DeepHash for Image Instance Retrieval: Getting Regularization, Depth and Fine-Tuning Right

Jie Lin
Institute for Infocomm Research, Singapore
lin-j@i2r.a-star.edu.sg

Olivier Morère
Université Pierre et Marie Curie, Paris, France
olivier.morere@gmail.com

Antoine Veillard
Université Pierre et Marie Curie, Paris, France
antoine.veillard@gmail.com

Ling-Yu Duan
School of EE&CS, Peking University, China
lingyu@pku.edu.cn

Hanlin Goh
Institute for Infocomm Research, Singapore
gohhanlin@gmail.com

Vijay Chandrasekhar*
Institute for Infocomm Research / Nanyang Technological University, Singapore
vijay@i2r.a-star.edu.sg

ABSTRACT

This work focuses on representing very high-dimensional global image descriptors using very compact 64-1024 bit binary hashes for instance retrieval. We propose DeepHash: a hashing scheme based on deep networks. Key to making DeepHash work at extremely low bitrates are three important considerations – regularization, depth and fine-tuning – each requiring solutions specific to the hashing problem. In-depth evaluation shows that our scheme outperforms state-of-the-art methods over several benchmark datasets for both Fisher Vectors and Deep Convolutional Neural Network features, by up to 8.5% over other schemes. The retrieval performance with 256-bit hashes is close to that of the uncompressed floating point features – a remarkable 512× compression.

CCS CONCEPTS

•**Information systems → Image search;**

KEYWORDS

RBM; Hashing; Regularization; Siamese Network; Fisher Vectors; CNN; Image Instance Retrieval

1 INTRODUCTION

A compact binary image representation such as a 64-bit hash is a definite must for fast image retrieval. 64 bits provide more than enough capacity for any practical purposes, including internet-scale problems. In addition, a 64-bit hash is directly addressable in RAM and enables fast matching using Hamming distances.

State-of-the-art global image descriptors such as Fisher Vectors (FV) [40] and Deep Convolutional Neural Network (DCNN) features [27] allow for robust image matching. However, the dimensionality of such descriptors is typically very high: 8192 to 65536 floating point numbers for FVs[40] and 512 to 4096 for DCNNs [5, 4, 45]. Bringing such high-dimensional floating point representations down to a 64-bit hash is a considerable challenge.

Deep learning has achieved remarkable success in many visual tasks such as image classification [27, 43], image retrieval [5], face recognition [1, 44] and pose estimation [47]. Furthermore, specific architectures such as stacked restricted Boltzmann machines (RBM) are primarily known as powerful dimensionality reduction techniques [42].

We propose *DeepHash*, a deep binary hashing scheme that combines purpose-specific regularization with weakly-supervised fine-tuning. A thorough empirical evaluation on a number of publicly available data sets shows that DeepHash surpasses other state-of-the-art methods at bitrates from 1024 down to 64. This is due to the correct mix of regularization, depth and fine-tuning. This work represents a strong step towards the Holy Grail of a perfect 64-bit hash.

2 RELATED WORK AND CONTRIBUTIONS

Hashing schemes can be broadly categorized into unsupervised and supervised (including semi-supervised) schemes. Examples of unsupervised schemes are Iterative Quantization [17], Spectral Hashing [50], Restricted Boltzmann Machines [42], while some examples of state-of-the-art supervised schemes include Minimal Loss Hashing [39], Kernel-based Supervised Hashing [28], Ranking-based Supervised Hashing [49] and Column Generation Hashing [31]. Supervised hashing schemes are typically applied to the semantic retrieval problem. In this work, we are focused on instance retrieval: semantic retrieval is outside the scope of this work.

There is plenty of work on binary codes for descriptors like SIFT or GIST [17, 48, 30, 18, 50, 28, 34, 39, 46, 8, 9]. There is comparatively little work on hashing descriptors like Fisher Vectors (FV) which are two orders of magnitude higher in dimensionality. Perronnin et al. [40] propose ternary quantization of FV, quantizing each dimension to +1,-1 or 0. Perronnin et al. also explore Locality Sensitive Hashing [51] and Spectral Hashing [50]. Spectral Hashing

*J. Lin, O. Morère, A. Veillard and V. Chandrasekhar contributed equally to this work.

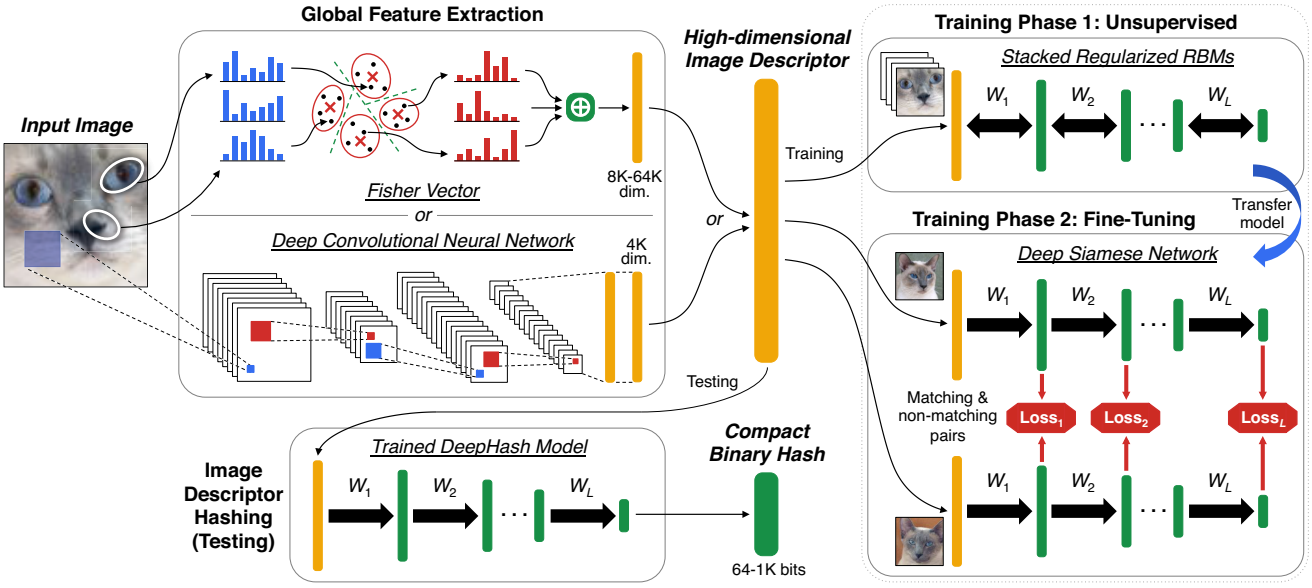


Figure 1: Our proposed hashing and model training pipeline. A high-dimensional global image descriptor, such as fisher vector and deep convolutional neural net feature, is extracted from an image. The trained DeepHash model transforms this image descriptor to a compact binary hash (between 64 to 1K bits), via a succession of L nonlinear feedforward projections. The DeepHash model is trained in two phases: a unsupervised pre-training phase and a weakly supervised fine-tuning phase. In phase 1, restricted Boltzmann machines (RBMs) are trained in a layer-wise manner and stacked into a deep network. In phase 2, matching and non-matching pairs are used to construct deep Siamese networks for parameter fine-tuning.

performs poorly at high rates, while LSH and simple ternary quantization need thousands of bits to achieve good performance. Gong et al. propose the popular Iterative Quantization (ITQ) scheme and apply it to GIST [17]. In subsequent work, Gong et al. [16] focus on generating very long codes for global descriptors, and the Bilinear Projection-based Binary Codes (BPBC) scheme requires tens of thousands of bits to match the performance of the uncompressed global descriptor. Jegou et al. propose Product Quantization (PQ) for obtaining compact representations [25]. While this produces compact descriptors, the resulting representation is not binary and cannot be compared with Hamming distances. As opposed to previous work, our focus is on generating extremely compact binary representations for FV and DCNN features in the 64 bits-1024 bits range.

In this paper, we propose *DeepHash* (Figure 1), a hashing scheme based on deep networks for high-dimensional global descriptors. The key to making the DeepHash scheme work at extremely low bitrates are three important considerations – regularization, depth and fine-tuning – each requiring solutions specific to the hashing problem.

- We pre-train a deep network using a RBM regularization scheme that is specifically adapted to the hashing problem. This enhances the efficiency of compact hashes, while achieving performance close to the uncompressed descriptor.
- Using stacked RBMs as a starting point, we fine-tune the model as a deep Siamese network. Critical improvements in the loss function lead to further improvements in retrieval results.
- DeepHash training is only required to be performed once on a single large independent training set. Through a thorough evaluation against state-of-the-art hashing schemes used for instance retrieval across a wide range of data sets with both DCNN and FV descriptors, we show that DeepHash outperforms other schemes (up 8.5%), particularly at low bit rates.

3 DEEPHASH

DeepHash is a hashing scheme based on a deep network to generate binary compact hashes for image instance retrieval (Figure 1).[1] Given a global image descriptor $\mathbf{z}^0$, a deep network performs a series of L layers of nonlinear projections to generate a compact hash $\mathbf{z}^L$. The model is trained in two phases: 1) greedy layer-wise unsupervised pre-training with hashing regularization and 2) weakly-supervised Siamese fine-tuning.

In the unsupervised phase, stacked restricted Boltzmann machines (RBMs) [22] are used to learn the initial parameters of the deep network. Each new layer in the network is trained to model the data distribution of the previous layer and is regularized specifically for hashing. A key feature is that this unsupervised pre-trained model is easily transferable. The unsupervised RBM parameters, which can be used to generate good hashes, can be further optimized with a fine-tuning phase. Fine-tuning is done through weak supervision by treating the deep model as a Siamese network [6]. Fine-tuning is also carried out an independent data set. In the rest

[1]DeepHash will be made publicly available on Caffe Model Zoo [26] (https://github.com/BVLC/caffe/wiki/Model-Zoo).

of this section, we will describe the details of the training process for our deep hashing scheme.

3.1 Stacked Reguarized RBMs

The deep network with L layers is initially pre-trained layer-by-layer from the bottom up through unsupervised learning, where each pair of successive layers ($\mathbf{z}^{l-1}$ and $\mathbf{z}^{l}$) is trained as an RBM building block. An RBM is an bipartite Markov random field with the input layer $\mathbf{z}^{l-1} \in \mathbb{R}^I$ connected to a latent layer $\mathbf{z}^{l} \in \mathbb{R}^J$ via a set of undirected weights $\mathbf{W}^{l} \in \mathbb{R}^{IJ}$. The input units z_i^{l-1} and latent units z_j^{l} are also parameterised by their corresponding biases c_i^{l-1} and b_j^{l}, respectively.

Binary RBMs. The first layer of the deep network takes a high-dimensional image descriptor as input. Previous works [40, 3] have shown that binarization of FV and DCNN features results in negligible loss in performance. For this work, binarization is done by component-wise mean thresholding for the inputs. We use binary latent units with sigmoid activation function, because binary output bits are desired for our hash. Binary RBMs are also faster and simpler to train as compared to continuous RBMs [20]. All layers in the deep network will consist of binary units and binary hashes can be extracted from all intermediate layers.

The units within a layer are conditionally independent pairwise. Therefore, the activation probabilities of one layer can be sampled by fixing the states of the other layer, and using distributions given by logistic functions for binary RBMs:

$$\mathbb{P}(z_j^{l}|\mathbf{z}^{l-1}) = 1/(1+\exp(-\mathbf{w}_j\mathbf{z}^{l-1} - b_j)), \tag{1}$$

$$\mathbb{P}(z_i^{l-1}|\mathbf{z}^{l}) = 1/(1+\exp(-\mathbf{w}_i^{\top}\mathbf{z}^{l} - c_i)). \tag{2}$$

As a result, alternating Gibbs sampling can be performed between the two layers. The sampled states are used to update the parameters $\{\mathbf{W}^{l}, \mathbf{b}^{l}, \mathbf{c}^{l-1}\}$ through minibatch gradient descent using the contrastive divergence algorithm [21] to approximate the maximum likelihood of the input distribution.

Given a trained RBM with fixed parameters and an input vector, a hash can be generated through a feedforward projection and thresholding Equation (1) at 0.5.

$$z_j^{l} = \begin{cases} 1, & \text{if } \mathbb{P}(z_i^{l-1}|\mathbf{z}^{l}) > 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Hashing Regularization. The unsupervised RBM is naively trained without considering the task, which in this case is image hashing. It is, however, important for the RBMs to project the data in a latent subspace that is suitable for hashing. One way to encourage the learning of suitable representations is to perform regularization, such as sparsity [29, 37, 15]. For classification, representations are encouraged to be very sparse to improve separability. For hashing, however, it is desirable to encourage the representation to make efficient use of the limited latent subspace.

For a given l and a minibatch of input instances $\mathbf{z}_\alpha^{l-1}$, we add a regularization term to the RBM optimization problem to encourage (a) half the bits to be active for a given hash, and (b) each bit value to be equiprobable across hashes:

$$\underset{\{\mathbf{W}^{l}, \mathbf{b}^{l}, \mathbf{c}^{l-1}\}}{\arg\min} -\sum_{\alpha} \log\Bigg(\sum_{\mathbf{z}_\alpha^{l} \in \mathcal{E}_\alpha} \mathbb{P}(\mathbf{z}_\alpha^{l-1}, \mathbf{z}_\alpha^{l}) + \lambda h(\mathcal{E}_\alpha) \Bigg), \tag{4}$$

where $\mathcal{E}_\alpha$ is the minibatch of sampled latent units for layer l and λ is the regularization constant.

We adapt the fine-grained regularization proposed in [15] to suit our hashing problem. For each instance $\mathbf{z}_\alpha^{l}$, the regularization term for binary units penalises each unit $z_{j\alpha}^{l}$ with the cross entropy loss with respect to a target activation $t_{j\alpha}^{l}$ based on a predefined distribution,

$$h(\mathcal{E}_\alpha) = -\sum_{\mathbf{z}_\alpha^{l} \in \mathcal{E}_\alpha} \sum_{j} t_{j\alpha}^{l} \log z_{j\alpha}^{l} + (1 - t_{j\alpha}^{l}) \log(1 - z_{j\alpha}^{l}). \tag{5}$$

Unlike [15], we choose the $t_{j\alpha}^{l}$ such that each $\{t_{j\alpha}^{l}\}_j$ for fixed α and each $\{t_{j\alpha}^{l}\}_\alpha$ for fixed j is distributed according to $\mathcal{U}(0,1)$. The uniform distribution is suitable for hashing high-dimensional vectors because the regularizer encourages the each latent unit to be active with a mean of 0.5, while avoiding activation saturation. The result is a space-filling effect in the latent subspace, where data is efficiently represented.

After RBM training, we further enforce space utilization by substituting the learned RBM bias by the data set mean $\langle \mathbf{w}_j\mathbf{z}^{l-1} \rangle$ of the linear projection preceding the logistic. Equation (3) is modified such that the final hash is centered around 0.5:

$$z_j^{l} = \begin{cases} 1, & \text{if } \mathbf{w}_j\mathbf{z}^{l-1} - \langle \mathbf{w}_j\mathbf{z}^{l-1} \rangle > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

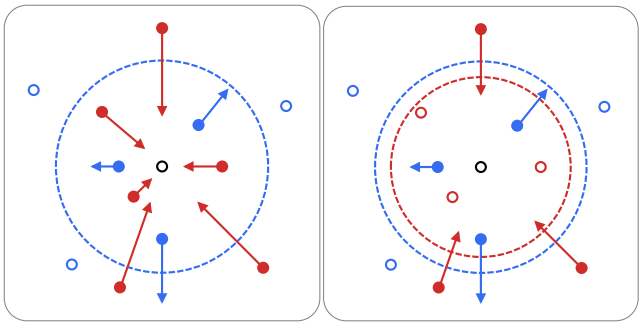
Stacked RBMs. The set of global image descriptors lie in a complex manifold in a very high-dimensional feature space. Deeper networks have the potential to discover more complex nonlinear hash functions and improve image instance retrieval performance. Following [22], we stack multiple RBMs by training one layer at a time to create a deep network with several layers.

Each layer models the activation distribution of the previous layer and captures higher order correlations between those units. For the hashing problem, we are interested in low-rate points of 64, 256 and 1024 bits, which are typical operating points as discussed in Section 4. We progressively decrease the dimensionality of latent layers by a factor of 2^n per layer, where n is a tuneable parameter. For our final models, n is empirically selected for each layer resulting in variable network depth.

3.2 Deep Siamese Fine-Tuning

Retrieval results are driven by the structure of the local neighborhood around the query. The unsupervised training is followed by a fine-tuning step in order to improve the local structure of the embedding. The fine-tuning is performed with a learning architecture known as Siamese networks first introduced in [6]. The principle was later successfully applied to deep architectures for face identification [12] and shown to produce representations robust to various transformations in the input space [19]. The use of Siamese architectures in the context of image retrieval from DCNN features was recently suggested as a possible improvement to the state-of-the-art on the subject [5].

A Siamese network is a weakly-supervised scheme for learning a similarity measure from pairs of data instances labeled as matching or non-matching. In our adaptation of the concept, the weights of the trained RBM network are fine-tuned by learning a similarity measure at every intermediate layer in addition to the target space.



(a) (b)

Figure 2: **A sample point (black dot) with corresponding matching (red dots) and non-matching (blue dots) samples. The contrastive divergence loss used for fine-tuning can be interpreted as applying attractive forces between matching elements (red arrows) and repulsive forces between non-matching elements (blue arrows). (a) The loss function (7) proposed in [19] with a single margin parameter for non-matching pairs (blue circle). Matching elements are subject to attractive forces regardless of whether they are already close enough from each other which adversely affects fine-tuning. (b) Our proposed loss function (8) with an additional margin parameter affecting matching pairs reciprocally (red circle).**

Given a pair of data $(\mathbf{z}_\alpha^0, \mathbf{z}_\beta^0)$, a contrastive loss $\mathcal{D}_l$ is defined for every layer l and the error is back propagated though gradient descent. Back propagation for the losses of individual layers ($l = 1..L$) is performed at the same time. Applying the loss function proposed by Handsell et al. in [19] yields:

$$\mathcal{D}_l(\mathbf{z}_\alpha^0, \mathbf{z}_\beta^0) = y\|\mathbf{z}_\alpha^l - \mathbf{z}_\beta^l\|_2^2 + (1-y)\max(m - \|\mathbf{z}_\alpha^l - \mathbf{z}_\beta^l\|_2^2, 0) \quad (7)$$

where $y = 1$ if $(\mathbf{z}_\alpha^0, \mathbf{z}_\beta^0)$ is a matching pair or $y = 0$ otherwise, and $m > 0$ is a margin parameter affecting non-matching pairs. As shown in Figure 2(a), the effect is to apply a contractive force between elements of any matching pairs and a repulsive force between elements of non-matching pairs which element-wise distance is shorter than $\sqrt{m}$.

However, experiment results in Figure 3 show that the loss function (7) causes a quick drop in retrieval results. Results with non-matching pairs alone suggest that the handling of matching pairs is responsible for the drop. The indefinite contraction of matching pairs well beyond what is necessary to distinguish them from non-matching elements is a damaging behaviour, specially in a fine-tuning context since the network is first globally optimized with a different objective. Figure 4 shows that any two elements, even matching, are always far apart in high dimension. As a solution, we propose a double-margin loss with an additional parameter affecting matching pairs:

$$\begin{aligned}\mathcal{D}_l(\mathbf{z}_\alpha^0, \mathbf{z}_\beta^0) =& y\max(\|\mathbf{z}_\alpha^l - \mathbf{z}_\beta^l\|_2^2 - m_1, 0) \\ &+ (1-y)\max(m_2 - \|\mathbf{z}_\alpha^l - \mathbf{z}_\beta^l\|_2^2, 0)\end{aligned} \quad (8)$$

As shown in Figure 2(b), the new loss can thus be interpreted as learning "local large-margin classifiers" (if $m_1 \leq m_2$) to distinguish between matching and non-matching elements. In practice, we found that the two margin parameters can be set equal ($m_1 = m_2 = m$) and tuned automatically from the statistical distribution of the sampled matching and non-matching pairs (Figure 4).

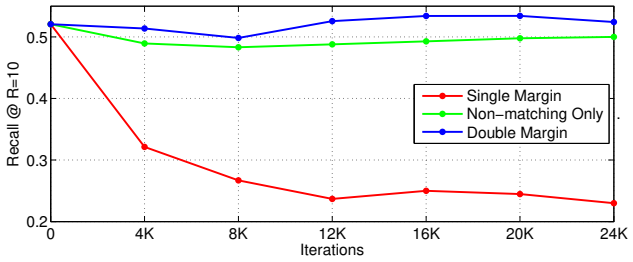


Figure 3: **Recall @ R=10 on the *Holidays* data set (See Section 4.1 for a description of the data sets) over several iterations of Siamese fine-tuning. The recall rate quickly collapses when using the single margin loss function suggested by Hadsell et al. [19] while performance is better retained when only non-matching pairs are passed. The double-margin loss solves the problem. The network is a stacked RBM (8192-4096-2048-64) trained with Fisher descriptors on the *ImageNet* data set. Matching pairs are sampled from the *Yandex* data set. For every matching pair, a random non-matching element is chosen from the data set to form two non-matching pairs. There are 33 matching pairs and 66 corresponding non-matching pair with every iteration. The test set is the Holidays data set.**

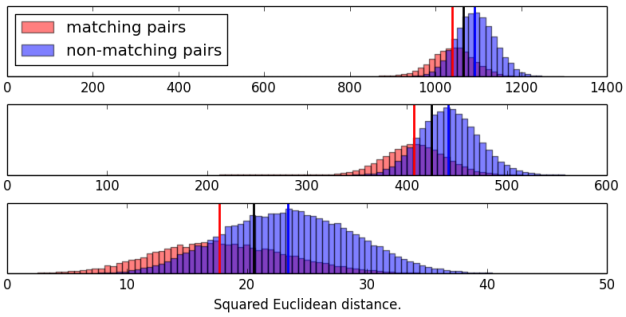


Figure 4: **Histograms of squared Euclidean distances for 20,000 matching pairs and corresponding 40,000 non-matching pairs for an 8192-4096(top)-2048(middle)-64(bottom) stacked RBM network. The red and blue vertical lines indicate the median values for the matching and non-matching pairs respectively. The Siamese loss shared margin value m is systematically set to be the mean of the two values (black vertical lines).**

4 EXPERIMENTAL RESULTS

4.1 Evaluation Framework

Global Descriptors. For the FV, we extract SIFT [35] features obtained from Difference-of-Gaussian (DoG) interest points. We use PCA to reduce dimensionality of the SIFT descriptor from 128 to 64 dimensions, which has shown to improve performance [24]. We use a Gaussian Mixture Model (GMM) with 128 centroids, resulting in 8192 dimensions each for first and second order statistics. Only the first-order statistics are retained in the global descriptor representation, as second-order statistics only results in a small improvement in performance [32]. The FV is L_2-normalized to unit-norm, after signed power normalization. We denote this configuration as the FV feature from here-on.

DCNN features are extracted using the open-source software Caffe [26] for AlexNet proposed for *ImageNet* classification in their seminal contribution [27]. For simplicity, we use the 4096-dimensional feature extracted from layer *fc6* (before softmax) for image retrieval, following [5]. We refer to this feature as the DCNN feature from here-on.

Training Data. Most schemes, including our proposed scheme, require a training step. We use the *ImageNet* data set for training, which consists of 1 million images from 1000 different image categories [13]. We randomly sample a subset of images from *ImageNet*. For the proposed deep Siamese fine-tuning scheme proposed, we use the 200K matching image pairs data set provided by *Yandex* in their recent work [5], consisting primarily of landmark images. For every matching pair, a random sample is picked to generate 2 corresponding non-matching pairs. This training set is independent of the query and database data described next.

Testing Data. We use 4 popular data sets for small scale experiments: *Oxford* (55 queries, 5062 database images) [41], *INRIA Holidays* (500 queries, 991 database images) [23], Stanford Mobile Visual Search *Graphics* (1500 queries, 1000 database images) [7, 2] and University of Kentucky Benchmark (UKB) (10200 queries, 10200 database images) [38]. For large-scale retrieval experiments, we present results on *Holidays* and *UKB* data sets, combined with the 1 million MIR-FLICKR distractor data set [36].

Comparisons. We compare several state-of-the-art schemes. Some have been proposed for lower dimensional vectors like SIFT and GIST, but we evaluate their performance on both FV and DCNN features.

- *ITQ* [17]. For the Iterative Quantization (ITQ) scheme, the authors propose signed binarization after applying two transforms: first the PCA matrix, followed by a rotation matrix, which minimizes the quantization error of mapping PCA-transformed data to the vertices of a zero-centered binary hypercube.
- *BPBC* [16]. Instead of the large PCA projection matrices used in [17], the authors apply bilinear projections, which require far less memory.
- *LSH* [51]. LSH is based on random unit-norm projections followed by signed binarization.
- *PQ* [25]. For FV and Product Quantization, we consider blocks of dimensions $D = 64, 256$ and 1024, and train $K = 256$ centroids for each block, resulting in $b = 64, 256$ and 1024 bit descriptors respectively. For DCNN, we consider blocks of dimensions $D = 32, 128$ and 1024, with $K = 256$ centroids, resulting in the same bitrates. Here, we do not apply Random Rotations, or PCA before applying PQ [25]. Such preprocessing can be applied to other schemes too. This is not a binary hashing scheme and only included for reference.

We ignore Spectral Hashing [50] due to its inferior performance on FV in [40].

4.2 DeepHash Experiments

Hashing Regularization. In Figure 6(a), we show the effect of applying regularization proposed in Section 3.1 on a single layer RBM 8192-b, for $b = 64, 256, 1024$. The *Holidays* data set and FV features are chosen. Hashing regularization improves performance significantly, $\sim$10% absolute recall @ $R = 10$ at low-rate point $b = 64$. The performance gap increases as rate decreases. This is intuitive as the regularization pushes the network towards keeping half the bits alive and equiprobable (across hashes), with its effect being more pronounced at lower rates.

Depth. In Figure 6(b), we plot recall @ $R = 10$ for the *Holidays* data set and *FV* features, as depth is increased for a given rate point b. For $b = 1024$, we consider configurations 8192-1024, 8192-4096-1024, and 8192-4096-2048-1024 corresponding to depth $1, 2, 3$ respectively. For rate points $b = 64$ and 256, similar configurations of varying depth are chosen. We observe that, with no regularization, recall improves as depth is increased for $b = 256$ and $b = 64$, with optimal depth of 3 and 4 respectively, beyond which performance drops. At higher rates of $b = 1024$ and beyond, increasing depth does not improve as performance saturates. For hashing, a sweet spot in performance for the depth parameter is observed for each rate point, as deeper networks can cause performance to drop due to loss of information over the layers. Similar trends are obtained for recall @ $R = 100$. Importantly, we observe that with the proposed regularization, we can achieve the same performance with lower depth at each rate point. This is critical, as lower the depth, the faster the hash generation, and lower the memory requirements.

Table 1: Retrieval results before and after Siamese fine-tuning, with corresponding differences. The stacked RBM network (8192-4096-2048-64) is trained with Fisher descriptors from the ImageNet data set. Fine-tuning consistently improves retrieval results at any bit-rate.

	Layer	Recall @ R=10			Recall @ R=100		
		bef.	aft.	diff.	bef.	aft.	diff.
Holidays	4096	70.83	73.67	**2.84**	89.92	91.40	**1.48**
	2048	67.74	71.12	**3.38**	88.77	92.04	**3.27**
	64	52.06	53.04	**0.98**	80.38	83.91	**3.53**
Oxford	4096	19.38	21.73	**2.35**	41.09	45.19	**4.10**
	2048	14.32	17.23	**2.91**	36.03	41.03	**5.00**
	64	10.69	12.01	**1.32**	23.75	29.99	**6.24**
UKB	4096	79.22	82.22	**3.00**	92.04	93.73	**1.69**
	2048	75.62	79.37	**3.75**	90.79	92.82	**2.03**
	64	47.94	49.25	**1.31**	73.02	73.94	**0.92**

Fine-Tuning. Table 1 provides detailed retrieval results for a 3-layer model before and after Siamese fine-tuning. The results show consistent improvements with every training data set and at any bit-rate with a global average difference of 2.78% (up to 6.24%). The difference is more significant at higher recall rates with an average of 2.43% @ R=10 compared to 3.13% @ R=100. They are however quite comparable when relative improvement rate is considered: 7.46% @ R=10 and 7.24% @ R=100 relatively.

We notice differences across test sets with improvements on the *Oxford* set being more pronounced. The *Yandex* data set used for fine-tuning is made with matching pairs of landmark structures which can explain the better performance of the *Oxford* data set made of buildings only. The systematic improvements on all sets are nevertheless evidence of the high transferability of both unsupervised training and semi-supervised fine-tuning.

Fast Optimization with ROC Experiments. The Stanford Mobile Visual Search (*SMVS*) data set [7] contains a list of 16,319 matching image pairs, comprising a wide range of object categories. We extract FV for matching and non-matching pairs from the *SMVS* data set, hash the data to different rate points, and compute the Receiver Operating Characteristic (ROC) curve. In Figure 5(a), we plot ROC Area Under Curve (AUC) for different schemes for the *SMVS* data set. In Figure 5(b) and 5(c), we plot recall @ 10 and MAP on the *Holidays* data set. The retrieval performance of a scheme at a given database size depends on the ROC curve at different

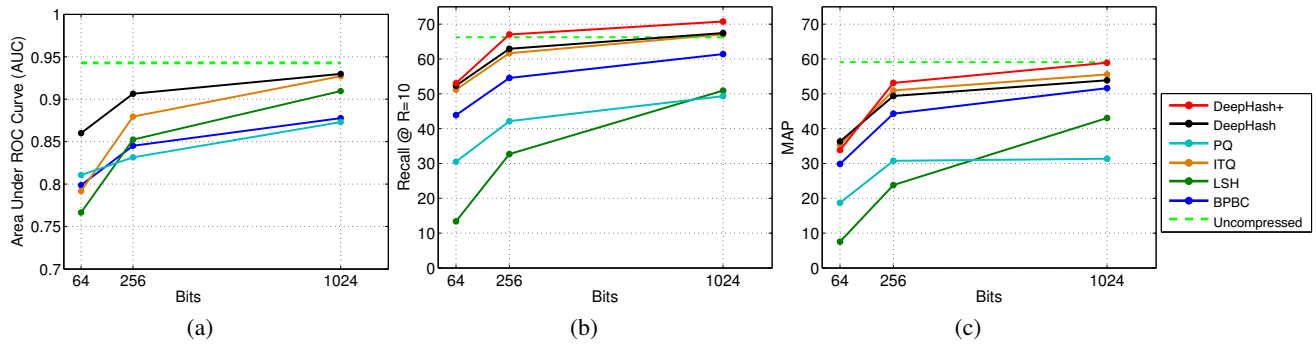


Figure 5: **Comparing AUC, Recall and MAP performance of different schemes at varying b in (a),(b) and (c) respectively. *Holidays* and FV are used for retrieval experiments, and *SMVS* for AUC. DeepHash outperforms all schemes. Also, the performance ordering of schemes is largely consistent between AUC results and retrieval results, both MAP and Recall. AUC can be used for fast optimization of parameters.**

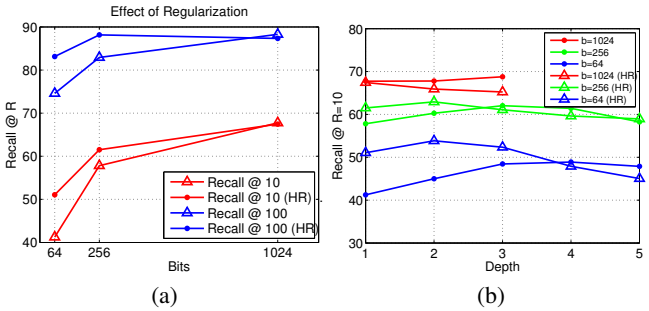


Figure 6: **Hashing FV for *Holidays*. (HR) refers to schemes trained with hashing regularization. (a) Hashing regularization improves performance significantly for single layer models 8192-b as b is decreased. (b) Recall improves as depth is increased for lower rate points $b = 64$ and $b = 256$. With regularization, we can achieve the same or better recall at lower depth.**

TPR/FPR operating points, as shown in [11]. Low FPR points are important [11, 10]. We observe that the Area Under Curve (AUC) results predict well the performance ordering (MAP and Recall) of different schemes for retrieval experiments. The retrieval and AUC experiments are performed on very different data sets, but the AUC results generalize well, and are used for fast optimization of parameters.

DeepHash Parameters. For RBM learning, we set the learning rate to 0.001 for the weight and bias parameters, momentum to 0.9, and ran the training on 150,000 images from the *ImageNet* data set for a maximum 30 epochs. Binary descriptors for the first layer are generated by subtracting the mean for each data set. For each rate point, we consider a set of models with dimensionality progressively reduced by a factor of 2 from the starting representation for FV and DCNN respectively. The best model is chosen based on greedy optimization of AUC on the *SMVS* data set, which works well as seen in detailed experimental results of Section 4.3. The chosen architectures are described in Table 2. The depth of the network increases as hash size decreases. Each target setting requires several hours to train on a modern CPU.

Bits	FV	DCNN
1024	8192-1024	4096-1024
256	8192-4096-256	4096-2048-256
64	8192-4096-2048-64	4096-2048-1024-64

Table 2: DeepHash Architecture

4.3 Retrieval Experiments

We present retrieval results using FV and DCNN features in Figure 7 and Figure 8. For instance retrieval, it is important for the relevant image to be present in the first step of the pipeline, matching global descriptors, so that a Geometric Consistency Check (GCC) [14] step can find it subsequently. We present recall @ typical operating points, R = 100 and R = 1000 for small and large data sets respectively. For *UKB* small experiments, we plot 4× recall @ $R = 4$ to be consistent with the literature. We refer to hashes before and after fine-tuning as DeepHash and DeepHash+ respectively in all figures. We refer to deep hashes based on DCNN and FV features as DCNN-DeepHash and FV-DeepHash respectively.

Performance of DeepHash. For DCNN and FV features, the proposed DeepHash outperforms state-of-the-art schemes on most data sets. The statistics of FV and DCNN features are very different. FV are dense descriptors with zero blocks corresponding to centroids not visited, while deep DCNN features tend to be sparse. Our method works well for both types of features.

For the retrieval experiments in Figure 7, there is up to 7.4% improvement in absolute recall at $b = 64$ bits compared to the second performing scheme. Up to 8.5% improvement is seen at $b = 256$, which can be a practical rate point for many applications, as there is only a marginal drop in performance for DCNN features compared to uncompressed features. Similar trends are obtained for recall @ $R = 10$ and MAP, as seen by comparing *Holidays* results in Figure 5(b),(c) and Figure 7(a), with a higher gap for larger R. Consistent trends are also obtained for the large-scale retrieval results in Figure 8.

The performance ordering of other schemes depends on the bitrate and type of feature, while DeepHash is consistent across data sets. Compared to *ITQ* scheme which applies a single PCA transform, each output bit for DeepHash is generated by a series of

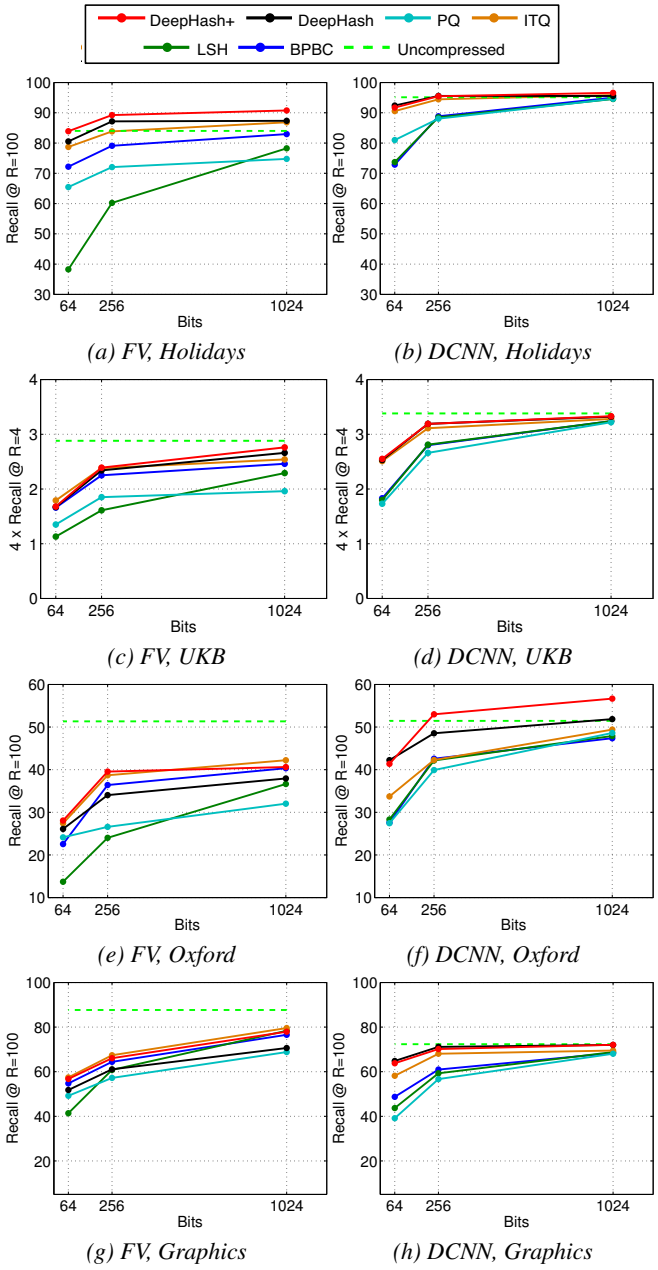


Figure 7: **Small-scale retrieval results.** *DeepHash* **outperforms other schemes by a significant margin.**

projections. The *PQ* scheme performs poorly at the low rates in consideration, as large blocks of the global descriptor are quantized with a small number of centroids, as previously observed in [16]. *LSH* performs poorly at low rates, but catches up given enough bits.

We observe a consistent improvement using Siamese fine-tuning, which learns more discriminative projections. The learnt projections generalize well, which is key for diverse retrieval tasks, thus showing the robustness of our proposed method.

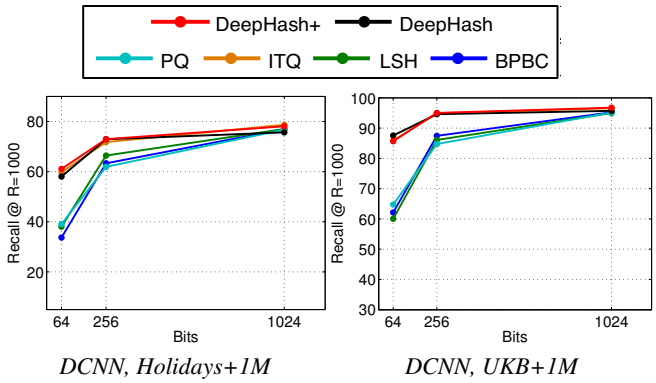


Figure 8: **Large-scale retrieval results (with 1M distractor images) for different compression schemes.** *DeepHash* **outperforms other schemes at most rate points and data sets.**

Comparing FV-DeepHash and DCNN-DeepHash. At a given rate point, DCNN-DeepHash outperforms FV-DeepHash hashes for all data sets except *Graphics* as seen by comparing across rows of Figure 7. At low rates, DCNN-DeepHash improves performance by more than 10% on some of the small data sets.

The results are data-set dependent. DCNN features are able to capture more complex low level features and have a lower starting dimensionality compared to FV. However, DCNN features have limited rotation and scale invariance, based on the level of data invariance seen at training time. FV, on the other hand, aggregate hand-crafted SIFT descriptors from scale and rotation invariant interest points, which results in a scale and rotation invariant representation. The *Graphics* data set has more objects with large variations in scale and rotation compared to the other data sets: this is one of the reasons, why peak performance of FV is higher than DCNN for *Graphics*.

Comparison to Uncompressed Descriptors. We compare the performance of DeepHash to the uncompressed descriptor in Figure 7. We obtain remarkable compression performance - at 256 bits for DCNN hashes, we only observe a marginal drop (a few%) compared to the uncompressed representation for retrieval on a wide range of data sets: a 512 $\times$ compression compared to a floating point representation, and 16 $\times$ compared to a binary representation. For FV, we can match the performance of the uncompressed descriptor with 1024 bits for *Holidays* and *UKB*, with a drop for *Graphics* and *Oxford*. At some rate points, DeepHash performs better than the uncompressed descriptor, which is probably due to quantization of noise in the uncompressed descriptor.

The instance retrieval hashing problem becomes increasingly difficult as we move towards a 64-bit hash. At 64 bits, there is a 5-10% drop in performance compared to 256 bits for DCNN features, while a drop is also observed for FV. For the million scale experiments, however, we observe a 10-20% drop in performance at 64 bits compared to 1024 bits for DCNN features.

Future Work. Improving performance further at very low-rate points like 64 bits for even larger databases is an interesting direction for future work. Studying mathematical models which relate hash size to performance for varying database size is also an exciting direction to pursue. Finally, in this work, we learnt compact

hashes starting from a pre-trained DCNN model. Learning the hash directly from pixels in a DCNN framework might lead to further improvements [33].

5 CONCLUSIONS

A perfect image hashing scheme would convert a high-dimensional descriptor into a low-dimensional bit representation without losing retrieval performance. We believe that DeepHash, which focuses on achieving complex hash functions with deep learning, is a significant step in this direction. Our method is focused on a deep network which efficiently utilizes the binary subspace through hashing regularization and further fine-tuning using a Siamese training algorithm. Through a rigorous evaluation process, we show that our model performs well across various data sets, regardless of the type of image descriptors used, sparse or dense. The improvement over existing hashing schemes attests to the importance of regularization, depth and fine-tuning for hashing image descriptors.

6 ACKNOWLEDGMENTS

This work was partially supported by grants from National Natural Science Foundation of China (U1611461, 61661146005) and National Hightech R&D Program of China (2015AA016302).

REFERENCES

[1] DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[2] *MPEG CDVS (Compact Descriptors for Visual Search) Benchmark. Stanford Digital Repository*. http://purl.stanford.edu/qy869qz5226.

[3] Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[4] A. Babenko and V. Lempitsky. Aggregating local deep features for image retrieval. In *International Conference on Computer Vision (ICCV)*, 2015.

[5] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural Codes for Image Retrieval. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2014.

[6] Jane Bromley, Isabelle Guyon, Yann Lecun, Eduard Sckinger, and Roopak Shah. Signature Verification using a "Siamese" Time Delay Neural Network. In *Advances in Neural Information Processing Systems (NIPS)*, 1994.

[7] V. Chandrasekhar, D.M.Chen, S.S.Tsai, N.M.Cheung, H.Chen, G.Takacs, Y.Reznik, R.Vedantham, R.Grzeszczuk, J.Back, and B.Girod. Stanford Mobile Visual Search Data Set. In *Proceedings of ACM Multimedia Systems Conference (MMSys), San Jose, California, February 2011*.

[8] V. Chandrasekhar, M. Makar, G. Takacs, D.M. Chen, S. S. Tsai, N. M. Cheung, R. Grzeszczuk, Y. Reznik, and B. Girod. Survey of SIFT Compression Schemes. In *Proceedings of International Mobile Multimedia Workshop (IMMW), IEEE International Conference on Pattern Recognition (ICPR)*, Istanbul, Turkey, August 2010.

[9] V. Chandrasekhar, G. Takacs, D. M. Chen, S. S. Tsai, R. Grzeszczuk, and B. Girod. CHoG: Compressed Histogram of Gradients - A Low Bit Rate Feature Descriptor. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2504–2511, Miami, Florida, June 2009.

[10] D. M. Chen, S. S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod. Residual Enhanced Visual Vector as a Compact Signature for Mobile Visual Search. In *Signal Processing, Elsevier, In Press*, June 2012.

[11] David M. Chen. Memory-Efficient Image Databases for Mobile Visual Search. *Ph.D. thesis, Department of Electrical Engineering, Stanford University*, April 2014.

[12] Sumit Chopra, Raia Hadsell, and Yann Lecun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–546. IEEE Press, 2005.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[14] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of ACM*, 24(6):381–395, June 1981.

[15] Hanlin Goh, Nicolas Thome, Matthieu Cord, and Joo-Hwee Lim. Unsupervised and supervised visual codes with restricted Boltzmann machines. In *European Conference on Computer Vision (ECCV)*, 2012.

[16] Yunchao Gong, Sanjiv Kumar, Henry Rowley, and Svetlana Lazebnik. Learning Binary Codes for High-Dimensional Data Using Bilinear Projections. In *Proceedings of CVPR*, pages 484–491, 2013.

[17] Yunchao Gong and S. Lazebnik. Iterative Quantization: A Procrustean Approach to Learning Binary Codes. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 817–824, Washington, DC, USA, 2011.

[18] Kristen Grauman and Rob Fergus. Learning Binary Hash Codes for Large-Scale Image Search. In *Machine Learning for Computer Vision*, volume 411, pages 49–87. 2013.

[19] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1735–1742, 2006.

[20] Geoffrey Hinton. A Practical Guide to Training Restricted Boltzmann Machines. In *Neural Networks: Tricks of the Trade*, volume 7700 of *LNCS*, pages 599–619. Springer Berlin Heidelberg, 2012.

[21] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[22] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief networks. *Neural Computation*, 18(7):1527–1554, 2006.

[23] H. Jégou, M. Douze, and C. Schmid. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 304–317, Berlin, Heidelberg, October 2008.

[24] H. Jégou, M. Douze, C. Schmid, and P. Perez. Aggregating Local Descriptors into a Compact Image Representation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3304–3311, San Francisco, CA, USA, June 2010.

[25] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716, September 2012.

[26] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[28] Brian Kulis and Kristen Grauman. Kernelized Locality-Sensitive Hashing for Scalable Image Search. In *Proceedings of IEEE International Conference on Computer Vision (ICCV*, 2009.

[29] Honglak Lee, Chaitanya Ekanadham, and Andrew Ng. Sparse deep belief net model for visual area V2. In *Advances in Neural Information Processing Systems (NIPS)*, pages 873–880, 2008.

[30] Youngwoon Lee. Spherical Hashing. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2957–2964, Washington, DC, USA, 2012. IEEE Computer Society.

[31] Xi Li, Guosheng Lin, Chunhua Shen, Anton van den Hengel, and Anthony R. Dick. Learning Hash Functions Using Column Generation. In *Proceedings of ICML*, volume 28, pages 142–150, 2013.

[32] Jie Lin, Ling-Yu Duan, Tiejun Huang, and Wen Gao. Robust Fisher Codes for Large Scale Image Retrieval. In *Proceedings of International Conference on Acoustics and Signal Processing (ICASSP)*, 2013.

[33] K. Lin, J. Lu, C. Chen, and J. Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[34] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised Hashing with Kernels. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[35] D. Lowe. Distinctive Image Features from Scale-invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[36] B. Thomee Mark J. Huiskes and Michael S. Lew. New Trends and Ideas in Visual Concept Detection: The MIR Flickr Retrieval Evaluation Initiative. In *Proceedings of the 2010 ACM International Conference on Multimedia Information Retrieval*, pages 527–536, New York, NY, USA, 2010.

[37] Vinod Nair and Geoffrey Hinton. 3D Object Recognition with Deep Belief Nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1339–1347, 2009.

[38] D. Nistér and H. Stewénius. Scalable Recognition with a Vocabulary Tree. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2161–2168, New York, USA, June 2006.

[39] Mohammad Norouzi and David Fleet. Minimal Loss Hashing for Compact Binary Codes. In *Proceedings of ICML*, pages 353–360, 2011.

[40] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. Large-scale Image Retrieval with Compressed Fisher Vectors. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3384–3391, San Francisco, CA, USA, June 2010.

[41] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *Proceedings of CVPR*, pages 1–8, Minneapolis, Minnesota, June 2007.

[42] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann Machines for Collaborative Filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 791–798, New York, NY, USA, 2007. ACM.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[44] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. June 2014.

[45] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of CNN activations. In *International Conference on Learning Representations (ICLR)*, 2016.

[46] A. Torralba, R. Fergus, and Y. Weiss. Small Codes and Large Image Databases for Recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, Alaska, June 2008.

[47] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. June 2014.

[48] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-Supervised Hashing for Scalable Image Retrieval. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, USA, June 2010.

[49] Jun Wang, Wei Liu, Andy X Sun, and Yu-Gang Jiang. Learning Hash Codes with Listwise Supervision. In *Proceedings of IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 3032–3039, 2013.

[50] Y. Weiss, A. Torralba, and R. Fergus. Spectral Hashing. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 1753–1760, Vancouver, BC, Canada, December 2008.

[51] C. Yeo, P. Ahammad, and K. Ramchandran. Rate-efficient Visual Correspondences using Random Projections. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pages 217–220, San Diego, California, October 2008.