CANDE: A Lightweight Locality-Sensitive Hashing Add-on for Candidate-Based Distribution Estimation

Jingfan Meng Huayi Wang Kexin Rong Jun Xu
Georgia Institute of Technology, Atlanta, GA, USA
jfmeng@gatech.edu huayiwang@gatech.edu krong@gatech.edu jx@cc.gatech.edu

***Abstract*—Locality sensitive hashing (LSH) is a widely used technique for approximate nearest neighbor search (ANNS). In an LSH-based solution for ANNS, the computation of query-to-data (Q2D) distances accounts for a considerable fraction of the query time, but such distance information is thrown away after nearest neighbors are identified. In this paper, we propose CANDE (Candidate-based Distribution Estimation), a lightweight add-on to LSH that reuses such information for a wide range of analytics tasks including Q2D distance distribution estimation (QDDE), kernel density estimation (KDE), and query-time recall estimation (QTRE). This allows for significant savings in indexing costs and query time for multiple tasks associated with the original query.**

The main technical hurdle that CANDE addresses is the accurate estimation of some important statistics of the dataset via importance sampling. We discover that the existing estimators of these statistics are not accurate, because they approximate the actual number of collisions (called *collision rate*) in the LSH index using the theoretical *collision probability* (of the LSH function family), and this approximation is crude. To address this issue, we propose more accurate estimators based on a novel scheme called *inferred collision rate* (ICR), which gives a much better approximation to the actual collision rate. Furthermore, we propose an efficient algorithm for computing ICR from the nearest neighbor candidates returned by ANNS. Our evaluation shows that CANDE outperforms existing solutions on multiple analytics tasks while adding only about 8% to 19% query time overhead to ANNS.

I. INTRODUCTION

Locality sensitive hashing (LSH) is a widely used technique for approximate nearest neighbor search (ANNS), a.k.a. similarity search, over high-dimensional datasets [1], [2], [3]. LSH offers a small index size and low resource usage for index construction, as well as reduced costs for data addition and deletion, making it suitable for ANNS applications with limited computational power and constantly updating datasets [4]. In the past two decades, LSH has become a fruitful research area with various schemes for different distance metrics such as L_p [2], angular [5], and Jaccard [6], as well as variants optimized for index size [3], query time [7], or I/O cost [8].

Central to an LSH scheme is the design of its family of LSH functions, which ensures similar data items are much more likely to collide in the same hash bucket compared with dissimilar items. This feature expedites ANNS by narrowing the search to a small subset of data items that hash-collide with the query, known as *nearest neighbor (NN) candidates*. For example, a typical LSH scheme [3] for ANNS only needs to inspect about 5% to 10% of a million-item dataset to achieve a recall of around 0.8, thereby providing significant speedups compared to a naïve linear scan.

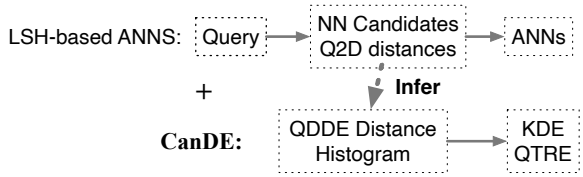


Fig. 1: As an add-on, CANDE augments LSH-based ANNS for various analytics tasks.

In order to identify true nearest neighbors from NN candidates, the LSH scheme needs to retrieve the original vector of every NN candidate and compute its query-to-data (Q2D) distance. This process accounts for a considerable fraction of the query time. We observe that these Q2D distances, which are typically discarded post-query, are actually a valuable "sketch" of the neighborhood of the query.

In light of this observation, this paper presents CANDE, a lightweight LSH add-on that reuses the NN candidates and their Q2D distances for additional analytics tasks with minimal added cost (Figure 1). The basic functionality of CANDE is Q2D distance distribution estimation (QDDE), which returns a histogram on the Q2D distances of all dataset items. This histogram is useful for various analytics tasks, such as kernel density estimation (KDE) [9], ANNS query-time recall estimation (QTRE), distance-based outlier detection [10] and cardinality estimation [11], [12], and fair near neighbor sampling (FNNS) [13]. Compared with existing solutions, which perform each task separately of others, CANDE is more cost-effective in two aspects. First, CANDE uses the same LSH index built for ANNS for multiple tasks, thereby saving considerable indexing and storage overhead compared with constructing a separate index for each task [9], [13], [14]. Second, CANDE amortizes the expensive Q2D distance computation across multiple tasks on the same query, leading to substantial savings in total query time. Indeed, our evaluations show that CANDE can perform a suite of tasks (e.g., KDE and ANNS) on large datasets with only 8% to 19% query time overhead on top of executing a standalone ANNS query.

Probability vs. rate: To obtain accurate estimates based on NN candidates, CANDE recognizes an important distinction

between the theoretical *collision probability* and the actual *collision rate* in LSH. In our observations (in § III-C), the actual collision rate almost always deviates significantly from its expectation (computed from collision probability). Despite this discrepancy, the mainstream approach has been using the collision probability in place of the collision rate, because the latter is challenging even to approximate at the query stage. This has lead to limited estimation accuracy [15], [16] or high resource usage [9] for all existing schemes so far.

As our second key contribution, we propose a novel framework that overcomes this limitation based on a new concept of *inferred collision rate* (ICR) derived statistically from the NN candidates of ANNS. Unlike existing schemes, ICR accurately approximates the actual collision rate without relying on any data-dependent assumption. Furthermore, it allows our estimators to offer superior estimation accuracy compared with those in [9], [16], [14]. Finally, we have designed a highly efficient algorithm for computing ICR, to be described in §IV-C.

In summary, we make three major contributions.

1) We discover that the NN candidates and their Q2D distances obtained from LSH-based ANNS can be reused to solve a number of analytics tasks efficiently and accurately, including QDDE, KDE, and a novel research problem of query-time recall estimation (QTRE).
2) We propose ICR, an accurate approximation to LSH collision rate, which improves the estimation accuracy of a class of importance sampling estimators that are based on LSH. In addition, we introduce a novel algorithm that computes ICR efficiently.
3) We evaluate the accuracy and efficiency of CANDE on six real-world datasets. CANDE achieves higher accuracy than existing solutions while being highly efficient, using only 8% to 19% of the ANNS query time.

II. BACKGROUND AND OVERVIEW

We start this section with background on LSH-based ANNS in §II-A. We provide an overview of three example analytics tasks that CANDE supports in §II-B.

A. Background: Locality Sensitive Hashing

LSH-based ANNS: We first explain how to process an approximate nearest neighbor search (ANNS) query using LSH. An LSH index usually consists of multiple (say L) LSH tables. Each LSH table stores the hash values $h(\vec{x})$ of all data items $\vec{x}$ in a dataset $\mathcal{D}$ under a (typically compound, see example at the end of §II-A) LSH function $h(\cdot)$. Usually, these L compound LSH functions (one for each LSH table) are i.i.d. random realizations from a family of functions $\mathcal{H}$ that satisfy the distance preserving property (Definition 1).

Given a query $\vec{q}$, each LSH table generates a set of NN candidates denoted by $\mathcal{A}$, which likely contains some nearest neighbors of $\vec{q}$. In most LSH schemes, $\mathcal{A}$ is defined as the set of data items $\vec{x}$ such that $h(\vec{x}) = h(\vec{q})$ ($\vec{x}$ having a *hash collision* with $\vec{q}$). We stick to this definition in this paper, although all our results generalize to other definitions of $\mathcal{A}$ such as that in multi-probe LSH [3].

Once we have the set of NN candidates A_i generated from each LSH table $i = 1, 2, \ldots, L$, the next step is to compute their union $\mathcal{A} \triangleq \cup_{i=1}^{L} A_i$ and the query-to-data (Q2D) distance, denoted by $d(\vec{q}, \vec{x})$, for each NN candidate $\vec{x} \in \mathcal{A}$. The ANNS query result consists of the top-K data items in $\mathcal{A}$ with shortest Q2D distances.

Collision Probability: Collision probability is a concept central to LSH. It is defined as the probability that a fixed data item $\vec{x}$ collides with the query $\vec{q}$ under a randomly seeded (realized) hash function $h(\cdot)$ from the LSH family $\mathcal{H}$ (in a single LSH table). We denote this value as the *table-level* collision probability $p(\vec{x})$. We will drop the part "$(\vec{x})$" from $p(\vec{x})$ and other related values in the sequel when $\vec{x}$ is obvious from context. The key technical idea of LSH is that its collision probability satisfies the following *distance preserving* property.

Definition 1. *A family $\mathcal{H}$ of LSH functions is distance preserving if for any query $\vec{q}$ and two data items $\vec{x}_1$ and $\vec{x}_2$, we have $p(\vec{x}_1) \leq p(\vec{x}_2)$ whenever $d(\vec{q}, \vec{x}_1) \geq d(\vec{q}, \vec{x}_2)$ and vice versa. In other words, distance preserving means collision probability p is a monotonically decreasing function of the Q2D distance.*

We denote by $P(\vec{x})$ the *index-level* collision probability that a fixed data item $\vec{x}$ collides with $\vec{q}$ in *any* of the L LSH tables, or equivalently, appears in the union of NN candidates $\mathcal{A}$. As a convention, an upper case letter, such as P, denotes an index-level value (collision probability or rate), and the corresponding lower case letter, such as p, denotes its table-level counterpart. Since each LSH table is independently seeded, we have $P = 1 - (1 - p)^L$ for any data item. As we will show shortly, this index-level collision probability P is central to importance sampling schemes that infer properties of the dataset from $\mathcal{A}$.

Example: E2LSH for L_2: We now describe E2LSH [2], a classic scheme for L_2 distance[1] that many of our examples are based on. In E2LSH, $h(\cdot)$ for each LSH table is a compound LSH function comprising many (say M) *constituents* ("element" LSH functions): $h(\vec{x}) \triangleq \Big(g_1(\vec{x}), g_2(\vec{x}), \ldots, g_M(\vec{x})\Big)$. Each constituent $g_i(\cdot)$ (for $i = 1, 2, \ldots, M$) takes the form $g_i(\vec{x}) = \lfloor (\langle \vec{\theta}_i, \vec{x} \rangle + \beta_i)/w \rfloor$. In this formula, $\vec{\theta}_i$ is an i.i.d. Gaussian random vector, and the inner product $\langle \vec{\theta}_i, \vec{x} \rangle$ is called the *raw hash value* [17], denoted by $f_i(\vec{x})$. The number w is a fixed parameter called the *bucket width*, and β_i is a random variable uniformly distributed in $[0, w)$. Seeding (realizing) an LSH function $h(\cdot)$ from the family $\mathcal{H}$ involves generating and fixing the random vectors (or variables) $\vec{\theta}_i$'s and β_i's for all M constituent LSH functions, which determine the raw hash values (given a certain $\vec{x}$) and bucket boundaries, respectively.

B. CanDE: Analytic Add-on for LSH Index

CANDE is a novel add-on framework that supplements existing LSH-based ANNS indices (already built and tuned for the given dataset) for a range of analytics tasks. Given an

[1] In this paper, we use L_2 as the default distance metric $d(\cdot, \cdot)$. Indeed, CANDE can work with any distance metric that has a distance-preserving LSH for ANNS.

ANNS query $\vec{q}$, CANDE reuses all information gathered and computed for processing the ANNS query (described in §II-A), and return multiple analytics results while adding a negligible overhead to its running time.

Such information is not utilized in existing solutions. For example, for kernel density estimation, the state-of-the-art estimators [9], [18] draws data samples using a separate, specifically-built LSH index. The Q2D distances of their samples (used in a similar estimator as (2)) are also computed separately from those for ANNS. This not only leads to overheads of building and storing a second index, but also long query times due to the repeated computations of Q2D distances.

CANDE can flexibly adapt to a wide range of analytics tasks about counting the "data distribution" in the neighborhood of the query. In this paper, we focus on the following three aforementioned data analytics tasks that can provide important insights into the dataset or ANNS performance. Any of these tasks, either separately or in combination, can be performed, using this framework, with only a small additional overhead to the ANNS query time.

1. QDDE (query-to-data distance distribution estimation): This task is to compute a histogram on the distribution of Q2D distances of all data items. Specifically, a QDDE task involves a query $\vec{q}$; and a list of bins $B_1, B_2, \ldots, B_N$ given by the user, each of which covers a range of Q2D distances. The height of a bin B, say $[b, b')$ is defined as the number of NNs whose Q2D distance falls into this range, i.e., $U \triangleq |\mathcal{D} \cap B| = |\{\vec{x} \in \mathcal{D} \mid b \leq d(\vec{q}, \vec{x}) < b'\}|$. The result of QDDE is the estimated heights of all given bins. QDDE provides a spatial relational summary on the density of data on each annulus (a specific Q2D distance range) centered around the query, which serves as the foundation of other tasks such as KDE and QTRE. Despite such significance, efficient and accurate QDDE is challenging for high-dimensional datasets, because methods based on standard collision probability barely achieves higher accuracy than naïve random sampling (see Figure 6).

2. KDE (kernel density estimation): KDE is a popular non-parametric method for estimating the probability density function of a random variable defined as a certain function of the dataset $\mathcal{D}$ [9], [18]. It has been used, for a long time, in various data analytics applications such as outlier detection, regression, and clustering. In recent years, there has been a growing research interest in improving the efficiency of KDE on high-dimensional datasets [9], [18], [14]. The goal of KDE is to estimate the kernel density Z, defined as the sum of kernel weights of all data items: $Z \triangleq \sum_{\vec{x} \in \mathcal{D}} k(\vec{q}, \vec{x})$, where $k(\cdot, \cdot)$ is called the *kernel weight function*. In this paper, we focus on the most commonly used kernel weight function [9], namely the Gaussian kernel, defined as $k(\vec{q}, \vec{x}) \triangleq \exp(-d^2(\vec{q}, \vec{x})/2\sigma^2)$, wherein σ is a hyperparameter called the *bandwidth*. Since the Gaussian kernel decays exponentially, KDE focuses on the accurate counting of data items near the query.

3. QTRE (query-time recall estimation): This task is to self-report an estimated recall of the ANNS answers returned by an LSH index for a specific query. *Recall* is one of the most important accuracy metrics for ANNS [4]. It is defined as $|\mathcal{K} \cap \mathcal{K}^*|/K$ for a top-K ANNS query $\vec{q}$, wherein $\mathcal{K}$ is the set of top-K approximate NNs returned by the ANNS algorithm, and $\mathcal{K}^*$ is the set of actual top-K NNs of $\vec{q}$.

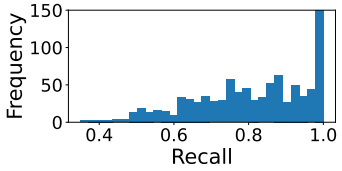


Fig. 2: Histogram of actual recalls of 1000 top-100 queries on SIFT dataset with the average recall tuned to 0.8. The actual recalls are very different, some of which as low as 0.4.

The standard practice for LSH-based ANNS is to tune its parameters to achieve a desired *average recall* on sampled queries [2], [3]. To the best of our knowledge, no existing work allows the self-reporting of the estimated recall of the answers to a specific ANNS query. However, in practice, the actual recalls of ANNS queries vary greatly from one to another (from 0.4 to 1 as in Figure 2). Therefore, knowing the actual recall is crucial for providing consistent ANNS query accuracy. For example, if the recall of a specific query is very low, a remedial action can be taken such as linearly scanning the entire dataset, which takes 10 to 20 times the query time of scanning only the NN candidates [4].

TABLE I: Summary of notations.

Notation	Definition
$\mathcal{D}$	Dataset of points in $\mathbb{R}^d$
$d(\vec{q}, \vec{x})$	Query-to-data (Q2D) distance
$\mathcal{A}$	Set of NN candidates
A_i	Set of NN candidates from the i-th table
$\mathcal{C}$	Set of NN candidates in histogram bin B (NN-B candidates)
C_i	Set of NN-B candidates from the i-th table
$P(\vec{x})$	Index-level collision *probability* (CP) of a data point $\vec{x}$
$p(\vec{x})$	Table-level collision *probability* of a data point $\vec{x}$, see (3)
$R(\vec{h}, \{\vec{x}\})$	Index-level collision *rate* (CR) of a set of data items $\{\vec{x}\}$ under all LSH functions $\vec{h}$ in the index, see (6)
$r(h, \{\vec{x}\})$	Table-level collision *rate* of a set of data items $\{\vec{x}\}$ under a LSH function $h(\cdot)$, see (4)
$\hat{R}(C_{1\ldots L})$	Index-level inferred collision *rate* (ICR) computed from NN-B candidates $C_{1\ldots L} = \{C_1, C_2, \ldots, C_L\}$, see (8)
$\hat{r}_i(C_{1\ldots L})$	Table-level inferred collision *rate* for LSH table i computed from NN-B candidates $C_{1\ldots L}$, see (7)

III. CANDE-CP ESTIMATORS AND LIMITATION

In this section, we introduce in § III-A CANDE-CP, the commonsense estimators for QDDE, KDE, and QTRE that are based on (the standard concept of) collision probability

(CP), with a brief description of their implementation. We find that these estimators are not as accurate as expected due to a long overlooked discrepancy between CP and collision rate (CR) in LSH (§III-B). The cause and the implication of this discrepancy is elaborated in §III-C.

A. CanDE-CP: CP-Based Estimators

CANDE-CP are importance sampling estimators that use the collision probability P of the LSH index for estimating QDDE, KDE, and recall. Importance sampling is a statistical technique for estimating properties of a certain distribution using samples generated from a separate sampling distribution. In the context of CANDE, this technique is utilized to make inference using the set $\mathcal{A}$ of NN candidates, which are samples generated (for processing an ANNS query) via a distribution determined by the LSH scheme.

CanDE-CP Estimators: All three estimators below can be computed using only the Q2D distances of NN candidates, since the collision probability $P(\vec{x})$ is a function of the Q2D distance of $\vec{x}$ by Definition 1, and so is the kernel density function $k(\vec{q},\vec{x})$ as mentioned above.

- For QDDE, focusing on one target bin B, the estimator for the bin height $U = |\mathcal{D} \cap B|$ is

$$\hat{U} = \sum_{\vec{x} \in \mathcal{A} \cap B} 1/P(\vec{x}). \tag{1}$$

- For KDE, our estimator for the kernel density $Z \triangleq \sum_{\vec{x} \in \mathcal{D}} k(\vec{q},\vec{x})$ is

$$\hat{Z} = \sum_{\vec{x} \in \mathcal{A}} k(\vec{q},\vec{x})/P(\vec{x}). \tag{2}$$

- For QTRE, let $\vec{x}_1, \vec{x}_2, \ldots$ denote all NN candidates in $\mathcal{A}$ in the increasing order of Q2D distance. We estimate $Y = |\mathcal{K} \cap \mathcal{K}^*|$ (the number of true NNs found by LSH) as the largest integer $\hat{Y}$ such that $\sum_{i=1}^{\hat{Y}} 1/P(\vec{x}_i) \le K$. The recall of this query is estimated as $\hat{Y}/K$.

Implementation: The values of all CANDE-CP estimators above can be calculated simultaneously in one pass over $\mathcal{A}$: At the arrival of each NN candidate $\vec{x}$, we can directly update $\hat{U}$ and $\hat{Z}$ according to (1) and (2) respectively; and update a heap that stores top-$\hat{Y}$ ($\hat{Y} \le K$) NN candidates for QTRE. Such calculation would however require computing $P(\vec{x})$ for each $\vec{x} \in \mathcal{A}$. Since each $P(\vec{x})$ is not "dirt cheap" to compute, computing all collision probability values on-the-fly is not practical especially on large datasets. Hence, we opt to use precomputed values of $P(\vec{x})$, which are values of a function on the Q2D distances of $\vec{x}$'s by Definition 1. In our experiment, we store all collision probability values in a lookup table for Q2D distances quantized up to 4 digits of precision, which incurs a small memory overhead of around 80 KB. This way, CANDE-CP incurs a negligible additional computational cost on top of ANNS.

Limitation: We were surprised that the relative errors of CANDE-CP estimators are actually way higher than their typical values of about $O(1/\sqrt{|\mathcal{A}|})$, where $|\mathcal{A}|$ is the number of NN candidates [19]. For example, on Deep and SIFT datasets where $|\mathcal{A}| \approx 10^5$, we observe a 10% to 20% relative error using the CANDE-CP estimator, which is 30 to 60 times larger than expected. We now know that this inaccuracy results from the discrepancy between two closely related values concerning LSH: collision probability and collision rate. Unfortunately, this difference has been largely overlooked in standard importance sampling estimators that are grounded in collision probability [15], [16], [9]. The subsequent sections delve into a detailed discussion of these two values, illustrating the cause and practical implications of their differences.

B. Collision Probability vs. Rate

In this subsection, we define *collision probability* and *collision rate*, and explain the relationship between these two values. For ease of presentation, we focus the discussion on the level of a single LSH table here and will extend to the index level in §III-C.

Collision Probability: Collision probability measures the expected chance of collision over all possible hash functions in the given LSH family.

Definition 2. *Given a query $\vec{q}$, a data item $\vec{x}$, and a family $\mathcal{H}$ of LSH functions, the* collision probability $p(\vec{x})$ *is defined as the probability of realizing (generating) a random function $h(\cdot)$ from $\mathcal{H}$ such that $\vec{x}$ collides with $\vec{q}$ under $h(\cdot)$, i.e.,*

$$p(\vec{x}) \triangleq \Pr_{h \in \mathcal{H}} \left(\vec{x} \text{ collides with } \vec{q} \text{ under } h(\cdot)\right). \tag{3}$$

For most LSH schemes, the collision probability is a well-studied value, which can be computed either from closed-form expressions such as for E2LSH [2], or using Monte Carlo simulation such as for multi-probe LSH [3].

Collision Rate: Collision rate measures the actual number of collisions on a specific (realized) LSH table.

Definition 3. *Given a query $\vec{q}$, a set of data items $\{\vec{x}\}$ and a fully realized LSH function $h(\cdot)$, the* collision rate *of these data items under $h(\cdot)$, denoted by $r(h,\{\vec{x}\})$, is defined as the ratio of the number of data items that collide with $\vec{q}$ to the cardinality of $\{\vec{x}\}$. In other words,*

$$r(h,\{\vec{x}\}) \triangleq \frac{|\{\vec{x} \mid \vec{x} \text{ collides with } \vec{q} \text{ under } h(\cdot)\}|}{|\{\vec{x}\}|}. \tag{4}$$

We illustrate how collision rate is affected by the seeding of LSH tables using an example of E2LSH. Suppose each LSH table uses a compound E2LSH function $h(\cdot)$ that consists of $M = 2$ constituents. As described in §II-A, the hash value $h(\vec{x})$ of a data item $\vec{x}$ is determined by the raw hash values $f_i(\vec{x}) = \langle \vec{\theta_i}, \vec{x} \rangle$ and the "bucket boundaries" β_i, for $i = 1, 2$, all of which are fixed during the seeding process of $h(\cdot)$.

Figure 3 shows two LSH tables, each containing (the raw hash values of) 20 data items $\{\vec{x}\}$ (black dots) and a query item $\vec{q}$ (the red plus sign). These two tables are seeded with the same $\vec{\theta_1}$, $\vec{\theta_2}$, and β_2, but different β_1. As a result, the raw

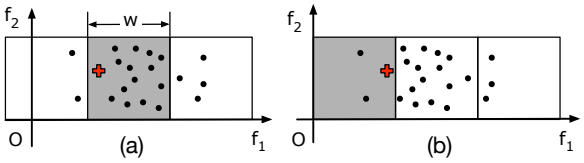


Fig. 3: Two example E2LSH tables with different seeds. The collision rate differ greatly (0.7 versus 0.1) between these two LSH tables, because bucket boundaries are realized differently. Note that the raw hash values of data items are distributed unevenly around $f(\vec{q})$, although the marginal distribution of each of them centers at $f(\vec{q})$. This is because they are strongly correlated by the seeding of $f(\cdot)$.

hash values f_1 and f_2 for all data and query items are the same in the two subfigures, but the horizontal bucket boundaries are different. By Definition 3, collision rate is the number of black dots that fall into the same bucket as the query (the shaded squares) divided by the total number of data items. Therefore, the collision rate in Figure 3 (a) is $14/20 = 0.7$, whereas that in Figure 3 (b) is only $2/20 = 0.1$. This example demonstrates that even a slight difference in bucket boundaries caused by the seeding of LSH can lead to vastly different collision rates.

Relationship Between These Two Values: Supposing all data items in $\{\vec{x}\}$ has the same collision probability $p(\vec{x})$ (or they are equally far from the query), the expectation of collision rate across all *random seeds* (that generates $h(\cdot)$ from $\mathcal{H}$) is exactly $p(\vec{x})$ [20], i.e.,

$$p(\vec{x}) = \mathbb{E}_{h\in\mathcal{H}}[r(h, \{\vec{x}\})]. \tag{5}$$

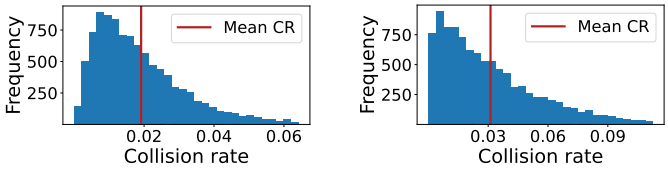
According to (5), collision probability can be regarded as the "expected collision rate", which is a constant in all LSH tables independent of the seeding. Just as the actual value of a random variable is not necessarily close to its expectation, the actual collision rate (in a realized LSH table) can be very different from collision probability.

C. Gap Between Collision Probability and Rate

In this subsection, we highlight the significance of the difference between collision probability and collision rate, as well as its impact on the accuracy of CANDE-CP estimators.

Gap on One LSH Table: We first explain the difference between collision probability p and the actual collision rate $r(h)$ (as a function of the seeding of $h(\cdot)$) on a single LSH table indexed using $h(\cdot)$. Except for a few works such as [20], prior LSH studies often assume that these two values are interchangeable. Here, we show experimental evidence that because of the aforementioned "boundary effect", the variability of collision rate $r(h)$ across different LSH tables is too large to safely assume that $p \approx r(h)$.

We use two datasets in this experiment: a real-world dataset Audio [21], which consists of 53,300 192-dimensional vectors converted from audio data, and a synthetic dataset Uniform that consists of 100,000 128-dimensional data items uniformly distributed on the unit hypersphere. For each dataset, we build 10,000 LSH tables using independent realizations of E2LSH



(a) Audio (b) Uniform

Fig. 4: The histogram of collision rates measured from 10,000 LSH tables. Notably, the collision rates of many tables significantly deviate from the mean.

functions $h(\cdot)$. Figure 4 shows the histogram of the collision rates $r(h)$ of each dataset across all LSH tables. It is clear that the distributions of $r(h)$ are far from their respective means (vertical red lines) on both datasets, with many LSH tables having collision rates either close to 0 or several times higher than the mean.

Gap on a Multi-Table Index: Recall from § II-A that the index-level collision probability P is the probability of colliding in at least one LSH table and can be computed from p as $P = 1 - (1-p)^L$. Similarly, we can define the index-level collision rate $R(\vec{h}, \{\vec{x}\})$ for any set of data items $\{\vec{x}\}$ as the ratio of the data items in $\{\vec{x}\}$ that appear in the union of NN candidates $\mathcal{A}$ to the cardinality of $\{\vec{x}\}$, i.e., $R(\vec{h}, \{\vec{x}\}) \triangleq |\{\vec{x}\} \cap \mathcal{A}|/\{\vec{x}\}|$. Here, we denote as "$\vec{h}$" the concatenation of all L compound LSH functions (one for each table) in the index. Also, for succinctness, we drop the part "$\{\vec{x}\}$" from all notations such as $R(\vec{h}, \{\vec{x}\})$ in the sequel. By the independence between LSH functions in different tables, $R(\vec{h})$ can be approximated by

$$R(\vec{h}) \approx 1 - \prod_{i=1}^{L}(1 - r(h_i)). \tag{6}$$

Since the formulas of P and $R(\vec{h})$ take similar forms, the large deviations of $r(h)$ from p on the table-level cause the index-level collision rate $R(\vec{h})$ to also deviate from P. On real datasets, P and $R(\vec{h})$ can differ by about 10% to 20%.

Impact on Estimator Accuracy: As we explain in the following QDDE example, a key assumption in traditional importance sampling estimators is that the collision rate $R(\vec{h})$ is close to the collision probability P. For a fixed target bin B, we let $U = \mathcal{D} \cap B$ be the NN-B's (data items whose Q2D distances fall into B) and $\mathcal{C} = \mathcal{A} \cap B$ be the *NN-B candidates* found by the LSH index. We know by definition that $|U| = |\mathcal{C}|/R(\vec{h})$. Hence, the estimator $|\mathcal{C}|/P$ is close to $|U|$ only when $R(\vec{h}) \approx P$, which is not the case as just shown.

The challenge here is that $R(\vec{h})$ is too costly to compute. It is a *query-specific* value (which varies from one query to another) subject to complex "boundary effects" (as shown in Figure 3). As we will show in § VI, even to approximate it requires making *data-dependent* assumptions on the queries and dataset. Our key contribution in this paper is a novel *data-independent* scheme called ICR (inferred collision rate). Instead of predicting the collision rate (as previous works did), ICR statistically infers it from the NN candidates and their

Q2D distances (already calculated for ANNS). Powered by ICR, we can greatly improve the accuracy of our estimators.

IV. CANDE-ICR ESTIMATORS

In this section, we describe the inferred collision rate (ICR), a novel data-independent approximation to the actual collision rate (§IV-A). Next, we introduce CANDE-ICR, our estimators powered by ICR in § IV-B. We present an algorithm for efficiently computing CANDE-ICR in §IV-C.

A. ICR: Data-Independent CR Approximation

In this subsection, we show how the collision rate $R(\vec{h})$ (on a target histogram bin B) is approximated by our ICR $\hat{R}(C_{1...L})$, which is computed from (as a function of) the collection of NN-B candidates $C_{1...L} \triangleq \{C_1, C_2, \ldots, C_L\}$. Here, each C_i is the set of NN-B's that collide with the query in LSH table i. Similar to (6), we calculate the index-level ICR $\hat{R}(C_{1...L})$ as $1-\prod_{i=1}^{L}(1-\hat{r}_i(C_{1...L}))$, wherein each $\hat{r}_i(C_{1...L})$ approximates the table-level collision rate $r(h_i)$ for LSH table i. Hence, it suffices to derive the estimator $\hat{r}_i(C_{1...L})$ for a fixed i only, which we focus on in the sequel.

Statistical Inference Background: We first describe an urn model and its MLE (maximum likelihood estimator) from which $\hat{r}_i(C_{1...L})$ is derived. Let an urn contain M balls of two different colors: N of them are red, the other $M-N$ are black, and the values of M and N are not known. Our statistical inference problem is to estimate N/M, the *fraction* of red balls in the urn. Suppose m balls are drawn from this urn *uniformly at random without replacement, and independently of colors*, and suppose n out of these m balls are red. Given the values of m and n as our observation, n/m is known to be an unbiased MLE [19] for N/M, since n follows a hypergeometric distribution parameterized by m, M, and N.

MLE of Collision Rate: Our task of approximating $r(h_i)$, the collision rate on LSH table i, can be cast into the above statistical inference problem, as follows. Since $r(h_i)$ is defined as $|C_i|/|\mathcal{D} \cap B|$ in (4), it can be faithfully modeled as the fraction of red balls in the urn above wherein each ball corresponds to an NN-B ($M = |\mathcal{D} \cap B|$), has red color if it is in C_i ($N = |C_i|$), and has black color otherwise.

The question remains how to draw a uniform sample of m balls from the urn independently of colors. One straightforward idea is to view the L LSH tables (functions) as the drawing mechanism, and consider the union of all NN-B candidates $\mathcal{C} \triangleq \cup_{j=1}^{L} C_j$ as the m balls drawn from the urn. However, since all balls in C_i are red by definition (which violates the color-independence), we must *exclude* LSH table i from the drawing mechanism. Hence, we instead use the union of NN-B candidates from *the other* LSH tables, that is, $S_i \triangleq \cup_{j=1,j\neq i}^{L} C_j$, since in this way the LSH functions used to determine their selection (in LSH tables except the i^{th}) are independent of that used in LSH table i to determine the color of balls. The choice of S_i also ensures that the balls therein are drawn (almost) uniformly: Given that bins are narrow, every NN-B has a similar Q2D distance and therefore a similar collision probability in each independent LSH table.

Therefore, we use $m = |S_i|$ and $n = |C_i \cap S_i|$, so the MLE becomes $\hat{r}_i(C_{1...L}) \triangleq n/m = |C_i \cap S_i|/|S_i|$. Furthermore, for efficient implementation of ICR (in §IV-C), we can avoid intersecting C_i with S_i (which is costly) by using the following alternative definition. Denote as V_i the set of NN-B candidates *exclusive to* LSH table i, i.e., $V_i \triangleq C_i \setminus S_i$. Then we have $\hat{r}_i(C_{1...L}) = |C_i \cap S_i|/|S_i| = (|C_i| - |V_i|)/(|\mathcal{C}| - |V_i|)$.

Bayesian Enhancement: An issue with the MLE above is that it may produce very large errors if the sample size m is very small (say less than 10). We overcome this issue by using CP, which consistently has 10% to 20% error, as informative *prior knowledge* to enhance the MLE estimator using the following Bayesian technique.

In practice, we can approximate the distribution of the number of red balls in our sample n, which is hypergeometric in principle, with the binomial distribution $Binom(m, r(h_i))$ and use the standard Bayesian estimator [22] $(\alpha+n)/(\alpha+\beta+m)$ for the success probablity $r(h_i)$ of the binomial distribution. Here, α and β are two hyperparameters expressing the prior knowledge. In ICR, to use the CP p as the informative prior, we let $\alpha/(\alpha+\beta) = p$, which implies $\beta = \alpha(1-p)/p$. The remaining hyperparameter α expresses the "strength" of our prior knowledge: The larger α is, the more confident we are about the collision probability p relative to our observations. The appropriate value of α needs to be tuned in experiments.

Substituting β, n and m to values mentioned above, we arrive at our final definition of ICR as follows.

Definition 4 (ICR estimator)**.** *The inferred collision rate $\hat{r}_i(C_{1...L})$ for LSH table i ($i = 1, 2, \ldots, L$) is*

$$\hat{r}_i(C_{1...L}) \triangleq \frac{\alpha + |C_i| - |V_i|}{\alpha/p + |\mathcal{C}| - |V_i|} \approx r(h_i), \tag{7}$$

wherein α is a Bayesian hyperparameter, p is the standard collision probability, and C_i, $\mathcal{C}$, and V_i are defined above.

The ICR for the LSH index (realized by $\vec{h}$) is calculated from $\hat{r}_i(C_{1...L})$, $i = 1, 2, \ldots, L$, under the following formula

$$\hat{R}(C_{1...L}) \triangleq 1 - \prod_{i=1}^{L}(1 - \hat{r}_i(C_{1...L})) \approx R(\vec{h}). \tag{8}$$

B. CanDE-ICR: ICR-Based Estimators

As mentioned above, ICR can approximate the actual collision rate much better than collision probability (CP). Hence, we can significantly improve the accuracy of the three CANDE-CP estimators (for QDDE, KDE, and QTRE respectively) described in § III-A, by replacing the collision probability $P(\vec{x})$ therein with ICR $\hat{R}(B)$, for any target bin B. We call the resulting estimators (with this replacement) the ICR-based estimators. For example, the ICR-based estimator for QDDE is $\hat{u} = \sum_{\vec{x}\in\mathcal{A}\cap B} 1/\hat{R}(B)$.

Our ICR technique has wide applicability: Almost all CP-based importance sampling estimators in the literature (such as [16]) can be similarly changed to ICR-based ones to achieve much better estimation accuracy. However, unlike CP, ICR cannot be precomputed, and it is a nontrivial problem to

compute ICR and implement ICR-based estimators efficiently on large datasets. We will address this problem in the next subsection.

C. Efficient Computation of CanDE-ICR

With the understanding that the ICR $\hat{R}(B)$ on all bins (B's) can be computed simultaneously as we did in §III-A for CANDE-CP estimators, we again focus on the ICR of a target bin B. From (7) and (8), to compute $\hat{R}(C_{1...L})$ on B, it suffices to know the cardinalities of the following sets defined above: C_i (the set of NN-B candidates generated from LSH table i), $\mathcal{C}$ (the union of C_i's), and V_i (the set of NN-B candidates unique to C_i), for $i = 1, 2, \ldots, L$.

These cardinalities can be obtained via post-processing the output from the LSH-based ANNS procedure as follows. The post-processing takes a sequential scan of NN candidates from all LSH tables, namely A_i's for $i = 1, 2, \ldots, L$. For each NN candidate $\vec{x} \in A_i$, we update counters for these cardinalities if $\vec{x}$ falls into the current histogram bin B, which can be determined only from the Q2D distance of $\vec{x}$. This procedure, however, can be better implemented, since looking up the Q2D distance for each $\vec{x}$ in the scan in effect duplicates the memory access pattern for the computation of the union of NN candidates $\mathcal{A}$ in ANNS. On large datasets, such random memory access accounts for between 13% and 16% of the ANNS query time.

Input: NN candidates A_i from LSH tables $i = 1, 2, \ldots, L$.
Output: The set $\mathcal{K}$ of K-ANNs of $\vec{q}$ and cardinalities of $\mathcal{C}$, C_i's, and V_i's (used to compute ICR).

```
1  Initialize K ← ∅ and all counters to 0;
2  for i = 1, 2, ..., L do
3      for each data item x⃗ ∈ A_i do
4          Suppose this is the j^th appearance of x⃗ for
           this query;
5          if j = 1 then
6              Compute its Q2D distance;
7              Update K so that it contains top-K data
               items of shortest Q2D distances;
8          if x⃗ does not belong to the target bin then
9              continue;
10         Increment |C_i| counter by 1;
11         if j = 1 then
12             Increment |C| and |V_i| counters by 1;
13             SRC[x⃗] ← i;
14         else if j = 2 then
15             Decrement |V_SRC[x⃗]| counter by 1;
16 return K and all counters;
```

Procedure 1: Efficient counting algorithm for ICR. The two (**if** $j = 1$) conditions at Lines 5 and 11 are not equivalent due to the **continue** statement at Line 9.

Proc. 1 shows our efficient counting algorithm for ICR that avoids this duplicated random memory access. It carries out the following three computational tasks together while performing a single pass over $A_1, A_2, \ldots, A_L$: 1) computing the union (implicitly in Line 5); 2) Q2D distance computation for ANNS query processing (Lines 6 and 7); and 3) counting cardinalities (Lines 10 through 15). Note that information such as $\vec{x}$'s number of appearances j and the source LSH table $SRC[\vec{x}]$ at $j = 1$ can be stored in a hash table, so they all can be retrieved in $O(1)$ time. We have custom-made the hash table using aggressive compression techniques similar to that in [23] to make it compact enough to fit into the CPU cache for large (ten million) datasets. Hence, the time complexity of Proc. 1, excluding the computation of Q2D distances (which should be charged for ANNS), is $O(\sum_{i=1}^{L} |A_i|)$, or linear to the total number of NN candidates.

V. EVALUATION

In this section, we evaluate the accuracy and efficiency of CANDE on real-world datasets on the aforementioned KDE, QDDE, and QTRE tasks. Overall, our experiments show that:

1) As expected, CANDE-ICR is more accurate than CANDE-CP, and they both achieve higher estimation accuracy than random sampling on KDE, provided that the bandwidth σ is not too large (§ V-B) and QDDE, provided that Q2D distances are not too long (§V-C).
2) On QTRE, CANDE-ICR is more accurate than CANDE-CP, and its error does not increase with the number of NNs (§V-D).
3) CANDE-ICR is a lightweight add-on. Its time overhead is at most 18.6% on top of an ANNS query (§V-E).
4) Finally, in a fixed histogram bin, we show that ICR accurately approximates the colllision rates for most queries (§V-F).

A. Experimental Setup

Our evaluation uses the following six real-world datasets widely used in the ANNS literature [4], [17]. These datasets are different in dimension, size, and data distribution as show in Table II. Each dataset is associated with 1000 ANNS queries.

TABLE II: Summary of Datasets.

	Dataset	# Data	Dim.	Type
Small	MNIST [24]	69.0K	784	Image
	Trevi [25]	99.9K	4096	Image
Medium	GIST [26]	1.0M	960	Image
	GloVe [27]	1.2M	100	Text
Large	Deep [28]	10.0M	96	Image
	SIFT [26]	10.0M	128	Image

We implement CANDE on top of multi-probe E2LSH [3], the state-of-the-art LSH-based ANNS solution for L_2 distance. For each dataset, we tune the parameters so that the query time is near-optimal for top-100 ANNS queries at 0.8 average

recall. We set the Bayesian hyperparameter α to 25 on all datasets, which consistently leads to high estimation accuracy.

We measure estimation accuracy of all three evaluated tasks using the *mean relative error* (MRE) averaged across queries. Lower MRE values indicates higher estimation accuracy. We measure efficiency using the query time (in milliseconds). We conduct all time measurements on C++ programs compiled by g++-13.2 with "-O3" optimization option using a workstation running Ubuntu 22.04 with Intel Core i9–10980XE 3.0 GHz CPU (24.75 MB cache) and 256 GB DRAM. All reported run times are the average of 5 repeated measurements.

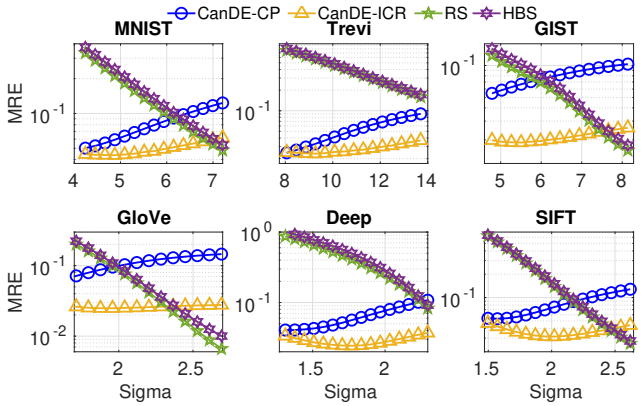


Fig. 5: MRE values of KDE schemes on six datasets. CANDE-ICR is significantly more accurate than RS, HBS, as well as CANDE-CP, on a wide range of bandwidth values.

B. KDE Accuracy

KDE Setup: As we mentioned above, the goal of KDE is to estimate the following Gaussian kernel density: $Z = \sum_{\vec{x}\in\mathcal{D}} \exp(-d^2(\vec{q},\vec{x})/2\sigma^2)$. The hyperparameter σ, known as the *bandwidth*, determines the level of locality in KDE. In this experiment, we choose a range of bandwidth values as follows. The lower bound of the range is selected such that the ground truth kernel density is at least 10^{-6} for at least 80% of the queries[2]. The upper bound is 1.8 times the lower bound. This leads to a wide range in which kernel density values vary by at least six orders of magnitude. We filter out queries whose kernel density is less than 10^{-6} on the smallest bandwidth, so that the reported MRE values are not dominated by large relative errors from a few queries. We standardize all datasets in Table II so that the mean is 0 and variance is 1 on each dimension, following prior practices [18].

We compare CANDE estimators with two baselines:

- RS (random sampling): We uniformly sample m data items $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_m$ from the dataset and use the estimator $\hat{Z} = M/m \cdot \sum_{i=1}^{m} k(\vec{q},\vec{x}_i)$, where M is the size of dataset, and $k(\cdot,\cdot)$ is the kernel weight function.
- HBS (hashing-based-sketch) [18]: HBS makes inference from a set of weighted samples created from hashing and non-uniform sampling, with the goal of better sampling sparse regions of the dataset. Its estimator is a weighted version of that of RS.

In all experiments, we use the average number of NN candidates in ANNS as the sample size m for RS and HBS.

Evaluation Results: Figure 5 reports the MRE values of KDE schemes on six datasets. Overall, we observe that HBS performs similarly to RS and that both CANDE estimators (especially CANDE-ICR) are more accurate than RS and HBS on at least the lower 50% of the range shown, reducing MRE by a factor of up to 18.6. Note that this comparison is performed under the same sample size, and as we will show in Table III, the query speed of CANDE is at least five times as fast as RS and HBS.

As bandwidth increases, CANDE-CP and CANDE-ICR become less accurate while RS and HBS become more accurate. This is because CANDE relies on NN candidates, which are typically close to the query, whereas RS and HBS use random samples likely to be farther away. At lower bandwidths, kernel density is mainly dictated by data items that are close to the query, making CANDE more accurate than RS and HBS. In contrast, when bandwidth increases, many data items other than near neighbors start to contribute to the kernel density, making RS and HBS more accurate.

Comparing the two CANDE estimators, CANDE-ICR is much more accurate than CANDE-CP on most bandwidth values, reducing MRE by a factor of up to 5.2 on all datasets. This shows that our ICR, powered by Bayesian inference, approximates the actual collision rate much more accurately than the standard collision probability, especially on large bandwidths (Q2D distances).

C. QDDE Accuracy

QDDE Setup: Recall that the objective of QDDE is to estimate a distance histogram, wherein the height of each bin represents the number of data items within a particular Q2D distance range. Unlike KDE, we are not aware of existing algorithms for QDDE, so we compare only with RS (random sampling). RS makes estimation from m uniformly sampled items from a dataset of size M. If n out of m samples fall into the target bin, then the estimated bin height is Mn/m. Similar to what we did in KDE, m is set to match the average number of NN candidates from LSH.

As mentioned above, applications such as KDE need high accuracy on close Q2D distances, so we mostly focus on this range in evaluation. On the first four datasets, our histogram covers the 10% nearest data items of the entire dataset on average. On the two ten-million datasets (Deep and SIFT), we cover the 1% nearest, which is still 100,000 data items on average. We also exclude very short Q2D distances, since the error of RS therein is too large compared with CANDE's. Each resulting histogram is divided into at least 10 bins.

Evaluation Results: Figure 6 shows the MRE values of each histogram bin in six datasets. The x-axis of Figure 6 is shown in log scale, since the bins to the right contain much more data items and have much larger Q2D distance quantiles than those

[2]For reference, bandwidth suggested by Scott's rule [29] leads to very small kernel densities (less than 10^{-9}) for our high-dimensional datasets.

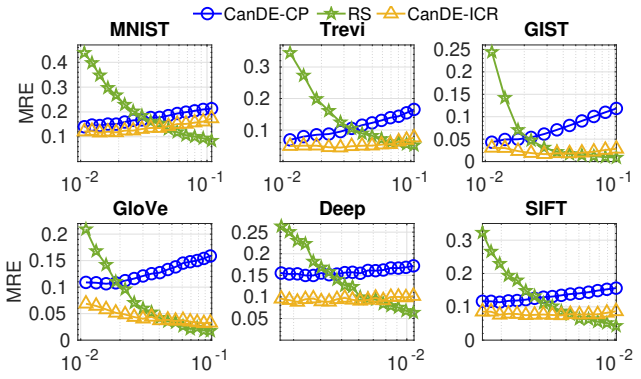


Fig. 6: MRE values of QDDE schemes on six datasets. CANDE-ICR is more accurate than CANDE-CP on the entire range and more accurate than RS on close Q2D distances.

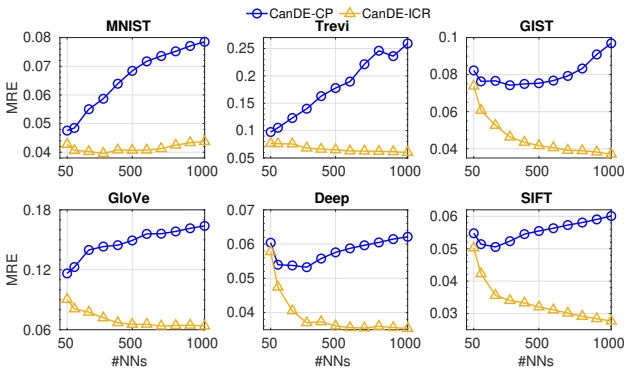


Fig. 7: MRE values of QTRE schemes on six datasets. CANDE-ICR is more accurate than CANDE-CP, and its error does not increase with K for K up to 1000.

to the left. All MRE values in Figure 6 are computed across queries that have at least one data item in every bin. At least 70% of all queries satisfy the above criteria in all datasets.

Similar to the trend in the KDE experiment, as the Q2D distance increases, the accuracy of CANDE decreases while that of RS increases. This is expected result as we have explained above. CANDE-ICR reduces MRE by a factor of up to 9.3 compared with RS. The fact that CANDE is more accurate at small Q2D distances (the left half of range in Figure 6) makes it particularly well suited for applications such as KDE, which focus heavily on nearby data points. Also similarly to the KDE experiment, this comparison is performed under the same sample size, and the running time of CANDE-ICR is much faster than RS in practice.

Comparing both CANDE estimators, CANDE-ICR consistently outperforms CANDE-CP (having smaller MRE by a factor of up to 4.6). Furthermore, in all datasets except MNIST, CANDE-ICR's MRE remains at a small value (less than 0.1) as Q2D distance increases, whereas CANDE-CP's MRE increases continuously with Q2D distance. This indicates that ICR is much more robust and accurate than CP.

D. QTRE Accuracy

QTRE Setup: The goal of QTRE is to self-report the *recall* of ANNS results returned by an LSH-based index for a specific query. In our evaluation, we fix (the seeds and parameters of) a multi-probed E2LSH index for each dataset as mentioned above. This index is tuned to 0.8 average recall across all queries, but as already shown in Figure 2, the actual recalls of queries vary significantly, whose standard deviation ranges between 0.16 and 0.35 across these datasets.

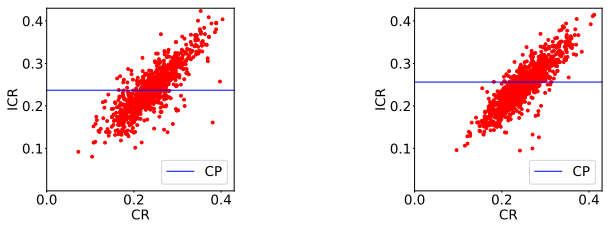
For each top-K ANNS query, the reference (ground truth) recall is the ratio Y/K, wherein Y is the number of true NNs in the query result. QTRE accuracy is measured by the MRE of the estimated recalls over reference ones across queries. We vary K from 50 to 1000, which are respectively typical and large values in the literature [4]. Since we are not aware of any existing QTRE solution, CANDE-ICR is compared only with CANDE-CP, the straightforward solution based on the standard collision probability (CP). Our results, again, show the superior accuracy of ICR over CP.

Evaluation Results: Figure 7 shows the MRE values of all queries associated with each dataset. It is clear that CANDE-ICR is much more accurate than CANDE-CP on all datasets. As K increases, the accuracy of CANDE-ICR either improves or remains the same, whereas that of CANDE-CP degrades steadily. For example, at $K = 100$, CANDE-ICR's MRE is 7.6% to 34.2% less than that of CANDE-CP and are 43.1% to 93.0% less at $K = 1000$. In many cases, CANDE-ICR's MRE is only about 0.05, which is significantly less than the standard deviation of actual recalls (0.20 to 0.44) divided by their means (0.8).

The difference in the trends of MRE as K increases can be attributed to the approximation errors having very different natures (correlated or not) in these two schemes, as follows. In QTRE, the number of true NNs Y in query results is related to K under the formula $\sum_{i=1}^{Y} 1/R(\vec{x}_i) = K$. CANDE-ICR and CANDE-CP approximate the CR R with ICR $\hat{R}$ and CP P, respectively. As K increases, the above formula contains more terms. In the case of CANDE-CP, the estimation errors $1/R(\vec{x}_i) - 1/P(\vec{x}_i)$ for different terms are positively correlated, since as mentioned above, the collision rates $R(\vec{x}_i)$ are positively correlated. As a result, estimation error builds up as K increases. In contrast, in the case of CANDE-ICR, the errors $1/R(\vec{x}_i) - 1/\hat{R}(\vec{x}_i)$ from different bins are almost uncorrelated, since in every bin B, the ICR $\hat{R}$ is calculated from the NN-B candidates independently of those in other bins. As a result, their errors cancel out, so estimation errors do not increase with K.

E. Time Efficiency of CanDE

Table III shows the average query times of running KDE using CANDE-CP and CANDE-ICR compared with those of running an ANNS query. Running the other two tasks (QDDE and QTRE) leads to similar results. The left half of the last two columns show the total query times of answering ANNS and running CANDE on top of it (reusing the NN candidates and their respective Q2D distances). The percentage overhead of CANDE (over answering ANNS alone) is shown on the right



(a) Deep (b) SIFT

Fig. 8: Scatter plots of ICR versus CR. Each dot represents a query. ICR is strongly correlated to CR, whereas CP is a constant for all queries (shown as the blue horizontal line).

half of these columns. The results show that both CANDE-CP and CANDE-ICR are lightweight, with overheads ranging from 3.8% to 8.9%, and 8.4% to 18.6%, respectively.

TABLE III: Query times (ms) of KDE. Both CANDE-CP and CANDE-ICR are lightweight, imposing no more than about 9% and 19% overhead to query time, respectively.

Dataset	ANNS	CANDE-CP	CANDE-ICR
MNIST	9.78	10.65 (8.9%)	11.33 (15.8%)
Trevi	32.1	33.9 (5.6%)	34.8 (8.4%)
GIST	181.3	192.8 (6.3%)	212.6 (17.3%)
GloVe	66.3	71.1 (7.2%)	78.6 (18.6%)
Deep	151.2	160.9 (6.4%)	168.1 (11.2%)
SIFT	201.3	208.9 (3.8%)	221.3 (9.9%)

F. Accuracy of ICR in Approximating CR

To demonstrate the accuracy of the proposed ICR after employing the distance information of NN candidates, we plot the index-level CP, CR, and ICR values in two real datasets (Deep and SIFT). These values are computed or measured for a fixed QDDE bin close to the right boundary of Figure 6. Figure 8 shows a strong correlation between ICR and CR, where most dots except for a few outliers (as a result of random noise) are distributed on the diagonal. In contrast, CP, shown as horizontal lines in Figure 8, does not vary with CR. This result clearly shows that ICR is a much better approximation to CR than CP.

VI. RELATED WORK

In this section, we delve deeper into the historical developments of collision probability and collision rate in LSH-based schemes. Collision probability is a central concept to the theoretical foundation of LSH [2], [30], giving guarantees on the average recall and query time under random LSH functions. In fact, almost every proposal of new LSH scheme has a dedicated section for calculating or analyzing collision probability [2], [5], [8]. Collision probability has also been applied to other settings. For example, some LSH schemes such as multi-probe [3] select promising buckets (that are most likely to contain NNs) based on collision probability. Many emerging applications of LSH, such as sampling [31], [32] and inference [9], [18], [16] also use collision probability as an easy-to-compute alternative to the collision rate.

However, as we mentioned above, these works often have large estimation errors due to the discrepancy between collision probability and collision rate. This is a long-standing issue without a widely-accepted solution. On one hand, some schemes such as [15], [31], [32] have recognized such inaccuracy and focus on applications, such as outlier detection and machine learning, that can tolerate such inaccuracy. On the other hand, many workarounds have been proposed, but they all have to pay significantly more indexing and storage costs than what is needed by ANNS, to compensate for not knowing the actual collision rate. For example, HBE-like algorithms [9], [18] only use one NN candidate from each hash table to estimate the KDE, so they need as many LSH tables as the number of samples. Similarly, the algorithms in [14], [13] assume that the LSH index can find all near neighbors of the query with high probability, which is the case only when many LSH tables are used.

At least two independent threads of works have attempted to close the gap between collision probability and collision rate, but all these attempts are data-dependent in that they rely on assumptions on the queries or dataset distribution. The work [33] proposed to select promising buckets in multi-probe LSH according to a "collision rate" calibrated by Bayesian knowledge conditioning on a specific realization of LSH function. However, such calibration requires training a model using foretold queries. Another work [20] showed that the observed recall of LSH-based ANNS can be better explained from the distribution of collision rates rather than the expected collision probability, and a follow-up work [34] attempted to calculate such distribution under a strong assumption that data items are uniformly distributed on the unit hypersphere. In comparison, our proposed ICR accurately reveals the realization of this distribution in a data-independent way.

VII. CONCLUSION

In this paper, we propose CANDE, a lightweight add-on to LSH-based ANNS that reuses the NN candidates and their Q2D distances for many analytics tasks. We formulate and tackle QTRE, a novel research problem that is important to consistent ANNS accuracy for all queries. We propose ICR, a novel, accurate approximation to the actual collision rate of the LSH index, which makes CANDE much more accurate than existing importance sampling estimators. Furthermore, we propose an efficient algorithm for computing ICR. Finally, we show the high accuracy of CANDE-ICR on KDE, QDDE, and QTRE by extensive experiments.

A limitation of our proposed ICR is that it relies on many NN candidates to be accurate. As a result, this idea only works with LSH but not with other indices such as HNSW [35]. It remains open as to how to design an index that achieves competitive query times (so as not to slow down users who are not interested in analytics tasks) and also facilitates importance sampling on the dataset.

Acknowledgment. This material is based upon work partially supported by the National Science Foundation under grant

numbers CNS-2007006 and IIS-2335881. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," in *Symp. on Theory of Comput.*, ser. STOC '98. New York, NY, USA: ACM, 1998, pp. 604–613.

[2] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Symp. on Comput. Geometry*, ser. SoCG '04. New York: ACM, 2004, pp. 253–262.

[3] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: Efficient indexing for high-dimensional similarity search," in *Very Large Data Bases*, ser. VLDB '07. Vienna, Austria: VLDB Endowment, 2007, pp. 950–961.

[4] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, "Approximate nearest neighbor search on high dimensional data – experiments, analyses, and improvement," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 8, pp. 1475–1488, 2020.

[5] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, "Practical and Optimal LSH for Angular Distance," in *Neural Inf. Process. Syst.*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 1225–1233.

[6] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," in *Symp. on Theory of Comput.*, ser. STOC '98. New York, NY, USA: ACM, 1998, pp. 327–336.

[7] V. Satuluri and S. Parthasarathy, "Bayesian locality sensitive hashing for fast similarity search," ser. VLDB '12, vol. 5, no. 5. VLDB Endowment, 2012, pp. 430–441.

[8] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng, "Query-aware locality-sensitive hashing for approximate nearest neighbor search," ser. VLDB '15, vol. 9, no. 1. VLDB Endowment, 2015, pp. 1–12.

[9] M. Charikar and P. Siminelakis, "Hashing-based-estimators for kernel density in high dimensions," in *Found. of Comput. Sci.*, ser. FOCS '17. Berkeley, CA, USA: IEEE, 2017, pp. 1032–1043.

[10] M. R. Pillutla, N. Raval, P. Bansal, K. Srinathan, and C. V. Jawahar, "Lsh based outlier detection and its application in distributed setting," in *Proc. of Inf. and Knowl. Manage.*, ser. CIKM '11. New York, NY, USA: ACM, 2011, pp. 2289–2292.

[11] J. Sun, G. Li, and N. Tang, "Learned cardinality estimation for similarity queries," in *Proc. of the 2021 Int. Conf. on Manage. of Data*, ser. SIGMOD '21. New York, NY, USA: ACM, 2021, pp. 1745–1757.

[12] M. Mattig, T. Fober, C. Beilschmidt, and B. Seeger, "Kernel-based cardinality estimation on metric data," in *Proc. of the 21st Int. Conf. on Extending Database Technol.*, ser. EDBT '18. Vienna, Austria: EDBT Association, 2018, pp. 349–360.

[13] M. Aumüller, R. Pagh, and F. Silvestri, "Fair near neighbor search: Independent range sampling in high dimensions," in *Proc. of the 39th ACM Symp. on Princ. of Database Syst.*, ser. PODS'20. New York, NY, USA: ACM, 2020, pp. 191–204.

[14] M. Charikar, M. Kapralov, N. Nouri, and P. Siminelakis, "Kernel density estimation through density constrained near neighbor search," in *Symp. on Found. of Comput. Sci.* Virtual: IEEE, 2020, pp. 172–183.

[15] C. Luo and A. Shrivastava, "Arrays of (locality-sensitive) count estimators (ace): Anomaly detection on the edge," in *World Wide Web Conf.*, ser. WWW '18. Intern. WWW Conf., 2018, pp. 1439–1448.

[16] R. Spring and A. Shrivastava, "Mutual information estimation using lsh sampling," in *Proc. of the Int. Joint Conf. on Artif. Intell.*, ser. IJCAI '20. Yokohama, Japan: IJCAI Organization, 7 2020, pp. 2807–2815.

[17] H. Wang, J. Meng, L. Gong, J. Xu, and M. Ogihara, "Mp-rw-lsh: An efficient multi-probe lsh solution to anns-l1," *Proc. VLDB Endow.*, vol. 14, no. 13, pp. 3267–3280, sep 2021.

[18] P. Siminelakis, K. Rong, P. Bailis, M. Charikar, and P. Levis, "Rehashing kernel evaluation in high dimensions," in *Int. Conf. on Machine Learning*, ser. ICML '19, vol. 97. Lugano, Switzerland: PMLR, 09–15 Jun 2019, pp. 5789–5798.

[19] E. Lehmann and G. Casella, *Theory of Point Estimation, 2nd edition*. New York, NY, USA: Springer, 1998.

[20] H. Wang, J. Cao, L. Shu, and D. Rafiei, "Locality sensitive hashing revisited: Filling the gap between theory and algorithm analysis," in *Int. Conf. on Inform. & Knowl. Manage.*, ser. CIKM '13. New York, NY, USA: ACM, 2013, pp. 1969–1978.

[21] Princeton University Computer Science Department, "Audio dataset," http://www.cs.princeton.edu/cass/audio.tar.gz, 2006.

[22] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*, 2nd ed. Chapman and Hall/CRC, 2004.

[23] N. Hua, H. Zhao, B. Lin, and J. Xu, "Rank-indexed hashing: A compact construction of bloom filters and variants," in *Int. Conf. on Netw. Protocols*, ser. ICNP '08. Orlando, Florida: IEEE, 2008, pp. 73–82.

[24] L. Yann, C. Corinna, and J. B. Christopher, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/, 1994.

[25] S. Winder, M. Brown, N. Snavely, S. Seitz, and R. Szeliski, "Trevi: Local Image Descriptors Data," http://phototour.cs.washington.edu/patches/default.htm, 2007.

[26] L. Amsaleg and H. Jégou, "Datasets for ANN neighbor search," http://corpus-texmex.irisa.fr/, 2010.

[27] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for Word representation," https://nlp.stanford.edu/projects/glove/, 2014.

[28] A. Babenko and V. Lempitsky, "Deep: Datasets of deep descriptors," http://sites.skoltech.ru/compvision/noimi/, 2016.

[29] D. W. Scott, "Scott's rule," *WIREs Computational Statistics*, vol. 2, no. 4, pp. 497–502, 2010.

[30] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Very Large Data Bases*, ser. VLDB '99. San Francisco, CA, USA: VLDB Endowment, 1999, pp. 518–529.

[31] B. Chen, Y. Xu, and A. Shrivastava, "Fast and accurate stochastic gradient estimation," in *Neural Inform. Process. Systems*, ser. NeurIPS '19, vol. 32. Vancouver, BC, Canada: NeurIPS Foundation, 2019.

[32] Z. Xu, B. Chen, C. Li, W. Liu, L. Song, Y. Lin, and A. Shrivastava, "Locality sensitive teaching," in *Neural Inform. Process. Systems*, vol. 34. Virtual: NeurIPS Foundation, 2021, pp. 18 049–18 062.

[33] A. Joly and O. Buisson, "A posteriori multi-probe locality sensitive hashing," in *Proc. of the 16th ACM Int. Conf. on Multimedia*, ser. MM '08. New York, NY, USA: ACM, 2008, pp. 209–218.

[34] K. Lu, H. Wang, Y. Xiao, and H. Song, "Why locality sensitive hashing works: A practical perspective," *Inform. Process. Lett.*, vol. 136, pp. 49–58, 2018.

[35] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE T. Pattern Anal. and Mach. Intell.*, vol. 42, no. 4, pp. 824–836, 2020.