Nonlinear Asymmetric Multi-Valued Hashing

Cheng Da, Gaofeng Meng, Shiming Xiang *, Kun Ding, Shibiao Xu, Qing Yang, and Chunhong Pan

Abstract—Most existing hashing methods resort to binary codes for large scale similarity search, owing to the high efficiency of computation and storage. However, binary codes lack enough capability in similarity preservation, resulting in less desirable performance. To address this issue, we propose Nonlinear Asymmetric Multi-Valued Hashing (NAMVH) supported by two distinct non-binary embeddings. Specifically, a real-valued embedding is used for representing the newly-coming query by an ideally nonlinear transformation. Besides, a multi-integer-embedding is employed for compressing the whole database, which is modeled by Binary Sparse Representation (BSR) with fixed sparsity. With these two non-binary embeddings, NAMVH preserves more precise similarities between data points and enables access to the incremental extension with database samples evolving dynamically. To perform meaningful asymmetric similarity computation for efficient semantic search, these embeddings are jointly learnt by preserving the pairwise label-based similarity. Technically, this results in a mixed integer programming problem, which is efficiently solved by a well-designed alternative optimization method. Extensive experiments on seven large scale datasets demonstrate that our approach not only outperforms the existing binary hashing methods in search accuracy, but also retains their query and storage efficiency.

Index Terms—Asymmetric hashing, multi-valued embeddings, binary sparse representation, nonlinear transformation.

1 Introduction

Recently, the amount of available data has increased explosively on the web. How to measure the similarity between these tremendous data points embedded in high-dimensional space efficiently is a fundamental but intractable task in real-world applications. Technically, hashing methods [1], [2], [3], [4], [5], [6] devise a series of hash functions to compress high-dimensional data points into compact binary codes, which simultaneously preserve the similarity and (or) structural information of the original data. Owing to the binary representation, hashing not only enables high computational efficiency by using Hamming distance or look-up tables, but also allows extremely efficient memory footprint. Substantially, extensive applications of hashing have been prevalent in various tasks, including content-based retrieval [7], [8], [9], feature matching [10], person re-identification [11], [12], [13], data clustering [14], neural network compression [15], and so on.

In the literature, many hashing methods have been proposed [16], [17], [18], [19], [20], [21], which can be roughly divided into two categories: data-independent and data-dependent hashing methods. Locality Sensitive Hashing (LSH) [22] is a representative of the first class that employs random projection as hash function. However, the major drawback of LSH and its variants [4], [23] is that many hash bits are required to guarantee good performance. In parallel, recent efforts mainly focus on designing data-dependent hashing methods that learn compact binary codes by fully exploiting the data information. These methods are categorized into unsupervised [3], [24], [25], [26], semi-supervised [8], [27] and supervised methods [28], [29], [30], [31]. Contrary to unsupervised hashing methods that learn hash functions by only preserving the structural information of the original unlabeled data (*e.g.* , the Euclidean distance), supervised hashing attempts to learn semantic-preserving binary codes by leveraging the semantics (*e.g.* , classification information [7], [32], pairwise similarity [28], [30] and triplet ranking [33], [34]) induced from label information. Hence, supervised hashing yields promising performance in terms of search accuracy, attracting extensive attention.

From the perspective of encoding strategy, most of supervised hashing methods [3], [30] treat the query points and database ones in a symmetric manner. Namely, the both binary codes are derived from the same hash function in symmetric hashing. On the contrary, some pioneering work [35], [36] has theoretically proven that asymmetric hashing can attain superior accuracy with shorter codes by using two different hash functions for query and database points. For example, Gordo *et al.* [37] claimed that compressing the query into binary codes is not a strict requirement. The key insight is that query is real-valued and database is still binary, so that the more precise information of the query is capable of facilitating better similarity search. Recently, supervised discrete hashing (SDH) [32] and column sampling discrete supervised hashing (COSDISH) [28] adopt an asymmetric learning strategy, where the binary codes of database points are directly learned by discrete optimization methods, and the learned hash function is only applied for the unseen query points. As the accumulation errors of encoding database points are reduced, the considerable performance is achieved in these asymmetric hashing methods.

Such work enriches the research on hashing techniques and encouraging results have been witnessed on the advent of large scale data, but they all resort to binary codes to some extent. However, even though binary codes are efficient for retrieval and storage, they apparently can not preserve the semantic similarity powerfully, and thus leading to degraded search accuracy. In view of the intrinsical binary characteristic of hashing, a paradoxical question worthy of

- *C. Da, G. Meng, S. Xiang, K. Ding, S. Xu, Q. Yang and C. Pan are with the Department of National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. E-mail: {cheng.da, gfmeng, smxiang, kding, shibiao.xu, qyang, chpan}@nlpr.ia.ac.cn*
- *C. Da and S. Xiang are also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 101408, China.*
- ** Corresponding author: Shiming Xiang. E-mail: smxiang@nlpr.ia.ac.cn*

study is raised: *Beyond binary codes, can we employ non-binary codes to execute more precisely search and retain the query and storage efficiency derived from binarization simultaneously?*

In this paper, we answer the aforementioned question by a novel asymmetric hashing method, named Nonlinear Asymmetric Multi-Valued Hashing (NAMVH), which achieves the goal of alleviating the binary limitation. The idea of our approach is quite intuitive — exploiting real values and multiple integer values instead of binary ones should permit the better preservation of the similarity between data points. To instantiate this proposal, we investigate two multi-valued embeddings: the real-valued embedding and the multi-integer-valued embedding. The former is employed to map the query point into a real-valued low-dimensional space. The latter is utilized to compress the database point in a multi-integer-valued space. Thanks to the multi-valued encoding scheme, the more precise information of query points is preserved and the space partition of database is denser than that of basic binary space. Consequently, NAMVH is able to allow the larger number of possible distances and the more accurate distance approximation, in terms of the similarity search.

According to the distinct purpose of these two embeddings, we implement them in an asymmetric fashion. To be specific, an ideal transformation function parameterized by Θ is learned to directly produce real-valued embeddings of the newly-coming query points. Rather than searching Θ in the linear or kernel space, we instead optimize Θ in a multi-layer neural network, which can be trained across the whole supervised information efficiently, as opposed to the limited corpus of incomplete pair constructions or sampled subset in [18], [29]. Notably, no mandatory binary constraints are imposed to guarantee the binary-like property of network output as in [38], [39]. In doing so, the nonlinear relationship between data points can be well explored and exploited.

On the other hand, the multi-integer-valued embeddings of database points are directly optimized in training step, which can eliminate the quantization error accumulation. Nevertheless, general multi-integer-valued embeddings do not come with provable query time guarantees, as they are not conducive for building lookup table with tremendous combinations of integer values, which is commonly used for efficient search. Binary Sparse Representation (BSR) is proposed to circumvent this problem, where a multi-integer-valued vector is represented by a product of a binary dictionary and an indicator vector. In this way, the lookup table about dictionary can be built efficiently, and hence the efficiency of similarity search and storage can be guaranteed. Moreover, the sophisticated product form is amenable to the incremental extension of database efficiently. Namely, NAMVH is able to generate and update embeddings of database with data evolving dynamically, while some current hashing methods [28], [29], [32] necessitate retraining from scratch to accommodate the newly-coming samples.

To make the two different embeddings to be useful representations for asymmetric similarity computation in the query stage, these embeddings should be optimized to allow the meaningful comparison between them. For this reason, our approach is formulated as an asymmetric inner product fitting problem, of which the aim is to minimize the alignment error between the predictive similarity and the ground-truth one provided by the semantic labels in a pairwise manner. However, the discrete constraints bring us a mixed integer programming problem that is generally NP-hard [28]. To solve the problem highly coupled with neural network and binary constraints, a well-designed alternative optimization algorithm is exploited, where each subproblem is solved efficiently, yielding satisfactory solutions.

The main contributions of this work are as follows:

- A novel asymmetric supervised hashing method supported by multi-valued embeddings is proposed, which can alleviate the intrinsic binary limitation and remarkably improve the capability of similarity preservation, resulting in superior search accuracy.
- We introduce Binary Sparse Representation (BSR) to model the multi-integer-valued embeddings for database points. BSR not only guarantees the efficiency of similarity search and storage, but also enables access to the incremental extension of database. To the best of our knowledge, the multi-integer-valued embedding is first proposed in hashing methods.
- Rather than attempting to optimize real-valued embeddings in the kernel space [40], we instead integrate them with multi-integer-valued embeddings as the loss layer of a multi-layer neural network without any binary constraints. Having done so, the inherently nonlinear relationship of the data samples may be better explored and exploited. Once optimized, the unseen query points can be transformed into more effective real-valued embeddings than that in [40].
- A well-designed alternative optimization algorithm is proposed to solve the highly coupled problem efficiently. Even though the pairwise objective function is implemented, NAMVH is able to perform neural network learning simply in a pointwise manner without pair constructions or even subset sampling. Along with the efficient optimization of other subproblems, NAMVH has linear time complexity, which is fairly scalable to large scale data learning.

It should be noted that this article extends our early conference work [40] through the following aspects. (1) Beyond performing optimization of real-valued embeddings in the kernel space [40], we instead train a neural network across the whole supervised information to optimize them efficiently, by recasting the problem into a simple regression framework without mandatory binary constraints. In doing so, the intrinsically nonlinear similarities are effectively preserved, such that NAMVH reaches more appealing performance than the kernel-based one [40]. (2) We further demonstrate that the similarity metric of NAMVH can be regarded as the query sensitive weighted Hamming distance. (3) Incremental extension of database is proposed to accommodate the dynamically evolving database. (4) Extensive experiments and analysis on more larger datasets are conducted to verify the effectiveness.

2 Background and Related Work

2.1 Asymmetric Hashing

Most classical hashing methods [3], [22], [25] handle query and database points in a symmetric way. While some efforts have been devoted to delving into the merits of the asymmetric form in similarity search problem over the past years. Some methods incipiently attempt to employ "asymmetric distance" for the more precise distance measure. Dong *et al.* [35] proposed to directly estimate the distance between the original feature vector of the query and the binary codes in the database, where this asymmetric distance can be regarded as a weighted Hamming distance. Since [35] is on the basis of random projections, Gordo *et al.* [37] presented two generic asymmetric distances that can be broadly applied to many hashing methods. Accordingly, these methods mainly focus on the distance measure between the codes derived from the same real-valued embedding function that is previously defined. In our work, we focus on the functional space that can yield powerful embedding functions to radically alleviate the binary limitation.

Besides the early research about the asymmetric distance measure, many hashing methods have concentrated on designing or learning two different hash functions for database and query points, respectively. Asymmetric LSH [41] extends the traditional LSH framework to allow two pre-designed hash functions for solving Maximum Inner Product Search (MIPS) problem. Neyshabur *et al.* [36] went a step further by learning two distinct binary hash functions directly, proposing an intrinsically asymmetric hashing method. Gearing to this line, Shen *et al.* [19] presented a binary codes learning framework for the MIPS problem by directly learning different hash functions. Recently, combined with feature learning, Shen *et al.* [42] devised two different deep convolutional models as hash functions. Jiang and Li [43] simultaneously learned the binary codes of database points and the deep hash functions only for query points in an efficient training procedure.

Notably, two-step hashing methods [44], [45] have attracted broad research interests, which explicitly decompose the hashing learning problem into two steps: a binary code inference step and a hash function learning step based on the learned codes. Subsequently, many hashing methods generalize the idea of two-step hashing by introducing some coupling between the two steps mentioned above. For example, Lin *et al.* [29] proposed FastHash that solves the two steps alternatively in a bit-wise optimization strategy, where the binary classifier of decision tree is learned for one bit at a time. Kang *et al.* [28] presented an efficient discrete optimization by sample strategy in the first step, and applied decision tree as classifiers in the second step. Shen *et al.* [32] jointly learned hash codes that are ideal for classification and the hash functions in a discrete optimization manner. As mentioned in [46], asymmetric encoding strategies can be adopted in two-step hashing, and thereby it is essentially an asymmetric hashing method. Nevertheless, aforementioned methods lack enough ability in similarity preservation, due to the binary limitation to some extent, and thus achieving less desirable search performance.

2.2 Nonlinear Hashing

Conventional hashing methods often formulate hash function as a linear transformation to encode data points [3]. To further exploit the underlying structure of data points, nonlinear hash functions have been extensively used recently. For example, some hashing methods learn hash function in kernel space, such as kernel-based supervised hashing (KSH) [30] and SDH [32]. Rather than adopting shallow models, Do *et al.* [38] utilized multi-layer neural network to produce binary codes. Recently, remarkable advances of deep learning [47] have been demonstrated in image classification [48], image segmentation [49] and object detection [50]. Many deep supervised hashing methods driven by different semantic similarity preserving manners have been proposed, which integrate feature learning with nonlinear hash function learning in a unified framework. Yang *et al.* [7] transformed a latent layer of a classification CNN as binary-like hash functions and performed the joint learning of image representations, hash codes and classification in a pointwise manner. Xia *et al.* [45] trained a DNN model to fit the hash codes pre-decomposed from the pairwise similarity matrix in a two-step fashion. Lai *et al.* [34] designed a triplet ranking loss to preserve the relative similarities.

In the view of scalability, pointwise methods allow large scale datasets training. While the computational and storage cost of pairwise methods scale in $\mathcal{O}(N^2)$, where N is the size of database, and thus is difficult to exploit the whole information of database points efficiently. Clearly, the training cost of triplet-based methods is more higher. However, the network in NAMVH can be learned in a pointwise fashion, even if the pairwise information is used.

In addition, the inherent binary constraints still enforce the deep hashing methods to generate binary codes. Most of them [33], [45] solve this discrete problem with continuous relaxation. While the modified problem with relaxation deviates from the original one due to the large quantization error, resulting in suboptimal solutions. For this purpose, recent studies have paid attention to handling the binary constraints well. Liu *et al.* [51] imposed a regularizer to replace the binary constraints by encouraging the outputs to approximate discrete values (*i.e.* , ± 1) directly. Zhu *et al.* [52] utilized bimodal Laplacian prior to enforce that the learned codes are assigned to discrete values with the largest probability. HashNet [53] trains the network with an alterable activation function, where the degree of smooth is gradually reduced as the training continues, hence eventually converging to the original binary optimization problem.

Some work [18], [38], [42], [54] also adopts smooth activation function (*e.g.* , Tanh) to approximate non-smooth sign function. And the quantization loss between binary codes and continuous outputs of network is then introduced to guarantee the binary property indirectly. However, due to the multi-valued embeddings of NAMVH, the network can be readily trained without binary limitations for directly encoding queries as real-valued ones, and multi-integer-valued embeddings are simultaneously learned in training stage. In doing so, the accumulated quantization error can be almost discarded in both training and query stage.

3 Nonlinear Asymmetric Multi-Valued Hashing

3.1 Basic Formulation

Suppose that database points are represented by a set of N D-dimensional vectors $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$. Their associated labels are denoted by $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \{0,1\}^{C \times N}$, where C is the number of classes. When the c-th entry y_{ci} equals to 1, it means $\mathbf{x}_i$ belongs to the c-th class. The goal of traditional supervised hashing is to learn K-bit binary embeddings $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_N] \in \{-1,1\}^{K \times N}$ such that the intrinsic correlation of the original space can be well preserved in the Hamming space. A commonly used objective to learn these codes is to preserve the Hamming affinity [28], [29], [30]. Concretely, it minimizes the discrepancy between the predictive and ground-truth affinities, formulated as $\min_{\mathbf{B}} \|\frac{1}{K}\mathbf{B}^T\mathbf{B} - \mathbf{S}\|_F^2$, where $\mathbf{S}$ is a pairwise similarity matrix derived by the labels and $\|\cdot\|_F$ is the Frobenius norm. This model is first introduced in KSH [30], and becomes a generic optimization problem for hashing learning [28], [29]. However, one major deficiency of this formulation is the symmetric binary inner product form, which can only preserve $K+1$ kinds of Hamming distance, limiting in approximating the real-valued similarity accurately. In brief, binary codes lack enough capability in similarity preservation, resulting in less desirable performance.

We therefore propose NAMVH to alleviate the binary limitation, in which the traditional binary values are extended to real values or multiple integer values. In the following subsections, we firstly introduce the construction of the real-valued embedding and the multi-integer-valued embedding. Then, the architecture and optimization method are elaborated. We also present the incremental extension of database samples. Finally, complexity analyses are reported.

3.2 Real-valued Embedding for Query

Since directly using the real-valued information should permit more accurate approximation of similarity, we aim to explicitly exploit the real-valued embeddings in our approach, constructing the asymmetric objective function as follows:

$$\begin{aligned} \min_{\Theta,\mathbf{B}} \quad & \|F(\mathbf{X};\Theta)^T\mathbf{B} - \mathbf{S}\|_F^2 \\ \text{s.t.} \quad & \mathbf{B} \in \{-1,1\}^{K \times N}, \end{aligned} \tag{1}$$

where $F(\cdot)$ serves as an ideal transformation parameterized by Θ to generate real-valued embeddings $F(\mathbf{X};\Theta) \in \mathbb{R}^{K \times N}$ without any binary constraints, and $\mathbf{B}$ is binary ones. The similarity between them is measured in an asymmetric inner product form, which is of more diversity than those are both binary. Meanwhile, the asymmetric objective in problem (1) may have potential to achieve a smaller objective value than symmetric one, which usually suggests more similarity preservation and better retrieval accuracy [55]. The two kinds of embeddings are jointly learned by preserving the pairwise semantic similarities. Once problem (1) is solved, the binary embeddings of database points can be rendered by $\mathbf{B}$ and the unseen query $\mathbf{q}$ can be encoded via $F(\mathbf{q};\Theta)$.

As the produced embeddings should be capable of preserving the inherent structure of data points, the power of $F(\cdot)$ affects the performance of our approach to large extent. RBF kernel mapping is employed to construct a simple transformation in our previous version [40]. But, kernel-based projection is susceptible to the selected anchor points, resulting in limited and unstable performance. Furthermore, together with the subsequently imposed term or constraints, the combined objective function is non-convex. Rather than searching Θ in the kernel space, we instead optimize Θ in a multi-layer neural network, which can be trained across the whole supervised information efficiently. One thus realizes the desideration to generate more desirable real-valued embeddings and simultaneously captures the intrinsically nonlinear relationship of the data samples.

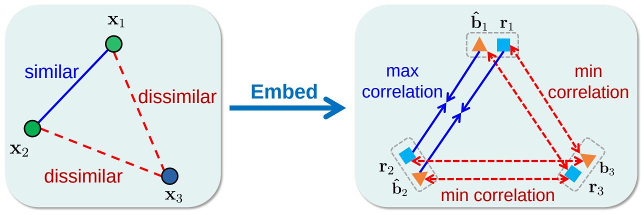


Fig. 1. Illustration of asymmetric inner product fitting. The similarity between the two distinct embeddings is crosswise calculated. In order to preserve the semantic similarity, the similarity of similar pairs is maximized, otherwise, the similarity of dissimilar pairs is minimized.

3.3 Multi-integer-valued Embedding for Database

Although problem (1) is modeled in the asymmetric viewpoint, the database points are still supported by binary codes, which have limited capability in approximating diverse similarities. To alleviate the binary limitation thoroughly, we introduce the multi-integer-valued embedding into problem (1), whereby database points are compressed into multiple integer vectors. Consequently, the capability of similarity preservation is further improved.

However, multiple integer values are not conducive for fast search, because lookup table can not be build efficiently for massive combinations of huge integer values. Therefore, we propose Binary Sparse Representation (BSR) to model the multi-integer-valued embeddings with fixed sparsity. Technically, a multi-integer-valued vector $\hat{\mathbf{b}}_i$ is denoted by a product of a binary dictionary $\mathbf{C}$ and an indicator vector $\mathbf{a}_i$, *i.e.*, $\hat{\mathbf{b}}_i = \mathbf{C}\mathbf{a}_i$. Here $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_M] \in \{-1,1\}^{K \times M}$ is the binary dictionary with $M\,(M \ll N)$ atoms and $\mathbf{a}_i \in \{0,1\}^M$ is the sparse indicator vector, which strictly contains l 1s to indicate that only l atoms can be selected from the dictionary. $l\,(0 < l < M)$ is a crucial hyper-parameter, reflecting the sparsity of $\mathbf{a}$ and the diversity of multi-integer-valued embeddings indirectly.

Based on BSR, $\hat{\mathbf{B}} = [\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_N] \in \{-l, -l+2, \ldots, l-2, l\}^{K \times N}$ includes at most $l+1$ kinds of integer values. Meanwhile, we can build the lookup table about the dictionary $\mathbf{C}$ and apply $\mathbf{a}_i$ for table lookup operation, whereby the efficiency of similarity search and storage is guaranteed to some degree by BSR. By replacing $\mathbf{B}$ in problem (1) with $\hat{\mathbf{B}} = \mathbf{C}\mathbf{A}$, the problem is reformulated as follows:

$$\begin{aligned} \min_{\Theta,\mathbf{C},\mathbf{A}} \quad & \|F(\mathbf{X};\Theta)^T\mathbf{C}\mathbf{A} - \mathbf{S}\|_F^2 \\ \text{s.t.} \quad & \mathbf{1}_M^T\mathbf{A} = l\,\mathbf{1}_N^T, \\ & \mathbf{A} \in \{0,1\}^{M \times N},\ \mathbf{C} \in \{-1,1\}^{K \times M}, \end{aligned} \tag{2}$$

where these two embeddings can also be jointly learnt and $\mathbf{1}_M$ represents a column vector with M 1s. Technically, problem (2) turns out to be an asymmetric inner product fitting problem, where the semantic similarity between asymmetric embeddings should be preserved, as shown in Fig. 1. Note that when $\mathbf{C}$ is a complete dictionary and $l = 1$, problem (2) degrades to (1). Theoretically, problem (1) can be regarded as a special case of problem (2).

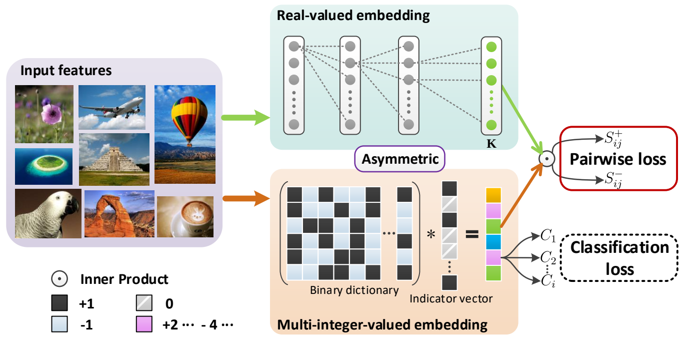


Fig. 2. The overview architecture of NAMVH. Each input is represented as two asymmetric embeddings. One is the real-valued embedding (the green points), generated by a multi-layer neural network. Another is the multi-integer-valued embedding (the colorful squares), modeled by BSR. We minimize the pairwise loss between these embeddings and the classification errors in one unified framework. Best viewed in colors.

Notably, the compositional form of $\mathbf{CA}$ is analogous to Cartesian k-means [56]. However, there exist intrinsically distinguishable differences. Cartesian k-means employs $\mathbf{CA}$ to construct more real-valued cluster centers so as to reconstruct the original features with smaller quantization errors and storage, and thus it only pays attention to the number of cluster centers. By contrast, NAMVH intends to alleviate the binary limitation so that it utilizes this form to construct integer values based on binary atoms in a concise format, which lays emphasis on the value of the compact codes.

3.4 Architecture

Problem (2) mainly focuses on the pairwise relations. In order to take full advantage of supervisory information, pointwise information of the class labels can be formulated as a classification term for guiding the hash function learning [32]. We therefore consider to additionally introduce a classification error term into the problem (2) as a regularization term, obtaining the overall objective function:

$$\begin{aligned}\min_{\Theta,\mathbf{C},\mathbf{A},\mathbf{V}} \quad & \|F(\mathbf{X};\Theta)^T\mathbf{CA}-\mathbf{S}\|_F^2+\lambda\|\mathbf{V}^T\mathbf{CA}-\mathbf{Y}\|_F^2 \\ \text{s.t.} \quad & \mathbf{1}_M^T\mathbf{A}=l\,\mathbf{1}_N^T, \\ & \mathbf{A}\in\{0,1\}^{M\times N},\ \mathbf{C}\in\{-1,1\}^{K\times M},\end{aligned} \tag{3}$$

where $\mathbf{V}\in\mathbb{R}^{K\times C}$ is the weight of a linear classifier, which guarantees that the good multi-integer-valued embeddings are beneficial for classification, and λ is penalty parameter that determines the strength of the classification term. As for similarity matrix $\mathbf{S}$, we generate it by $\mathbf{Y}^T\mathbf{Y}$. Compared to the binary similarity matrix used in [29], [30], $\mathbf{Y}^T\mathbf{Y}$ may make better use of the supervised information implied in the semantic labels and facilitate the training of large scale dataset efficiently. However, due to the label imbalance problem [46], $\mathbf{Y}^T\mathbf{Y}$ is improper for direct use. For this reason, we construct a new semantic similarity matrix: $\hat{\mathbf{S}}=(1+\alpha)\mathbf{Y}^T\mathbf{Y}-\alpha$. Then, $\hat{\mathbf{S}}$ is decomposed into a product of two smaller matrices: $\hat{\mathbf{S}}=\mathbf{P}^T\mathbf{Q}$, where $\mathbf{P}=[\sqrt{1+\alpha}\mathbf{Y};\sqrt{\alpha}\mathbf{1}^T]$, $\mathbf{Q}=[\sqrt{1+\alpha}\mathbf{Y};-\sqrt{\alpha}\mathbf{1}^T]\in\mathbb{R}^{(C+1)\times N}$. Thus, we employ the factorization form instead of the original $\hat{\mathbf{S}}$ for computation, reducing the complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(NC)$. Once problem (3) is optimized, the database points can be represented by $\mathbf{CA}$, and the network will be employed for encoding the newly-coming queries. An overview architecture is illustrated in Fig. 2.

3.5 Query Sensitive Weighted Hamming Distance

The similarity between two asymmetric embeddings is defined as a inner product form in (2). For the case of using K-bit multi-integer-valued embedding $\mathbf{Ca}$ with l sparsity and K-bit real-valued embedding $\tilde{\mathbf{x}}$ generated by $F(\mathbf{x};\Theta)$, the similarity between $\mathbf{Ca}$ and $\tilde{\mathbf{x}}$ is calculated by $\tilde{\mathbf{x}}^T\mathbf{Ca}$. We denote the i-th selected atom from binary dictionary $\mathbf{C}$ by $\mathbf{p}_i$ according to $\mathbf{a}$, and the absolute value of $\tilde{\mathbf{x}}$ is denoted by $\hat{\mathbf{x}}$. Therefore, the similarity can be rewritten as follows:

$$\begin{aligned}\tilde{\mathbf{x}}^T\mathbf{Ca} &= \tilde{\mathbf{x}}^T\sum_{i=1}^{l}\mathbf{p}_i=[\tilde{\mathbf{x}}^T,\ldots,\tilde{\mathbf{x}}^T]\,[\mathbf{p}_1^T,\ldots,\mathbf{p}_l^T]^T \\ &= [\text{sign}\,(\tilde{\mathbf{x}})^T,\ldots,\text{sign}\,(\tilde{\mathbf{x}})^T]\,\Gamma\,[\mathbf{p}_1^T,\ldots,\mathbf{p}_l^T]^T \\ &= \sum_{i=1}^{l}\sum_{j=1}^{K}\hat{x}_j(-2\|e_{(i-1)l+j}-\tilde{p}_{(i-1)l+j}\|_H+1),\end{aligned} \tag{4}$$

where the concatenation of $\mathbf{p}_i$ is denoted by $\tilde{\mathbf{p}}=[\mathbf{p}_1^T,\ldots,\mathbf{p}_l^T]^T\in\{-1,1\}^{lK}$, l-times repetitive concatenation of $\text{sign}\,(\tilde{\mathbf{x}})^T$ is denoted by $\mathbf{e}=[\text{sign}\,(\tilde{\mathbf{x}})^T,\ldots,\text{sign}\,(\tilde{\mathbf{x}})^T]^T\in\{-1,1\}^{lK}$, and $\|\cdot\|_H$ represents the Hamming distance. More specifically, $\Gamma=\text{diag}\,(\Lambda,\ldots,\Lambda)$ is a diagonal matrix, where Λ denotes the diagonalizable form of $\hat{\mathbf{x}}$, *i.e.*, $\Lambda=\text{diag}\,(\hat{x}_1,\ldots,\hat{x}_k)=\text{diag}\,(|\tilde{x}_1|,\ldots,|\tilde{x}_k|)$.

Eqn. (4) shows that the asymmetric inner product can be reformulated as the form of weighted Hamming distance of two concatenate embeddings. Moreover, the weights $\hat{x}_j$ for each bit is only determined on the absolute value of the query, which is query sensitive [57], [58]. Consequently, the similarity computed by $\tilde{\mathbf{x}}^T\mathbf{Ca}$ is equivalent to a query sensitive weighted Hamming distance of concatenate embeddings, which is of more capability of similarity measure than that of the conventional Hamming distance. To circumvent the task of time-consuming computation of between lK-bit codes, an efficient query strategy is proposed in Section 3.8.

3.6 Optimization

The optimization of problem (3) consists of neural network learning and discrete optimization problem. Meanwhile, $F(\mathbf{X};\Theta)$ is highly coupled with $\mathbf{CA}$ and $\mathbf{S}$ in a pairwise form, so that it is cumbersome to update Θ by Back Propagation (BP) algorithm in a minibath manner. To this end, we introduce an auxiliary variable $\mathbf{Z}\in\mathbb{R}^{K\times N}$ to isolate network dexterously, obtaining the following formulation:

$$\begin{aligned}\min_{\Theta,\mathbf{Z},\mathbf{C},\mathbf{A},\mathbf{V}} \quad & \|\mathbf{Z}^T\mathbf{CA}-\mathbf{S}\|_F^2+\lambda\|\mathbf{V}^T\mathbf{CA}-\mathbf{Y}\|_F^2 \\ \text{s.t.} \quad & \mathbf{Z}=F(\mathbf{X};\Theta),\ \mathbf{1}_M^T\mathbf{A}=l\,\mathbf{1}_N^T, \\ & \mathbf{A}\in\{0,1\}^{M\times N},\ \mathbf{C}\in\{-1,1\}^{K\times M}.\end{aligned} \tag{5}$$

Moreover, we convert problem (5) to the following problem:

$$\begin{aligned}\min_{\Theta,\mathbf{Z},\mathbf{C},\mathbf{A},\mathbf{V}} \quad & \|\mathbf{Z}^T\mathbf{CA}-\mathbf{S}\|_F^2+\mu\|\mathbf{Z}-F(\mathbf{X};\Theta)\|_F^2 \\ & +\lambda\|\mathbf{V}^T\mathbf{CA}-\mathbf{Y}\|_F^2 \\ \text{s.t.} \quad & \mathbf{1}_M^T\mathbf{A}=l\,\mathbf{1}_N^T, \\ & \mathbf{A}\in\{0,1\}^{M\times N},\ \mathbf{C}\in\{-1,1\}^{K\times M},\end{aligned} \tag{6}$$

where μ is the penalty parameter. When μ is set sufficiently large, the output of the topmost layer of network is forced closer to the real-valued embeddings $\mathbf{Z}$. Generally, problem (6) is a mixed integer programming problem, which is non-convex with Θ, $\mathbf{Z}$, $\mathbf{C}$, $\mathbf{A}$ and $\mathbf{V}$ together. To address this problem, a well-designed alternative optimization algorithm is presented, where only one variable is optimized with the others fixed at each step. The details of the optimization algorithm are elaborated as follows.

We initialize $\hat{\mathbf{B}}$ by ITQ, randomly select M columns to construct $\mathbf{C}$, and randomly set each column $\mathbf{A}$ with one 1.

Z-Step. The problem about $\mathbf{Z}$ is formulated as

$$\min_{\mathbf{Z}} \quad \|\mathbf{Z}^T\mathbf{C}\mathbf{A} - \mathbf{S}\|_F^2 + \mu\|\mathbf{Z} - F(\mathbf{X};\Theta)\|_F^2, \tag{7}$$

which can be easily solved by using matrix manipulations, resulting in a closed-form solution:

$$\mathbf{Z} = (\mathbf{C}\mathbf{A}(\mathbf{C}\mathbf{A})^T + \mu\mathbf{I})^{-1}(\mathbf{C}\mathbf{A}\mathbf{S}^T + \mu F(\mathbf{X};\Theta)), \tag{8}$$

where $\mathbf{I}$ is an identity matrix.

Θ-Step. When all variables are fixed but Θ, the subproblem of network is simplified as a regression problem:

$$\min_{\Theta} \quad \|\mathbf{Z} - F(\mathbf{X};\Theta)\|_F^2. \tag{9}$$

This problem can be solved by SGD readily without any binary constraints and minibatch can be constructed directly. Thus, it allows network learning in a pointwise manner without pair construction or even subset sampling. However, it is intricate to construct minbatch in the original problem (6) in a pairwise manner. Once Θ is updated, the updated $F(\mathbf{X};\Theta)$ can be obtained.

C-Step. Fixing the all variables but $\mathbf{C}$, the subproblem is

$$\begin{aligned} \min_{\mathbf{C}} \quad & \|\mathbf{Z}^T\mathbf{C}\mathbf{A}\|_F^2 + \lambda\|\mathbf{V}^T\mathbf{C}\mathbf{A}\|_F^2 - 2\mathrm{Tr}(\mathbf{R}^T\mathbf{C}) \\ \text{s.t.} \quad & \mathbf{C} \in \{-1,1\}^{K\times M}, \end{aligned} \tag{10}$$

where $\mathbf{R} = \mathbf{Z}\mathbf{S}\mathbf{A}^T + \lambda\mathbf{V}\mathbf{Y}\mathbf{A}^T$, and $\mathrm{Tr}(\cdot)$ is the trace of matrix. $\mathbf{C}$ is updated row by row via the discrete cyclic coordinate descent method [32]. Suppose that $\mathbf{C} = [\mathbf{c}^T;\mathbf{C}']$, $\mathbf{Z} = [\mathbf{z}^T;\mathbf{Z}']$, $\mathbf{V} = [\mathbf{v}^T;\mathbf{V}']$, and $\mathbf{R} = [\mathbf{r}^T;\mathbf{R}']$, where $\mathbf{c}^T$ is one row of $\mathbf{C}$ and $\mathbf{C}'$ is the matrix of $\mathbf{C}$ excluding $\mathbf{c}^T$. In addition, $\mathbf{z}^T$, $\mathbf{Z}'$, $\mathbf{v}^T$, $\mathbf{V}'$, $\mathbf{r}^T$ and $\mathbf{R}'$ are denoted in the similar way.

For different l, there are different complexities for solving $\mathbf{C}$. Thus, we discuss the solving procedure in two cases, *i.e.*, $l = 1$ and $l > 1$. In case of $l = 1$, $\mathbf{U} = \mathbf{A}\mathbf{A}^T$ is a diagonal matrix, which makes the quadratic term about $\mathbf{c}$ be a constant, leading to a binary linear programming problem:

$$\begin{aligned} \min_{\mathbf{c}} \quad & \mathrm{Tr}(\mathbf{c}^T\mathbf{U}(\mathbf{C}')^T\mathbf{Z}'\mathbf{z}) + \mathrm{Tr}(\mathbf{c}^T\mathbf{U}(\mathbf{C}')^T\mathbf{V}'\mathbf{v}) \\ & - \mathrm{Tr}(\mathbf{c}^T\mathbf{r}) \\ \text{s.t.} \quad & \mathbf{c} \in \{-1,1\}^M. \end{aligned} \tag{11}$$

Clearly, the closed-form solution to this simple problem is:

$$\mathbf{c} = \mathrm{sign}(\mathbf{o}), \tag{12}$$

where $\mathbf{o} = \mathbf{r} - \mathbf{U}(\mathbf{C}')^T\mathbf{Z}'\mathbf{z} - \lambda\mathbf{U}(\mathbf{C}')^T\mathbf{V}'\mathbf{v}$. In the case of $l > 1$, $\mathbf{U}$ is not a diagonal matrix. By some simplifications, a binary quadratic programming problem is obtained

$$\min_{\mathbf{c}} \quad \mathbf{c}^T\mathbf{U}\mathbf{c} - 2\mathbf{c}^T\mathbf{f}, \quad \text{s.t.} \quad \mathbf{c} \in \{-1,1\}^M, \tag{13}$$

Algorithm 1 Forward Greedy Algorithm

```
Input: Matrix $\mathbf{M} \in \mathbb{R}^{M\times M}$; the corresponding vector $\mathbf{h} \in \mathbb{R}^M$; hyper-parameters $l$; two index sets $\mathcal{S}$ and $\bar{\mathcal{S}}$.
1:  Initialize $\mathcal{S} = \varnothing$ and $\bar{\mathcal{S}} = \{1,\dots,M\}$.
2:  for $i = 1 \to l$ do
3:      for each index $j$ in $\bar{\mathcal{S}}$ do
4:          Select the index $j$ from $\bar{\mathcal{S}}$, let $\hat{\mathcal{S}} = \mathcal{S} \bigcup j$.
5:          Conduct $\mathbf{a}$ using Eqn. (17) by $\hat{\mathcal{S}}$, where $\|\mathbf{a}\|_0 = i$.
6:          Compute $\mathcal{L}_a(j) = \mathbf{a}^T\mathbf{M}\mathbf{a} - 2\mathbf{a}^T\mathbf{h}$.
7:      end for
8:      $p = \arg\min_j \mathcal{L}_a(j)$.
9:      $\mathcal{S} = \mathcal{S} \bigcup p$, $\bar{\mathcal{S}} = \bar{\mathcal{S}} - p$.
10: end for
Output: Set $\mathcal{S}$; indicator vector $\mathbf{a}$.
```

where $\mathbf{f} = \mathbf{o}/\|\mathbf{z}\|_2^2$. By dropping the binary constraints, the approximate solution in relaxation objective is obtained:

$$\mathbf{c} = \mathrm{sign}(\mathbf{U}^{-1}\mathbf{f}). \tag{14}$$

A-Step. Let $\mathbf{G}_1 = \mathbf{Z}^T\mathbf{C} \in \mathbb{R}^{N\times M}$ and $\mathbf{G}_2 = \mathbf{V}^T\mathbf{C} \in \mathbb{R}^{C\times M}$, and the problem of $\mathbf{A}$ can be rewritten as follows:

$$\begin{aligned} \min_{\mathbf{A}} \quad & \|\mathbf{G}_1\mathbf{A} - \mathbf{S}\|_F^2 + \lambda\|\mathbf{G}_2\mathbf{A} - \mathbf{Y}\|_F^2 \\ \text{s.t.} \quad & \mathbf{A} \in \{0,1\}^{M\times N}. \end{aligned} \tag{15}$$

Clearly, the subproblems about the columns of $\mathbf{A}$ are separable so that $\mathbf{A}$ can be optimized column by column. When solving one column $\mathbf{a}$ of $\mathbf{A}$, the corresponding problem is

$$\begin{aligned} \min_{\mathbf{a}} \quad & \mathcal{L}_a = \mathbf{a}^T\mathbf{M}\mathbf{a} - 2\mathbf{a}^T\mathbf{h} \\ \text{s.t.} \quad & \|\mathbf{a}\|_0 = l,\ \mathbf{a} \in \{0,1\}^M, \end{aligned} \tag{16}$$

where $\|\cdot\|_0$ is ℓ_0 norm, $\mathbf{M} = \mathbf{G}_1^T\mathbf{G}_1 + \lambda\mathbf{G}_2^T\mathbf{G}_2 \in \mathbb{R}^{M\times M}$, and $\mathbf{h} = \mathbf{G}_1^T\mathbf{s} + \lambda\mathbf{G}_2^T\mathbf{y} \in \mathbb{R}^M$. Besides, $\mathbf{s}$ and $\mathbf{y}$ are the corresponding columns to $\mathbf{a}$ in $\mathbf{S}$ and $\mathbf{Y}$, respectively.

Problem (16) is a constrained binary quadratic programming problem, which is generally difficult to solve. Instead of the continuous relaxation, we investigate how to discretely solve $\mathbf{a}$ in two cases, *i.e.*, $l = 1$ and $l > 1$. Specifically, an index set is defined by $\mathcal{S}$ that recorders the indexes of 1 entries in $\mathbf{a}$, so that the solution of $\mathbf{a}$ can be represented by Eqn. (17). Evidently, in the case of $l = 1$, the optimal index of $\mathbf{a}$ can be readily attained by $j = \arg\min_{j=1,\dots,M} m_{jj} - 2h_j$.

While a forward greedy algorithm is proposed to address the case of $l > 1$. The main procedure of this algorithm is detailed as follows. First, we define two index sets of $\mathbf{a}$, that is, $\mathcal{S} = \varnothing$ and $\bar{\mathcal{S}} = \{1,\dots,M\}$. Second, an index p in $\bar{\mathcal{S}}$ is identified if $\mathcal{L}_a$ in (16) is minimized with $a_p = 1$. Then, the index p is moved from $\bar{\mathcal{S}}$ to $\mathcal{S}$. Third, we successively find one index in $\bar{\mathcal{S}}$ at a time that minimizes $\mathcal{L}_a$ combined with the indices in $\mathcal{S}$, and move this index to $\mathcal{S}$. Once the cardinality of $\mathcal{S}$ equals to l, the algorithm stops. Finally, the approximate solution to (16) is obtained by

$$a_i = \begin{cases} 1, & \text{if } i \in \mathcal{S}; \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Algorithm 1 summarizes the forward greedy method.

Algorithm 2 Nonlinear Asymmetric Multi-Valued Hashing

```
Input: Training data $\mathbf{X} \in \mathbb{R}^{D\times N}, \mathbf{Y} \in \{0,1\}^{C\times N}$, and $\mathbf{S} \in \mathbb{R}^{N\times N}$; code length $K$; dictionary size $M$; maximum iteration number $t$; hyper-parameters $l, \lambda, \mu, \alpha$.
1: Initialize $\mathbf{C}$ by ITQ and $\mathbf{A}$ by random initialization. And $F(\mathbf{X};\Theta)$ is initialized by zeros.
2: for iter $i = 1 \to t$ do
3:     Z-Step: Update $\mathbf{Z}$ using Eqn. (8)
4:     $\Theta$-Step: Update the parameters of neural network using chain rule, and attain the updated $F(\mathbf{X};\Theta)$.
5:     C-Step: Circularly update $\mathbf{C}$ row by row using Eqn. (12) for $l = 1$ and using Eqn. (14) for $l > 1$.
6:     A-Step: Update $\mathbf{A}$ column by column using optimal solution for $l = 1$ and using Algorithm 1 for $l > 1$.
7:     V-Step: Update $\mathbf{V}$ using Eqn. (18).
8: end for
Output: Network parameters $\Theta$; dictionary $\mathbf{C}$; indicator matrix $\mathbf{A}$; classification matrix $\mathbf{V}$.
```

V-Step. The subproblem of $\mathbf{V}$ is a least squares regression problem. The closed-form solution is obtained as

$$\mathbf{V} = (\mathbf{CA}(\mathbf{CA})^T)^{-1}(\mathbf{CA})\mathbf{Y}^T. \tag{18}$$

For clarity, the whole alternative optimization algorithm of NAMVH is summarized in Algorithm 2. We made the code available at https://github.com/dcfucheng/NAMVH.

3.7 Incremental Extension of Database Samples

The obvious defect of coupled asymmetric hashing methods is that the entire codes of database remain unchanged completely after training procedure [28], [32]. Namely, when the database samples evolve dynamically, the methods are obliged to retrain from scratch with the newly entire data samples, resulting in inefficient computation. Fortunately, although NAMVH directly learns embeddings of database points, it can also allow a simple method for incremental extension of database samples, benefiting from the sophisticated structure of our method. Particularly, due to the product form of the multi-integer-valued embeddings, NAMVH is capable of updating the embeddings indirectly by updating the dictionary, with the existing indicator matrix fixed [59]. Moreover, the optimization of indicator matrix is column-separable, and hence enabling access to the indicator of each newly-coming sample individually. Formally, the incremental extension of database is formulated as follows:

$$\begin{aligned}\min_{\mathbf{Z}_n,\mathbf{C},\mathbf{A}_n,\mathbf{V}} \quad & \|[\mathbf{Z},\mathbf{Z}_n]^T\mathbf{C}[\mathbf{A},\mathbf{A}_n] - \mathbf{S}_{N+n}\|_F^2 \\ & + \mu\|[\mathbf{Z},\mathbf{Z}_n] - F([\mathbf{X},\mathbf{X}_n];\Theta)\|_F^2 \\ & + \lambda\|\mathbf{V}^T\mathbf{C}[\mathbf{A},\mathbf{A}_n] - [\mathbf{Y},\mathbf{Y}_n]\|_F^2 \\ \text{s.t.} \quad & \mathbf{1}_M^T\mathbf{A}_n = l\,\mathbf{1}_n^T, \\ & \mathbf{A}_n \in \{0,1\}^{M\times n},\ \mathbf{C} \in \{-1,1\}^{K\times M},\end{aligned} \tag{19}$$

where $\mathbf{Z}_n, \mathbf{A}_n$ and $\mathbf{Y}_n$ are the variables corresponding to n newly-coming data samples $\mathbf{X}_n \in \mathbb{R}^{D\times n}$. The term of $[\mathbf{Z},\mathbf{Z}_n] \in \mathbb{R}^{d\times(N+n)}$ represents the concatenated matrix of $\mathbf{Z}$ and $\mathbf{Z}_n$ by column, and the other concatenated terms have the similar definitions. And $\mathbf{S}_{N+n} \in \mathbb{R}^{(N+n)\times(N+n)}$ represents the similarity matrix of the all data points.

Algorithm 3 Incremental Extension of Database Samples

```
Input: The variables from Algorithm 2: Network parameters $\Theta$, $\mathbf{Z} \in \mathbb{R}^{K\times N}$, $\mathbf{C} \in \{-1,1\}^{K\times M}$, $\mathbf{A} \in \{0,1\}^{M\times N}$ and $\mathbf{V} \in \mathbb{R}^{K\times C}$; the newly-coming data samples $\mathbf{X}_n \in \mathbb{R}^{D\times n}, \mathbf{Y}_n \in \{0,1\}^{C\times n}$ and the whole similarity $\mathbf{S}_{N+n} \in \mathbb{R}^{(N+n)\times(N+n)}$; maximum iteration number $t$; hyper-parameters $l, \lambda, \mu$.
1: Initialize $\mathbf{Z}_n$ by $F(\mathbf{X}_n;\Theta)$.
2: for iter $i = 1 \to t$ do
3:     $\mathbf{A}_n$-Step: Update $\mathbf{A}_n$ column by column using optimal solution for $l = 1$ and using Algorithm 1 for $l > 1$, while $\mathbf{A}$ is fixed.
4:     C-Step: Integrate $\mathbf{Z}_n, \mathbf{A}_n, \mathbf{Y}_n$ into Problem (10), and update $\mathbf{C}$ row by row using Eqn. (12) for $l = 1$ or Eqn. (14) for $l > 1$ circularly.
5:     V-Step: Integrate $\mathbf{C}_n, \mathbf{A}_n, \mathbf{Y}_n$ into Eqn. (18), and update $\mathbf{V}$ by it.
6:     $\mathbf{Z}_n$-Step: Integrate $\mathbf{C}_n, \mathbf{A}_n, \mathbf{S}_{N+n}$, $F(\mathbf{X}_n;\Theta)$ into Eqn. (8), and update $\mathbf{Z}_n$ by it, while $\mathbf{Z}$ is fixed.
7: end for
Output: Dictionary $\mathbf{C}$; indicator matrix $\mathbf{A}_n$.
```

The goal of problem (19) is to optimize $\mathbf{Z}_n, \mathbf{C}, \mathbf{A}_n$ and $\mathbf{V}$ for accommodating the newly-coming database samples with the already learned network fixed. Once the Algorithm 2 is done, the variables $\mathbf{V}, \mathbf{C}, \mathbf{A}$ and $\mathbf{Z}$ for the original samples can be initialized, and $F([\mathbf{X},\mathbf{X}_n];\Theta)$ can be produced by the learned neural network. $\mathbf{Z}_n$ is simply initialized by $F(\mathbf{X}_n;\Theta)$. Then, due to the column separability of the optimization in $\mathbf{A}$-step, we can just update each $\mathbf{a}$ in $\mathbf{A}_n$ respectively, while remain the original $\mathbf{A}$ fixed. Meanwhile, $\mathbf{Z}_n$ can also be updated individually with $\mathbf{Z}$ fixed. Finally, all the variables $\mathbf{Z}_n, \mathbf{C}, \mathbf{A}_n$ and $\mathbf{V}$ can be updated in an alternative manner. Having done so, the new dictionary suitable for the whole database samples is attained, the indicators of the newly-coming points $\mathbf{A}_n$ are produced efficiently, and hence leading to the updated multi-integer-valued embeddings of the whole database samples.

For clarity, the detailed optimization algorithm of incremental extension of database samples is summarized in Algorithm 3. It is crucial to noted that the encoding mechanism for newly-coming database samples can be regarded as a specific "out of sample" of database samples with none explicit hash function, which is different from "out of sample" of query. Conclusively, the proposed mechanism differs from online hashing [60], which can solve the hashing learning task on the sequential data (or stream data).

3.8 Efficient Query Strategy

Given a query vector $\mathbf{q}$, the goal is to find some items from the database that are similar to the query $\mathbf{q}$. We first project it onto low-dimensional space through the network by $\hat{\mathbf{q}} = F(\mathbf{q};\Theta)$. Then, the similarities between $\hat{\mathbf{q}}$ and the multi-integer-valued embeddings $\mathbf{CA}$ of the database points are calculated by $\mathbf{S}_q = (\hat{\mathbf{q}})^T\mathbf{CA}$. Once $\mathbf{S}_q$ is obtained, the nearest neighbors are returned by sorting these similarities.

To improve the computational efficiency, a lookup table is built for the query, denoted by $\mathbf{T}_c = (\hat{\mathbf{q}})^T\mathbf{C}$. As for computing $\mathbf{S}_q = \mathbf{T}_c\mathbf{A}$, instead of multiplication, only the table lookup and addition operations are required, according to the binary sparse indicator matrix $\mathbf{A} \in \{0,1\}^{M\times N}$. Specifically, we still discuss how to calculate $\mathbf{S}_q$ in two cases. In the case of $l = 1$, the multi-integer-valued embeddings degrade into binary codes so that only the table lookup operations on $\mathbf{T}_c$ are required. As $l > 1$, besides the table lookup operations, l selected atoms have to be summed together. Notably, if we construct the binary codes of the query by sign($\hat{\mathbf{q}}$), NAMVH can also perform similarity search based on the Hamming distance. The differences of these query strategies are clearly shown in Table 1, where the superscript represents the encoding type of the query and the subscript represents the hyper-parameter l. 'Ham', 'QsHam', 'Ori', and 'Con' represent the Hamming distance, the query sensitive Hamming distance, the original embeddings, and the concatenate embeddings, respectively.

TABLE 1
A summary of different query strategies of NAMVH.

Method		Query	Database	Distance
$l = 1$	NAMVH$^b_1$	Binary	Binary	Ham+Ori
	NAMVH$^r_1$	Real	Binary	QsHam+Ori
$l > 1$	NAMVH$^b_{10}$	Binary	Multi-integer	Ham+Con
	NAMVH$^r_{10}$	Real	Multi-integer	QsHam+Con

3.9 Complexity Analysis

We discuss the complexity of the proposed NAMVH. Assume that $N \gg D > M > C, K$, where N is the number of training samples, D is the dimension of features, K is the code length, M is the number of atoms of dictionary.

3.9.1 Analysis on computational complexity

We analyze the complexity of updating $\mathbf{Z}, \mathbf{C}, \mathbf{V}$ and $\mathbf{A}$ in one iteration. The computational complexity of $\mathbf{Z}$-step is $\mathcal{O}(NKM)$, $\mathbf{C}$-step is $\mathcal{O}(NMK^2)$ and $\mathbf{V}$-step is $\mathcal{O}(NKM)$. As for $\mathbf{A}$-step, since the sparsity of indicator vector $\mathbf{a}$ and the symmetry of $\mathbf{M}$ in $\mathcal{L}_a$, it takes $\mathcal{O}(N\sum_{j=1}^{l}(M-j)j)$ to perform forward greedy algorithm. Meanwhile, the computational complexity of $\mathbf{M}$ scales in $\mathcal{O}(NK^2)$. Consequently, the complexity of $\mathbf{A}$-step scales in $\mathcal{O}(max(N\sum_{j=1}^{l}(M-j)j, NK^2))$. As for incremental extension, the complexity of $\mathbf{Z}_n, \mathbf{C}, \mathbf{V}$ and $\mathbf{A}_n$-step is $\mathcal{O}((N+n)KM)$, $\mathcal{O}((N+n)MK^2)$, $\mathcal{O}((N+n)KM)$ and $\mathcal{O}(max(n\sum_{j=1}^{l}(M-j)j, (N+n)K^2))$. Additionally, the neural network is trained in a pointwise manner by leveraging the auxiliary variable. In brief, even if the pairwise information is employed, our algorithm has linear time complexity with respect to the size of training set, which is fairly scalable to large training data.

3.9.2 Analysis on query complexity

We evaluate the query complexity of NAMVH in terms of the Hamming ranking. As mentioned in Section 3.8, for a single query, we only need the computation of $\mathbf{T}_c$ that scales in $\mathcal{O}(KM)$ and some table lookup and addition operations to calculate $\mathbf{T}_c\mathbf{A}$, which scale in $\mathcal{O}(l\,N)$. Due to $KM \ll N$, the query complexity mainly relies on the size of search database N and the hyper-parameter l, rather than the code length K. Therefore, the query time of NAMVH is constant for all code lengths, differing from the traditional binary hashing methods, whose query complexity depends on the code length K, which prevents the use of long codes.

3.9.3 Analysis on storage complexity

Compared with conventional binary hashing methods, the storage complexity of NAMVH is acceptable. Due to the fact that the memory cost of the dictionary $\mathbf{C} \in \{-1,+1\}^{K\times M}$ is negligible, the database storage mainly lies on the indicator matrix $\mathbf{A} \in \{0,1\}^{M\times N}$. In view of the fixed sparsity of $\mathbf{a}$, only l indices of 1s in each $\mathbf{a}$ are required to store for each database point. Specifically, the storage of each $\mathbf{a}$ is $l \log M$ bits. Therefore, an interesting observation is that the storage of NAMVH only lies on the hyper-parameter l and the number of atoms M, rather than the code length K. Therefore, if appropriate values of l and M are set, the storage of NAMVH might be less than that of conventional binary ones, even if the non-binary embeddings are employed.

4 Experiments

4.1 Experimental Settings

4.1.1 Datasets

Five relatively small public datasets (ESP-GAME [61], MIR-FLICKR [62], NUS-WIDE [63], CIFAR10[1], and MNIST[2]) and two large ones (RCV [64], ILSVRC14 [65]) are adopted to evaluate the performance of NAMVH. The ESP-GAME dataset consists of $20,768$ images, with each image labeled with multiple semantic labels from 268 categories. The MIR-FLICKR dataset includes $25,000$ images crawled from Flickr, with each image associated with multiple semantic labels from 38 categories. Each image of both ESP-GAME and MIR-FLICKR is represented by a 512-dimensional GIST [66] feature vector. NUS-WIDE contains $269,648$ images collected from Flickr, each of which belongs to multiple categories taken from 81 concept tags. $209,347$ images are collected by removing the images without any labels [28]. And the provided 500-dimensional bag-of-words (BoW) features are utilized. The CIFAR10 dataset, a subset of well-known 80M Tiny image benchmark [67], contains $60,000$ color images classified into 10 categories, where each image is represented by a 512-dimensional GIST feature vector. The MNIST dataset consists of $70,000$ handwritten digit images from '0' to '9'. Each image is represented by a 784-dimensional feature vector of gray scale pixels.

Reuters Corpus Volume I (RCV1) is a large collection of over $800,000$ manually categorized newswire stories with 2456 multiple labels. The high-dimensional sparse feature of each news is reduced to 1000 by sparse random projection technique [68]. ILSVRC14, a subset of ImageNet [65], consists of 1.2 million images with 1000 categories. 4096-dimensional convolutional features are extracted, and then the dimension is reduced to 512 by principal component analysis. For five small datasets, each dataset is randomly split into a query set with 1000 samples and a training set with the remaining samples for evaluation. In RCV, $10K$ samples are randomly selected to conduct a query set and the remain ones for training. As for ILSVRC14, we randomly select 3 samples in each categories to form a query set with 3000 samples and a training set with the remaining samples.

1. http://www.cs.toronto.edu/kriz/cifar.html
2. http://yann.lecun.com/exdb/mnist/

4.1.2 Baseline methods

We compare our NAMVH against several classical binary hashing methods, including eight shallow hashing methods, *i.e.*, ITQ-CCA [3], KSH [30], FastHash [29], SDH [32], COSDISH [28], SGH [26], LIN:V [36], AMVH [40] and two deep hashing methods, *i.e.*, SH-BDNN [38] and SSDH [7]. Typically, the published codes of SSDH dose not support the multi-label dataset. For the pairwise hashing methods with high computational complexity (*i.e.*, KSH, FastHash, LIN:V and SH-BDNN), $10K$ samples are randomly selected from training set for learning. While all the training samples are utilized for the remaining methods, due to their favorable scalability. Notably, LIN:V is a representatively asymmetric hashing method that learns binary codes of database in training stage directly. Thus, only $10K$ training points in database can be searched in evaluation.

4.1.3 Evaluation criteria

Three widely used criteria: mean average precision (MAP), top-K precision and NDCG@100 are adopted to evaluate retrieval performance. As our approach is applied for semantic search, the true neighbors (groundtruth) are the samples that share at least one label with the query. Since our approach is beyond binary coding, the construction of the lookup table to find the nearest neighbors within a specific distance (*e.g.*, Hamming radius 2) may not be realized directly. Therefore, our approach mainly focuses on the efficiency in terms of the Hamming ranking. We report the query time under different code lengths to evaluate the search efficiency, which consists of two parts time for all methods. Specifically, one is the encoding time for single one query sample, namely the time of feature mapping with the hash functions. The other one is the time of similarity matrix construction, which determines the similarities between the query and all database samples. As the top ranked samples are returned as the retrieved results subsequently in the same experimental settings, the sort time is omitted.

4.1.4 Implementation details

For the compared methods, the public codes and the suggested parameters from the corresponding authors are utilized. In the kernel-based methods (*i.e.*, KSH, SGH, SDH, COSDISH, LIN:V and AMVH), 1000 anchors are randomly chosen for kernel mapping. Boost trees and kernel ridge regression are adopted for FastHash and COSDISH. We employ a 4-layer neural network with $200, 120, 100, K$ dimension of each layer as nonlinear transformation $F(\cdot)$. ReLU or TanH nonlinear activation functions are used for previous layers, but the linear function is used for the last layer. Batch normalization [69] is used in each layer. The parameters of each layer are initialized by Gaussian. As for SSDH, we replace the CNN feature extractor of SSDH by the previous layers of our proposed network for fair comparison. Here α for similarity transformation is set to $\nu n_s/n_d$, where ν is a hyper-parameter; n_s and n_d are the number of similar and dissimilar pairs, respectively. Moreover, we set parameter λ to $10\,N/C$, μ to $10\,N/K$ and ν to 0.5 via cross-validation; dictionary size to 256, maximum iteration number t to 10, and l to 1 or 10; iteration, momentum and weight decay to 50000, 0.9 and 0.004; learning rate and batch size of small datasets to 0.02 and 100; large ones to 0.05 and 1000; dictionary $\mathbf{C}$ is circularly updated 10 times. We implement our method with CAFFE [70] on an NVIDIA Titan Xp GPU. All the experiments are conducted on a 64-bit Linux Server with 512 GB RAM and 2.40 GHz CPU.

4.2 Retrieval Performance

The exhaustive retrieval results on five relatively small scale datasets are illustrated in Fig. 3, and those on two large scale datasets are reported in Table 2. In general, these experimental results show that both NAMVH$^r_{10}$ and AMVH$^r_{10}$ achieve appealing performance over the other competitors, demonstrating the effectiveness of the multi-valued encoding strategy. Furthermore, NAMVH$^r_{10}$ outperforms AMVH$^r_{10}$ by a large margin in various criteria. This empirically verifies the capability of the learned neural network, which can construct a powerful nonlinear transformation for encoding effective real-valued embeddings.

For the three multi-label datasets (*i.e.*, ESP-GAME, MIRFLICKR and NUS-WIDE) in the first three rows of Fig. 3, we have the following observations. First, the performance of the unsupervised method (SGH) is worse than the supervised ones. Moreover, the asymmetric hashing methods (COSDISH, SDH, LIN:V, AMVH and NAMVH) achieve preferable performance over the symmetric ones. Second, NAMVH$^r_1$ outperforms the methods in which the binary codes are utilized for database (*e.g.*, NAMVH$^b_1$,COSDISH and SDH) in most cases. This shows that the real-valued query can improve the search accuracy. Third, compared with NAMVH$^r_1$, NAMVH$^r_{10}$ further enhances the capability of similarity preservation, achieving the highest performance. These results verify the superiority of the multi-integer-valued encoding strategy. Fourth, for MAP and NDCG, it is worthy to note that the performance of NAMVH$^r_1$ and NAMVH$^r_{10}$ with 8 bits is remarkably superior to that of SDH and COSDISH with 64 bits, indicating that NAMVH can reach satisfying performance with short code length.

Referring to the two multi-class datasets (*i.e.*, CIFAR10 and MNIST) in the bottom rows of Fig. 3, the similar conclusion that asymmetric hashing methods (COSDISH, SDH, LIN:V, AMVH and NAMVH) achieve promising results can also be drawn. NAMVH still surpasses the other competitors, while SSDH [7] also provides comparable results on these two datasets, due to the practical nonlinear hash function derived from a classification network. Nevertheless, NAMVH also obtains 24.01% and 26.33% improvements over SSDH on Precision@51 and NDCG with 16 bits, respectively. Additionally, the performance of NAMVH$^r_1$ is nearly close to that of NAMVH$^r_{10}$ in some cases. One possible reason is that NAMVH with lower l already obtains satisfactory search accuracy, while the optimization algorithm may reach suboptimal solutions with larger l in these datasets.

To evaluate our approach on large scale datasets, experiments are also conducted on ILSVRC14 (1.2M) and RCV (800K). Then, COSDISH, SDH and SSDH are adopted as compared methods, due to their favorable scalability. The retrieval results are listed in Table 2. Specifically, on ILSVRC14, the MAP performance of SSDH is superior to conventional hashing methods (*i.e.*, COSDISH and SDH), demonstrating the effectiveness of nonlinear hash function. NAMVH$^r_{10}$ remarkably surpasses SSDH in term of all

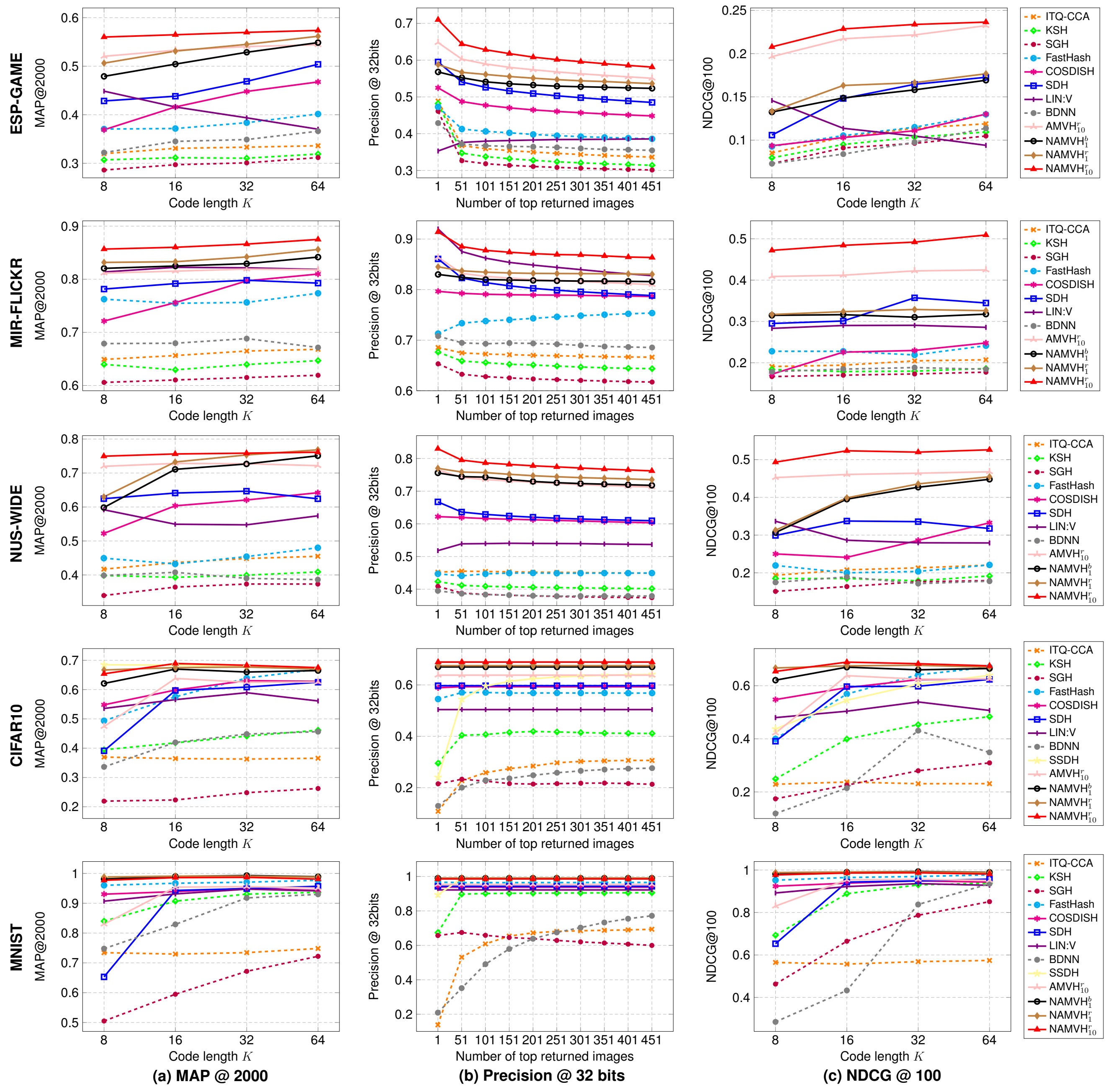
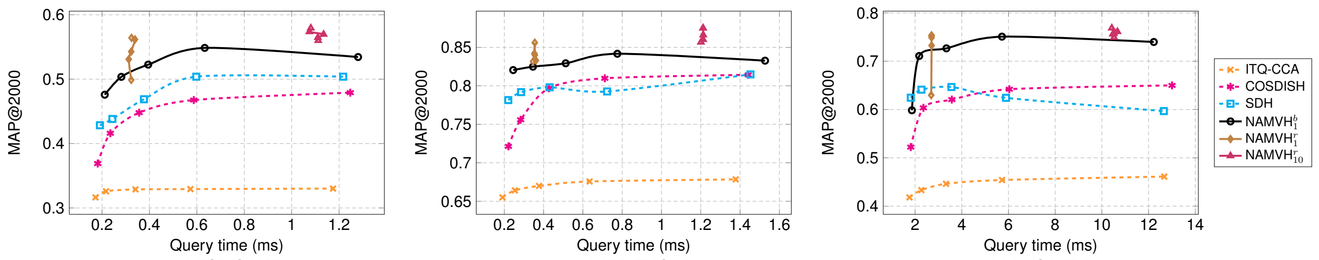


Fig. 3. Retrieval performance of different hashing methods on five relatively small datasets. From top row to bottom row, they are ESP-GAME, MIR-FLICKR, NUS-WIDE, CIFAR10 and MNIST, respectively. First column: (a) MAP@2000 with different codes lengths; Second column: (b) top-K precision@32 bits w.r.t. different numbers of top returned images; Third column: (c) NDCG@100 with different codes lengths. Best viewed in colors.

TABLE 2
Retrieval performance of different hashing methods on two large scale dataset RCV and ILSVRC14.

Method	RCV (800K, 512-SPR)						ILSVRC14 (1.2M, 512-CNN)					
	MAP@2000				Precision@101	NDCG@100	MAP@2000				Precision@101	NDCG@100
	8 bits	16 bits	32 bits	64 bits	32 bits	32 bits	8 bits	16 bits	32 bits	64 bits	32 bits	32 bits
COSDISH	0.5182	0.6973	0.7405	0.8091	0.7308	0.3884	0.0092	0.0259	0.0643	0.1620	0.0500	0.0477
SDH	0.5644	0.6714	0.7569	0.7814	0.7351	0.5382	0.0249	0.0841	0.1782	0.2754	0.1710	0.1710
AMVH^r_{10}	0.6714	0.7464	0.7852	0.8002	0.7885	0.6156	0.0422	0.1221	0.2122	0.2709	0.1957	0.1957
SSDH	-	-	-	-	-	-	0.0393	0.1666	0.3094	0.3615	0.2114	0.1895
NAMVH^b_1	0.5901	0.7386	0.7874	0.8210	0.7816	0.5213	0.0651	0.1180	0.1295	0.1585	0.1137	0.1137
NAMVH^r_1	0.5943	0.7509	0.7894	0.8276	0.7875	0.5241	0.0691	0.1233	0.1325	0.1602	0.1163	0.1163
NAMVH^b_{10}	0.6689	0.7733	0.8482	0.8634	0.8527	0.6551	0.0552	0.1117	0.2245	0.3463	0.2117	0.2117
NAMVH^r_{10}	0.6795	0.7974	0.8757	0.8928	0.8776	0.7066	0.0837	0.2179	0.3542	0.4093	0.3230	0.3230



(a) ESP-GAME (b) MIR-FLICKR (c) NUS-WIDE

Fig. 4. Query time comparison and the correspondingly search performance MAP@2000 under various code lengths on (a) ESP-GAME, (b) MIR-FLICKR and (c) NUS-WIDE. The markers from left to right on each curve indicate the code length of $8, 16, 32, 64$, and 128, respectively. The vertical axis represents the search performance MAP, and the horizontal axis corresponds to the query time cost (milliseconds).

criteria. Concretely, NAMVH$_{10}^r$ gains 52.79% and 70.45% improvement over SSDH on Precision@101 and NDCG, respectively. On RCV, we can see that the performance of NAMVH$_1^r$ already prevails over that of COSDISH and SDH, and NAMVH$_{10}^r$ also achieves the best performance. Overall, NAMVH$_{10}^r$ surpasses AMVH$_{10}^r$ in various criteria.

In summary, SSDH sufficiently exploits pointwise label information to learn nonlinear hash functions, and thus it achieves promising results on small multi-class datasets, while the performance on large ones is not good enough. SH-BDNN, LIN:V and FastHash can only be constructed on the sampled pairwise similarity, leading to suboptimal results. SDH and COSDISH depend on the kernel-based functions, so the performance is limited. Typically, all the hashing methods are subject to the binary constraints. In contrast, we attribute the merit of NAMVH to the proposed multi-valued embeddings. NAMVH can efficiently perform network learning with the whole pairwise information, and the real-valued outputs are preserved for use. Also, mostly important, the multi-integer-valued embeddings can further enhance the capability of similarity preservation, resulting in preferable performance on various datasets.

4.3 Evaluation on Search Efficiency

To evaluate the efficiency of NAMVH, the MAP performance and the correspondingly query time under different code lengthes are illustrated in Fig. 4. Some conclusions can be drawn. First, as the Hamming distance is utilized to perform similarity search in ITQ-CCA, COSDISH, SDH and NAMVH$_1^b$, the query time of these methods reveals the same increasing tendency as the code length increases. The longer codes take more query time, and achieve more preferable performance. ITQ-CCA takes a little shorter query time as the liner hash function is used. While COSDISH and SDH employ kernel-based ones, and NAMVH$_1^b$ adopts neural network, resulting in longer time. Second, the query time of NAMVH$_1^r$ and NAMVH$_{10}^r$ is almost the same for all code lengths. The underlying reason is that the query time only depends on l and the database size N, rather than the code length K. Specifically, NAMVH$_1^r$ only takes slightly longer query time than SDH with 16 bits, and NAMVH$_{10}^r$ requires shorter time than SDH with 128 bits. As a result, confronted with the calculation of the weighted Hamming distance, NAMVH leverages BSR to perform the efficient search, and hence achieving such desirable performance and retaining the search efficiency simultaneously. Moreover, both the query time and retrieval performance rely on l in our approach, it is flexible to make a trade-off between them.

4.4 Evaluation on Storage Efficiency

As analysed in section 3.9.2, the storage of our method relies on the dictionary size M and hyper-parameter l, rather than the code length K. For example, the storage of NAMVH$_{10}^r$ with 32 atoms is $10 \times \log_2 32 = 50$ bits per sample, which approximately equals to that of the binary hashing methods with 48 bits. Referring to the Fig. 5 that illustrates the effect of the dictionary size, NAMVH$_{10}^r$ with 32 atoms is greatly superior to the binary hashing methods with approximate storage cost (*i.e.*, 48 bits). Typically, NAMVH$_{10}^r$ with 16 atoms (*i.e.*, 40 bits) already outperforms the binary ones by 13.50%, 4.04% and 19.20% on three datasets, respectively. In addition, NAMVH$_1^r$ of 32 atoms with $1 \times \log_2 32 = 5$ bits far less than 48 bits already surpasses the binary hashing methods. To sum up, NAMVH in a way of the multi-valued embeddings is able to achieve favorable performance with even less storage cost than binary hashing methods.

4.5 Performance on Incremental Extension

We split training set into a sub-training set randomly sampled with various sample rates $\pi = [0.05, 0.2, 0.4, 0.6, 0.8]$ and an extended set with the remaining samples for the evaluation of incremental extension. For all the methods, only the sub-training set is used for training, while the relevant extended set is just for incremental extension. To be specific, there are two variants of each method (*i.e.*, non-extension and extension). The original methods (COSDISH, SDH and NAMVH$_{10}^r$) represent for non-extension, where the embeddings of the sub-training set are directly learned in the asymmetric manner. Thus, only the sub-training set can be searched by the queries. The histograms in Fig. 6 illustrate the MAP performance of these methods.

Additionally, e-COSDISH, e-SDH and e-NAMVH$_{10}^r$ represent the methods for extension, where the extended set should be further encoded for search after the training on the sub-training set. Typically, generic asymmetric hashing can not perform incremental extension of database explicitly, so that we adopt a compromising (symmetric) strategy to perform extension by utilizing the learned hash functions from the sub-training set to encode the whole data set. On the contrary, e-NAMVH$_{10}^r$ first learns the embeddings of the sub-training set, and then executes incremental extension to further update the embeddings of the sub-training set and produce the new ones of the extended set. The curves in Fig. 6 demonstrate the results of the extension methods.

Referring to Fig. 6, we can obviously see that the higher MAP is achieved as the size of the sub-training set increases, since the more training data can facilitate the hash function

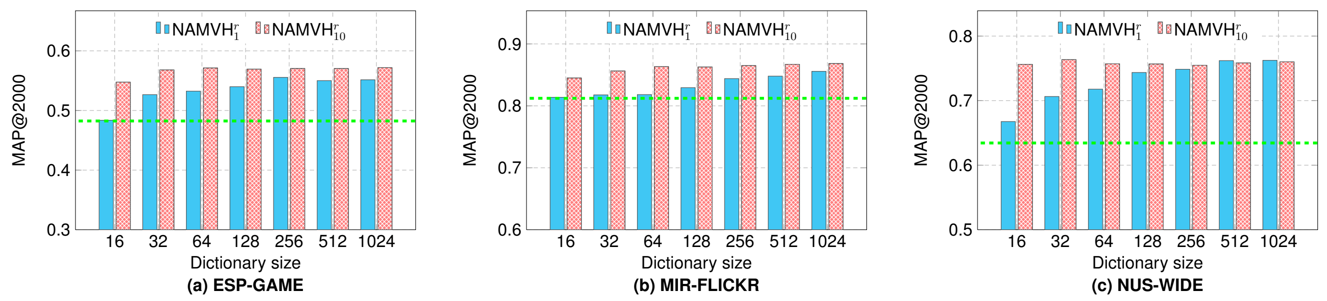


Fig. 5. The search performance MAP@2000 of NAMVH$_1^r$ and NAMVH$_{10}^r$ on the different dictionary sizes on (a) ESP-GAME, (b) MIR-FLICKR and (c) NUS-WIDE with 32-bit codes. Green dashed lines represent the highest MAP of the compared binary hashing methods with 48-bit codes.

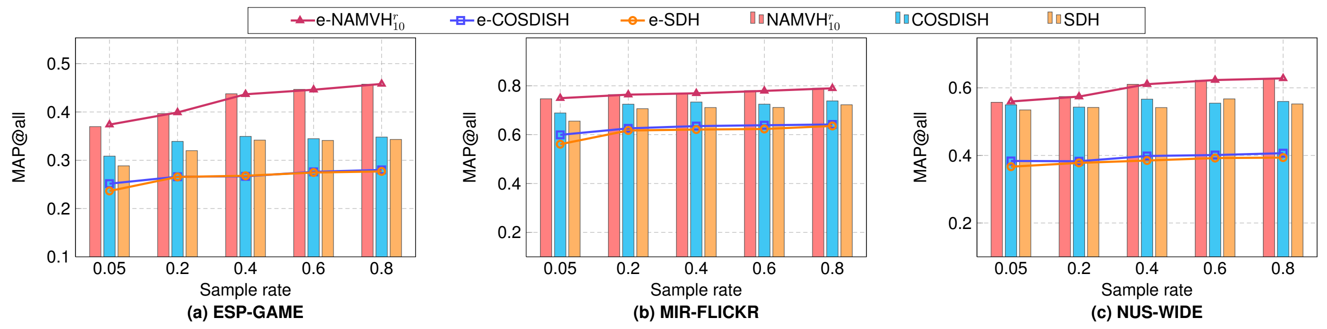


Fig. 6. The search performance MAP@all of the extension methods (e-COSDISH, e-SDH and e-NAMVH$_{10}^r$) and the correspondingly non-extension methods (COSDISH, SDH and NAMVH$_{10}^r$) under various sample rates π on (a) ESP-GAME, (b) MIR-FLICKR and (c) NUS-WIDE with 16-bit codes.

learning. Typically, the performance of e-COSDISH and e-SDH is inferior to that of COSDISH and SDH, respectively. This indicates that the symmetric encoding strategy is worse than asymmetric one, and using the learned hash function to indirectly implement the incremental extension of database does not prove effective. In contrast, e-NAMVH$_{10}^r$ achieves comparable results as NAMVH$_{10}^r$ at all sample rates (even 5%), which verifies that the embeddings generated by the incremental extension indeed guarantee the approximate search accuracy of NAMVH$_{10}^r$. Meanwhile, e-NAMVH$_{10}^r$ surpasses e-COSDISH and e-SDH by a large margin. Conclusively, these results demonstrate the effectiveness of the incremental extension of database samples in NAMVH.

4.6 More In-Depth Analysis

4.6.1 *The effect of sparsity l*

The setting of the hyper-parameter l is pivotal in our method, which controls the capability of the multi-integer-valued embeddings. To empirically verify the effect of l, the MAP performance with diverse l is reported in Fig. 7. In general, NAMVH on 16 bits is better than that on 8 bits in most cases, which conforms to the observations in Fig. 3. Moreover, the search performance has a strong relation to l. On ESP-GAME and NUS-WIDE, the MAP increases rapidly with l from 1 to 10, and already reaches high MAP with a small l (*i.e.*, $l < 5$). Finally, it tends to be saturated when $l > 10$. On CIFAR10, NAMVH often obtains a little improvement as l increases on 16 bits, while NAMVH with 8-bit codes gets worse when $l > 10$. As for MIR-FLICKR, the MAP also turns out an increasing trend with l, but is subject to a slight fluctuation. Consequently, this implies that choosing a larger l is not a strict requirement. The underlying reason is that the optimization algorithm may obtain suboptimal solutions when l is large. For this reason, l is set to 10 in NAMVH to demonstrate the superiority of the multi-integer-valued embeddings empirically.

4.6.2 *The effect of dictionary size M*

To investigate the effect of dictionary size M on the search performance, we show the MAP results w.r.t. different dictionary sizes in Fig. 5. Apparently, the MAP of NAMVH$_1^r$ gradually becomes higher with more atoms. While NAMVH$_{10}^r$ always keeps stably satisfying MAP values even in the extreme case of 16 atoms. Typically, NAMVH$_{10}^r$ with 16 atoms outperforms the binary hashing. While NAMVH$_1^r$ requires more atoms to keep pace with NAMVH$_{10}^r$, but still surpasses the binary ones in most cases. This is because that NAMVH$_1^r$ chooses only one atom to represent the database points. In this case, there are only M kinds of hash code, and thus the larger M is desiderated to generate more candidates. While NAMVH$_{10}^r$ adopts the sum of 10 atoms selected from M atoms for encoding so that using even less atoms can produce nearly $\binom{M}{l}$ combinations far more than M. It therefore reveals that a small dictionary is capable of guaranteeing the remarkable search accuracy.

4.6.3 *Energy comparison*

Some classical supervised hashing methods [28] learn binary codes by minimizing $E = \|\frac{1}{K}\mathbf{B}^T\mathbf{B} - \mathbf{S}\|_F^2$, where $s_{ij} = \pm 1$ represents similar or dissimilar pairs. Thus, energy E can be used to empirically evaluate the goodness [46] of the learned binary codes produced by NAMVH$_1^b$. Due to the asymmetry of NAMVH$_1^b$, two types of energy are used for evaluation. One is the symmetric energy E, the other is the asymmetric one $\hat{E} = \|\frac{1}{K}\text{sign}(\mathbf{Z})^T\mathbf{B} - \mathbf{S}\|_F^2$, where $\mathbf{Z}$ represents the real-valued codes and $\mathbf{B} = \mathbf{CA}$ represents the binary codes as $l = 1$. The energy curves at each iteration are shown in Fig. 8. COSDISH uses the column sampling discrete optimization to generate binary codes, achieving relatively low energies with a violent fluctuation. Although SDH is a discrete method, it is learned in a classification framework, hence resulting in high energies for pairwise evaluation. Obviously, the energies of NAMVH$_1^b$ are more

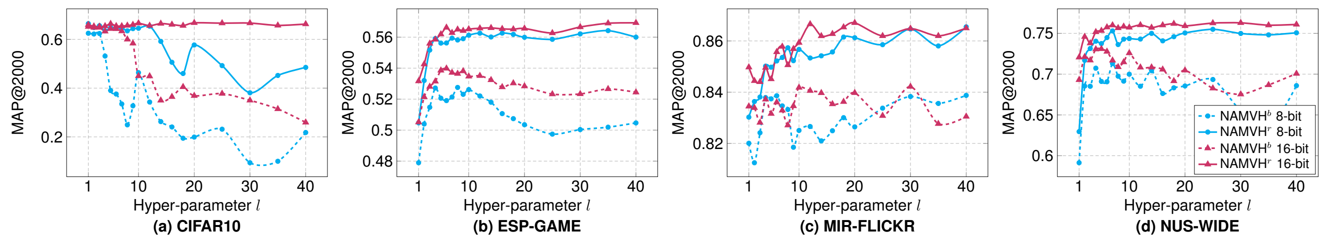


Fig. 7. The effect of l on MAP@2000 performance on (a) CIFAR10, (b) ESP-GAME, (c) MIR-FLICKR and (d) NUS-WIDE with 8-bit and 16-bit codes.

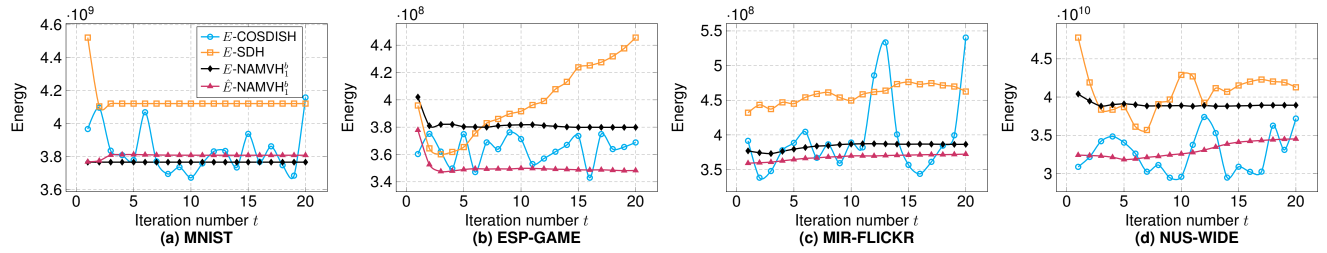


Fig. 8. Energy curves of different methods on (a) MNIST, (b) ESP-GAME, (c) MIR-FLICKR and (d) NUS-WIDE with 16-bit codes.

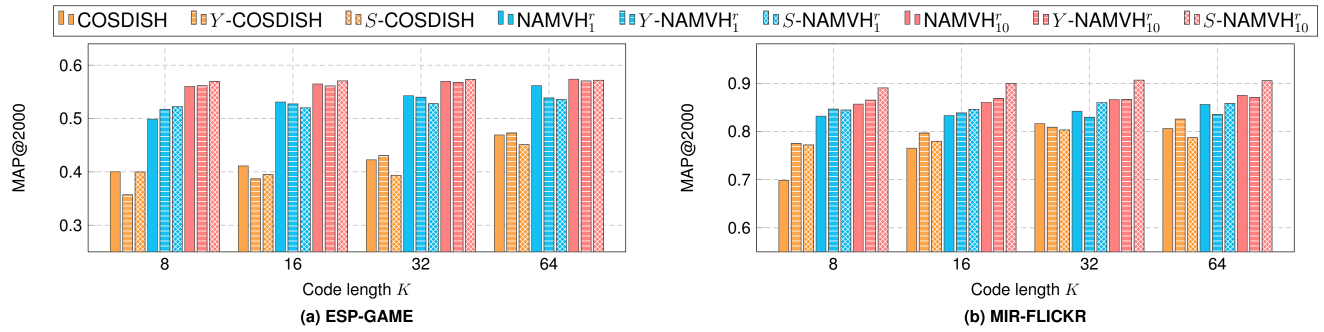


Fig. 9. The MAP@2000 performance of COSDISH and NAMVH with different similarity constructions on (a) ESP-GAME and (b) MIR-FLICKR.

stable than those of the others. Moreover, $\hat{E}$-NAMVH$_1^b$ has lower energies than E-NAMVH$_1^b$ and gets almost the lowest energies in multi-label datasets, while they are close to each other on MNIST, getting nearly the same performance that is consistent to the results in Fig. 3. To sum up, the construction of binary codes in NAMVH$_1^b$ is comparable to that of the discrete methods. Meanwhile, the asymmetric formulation may achieve better performance than the symmetric one, proving once again the effectiveness of the asymmetry.

4.6.4 Type of activation function

We investigate the influence of various activation functions (*i.e.*, ReLU [71], TanH, Sigmoid), and the results are shown in Table 3. In general, ReLU activation function achieves reliable results in each dataset. Specifically, for ESP-GAME, MIR-FLICKR, NUS-WIDE, all of the activation functions obtain comparable results, but TanH attains a little superiority to the others. Thus, we employ TanH as activation function in these three datasets. While for CIFAR10 and MNIST, the performance of ReLU is obviously better than the others, especially in the case of $l = 10$. Additionally, for larger datasets RCV and ILSVRC14, ReLU still prevails over the other activation functions. Conclusively, ReLU is employed as activation function in these four datasets.

4.6.5 Type of similarity construction

It is necessary to deeply analyze the effect of different types of similarity construction on hashing methods. As such, three similarity matrixes are considered as follows: (1) the original similarity matrix as their own definitions; (2) non-binary similarity construction, directly generated from label, *i.e.*, $\mathbf{Y}^T\mathbf{Y}$, denoted by Y; (3) binary similarity matrix $\mathbf{S}$ ($s_{ij} = 1$ for similar pairs; $s_{ij} = -1$ for dissimilar ones), denoted by S. We compare our approach with COSDISH, due to its relatively good search accuracy and high efficiency. Notably, $\mathbf{Y}^T\mathbf{Y}$ construction is the same to binary similarity definition, due to the one-hot label vectors on multi-class datasets. And large dataset with an enormous similarity matrix will lead to much low efficiency of storage and computation. Thus, the experiments are conducted on two relatively small multi-label datasets, and the results are shown in Fig. 9. Clearly, when the fairly same amount of affinity information is exploited, our approach outperforms COSDISH. Moreover, Y-COSDISH can attain higher MAP than COSDISH in most cases, implicitly revealing the effectiveness of $\mathbf{Y}^T\mathbf{Y}$ construction. On ESP-GAME, NAMVH$_1^r$ is superior to S-NAMVH$_1^r$ in most cases, while

TABLE 3
The MAP@2000 results with various activation functions.

Activation	ReLU		TanH		Sigmoid	
	$l = 1$	$l = 10$	$l = 1$	$l = 10$	$l = 1$	$l = 10$
ESP-GAME	0.5066	0.5412	0.5309	0.5648	0.5370	0.5586
MIR-FLICKR	0.8434	0.8567	0.8458	0.8667	0.8462	0.8646
NUS-WIDE	0.7214	0.7504	0.7324	0.7561	0.7173	0.7574
CIFAR10	0.6640	0.6890	0.6730	0.5490	0.6300	0.4970
MNIST	0.9890	0.9860	0.9700	0.9220	0.9480	0.8420
RCV	0.7509	0.7974	0.7548	0.7689	0.7229	0.7325
ILSVRC14	0.1233	0.2179	0.0737	0.0797	0.0694	0.0771

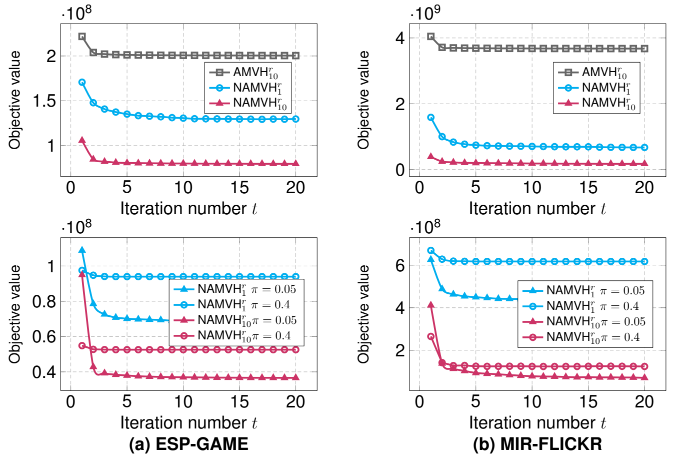


Fig. 10. Convergence curves of NAMVH$^r_1$ and NAMVH$^r_{10}$ on ESP-GAME and MIR-FLICKR with 16-bit codes, respectively. Top row: the original convergence curves; bottom row: the convergence cures of the incremental extension of database samples with two sample rates.

NAMVH$^r_{10}$ is nearly equal to S-NAMVH$^r_{10}$. Typically, S-NAMVH is better than NAMVH on MIR-FLICKR. In brief, different similarity constructions may lead to a slight fluctuation in performance for hashing methods. Nevertheless, S-NAMVH incurs cumbersome model training, as $\mathcal{O}(N^2)$ complexity restricts the application of large dataset. Hence, $\mathbf{Y}^T\mathbf{Y}$ construction does not always play a decisive role in search accuracy, but a crucial one in the training efficiency.

4.6.6 Convergence

We experimentally validate the convergence of Algorithm 2 and Algorithm 3. By taking ESP-GAME and MIR-FLICKR as examples, the original convergence curves of AMVH$^r_{10}$, NAMVH$^r_1$ and NAMVH$^r_{10}$ are illustrated in the first row of Fig. 10. Each point records the objective value of Eqn. (6) at each iteration. Obviously, the objective values monotonously decline as the iteration increases, and tend to be stable after a few iterations, hence demonstrating high convergence rate. Thus, Algorithm 2 can converge practically in less than 10 iterations. It is seen that the objective value of NAMVH$^r_{10}$ is significantly lower than that of NAMVH$^r_1$. This indicates the superiority of the multi-integer-valued embeddings. Typically, compared with AMVH$^r_{10}$, NAMVH reaches a much lower value, revealing again the effectiveness of the nonlinear transformation.

Referring to the second row of Fig. 10, each point records the objective value of Eqn. (19) at each iteration. Clearly, after the original training is done, the objective values still monotonously decline as the incremental extension of the dataset is performed. These results explicitly verify the effectiveness of the incremental extension of database samples. Meanwhile, the objective value with low sample rate is lower than that with high sample rate, demonstrating that the incremental extension algorithm is able to generate embeddings of massive database points.

4.6.7 Parameter sensitivity

We conduct the sensitivity analysis on parameters λ and μ, and the MAP performance of NAMVH$^r_{10}$ is illustrated in Fig. 11. By fixing μ, NAMVH$^r_{10}$ obtains stable performance with $\log_{10}(\lambda C/N)$ in $[-5, 1]$. Thus, it reveals that the performance is not much sensitive to λ, when λ is small. As

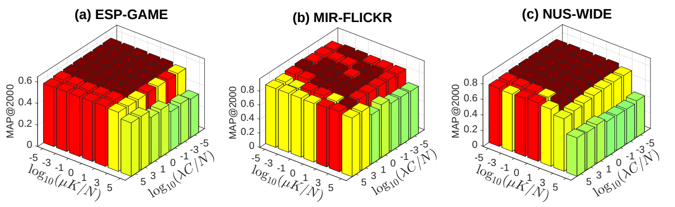


Fig. 11. Parameter sensitivity analysis on λ and μ in NAMVH$^r_{10}$ on (a) ESP-GAME, (b) MIR-FLICKR and (c) NUS-WIDE with 16-bit codes. Best viewed in colors.

for parameter μ, when $\log_{10}(\mu K/N)$ is larger than 3, the performance gets worse. Therefore, we empirically set λ and μ to $10\,N/C$ and $10\,N/K$, respectively.

5 Discussion

5.1 Notes on Asymmetric Hashing

We summarize the underlying reasons of why asymmetric hashing is better than symmetric one from the following aspects. First, Neyshabur *et al.* [36] theoretically proved that there exist representations of data where the code length of asymmetric binary hashing might be much shorter than that of symmetric one, which demonstrates that the asymmetric hashing could be much more powerful and allow better approximation of target similarity with shorter code length. Second, assuming two binary code matrixes $\mathbf{B}, \tilde{\mathbf{B}}$ and similarity matrix $\mathbf{S}$, the asymmetry can be measured by $\mathcal{L}_a = \|\mathbf{B}^T\tilde{\mathbf{B}} - \mathbf{S}\|_F^2$, while the symmetry is measured by $\mathcal{L}_s = \|\mathbf{B}^T\mathbf{B} - \mathbf{S}\|_F^2$. Intrinsically, $\mathcal{L}_s$ can be regarded as a constrained case of $\mathcal{L}_a$ with $\tilde{\mathbf{B}} = \mathbf{B}$. Thus, the solution space of $\mathcal{L}_a$ is larger than that of $\mathcal{L}_s$. Third, the accumulation errors of encoding database samples can be reduced in some asymmetric hashing methods, such as SDH, COSDISH and ours, since they directly generate codes of database without explicit hash functions. And thus the considerable performance can be achieved in these asymmetric hashing methods. Conclusively, the asymmetric hashing methods may have potential to prevail over the symmetric ones.

5.2 Limitations

Some potential limitations are listed as follows for the integrity of our approach. First, the applicability of the similarity decomposition is relatively limited, which is heavily depended on the optimization method and the specific similarity configuration. Actually, unless there is explicit computation on the $N \times N$ similarity matrix, the decomposition of similarity cannot reduce any computational complexity. Meanwhile, without pointwise “label” information, there are no general methods for the decomposition of similarity matrix on all dataset. Second, since our approach is beyond binary coding, the construction of the lookup table to find the nearest neighbors at a specific distance in $\mathcal{O}(1)$ cannot be realized directly, where the points falling within certain Hamming radius from the query codes are returned. Third, since the training is interweaved with multi-integer-valued embeddings learning in the unified alternative optimization method, the network is required to retrain iteratively. Thus, the training complexity is a little high.

6 Conclusion

In this paper, we proposed a novel asymmetric hashing approach, NAMVH, to alleviate the intrinsic binary limitation, achieving appealing search performance. NAMVH employs neural network to generate the real-valued embeddings of queries, and produces the multi-integer-valued embeddings of database by BSR. Due to the elaborate form of BSR, NAMVH remains high efficiency of query and storage of the non-binary embeddings, and has access to the incremental extension of database samples. Experiments on seven large scale datasets demonstrate the superiority and scalability of NAMVH to the conventional binary hashing methods.

In the future work, instead of approximate solutions, more effective discrete optimization methods deserve a further study. In addition, it is worthy to point out that the technique of the proposed multi-valued encoding scheme is not restricted to the supervised hashing, and it can also be applied to a wide range of hashing methods, such as unsupervised hashing, cross-modal hashing, and so on.

Acknowledgments

The authors would like to thank the reviewers for their valuable suggestions. This work was supported by the National Natural Science Foundation of China under Grants 91646207, 61620106003, 61573352 and 61671451.

References

[1] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, 2018.

[2] J. Wang, W. Liu, S. Kumar, and S. Chang, "Learning to hash for indexing big data - A survey," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2016.

[3] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, 2013.

[4] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1092–1104, 2012.

[5] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *NIPS*, 2009, pp. 1042–1050.

[6] M. Norouzi, D. J. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *NIPS*, 2012, pp. 1070–1078.

[7] H. Yang, K. Lin, and C. Chen, "Supervised learning of semantics-preserving hashing via deep convolutional neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PP, pp. 1–14, 2017.

[8] J. Wang, O. Kumar, and S. Chang, "Semi-supervised hashing for scalable image retrieval," in *CVPR*, 2010, pp. 3424–3431.

[9] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *CVPR*, 2008.

[10] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua, "Ldahash: Improved matching with smaller descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 1, pp. 66–78, 2012.

[11] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification," *IEEE Trans. Image Processing*, vol. 24, no. 12, pp. 4766–4779, 2015.

[12] F. Zhu, X. Kong, L. Zheng, H. Fu, and Q. Tian, "Part-based deep hashing for large-scale person re-identification," *IEEE Trans. Image Processing*, vol. 26, no. 10, pp. 4806–4817, 2017.

[13] J. Chen, Y. Wang, J. Qin, L. Liu, and L. Shao, "Fast person re-identification via cross-camera semantic binary transformation," in *CVPR*, 2017, pp. 5330–5339.

[14] Y. Gong, M. Pawlowski, F. Yang, L. Brandy, L. D. Bourdev, and R. Fergus, "Web scale photo hash clustering on a single machine," in *CVPR*, 2015, pp. 19–27.

[15] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *ICML*, 2015, pp. 2285–2294.

[16] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *SCG*, 2004, pp. 253–262.

[17] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik, "Learning binary codes for high-dimensional data using bilinear projections," in *CVPR*, 2013, pp. 484–491.

[18] W. Li, S. Wang, and W. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *IJCAI*, 2016, pp. 1711–1717.

[19] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. T. Shen, "Learning binary codes for maximum inner product search," in *ICCV*, 2015, pp. 4148–4156.

[20] W. Liu, C. Mu, S. Kumar, and S. Chang, "Discrete graph hashing," in *NIPS*, 2014, pp. 3419–3427.

[21] K. Ding, B. Fan, C. Huo, S. Xiang, and C. Pan, "Cross-modal hashing via rank-order preserving," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 571–585, 2017.

[22] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB*, 1999, pp. 518–529.

[23] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *NIPS*, 2009, pp. 1509–1517.

[24] W. Kong and W. Li, "Isotropic hashing," in *NIPS*, 2012, pp. 1655–1663.

[25] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *NIPS*, 2008, pp. 1753–1760.

[26] Q. Jiang and W. Li, "Scalable graph hashing with feature transformation," in *IJCAI*, 2015, pp. 2248–2254.

[27] J. Zhang, Y. Peng, and J. Zhang, "SSDH: semi-supervised deep hashing for large scale image retrieval," *CoRR*, vol. abs/1607.08477, pp. 1–13, 2016.

[28] W. Kang, W. Li, and Z. Zhou, "Column sampling based discrete supervised hashing," in *AAAI*, 2016, pp. 1230–1236.

[29] G. Lin, C. Shen, and A. van den Hengel, "Supervised hashing using graph cuts and boosted decision trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2317–2331, 2015.

[30] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang, "Supervised hashing with kernels," in *CVPR*, 2012, pp. 2074–2081.

[31] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *ICML*, 2011, pp. 353–360.

[32] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *CVPR*, 2015, pp. 37–45.

[33] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *CVPR*, 2015, pp. 1556–1564.

[34] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *CVPR*, 2015, pp. 3270–3278.

[35] W. Dong, M. Charikar, and K. Li, "Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces," in *SIGIR*, 2008, pp. 123–130.

[36] B. Neyshabur, N. Srebro, R. Salakhutdinov, Y. Makarychev, and P. Yadollahpour, "The power of asymmetry in binary hashing," in *NIPS*, 2013, pp. 2823–2831.

[37] A. Gordo, F. Perronnin, Y. Gong, and S. Lazebnik, "Asymmetric distances for binary embeddings," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 33–47, 2014.

[38] T. Do, A. Doan, and N. Cheung, "Learning to hash with binary deep neural network," in *ECCV*, 2016, pp. 219–234.

[39] Z. Chen, J. Lu, J. Feng, and J. Zhou, "Nonlinear discrete hashing," *IEEE Trans. Multimedia*, vol. 19, no. 1, pp. 123–135, 2017.

[40] C. Da, S. Xu, K. Ding, G. Meng, S. Xiang, and C. Pan, "AMVH: asymmetric multi-valued hashing," in *CVPR*, 2017, pp. 898–906.

[41] A. Shrivastava and P. Li, "Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS)," in *NIPS*, 2014, pp. 2321–2329.

[42] F. Shen, X. Gao, L. Liu, Y. Yang, and H. T. Shen, "Deep asymmetric pairwise hashing," in *MM*, 2017, pp. 1522–1530.

[43] Q. Jiang and W. Li, "Asymmetric deep supervised hashing," *CoRR*, vol. abs/1707.08325, pp. 1–8, 2017.

[44] G. Lin, C. Shen, D. Suter, and A. van den Hengel, "A general two-step approach to learning-based hashing," in *ICCV*, 2013, pp. 2552–2559.

[45] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *AAAI*, 2014, pp. 2156–2162.

[46] K. Ding, C. Huo, B. Fan, S. Xiang, and C. Pan, "In defense of locality-sensitive hashing," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 29, no. 1, pp. 87–103, 2018.
[47] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1106–1114.
[49] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, 2017.
[50] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.
[51] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *CVPR*, 2016, pp. 2064–2072.
[52] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *AAAI*, 2016, pp. 2415–2421.
[53] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *ICCV*, 2017, pp. 5609–5618.
[54] Q. Li, Z. Sun, R. He, and T. Tan, "Deep supervised discrete hashing," in *NIPS*, 2017, pp. 2479–2488.
[55] X. Shi, F. Xing, K. Xu, M. Sapkota, and L. Yang, "Asymmetric discrete graph hashing," in *AAAI*, 2017, pp. 2541–2547.
[56] M. Norouzi and D. J. Fleet, "Cartesian k-means," in *CVPR*, 2013, pp. 3017–3024.
[57] X. Zhang, L. Zhang, and H. Shum, "Qsrank: Query-sensitive hash code ranking for efficient ϵ-neighbor search," in *CVPR*, 2012, pp. 2058–2065.
[58] L. Zhang, Y. Zhang, J. Tang, K. Lu, and Q. Tian, "Binary code ranking with weighted hamming distance," in *CVPR*, 2013, pp. 1586–1593.
[59] D. Xu, I. W. Tsang, and Y. Zhang, "Online product quantization," *IEEE Trans. Knowl. Data Eng.*, vol. Preprint, pp. 1–14, 2018.
[60] F. Çakir, K. He, S. A. Bargal, and S. Sclaroff, "Mihash: Online hashing with mutual information," in *ICCV*, 2017, pp. 437–445.
[61] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *CHI*, 2004, pp. 319–326.
[62] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *MIR*, 2008, pp. 39–43.
[63] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "NUS-WIDE: a real-world web image database from national university of singapore," in *CIVR*, 2009, pp. 1–9.
[64] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, 2004.
[65] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.
[66] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.
[67] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, 2008.
[68] P. Li, T. Hastie, and K. W. Church, "Very sparse random projections," in *SIGKDD*, 2006, pp. 287–296.
[69] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, pp. 448–456.
[70] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.
[71] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011, pp. 315–323.

Cheng Da received the B.S. degree in computer science from Sichuan University, Chengdu, China, in 2014. He is currently working toward the Ph.D. degree in National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include computer vision and machine learning.

Gaofeng MENG received the B.S. degree in applied mathematics from Northwestern Polytechnical University in 2002, and the M.S. degree in applied mathematics from Tianjin University in 2005, and the Ph.D degree in control science and engineering from Xi'an Jiaotong University in 2009. He is now an associate professor of National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests include document image processing and computer vision. He serves as an Associate Editor for Neurocomputing since 2014. He is now a senior member of IEEE.

Shiming Xiang received the B.S. degree in mathematics from Chongqing Normal University, Chongqing, China, in 1993, the M.S. degree from Chongqing University, Chongqing, China, in 1996, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2004. From 1996 to 2001, he was a Lecturer with the Huazhong University of Science and Technology, Wuhan, China. He was a Postdoctoral Researcher with the Department of Automation, Tsinghua University, Beijing, China, until 2006. He is currently a Professor with the Institute of Automation, Chinese Academy of Sciences. His current research interests include pattern recognition and machine learning.

Kun Ding received the B.S. degree in automation from the Tianjin University of Science and Technology, Tianjin, China, in 2011, the M.S. and Ph.D. degrees in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2014, and 2017, respectively. His current research interests include machine learning, computer vision, and remote sensing.

Shibiao Xu received the B.S. degree in Information Engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2009, and the Ph.D. degree in Computer Science from Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2014. He is currently an associate professor in the Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include image based three-dimensional scene reconstruction and scene semantic understanding.

Qing Yang received the Ph.D. degree in computer science from the Institute of Automation, Chinese Academy of Sciences, Beijing, China. From 1999 to 2003, he worked as a Computer Scientist in the Computing Sciences of Lawrence Berkeley National Laboratory, Berkeley, CA. Since 2004, he has been a Full Professor with the Institute of Automation, Chinese Academy of Sciences. He conducts research on computer vision, web search engine and recommender systems.

Chunhong Pan received the B.S. degree in automatic control from Tsinghua University, Beijing, China, in 1987, the M.S. degree from Shanghai Institute of Optics and Fine Mechanics, Chinese Academy of Sciences, Shanghai, China, in 1990, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2000. He is currently a Professor with the Institute of Automation, Chinese Academy of Sciences. His current research interests include computer vision, image processing, computer graphics, and remote sensing.