# Asymmetric Deep Supervised Hashing


Qing-Yuan Jiang & Wu-Jun Li

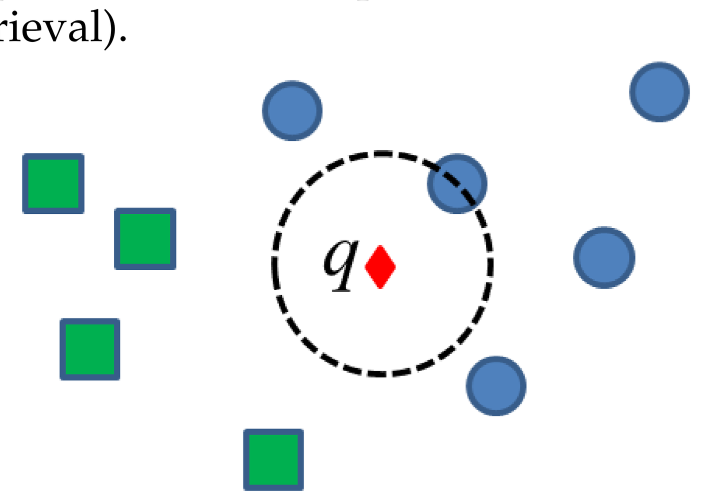
LAMDA Group, Department of Computer Science and Technology, Nanjing University, Nanjing, China.

jiangqy@lamda.nju.edu.cn, liwujun@nju.edu.cn


## Introduction

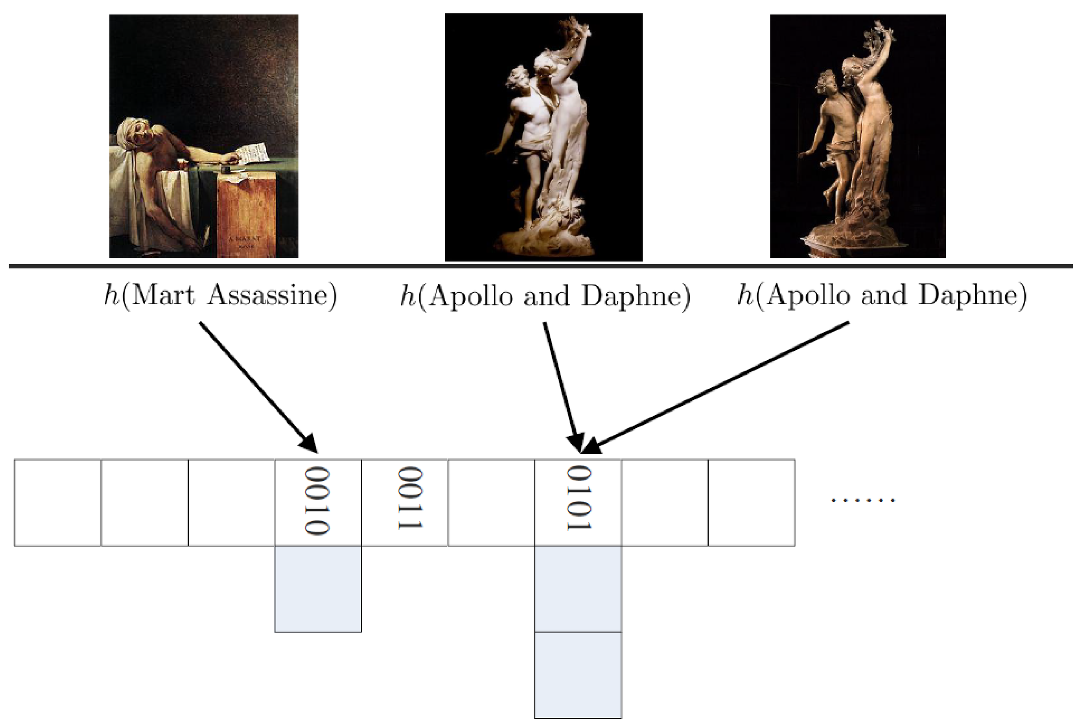
### Nearest Neighbor Search (NNS)

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




- Underlying many machine learning, data mining, information retrieval problems.
- Challenge for NNS in big data applications:
  - Curse of dimensionality
  - Storage cost
  - Search (query) speed

### Similarity Preserving Hashing for NNS




- Reduce dimensionality and storage cost
- Fast search speed
  - By using hash-code to construct an index, we can achieve constant or sub-linear search time complexity.
  - Exhaustive search is also acceptable because the distance calculation cost is cheap with binary representation.

### Motivation

- Deep supervised hashing can achieve better performance.
- Most existing deep hashing methods adopt a symmetric strategy to learn one deep hash function.
- The training of existing symmetric methods is typically time-consuming ($\mathcal{O}(n^2)$ or even $\mathcal{O}(n^3)$).
- It is hard for these methods to effectively utilize the supervised information.

### Contribution

- The first deep hashing method which treats query points and database points in an asymmetric way.
- The training of ADSH is much more efficient than that of symmetric deep hashing methods.
- Achieve state-of-the-art retrieval performance.

## Source Code

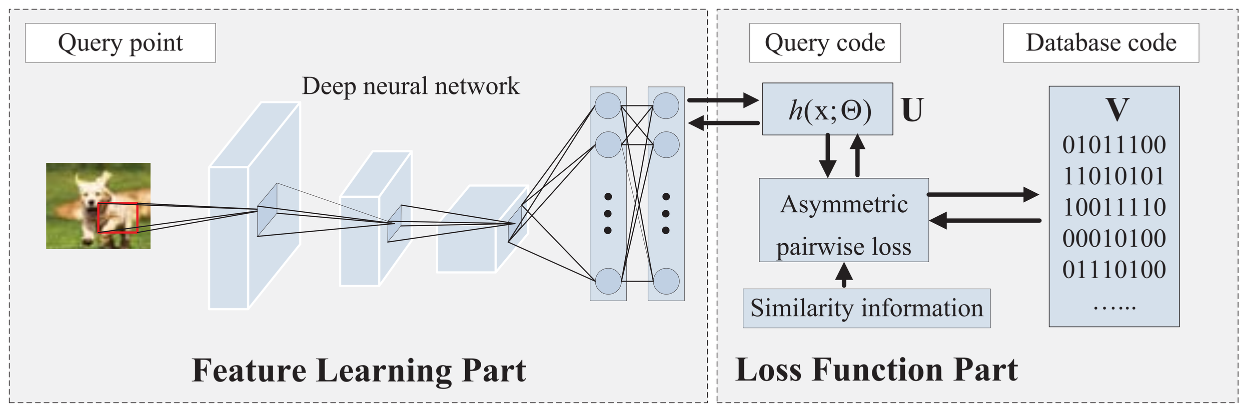
**Source code is available at:**

`https://cs.nju.edu.cn/lwj/.`

## Model

- **Model architecture.**

The end-to-end deep learning architecture for ADSH




- **Feature learning part.**
  - Feature learning part contains a deep neural network model.
  - Pretrained CNN-F model: 5 convolutional layers + 3 fully-connected layers.
  - Learning deep neural network only for query data points.
- **Loss function part.**

$$\min_{\Theta,\mathbf{V}} J(\Theta,\mathbf{V}) = \sum_{i\in\Omega}\sum_{j\in\Gamma}\left[\tanh(F(\mathbf{y}_i;\Theta))^T\mathbf{v}_j - cS_{ij}\right]^2 + \gamma\sum_{i\in\Omega}[\mathbf{v}_i - \tanh(F(\mathbf{y}_i;\Theta))]^2$$

$$\text{s.t.} \quad \mathbf{V}\in\{-1,+1\}^{n\times c},$$

  - Query images set $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^m$.
  - Pairwise labels $\mathbf{S} = \{S_{ij}\}$ with $S_{ij}\in\{-1,1\}$.
  - Binary codes $\mathbf{V} = \{\mathbf{v}_j^T\}_{j=1}^n$ for database set.
  - $\Theta$ is the parameters for deep neural network.
  - The output of deep neural network is defined as $F(\mathbf{y}_i;\Theta)$ for given query point $\mathbf{y}_i$.
  - Relaxed binary codes $\widetilde{\mathbf{U}} = \{\widetilde{\mathbf{u}}_i^T\}_{i=1}^m$ for query set. Here, $\widetilde{\mathbf{u}}_i = \tanh(F(\mathbf{y}_i;\Theta))$ and $\mathbf{z}_i = F(\mathbf{y}_i;\Theta)$.

## Learning

### Alternating Learning Algorithm

- **Learn $\Theta$ with V fixed:** back propagation.
  - update deep neural network parameters by using BP algorithm.

$$\frac{\partial J}{\partial \mathbf{z}_i} = \Big\{2\sum_{j\in\Gamma}[\widetilde{\mathbf{u}}_i^T\mathbf{v}_j - cS_{ij}] + 2\gamma(\widetilde{\mathbf{u}}_i - \mathbf{v}_i)\Big\}\odot(1-\widetilde{\mathbf{u}}_i^2). \quad (1)$$

- **Learn V with $\Theta$ fixed:** discrete cyclic coordinate descent.
  - update binary codes V bit by bit.
  - for each bit $\mathbf{V}_{*k}$:

$$\mathbf{V}_{*k}^* \doteq \operatorname*{argmin}_{\mathbf{V}_{*k}\in\{-1,+1\}^n} J(\mathbf{V}_{*k}) = \|\mathbf{V}\widetilde{\mathbf{U}}^T\|_F^2 + \mathrm{tr}(\mathbf{V}\mathbf{Q}^T) + \text{const}. \quad (2)$$

where $\mathbf{Q} = -2c\mathbf{S}^T\widetilde{\mathbf{U}} - 2\gamma\bar{\mathbf{U}}$. Hence we have:

$$\mathbf{V}_{*k} = -\mathrm{sign}\big(2\widehat{\mathbf{V}}_k\widehat{\mathbf{U}}_k^k\widetilde{\mathbf{U}}_{*k}^T + \mathbf{Q}_{*k}\big).$$

- **Learning algorithm**

**Algorithm 1** The learning algorithm of ADSH.

```
procedure ADSH(Y, S)                                  ▷ Images Y and similarity S.
    Initialize Θ, V, batch size: M and iteration number T_out, T_in.
    for w < T_out do
        Sample query set Y^Ω ⊂ X and similarity S ⊂ S.
        for s = 1, 2, ..., m/M do                        ▷ Update DNN.
            Randomly sample M data points.
            Calculate output F(y_i; Θ) by forward propagation.
            Calculate gradient according to Equation (1).
            Update parameter Θ by using back propagation.
        end for
        for k = 1 → c do                                 ▷ Update binary codes.
            Update V_*k according to Equation (2).
        end for
    end for
    return Θ and V.          ▷ DNN parameters Θ and binary code V.
end procedure
```

## Experiment

### Datasets.

- **MS-COCO**: 82,783 training, 40,504 validation images which belong to 91 categories. 5,000 images (250 images per category from validation set) are adopted as test set.
- **CIFAR-10**: 60,000 32×32 color images which are categorized into 10 classes (6000 images per class). It is a single-label dataset.
- **NUS-WIDE**: 270,000 images which belong to 81 classes. The 21 most frequent classes are adopted for evaluation. For these classes, the number of images of each class is at least 5000.

### Baselines.

- **Unsupervised hashing**: ITQ [CVPR-11].
- **Supervised hashing**: Lin:Lin [NIPS-12], Lin:V [NIPS-12], LFH [SIGIR-14], FastH [CVPR-14], SDH [CVPR-15], COSDISH [AAAI-16], ADGH [AAAI-17].
- **Deep Supervised Hashing**: DPSH [IJCAI-16], DSH [CVPR-16], DHN [AAAI-16].
- Among all baselines, Lin:Lin, Lin:V and ADGH are asymmetric hashing methods.

### MAP on three datasets.

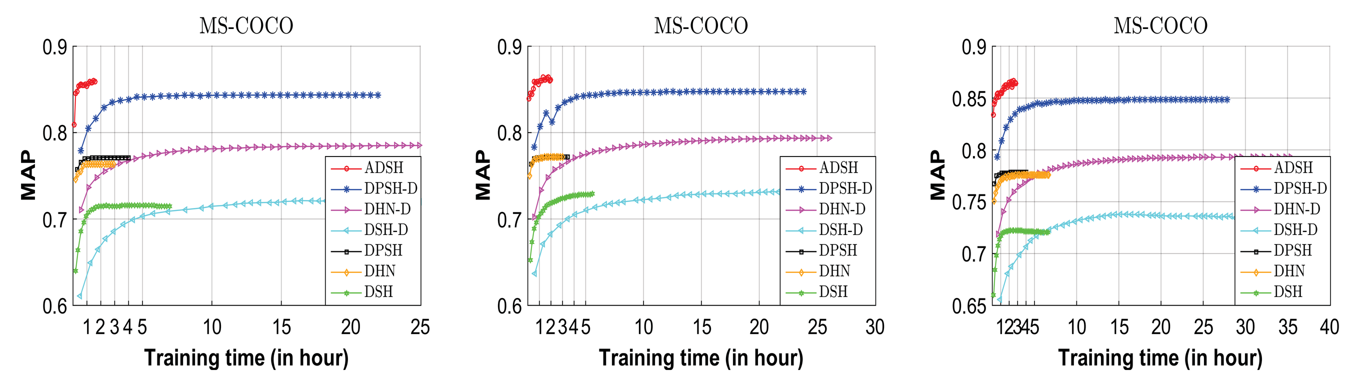
| Method | MS-COCO | | | CIFAR-10 | | | NUS-WIDE | | |
|---|---|---|---|---|---|---|---|---|---|
| | #24 | #32 | #48 | #24 | #32 | #48 | #24 | #32 | #48 |
| ITQ | .6326 | .6308 | .6338 | .2754 | .2861 | .2941 | .7361 | .7457 | .7553 |
| Lin:Lin | .6722 | .6701 | .6736 | .6312 | .6079 | .6013 | .5704 | .5627 | .5555 |
| LFH | .7389 | .7580 | .7725 | .5738 | .6414 | .6927 | .7681 | .7949 | .8135 |
| FastH | .7478 | .7544 | .7604 | .6632 | .6847 | .7020 | .7692 | .7817 | .8037 |
| SDH | .7078 | .7115 | .7164 | .6334 | .6514 | .6603 | .7998 | .8017 | .8124 |
| COSDISH | .6924 | .7312 | .7589 | .6614 | .6802 | .7016 | .7406 | .7843 | .7964 |
| KADGH | N/A | N/A | N/A | .6607 | .6701 | .6829 | N/A | N/A | N/A |
| DPSH | .7667 | .7729 | .7777 | .7204 | .7341 | .7464 | .8249 | .8351 | .8442 |
| DSH | .7176 | .7156 | .7220 | .7421 | .7703 | .7992 | .7313 | .7401 | .7485 |
| DHN | .7656 | .7691 | .7740 | .7213 | .7233 | .7332 | .8013 | .8051 | .8146 |
| ADSH | **.8590** | **.8633** | **.8651** | **.9280** | **.9310** | **.9390** | **.8784** | **.8951** | **.9055** |
| ADSH* | **.8673** | **.8641** | **.8503** | **.9636** | **.9632** | **.9605** | **.9049** | **.9088** | **.9119** |

ADSH* adopts pretrained resnet50 for feature learning.

### MAP with larger training set.

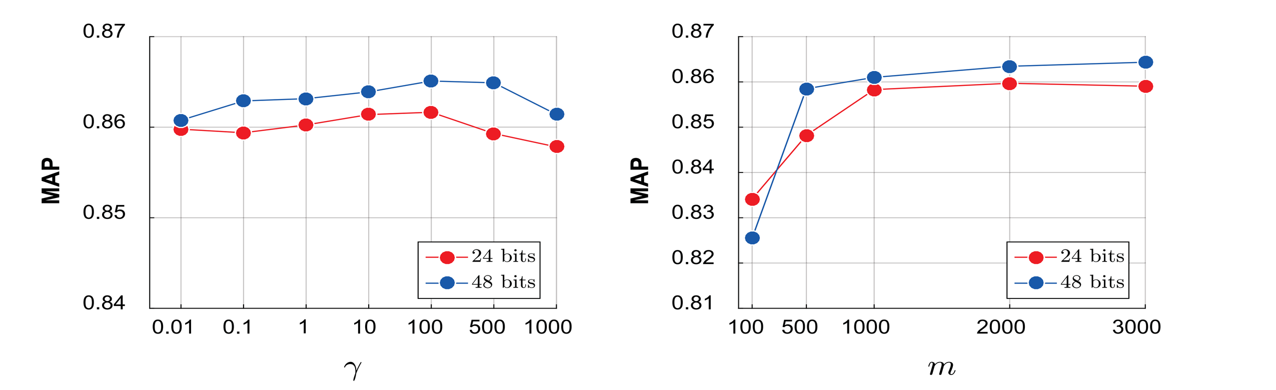
| Method | MS-COCO | | | CIFAR-10 | | | NUS-WIDE | | |
|---|---|---|---|---|---|---|---|---|---|
| | #24 | #32 | #48 | #24 | #32 | #48 | #24 | #32 | #48 |
| Lin:V | .7803 | .7812 | .7842 | .8160 | .8038 | .7993 | .7565 | .7615 | .7605 |
| LFH-D | .7729 | .8063 | .8165 | .7267 | .7712 | .8333 | .8351 | .8604 | .8790 |
| SDH-D | .7724 | .7708 | .7711 | .8466 | .8501 | .8501 | **.8808** | .8815 | .8756 |
| COSDISH-D | .7685 | .8052 | .7943 | .8700 | .8802 | .8771 | .8546 | .8636 | .8752 |
| KADGH-D | N/A | N/A | N/A | .8710 | .8759 | .8749 | N/A | N/A | N/A |
| ADSH | **.8590** | **.8633** | **.8651** | **.9280** | **.9310** | **.9390** | .8784 | **.8951** | **.9055** |
| ADSH* | **.8673** | **.8641** | **.8503** | **.9636** | **.9632** | **.9605** | **.9049** | **.9088** | **.9119** |

### Training time on MS-COCO dataset.




(a). 24bits (b). 32bits (c). 48bits

### Hyper-parameters on MS-COCO dataset.




(a). $\gamma$ (b). # sampled query points $m$