Transfer Adversarial Hashing for Hamming Space Retrieval

Zhangjie Cao[1], Mingsheng Long[1], Chao Huang[1], and Jianmin Wang[1]

[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: $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and Query: $\mathbf{q}$
  - NN: $\mathrm{NN}(\mathbf{q}) = \min_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x}, \mathbf{q})$

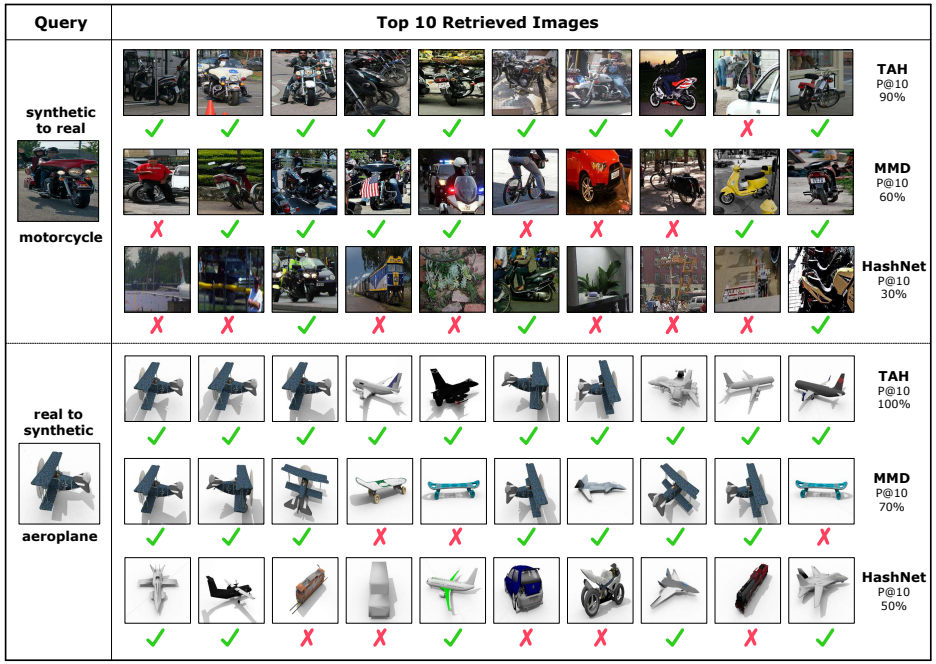
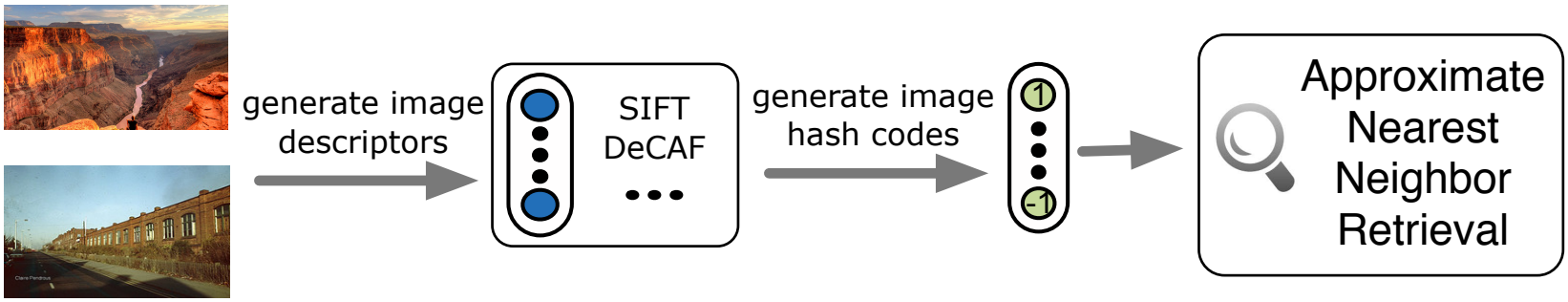


Figure: Image Retrieval: Similarity Retrieval in Hamming Space.

Hashing Methods



Superiorities

Memory

- 128-d float : 512 bytes $\rightarrow$ 16 bytes
- 1 billion items : 512 GB $\rightarrow$ 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

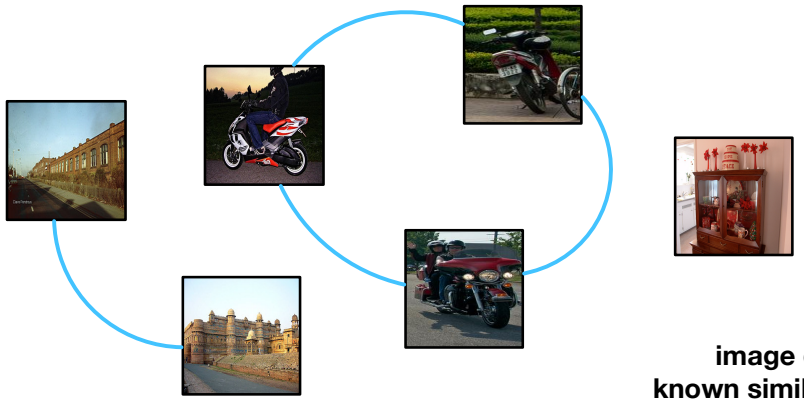


image domain with
known similarity relationship

traditional image retrieval

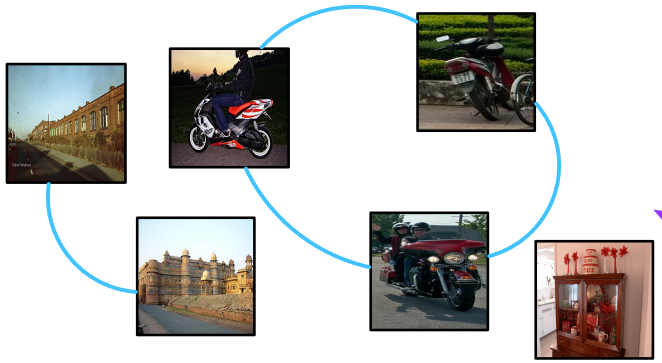


image domain with
known similarity relationship

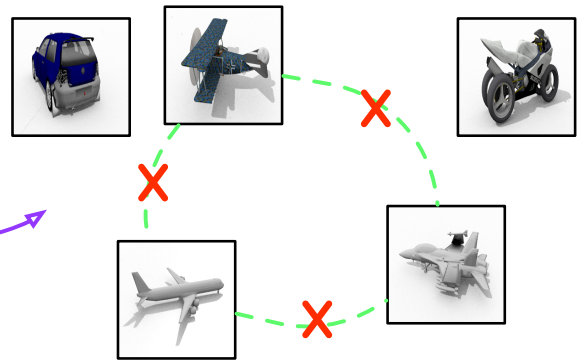


image domain with
unknown similarity relationship

transfer image retrieval

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.

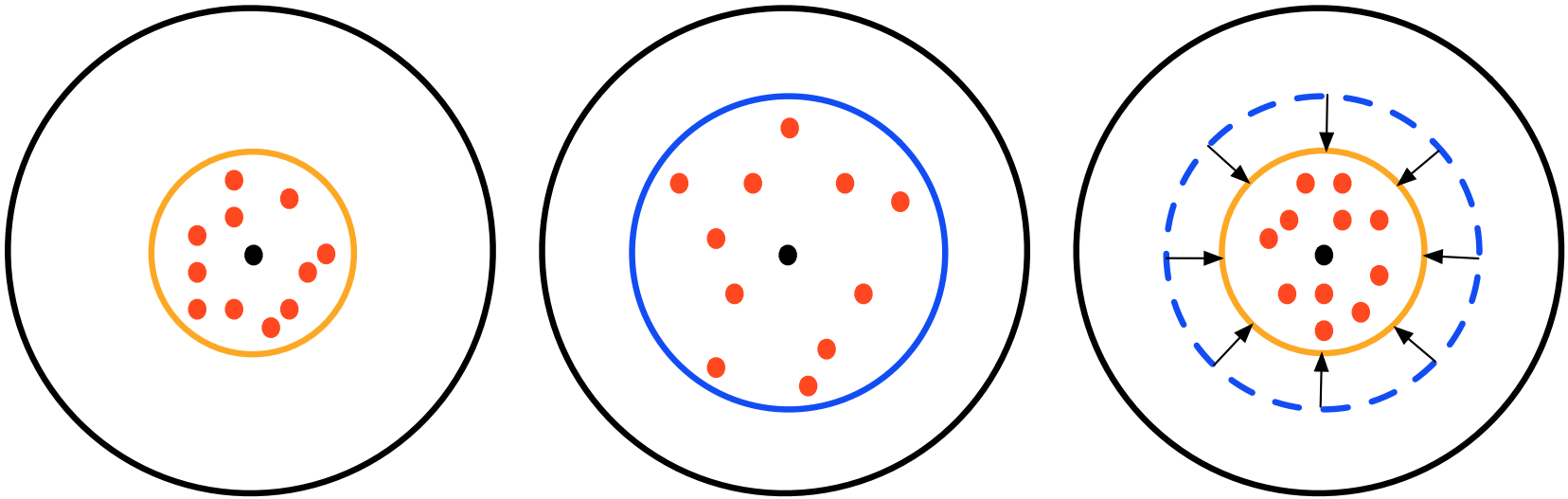
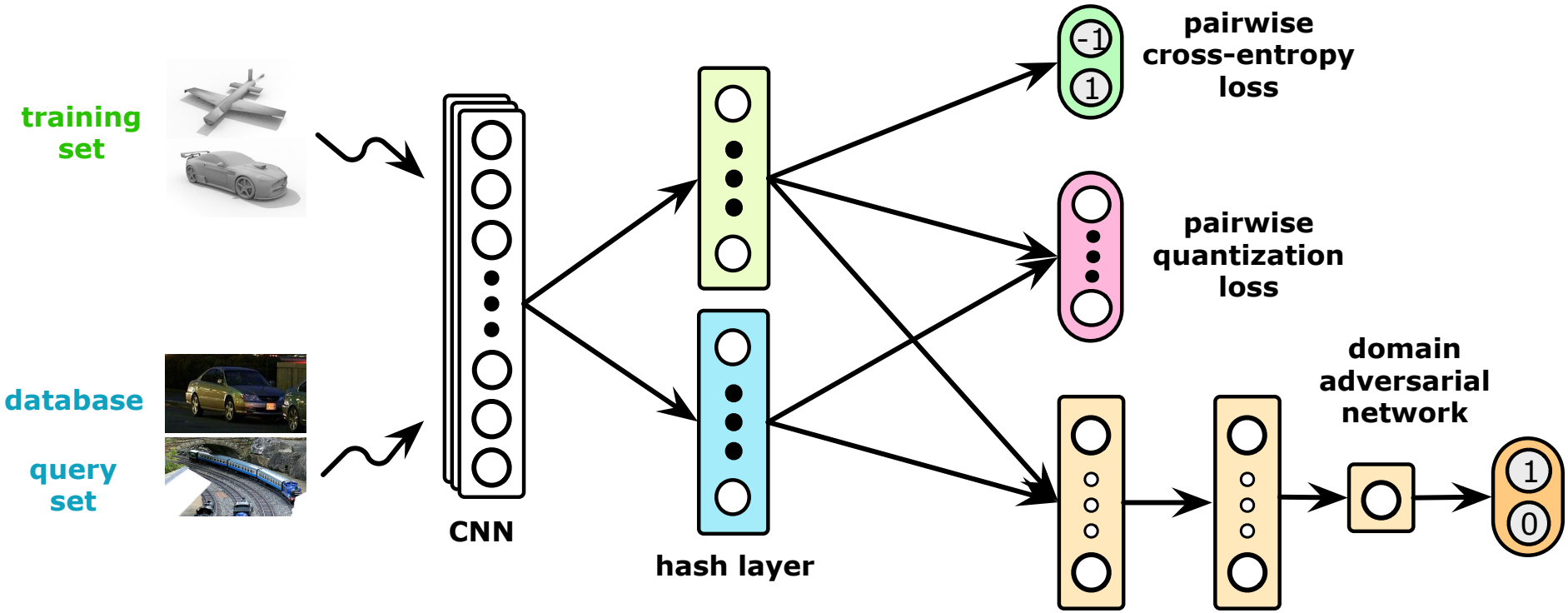
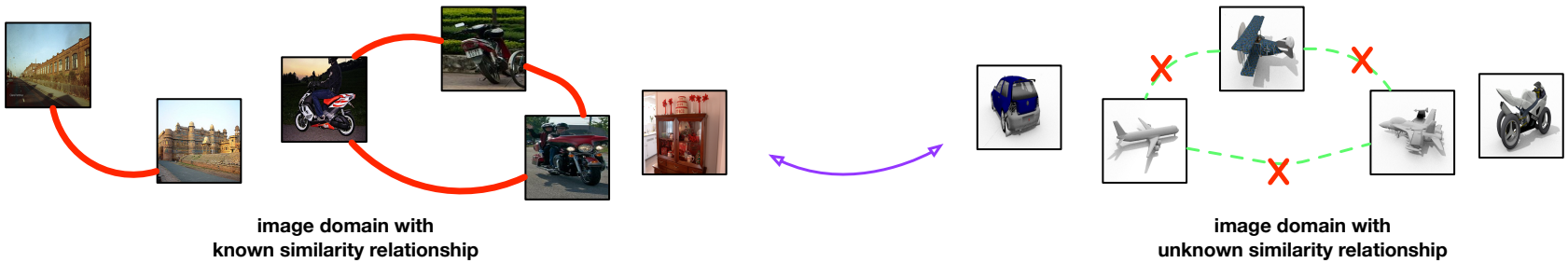


Figure: Concentration Problem in Transfer Hamming Space Retrieval

Network Architecture



Hash Function Learning



Logarithm Maximum a Posteriori estimation

Given the set of pairwise similarity labels $\mathcal{S} = \{s_{ij}\}$, the logarithm Maximum a Posteriori (MAP) estimation of training hash codes $\boldsymbol{H}^x = [\boldsymbol{h}_1^x, \ldots, \boldsymbol{h}_n^x]$ can be defined as

$$\begin{aligned} \log p\left(\boldsymbol{H}^x|\mathcal{S}\right) &\propto \log p\left(\mathcal{S}|\boldsymbol{H}^x\right) p\left(\boldsymbol{H}^x\right) \\ &= \sum_{s_{ij}\in\mathcal{S}} \log p\left(s_{ij}|\boldsymbol{h}_i^x, \boldsymbol{h}_j^x\right) p\left(\boldsymbol{h}_i^x\right) p\left(\boldsymbol{h}_j^x\right), \end{aligned} \tag{1}$$

where $p(\mathcal{S}|\boldsymbol{H}^x)$ is likelihood function, and $p(\boldsymbol{H}^x)$ is prior distribution.

Hash Function Learning

Conditional Probability

For each pair of points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $p(s_{ij}|\boldsymbol{h}_i^x, \boldsymbol{h}_j^x)$ is the conditional probability of their relationship s_{ij} given their hash codes $\boldsymbol{h}_i^x$ and $\boldsymbol{h}_j^x$, which can be defined using the pairwise logistic function,

$$p\left(s_{ij}|\boldsymbol{h}_i^x, \boldsymbol{h}_j^x\right) = \begin{cases} \sigma\left(\text{sim}\left(\boldsymbol{h}_i^x, \boldsymbol{h}_j^x\right)\right), & s_{ij} = 1 \\ 1 - \sigma\left(\text{sim}\left(\boldsymbol{h}_i^x, \boldsymbol{h}_j^x\right)\right), & s_{ij} = 0 \end{cases} \tag{2}$$

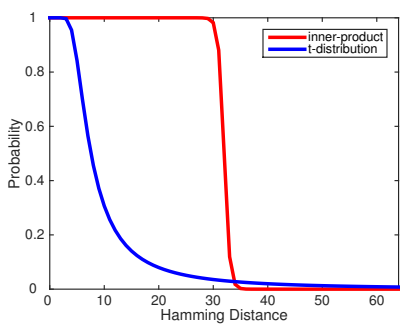
$$= \sigma\left(\text{sim}\left(\boldsymbol{h}_i^x, \boldsymbol{h}_j^x\right)\right)^{s_{ij}} \left(1 - \sigma\left(\text{sim}\left(\boldsymbol{h}_i^x, \boldsymbol{h}_j^x\right)\right)\right)^{1-s_{ij}},$$

where $\text{sim}\left(\boldsymbol{h}_i^x, \boldsymbol{h}_j^x\right)$ is the similarity function of code pairs $\boldsymbol{h}_i^x$ and $\boldsymbol{h}_j^x$ and $\sigma(x)$ is the probability function.

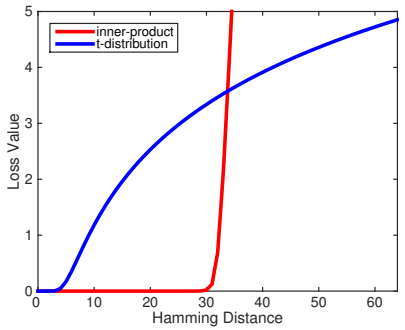
Hash Function Learning

Similarity Function and Probability Function

Previous methods [12, 2] usually use inner product $\left\langle \boldsymbol{h}_i^x, \boldsymbol{h}_j^x \right\rangle$ 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 proposes a new similarity function $\text{sim}\left(\boldsymbol{h}_i^x, \boldsymbol{h}_j^x\right) = \frac{b}{1+\left\|\boldsymbol{h}_i^x - \boldsymbol{h}_j^x\right\|^2}$ and the corresponding probability function is defined as $\sigma(x) = \tanh(\alpha x)$.



(a) Probability



(b) Loss Value

Hash Function Learning

Prior

Similar to previous work [11, 6, 12], defining that $\boldsymbol{h}_i^x = \mathrm{sgn}(\boldsymbol{z}_i^x)$ where $\boldsymbol{z}_i^x$ 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

$$p\left(\boldsymbol{z}_i^x\right) = \frac{1}{2\varepsilon} \exp\left(-\frac{\left|\boldsymbol{z}_i^x\right| - \mathbf{1}}{\varepsilon}\right) \tag{3}$$

where ε is the parameter of the exponential distribution.

Hash Function Learning

Optimization Poblem

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

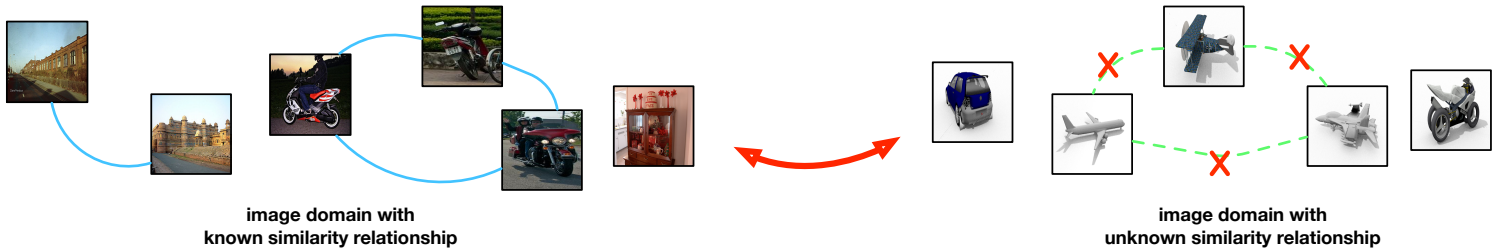
$$\min_{\theta} J = L + \lambda Q, \tag{4}$$

where λ is the trade-off parameter and θ is network parameters.

$$L = \sum_{s_{ij} \in \mathcal{S}} \log \left(1 + \exp \left(\frac{b}{1 + \left\| \boldsymbol{z}_i^x - \boldsymbol{z}_j^x \right\|_2} \right) \right) - s_{ij} \frac{b}{1 + \left\| \boldsymbol{z}_i^x - \boldsymbol{z}_j^x \right\|_2} \tag{5}$$

$$Q = \sum_{s_{ij} \in \mathcal{S}} \sum_{t=1}^{b} \left(- \log \cosh \left(|z_{it}^x| - 1 \right) - \log \cosh \left(|z_{jt}^x| - 1 \right) \right) \tag{6}$$

Homogeneous Distribution Alignment



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

Domain Distribution Alignment with Adversarial Nertwork

To extract domain-invariant hash codes $\mathbf{h}$, the parameters θ_f of deep hashing network G_f are learned by maximizing the loss of domain discriminator G_d, while the parameters θ_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:

$$D\left(\theta_f, \theta_y, \theta_d\right) = \frac{1}{n+m} \sum_{\mathbf{v}_i \in \mathcal{X} \cup \mathcal{Y}} L_d\left(G_d\left(G_f\left(\mathbf{v}_i\right)\right), d_i\right), \tag{7}$$

where L_d is the cross-entropy loss and d_i is the domain label of data point $\mathbf{v}_i$. $d_i = 1$ means $\mathbf{v}_i$ belongs to target domain and $d_i = 0$ means $\mathbf{v}_i$ belongs to source domain.

Transfer Adversarial Hashing

Unified optimization problem

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

$$C = J - \mu D, \tag{8}$$

where μ 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):

$$\begin{aligned} (\hat{\theta}_f, \hat{\theta}_y) &= \arg\min_{\theta_f, \theta_y} C\left(\theta_f, \theta_y, \theta_d\right), \\ (\hat{\theta}_d) &= \arg\max_{\theta_d} C\left(\theta_f, \theta_y, \theta_d\right). \end{aligned} \tag{9}$$

Experiments Setup

- **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

Method	NUS-WIDE				VisDA2017							
					synthetic → *real*				*real* → *synthetic*			
	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits
TAH	**0.722**	**0.729**	**0.692**	**0.680**	**0.465**	**0.423**	**0.433**	**0.404**	**0.672**	**0.695**	**0.784**	**0.761**
THN	0.671	0.676	0.662	0.603	0.415	0.396	0.228	0.127	0.647	0.687	0.664	0.532
HashNet	0.709	0.693	0.681	0.615	0.412	0.403	0.345	0.274	0.572	0.676	0.662	0.642
DHN	0.669	0.672	0.661	0.598	0.331	0.354	0.309	0.281	0.545	0.612	0.608	0.604
DNNH	0.568	0.622	0.611	0.585	0.241	0.276	0.252	0.243	0.509	0.564	0.551	0.503
CNNH	0.542	0.601	0.587	0.535	0.221	0.254	0.238	0.230	0.487	0.568	0.530	0.445
SDH	0.555	0.571	0.517	0.499	0.196	0.238	0.229	0.212	0.330	0.388	0.339	0.277
ITQ	0.498	0.549	0.517	0.402	0.187	0.175	0.146	0.123	0.163	0.193	0.176	0.158
SH	0.496	0.543	0.437	0.371	0.154	0.141	0.130	0.105	0.154	0.182	0.145	0.123
KSH	0.531	0.554	0.421	0.335	0.176	0.183	0.124	0.085	0.143	0.178	0.146	0.092
LSH	0.432	0.453	0.323	0.255	0.122	0.092	0.083	0.071	0.130	0.145	0.122	0.063

Results and Discussion

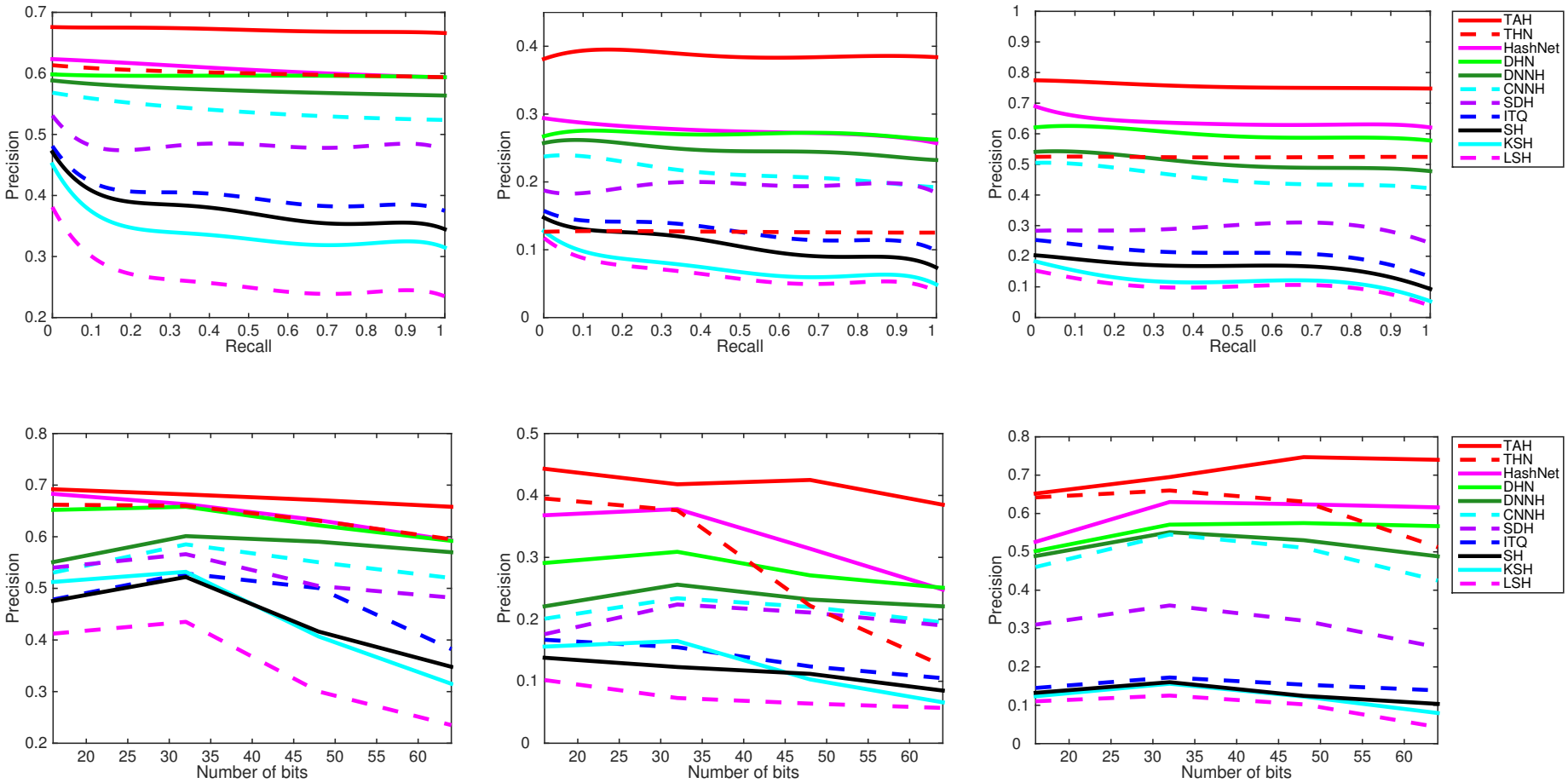


Figure: The Precision-recall curve @ 64 bits and the Precision within Hamming radius 2 of TAH and comparison methods on three tasks.

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

Method	*synthetic* → *real*				*real* → *synthetic*			
	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits
TAH-t	0.443	0.405	0.390	0.364	0.660	0.671	0.717	0.624
TAH-A	0.305	0.395	0.382	0.331	0.605	0.683	0.725	0.724
TAH	**0.465**	**0.423**	**0.433**	**0.404**	**0.672**	**0.695**	**0.784**	**0.761**

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.

Summary

- 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.
In *AAAI*, pages 81–87, 2017.


Z. Cao, M. Long, J. Wang, and P. S. Yu.
Hashnet: Deep learning to hash by continuation.
In *ICCV*, 2017.


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.
In *VLDB*, volume 99, pages 518–529. ACM, 1999.


Y. Gong and S. Lazebnik.
Iterative quantization: A procrustean approach to learning binary codes.
In *CVPR*, pages 817–824, 2011.


H. Lai, Y. Pan, Y. Liu, and S. Yan.
Simultaneous feature learning and hash coding with deep neural networks.
In *CVPR*. IEEE, 2015.


W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang.
Supervised hashing with kernels.
In *CVPR*. IEEE, 2012.


F. Shen, C. Shen, W. Liu, and H. Tao Shen.
Supervised discrete hashing.
In *CVPR*. IEEE, June 2015.


E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell.
Simultaneous deep transfer across domains and tasks.
In *ICCV*, 2015.


Y. Weiss, A. Torralba, and R. Fergus.
Spectral hashing.
In *NIPS*, 2009.


R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan.
Supervised hashing for image retrieval via image representation learning.
In *AAAI*, pages 2156–2162. AAAI, 2014.


H. Zhu, M. Long, J. Wang, and Y. Cao.
Deep hashing network for efficient similarity retrieval.
In *AAAI*. AAAI, 2016.