ENHANCING FEATURE DISCRIMINATION FOR UNSUPERVISED HASHING

Tuan Hoang* *Thanh-Toan Do*† *Dang-Khoa Le Tan** *Ngai-Man Cheung**

*Singapore University of Technology and Design (SUTD)
†The University of Adelaide

ABSTRACT

We introduce a novel approach to improve unsupervised hashing. Specifically, we propose a very efficient embedding method: *Gaussian Mixture Model embedding (Gemb)*. The proposed method, using Gaussian Mixture Model, embeds feature vector into a low-dimensional vector and, simultaneously, enhances the discriminative property of features before passing them into hashing. Our experiment shows that the proposed method boosts the hashing performance of many state-of-the-art, e.g. Binary Autoencoder (BA) [1], Iterative Quantization (ITQ) [2], in standard evaluation metrics for the three main benchmark datasets.

Index Terms— Embedding, Hashing, Discrimination enhancement, Gaussian mixture model

1. INTRODUCTION

Earlier hashing methods [2, 1, 3] use hand-crafted global features, e.g. GIST [4]. Recently, the convolutional neural networks (CNNs) have emerged as the state-of-art method for global descriptors in image retrieval task [5, 6]. These CNN descriptors inherit the highly discriminative property in visual recognition task. Therefore, we hypothesize that: using the image descriptors based on the activations of CNNs can boost the performance of state-of-the-art hashing methods, compared to hand-crafted features. Note that image descriptors from the fully-connected network layers have been evaluated for hashing [7, 8]. However, these works only focus on evaluating the hashing performance of their proposed methods; they do not make any explicit comparison between using hand-crafted and CNN descriptors for hashing.

Given the possibility that the discriminative CNN descriptors can increase the hashing performance of various methods, we delve deeper into the problem: "*How can we further enhance discrimination of the features for hashing purpose?*" Inspired by state-of-the-art embedding methods including Vector of Locally Aggregated Descriptors [9], Function Approximation-based Embedding [10, 11], Fisher Vector [12], we propose a method to enhance the feature discrimination. In particular, different from those embedding methods which transform an incoming variable-size set of independent samples (e.g. SIFT features [13]) into a higher dimensional fixed size vector representation, our proposed method maps a single descriptor into a lower dimensional fixed-size vector representation.

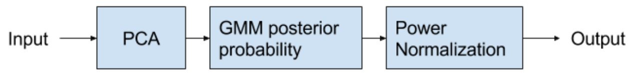


Fig. 1: Our proposed Gemb method: The inputs are global descriptors such as GIST or CNN, and outputs are the embedding features for hashing.

Contribution. In this paper, we address the problem of producing compact but very discriminative features [9]. Our proposed embedding method improves hashing performance. Specifically, to the best of our knowledge, our work is the first to propose to improve unsupervised hashing performance by an explicit embedding step to enhance feature discrimination. Our experiments show that our embedding method in combination with ITQ [2] consistently outperforms other state-of-art hashing methods in several evaluation metrics and various datasets.

The remaining of this paper is organized as follows. Section 2 introduces the proposed method in detail. Section 3 presents the settings and results of experiments. Finally, we conclude the paper in Section 4.

2. PROPOSED METHOD

In this section, we describe the details of our proposed method with three main steps. In the first step (Section 2.1), we preprocess descriptors by Principal Component Analysis (PCA). The main novelty of our method is the second step (Section 2.2), in which we attempt to embedding data to lower dimensional space and enhancing feature discrimination simultaneously, using the posterior probabilities of Gaussian Mixture Model (GMM). In the last step (Section 2.3), the embedding features are post-processed by Power Normalization to make them more robust to $l2$-distance similarity [14].

2.1. Dimensionality reduction

Our input is a set of m high-dimensional data points: $\tilde{X} = \{\tilde{x}^{(1)}, \tilde{x}^{(2)}, ..., \tilde{x}^{(m)}\}, \tilde{x}^{(i)} \in \mathbb{R}^d$. To reduce the computational cost and enhance the discriminative property, we want to produce a compact feature in which the variance of each variable is maximized, the variables are pairwise uncorrelated, and noise and redundancy are removed. This can be accomplished by PCA. Note that PCA also facilitates our second step, as will be discussed.

We need to decide the number of PCA components to retain. In this work, we choose the number of retained components based on the percentage of variance retained γ, $0 \leq \gamma \leq 1$:

$$\frac{\sum_{j=1}^{D} \lambda_j}{\sum_{j=1}^{n} \lambda_j} \geq \gamma. \tag{1}$$

Here $\lambda_1, ..., \lambda_n$ are the eigenvalues (sorted in decreasing order) of the covariance matrix $S = \frac{1}{m}\sum_{i=1}^{m}(\tilde{x}^{(i)})(\tilde{x}^{(i)})^T$. The number of retained PCA components D is the smallest value that satisfies the inequality (1). Then, the compressed features can be obtained by: $X = U^T\tilde{X}$, where U is the matrix of eigenvectors corresponding to the top-D eigenvalues of S in columns.

Specific to our method, reducing the dimension of the data can also reduce the complexity of the hypothesis class considered and help avoid overfitting in learning a GMM (Section 2.2). Pairwise uncorrelated variables is also a desirable property to avoid ill-condition covariance matrix in fitting GMM.

2.2. Posterior probability as embedding features

Let $\lambda = \{w_i; \mu_i; \Sigma_i; i = 1...N\}$ be the set of Gaussian Mixture Model (GMM) parameters learned from the compressed data $X = \{x^{(1)}, x^{(2)}, ..., x^{(m)}\}; x^{(i)} \in \mathbb{R}^D$. Specifically, w_i, μ_i and Σ_i denote respectively the weight, mean vector and covariance matrix of the i-th Gaussian in the model of total N Gaussians.

The posterior probability captures the strength of relationship between a sample $x^{(t)}$ and a Gaussian model $\mathcal{N}(\mu_j, \Sigma_j)$. It is given by:

$$P(j|x^{(t)}, \mu_j, \Sigma_j) = \frac{w_j p_j(x^{(t)}|\mu_j, \Sigma_j)}{\sum_{i=1}^{N} w_i p_i(x^{(t)}|\mu_i, \Sigma_i)} \tag{2}$$

Here $p_j(x^{(t)}|\mu_j, \Sigma_j)$ is the probability of $x^{(t)}$ given the j-th Gaussian distribution:

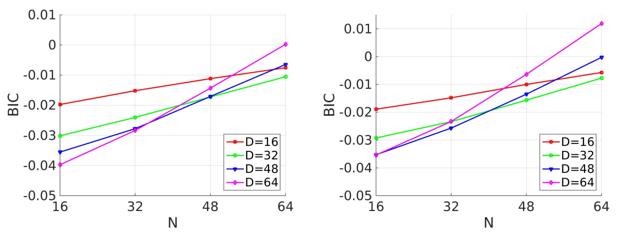
$$p_j(x^{(t)}|\mu_j, \Sigma_j) = \frac{\exp(-\frac{1}{2}(x^{(t)} - \mu_j)^T \Sigma_j^{-1}(x^{(t)} - \mu_j))}{(2\pi)^{D/2}|\Sigma_j|^{1/2}} \tag{3}$$

where $|.|$ denotes the determinant operator.

We propose to construct the embedding feature of a sample $x^{(t)}$ by:

$$z^{(t)} = \left[P(j|x^{(t)}, \mu_j, \Sigma_j); j = 1...N\right] \tag{4}$$

In our proposed method, since we do not have the diagonal constraint on the covariances as in Fisher Vector encoding [12], we decide the use full covariances in the GMM to achieve better mixture models. With full covariances instead of diagonal covariances, it is possible to increase the likelihood. However, there is a risk of overfitting as there are more



(a) CIFAR-10 [16] (b) LabelMe-12-50k [17]

Fig. 2: The relative difference between BIC values of GMM with full covariances and GMM with diagonal covariances using VGG-FC7 descriptors of CIFAR-10 and LabelMe-12-50k dataset[1]. A negative difference means that the former one is better under BIC.

parameters to be estimated. Therefore, in order to make a systematic comparison between GMM with full and diagonal covariances, we utilize the Bayesian Information Criterion (BIC) [15] as this criterion reduces the risk of overfitting by introducing the penalty term on the number of parameters. BIC results are shown in Fig.2. We observe that with a small N, e.g. 16, using full covariances can help achieve much better mixture models under BIC (or smaller BIC). At a larger N, e.g. 64, the BIC values of GMM with diagonal covariances are more comparable to, or even higher than (at large D) those of GMM with full covariances. Therefore, full covariance is preferable.

2.3. Unsparsifying by Power Normalization

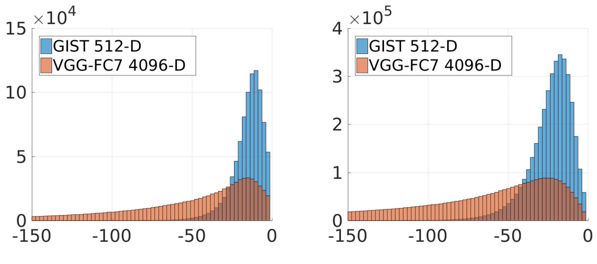
Similar to the finding in [14], we observe that as the number of Gaussians increases, the embedding features become sparser. The distributions of embedding features (log-scale) in Fig. 3 shift to more negative regions as N increases. This effect can be explained: as the number of Gaussians increases, it is easier for a embedding feature $z^{(t)}$ to fit in a distribution with high probability.

Additionally, we also observe in Fig. 3 that more discriminative global descriptors $\tilde{x}^{(t)}$ result in sparser embedding features $z^{(t)}$. This property can be explained: with more discriminative descriptors, the descriptors of semantically similar images tend to be very close in the $l2$-space, while those of semantically different images tend to locate far away from each other. Consequently, GMM groups the semantically similar descriptors into the same clusters with high probability.

With sparse vectors, the $l2$-distance is a poor measurement of similarity. Therefore, we follow [14] to "*unsparsify*" the embedding features by applying the Power Normalization function (5), so that we can still use the $l2$-distance similarity. In particular, the $l2$-distance similarity is desirable as hashing methods preserve the similarity between Euclidean and Hamming distances.

$$f(z) = \text{sign}(z)|z|^{\alpha} \tag{5}$$

[1]Please refer to Section 3.1 for detail information about the datasets.

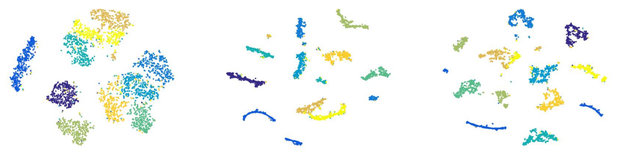


(a) $N = 16$ (b) $N = 64$

Fig. 3: Histogram of embedding features (in log scale) for CIFAR-10 [16] using GIST 512-D and VGG-FC7 4096-D descriptors[1]. Horizontal-axis is in log-scale.

2.4. Visualize descriptors

We utilize t-SNE [18] to visualize (Fig. 4) the scatter plot of GIST 512-D descriptors of a subset of MNIST dataset. We can clearly observe that after processing with Gemb with a certain number of Gaussians (Fig. 4b, 4c), the descriptors of the same class locate closer. Furthermore, the boundaries between different classes are clearer. These suggest that Gemb can lead to more discriminative features.



(a) Without Gemb (b) Gemb ($N = 32$) (c) Gemb ($N = 64$)

Fig. 4: Visualizing GIST descriptors of a subset of MNIST dataset[1]. (a): without Gemb. (b, c): with Gemb. Different colors correspond to different classes. Best viewed in color.

3. EXPERIMENTS

Firstly, we conduct comprehensive experiments to show that, in comparison with GIST 512-D descriptors [4], descriptors from CNN (e.g. VGG-FC7 [19]) can significantly boost the retrieval performance of many state-of-the-art unsupervised hashing methods: Spectral Hashing (SH) [20], Spherical Hashing (SpH) [21], Iterative Quantization (ITQ) [2], and Binary Autoencoder (BA) [1] (Section 3.2). More importantly, we demonstrate that our proposed method in combination with ITQ and BA further enhance the hashing quality for both GIST hand-crafted features and CNN descriptors (Section 3.3).

3.1. Dataset, Evaluation protocol, and Implementation notes

The **CIFAR-10 dataset** [16] contains 60,000 fully-annotated color images of 32×32 from 10 object classes. We randomly sampled 10% images per class, as the query data, and used the remaining images as the training set and retrieval database. This sampling strategy is applied to all other datasets to handle unbalanced samples in different classes.

The **LabelMe-12-50k dataset** [17] is a subset of LabelMe dataset [22]. The LabelMe-12-50k dataset includes 50,000 fully-annotated color images of 256×256 of 12 classes. In this dataset, for images which have more than one label values in $[0.0, 1.0]$, we choose the object class corresponding to the largest label value as image labels.

The **MNIST dataset** [23] consists of 70,000 fully-annotated grayscale handwritten digit images of 28×28 from 10 classes.

Evaluation protocols. To evaluate retrieval performance of methods, we apply three common metrics: 1) mean Average Precision (***mAP***); 2) precision of Hamming radius of 2 (***precision@r2***) which measures precision on retrieved images having Hamming distance to query ≤ 2 (we report zero precision for the queries that return no image); 3) precision at top 1000 return images (***precision@1k***) which measures the precision on the top 1000 retrieved images. Class labels are used as ground truths for all evaluation. Additionally, to avoid biased results due to unbalanced samples of different classes in query sets, we calculate the average results for all classes.

Implementation notes. In our method, the two parameters $\{\gamma, \alpha\}$ is empirically set as $\{0.85, 0.15\}$ and $\{0.65, 0.05\}$ for GIST 512-D [4] and VGG-FC7 [2] [19] descriptors respectively. The number of Gaussians N is set as the number of hashing bits.

For all compared methods [1, 2, 20, 21], we use the implementations and the suggested parameters provided by the authors. Besides, in order to improve the statistical stability in the results, we execute the experiments 5 times for all methods and report the average values.

3.2. Hand-crafted descriptor vs. CNN descriptor

Table 1 and Table 2 compare the hashing performance of different methods [20, 21, 2, 1] using GIST 512-D descriptors and the activation of the fully connected layer of VGG (VGG-FC7). As shown in the tables, all hashing methods consistently achieve higher performances in the majority of evaluation metrics when using the VGG-FC7 descriptors in comparison with GIST 512-D descriptors.

3.3. Evaluate Gemb

In order to evaluate Gemb for hashing purpose, we combine Gemb with BA and ITQ[3] and conduct experiments on CIFAR-10, LabelMe-12-50k, and MNIST[4] datasets. We report experimental results for these datasets on Table 1, Table 2, and Table 3 respectively. Gemb clearly helps to boost

[2]Descriptors are extracted from the Fully Connected layer 7 of VGG.

[3]Due to space constraint, we only evaluate our Gemb with BA and ITQ. However, Gemb also works well with other hashing methods [20, 21, 24, 7].

[4]We do not evaluate the descriptor from VGG for MNIST dataset since VGG is trained on RGB images while MNIST is grayscale.

Table 1: Results on the CIFAR-10 [16] dataset. We report the results using GIST 512-D descriptor [4] on the top section and using VGG-FC7 descriptor [19] on the bottom section.

Methods	mAP			precision@1k		precision@r2	
	16	32	64	16	32	16	32
SH [20]	12.88	12.71	12.99	17.32	17.69	18.38	21.00
SpH [21]	14.46	15.13	15.88	19.38	21.30	21.20	13.86
BA [1]	15.34	16.86	17.74	21.64	24.30	24.65	13.67
ITQ [2]	16.59	17.42	18.02	22.36	24.49	23.84	18.24
Gemb+BA	20.80	22.20	22.45	28.86	**32.86**	**28.31**	32.58
Gemb+ITQ	**21.36**	**22.44**	**22.59**	**29.02**	31.48	28.25	**32.97**
SH [20]	18.31	16.54	15.78	28.61	26.74	32.90	18.95
SpH [21]	18.82	20.93	23.40	27.33	31.60	30.34	22.33
BA [1]	25.38	26.16	27.99	36.45	38.19	**39.58**	25.54
ITQ [2]	26.82	27.38	28.73	**37.08**	38.86	38.83	29.53
Gemb+BA	27.24	28.52	29.97	36.37	**39.64**	35.45	**42.05**
Gemb+ITQ	**27.61**	**29.12**	**30.01**	35.51	38.36	34.03	41.26

Table 2: Results on the LabelMe-12-50k dataset [17]. We report the results using GIST 512-D descriptor [4] on the top section and using VGG-FC7 descriptor [19] on the bottom section.

Methods	mAP			precision@1k		precision@r2	
	16	32	64	16	32	16	32
SH [20]	10.74	10.76	10.95	13.73	13.82	14.52	18.54
SpH [21]	11.86	13.02	13.67	15.17	17.07	16.96	13.89
BA [1]	14.21	14.55	15.43	18.93	19.97	19.92	12.97
ITQ [2]	15.07	16.06	16.58	18.83	20.31	20.24	19.43
Gemb+BA	19.95	21.64	21.69	24.43	**26.33**	24.56	27.92
Gemb+ITQ	**20.79**	**21.69**	**21.92**	**24.77**	26.01	**24.75**	**28.67**
SH [20]	12.60	12.59	12.24	17.23	17.20	20.32	15.23
SpH [21]	13.59	15.10	17.03	17.81	20.08	20.39	14.29
BA [1]	16.96	18.42	20.80	21.99	23.85	25.80	15.40
ITQ [2]	18.06	19.40	20.71	23.13	24.84	**26.30**	19.54
Gemb+BA	22.63	24.05	24.19	**27.15**	28.47	25.95	**30.77**
Gemb+ITQ	**23.37**	**24.26**	**25.37**	25.65	**28.92**	24.91	29.38

performance of BA and ITQ in majority of evaluation metrics, ***mAP***, ***precision@1k*** and ***precision@r2***, for various code lengths and datasets. Furthermore, Gemb+ITQ consistently achieves the best ***mAP*** among all compared methods.

Comparison with Hashing method using Deep Neural Network (DNN). Recently, there are several methods [3, 7, 25] to apply DNN to learn binary hash code. These methods achieve very competitive performances.

- ***Deep Hasing (DH) [3]*** and ***Unsupervised Hashing with Binary Deep Neural Network (UH-BDNN) [7].*** Following the experiments settings in [3] and [7], we conduct experiments on CIFAR-10 to make a fair comparison. In this experiment, 100 images are randomly sampled for each class as query set; the remaining images are for training and database query. The images are presented by GIST 512-D descriptors. In addition, to avoid bias results due to test samples, we repeat the experiment 5 times with 5 different random test sets. The comparative results in term of ***mAP*** and ***precision@r2*** are presented in Table 4. Clearly, Gemb + ITQ consistently outperforms DH and UH-BDNN.

Table 3: Results on the MNIST [23] dataset. We report the results using GIST 512-D descriptors [4].

Methods	mAP			precision@1k		precision@r2	
	16	32	64	16	32	16	32
SH [20]	32.59	33.23	30.65	56.23	61.03	60.12	78.37
SpH [21]	31.27	36.80	41.40	51.28	62.17	57.66	68.62
BA [1]	48.48	51.72	52.73	70.83	76.23	75.17	74.70
ITQ [2]	46.37	50.59	53.69	69.29	75.85	70.66	82.06
Gemb+BA	79.35	82.59	83.13	85.66	92.74	84.56	92.56
Gemb+ITQ	**79.97**	**83.81**	**83.72**	**86.09**	**93.33**	**85.13**	**93.23**

Table 4: Comparison with Deep Hashing (DH) [3] and Unsupervised Hashing with Binary Deep Neural Network (UH-BDNN) [7]. The results of DH and UH-BDNN are cited from [3] and [7] respectively.

Methods	CIFAR10			
	mAP		precision@r2	
	16	32	16	32
DH [3]	16.17	16.62	23.33	15.77
UH-BDNN [7]	17.83	18.52	24.97	18.85
Gemb+ITQ	**21.14**	**22.27**	**28.60**	**31.18**

- ***DeepBit [25].*** Similarly, we compare Gemb + ITQ with DeepBit on CIFAR-10. In this experiment, since DeepBit utilized pretrained VGG to learn the compact binary code, we present each image by a VGG-FC7 descriptor. We report the ***mAP*** of the top 1,000 returned images in Table 5 as in [25]. The results clearly show that Gemb + ITQ outperforms DeepBit by a fair margin.

Table 5: Comparison with DeepBit [25]. The results of DeepBit are cited from [25].

Methods	16	32	64
DeepBit [25]	19.43	24.86	27.73
Gemb+ITQ	**37.50**	**41.00**	**44.27**

4. CONCLUSION

We have discussed a novel approach to improve unsupervised hashing by an explicit embedding step to enhance feature discrimination. In particular, we have discussed our proposed Gemb method using GMM posterior probabilities. Our method attempts to embed a high-dimensional global descriptor into a low-dimensional space and, simultaneously, enhance the embedding feature discrimination. The solid experimental results on three benchmark datasets have demonstrated that Gemb can help to enhance feature discrimination and boost the performances of different hashing methods.

References

[1] Miguel Á. Carreira-Perpiñán and Ramin Raziperchikolaei, "Hashing with binary autoencoders," in *CVPR*, 2015.

[2] Yunchao Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *CVPR*, 2011.

[3] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou, "Deep hashing for compact binary codes learning," in *CVPR*, 2015.

[4] Aude Oliva and Antonio Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *IJCV*, pp. 145–175, 2001.

[5] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky, "Neural codes for image retrieval," in *ECCV*, 2014.

[6] Y. Gong, R. Guo L. Wang, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *ECCV*, 2014.

[7] Thanh-Toan Do, Anh-Dzung Doan, and Ngai-Man Cheung, "Learning to hash with binary deep neural network," in *ECCV*, 2016.

[8] Y. Li, Y. Xu, Z. Miao, H. Li, J. Wang, and Y. Zhang, "Deep feature hash codes framework for content-based image retrieval," in *International Conference on Wireless Communications Signal Processing (WCSP)*, 2016.

[9] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, 2010.

[10] Thanh-Toan Do, Quang Tran, and Ngai-Man Cheung, "FAemb: a function approximation-based embedding method for image retrieval," in *CVPR*, 2015.

[11] Thanh-Toan Do and Ngai-Man Cheung, "Embedding based on function approximation for large scale image search," *TPAMI*, 2017.

[12] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *CVPR*, 2007.

[13] David G. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, 1999.

[14] Florent Perronnin, Jorge Sánchez, and Thomas Mensink, "Improving the fisher kernel for large-scale image classification," in *ECCV*, 2010.

[15] Ernst Wit, Edwin van den Heuvel, and Jan-Willem Romeijn, ""All models are wrong...": an introduction to model uncertainty," *Statistica Neerlandica*, vol. 66, no. 3, pp. 217–236, 2012.

[16] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," in *Technical report, University of Toronto*, 2009.

[17] Rafael Uetz and Sven Behnke, "Large-scale object recognition with cuda-accelerated hierarchical neural networks," in *IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, 2009.

[18] Laurens van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne," *The Journal of Machine Learning Research*, vol. 9, no. 2579-2605, pp. 85, 2008.

[19] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, 2014.

[20] Yair Weiss, Antonio Torralba, and Rob Fergus, "Spectral hashing," in *NIPS*, 2009.

[21] J. P. Heo, Y. Lee, J. He, S. F. Chang, and S. E. Yoon, "Spherical hashing," in *CVPR*, 2012.

[22] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman, "Labelme: A database and web-based tool for image annotation," *IJCV*, pp. 157–173, 2008.

[23] Yann LeCun and Corinna Cortes, "MNIST handwritten digit database," 2010.

[24] Kaiming He, Fang Wen, and Jian Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *CVPR*, 2013.

[25] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *CVPR*, 2016.