APHASH: ANCHOR-BASED PROBABILITY HASHING FOR IMAGE RETRIEVAL

Junjie Chen[1] *Anran Wang*[2] *William K. Cheung*[1]

[1] Computer Science Department, Hong Kong Baptist University
[2] Institute for Infocomm Research, A*STAR, Singapore

ABSTRACT

In this paper, we propose a novel unsupervised hashing method called Anchor-based Probability Hashing (APHash) to preserve the similarities by exploiting the distribution of data points. In particular, distances are transformed into probabilities in both original and hash code spaces. Our method aims to learn hash codes which minimize the mismatch between probability distributions of these two spaces. To address the high complexity issue, our method randomly selects a set of anchors and constructs asymmetric probability matrices. In this way, APHash can make use of the correlation between anchors and data points to learn hash codes more efficiently. Experimental results on two benchmark datasets demonstrate the effectiveness of the proposed APHash method, outperforming state-of-the-art hashing approaches in the application of image retrieval.

Index Terms— Hashing, Image Retrieval, Unsupervised Learning

1. INTRODUCTION

With the availability of large volumes of high-dimensional image data, large-scale visual search has attracted extensive research attention in computer vision and machine learning communities [1, 2, 3, 4]. To tackle heavy computational cost with high-dimensional real-valued descriptors, hashing methods have been proposed. The target of hashing is to encode high-dimensional feature vectors into short binary hash codes while preserving similarities of interest. By using binary hash codes, significant reduction on both storage and computation complexity can be achieved.

In this paper, we focus on unsupervised hashing. This category of methods makes the Hamming distance of learned hash codes pairs and the corresponding similarity (typically computed with Euclidean distances) given in the original space as consistent as possible. Among various unsupervised hashing methods, graph-based hashing methods are very representative, including Spectral Hashing [5], Anchor Graph Hashing [6], Discrete Graph Hashing [7] and Scalable Graph Hashing [8]. Typically, the graph-based hashing methods can be considered as a Laplacian Eigenmap problem. Other existing methods like Iterative Quantization (ITQ) [9] attempt to obtain binary hash codes by imposing optimized rotation on the principal vectors. Ordinal relation is also considered in the hash code learning process. Ordinal Embedding Hashing (OEH) [10] was proposed to preserve the relative orders among data points in the Hamming space.

Instead of treating it an Laplacian Eigenmap or ordinal preserving problem, we propose a novel unsupervised method to preserve the similarities of the original space in the learned hash codes by exploiting the distribution of data points, which we refer as Anchor-based Probability Hashing (APHash). In the proposed framework, distances between data points in the original space and the hash code space are transformed into probability distributions $\mathcal{P}$ and $\mathcal{Q}$ that represent data similarities. If derived hash codes correctly capture the similarities of the original data space, the mismatch between $\mathcal{P}$ and $\mathcal{Q}$ should be minimized.

However, assume there are n data points, $\mathcal{P}$ and $\mathcal{Q}$ will be represented by $n \times n$ probability matrices, and the complexity will be $\mathcal{O}(n^2)$ as in SePH [11]. Thus, it is hard to employ this configuration for the image retrieval task, as the number of data points is typically large. To overcome this obstacle, we propose to randomly select a set of m anchors and construct asymmetric probability matrices of size $m \times n$ to represent the the probability distributions $\mathcal{P}$ and $\mathcal{Q}$. In this way, our APHash method makes use of the correlation between anchors and data points to learn hash codes more efficiently. Extensive experiments are conducted on publicly available CIFAR-10 [12] and Youtube Faces [13] datasets and the results demonstrate that our proposed APHash method can achieve superior performance over the state-of-the-art hashing approaches.

2. ANCHOR-BASED PROBABILITY HASHING

2.1. Problem Formulation

Distribution-preserving Loss

To capture the data structure in the original space, we propose a novel method called Anchor-based Probability Hashing (APHash) which aims to preserve the distribution of data points. Specifically, Euclidean distances between data points are transformed into probability distribution $\mathcal{P}$ which represent similarities in the original space. Similarly, Hamming

distances in the hash code space are transformed into probability distribution $\mathcal{Q}$. Assume we are given n data points, $\mathcal{P}$ and $\mathcal{Q}$ will be represented by probability matrices of $n \times n$. The data distribution can be preserved by minimizing the mismatch between $\mathcal{P}$ and $\mathcal{Q}$.

However, there is high complexity issue under this configuration, where the complexity grows quadratically with increasing of the number of data points. To address this issue, we select a set of m anchors $C = \{\mathbf{c}_i\}_{i=1}^m$ from the whole training set containing n data points $X = \{\mathbf{x}_i\}_{i=1}^n$ and construct asymmetric probability matrices $\mathcal{P}$ and $\mathcal{Q}$ of $m \times n$. Here we randomly select the anchors for simplicity. Note that m is much smaller than n (i.e. $m \ll n$). In this way, APHash can make use of the correlations between anchors and all the data points to learn hash codes which preserve the data structure. In our APHash method, KL-divergence is utilized to measure the difference between these two probability distributions.

In the original space, to derive the probability distribution $\mathcal{P}$, we define $p_{j|i}$ as the probability of assigning data point $\mathbf{x}_j$ to anchor $\mathbf{c}_i$, in other words, $p_{j|i}$ indicates the similarity between data point $\mathbf{x}_j$ and anchor $\mathbf{c}_i$. $p_{j|i}$ is represented as follows:

$$p_{j|i} = \begin{cases} 1, & \text{if} \quad d(\mathbf{c}_i, \mathbf{x}_j) \leq \theta \\ 0, & \text{if} \quad d(\mathbf{c}_i, \mathbf{x}_j) > \theta \end{cases} \tag{1}$$

where θ is the threshold indicating the average distance between anchor $\mathbf{c}_i$ and its k nearest neighbors:

$$\theta = \frac{\sum_{j \in \mathcal{N}_k(\mathbf{c}_i)} d(\mathbf{c}_i, \mathbf{x}_j)}{k} \tag{2}$$

where $d(\mathbf{c}_i, \mathbf{x}_j) = \|\mathbf{c}_i - \mathbf{x}_j\|_2^2$ denotes the Euclidean distance between anchor $\mathbf{c}_i$ and data point $\mathbf{x}_j$, and $\mathcal{N}_k(\mathbf{c}_i)$ denotes the set of k nearest neighbors of anchor $\mathbf{c}_i$. To some extent, the threshold θ can help filter out those remote data points within these k nearest neighbors.We normalize the probabilities such that the sum of probabilities related to each anchor is 1, that is $\sum_{j=1}^n p_{j|i} = 1$.

Similarly, we define a probability distribution $\mathcal{Q}$ in the hash code space using Hamming distances. Assume r-bit hash code matrices for the anchor set and the whole training set are $H = \{\mathbf{h}_i\}_{i=1}^m \in \{+1, -1\}^{r \times m}$ and $B = \{\mathbf{b}_i\}_{i=1}^n \in \{+1, -1\}^{r \times n}$. $q_{j|i}$ denotes the probability of of assigning data point $\mathbf{b}_j$ to anchor $\mathbf{h}_i$ in the hash code space. Inspired by t-SNE [14], we utilize a Student t-distribution with one degree of freedom to transform Hamming distances into probabilities, as shown in the following formula:

$$q_{j|i} = \frac{(1 + g(\mathbf{h}_i, \mathbf{b}_j))^{-1}}{\sum_{t=1}^n (1 + g(\mathbf{h}_i, \mathbf{b}_t))^{-1}} \tag{3}$$

where $g(\mathbf{h}_i, \mathbf{b}_j)$ denotes the Hamming distance between anchor $\mathbf{h}_i$ and data point $\mathbf{b}_j$ in the hash code space. Similar with the setting in the original space, we normalize the probabilities such that $\sum_{j=1}^n q_{j|i} = 1$. With $\mathbf{h}_i$ and $\mathbf{b}_j \in \{+1, -1\}^r$ for any i and j, the Hamming distance between hash codes of two instances can be calculated from their squared Euclidean distance as follows:

$$g(\mathbf{h}_i, \mathbf{b}_j) = \frac{1}{4}\|\mathbf{h}_i - \mathbf{b}_j\|_2^2 \tag{4}$$

thus $q_{j|i}$ can be further rewritten as follows:

$$q_{j|i} = \frac{(1 + \frac{1}{4}\|\mathbf{h}_i - \mathbf{b}_j\|_2^2)^{-1}}{\sum_{t=1}^n (1 + \frac{1}{4}\|\mathbf{h}_i - \mathbf{b}_t\|_2^2)^{-1})} \tag{5}$$

The objective of APHash is to learn optimal hash codes H and B of anchors and all data points that can minimize the mismatch between probability distributions $\mathcal{P}$ and $\mathcal{Q}$. By incorporating the correlation between anchors and data points, the distribution-preserving loss using KL-divergence to learn hash codes H and B is defined as follows:

$$J_0 = \sum_{i=1}^m \sum_{j=1}^n p_{j|i} log \frac{p_{j|i}}{q_{j|i}} \tag{6}$$

where $p_{j|i}$ and $q_{j|i}$ are defined as formula (1) and (5) respectively. Minimizing the cost function J_0 means to make the distributions $\mathcal{P}$ and $\mathcal{Q}$ as consistent as possible, so that the data structure in the training set can be well preserved in the learned hash code H and B.

To make the problem tractable, we follow the previous work [5, 15] and relax the binary constrained H and B to be real-valued, which are denoted as $\widehat{H}$ and $\widehat{B}$. Then, the formulation of $q_{j|i}$ will become:

$$q_{j|i} = \frac{(1 + \frac{1}{4}\|\hat{\mathbf{h}}_i - \hat{\mathbf{b}}_j\|_2^2)^{-1}}{\sum_{t=1}^n (1 + \frac{1}{4}\|\hat{\mathbf{h}}_i - \hat{\mathbf{b}}_t\|_2^2)^{-1})} \tag{7}$$

Quantization Loss

To minimize the quantization error between real-valued $\widehat{H}$ and $\widehat{B}$ and binary H and B, we introduce a regularization term to force the entries of $\widehat{H}$ and $\widehat{B}$ to be near to $+1$ or -1, which is defined in the following formula:

$$\begin{aligned} J_1 &= Q(\widehat{H}) + Q(\widehat{B}) \\ &= \frac{1}{Z_H}\||\widehat{H}| - \mathbf{1}_H\|_2^2 + \frac{1}{Z_B}\||\widehat{B}| - \mathbf{1}_B\|_2^2 \end{aligned} \tag{8}$$

where $\mathbf{1}_H$ and $\mathbf{1}_B$ are the matrices of the same dimensionalities as H and B with all entries being 1. $Z_H = r \times m$ and $Z_B = r \times n$ are normalizing factors to eliminate the effect of the training set size and the hash code length. We choose the L2-norm regularizer for the simplicity of its optimization.

Overall Formulation

We formulate the cost function of our APHash method as:

$$\begin{aligned} J &= J_0 + \lambda J_1 \\ &= \min_{\widehat{H}, \widehat{B}} \sum_{i=1}^m \sum_{j=1}^n p_{j|i} log \frac{p_{j|i}}{q_{j|i}} \\ &\quad + \lambda(\frac{1}{Z_H}\||\widehat{H}| - \mathbf{1}_H\|_2^2 + \frac{1}{Z_B}\||\widehat{B}| - \mathbf{1}_B\|_2^2) \end{aligned} \tag{9}$$

Table 1. Mean Average Precision of Hamming Ranking for different numbers of bits on two datasets.

Method	CIFAR-10 (mAP)				Youtube Faces (mAP)			
	8 bits	16 bits	32 bits	64 bits	8 bits	16 bits	32 bits	64 bits
LSH	0.1170	0.1222	0.1428	0.1515	0.1116	0.1586	0.2948	0.4354
SH	0.1295	0.1301	0.1303	0.1317	0.3900	0.5857	0.6652	0.5992
AGH	0.1507	0.1575	0.1483	0.1440	0.4527	0.6362	0.7299	0.6070
DSH	0.1470	0.1580	0.1625	0.1696	0.2754	0.3721	0.5207	0.5424
SpH	0.1465	0.1487	0.1537	0.1617	0.2646	0.3865	0.5030	0.5877
OEH	0.1373	0.1531	0.1572	0.1625	0.2182	0.4774	0.5901	0.6386
ITQ	0.1545	0.1650	0.1733	0.1787	0.4980	0.6709	0.7454	0.7525
APHash	**0.1630**	**0.1698**	**0.1779**	**0.1850**	**0.6160**	**0.6975**	**0.7499**	**0.7690**

where λ is a parameter to balance the effects of the distribution-preserving loss and the quantization loss. The asymmetric property of APHash provides an opportunity to learn more informative hash codes. As demonstrated in [16], the asymmetric structure tends to lead to a better retrieval performance, especially under the configuration of a short hash code length.

2.2. Optimization

We propose to utilize gradient descent based optimization technique to solve this unconstrained optimization problem and learn locally optimal $\widehat{B}$ and $\widehat{H}$. We calculate the gradients with respect to $\widehat{B}$ and $\widehat{H}$ respectively. The derivative w.r.t. $\hat{\mathbf{h}}_i$ is as follows:

$$\begin{aligned}\frac{\partial J}{\partial \hat{\mathbf{h}}_i} = \sum_{j=1}^{n}(1+\frac{1}{4}\|\hat{\mathbf{h}}_i-\hat{\mathbf{b}}_j\|_2^2)^{-1}\times(p_{j|i}-q_{j|i}) \\ \times(\hat{\mathbf{h}}_i-\hat{\mathbf{b}}_j)+\frac{\lambda}{Z_H}(|\hat{\mathbf{h}}_i-\mathbf{1}|)\odot sign(\hat{\mathbf{h}}_i)\end{aligned} \tag{10}$$

Similarly, the derivative w.r.t $\hat{\mathbf{b}}_j$ is:

$$\begin{aligned}\frac{\partial J}{\partial \hat{\mathbf{b}}_j} = -\sum_{j=1}^{n}(1+\frac{1}{4}\|\hat{\mathbf{h}}_i-\hat{\mathbf{b}}_j\|_2^2)^{-1}\times(p_{j|i}-q_{j|i}) \\ \times(\hat{\mathbf{h}}_i-\hat{\mathbf{b}}_j)+\frac{\lambda}{Z_B}(|\hat{\mathbf{b}}_j-\mathbf{1}|)\odot sign(\hat{\mathbf{b}}_j)\end{aligned} \tag{11}$$

where $\odot$ denotes entry-wise multiplication, $\mathbf{1}$ is a column vector with all entries being 1. The optimization process will be performed in an alternative way. Stochastic Gradient Descent is applied to optimize $\widehat{H}$ with formula (10) while fixing $\widehat{B}$, and update $\widehat{B}$ with (11) while fixing $\widehat{H}$. The corresponding binary hash code matrices can be derived via the sign function, i.e. $H = sign(\widehat{H})$ and $B = sign(\widehat{B})$.

2.3. Hash Function Learning

We utilize the original data matrix C and the binary matrix H of the anchors as the guidance to learn the linear hash functions, the objective function is as follows:

$$L = \min_{W} \|H - W^T C\|_2^2 + \alpha\|W\|_2^2 \tag{12}$$

where $W \in R^{r\times d}$ is the projection matrix to be learned, α is a hyperparameter to weight the L2 norm imposed on W. Finally, we obtain the analytical solution of W:

$$W = (CC^T + \alpha I_{d\times d})^{-1}CH^T \tag{13}$$

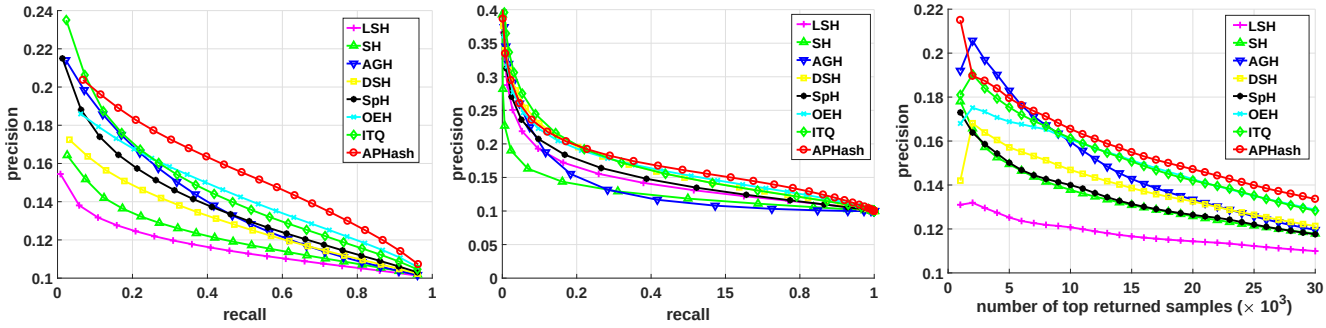
where $I_{d\times d}$ is an identity matrix of dimensions $d \times d$.

3. EXPERIMENT

3.1. Datasets and Experimental Setup

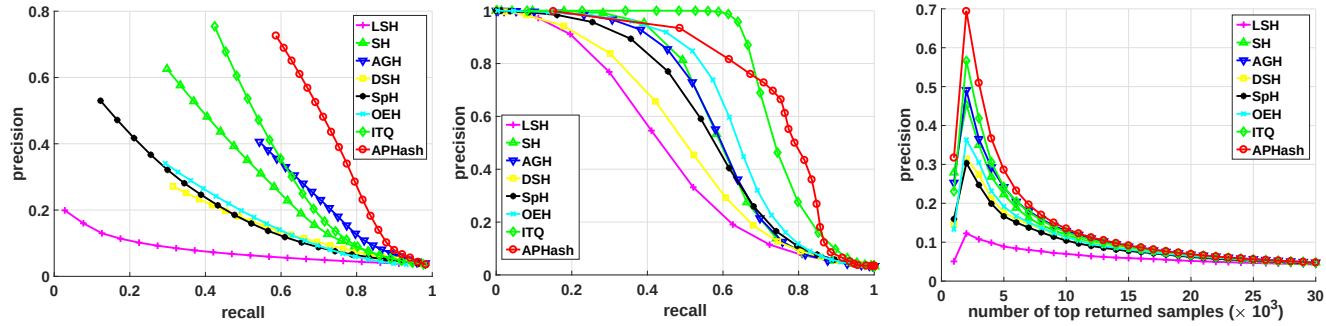
Datasets: We conduct experiments on two publicly available benchmark datasets: CIFAR-10 [12] and Youtube Faces [13]. The CIFAR-10 dataset consists of $60,000$ color images from 10 classes. Following [10, 15], each image is represented by a 512-dimensional GIST feature [17]. We randomly select 1000 images from dataset for query and the rest 59000 images for training and gallery database. This Youtube Faces dataset consists of face data involved with 1,595 persons. We first randomly choose 30 individuals, each of which has at least 1600 images. Each face image is represented by a 1,770-dimensional LBP feature vector [18]. Specifically, we randomly sample 100 images from each individual to form query set. We further randomly select 1500 images from each individual to form a training set of $45,000$ images. Since two datasets are fully annotated, we evaluate retrieval performance using the ground-truth semantic labels.

Parameters: For the setting of parameters, we set the number of nearest neighbors k as $1,000$ for two datasets empirically. The size of the anchor set is set as $10,000$ in all experiments. For the hyperparameters α to control the weight of L2 norm on W and λ to balance the effect of quantization loss, we set $\alpha = 1$ and $\lambda = 0.01$ respectively. In our experiments, we observed that the objective typically converges at about 50 iterations. Therefore, we set the maximum number of iterations as 50. The results of the compared methods are obtained with the codes provided by the authors, and the setting of parameters follows the suggestions in the original papers.



(a) precision-recall curve @8bit (b) precision-recall curve @64bit (c) precision curve @8bit

Fig. 1. ANN search performance evaluated with precision-recall curve (@8 bits and @64 bits respectively) and precision curve *w.r.t.* top returned samples (@8 bits) on CIFAR-10 Dataset. Best view in color.



(a) precision-recall curve @8bit (b) precision-recall curve @64bit (c) precision curve @8bit

Fig. 2. ANN search performance evaluated with precision-recall curve (@8 bits and @64 bits respectively) and precision curve *w.r.t.* top returned samples (@8 bits) on Youtube Faces Dataset. Best view in color.

3.2. Comparison with State-of-the-art Methods

We follow the recently published paper [10] and compare the proposed APHash with several state-of-the-art methods, including Locality Sensitive Hashing (LSH) [19], Spectral Hashing (SH) [5], Density Sensitive Hashing (DSH) [20], Spherical Hashing (SpH) [21], Anchor Graph Hashing (AGH) [6], Iterative Quantization (ITQ) [9], and Ordinal Embedding Hashing (OEH) [10].Following [20, 9], we evaluate APHash with three criterions: mean average precision (mAP), precision-recall curve, and precision curve. All the experiment results reported are the average of performance over 10 runs.

The quantitative results of comparison with seven state-of-the-art methods listed above are shown in Tab. 1, Fig. 1, and Fig. 2. For the comparison with mAP criteria, the length of hash codes varies from 8 bits to 64 bits. From Tab. 1, we can observe that the proposed APHash consistently outperforms all the compared approaches and achieves the state-of-the-art retrieval performance on both CIFAR-10 and Youtube Faces datasets. As for the precision-recall curve, we show the experimental results with short hash codes (8 bits) and long hash codes (64 bits) on two datasets in Fig. 1(a)(b) and Fig. 2(a)(b). We also show the precision curves to demonstrate the effectiveness of the APHash on two datasets in Fig. 1(c) and Fig. 2(c). As we can see, APHash also outperforms state-of-the-art methods. Experimental results show that APHash can learn more informative hash codes and more powerful hash function. Especially, with short hash codes (e.g. 8 bits), the proposed APHash outperforms the compared methods by a large margin, which demonstrates that such an asymmetric structure leads to the accommodation of much more information as discussed in Sec. 2.1.

4. CONCLUSION

In this paper, we proposed an unsupervised Anchor-based Probability Hashing (APHash) method by preserving the distribution of data points. Our method makes use of the correlation between anchors and data points to learn informative hash codes. It transforms the distances between data points into probability distributions and minimizes the KL-divergence to preserve the data structure. An alternating optimization algorithm is proposed to solve the problem. Experiments conducted on two publicly available datasets demonstrate that the proposed APHash outperforms the compared approaches and achieves state-of-the-art retrieval performance.

5. REFERENCES

[1] Ping Li, Anshumali Shrivastava, Joshua L Moore, and Arnd C König, "Hashing algorithms for large-scale learning," in *Advances in neural information processing systems*, 2011, pp. 2672–2680.

[2] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji, "Bilinear cnn models for fine-grained visual recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1449–1457.

[3] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han, "Large-scale image retrieval with attentive deep local features," 2016.

[4] Jifei Song, Yu Qian, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales, "Deep spatial-semantic attention for fine-grained sketch-based image retrieval," ICCV, 2017.

[5] Yair Weiss, Antonio Torralba, and Rob Fergus, "Spectral hashing," in *Advances in neural information processing systems*, 2009, pp. 1753–1760.

[6] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang, "Hashing with graphs," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 1–8.

[7] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang, "Discrete graph hashing," in *Advances in Neural Information Processing Systems*, 2014, pp. 3419–3427.

[8] Qing-Yuan Jiang and Wu-Jun Li, "Scalable graph hashing with feature transformation.," in *IJCAI*, 2015, pp. 2248–2254.

[9] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

[10] Hong Liu, Rongrong Ji, Yongjian Wu, and Wei Liu, "Towards optimal binary code learning via ordinal embedding.," in *AAAI*, 2016, pp. 1258–1265.

[11] Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang, "Semantics-preserving hashing for cross-view retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3864–3872.

[12] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," 2009.

[13] Lior Wolf, Tal Hassner, and Itay Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 529–534.

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

[15] 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 IEEE Conference on*. IEEE, 2012, pp. 2074–2081.

[16] Behnam Neyshabur, Nati Srebro, Ruslan R Salakhutdinov, Yury Makarychev, and Payman Yadollahpour, "The power of asymmetry in binary hashing," in *Advances in Neural Information Processing Systems*, 2013, pp. 2823–2831.

[17] Aude Oliva and Antonio Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

[18] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[19] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al., "Similarity search in high dimensions via hashing," in *VLDB*, 1999, vol. 99, pp. 518–529.

[20] Zhongming Jin, Cheng Li, Yue Lin, and Deng Cai, "Density sensitive hashing," *IEEE transactions on cybernetics*, vol. 44, no. 8, pp. 1362–1371, 2014.

[21] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon, "Spherical hashing," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2957–2964.