Voronoi Diagram Encoded Hashing

Yang Xu and Kai Ming Ting (✉)

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
xuyang@lamda.nju.edu.cn, tingkm@nju.edu.cn

Abstract. The goal of learning to hash (L2H) is to derive data-dependent hash functions from a given data distribution in order to map data from the input space to a binary coding space. Despite the success of L2H, two observations have cast doubt on the source of the power of L2H, i.e., learning. First, a recent study shows that even using a version of locality sensitive hashing functions without learning achieves binary representations that have comparable accuracy as those of L2H, but with less time cost. Second, existing L2H methods are constrained to three types of hash functions: thresholding, hyperspheres, and hyperplanes only. In this paper, we unveil the potential of Voronoi diagrams in hashing. Voronoi diagram is a suitable candidate because of its three properties. This discovery has led us to propose a simple and efficient no-learning binary hashing method, called Voronoi Diagram Encoded Hashing (VDeH), which constructs a set of hash functions through a data-dependent similarity measure and produces independent binary bits through encoded hashing. We demonstrate through experiments on several benchmark datasets that VDeH achieves superior performance and lower computational cost compared to existing state-of-the-art methods under the same bit length. Our code is available at: https://github.com/Phantom-Det/VDeH/tree/main.

Keywords: Learning to hash · Binary representation · Voronoi diagram

1 Introduction

For decades, hashing techniques have been one of the most effective tools commonly used to compress data for fast access, analysis, and learning [5, 6, 36]. Hashing techniques are popular for their simplicity and offer significant advantages in data processing and security [2, 16]. Their primary strength lies in the ability to efficiently map arbitrary-sized input data to fixed-size embeddings, known as hash values. This process is deterministic, meaning the same input always yields the same output, ensuring data consistency. These attributes facilitate a wide range of applications, including data integrity verification, data indexing, blockchain technology, and distributed systems [3, 25, 26, 29].

Learning to hash (L2H) [22, 31, 35], a prominent subfield within hashing techniques, aims to map data from the input space to a binary coding space, in which the Hamming distance well approximates the distance (e.g., ℓ_p norm) in input

space. Then, efficient retrieval and learning can be performed in the binary coding space. The general problem in L2H is to learn long binary codes that approximate the input distance[1].

Given input data of size N with dimensionality d, the core task of L2H is to construct binary hash functions that map the data into L-bit binary codes. Broadly, existing L2H methods can be categorized into three classes: thresholding [9, 10, 32, 34, 35], hyperspheres [15], and hyperplanes [4, 17, 22, 37]. Thresholding-based methods initiate the process by transforming distance or similarity relationships from the input space into a new matrix of size $N \times L$ through metric learning [32, 35] or PCA-based methods [9, 10, 34]. Then, matrix values are binarized based on a given threshold. Hypersphere and hyperplane-based methods construct hash functions by partitioning the input space. Points falling within the same partition are mapped to identical binary code values. Due to randomness in the partitioning process, these methods typically improve binary codes performance by learning the partitioning strategy to achieve a uniform distribution of points across distinct partitions [15, 22].

Despite the success of L2H, the following two observations cast doubt on the source of the power of methods, i.e., learning:

1. Existing L2H methods all claim that their learning process (in optimizing some defined objective) plays a crucial role in obtaining effective binary codes. Yet, a recent study [4] shows that even using brLSH, a version of locality-sensitive hashing [5] without learning, achieves binary codes that have comparable performance as those of L2H methods, but with less computation cost.
2. They are constrained to three types of hash functions. Learning is required in order to ensure that these functions satisfy three key properties for effective binary codes [10, 15, 17, 22, 32, 34], i.e., (a) *full space coverage*: each hash function covers the entire space; (b) *entropy maximization*: each hash function covers approximate the same number of points from the input data, such that the total set of hash functions maximizes information entropy; and (c) *bit independence*: all hash functions have mutually independent hash bits. Their importance will be discussed in detail in Section **3**.

In this work, we provide the insight that Voronoi diagram is a suitable alternative to L2H, attributed to its three properties: first, a Voronoi diagram naturally covers the entire input space; second, for a given dataset, every Voronoi cell

[1]Note that another related area called *deep hashing* [24, 28] conducts a similar task but with a totally different goal: generating short compact binary codes, where proximate or identical binary codes represent semantic (categorical) similarity between instances, while distant binary codes represent semantic dissimilarity. However, deep hashing is specifically designed for datasets with explicit categorical information, notably image data [11, 14, 33], and is incompatible with datasets lacking this information, including extensive text and tabular data. Besides, these short binary codes do not adequately approximate the input distance. Existing studies demonstrate that achieving an effective approximation of the input distance through binary codes requires a code length of $\mathcal{O}(d)$, where d is the input dimensionality [9, 22, 34, 35].

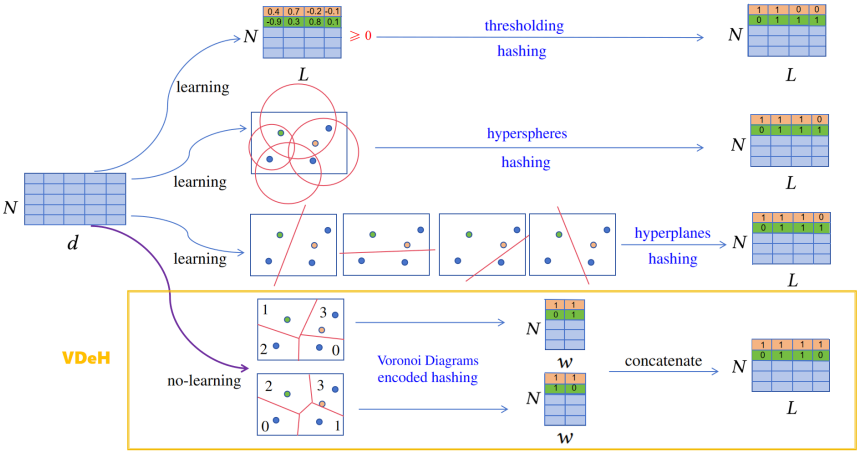


Fig. 1: An illustrative comparison of VDeH using Voronoi diagram and three existing types of hash functions (thresholding, hyperspheres and hyperplanes). The parameters $N = 5$, $d = 5$, $L = 4$, and $w = 2$ are used in this example.

in a Voronoi diagram covers approximately the same number of points [7]; third, it is feasible to transform a Voronoi diagram into a set of hash functions, which generates mutually independent bits (hash values). Unlike existing L2H methods, all these properties are obtained in Voronoi diagrams without any extra learning process.

Voronoi diagrams facilitate the generation of *data-dependent* hash functions, as these hash functions are derived by samples from a given dataset and are dependent on the distribution of the data. Specifically, they yield small Voronoi cells in areas of high densities, while large Voronoi cells are produced in areas of low densities [7, 30]. This discovery has led us to propose a simple and effective no-learning approach called Voronoi Diagram Encoded Hashing (VDeH), which is the first data-dependent binary hashing scheme based on Voronoi diagram already used in a kernel. Figure 1 shows an illustrative comparison of VDeH using Voronoi diagram and three existing types of hash functions.

Our main contributions are summarized as follows:

- Providing the insight that Voronoi diagram can be leveraged to realize a family of data-dependent hash functions, without learning.
- Creating a new definition of hashing scheme derived directly from a kernel based on Voronoi diagrams.
- Proposing VDeH, which creates the first kernel-based implementation of a data-dependent binary hashing scheme, and formulating a binary distance function intrinsic to VDeH. We have theoretically proved that VDeH is capable of generating mutually independent binary bits.
- Conducting comprehensive empirical comparisons and analyses between VDeH and existing state-of-the-art L2H methods to demonstrate the effectiveness and efficiency of VDeH.

2 Background and Related Work

With the emergence of large-scale data, numerous learning to hash (L2H) methods have been developed [12, 22, 31]. The generic binary hashing problem is the following. Given N data points $\mathcal{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{d \times N}$, generate L hash functions to map a data point $\mathbf{x}$ into a L-bit binary hash code $\mathcal{B}(\mathbf{x}) = [h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_L(\mathbf{x})]$ where $h_l(\mathbf{x}) \in \{0, 1\}$ is the l-th hash function.

For the linear binary projection-based hashing [9, 22, 35]:

$$h_l(\mathbf{x}) = sgn(F(\mathbf{R}_l\mathbf{x} + t_l)),$$

where $\mathbf{R}_l \in \mathbb{R}^{L \times d}$ is the projection matrix, $sgn(\cdot)$ is a binary map, and t_l is the intercept. Different hashing methods aim at finding different F, $\mathbf{R}_l$ and t_l with respect to different objective functions.

Existing L2H methods seek to ensure that the Hamming distance of generated binary codes closely approximates the input distance by imposing specific constraints on these codes. For example, given $b_i \in \{0, 1\}^L$, Spectral Hashing (SH) [32] aims to minimize the average Hamming distance between similar neighbors, represented by $\sum_{ij} W_{ij}||b_i - b_j||^2$, where W_{ij} is the similarity between points i and j as measured by Gaussian kernel, and $||b_i - b_j||^2$ is the Hamming distance between binary codes b_i and b_j. SH uses two constraints: (i) $\sum_i b_i = \frac{1}{2}$ to ensure entropy maximization, and (ii) $\frac{1}{n}\sum_i b_i b_i^T = I$ to guarantee bit independence. In addition, Spherical Hashing (SpH) [15] generates hash functions by partitioning the input space using hyperspheres. SpH achieves bit independence by ensuring that each hashing function has an equal probability of output 0 and 1. To achieve entropy maximization, it requires each hypersphere to contain an approximately equal number of points. Since hyperspheres do not cover the entire input space, SpH increases the radius of the hyperspheres to cover at least all training data, thus ensuring full space coverage. It is worth noting that most existing methods explicitly or implicitly ensure the three key properties: full space coverage, entropy maximization, and bit independence by constructing similar constraints. Methods operated on spatial partitioning necessitate that the hash functions provide complete coverage of the input space [4, 15, 17, 22, 37]. Besides, existing methods ensure entropy maximization by explicitly constraining each bit to have a half probability of being 0 or 1, while maintaining the independence of each bit's output [9, 10, 15, 17, 22, 32, 34, 35, 37]. The detailed explanations of how these methods guarantee these properties and their categorizations can be found in comprehensive survey papers [24, 31].

The three properties have been demonstrated to be crucial for effective binary codes in existing methods. However, these methods invariably require a learning process to gain these properties, as their employed hash functions, including those based on thresholding, hyperspheres, and hyperplanes, do not naturally possess these properties.

Table 1: Key notations used.

Notation	Definition
$\mathcal{X}$	Input dataset with N points in $\mathbb{R}^d$
$\mathcal{D}$	A set of ψ points randomly sampled from $\mathcal{X}$ to generate a Voronoi diagram
$\mathcal{B}$	L-bit binary codes correspond to all points in $\mathcal{X}$
H	A set of hash functions derived from a Voronoi Daigram
κ	kernel $\kappa(x, y)$, where $x, y \in \mathbb{R}^d$
$\mathbf{S}$	A similarity function defined in $\mathbb{R}^d$
w	The number of bits generated by encoded hashing over a Voronoi diagram
$\mathcal{H}$	A hashing scheme consisting of a family of hash functions.
$P_{\mathcal{H}}$	Distribution over a family of hash functions $\mathcal{H}$

3 Insight: Voronoi Diagram is a Suitable Candidate for Binary Hashing

The key notations used in this paper are provided in Table 1.

Given a point $\mathbf{x} \in \mathbb{R}^d$, a hashing scheme of a family H of hash functions $h \in H$, where $h(\mathbf{x}) \in \{0, 1\}$, must have the following three properties: full space coverage, entropy maximization, and bit independence.

Full space coverage. All $h \in H$ cover the entire $\mathbb{R}^d$, i.e., $\forall \mathbf{x} \in \mathbb{R}^d, \exists h \in H$ s.t. $h(\mathbf{x}) = 1$. The failure to achieve this coverage can severely impair retrieval effectiveness. Two possible current treatments of this shortcoming are unsatisfactory: (a) All points outside the covered regions have no binary code representations, rendering them irretrievable. (b) Using a common binary mapping to all points outside the covered regions risks conflating vastly distant points in the input space with a same binary code. This issue is particularly pronounced when adopting hypersphere-based hashing strategies [15]. Therefore, ensuring that hash functions cover the entire space is a critical aspect of maintaining high retrieval accuracy in binary hashing-based information retrieval systems. This coverage guarantees that every point $\mathbf{x} \in \mathbb{R}^d$ can be effectively indexed and retrieved, thereby maximizing the utility of binary hashing in real-world applications.

Entropy maximization. For a given dataset $\mathcal{X} \subset \mathbb{R}^d$ with N points, every hash function $h \in H$ shall cover approximately the same number of points $x \in \mathcal{X}$, i.e., uniformly distributed over all hash functions:

$$\forall h \in H, \left| |h(\mathcal{X})| - \tfrac{N}{|H|} \right| \leqslant \varepsilon,$$

where $|H|$ denotes the total number of hash functions in H and ε is a predefined non-negative small number indicating the tolerance level for the approximation. This is to ensure that there are no under-utilized hash functions. This uniform coverage can be understood as maximizing the information entropy of the resulting binary codes [21], where high information entropy is indicative of a rich, diverse representation of the dataset. It avoids the scenario where a large proportion of the dataset is lumped in a few binary codes only.

Bit independence. All hash functions in H are mutually independent. This means that for any pair of distinct hash functions $h_i \neq h_j \in H$, and for any point $\mathbf{x} \in \mathbb{R}^d$, the outputs $h_i(\mathbf{x})$ and $h_j(\mathbf{x})$ are statistically independent. It has been proven to be indispensable for optimizing performance [13, 15, 18, 32]. This independence ensures that each hash function contributes uniquely to the hashing process, thereby eliminating potential redundancy in the generated binary bits. When hash functions are not mutually independent, the resulting binary representations suffer from information redundancy, which in turn, dilutes the effectiveness of the binary hashing scheme, leading to poor retrieval efficiency. Such redundancy also inflates the storage requirements unnecessarily.

It is interesting to note that none of the existing hash functions (i.e., thresholding, hyperspheres and hyperplanes) have the three properties naturally. That is the reason why learning has been employed to ensure that the final hash functions satisfy the three properties.

Here we have the insight that Voronoi diagrams is a suitable candidate for hash functions because they have the following properties, without learning:

1. A Voronoi diagram naturally covers the entire input space.
2. For a given dataset, every Voronoi cell in a Voronoi diagram covers approximately the same number of points. This occurs when a set of points, that represents the data distribution, is used to construct the Vonoroi diagram. And it can be easily achieved through a random Voronoi partition created by a set of points drawn independently and randomly from the dataset [7].
3. It is feasible to transform a Voronoi diagram into a set of mutually independent hash functions (see the analysis in Section **4.3**).

This insight has led us to propose a no-learning data-dependent hashing scheme based on Voronoi diagrams, described in the next section.

4 Proposed Approach

Here we give the formal definition of our proposed binary hashing method called Voronoi Diagram Encoded Hashing (VDeH), which is the first binary hashing scheme based on a data-dependent similarity using Voronoi diagram partitioning.

4.1 A New Definition of Hashing

A kernel based on Voronoi diagrams called Isolation Kernel [30] is defined as:

$$\kappa(\mathbf{x}, \mathbf{y}|\mathcal{H}(\mathcal{X})) = \mathbb{E}_{H\sim\mathcal{H}(\mathcal{X})}[\mathbb{1}(\mathbf{x}, \mathbf{y} \in \theta \mid \theta \in H)],$$

where each Voronoi diagram H has cells θ, and $\mathbb{1}(\cdot)$ is an indicator function.

By re-writing each cell θ as a hash function h, it can be re-expressed as:

$$\kappa(\mathbf{x}, \mathbf{y}|\mathcal{H}(\mathcal{X})) = \mathbb{E}_{H\sim\mathcal{H}(\mathcal{X})}[\mathbb{1}(h(\mathbf{x}) = h(\mathbf{y}) = 1 \mid h \in H)].$$

The above revelation, together with the three properties of Voronoi diagram (stated in Section 3), prompt us to propose a new definition of hashing. Given a dataset $\mathcal{X} \subset \mathbb{R}^{d\times N}$, the proposed VDeH scheme is defined as follows:

Definition 1. *The VDeH scheme is a family $\mathcal{H}(\mathcal{X})$ of hash functions created by Voronoi diagrams associated with a distribution $P_{\mathcal{H}}$ over $\mathcal{H}(\mathcal{X})$ generated from a dataset $\mathcal{X}$ such that it satisfies*

$$Pr_{h \in H \in \mathcal{H}(\mathcal{X})}[h(\mathbf{x}) = h(\mathbf{y}) = 1] = \kappa(\mathbf{x}, \mathbf{y} | \mathcal{H}(\mathcal{X})), \tag{1}$$

where $\kappa(\mathbf{x}, \mathbf{y}|\mathcal{H}(\mathcal{X}))$ is derived from Voronoi diagrams generated from $\mathcal{X}$; and H is a set of all hash functions h derived from a Voronoi diagram.

This definition makes the implementation of VDeH extremely simple because the hashing functions can be obtained with a simple additional encoding from the Voronoi diagrams already used in Isolation Kernel. Unlike existing hashing methods, VDeH needs no special design or learning of hash functions to gain the three properties mentioned in Section 3.

4.2 Implementation Details of VDeH

Given a subset $\mathcal{D} = \{\mathbf{s}_1, \ldots, \mathbf{s}_\psi\} \subset \mathcal{X}$ with ψ randomly selected points. A Voronoi diagram H, created by $\mathcal{D}$, partitions the $\mathbb{R}^d$ space into ψ Voronoi cells, where $\mathbf{s}_i$ is at the center of Voronoi cell i. Let a set of hash functions created by $\mathcal{D}$ be $H = \{h_1, h_2, \ldots, h_\psi\}$. For any $\mathbf{x} \in \mathbb{R}^d$, $h_i(\mathbf{x})$ is defined as:

$$h_i(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad dist(\mathbf{x}, \mathbf{s}_i) > dist(\mathbf{x}, \mathcal{D}) \\ 1 & \text{if} \quad dist(\mathbf{x}, \mathbf{s}_i) = dist(\mathbf{x}, \mathcal{D}), \end{cases} \tag{2}$$

where $dist(\mathbf{x}, \mathbf{s}_i) = \| \mathbf{x} - \mathbf{s}_i \|$ denotes the ℓ_2 distance between $\mathbf{x}$ and $\mathbf{s}_i$; and $dist(\mathbf{x}, \mathcal{D}) = \min_{\mathbf{y} \in \mathcal{D}/\{\mathbf{x}\}} \| \mathbf{x} - \mathbf{y} \|$. When a point is located on a boundary, it is randomly assigned to any one of the cells sharing that boundary.

Given a fixed length of binary bits, existing studies have established that ensuring independence among generated binary bits is key to effective binary hashing. However, directly converting Voronoi cells into hash functions results in dependencies between them, due to the fact:

$$\forall h \in H, \forall \mathbf{x} \in \mathbb{R}^d, \sum_{i \in [1, \psi]} \mathbb{1}[h_i(\mathbf{x}) = 1] = 1. \tag{3}$$

This dependency arises because if $\mathbf{x}$ belongs to the i-th Voronoi cell, indicating $h_i(\mathbf{x}) = 1$, then $\mathbf{x}$ can not fall into other regions, hence $h(\mathbf{x}) = 0$ for those. As a result, any $h(\mathbf{x})$ is influenced by other hash functions.

To address this issue, we have adopted a simple encoded hashing mechanism to generate mutually independent binary bits. Let $\psi = 2^w$, and as 2^w Voronoi cells can be encoded with a binary code of w mutually independent bits, a L-bit code of the VDeH scheme represents $L/w \times 2^w$ hash functions (the proof is presented in the Section **4.3**).

The process for generating a binary code corresponding to a point $\mathbf{x} \in \mathbb{R}^d$ via VDeH is outlined as follows:

1. Given a dataset $\mathcal{X}$, we randomly sample ψ points to form a set $\mathcal{D}$, and generate the set of Voronoi diagram hash functions $H = \{h_1, h_2, \ldots, h_\psi\}$. According to Eq.(3), there is one and only one $h_i(\mathbf{x}) = 1$ for any point $\mathbf{x}$, and $\forall u \neq i, h_u(\mathbf{x}) = 0$.
2. The above 'raw' ψ-bit vector $[h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_\psi(\mathbf{x})]$ can then be encoded as w-bit vector $b(\mathbf{x}) = [e_1(\mathbf{x}), e_2(\mathbf{x}), \ldots, e_w(\mathbf{x})]$, where $\psi \leqslant 2^w$, through the following encoded hashing function:

$$e_j(\mathbf{x}) = \begin{cases} 0 & \text{if } \lfloor \frac{\arg_{i\in[1,\psi]} h_i(\mathbf{x})=1}{2^{j-1}} \rfloor \mod 2 = 0 \\ 1 & \text{if } \lfloor \frac{\arg_{i\in[1,\psi]} h_i(\mathbf{x})=1}{2^{j-1}} \rfloor \mod 2 \neq 0, \end{cases} \tag{4}$$

   where $j = 1, 2, \ldots, w$.
3. After repeating the above two steps L/w times, we concatenate all w-bit vectors $b(\mathbf{x})$ to formulate a L-bit code $\mathcal{B}(\mathbf{x}) = [b^1(\mathbf{x}), \ldots, b^{L/w}(\mathbf{x})]$ for any point $\mathbf{x}$.

It is interesting to note that this simple encoding is not applicable to existing L2H methods where a L-bit code could represent L hash functions only. In short, the VDeH scheme represents $(\frac{\psi}{w} - 1)L$ more hash functions than existing L2H methods when both have codes of the same number of L bits. This is a key to VDeH's better performance over existing L2H methods. Also note that the Voronoi diagram does not necessarily have to have 2^w Voronoi cells. It is simply a convenience for a binary encoded hashing with mutually independent bits.

The time complexity of VDeH. VDeH has a linear time complexity with respect to the size of the dataset, denoted as N, and the dimensionality of the data, denoted as d. A detailed analysis of the time complexity of VDeH is presented as follows: to convert a dataset $\mathcal{X}$ of N points of d dimensions, VDeH first finds the nearest neighbor of each point in $\mathcal{X}$ from the set $\mathcal{D}$ of ψ points (or Voronoi cells), resulting in a complexity of $\mathcal{O}(\psi N d)$. This process is repeated L/w times, leading to a total complexity of $\mathcal{O}(\frac{\psi L N d}{w})$. VDeH then uses the encoded hash functions h to generate L-bit binary codes for all N points, resulting in a complexity of $\mathcal{O}(LN)$. Thus, the overall time complexity for VDeH's 'training' process is $\mathcal{O}(\frac{\psi L N d}{w})$, which is linear to the dataset size N and the input dimensionality d.

4.3 Distance Measure for VDeH Codes

Hamming distance, renowned for its simplicity and efficiency, is predominantly employed in the comparison of binary codes due to its methodological advantage of merely counting the differing bits between two codes, thereby circumventing the necessity for intricate arithmetic operations such as multiplication and square root calculations. This stands in stark contrast to the computationally more demanding Euclidean distance.

For VDeH, the measurement of similarity between any two points is conducted by calculating the probability that both points fall into the same regions across all generated Voronoi cells. To fully utilize the three properties of the

Voronoi diagram hashing functions, we propose the following distance metric, VDeH distance ($\mathbf{d}_V(\mathcal{B}_i, \mathcal{B}_j)$), between two binary codes $\mathcal{B}_i$ and $\mathcal{B}_j$ generated by VDeH:

$$\mathbf{d}_V(\mathcal{B}_i, \mathcal{B}_j) = \frac{w}{L} \sum_{k=1}^{L/w} \mathbb{1}(b_i^k \neq b_j^k), \tag{5}$$

where $\mathcal{B}_i = [b_i^1, \ldots, b_i^{L/w}]$ and $\mathcal{B}_j = [b_j^1, \ldots, b_j^{L/w}]$. Then, for any given query point $\mathbf{q}$, after computing its binary code $\mathcal{B}(\mathbf{q})$ through VDeH and its distance to every point in the dataset can be calculated using Eq.(5), the retrieval process returns a point in the dataset having the shortest distance to the query point $\mathbf{q}$.

4.4 Independence among VDeH Bits

The importance of independence among hash bits are mentioned in existing studies [32, 13, 18]. It can be defined as follows:

Definition 2. *(**Independence among hash bits.**) Given a family of hash functions $\{h_1, h_2, \ldots, h_L\}$, for any $\mathbf{x} \in \mathbb{R}^d$, $h_1(\mathbf{x})$, $h_2(\mathbf{x})$, $\ldots$, $h_L(\mathbf{x})$ are said to be mutually independent for $\forall \mathbf{x}$ if for any set of values $v_1, v_2, \ldots, v_L \in \{0, 1\}$, it holds that:*

$$Pr(h_1(\mathbf{x}) = v_1, h_2(\mathbf{x}) = v_2, \ldots, h_L(\mathbf{x}) = v_L) =$$
$$Pr(h_1(\mathbf{x}) = v_1) \cdot Pr(h_2(\mathbf{x}) = v_2) \cdot \cdots \cdot Pr(h_L(\mathbf{x}) = v_L).$$

This means that the value of one hash bit does not influence the value of another hash bit. Although directly converting regions generated by Voronoi diagrams into hash functions results in dependencies between them based on the Eq.(3), we prove that the encoded hashing enables the achievement of such independence among hash bits. VDeH generates mutually independent binary bits through encoded hashing, as demonstrated by the following theorem.

Lemma 1 *Let $\mathcal{D} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_\psi\} \sim G^\psi$ be a dataset, where every $\mathbf{s}_i$ is i.i.d drawn from an unknown probability distribution G on the input space Ω. Let $\mathcal{D}$ forms its Voronoi cells $V_i \subset \Omega$, the probability that $\mathbf{s}_i$ is the nearest neighbor of any $\mathbf{x} \in \Omega$ in $\mathcal{D}$ is given as: $Pr(\mathbf{x} \in V_i) = 1/\psi$, for every $i = 1, 2, \ldots, \psi$.*

Proof. Let $R(\mathbf{x}) = \{\mathbf{y} \in \Omega \mid dist(\mathbf{x}, \mathbf{y}) \leqslant r\}$ be r-neighborhood region of $\mathbf{x}$, and $\Delta R(\mathbf{x}) = \{\mathbf{y} \in \Omega \mid r \leqslant dist(\mathbf{x}, \mathbf{y}) \leqslant r + \Delta r\}$ be Δr incremental r-neighborhood region of x. Let ρ_G be the probability density of G, for $\Delta r > 0$, $f = \int_{R(\mathbf{x})} \rho_G(\mathbf{y}) d\mathbf{y}$ and $\Delta f = \int_{\Delta R(\mathbf{x})} \rho_G(\mathbf{y}) d\mathbf{y}$.

Then, let two events T and Q be as follows:

$$T \equiv \mathbf{s}_k \notin R(\mathbf{x}) \text{ for all } \mathbf{s}_k \in \mathcal{D} \ (k = 1, 2, \ldots, \psi), \text{ and}$$
$$Q \equiv \begin{cases} \mathbf{s}_i \in \Delta R(\mathbf{x}) \text{ for } \mathbf{s}_i \in \mathcal{D}, \text{ and} \\ \mathbf{s}_k \notin \Delta R(\mathbf{x}) \text{ for all } \mathbf{s}_k \in \mathcal{D} \ (k = 1, 2, \ldots, \psi, i \neq k). \end{cases}$$

Next, the probability $Pr(T \wedge Q) = Pr(T)Pr(Q \mid T)$ denotes that $\mathbf{s}_i$ is the nearest neighbor of $\mathbf{x}$ in $\mathcal{D}$, *i.e.*, $\mathbf{x} \in V_i$, where

$$Pr(T) = (1-f)^{\psi}, \text{ and}$$
$$Pr(Q \mid T) = \frac{\Delta f}{1-f}\left\{1 - \frac{\Delta f}{1-f}\right\}^{\psi-1}.$$

By letting Δf be infinite decimal df, $\Delta f/(1-f) \to df/(1-f)$ and $\{1-\Delta f/(1-f)\} \to 1$. Thus, we obtain the following total probability that $\mathbf{s}_i$ is the nearest neighbor of $\mathbf{x}$ in $\mathcal{D}$, *i.e.*, $\mathbf{x} \in V_i$, by integrating $Pr(T \wedge Q)$ on $f \in [0,1]$ for every $i = 1, 2, \ldots, \psi$.

$$Pr(\mathbf{x} \in V_i) = \int_0^1 (1-f)^{\psi} \frac{df}{1-f} = \frac{1}{\psi}.$$

Theorem 1 *Let a L-bit code vector $\mathcal{B}(\mathbf{x}) = [b_1(\mathbf{x}), b_2(\mathbf{x}), \ldots, b_T(\mathbf{x})]$ denote the VDeH encoded binary vector for any point $\mathbf{x}$, where $T = L/w$ is the number of Voronoi diagrams generated and every $b_i(\mathbf{x})$ is the w-bit vector generated by the i-th VDeH encoded hashing. When each Voronoi diagram has 2^w regions, every bit in $\mathcal{B}(\mathbf{x})$ is mutually independent.*

Proof. For any two bits $\alpha, \beta \in \mathcal{B}(\mathbf{x})$ located at different Voronoi cells, there are the following two cases:
Case 1: α and β are generated by the different encoded hashing.
Let $\alpha \in b_j$ and $\beta \in b_k$ $(1 \leqslant j < k \leqslant T)$. Given that Voronoi diagrams generated in each random sampling are mutually independent, the bits obtained from each encoded hashing are also mutually independent. Consequently, it follows that:

$$Pr(b_j = s_j, b_k = s_k) = Pr(b_j = s_j) \cdot Pr(b_k = s_k),$$

where $s_j, s_k \in \{0,1\}^w$. Moreover, for the bits $\alpha \in b_j$ and $\beta \in b_k$

$$Pr(\alpha = v_\alpha, \beta = v_\beta) = Pr(\alpha = v_\alpha) \cdot Pr(\beta = v_\beta),$$

where $v_\alpha, v_\beta \in \{0,1\}$.
Case 2: α and β are generated by the same encoded hashing.
Let $\alpha, \beta \in b_i$ $(1 \leqslant i \leqslant T)$, $b_i = [e_1, e_2, \ldots, e_w]$, and $\mathcal{B}$ denote a set of ψ different w-bit binary codes correspond to the ψ distinct Voronoi cells.
When $\psi = 2^w$, $\mathcal{B}$ exhaustively represents all probable w-bit binary codes corresponding to ψ Voronoi cells. Let $v_\alpha, v_\beta \in \{0,1\}$, then we have $\sum \mathbb{1}[\alpha = v_\alpha] = \frac{\psi}{2}$, and $\sum \mathbb{1}[\alpha = v_\alpha, \beta = v_\beta] = \frac{\psi}{4}$. Based on Lemma 1, it follows that

$$\begin{aligned}
Pr(\alpha = v_\alpha, \beta = v_\beta) &= \textstyle\sum_{i=1}^{\psi}(Pr(\mathbf{x} \in V_i)\mathbb{1}[\alpha = v_\alpha, \beta = v_\beta]) \\
&= \tfrac{1}{\psi}\textstyle\sum_{i=1}^{\psi}\mathbb{1}[\alpha = v_\alpha, \beta = v_\beta] \\
&= \frac{\sum_{i=1}^{\psi}\mathbb{1}[\alpha = v_\alpha]}{\psi} \times \frac{\sum_{i=1}^{\psi}\mathbb{1}[\beta = v_\beta]}{\psi} \\
&= \textstyle\sum_{i=1}^{\psi}(Pr(\mathbf{x} \in V_i)\mathbb{1}[\alpha = v_\alpha]) \times \sum_{i=1}^{\psi}(Pr(\mathbf{x} \in V_i)\mathbb{1}[\beta = v_\beta]) \\
&= Pr(\alpha = v_\alpha) \cdot Pr(\beta = v_\beta).
\end{aligned}$$

Consequently, in all cases, α and β are mutually independent. Since α and β are arbitrary bits in $\mathcal{B}$, when each Voronoi diagram has 2^w regions, every bit in $\mathcal{B}(\mathbf{x})$ generated by VDeH is mutually independent.

5 Experiment

5.1 Datasets and Settings

To evaluate the proposed Voronoi diagram encoded hashing, we conducted experiments on four public datasets, including two image datasets and two text datasets. It is important to note that existing studies mainly evaluate their performance on image datasets [9, 10, 15, 22, 32, 35, 34], our work extends the evaluation to text datasets for a more comprehensive analysis. Unlike images, which have obviously discriminating features and can be easily identified to guide retrieval tasks, text data presents a greater challenge due to its inherent complexity. The datasets vary in size and dimensionality: CIFAR-10 [20] contains 60,000 images (512 dimensions), GIST [19] contains one million images represented by 960-dimensional descriptors, Nytimes [27] includes 290,000 articles (256 dimensions), and Kosarak [8] includes 74,962 click-stream news (27,983 dimensions).

Following existing studies [15, 17, 22], we use 10,000 randomly sampled instances for training. We then randomly sample 500 instances, different from the training set as queries. The retrieval performance is assessed using two frequently used evaluation metrics, i.e., mean average precision (mAP) and the precision-recall curve (PR curve). We compared the performance of VDeH with the three types of state-of-the-art methods, i.e., (1) thresholding-based methods: spectral hashing (SH), circulant binary embedding (CBE), iterative quantization (ITQ), bilinear projections (BP), and sparse projections (SP); (2) hyperspheres-based method: spherical hashing (SpH); (3) hyperplanes-based methods: density sensitive hashing (DSH), sparse embedding and least variance encoding (SELVE), and refining codes for locality sensitive hashing (rcLSH). The parameters within each hashing method were assigned to default or suggested values by authors. For VDeH, the parameter ψ is searched in $\{2^2, 2^3, \ldots, 2^8\}$. All experiments are executed on a Linux CPU machine: AMD 128-core CPU with each core running at 2 GHz and 1T GB RAM.

5.2 Results and discussion

Comparing to the state-of-the-art. Table 2 presents the mAP scores of our proposed VDeH and the competing hashing methods conducted on the benchmark datasets. VDeH has the best results, compared with all the state-of-the-art methods, across all the tested number of hash bits ranging from 128 bits to 2048 bits. The mAP of VDeH increases consistently as the number of hash bits increases. Overall, only DSH demonstrated comparable performance across all datasets, positioning it as VDeH's closest contender. Among existing methods, rcLSH exhibited the best performance on image datasets. Notably, many existing methods showed a marked decrease in performance on the text datasets as

Table 2: The mAP scores on four datasets with different number of hash bits. The highest score in each row is marked with bold font.

Dataset	bits	VDeH	rcLSH	SH	CBE	ITQ	BP	SP	DSH	SELVE	SpH
CIFAR-10	128	**0.366**	0.283	0.117	0.315	0.355	0.344	0.352	0.318	0.199	0.242
	256	**0.438**	0.351	0.153	0.342	0.377	0.343	0.370	0.372	0.176	0.289
	512	**0.468**	0.392	0.193	0.396	0.389	0.339	0.397	0.401	0.167	0.304
	1024	**0.501**	0.412	0.201	0.401	0.398	0.332	0.407	0.429	0.169	0.314
	2048	**0.525**	0.432	0.214	0.407	0.401	0.335	0.426	0.457	0.169	0.321
GIST	128	**0.664**	0.582	0.338	0.617	0.644	0.645	0.604	0.501	0.430	0.222
	256	**0.714**	0.631	0.454	0.644	0.650	0.641	0.614	0.580	0.399	0.223
	512	**0.763**	0.651	0.501	0.656	0.654	0.634	0.638	0.618	0.402	0.232
	1024	**0.774**	0.659	0.514	0.657	0.657	0.642	0.638	0.610	0.399	0.233
	2048	**0.793**	0.687	0.532	0.662	0.663	0.644	0.641	0.621	0.398	0.242
Nytimes	128	**0.116**	0.017	0.033	0.022	0.023	0.023	0.022	0.016	0.003	0.100
	256	**0.165**	0.026	0.055	0.029	0.029	0.029	0.027	0.033	0.004	0.110
	512	**0.340**	0.028	0.160	0.026	0.030	0.026	0.029	0.185	0.004	0.146
	1024	**0.348**	0.103	0.295	0.108	0.104	0.008	0.106	0.339	0.006	0.088
	2048	**0.376**	0.113	0.310	0.100	0.097	0.008	0.100	0.326	0.008	0.089
Kosarak	128	**0.457**	0.235	0.375	0.255	0.261	0.254	0.258	0.368	0.203	0.256
	256	**0.603**	0.271	0.512	0.286	0.293	0.295	0.304	0.494	0.226	0.296
	512	**0.607**	0.308	0.571	0.301	0.298	0.346	0.341	0.437	0.279	0.346
	1024	**0.615**	0.307	0.569	0.300	0.302	0.351	0.357	0.415	0.289	0.372
	2048	**0.638**	0.313	0.579	0.309	0.312	0.359	0.369	0.412	0.300	0.381

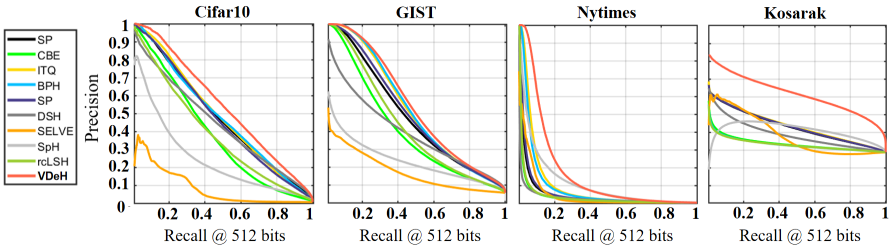
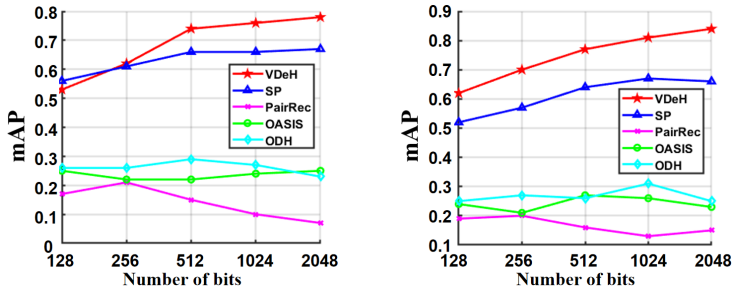


Fig. 2: PR curves on four datasets with the 512-bit binary codes

the hash bits increases, except VDeH, SH, and DSH. This is probably because text data have complex density variations. DSH and SH are adapted by using more hash functions in dense distribution and fewer in sparse distribution; while VDeH naturally adapts because it creates smaller Voronoi cells in dense distribution and larger ones in sparse distribution [7, 30]. In addition, Figure 2 provides the PR curves of VDeH and the competing hashing methods, when the number of hash bits is 512. We can observe that the VDeH curve (in red) consistently dominates the curves of other methods across all datasets, demonstrating its superior precision and recall scores. For the cases with different hash bits, similar results can be observed (not presented due to lack of space).

Comparing to the deep hashing methods. Deep hashing is known for its high accuracy in semantic-based image retrieval [24]. However, deep hashing methods cannot accurately retrieve input distance similarities, even after extracting semantic information from images. To show this, we use CIFAR-10



(a) Cifar10 (b) GIST

Fig. 3: The mAP scores comparison for five methods on image datasets.

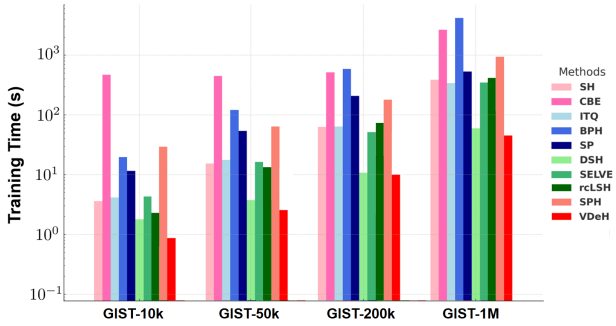
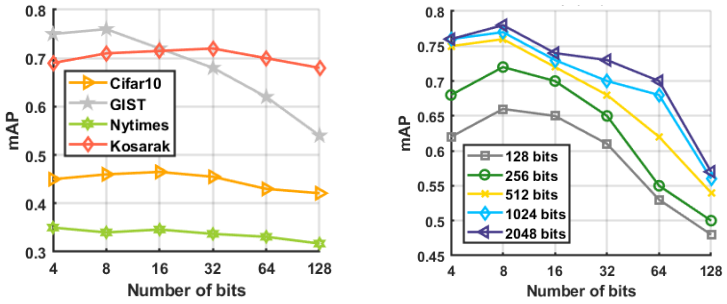


Fig. 4: Training time comparison on GIST dataset with the 512-bit binary codes.

and GIST datasets, which possess semantic information in images. Initially, we utilized ResNet18 to extract embeddings from these two datasets. Since the Euclidean distance between the obtained embeddings can reflect the semantic information between images, we used the Euclidean distances derived from these embeddings as ground truth to test the performance of VDeH and SP, as well as three deep hashing methods PairRec [11], OASIS [33] and ODH [14]. The comparison results are shown in Figure 3. We can observe that typical L2H methods can effectively preserve the Euclidean distances between instances, whereas deep methods lack this capability. This is due to the fact that deep methods incorporate labels as part of their loss function during training, aiming to map instances with the same label to close hash codes, rather than being designed to preserve the input distance similarities between instances.

Training time comparison. Unlike existing hashing methods that rely on optimization for learning hash functions, VDeH's training process is straightforward, requiring only the random sampling of a given number of points from the training data, each corresponding to a nearest-neighbor-based hash function (as shown in Eq. 2). Note that all methods require the transformation of training data into binary codes via hash functions. We tested the training time of various methods on the GIST dataset, shown in Figure 4. VDeH took the shortest



(a) 512-bit binary codes (b) GIST dataset

Fig. 5: The impact of Parameter ψ: (a) testing on different datasets with 512-bit binary codes; (b) Testing on GIST dataset with different number of hash bits.

time for every training data size, from 10k to 1M, demonstrating its efficiency superiority over other methods. This is because VDeH needs no learning, but all other existing methods must perform learning.

Effects of hyper-parameter ψ. Figure 5 illustrates the impact of the parameter ψ on the performance of the proposed VDeH method across different datasets and binary code lengths. In subfigure (a), we observe that only the GIST dataset shows a degree of sensitivity to ψ, while Cifar10, Nytimes, and Kosarak show relatively stable mAP scores across the different ψ values. Subfigure (b) further explores this by examining VDeH's mAP scores on the GIST dataset with varying binary code lengths. We can observe that the optimal ψ value remains consistent regardless of the number of hash bits, ranging from 128 to 2048 bits. This suggests that the optimal selection of ψ is intrinsic to the characteristics of the specific dataset, rather than the code length.

6 Discussion

We acknowledge that the integration of Voronoi diagrams with hashing techniques is not an entirely novel concept [1, 23]. However, our proposed VDeH distinguishes itself from existing works through its unique mechanism for generating data-dependent hash functions and its redefinition of the hashing process. Prior approaches, such as VLSH by Loi *et al.* [23], use Voronoi regions to primarily localize and adapt standard Locality Sensitive Hashing (LSH); its core hashing still relies on LSH's random projection-based mechanism within these regions. Similarly, the work by Ajani & Wanjari combines Voronoi clustering with hash indexing, where hash indexing serves to optimize their k-means clustering algorithm for uncertain data rather than generating binary codes for similarity search. In contrast, VDeH directly derives hash functions from the partitions of Voronoi diagrams themselves. The goal of VDeH is to generate highly efficient binary representations for large-scale datasets, thereby enabling

rapid similarity search and retrieval. The main novelty of VDeH lies in its utilization of Voronoi diagrams as the core data-dependent hash function generator. It proposes a method that can achieve key hashing properties, including full space coverage, entropy maximization, and bit independence, without resorting to complex learning procedures, notably through its specifically defined encoded hashing mechanism.

7 Conclusion

We introduce VDeH, a novel no-learning data-dependent method for binary hashing. VDeH distinguishes itself from L2H methods in three aspects. First, VDeH employs the unique space partitioning of Voronoi diagrams, leveraging its three previously concealed properties that match perfectly those required for hashing. Second, VDeH is an implementation of a new definition of hashing which equates the probability of some hashed condition to a similarity function. The definition enables Voronoi diagrams, already used to compute the similarity of Isolation Kernel, to construct hash functions easily without much effort. No existing L2H methods have used a similar definition as far as we know. Third, the integration of encoded hashing enables VDeH to generate mutually independent binary bits, which could not be utilized by existing methods because they have already employed independent hash functions. Our experiments show that VDeH exhibits superior performance with lower computational cost compared to the state-of-the-art methods.

Acknowledgments. We thank the anonymous reviewers for their valuable comments. This work was supported in part by the National Natural Science Foundation of China (Grant No. 92470116).

References

1. Ajani, S., Wanjari, M.: An efficient approach for clustering uncertain data mining based on hash indexing and voronoi clustering. In: 2013 5th International Conference and Computational Intelligence and Communication Networks, pp. 486–490. IEEE (2013)
2. Al-Odat, Z.A., Ali, M., Abbas, A., Khan, S.U.: Secure hash algorithms and the corresponding FPGA optimization techniques. ACM Computing Surveys (CSUR) **53**(5), 1–36 (2020)
3. Borthwick, A., Ash, S., Pang, B., Qureshi, S.: Scalable Blocking for Very Large. In: ECML PKDD 2020 Workshops, September 14–18, 2020, Proceedings, vol. 1323, p. 303. Springer Nature (2021)
4. Cai, D.: A revisit of hashing algorithms for approximate nearest neighbor search. IEEE Transactions on Knowledge and Data Engineering **33**(6), 2337–2348 (2021)
5. Charikar, M.S.: Similarity Estimation Techniques from Rounding Algorithms. In: Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing, pp. 380–388 (2002)

6. Chi, L., Zhu, X.: Hashing techniques: A survey and taxonomy. ACM Computing Surveys (Csur) **50**(1), 1–36 (2017)
7. Devroye, L., Györfi, L., Lugosi, G., Walk, H.: On the measure of Voronoi cells. Journal of Applied Probability **54**(2), 394–408 (2017)
8. FIMI: FIMI datasets. http://fimi.ua.ac.be/data/. Accessed 2 Jan 2017 (2017)
9. Gong, Y., Kumar, S., Rowley, H.A., Lazebnik, S.: Learning binary codes for high-dimensional data using bilinear projections. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2013)
10. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(12), 2916–2929 (2013)
11. Hansen, C., Hansen, C., Simonsen, J.G., Alstrup, S., Lioma, C.: Unsupervised Semantic Hashing with Pairwise Reconstruction. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (2020)
12. Haq, I.U., Caballero, J.: A survey of binary code similarity. Acm computing surveys (csur) **54**(3), 1–38 (2021)
13. He, J., Chang, S.-F., Radhakrishnan, R., Bauer, C.: Compact hashing with joint optimization of search accuracy and time. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 753–760 (2011)
14. He, L., Huang, Z., Liu, C., Li, R., Wu, R., Liu, Q., Chen, E.: One-bit Deep Hashing: Towards Resource-Efficient Hashing Model with Binary Neural Network. In: Proceedings of the 32nd ACM International Conference on Multimedia, pp. 7162–7171 (2024)
15. Heo, J.-P., Lee, Y., He, J., Chang, S.-F., Yoon, S.-E.: Spherical hashing: Binary code embedding with hyperspheres. IEEE transactions on pattern analysis and machine intelligence **37**(11), 2304–2316 (2015)
16. Hu, D., Chen, Z., Wu, J., Sun, J., Chen, H.: Persistent memory hash indexes: An experimental evaluation. Proceedings of the VLDB Endowment **14**(5), 785–798 (2021)
17. Jin, Z., Li, C., Lin, Y., Cai, D.: Density sensitive hashing. IEEE Transactions on Cybernetics **44**(8), 1362–1371 (2014)
18. Joly, A., Buisson, O.: Random Maximum Margin Hashing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2011)
19. Jégou, H., Douze, M., Schmid, C.: Product Quantization for Nearest Neighbor Search. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 117–128 (2011)
20. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical Report from the Department of Computer Science at the University of Toronto (2009)
21. Li, Y., van Gemert, J.: Deep Unsupervised Image Hashing by Maximizing Bit Entropy. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 3, pp. 2002–2010 (2021)
22. Liu, H., Zhou, W.H., Wu, Z., Zhang, S.C., Li, G., Li, X.L.: Refining Codes for Locality Sensitive Hashing. IEEE Transactions on Knowledge and Data Engineering, pp. 1–11 (2023)
23. Loi, T.L., Heo, J.-P., Lee, J., Yoon, S.-E.: VLSH: Voronoi-based locality sensitive hashing. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5345–5352. IEEE (2013)

24. Luo, X., Wang, H., Wu, D., Chen, C., Deng, M., Huang, J., Hua, X.-S.: A survey on deep hashing methods. ACM Transactions on Knowledge Discovery from Data **17**(1), 1–50 (2023)
25. Mao, J.-Y., Pan, Q.-K., Miao, Z.-H., Gao, L., Chen, S.: A hash map-based memetic algorithm for the distributed permutation flowshop scheduling problem with preventive maintenance to minimize total flowtime. Knowledge-Based Systems **242**, 108413 (2022)
26. Min, X., Lu, K., Liu, P., Wan, J., Xie, C., Wang, D., Yao, T., Wu, H.: SepHash: A Write-Optimized Hash Index On Disaggregated Memory via Separate Segment Structure. Proceedings of the VLDB Endowment **17**(5), 1091–1104 (2024)
27. Pham, N., Liu, T.: Falconn++: A Locality-sensitive Filtering Approach for Approximate Nearest Neighbor Search. In: Proceedings of the 36th Annual Conference on Neural Information Processing Systems (2022)
28. Singh, A., Gupta, S.: Learning to hash: a comprehensive survey of deep learning-based hashing methods. Knowledge and Information Systems **64**(10), 2565–2597 (2022)
29. Thangavel, M., Varalakshmi, P.: Enabling ternary hash tree based integrity verification for secure cloud data storage. IEEE Transactions on Knowledge and Data Engineering **32**(12), 2351–2362 (2019)
30. Ting, K.M., Zhu, Y., Zhou, Z.-H.: Isolation Kernel and Its Effect on SVM. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2018)
31. Wang, J., Zhang, T., Song, J., Sebe, N., Shen, H.T.: A Survey on Learning to Hash. IEEE Transactions on Pattern Analysis and Machine Intelligence **13**(9) (2017)
32. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Proceedings of the 21st Annual Conference on Neural Information Processing Systems, pp. 1753–1760 (2008)
33. Wu, X.-M., Luo, X., Zhan, Y.-W., Ding, C.-L., Chen, Z.-D., Xu, X.-S.: Online Enhanced Semantic Hashing: Towards Effective and Efficient Retrieval for Streaming Multi-Modal Data. In: AAAI Conference on Artificial Intelligence, pp. 4263–4271 (2022)
34. Xia, Y., He, K., Kohli, P., Sun, J.: Sparse projections for high-dimensional binary codes. In: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3332–3339 (2015)
35. Yu, F.X., Kumar, S., Gong, Y., Chang, S.-F.: Circulant binary embedding. In: Proceedings of the 31st International Conference on Machine Learning (2014)
36. Zhu, L., Zheng, C., Guan, W., Li, J., Yang, Y., Shen, H.T.: Multi-modal hashing for efficient multimedia retrieval: A survey. IEEE Transactions on Knowledge and Data Engineering **36**(1), 239–260 (2023)
37. Zhu, X., Zhang, L., Huang, Z.: A Sparse Embedding and Least Variance Encoding Approach to Hashing. IEEE Transactions on Image Processing (2014)