This CVPR2015 paper is the Open Access version, provided by the Computer Vision Foundation.
The authoritative version of this paper is available in IEEE Xplore.

Deep Hashing for Compact Binary Codes Learning

Venice Erin Liong[1], Jiwen Lu[1*], Gang Wang[1,2], Pierre Moulin[1,3], and Jie Zhou[4]
[1]Advanced Digital Sciences Center, Singapore
[2]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
[3]Department of ECE, University of Illinois at Urbana-Champaign, IL USA
[4]Department of Automation, Tsinghua University, Beijing, China
venice.l@adsc.com.sg; jiwen.lu@adsc.com.sg; wanggang@ntu.edu.sg;
moulin@ifp.uiuc.edu;jzhou@tsinghua.edu.cn

Abstract

In this paper, we propose a new deep hashing (DH) approach to learn compact binary codes for large scale visual search. Unlike most existing binary codes learning methods which seek a single linear projection to map each sample into a binary vector, we develop a deep neural network to seek multiple hierarchical non-linear transformations to learn these binary codes, so that the nonlinear relationship of samples can be well exploited. Our model is learned under three constraints at the top layer of the deep network: 1) the loss between the original real-valued feature descriptor and the learned binary vector is minimized, 2) the binary codes distribute evenly on each bit, and 3) different bits are as independent as possible. To further improve the discriminative power of the learned binary codes, we extend DH into supervised DH (SDH) by including one discriminative term into the objective function of DH which simultaneously maximizes the inter-class variations and minimizes the intra-class variations of the learned binary codes. Experimental results show the superiority of the proposed approach over the state-of-the-arts.

1. Introduction

Large scale visual search has attracted great attention in computer vision due to its wide potential applications in recent years [4]. Hashing is a powerful technique for large-scale visual search and a variety of hashing-based methods have been proposed [6, 7, 20, 34, 36]. The basic idea of hashing-based approach is to construct a series of hash functions to map each visual object into a binary feature vector so that visually similar samples are mapped into similar binary codes.

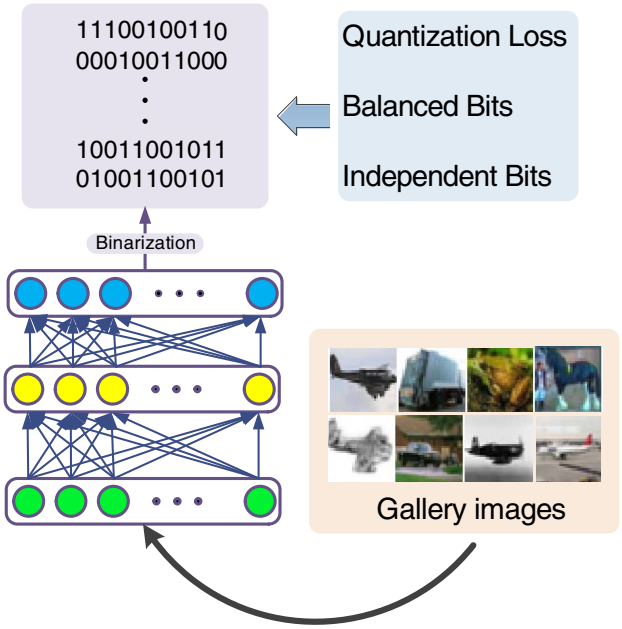


Figure 1. The basic idea of our proposed approach for compact binary codes learning. Given a gallery image set, we develop a deep neural network and learn the parameters of the network by using three criterions for the codes obtained at the top layer of the network: 1) minimizing loss between the original real-valued feature and the learned binary vector; 2) binary codes distribute evenly on each bit, and 3) each bit is as independent as possible. The parameters of the networks are updated by back-propagation based on the optimization objective function at the top layer.

Existing hashing-based methods can be classified into two categories: *data-independent* [1, 3, 11, 23] and *data-dependent* [5, 6, 8, 15, 21, 34]. For the first category, random projections are first employed to map samples into a feature space and then binarization is performed. Representative methods in this category are locality sensitive hashing (LSH) [1] and its kernelized or discriminative exten-

*Corresponding author.

sions [11, 16, 23]. For the second category, various statistical learning techniques are used to learn hashing functions to map samples into binary codes. State-of-the-art methods in this category include spectral hashing [36], binary reconstructive embedding (BRE) [15], iterative quantization (ITQ) [6], K-means hashing (KMH) [8], minimal-loss hashing (MLH) [21], and sequential projection learning hashing (SPLH) [34]. However, most these data-dependent hashing methods cannot well capture the nonlinear manifold structure of samples. While several kernel-based hashing methods [7, 19] have been proposed, they suffer from the scalability problem.

In this paper, we propose a new deep hashing (DH) method to learn compact binary codes for large scale visual search. Figure 1 illustrates the basic idea of the proposed approach. Unlike existing binary codes learning methods which seek a single linear projection to map each sample into a binary vector, we develop a deep neural network to seek multiple hierarchical non-linear transformations to learn these binary codes. Our model is learned under three constraints at the top layer of the deep network: 1) the loss between the original feature descriptor and the learned binary vector is minimized, 2) the binary codes distribute evenly on each bit, and 3) different bits are as independent as possible. To further improve the discriminative power of the learned binary codes, we extend DH into supervised DH (SDH) by including one discriminative term into the objective function of DH which simultaneously maximizes the inter-class variations and minimizes the intra-class variations of the learned binary codes. Experimental results on three widely used datasets are presented to show the efficacy of the proposed approach.

2. Related Work

Learning-based Hashing: Existing learning-based hashing methods can be classified into three categories: unsupervised, semi-supervised and supervised. For the first category, label information of the training set is not required in the learning procedure. For example, Weiss *et al.* [36] presented a spectral hashing method to obtain balanced binary codes by solving a spectral graph partitioning problem. Gong *et al.* [6] developed an ITQ method by simultaneously maximizing the variance of each binary bit and minimizing the binarization loss. He *et al.* [8] developed a KMH method by minimizing the hamming distance between the quantized cells and the cluster centers. Heo *et al.* [9] proposed a hypersphere-based hashing method by minimizing the spherical distance between the original real-valued features and the learned binary features. For the second category, the pairwise label information is used to learn hashing functions. For example, Wang *et al.* [33] developed a semi-supervised hashing (SSH) method by minimizing the empirical error for pairwise labeled training samples and maximizing the variance of both labeled and unlabeled training samples. Kulis and Darrell [15] presented a BRE method by minimizing the reconstruction error between the original Euclidean distance and the learned hamming distance. Norouzi and Fleet [21] presented a MLH method by minimizing the loss between the learned Hamming distance and the quantization error. For the third category, the class label information of each sample is used in hashing function learning. For example, Stretcha *et al.* [27] developed a LDA hashing by minimizing the intra-class variations and maximizing the inter-class variations of binary codes. Rastegari *et al.* [25] proposed a discriminative hashing method by learning multiple linear-SVMs with the large margin criterion. While these hashing methods have achieved reasonably good performance in many applications, most of them usually seek a single linear projection, which cannot well capture the nonlinear structure of samples.

Deep Learning: Deep learning aims to learn hierarchical feature representations by building high-level features from raw data. In recent years, a variety of deep learning algorithms have been proposed in computer vision and machine learning [2, 10, 12, 18, 17, 24, 30], and some of them have successfully applied to many visual analysis applications image classification [14], object detection [28], action recognition [17], face verification [29], and visual tracking [35]. Representative deep learning methods include deep stacked auto-encoder [17], deep convolutional neural networks [12], and deep belief network [10]. While deep learning has achieved great success in various visual application, little progress of deep learning have been made in hashing-based large scale visual search. To our knowledge, semantic hashing [26] is the first work on using deep learning techniques for hashing. They applied the stacked Restricted Boltzmann Machine (RBM) learn compact binary codes for visual search. However, their model is complex and requires pre-training, which is not efficient for practical applications.

3. Proposed Approach

In this section, we first present some basic knowledge of the learning-based hashing method and then detail our proposed DH and SDH methods.

3.1. Hashing

Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N] \in \mathbb{R}^{d \times N}$ be the training set which contains N samples, where $\mathbf{x}_n \in \mathbb{R}^d$ $(1 \leq n \leq N)$ is the nth sample in $\mathbf{X}$. Learning-based hashing methods aim to seek multiple hash functions to map and quantize each sample into a compact binary vector. Assume there are K hashing functions to be learned, which map each $\mathbf{x}_n$ into a K-bit binary codes vector $\mathbf{b}_n = [\mathbf{b}_{n1}, \cdots, \mathbf{b}_{nK}] \in \{-1, 1\}^{K \times 1}$, and the kth binary bit $\mathbf{b}_{nk}$ of $\mathbf{x}_n$ is computed

as follows:

$$\mathrm{b}_{nk} = f_k(\mathbf{x}_n) = \mathrm{sgn}\left(g_k(\mathbf{x}_n)\right) = \mathrm{sgn}(\mathrm{w}_k^T \mathbf{x}_n) \quad (1)$$

where f_k is the kth hashing function, and $\mathrm{w}_k \in \mathbb{R}^d$ is the projection in f_k, $\mathrm{sgn}(v)$ returns 1 if $v > 0$ and -1 otherwise.

Let $\mathrm{W} = [\mathrm{w}_1, \mathrm{w}_2, \cdots, \mathrm{w}_K] \in \mathbb{R}^{d \times K}$ be the projection matrix. Then, the mapping of $\mathbf{x}_n$ can be computed as: $g(\mathbf{x}_n) = \mathrm{W}^T \mathbf{x}_n$, which can be further binarized to obtain the binary codes as follows:

$$\mathrm{b}_n = \mathrm{sgn}(\mathrm{W}^T \mathbf{x}_n) \quad (2)$$

While a variety of learning-based hashing methods have been proposed with different motivations in recent years [5, 6, 8, 15, 21, 34], most of them aim to learn the projection matrix W with different objective functions and constraints. However, most existing hashing methods only learn a single projection matrix, which is in essence linear and cannot well capture the nonlinear manifold of samples. While some kernel-based hashing methods have been presented [7, 19], they still suffer from the scalability problem because these kernel-based methods cannot obtain the explicit nonlinear mapping. In this work, we propose a deep learning approach to learn multiple nonlinear transformations to obtain the compact binary codes.

3.2. Deep Hashing

As shown in Figure 1, for a given sample $\mathbf{x}_n$, we obtain a binary vector $\mathbf{b}_n$ by passing it to a network which contains multiple stacked layers of nonlinear transformations. Assume there are $M+1$ layers in our deep network, and there are p^m units for the mth layer, where $m = 1, 2, \cdots, M$. For a given sample $\mathbf{x}_n \in \mathbb{R}^d$, the output of the first layer is: $\mathbf{h}_n = s(\mathbf{W}^1 \mathbf{x}_n + \mathbf{c}^1) \in \mathbb{R}^{p^1}$, where $\mathbf{W}^1 \in \mathbb{R}^{p^1 \times d}$ is the projection matrix to be learned at the first layer of the network, $\mathbf{c}^1 \in \mathbb{R}^{p^1}$ is the bias, and $s(\cdot)$ is a nonlinear activation function. The output of the first layer is then considered as the input for the second layer, so that $\mathbf{h}_n^2 = s(\mathbf{W}^2 \mathbf{h}_n^1 + \mathbf{c}^2) \in \mathbb{R}^{p^2}$, where $\mathbf{W}^2 \in \mathbb{R}^{p^2 \times p^1}$ and $\mathbf{c}^2 \in \mathbb{R}^{p^2}$ are the projection matrix and bias vector for the second layer, respectively. Similarly, the output for the mth layer is: $\mathbf{h}_n^m = s(\mathbf{W}^m \mathbf{h}_n^{m-1} + \mathbf{c}^m)$, and the output at the top layer of our network is:

$$g_{DH}(\mathbf{x}_n) = \mathbf{h}_n^M = s(\mathbf{W}^M \mathbf{h}_n^{M-1} + \mathbf{c}^M) \quad (3)$$

where the mapping $g_{DH} : \mathbb{R}^d \to \mathbb{R}^{p^M}$ is parameterized by $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M, 1 \le m \le M$.

Now, we perform hashing for the output $\mathbf{h}^M$ at the top layer of the network to obtain binary codes as follows:

$$\mathbf{b}_n = \mathrm{sgn}(\mathbf{h}_n^M) \quad (4)$$

Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_N] \in \{-1, 1\}^{K \times N}$ and $\mathbf{H}^m = [\mathbf{h}_1^m, \mathbf{h}_2^m, \cdots \mathbf{h}_N^m] \in \mathbb{R}^{p^m \times N}$ be the matrix representation of the binary codes vectors and the output of the mth layer of the network, we formulate the following optimization problem to learn the parameters of our network:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{c}} J &= J_1 - \lambda_1 J_2 + \lambda_2 J_3 + \lambda_3 J_4 \\ &= \frac{1}{2}\|\mathbf{B} - \mathbf{H}^M\|_F^2 - \frac{\lambda_1}{2N}\mathrm{tr}((\mathbf{H}^M \mathbf{H}^M)^T) \\ &+ \frac{\lambda_2}{2}\sum_{m=1}^{M} \|\mathbf{W}^m (\mathbf{W}^m)^T - \mathbf{I}\|_F^2 \\ &+ \frac{\lambda_3}{2}(\|\mathbf{W}^m\|_F^2 + \|\mathbf{c}^m\|_2^2) \end{aligned} \quad (5)$$

The first term J_1 aims to minimize the quantization loss between the learned binary vectors and the original real-valued vectors. The second term J_2 aims to maximize the variance of learned binary vectors to ensure balanced bits. The third term J_3 enforces a relaxed orthogonality constraint on those projection matrices so that the independence of each transform is maximized. The last term J_4 are regularizers to control the scales of the parameters. λ_1, λ_2, and λ_3 are three parameters to balance the effect of different terms.

To solve this optimization problem, we employ the stochastic gradient descent method to learn parameters $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$. The gradient of the objective function in (5) with respect to different parameters are computed as fol-

Algorithm 1: DH

Input: Training set $\mathbf{X}$, network layer number M, learning rate η, iterative number R, parameters λ_1, λ_2 and λ_3, and convergence error ε.
Output: Parameters $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$.
Step 1 (Initialization):
Initialize $\mathbf{W}^1$ by getting the top p^1 eigenvectors from the covariance matrix.
Initialize $\{\mathbf{W}^m\}_{m=2}^{m=M} = \mathbf{I}^{p_{m-1} \times p_m}$ and $\{\mathbf{c}^m\}_{m=1}^{m=M} = \mathbf{1}^{p_m \times 1}$.
Step 2 (Optimization by back propagation):
for $r = 1, 2, \cdots, R$ **do**
  Set $\mathbf{H}^0 = \mathbf{X}$
  for $m = 1, 2, \cdots, M$ **do**
    Compute $\mathbf{H}^m$ using the deep networks from (3).
  end
  for $m = M, M-1, \cdots, 1$ **do**
    Obtain the gradients according to (6)-(7).
  end
  for $m = 1, 2, \cdots, M$ **do**
    Update $\mathbf{W}^m$ and $\mathbf{c}^m$ according to (10)-(11).
  end
  Calculate J_t using (5).
  If $r > 1$ and $|J_r - J_{r-1}| < \varepsilon$, go to **Return**.
end
Return: $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$.

lows:

$$\frac{\partial J}{\partial \mathbf{W}^m} = \Delta^m (\mathbf{H}^{m-1})^T + \lambda_2 \mathbf{W}^m (\mathbf{W}^m (\mathbf{W}^m)^T - \mathbf{I}) + \lambda_3 \mathbf{W}^m \quad (6)$$

$$\frac{\partial J}{\partial \mathbf{c}^m} = \Delta^m + \lambda_3 \mathbf{c}^m \quad (7)$$

where

$$\Delta^M = (-(\mathbf{B} - \mathbf{H}^M) - \lambda_1 \mathbf{H}^M) \odot s'(\mathbf{Z}^M) \quad (8)$$

$$\Delta^m = ((\mathbf{W}_1^{m+1})^T \Delta^{m+1}) \odot s'(\mathbf{Z}^m) \quad (9)$$

Here $\odot$ denotes element-wise multiplication, and $\mathbf{Z}^m = \mathbf{W}^m \mathbf{H}^{m-1} + \mathbf{c}^m$.

The parameters are updated by using the following gradient descent algorithm until convergence.

$$\mathbf{W}^m = \mathbf{W}^m - \eta \frac{\partial J}{\partial \mathbf{W}^m} \quad (10)$$

$$\mathbf{c}^m = \mathbf{c}^m - \eta \frac{\partial J}{\partial \mathbf{c}^m} \quad (11)$$

where η is the step-size. **Algorithm 1** summarizes the detailed procedure of the proposed DH method.

3.3. Supervised Deep Hashing

Since DH is an unsupervised learning approach, it is desirable to further improve its performance by using the label information of training samples. In this subsection, we propose a supervised deep hashing (SDH) method which extends DH into a supervised version to enhance the discriminative power of DH. For each pair of training samples $(\mathbf{x}_i, \mathbf{x}_j)$, we know whether they are from the same class or not. Hence, we can construct two sets $\mathcal{S}$ or $\mathcal{D}$ from the training set, which represents the positive samples pairs and the negative samples pairs in the training set, respectively. Then, we formulate the following optimization problem for our SDH method:

$$\begin{aligned} \arg\min_{\mathbf{W},\mathbf{c}} J = & \frac{1}{2}\|\mathbf{B} - \mathbf{H}^M\|_F^2 \\ & - \frac{\lambda_1}{2}(\mathrm{tr}(\frac{1}{N}\mathbf{H}^M(\mathbf{H}^M)^T) + \alpha \mathrm{tr}(\Sigma_B - \Sigma_W)) \\ & + \frac{\lambda_2}{2}\sum_{m=1}^{M}\|\mathbf{W}^m(\mathbf{W}^m)^T - \mathbf{I}\|_F^2 \\ & + \frac{\lambda_3}{2}\sum_{m=1}^{M}(\|\mathbf{W}^m\|_F^2 + \|\mathbf{c}^m\|_2^2) \end{aligned} \quad (12)$$

Algorithm 2: SDH

Input: Training set $\mathbf{X}$, pairwise sample indices, network layer number M, learning rate η, iterative number R, parameter λ_1, λ_2, λ_3 and α, and convergence error ϵ.

Output: Parameters $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$.

Step 1 (Initialization):

Initialize $\mathbf{W}^1$ by getting the top p^1 eigenvectors from a semi-supervised hashing method in [33].
Initialize $\{\mathbf{W}^m\}_{m=2}^{m=M} = \mathbf{I}^{p_{m-1} \times p_m}$ and $\{\mathbf{c}^m\}_{m=1}^{m=M} = \mathbf{1}^{p_m \times 1}$.

Step 2 (Optimization by back propagation):

for $r = 1, 2, \cdots, R$ **do**
  Set $\mathbf{H}^0 = \mathbf{X}$, $\{\mathbf{H}_{s1}^0, \mathbf{H}_{s2}^0\}$ and $\{\mathbf{H}_{d1}^0, \mathbf{H}_{d2}^0\}$ for pairwise samples in $\mathcal{S}$ and $\mathcal{D}$ respectively from set $\mathbf{X}$
  for $m = 1, 2, \cdots, M$ **do**
    Compute $\mathbf{H}^m$, $\mathbf{H}_{s1}^m$, $\mathbf{H}_{s2}^m$, $\mathbf{H}_{d1}^m$, and $\mathbf{H}_{d2}^m$ using the deep networks.
  end
  for $m = M, M-1, \cdots, 1$ **do**
    Obtain the gradients according to (15)-(16).
  end
  for $m = 1, 2, \cdots, M$ **do**
    Update $\mathbf{W}^m$ and $\mathbf{c}^m$ according to (10)-(11).
  end
  Calculate J_t using (12).
  If $r > 1$ and $|J_r - J_{r-1}| < \varepsilon$, go to **Return**.
end

Return: $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$.

where

$$\Sigma_W = \frac{1}{N_S}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}}(\mathbf{h}_i^M - \mathbf{h}_j^M)(\mathbf{h}_i^M - \mathbf{h}_j^M)^T = \frac{1}{N_S}\mathrm{tr}((\mathbf{H}_{s1}^M - \mathbf{H}_{s2}^M)(\mathbf{H}_{s1}^M - \mathbf{H}_{s2}^M)^T) \quad (13)$$

$$\Sigma_B = \frac{1}{N_D}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}}(\mathbf{h}_i^M - \mathbf{h}_j^M)(\mathbf{h}_i^M - \mathbf{h}_j^M)^T = \frac{1}{N_D}\mathrm{tr}((\mathbf{H}_{d1}^M - \mathbf{H}_{d2}^M)(\mathbf{H}_{d1}^M - \mathbf{H}_{d2}^M)^T) \quad (14)$$

N_S and N_D are the number of neighbor and non-neighbor pairs[1], $\{\mathbf{H}_{s1}^m, \mathbf{H}_{s2}^m\}_{m=1}^M$ are the hidden representation of sample pairs in $\mathcal{S}$ and $\{\mathbf{H}_{d1}^m, \mathbf{H}_{d2}^m\}_{m=1}^M$ in $\mathcal{D}$. The objective of J_2 is to minimize the intra-class variations and maximize the inter-class variations, α is the parameter to balance these two parts in this term. The aims of J_1, J_3, and J_4 are the same as those of the DH method.

Similar to DH, we also use the stochastic gradient descent method to learn parameters $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$ in SDH.

[1]In our experiments, we randomly sampled 1000 positive and 1000 negative pairs from the training set to learn the SDH model.

Table 1. Results on the CIFAR-10 dataset. Results on the top section are from unsupervised methods and those on the bottom section are from the supervised methods. The first two columns show the Hamming ranking results evaluated by mAP and precision@N (where N=1000). The right column shows the Hamming look-up results when the Hamming radius $r = 2$. Hamming look-up for $K = 64$ is not evaluated because this evaluation is impractical for longer codes.

Method	Hamming ranking (mAP, %)			precision (%) @ sample = 500			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
PCA-ITQ [6]	15.67	16.20	16.64	22.46	25.30	27.09	22.60	14.99
KMH [8]	13.59	13.93	14.46	20.28	21.97	22.80	22.08	5.72
Spherical [9]	13.98	14.58	15.38	20.13	22.33	25.19	20.96	12.50
SH [36]	12.55	12.42	12.56	18.83	19.72	20.16	18.52	**20.60**
Semantic [26]	12.95	14.09	13.89	14.79	17.87	18.27	11.49	13.78
PCAH [34]	12.91	12.60	12.10	18.89	19.35	18.73	21.29	2.68
LSH [1]	12.55	13.76	15.07	16.21	19.10	22.25	16.73	7.07
DH	**16.17**	**16.62**	**16.96**	**23.79**	**26.00**	**27.70**	**23.33**	15.77
SPLH [34]	17.61	20.20	20.98	25.32	29.43	32.22	23.05	30.47
MLH [21]	18.37	20.49	21.89	24.43	29.60	33.01	23.52	28.72
BRE [15]	14.42	15.14	15.88	20.68	22.86	25.14	20.89	20.29
SDH	**18.80**	**20.83**	**22.51**	**26.32**	**30.42**	**33.60**	**23.26**	**31.48**

The gradient of the objective function in (12) with respect to these parameters can be computed as follows:

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{W}^m} &= \Delta^m \mathbf{H}^{m-1} + \frac{\alpha}{\lambda_1}(\Delta^m_{s1}\mathbf{H}^{m-1}_{s1} \\ &- \Delta^m_{s2}\mathbf{H}^{m-1}_{s2} - \Delta^m_{d1}\mathbf{H}^{m-1}_{d1} \\ &+ \Delta^m_{d2}\mathbf{H}^{m-1}_{d2}) + \lambda_3 \mathbf{W}^m \\ &+ \lambda_2 \mathbf{W}^m(\mathbf{W}^m)^T - \mathbf{I}) \qquad (15)\\ \frac{\partial J}{\partial \mathbf{c}^m} &= \Delta^m + \frac{\alpha}{\lambda_1}(\Delta^m_{s1} - \Delta^m_{s2} \\ &- \Delta^m_{d1} + \Delta^m_{d2}) + \lambda_3 \mathbf{c}^m \qquad (16)\end{aligned}$$

where the Δ terms for the top layer can be computed as follows:

$$\Delta^M_{s1} = \frac{1}{N_S}(\mathbf{H}^M_{s1} - \mathbf{H}^M_{s2}) \odot s'(\mathbf{Z}^M_{s1}) \qquad (17)$$

$$\Delta^m_{s2} = \frac{1}{N_S}(\mathbf{H}^M_{s1} - \mathbf{H}^M_{s2}) \odot s'(\mathbf{Z}^M_{s1}) \qquad (18)$$

$$\Delta^M_{d1} = \frac{1}{N_D}(\mathbf{H}^M_{d1} - \mathbf{H}^M_{d2}) \odot s'(\mathbf{Z}^M_{d1}) \qquad (19)$$

$$\Delta^m_{d2} = \frac{1}{N_D}(\mathbf{H}^M_{d1} - \mathbf{H}^M_{d2}) \odot s'(\mathbf{Z}^M_{d1}) \qquad (20)$$

For the hidden layer, they can be computed as follows:

$$\Delta^m_{s1} = ((\mathbf{W}^{m+1})^T \Delta^{m+1}_{s1}) \odot s'(\mathbf{Z}^m_{s1}) \qquad (21)$$

$$\Delta^m_{s2} = ((\mathbf{W}^{m+1})^T \Delta^{m+1}_{s2}) \odot s'(\mathbf{Z}^m_{s2}) \qquad (22)$$

$$\Delta^m_{d1} = ((\mathbf{W}^{m+1})^T \Delta^{m+1}_{d1}) \odot s'(\mathbf{Z}^m_{d1}) \qquad (23)$$

$$\Delta^m_{d2} = ((\mathbf{W}^{m+1})^T \Delta^{m+1}_{d2}) \odot s'(\mathbf{Z}^m_{d2}) \qquad (24)$$

Algorithm 2 summarizes the detailed procedure of the proposed SDH method.

4. Experiments

We conduct experiments on three widely used datasets to evaluate our proposed DH and SDH methods for compact binary codes learning and compare them with several state-of-the-art methods. The following describes the details of the experiments and results.

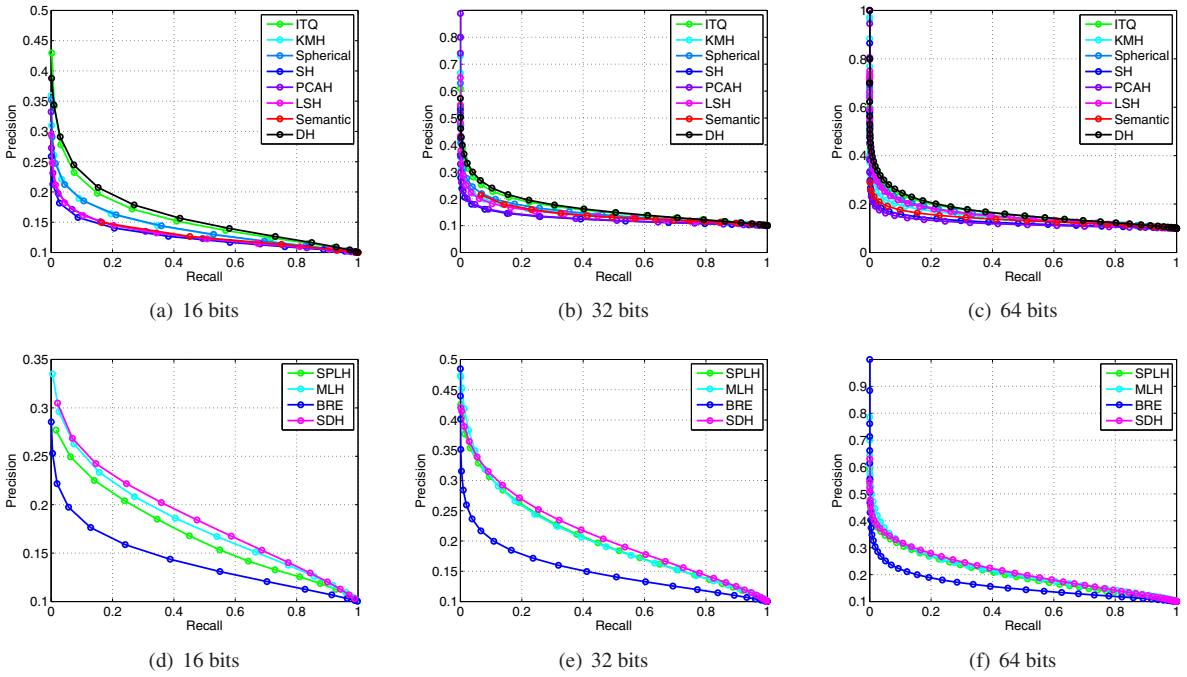
4.1. Results on CIFAR-10

The CIFAR-10 dataset [13] contains 60000 color images from 10 object classes, which are from the Tiny image dataset [31]. The size of each image is 32×32. Following the same setting in [34], We randomly sampled 1000 samples, 100 per class, as the query data, and used the remaining 59000 images as the gallery set. Each image was represented as a 512-D GIST feature vector [22].

For our DH and SDH methods, we trained our deep model with 3 layers by setting $M = 2$, where the dimensions for these layers were empirically set as $[60 \rightarrow 30 \rightarrow 16]$, $[80 \rightarrow 50 \rightarrow 32]$, and $[100 \rightarrow 80 \rightarrow 64]$ for the 16, 32 and 64 bits experiments, respectively. The parameters λ_1, λ_2 and λ_3 were empirically set as 100, 0.001 and 0.001, respectively. We used the hyperbolic tangent function as the non-linear activation function in our models. The parameter α for SDH was empirically set as 1.

We compared our methods with nine state-of-the-art hashing methods, where six of them are unsupervised and the other three are supervised. The six unsupervised methods include PCA-ITQ [6], KMH [8], Spherical [9], SH [36], PCAH [34] and LSH [1]. The three supervised methods are SPLH [34], MLH [21], and BRE [15]. For all of these nine compared methods, we employed the implementations of these compared methods provided by the original authors and used the default parameters recommended by the corresponding papers. Semantic [26]

We used the following three evaluation metrics to measure the performance of different methods: 1) mean average precision (mAP): which computes the area under the precision-recall curve and evaluates the overall performance of different hashing algorithms; 2) precision at N samples:



(a) 16 bits (b) 32 bits (c) 64 bits

(d) 16 bits (e) 32 bits (f) 64 bits

Figure 2. Recall vs. precision curve on the CIFAR dataset. The first row shows the results of unsupervised hashing methods and the second row for supervised hashing methods at 16, 32 and 64 bits, respectively.

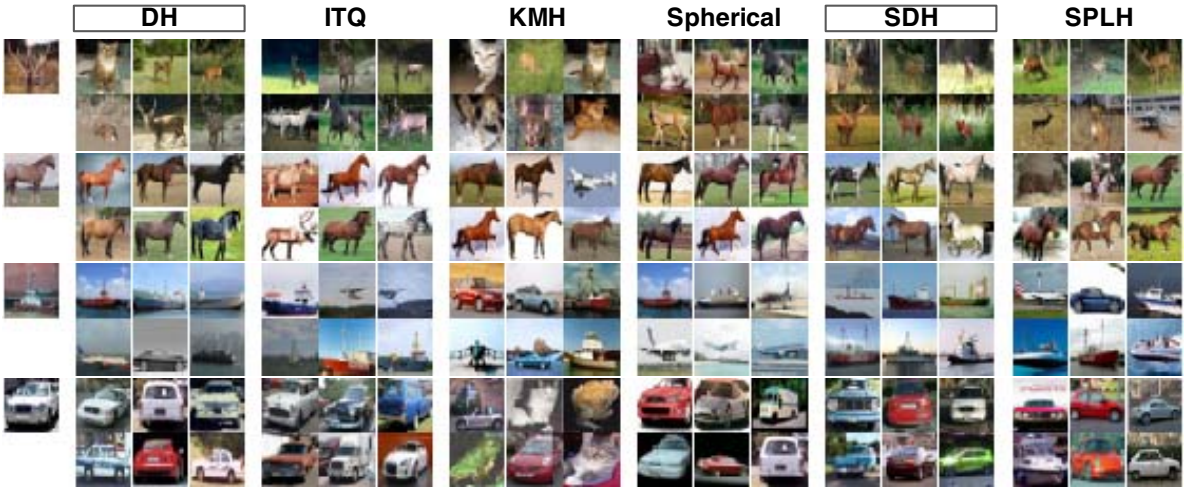
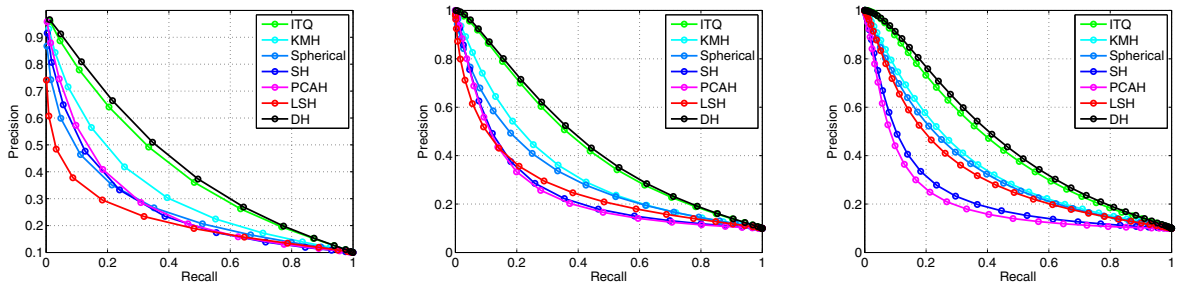


Figure 3. Top retrieved 6 images of 4 queries returned by different hashing methods on the CIFAR dataset. The image on the first column is the query sample. From left to right are the retrieved images by DH, ITQ, KMH, Spherical, SDH and SPLH when 64-bit binary codes are used for search.

which is the percentage of true neighbors among top N retrieved samples; and 3) Hamming look-up result when the hamming radius is set as r: which measures the precision over all the points in the buckets that fall within a hamming radius of $r = 2$ provided that a failed search would have zero precision. We repeated the experiments 10 times and took the average as the final result.

Table 1 shows the search results of different hashing methods on the CIFAR-10 dataset. Figure 2 shows the recall vs. precision curves for different methods on 16, 32 and 64 bits, respectively. As can be seen, our DH method outperforms the other compared unsupervised hashing methods, and our SDH outperforms the other compared supervised hashing methods, respectively. In addition, we also compare our model with the Semantic hashing [26] technique which also uses a deep model. Our method performs much better than the Semantic Hashing since our model not only minimize the reconstruction cost but also ensures balanced



(a) 16 bits (b) 32 bits (c) 64 bits

Figure 4. Recall vs. precision curve on the MNIST dataset for unsupervised hashing methods at 16, 32 and 64 bits, respectively.

Table 2. Results on the MNIST dataset. The top section are the unsupervised methods and the bottom section are the supervised methods. The first two columns show the Hamming ranking results evaluated by mAP and precision@N where N=1000, and the last column shows the Hamming look-up results when the Hamming radius $r = 2$.

Method	Hamming ranking (mAP, %)			precision (%) @ sample = 500			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
PCA-ITQ [6]	41.18	43.82	45.37	66.39	74.04	77.42	65.73	73.14
KMH [8]	32.12	33.29	35.78	60.43	67.19	72.65	61.88	68.85
Spherical [9]	25.81	30.77	34.75	49.48	61.27	69.85	51.71	64.26
SH [36]	26.64	25.72	24.10	56.29	61.29	61.98	57.52	65.31
PCAH [34]	27.33	24.85	21.47	56.56	59.99	57.97	36.36	65.54
LSH [1]	20.88	25.83	31.71	37.77	50.16	61.73	25.10	55.61
DH	**43.14**	**44.97**	**46.74**	**67.89**	**74.72**	**78.63**	**66.10**	**73.29**
SPLH [34]	44.20	48.29	48.34	62.98	67.89	67.99	63.71	74.06
BRE [15]	33.34	35.09	36.80	60.72	68.86	**73.08**	34.09	64.21
SDH	**46.75**	**51.01**	**52.50**	**65.19**	**70.18**	72.33	**63.92**	**77.07**

bits and independence of each transformation.

Figure 3 presents some example query images and the retrieved neighbors on the CIFAR-10 dataset when 64 bits were used to learn binary codes for different hashing methods. We see that our DH and SDH methods shows better search performance because higher semantic relevance can be obtained in the top retrieved samples.

4.2. Results on MNIST

The MNIST dataset[2] consists of 70000 handwritten digit images from 10 classes (labeled from 0 to 9). The size of each image is 28×28. We randomly sampled 1000 samples, 100 per class, as the query data, and used the remaining 69000 images as the gallery set. Each image was represented as a 784-D gray-scale feature vector by using its intensity [22]. We followed the same setting in the CIFAR-10 dataset and also used the same three evaluation metrics to compare different hashing methods. Table 2 shows the search results of different hashing methods on the MNIST dataset. Figure 4 shows the recall vs. precision curves for different unsupervised hashing methods on 16, 32 and 64 bits, respectively. As can be seen, our DH method outperforms the other compared unsupervised hashing methods, and our SDH outperforms the other compared supervised hashing methods, respectively.

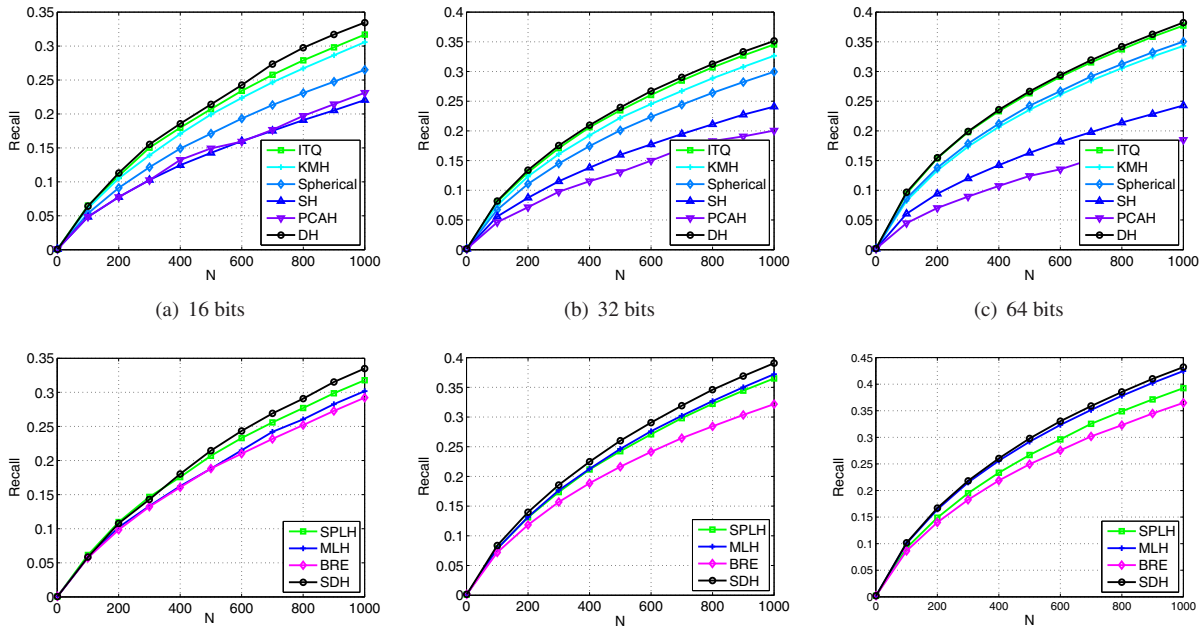
[2]http://yann.lecun.com/exdb/mnist/.

4.3. Results on LabelMe22k

The LabelMe dataset [32] consists of 22000 object images, where each image was represented as 512-D GIST feature. For each sample, a maximum of 50 semantic neighbors were provided as the ground truths for evaluation. Following the same evaluation protocol in [15], we randomly sampled 2000 images as the query data and used the remaining 20000 images are the gallery set. We measure the recall at the top N ranked samples to evaluate the different methods, which is defined as the fraction of retrieved true neighbors to the total number of true neighbors. Table 3 that the search results and Figure 5 shows the Recall@N results of different hashing methods on the LabelMe22k dataset, respectively. We see that our DH achieved very competitive results with the state-of-the-art unsupervised hashing methods and SDH outperforms the compared supervised hashing methods, respectively.

4.4. Computational Time

Lastly, we investigated the computational time of our proposed DH and SDH methods, and compared them with those of other hashing methods. Our PC is configured with a 3.40GHz CPU and 24.0 GB RAM. Table 4 shows the training and test time of different hashing methods on the CIFAR-10 dataset when 16-bit binary codes are used for evaluation, where the test time is computed for each query



(a) 16 bits (b) 32 bits (c) 64 bits

(d) 16 bits (e) 32 bits (f) 64 bits

Figure 5. Recall@N results for the LabelMe22k dataset. The first row shows the results of unsupervised methods and the second row for the supervised methods at 16, 32 and 64 bits, respectively.

Table 3. Recall@N results on the LabelMe22k database when N=1000. On the top section are the unsupervised methods and on the bottom section are the semi-supervised ones.

Method	Recall @ $N = 1000$		
	16	32	64
PCA-ITQ [6]	31.29	34.54	37.97
KMH [8]	30.58	32.65	34.31
Spherical [9]	26.51	29.96	35.04
SH [36]	22.04	24.09	24.28
PCAH [34]	23.11	20.06	18.49
LSH [1]	22.01	28.22	32.42
DH	**32.60**	**35.51**	**38.42**
SPLH [34]	32.29	36.90	37.82
MLH [21]	32.05	37.21	36.66
BRE [15]	28.46	33.02	38.30
SDH	**33.79**	**39.10**	**42.98**

Table 4. Computational time of different hashing methods on the CIFAR-10 dataset.

Method	Training (seconds)	Test (microseconds)
PCA-ITQ [6]	5.0	2.6
KMH [8]	72.4	29.0
Spherical [9]	6.7	6.5
SH [36]	0.7	6.9
Semantic [26]	0.0023	43.1
PCAH [34]	0.7	0.06
LSH [1]	0.1	0.08
DH	27.5	4.1
SPLH [34]	8.0	2.5
MLH [21]	1770.0	2.5
BRE [15]	37.6	4.3
SDH	54.8	4.8

image. We see that the training time of our methods are higher than previous hashing methods, and the test time are comparable to the existing methods.

5. Conclusion

In this paper, we have proposed two hashing methods called deep hashing (DH) and supervised deep hashing (SDH) for compact binary codes learning. Experimental results on three widely used datasets showed the effectiveness of the proposed methods. How to apply our proposed methods to other vision applications such as object recognition and visual tracking seems an interesting future work.

Acknowledgement

This work was supported by a research grant from the Agency for Science, Technology and Research of Singapore for the Human Cyber Security Systems (HCSS) Program at the Advanced Digital Sciences Center, the research grant of Singapore Ministry of Education (MOE) Tier 2 ARC28/14, Singapore A*STAR Science and Engineering Research Council PSF1321202099, and the National Natural Science Foundation of China under Grant 61225008, the National Basic Research Program of China under Grant 2014CB349304, the Ministry of Education of China under Grant 20120002110033, and the Tsinghua University Initiative Scientific Research Program.

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006. 1, 5, 7, 8

[2] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. 2

[3] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, pages 812–815, 2008. 1

[4] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM CSUR*, 40(2):5, 2008. 1

[5] Y. Gong, S. Kumar, V. Verma, and S. Lazebnik. Angular quantization-based binary codes for fast similarity search. In *NIPS*, pages 1196–1204, 2012. 1, 3

[6] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824, 2011. 1, 2, 3, 5, 7, 8

[7] J. He, W. Liu, and S.-F. Chang. Scalable similarity search with optimized kernel hashing. In *KDD*, pages 1129–1138, 2010. 1, 2, 3

[8] K. He, F. Wen, and J. Sun. K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In *CVPR*, pages 2938–2945, 2013. 1, 2, 3, 5, 7, 8

[9] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *CVPR*, pages 2957–2964, 2012. 2, 5, 7, 8

[10] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 2

[11] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *CVPR*, pages 1–8, 2008. 1, 2

[12] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *PAMI*, 35(1):221–231, 2013. 2

[13] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Univ. of Toronto, 2009. 5

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 2

[15] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1042–1050, 2009. 1, 2, 3, 5, 7, 8

[16] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137, 2009. 2

[17] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, pages 3361–3368, 2011. 2

[18] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ACM*, pages 609–616, 2009. 2

[19] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012. 2, 3

[20] X. Liu, J. He, B. Lang, and S.-F. Chang. Hash bit selection: a unified solution for selection problems in hashing. In *CVPR*, pages 1570–1577, 2013. 1

[21] M. Norouzi and D. M. Blei. Minimal loss hashing for compact binary codes. In *ICML*, pages 353–360, 2011. 1, 2, 3, 5, 8

[22] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001. 5, 7

[23] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517, 2009. 1, 2

[24] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, pages 1–8, 2007. 2

[25] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV*, pages 876–889, 2012. 2

[26] R. Salakhutdinov and G. Hinton. Semantic hashing. *IJAR*, 50(7):969–978, 2009. 2, 5, 6, 8

[27] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. *PAMI*, 34(1):66–78, 2012. 2

[28] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *NIPS*, pages 2553–2561, 2013. 2

[29] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014. 2

[30] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *ECCV*, pages 140–153, 2010. 2

[31] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *PAMI*, 30(11):1958–1970, 2008. 5

[32] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, pages 1–8, 2008. 7

[33] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, pages 3424–3431, 2010. 2, 4

[34] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *PAMI*, 34(12):2393–2406, 2012. 1, 2, 3, 5, 7, 8

[35] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, pages 809–817, 2013. 2

[36] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008. 1, 2, 5, 7, 8