# Relaxation-Free Deep Hashing via Policy Gradient

Xin Yuan, Liangliang Ren, Jiwen Lu *, and Jie Zhou

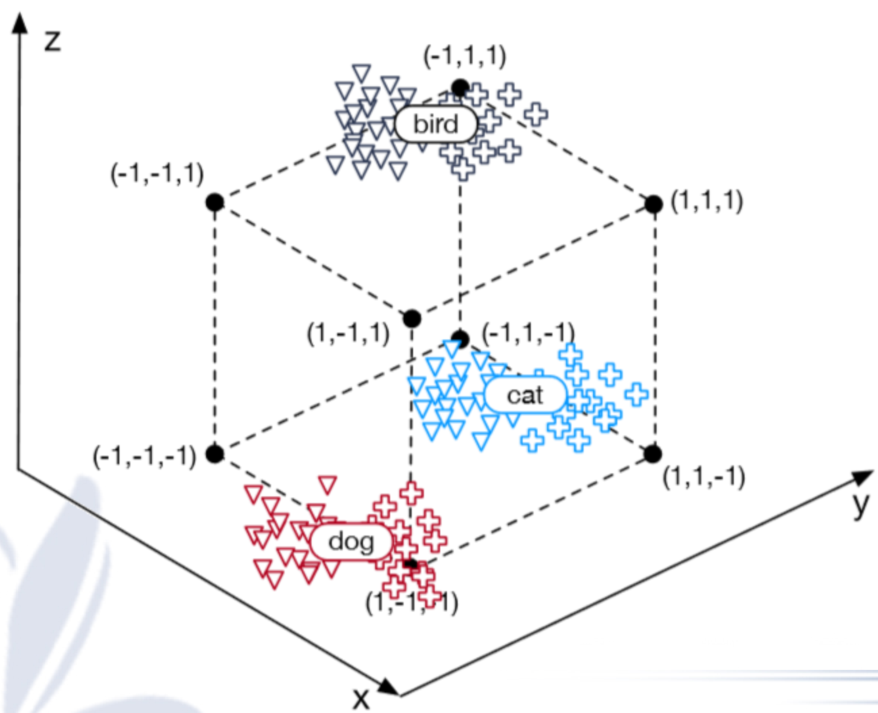
Department of Automation, Tsinghua University
yuanx16@mails.tsinghua.edu.cn; renll16@mails.tsinghua.edu.cn;
lujiwen@tsinghua.edu.cn; jzhou@tsinghua.edu.cn

ECCV 2018

Made by Godder
2019/10/25

# introduction

- Hash evaluate
  - Hamming distance



# introduction

- Hash evaluate
  - Hamming distance
  - cosine

# introduction

- European space to Hamming distance.
  - Sign function

$$sign(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0. \end{cases}$$

# introduction

- European space to Hamming distance.
  - Sign function
  - relaxation

# introduction

- European space to Hamming distance.
  - Sign function
  - relaxation
  - Probability

# Related work

- Unsupervised
  - SH: graph partition
  - AGH: neighborhoods by tractable graph
  - DH: neural network & non-linear function
  - ITQ: iterative manner to learn a rotation matrix
  - MH: cluster center & dimensionally reduction
  - DGH: optimization method for discrete code space.

# Related work

- Machine learning Supervised
  - KSH: inner product for hamming distance
  - FSH: decision tree.
  - SDH: auxiliary variable & kernel based.
  - SEDH: learn multi-layer function by label information

# Related work

- Deep learning Supervised
  - CNNH: two stages strategy.
    <1> learn hash code
    <2> learn network by hash code
  - DNNH: CNNH with simultaneous feature learning.
  - DSH: DNNH + max-margin loss + quantization loss
  - HashNet: non-smooth sign activation.

# Related work

- Deep learning hash code generation

  discrete hash code is non-differentiable

# Related work

- Deep learning hash code generation

    discrete hash code is non-differentiable

  - Continuous relaxation by a quantization function

# Related work

- Deep learning hash code generation

  discrete hash code is non-differentiable

  - Continuous relaxation by a quantization function

  quantization loss is np-hard, the loss is disconvergent

# Related work

- Deep learning hash code generation

  discrete hash code is non-differentiable

  - Continuous relaxation by a quantization function
  - Replace sign function with sigmoid or tanh function

# Related work

- Deep learning hash code generation

  discrete hash code is non-differentiable

  - Continuous relaxation by a quantization function
  - Replace sign function with sigmoid or tanh function

  Model learning difficult, relaxation becomes more non-smooth

# PGDH

- Hash representation

$$p(a_{i,k}) = \begin{cases} \pi_{\boldsymbol{x}_i,\theta}^{(k)}, & \text{if } a_{i,k} = 1 \\ 1 - \pi_{\boldsymbol{x}_i,\theta}^{(k)}, & \text{if } a_{i,k} = 0 \end{cases}$$

# PGDH

- Hash representation to discrete hash codes

$$b_q^k = \begin{cases} +1, & \text{if } \pi_{\boldsymbol{x}_q,\theta}^{(k)} > 0.5 \\ -1, & \text{otherwise} \end{cases}$$

$$b_q^k = \begin{cases} +1, & \text{with probability} \quad \pi_{\boldsymbol{x}_q,\theta}^{(k)} \\ -1, & \text{with probability} \quad 1 - \pi_{\boldsymbol{x}_q,\theta}^{(k)} \end{cases}$$

# PGDH

- Hash constraint

$$
\begin{aligned}
\boldsymbol{b}_i &= 2 * (\boldsymbol{a}_i - 0.5) \\
r(\boldsymbol{a}_i) &= -\frac{1}{2}\sum_{j=1}^{n} \hat{s}_{ij}(K - \boldsymbol{b}_i^T \hat{\boldsymbol{b}}_j) \\
& \quad s.t. \quad \boldsymbol{b}_i, \hat{\boldsymbol{b}}_j \in \{-1, +1\}^K \qquad (4) \\
\hat{s}_{ij} &= \begin{cases} \beta, & \text{if } s_{ij} = 1 \\ \beta - 1, & \text{otherwise} \end{cases}
\end{aligned}
$$

# PGDH

- Objective function

$$\mathcal{L}(\theta) = -\sum_i \mathbb{E}_{\boldsymbol{a}_i \sim P_\theta(\boldsymbol{x}_i)}[r(\boldsymbol{a}_i)]$$

# PGDH

- Policy gradient

$$\nabla_\theta \mathcal{L}(\theta) = -\sum_i \mathbb{E}_{\boldsymbol{a}_i \in \mathcal{A}_i}[r(\boldsymbol{a}_i)\nabla_\theta \log(P_\theta(\boldsymbol{a}_i|\boldsymbol{x}_i))]$$

where $A_i$ is the set of all possible actions for i-th input data in the minibatch

$T$-samples Monte Carlo on $\boldsymbol{a}_i$

$$\mathcal{A}_i = \{\boldsymbol{a}_i^1, \boldsymbol{a}_i^2, ..., \boldsymbol{a}_i^T\} = MC^{P_\theta(\boldsymbol{a}_i|\boldsymbol{x}_i)}(T)$$

$$\nabla_\theta \mathcal{L}(\theta) \approx -\frac{1}{T}\sum_i \sum_t [r(\boldsymbol{a}_i^t)\nabla_\theta \log(P_\theta(\boldsymbol{a}_i^t|\boldsymbol{x}_i))]$$

# PGDH

- REINFORCE

$$\nabla_\theta \mathcal{L}(\theta) \approx -\frac{1}{T}\sum_i \sum_t [(r(\boldsymbol{a}_i^t) - r')\nabla_\theta \log(P_\theta(\boldsymbol{a}_i^t|\boldsymbol{x}_i))] \tag{10}$$

r‘ the average of all rewards in each mini-batch.

$$\sum_i \mathbb{E}_{\boldsymbol{a}_i \in \mathcal{A}_i}[r'\nabla_\theta \log(P_\theta(\boldsymbol{a}_i^t|\boldsymbol{x}_i))] = \sum_i r'\nabla_\theta \sum_{\boldsymbol{a}_i} P_\theta(\boldsymbol{a}_i^t|\boldsymbol{x}_i) = \sum_i r'\nabla_\theta 1 = 0$$

$$\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}(\theta) \tag{11}$$

# PGDH

**Algorithm 1:** PGDH

**Input:** Training set: $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^n$, pairwise labels: $\mathcal{S} = \{s_{ij}\}$ and codebook update interval $R > 1$.

**Output:** Learning model $\theta$ and codebook $\boldsymbol{C}$

Initialize $p_\theta$ and $\boldsymbol{C}$;
**for** $iter = 1, 2, \ldots, M$ **do**
    Sample random minibatch from $\boldsymbol{X}$;
    Compute the action probability by feeding minibatch to the model;
    Compute the rewards for MC samples of the minibatch according to Eq. (4)
    Compute policy gradient according to Eq. (10);
    Update the model $\theta$ according to Eq. (11);
    **if** iter % $R = 0$ **then**
        Update codebook $\boldsymbol{C}$;
    **end if**
**end for**
**return** model $\theta$ and codebook $\boldsymbol{C}$;

# Experiments

- Dataset

| | CIFAR10 | NUSWIDE | ImageNet |
|---|---|---|---|
| category number | 10 | 21 | 100 |
| Train set | 500×10 | 500×21 | 100×100 |
| Query set | 100×10 | 100×21 | Validation set |
| Retrieval set | All - query | All - query | 100 categories |

# Experiments

- Evaluate
  - MAP
  - RP
  - HLP@2 (Hamming lookup precision within radius r=2)

# Experiments

- Hash code generation method

| Training Epochs | 1 | 5 | 10 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Deterministic | 24.51 | 47.18 | 66.56 | 72.13 | 74.73 | 74.78 | 74.77 | 75.17 | 75.50 | 75.54 | √ |
| Stochastic | 10.10 | 18.18 | 58.32 | 73.54 | 74.18 | 74.93 | 75.12 | 75.18 | 74.90 | 75.21 | |

# Experiments

- Setting
  - Hyper-parameter

| $\beta$ | **0.1** | **0.2** | **0.3** | **0.4** | **0.5** | **0.6** | **0.7** | **0.8** | **0.9** |
|---|---|---|---|---|---|---|---|---|---|
| CIFAR-10 | 10.12 | 18.38 | 20.08 | 49.43 | 73.65 | 70.32 | **75.23** | 75.12 | 34.12 |
| NUS-WIDE | 31.32 | 43.65 | 54.13 | 66.12 | 77.95 | 76.12 | 77.32 | **79.18** | 78.80 |
| ImageNet | 1.14 | 1.14 | 33.12 | 43.64 | 69.65 | 68.69 | **70.32** | 70.11 | 70.03 |

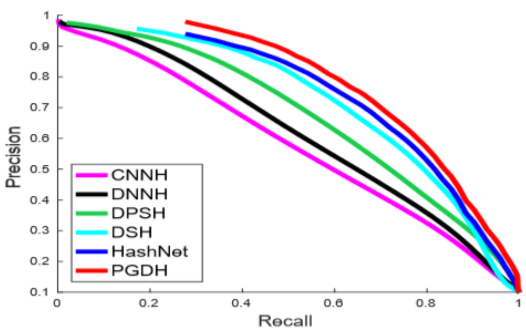
R = 5, Monte Carlo samples T = 10

- Network:  AlexNet pretrained on ImageNet.
- Optimizer: Adam with lr=0.005
- Batch-size: 128

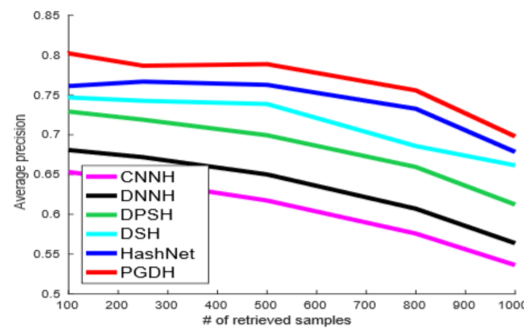
# Experiments

| Methods | CIFAR-10 (%) | | | | NUS-WIDE (%) | | | | ImageNet (%) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 16 | 32 | 48 | 64 | 16 | 32 | 48 | 64 | 16 | 32 | 48 | 64 |
| LSH [8] | 12.9 | 15.2 | 16.9 | 17.8 | 40.3 | 49.2 | 49.3 | 55.1 | 10.1 | 23.5 | 30.1 | 34.9 |
| SH [40] | 12.2 | 13.5 | 12.1 | 12.6 | 47.9 | 49.1 | 49.8 | 51.5 | 20.8 | 32.7 | 39.5 | 42.0 |
| ITQ [9] | 21.3 | 23.4 | 23.8 | 25.3 | 56.7 | 60.3 | 62.2 | 62.6 | 32.5 | 46.2 | 51.3 | 55.6 |
| CCA-ITQ [9] | 31.4 | 36.1 | 36.6 | 37.9 | 50.9 | 54.4 | 56.8 | 67.6 | 26.6 | 43.6 | 54.8 | 58.0 |
| KSH [25] | 35.6 | 40.8 | 53.1 | 44.1 | 40.6 | 40.8 | 38.7 | 39.8 | 16.0 | 28.8 | 34.2 | 39.4 |
| FastH [19] | 45.3 | 46.1 | 48.7 | 50.3 | 51.9 | 61.0 | 64.7 | 65.2 | 22.8 | 44.7 | 51.7 | 55.6 |
| SDH [30] | 40.2 | 42.0 | 44.9 | 45.6 | 53.4 | 61.8 | 63.1 | 64.5 | 29.9 | 45.1 | 54.9 | 59.3 |
| CNNH [42] | 48.8 | 51.2 | 53.4 | 53.6 | 61.2 | 62.3 | 62.1 | 63.7 | 28.8 | 44.7 | 52.8 | 55.6 |
| DNNH [17] | 55.5 | 55.8 | 58.1 | 62.3 | 68.1 | 71.3 | 71.8 | 72.0 | 29.7 | 46.3 | 54.0 | 56.6 |
| DPSH [18] | 64.6 | 66.1 | 67.7 | 68.6 | 71.5 | 72.6 | 73.8 | 75.3 | 32.6 | 54.6 | 61.7 | 65.4 |
| DSH [22] | 68.9 | 69.1 | 70.3 | 71.6 | 71.8 | 72.3 | 74.2 | 75.6 | 34.8 | 55.0 | 62.9 | 66.5 |
| HashNet [2] | 70.3 | 71.1 | 71.6 | 73.9 | 73.3 | 75.2 | 76.2 | 77.6 | 50.6 | 62.9 | 66.3 | 68.4 |
| **PGDH** | **73.6** | **74.1** | **74.7** | **76.2** | **76.1** | **78.0** | **78.6** | **79.2** | **51.8** | **65.3** | **70.7** | **71.6** |

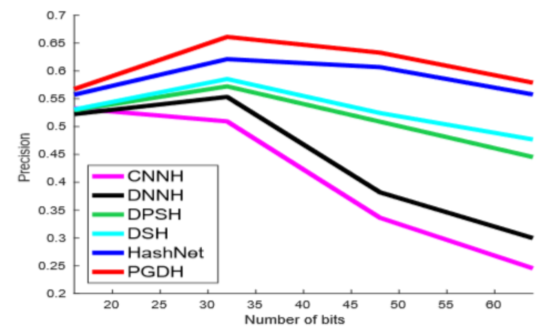
# Experiments



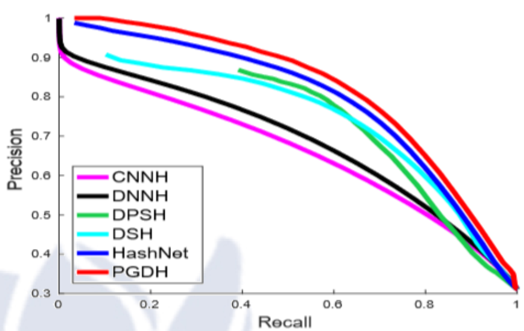

(a) P-R curve at 64 bits



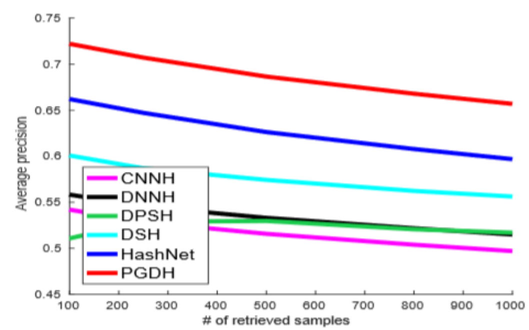

(b) P@N at 64 bits



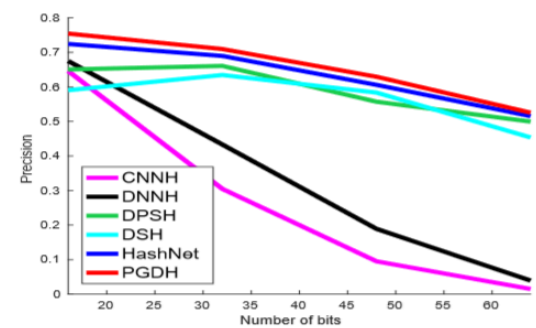

(c) HLP@2 at 64 bits




(a) P-R curve at 64 bits




(b) P@N at 64 bits




(c) HLP@2 at 64 bits

# Reference


- Liu, W., Wang, J., Kumar, S., Chang, S.: Hashing with graphs. In: ICML (2011)
- Liong, V.E., Lu, J., Wang, G., Moulin, P., Zhou, J.: Deep hashing for compact binary codes learning. In: CVPR. pp. 2475–2483 (2015). https://doi.org/10.1109/CVPR.2015.7298862
- Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. TPAMI 35(12), 2916–2929 (2013). https://doi.org/10.1109/TPAMI.2012.193
- Etc.