Transformed Anti-Sparse Learning for Unsupervised Hashing

Zhangyang Wang
atlaswang@tamu.edu
Department of Computer Science and Engineering, Texas A&M University

Ji Liu
jliu@cs.rochester.edu
Department of Computer Science, University of Rochester

Shuai Huang
shuaih@uw.edu
Department of Industrial and Systems Engineering, University of Washington

Xinchao Wang
xinchao@illinois.edu
Beckman Institute, University of Illinois Urbana-Champaign

Shiyu Chang
shiyu.chang@ibm.com
IBM T. J. Watson Research Center

Abstract

Anti-sparse representation was recently considered for unsupervised hashing, due to its remarkable robustness to the binary quantization error. We relax the existing spread property [4, 22] for anti-sparse solutions, to a new *Relaxed Spread Property* (RSP) that demands milder conditions. We then propose a novel *Transformed Anti-Sparse Hashing* (**TASH**) model to overcome several major bottlenecks, that have significantly limited the effectiveness of anti-sparse hashing models. TASH jointly learns a dimension-reduction transform, a dictionary and the anti-sparse representations in a unified formulation. We have conducted extensive experiments on real datasets and practical settings, and demonstrate the highly promising performance of TASH.

1 Introduction

Hashing is a set of approaches to transform a data sample to a low-dimensional bit sequence [17, 23, 24]. When the semantic similarity information like the class label is available, *supervised hashing* methods obtain promising performance by explicitly considering semantic consistency. *Unsupervised hashing* methods, as another main category, rely only on unlabeled data to generate binary hashing codes, making them more challenging to achieve good performance. In this paper, our main focus is on the task of unsupervised hashing.

The seminal work of [8] formulated the paradigm for the locality-sensitive hashing (LSH), which guarantees that the hashing codes of similar data will be closer in the Hamming space. Despite its elegant theoretic guarantee and computational simplicity, the classic random hyperplane-based LSH methods [2] suffer from high code redundancy. More recently, *learning to hash* has gained increasing popularity. A few representation methods include spectral hashing [28], spherical hashing [7], density sensitive hashing [11], discrete graph hashing

[16], and iterative quantization (ITQ) [5]. Such methods involve learning a compound hashing function that maps vectors $\in \mathbb{R}^p$ to binary codes $\in \mathbb{B}^c$, such that the Hamming distance between codes preserves the similarity between original vectors.

As the critical problem of learning to hash, the choice of hashing functions influences the flexility of partitioning the space and the efficiency of computing hashing codes. The most popular choice is arguably the linear hashing function, e.g., in [5], that projects the vectors onto random or learned directions and then takes the sign to produce binary codes. Such a typical "project and sign" strategy, however, was suggested by [1] to be sub-optimal for the typical hashing case $p > c$, where the projection cannot be orthogonal and the strategy may lead to a bad quantizer (check Example (9) in [1]).

Motivated by such observations, recent approaches adopt more sophisticated encoding strategies, such as spherical function [7], kernelized function [13], sparse coding [32], and neural network [14]. Our particular interest is paid to the anti-sparse coding model for hashing [4, 10, 26]. For a data sample $x \in \mathbb{R}^p$, its anti-sparse code $a \in \mathbb{R}^c$ can be solved via:

$$\min_a ||a||_\infty, \quad s.t. \quad \mathbf{D}a = x \tag{1}$$

where $\mathbf{D} \in \mathbb{R}^{p\times c}$ is the dictionary. The binary code is obtained by simply thresholding a at zero. This objective of anti-sparse coding (1) resembles the better-known sparse coding [27], except the ℓ_0 or ℓ_1 norm being replaced by the ℓ_∞ norm. As a result, in contrast to concentrating the signal representation on few elements, the solution of (1) is called "spread" or "democratic": it offers a robust representation of a vector in a higher dimensional space with all the components sharing the information approximately evenly. [4, 22] showed that a usually has most entries reaching the same absolute maximum magnitude, therefore resembling to an antipodal signal $\in \mathbb{B}^c$. Therefore, converting x to its anti-sparse representation a reduces the quantization error underpinning the binarization (sign) function.

Despite the appealing advantages of minimizing quantization distortions and introducing a non-linear hashing function, the study of anti-sparse coding for hashing has so far been limited and preliminary. First, while *dictionary learning* has become the standard tool for sparse coding [27], existing anti-sparse hashing models [1, 10] only exploited *fixed tight frames* $\mathbf{D}$. [22] proved that given a *full-spark* overcomplete $\mathbf{D}$, at least $c - p + 1$ elements of the anti-sparse solution would be stuck to the extreme values $\pm||a||_\infty$, referred to as the *spread property* hereinafter (see Lemma 2 in [22]). However, the overcomplete assumption becomes **impractical** when it comes to hashing, whose goal is to represent high-dimensional x (e.g., image descriptors with the dimensionality of thousands) with a compact code (tens or hundreds of bits). Following the above theoretical results, [1, 10] either tested only on synthetic data, or conducted benchmark evaluations in the unusual setting of $c > p$. Their demonstrated efficiency thus cannot be on par with many state-of-the-art models that work effectively under $c < p$. A straightforward idea is to apply principal component analysis (PCA) as pre-processing, to project x to a lower dimension $d \leq c$. However, it is not jointly considered with the hashing step, and will inevitably cause considerable information loss.

This paper makes multi-fold contributions to overcome the above theoretical and practical limitations. First, we derive the new *Relaxed Spread Property* (**RSP**), that avoids the strong assumption of $\mathbf{D}$, as well as unties the lower bound of extreme elements with the number of rows p. Next, to reduce the practical gap between c and p, we propose a novel *Transformed Anti-Sparse Hashing* (**TASH**) model, which learns *dimension-reduction transform*, *dictionary* and anti-sparse *representation* simultaneously. To our best knowledge, TASH is the first model that optimizes the three in a unified formulation. We then present a systematical evaluation of TASH, which clearly manifests its highly competitive performance.

2 Theoretical Result

[4] observed the spread property of ℓ_∞ minimization (1) w.r.t. a tight frame $\mathbf{D} \in \mathbb{R}^{p\times c}$ ($p < c$), and claimed the lower bound of $c-p+1$ on the minimum number of entries whose magnitudes reach $\|a\|_\infty$. However, it remains unclear how their result can be extended from the "noiseless" setting (1) to the "noisy" case such as (7). Besides, their claim was stated without explicitly specifying requirements on the dictionary, except for being a tight frame. [22] formally analyzed the solution to the ℓ_∞ minimization under the inequality constraint of the approximation accuracy ε, denoted by $\mathrm{P}_\infty^\varepsilon$. They established that if the frame has full spark, then every solution to $\mathrm{P}_\infty^\varepsilon$ has at least $c-p+1$ extreme entries. The results imply that for full-spark and highly overcomplete $\mathbf{D}$, solving constrained ℓ_∞ minimization is guaranteed to lead to an anti-sparse representation with most entries being extreme.

Despite a few deterministic and random constructions of full-spark frames being known, it is challenging to ensure that a learned dictionary $\mathbf{D}$ in (6) will satisfy the full spark condition. Further, the lower bound $c-p+1$ remains to be associated with the redundancy (c/p). We thus seek an alternative to replace the strong full spark condition and the high redundancy implication of $\mathbf{D}$.

Our result, termed as the *Relaxed Spread Property* (RSP), is stated below. Define the *independent cardinality* of a matrix $\mathbf{D} \in \mathbb{R}^{p\times c}$ to be the minimal number of columns *arbitrarily* selected from $\mathbf{D}$ such that

$$\text{span of selected columns} \supseteq \text{span of remaining columns}.$$

As an equivalent definition, the independent cardinality is the smallest number of columns arbitrarily selected from $\mathbf{D}$, such that their span is equal to the span of $\mathbf{D}$.

As a special case, if $\mathbf{D}$ has full spark (which requires $p \le c$), then its independent cardinality is p, and RSP is reduced to the result of [22]. For a non-full spark $\mathbf{D}$, we do not see a general relation between its spark and independent cardinality, as the two are formulated for different purposes. Just like spark, the independent cardinality of a matrix is NP-hard to compute. Importantly, all our conclusions below stay valid for arbitrary $\mathbf{D}$, even $p > c$.

Theorem 2.1. *Let a^* be the optimal solution to the following ℓ_∞ regularized optimization*

$$\min_a \quad \tfrac{1}{2}\|\mathbf{D}a - x\|^2 + \lambda\|a\|_\infty, \tag{2}$$

where $\mathbf{D} \in \mathbb{R}^{p\times c}$ and $\lambda > 0$. Then a^* includes at least $c-t+1$ extreme entries with the magnitude $\|a^*\|_\infty$, where t is the independent cardinality of $\mathbf{D}$. Further, it indicates that the intensity of a^* is evenly distributed

$$\|a^*\|_\infty \le \frac{1}{\sqrt{c-t+1}}\|a^*\|.$$

Proof. First, if $\mathbf{D}a^*$ is 0, then a^* must be 0 and the claim is satisfied automatically. Second, we consider the case $\mathbf{D}a^*$ is nonzero, and prove the claim by contradiction. Assume that k elements in a^* are extreme values with $k \le c-t$. Let Ω denote the index of elements in a^* that are extreme, and $\bar{\Omega}$ be the complementary set. Note that $|\Omega| = k$ and $|\bar{\Omega}| = c-k \ge t$. From the definition of the independent cardinality, we know that

$$\text{range}(\mathbf{D}_{\bar{\Omega}}) \supseteq \text{range}(\mathbf{D}_\Omega).$$

It follows that there must exist a nonzero vector v satisfying

$$\mathbf{D}_{\bar{\Omega}}v = \mathbf{D}a^* = \mathbf{D}_\Omega a^*_\Omega + \mathbf{D}_{\bar{\Omega}}a^*_{\bar{\Omega}}.$$

Next we construct a new solution a' with $a'_{\Omega} = (1-\theta)a^*_{\Omega}$, $a'_{\bar{\Omega}} = (1-\theta)a^*_{\bar{\Omega}} + \theta v$. One can verify that the $\mathbf{D}a' = \mathbf{D}a^*$ for any θ. Thus, the function value of the square term are the same for a^* and x'. But next we will see $\|a^*\|_\infty > \|a'\|_\infty$ if θ is chosen appropriately, which leads the contradiction that a^* is not the optimal solution. To see that, we have

$$\|a'\|_\infty = \max\left\{|1-\theta|\|a^*_{\Omega}\|_\infty, |1-\theta|\left\|a^*_{\bar{\Omega}} + \frac{\theta}{1-\theta}v\right\|_\infty\right\}.$$

To show $\|a^*\|_\infty > \|a'\|_\infty$, it suffices to show that there exist a $\theta \in (0,1)$ such that

$$\left\|a^*_{\bar{\Omega}} + \frac{\theta}{1-\theta}v\right\|_\infty \leq \|a^*_{\Omega}\|_\infty = \|a^*\|_\infty.$$

Since $\|a^*_{\bar{\Omega}}\|_\infty < \|a^*_{\Omega}\|_\infty$, $a^*_{\bar{\Omega}}$ is an interior point in the box set $\{\mathbf{y} \in \mathbb{R}^{c-k} : \ \|\mathbf{y}\| \leq \|a^*\|_\infty\}$. Therefore, given an arbitrary direction $v \neq 0$, shifting $a^*_{\bar{\Omega}}$ alone such direction by a tiny (as long as θ is small enough) step $\frac{\theta}{1-\theta}$ can guarantee that $a^*_{\bar{\Omega}} + \frac{\theta}{1-\theta}v$ is still in the box set, which proves (2). The second claim is implied similarly. □

RSP holds without hinging on any assumption of $\mathbf{D}$, e.g., being a tight frame or full spark. Moreover, the lower bound by RSP on the number of extreme entries in a is not tied with the number of rows p in $\mathbf{D}$. That is different from the lower bound given by [4, 22], and potentially allows for less overcomplete $\mathbf{D}$ with sufficiently small independent cardinality. In future work, we also aim to minimize the independent cardinality of $\mathbf{D}$ using some tractable loss function, which can be incorporated into TASH and regularize the learning of $\mathbf{D}$.

3 Model and Algorithm

RSP relaxes the requirement of the classical spread property, i.e., it is no longer necessary to ensure a full-spark or tight-frame $\mathbf{D}$. Although it is infeasible to directly minimize the independent cardinality of $\mathbf{D}$, RSP provides the promise in choosing $\mathbf{D}$ more flexibly, motivating us to introduce a dictionary learning step. However, with $c \leq p$ in the hashing practice, solving (2) usually will not produce a with most entries reaching the extreme. *Our most important innovation* is to jointly learn a dimensionality reduction **transform matrix** $\mathbf{P} \in \mathbb{R}^{d\times p}$, where $d < p$, so that the input for anti-sparse coding becomes $\mathbb{R}^d$ instead of $\mathbb{R}^p$.

3.1 Model Formulation

Assume the data matrix $\mathbf{X} \in \mathbb{R}^{p\times n}$, consisting of n training samples $x_i \in \mathbb{R}^p$, $i = 1, 2, \ldots n$. The goal is to obtain a hashing code matrix $\mathbf{B} \in \mathbb{B}^{c\times n}$. While the integer constraint is intractable, we instead seek to solve the anti-sparse representation $\mathbf{A} \in \mathbb{R}^{c\times n}$, with $a_i \in \mathbb{R}^c$ denoting its i-th column, $i = 1, 2, \ldots n$, such that $\mathbf{B} = \mathbb{I}_{\mathbf{A}>0}$. With the transform matrix $\mathbf{P} \in \mathbb{R}^{d\times p}$, we correspondingly define the dictionary $\mathbf{D} \in \mathbb{R}^{d\times c}$, where $d_j \in \mathbb{R}^d$ denotes its j-th column, $j = 1, 2, \ldots c$. Note that by adjusting d, $\mathbf{D}$ may be overcomplete ($d < c$), complete ($d = c$), or undercomplete ($d > c$). Assume that $\mathbf{P}$ is pre-selected, the following model implements the anti-sparse coding with dictionary learning and fixed transform (λ is a constant):

$$\min_{\mathbf{D},\mathbf{A}} \tfrac{1}{2}||\mathbf{DA} - \mathbf{PX}||_2^2 + \lambda \textstyle\sum_{i=1}^n ||a_i||_\infty, \quad s.t. \quad ||d_j|| \leq 1, j = 1, 2, \ldots, c. \tag{3}$$

P could be chosen as a PCA matrix or other common dimensionality reduction operators. An alternative would be to learn **P** jointly as well, with the key problem to design a proper regularization function on **P**. Directly setting **P** as a variable in (3) leads to the trivial solution of $\mathbf{P} = \mathbf{0}$ and $\mathbf{A} = \mathbf{0}$. Another option inspired by the transform learning literature [29, 30] is to set $\mathbf{P}\mathbf{P}^T = \mathbf{I}$, which seeks to learn **P** with normalized and mutually uncorrelated rows. Seemingly plausible at the first glance, that would lead to another set of trivial solutions, i.e., $\mathbf{A} \approx \mathbf{0}$ and $\mathbf{P}\mathbf{X} \approx \mathbf{0}$, implying that **P** is always nearly orthogonal to **X** and **PX** hardly captures any meaningful information. We recall that solving the following form gives out $\mathbf{P}^*$ as the $d \times p$ PCA projection matrix of **X** ($\mathbf{I}_d$ denotes a $d \times d$ identity matrix):

$$\min_{\mathbf{P}} ||\mathbf{P}^\mathbf{T}\mathbf{P}\mathbf{X} - \mathbf{X}||_F^2, \quad s.t. \quad \mathbf{P}\mathbf{P}^T = \mathbf{I}_d. \tag{4}$$

Inspired by (4), one viable regularization would be to minimize the information loss through the transformation **P**, measured by $||\mathbf{P}^\mathbf{T}\mathbf{P}\mathbf{X} - \mathbf{X}||_F^2$, in addition to $\mathbf{P}\mathbf{P}^T = \mathbf{I}$. That leads to the form below (β is a constant):

$$\begin{array}{c} \min_{\mathbf{D},\mathbf{A},\mathbf{P}} \frac{1}{2}||\mathbf{D}\mathbf{A} - \mathbf{P}\mathbf{X}||_F^2 + \lambda \sum_{i=1}^n ||a_i||_\infty + \frac{\beta}{2}||\mathbf{P}^\mathbf{T}\mathbf{P}\mathbf{X} - \mathbf{X}||_F^2, \\ s.t. \quad ||d_j|| \leq 1, j = 1,2,...,c; \ \mathbf{P}\mathbf{P}^T = \mathbf{I}_d. \end{array} \tag{5}$$

In order to alleviate the entangled constraints and avoid higher-order terms on **P**, we introduce an auxiliary variable $\mathbf{Q} \in \mathbb{R}^{d \times p}$, and relax (5) as:

$$\begin{array}{c} \min_{\mathbf{D},\mathbf{A},\mathbf{P},\mathbf{Q}} \frac{1}{2}||\mathbf{D}\mathbf{A} - \mathbf{P}\mathbf{X}||_F^2 + \lambda \sum_{i=1}^n ||a_i||_\infty + \frac{\beta}{2}(||\mathbf{Q}^T\mathbf{P}\mathbf{X} - \mathbf{X}||_F^2 + ||\mathbf{P} - \mathbf{Q}||_F^2), \\ s.t. \quad ||d_j|| \leq 1, j = 1,2,...,c; \ \mathbf{Q}\mathbf{Q}^T = \mathbf{I}_d. \end{array} \tag{6}$$

The formulation (6) will be the **default TASH model** throughout the paper.

Previous anti-sparse models [1, 10] are simplistic instances of TASH (6) that learn only **A**, with fixed **D**, **P** being the identity matrix and no **Q**. Another well-known hashing model, ITQ [5], is also related to (6). To see this, one may let $d = c$, fix **P** as the PCA matrix, set **D** as a unitary (rotation) matrix to be jointly learned, and view ℓ_∞-regularization as a convex relaxation from the nonconvex binary constraint.

After obtaining **D** and **P** from the training model (6), the testing stage for any x_{new} solves:

$$\min_{a_{new}} \frac{1}{2}||\mathbf{D}a_{new} - \mathbf{P}x_{new}||_2^2 + \lambda||a_{new}||_\infty. \tag{7}$$

3.2 Algorithm Development

We hereby present an iterative coordinate descent algorithm for (6). At the k-th iteration (k = 0, 1, 2, ...), we sequentially solve the following four subproblems.

A-subproblem The **A**-subproblem can be updated separably for each a^i:

$$\min_{a_i} \frac{1}{2}||\mathbf{D}_k a_i - \mathbf{P}_k x_i||_2^2 + \lambda||a_i||_\infty, \quad i = 1,2,...,n. \tag{8}$$

We implement the sub-differential set-based algorithm proposed by [4] to solve (8), which is similar to path-following methods based on continuation techniques.

D-subproblem The **D**-subproblem updates $\mathbf{D}_{k+1}$ via a least-squares problem with spherical constraints. It is solved using the Lagrange dual method [15] for efficiency, as only c dual variables are involved:

$$\min_{\mathbf{D}} \frac{1}{2}||\mathbf{D}\mathbf{A}_{k+1} - \mathbf{P}_k\mathbf{X}||_F^2 \quad s.t. \quad ||d_j|| \leq 1, j = 1,2,...,c. \tag{9}$$

P-subproblem The **P**-subproblem solves an unconstrained quadratic optimization, with an closed-form update formula:

$$\mathbf{P}_{k+1} = \{\mathbf{D}_k\mathbf{A}_k\mathbf{X}^T + \beta\mathbf{Q}_k(\mathbf{X}\mathbf{X}^T + \mathbf{I}_p)\} * \{(1+\beta)\mathbf{X}\mathbf{X}^T + \beta\mathbf{I}_p\}^{-1}. \tag{10}$$

Q-subproblem The **Q**-subproblem minimizes a quadratic form with the orthogonality constraint (let $\mathbf{Z} = \mathbf{P}_{k+1}\mathbf{X}$ for simplicity):

$$\min_{\mathbf{Q}} ||\mathbf{Q}^T\mathbf{Z} - \mathbf{X}||_F^2 + ||\mathbf{P}_{k+1} - \mathbf{Q}||_F^2, \quad s.t. \quad \mathbf{Q}\mathbf{Q}^T = \mathbf{I}_d. \tag{11}$$

Despite its concise form, solving (11) can be difficult due to the non-convex constraint. The feasible set of (11) is often referred to as the Stiefel manifold, and can lead to many local minimizers [31]. We develop an algorithm based on the alternating direction method of multipliers (ADMM) to efficiently solve (11). By variable splitting and introducing another auxiliary variable $\mathbf{R} \in \mathbb{R}^{d\times p}$, (11) can be first re-written as:

$$\min_{\mathbf{Q},\mathbf{R}} ||\mathbf{Q}^T\mathbf{Z} - \mathbf{X}||_F^2 + ||\mathbf{P}_{k+1} - \mathbf{Q}||_F^2, \quad s.t. \quad \mathbf{Q} = \mathbf{R}, \mathbf{R}\mathbf{R}^T = \mathbf{I}_d. \tag{12}$$

The augmented Lagrangian function of (12) is:

$$||\mathbf{Q}^T\mathbf{Z} - \mathbf{X}||_F^2 + ||\mathbf{P}_{k+1} - \mathbf{Q}||_F^2 + <\mathbf{Y}, \mathbf{Q} - \mathbf{R}> + \frac{\delta}{2}||\mathbf{Q} - \mathbf{R}||_F^2 + \Phi(\mathbf{R}). \tag{13}$$

Here $Y \in \mathbb{R}^{d\times p}$ is the Lagrange multiplier attached to the equality constraint, δ is a positive constant with the default value 1, and $\Phi(\mathbf{R})$ is the penalty function that takes 0 when $\mathbf{R}\mathbf{R}^T = \mathbf{I}_d$ and infinity elsewhere. ADMM minimizes (13) with respect to **Q**, **R** and updates **Y** in an alternating manner (t = 0, 1, 2... denotes the iteration number). Minimizing (13) w.r.t. **Q** enjoys the closed-form solution:

$$\mathbf{Q}_{t+1} = (\mathbf{Z}\mathbf{Z}^T + (\delta+1)\mathbf{I}_d)^{-1} * (\mathbf{Z}\mathbf{X}^T + \mathbf{P} - \mathbf{Y}_t + \delta\mathbf{R}_t) \tag{14}$$

Minimizing (13) over **R** leads to the following problem:

$$\min_{\mathbf{R}} \Phi(\mathbf{R}) - <\mathbf{Y}_t, \mathbf{R}> + \frac{\delta}{2}||\mathbf{Q}_{t+1} - \mathbf{R}||_F^2, \tag{15}$$

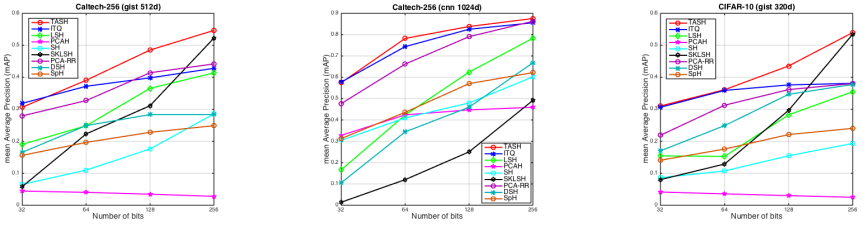
whose solution is characterized below (the proof follows the proof of Theorem 2.1 in [3]):

Theorem 3.1. *The optimal solution of (15) is* $\mathbf{R} = \mathbf{U}\mathbf{V}^T$, *where* $\mathbf{U} \in \mathbb{R}^{d\times d}$, $\mathbf{V} \in \mathbb{R}^{p\times d}$ *are composed of the top-d left and right singular vectors of* $\mathbf{Q}_{t+1} + \frac{\mathbf{Y}_t}{\delta}$, *respectively.*

Y is then updated as: $\mathbf{Y}_{t+1} = \mathbf{Y}_t + \delta(\mathbf{Q}_{t+1} - \mathbf{R}_{t+1})$.

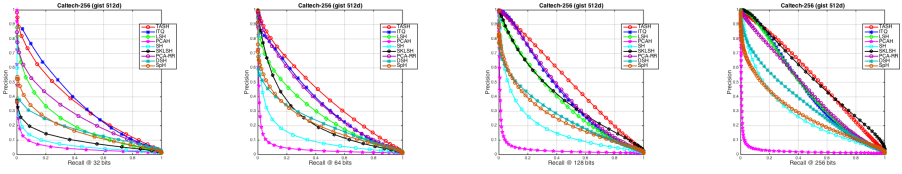
3.3 Complexity Analysis

For each a_i problem (8), the complexity of one iteration is dominated by the inverse of a Gram matrix of a dynamic partition of **D**, which is (loosely) upper-bounded by $O(c^3)$ when applying Gauss-Jordan elimination. It thus roughly takes $O(C_1nc^3)$ to update the entire **A** once, where C_1 is a constant accounting for iterations, etc. For optimizing (9), each step inverses a $c \times c$ matrix, leading to the complexity of $O(C_2c^3)$. Computing (10) causes a negligible cost of $O(dcn + dpn + dp^2)$, as the matrix inverse can be pre-calculated and no iteration is used. The **Q**-subproblem has a complexity of $O(C_4pd^2)$, owing to the singular value decomposition required to solve (15) per iteration.



(a) (b) (c)

Figure 1: The average mAP comparisons at different numbers of bits: (a) Caltech-256 (gist 512d); (b) Caltech-256 (cnn 1024d); (c) CIFAR-10 (gist 320d).



(a) (b) (c) (d)

Figure 2: The comparisons of precision-recall curves on Caltech-256 (gist 512d) at: (a) 32 bits; (b) 64 bits; (c) 128 bits; (d) 256 bits.

The overall complexity of the iterative coordinate descent algorithm for training TASH is hence: $O(C'_1nc^3 + C'_2c^3 + C'_3(dcn + dpn + dp^2) + C'_4pd^2)$, where C'_1, C'_2, C'_3 and C'_4 are all constants absorbing outer loop, and inner loop iterations when necessary. The overall complexity scales linearly with the data volume n, quadratically with the input feature dimension p, quadratically with the transformed dimension d, and cubically with the hash length c. For the testing stage (7), its complexity is the same as the **A**-subproblem: $O(C_1c^3)$ per sample.

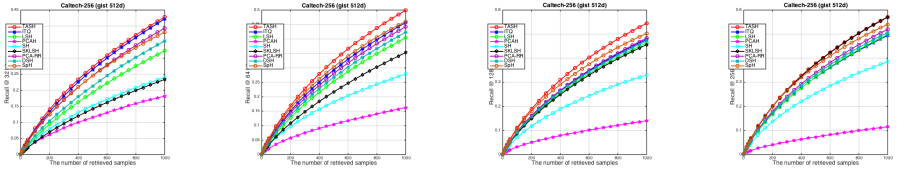
In practice, the main efficiency bottleneck lies in solving the **A**-subproblem per iteration. That implies a potential acceleration strategy to solve all a_i problems parallelly. Increasing c has the most notable impact on the practical running time. The influences of raising d and p are not as obvious. Besides, we usually only need to solve **A**, **D**, and **Q** subproblems inexactly, which reduces C'_1, C'_2 and C'_4.

4 Experiments for Unsupervised Hashing

4.1 Comparison with State-of-the-Art Methods

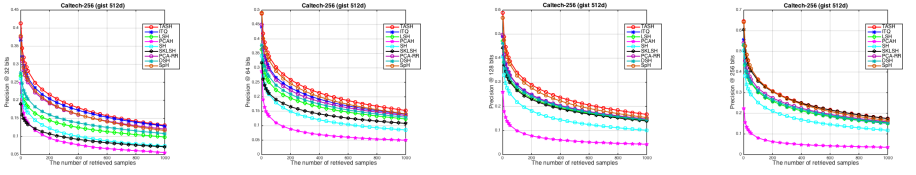
We evaluate our methods on two popular benchmarks: Caltech-256 [6] and CIFAR-10 [12]. Both have been widely used to evaluate unsupervised hashing methods. Caltech-256 consists of 29,780 images, each of which is associated with one of 256 object categories. CIFAR-20 contains 60,000 images from 10 classes. We use 512-dimension GIST [19] descriptors and 1024-dimension CNN features from the VGG model [21] to create two views of Caltech-256, referred to as *Caltech-256 (gist 512d)* and *Caltech-256 (cnn 1024d)*, respectively. We use 320-dimension GIST descriptors to produce the *CIFAR-10 (gist 320d)* set. In each of these datasets, we form a query set by randomly choosing 1,000 samples and construct a non-overlapping training set using the rest[1].

[1]We used the pre-processed datasets as well as visualization tools from: https://github.com/willard-yuan/hashing-baseline-for-image-retrieval



(a) (b) (c) (d)

Figure 3: The comparison of curves of recall versus number of retrieved samples on Caltech-256 (gist 512d) at: (a) 32 bits; (b) 64 bits; (c) 128 bits; (d) 256 bits.



(a) (b) (c) (d)

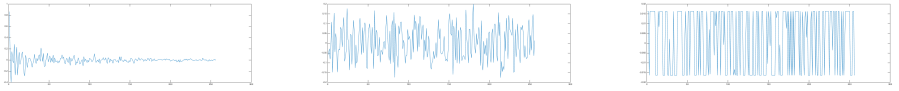
Figure 4: The comparison of curves of precision versus number of retrieved samples on Caltech-256 (gist 512d) at: (a) 32 bits; (b) 64 bits; (c) 128 bits; (d) 256 bits.

We compare TASH with a variety of state-of-the-art unsupervised hashing methods: **ITQ** [5]; PCA Hashing (**PCAH**); [13]; PCA projection followed by random orthogonal transformation (**PCA-RR**) [9]; Spectral Hashing (**SH**) [28]; Spherical Hashing (**SpH**) [7]; Local Sensitive Hashing (**LSH**) [2]; Shift-invariant Kernel-based Locality-sensitive Hashing (**SKLSH**) [20]; Density Sensitive Hashing (**DSH**) [11]. LSH and SKLSH rely only on randomized data-independent linear projections, while all others utilize the training dataset.

We initialize $\mathbf{D}$ using a tight frame $\mathbf{D}_0$, generated by QR decomposition from an i.i.d. Gaussian matrix. We initialize both $\mathbf{P}$ and $\mathbf{Q}$ as the PCA matrix of $\mathbf{X}$. These proper initializations make TASH converge fast, and for simplicity we use 10 iterations for all experiments. TASH also has a few hyperparameters to decide. d is usually chosen no larger than c: we fix $d = c$ here, and will discuss the trade-off of d in the next section. We select λ proportionally to c: $\lambda = \delta c$, while in this paper we use the same empirical $\delta = \frac{5}{128}$ for all datasets. However, we observe that fine-tuning δ per each dataset may lead to TASH performance better than reported. The default value of β is 5.

We vary the hashing code length c from 32 bits to 256 bits, to evaluate the performance of all the methods on compact codes and relatively long codes. We adopt the following four classical criteria for evaluation: 1) the *mean average precision* (mAP); 2) the *precision and recall* (PR) curve; 3) the recall values at different number of retrieved samples; 4) the precision values at different number of retrieved samples. Each measure is computed for all comparison methods at different numbers of bits (32, 64, 128, 256).

Fig. 1 (a) compares all the methods in terms of mAP on the Caltech-256 (gist 512d) set. While PCAH fails to perform well due its simplistic nature, two relevant methods, ITQ and PCA-RR, show competitive results for most code sizes. ITQ even marginally outperform TASH at 32 bits. But as the bit number rises, ITQ and PCA-RR lag behind TASH with growingly large margins. Another notable competitor, SKLSH, shows a strongly upward trajectory and gets the second best mAP at 256 bits. It is owing to the theoretical convergence guarantee of SKLSH that the Hamming distance between binary codes can precisely approximate the distance in the kernel space when enough bits are assigned. LSH also im-



(a) (b) (c)

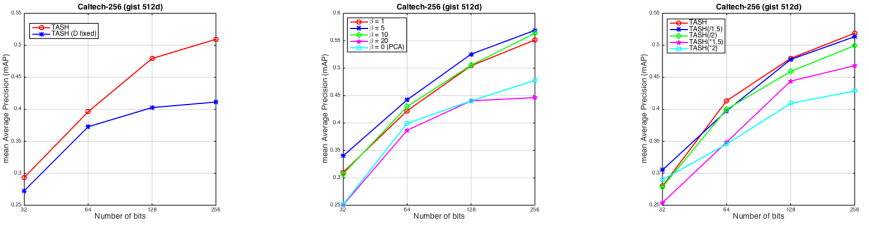
Figure 5: The learned representations of one test sample from Caltech-256 (gist 512d), through (a) PCAH, (b) ITQ and (c) TASH models.

proves as the code length increases, and almost reaches the performance level of ITQ and PCA-RR at 256 bits. However, both SKLSH and LSH have unsatisfactory performance at small number of bits (the mAP of SKLSH is almost as poor as that of PCAH at 32 bits). That observation certifies the importance of learning from data, in particular when the code length is small. Finally, while some methods (ITQ, PCA-RR) perform better with compact codes and some others (SKLSH, LSH) better for longer codes, TASH outperforms all its competitors in almost all cases, with noticeable margins.

Figs. 1 (b) and (c) demonstrate the mAP results on Caltech-256 (cnn 1024d) and CIFAR-10 (gist 320d), respectively. Fig. 1 (c) share similar trends as (a). It is interesting to compare Figs. 1 (a) and (b) to find out that more sophisticated features (such as CNN features) greatly benefit the mAPs of data-dependent hashing methods, such as ITQ, PCA-RR and even PCAH, while SKLSH seems to be negatively affected here. In both settings, TASH maintains its persistent advantage over all other competitors.

Fig. 2 further investigates the key trade-off between the precision and recall of all methods on Caltech-256 (gist 512d). PCAH actually deteriorates as the number of bits increases. ITQ and SKLSH performs comparably with TASH at 32 bits and 256 bits, respectively. But TASH is the only one that maintains top performance at all numbers of bits, and significantly outperform all else at 64 and 128 bits. Figs. 3 and 4 record the precision and recall changes w.r.t the number of retrieved samples on Caltech-256 (gist 512d), both of which re-confirm the consistent performance superiority and robustness of TASH. Interestingly, some methods with less competitive mAPs show promising in these two curves, such as SpH.

4.2 Visualization and Analysis



(a) (b) (c)

Figure 6: The average mAP comparisons on Caltech-256 (gist 512d), with (a) learned or fixed dictionary $\mathbf{D}$, (b) different coefficients β and (c) different transformed dimensions d.

We visualize three learned representations (before quantization) of one sample from Caltech-256 (gist 512d), in Fig. 5, by PCAH, ITQ and TASH, respectively. It is easy to see that the quantization from the PCAH feature (a) will suffer the strongest distortion in the binarization step, since most energy concentrates on a few first elements, which also impairs

the discriminability of the resulting binary codes. The ITQ feature (b) greatly alleviates the problem by rotating the PCA feature and re-balancing the magnitudes. The TASH feature (c) will have the minimum quantization error, as it naturally resembles an antipodal signal and most of its entries already take the two extreme values.

Fig. 6 (a) compares solving TASH with the learned dictionary $\mathbf{D}$, and with the fixed tight frame $\mathbf{D}_0$. As expected, the capability to adapt the basis to the training data greatly benefits TASH, especially for large c. Fig. 6 (b) studies the effect of choosing β, with the baseline $\beta = 0$ defined as using PCA matrix for $\mathbf{P}$ and no $\mathbf{Q}$. Clearly, learning the transform $\mathbf{P}$ contributes dramatically to the outstanding performance of TASH, as compared to the baseline. On the other hand, overly large βs like 10 and 20 start to weaken the results. When $\beta \to \infty$, the objective of (6) turns close to that of (3), which is also reflected by the major overlapped parts between the β =20 and $\beta = 0$ (PCA) curves.

Fig. 6 (c) presents particularly interesting results, on **varying the transformed dimension** d. Our default TASH models sets $d = c$. We also test and compare the performance of different TASH models with $d = c/1.5$, $d = c/2$, $d = c \times 1.5$, $d = c \times 2$. As the main observation, it is always preferable to choose $d \leq c$ than $d > c$. Moreover, a smaller $d \leq c$ is not necessarily more favorable, e.g., $d = c/2$, due to heavier information loss in the dimensionality reduction process. The optimal d is thus decided by the **trade-off** between: 1) adopting reasonably large d in order to preserve more information during the transform; and 2) keeping d sufficiently small w.r.t. c, which usually implies that more elements of anti-sparse solutions are stuck to the extreme. Actually, the two curves of $d = c/1.5$ and $d = c$ get fairly close: the former is even slightly better at 32 bits.

5 Discussions and Conclusions

The paper contributes both theoretically and practically to making anti-sparse models more effective for hashing. The proposed TASH model jointly learns a dimension-reduction transform, a dictionary and the anti-sparse representations in a unified formulation. We evaluate TASH on several real-world datasets and compare it against a large variety of unsupervised hashing methods, which confirms its highly competitive performance and strong robustness. Visualizations and analysis add to the explainability of results. Since the $\mathbf{A}$-subproblem is relatively costly to solve, whose complexity strongly depends on the data scale n and the hash length c, our immediate future work is to learn its fast approximations [25, 26].

References

[1] Raghavendran Balu, Teddy Furon, and Hervé Jégou. Beyond project and sign for cosine estimation with binary codes. In *ICASSP*, pages 68884–6888. IEEE, 2014.

[2] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Symposium on Theory of computing*, pages 380–388. ACM, 2002.

[3] Xi Chen, Yanjun Qi, Bing Bai, Qihang Lin, and Jaime G Carbonell. Sparse latent semantic analysis. In *SDM*, pages 474–485. SIAM, 2011.

[4] Jean-Jacques Fuchs. Spread representations. In *ASILOMAR Conference*, pages 814–817. IEEE, 2011.

[5] 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*, 35(12): 2916–2929, 2013.

[6] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[7] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *CVPR*, pages 2957–2964. IEEE, 2012.

[8] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Symposium on Theory of Computing*, pages 604–613. ACM, 1998.

[9] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311. IEEE, 2010.

[10] Hervé Jégou, Teddy Furon, and Jean-Jacques Fuchs. Anti-sparse coding for approximate nearest neighbor search. *arXiv preprint arXiv:1110.3767*, 2011.

[11] Zhongming Jin, Cheng Li, Yue Lin, and Deng Cai. Density sensitive hashing. *IEEE Transactions on Cybernetics*, 44(8):1362–1371, 2014.

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

[13] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137. IEEE, 2009.

[14] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015.

[15] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. *NIPS*, 19:801, 2007.

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

[17] Xiaoqiang Lu, Xiangtao Zheng, and Xuelong Li. Latent semantic minimal hashing for image retrieval. *IEEE Transactions on Image Processing*, 2016.

[18] Yusuke Matsushita and Toshikazu Wada. Principal component hashing: An accelerated approximate nearest neighbor search. In *Pacific-Rim Symposium on Image and Video Technology*, pages 374–385. Springer, 2009.

[19] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 2001.

[20] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517, 2009.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] Christoph Studer, Tom Goldstein, Wotao Yin, and Richard G Baraniuk. Democratic representations. *arXiv preprint arXiv:1401.3420*, 2014.

[23] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[24] Xinchao Wang, Zhu Li, and Dacheng Tao. Subspaces indexing model on grassmann manifold for image search. *IEEE Transactions on Image Processing*, 20(9):2627–2635, 2011.

[25] Zhangyang Wang, Qing Ling, and Thomas S Huang. Learning deep ℓ_0 encoders. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[26] Zhangyang Wang, Yingzhen Yang, Shiyu Chang, Qing Ling, and Thomas S Huang. Learning a deep ℓ_∞ encoder for hashing. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2174–2180. AAAI Press, 2016.

[27] Zhaowen Wang, Jianchao Yang, Haichao Zhang, Zhangyang Wang, Yingzhen Yang, Ding Liu, and Thomas S Huang. *Sparse Coding and its Applications in Computer Vision*. World Scientific, 2015.

[28] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2009.

[29] Bihan Wen, Saiprasad Ravishankar, and Yoram Bresler. Frist-flipping and rotation invariant sparsifying transform learning and applications to inverse problems. *arXiv preprint arXiv:1511.06359*, 2015.

[30] Bihan Wen, Saiprasad Ravishankar, and Yoram Bresler. Structured overcomplete sparsifying transform learning with convergence guarantees and applications. *International Journal of Computer Vision*, 2015.

[31] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[32] Xiaofeng Zhu, Zi Huang, Hong Cheng, Jiangtao Cui, and Heng Tao Shen. Sparse hashing for fast multimedia search. *ACM Transactions on Information Systems*, 31(2): 9, 2013.