# Factorized Binary Codes for Large-Scale Nearest Neighbor Search

Frederick Tung
ftung@cs.ubc.ca
James J. Little
little@cs.ubc.ca

Department of Computer Science
University of British Columbia
Vancouver, Canada

Nearest neighbor search is a ubiquitous problem in computer vision. Given a previously unseen query point $\mathbf{q} \in \mathbf{R}^d$, we seek its closest matches in a database $\mathbf{X} \in \mathbf{R}^{n \times d}$. One class of techniques for nearest neighbor search is *hashing algorithms* for constructing compact binary codes. Hashing algorithms transform the original data points into compact bit string signatures that require significantly less storage space and can be compared quickly using bit operations.

We can think of the bits in a binary code as the decisions of a set of hash functions or hyperplanes, possibly in some kernelized space. These hyperplanes are learned or generated by the hashing algorithm. In matrix form, we have

$$\mathbf{Y} = \mathrm{sgn}(\mathbf{XW}) \tag{1}$$

where $\mathbf{X} \in \mathbf{R}^{n \times d}$, $\mathbf{W} \in \mathbf{R}^{d \times c}$, $\mathbf{Y} \in \{0,1\}^{n \times c}$, and $c$ is the number of hash functions, or the number of bits in the generated binary code.

Typically, nearest neighbor search performance improves as the number of hash functions increases, i.e. as $c$ increases. However, as the number of hash functions increases, the matrix $\mathbf{Y}$ of binary codes also increases in size, leading to higher storage requirements. For example, if we wish to improve retrieval performance by doubling the number of hash functions, we have to store binary codes that are twice the length.

In this paper, we present a novel factorized binary codes approach that uses an approximate matrix factorization of the binary codes to increase the number of hash functions while maintaining the original storage requirements. Fig. 1 illustrates the factorized binary codes approach. Given $\mathbf{X}$, $\mathbf{W}$, and $\mathbf{Y}$ as defined in Eq. (1), define a 'long' code length $c^l > c$, and form the matrix $\mathbf{W^l} \in \mathbf{R}^{d \times c^l}$, which appends $(c^l - c)$ new hash functions to the $c$ existing hash functions in $\mathbf{W}$. The new hash functions are generated using the same procedure as the existing hash functions, according to the underlying hashing algorithm. The augmented matrix $\mathbf{W^l}$ produces 'long' binary codes $\mathbf{Y}^l \in \{0,1\}^{n \times c^l}$. We approximate $\mathbf{Y}^l$

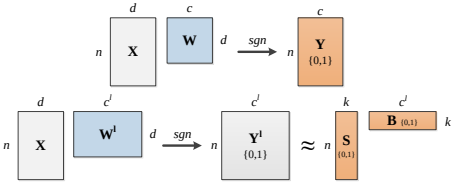


Figure 1: Graphical overview of the factorized binary codes approach

as the Boolean product of two factor matrices $\mathbf{S}$ and $\mathbf{B}$, both of which are also binary:

$$\mathbf{Y}^l \approx \mathbf{S} \circ \mathbf{B} \tag{2}$$

where $\mathbf{S} \in \{0,1\}^{n \times k}$, $\mathbf{B} \in \{0,1\}^{k \times c^l}$, $\circ$ denotes the Boolean product, and $k$ is set such that the factor matrices require no more storage than the original binary codes $\mathbf{Y} \in \{0,1\}^{n \times c}$ (in Fig. 1, the areas highlighted in orange are the same). Given a query $\mathbf{q} \in \mathbf{R}^d$, the 'long' binary code $\mathbf{y_q} \in \{0,1\}^{c^l}$ is computed using the augmented set of hash functions $\mathbf{W}^l$ and matched with the approximate binary codes $\tilde{\mathbf{Y}}^l$ as reconstructed using $\mathbf{S}$ and $\mathbf{B}$. Fig. 2 shows experimental results on the LM+SUN dataset with 384-dimensional Gist descriptors.

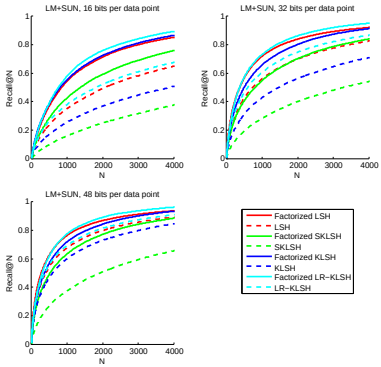


Figure 2: Experimental results on LM+SUN