Fast content identification in high-dimensional feature spaces using Sparse Ternary Codes

Sohrab Ferdowsi, Slava Voloshynovskiy, Dimche Kostadinov, Taras Holotyak
Department of Computer Science, University of Geneva
7 route de Drize, 1227 Carouge, Switzerland

***Abstract*—We consider the problem of fast content identification in high-dimensional feature spaces where a sub-linear search complexity is required. By formulating the problem as sparse approximation of projected coefficients, a closed-form solution can be found which we approximate as a ternary representation. Hence, as opposed to dense binary codes, a framework of Sparse Ternary Codes (STC) is proposed resulting in sparse, but robust representation and sub-linear complexity of search. The proposed method is compared with the Locality Sensitive Hashing (LSH) and the memory vectors on several large-scale synthetic and public image databases, showing its superiority.**

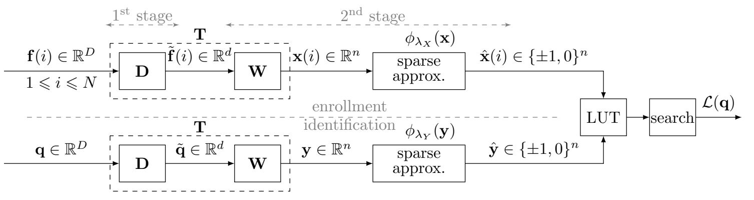

1st stage
2nd stage
T
$\mathbf{f}(i) \in \mathbb{R}^D$
$1 \leqslant i \leqslant N$
D
$\tilde{\mathbf{f}}(i) \in \mathbb{R}^d$
W
$\mathbf{x}(i) \in \mathbb{R}^n$
$\phi_{\lambda_X}(\mathbf{x})$
sparse approx.
$\hat{\mathbf{x}}(i) \in \{\pm 1, 0\}^n$
enrollment
identification
LUT
search
$\mathcal{L}(\mathbf{q})$
T
$\mathbf{q} \in \mathbb{R}^D$
D
$\tilde{\mathbf{q}} \in \mathbb{R}^d$
W
$\mathbf{y} \in \mathbb{R}^n$
$\phi_{\lambda_Y}(\mathbf{y})$
sparse approx.
$\hat{\mathbf{y}} \in \{\pm 1, 0\}^n$

Fig. 1: The schematic diagram of the proposed system. $\mathbf{D}$ is an energy-compacting operator which is used for de-correlating the features. $\mathbf{W}$ is a random projection matrix.

I. Introduction

Many security applications require fast and reliable identification from high-dimensional extracted features. The identification is based on a similarity search between a given query feature vector and those in a database of registered items. If the database is large, i.e., typically hundreds of thousands or millions of items, the linear search required to match the query with the items in the database becomes the bottleneck of processing, if not prohibitive.

A typical identification problem includes, but it is not limited to: *people identification* based on their biometrics, *physical object identification* based on the graphical design of printed matters on object surface and Physical Unclonebale Functions (PUFs), *device identification* based on sensor fingerprints and *multimedia content identification* for copyright protection, filtering and tracking. The generalized identification system architecture under analysis is shown in Figure 1. For this problem, the current trend in high dimensional feature extraction is based on either the usage of last layers of deep nets trained in unsupervised or supervised way, a.k.a. *neural codes* [1], or aggregation of low- or mid- dimensional local descriptors such as SIFT, ORB, KAZE, etc., resulting in high dimensional global features of the range of several thousands. Among the successful methods in this line of work are BOV [2], VLAD [3], Fisher Vector (FV) [4], and their recent interesting generalizations such as triangulation embedding [5] and generalized max-pooling (GMP) [6].

The resulting high-dimensional descriptors are collected in a database consisting of N enrolled feature vectors and the identification system should produce a list $\mathcal{L}(\mathbf{q})$ of indices of enrolled features $\mathbf{f}(i) \in \mathbb{R}^D$, $1 \leq i \leq N$ closest to the probe feature vector $\mathbf{q} \in \mathbb{R}^D$. The complexity of direct identification using linear scan is of $\mathcal{O}(DN)$.

At the same time, many algorithms have been proposed aiming at reducing the search complexity. In particular, a large family of methods uses indexing structures like k-d tree, achieving a sub-linear search complexity [7]. However, the efficiency of these methods are strictly bounded to very low-dimensional features around several tens, making them far from appealing for the above applications.

Another very popular line of research aims at extracting compact binary codes to replace original features. The search is then performed in the Hamming space of codes, in some cases providing the further advantage of more efficient memory storage. One group of methods in this family is based on projection followed by binarization. Among these methods, there is the popular family of LSH (e.g., [8], [9] or [10]) where the projection is chosen from a pre-defined set (mainly random projections), or is learned to adapt to the data or satisfy some properties of codes (e.g., [11], also see [12], [13]). Another group of methods produce binary codes by different kinds of quantization, e.g., product quantization as in [14] and [15].

These methods, although can reduce the search time, their complexity is still exhaustive and is linear in $\mathcal{O}(N)$. In fact, they gain in search time by replacing the floating-point operations with those of binary codes. Furthermore, in order to maintain a small false negative rate, necessary for successful identification, the length of binary codes should be chosen large. Other than storage issues, this will result also in a higher false positive rate.

Memory vectors [16] and group testing [17] are among the first methods that discuss the problem of similarity search in high-dimensional spaces achieving sub-linear complexities. It should also be mentioned that LSH family can also provide sub-linear search complexity using multiple hash tables, however, this requires a prohibitive memory. As an example of LSH, the algorithm proposed in [18] requires $\mathcal{O}(DN^\rho)$ search complexity and $\mathcal{O}(DN^{1+\rho})$ memory storage, where ρ is a parameter that should be chosen slightly less than one to maintain high search performance.

This work introduces a scheme (Fig. 1) to address the problem of low-complexity search in high dimensional spaces. We make the following **contributions**: (1) We formulate the hashing problem as sparse feature approximation which results in hard-thresholding in the projected domain. (2) As an approximation to this optimal solution, we introduce *Sparse Ternary Codes* (STC) as an alternative to binary codes providing two advantages. First, we show that they produce more robust values avoiding the high false positive rate of binary codes due to bit-flipping. Second, they naturally provide a data structure suitable for hashing and sub-linear complexity search. (3) We perform experiments on several large scale (1 Mio), high-dimensional synthetic and public databases and provide competitive results.

The rest of the paper is organized as follows. In section II we briefly introduce the problem of Nearest Neighbour search. In section III we pose the problem of hashing as sparse representation and discuss its solution. In section IV we describe the proposed architecture. We conduct several experiments on different databases in section V. Finally, section VI concludes the paper.

Notation remark: All vectors are considered as column vectors and are represented using bold face, e.g., $\mathbf{x}(i) = [x_1(i), \cdots, x_j(i), \cdots, x_n(i)]^T$, where j indexes the vector elements and i indexes the vector itself. Matrices are depicted with capital boldface letters. The identity matrix of dimension D is denoted as $\mathbf{I}_D$ and the indicator function $\mathbb{1}_{\{E\}}$ equals unity, if E occurs and equals zero, otherwise.

II. Problem formulation: Nearest Neighbour search

Given a query $\mathbf{q} = [q_1, \cdots, q_D]^T \in \mathbb{R}^D$, we consider the identification problem as finding the Nearest Neighbours (NN's) to the features stored in the database $\mathcal{F} = [\mathbf{f}(1), \cdots, \mathbf{f}(i), \cdots, \mathbf{f}(N)]$, where $\mathbf{f}(i) = [f_1(i), \cdots, f_D(i)]^T \in \mathbb{R}^D$. The similarity is considered in terms of distance similarity $d(\mathbf{q}, \mathbf{f}(i)) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^+$, where $d(\cdot,\cdot)$ is usually assumed to be the Euclidean distance $d(\mathbf{q}, \mathbf{f}(i)) = ||\mathbf{q} - \mathbf{f}(i)||_2$.

The identification system produces a list $\mathcal{L}(\mathbf{q})$ of either k most similar items or all items within a specified distance ϵD, with $\epsilon > 0$, such that:

$$\mathcal{L}(\mathbf{q}) = \{1 \leqslant i \leqslant N : d(\mathbf{q}, \mathbf{f}(i)) \leqslant \epsilon D\}, \tag{1}$$

which can be ordered according to their distances to $\mathbf{q}$.

We will consider Approximate Nearest Neighbour (ANN) solutions with complexity of search that is a fraction of the direct search complexity $\mathcal{O}(DN)$.

III. The STC: from sparse coding to hashing

In this section, we consider the problem of hashing from a signal approximation point of view and formulate it as sparse approximation of projected data which results in a ternary representation.

A. From feature approximation to sparse code approximation

Many problems in signal processing and classification are based on the sparse feature approximation which is an inverse problem in nature and can be generalized as [19]:

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathcal{X}^n}{\operatorname{argmin}} ||\mathbf{f} - \mathbf{A}\mathbf{x}||_2^2 + \beta \Omega(\mathbf{x}), \tag{2}$$

where $\mathcal{X}$ is the alphabet of sparse approximation vector, $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_n] \in \mathbb{R}^{D \times n}$ is a codebook, β is a regularization parameter and $\Omega(\mathbf{x})$ is chosen to encourage some desired properties for representations such as sparsity or inter- and intra-class code similarity.

In the general case, the feature approximation problem is known to be NP-hard for a generic $\Omega(\mathbf{x})$ and various surrogate norms are applied to get an approximate solution in practice.

In contrast, in this paper, we consider a transform-based approach to approximation, where, rather than the feature $\mathbf{f}$ itself, its projection $\mathbf{Tf}$ is approximated as:

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathcal{X}^n}{\operatorname{argmin}} ||\mathbf{Tf} - \mathbf{x}||_2^2 + \beta \Omega(\mathbf{x}). \tag{3}$$

This is a direct problem[1] and can be solved in closed-form for some important regularizations like $\Omega(\mathbf{x}) = ||\mathbf{x}||_1$ and $\Omega(\mathbf{x}) = ||\mathbf{x}||_0$[2]. In fact, the latter case based on ℓ_0 norm, which we consider in this work, is of particular interest for the search problem where a compact representation is desired for $\mathbf{x}$. In this case, a closed-form solution is derived as [19]:

$$\mathbf{x}^* = \psi_\lambda(\mathbf{Tf}), \tag{4}$$

where λ is a constant value and the hard-thresholding function is defined as $\psi_\lambda(t) = t \cdot \mathbb{1}_{\{|t| > \lambda\}}$. Therefore, $\psi_\lambda(\cdot)$ essentially keeps only the strong values of $\mathbf{Tf}$ and sets the values with low magnitude to zero (Fig. 2a).

B. Hashing as sparse approximation

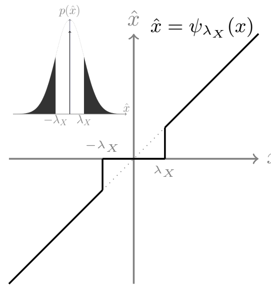
The sparse code formulation (3) bears an important similarity with the existing hashing methods. With the mapping function $\hat{x} = sign(x)$ (with $sign(x) = +1$, if $x \geqslant 0$ and -1, otherwise), one obtains a classical binary hashing with $\mathcal{X} = \{-1, +1\}$. However, according to (4), since the resulting coefficients are real valued and require additional storage and search complexity, in order to provide a compact representation for hashing and fast search, rather than dense binary hashing (Fig. 2c), a more accurate and a natural approximation of this solution should be in the form of sparse ternary (Fig. 2b). In other words, the ternary mapping $\phi_\lambda(\cdot)$, is a "hard-coded" version of $\psi_\lambda(\cdot)$. Obviously, if $\lambda = 0$, the STC reduces to the binary hash mapping with $\mathcal{X} = \{-1, +1\}$ shown in Fig. 2c.

The resulting alphabet of the ternarized projection is $\mathcal{X} = \{-1, 0, +1\}$. We will consider two different thresholds, λ_X and λ_Y for the enrollment and query sides, respectively[3]. The choice of these thresholds directly influences α_X and α_Y, the sparsity ratios of the enrolment and the query codes, respectively. These sparsity ratios play important roles in the overall performance and memory storage of the system.

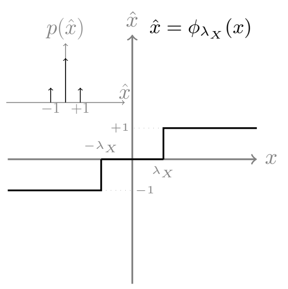
[1]Refer to [20] for a more detailed description of direct vs. inverse problems in signal approximation.

[2]Problems (2) and (3) are equivalent when $\mathbf{A}$ is orthogonal and $\mathbf{T} = \mathbf{A}^T$.

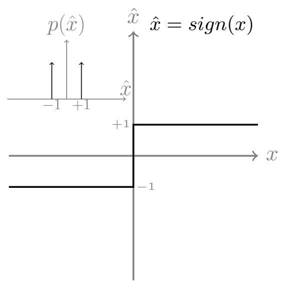
[3]Therefore, the non-linearities of the encoder and decoder can be different.



(a) hard thresholding



(b) ternarization



(c) binarization

Fig. 2: Thresholding functions.

IV. Proposed framework: Sparse Ternary Codes

In practice, the feature vectors might not be uniformly distributed in the D-dimensional space and tend to form clusters in $\mathbb{R}^D$. To proceed with the uniform approach to the features of different origin, we can decompose the transform matrix $\mathbf{T}$ introduced in section III in two stages as $\mathbf{T} = \mathbf{WD}$. The first stage $\mathbf{D}$ is of signal processing nature, like the DCT, PCA or dictionary learning. The second stage is the hashing stage where a random projection matrix $\mathbf{W}$ is used to project the decorrelated features $\tilde{\mathbf{f}}$ (or $\tilde{\mathbf{q}}$ on the query side) to the space of $\mathbb{R}^n$, providing $\mathbf{x}$ (and $\mathbf{y}$ for query).

A. Proposed algorithm

The search algorithm consists of the enrolment and search stages (Fig. 1). At the enrolment stage, all feature vectors $\{\mathbf{f}(i)\}_{i=1}^N$ of the database $\mathcal{F}$ are converted to the STC representations $\{\hat{\mathbf{x}}(i)\}_{i=1}^N = \text{STC}(\{\mathbf{f}(i)\}_{i=1}^N)$ and described in the form of inverted file or LUT. Algorithm 1 discusses this procedure in more details.

At the search stage, after the query $\mathbf{q}$ is encoded to the ternary $\hat{\mathbf{y}}$, its non-zero elements refer to the corresponding elements of the look-up table where the list of registered database items are retrieved. The voting vector $\mathbf{v}(\mathbf{q})$ of size N is initialized by zero. Each time an item has a match, its current value of $\mathbf{v}(\mathbf{q})$ is increased by a constant factor ν. The mismatch values, i.e., the cases where the sign of active coefficients are different, that item could be penalized in the voting procedure by a constant ν'.

After all the relevant items are counted, the list $\mathcal{L}(\mathbf{q})$ is generated by picking the largest $k = |\mathcal{L}(\mathbf{q})|$ values of $\mathbf{v}(\mathbf{q})$. In fact, the voting vector $\mathbf{v}(\mathbf{q})$ is a very sparse vector with most values near or exactly zero. To achieve further sparsity, values below an empirically chosen threshold could be set to zero.

Algorithm 1 database enrollment using STC

```
Input: database items $\mathcal{F} = [\mathbf{f}(1), \cdots, \mathbf{f}(i), \cdots, \mathbf{f}(N)]^T$,
    knowledge of the query noise level
Output: Look-Up-Tables $LUT^+$ and $LUT^-$
 1: Initialize empty lists $LUT^+ = \{LUT_1^+, \cdots, LUT_n^+\}$ and
    $LUT^- = \{LUT_1^-, \cdots, LUT_n^-\}$.
 2: Depending on the signal, choose $\mathbf{D}$ as a suitable energy
    compacting operator (dictionary).
 3: Generate elements of $\mathbf{W}_{(d \times n)}$ independently from the
    unit-norm, zero-mean Gaussian distribution and orthogo-
    nalize the columns using Gram-Schmidth.
 4: Choose $\lambda_X$ w.r.t the memory/complexity constraints and
    the noise level
 5: for $i = 1, \cdots, N$ do
 6:     $\tilde{\mathbf{f}}(i) = \mathbf{D}[\mathbf{f}(i)]$   ▷ Energy compaction depends on the
    type of $\mathbf{D}$
 7:     $\mathbf{x}(i) = \mathbf{W}^T \tilde{\mathbf{f}}(i)$
 8:     $\hat{\mathbf{x}}(i) = \phi_{\lambda_X}(\mathbf{x}(i))$
 9:     for $j = 1, \cdots, n$ do
10:         if $\hat{\mathbf{x}}_j(i) = 1$ then
11:             $LUT_j^+ \leftarrow \{LUT_j^+, i\}$
12:         else if $\hat{\mathbf{x}}_j(i) = -1$ then
13:             $LUT_j^- \leftarrow \{LUT_j^-, i\}$
14:         end if
15:     end for
16: end for
```

Therefore, we neglect the complexity of finding the $|\mathcal{L}(\mathbf{q})|$ largest values of $\mathbf{v}(\mathbf{q})$. Algorithm 2 describes these steps in more details.

For some application, the (initial) list $\mathcal{L}(\mathbf{q})$ could be further refined by directly searching the retrieved items in the original space with NN search to produce a final list of $\mathcal{L}'(\mathbf{q})$.

V. Experiments

We conduct several experiments[4] to validate the performance of our method and to compare it with the state-of-the-art. We first use synthetic data and then consider three public databases, Inria Holidays+Flickr1M, the UKB and the FAMOS which we describe later.

We compare our method with the sign-random-projection LSH or Sim-hash [10], which is the binary version of LSH and the memory vectors [16] approach which, similar to this work, considers the problem of similarity search in high-dimensional spaces and provides a complexity ratio $\tau_{N,D}$ which is the ratio between the complexity of the algorithm and that of naïve exhaustive search which is $\mathcal{O}(ND)$, in terms of "big-O-notations". Unlike the algorithm run time, this makes the comparison independent of implementation and hardware architecture.

Evaluation protocol: We evaluate the results based on the mean Average Precision (*mAP*) and the *R-Recall@T*.

The area under the precision-recall curve of a query is the Average Precision and the *mAP* is its mean value calculated for

[4]The presented results in this section should be easily reproduced as long as the thresholds λ_X and λ_Y are properly selected in view of the reported complexity/memory values.

Algorithm 2 fast search using STC

```
Input: query vector $\mathbf{q}$, $\mathbf{D}$, $W$, Look-Up-Tables $LUT^{+}$ and $LUT^{-}$
Output: $\mathcal{L}(\mathbf{q})$
 1: Choose $\lambda_Y$, w.r.t. complexity requirements   ▷ One could use a priori knowledge to guess a valid range and a validation set for fine-tuning.
 2: $\tilde{\mathbf{q}} = \mathbf{D}[\mathbf{q}]$
 3: $\mathbf{y} = W^T\tilde{\mathbf{q}}$
 4: $\hat{\mathbf{y}} = \phi_{\lambda_Y}(\mathbf{y})$
 5: Initialize a voting vector $\mathbf{v}(\mathbf{q})_{(N\times 1)}$ with zeros.
 6: Choose voting factor $\nu$ to encourage match and $\nu'$ to penalize mismatch.
 7: List the indices of all $+1$ elements of $\hat{\mathbf{y}}$ in $\mathcal{L}_W^{+}$ and all $-1$ elements in $\mathcal{L}_W^{-}$.
 8: for all $i^+ \in \mathcal{L}_W^{+}$ and $i^- \in \mathcal{L}_W^{-}$ do   ▷ The complexity of this stage is of $\mathcal{O}(\alpha_X\alpha_Y Nn)$.
 9:     $\mathbf{v}(\mathbf{q})_{I^+} \leftarrow \mathbf{v}(\mathbf{q})_{I^+} + \nu$ and $\mathbf{v}(\mathbf{q})_{I^-} \leftarrow \mathbf{v}(\mathbf{q})_{I^-} + \nu$ ($I^+ = LUT^{+}_{i^+}$ and $I^- = LUT^{-}_{i^-}$)
10: end for
11: for all $i' \in \mathcal{L}_W^{-}$ and $i'' \in \mathcal{L}_W^{+}$ do
12:     $\mathbf{v}(\mathbf{q})_{I'} \leftarrow \mathbf{v}(\mathbf{q})_{I'} - \nu'$ and $\mathbf{v}(\mathbf{q})_{I''} \leftarrow \mathbf{v}(\mathbf{q})_{I''} - \nu'$ ($I' = LUT^{+}_{i'}$ and $I'' = LUT^{-}_{i''}$)
13: end for
14: Rank the top $k$ values of $\mathbf{v}(\mathbf{q})$ and report them as $\mathcal{L}(\mathbf{q})$.   ▷ $\mathbf{v}(\mathbf{q})$ is highly sparse, so we neglect the complexity of finding its $k$ largest values.
```

all the queries introduced. When a list of T items are retrieved by an algorithm, the *R-Recall@T* indicates the frequency ratio of the presence of top R correct items as specified by the ground-truth in that list, averaged for all the queries.

A. Synthetic random data

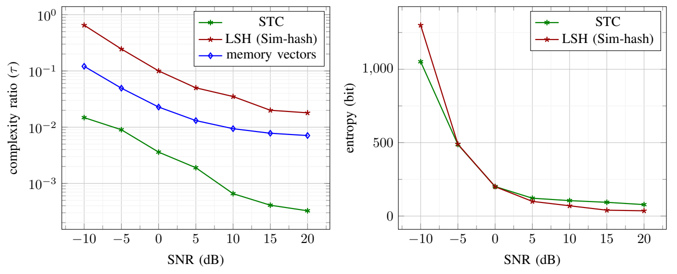
It is a known fact that random data are very difficult to index since they do not have structures in them. This is a reason that in Fig. 5 of [7], the authors report a very low performance in terms of complexity speed-up for 100K of synthetic data with dimensions as low as several tens and they conclude that these datasets represent the greatest challenge for the NN search. Here we evaluate the performance of our algorithm on this type of data.

We consider N feature vectors as random vectors $\mathbf{F}$ generated from $\mathcal{N}(\mathbf{0}, \mathbf{I}_D)$ distribution as the items registered in the database. To model different degradation levels of queries with respect to their respective original item, we consider query random vector $\mathbf{Q} = \mathbf{F} + \mathbf{P}$, generated from $\mathcal{N}(\mathbf{f}(i), \sigma_P^2\mathbf{I}_D)$, where $\mathbf{f}(i)$ is a realization of the random vector $\mathbf{F}$, stored as the i^{th} item in the database with $1 \leqslant i \leqslant N$. Thus, we model the degradation as a white Gaussian[5] noise of variance σ_P^2, where we control the degradation level by SNR $= 10\log_{10}\frac{1}{\sigma_P^2}$, a parameter known as the signal-to-noise ratio.

The choice of this model is justified by the fact that when the NN search is considered as the optimal measure of similarity, the degradation is assumed to be of Additive White Gaussian Noise (AWGN) nature. Otherwise, when there is correlation in the degradation model, the original NN search on the database with minimum Euclidean distance as $\mathbf{f}(i) = \text{argmin}_i \,||\mathbf{q} - \mathbf{f}(i)||_2$ is not the optimal measure.

Fig. 3a and 3b show respectively the achieved complexity ratio and the code entropy[6] of each of the items of the database to achieve near perfect performance in terms of *1-Recall@1* or equivalently, when the desired P_d, the probability of correctly detecting the true 1-NN in a list of size 1 is fixed between $0.99 \leqslant P_d \leqslant 1$.

The experiments were carried out on a synthetic database, consisting of $N = 1$Mio *i.i.d.* Gaussian data items of dimension $D = 2000$, for varying SNR. The parameters of the STC were chosen such that they provide comparable entropy with the Sim-hash. While, depending on the SNR, the STC can require slightly more memory, as is shown in these figures, its computational complexity is near two orders of magnitude less than the Sim-hash LSH.



(a) complexity ratio (b) codeword length

Fig. 3: Search complexity and memory storage requirements per codeword to achieve *1-Recall@1* $\simeq 1$ for varying SNR on synthetic Gaussian *i.i.d.* database ($N = 1000000$, $D = 2000$).

In these experiments, one layer of random projections with varied dimensionality ($150 \leqslant n \leqslant 1500$) was used followed by ternary thresholding the projected data ($0.5 \leqslant \lambda_X, \lambda_Y \leqslant 2.0$) for the STC and binarization for the Sim-Hash.

As is seen from Fig. 3, the required complexity to achieve this performance reduces with increasing SNR. This is a very important advantage of our method indicating its flexibility of design to adjust complexity gain and compactness of the codewords with the level of difficulty of a given database. It is also seen from the figure that the efficiency of the proposed STC, in terms of both memory and complexity, is more evident in the low-SNR regime where the effect of noise on the binary LSH makes it less efficient.

In this setup, we did not use any re-ranking of the items by referring to a number of short-listed items of the database and searching in the original domain. Therefore, we are doing search complexity reduction with compression simultaneously.

For example, for the case of SNR $= 0$ dB which we consider as a typical scenario for many content identification applications, we have a complexity 278 times smaller than the naïve exhaustive search while compressing the whole high dimensional data of original size of 8 GB in single precision

[5]Due to the CLT, anyway the data will be close to Gaussian after projection.

[6]The entropy of a binary code of length n is $H_b = n$, while the entropy of the ternary code of length n with sparsity level α_X is given as $H_t = n[-2\alpha_X log_2(\alpha_X) - (1 - 2\alpha_X)log_2(1 - 2\alpha_X)]$.

to short codes, or their equivalent LUT's as small as $\simeq 25$ MB, plus an overhead of 2.4 MB to store the projection matrix.

Since the data were *i.i.d.*, we did not use any signal processing to take advantage of the redundancy or the structures in the data. Therefore, these results reflect very clearly the indexing capability of the method in reducing the high-dimensional space, independent of the influence of the signal type.

Comparison with memory vectors: Also in Fig. 3a, it is shown the performance of memory vectors[7] [16] on the *i.i.d.* Gaussian data. Instead of SNR, the authors use α[8] and β to represent the degradation of query with respect to the true item. However, we can express them in terms of SNR using $\alpha = \sqrt{\frac{1}{1+10^{-\mathrm{SNR}/10}}}$ and $\beta = \sqrt{1-\alpha^2}$ where it is assumed $\mathbf{Y} = \alpha\mathbf{X} + \beta\mathbf{Z}$ and all random variables set to be of unit ℓ_2-norm (refer to section 2 of [16]).

In calculating the performance of the memory vectors for these experiments and under their hypothesis testing framework, we fix a constraint on the probability of false negative as $\mathbb{P}_{fn} < \epsilon$ (here $\epsilon = 0.01$). This is equivalent to ensuring that *1-Recall@1*, or probability of correct detection be bounded as $1-\epsilon \leqslant P_d \leqslant 1$, from which we can derive the optimal threshold value for the binary decision which gives the probability of false positive, $\mathbb{P}_{fp}$ that dictates the list size in memory vectors [16] (equations 12 and 13 under the "pinv" optimization which gives better performance than "sum-memory"). We then use equation 15 in [16] to calculate the complexity ratio of the memory vectors. Since the optimal memory unit size is not known in advance, for all experiments, we consider all possible values and report the best.

As is shown in Fig. 3, for all considered SNR values, the STC outperforms also the memory vectors in terms of complexity. Moreover, while our approach also compresses the data along with search complexity reduction, the memory vectors, besides introducing a memory overhead, requires all the original database to be in memory since the re-rank of the list of all memory units identified by direct search at the first stage is required.

It should also be mentioned that the reported results for memory vectors are the best case theoretical behaviour with underlying assumptions like $D \to \infty$, while the reported performance of the proposed STC are all based on experiments.

B. Public databases

Here we briefly describe the experimented databases.

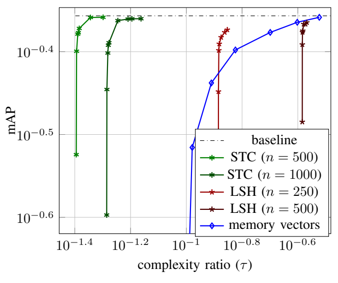
INRIA Holidays [21] is a set of images of various scenes in 500 classes. The first member of each class is excluded from the set and is considered as a query and the rest of 991 remaining images are considered as database items. Since this database is small, **Holidays+Flickr1M** adds one million distractor images to the set, totalling 1000991 items. As for image features we use **Triangulation Embedding** [5], the state-of-the-art local feature aggregation method which provides one global descriptor per image as a high-dimensional ($D = 1920$) feature vector[9]. These features are provided already after the PCA, so the informativeness of the features decreases with growing dimension. For our method and the Sim-hash, in order to provide the initial list, we use only the first 700 features. For the memory vectors, we use the results provided by the authors in [16]. Algorithms for this set are evaluated based on the *mAP* measure.

The **UKB** [22] contains 2550 categories of objects each containing 4 images. The first image of each set is considered as query and, by convention, is not excluded from the database of 10200 items. The evaluation is based on *4-Recall@4*. We use neural codes of $D = 4096$ as the image features. As was also mentioned in the original work of neural codes [1], we observe that these features are highly compressible using the PCA. Therefore, we use the first $d = 500$ PCA coefficients corresponding to biggest eigenvalues as the processing unit $\mathbf{D}$.

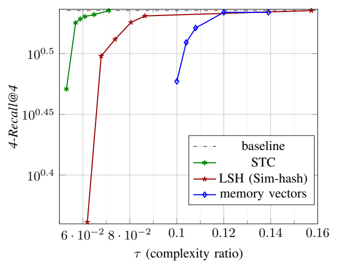
The **FAMOS** [23] contains 5000 unique micro-structures (PUF's) from consumer packages for the development, testing and benchmarking of forensic object identification and authentication technologies. From each package, there are 6 different images taken from two different cameras, each 128 by 128. From the first 500 packages, one image is taken as the query and is not excluded from the set. The evaluation is based on *6-Recall@6*. We use the first 100 elements of the zigzag-ordered 2D-DCT of the gray-scale image as feature vectors. Instead of the original 16384 dimensions, these 100 dimensions are enough to capture all the nearest neighbour content of the data without degrading the useful semantic information.

For the above databases, we consider cosine similarity as the underlying measure. In the case of Holidays+Flickr1M and UKB sets, the ground-truths based on this measure do not exactly correspond to the semantic similarity. For the FAMOS, however, the ground truth corresponds to *6-Recall@6* $= 5.99$.

Fig. 4 compares the performance of the proposed STC with the Sim-hash and the memory vectors for varying complexity ratios on the Holidays+Flickr1M and the UKB sets. A significant complexity advantage is observed compared to the Sim-hash and the memory vectors on these sets.



(a) Holidays+Flickr1M



(b) UKB

Fig. 4: Performance of the image retrieval sets (a) Holidays+Flickr1M with triangulation embedding ($N = 1000991$, $D = 1920$, 500 queries), (b) UKB with neural codes ($N = 10200$, $D = 4096$, 2550 queries).

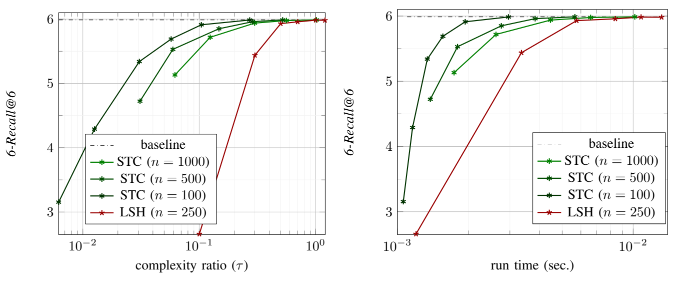
[7]There, the measure of similarity is cosine distance. However, under the *i.i.d.* setup, since the data dimension is high, all vectors will have very similar norms and the maximum cosine similarity translates to minimum Euclidean distance criterion.

[8]Not to be mistaken with α in this paper characterizing the sparsity ratio of the STC.

[9]ftp://ftp.irisa.fr/local/texmex/corpus/memvec/

It is important to note that for these two experiments, like the memory vectors, we re-rank the final list by searching the short-listed candidate items in the original domain. However, the number of *I/O* access to the memory we require is considerably less, accounting for an equivalent complexity ratio of maximum 0.01. It is also interesting to notice that, according to Fig. 4a, the Sim-hash does not achieve the baseline at the experimented regimes and requires either the code-length n or the size of the short-listed items for re-ranking to be increased.

The overall complexity ratio of our method on the Holidays+Flickr1M consists of random projections, table lookup and the direct domain search which can be written as $\tau_{N,D} = \frac{(Dn+\alpha_X \alpha_Y Nn + D|\mathcal{L}|)}{ND}$, where the short-list size for all experiments varies as $0 \leqslant |\mathcal{L}| \leqslant 10000$. On the UKB set, since we also perform the PCA, the complexity ratio is of $\tau_{N,d} = \frac{(Dd+dn+\alpha_X \alpha_Y Nn + D|\mathcal{L}|)}{Nd}$ with $d = 500$ and $0 \leqslant |\mathcal{L}| \leqslant 120$. For the Sim-hash LSH, these complexities should be calculated with $\alpha_X = \alpha_Y = 1$.



(a) complexity ratio (b) approx. run-time

Fig. 5: PUF identification on FAMOS ($N = 30000$, $D = 100$, 500 queries).

Figures 5a and 5b compare the identification performance of STC with that of Sim-hash on the FAMOS, for varying complexity ratio and the corresponding run-time, respectively. For this experiment, varying complexity ratios of STC are reported based on varying sparsity levels for three different values of n. For the Sim-hash, the complexity ratios are changed by changing values of n (from 10 to 120). As it is seen from the figure, the STC achieves the baseline performance with a large gap compared to the Sim-hash. In this experiment, unlike the previous two, we do not re-rank the items by searching the short-listed items from the original domain. Therefore, we do not compare with memory vectors that should always perform the search with the items in the selected memory units from the original data.

VI. Conclusions

The problem of similarity search in high-dimensional feature spaces was addressed by formulating it as sparse approximation in the projection domain. The optimal solution derived motivated the Sparse Ternary Codes (STC), a framework for fast search which, as opposed to the dense codes in the binary hashing framework, considers the reliability of the projection coefficients resulting into a more efficient data structure for search. It was then shown both on the synthetic and several public datasets that the STC compared with the binary LSH and the memory vectors, while being memory efficient, can result into considerable search complexity reduction.

References

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

[2] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, 2004.

[3] H. Jegou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010.

[4] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.

[5] H. Jegou and A. Zisserman, "Triangulation embedding and democratic aggregation for image search," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014.

[6] N. Murray and F. Perronnin, "Generalized max pooling," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014.

[7] M. Muja and D.G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2014.

[8] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998.

[9] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 2004.

[10] M. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002.

[11] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in neural information processing systems*, 2009.

[12] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for Similarity Search: A Survey," *ArXiv e-prints*, 2014.

[13] J. Wang, S. Kumar, and S. F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[14] H. Jégou, M. Douze, and C. Schmid, "Product Quantization for Nearest Neighbor Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[15] M. Norouzi and D. Fleet, "Cartesian k-means," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[16] A. Iscen, T. Furon, V. Gripon, M. Rabbat, and H. Jégou, "Memory vectors for similarity search in high-dimensional spaces," *ArXiv e-prints*, 2014.

[17] M. Shi, T. Furon, and H. Jégou, "A group testing framework for similarity search in high-dimensional spaces," in *Proceedings of the ACM International Conference on Multimedia*, 2014.

[18] A. Andoni, *Nearest neighbor search: the old, the new, and the impossible*, Ph.D. thesis, Massachusetts Institute of Technology, 2009.

[19] Julien Mairal, Francis Bach, and Jean Ponce, "Sparse modeling for image and vision processing," *arXiv preprint arXiv:1411.3230*, 2014.

[20] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1072–1086, 2013.

[21] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *European Conference on Computer Vision*, 2008.

[22] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[23] S. Voloshynovskiy, M. Diephuis, F. Beekhof, O. Koval, and B. Keel, "Towards reproducible results in authentication based on physical non-cloneable functions: The forensic authentication microstructure optical set (famos)," in *Proceedings of IEEE International Workshop on Information Forensics and Security*, 2012.