SNR Maximization Hashing

Honghai Yu, Student Member, IEEE and Pierre Moulin, Fellow, IEEE

Abstract

We propose a novel robust hashing algorithm based on signal-to-noise ratio (SNR) maximization to learn compact binary codes, where the SNR metric is used to select a set of projection directions, and one hash bit is extracted from each projection direction. We first motivate this approach under a Gaussian model for the underlying signals, in which case maximizing SNR is equivalent to minimizing the robust hashing error probability. A globally optimal solution can be obtained by solving a generalized eigenvalue problem. We also develop a multi-bit per projection algorithm to learn longer hash codes when the number of high-SNR projections is limited. The proposed algorithms are tested on both synthetic and real datasets, showing significant performance gains over existing hashing algorithms.

I. INTRODUCTION

Robust hashing, a.k.a. semantic hashing and fingerprinting, has received considerable attention from both academia and industry. For instance, robust hashing is used in several content identification (ID) systems to detect registered audio and video uploads in real time. Robust hashing is also used for music identification on mobile devices. Moreover, robust hashing has been applied to content-based image retrieval (CBIR) tasks in large databases. Other applications include advertisement tracking, broadcast monitoring, copyright control, and law enforcement [1]–[3]. In these applications, the content is encoded into compact binary hash codes (fingerprint) which allows real-time search. The fingerprint must be robust to various content-preserving distortions, while being discriminative enough to distinguish perceptually different signals.

A popular family of hash functions, which assumes centered (mean-subtracted) inputs $\mathbf{x} \in \mathbb{R}^d$, a projection matrix $W \in \mathbb{R}^{d \times k}$, and binary scalar quantization, is given by

$$h(\mathbf{x}, W) = \text{sgn}(W^T \mathbf{x}) \in \{\pm 1\}^k,$$

(1)

H. Yu and P. Moulin are with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign (UIUC), Urbana, IL 61801, USA, and Advanced Digital Sciences Center, Singapore (e-mail: yu75@illinois.edu; moulin@ifp.uiuc.edu)

This work was supported by NSF grant CCF 12-19145 and by AUIP fellowship from the A*STAR Graduate Academy, Singapore.
where \( \text{sgn}(v) = 1 \) if \( v \geq 0 \) and \(-1\) otherwise. For a matrix or vector, \( \text{sgn}(\cdot) \) denotes the element-wise operation. Many robust hashing algorithms fall in this category [4]–[8]. Other families of hash functions based on learning kernels [9], multilayer neural networks [10], [11], and boosting [12], [13] are more expensive to train and evaluate.

Traditionally, \( W \) was generated by randomly sampling a distribution that satisfies the locality-sensitive property [14], [15]. However, data-independent \( W \) can lead to inefficient codes, and thus require much longer codes (larger \( k \)) to work well. Recently, learning \( W \) from training datasets has been shown to outperform data-independent \( W \) for the same code length [5], [7], [8], [11]. To learn \( W \), the nondifferentiable and nonconvex \( \text{sgn} \) function of (1) is often approximated by either the identity function \( h(x; W) = W^T x \) [5]–[7], which obviously introduces a large approximation error when the magnitude of \( W^T x \) is large, or the hyperbolic tangent function \( \tanh(W^T x) \) [8], which can cause the optimization to be trapped in a bad local optimum due to the nonconvexity of the tanh function.

In this paper, we introduce SNR maximization as a candidate for selecting the projection directions. We show that this approach minimizes the robust hashing error probability under a Gaussian model. We propose a SNR maximization hashing (SNR-MH) algorithm that iteratively finds uncorrelated projections that maximize the SNR. Our method does not require any approximation to the \( \text{sgn} \) function and finds the global optimal solution.

SNR has been used as the performance measure in many applications, such as lossy compression [16], matched filtering [17], relay functionality in memoryless relay networks [18], and beamforming in narrowband sensor arrays [19], [20]. Among these applications, matched filtering and beamforming are closely related to our linear projection learning. In both matched filtering and beamforming, the observed signal \( Y = X + Z \in \mathbb{R}^d \) consists of the desired signal \( X \in \mathbb{R}^d \) corrupted by independent additive noise \( Z \in \mathbb{R}^d \).

In matched filtering, the signal \( X \) is deterministic and consists of temporal samples. In beamforming, the signal \( X \) is stochastic and consists of samples from spatially separated sensors. In both cases, the goal is to construct a linear filter \( w^* \in \mathbb{R}^d \) such that the signal-to-noise ratio at the filter output is maximized:

\[
w^* = \arg \max_w \frac{w^T R_X w}{w^T R_Z w},
\]

where \( R_Z \triangleq \mathbb{E}[ZZ^T] \) is the noise auto-correlation matrix and \( R_X \triangleq \mathbb{E}[XX^T] \) is the signal autocorrelation matrix. Note that \( R_X \) reduces to \( R_X = XX^T \) when \( X \) is deterministic. The solution to (2) can be obtained by solving a generalized eigenproblem. To guarantee uniqueness of the solution, the noise power is usually
normalized, e.g., $w^T R_Z w = 1$. For matched filtering, when $Z$ is additive white Gaussian noise (AWGN) with covariance matrix $R_Z = \sigma_Z^2 I$, the solution $w^*$ is a scaled version of $X$, i.e., $w^* = cX$, which not only maximizes SNR, but can be used for optimal detection as well. For beamforming, $w^*$ is the optimal transformation that linearly combines $d$ different copies of the desired signal from $d$ sensors.

Unlike matched filtering and beamforming, robust hashing does not aim to recover $X$. Rather, the decision to be made is whether two signals $X$ and $Y$ are related or not. The decision is not based on $X$ and $Y$ directly but on binary hash codes extracted from $X$ and $Y$. Moreover, robust hashing learns $k$ projection vectors $\{w_1, \ldots, w_k\}$ instead of just one vector as in the matched filtering and beamforming applications.

To our knowledge, SNR has not yet been used as the performance measure for robust hashing in the literature. In the next section, we show that maximizing SNR is equivalent to minimizing the robust hashing error probability in a Gaussian model. In Section III, we derive the SNR-MH algorithm and relate it to other hashing algorithms. Section IV contains simulation results for both synthetic and real datasets, demonstrating SNR-MH’s superior performance in learning compact binary codes. In Section V, we extend SNR-MH to learn longer hash codes. Section VI concludes the paper with possible directions for future research.

II. STATISTICAL MODEL

In this section, we introduce a second-order statistical model for robust hashing and motivate SNR maximization by showing that under an additional Gaussian assumption, a larger SNR results in a smaller robust hashing error probability.

A. Statistical Model for Robust Hashing

The second-order statistical model consists of the following ingredients:

(A1) The signal $X \in \mathbb{R}^d$ follows a distribution $P_X$ with mean $0$ and covariance matrix $C_X \in \mathbb{R}^{d \times d}$.

(A2) If the query item $Y$ is related to $X$, the following distortion model holds:

$$Y = X + Z,$$

where the noise $Z$ is independent of $X$ and follows a distribution $P_Z$ with mean $0$ and positive-definite covariance matrix $C_Z$.

(A3) If $X$ and $Y$ are unrelated, $Y$ is independent of $X$ and follows a distribution $P_Y$.

The hashing code is as follows:
(A4) The projection matrix \( W \in \mathbb{R}^{d \times k}, k \leq d \) is such that the \( k \times k \) matrices \( W^T C_X W \) and \( W^T C_Z W \) are both diagonal\(^1\). Hence, the transformed feature components \( \{ w_i^T X \}_{i=1}^k \) are uncorrelated, and the transformed noise components \( \{ w_i^T Z \}_{i=1}^k \) are also uncorrelated. Denote by \( \{ \sigma_i^2 \}_{i=1}^d \) and \( \{ \lambda_i^2 \}_{i=1}^d \) the diagonal entries of \( W^T C_X W \) and \( W^T C_Z W \) respectively. For the \( i \)-th projection, we have \( w_i^T Y = w_i^T X + w_i^T Z \). We define the \( i \)-th signal-to-noise ratio for the \( i \)-th projection as

\[
\text{SNR}_i \triangleq \frac{\sigma_i^2}{\lambda_i^2}, \quad 1 \leq i \leq k.
\]

(4)

(A5) Binary fingerprints are extracted using the component-wise \( \text{sgn} \) function:

\[
F = \text{sgn}(W^T X) \in \{-1\}^k
\]

\[
G = \text{sgn}(W^T Y) \in \{-1\}^k
\]

(5)

with \( F_i = \text{sgn}(w_i^T X) \) and \( G_i = \text{sgn}(w_i^T Y), 1 \leq i \leq k \).

(A6) Upon seeing a pair \((x, y)\), a binary decision about whether \( x \) and \( y \) are similar or dissimilar is made based on the fingerprints \( f = \text{sgn}(W^T x) \) and \( g = \text{sgn}(W^T y) \). Similar and dissimilar \((x, y)\) pairs are defined as follows:

Similar \((S)\) : \( x \) and \( y \) are related by (3);

Dissimilar \((D)\) : \( x \) and \( y \) are independent.

(A7) The decision rule is

\[
d_H(f, g) \begin{cases} S \quad \text{if} \quad d_H(f, g) \leq \tau \\ D \quad \text{if} \quad d_H(f, g) > \tau \end{cases}
\]

(6)

where \( d_H(f, g) \triangleq \sum_{i=1}^k 1_{\{f_i \neq g_i\}} \) is the Hamming distance between \( f \) and \( g \), and \( \tau \in \{0, 1, \ldots, k\} \) is a decision threshold. The rule declares \((x, y)\) similar when \( d_H(f, G) \leq \tau \) and dissimilar when \( d_H(f, G) > \tau \). A refinement on (6) would be to randomize the decision in the event that \( d_H(f, g) = \tau \). This make it possible to achieve a desired false-positive error probability, as will be discussed in Sections IV-A and V-B.

These assumptions on the robust hashing system are motivated by practical designs such as in [4]–[8], [22]. In particular, the uncorrelatedness property of hash codes in (A4) was first proposed in [22] and used by many subsequent hashing algorithms [5], [23], [24], hash functions in the form of (A5) were used

\(^1\)The existence of such \( W \) is guaranteed [21, Theorem 15.3.2].
in [4]–[8], and the decision rule (A7) is widely used as Hamming distance can be computed extremely fast using bitwise XOR.

B. Error Probability Analysis under Gaussian Model

Based on the above statistical model, we analyze the robust hashing error probabilities under the additional assumption that \(X \sim N(0, C_X)\) and \(Z \sim N(0, C_Z)\). For Gaussian random vectors, uncorrelatedness of \(W^T X\) and \(W^T Z\) implies independence. It then follows from (A4) that \(\{F_i\}_{i=1}^k\) are independent and from (A2) that so are \(\{G_i\}_{i=1}^k\). For non-Gaussian \(X\) and \(Y\), we would only have uncorrelated \(\{F_i\}_{i=1}^k\) and uncorrelated \(\{G_i\}_{i=1}^k\).

Denote by \(P_{FG}(f, g) = \prod_{i=1}^k P_{F_i,G_i}(f_i, g_i)\) the joint distribution of \((F, G)\) when \(X\) and \(Y\) are similar and by \(P_F P_G(f, g) = \prod_{i=1}^k P_{F_i,P_G}(f_i, g_i)\) the distribution when \(X\) and \(Y\) are dissimilar. The performance of the hashing system is quantified using probability of miss

\[
P_M \triangleq P_{FG}\{d_H(F, G) > \tau\}
\]

and probability of false alarm

\[
P_F \triangleq P_F P_G\{d_H(F, G) \leq \tau\}.
\]

In the rest of this section, we prove the following proposition with the help of Lemma 1 and 2 below.

Proposition 1: Under the Gaussian model, for a fixed \(\tau\), \(P_M\) is a decreasing function of \(\{SNR_i\}_{i=1}^k\) and \(P_F\) is independent of \(\{SNR_i\}_{i=1}^k\).

Proof: When \(F = \text{sgn}(W^T X)\) and \(G = \text{sgn}(W^T Y)\) are generated from independent \(X\) and \(Y\), we have

\[
P_{F_i,G_i}\{F_i \neq G_i\} = \frac{1}{2}, \ 1 \leq i \leq k.
\]

As the pairs \((F_i, G_i), 1 \leq i \leq k\) are independent, \(d_H(F, G)\) follows the binomial distribution with \(k\) trials and parameter \(\frac{1}{2}\):

\[
d_H(F, G) \sim \text{Bi}(k, \frac{1}{2}).
\]

Hence, \(P_F\) does not depend on \(\{SNR_i\}_{i=1}^k\).

When \(F\) and \(G\) are generated from similar \(X\) and \(Y\), define

\[
p_i \triangleq P_{F_i,G_i}\{F_i \neq G_i\}, \ 1 \leq i \leq k.
\]

Since the pairs \((F_i, G_i), 1 \leq i \leq k\) are independent, the Hamming distance between \(F\) and \(G\) follows the
Poisson binomial distribution (PBD) with parameter \( \{p_1, \ldots, p_k\} \in [0, 1]^k \):

\[
Pr_{FG}\{d_H(F, G) = l\} = \sum_{A \in E_l} \prod_{i \in A} p_i \prod_{j \notin A} (1 - p_j), \quad 0 \leq l \leq k,
\]

where \( E_l \) is the set of all subsets of \( l \) integers that can be selected from \( \{1, 2, \ldots, k\} \) and \( A^c = \{1, 2, \ldots, k\} \setminus A \) is the complement of \( A \). In the special case of uniform probabilities \( p_i \equiv p \), we have \( d_H(F, G) \sim Bi(k, p) \).

Define the random variable \( T^S_k = d_H(F, G) \) for similar \( F \) and \( G \), so \( T^S_k \sim PBD(\{p_1, \ldots, p_k\}) \). Then we have \( P_M = Pr\{T^S_k > \tau\} \).

**Lemma 1:** For a given decision threshold \( \tau \in \{0, 1, \ldots, k - 1\} \) and probabilities \( \{p_1, p_2, \ldots, p_{k-1}\} \), \( Pr\{T^S_k > \tau\} \) is an increasing function of \( p_k \).

**Proof:** Let \( T^S_{k-1} \sim PBD(\{p_1, \ldots, p_{k-1}\}) \). For \( l = 0, 1, \ldots, k \), we have

\[
Pr\{T^S_k = l\} = p_k \times Pr\{T^S_{k-1} = l - 1\} + (1 - p_k) \times Pr\{T^S_{k-1} = l\}.
\]

Since every PBD is unimodal, and the mode is either unique or shared by two adjacent integers [25], let \( l^* \) be the unique mode (or the smaller of the two modes) of \( T^S_{k-1} \). When \( l \leq l^* \), we have \( Pr\{T^S_{k-1} = l - 1\} < Pr\{T^S_{k-1} = l\} \), so \( Pr\{T^S_k = l\} \) decreases with \( p_k \). When \( l > l^* \) (or \( l > l^* + 1 \) when there are two modes), we have \( Pr\{T^S_{k-1} = l - 1\} > Pr\{T^S_{k-1} = l\} \), so \( Pr\{T^S_k = l\} \) increases with \( p_k \).

Therefore, when \( 0 \leq \tau \leq l^* \), \( Pr\{T^S_k > \tau\} = 1 - \sum_{l=0}^{\tau} Pr\{T^S_k = l\} \) is an increasing function of \( p_k \).

When \( l^* + 1 \leq \tau \leq k - 1 \), \( Pr\{T^S_k > \tau\} = \sum_{l=\tau+1}^{k} Pr\{T^S_k = l\} \) is also an increasing function of \( p_k \).

**Lemma 2:** Under (A2) and (A4), \( p_i \) is a decreasing function of \( \text{SNR}_i \) for \( i = 1, \ldots, k \).

**Proof:** Denote by \( \tilde{X}_i = w_i^T X \) and \( \tilde{Z}_i = w_i^T Z \) the \( i \)-th transformed feature random variable and transformed noise random variable respectively. Then \( \tilde{X}_i \sim N(0, \sigma_i^2) \) and \( \tilde{Z}_i \sim N(0, \lambda^2) \). By (3) and (5), \( F_i = \text{sgn}(\tilde{X}_i) \) and \( G_i = \text{sgn}(\tilde{X}_i + \tilde{Z}_i) \) are independent. It has been shown in [26, Equations 16 and 17] that

\[
p_i = P_{F,G, i}\{F_i \neq G_i\} = \frac{1}{\pi} \arctan \left( \frac{1}{\text{SNR}_i} \right),
\]

which is a decreasing function of \( \text{SNR}_i \). 

It follows from (7), Lemma 1, and Lemma 2 that for fixed \( \tau \), \( P_M \) is a decreasing function of \( \{\text{SNR}_i\}_{i=1}^k \).
III. SNR Maximization Hashing

Motivated by Proposition 1, we propose SNR-MH, a hashing algorithm that finds the globally optimal projection directions \( \{w_i\}_{i=1}^k \) and then extracts binary fingerprints according to (5).

Denote by \( \mathbf{X} \in \mathbb{R}^d \) and \( \mathbf{Z} \in \mathbb{R}^d \) the feature random vector and noise vector respectively (both have mean zero). The problem is to find a \( d \times k \) transformation matrix \( \mathbf{W} = [w_1, \ldots, w_k] \) such that the transformed feature vector \( \mathbf{W}^T \mathbf{X} \in \mathbb{R}^k \) is uncorrelated and the SNR at each projection \( \text{SNR}_i = \frac{\text{var}(w_i^T \mathbf{X})}{\text{var}(w_i^T \mathbf{Z})} \) is maximized. Mathematically, the projection vectors \( w_i, i = 1, 2, \ldots, k \), are learnt sequentially via the following optimization:

\[
\begin{align*}
    w_i = & \arg \max_w \quad \frac{w^T \mathbf{C}_X w}{w^T \mathbf{C}_Z w} \\
    \text{subject to} & \quad w^T \mathbf{C}_X w_j = 0, \quad \forall j < i \\
    & \quad w^T \mathbf{C}_Z w_j = 0, \quad \forall j < i \\
    & \quad w^T \mathbf{C}_Z w = 1,
\end{align*}
\]

where \( \mathbf{C}_X \) and \( \mathbf{C}_Z \) are the covariance matrices of \( \mathbf{X} \) and \( \mathbf{Z} \) respectively, and the last constraint is to normalize the transformed noise to unit power so the solution is unique. To ensure \( \mathbf{C}_Z \) is invertible, a small constant \( \alpha > 0 \) is often added to the diagonal entries of \( \mathbf{C}_Z \), i.e., \( \mathbf{C}_Z \) is replaced with \( \mathbf{C}_Z + \alpha \mathbf{I} \) where \( \mathbf{I} \) denotes the identity matrix.

The optimization (15) is used in multiclass Fisher discriminant analysis (FDA) [27] to learn up to \( k \) linear projections when there are \( k + 1 \) different classes. In multiclass FDA, \( \mathbf{C}_X \) is the inter-class scatter matrix and \( \mathbf{C}_Z \) is the intra-class scatter matrix. The solution of (15) is given by the \( k \) eigenvectors corresponding to the first \( k \) largest eigenvalues of the generalized eigenproblem [27]

\[
\mathbf{C}_X w = \gamma \mathbf{C}_Z w,
\]

where \( \gamma \) is the eigenvalue (to be interpreted as the SNR in the direction \( w \)).

There are several ways to reduce (16) to a standard eigendecomposition problem [21]. One way is to form \( \mathbf{C}_Z^{-1} \mathbf{C}_X \), but in general \( \mathbf{C}_Z^{-1} \mathbf{C}_X \) is not symmetric, so all the nice properties about diagonalizing symmetric matrices will be lost.

Another way to solve (16) is by using the Cholesky decomposition on \( \mathbf{C}_Z \) [21]. Let \( \mathbf{C}_Z = \mathbf{L} \mathbf{L}^T \) where \( \mathbf{L} \) is a lower triangular matrix. Then (16) becomes

\[
\begin{bmatrix}
    \mathbf{L}^{-1} \mathbf{C}_X \mathbf{L}^{-T}
\end{bmatrix} \begin{bmatrix}
    \mathbf{L}^T w
\end{bmatrix} = \gamma \begin{bmatrix}
    \mathbf{L}^T w
\end{bmatrix},
\]
which is a standard eigendecomposition problem.

Note that the above procedure is equivalent to applying a whitening transformation $L^{-1}$ on the noise. After whitening, $L^{-1}Z$ and $L^{-1}X$ have covariance matrices $L^{-1}C_LL^{-T} = I$ and $L^{-1}C_XL^{-T}$ respectively.

**Connection to PCA Hashing (PCAH):** In PCA hashing [5], [7], $W$ is given by the top $k$ eigenvectors of $C_X$. This is equivalent to assuming $C_Z$ is the identity matrix in (16). PCA hashing maximizes the transformed feature variance without considering the noise. The only case PCA hashing is optimal in the sense of SNR maximization is when the noise $Z$ has uncorrelated components with equal variances.

**Connection to Semi-Supervised Hashing (SSH):** SSH [5] was formulated as maximizing a measure of classification accuracy while having large variance and quasi-independence of the hash bits. After approximating the sgn function with the identity function, SSH maximizes the following objective function subject to the constraint $W^TW = I$:

$$
\sum_{i=1}^{k} \left[ w_i^T C_{XY} w_i - w_i^T C_{X\hat{Y}} w_i + \beta w_i^T C_X w_i \right],
$$

where $C_{XY}$ and $C_{X\hat{Y}}$ denote the cross-covariance matrices between similar and dissimilar $X$ and $Y$ respectively, and $\beta > 0$ is a weighting parameter chosen by cross-validation. The optimal projection matrix $W$ then consists of the top eigenvectors of the matrix $C_{XY} - C_{X\hat{Y}} + \beta C_X$.

Under the second-order statistical model of Section II, $C_{XY}$ becomes $C_X$ and $C_{X\hat{Y}}$ is the zero matrix. As a result, the optimal projections of SSH are equivalent to those of PCA Hashing.

In the next section, we will compare the empirical performance of SNR-MH with that of PCAH, SSH, and other hashing algorithms.

**IV. Experimental Results**

**A. Results on Synthetic Data**

We first run simulations on synthetic datasets and compare SNR-MH and PCAH under the Gaussian model of Section II$^2$. We fix the feature dimension $d = 128$. The feature vector $X$ consists of i.i.d. samples from $\mathcal{N}(0, C_X)$. The covariance matrix $C_X = U D_X U^T$, where $U$ is a random $d \times d$ orthogonal matrix and $D_X$ is a $d \times d$ diagonal matrix with diagonal entries uniformly sampled from $(0.5, 1)$ and normalized so that their sum equals to $P = 128$, where $P$ is the total signal power. The noise vector $Z$ consists of i.i.d. samples from $\mathcal{N}(0, C_Z)$ where $C_Z = V D_Z V^T$ where $V$ is a random orthogonal matrix.

$^2$We only show results for PCAH in the synthetic experiments since PCAH and SSH are equivalent under the statistical model.
and $D_Z = \text{diag}\{d_{z1}, d_{z2}, \ldots, d_{zd}\}$. Fixing the total noise power equal to $P$ above, we consider three different scenarios depending on how $\{d_{z1}, d_{z2}, \ldots, d_{zd}\}$ are designed:

1) **Uniform**: $d_{zi} = P/d$, $1 \leq i \leq d$.
2) **Linear**: $d_{zi} = a + (i - 1)r$, $1 \leq i \leq d$, where $a = 0.1$ and $\sum_{i=1}^{d} d_{zi} = P$.
3) **Exponential**: $d_{zi} = ar^{(i-1)}$, $1 \leq i \leq d$, where $r = 1.05$ and $\sum_{i=1}^{d} d_{zi} = P$.

We generate 500,000 similar pairs for training, from which we estimate $C_X$ and $C_Z$ and solve (15) with these estimates. We generate another 500,000 similar and dissimilar pairs for testing. Simulation results are shown in Fig. 1. The left column shows $\text{SNR}_i$ for each projection $w_i$ learnt by SNR maximization and PCA; the right column shows ROC curves for SNR-MH and PCAH at different code lengths. The rows correspond to uniformly, linearly and exponentially generated $\{d_{z1}, d_{z2}, \ldots, d_{zd}\}$ respectively.

Consider the left column first. As noted in Section III, SNR-MH and PCAH coincide in the uniform scenario. In the linear (second row) and exponential (third row) scenarios where noise power is not evenly distributed, $\text{SNR}_1$ increases from 1.32 to 9.98 and 81.83 respectively in SNR maximization but remains largely unchanged in PCA. Additionally, more high-SNR projections are learnt in the exponential case relative to the linear case because the number of small $d_{zi}$'s is larger. On the contrary, PCAH performs similarly across the three scenarios and $\text{SNR}_i$ is generally not a monotone function of the PCA projections as PCAH seeks the variance-maximizing projections of the signal and ignores the noise structure.

To compare the robust hashing performance of the methods considered, we show ROC curves in the right column. We obtain $k+1$ points on the ROC curves, indexed by the integer threshold $\tau \in \{0, 1, \ldots, k\}$. The line segments between consecutive points are obtained by linear interpolation. They can be achieved by randomizing between the two deterministic tests corresponding to the endpoints of each such segment. Showing in the right column, SNR-MH and PCAH performs indistinguishably in the uniform scenario, whereas SNR-MH outperforms PCAH significantly in the linear and exponential scenarios, especially in the latter case where the gain is in orders of magnitude due to the high-SNR projections learnt by SNR maximization.

We notice that in the exponential scenario, 64-bit SNR-MH performs better than 128-bit SNR-MH. Though compact codes are more desirable in many applications, we expect performance to improve with longer codes. This behavior also arises in real datasets where error performance starts to deteriorate when more bits are added to the hashing codes. We will address this issue in more detail in Section V and propose strategies to keep improving performance with longer codes.
Fig. 1: Experiments on the synthetic dataset. The left column shows SNR for each projection $i$ learnt by SNR maximization or PCA; the right column shows ROC curves for SNR-MH and PCAH at different code lengths. Rows correspond to uniformly, linearly and exponentially generated $\{d_{z1}, d_{z2}, \ldots, d_{zd}\}$ respectively.
B. Results on Audio Content Identification

Next, we test our proposed SNR-MH on an audio content identification (ID) system. The problem is to determine whether a given query $y$ is related to some element of the database with $M$ elements, and if so, identify which one. To this end, an algorithm must be designed, returning the decision $\psi(y) \in \{0, 1, 2, \ldots, M\}$, where $\psi(y) = 0$ indicates that $y$ is unrelated to any of the database elements. This is a *single-output decoder*. Alternatively, a *variable-size list decoder* $L(y) \subseteq \{1, 2, \ldots, M\}$ might be used, returning 0, 1, 2 or more matches.

Our audio dataset is a collection of 1,700 songs spanning a variety of music genres including classical, vocal, rock and pop. We randomly divide the 1,700 songs into training, validation, and testing subsets consisting of 100, 100, and 1,500 songs respectively. From the training songs, we generate 22,400 matching and 22,400 nonmatching feature pairs. The audio distortions are created by the software GoldWave [28] and the audio distortions considered are as follows:

1) Bandpass filtering (BPF): 400 Hz to 4 kHz bandpass filtering.
2) Echo (E): Tunnel reverberation.
3) Equalization 1 (EQ1): Boost bass.
4) Equalization 2 (EQ2): Recording industry association of America (RIAA).
5) Sampling rate change (SR): Down-sampling to 16 kHz.
7) WMA encoding (WMA): 64 kb/s WMA encoding.

On top of the above distortions, each audio signal is encoded by 96 kb/s MP3 encoding and added a time delay of 92.9 ms.

We follow the same experimental setup as in [12], [29] for audio fingerprinting. An audio signal is first normalized to mono with 11,025 Hz sampling rate, and then converted into overlapping segments with 371.52 msec window size and 185.76 msec shift. For every segment, an $M$-dimensional spectral subband centroid (SSC) vector is computed [30] from $M = 16$ critical subbands linearly spaced in mel scale from 300 Hz to 5300 Hz. A SSC image, built from $N = 10$ consecutive SSC vectors, is the basic building block for fingerprint extraction. Given a $16 \times 10$ SSC image, we first convert it into a 160 dimensional vector and extract 32 bits from it by SNR-MH as the fingerprint. For every shift of 185.76 msec, an SSC image is obtained from an audio segment of length 2.04 sec. Every audio query is fixed to be 5 sec long, corresponding to 16 SSC images.

To compare performance, we estimate probability of false positive ($P_{FP}$) and probability of false
negative ($P_{FN}$) for the single-output decoder, and expected number of incorrect items on the list ($\mathbb{E}(N_i)$) and probability of miss ($P_{miss}$) for the list decoder [29]. Besides PCAH and SSH, we also compare with two boosting-based hashing algorithms, symmetric pairwise boosting (SPB) [12] and a regularized Adaboost (ACCR Adaboost) [13], [29], which have achieved excellent content ID performance on audio.

Fig. 2 shows the performance comparison on the audio content identification experiments. For both decoders, SNR-MH outperforms all other methods. For the list decoder, SNR-MH outperforms the next best by almost an order of magnitude.

C. Results on Object Retrieval

We also evaluate SNR-MH on the University of Kentucky Object Recognition dataset [31]. There are 2,550 different objects in the dataset, each of which contains four images taken under different viewpoint, orientation, scale, or lighting conditions. See Fig. 3 for some example objects.

Images in the dataset are $640 \times 480$ pixels. Each image is represented by a 512-dimensional bag-of-SIFT-features (BoSF) [32]. In BoSF, dense SIFT descriptors are first extracted from every $16 \times 16$ pixel patches over a grid with spacing of 8 pixels and assigned to 512 visual words learnt by k-means clustering.

We randomly take one image from each object as query and the rest are used as database and training
set. We compute the recall@$K$ for each query, where $K$ is the number of top retrieved samples based on the Hamming distance between the query and database samples, and we report the average recall@$K$ over all queries.

![Fig. 3: Examples from the University of Kentucky Object Recognition dataset.](image)

Fig. 4: Performance comparison of different hashing algorithms on object retrieval. Results in (a) are generated using 64 bits.

Fig 4 shows the performance comparison of SNR-MH and other hashing algorithms including spectral hashing (SH) [22] and iterative quantization (ITQ) [7]. Fig. 4a compares hashing algorithms for 64-bit codes, showing Recall@$K$ vs $K$, while Fig. 4b shows Recall@10 as a function of code size. The figures show that SNR-MH outperforms all other algorithms by a large margin, and Recall@10 increases rapidly as code size increases.
V. LEARNING LONGER HASH CODES

In the previous sections, we have shown that SNR-MH learns compact binary codes that outperform other robust hashing algorithms on both synthetic datasets and real datasets. The ability to learn compact binary codes that preserve semantic similarity is extremely valuable in large-scale retrieval systems as compact binary codes are both search- and storage-efficient. However, in certain applications with a higher bit budget, finding high-SNR projections that are uncorrelated to the previous chosen ones may become challenging.

As observed by several authors [7], [15], the performance of training data dependent hashing algorithms\(^3\) does not always improve with longer code length. One possible reason is that learning a large number of projections overfits the training dataset. For SNR-MH, another reason is the deteriorating effect of low SNR projections. We show this effect both theoretically and empirically, and propose a remedy which we call SNR multi-bit hashing (SNR-MBH). SNR-MBH is comprised of the following three components: (i) a simple iterative procedure that automatically determines the cutoff number of projections beyond which adding bits from the discarded low-SNR projections would hurt performance; (ii) a bit allocation strategy to allocate multiple bits to each high-SNR projection when the number of bits needed exceeds the number of projections; (iii) a multi-bit quantization scheme that assigns multiple bits to each projection. Experiments on a synthetic dataset and real datasets demonstrate the superior performance of SNR-MBH.

A. Deteriorating Effect of Low-SNR Projections

Some of our experiments have shown that when bits are extracted from low-SNR projections, performance deteriorates (see the last subfigure in Fig. 1 for an example). In theory, adding more bits can never hurt performance if the optimal decision rule is used. Under the Gaussian model in Section II, the

\(^3\)Randomized algorithms such as locality sensitive hashing (LSH) [4] and shift-invariant kernels LSH (SKLSH) [15] are data-independent. Though data-independent algorithms enjoy the theoretical guarantee that the underlying metrics are increasingly well preserved as the code length increases, they require much longer codes to work well.
optimal decision rule is a likelihood ratio test (LRT). The loglikelihood ratio

\[ \Lambda = \sum_{i=1}^{k} \log_2 \frac{P_{F,G_i}(f_i, g_j)}{P_{F_i}(f_i)P_{G_i}(g_j)} \]

\[ = 2k + \sum_{i=1}^{k} \left( 1_{\{f_i \neq g_i\}} \log_2 p_i + 1_{\{f_i = g_i\}} \log_2 (1 - p_i) \right) \]

\[ = C + \sum_{i=1}^{k} 1_{\{f_i \neq g_i\}} \log_2 \frac{p_i}{1 - p_i}, \]

where \( C = 2k + \sum_{i=1}^{k} \log_2 (1 - p_i) \) is a constant, is to be compared with a threshold. Hence the LRT can be expressed in terms of the weighted Hamming distance

\[ \sum_{i=1}^{k} \left( \log_2 \frac{1 - p_i}{p_i} \right) 1_{\{f_i \neq g_i\}} \leq \frac{S}{\delta}, \tag{19} \]

where \( \tau \) is the threshold of the test and \( p_i \) is the probability that \( F_i \neq G_i \) for similar fingerprints, as defined in (11). If \( p_i = 1/2 \), the weight \( \log \frac{1}{p_i} \) is zero, rendering the \( i \)-th bit useless. For \( p_i < 1/2 \), a positive weight is assigned, and the smaller the \( p_i \), the larger the weight\(^4\).

On the other hand, the Hamming distance decision rule of (6) gives equal weight to each hash bit, which is a mismatched detector. The deteriorating effect of using bits from low-SNR projections can only be caused by this suboptimal decision rule.

To demonstrate the difference between the LRT of (19) and the suboptimal Hamming distance detector of (6), we run simulations on a synthetic dataset which is generated according to the ‘exponential’ scenario of Section IV-A, with the difference that the total noise power is five times of the total signal power. As shown in Fig. 5, LRT and Hamming distance detector perform similarly up to 64 bits. Then, performance deteriorates from 64 bits to 96 bits and 128 bits for the Hamming distance detector, while performance keeps improving with longer codes for the LRT detector though the improvement is marginal.

However, most robust hashing systems use Hamming distance because of its simplicity and search efficiency [33]. To enjoy these properties without sacrificing too much performance, we propose next a SNR multi-bit hashing (SNR-MBH) for which performance keeps improving with longer code length under the Hamming distance detector. In the following three subsections, we describe the three components of SNR-MBH, and in Section V-E we show SNR-MBH’s superior performance on both the synthetic and real datasets. Note that SNR-MBH could also be used with LRT in a way similar to the model-based

\(^4\)From (14), we know \( p_i < \frac{1}{2} \). In practice, \( p_i \) is estimated from a training dataset. To avoid infinite weights, a small positive constant is added to the estimate.
(a) Hamming distance detector (6) (b) LRT (19)

Fig. 5: ROC curves of SNR-MH on the synthetic dataset.

decision rule in [34], but we leave this for future work.

B. Cutoff Number of Projections

The first objective is to establish a simple procedure that determines whether including the \((k+1)\)-th projection hurts performance. If so, we keep the first \(k\) projections only and discard the rest.

Denoting by \(T^D_k\) and \(T^S_k\) the Hamming distance between independent and related \(k\)-bit fingerprints respectively, then we express (7) and (8) in terms of the test threshold: \(P^k_M(\tau) \triangleq \Pr\{T^S_k > \tau\}\) and \(P^k_F(\tau) \triangleq \Pr\{T^D_k \leq \tau\}\). The goal is to determine the smallest \(k \in \{1, 2, \ldots, d\}\) such that for thresholds \(\tau_k\) and \(\tau_{k+1}\) corresponding to a fixed false alarm probability \(P^k_F(\tau_k) = P^{k+1}_F(\tau_{k+1}) = \alpha\), the probability of miss increases from \(k\) to \(k+1\): \(P^k_M(\tau_k) < P^{k+1}_M(\tau_{k+1})\). This smallest \(k\) will be our cutoff number of projections and denoted by \(K_c\).

Under the Gaussian model of Section II, we have \(T^D_k \sim Bi(k, \frac{1}{2})\) and \(T^S_k \sim PBD(\{p_1, \ldots, p_k\})\). However, working with the binomial distribution and PBD in determining \(K_c\) poses two challenges. First, computing \(P^k_M\) is infeasible for large \(k\) as the number of terms in PBD is combinatorial (12). Second, there generally does not exist \(\tau_k\) and \(\tau_{k+1}\) achieving \(P^k_F(\tau_k) = P^{k+1}_F(\tau_{k+1}) = \alpha\) due to the discrete nature of binomial distributions. To overcome these challenges, we use the fact that both \(T^D_k\) and \(T^S_k\) are sums of independent random variables. For large \(k\), we use Gaussian approximations in the small- and moderate-deviations regime. The means of \(T^D_k\) and \(T^S_k\) are respectively \(k/2\) and \(\sum_{i=1}^{k} p_i\), and
their variances are respectively $k/4$ and $\sum_{i=1}^{k} p_i (1-p_i)$. When the probabilities $P^k_F(\tau)$ and $P^k_M(\tau)$ are in the moderate-deviations regime ($\tau$ is a few standard deviations away from the mean), these two error probabilities can be reasonably well approximated using the Q-function.

Denote by

$$\tilde{P}^k_F(\tau) \doteq Q\left( \frac{k/2 - \tau}{\sqrt{k/4}} \right), \quad (20)$$

$$\tilde{P}^k_M(\tau) \doteq Q\left( \frac{\tau - \sum_{i=1}^{k} p_i}{\sqrt{\sum_{i=1}^{k} p_i (1-p_i)}} \right) \quad (21)$$

the Gaussian approximations of $P^k_F(\tau)$ and $P^k_M(\tau)$ respectively, and arrange $p_i$’s in ascending order. We use the following procedure to determine $K_c$. For each $k = 1, 2, \ldots, d-1$, obtain $\tau_k = k/2 - \sqrt{k/4} Q^{-1}(\alpha)$ and $\tau_{k+1} = (k+1)/2 - \sqrt{(k+1)/4} Q^{-1}(\alpha)$ from $\tilde{P}^k_F(\tau_k) = \tilde{P}^{k+1}_F(\tau_{k+1}) = \alpha$, and check whether the condition $\tilde{P}^{k+1}_M(\tau_{k+1}) < \tilde{P}^k_M(\tau_k)$ fails. The smallest $k$ such that the condition fails will be the cutoff number of projections $K_c$. We keep the first $K_c$ projections and discard the rest.

Throughout our experiments, we use $\alpha = 10^{-3}$ corresponding to a threshold $\tau$ that is about three standard deviations away from the mean $k/2$. Moreover, in calculating $\tilde{P}^k_M(\tau)$, one needs to estimate $\{p_i\}$ from a training dataset. While based on approximations, this cutoff selection procedure worked well in our experiments.

To evaluate this procedure, we run simulations on the synthetic dataset of Section V-A. We fix $P^k_F$ and $\tilde{P}^k_F$ at $\alpha = 10^{-3}$, where $P^k_F = 10^{-3}$ is achieved by the randomized version of the decision rule (6). As shown in Fig. 6, the Gaussian approximations are reasonably close to the simulated results. Moreover, $P^k_M$ and $\tilde{P}^k_M$ achieve their minima at similar values of $k$ which is important because the minimum of $P^k_M$ is our cutoff $K_c$. The cutoff number of projections returned by the procedure is $K_c = 72$, which indicates that using hash codes that are larger than 72 bits could hurt performance. Indeed, we see in Fig. 5a that 96-bit and 128-bit SNR-MH perform worse than 64-bit SNR-MH.

C. A Bit Allocation Strategy

To generate $N > K_c$ bits, one must extract more than one bit from some projections. Hence the first task is to decide how to allocate $N$ bits across $K_c$ projections. This type of problem often arises in information theory. In particular, for a $\mathcal{N}(0, \sigma^2)$ source, the rate distortion function $R(D) = \frac{1}{2} \log \frac{\sigma^2}{D}$ gives the minimum bit rate needed to describe the source with MSE not exceeding $D$ [35]. This motivated
us to develop a bit allocation strategy based on $\log \text{SNR}_i$. Denote by

$$B_i = \left\lceil N \frac{\log (\text{SNR}_i + 1)}{\sum_{i=1}^{K_c} \log (\text{SNR}_i + 1)} \right\rceil$$

the number of bits projection $w_i$ can accommodate, where $\lceil \cdot \rceil$ is the ceiling function and adding one to each $\text{SNR}_i$ ensures each projection is allocated at least one bit. The total number of bits for the $K_c$ projections is $B = \sum_{i=1}^{K_c} B_i \geq N$. If the inequality is strict, we need to prune $B - N$ bits from the total.

To do so, we first assign $N_i = B_i$ bits to each projection, and then prune bits from each projection until $\sum_{i=1}^{K_c} N_i = N$, as described in Table I. The pruning follows two rules: (i) $N_i \geq N_j, \forall i < j$; (ii) whenever $N_j < B_j$, $N_i \leq N_j + 1, \forall i < j$. These two rules ensure that bits are spread as evenly as possible among the $K_c$ projections while satisfying $N_i \leq B_i$.

D. Multi-Bit Quantization

After bit allocation, we need to extract $N_i$ bits from projection $w_i$. A $N_i$-bit scalar quantizer partitions the real line into $2^{N_i}$ bins separated by $2^{N_i} - 1$ thresholds. As $N_i$ could be large, we propose a multi-bit quantization scheme where the number of thresholds grows linearly, rather than exponentially, with $N_i$. 

Fig. 6: Comparison between probability of miss from simulation results on the synthetic dataset and their Gaussian approximations. Both $P^k_F$ and $\tilde{P}^k_F$ are fixed at $10^{-3}$, where $P^k_F = 10^{-3}$ is obtained by a randomized decision rule.
TABLE I: The bit allocation strategy.

| Input: number of high-SNR projections $K$, bit budget $N$, and $B_i$ from (22) for $1 \leq i \leq K_c$. |
| Initialization: $N_i = B_i, \forall i$ and $M = \sum_{i=1}^{K_c} N_i - N$ |
| while $M > 0$ |
| $i = \max\{j : N_j = \max\{N_k, k = 1, 2, \ldots, K_c\}\}$, |
| $N_i = -$, |
| $M = \sum_{i=1}^{K_c} N_i - N$. |
| Output: number of bits assigned to each projection $N_i, 1 \leq i \leq K_c$. |

Motivated by the 2-bit quantization procedure of [12], [29], we define the $i$-th threshold, $i = 1, \ldots, N_i$, as the $i/(N_i + 1) \times 100\%$ quantile of the distribution of the transformed feature $w_i^T X$. The $N_i$ thresholds induce $N_i + 1$ bins, and we assign a length-$N_i$ bit string to the $t$-th bin with $N_i + 1 - t$ zeros followed by $t - 1$ ones for $t = 1, \ldots, N_i + 1$. Besides the linear growth of the number of thresholds with $N_i$, another advantage is that the Hamming distance between the binary code for the $t$-th bin and the binary code for the $(t + s)$-th bin is exactly $s$, which makes the binary code distance preserving. The quantization scheme is illustrated in Fig. 7 with $N_i = 3$.

![Fig. 7: The quantization scheme. The three thresholds are the (25%, 50%, 75%) quantiles of the distribution.](image)

E. Experimental Results

1) Results on Synthetic Data: We show results on the synthetic dataset of Section V-A. We select $K_c = 72$ high-SNR projections as determined by the procedure described in Section V-B. To generate 96 and 128 bits, we use the bit allocation strategy of Section V-C (shown in Fig. 8a is the bit allocation result for the 128-bit SNR-MBH) and the multi-bit quantization scheme of Section V-D. As shown in
Fig. 8b, 96-bit and 128-bit SNR-MBH using 72 projections outperform the corresponding SNR-MH using all 128 projections.

![Diagram](image1)

(a) Bit allocation for 128-bit SNR-MBH.

![Diagram](image2)

(b) ROC curves for SNR-MH and SNR-MBH.

Fig. 8: Experiments on the synthetic dataset generated the same way as Fig. 5. In (b), SNR-MBH uses only the first 72 projections to generate binary codes.

2) Results on MNIST Dataset: Next, we conduct experiments on the MNIST handwritten digit dataset\(^5\) to demonstrate the power of SNR-MH to learn compact codes and SNR-MNH to learn longer codes. The MNIST dataset contains 60,000 training images and 10,000 testing images of ten handwritten digits. Each image is of size \(28 \times 28\) pixels, from which we extract 512-dimensional GIST features [36].

As shown in Fig. 9a, the cutoff number of projections is very small \(K_c = 9\), so we expect bits generated from the tenth projection onward start to hurt performance. Fig. 9b shows the 5 nearest neighbor (NN) classification performance, based on Hamming distance ranking, for different methods at different code lengths. SNR-MH performs impressively well with the first 9 bits. However, as we include more low SNR projections to generate hash bits, the performance of SNR-MH deteriorates.

The bit allocation for 64-bit SNR-MBH with nine high-SNR projections is shown in Fig. 10a, and 5NN performance is shown in Fig. 10b. Now, the classification error keeps decreasing with longer codes and is lower than that for competing methods. SNR-MBH also exhibits superior performance in retrieval tasks, as shown in Fig. 11.

\(^5\)http://yann.lecun.com/exdb/mnist/
3) Results on CIFAR-10 Dataset: The CIFAR-10 dataset [37] consists of 50,000 training and 10,000 test color images of size $32 \times 32$ pixels. Images have been manually grouped into ten classes, namely airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. From each image, we extract the state-of-the-art Convolutional Network-based image features using the feature extractor Overfeat [38],
Fig. 11: Retrieval performance on the MNIST dataset. SNR-MBH uses at most 9 projections.

resulting in a 4,096-dimensional feature vector. Afterwards, we use PCA to reduce the feature dimension to 512, which retains 99.8% of the total signal variance.

Similar to the experiments on the MNIST dataset, $K_c = 9$ high-SNR projections are selected by the procedure in Section V-B. As shown in Fig. 12a, the retrieval performance of SNR-MH drops drastically from 9-bit code to 16-bit code. However in Fig. 12b, we see a strong upward trajectory for SNR-MBH and it outperforms the next best by a large margin across different code lengths.

VI. CONCLUSIONS

In this paper, we have proposed a SNR maximization framework to extract both compact and longer binary hashing codes that preserves semantic similarity. We have shown that the hash bits generated from SNR maximization projections minimize the robust hashing error probability under a Gaussian model for the underlying signals. Despite the simple linear model (3) and the simple training procedure (solving generalized eigenproblems and parameter-free), the proposed SNR-MH and SNR-MBH exhibit excellent retrieval and classification performance on both synthetic and various real datasets. For future work, we plan to develop a kernelized version to learn nonlinear feature transformations and extend the framework to a multi-feature hashing where more than one feature vector is available for both training and testing.
Fig. 12: Experiments on the CIFAR-10 dataset.

REFERENCES


