32.31	21.33	7.05	9.28	5.94	23.74
	256	45.42	44.42	42.08	46.60	47.41	24.86	7.45	10.21	5.99	39.54
GIST1M	32	2.00	1.74	1.14	1.68	1.39	1.15	0.94	0.68	0.26	0.56
	64	4.12	3.51	2.67	3.27	3.25	2.21	1.05	1.08	0.32	1.50
	128	6.97	5.96	5.24	5.14	5.21	3.83	1.09	1.45	0.31	3.12
	256	10.43	7.20	8.18	6.95	7.70	5.26	1.08	2.19	0.33	6.04
SIFT100K	32	12.40	9.97	5.14	8.14	7.06	4.75	4.97	10.09	8.01	5.02
	64	25.17	18.63	11.31	18.37	15.47	8.87	5.98	17.36	12.67	12.52
	128	37.41	32.13	20.76	32.44	28.27	12.69	6.46	22.51	16.19	24.50
	256	51.57	46.42	32.78	-	-	15.97	6.67	32.42	18.56	40.69
SIFT1M	32	5.07	3.07	1.51	2.69	2.31	1.26	1.11	4.23	3.92	1.49
	64	13.58	8.11	4.46	8.16	7.24	2.94	1.50	9.81	8.19	5.68
	128	26.00	18.01	10.58	17.87	16.53	5.04	1.68	15.56	10.52	14.05
	256	39.88	31.69	20.16	-	-	7.38	1.77	26.10	12.05	28.59

Learning Hash Codes with Listwise Supervision (Wang et al., 2013a)



Ranking-based supervised hashing (RSH) defines a rank triplet S_{mij} :

$$S_{mij} = S(\mathbf{q}_m; \mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & : r_i^q < r_j^q \\ -1 & : r_i^q > r_j^q \\ 0 & : r_i^q = r_j^q \end{cases}$$

The loss function can be represented as :

$$L_H = - \sum_m \sum_{i,j} H(\mathbf{q}_m)^T [H(\mathbf{x}_i) - H(\mathbf{x}_j)] S_{mij}$$

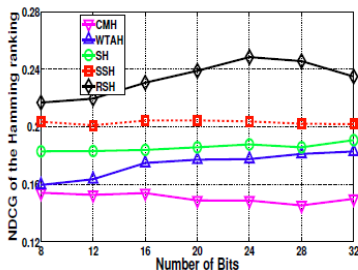
Learning Hash Codes with Listwise Supervision (Wang et al., 2013a)

Utilizing a linear hash function, the relaxed objective function is given by:

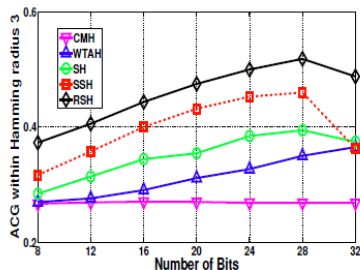
$$\begin{aligned} \min_W & - \sum_m \sum_{i,j} \mathbf{q}_m^T W W^T (\mathbf{x}_i - \mathbf{x}_j) S_{mij} \\ \text{s.t. } & W^T W = I \end{aligned}$$

Solution: augmented Lagrangian multiplier is utilized.

Learning Hash Codes with Listwise Supervision (Wang et al., 2013a)



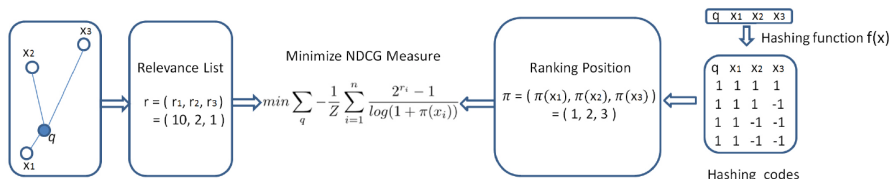
(a)



(b)

Figure 3. Performance evaluation on **NUSWIDE** dataset using different number of hash bits. a) NDCG of the Hamming ranking; b) ACG within Hamming radius 3.

Ranking Preserving Hashing (Wang et al., 2015)



The **NDCG** objective is a **non-convex non-smooth** optimization problem, since it depends on the ranking positions of data.

Ranking Preserving Hashing (Wang et al., 2015)

Utilizing a tractable probabilistic framework with a logistic function to approximate it:

$$\begin{aligned}\hat{\pi}_j(x_i) &= 1 + E\left[\sum_{k=1}^n I(c_{q_j}^T(c_k - c_i) > 0)\right] \\ &= 1 + \sum_{k=1}^n \Pr(c_{q_j}^T(c_k - c_i) > 0) \\ &= 1 + \sum_{k=1}^n \frac{1}{1 + \exp(-q_j^T W^T W(x_k - x_i))}\end{aligned}$$

The overall objective is given by:

$$J(W) = -\sum_{j=1}^m \frac{1}{Z_j} \sum_{i=1}^n \frac{2^{r_i^j} - 1}{\log(1 + \hat{\pi}_j(x_i))} + \alpha \|WW^T - I\|_F^2$$

Solution: This guarantees the objective is smooth and differentiable. Then L-BFGS quasi-Newton method is applied to solve the optimization problem.

Ranking Preserving Hashing (Wang et al., 2015)

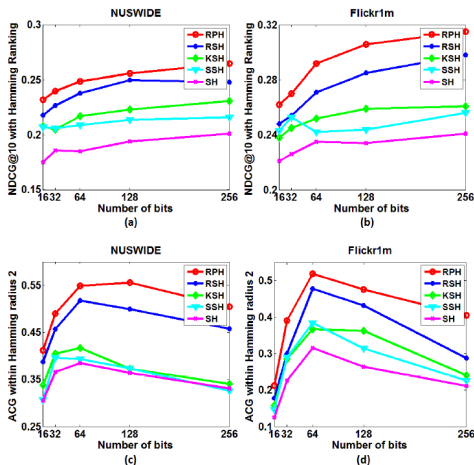


Figure 2: Performance evaluation on both datasets with different number of hashing bits. (a)-(b): NDCG@10 using *Hamming Ranking*. (c)-(d): ACG with Hamming radius 2 using *Hash Lookup*.

Outline

- 1 Introduction
- 2 Unsupervised Hashing
- 3 Supervised Hashing
- 4 Ranking-based Hashing
- 5 Multimodal Hashing**
- 6 Deep Hashing
- 7 Quantization
- 8 Conclusion
- 9 Reference

Multimodal Hashing

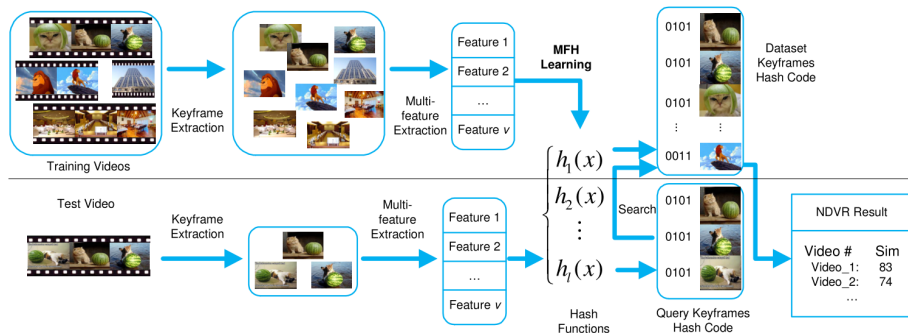
Multi-Source Hashing:

- **MFH**: Multiple feature hashing (Song et al., 2011)
- **CH**: Composite hashing (Zhang et al., 2011)

Cross-Modal Hashing:

- **CVH**: Cross view hashing (Kumar and Udupa, 2011)
- **MLBE**: Multimodal latent binary embedding (Zhen and Yeung, 2012a)
- **CRH**: Co-regularized hashing (Zhen and Yeung, 2012b)
- **IMH**: Inter-media hashing (Song et al., 2013)
- **RaHH**: Relation-aware heterogeneous hashing (Ou et al., 2013)
- **SCM**: Semantic correlation maximization (Zhang and Li, 2014)
- **CMFH**: Collective matrix factorization hashing (Ding et al., 2014)
- **QCH**: Quantized correlation hashing (Wu et al., 2015)
- **SePH**: Semantics-preserving hashing (Lin et al., 2015b)

Multiple Feature Hashing (Song et al., 2011)



Multiple Feature Hashing (Song et al., 2011)

MFH define v affinity matrices:

$$(A^g)_{pq} = \begin{cases} 1, & \text{if } (x^g)_p \in \mathcal{N}_k((x^g)_q) \text{ or } (x^g)_q \in \mathcal{N}_k((x^g)_p) \\ 0, & \text{else} \end{cases}$$

The relaxed objective function is given by

$$\begin{aligned} \min_{Y, Y^g, W, b} & \sum_{g=1}^v \sum_{p,q=1}^n (A^g)_{pq} \|(y^g)_p - (y^g)_q\|_F^2 + \\ & \gamma \sum_{g=1}^v \sum_{t=1}^n \|y_t - (y^g)_t\|_F^2 + \\ & \alpha (\|X^T W + \mathbf{1}b - Y\|_F^2 + \beta \|W\|_F^2) \\ \text{s.t. } & YY^T = I \end{aligned}$$

Multiple Feature Hashing (Song et al., 2011)

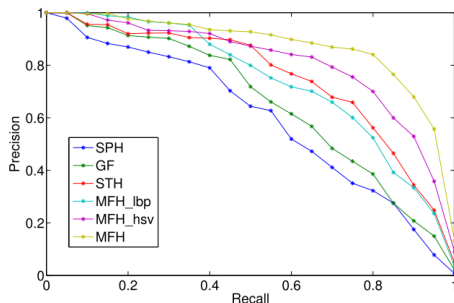


Table 2: Comparison of MAP and time on Combined Dataset

Methods	MAP	Time(s)
SPH	0.5941	0.4907
GF	0.6466	1.3917
STH	0.7536	0.6439
MFH_lbp	0.7526	0.6445
MFH_hsv	0.8042	0.4508
MFH	0.8656	0.5533

Composite Hashing (Zhang et al., 2011)

Composite Hashing follows SH's idea and it learns hashing codes from M information sources.

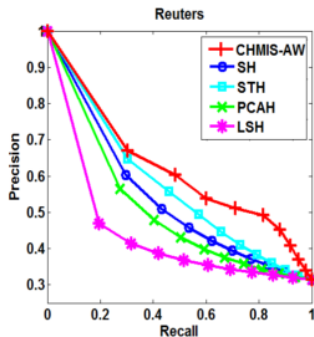
For t -th source, the affinity matrix is defined by

$$S_{i,j}^{(t)} = \begin{cases} e^{-\frac{\|x_i^{(t)} - x_j^{(t)}\|^2}{\delta_{i,j}^2}} & \text{if } x_i^{(t)} \in N_k(x_j^{(t)}) \text{ or } x_j^{(t)} \in N_k(x_i^{(t)}) \\ 0 & \text{otherwise} \end{cases}$$

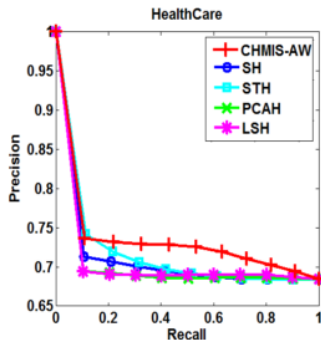
The overall objective function of Composite Hashing combines the similarity preservation part and the consistency part

$$\begin{aligned} \min_{Y, \tilde{W}, \alpha} & \operatorname{tr}(C_1 Y^T \sum_{t=1}^M \tilde{L}^{(t)} Y) + C_2 \|Y - \sum_{t=1}^M \alpha_t (\tilde{W})^T \tilde{X}^{(t)}\|^2 + \|\tilde{W}\|^2 \\ \text{s.t. } & Y \in \{-1, 1\}^{K \times n} \\ & Y\mathbf{1} = 0, \quad YY^T = I, \quad \alpha^T \mathbf{1} = 1, \quad \alpha \geq 0 \end{aligned}$$

Composite Hashing (Zhang et al., 2011)



(a)



(b)

Figure 3: Precision and Recall Curve on Reuters and the healthcare dataset, with fixed bit number 32. CHMIS-AW demonstrates its superior performance over the other three algorithms.

Cross View Hashing (CVH) (Kumar and Udupa, 2011)

CVH extends the SH model to cross view settings.

The Hamming distance summed over all the views

$$d_{ij} = \sum_{k=1}^K d(y_i^{(k)}, y_j^{(k)}) + \sum_{k=1}^K \sum_{k' > k}^K d(y_i^{(k)}, y_j^{(k')})$$

The objective function of CVH is a generalization of SH

$$\min \bar{d} = \sum_{i=1}^n \sum_{i=1}^n W_{ij} d_{ij} = \sum_{k=1}^K \text{Tr}(Y^{(k)} L' Y^{(k)T}) - 2 \sum_{k=1}^K \sum_{k' > k}^K \text{Tr}(Y^{(k)} W Y^{(k')})$$

$$s.t. Y^{(k)} e = 0, \text{ for } k = 1, \dots, K$$

$$\frac{1}{n} Y^{(k)} Y^{(k)T} = I_d, \text{ for } k = 1, \dots, K$$

$$Y_{ij}^{(k)} \in \{-1, 1\}, \text{ for } k = 1, \dots, K$$

Cross View Hashing (CVH) (Kumar and Udupa, 2011)

Interesting special cases:

- $K = 1$

$$X^{(1)} L X^{(1)T} A^{(1)} = X^{(1)} X^{(1)T} A^{(1)} \Lambda$$

This formulation is similar in structure to Locality Preserving Indexing (Cai et al., 2007).

- $W = I_{n \times n}$

$$\begin{aligned} X^{(1)} X^{(2)T} A^{(2)} &= X^{(1)} X^{(1)T} A^{(1)} \Lambda' \\ X^{(2)} X^{(1)T} A^{(1)} &= X^{(2)} X^{(2)T} A^{(2)} \Lambda' \end{aligned}$$

This is the generalized eigenvalue formulation of Canonical Correlation Analysis.

Cross View Hashing (CVH) (Kumar and Udupa, 2011)

Table 1: Comparative Performance in Japanese Language People Search

	P@1	P@3
HASH	0.765	0.855
DM	0.408	0.497

Table 2: Summary of $K = 2$ Experiments

	Training Set Size	Test Set Size	MRR HASH	MRR GEOM
ENG-HIN	15541	1027	0.725	0.686
ENG-RUS	11527	1124	0.629	0.563
ENG-HEB	16317	998	0.794	0.723

Table 3: Summary of $K > 2$ Experiments

Training Data	ENG-HIN	ENG-KAN	ENG-TAM
ENG-HIN	0.725	-	-
ENG-KAN	-	0.712	-
ENG-TAM	-	-	0.71
ENG-HIN-KAN	0.716	0.722	-
ENG-HIN-KAN-TAM	0.69	0.678	0.708

Multimodal Latent Binary Embedding (MLBE) (Zhen and Yeung, 2012a)

MLBE proposed a probabilistic model for multimodal hash function learning.

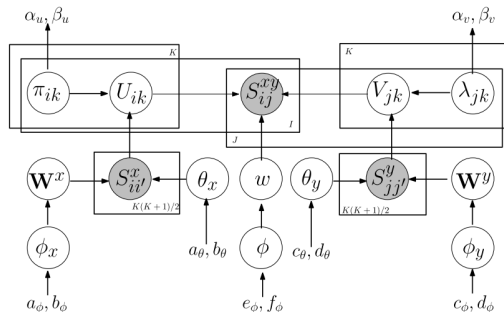


Figure 1: Graphical model representation of MLBE

Multimodal Latent Binary Embedding (MLBE) (Zhen and Yeung, 2012a)

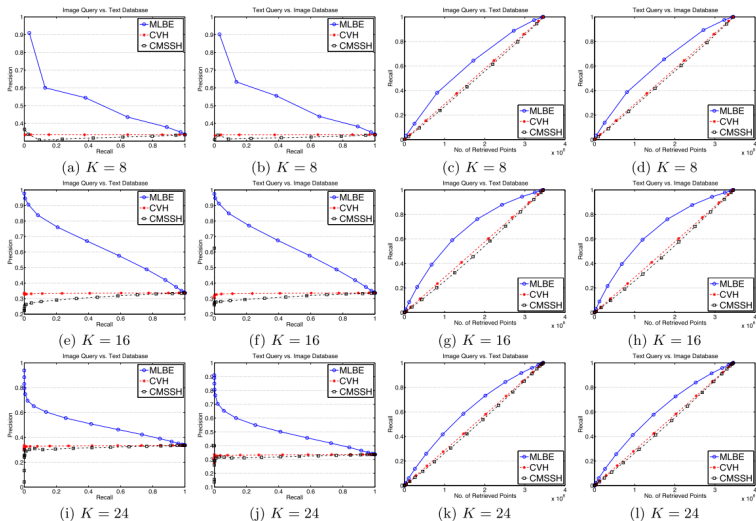
$$\begin{aligned}
 p(\mathbf{S}^x \mid \mathbf{U}, \mathbf{W}^x, \theta_x) &= \prod_{i=1}^I \prod_{i'=1}^I \mathcal{N}(S_{ii'}^x \mid \mathbf{u}_i^T \mathbf{W}^x \mathbf{u}_{i'}, \frac{1}{\theta_x}), \\
 p(\mathbf{S}^y \mid \mathbf{V}, \mathbf{W}^y, \theta_y) &= \prod_{j=1}^J \prod_{j'=1}^J \mathcal{N}(S_{jj'}^y \mid \mathbf{v}_j^T \mathbf{W}^y \mathbf{v}_{j'}, \frac{1}{\theta_y}), \\
 p(\mathbf{S}^{xy} \mid \mathbf{U}, \mathbf{V}, w) &= \prod_{i=1}^I \prod_{j=1}^J [\text{Bern}(S_{ij}^{xy} \mid \gamma(w \mathbf{u}_i^T \mathbf{v}_j))]^{O_{ij}} \\
 p(U_{ik} \mid \pi_{ik}) &= \text{Bern}(U_{ik} \mid \pi_{ik}), \\
 p(\pi_{ik} \mid \alpha_u, \beta_u) &= \text{Beta}(\pi_{ik} \mid \alpha_u, \beta_u), \\
 p(\mathbf{U} \mid \alpha_u, \beta_u) &= \prod_{i=1}^I \prod_{k=1}^K \text{Bern}(U_{ik} \mid \frac{\alpha_u}{\alpha_u + \beta_u}). \\
 p(\mathbf{V} \mid \alpha_v, \beta_v) &= \prod_{j=1}^J \prod_{k=1}^K \text{Bern}(V_{jk} \mid \frac{\alpha_v}{\alpha_v + \beta_v}). \\
 p(\mathbf{W}^x \mid \phi_x) &= \prod_{k=1}^K \prod_{d=k}^K \mathcal{N}(W_{kd}^x \mid 0, \frac{1}{\phi_x}), \\
 p(\mathbf{W}^y \mid \phi_y) &= \prod_{k=1}^K \prod_{d=k}^K \mathcal{N}(W_{kd}^y \mid 0, \frac{1}{\phi_y}).
 \end{aligned}$$

Multimodal Latent Binary Embedding (MLBE) (Zhen and Yeung, 2012a)

Table 2: mAP comparison on Flickr

Task	Method	Code Length		
		$K = 8$	$K = 16$	$K = 24$
Image Query vs. Text Database	MLBE	0.6322	0.6608	0.5104
	CVH	0.5361	0.4871	0.4605
	CMSSH	0.5155	0.5333	0.5150
Text Query vs. Image Database	MLBE	0.5626	0.5970	0.4296
	CVH	0.5260	0.4856	0.4553
	CMSSH	0.5093	0.4594	0.4053

Multimodal Latent Binary Embedding (MLBE) (Zhen and Yeung, 2012a)



Co-Regularized Hashing (CRH) (Zhen and Yeung, 2012b)

CRH is a novel multimodal hash function learning method based on a boosted co-regularization framework. The objective function is

$$\mathcal{O} = \frac{1}{I} \sum_{i=1}^I l_i^x + \frac{1}{J} \sum_{j=1}^J l_j^y + \gamma \sum_{n=1}^N \omega_n l_n^* + \frac{\lambda_x}{2} \|w_x\|^2 + \frac{\lambda_y}{2} \|w_y\|^2$$

The intra-modality loss l_i^x and l_j^y are

$$\begin{aligned} l_i^x &= [1 - f(x_i)(w_x^T x_i)]_+ = [1 - |w_x^T x_i|]_+ \\ l_j^y &= [1 - g(y_j)(w_y^T y_j)]_+ = [1 - |w_y^T y_j|]_+ \end{aligned}$$

The inter-modality loss l_n^* is

$$l_n^* = s_n d_n^2 + (1 - s_n) \tau(d_n)$$

where $d_n = w_x^T x_{a_n} - w_y^T y_{b_n}$, τ is the smoothly clipped inverted squared deviation (SCISD) function.

Co-Regularized Hashing (CRH) (Zhen and Yeung, 2012b)

The CRH formulation can be solved by using CCCP to minimize the upper bound w.r.t. w_x

$$\mathcal{O}_x = \frac{\lambda_x \|w_x\|^2}{2} + \gamma \sum_{n=1}^N \omega_n (s_n d_n^2 + (1 - s_n) \zeta_n^x) + \frac{1}{I} \sum_{i=1}^I l_i^x$$

where

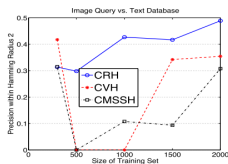
$$\zeta_n^x = \tau_1(d_n) - \tau_2(d_n^{(t)}) - d_n^{(t)} x_{a_n}^T (w_x - w_x^{(t)}), d_n^{(t)} = (w_x^{(t)})^T x_{a_n} - w_y^T y_{b_n}$$

Co-Regularized Hashing (CRH) (Zhen and Yeung, 2012b)

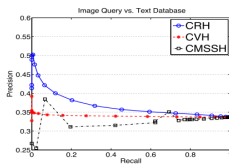
Table 2: mAP comparison on Flickr

Task	Method	Code Length		
		$K = 24$	$K = 48$	$K = 64$
Image Query vs. Text Database	CRH	0.5259 ± 0.0094	0.4990 ± 0.0075	0.4929 ± 0.0064
	CVH	0.4717 ± 0.0035	0.4515 ± 0.0041	0.4471 ± 0.0023
	CMSSH	0.5287 ± 0.0123	0.5098 ± 0.0141	0.4911 ± 0.0220
Text Query vs. Image Database	CRH	0.5364 ± 0.0021	0.5185 ± 0.0050	0.5064 ± 0.0055
	CVH	0.4598 ± 0.0020	0.4519 ± 0.0029	0.4477 ± 0.0058
	CMSSH	0.5029 ± 0.0321	0.4815 ± 0.0101	0.4660 ± 0.0298

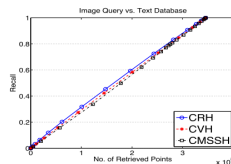
Co-Regularized Hashing (CRH) (Zhen and Yeung, 2012b)



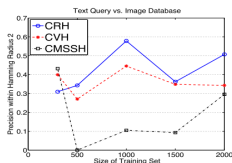
(a) Varying training set size



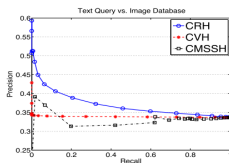
(b) Precision-recall curve



(c) Recall curve



(d) Varying training set size



(e) Precision-recall curve

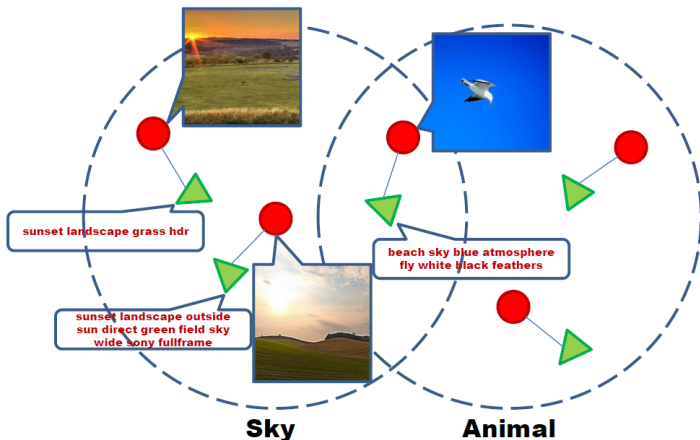


(f) Recall curve

Figure 2: Results on Flickr

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Supervised Multimodal Similarity Search



Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Motivation

- Existing supervised methods are not scalable
- Train hash functions on large-scale multimodal dataset

Contribution

- Avoiding explicitly computing the pairwise similarity matrix.
Linear-time complexity w.r.t. the size of training data.
- A sequential learning method with **closed-form solution** to each bit.
No hyper-parameters and **stopping conditions** are needed.

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Notations

- n - the number of training entities (data points)
- d_x, d_y - the dimensions of feature space in each modality
- $\{x_1, x_2, \dots, x_n | \mathbf{x}_i \in \mathbb{R}^{d_x}\}, \{y_1, y_2, \dots, y_n | \mathbf{y}_i \in \mathbb{R}^{d_y}\}$
- denote the feature vectors of the two modalities
- $\mathbf{X} \in \mathbb{R}^{n \times d_x}, \mathbf{Y} \in \mathbb{R}^{n \times d_y}$
- feature vectors form the rows of the data matrix
- m - number of semantic categories
- $\{l_1, l_2, \dots, l_n | \mathbf{l}_i \in \{0, 1\}^m\}$
- semantic labels, $l_{i,k} = 1$ denotes that the i th entity belongs to the k th semantic category

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Similarity Definition

Cosine similarity between the semantic label vectors. The similarity between the i th entity and the j th entity is defined as follows

$$\tilde{S}_{ij} = \frac{l_i \cdot l_j}{\|l_i\|_2 \|l_j\|_2},$$

$$\tilde{L}_{ik} = \frac{l_{i,k}}{\|l_i\|_2}, \text{ then } \tilde{S} = \tilde{L}\tilde{L}^T$$

Perform element-wise linear transformation on \tilde{S} to get our final **semantic similarity matrix** $S \in [-1, 1]^{n \times n}$

$$S = 2\tilde{S} - E = 2\tilde{L}\tilde{L}^T - \mathbf{1}_n \mathbf{1}_n^T,$$

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Objective

The two hash functions should preserve the semantic similarity cross modalities. We try to reconstruct the **semantic similarity matrix** by the learned hash codes. Hence, the objective function of our model is to minimize the following square error:

$$\begin{aligned} \min_{f,g} \sum_{i,j} \left(\frac{1}{c} f(x_i) g(y_j) - S_{ij} \right)^2 \\ \text{s.t. } \sum_i f(x_i) f(x_i)^T = nI_c \\ \sum_j g(y_j) g(y_j)^T = nI_c, \end{aligned}$$

The constraints try to make the bits between different hash functions uncorrelated.

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

In matrix form, we can rewrite the problem as follows

$$\begin{aligned} \min_{W_x, W_y} \quad & \| \text{sgn}(XW_x) \text{sgn}(YW_y)^T - cS \|_F^2 \\ \text{s.t.} \quad & \text{sgn}(XW_x)^T \text{sgn}(XW_x) = nI_c \\ & \text{sgn}(YW_y)^T \text{sgn}(YW_y) = nI_c. \end{aligned}$$

With simple algebra, we can transform the objective function into the following form

$$\begin{aligned} & \| \text{sgn}(XW_x) \text{sgn}(YW_y)^T - cS \|_F^2 \\ &= \text{tr}[(\text{sgn}(XW_x) \text{sgn}(YW_y)^T - cS) \times (\text{sgn}(XW_x) \text{sgn}(YW_y)^T - cS)^T] \\ &= \text{tr}(\text{sgn}(XW_x) \text{sgn}(YW_y)^T \text{sgn}(YW_y) \text{sgn}(XW_x)^T) \\ &\quad - 2c \cdot \text{tr}(\text{sgn}(XW_x)^T S \text{sgn}(YW_y)) + \text{tr}(c^2 S^2) \\ &= -2c \cdot \text{tr}(\text{sgn}(XW_x)^T S \text{sgn}(YW_y)) + \text{const}, \end{aligned}$$

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Then, we can reformulate the problem as the following equivalent problem:

$$\begin{aligned} \max_{W_x, W_y} \quad & tr \left(sgn(XW_x)^T S sgn(YW_y) \right) \\ \text{s.t.} \quad & sgn(XW_x)^T sgn(XW_x) = nI_c \\ & sgn(YW_y)^T sgn(YW_y) = nI_c \end{aligned}$$

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Semantic Correlation Maximization

Since it is NP hard to solve the problem, we apply the **spectral relaxation** trick to our problem

$$\begin{aligned} \max_{W_x, W_y} \quad & tr(W_x^T X^T S Y W_y) \\ \text{s.t.} \quad & W_x^T X^T X W_x = I_c \\ & W_y^T Y^T Y W_y = I_c. \end{aligned}$$

The term $X^T S Y$ actually measures the correlation between the two modalities with respect to the **semantic labels**. This correlation is called **semantic correlation**.

The objective function will degenerate to the CCA formulation when $S = I_n$.

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Generalized Eigenvalue Problem

The problem is equivalent to a **generalized eigenvalue problem**

Let $C_{xy} = X^T S Y$, $C_{xx} = X^T X$, and $C_{yy} = Y^T Y$.

The optimal solution of W_x is the eigenvectors corresponding to the c largest eigenvalues of

$$C_{xy} C_{yy}^{-1} C_{xy}^T W_x = \Lambda^2 C_{xx} W_x$$

W_y can be obtained by $W_y = C_{yy}^{-1} C_{xy}^T W_x \Lambda^{-1}$.

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Solved perfectly? **No!**

The **quantization loss** from projection space to Hamming space should not be neglected by **spectral relaxation**.

In eigen-decomposition, larger-variance projected dimensions carry more information. Hence, using each eigenvector to generate one bit of hash code is not reasonable (Kong and Li, 2012b).

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Sequential Strategy

Assuming that the projection vectors $w_x^{(1)}, \dots, w_x^{(t-1)}$ and $w_y^{(1)}, \dots, w_y^{(t-1)}$ have been learned. To learn the next projection vectors $w_x^{(t)}$ and $w_y^{(t)}$, we define a residue matrix

$$R_t = cS - \sum_{k=1}^{t-1} \text{sgn}(Xw_x^{(k)})\text{sgn}(Yw_y^{(k)})^T.$$

Objective function can be written as

$$\min_{w_x^{(t)}, w_y^{(t)}} \left\| \text{sgn}(Xw_x^{(t)})\text{sgn}(Yw_y^{(t)})^T - R_t \right\|_F^2.$$

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

The above objective function can be solved as a generalized eigenvalue problem which can be got by substituting $w_x^{(t)}$, $w_y^{(t)}$, and R_t for W_x , W_y , and S in the original formulation.

Note that the **semantic correlation** can be still efficiently calculated in linear time

$$\begin{aligned}
 C_{xy}^{(t)} &= X^T R_t Y \\
 &= c X^T S Y - \sum_{k=1}^{t-1} X^T \operatorname{sgn}(X w_x^{(k)}) \operatorname{sgn}(Y w_y^{(k)})^T Y \\
 &= C_{xy}^{(t-1)} - X^T \operatorname{sgn}(X w_x^{(t-1)}) \operatorname{sgn}(Y w_y^{(t-1)})^T Y,
 \end{aligned}$$

SCM (Zhang and Li, 2014)

Algorithm 3 Learning Algorithm of SCM Hashing Method.

$$C_{xy}^{(0)} \leftarrow 2(X^T \tilde{L})(Y^T \tilde{L})^T - (X^T \mathbf{1}_n)(Y^T \mathbf{1}_n)^T;$$

$$C_{xy}^{(1)} \leftarrow c \times C_{xy}^{(0)};$$

$$C_{xx} \leftarrow X^T X + \gamma I_{d_x};$$

$$C_{yy} \leftarrow Y^T Y + \gamma I_{d_y};$$

for $t = 1 \rightarrow c$ **do**

Solving the following generalized eigenvalue problem

$$C_{xy}^{(t)} C_{yy}^{-1} [C_{xy}^{(t)}]^T w_x = \lambda^2 C_{xx} w_x,$$

we can obtain the optimal solution $w_x^{(t)}$ corresponding to the largest eigenvalue

$$\lambda_{max};$$

$$w_y^{(t)} \leftarrow \frac{C_{yy}^{-1} C_{xy}^T w_x^{(t)}}{\lambda_{max}};$$

$$h_x^{(t)} \leftarrow \text{sgn}(X w_x^{(t)});$$

$$h_y^{(t)} \leftarrow \text{sgn}(Y w_y^{(t)});$$

$$C_{xy}^{(t+1)} \leftarrow C_{xy}^{(t)} - (X^T \text{sgn}(X w_x^{(t)}))(Y^T \text{sgn}(Y w_y^{(t)}))^T;$$

end for

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Scalability

Table: Training time (in seconds) on NUS-WIDE dataset by varying the size of training set.

Method \ Training Size	500	1000	1500	2000	2500	3000	5000	10000	20000
SCM	276	249	303	222	236	260	248	228	230
CCA	25	20	23	22	25	22	28	38	44
CVH	62	116	123	149	155	170	237	774	1630
CRH	68	253	312	515	760	1076	-	-	-
MLBE	67071	126431	-	-	-	-	-	-	-

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Scalability

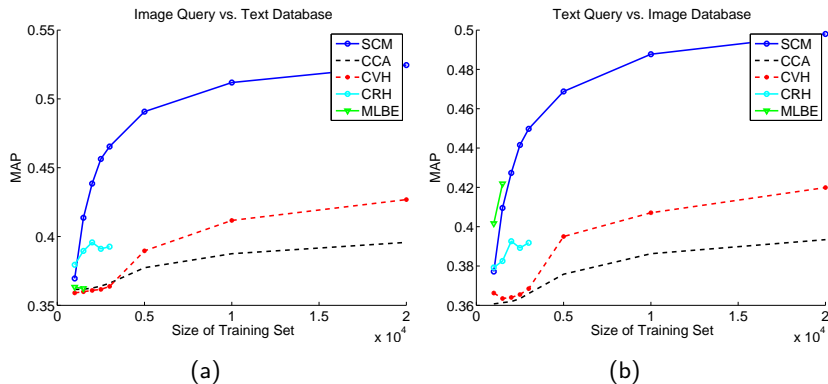


Figure: MAP on NUS-WIDE dataset by varying the size of training set.

Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)

Table: MAP results on small-scale training set of NUS-WIDE

Task	Method	Code Length		
		$c = 16$	$c = 24$	$c = 32$
Image Query v.s. Text Database	SCM	0.4385	0.4397	0.4390
	CCA	0.3625	0.3586	0.3565
	CVH	0.3608	0.3575	0.3562
	CRH	0.3957	0.3965	0.3970
	MLBE	0.3697	0.3620	0.3540
Text Query v.s. Image Database	SCM	0.4273	0.4265	0.4259
	CCA	0.3619	0.3580	0.3560
	CVH	0.3640	0.3596	0.3581
	CRH	0.3926	0.3910	0.3904
	MLBE	0.3877	0.3636	0.3551

Table: MAP results on large-scale training set of NUS-WIDE

Task	Method	Code Length		
		$c = 16$	$c = 24$	$c = 32$
Image Query v.s. Text Database	SCM	0.5451	0.5501	0.5412
	CCA	0.4078	0.3964	0.3886
Text Query v.s. Image Database	SCM	0.5147	0.5153	0.5105
	CCA	0.4038	0.3934	0.3861

Collective Matrix Factorization Hashing (Ding et al., 2014)

Employ collective matrix factorization (CMF) to learn cross-view hash functions.

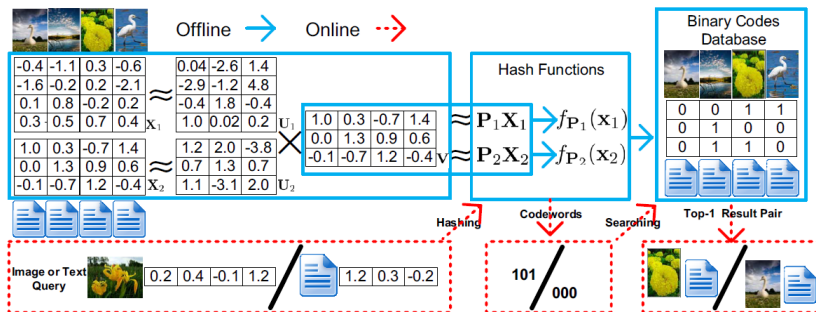


Figure 2. Framework of CMFH, illustrated with toy data.

Collective Matrix Factorization Hashing (Ding et al., 2014)

Decompose $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ jointly with the constraint $\mathbf{V}_1 = \mathbf{V}_2 = \mathbf{V}$

$$\lambda \|\mathbf{X}^{(1)} - \mathbf{U}_1 \mathbf{V}\|_F^2 + (1 - \lambda) \|\mathbf{X}^{(2)} - \mathbf{U}_2 \mathbf{V}\|_F^2$$

The overall objective function contains the **collective matrix factorization**, the **linear embedding** and the **regularization term**:

$$\min_{\mathbf{U}_1, \mathbf{U}_2, \mathbf{P}_1, \mathbf{P}_2, \mathbf{V}} G(\mathbf{U}_1, \mathbf{U}_2, \mathbf{P}_1, \mathbf{P}_2, \mathbf{V})$$

where

$$\begin{aligned} G = & \lambda \|\mathbf{X}^{(1)} - \mathbf{U}_1 \mathbf{V}\|_F^2 + (1 - \lambda) \|\mathbf{X}^{(2)} - \mathbf{U}_2 \mathbf{V}\|_F^2 \\ & + \mu (\|\mathbf{V} - \mathbf{P}_1 \mathbf{X}^{(1)}\|_F^2 + \|\mathbf{V} - \mathbf{P}_2 \mathbf{X}^{(2)}\|_F^2) \\ & + \gamma R(\mathbf{U}_1, \mathbf{U}_2, \mathbf{P}_1, \mathbf{P}_2, \mathbf{V}) \end{aligned}$$

Collective Matrix Factorization Hashing (Ding et al., 2014)

Mean Average Precision(MAP) on Wiki dataset and NUS-WIDE dataset.

Task	Method	Wiki				NUS-WIDE			
		16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
View1 to View2	CVH	0.2041	0.1604	0.1296	0.1308	0.3722	0.3632	0.4060	0.3875
	IMH	0.2017	0.2119	0.2025	0.1926	0.4721	0.4716	0.4668	0.4594
	CMSSH	0.2026	0.2113	0.2011	0.2084	0.4997	0.5189	0.5142	0.5086
	CHMIS	0.2207	0.2189	0.2126	0.2035	0.4870	0.4896	0.4835	0.4762
	CMFH	0.2538	0.2582	0.2619	0.2648	0.5591	0.5698	0.5780	0.5837
View2 to View1	CVH	0.2962	0.1944	0.1337	0.1125	0.4042	0.3991	0.4480	0.4315
	IMH	0.4865	0.5295	0.4935	0.4628	0.4793	0.4800	0.4744	0.4643
	CMSSH	0.2928	0.2732	0.2753	0.2814	0.5015	0.5103	0.5039	0.4984
	CHMIS	0.2207	0.2189	0.2126	0.2035	0.4870	0.4896	0.4835	0.4762
	CMFH	0.6116	0.6298	0.6398	0.6477	0.6614	0.6921	0.7164	0.7185

Quantized Correlation Hashing (Wu et al., 2015)

Contributions

- First attempt to integrate hash function learning with quantization together for cross-modal hashing
- Multi-modality objective function is transformed to a single-modality formulation

Framework

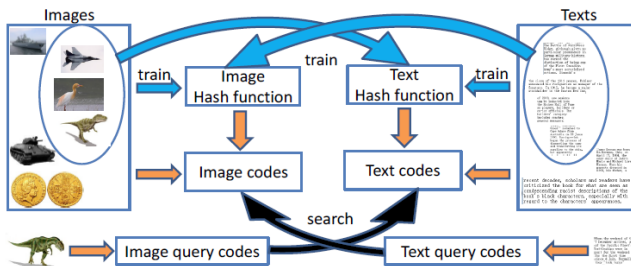


Figure 1: The framework of cross-modal hashing algorithm

Quantized Correlation Hashing (Wu et al., 2015)

Objective function

$$\min O(\mathbf{B}, \mathbf{W}) = \|\mathbf{B} - \mathbf{Z}\mathbf{W}\|_F^2 - \text{tr}(\mathbf{W}^T \mathbf{Z}^T \tilde{\mathbf{S}} \mathbf{Z} \mathbf{W})$$

$$\text{where } \mathbf{W} = \begin{bmatrix} \mathbf{W}_x \\ \mathbf{W}_y \end{bmatrix}, \tilde{\mathbf{S}} = \begin{bmatrix} \beta \mathbf{L}_x & \alpha \mathbf{S} \\ \alpha \mathbf{S}^T & \beta \mathbf{L}_y \end{bmatrix}, \mathbf{Z} = \begin{bmatrix} \mathbf{X} & \\ & \mathbf{Y} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_x \\ \mathbf{B}_y \end{bmatrix}$$

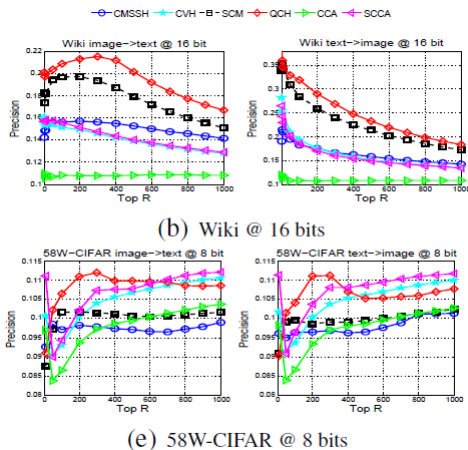
Learn hash function

Adopt an alternating optimization procedure to iteratively optimize \mathbf{W} and \mathbf{B}

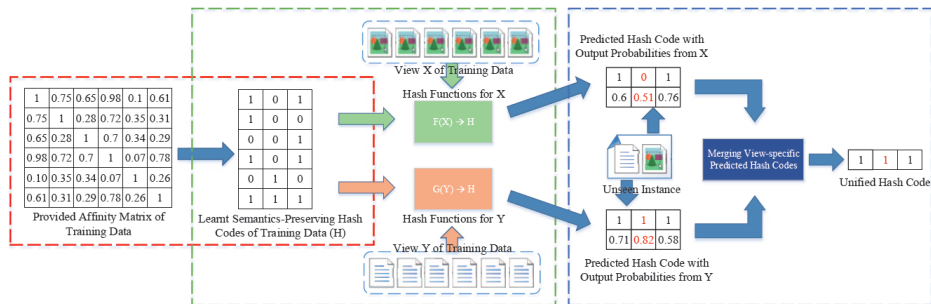
- Optimize \mathbf{B} using $\mathbf{B} = \text{sgn}(\mathbf{Z}\mathbf{W})$ when fixing \mathbf{W}
- Iteratively optimize \mathbf{W} using Crank-Nicolson-like scheme (Smith, 1985) when fixing \mathbf{B}

Quantized Correlation Hashing (Wu et al., 2015)

Precision-Recall curve on Wiki dataset and 58W-CIFAR dataset



Semantics-Preserving Hashing (Lin et al., 2015b)



Training of SePH: 1) Learning semantics-preserving hash codes of the training data (red dotted rectangle),
2) Learning hash functions for each view (green dotted rectangle)

Out-of-sample Extension: 1) Predicting hash codes from observed views,
2) Determining the unified hash code using a novel probabilistic approach

Semantics-Preserving Hashing (Lin et al., 2015b)

Objective function: the KL-divergence between two distributions \mathcal{P} and \mathcal{Q}

$$\Psi = \min_{\hat{H} \in \mathbb{R}^{n \times d_c}} \sum_{i \neq j} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}} + \frac{\alpha}{C} |||\hat{H}| - \mathbf{I}||_2^2$$

where \hat{H} is the relaxed hash-code matrix, $p_{i,j} = \frac{A_{i,j}}{\sum_{i \neq j} A_{i,j}}$ with $A_{i,j}$ being the supervised affinity between points i and j ,

$$q_{i,j} = \frac{(1 + \frac{1}{4} ||\hat{H}_{i,\cdot} - \hat{H}_{j,\cdot}||_2^2)^{-1}}{\sum_{k \neq m} (1 + \frac{1}{4} ||\hat{H}_{k,\cdot} - \hat{H}_{m,\cdot}||_2^2)^{-1}}$$

Learning algorithm:

- Utilize gradient descent based optimization methods.
- The gradient w.r.t. $\hat{H}_{i,\cdot}$ can be derived as follows

$$\begin{aligned} \frac{\partial \Psi}{\partial \hat{H}_{i,\cdot}} = & \sum_{j \neq i} (p_{i,j} - q_{i,j}) (1 + \frac{1}{4} ||\hat{H}_{i,\cdot} - \hat{H}_{j,\cdot}||_2^2)^{-1} (\hat{H}_{i,\cdot} - \hat{H}_{j,\cdot}) \\ & + \frac{2\alpha}{C} (||\hat{H}_{i,\cdot}|| - \mathbf{1}^T) \odot \sigma(\hat{H}_{i,\cdot}) \end{aligned}$$

Semantics-Preserving Hashing (Lin et al., 2015b)

Learn hash functions

After getting the hash code, kernel logistic regression is used to learn the hash functions:

$$\Theta = \min_{\mathbf{w}^{(k)}} \sum_{i=1}^n \log(1 + e^{-\mathbf{h}_i^{(k)} \phi(\mathbf{X}_{i,\cdot}) \mathbf{w}^{(k)}}) + \lambda \|\mathbf{w}^{(k)}\|_2^2$$

Generate hash codes

- For any unseen instance with only one view observed, utilize $p(\mathbf{c}_k^{\mathcal{X}} = b | \mathbf{x}) = (1 + e^{-b(\phi(\mathbf{x}) \hat{\Phi}^T) \hat{\mathbf{v}}^{(k)}})^{-1}$
- For those with both views observed, utilize $\mathbf{c}_k = \text{sgn}(p(\mathbf{c}_k = 1 | \mathbf{x})p(\mathbf{c}_k = 1 | \mathbf{y}) - p(\mathbf{c}_k = -1 | \mathbf{x})p(\mathbf{c}_k = -1 | \mathbf{y}))$ to generate hash code.

Semantics-Preserving Hashing (Lin et al., 2015b)

Mean Average Precision (MAP):

		Wiki				MIRFlickr				NUS-WIDE			
		16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
Image Query v.s. Text Database	CMSSH [1]	0.1877	0.1771	0.1646	0.1552	0.5728	0.5743	0.5706	0.5706	0.4063	0.3927	0.3939	0.3739
	CVH [9]	0.1257	0.1212	0.1215	0.1171	0.6067	0.6177	0.6157	0.6074	0.3687	0.4182	0.4602	0.4466
	IMH [14]	0.1573	0.1575	0.1568	0.1651	0.6016	0.6120	0.6070	0.5982	0.4187	0.3975	0.3778	0.3668
	LSSH [25]	0.2141	0.2216	0.2218	0.2211	0.5784	0.5804	0.5797	0.5816	0.3900	0.3924	0.3962	0.3966
	CMFH [3]	0.2132	0.2259	0.2362	0.2419	0.5861	0.5835	0.5844	0.5849	0.4267	0.4229	0.4207	0.4182
	KSH-CV [26]	0.1965	0.1839	0.1701	0.1662	0.5793	0.5767	0.5732	0.5744	0.4229	0.4162	0.4026	0.3877
	SCM-Orth [19]	0.1598	0.1460	0.1383	0.1131	0.5854	0.5751	0.5704	0.5649	0.3787	0.3668	0.3593	0.3520
	SCM-Seq [19]	0.2210	0.2337	0.2442	0.2596	0.6237	0.6343	0.6448	0.6489	0.4842	0.4941	0.4947	0.4965
	SePH _{rnd}	0.2762	0.2965	0.3049	0.3131	0.6720	0.6761	0.6794	0.6814	0.5394	0.5454	0.5499	0.5556
	SePH _{km}	0.2787	0.2956	0.3064	0.3134	0.6723	0.6771	0.6783	0.6817	0.5421	0.5499	0.5537	0.5601
Text Query v.s. Image Database	CMSSH [1]	0.1630	0.1617	0.1539	0.1517	0.5715	0.5732	0.5699	0.5697	0.3874	0.3849	0.3704	0.3699
	CVH [9]	0.1185	0.1034	0.1024	0.0990	0.6026	0.6041	0.6017	0.5972	0.3646	0.4024	0.4339	0.4255
	IMH [14]	0.1463	0.1311	0.1290	0.1301	0.5895	0.6031	0.6010	0.5930	0.4053	0.3892	0.3758	0.3627
	LSSH [25]	0.5031	0.5224	0.5293	0.5346	0.5898	0.5927	0.5932	0.5932	0.4286	0.4248	0.4248	0.4175
	CMFH [3]	0.4884	0.5132	0.5269	0.5375	0.5937	0.5919	0.5931	0.5919	0.4627	0.4556	0.4518	0.4478
	KSH-CV [26]	0.1710	0.1665	0.1696	0.1576	0.5786	0.5763	0.5728	0.5715	0.4088	0.3906	0.3869	0.3834
	SCM-Orth [19]	0.1553	0.1389	0.1262	0.1096	0.5857	0.5747	0.5672	0.5604	0.3756	0.3641	0.3565	0.3523
	SCM-Seq [19]	0.2134	0.2366	0.2479	0.2573	0.6133	0.6209	0.6295	0.6340	0.4536	0.4620	0.4630	0.4644
	SePH _{rnd}	0.6312	0.6581	0.6637	0.6695	0.7178	0.7243	0.7287	0.7313	0.6230	0.6331	0.6407	0.6489
	SePH _{km}	0.6318	0.6577	0.6646	0.6709	0.7197	0.7271	0.7309	0.7354	0.6302	0.6425	0.6506	0.6580

Table 2. Cross-view retrieval performance of the proposed SePH (*i.e.* SePH_{rnd} and SePH_{km}) and compared baselines on all benchmark datasets with different hash code lengths, in terms of *mAP*.

Outline

- 1 Introduction
- 2 Unsupervised Hashing
- 3 Supervised Hashing
- 4 Ranking-based Hashing
- 5 Multimodal Hashing
- 6 Deep Hashing**
- 7 Quantization
- 8 Conclusion
- 9 Reference

Deep Hashing

Deep learning for hashing

- **CNNH**: Supervised hashing via image representation learning (Xia et al., 2014)
- **NINH**: Simultaneous feature learning and hash coding with deep neural networks (Lai et al., 2015)
- **DSRH**: Deep semantic ranking based hashing (Zhao et al., 2015)
- **DRSCH**: Bit-scalable deep hashing (Zhang et al., 2015)
- **DH**: Deep hashing for compact binary codes learning (Liong et al., 2015)
- Deep learning of binary hash codes (Lin et al., 2015a)
- **DPSH**: Feature learning based deep supervised hashing with pairwise labels (Li et al., 2015)

Supervised Hashing via Image Representation Learning (CNNH) (Xia et al., 2014)

Two-stage framework

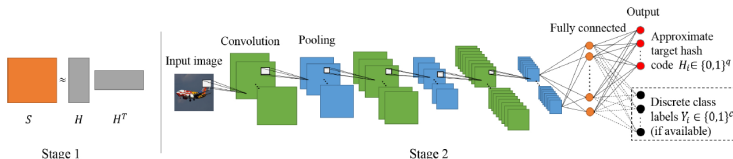


Figure 1: Overview of the proposed two-stage method. In stage 1, the pairwise similarity matrix S is decomposed into a product HH^T , where H is a matrix of approximate target hash codes. In stage 2, we use a convolutional network to learn the feature representation for the images as well as a set of hash functions. The network consists of three convolution-pooling layers, a fully connected layer and an output layer. The output layer can be simply constructed with the learned hash codes in H (the red nodes). If the image tags are available in training, one can add them in the output layer (the black nodes) so as to help to learn a better shared representation of the images. By inputting an test image to the trained network, one can obtain the desired hash code from the values of the red nodes in the output layer.

Supervised Hashing via Image Representation Learning (CNNH) (Xia et al., 2014)

Method	MNIST(MAP)				CIFAR10(MAP)				NUS-WIDE(MAP)			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48bits	12 bits	24 bits	32 bits	48 bits
CNNH+	0.969	0.975	0.971	0.975	0.465	0.521	0.521	0.532	0.623	0.630	0.629	0.625
CNNH	0.957	0.963	0.956	0.960	0.439	0.511	0.509	0.522	0.611	0.618	0.625	0.608
KSH	0.872	0.891	0.897	0.900	0.303	0.337	0.346	0.356	0.556	0.572	0.581	0.588
ITQ-CCA	0.659	0.694	0.714	0.726	0.264	0.282	0.288	0.295	0.435	0.435	0.435	0.435
MLH	0.472	0.666	0.652	0.654	0.182	0.195	0.207	0.211	0.500	0.514	0.520	0.522
BRE	0.515	0.593	0.613	0.634	0.159	0.181	0.193	0.196	0.485	0.525	0.530	0.544
SH	0.265	0.267	0.259	0.250	0.131	0.135	0.133	0.130	0.433	0.426	0.426	0.423
ITQ	0.388	0.436	0.422	0.429	0.162	0.169	0.172	0.175	0.452	0.468	0.472	0.477
LSH	0.187	0.209	0.235	0.243	0.121	0.126	0.120	0.120	0.403	0.421	0.426	0.441

Table 1: MAP of Hamming ranking w.r.t different number of bits on three datasets. For NUS-WIDE, we calculate the MAP values within the top 5000 returned neighbors. The results of CNNH / CNNH+ are the average of 5 trials.

Supervised Hashing via Image Representation Learning (CNNH) (Xia et al., 2014)

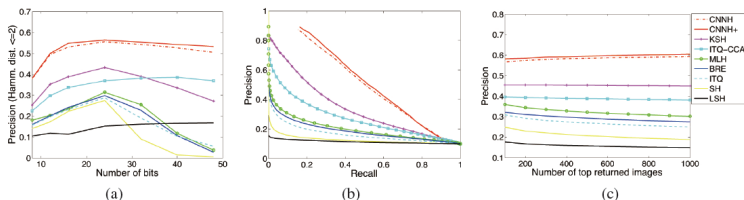


Figure 3: The results on CIFAR10. (a) precision curves within Hamming radius 2; (b) precision-recall curves of Hamming ranking with 48 bits; (c) precision curves with 48 bits w.r.t. different number of top returned samples

Simultaneous Feature Learning and Hash Coding with Deep Neural Networks (NINH) (Lai et al., 2015)

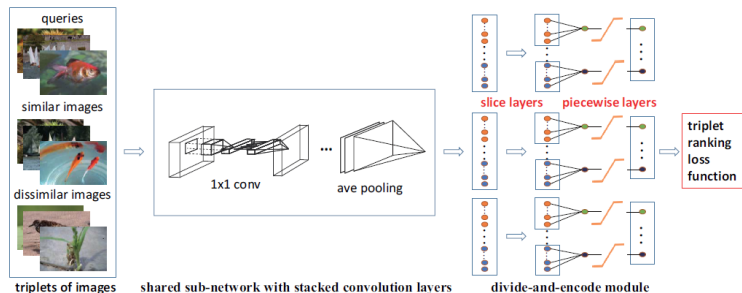


Figure 1. Overview of the proposed deep architecture for hashing. The input to the proposed architecture is in the form of triplets, i.e., (I, I^+, I^-) with a query image I being more similar to an image I^+ than to another image I^- . Through the proposed architecture, the image triplets are first encoded into a triplet of image feature vectors by a shared stack of multiple convolution layers. Then, each image feature vector in the triplet is converted to a hash code by a divide-and-encode module. After that, these hash codes are used in a triplet ranking loss that aims to preserve relative similarities on images.

Simultaneous Feature Learning and Hash Coding with Deep Neural Networks (NINH) (Lai et al., 2015)

The loss function:

$$\begin{aligned} & \ell_{\text{triplet}}(\mathcal{F}(I), \mathcal{F}(I^+), \mathcal{F}(I^-)) \\ &= \max(0, \|\mathcal{F}(I) - \mathcal{F}(I^+)\|_2^2 - \|\mathcal{F}(I) - \mathcal{F}(I^-)\|_2^2 + 1) \\ & \text{s.t. } \mathcal{F}(I), \mathcal{F}(I^+), \mathcal{F}(I^-) \in [0, 1]^q. \end{aligned}$$

Simultaneous Feature Learning and Hash Coding with Deep Neural Networks (NINH) (Lai et al., 2015)

Table 2. MAP of Hamming ranking w.r.t different numbers of bits on three datasets. For NUS-WIDE, we calculate the MAP values within the top 5000 returned neighbors. The results of CNNH is directly cited from [27]. CNNH* is our implementation of the CNNH method in [27] using Caffe, by using a network configuration comparable to that of the proposed method (see the text in Section 4.1 for implementation details).

Method	SVHN(MAP)				CIFAR-10(MAP)				NUS-WIDE(MAP)			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48bits	12 bits	24 bits	32 bits	48 bits
Ours	0.899	0.914	0.925	0.923	0.552	0.566	0.558	0.581	0.674	0.697	0.713	0.715
CNNH*	0.897	0.903	0.904	0.896	0.484	0.476	0.472	0.489	0.617	0.663	0.657	0.688
CNNH [27]	N/A				0.439	0.511	0.509	0.522	0.611	0.618	0.625	0.608
KSH [12]	0.469	0.539	0.563	0.581	0.303	0.337	0.346	0.356	0.556	0.572	0.581	0.588
ITQ-CCA [4]	0.428	0.488	0.489	0.509	0.264	0.282	0.288	0.295	0.435	0.435	0.435	0.435
MLH [16]	0.147	0.247	0.261	0.273	0.182	0.195	0.207	0.211	0.500	0.514	0.520	0.522
BRE [8]	0.165	0.206	0.230	0.237	0.159	0.181	0.193	0.196	0.485	0.525	0.530	0.544
SH [26]	0.140	0.138	0.141	0.140	0.131	0.135	0.133	0.130	0.433	0.426	0.426	0.423
ITQ [4]	0.127	0.132	0.135	0.139	0.162	0.169	0.172	0.175	0.452	0.468	0.472	0.477
LSH [2]	0.110	0.122	0.120	0.128	0.121	0.126	0.120	0.120	0.403	0.421	0.426	0.441

Deep Semantic Ranking Based Hashing for Multi-Label Image Retrieval (DSRH) (Zhao et al., 2015)

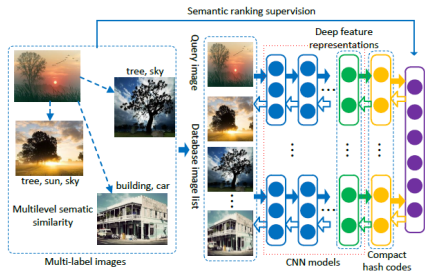


Figure 1. The proposed deep semantic ranking based hashing. Solid and hollow arrows indicate forward and backward propagation directions of features and gradients respectively. Hash functions consist of deep convolutional neural network (CNN) and binary mappings of the feature representation from the top hidden layers of CNN. Multilevel semantic ranking information is used to learn such deep hash functions to preserve the semantic structure of multi-label images.

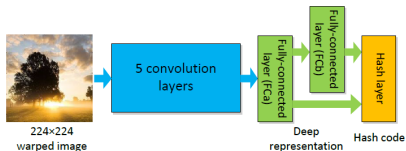


Figure 2. The structure of deep hash functions. An input image is first transformed to a fixed size, and then goes through five convolution layers and two fully-connected layers, which provides a deep feature representation. Finally, the hash layer generates a compact binary code. The hash layer is also directly connected to the first fully-connected layer (FCa) in order to utilize diverse feature information biased toward visual appearance.

Deep Semantic Ranking Based Hashing for Multi-Label Image Retrieval (DSRH) (Zhao et al., 2015)

The major part:

- Hash functions:

$$h(\mathbf{x}; \mathbf{w}) = \text{sign}(\mathbf{w}^T [f_a(\mathbf{x}); f_b(\mathbf{x})]),$$

- Semantic ranking supervision
- Optimization with surrogate loss (weighted triplet loss):

$$\begin{aligned} \mathcal{F}(\mathbf{W}) = & \sum_{\mathbf{q} \in \mathcal{D}, \{\mathbf{x}_i\}_{i=1}^M \subset \mathcal{D}} L_{\omega}(\mathbf{h}(\mathbf{q}; \mathbf{W}), \{\mathbf{h}(\mathbf{x}_i; \mathbf{W})\}_{i=1}^M) \\ & + \frac{\alpha}{2} \left\| \text{mean}_{\mathbf{q}}(\mathbf{h}(\mathbf{q}; \mathbf{W})) \right\|_2^2 + \frac{\beta}{2} \|\mathbf{W}\|_2^2. \end{aligned}$$

Deep Semantic Ranking Based Hashing for Multi-Label Image Retrieval (DSRH) (Zhao et al., 2015)

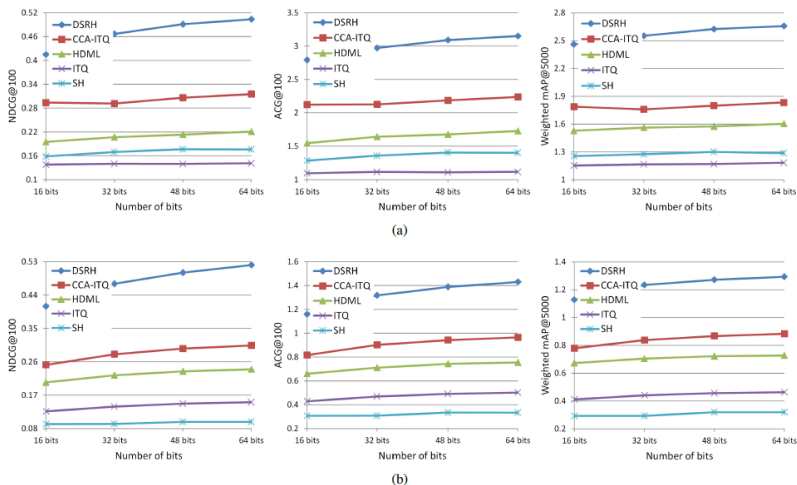


Figure 4. Comparison of ranking performance of our DSRH and other hashing methods based on hand-crafted features on two datasets: (a) MIRFLICKR-25K and (b) NUS-WIDE.

Deep Semantic Ranking Based Hashing for Multi-Label Image Retrieval (DSRH) (Zhao et al., 2015)

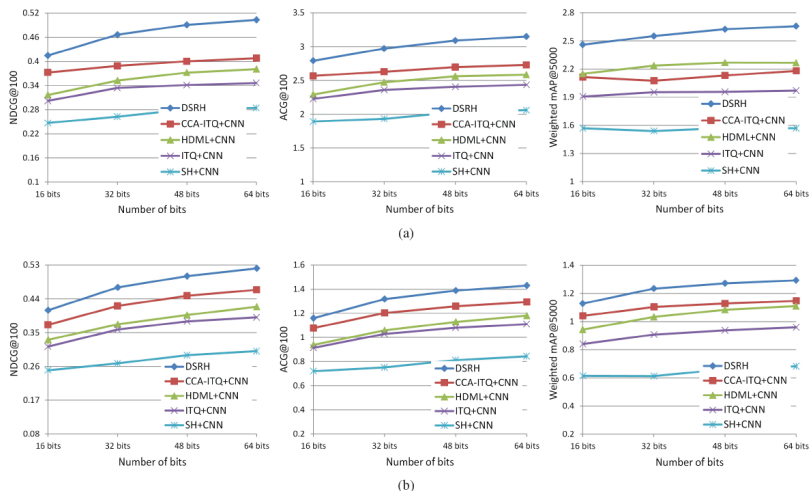


Figure 5. Comparison of ranking performance of our DSRH and other hashing methods based on activation features of pre-trained CNN on two datasets: (a) MIRFLICKR-25K and (b) NUS-WIDE.

Bit-Scalable Deep Hashing (DRSCH) (Zhang et al., 2015)

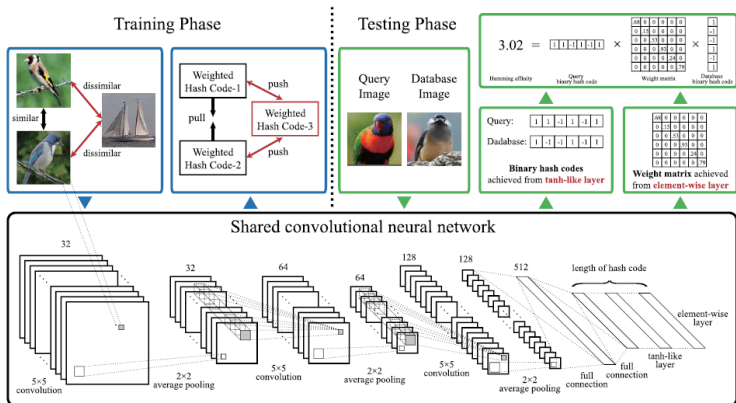


Fig. 2. The bit-scalable deep hashing learning framework. The bottom panel shows the deep architecture of neural network that produces the hashing code with the weight matrix by taking raw images as inputs. The training stage is illustrated in the left up panel, where we train the network with triplet-based similarity learning. An example of hashing retrieval is presented in the right up panel, where the similarity is measured by the Hamming affinity.

Bit-Scalable Deep Hashing (DRSCH) (Zhang et al., 2015)

Method	MNIST (MAP %)				
	16 bits	24 bits	32 bits	48 bits	64 bits
DRSCH	96.92	97.37	97.88	97.91	98.09
DSCH	96.51	96.63	97.21	97.48	97.68
DSRH [40]	96.48	96.69	97.21	97.53	97.75
KSH-CNN [7]	83.89	86.67	88.51	89.41	89.67
MLH-CNN [12]	71.03	76.18	78.06	80.66	80.87
BRE-CNN [39]	61.00	64.05	64.11	66.33	67.02
KSH [7]	82.85	86.03	87.37	88.48	88.82
MLH [12]	45.77	62.16	63.07	65.23	66.70
BRE [39]	41.96	57.19	56.52	64.74	66.55
PCA-RR [14]	35.96	39.93	38.17	43.81	45.76
ITQ [14]	34.44	38.99	40.62	43.04	41.76
SH [13]	13.40	14.81	15.28	16.29	17.11
LSH [18]	22.65	21.39	35.56	27.85	37.78
Euclidean	89.55	87.83	86.89	83.76	82.92

TABLE I

IMAGE RETRIEVAL RESULTS (MEAN AVERAGE PRECISION) WITH VARIOUS NUMBER OF BITS ON THE MNIST DATASET. THE SCALE OF TEST QUERY SET IS 10K. OUR METHOD OUTPERFORMS THE STATE-OF-THE-ART METHODS.

Method	CIFAR-10 (MAP %)				
	16 bits	24 bits	32 bits	48 bits	64 bits
DRSCH	61.46	62.19	62.87	63.05	63.26
DSCH	60.87	61.33	61.74	61.98	62.35
DSRH [40]	60.84	61.08	61.74	61.77	62.91
KSH-CNN [7]	40.08	42.98	44.39	45.77	46.56
MLH-CNN [12]	25.04	28.86	31.29	31.88	31.83
BRE-CNN [39]	19.80	20.57	20.59	21.64	21.96
KSH [7]	32.15	35.17	36.51	38.26	39.50
MLH [12]	13.33	15.78	16.29	18.03	18.84
BRE [39]	12.19	15.63	16.10	17.19	17.56
PCA-RR [14]	12.06	12.24	13.61	13.46	13.80
ITQ [14]	11.45	11.63	11.53	10.97	11.24
SH [13]	19.22	19.28	20.09	20.79	21.46
LSH [18]	12.36	11.74	12.30	13.57	12.42
Euclidean	35.46	34.07	33.91	32.18	31.09

TABLE II

IMAGE RETRIEVAL RESULTS (MEAN AVERAGE PRECISION) WITH VARIOUS NUMBER OF BITS ON THE CIFAR-10 DATASET. THE SCALE OF TEST QUERY SET IS 10K (1K PER CLASS). THE PROPOSED METHOD OUTPERFORMS THE STATE-OF-THE-ART METHODS.

Bit-Scalable Deep Hashing (DRSCH) (Zhang et al., 2015)

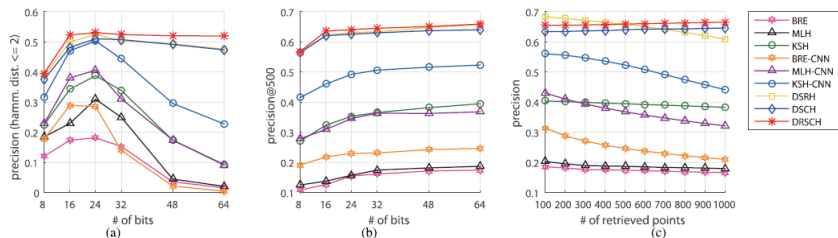


Fig. 4. The results on the CIFAR-10 dataset. (a) Precision curves within Hamming radius 2; (b) Precision curves with top 500 returned; (c) Precision curves with 64 hash bits.

Deep Supervised Hashing with Pairwise Labels (Li et al., 2015)

Motivation

- Most existing hashing methods are based on **hand-crafted features** which might not be optimally compatible with the hashing procedure.
- Recently, **deep hashing** is proposed for simultaneous feature learning and hash-code learning, with better performance.
- Most existing deep hashing methods are supervised with **triplet labels**. For **pairwise labels**, there have not existed methods for simultaneous feature learning and hash-code learning.

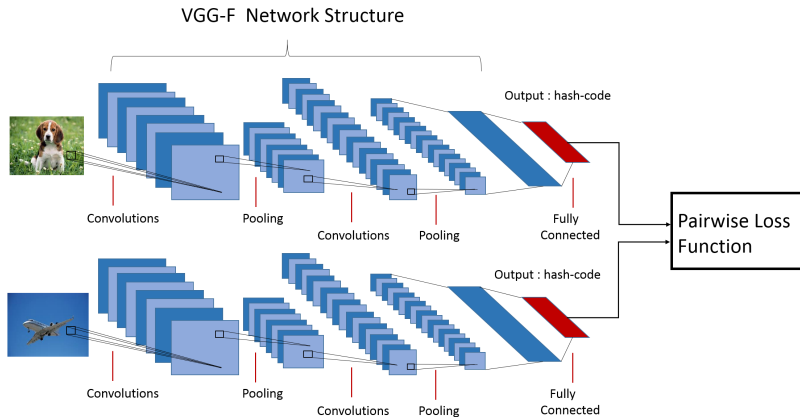
Our contribution:

- An end-to-end framework, called **deep pairwise-supervised hashing (DPSH)**, to perform simultaneous feature learning and hash-code learning for applications with pairwise labels.

Deep Supervised Hashing with Pairwise Labels (Li et al., 2015)

VGG-F + hash-code: $7 + 1 = 8$ layers.

Five conv+pooling, two 4096 fully-connected, one fully-connected for hash-code



Deep Supervised Hashing with Pairwise Labels (Li et al., 2015)

The same pairwise loss function as LFH:

$$\begin{aligned} L &= -\log p(\mathcal{S}|\mathcal{B}) = -\sum_{s_{ij} \in \mathcal{S}} \log p(s_{ij}|\mathcal{B}) \\ &= -\sum_{s_{ij} \in \mathcal{S}} (s_{ij}\Theta_{ij} - \log(1 + e^{\Theta_{ij}})), \end{aligned}$$

where $s_{ij} \in \{0, 1\}$ is the supervised label, $\mathcal{B} = \{\mathbf{b}_i\}_{i=1}^n$ is the binary codes, $\Theta_{ij} = \frac{1}{2}\mathbf{b}_i^T \mathbf{b}_j$.

Similar continuous relaxation as that in LFH is adopted for learning.

Deep Supervised Hashing with Pairwise Labels (Li et al., 2015)

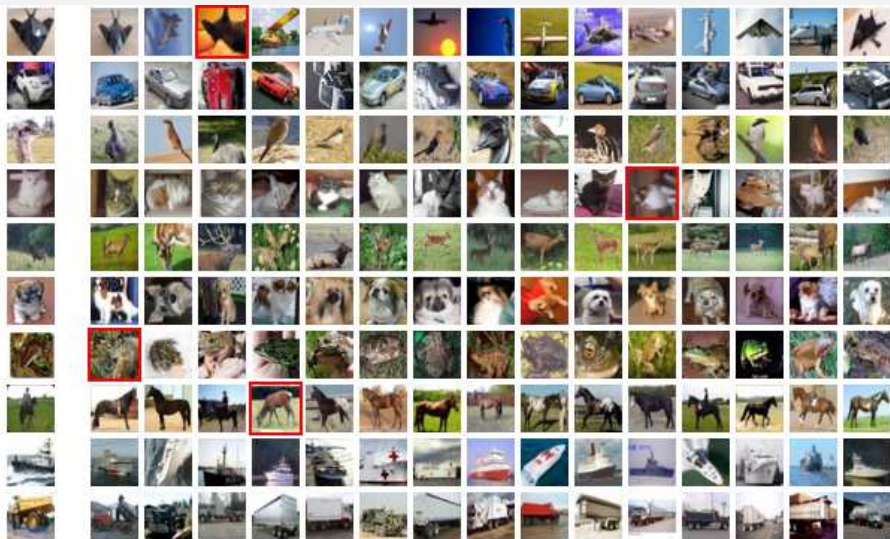
Method	CIFAR-10 (MAP)				NUS-WIDE (MAP)			
	12-bits	24-bits	32-bits	48-bits	12-bits	24-bits	32-bits	48-bits
DPSH	0.682	0.686	0.725	0.733	0.785	0.810	0.824	0.833
DPSH0	0.458	0.512	0.521	0.525	0.723	0.743	0.748	0.758
NINH [CVPR15]	0.552	0.566	0.558	0.581	0.674	0.697	0.713	0.715
CNNH [AAAI14]	0.439	0.476	0.472	0.489	0.611	0.618	0.625	0.608
FastH	0.305	0.349	0.369	0.384	0.621	0.650	0.665	0.687
SDH	0.285	0.329	0.341	0.356	0.568	0.600	0.608	0.637
KSH	0.303	0.337	0.346	0.356	0.556	0.572	0.581	0.588
LFH	0.278	0.435	0.518	0.561	0.747	0.784	0.808	0.802
LFH0	0.176	0.231	0.211	0.253	0.571	0.568	0.568	0.585
SPLH	0.171	0.173	0.178	0.184	0.568	0.589	0.597	0.601
ITQ	0.162	0.169	0.172	0.175	0.452	0.468	0.472	0.477
SH	0.127	0.128	0.126	0.129	0.454	0.406	0.405	0.400

Deep Supervised Hashing with Pairwise Labels (Li et al., 2015)

Compare to baselines with triplet labels:

Method	CIFAR-10 (MAP)				NUS-WIDE (MAP)			
	16-bits	24-bits	32-bits	48-bits	16-bits	24-bits	32-bits	48-bits
DPSH	0.742	0.764	0.765	0.770	0.698	0.711	0.726	0.730
DRSCH [TIP15]	0.615	0.622	0.629	0.631	0.618	0.622	0.623	0.628
DSCH [TIP15]	0.609	0.613	0.617	0.620	0.592	0.597	0.611	0.609
DSRH [CVPR15]	0.608	0.611	0.617	0.618	0.609	0.618	0.621	0.631

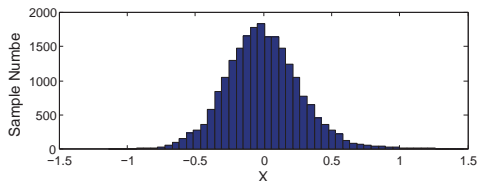
Deep Supervised Hashing with Pairwise Labels (Li et al., 2015)



Outline

- 1 Introduction
- 2 Unsupervised Hashing
- 3 Supervised Hashing
- 4 Ranking-based Hashing
- 5 Multimodal Hashing
- 6 Deep Hashing
- 7 Quantization**
- 8 Conclusion
- 9 Reference

Double Bit Quantization (Kong and Li, 2012a)



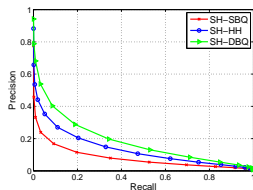
(a)	A 0 BC		1 D	
(b)	01	00	10	11
(c)	01	00	10	

Point distribution of the real values computed by PCA on *22K LabelMe* data set, and different coding results based on the distribution:

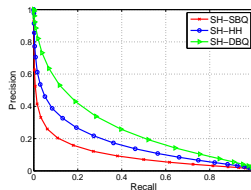
- (a) single-bit quantization (SBQ);
- (b) hierarchical hashing (HH) (Liu et al., 2011);
- (c) double-bit quantization (DBQ).

Double Bit Quantization (Kong and Li, 2012a)

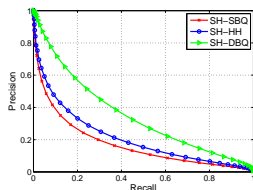
Precision-recall curve on 22K LabelMe data set



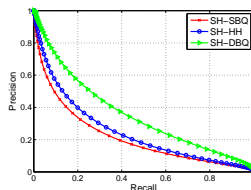
SH 32 bits



SH 64 bits



SH 128 bits



SH 256 bits

Double Bit Quantization (Kong and Li, 2012a)

mAP on LabelMe data set

# bits	32			64		
	SBQ	HH	DBQ	SBQ	HH	DBQ
ITQ	0.2926	0.2592	0.3079	0.3413	0.3487	0.4002
SH	0.0859	0.1329	0.1815	0.1071	0.1768	0.2649
PCA	0.0535	0.1009	0.1563	0.0417	0.1034	0.1822
LSH	0.1657	0.105	0.12272	0.2594	0.2089	0.2577
SIKH	0.0590	0.0712	0.0772	0.1132	0.1514	0.1737

# bits	128			256		
	SBQ	HH	DBQ	SBQ	HH	DBQ
ITQ	0.3675	0.4032	0.4650	0.3846	0.4251	0.4998
SH	0.1730	0.2034	0.3403	0.2140	0.2468	0.3468
PCA	0.0323	0.1083	0.1748	0.0245	0.1103	0.1499
LSH	0.3579	0.3311	0.4055	0.4158	0.4359	0.5154
SIKH	0.2792	0.3147	0.3436	0.4759	0.5055	0.5325

Manhattan Quantization (Kong et al., 2012)

Quantization Stage

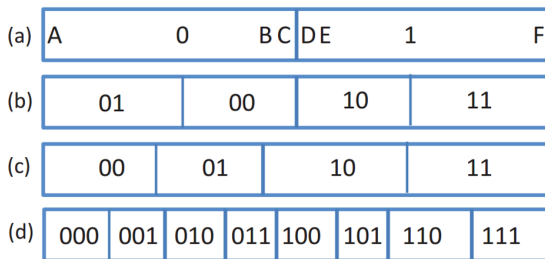
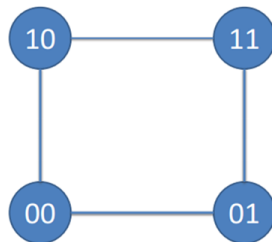


Figure 1: Different quantization methods: (a) single-bit quantization (SBQ); (b) hierarchical quantization (HQ); (c) 2-bit Manhattan quantization (2-MQ); (d) 3-bit Manhattan quantization (3-MQ).

Manhattan Quantization (Kong et al., 2012)

Natural Binary Code(NBC)



(a) Hamming distance



(b) Decimal distance with NBC

Manhattan Quantization (Kong et al., 2012)

Manhattan Distance

Let $\mathbf{x} = [x_1, x_2, \dots, x_d]^T$, $\mathbf{y} = [y_1, y_2, \dots, y_d]^T$, the Manhattan distance between \mathbf{x} and \mathbf{y} is defined as follows:

$$d_m(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^d |x_i - y_i|,$$

where $|x|$ denotes the absolute value of x .

Manhattan Quantization (Kong et al., 2012)

- We divide each projected dimension into 2^q regions and then use q bits of natural binary code to encode the index of each region.
- For example, If $q = 3$, the indices of regions are $\{0, 1, 2, 3, 4, 5, 6, 7\}$ and the natural binary codes are $\{000, 001, 010, 011, 100, 101, 110, 111\}$

Manhattan Quantization (Kong et al., 2012)

- Manhattan quantization(MQ) with q bits is denoted as q -MQ.
- For example, if $q = 2$,

$$\begin{aligned}d_m(000100, 110000) &= d_d(00, 11) + d_d(01, 00) + d_d(00, 00) \\&= 3 + 1 + 0 \\&= 4.\end{aligned}$$

Manhattan Quantization (Kong et al., 2012)

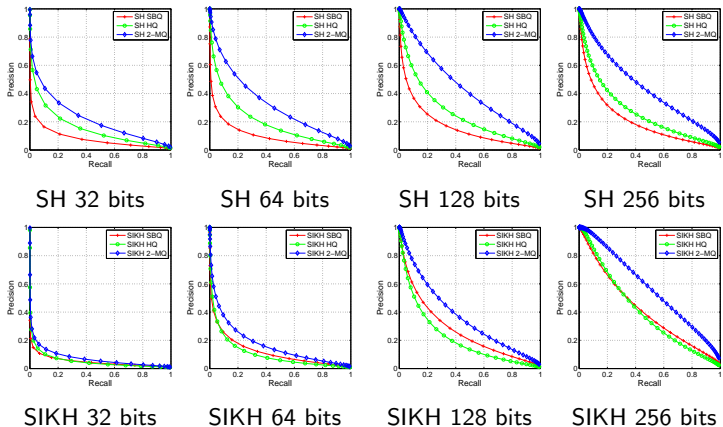


Figure: Precision-recall curve on 22K *LabelMe* data set

Manhattan Quantization (Kong et al., 2012)

Table: mAP on ANN_SIFT1M data set. The best mAP among SBQ, HQ and 2-MQ under the same setting is shown in bold face.

# bits	32			64			96		
	SBQ	HQ	2-MQ	SBQ	HQ	2-MQ	SBQ	HQ	2-MH
ITQ	0.1657	0.2500	0.2750	0.4641	0.4745	0.5087	0.5424	0.5871	0.6263
SIKH	0.0394	0.0217	0.0570	0.2027	0.0822	0.2356	0.2263	0.1664	0.2768
LSH	0.1163	0.0961	0.1173	0.2340	0.2815	0.3111	0.3767	0.4541	0.4599
SH	0.0889	0.2482	0.2771	0.1828	0.3841	0.4576	0.2236	0.4911	0.5929
PCA	0.1087	0.2408	0.2882	0.1671	0.3956	0.4683	0.1625	0.4927	0.5641

Variable Bit Quantization (Moran et al., 2013)

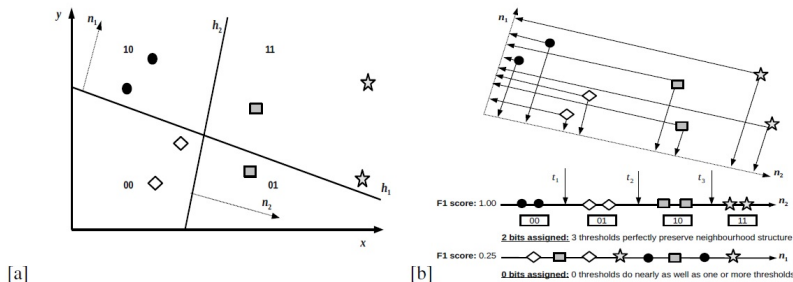


Figure 1: **Left:** Data points with identical shapes are 1-NN. Two hyperplanes h_1, h_2 are shown alongside their associated normal vectors (n_1, n_2). **Right top:** Projection of points onto the normal vectors n_1 and n_2 of the hyperplanes (arrows denote projections). **Right middle:** Positioning of the points along normal vector n_2 . Three quantisation thresholds (t_1, t_2, t_3 , and consequently 2 bits) can maintain the neighbourhood structure. **Right bottom:** the high degree of mixing between the 1-NN means that this hyperplane (h_1) is likely to have 0 bits assigned (and therefore be discarded entirely).

Variable Bit Quantization (Moran et al., 2013)

Dataset	CIFAR-10					TDT-2				Reuters-21578			
	SBQ	MQ	DBQ	NPQ	VBQ	SBQ	MQ	DBQ	VBQ	SBQ	MQ	DBQ	VBQ
SIKH	0.042	0.063	0.047	0.090	0.161	0.034	0.045	0.031	0.092	0.102	0.112	0.087	0.389
LSH	0.119	0.093	0.066	0.153	0.207	0.189	0.097	0.089	0.229	0.276	0.201	0.175	0.538
BLSI	0.038	0.135	0.111	0.155	0.231	0.283	0.210	0.087	0.396	0.100	0.030	0.030	0.156
SH	0.051	0.135	0.111	0.167	0.202	0.146	0.212	0.167	0.370	0.033	0.028	0.030	0.154
PCAH	0.036	0.137	0.107	0.153	0.219	0.281	0.208	0.094	0.374	0.095	0.034	0.027	0.154

Table 1: Area under the Precision Recall curve (AUPRC) for all five projection methods. Results are for 32 bits (images) and at 128 bits (text). The best overall score for each dataset is shown in bold face.

Outline

- 1 Introduction
- 2 Unsupervised Hashing
- 3 Supervised Hashing
- 4 Ranking-based Hashing
- 5 Multimodal Hashing
- 6 Deep Hashing
- 7 Quantization
- 8 Conclusion**
- 9 Reference

Conclusion

- Hashing can significantly **improve searching speed** and **reduce storage cost**.
- Projections with **isotropic variances** will be **better than** those with **anisotropic variances**. (IsoHash)
- **Avoiding pairwise computation via feature transformation** can be used for scalable graph hashing. (SGH)
- **Stochastic learning** or **avoiding pairwise computation** can be used for scalable supervised hashing. (LFH/SCM)
- **Discrete hashing** methods might outperform **continuous relaxation** methods. (COSDISH)
- **Deep hashing** can improve accuracy by integrating feature learning into code learning procedure. (DPSH).
- The **quantization** stage is at least as important as the **projection** stage. (DBQ/MQ)

Q & A

Thanks!



Question?

A Learning to Hash website: <http://cs.nju.edu.cn/lwj/L2H.html>

Some code available at: <http://cs.nju.edu.cn/lwj>

Outline

- 1 Introduction
- 2 Unsupervised Hashing
- 3 Supervised Hashing
- 4 Ranking-based Hashing
- 5 Multimodal Hashing
- 6 Deep Hashing
- 7 Quantization
- 8 Conclusion
- 9 Reference**

- A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.
- A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 327–336, 1998.
- M. Chu. Constructing a Hermitian matrix from its diagonal entries and eigenvalues. *SIAM Journal on Matrix Analysis and Applications*, 16(1): 207–217, 1995.
- M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the ACM Symposium on Computational Geometry*, 2004.
- G. Ding, Y. Guo, and J. Zhou. Collective matrix factorization hashing for multimodal data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2083–2090, 2014.
- A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions  

- via hashing. In *Proceedings of International Conference on Very Large Data Bases*, 1999.
- Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of Computer Vision and Pattern Recognition*, 2011.
- A. Horn. Doubly stochastic matrices and the diagonal of a rotation matrix. *American Journal of Mathematics*, 76(3):620–630, 1954.
- Q.-Y. Jiang and W.-J. Li. Scalable graph hashing with feature transformation. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 2248–2254, 2015.
- W.-C. Kang, W.-J. Li, and Z.-H. Zhou. Column sampling based discrete supervised hashing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, 2016.
- W. Kong and W.-J. Li. Double-bit quantization for hashing. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, 2012a.
- W. Kong and W.-J. Li. Isotropic hashing. In *Proceedings of the 26th*

- Annual Conference on Neural Information Processing Systems (NIPS)*, 2012b.
- W. Kong, W.-J. Li, and M. Guo. Manhattan hashing for large-scale image retrieval. In *The 35th International ACM SIGIR conference on research and development in Information Retrieval (SIGIR)*, 2012.
- B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Proceedings of International Conference on Computer Vision*, 2009.
- B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12):2143–2157, 2009.
- S. Kumar and R. Udupa. Learning hash functions for cross-view similarity search. In *IJCAI*, pages 1360–1365, 2011.
- H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3278, 2015.
- P. Li and A. C. König. Theory and applications of b -bit minwise hashing. *Commun. ACM*, 54(8):101–109, 2011.

- W.-J. Li, S. Wang, and W.-C. Kang. Feature learning based deep supervised hashing with pairwise labels. *arXiv:1511.03855*, 2015.
- G. Lin, C. Shen, D. Suter, and A. van den Hengel. A general two-step approach to learning-based hashing. In *IEEE International Conference on Computer Vision*, pages 2552–2559, 2013.
- G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1971–1978, 2014.
- K. Lin, H. Yang, J. Hsiao, and C. Chen. Deep learning of binary hash codes for fast image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops*, pages 27–35, 2015a.
- Z. Lin, G. Ding, M. Hu, and J. Wang. Semantics-preserving hashing for cross-view retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3864–3872, 2015b.
- V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2475–2483, 2015.

- W. Liu, J. Wang, S. Kumar, and S. Chang. Hashing with graphs. In *Proceedings of International Conference on Machine Learning*, 2011.
- W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.
- W. Liu, C. Mu, S. Kumar, and S. Chang. Discrete graph hashing. In *Advances in Neural Information Processing Systems*, pages 3419–3427, 2014.
- S. Moran, V. Lavrenko, and M. Osborne. Variable bit quantisation for LSH. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2013.
- M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *Proceedings of International Conference on Machine Learning*, 2011.
- M. Norouzi, D. J. Fleet, and R. Salakhutdinov. Hamming distance metric learning. In *Advances in Neural Information Processing Systems*, pages 1070–1078, 2012.
- M. Ou, P. Cui, F. Wang, J. Wang, W. Zhu, and S. Yang. Comparing

- apples to oranges: a scalable solution with heterogeneous hashing. In *KDD*, pages 230–238, 2013.
- M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Proceedings of Neural Information Processing Systems*, 2009.
- R. Salakhutdinov and G. Hinton. Semantic Hashing. In *SIGIR workshop on Information Retrieval and applications of Graphical Models*, 2007.
- R. Salakhutdinov and G. E. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, 2009.
- F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 37–45, 2015.
- A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014.
- G. D. Smith. *Numerical solution of partial differential equations: finite difference methods*. Oxford university press, 1985.

- J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM Multimedia*, pages 423–432, 2011.
- J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *SIGMOD Conference*, pages 785–796, 2013.
- C. Strecha, A. A. Bronstein, M. M. Bronstein, and P. Fua. Ldhash: Improved matching with smaller descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(1):66–78, 2012.
- A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Proceedings of Computer Vision and Pattern Recognition*, 2008.
- J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. In *Proceedings of International Conference on Machine Learning*, 2010a.
- J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale image retrieval. In *Proceedings of Computer Vision and Pattern Recognition*, 2010b.

- J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang. Learning hash codes with listwise supervision. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3032–3039, 2013a.
- J. Wang, J. Wang, N. Yu, and S. Li. Order preserving hashing for approximate nearest neighbor search. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 133–142, 2013b.
- Q. Wang, Z. Zhang, and L. Si. Ranking preserving hashing for fast similarity search. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 3911–3917, 2015.
- Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proceedings of Neural Information Processing Systems*, 2008.
- B. Wu, Q. Yang, W. Zheng, Y. Wang, and J. Wang. Quantized correlation hashing for fast cross-modal search. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 3946–3952, 2015.
- R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the*

- Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2156–2162, 2014.
- D. Zhang and W. Li. Large-scale supervised multimodal hashing with semantic correlation maximization. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, 2014.
- D. Zhang, F. Wang, and L. Si. Composite hashing with multiple information sources. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2011.
- P. Zhang, W. Zhang, W. Li, and M. Guo. Supervised hashing with latent factor models. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2014.
- R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Transactions on Image Processing*, 24(12):4766–4779, 2015.
- F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based

hashing for multi-label image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1556–1564, 2015.

Y. Zhen and D.-Y. Yeung. A probabilistic model for multimodal hash function learning. In *KDD*, pages 940–948, 2012a.

Y. Zhen and D.-Y. Yeung. Co-regularized hashing for multimodal data. In *NIPS*, pages 1385–1393, 2012b.