VDeH: Voronoi Diagram Encoded Hashing for Effective and Efficient Similarity Search, Applicable to Different Types of Databases

YANG XU, State Key Laboratory for Novel Software Technology & School of Artificial Intelligence, Nanjing University, China
KAI MING TING*, State Key Laboratory for Novel Software Technology & School of Artificial Intelligence, Nanjing University, China
XINPENG LI, State Key Laboratory for Novel Software Technology & School of Artificial Intelligence, Nanjing University, China
YUNPENG LI, State Key Laboratory for Novel Software Technology & School of Artificial Intelligence, Nanjing University, China

The goal of learning to hash (L2H) is to derive data-dependent hash functions from a given data distribution 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 its power, *i.e.*, learning. First, a recent study shows that a version of locality-sensitive hashing without learning can achieve comparable accuracy to L2H methods with less time cost. Second, existing L2H methods are constrained to only three types of hash functions: thresholding, hyperspheres, and hyperplanes. In this paper, we identify Voronoi diagrams as a superior alternative for hashing, as they naturally possess three key properties, *i.e.*, full space coverage, entropy maximization, and bit independence, that existing methods must acquire through complex learning. This insight leads us to propose Voronoi Diagram Encoded Hashing (VDeH), a simple and efficient no-learning approach. VDeH constructs hash functions directly from data-driven Voronoi partitions and leverages a simple encoding scheme to generate mutually independent binary bits. The no-learning and data-dependent nature of VDeH makes it an ideal and highly effective plug-and-play component for similarity search, not only in standard vector databases but also for complex data objects such as trajectories and graphs. For complex data objects, VDeH enables a straightforward two-stage process: an appropriate embedding method first maps the data into a vector space, after which VDeH efficiently generates binary hash codes to accelerate retrieval. Comprehensive experiments on multiple large-scale public benchmarks (consisting of databases of vectors, graphs, and trajectories) demonstrate that VDeH significantly outperforms state-of-the-art hashing methods in both retrieval accuracy and computational efficiency, establishing it as a versatile, efficient, and high-performance solution for similarity search across diverse data domains.

JAIR Associate Editor: Julia Handl

JAIR Reference Format:
Yang Xu, Kai Ming Ting, Xinpeng Li, and Yunpeng Li. 2026. VDeH: Voronoi Diagram Encoded Hashing for Effective and Efficient Similarity Search, Applicable to Different Types of Databases. *Journal of Artificial Intelligence Research* 86, Article 43 (August 2026), 31 pages. DOI: 10.1613/jair.1.21934

*Corresponding Author.

Authors' Contact Information: Yang Xu, ORCID: 0000-0003-0291-6250, xuyang@lamda.nju.edu.cn, State Key Laboratory for Novel Software Technology & School of Artificial Intelligence, Nanjing University, Nanjing, Jiangsu, China; Kai Ming Ting, ORCID: 0000-0001-7892-6194, tingkm@nju.edu.cn, State Key Laboratory for Novel Software Technology & School of Artificial Intelligence, Nanjing University, Nanjing, Jiangsu, China; Xinpeng Li, ORCID: 0000-0002-9959-4691, lixp@lamda.nju.edu.cn, State Key Laboratory for Novel Software Technology & School of Artificial Intelligence, Nanjing University, Nanjing, Jiangsu, China; Yunpeng Li, ORCID: 0009-0002-7509-1085, liypnju@foxmail.com, State Key Laboratory for Novel Software Technology & School of Artificial Intelligence, Nanjing University, Nanjing, Jiangsu, China.

DOI: 10.1613/jair.1.21934

1 Introduction

For decades, hashing techniques have been one of the most effective tools commonly used for fast access, analysis, and learning (Charikar 2002; Chi and Zhu 2017; Zhu, Zheng, et al. 2023). Hashing techniques are popular for their simplicity and offer significant advantages in data processing and security (Hu et al. 2021; Al-Odat et al. 2020). 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 (Borthwick et al. 2021; Mao et al. 2022; Min et al. 2024; Thangavel and Varalakshmi 2019).

Learning to hash (L2H) (Liu et al. 2023; Wang, Zhang, et al. 2017; Yu et al. 2014), a prominent subfield within hashing techniques, aims to map data in the input space to a binary coding space, where the Hamming distance can approximate well 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 the 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 (Gong, Kumar, et al. 2013; Gong, Lazebnik, et al. 2013; Weiss et al. 2008; Xia et al. 2015; Yu et al. 2014), hyperspheres (Heo et al. 2015), and hyperplanes (Cai 2021; Jin et al. 2014; Liu et al. 2023; Zhu, Zhang, et al. 2014). 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 (Weiss et al. 2008; Yu et al. 2014) or PCA-based methods (Gong, Kumar, et al. 2013; Gong, Lazebnik, et al. 2013; Xia et al. 2015). Then, matrix values are binarized based on a given threshold. Hypersphere and hyperplane-based methods construct hash functions by first partitioning the input space with their respective space partitioning methods. Points falling into the same partition are mapped to the identical binary code value. Due to randomness in the partitioning process, these methods typically improve binary codes performance by learning a partitioning strategy such that the generated partitions cover all points in the given database uniformly (Heo et al. 2015; Liu et al. 2023).

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

(1) Existing L2H methods claim that their learning process (in optimizing some objective) plays a crucial role in obtaining effective binary codes. Yet, a recent study (Cai 2021) shows that even using brLSH, a version of locality-sensitive hashing (Charikar 2002) without learning, achieves binary codes that have comparable performance as 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 (Gong, Lazebnik, et al. 2013; Heo et al. 2015; Jin et al. 2014; Liu et al. 2023; Weiss et al. 2008; Xia et al. 2015), *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 in the given database such that the total set of hash functions maximizes information entropy;

[1]Note that a related area called *deep hashing* (Luo et al. 2023; Singh and Gupta 2022) 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. In other words, deep hashing is specifically designed for datasets with explicit categorical information, notably image data (Hansen et al. 2020; He, Huang, et al. 2024; Wu, Luo, et al. 2022). But, it is incompatible with datasets lacking this information, including extensive text and tabular data. In addition, it typically generates short binary codes which do not adequately approximate the input distance for the text and tabular data. In essence, deep hashing aims to preserve semantic similarity, whereas the primary goal of L2H is to preserve the distance structure of the input space. Existing studies demonstrate that achieving an effective approximation of the input distance through binary codes requires a code length of $O(d)$, where d is the input dimensionality (Gong, Kumar, et al. 2013; Liu et al. 2023; Xia et al. 2015; Yu et al. 2014).

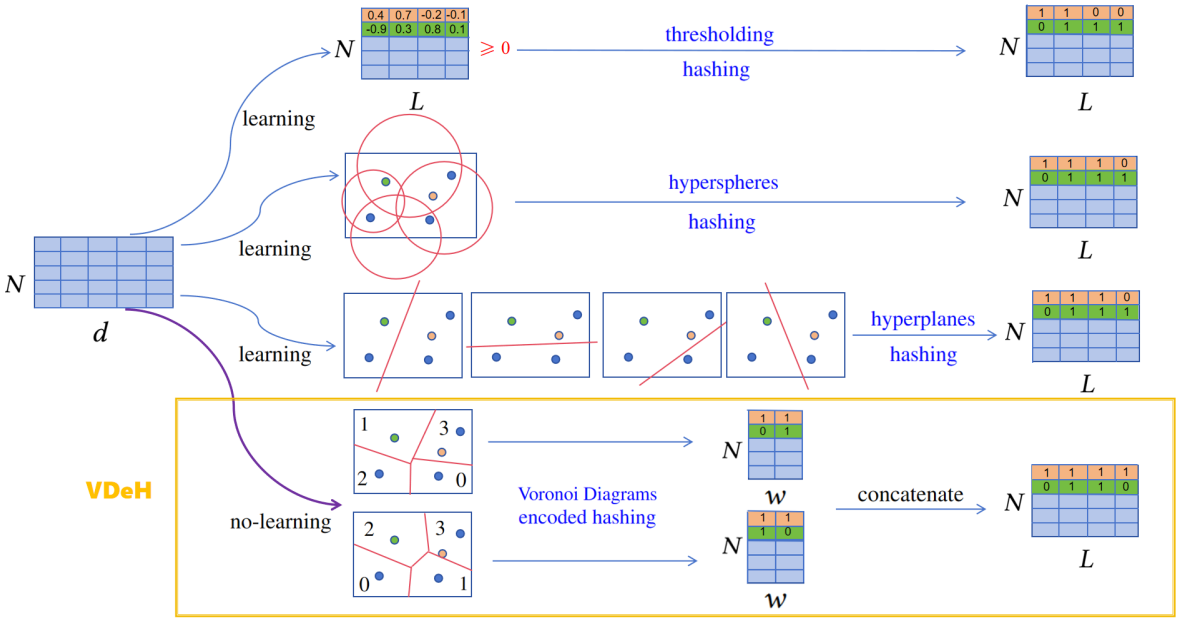


Fig. 1. An illustrative comparison of VDeH using Voronoi diagram and three existing types of hash functions (thresholding, hyperspheres and hyperplanes). The parameters $N = 5$, $d = 5$, $L = 4$, and $w = 2$ are used in this example. Each of the L/w Voronoi diagrams (here 2 diagrams for $L = 4, w = 2$) generates w bits, which are then concatenated.

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 in a Voronoi diagram covers approximately the same number of points (Devroye et al. 2017); 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 readily available in Voronoi diagrams without learning.

Voronoi diagrams facilitate the generation of *data-dependent* hash functions, as these hash functions are derived by data samples in a given dataset that follow the distribution of the data. Specifically, they yield small Voronoi cells in dense regions, and large Voronoi cells are produced in sparse regions (Devroye et al. 2017; Ting, Zhu, et al. 2018). This discovery has led us to propose a simple and effective no-learning method called Voronoi Diagram Encoded Hashing (VDeH), which is the first data-dependent binary hashing scheme based on Voronoi diagram. Figure 1 shows an illustrative comparison of VDeH using Voronoi diagram and three existing types of hash functions.

Moreover, the no-learning and data-dependent nature of VDeH makes it not only superior for standard vector retrieval but also naturally applicable to similarity search on more challenging domains in complex data objects (*e.g.*, graphs and trajectories). In these scenarios, a direct and effective workflow is to first employ a domain-specific embedding method to map the objects into a vector space (Chang et al. 2024; Khoshraftar and An 2024). Subsequently, VDeH is used to efficiently convert these vector representations into compact binary codes for fast retrieval. This decoupled approach provides greater flexibility and generality than end-to-end models, which

require specialized architectures for specific data types (Deng et al. 2024; Wang, Xu, et al. 2021). This process highlights the flexibility of VDeH as an ideal plug-and-play component, extending the reach of efficient hashing to a wide variety of domains.

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 (Section 3).
- Creating a new definition of hashing scheme derived directly from a kernel based on Voronoi diagrams, and proposing VDeH that creates the first kernel-based implementation of a data-dependent binary hashing scheme, and formulating a binary distance function intrinsic to VDeH. In addition, we theoretically prove that VDeH is capable of generating mutually independent binary bits (Section 4).
- Conducting comprehensive empirical comparisons and analyses to demonstrate VDeH's superior effectiveness and efficiency compared to state-of-the-art L2H methods. Its performance is validated on standard vector databases and complex data objects such as trajectories and graphs (Section 5).

2 Background and Related Work

This section reviews the essential literature with a primary focus on Learning to Hash (L2H), the foundational domain of our method. We first detail its core principles and dominant methods to establish a thorough technical background. Then, we briefly survey representative methods of similarity search for complex data objects, taking graph and trajectory data as examples, to illustrate how the L2H method can be used for retrieval of these data.

2.1 Learning to Hash

With the emergence of large-scale data, numerous L2H methods have been developed (Haq and Caballero 2021; Liu et al. 2023; Wang, Zhang, et al. 2017). The generic binary hashing problem is the following. Given N data points $D = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{d \times N}$, generate L hash functions to map a data point $\mathbf{x}$ into a L-bit binary hash code $\mathcal{B}_\mathbf{x} = [h_1(\mathbf{x_i}), h_2(\mathbf{x_i}), \ldots, h_L(\mathbf{x_i})]$ where $h_l(\mathbf{x}) \in \{0, 1\}$ is the l-th hash function. For the linear binary projection-based hashing (Gong, Kumar, et al. 2013; Liu et al. 2023; Yu et al. 2014):

$$h_l(\mathbf{x}) = sgn(F(\mathbf{R}_l\mathbf{x} + t_l)), \tag{1}$$

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 can be broadly categorized into three classes: thresholding (Gong, Kumar, et al. 2013; Gong, Lazebnik, et al. 2013; Weiss et al. 2008; Xia et al. 2015; Yu et al. 2014), hyperspheres (Heo et al. 2015), and hyperplanes (Cai 2021; Jin et al. 2014; Liu et al. 2023; Zhu, Zhang, et al. 2014). Thresholding-based methods initiate the process by transforming distance or similarity relationships from the input space into a new matrix of size $N \times L$. Then, matrix values are binarized based on a given threshold. For example, *iterative quantization* (ITQ) employ projections derived from the PCA preprocessing (Maćkiewicz and Ratajczak 1993), and subsequently determine an optimal rotation matrix to substitute the input data, followed by scalar quantization. The goal of ITQ is to ensure that bits in the binary codes contribute equally to distance evaluation by equalizing variances along the principal component directions. Hypersphere and hyperplane-based methods construct hash functions by partitioning the input space. For example, *density sensitive hashing* (DSH) (Jin et al. 2014) uses random hyperplanes to partition the space. It then refines these partitions based on data density variations, performing finer-grained partitioning in dense regions while applying coarser-grained partitioning in sparse regions, thereby enhancing the performance of the binary codes.

Moreover, 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 to ensure three key properties:

full space coverage, *entropy maximization*, and *bit independence*. For example, *spectral hashing* (SH) (Weiss et al. 2008) considers the similarity-distance product minimization. Given $\mathcal{B}_\mathbf{x} \in \{0,1\}^L$, SH aims to minimize the average Hamming distance between similar neighbors, represented by $\sum_{ij} W_{ij}||\mathcal{B}_\mathbf{x} - \mathcal{B}_\mathbf{y}||^2$, where W_{ij} is the similarity between points i and j as measured by Gaussian kernel, and $||\mathcal{B}_\mathbf{x} - \mathcal{B}_\mathbf{y}||^2$ is the Hamming distance between binary codes $\mathcal{B}_\mathbf{x}$ and $\mathcal{B}_\mathbf{y}$. SH uses two constraints: (i) $\sum_i \mathcal{B}_\mathbf{x} = \frac{1}{2}$ to ensure entropy maximization, and (ii) $\frac{1}{n}\sum_i \mathcal{B}_\mathbf{x}\mathcal{B}_\mathbf{x}^T = I$ to guarantee bit independence. In addition, Spherical Hashing (SpH) (Heo et al. 2015) enables the mapping of similar data points to nearby regions in the spherical region, facilitating efficient similarity search. The method incorporates iterative optimization of the radius and center of the sphere to ensure comprehensive coverage of all data points, while maintaining that different spherical regions contain approximately the same number of data points. 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 by constructing similar constraints. Methods operated on spatial partitioning necessitate that the hash functions provide complete coverage of the input space (Cai 2021; Heo et al. 2015; Jin et al. 2014; Liu et al. 2023; Zhu, Zhang, et al. 2014). 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 (Gong, Kumar, et al. 2013; Gong, Lazebnik, et al. 2013; Heo et al. 2015; Jin et al. 2014; Liu et al. 2023; Weiss et al. 2008; Xia et al. 2015; Yu et al. 2014; Zhu, Zhang, et al. 2014). The detailed explanations of how these methods guarantee these properties and their categorizations can be found in comprehensive survey papers (Luo et al. 2023; Wang, Zhang, et al. 2017). These 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. This reliance on a learning phase not only introduces significant computational overhead and complexity, but can also lead to sub-optimal solutions if the chosen optimization objective does not perfectly align with the intrinsic data structure for a given task (Cai 2021).

2.2 Similarity Search on Complex Data

Similarity search in large-scale databases of complex data objects, such as trajectories and graphs, is a foundational but challenging task.

Initially, this field relied on traditional distance metrics defined directly on the data structures, like *Hausdorff distance* (Hausdorff 1914), *Fréchet distance* (Ward 1954), and *Dynamic Time Warping (DTW)* (Myers et al. 1980) for trajectories and *Graph Edit Distance (GED)* (Gao et al. 2010) for graphs. Although effective at capturing structural nuances, the high computational complexity of these metrics became a primary bottleneck for retrieval at scale. To overcome this, a dominant trend shifted towards learning-based approaches that map complex objects into fixed-dimensional vector representations (Li, Zhao, et al. 2018b; Wu, Pan, et al. 2020; Yang et al. 2021; Zhang et al. 2018). Although comparing these embeddings is significantly more efficient than traditional metrics, the problem of searching through massive databases of the resulting vectors remains, mirroring the challenges of large-scale vector retrieval (Chang et al. 2024; Khoshraftar and An 2024). Consequently, the latest approaches focus on learning end-to-end hashing models that generate compact binary codes directly from the raw complex data, enabling rapid search via bitwise operations (Deng et al. 2024; Wang, Xu, et al. 2021). For example, to approximate metrics like DTW, the Traj2Hash model (Deng et al. 2024) trains a bespoke two-channel trajectory encoder using a loss against pre-computed similarities. Simultaneously, to learn the binary codes, it applies a ranking-based hashing loss which requires a massive number of trajectory triplets, which are generated using a

Table 1. Key notations used.

Notation	Definition
D	Input dataset with N points in $\mathbb{R}^d$
$\mathcal{X}$	The input space from which the dataset D is drawn
$\mathcal{D}$	A set of ψ points randomly sampled from D to generate a Voronoi diagram
$\mathcal{B}_{\mathbf{x}}$	L-bit binary code corresponds to $\mathbf{x}$ in D
κ	Kernel $\kappa(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$
$\varphi(\mathbf{x}\|D)$	The kernel feature map of instance $\mathbf{x}$ obtained from D
$\mathcal{U}$	A set of complex data objects
$\mathcal{E}$	A set of embedded representations of $\mathcal{U}$ in $\mathbb{R}^d$
S	A similarity function defined in $\mathbb{R}^d$
w	The number of bits generated by encoded hashing over a Voronoi diagram
H	A set of hash functions derived from 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 $\mathcal{H}$

heuristic based on coarse spatial grids, a strategy deeply coupled with the nature of spatial trajectories. However, these approaches are inherently specialized, requiring bespoke architectures tailored to specific data structures, and are not generalizable across different data types.

This landscape highlights the value of a decoupled approach, where a universal and efficient hashing method like VDeH can be combined with any of the powerful learning-based embedding models. Such a two-stage process offers a plug-and-play solution that is both highly scalable and readily adaptable to any type of complex data, seamlessly integrating the strengths of deep representation learning with the speed of binary code retrieval.

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 (Heo et al. 2015). 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 $D \subset \mathbb{R}^d$ with N points, every hash function $h \in H$ shall cover approximately the same number of points $\mathbf{x} \in D$, *i.e.*, uniformly distributed over all hash functions:

$$\forall h \in H, \left| |h(D)| - \tfrac{N}{|H|} \right| \leqslant \varepsilon \tag{2}$$

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 (Li and Gemert 2021), 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 (He, Chang, et al. 2011; Heo et al. 2015; Joly and Buisson 2011; Weiss et al. 2008). 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 (Devroye et al. 2017). This natural balancing occurs because random sampling places more partition centers in high-density regions and fewer in sparse ones. Consequently, dense areas are divided into many small Voronoi cells, while sparse areas are covered by a few large cells, leading to a roughly equal number of data points residing within each cell (Ting, Washio, et al. 2024).
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 The Proposed Hashing Method: VDeH

Here we give the formal definition of our proposed 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 κ (Ting, Washio, et al. 2024; Ting, Zhu, et al. 2018) is defined as:

$$\kappa(\mathbf{x}, \mathbf{y}|\mathcal{H}(D)) = \mathbb{E}_{H\sim\mathcal{H}(D)}[\mathbb{1}(\mathbf{x}, \mathbf{y} \in \theta \mid \theta \in H)], \tag{3}$$

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:

$$\kappa(\mathbf{x}, \mathbf{y}|\mathcal{H}(D)) = \mathbb{E}_{H\sim\mathcal{H}(D)}[\mathbb{1}(h(\mathbf{x}) = h(\mathbf{y}) = 1 \mid h \in H)]. \tag{4}$$

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 $D \subset \mathbb{R}^{d\times N}$, the proposed VDeH scheme is defined as follows:

Definition 4.1. The VDeH scheme is a family $\mathcal{H}(D)$ of hash functions created by Voronoi diagrams associated with a distribution $P_{\mathcal{H}}$ over $\mathcal{H}(D)$ generated from a dataset D such that it satisfies

$$Pr_{h\in H\in\mathcal{H}(D)}[h(\mathbf{x}) = h(\mathbf{y}) = 1] = \kappa(\mathbf{x}, \mathbf{y}|\mathcal{H}(D)), \tag{5}$$

where Isolation Kernel $\kappa(\mathbf{x}, \mathbf{y}|\mathcal{H}(D))$ is derived from Voronoi diagrams generated from D; 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. It is essential to distinguish the *data-dependent* property of VDeH from the learning process characteristic of existing L2H methods. In L2H, learning typically involves the iterative optimization of a loss function to derive optimal hash parameters (Luo et al. 2022; Wang, Zhang, et al. 2017). VDeH, however, requires *no-learning* where the hash functions are constructed through a single-pass random sampling of ψ data points to form ψ Voronoi partitions. In this sense, VDeH is conceptually aligned with Locality-Sensitive Hashing (LSH), where the hash functions are defined by geometric structures rather than iterative data fitting (Charikar 2002; Datar et al. 2004). Although VDeH's hashing scheme is data-dependent, as the sampled points naturally reflect the underlying data distribution, it does not need optimization. This approach ensures that the three key properties of effective hashing mentioned in Section 3 are obtained as inherent geometric consequences of the Voronoi partitions rather than through computationally intensive optimization.

4.2 Implementation Details of VDeH

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

$$h_l(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad dist(\mathbf{x}, \mathbf{s}_l) > dist(\mathbf{x}, \mathcal{D}) \\ 1 & \text{if} \quad dist(\mathbf{x}, \mathbf{s}_l) = dist(\mathbf{x}, \mathcal{D}), \end{cases} \tag{6}$$

where $dist(\mathbf{x}, \mathbf{s}_l) = \| \mathbf{x} - \mathbf{s}_l \|$ denotes the ℓ_2 distance between $\mathbf{x}$ and $\mathbf{s}_l$; 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:

$$\forall h \in H, \forall \mathbf{x} \in \mathbb{R}^d, \sum_{l\in[1,\psi]} \mathbb{1}[h_l(\mathbf{x}) = 1] = 1. \tag{7}$$

This dependency arises because if $\mathbf{x}$ belongs to the u-th Voronoi cell, *i.e.*, $h_u(\mathbf{x}) = 1$, then $\mathbf{x}$ can not fall into other regions, hence $h_l(\mathbf{x}) = 0\ \forall l \neq u$. 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 for a point $\mathbf{x} \in \mathbb{R}^d$ via VDeH is outlined as follows:

1. Given a dataset D, 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.(7), there is one and only one $h_l(\mathbf{x}) = 1$ for any point $\mathbf{x}$, and $\forall u \neq l, h_u(\mathbf{x}) = 0$.

2. The 'raw' ψ-bit vector $[h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_\psi(\mathbf{x})]$, with $\psi \leqslant 2^w$, is encoded as a new vector with w bits:

$$b(\mathbf{x}) = [e_1(\mathbf{x}), e_2(\mathbf{x}), \ldots, e_w(\mathbf{x})], \tag{8}$$

where the encoded hashing function $e_k(\mathbf{x}), k = 1, 2, \ldots, w$, is given as:

$$e_k(\mathbf{x}) = L_{k-1} \bmod 2, \text{ where } L_0 = [\arg_{l\in[1,\psi]} h_l(\mathbf{x}) = 1] - 1 \text{ and } L_k = \lfloor \frac{L_{k-1}}{2} \rfloor. \tag{9}$$

This encoding converts the 'raw' ψ-bit vector into its dense multi-bit binary representation, effectively converting an integer into a binary number.

3. After repeating the above two steps L/w times, we concatenate all w-bit vectors $\{b_1(\mathbf{x}), b_2(\mathbf{x}), \ldots, b_{L/w}(\mathbf{x})\}$ to formulate a L-bit code for any point $\mathbf{x}$

$$\mathcal{B}_\mathbf{x} = [b_1(\mathbf{x}), \ldots, b_{L/w}(\mathbf{x})]. \tag{10}$$

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. In addition, following the mechanism of the Isolation Kernel (Ting, Zhu, et al. 2018), these points (sites) are obtained via uniform random sampling from the dataset D. This ensures that the resulting Voronoi diagram H is data-dependent, where the density of sites s_l naturally follows the data distribution, creating smaller Voronoi cells in dense regions and larger cells in sparse regions (Devroye et al. 2017; Qin et al. 2019; Ting, Washio, et al. 2024). This adaptive partitioning is the key to capturing the data structure without explicit learning. Although individual random partitions may exhibit variability, VDeH has the property that the number of independent samplings T is governed by the code length L, where $T = L/w$. In contrast, increasing bits in a traditional L2H method does not necessarily mean more independent distribution-aware partitions. VDeH's effectiveness and stability scale directly with L. Our experiments show that as the bit length increases, the ensemble of random partitions becomes more stable, and the binary representation provides a more effective approximation of the data structure.

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

Note that VDeH produces the Voronoi partitions *implicitly* via 1-nearest-neighbor search among the sampled sites, and no additional computation is required to build the Voronoi diagram explicitly (Ting, Zhu, et al. 2018). This ensures that the complexity remains linear with respect to the input dimensionality d. Furthermore, unlike traditional metrics that succumb to the curse of dimensionality, VDeH leverages the *distinguishability* of the isolation mechanism (Ting, Washio, et al. 2024), which allows it to effectively preserve similarity structures even in extremely high-dimensional spaces.

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 $\mathbf{d}_V(\mathcal{B}_\mathbf{x}, \mathcal{B}_\mathbf{y})$, between two binary codes $\mathcal{B}_\mathbf{x}$ and $\mathcal{B}_\mathbf{y}$ generated by VDeH:

$$\mathbf{d}_V(\mathcal{B}_\mathbf{x}, \mathcal{B}_\mathbf{y}) = \frac{w}{L} \sum_{k=1}^{L/w} \mathbb{1}(b_k(\mathbf{x}) \neq b_k(\mathbf{y})), \tag{11}$$

where $\mathcal{B}_\mathbf{x} = [b_1(\mathbf{x}), \ldots, b_{L/w}(\mathbf{x})]$ and $\mathcal{B}_\mathbf{y} = [b_1(\mathbf{y}), \ldots, b_{L/w}(\mathbf{y})]$.

This distance metric d_V possesses a theoretical connection with the Isolation Kernel (IK) (Qin et al. 2019; Ting, Zhu, et al. 2018). Specifically, as the code length L increases, the VDeH distance $\mathbf{d}_V$ converges to the complement of the IK similarity:

$$\lim_{L\to\infty} \mathbf{d}_V(\mathcal{B}_\mathbf{x}, \mathcal{B}_\mathbf{y}) = 1 - \kappa(\mathbf{x}, \mathbf{y}|\mathcal{D}), \tag{12}$$

where $\kappa(\mathbf{x}, \mathbf{y}|\mathcal{D})$ denotes the Isolation Kernel similarity defined by the probability that two points $\mathbf{x}$ and $\mathbf{y}$ fall into the same Voronoi cell. This equivalence demonstrates that VDeH effectively transforms the powerful discriminative capability of Isolation Kernels into an efficient binary representation, enabling high-speed similarity search while preserving the underlying data distribution's geometric structure.

Then, for any given query point $\mathbf{q}$, after computing its binary code via VDeH and its distance to every point in the dataset via Eq.(11), the retrieval process returns the top-ranked points with the shortest distances to $\mathbf{q}$.

Unlike standard Hamming distance which operates on individual bits, the VDeH distance is a block-wise metric. Each w-bit block $b_k(\mathbf{x})$ corresponds to the cell index in the k-th independent Voronoi diagram. The motivation for this design is to eliminate the unjustified bias inherent in binary indexing; a mismatch in cell assignment should contribute equally to the distance regardless of which specific bits in the binary representation of the index differ. As established in Eq.(12), this consistency allows VDeH to accurately preserve the geometric structure of the input space.

4.4 Independence among VDeH Bits

The importance of independence among hash bits are mentioned in existing studies (He, Chang, et al. 2011; Joly and Buisson 2011; Weiss et al. 2008). It can be defined as follows:

Definition 4.2. **(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:

$$Pr[h_1(\mathbf{x}) = v_1, h_2(\mathbf{x}) = v_2, \ldots, h_L(\mathbf{x}) = v_L] = Pr[h_1(\mathbf{x}) = v_1] \cdot Pr[h_2(\mathbf{x}) = v_2] \cdot \ \ldots \ \cdot Pr[h_L(\mathbf{x}) = v_L]. \tag{13}$$

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 Eq.(7), we prove that the encoded hashing achieves the independence among hash bits. VDeH generates mutually independent binary bits through the encoded hashing, given in Theorem 1 below.

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:

$$T \equiv \mathbf{s}_k \notin R(\mathbf{x}) \text{ for all } \mathbf{s}_k \in \mathcal{D}\ (k = 1, 2, \ldots, \psi), \text{ and} \tag{14}$$

$$Q \equiv \begin{cases} \mathbf{s}_i \in \Delta R(\mathbf{x}) \text{ for } \mathbf{s}_i \in \mathcal{D}, \text{ and} \\ \mathbf{s}_k \notin \Delta R(\mathbf{x}) \text{ for all } \mathbf{s}_k \in \mathcal{D}\ (k = 1, 2, \ldots, \psi, i \neq k). \end{cases} \tag{15}$$

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

$$Pr(T) = (1 - f)^\psi, \text{ and} \tag{16}$$

$$Pr(Q \mid T) = \frac{\Delta f}{1 - f} \left\{1 - \frac{\Delta f}{1 - f}\right\}^{\psi - 1}. \tag{17}$$

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 $i = 1, 2, \ldots, \psi$.

$$Pr(\mathbf{x} \in V_i) = \int_0^1 (1 - f)^\psi \frac{df}{1 - f} = \frac{1}{\psi}. \tag{18}$$

□

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 $b_l(\mathbf{x})$ is the w-bit vector generated by the l-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:

$$Pr(b_j = s_j, b_k = s_k) = Pr(b_j = s_j) \cdot Pr(b_k = s_k), \tag{19}$$

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

$$Pr(\alpha = v_\alpha, \beta = v_\beta) = Pr(\alpha = v_\alpha) \cdot Pr(\beta = v_\beta), \tag{20}$$

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 $\mathcal{B}_\mathbf{x}$ denote a set of ψ different w-bit binary codes correspond to the ψ distinct Voronoi cells.

When $\psi = 2^w$, $\mathcal{B}_\mathbf{x}$ 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

$$\begin{aligned} Pr(\alpha = v_\alpha, \beta = v_\beta) &= \sum_{i=1}^{\psi} (Pr(\mathbf{x} \in V_i)\mathbb{1}[\alpha = v_\alpha, \beta = v_\beta]) \\ &= \frac{1}{\psi} \sum_{i=1}^{\psi} \mathbb{1}[\alpha = v_\alpha, \beta = v_\beta] \\ &= \frac{\sum_{i=1}^{\psi} \mathbb{1}[\alpha = v_\alpha]}{\psi} \times \frac{\sum_{i=1}^{\psi} \mathbb{1}[\beta = v_\beta]}{\psi} \\ &= \sum_{i=1}^{\psi} (Pr(\mathbf{x} \in V_i)\mathbb{1}[\alpha = v_\alpha]) \times \sum_{i=1}^{\psi} (Pr(\mathbf{x} \in V_i)\mathbb{1}[\beta = v_\beta]) \\ &= Pr(\alpha = v_\alpha) \cdot Pr(\beta = v_\beta). \end{aligned} \tag{21}$$

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

4.5 Application to Complex Data Objects

As established in Section 2.2, a primary challenge in modern data retrieval is applying hashing techniques to complex data objects which usually cannot be measured based on Euclidean distance. A powerful and flexible approach to address this is a decoupled, two-stage process that separates the domain-specific representation learning from the generic hashing mechanism. This process can be formally represented as:

$$\mathcal{B}(o) = \mathcal{H}(\mathcal{F}(o)). \tag{22}$$

Here, a domain-specific embedding function $\mathcal{F}$ first maps a complex data object o into a vector representation $\mathcal{F}(o) \in \mathbb{R}^d$. Subsequently, a generic hashing module $\mathcal{H}$ converts this vector into a final binary code $\mathcal{B}(o)$.

Although $\mathcal{H}$ can be any L2H method, our proposed VDeH is an ideal candidate for this role. Its no-learning nature and high efficiency make it a truly plug-and-play component that can be seamlessly integrated with any existing or future embedding technique without requiring retraining or architectural redesign. Algorithm 1 details the complete procedure of this two-stage process, using VDeH as the specific implementation for the hashing module $\mathcal{H}$.

5 Experiment

This section provides an empirical evaluation to validate the effectiveness and efficiency of VDeH. Our evaluation has two parts. First, we compare VDeH against a suite of state-of-the-art L2H methods on standard vector databases to evaluate its performance. Second, we demonstrate its practical utility and plug-and-play nature by conducting extensive case studies on similarity search for two types of complex data objects, *i.e.*, graphs and trajectories databases.

5.1 Experimental Setup

5.1.1 Datasets. The datasets used are summarized as follows[2]:

- **Vector datasets.** To evaluate the proposed VDeH, we conduct experiments on four public vector datasets, including two image datasets and two text datasets. It is important to note that existing studies mainly

[2]Further details on the characteristics and preprocessing steps for all datasets are provided in Appendix A.

Algorithm 1 VDeH for Complex Data Objects Retrieval

Input: Database of complex data objects $\mathcal{U} = \{o_1, \ldots, o_N\}$, query object q, embedding function $\mathcal{F}$, VDeH parameters $\{L, \psi\}$.
Output: The top-k objects in $\mathcal{U}$ with the shortest VDeH distances to q.

 # Stage 1. Use embedding function $\mathcal{F}$ to embed each object $o \in \mathcal{U}$ into a vector $\mathbf{x} \in \mathcal{E}$
$\mathbf{x}_o \leftarrow \mathcal{F}(o),\ \forall o \in \mathcal{U}$;
$\mathbf{x}_q \leftarrow \mathcal{F}(q)$;
 # Stage 2. VDeH Encoding
$i = 0$
while i++ $< \lfloor L/\log_2 \psi \rfloor$ **do**
    Randomly sample ψ points from $\mathcal{E}$ to generate a Voronoi diagram having ψ Voronoi partitions;
    Compute binary code $b_i(\mathbf{x})$ for each embedded vector $\mathbf{x} \in \mathcal{E}$ via Eq.(8);
end while
Compute $\mathcal{B}_\mathbf{x} = [b_1(\mathbf{x}), \ldots, b_{\lfloor L/\log_2 \psi \rfloor}(\mathbf{x})]$ via Eq.(10) for all $\mathbf{x} \in \mathcal{E}$;
 # Query processing
Compute the distance between q and every object $o \in \mathcal{U}$ based on their VDeH codes via Eq.(11);
return the top-k objects in $\mathcal{U}$ with the shortest VDeH distances to q.

evaluate their performance on image datasets only (Gong, Kumar, et al. 2013; Gong, Lazebnik, et al. 2013; Heo et al. 2015; Liu et al. 2023; Weiss et al. 2008; Xia et al. 2015; Yu et al. 2014). Our work extends the evaluation to text datasets for a more comprehensive analysis. Unlike images which have obvious discriminating features that 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` (Krizhevsky 2009) contains 60,000 images (512 dimensions), `GIST` (Jégou et al. 2011) contains one million images represented by 960-dimensional descriptors, `Nytimes` (Pham and Liu 2022) includes 290,000 articles (256 dimensions), and `Kosarak` (FIMI 2017) includes 74,962 click-stream news (27,983 dimensions).

- **Graph datasets.** For graph retrieval, we use six public graph datasets of varying sizes and characteristics[3]. These include `Letter-med`, `COIL-RAG`, `COLORS-3`, and three chemical compound datasets `aspirin`, `salicylic_acid`, and `toluene`. These graph datasets are provided with ground-truth class labels, which are used for retrieval evaluation.
- **Trajectory datasets.** For trajectory retrieval, we use three commonly-used large-scale taxi trajectory datasets `T-Drive`, `Porto`, and `NYC`. `T-Drive` (Yuan et al. 2011) is a dataset collected from GPS-equipped taxis in Beijing, China, over a period of three months. `Porto` (Moreira-Matias et al. 2016) comprises the trajectories of taxis in Porto, Portugal, spanning from July 2013 to June 2014. `NYC` (Donovan and Work 2016) records taxi data in New York City, USA, over a four-year period, including pick-up and drop-off locations of passengers, fare information, *etc.* Moreover, `GeoLife` (Zheng et al. 2010) contains GPS records of human users and includes ground-truth labels for evaluating downstream clustering and anomaly detection tasks.

5.1.2 Baselines. For vector datasets retrieval, we compare VDeH against nine state-of-the-art L2H methods covering the three main categories, *i.e.*, (1) *thresholding-based*: Spectral Hashing (SH) (Weiss et al. 2008), Circulant Binary Embedding (CBE) (Yu et al. 2014), Iterative Quantization (ITQ) (Gong, Lazebnik, et al. 2013), Bilinear Projections (BP) (Gong, Kumar, et al. 2013), and Sparse Projections (SP) (Xia et al. 2015); (2) *hypersphere-based*: Spherical Hashing (SpH) (Heo et al. 2015); and (3) *hyperplane-based*: Density Sensitive Hashing (DSH) (Jin et al.

[3]All the datasets can be found in https://chrsmrrs.github.io/datasets/docs/datasets/.

2014), Sparse Embedding and Least Variance Encoding (SELVE) (Zhu, Zhang, et al. 2014), and refining codes for Locality Sensitive Hashing (rcLSH) (Liu et al. 2023).

For graph and trajectory datasets retrieval, we compare VDeH against representative methods in each of the three categories, *i.e.*, ITQ, SpH and rcLSH, as well as typical methods in their respective fields. As rcLSH faces out-of-memory execution issues when processing large datasets, we use its other version brLSH for evaluation (Liu et al. 2023). Specifically, for the graph data, we select Graph Matching Network (GMN) (Li, Gu, et al. 2019), which is a powerful learning-based model that generates full-precision embeddings for graph similarity computation, serving as a strong non-hashing baseline. For the trajectory data, we select no-learning-based Hausdorff distance (Hausdorff 1914) and learning-based t2vec (Li, Zhao, et al. 2018a) as the non-hashing baselines. Moreover, for trajectory downstream tasks, we include additional specialized trajectory embedding methods such as E2DTC (Fang et al. 2021) for trajectory clustering and TrajGAT (Yao et al. 2022) for trajectory anomaly detection.

5.1.3 *Evaluation Protocol.*

Metrics. For retrieval tasks, we use Mean Average Precision (mAP) and Precision-Recall (PR) curves as the primary evaluation metrics (Liu et al. 2023; Xia et al. 2015). For downstream trajectory tasks, we use Normalized Mutual Information (NMI) (Cover 1999) and Adjusted Rand Index (ARI) (Steinley 2004) for clustering (Fang et al. 2021; Wang, Zhu, et al. 2023), and the Area Under the Curve (AUC) for anomaly detection (Wang, Wang, et al. 2024). The mAP is the mean of the Average Precision (AP) scores over the set of all queries Q. The AP for a single query is defined as:

$$\text{AP} = \frac{\sum_{k=1}^{N}(P(k) \times \text{rel}(k))}{\text{Number of relevant items}},$$

where N is the number of items in the database, $P(k)$ is the precision at cut-off k, and $\text{rel}(k)$ is an indicator function defined based on ground truth. The final mAP is then calculated as:

$$\frac{1}{|Q|}\sum_{q\in Q}\text{AP}(q).$$

In our experiment, for different retrieval datasets, the relevant item for mAP calculation is tailored to each data category. For the vector datasets, relevance is determined by the nearest neighbors in the original Euclidean feature space. For the graph datasets, items are considered relevant if they share the same ground-truth class label. For the label-free trajectory datasets, ground truth is established by finding the nearest neighbors in the Euclidean space of their pre-computed embeddings. Detailed formulations for all other metrics are provided in Appendix B. Furthermore, to ensure the statistical reliability of our results, all experiments are executed ten times independently with different random seeds for the Voronoi partition samplings. All values reported in the tables and figures represent the average across these ten runs.

Methods for Two-Stage Evaluation. For the experiments on complex data objects, we instantiate the two-stage process described in Section 4.5. The choice of first-stage methods is crucial for a fair evaluation of the second-stage hashing modules. In the first stage, we choose the same embedding method for all methods to ensure fairness in the evaluation. Specifically, we use Graph Convolutional Network (GCN) (Kipf and Welling 2016), a widely-used deep learning baseline and Isolation Graph Kernel (IGK) (Xu, Ting, and Jiang 2021), a highly efficient no-learning method for graph embedding. For trajectory embedding, we use Isolation Distributional Kernel (IDK) (Shang et al. 2024; Wang, Wang, et al. 2024; Wang, Zhu, et al. 2023), chosen for its strong performance on multiple trajectory tasks. In addition, the downstream trajectory tasks are performed using standard algorithms: k-Medoids (Park and Jun 2009) and Spectral Clustering (SC) (Ng et al. 2001) for clustering, and k-Nearest Neighbors (kNN) (Brito et al. 1997) and Local Outlier Factor (LOF) (Breunig et al. 2000) for anomaly detection. Crucially, in all complex

data experiments, the first-stage embedding or downstream algorithm is held constant for a given task. We then apply VDeH and the relevant baselines to the exact same set of vectors. This setup ensures that our comparison fairly isolates and evaluates the performance of the hashing modules themselves.

Parameter Settings. Following existing studies on vector datasets retrieval (Heo et al. 2015; Jin et al. 2014; Liu et al. 2023), we use 10,000 randomly sampled instances for training and 500 instances from the remainder as queries. For all baseline L2H methods, we use the parameter settings recommended by their original authors. For the first-stage embedding methods, we adopted the default parameters for GCN (Kipf and Welling 2016), IGK (Xu, Ting, and Jiang 2021), and IDK (Wang, Wang, et al. 2024; Wang, Zhu, et al. 2023). The parameters for the downstream task algorithms, including k-medoids (Park and Jun 2009), Spectral Clustering (SC) (Ng et al. 2001), k-Nearest Neighbors (kNN) (Brito et al. 1997), and Local Outlier Factor (LOF) (Breunig et al. 2000), were tuned following the protocols in their corresponding studies. For our proposed VDeH, there are two main parameters as specified in Algorithm 1: the final hash code length L, which is varied according to the experimental setup (*e.g.*, 128, 256, etc.), and the number of Voronoi cells ψ. ψ is the key hyperparameter, which we tune from the set $\{2^2, 2^3, \ldots, 2^8\}$ on a validation set. The parameter w is determined by ψ as $w = \log_2 \psi$ and is not an independent parameter. All experiments are conducted on a Linux server with an AMD 128-core CPU and 1TB RAM.

5.2 Performance Evaluation on Vector Datasets

5.2.1 Comparing to the State-of-the-art of L2H on Vector Databases. Table 2 presents the mAP scores of our proposed VDeH and the competing hashing methods conducted on the four benchmark vector 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. Overall, DSH, RcLSH and SH are VDeH's closest contenders. Among existing methods, rcLSH and DSH exhibited the best performance on the image datasets; SH and DSH are the best on the text datasets.

The mAP of VDeH increases consistently as the number of hash bits increases. Notably, many methods did not show a continued increase in performance on the text datasets as the number of 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 regions and fewer in sparse regions; while VDeH naturally adapts because it creates smaller Voronoi cells in dense regions and larger ones in sparse regions (Devroye et al. 2017; Ting, Zhu, et al. 2018). Unlike the rigid, global cuts made by hyperplanes, these Voronoi partitions create flexible, local boundaries that more faithfully capture the underlying data manifold. VDeH's unique encoding scheme then represents these numerous, fine-grained partitions with a high information density per bit, allowing it to better preserve local similarities and achieve higher retrieval accuracy. In addition, Figure 2 provides the PR curves of VDeH and the competing hashing methods, when the number of hash bits is 512. We observe that the VDeH curve (in red) consistently dominates the curves of other methods across all datasets, demonstrating its superior precision and recall scores.

5.2.2 Comparing to the Deep Hashing Methods. Deep hashing is known for its high accuracy in semantic-based image retrieval (Luo et al. 2023). However, deep hashing methods cannot accurately retrieve images with high similarities, even after extracting semantic information from images. To show this, we use the CIFAR-10 and GIST datasets, which possess the semantic information in images. As initialization, we utilized ResNet18 to extract embeddings from each of these two datasets. Since the Euclidean distance between two embedded vectors reflect the semantic differences between their associated images, we used the Euclidean distances derived from these embedded vectors as ground truth to test the performance of VDeH and SP, as well as three deep hashing methods PairRec (Hansen et al. 2020), OASIS (Wu, Luo, et al. 2022) and ODH (He, Huang, et al. 2024).

Table 2. The mAP scores on the four vector datasets with different number of hash bits. The highest score in each row is marked with bold font. A non-increasing mAP score compared with that of any shorter hash code is underlined. Methods are categorized as Voronoi Diagram-based (VD-based), Hypersphere-based (HS-based), Hyperplane-based (HP-based), and Thresholding-based (Th-based).

Dataset	bits	VD-based	HS-based	HP-based			Th-based				
		VDeH	SpH	rcLSH	DSH	SELVE	SH	CBE	ITQ	BP	SP
CIFAR-10	128	**.366**	.242	.283	.318	.199	.117	.315	.355	.344	.352
	256	**.438**	.289	.351	.372	.176	.153	.342	.377	.343	.370
	512	**.468**	.304	.392	.401	.167	.193	.396	.389	.339	.397
	1024	**.501**	.314	.412	.429	.169	.201	.401	.398	.332	.407
	2048	**.525**	.321	.432	.457	.169	.214	.407	.401	.335	.426
GIST	128	**.664**	.222	.582	.501	.430	.338	.617	.644	.645	.604
	256	**.714**	.223	.631	.580	.399	.454	.644	.650	.641	.614
	512	**.763**	.232	.651	.618	.402	.501	.656	.654	.634	.638
	1024	**.774**	.233	.659	.610	.399	.514	.657	.657	.642	.638
	2048	**.793**	.242	.687	.621	.398	.532	.662	.663	.644	.641
Nytimes	128	**.116**	.100	.017	.016	.003	.033	.022	.023	.023	.022
	256	**.165**	.110	.026	.033	.004	.055	.029	.029	.029	.027
	512	**.340**	.146	.028	.185	.004	.160	.026	.030	.026	.029
	1024	**.348**	.088	.103	.339	.006	.295	.108	.104	.008	.106
	2048	**.376**	.089	.113	.326	.008	.310	.100	.097	.008	.100
Kosarak	128	**.457**	.256	.235	.368	.203	.375	.255	.261	.254	.258
	256	**.603**	.296	.271	.494	.226	.512	.286	.293	.295	.304
	512	**.607**	.346	.308	.437	.279	.571	.301	.298	.346	.341
	1024	**.615**	.372	.307	.415	.289	.569	.300	.302	.351	.357
	2048	**.638**	.381	.313	.412	.300	.579	.309	.312	.359	.369

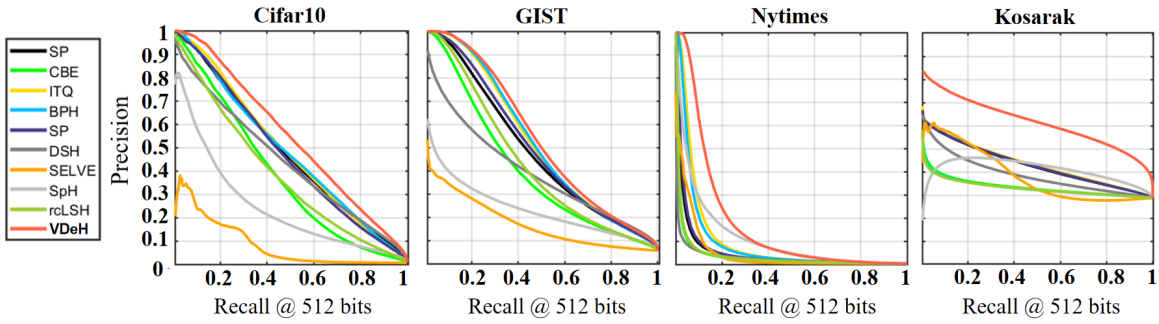
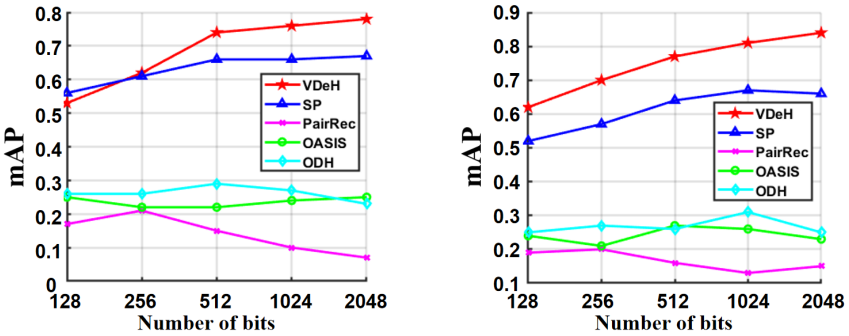


Fig. 2. PR curves on the four vector datasets with the 512-bit binary codes

The comparison results are shown in Figure 3. We observe that the representative L2H method SP and VDeH effectively preserve the Euclidean distances, whereas the deep hashing methods lack this capability. This is due to the fact that the deep hashing methods must have access to labels during the training process, aiming to



(a) Cifar10 (b) GIST

Fig. 3. Comparison of mAP scores for distance-preserving retrieval of deep hashing methods (PairRec, OASIS and ODH) and L2H methods (VDeH and SP) on CIFAR-10 and GIST. Ground truth is defined by Euclidean distance in the latent feature space.

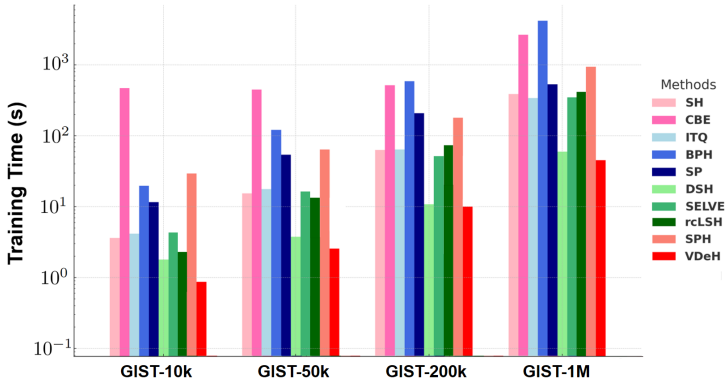


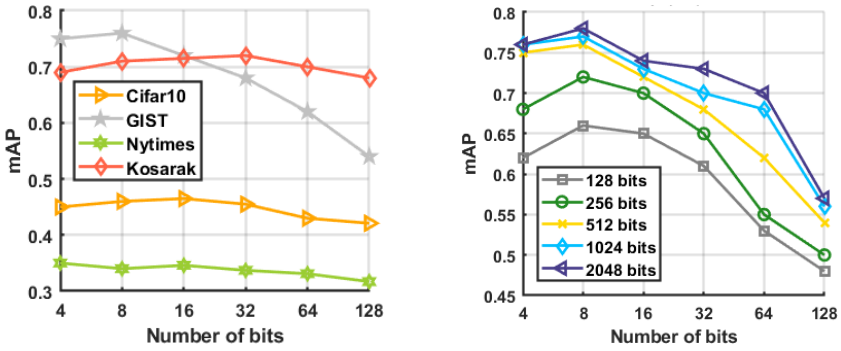
Fig. 4. Training time comparison on the largest GIST dataset with the 512-bit binary codes.

map instances with the same label into similar hash codes, and they are not designed to preserve the distances between instances in the input space.

5.2.3 Training Time Comparison. Unlike existing hashing methods that rely on optimization for learning hash functions, VDeH's training process is straightforward, requiring a small number of random samples only from the given dataset, where each sampled point corresponding to a nearest-neighbor-based hash function (as shown in Eq. 6). Note that all methods require the transformation of all the data points into binary codes via hash functions.

We tested the training time of various methods on the largest GIST dataset, shown in Figure 4. VDeH took the shortest 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.

5.2.4 Effects of Hyper-parameter ψ. We tested the robustness of VDeH to the core parameter ψ. Parameter ψ determines the number of partitions in each constructed Voronoi diagram. Conceptually, a larger ψ yields more partitions, more accurately reflecting the spatial relationships of instances. However, a larger ψ decreases the number of integrations for a given number of bits, making a properly tuned ψ crucial.



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

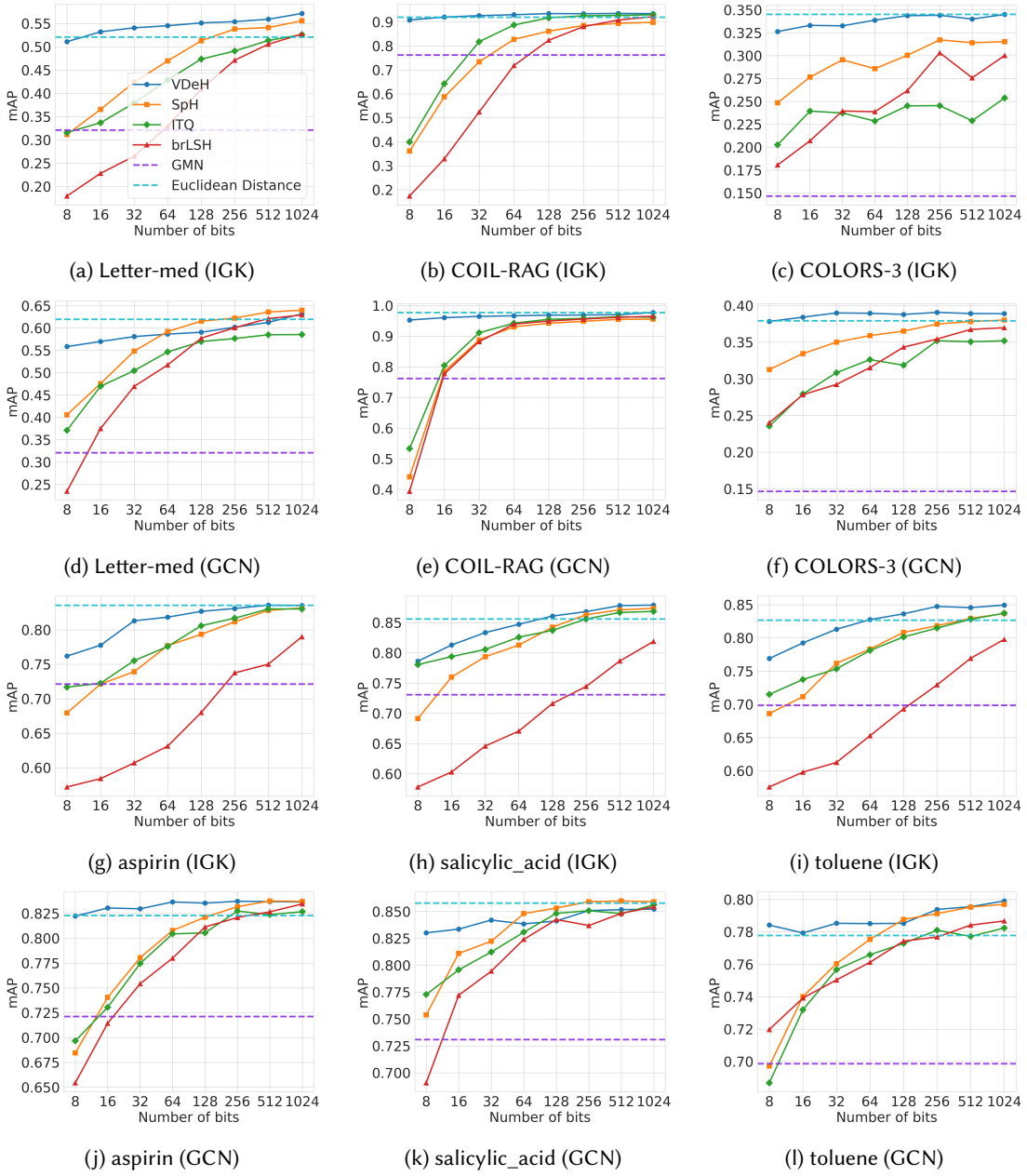
Fig. 5. The sensitivity of parameter ψ: (a) testing on different datasets with 512-bit binary codes; (b) Testing on the GIST dataset with different number of hash bits.

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 observe that the optimal ψ value remains the same regardless of the number of hash bits, ranging from 128 to 2048 bits. This suggests that the optimal ψ is intrinsic to the characteristics of the specific dataset, rather than the code length.

These observations further reveal a critical decoupling property between the hyperparameters ψ and L. As the optimal ψ is governed by the underlying data distribution and remains consistent across different code lengths L (as shown in Figure 5(b)), the parameter tuning process can be significantly simplified as follows. First, the optimal value of ψ can be identified using a short code length on a data subset, which minimizes the computational cost of the parameter sweep. Second, because retrieval performance improves predictably with the increase of L, the final code length can be determined solely based on the specific system constraints regarding storage and retrieval efficiency. Empirically, a search range of $\psi \in \{2^2, 2^3, \ldots, 2^6\}$ typically encompasses the optimal configuration for diverse datasets.

5.3 VDeH for Graph Retrieval

We first evaluate the performance of VDeH when applied to the task of large-scale graph retrieval. We adopt the two-stage process outlined in Section 4.5. In the first stage, we transform each graph into a vector embedding using two methods: the learning-based approach GCN (Kipf and Welling 2016) and the no-learning approach IGK (Xu, Ting, and Jiang 2021). In the second stage, we apply four L2H methods to these embeddings to generate binary codes for retrieval: our Voronoi diagram-based VDeH, the hypersphere-based SpH (Heo et al. 2015), the hyperplane-based brLSH (Liu et al. 2023) (used in place of rcLSH due to memory constraints on large datasets), and the thresholding-based ITQ (Gong, Lazebnik, et al. 2013). Furthermore, to provide a comprehensive evaluation, we include two fundamental baselines. The first is a direct Euclidean distance search on the embeddings, which represents the performance ceiling of the embedding method itself and serves as a benchmark for the effectiveness of the hashing stage. The second is the GMN (Li, Gu, et al. 2019), a state-of-the-art end-to-end graph similarity model, used to validate the competitiveness of our chosen first-stage embedding methods.



(a) Letter-med (IGK) (b) COIL-RAG (IGK) (c) COLORS-3 (IGK)

(d) Letter-med (GCN) (e) COIL-RAG (GCN) (f) COLORS-3 (GCN)

(g) aspirin (IGK) (h) salicylic_acid (IGK) (i) toluene (IGK)

(j) aspirin (GCN) (k) salicylic_acid (GCN) (l) toluene (GCN)

Fig. 6. Retrieval results on six graph datasets using two different first-stage embeddings.

Figure 6 shows the retrieval performance of VDeH and the baseline methods on the six graph datasets, using both IGK and GCN embeddings. From these results, we have the following key observations:

- The primary observation is the effectiveness of our two-stage framework. The mAP scores achieved by performing a direct Euclidean distance search on both the IGK and GCN embeddings (the light blue dashed line) consistently and substantially outperform the end-to-end GMN baseline (the purple dashed line) across all datasets. This result validates our choice of the first-stage embedding methods, confirming that they generate high-quality vector representations that effectively capture the structural similarities of the graphs, forming a strong foundation for the subsequent hashing stage.
- When comparing among different hashing methods, VDeH demonstrates superior performance over other L2H baselines in the majority of cases. In particular, its significant advantage is pronounced at shorter code lengths. On nearly every dataset, VDeH establishes a clear performance gap over its competitors with these compact codes. In contrast, other methods typically yield significantly poorer results with shorter codes, indicating a weakness in their information encoding process for complex graph embeddings. This highlights that VDeH's Voronoi-based partitioning and encoding scheme is highly effective, enabling it to generate discriminative binary codes that preserve crucial structural information even under tight storage constraints. Among the competitors, SpH emerges as the closest contender, though it often requires longer codes to approach VDeH's accuracy.
- We observe an intriguing phenomenon regarding the performance ceiling set by the Euclidean distance baseline. In several datasets (*e.g.*, on the `aspirin` and `toluene` datasets with GCN embeddings), the mAP results of VDeH and other L2H methods surpass the Euclidean distance baseline as the code length increases. This demonstrates that the hashing process can act as more than just an approximation, *i.e.*, it is a form of representation refinement. By mapping the continuous vectors to a structured binary space, VDeH performs a non-linear transformation that regularizes the feature space, effectively denoising the initial embeddings and capturing the underlying data manifold more robustly than the original L_2 norm. Conversely, in other scenarios where the L2H performance converges below this baseline, it aligns with the conventional understanding of hashing as an approximation method, where a degree of information loss is the inherent trade-off for the significant gains in retrieval efficiency.

5.4 VDeH for Trajectory Mining Tasks

We further evaluates the performance of VDeH when applied to the task of large-scale trajectory retrieval and mining. We continue to employ the two-stage framework, where in the first stage, we use the Isolation Distributional Kernel (IDK) (Wang, Wang, et al. 2024; Wang, Zhu, et al. 2023) to generate the trajectory embeddings. In the second stage, we apply the same four representative L2H methods, *i.e.*, VDeH, SpH, brLSH, and ITQ to convert these embeddings into binary codes. Our evaluation is multifaceted. First, we assess the retrieval accuracy of VDeH against the other L2H baselines. Second, we analyze the retrieval efficiency of our two-stage approach, benchmarking it against both first-stage embedding methods, *i.e.*, using Euclidean distance on IDK embeddings, and traditional Hausdorff distance. Finally, we demonstrate the practical effectiveness of the IDK+VDeH combination by evaluating its performance on two common downstream trajectory mining tasks: clustering and anomaly detection.

5.4.1 Trajectory Retrieval. We first evaluate the performance of VDeH on large-scale trajectory retrieval, a task where ground-truth class labels are typically unavailable. The primary objective of this experiment is to assess the extent to which different L2H methods can preserve the original distance relationships established by the first-stage embedding. A high-fidelity binary representation should yield a ranking of nearest neighbors that is highly consistent with the ranking produced by a direct Euclidean distance search on the initial embeddings. Table 3 shows the mAP scores for VDeH and three competing L2H methods on three large-scale trajectory datasets: `T-Drive`, `Porto`, and `NYC`. For this evaluation, each ground truth of a query is established by its nearest neighbor as measured by the Euclidean distance on the IDK-generated embeddings.

Table 3. mAP (%) on the three large-scale real-world trajectory datasets. The same IDK embedded vectors are used for all hashing methods, and we use the outcome of Euclidean distance on the IDK embedded vectors as the ground truth.

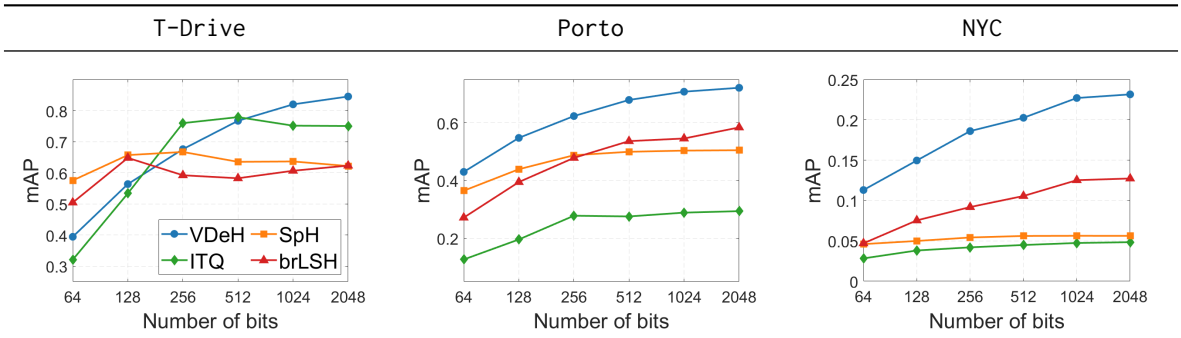


Table 4. Time cost for retrieving the 1,000 nearest neighbors of a query trajectory, average over 100 queries. The reported time includes both the conversion of the query trajectory and the subsequent retrieval process. The objects in each dataset are required to be converted only once, and this time is not included since it is pre-computed offline and does not affect the online retrieval time. The slowdown ratio for other methods is calculated against the VDeH (2048 bits) performance.

Mapping	Hashing/Distance	T-Drive	Porto	NYC
IDK	**VDeH (2048 bits)**	**33ms**	**77ms**	**539ms**
	Euclidean Distance	6s (∼182**x** ↓)	12s (∼156**x** ↓)	91s (∼169**x** ↓)
t2vec	Euclidean Distance	6s (∼182**x** ↓)	13s (∼169**x** ↓)	92s (∼171**x** ↓)
None	Hausdorff Distance	41s (∼1242**x** ↓)	192s (∼2494**x** ↓)	714s (∼1325**x** ↓)

The results demonstrate that VDeH shows the best overall performance, with its advantage becoming more pronounced at longer code lengths, where it surpasses all competitors on all datasets. On the Porto and NYC datasets, VDeH establishes a significant performance margin over all competitors. Furthermore, its mAP score demonstrates strong and consistent growth with the increase in code length from 64 to 2048 bits. This scalability highlights VDeH's ability to effectively leverage additional bits to create a more faithful and accurate binary approximation of the original embedding space. In contrast, the performance of the baseline methods is less robust. Interestingly, ITQ shows competitive performance on the T-Drive dataset, even slightly surpassing VDeH at 256 and 512 bits. However, its effectiveness sharply declines on the other two, larger datasets, suggesting that its performance is highly sensitive to the specific data distribution.

In addition, to quantify the retrieval speed advantage of mapping trajectories to binary codes within the two-stage framework, we measured the online query time on the three large-scale datasets. Table 4 reports the time required to retrieve the 1,000 nearest neighbors for a query trajectory, averaged over 100 queries. We benchmark the performance of VDeH (applied to IDK embeddings) against three non-hashing baselines: a direct Euclidean distance search on the same IDK embeddings, a Euclidean search on embeddings from the learning-based t2vec model, and the traditional, metric-based Hausdorff distance.

The results demonstrate the significant efficiency of VDeH, positioning it as a high-performance benchmark for retrieval. The process using VDeH's binary codes is consistently two to three orders of magnitude more efficient than baselines that operate on full-precision vectors or raw trajectories. For example, on the largest

Table 5. Clustering results in terms of NMI and ARI on the GeoLife dataset (Wang, Zhu, et al. 2023). The embedding times of IDK, t2vec, TrajGAT, and VDeH, as well as the pretraining time of E^{2}DTC, are included in the time presented in the table. Only TrajGAT (Yao et al. 2022) and E^{2}DTC (Fang et al. 2021) run on an NVIDIA RTX A6000 GPU, others use CPU. The best and the second-best results are in **bold** and underline, respectively. The down-arrow symbol "↓" denotes a degradation in computational performance (*i.e.*, a slower execution time) relative to the IDK+VDeH combination.

Clustering Method	Measure	NMI (%)	ARI (%)	Time (s)
Spectral Clustering (Ng et al. 2001)	IDK+VDeH	99.58	99.60	**11.2**
	IDK	**99.97**	**99.98**	32.9 (∼3**x** ↓)
	t2vec	3.85	0.01	247.4 (∼22**x** ↓)
	TrajGAT	72.56	36.36	460.2 (∼41**x** ↓)
	Hausdorff Distance	89.31	69.55	4106.7 (∼367**x** ↓)
k-Medoids Clustering (Park and Jun 2009)	IDK+VDeH	**88.73**	**71.30**	**9.1**
	IDK	88.19	67.65	24.8 (∼3**x** ↓)
	t2vec	32.79	11.49	238.1 (∼26**x** ↓)
	TrajGAT	75.20	55.30	510.6 (∼56**x** ↓)
	Hausdorff Distance	86.21	68.40	51964.0 (∼5710**x** ↓)
E^{2}DTC (Fang et al. 2021)		68.61	56.39	1705.9 (∼188**x** ↓)*

*Slowdown is calculated against the faster IDK+VDeH baseline (k-Medoids).

dataset, NYC, retrieving neighbors with a 2048-bit VDeH code takes only 539ms. In contrast, a direct search on the IDK embeddings is approximately 169**x** slower, requiring 91 seconds, while the computationally intensive Hausdorff distance is over 1,300**x** slower, taking 714 seconds. This substantial efficiency gain underscores the practical utility of VDeH as a plug-and-play component for building high-speed similarity search systems for complex data objects.

5.4.2 Trajectory Clustering and Anomaly Detection. Beyond similarity search, we further investigate the practical utility and scalability of the two-stage framework on common downstream trajectory mining tasks, *i.e.*, clustering and anomaly detection. For structured data such as trajectories, traditional mining algorithms often rely on a pairwise similarity matrix. While effective, computing this matrix using conventional distance metrics can incur prohibitive computational costs, creating a significant bottleneck for large-scale analysis. Our approach explores whether the binary codes generated by VDeH can serve as efficient surrogates for the original high-dimensional embeddings. The central hypothesis is that by leveraging these compact binary representations and a fast bitwise distance computation, we can substantially accelerate the execution of these downstream tasks. The key objective is to achieve these significant efficiency gains while preserving a high-quality outcome, ensuring that the performance on clustering and anomaly detection is not compromised. To this end, we employ the GeoLife dataset (Zheng et al. 2010), a standard benchmark widely used for such tasks due to the availability of ground-truth labels, which enables a rigorous quantitative evaluation of performance (Wang, Wang, et al. 2024; Wang, Zhu, et al. 2023).

Table 5 details the results of the trajectory clustering for different approaches with the following observations:

- First, the results demonstrate the superiority of the IDK embedding for trajectory representation. As shown in the table, the IDK measure (representing the IDK embedding with a Euclidean distance search) achieves the highest clustering accuracy among all non-hashing methods. For both Spectral Clustering and k-Medoids Clustering, its NMI and ARI scores are substantially higher than those from the learning-based embeddings, *i.e.*, t2vec and TrajGAT, the traditional Hausdorff Distance, and the specialized end-to-end

Table 6. Anomaly detection results in terms of AUC on the GeoLife dataset (Wang, Wang, et al. 2024).

Anomaly Detection Method	Measure	AUC (%)	Time (s)
k-Nearest Neighbors (Knox and Ng 1998)	IDK+VDeH	99.80	**381**
	IDK	98.31	4123 (∼11**x** ↓)
	t2vec	70.36	3098 (∼8**x** ↓)
	TrajGAT	81.43	6197 (∼16**x** ↓)
	Hausdorff Distance	**99.83**	535708 (∼1406**x** ↓)
Local Outlier Factor (Breunig et al. 2000)	IDK+VDeH	99.80	**473**
	IDK	**99.98**	4605 (∼10**x** ↓)
	t2vec	62.56	3219 (∼7**x** ↓)
	TrajGAT	83.72	6482 (∼14**x** ↓)
	Hausdorff Distance	99.89	553958 (∼1171**x** ↓)

model, E2DTC. Furthermore, it is also the most computationally efficient among these baselines, setting a strong benchmark for both accuracy and speed.

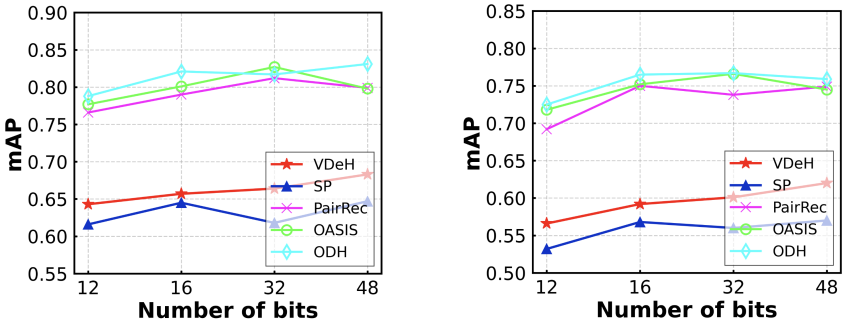
- Second, when VDeH is applied to the IDK embeddings, the resulting binary codes preserve this high representation quality while clearly improving efficiency. The performance of the IDK+VDeH approach is highly comparable to the IDK baseline. For Spectral Clustering, it achieves nearly identical, second-best scores. For k-Medoids Clustering, it even surpasses the baseline in both metrics, achieving the best results overall (**88.73%** NMI and **71.30%** ARI). The crucial advantage lies in the execution time, which is reduced by nearly a factor of three (*e.g.*, from 32.9s to **11.2s** for Spectral Clustering). This speedup is achieved because the numerous distance calculations in clustering algorithms are transformed from expensive floating-point Euclidean operations into highly efficient bitwise operations on compact binary codes.
- Third, these results highlight the disadvantages of existing approaches when compared to the two-stage IDK+VDeH framework. The learning-based embeddings from t2vec and TrajGAT yield poor clustering quality, justifying our selection of IDK as a more effective first-stage embedding method. Conversely, the traditional Hausdorff Distance, while providing reasonable accuracy, is rendered impractical by its extremely high computational cost, running three to four orders of magnitude slower than our framework. In addition, the end-to-end model E2DTC demonstrates significant deficiencies in both clustering accuracy and runtime efficiency, failing to offer a competitive solution.

The findings from the anomaly detection tasks, presented in Table 6, reinforce the conclusions drawn from the clustering results. Applying VDeH to the IDK embeddings again provides a substantial efficiency improvement, accelerating both k-Nearest Neighbors and Local Outlier Factor detection by approximately an order of magnitude. Crucially, this acceleration is achieved without sacrificing performance; the resulting AUC is comparable to the IDK baseline. In addition, while the traditional Hausdorff distance achieves good accuracy, its computational cost is prohibitively high. In contrast, the two-stage framework using VDeH generates comparable AUC scores but is over three orders of magnitude faster than Hausdorff distance, showcasing the applicability of VDeH for building practical and efficient solutions for large-scale trajectory mining.

6 Discussion

6.1 Distance Preservation vs. Semantic Awareness

A primary limitation of VDeH, as a distance-preserving hashing method, is its inability to explicitly perceive high-level semantic similarities that are not already captured within the input feature space. As VDeH is a



(a) MS-COCO (b) NUS-WIDE

Fig. 7. Comparison of mAP scores for semantic-based retrieval of deep hashing methods (PairRec, OASIS and ODH) and L2H methods (VDeH and SP) on MS-COCO and NUS-WIDE. Ground truth is defined by high-level categorical labels.

learning-free approach, its retrieval effectiveness is fundamentally constrained by the representational power of the upstream embeddings. To illustrate this boundary, we evaluated VDeH on common deep hashing benchmarks, including MS-COCO and NUS-WIDE image datasets (Hansen et al. 2020; He, Huang, et al. 2024; Wu, Luo, et al. 2022), where relevance is defined by categorical labels rather than geometric proximity. As shown in Figure 7, there is a significant performance gap between distance-preserving methods (VDeH, SP) and supervised deep hashing methods (ODH, OASIS, PairRec). The latter can leverage label information during training to warp the binary space toward semantic clusters.

Interestingly, despite the lower absolute mAP scores in these semantic tasks, VDeH maintains a stable performance gain as the bit length increases. In contrast, deep hashing methods exhibit noticeable oscillations across different bit configurations. This underscores that while VDeH is a general tool for distance-preserving retrieval, it is not intended to replace task-specific semantic hashing when high-level categorical labels are the primary retrieval target and are accessible for supervision.

6.2 Integration with Approximate Nearest Neighbor Indexing Structures

A critical distinction must be made between VDeH and industrial indexing structures like IVF, HNSW, and IVFPQ. VDeH is a distance-preserving binary encoding method focused on representation, whereas the others are search-space partitioning structures focused on pruning the candidate set. In fact, VDeH can be viewed as a plug-and-play component that serves as an efficient distance measure within these frameworks. In this complementary hierarchy, the indexing structure handles search-space pruning, while VDeH provides a highly compressed binary representation to accelerate kernel or distance calculations via bitwise operations. This efficiency is evidenced in our trajectory retrieval experiments shown in Table 4, where VDeH achieves a 150**x** speedup over Euclidean distance-based brute-force search.

This modularity has been further validated in recent extensions of this work (Zhibo et al. 2026). By generalizing the isolation mechanism and integrating it with advanced industrial indices like HNSW, it is possible to achieve up to 5**x** higher throughput and over 10**x** memory reduction compared to standard dense embeddings on million-scale text retrieval benchmarks. These results confirm that VDeH provides a superior foundation for downstream indexing algorithms by transforming high-dimensional vector spaces into computationally efficient binary manifolds.

6.3 Sensitivity to Embedding Quality

For structured data such as graphs and trajectories, VDeH is typically applied in a pipeline where an effective embedding encoder is first utilized to map the raw data into a continuous feature space. A critical consideration is the sensitivity of the hashing performance to the choice and quality of these upstream embeddings. We clarify that while the final retrieval accuracy is naturally constrained by the representational power of the chosen encoder, which serves as a performance ceiling, VDeH is designed to be an approximator of that specific input space. The robustness of VDeH comes from its data-dependent partitioning mechanism. By sampling Voronoi sites directly from the input distribution D, the hashing scheme adaptively captures the local density and geometric structure of the specific embedding manifold. This allows VDeH to maintain high fidelity regardless of whether the input is a learned representation or a non-learning one. As demonstrated in Figure 6, while absolute scores vary across different embedding methods, VDeH consistently exhibits a stable and predictable performance gain as the bit length L increases, effectively converging toward the optimal retrieval performance achievable within each respective space. This modularity ensures that VDeH is compatible with evolving representation learning techniques. Rather than being tied to a specific embedding methodology, it can serve as a universal, no-learning plug-and-play component that seamlessly inherits the benefits of advancements in feature extraction without requiring model retraining or structural modifications.

6.4 Robustness for Dimensionality and Noise

VDeH is fundamentally based on the Isolation Kernel (IK), which has been shown to overcome the curse of dimensionality by providing *distinguishability* in high-dimensional manifolds (Ting, Washio, et al. 2024). Unlike Lipschitz continuous kernels, the feature map of IK can effectively differentiate distinct points regardless of the input or intrinsic dimensionality of the data space. This theoretical advantage is reflected in our experiments, where VDeH yields superior and stable performance even on datasets with more than thousands of dimensions. Regarding noise sensitivity, VDeH maintains a locality-preserving property where the likelihood of a boundary crossing (and thus a bit-flip) remains low for minor input perturbations, which is similar to locality-sensitive hashing (Charikar 2002).

6.5 Empirical Evidence of Bit Independence

The bit independence property is evidenced by the convergence behavior of VDeH. In ensemble learning theory, when weak estimators are independent of each other, increasing the number of estimators can lead to an improvement in the accuracy of the final estimate (Breiman 2001). Our experimental results across all datasets consistently show that the retrieval precision improves with increasing of the number of bits L.

The absence of performance saturation even at high bit counts indicates that the information redundancy between different hashing blocks is minimal. This confirms that the independent random sampling of Voronoi seeds effectively explores diverse regions of the input distribution, allowing each bit segment to capture unique structural characteristics. Such empirical stability ensures that VDeH can reliably scale its bit length to achieve higher distance-preservation effectiveness (Ting, Washio, et al. 2024).

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 that 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 based

on a similar definition, as far as we know. Third, the encoded hashing enables VDeH to generate mutually independent binary bits. This encoding is a key advantage, as it allows a short w-bit code to represent a selection of up to 2^w distinct Voronoi partitions. This mechanism significantly increases the information content per bit, empowering VDeH to capture finer-grained data structures and achieve superior accuracy for a given code length. Our experiments show that VDeH exhibits superior performance with lower computational cost compared to the state-of-the-art methods.

Furthermore, our extensive experiments on complex data objects of graphs and trajectories validate that VDeH is a versatile and effective plug-and-play component. When integrated into a two-stage framework, VDeH is compatible with various embedding methods, whether learning-based or not. It not only accelerates similarity retrieval by orders of magnitude but also enhances downstream mining tasks such as clustering and anomaly detection, achieving state-of-the-art performance with a fraction of the computational cost of traditional methods.

In the near future, we will explore integrating this no-learning, plug-and-play hashing method with a wide array of data types and application domains where efficient similarity search remains a critical challenge, potentially enhancing the performance of various specialized tasks.

Acknowledgments

The authors would like to thank the anonymous reviewers and the editor for their valuable comments.

An earlier version of this work, titled "Voronoi Diagram Encoded Hashing", has been accepted for publication in the Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) 2025 (Xu and Ting 2025). This journal article is a substantial extension of the work that will appear in the conference proceedings. While the conference version introduces the core VDeH method and its evaluation on vector databases, this article significantly expands upon it by: (1) proposing a general two-stage framework for applying VDeH to complex data objects; (2) presenting new, comprehensive experiments on large-scale graph and trajectory databases; and (3) demonstrating VDeH's practical utility on downstream tasks, including clustering and anomaly detection.

Kai Ming Ting is supported by the National Natural Science Foundation of China (Grant No. 92470116 & W2531050). This project is also supported by Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118).

References

A. Borthwick, S. Ash, B. Pang, and S. Qureshi. 2021. "Scalable Blocking for Very Large." In: *Proceedings of the ECML PKDD Workshops*. Vol. 1323. Springer Nature, 303.

L. Breiman. 2001. "Random forests." *Machine learning*, 45, 1, 5–32.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. 2000. "LOF: identifying density-based local outliers." In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 93–104.

M. R. Brito, E. L. Chávez, A. J. Quiroz, and J. E. Yukich. 1997. "Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection." *Statistics & Probability Letters*, 35, 1, 33–42.

D. Cai. 2021. "A revisit of hashing algorithms for approximate nearest neighbor search." *IEEE Transactions on Knowledge and Data Engineering*, 33, 6, 2337–2348.

Y. Chang, E. Tanin, G. Cong, C. S. Jensen, and J. Qi. 2024. "Trajectory Similarity Measurement: An Efficiency Perspective." *Proceedings of the VLDB Endowment*, 17, 9, 2293–2306.

M. S. Charikar. 2002. "Similarity Estimation Techniques from Rounding Algorithms." In: *Proceedings of the ACM Symposium on Theory of Computing*, 380–388.

L. Chi and X. Zhu. 2017. "Hashing techniques: A survey and taxonomy." *ACM Computing Surveys (Csur)*, 50, 1, 1–36.

T. M. Cover. 1999. *Elements of information theory*. John Wiley & Sons.

M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. 2004. "Locality-sensitive hashing scheme based on p-stable distributions." In: *Proceedings of the Annual Symposium on Computational Geometry*, 253–262.

L. Deng, Y. Zhao, J. Chen, S. Liu, Y. Xia, and K. Zheng. 2024. "Learning to hash for trajectory similarity computation and search." In: *Proceedings of the IEEE International Conference on Data Engineering*. IEEE, 4491–4503.

L. Devroye, L. Györfi, G. Lugosi, and H. Walk. 2017. "On the measure of Voronoi cells." *Journal of Applied Probability*, 54, 2, 394–408.
B. Donovan and D. Work. 2016. *New York City Taxi Trip Data (2010-2013)*. https://doi.org/10.13012/J8PN93H8.
Z. Fang, Y. Du, L. Chen, Y. Hu, Y. Gao, and G. Chen. 2021. "E^2DTC: An end to end deep trajectory clustering framework via self-training." In: *Proceedings of the IEEE International Conference on Data Engineering*. IEEE, 696–707.
FIMI. 2017. "FIMI datasets." In: *http://fimi.ua.ac.be/data/. Accessed 2 Jan 2017.*
X. Gao, B. Xiao, D. Tao, and X. Li. 2010. "A survey of graph edit distance." *Pattern Analysis and applications*, 13, 1, 113–129.
Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. 2013. "Learning binary codes for high-dimensional data using bilinear projections." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. 2013. "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.
C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma. 2020. "Unsupervised Semantic Hashing with Pairwise Reconstruction." In: *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.
I. U. Haq and J. Caballero. 2021. "A survey of binary code similarity." *Acm computing surveys (csur)*, 54, 3, 1–38.
F. Hausdorff. 1914. *Grundzüge der Mengenlehre*. Veit & Comp, Leipzig.
J. He, S.-F. Chang, R. Radhakrishnan, and C. Bauer. 2011. "Compact hashing with joint optimization of search accuracy and time." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 753–760.
L. He, Z. Huang, C. Liu, R. Li, R. Wu, Q. Liu, and E. Chen. 2024. "One-bit Deep Hashing: Towards Resource-Efficient Hashing Model with Binary Neural Network." In: *Proceedings of the ACM International Conference on Multimedia*, 7162–7171.
J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. 2015. "Spherical hashing: Binary code embedding with hyperspheres." *IEEE transactions on pattern analysis and machine intelligence*, 37, 11, 2304–2316.
D. Hu, Z. Chen, J. Wu, J. Sun, and H. Chen. 2021. "Persistent memory hash indexes: An experimental evaluation." *Proceedings of the VLDB Endowment*, 14, 5, 785–798.
H. Jégou, M. Douze, and C. Schmid. 2011. "Product Quantization for Nearest Neighbor Search." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 117–128.
Z. Jin, C. Li, Y. Lin, and D. Cai. 2014. "Density sensitive hashing." *IEEE Transactions on Cybernetics*, 44, 8, 1362–1371.
A. Joly and O. Buisson. 2011. "Random Maximum Margin Hashing." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
S. Khoshraftar and A. An. 2024. "A survey on graph representation learning methods." *ACM Transactions on Intelligent Systems and Technology*, 15, 1, 1–55.
T. N. Kipf and M. Welling. 2016. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907*.
E. M. Knox and R. T. Ng. 1998. "Algorithms for mining distancebased outliers in large datasets." In: *Proceedings of the International Conference on Very Large Data Bases*. Citeseer, 392–403.
A. Krizhevsky. 2009. "Learning multiple layers of features from tiny images." *Technical Report from the Department of Computer Science at the University of Toronto*.
X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei. 2018a. "Deep representation learning for trajectory similarity computation." In: *Proceedings of the IEEE International Conference on Data Engineering*. IEEE, 617–628.
X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei. 2018b. "Deep Representation Learning for Trajectory Similarity Computation." In: *Proceedings of the IEEE International Conference on Data Engineering*, 617–628.
Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli. 2019. "Graph matching networks for learning the similarity of graph structured objects." In: *Proceedings of the International Conference on Machine Learning*. PMLR, 3835–3845.
Y. Li and J. van Gemert. 2021. "Deep Unsupervised Image Hashing by Maximizing Bit Entropy." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 3. Vol. 35, 2002–2010.
H. Liu, W. H. Zhou, Z. Wu, S. C. Zhang, G. Li, and X. L. Li. 2023. "Refining Codes for Locality Sensitive Hashing." *IEEE Transactions on Knowledge and Data Engineering*, 1–11.
X. Luo, H. Wang, D. Wu, C. Chen, M. Deng, J. Huang, and X.-S. Hua. 2022. "A Survey on Deep Hashing Methods." *ACM Transactions on Knowledge Discovery from Data*, 17, 1.
X. Luo, H. Wang, D. Wu, C. Chen, M. Deng, J. Huang, and X.-S. Hua. 2023. "A survey on deep hashing methods." *ACM Transactions on Knowledge Discovery from Data*, 17, 1, 1–50.
A. Maćkiewicz and W. Ratajczak. 1993. "Principal components analysis (PCA)." *Computers & Geosciences*, 19, 3, 303–342.
J.-Y. Mao, Q.-K. Pan, Z.-H. Miao, L. Gao, and S. Chen. 2022. "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.
X. Min, K. Lu, P. Liu, J. Wan, C. Xie, D. Wang, T. Yao, and H. Wu. 2024. "SepHash: A Write-Optimized Hash Index On Disaggregated Memory via Separate Segment Structure." *Proceedings of the VLDB Endowment*, 17, 5, 1091–1104.
L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas. 2016. "Time-evolving OD matrix estimation using high-speed GPS data streams." *Expert Systems with Applications*, 44, 275–288.

C. Myers, L. Rabiner, and A. Rosenberg. 1980. "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28, 6, 623–635. doi:10.1109/TASSP.1980.1163491.

A. Ng, M. Jordan, and Y. Weiss. 2001. "On spectral clustering: Analysis and an algorithm." *Advances in Neural Information Processing systems*, 14.

Z. A. Al-Odat, M. Ali, A. Abbas, and S. U. Khan. 2020. "Secure hash algorithms and the corresponding FPGA optimization techniques." *ACM Computing Surveys (CSUR)*, 53, 5, 1–36.

H.-S. Park and C.-H. Jun. 2009. "A simple and fast algorithm for K-medoids clustering." *Expert Systems with Applications*, 36, 2, 3336–3341.

N. Pham and T. Liu. 2022. "Falconn++: A Locality-sensitive Filtering Approach for Approximate Nearest Neighbor Search." In: *Proceedings of the Annual Conference on Neural Information Processing Systems*.

X. Qin, K. M. Ting, Y. Zhu, and V. C. Lee. 2019. "Nearest-neighbour-induced isolation similarity and its impact on density-based clustering." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 01. Vol. 33, 4755–4762.

Y. Shang, K. M. Ting, Z. Wang, and Y. Wang. 2024. "Distributional Kernel: An Effective and Efficient Means for Trajectory Retrieval." In: *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 271–283.

A. Singh and S. Gupta. 2022. "Learning to hash: a comprehensive survey of deep learning-based hashing methods." *Knowledge and Information Systems*, 64, 10, 2565–2597.

D. Steinley. 2004. "Properties of the hubert-arable adjusted rand index." *Psychological methods*, 9, 3, 386.

M. Thangavel and P. Varalakshmi. 2019. "Enabling ternary hash tree based integrity verification for secure cloud data storage." *IEEE Transactions on Knowledge and Data Engineering*, 32, 12, 2351–2362.

K. M. Ting, T. Washio, Y. Zhu, Y. Xu, and K. Zhang. 2024. "Is it possible to find the single nearest neighbor of a query in high dimensions?" *Artificial Intelligence*, 336, 104206.

K. M. Ting, Y. Zhu, and Z.-H. Zhou. 2018. "Isolation Kernel and Its Effect on SVM." In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

J. Wang, S. Xu, F. Zheng, K. Lu, J. Song, and L. Shao. 2021. "Learning efficient hash codes for fast graph-based data similarity retrieval." *IEEE Transactions on Image Processing*, 30, 6321–6334.

J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. 2017. "A Survey on Learning to Hash." *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Y. Wang, Z. Wang, K. M. Ting, and Y. Shang. 2024. "A principled distributional approach to trajectory similarity measurement and its application to anomaly detection." *Journal of Artificial Intelligence Research*, 79, 865–893.

Z. J. Wang, Y. Zhu, and K. M. Ting. 2023. "Distribution-Based Trajectory Clustering." In: *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 1379–1384.

A. Ward. 1954. "A generalization of the Frechet distance of two curves." *Proceedings of the National Academy of Sciences*, 40, 7, 598–602.

Y. Weiss, A. Torralba, and R. Fergus. 2008. "Spectral hashing." In: *Proceedings of the Annual Conference on Neural Information Processing Systems*, 1753–1760.

X.-M. Wu, X. Luo, Y.-W. Zhan, C.-L. Ding, Z.-D. Chen, and X.-S. Xu. 2022. "Online Enhanced Semantic Hashing: Towards Effective and Efficient Retrieval for Streaming Multi-Modal Data." In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 4263–4271.

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. 2020. "A comprehensive survey on graph neural networks." *IEEE Transactions on Neural Networks and Learning Systems*, 32, 1, 4–24.

Y. Xia, K. He, P. Kohli, and J. Sun. 2015. "Sparse projections for high-dimensional binary codes." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3332–3339.

B.-C. Xu, K. M. Ting, and Y. Jiang. 2021. "Isolation graph kernel." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 12. Vol. 35, 10487–10495.

Y. Xu and K. M. Ting. 2025. "Voronoi Diagram Encoded Hashing." In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer, 87–103.

P. Yang, H. Wang, Y. Zhang, L. Qin, W. Zhang, and X. Lin. 2021. "T3s: Effective representation learning for trajectory similarity computation." In: *Proceedings of the IEEE International Conference on Data Engineering*. IEEE, 2183–2188.

D. Yao, H. Hu, L. Du, G. Cong, S. Han, and J. Bi. 2022. "TrajGAT: A graph-based long-term dependency modeling approach for trajectory similarity computation." In: *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2275–2285.

F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. 2014. "Circulant binary embedding." In: *Proceedings of the International Conference on Machine Learning*.

J. Yuan, Y. Zheng, X. Xie, and G. Sun. 2011. "T-drive: Enhancing driving directions with taxi drivers' intelligence." *IEEE Transactions on Knowledge and Data Engineering*, 25, 1, 220–232.

M. Zhang, Z. Cui, M. Neumann, and Y. Chen. 2018. "An end-to-end deep learning architecture for graph classification." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 1. Vol. 32.

Y. Zheng, X. Xie, W.-Y. Ma, et al.. 2010. "GeoLife: A collaborative social networking service among user, location and trajectory." *IEEE Data Engineering Bulletin*, 33, 2, 32–39.

Z. Zhibo, X. Yang, T. Kai Ming, and N. Cam-Tu. 2026. "LLMs Meet Isolation Kernel: Lightweight, Learning-free Binary Embeddings for Fast Retrieval." *arXiv preprint arXiv:2601.09159*.

L. Zhu, C. Zheng, W. Guan, J. Li, Y. Yang, and H. T. Shen. 2023. "Multi-modal hashing for efficient multimedia retrieval: A survey." *IEEE Transactions on Knowledge and Data Engineering*, 36, 1, 239–260.

X. Zhu, L. Zhang, and Z. Huang. 2014. "A Sparse Embedding and Least Variance Encoding Approach to Hashing." *IEEE Transactions on Image Processing*.

A Dataset Details and Preprocessing

This appendix provides detailed information on the graph and trajectory datasets used in our experiments.

A.1 Graph Datasets

We use six public graph datasets that cover various scenarios, including representations of letters, images, and chemical molecular structures. Basic statistics for these datasets are summarized in Table 7. In our experiments, all graph edges are treated as undirected and unweighted.

Preprocessing for Chemical Datasets. The `aspirin`, `salicylic_acid`, and `toluene` datasets were originally for regression tasks. To adapt them for classification-based retrieval, we performed a two-step preprocessing. First, we converted them into binary classification datasets by labeling graphs in the top quartile of regression values as class 1 and those in the bottom quartile as class 0; graphs in the middle two quartiles were discarded. Second, due to the large size of the resulting datasets, we created the final versions by randomly sampling 1/4 of the graphs. For notational convenience, we refer to these final preprocessed datasets simply by their base names in the main paper.

A.2 Trajectory Datasets

For trajectory-related experiments, we use four widely-recognized real-world datasets. `T-Drive` (Yuan et al. 2011), `Porto` (Moreira-Matias et al. 2016), and `NYC` (Donovan and Work 2016) are large-scale taxi trajectory datasets used for retrieval tasks. `GeoLife` (Wang, Wang, et al. 2024; Wang, Zhu, et al. 2023; Zheng et al. 2010) contains GPS trajectories from human users and includes two distinct versions for downstream tasks. It is important to note that the version for clustering is a curated subset, from which a large number of potentially anomalous and overlapping trajectories have been removed to retain only large, well-separated trajectory groups, making it suitable for evaluating clustering quality (Wang, Zhu, et al. 2023). The version for anomaly detection, however, retains these trajectories. A statistical summary of these datasets is provided in Table 8.

B Evaluation Metrics and Parameter Settings

This appendix provides the detailed formulations of the evaluation metrics used in our experiments.

Table 7. Summary of the graph datasets used for retrieval experiments.

Dataset	#Graphs	Avg. Nodes	Avg. Edges	#Classes
`Letter-med`	2,250	4.67	3.21	15
`COIL-RAG`	3,900	3.01	3.02	100
`COLORS-3`	10,500	61.31	91.03	11
`aspirin`	13,868	21.00	151.52	2
`salicylic_acid`	27,089	16.00	104.13	2
`toluene`	42,270	15.00	96.15	2

Table 8. Summary of the four real-world trajectory datasets.

Statistic	T-Drive	Porto	NYC	GeoLife	
				Clustering	Anomaly Detection
#Trajectories	744,804	1,641,032	11,926,596	9,192	80,450
Total #Points	16,023,166	83,210,527	277,005,055	620,477	5,401,325
min–max $\|X\|$*	10–3,438	10–3,881	10–244	24–100	23–100
Avg. $\|X\|$	21.5	50.7	23.2	67.5	67.1

*$|X|$ denotes the number of points in a trajectory X.

Mean Average Precision (mAP). The mAP is the mean of the Average Precision (AP) scores over a set of queries Q. For a single query, AP is defined as:

$$\text{AP} = \frac{\sum_{k=1}^{N}(P(k) \times \text{rel}(k))}{\text{Number of relevant items}},$$

where N is the total number of items in the database, $P(k)$ is the precision at the k-th retrieved item, and $\text{rel}(k)$ is an indicator function that is 1 if the item at rank k is relevant and 0 otherwise. The final mAP is the average AP over all queries.

Precision-Recall (PR) Curve. The PR curve plots precision against recall for different ranking thresholds. Precision is the fraction of retrieved items that are relevant, while recall is the fraction of relevant items that have been retrieved. They are defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

where TP, FP, and FN are the numbers of true positives, false positives, and false negatives, respectively.

Normalized Mutual Information (NMI). NMI measures the agreement between two cluster assignments (*e.g.*, ground-truth U and predicted V), normalized by their entropies. It is defined as:

$$\text{NMI}(U, V) = \frac{2 \times I(U, V)}{H(U) + H(V)},$$

where $I(U, V)$ is the mutual information between U and V, and $H(U)$ and $H(V)$ are their respective entropies. NMI scores range from 0 (no correlation) to 1 (perfect correlation).

Adjusted Rand Index (ARI). ARI measures the similarity between two cluster assignments, correcting for chance. It computes a similarity score based on all pairs of samples that are assigned to the same or different clusters in both the predicted and true clusterings. ARI scores range from -1 (dissimilar) to 1 (perfectly similar), with values near 0 indicating random agreement.

Area Under the Curve (AUC). AUC refers to the area under the Receiver Operating Characteristic (ROC) curve. The ROC curve is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

where TN is the number of true negatives. An AUC of 1.0 represents a perfect model, while 0.5 represents a model with no discriminative ability.

C Statistical Significance Analysis

To validate the statistical reliability of VDeH's performance gains, we conduct a significance analysis. As noted in the main text, all experiments are executed over 10 independent runs with different random seeds for the Voronoi partition samplings. A key characteristic of VDeH is that its randomness stems solely from the sampling of Voronoi partitions, and the final binary code is an ensemble of $T = L/w$ independent partitions. As L increases, the law of large numbers ensures that the variance of the resulting distance estimates diminishes rapidly.

Table 9. Standard deviation ($\pm$) of VDeH's mAP scores over 10 independent runs on the four vector datasets.

Dataset	128 bits	256 bits	512 bits	1024 bits	2048 bits
CIFAR-10	$\pm.004$	$\pm.003$	$\pm.003$	$\pm.002$	$\pm.002$
GIST	$\pm.003$	$\pm.002$	$\pm.002$	$\pm.001$	$\pm.001$
Nytimes	$\pm.005$	$\pm.004$	$\pm.003$	$\pm.002$	$\pm.002$
Kosarak	$\pm.004$	$\pm.003$	$\pm.002$	$\pm.002$	$\pm.001$

Table 9 reports the standard deviation of VDeH's mAP scores across the 10 runs on the four vector datasets. The extremely small standard deviations (typically < 0.005) confirm that 10 runs are sufficient to produce stable and reliable estimates of VDeH's performance.

Received 12 February 2026; accepted 15 June 2026