K-Nearest Neighbors Hashing

Xiangyu He[1,2], Peisong Wang[1], Jian Cheng[1,2,3]✉
[1] NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Center for Excellence in Brain Science and Intelligence Technology, CAS, Beijing, China
{xiangyu.he, peisong.wang, jcheng}@nlpr.ia.ac.cn

Abstract

Hashing based approximate nearest neighbor search embeds high dimensional data to compact binary codes, which enables efficient similarity search and storage. However, the non-isometry $sign(\cdot)$ function makes it hard to project the nearest neighbors in continuous data space into the closest codewords in discrete Hamming space. In this work, we revisit the $sign(\cdot)$ function from the perspective of space partitioning. In specific, we bridge the gap between k-nearest neighbors and binary hashing codes with Shannon entropy. We further propose a novel K-Nearest Neighbors Hashing (KNNH) method to learn binary representations from KNN within the subspaces generated by $sign(\cdot)$. Theoretical and experimental results show that the KNN relation is of central importance to neighbor preserving embeddings, and the proposed method outperforms the state-of-the-arts on benchmark datasets.

1. Introduction

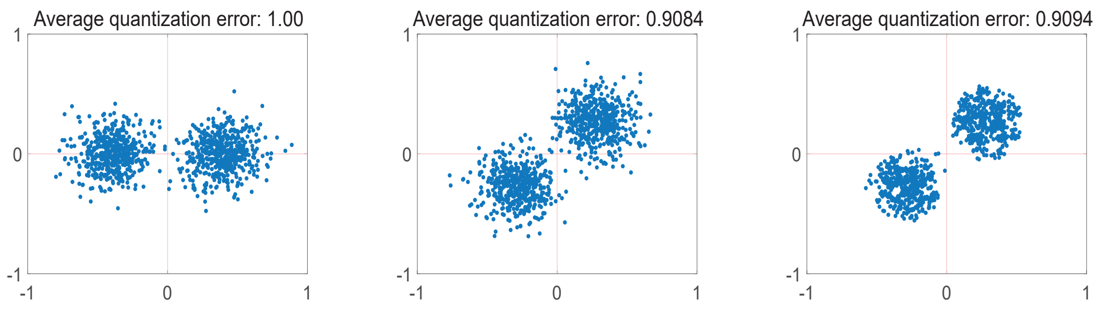
Similarity search is a fundamental problem in machine learning applications, such as clustering, matching, and classification. With the explosive growth of data size, traditional methods such as exhaustive search and Kd-tree, find themselves constrained by the huge size and high dimensionality. These problems lead to the boom of hashing based approximate nearest neighbor search [7, 28, 29, 20]. Hashing methods encode high dimensional data into vertices of binary hypercube while preserving the similarity in original data space. Due to its low computational cost and storage efficiency, the learning of similarity preserving binary codes has attracted much attention.

Traditional hashing methods consist of data independent and data dependent approaches. The classic data independent schemes include locality sensitive hashing (LSH) family [1, 6, 26, 8], using random projections to construct hashing functions. There is no doubt that LSH attains the preponderant influence in the context of extremely high-dimensional information retrieval. Nonetheless, LSH is still an inefficient learning strategy. Without considering the input data, the efficacy of LSH heavily relies on the coding length. In contrast with the limitations of data independent hashing, data dependent methods exploit the structure of data or semantic information to learn the compact binary representations.

Existing data dependent approaches can be further categorized into supervised and unsupervised schemes. Label-based hashing, such as supervised hashing with kernels [34], binary reconstruction embedding [25], supervised discrete hashing [39] and order preserving hashing [42] utilize semantic labels to optimize the binary hash codes or Hamming distances between clusters. Recently, deep learning has dramatically advanced the state-of-art [44, 46, 4, 45, 31, 30]. Both semantic representation and label information are used in deep neural networks to learn hash codes. However, high performance has been coupled with high computational and storage overhead. As pointed out in [12], hashing algorithms for learning binary codes and for encoding a new test image should be efficient and scalable.

Label-free hashing methods focus on the natural structure of data with no requirement on labels. Representative works include iterative quantization [12], anchor graph hashing [35], spectral hashing [43], spherical hashing [18], k-means hashing [17] and binary autoencoder [5]. Due to the redundancy of input features, a common initial technique in hashing schemes is principal component analysis (PCA) [43, 13, 22]. To deal with the variance (i.e., information) imbalance among different PCA directions, ITQ [12] utilizes an orthogonal rotation based on minimizing the quantization error. Although ITQ is still the seminal method in hashing, it is still unclear whether distortion minimization leads to optimal binary codes.

In this paper, we demonstrate that learning hashing function only from quantization error minimization may remain suboptimal. Inspired by Shannon entropy, we propose the



(a) PCA aligned. (b) ITQ rotation. (c) KNNH transformation.

Figure 1: Toy illustration of K-Nearest Neighbors Hashing (KNNH). The basic idea of binary embedding is to quantize data points to the closest vertex of the Hamming cube. (a) PCA leaves out the binary repesentation and splits each cluster to different vertices. (b) ITQ found the optimized rotation, in the context of lowest quantization error. (c) KNN Hashing endeavors to maintain the k-nearest neighbors within the same subspace during rotation (detailed in Section 2.4). Although it yields even larger quantization error than ITQ, the proposed transfomation is closer to ideal space partitioning.

conditional entropy minimization, which eludes analysis on $sign(\cdot)$ by transforming the hashing problem into a space partitioning problem. With Kozachenko-Leonenko estimator, we further prove that the conditional entropy minimization encourages the data point and its k-nearest neighbors to share the same hashing codes. As illustrated in Figure.1, the proposed K-Nearest Neighbors Hashing (KNNH) transformation approaches optimum by preserving KNN within the same subspaces (i.e., the same codewords). Extensive experiments show that our method outperforms the state-of-the-arts on benchmark datasets, which indicates the effectiveness of the proposed theory in real-world applications.

2. Approach

2.1. Preliminary

Formally, denote input matrix $X \in \mathbb{R}^{n\times d}$ as the concatenation of n vectors $X = \{x_i\}_{i=1}^n$. The vertices of an axis-aligned c-dimensional hypercube is $\{-1,+1\}^c$, denoted as $\mathbb{B}^c$. In general, the encoder $b_i = sign(x_i)$ maps a vector $x_i \in \mathbb{R}^d$ to the closest vertexes $b_j \in \mathbb{B}^c$; hence, we split $\mathbb{R}^d$ into 2^c disjoint subspaces $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{2^c}\}$ where $\mathcal{S}_j = \{x|sign(x) = b_j\}$. Given i.i.d. samples $\{x_i\}_{i=1}^n$ from the underlying probability density function $p(x)$ for $x \in \mathbb{R}^d$, we apply KNN estimator and re-substitution to non-parametric estimation (described in Section 2.3). Then, we have $p(b_j) = \int_x p(x, sign(x) = b_j)dx = \int_{x\in\mathcal{S}_j} p(x)dx$. For a discrete random variable Y with probability mass function $p(Y)$, Shannon entropy is defined as $H(Y) = -\mathbb{E}\{\log p(y)\} = -\sum_y p(y)\log p(y)$.

2.2. Mean-square minimization

The authors of [11] formulated a variety of hashing methods [12, 41, 43] and some other approximate nearest neighbor search schemes [36, 21] within a unified framework:

$$E = \sum_x ||x - d(e(x))||_2^2$$

where $e(\cdot)$ and $d(\cdot)$ refers to encoder and decoder, e.g., x is XR; $e(\cdot)$ is $sign(\cdot)$ and $d(\cdot)$ refers to scalar matrix in ITQ [12]. A quantizer that minimizes E should map any x to its nearest codeword in the codebook. We argure that this objective is simple and intuitive but may not straight to the hashing target.

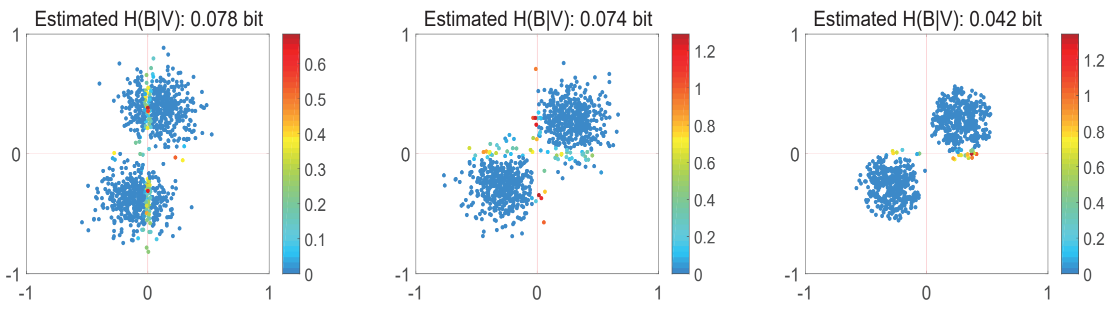
It is common practice to turn E into well-known signal-to-noise ratio (or signal-to-quantization-noise ratio) [14] :

$$SNR = 10\log_{10}\frac{\mathbb{E}(||x||^2)}{\mathbb{E}(||x - d(e(x))||^2)}.$$

This target alone reflects the compressibility of data instead of codewords similarity. In another word, distortion minimization is a data compression system where E and SNR focus on the minimization of reconstruction error. However, hashing aims at the approximation of nearest neighbors using codewords. There is no guarantee that optimized compression leads to the closest codewords since a cluster near the endpoints of a quantization interval can be split into different codewords.

2.3. Conditional entropy minimization

Under the constraint of orthogonal transformation, the relation of nearest neighbors can be preserved during rotation. However, this benefit may bring us to another pitfall: orthogonality is the guarantee of order-preserving. This condition holds when we relax our discrete hashing codes to be a continuous variable $\tilde{b} \in \mathbb{R}^c$. It is obvious that $||Rx_i - Rx_j||^2 = ||x_i - x_j||^2$ when $R^TR = I$. But the existence of non-smooth encoder makes $||e(Rx_i) - e(Rx_j)||^2 \leq$



(a) Random rotation. (b) ITQ rotation. (c) KNNH transformation.

Figure 2: The contribution (i.e., $\ln \frac{\epsilon(v_i;\mathcal{S}_{v_i})}{\epsilon(v_i)}$) of each data point to $\hat{H}(B|V)$, shown in colormap. KNN based shrinkage leads to lower conditional entropy than ITQ rotation due to less confusing points near boundary.

$||x_i - x_j||^2$ an open problem (i, j in a neighborhood). To overcome this issue, early works turn to relaxed hashing function such as $Sigmoid(\cdot)$ [42] and $tanh(\cdot)$ [33] to approximate original problem. We show that it is feasible to create the direct relation between features and hashing codes without approximation.

To directly model the connections between binary codewords and real-valued features, we circumvent the non-smooth $sign(\cdot)$ function and become interested in the subspaces partitioned by $sign(\cdot)$. Note that the number of components in a space partition usually plays a central role in probability and statistical learning theory, and the relationship between minimum mean-square and mutual information have been discussed in [38, 15, 16]. All those works inspire us to describe the hashing process in information-theoretic criteria :

$$\min H(B|V); \qquad V = f(X)$$

where $B = sign(V)$, the feature v_i is from a continuous distribution and b_j is the discrete output gallery codes. This target minimizes the uncertainty in B when V is known. In other words, the optimal feature representation should make it easy to determine their codewords.

The objective conforms to our intuition, but the plug-in method heavily relies on the approximation of probability density function over bins. Formally,

$$H(B|V) \equiv \int_v p(v)H(B|V=v)dv \tag{1}$$

$$\approx -\sum_{i,j} p_{v,b}(i,j) \log \frac{p_{v,b}(i,j)}{p_v(i)} \tag{2}$$

where $p_v(i) = \int_i \mu_i(v)dv$, the integration of estimated density $\mu(v)$ over ith bin and $p_{v,b}(i,j) = \int_i \mu_i(v \in \mathcal{S}_j)dv$. Clearly, it would be impractical to set the right side of E-q.(1) as the optimization target. To bridge over the obstacle, we utilize the alternate format of $H(B|V)$,

$$\begin{aligned} H(B) - H(B|V) = I(B;V) = H(V) - H(V|B) \\ H(B|V) = H(B) - H(V) + H(V|B). \end{aligned} \tag{3}$$

For the estimation of differential entropy, we further introduce the well-known Kozachenko-Leonenko estimator [23]

$$\hat{H}(V) = -\psi(k) + \psi(n) + \ln c_d + \frac{d}{n}\sum_{i=1}^{n} \ln \epsilon_k(v_i) \tag{4}$$

where $\psi(\cdot)$ is the digamma function, c_d is the volume of the d-dimensional unit ball and $\epsilon_k(v_i)$ is twice the disance from v_i to its kth nearest neighbor. Compared with conventional estimators based on binnings, Eq.(4) has minimal bias and estimates only from KNN distances. This property will ease the later derivations and lead to our hashing method.

From Eq.(4), we further estimate $\hat{H}(V|B) = \sum_j p(v \in \mathcal{S}_j)\hat{H}(V|v \in \mathcal{S}_j)$ in disjoint subspaces, which erases the error made in the individual integration over definite bins (E.q. (2)), so that

$$\begin{aligned} \hat{H}(B|V) =& \Big(-\sum_{j=1}^{2^c} \frac{|\mathcal{S}_j|}{n} \ln \frac{|\mathcal{S}_j|}{n}\Big) - \\ & \Big(-\psi(k) + \psi(n) + \ln c_d + \frac{d}{n}\sum_{i=1}^{n} \ln \epsilon_k(v_i)\Big) + \\ & \Big(\sum_{j=1}^{2^c} \frac{|\mathcal{S}_j|}{n}\big(-\psi(k) + \psi(|\mathcal{S}_j|) + \ln c_d + \\ & \frac{d}{|\mathcal{S}_j|}\sum_{i=1}^{n} \ln \epsilon_k(v_i; v \in \mathcal{S}_j)\big)\Big). \end{aligned} \tag{5}$$

Here $|\mathcal{S}_j|$ refers to the number of data points in space $\mathcal{S}_j$ and $\epsilon_k(v_i; v \in \mathcal{S}_j)$ is twice the distance from v_i to its KNN in $\mathcal{S}_j$, given $sign(v_i) = b_j$. Note that $\epsilon_k(v_i; v \in \mathcal{S}_j)$ makes

Algorithm 1: Unconstrained KNN Hashing.

```
Data: A set of data points $\{X_i\}_{i=1}^{n} \in \mathbb{R}^d$ in the form
      of $X \in \mathbb{R}^{n\times d}$.
Result: Codewords $B \in \{-1,+1\}^{n\times c}$ and
        transformation matrix $W \in \mathbb{R}^{c\times c}$.
$X \in \mathbb{R}^{n\times c} \leftarrow \phi(X)$;
// Dimensional reduction, PCA in this work;
while $W$ not converged do
    $V = XW$; // Here, $f(\cdot)$ is a linear projection;
    $\Omega \leftarrow$ Euclidean-KNNSearch$(V) \in \mathbb{N}^{n\times k}$;
    // Return the indexes of KNN;
    for $j = 1 : n$ do
        $V_j \leftarrow$ mean$(V[\Omega_j]) \in \mathbb{R}^{1\times c}$;
        // Update $V$, i.e. KNN Shrinkage;
    end
    $W = \arg\min ||\text{sign}(V) - XW||_F^2$;
end
$B = \text{sign}(V)$;
return $B, W$;
```

each feature point hold its KNN only when its KNN is in the same space as v_i. In this case, we simplify the double summation $\sum_j \sum_i \ln \epsilon_k(v_i; v \in \mathcal{S}_j)$ to $\sum_i \ln \epsilon_k(v_i; \mathcal{S}_{v_i})$ where $S_{v_i} = \{v | sign(v) = b_j = sign(v_i)\}$. By substituting this term into Eq.(5) and approximating digamma function as $\psi(n) = \ln n - \frac{1}{2n} - \frac{1}{12n^2} + O(\frac{1}{n^4})$ [2], $H(B|V)$ can be further written as

$$\hat{H}(B|V) = \underbrace{\frac{d}{n}\sum_{i=1}^{n} \ln \frac{\epsilon_k(v_i; \mathcal{S}_{v_i})}{\epsilon_k(v_i)}}_{\text{term I}} + \underbrace{\frac{1-m}{2n} - \frac{1}{n}\sum_{j=1}^{m} \frac{1}{12|\mathcal{S}_j|} + O(\frac{1}{n^2})}_{\text{term II}} \tag{6}$$

where m is the number of $\mathcal{S}_j$ satisfying $|\mathcal{S}_j| \neq 0$. For $m \ll n$, it is clear that $\hat{H}(B|V)$ dominated by term I with term II approaching zero. This result leaves us two insights: 1) the information loss of binarization comes from the data near the boundary, as shown in Figure.2 where warm colors are distributed among coordinate axes; and 2) $\hat{H}(B|V)$ should decrease when KNN of v_i all come from the same space. As for $m < n$, the second term is still lower bounded by $\frac{1-n}{2n} - \frac{n/12}{n} \approx -\frac{7}{12}$, and we can define a larger $\epsilon_k(v_i; \mathcal{S}_{v_i})$ when $k > |\mathcal{S}_j|$ to penalize the singletons (spaces with few samples) and to balance the contribution to the uncertainty between two terms.

2.4. K-nearest neighbors hashing

Although Eq.(6) has been more concrete than Eq.(1), it is still hard to put criteria into practice. Therefore, we propose a simple heuristic method to construct V, which reduces $\hat{H}(B|V)$ from the view of KNN distance. This idea begins with an unconstrained method and refined by an orthogonal constraint.

Algorithm 2: Orthogonal KNN Hashing.

```
Data: A set of data points $\{X_i\}_{i=1}^{n} \in \mathbb{R}^d$ in the form
      of $X \in \mathbb{R}^{n\times d}$.
Result: Codewords $B \in \{-1,+1\}^{n\times c}$ and rotation
        matrix $R \in \mathbb{R}^{c\times c}$.
$X \in \mathbb{R}^{n\times c} \leftarrow \phi(X)$;
// Dimensional reduction, PCA in this work;
$\Omega \leftarrow$ Euclidean-KNNSearch$(X) \in \mathbb{N}^{n\times k}$;
// Return the indexes of KNN;
for $j = 1 : n$ do
    $X_j \leftarrow$ mean$(X[\Omega_j]) \in \mathbb{R}^{1\times c}$;
    // Update $X$, i.e. KNN Shrinkage;
end
$B, R = \arg\min_{R^T R = I} ||\text{sign}(XR) - XR||_F^2$;
// Classical hashing problem;
return $B, R$;
```

Unconstrained method As shown in Eq.(6), the outlier of the cluster is more likely to drop into different spaces and is far more sensitive to its KNN than center points. In another word, $\epsilon(v_i; \mathcal{S}_{v_i})$ of outliers may hugely change in a small step towards cluster center, especially for each feature dimension $V^{(i)} \sim \mathcal{N}(0, \sigma^2)$. Hence, we propose the reduction in the volume of the cluster based on KNN, i.e., KNN shrinkage. To alleviate the impact of outliers, we reconstruct each feature point by the mean value of its k-nearest neighbors recursively (from 2-nn to k+1-nn in case 1-nn is still an outlier). That is, updated feature points $\hat{v}$ will be used in the following iterations (for loop in Algorithm 1,2). In fact, every feature point can be viewed as the weighted average with all the others. As illustrated in Figure.2, shrinkage step plays a key role in the decrease of $\hat{H}(B|V)$. Benefiting from this information gain, $\min ||sign(\hat{v}) - v||^2$ will better preserve the relation between original feature space and Hamming space. So far, we obtain the unconstrained KNNH, shown in Algorithm 1.

Orthogonal method Although the unconstrained method has taken $\epsilon(v_i; \mathcal{S}_{v_i})$ into consideration, there are two inevitable problems in practice: 1) computational time, e.g., KNN search and shrinkage in $while$ loop will take plenty of time due to massive computation and memory read/writes; and 2) without constraint, the only solution is the trivial one. Transformation matrix W minimizes $||sign(V) - XW||_F$ at the cost of losing KNN relations, where most data points

Table 1: Comparsions of different representative unsupervised hashing methods on the MNIST dataset. Each image was represented as a 784-D (28×28) gray-scale feature vector by using its intensity.

Method	Hamming Ranking (mAP,%)			precision (%)@N=1000			precision (%)@r=2	
	16	32	64	16	32	64	16	32
LSH[1]	20.88	25.83	31.71	37.77	50.16	61.73	25.10	55.61
SH[43]	26.64	25.72	24.10	56.29	61.29	61.98	57.52	65.31
PCAH[41]	27.33	24.85	21.47	56.56	59.99	57.97	36.36	65.54
SpH[18]	25.81	30.77	34.75	49.48	61.27	69.85	51.71	64.26
KMH[17]	32.12	33.29	35.78	60.43	67.19	72.65	61.88	68.85
ITQ[12]	41.18	43.82	45.37	66.39	74.04	77.42	65.73	**73.14**
KNNH	**47.33**	**53.25**	**56.03**	**67.95**	**75.89**	**79.04**	**71.82**	69.08

are packed into few buckets. Fortunately, an orthogonal constraint does address both problems simultaneously.

Since orthogonal transformation preserves lengths of vectors and angles between vectors, the KNN relation should be maintained during iterations. In this context, $\sum_{i\in\Omega_j}(XR)_i$ is equivalent to $(\sum_{i\in\Omega_j} X_i)R$ and we can move KNN search and shrinkage outside of the loop. Note that, the objective then becomes $\min ||sign(\hat{v}) - \hat{v}||^2$. That is a natural two-stage scheme: KNN based shrinkage and classical hashing problem, shown in Algorithm 2. Intuitively, one may argue that we just replace v by $\hat{v}$ in original hashing problem, but on the other hand, if we have V which satisfies $H(B|V) = 0$, a single "$sign(\cdot)$" should solve the hashing problem. At inference time, we directly apply the learned linear projections to unseen data points, i.e., testing samples, without KNN shrinkage.

3. Results

3.1. Datasets & Evaluation protocol

We evaluate the proposed K-Nearest Neighbors Hashing (KNNH) on three balanced benchmark datasets: CIFAR-10 [24], MNIST [27] and Places205 [47, 3], and we further verify the performance on an extremely imbalanced dataset: LabelMe-12-50K [40]. **CIFAR-10** dataset consists of 60,000 images of 10 classes. Each class contains 6,000 32x32 colour images. **MNIST** is a dataset containing 70,000 gray handwritten digit images in 10 classes. Each image is represented by a 28x28 gray-scale intensity matrix. Different from former ones, **LabelMe-12-50K** consists of 50,000 256x256 JPEG images of 12 classes, the data distribution among classes is imbalanced. Five large sample classes take 91% images and the smallest class contains only 0.6% samples. Besides, 50% of the images show a centered object and remaining 50% show a randomly selected region of a randomly selected image. This attribute matches the real-world challenge of image retrieval. As instances of other object classes may also be present in the image, we choose the object class with the largest label value as image labels. **Places205** is a very challenging dataset due to its large size and number of categories, which contains 2.5M images from 205 scene categories. Following [3], we use the CNN features extracted from the fc7 layer of ImageNet pre-trained AlexNet and reduce the dimensionality to 128 using PCA.

We use the following evaluation metrics to measure the performance of methods: 1) mean Average Precision (*mAP*) which evaluates the overall performance on different object classes; 2) precision of Hamming radius of 2 (*precision@r=2*) which measures precision on retrieved images having Hamming distance to query≤2 (we report zero precision for the queries if no image satisfy); 3) precision at N samples (*precision@N*) which refers to the percentage of true neighbors on top N retrieved samples. In our experiments, we strictly follow the same comparison settings in previous works, most of the results are directly reported by the authors. Besides, in order to improve the statistical stability, we repeated the experiments 10 times and took the average as the final result. Since the performance on small sample classes is more sensitive to the queries, we execute the experiments on LabelMe-12-50K 50 times to get the results. To prove that $\min H(B|X)$ leads to better codewords than straightforward quantization minimization, the hashing problem in KNNH was solved by *ITQ*, and we fully compare both methods on different datasets.

3.2. Results on balanced datasets

Following the same setting in [32, 9], we randomly selected 1000 samples from CIFAR-10, 100 per class, as the query data, and the remaining 59000 images as the gallery set. For MNIST dataset, we randomly sampled 100 per class, 1000 images in total, as the query data, and used the remaining 69000 images as the gallery set. The ground truths of queries are based on their class labels. Since hashing methods are independent of input features, we compare our method with representative hashing by using both handcrafted and deep features. In this subsection, we set *k* as 20 and make no effort on the fine-tuning.

Table 2: Comparsions of different representative unsupervised hashing methods on the CIFAR-10 dataset. Each image was represented as a 512-D GIST feature vector.

Method	Hamming Ranking (mAP,%)			precision (%)@N=1000			precision (%)@r=2	
	16	32	64	16	32	64	16	32
LSH[1]	12.55	13.76	15.07	16.21	19.10	22.25	16.73	7.07
SH[43]	13.19	12.97	13.18	17.74	17.93	18.43	19.42	**21.73**
PCAH[41]	13.23	12.89	12.30	17.86	17.91	16.91	21.80	3.00
SpH[18]	14.54	15.16	15.90	19.68	21.30	23.00	21.92	14.53
KMH[17]	16.05	16.19	15.79	21.21	22.56	22.83	23.48	12.80
ITQ[12]	16.57	17.34	17.91	22.08	23.98	25.21	**23.92**	16.90
KNNH	**17.32**	**18.76**	**19.54**	**22.52**	**25.48**	**27.08**	23.36	15.05

Table 3: Comparsions with deep learning methods and supervised methods. The top section are the unsupervised methods and the bottom section are the *supervised* methods (start from SDH).

Method	Hamming Ranking (mAP,%)			precision (%)@N=1000			precision (%)@r=2	
	16	32	64	16	32	64	16	32
CIFAR-10 Query=1,000								
Deepbit[31]	14.35	16.33	17.97	–	–	–	–	–
DH[32]	16.17	16.62	16.96	**23.79**	**26.00**	**27.70**	23.33	15.77
UHBDNN[9]	**17.83**	18.52	–	–	–	–	**24.97**	**18.85**
KNNH	17.32	**18.76**	**19.54**	22.52	25.48	27.08	23.36	15.05
MNIST Query=1,000								
DH[32]	43.14	44.97	46.74	67.89	74.72	78.63	66.10	73.29
UHBDNN[9]	45.38	47.21	–	–	–	–	69.13	75.26
KNNH	**47.33**	**53.25**	**56.03**	**67.95**	**75.89**	**79.04**	**71.82**	69.08
SDH[32]	46.75	51.01	52.50	65.19	70.18	72.33	63.92	**77.07**
SPLH[41]	44.20	48.29	48.34	62.98	67.89	67.99	63.71	74.06
BRE[25]	33.34	35.09	36.80	60.72	68.86	73.08	34.09	64.21

Table 1 shows the retrieval results of different hashing methods on the MNIST dataset. Each image is represented by a 784-dimensional gray-scale feature vector by using its intensity [37]. It is obvious that KNNH outperforms other representative unsupervised hashing methods on nearly every criteria. Table 2 obtains the same results on the CIFAR-10 dataset. KNNH mainly contributes to the increase of *mAP*, which directly reflects the changes in the order of recalls. Note that the *precision@r=2* values by KNNH are not very good but *p@r=2* is actually different from the well-known *R-precision* which describes one point on the precision-recall curve (we use *p@r=2* because it is popular in recent deep hashings). In our experiments, though the *p@r=2* values by KNNH are rather poor, the *recall@r=2* of KNNH are much higher than baselines. It is common to compare the precision of two points on different PR curves with the same recall, otherwise the comparison is unfair. Here is the same case. Since *r=2* can be an empirical setting to reduce the number of retrieval results, we also show the values of *precision@1K* as an alternative.

Although deep unsupervised hashing methods have attracted much attention, we show that linear transformation can achieve competitive results. Since Deepbit reported *mAP@1K* rather than *mAP* in [31], we rerun the open source codes and report the updated *mAP* in Table 3. Compared with deep hashing, our approach is still in the lead of *mAP* at 32/64bit. It is interesting that KNNH even surpasses a few supervised methods on MNIST.

In [46, 45, 9, 19], the authors use pre-trained CNN features as input for non-CNN-based hashing methods, all approaches achieve higher performances in most of evaluation metrics. Follow that setting, we use VGG features as input for CIFAR-10 and set up similar experiments. The query set contains 6,000 randomly sampled images (10% images per class) and the rest 54,000 image are used as the gallery set. For MNIST dataset, we evaluate the performances on GIST 512-D descriptor since MNIST is grayscale. Similar to CIFAR-10, we randomly sample 10% images per class, as the query data, and use the remaining images as the training set and retrieval database. Table 4 shows that

Table 4: mAP (%) for different unsupervised methods using high-level features. We reported the results on RGB datasets (CIFAR-10, LabelMe) using VGG-FC7 descriptor and MNIST using GIST 512-D descriptor.

Method	CIFAR-10			LabelMe-12-50K			MNIST		
	16	32	64	16	32	64	16	32	64
VGG+SH[43]	18.31	16.54	15.78	12.60	12.59	12.24	32.59	33.23	30.65
VGG+SpH[18]	18.82	20.93	23.40	13.59	15.10	17.03	31.27	36.80	41.40
VGG+KMH[17]	18.68	20.82	22.87	13.36	15.47	16.58	31.96	37.39	41.11
VGG+BA[5]	25.38	26.16	27.99	16.96	18.42	20.80	48.48	51.72	52.73
VGG+ITQ[12]	26.82	27.38	28.73	18.06	19.40	20.71	46.37	50.59	53.69
VGG+KNNH	**29.06**	**30.82**	**32.60**	**20.13**	**23.79**	**26.22**	**53.07**	**61.11**	**65.55**

Table 5: Comparsions with ITQ[12] on the small sample classes of imbalanced dataset. mAP (%, 32-bit) using GIST 512-D and VGG-FC7 descriptor was reported. Proportion reflects the number of each class accounts for the whole samples.

All Small Sample Classes in LabelMe-12-50K							
Class	sign	door	bookshelf	chair	table	keyboard	head
Proportion	2.5%	2.2%	1.0%	1.1%	0.6%	0.9%	0.7%
Hamming Ranking (mAP,%)							
ITQ[12]	3.46	4.76	2.43	1.38	0.74	6.94	1.56
KNNH	**3.80**	**4.82**	**2.50**	**1.42**	**0.75**	**12.75**	**1.86**
VGG+ITQ[12]	5.44	4.05	12.33	3.72	1.31	9.76	7.92
VGG+KNNH	**7.19**	**4.79**	**19.42**	**4.99**	**1.42**	**20.48**	**14.61**
Increase	1.32×	1.18×	1.58×	1.34×	1.08×	**2.10**×	1.84×

our method consistently outperforms others along with the increase of bit-width.

3.3. Results on imbalanced dataset

Imbalanced data problem has always been a hot topic in the machine learning community. In this section, we evaluate the performance of KNNH on an extremely imbalanced dataset: LabelMe-12-50K. As the smallest class contains only ~ 300 images, we randomly selected 10% images per class, as the query data, and used the $\sim 45,000$ images as the gallery set. The ground truths of queries are based on their class labels. In this subsection, we set *k* as 20 and make no effort on the fine-tuning.

To avoid the results dominated by the large sample classes, we report the *mAP* in Table 4 which is the macro average results for all classes. There is no doubt that KNNH takes the lead at 16/32/64 bits. However, the overall performance is not sufficiently convincing, we further report the *mAP* at all small classes in Table 5. The results show that KNNH does outperform the state-of-art method on difficult retrieval tasks. Besides, in combination with discriminating features, KNNH further enhances the hashing quality with the increase of bit-width. In some cases, we achieve 200% improvement, but to be honest, our results are still poor. 1.42% *mAP* is clearly a large space to investigate.

3.4. Results on large-scale dataset

To meet the real-world challenge, we further evaluate KNNH on the large-scale Places205 dataset [47]. We randomly sample 100 images from each class to construct a test set of 20,500 images and use the rest ~ 2.5M images as the retrieval set. The ground truths of queries are based on their class labels. Since Places205 is much larger than previous datasets, we change k to 200 and make no effort on the fine-tuning. As shown in Table 6, our approach consistently surpasses representative unsupervised hashing methods on mAP.

3.5. Performance under various k

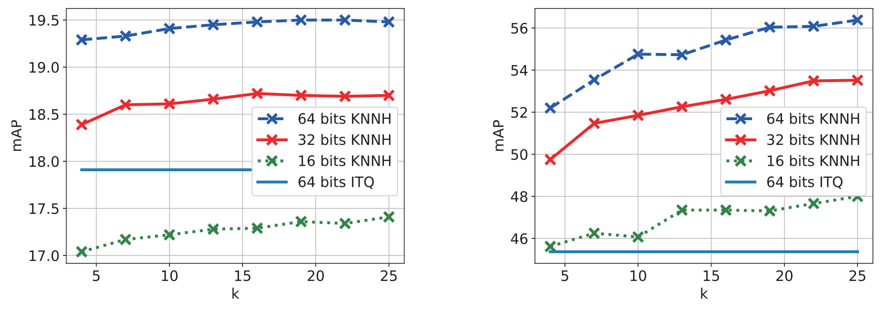
The performance of k-nn algorithms can be severely degraded by the selection of k. But, an effective method should achieve the consistent performance in a wide range of k. Figure.3 shows the robustness of our approach. KNNH consistently outperforms the leading method by a large margin. The results are compelling that our 16-bit KNNH has already surpassed 64-bit ITQ on MNIST datasets.

3.6. Computation time

Lastly, we analyze the computation time in practice. Since we did not introduce extra computation or storage at

Table 6: Comparsions of different representative unsupervised hashing methods on the Places205 dataset. Each image was represented by CNN features extracted from the AlexNet-FC7, and reduce the dimensionality to 128 using PCA.

Method	Hamming Ranking (mAP,%)			precision (%)@N=1000			precision (%)@r=2	
	16	32	64	16	32	64	16	32
SpH[18]	3.36	5.15	7.45	8.83	14.35	19.54	5.67	18.84
SH[43]	4.44	6.67	8.50	11.38	17.57	22.11	7.36	22.44
PCAH[41]	4.66	7.60	10.74	11.89	19.28	25.63	8.22	24.60
KMH[17]	4.78	7.65	10.60	12.10	19.22	25.32	8.29	**24.65**
BA[5]	5.73	9.65	13.44	12.40	20.24	26.03	7.96	23.65
ITQ[12]	5.89	9.69	13.53	12.53	20.16	26.28	7.99	23.32
KNNH	**7.60**	**12.17**	**15.92**	**13.45**	**21.04**	**26.43**	**8.76**	19.99



(a) CIFAR-10. (b) MNIST.

Figure 3: Comparsions on CIFAR-10 and MNIST with ITQ under different k. We reported the results on CIFAR-10 dataset using GIST 512-D descriptor and MNIST using 784-D (28×28) intensity feature vector. Since ITQ has no connection with k, the performance remains the same as the blue solid line depicted in both figures.

inference time, KNNH keeps the same speed as the algorithm to solve the second stage hashing problem. In our experiments, the testing time of KNNH on 32-bit Cifar-10 is 1.7×10^{-6} seconds with an Intel 3.0GHz CPU. Hence, we focus on the training time of KNN search and KNN shrinkage. Formally, given d dimensional features, KNN shrinkage has the linear time complexity: $\mathcal{O}(n)$ (approximately kn memory reads, n memory writes, $(k-1)nd$ additions/subtractions and nd multiplications). Therefore, the main issue of KNNH is its huge complexity in exhaustive KNN search: $\mathcal{O}(n^2 d)$ for the distances computing and $\mathcal{O}(n^2 \log n)$ for sortings.

By utilizing the power of GPU on parallel computing [10], we reduce the search time on CIFAR-10 (32-bit) from 27.06s to 1.81s (3.00GHz Intel CPU vs. Nvidia TITAN Xp). MNIST and LabelMe share a similar computation time on 32-bit, which is 2.43s and 1.09s, respectively. For Places205, the training time is about 110 minutes, since k is $10\times$ larger than training on small datasets and the huge data size limits the further GPU speedup. Besides, we noticed that the dimension of the feature points has only a small impact on the computation time, which makes it possible to increase the bit-width to achieve higher performance. In general, our training method can be implemented within a reasonable time without losing the performance of runtime speed.

4. Conclusion

We have introduced a hidden factor in learning hashing codes, which is the k-nearest neighbors' relation in subspaces. By adopting the view of conditional entropy minimization, we further propose a simple but effective method to enhance hashing quality. In a word, we create a direct connection between binary codewords and input features through k-nearest neighbors. Future works should extend those results to other datasets. The direct minimization of $H(B|V)$ is also worthy of further discussions.

5. Acknowledgement

This work was supported in part by National Natural Science Foundation of China (No.61876182, 61872364), the Strategic Priority Research Program of Chinese Academy of Science(No.XDB32050200).

References

[1] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[2] J. M. Bernardo. Algorithm as 103: Psi (digamma) function. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 25(3):315–317, 1976.

[3] Fatih Çakir, Kun He, Sarah Adel Bargal, and Stan Sclaroff. Mihash: Online hashing with mutual information. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 437–445, 2017.

[4] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. Deep quantization network for efficient image retrieval. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3457–3463, 2016.

[5] Miguel Á. Carreira-Perpiñán and Ramin Raziperchikolaei. Hashing with binary autoencoders. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 557–566, 2015.

[6] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 380–388, 2002.

[7] Jian Cheng, Cong Leng, Jiaxiang Wu, Hainan Cui, and Hanqing Lu. Fast and accurate image matching with cascade hashing for 3d reconstruction. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1–8, 2014.

[8] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*, pages 253–262, 2004.

[9] Thanh-Toan Do, Anh-Dzung Doan, and Ngai-Man Cheung. Learning to hash with binary deep neural network. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V*, pages 219–234, 2016.

[10] Vincent Garcia, Eric Debreuve, and Michel Barlaud. Fast k nearest neighbor search using GPU. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2008, Anchorage, AK, USA, 23-28 June, 2008*, pages 1–6, 2008.

[11] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 2946–2953, 2013.

[12] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2916–2929, 2013.

[13] Albert Gordo, Florent Perronnin, Yunchao Gong, and Svetlana Lazebnik. Asymmetric distances for binary embeddings. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(1):33–47, 2014.

[14] R. Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, April 1984.

[15] Dongning Guo, Shlomo Shamai, and Sergio Verdú. Mutual information and minimum mean-square error in gaussian channels. *IEEE Trans. Information Theory*, 51(4):1261–1282, 2005.

[16] Dongning Guo, Yihong Wu, Shlomo Shamai (Shitz), and Sergio Verdú. Estimation in gaussian noise: Properties of the minimum mean-square error. *IEEE Trans. Information Theory*, 57(4):2371–2385, 2011.

[17] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 2938–2945, 2013.

[18] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 2957–2964, 2012.

[19] Tuan Hoang, Thanh-Toan Do, Dang-Khoa Le Tan, and Ngai-Man Cheung. Enhancing feature discrimination for unsupervised hashing. In *2017 IEEE International Conference on Image Processing, ICIP 2017, Beijing, China, September 17-20, 2017*, pages 3710–3714, 2017.

[20] Qinghao Hu, Peisong Wang, and Jian Cheng. From hashing to cnns: Training binary weight networks via hashing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3247–3254, 2018.

[21] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011.

[22] Herve Jegou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 3304–3311, 2010.

[23] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Phys. Rev. E*, 69:066138, Jun 2004.

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

[25] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 1042–1050, 2009.

[26] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *IEEE 12th International Conference on Computer Vision, ICCV 2009,*

Kyoto, Japan, September 27 - October 4, 2009, pages 2130–2137, 2009.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Cong Leng, Jiaxiang Wu, Jian Cheng, Xiao Bai, and Hanqing Lu. Online sketching hashing. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 2503–2511, 2015.

[29] Cong Leng, Jiaxiang Wu, Jian Cheng, Xi Zhang, and Hanqing Lu. Hashing for distributed data. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1642–1650, 2015.

[30] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2479–2488, 2017.

[31] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1183–1192, 2016.

[32] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 2475–2483, 2015.

[33] Hong Liu, Rongrong Ji, Yongjian Wu, and Wei Liu. Towards optimal binary code learning via ordinal embedding. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 1258–1265, 2016.

[34] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 2074–2081, 2012.

[35] 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)*, pages 1–8. Citeseer, 2011.

[36] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[37] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[38] Sudhakar Prasad. Certain relations between mutual information and fidelity of statistical estimation. *CoRR*, abs/1010.1508, 2010.

[39] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 37–45, 2015.

[40] Rafael Uetz and Sven Behnke. Large-scale object recognition with cuda-accelerated hierarchical neural networks. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 1, pages 536–541. IEEE, 2009.

[41] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393–2406, 2012.

[42] Jianfeng Wang, Jingdong Wang, Nenghai Yu, and Shipeng Li. Order preserving hashing for approximate nearest neighbor search. In *ACM Multimedia Conference, MM '13, Barcelona, Spain, October 21-25, 2013*, pages 133–142, 2013.

[43] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1753–1760, 2008.

[44] Huei-Fang Yang, Kevin Lin, and Chu-Song Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(2):437–451, 2018.

[45] Ting Yao, Fuchen Long, Tao Mei, and Yong Rui. Deep semantic-preserving and ranking-based hashing for image retrieval. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3931–3937, 2016.

[46] Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, and Lei Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person reidentification. *IEEE Trans. Image Processing*, 24(12):4766–4779, 2015.

[47] Bolei Zhou, Àgata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 487–495, 2014.