Transfer Adversarial Hashing for Hamming Space Retrieval

Zhangjie Cao\textsuperscript{1}, Mingsheng Long\textsuperscript{1}, Chao Huang\textsuperscript{1}, and Jianmin Wang\textsuperscript{1}

\textsuperscript{1}KLiss, MOE; NEL-BDS; TNList; School of Software, Tsinghua University, China

The Thirty-second AAAI Conference on Artificial Intelligence, 2018
Image Retrieval

- Nearest Neighbor (NN) similarity retrieval
  - Database: $\mathcal{X} = \{x_1, \ldots, x_N\}$ and Query: $q$
  - NN: $NN(q) = \min_{x \in \mathcal{X}} d(x, q)$

Figure: Image Retrieval: Similarity Retrieval in Hamming Space.
Hashing Methods

Superiorities

Memory
- 128-d float : 512 bytes → 16 bytes
- 1 billion items : 512 GB → 16 GB

Time
- Computation: x10 - x100 faster
- Transmission (disk / web): x30 faster

Applications

- Approximate nearest neighbor search
- Compact representation, Feature Compression for large datasets
- Distribute and transmit data online
- Construct index for large-scale database
Traditional VS. Transfer

- Traditional Image Retrieval
  - Image domain with known similarity relationship

- Transfer Image Retrieval
  - Image domain with unknown similarity relationship
Challenges

- The hash model trained on the source domain cannot work well on the target domain due to the large distribution gap;
- The domain gap makes it difficult to concentrate the database points to be within a small Hamming ball.

Figure: Concentration Problem in Transfer Hamming Space Retrieval
Network Architecture
Logarithm Maximum a Posteriori estimation

Given the set of pairwise similarity labels $S = \{s_{ij}\}$, the logarithm Maximum a Posteriori (MAP) estimation of training hash codes $H^x = [h^x_1, \ldots, h^x_n]$ can be defined as

$$
\log p (H^x | S) \propto \log p (S | H^x) \ p (H^x) \\
= \sum_{s_{ij} \in S} \log p (s_{ij} | h^x_i, h^x_j) \ p (h^x_i) \ p (h^x_j), \tag{1}
$$

where $p(S | H^x)$ is likelihood function, and $p(H^x)$ is prior distribution.
Hash Function Learning

Conditional Probability

For each pair of points \( x_i \) and \( x_j \), \( p(s_{ij} \mid h_i^x, h_j^x) \) is the conditional probability of their relationship \( s_{ij} \) given their hash codes \( h_i^x \) and \( h_j^x \), which can be defined using the pairwise logistic function,

\[
p(s_{ij} \mid h_i^x, h_j^x) = \begin{cases} 
\sigma \left( \text{sim} \left( h_i^x, h_j^x \right) \right), & s_{ij} = 1 \\
1 - \sigma \left( \text{sim} \left( h_i^x, h_j^x \right) \right), & s_{ij} = 0 
\end{cases}
\]

\[
= \sigma \left( \text{sim} \left( h_i^x, h_j^x \right) \right)^{s_{ij}} \left( 1 - \sigma \left( \text{sim} \left( h_i^x, h_j^x \right) \right) \right)^{1-s_{ij}},
\]

where \( \text{sim} \left( h_i^x, h_j^x \right) \) is the similarity function of code pairs \( h_i^x \) and \( h_j^x \) and \( \sigma (x) \) is the probability function.
Hash Function Learning

Similarity Function and Probability Function

Previous methods [12, 2] usually use inner product $\langle h^x_i, h^x_j \rangle$ as similarity function and $\sigma(x) = 1/(1 + e^{-\alpha x})$ as probability function. However, they cannot force the Hamming distance between codes of similar data to be smaller than 2 since the probability cannot discriminate Hamming distances smaller than $b/2$ sufficiently.

Thus, we propose a new similarity function $\text{sim}(h^x_i, h^x_j) = \frac{b}{1 + \|h^x_i - h^x_j\|^2}$ and the corresponding probability function is defined as $\sigma(x) = \tanh(\alpha x)$.
**Hash Function Learning**

**Prior**

Similar to previous work [11, 6, 12], defining that $h_i^x = \text{sgn}(z_i^x)$ where $z_i^x$ is the activation of hash layer, we relax binary codes to continuous codes since discrete optimization of Equation (1) with binary constraints is difficult and adopt a quantization loss function to control quantization error. Specifically, we adopt the prior for quantization of [12] as

$$p(z_i^x) = \frac{1}{2\varepsilon} \exp \left( -\frac{|z_i^x| - 1}{\varepsilon} \right) \quad (3)$$

where $\varepsilon$ is the parameter of the exponential distribution.
Hash Function Learning

Optimization Problem

By substituting Equations (2) and (3) into the MAP estimation in Equation (1), we achieve the optimization problem,

\[
\min_{\theta} J = L + \lambda Q, \tag{4}
\]

where \(\lambda\) is the trade-off parameter and \(\theta\) is network parameters.

\[
L = \sum_{s_{ij} \in S} \log \left( 1 + \exp \left( \frac{b}{1 + \|z^x_i - z^x_j\|_2} \right) \right) - s_{ij} \frac{b}{1 + \|z^x_i - z^x_j\|_2} \tag{5}
\]

\[
Q = \sum_{s_{ij} \in S} \sum_{t=1}^{b} \left( - \log \cosh \left( |z^x_{it}| - 1 \right) - \log \cosh \left( |z^x_{jt}| - 1 \right) \right) \tag{6}
\]
Domain adversarial networks have been successfully applied to transfer learning [3, 9] by extracting features that can reduce the distribution shift between the source and the target domain. We reduce the distribution shifts between the source and the target domain by adversarial learning. The adversarial learning procedure is a two-player game, where the first player is the domain discriminator $G_d$ trained to distinguish the source domain from the target domain, and the second is the base hashing network $G_f$ fine-tuned simultaneously to confuse the domain discriminator.
Domain Distribution Alignment with Adversarial Network

To extract domain-invariant hash codes $h$, the parameters $\theta_f$ of deep hashing network $G_f$ are learned by maximizing the loss of domain discriminator $G_d$, while the parameters $\theta_d$ of domain discriminator $G_d$ are learned by minimizing the loss of the domain discriminator. The objective of domain adversarial network is the functional:

$$D(\theta_f, \theta_y, \theta_d) = \frac{1}{n + m} \sum_{v_i \in X \cup Y} L_d(G_d(G_f(v_i)), d_i),$$

where $L_d$ is the cross-entropy loss and $d_i$ is the domain label of data point $v_i$. $d_i = 1$ means $v_i$ belongs to target domain and $d_i = 0$ means $v_i$ belongs to source domain.
Transfer Adversarial Hashing

Unified optimization problem

The overall loss by integrating Equations (4) and (7),

\[ C = J - \mu D, \quad (8) \]

where \( \mu \) is a trade-off parameter between the MAP loss \( J \) and adversarial learning loss \( D \). The optimization of this loss is as follows. After training convergence, the parameters \( \hat{\theta}_f, \hat{\theta}_y, \hat{\theta}_d \) will deliver a saddle point of the functional (8):

\[
(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f,\theta_y} C(\theta_f, \theta_y, \theta_d), \\
(\hat{\theta}_d) = \arg\max_{\theta_d} C(\theta_f, \theta_y, \theta_d). \quad (9)
\]
Datasets: ImageNet, NUS-WIDE and MS-COCO

Protocols: Mean Average Precision (MAP), Precision-Recall curves and Precision all within Hamming radius 2

Parameter selection: cross-validation by jointly assessing

Methods to compare with: unsupervised methods LSH [4], SH [10], ITQ [5], supervised shallow methods KSH [7], SDH [8], supervised deep single domain methods CNNH [11], DNNH [6], DHN [12], HashNet [2] and supervised deep cross-domain method THN [1].
### Results and Discussion

**Table:** Mean Average Precision (MAP) of Hamming Ranking within Hamming Radius 2 for Different Number of Bits on the Three Image Retrieval Tasks

<table>
<thead>
<tr>
<th>Method</th>
<th>NUS-WIDE</th>
<th>VisDA2017</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16 bits</td>
<td>32 bits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAH</td>
<td>0.722</td>
<td>0.729</td>
</tr>
<tr>
<td>THN</td>
<td>0.671</td>
<td>0.676</td>
</tr>
<tr>
<td>HashNet</td>
<td>0.709</td>
<td>0.693</td>
</tr>
<tr>
<td>DHN</td>
<td>0.669</td>
<td>0.672</td>
</tr>
<tr>
<td>DNNH</td>
<td>0.568</td>
<td>0.622</td>
</tr>
<tr>
<td>CNNH</td>
<td>0.542</td>
<td>0.601</td>
</tr>
<tr>
<td>SDH</td>
<td>0.555</td>
<td>0.571</td>
</tr>
<tr>
<td>ITQ</td>
<td>0.498</td>
<td>0.549</td>
</tr>
<tr>
<td>SH</td>
<td>0.496</td>
<td>0.543</td>
</tr>
<tr>
<td>KSH</td>
<td>0.531</td>
<td>0.554</td>
</tr>
<tr>
<td>LSH</td>
<td>0.432</td>
<td>0.453</td>
</tr>
</tbody>
</table>
Results and Discussion

Figure: The Precision-recall curve @ 64 bits and the Precision within Hamming radius 2 of TAH and comparison methods on three tasks.
Empirical Analysis

**TAH-t** is the variant which uses the pairwise cross-entropy loss introduced in DHN [12] instead of our pairwise t-distribution cross-entropy loss. **TAH-A** is the variant removing adversarial learning module and trained without using the unsupervised training data.

**Table:** MAP within Hamming Radius 2 of TAH variants

<table>
<thead>
<tr>
<th>Method</th>
<th>synthetic $\rightarrow$ real</th>
<th>real $\rightarrow$ synthetic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16 bits 32 bits 48 bits 64 bits</td>
<td>16 bits 32 bits 48 bits 64 bits</td>
</tr>
<tr>
<td>TAH-t</td>
<td>0.443 0.405 0.390 0.364</td>
<td>0.660 0.671 0.717 0.624</td>
</tr>
<tr>
<td>TAH-A</td>
<td>0.305 0.395 0.382 0.331</td>
<td>0.605 0.683 0.725 0.724</td>
</tr>
<tr>
<td>TAH</td>
<td>0.465 0.423 0.433 0.404</td>
<td>0.672 0.695 0.784 0.761</td>
</tr>
</tbody>
</table>
Empirical Analysis

Key Observations

- TAH outperforms TAH-t by very large margins of 0.031 / 0.060 in average MAP, which confirms that the pairwise $t$ cross-entropy loss learns codes within Hamming Radius 2 better than pairwise cross-entropy loss.

- TAH outperforms TAH-A by 0.078 / 0.044 in average MAP for transfer retrieval tasks synthetic $\rightarrow$ real and real $\rightarrow$ synthetic. This convinces that TAH can further exploit the unsupervised train data of target domain to bridge the Hamming spaces of training dataset (real/synthetic) and database (synthetic/real) and transfer knowledge from training set to database effectively.
We formally define a new transfer hashing problem for image retrieval.

We propose a novel transfer adversarial hashing approach based on a hybrid deep architecture.

We align different domains in Hamming space and concentrate the hash codes to be within a small Hamming ball by Maximum a Posteriori estimation with carefully designed similarity function and probability function and adversarial learning.
Z. Cao, M. Long, J. Wang, and Q. Yang. 
Transitive hashing network for heterogeneous multimedia retrieval. 

Z. Cao, M. Long, J. Wang, and P. S. Yu. 
Hashnet: Deep learning to hash by continuation. 

Y. Ganin and V. Lempitsky. 
Unsupervised domain adaptation by backpropagation. 
In *ICML*, 2015.

A. Gionis, P. Indyk, R. Motwani, et al. 
Similarity search in high dimensions via hashing. 

Y. Gong and S. Lazebnik. 
Iterative quantization: A procrustean approach to learning binary codes. 


