Fast Diverse Nearest Neighbor Search

Justin Y. Chen
MIT
justc@mit.edu

Soham Nagawanshi
Texas A&M University
ndsoham@tamu.edu

Shenghao Xie
Carnegie Mellon University
xsh1302@gmail.com

Haike Xu
MIT
haikexu@mit.edu

Alan Zhou
Texas A&M University
alan.zhou.0@tamu.edu

Samson Zhou
Texas A&M University
samsonzhou@gmail.com

Abstract

In the k-diverse nearest neighbor search problem, the input is a dataset in which each point is assigned a color, and given a query q, the goal is to retrieve k approximate nearest neighbors with distinct colors. This color diversity metric is essential for information retrieval tasks where results are required to have different categories, e.g., recommending products from different sellers. Previous state-of-the-art solution by Anand et al. [ICML 2025] constructs a diversity-aware graph to meet this requirement. However, they suffer from an $\mathcal{O}(k^2)$ multiplicative query time overhead, creating a computational bottleneck when k is large.

To break this barrier, we propose a novel color bucketing framework that selects a subset of colors in each bucket and instantiates an independent approximate nearest neighbor search algorithm on the corresponding points. Inspired by group testing, we provide a randomized color sampling scheme that achieves an $\mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)}\log k\right)$ overhead in Euclidean spaces and an $\mathcal{O}\left(k\log k\right)$ overhead in general metric spaces, where c is the approximation factor, reducing a factor of k. In addition, the space usage to store the data structure matches or improves upon prior methods. Empirical results on semi-synthetic and real-world datasets demonstrate that our approach achieves significantly faster search times and higher recall compared to state-of-the-art diversity search baselines.

Furthermore, we introduce a dataset-oblivious deterministic bucketing scheme using expander graphs for a relaxed diverse search problem that only requires $k(1-\varepsilon)$ colors. Here, dataset-oblivious means that the color bucket construction does not depend on the spatial configuration of the dataset. We then establish an $\Omega(k^2)$ lower bound on the number of buckets for any dataset-oblivious deterministic bucketing scheme that guarantees k points of distinct colors within c times the optimal radius. We subsequently circumvent this by a dataset-aware adaptive segment tree bucketing scheme with an overhead of only $\mathcal{O}(k\log N)$, where N is the number of colors. As an extension of our results, we solve the problem with general diversity metric, where the goal is to maximize the minimum pair-wise distance of the k solutions, and improve the query time.

1 Introduction

In the nearest neighbor search (NNS) problem, the input is a set P of n points in a metric space $\mathcal{X}$, and the goal is to construct a data structure that, for any query q, reports a point $p \in P$ with minimum distance to q. This problem is a fundamental data structure problem with wide-ranging applications in modern computational tasks. In recommendation systems, NNS is a cornerstone

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

for collaborative filtering and content-based methods, which are significant in identifying the most relevant answer for a user's query. In computer vision, NNS facilitates large-scale image retrieval and facial recognition by matching feature vectors extracted from deep neural networks to massive databases of known descriptors [JDSP10]. Furthermore, in the recent rise of generative AI and Large Language Models (LLMs), NNS serves as an important primitive to retrieve contextually relevant documents from vector databases [MY18, KLJ+19] and has been used to accelerate attention computation in Transformer-based architectures [HJK+23]. Given these applications, we have seen a surge of interest in developing efficient algorithms for NNS [IM98, SDI06, AI08, MY18, IM18, SDS+19, IX23, AIK+25].

Given that existing data structures for *exact* nearest neighbor search are inefficient for high-dimensional point sets, the following relaxed c-approximate nearest neighbor (c-ANN) problem is widely adopted, where we allow to report a c-approximation to the exact nearest neighbor.

Definition 1.1 (The c-ANN problem)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$, the goal is to construct a data structure that satisfies the following: given any query q, if the nearest neighbor of q is within distance r, it reports a point $\widehat{p}$ that satisfies $\mathrm{dist}(\widehat{p}, q) \leq cr$.*

Classical algorithms address the c-ANN problem with efficient space usage and update time, including those by [IM98, AI08] using locality sensitive hashing (LSH) in Euclidean spaces, and [KL04] using navigating nets in general metric spaces. More recently, graph-based algorithms such as HNSW [MY18], NGT [IM18], and DiskANN [SDS+19] have gained increasing attention, as they achieve state-of-the-art performance in practice [ABF24], provable guarantees [IX23], and apply to general metrics. These methods build a navigation graph over the dataset P, and the query procedure performs a greedy search on this graph to locate points close to the query.

In many real-world applications, it is often necessary to return not just a single nearest neighbor but several of them, for instance, a search engine typically provides multiple websites in response to a user's query. While minimizing the distance of the reported points is the primary goal, it has been observed that the returned results can be redundant or nearly identical [CG98]. This issue is especially pronounced in applications in recommendation systems, where many similar versions of the same product, seller, or document may exist, e.g., a user buying shoes may prefer to be shown options from different sellers. This motivates the goal of diversity-aware search, as emphasized by Google's 2019 update, which ensures that no more than two pages from the same site are shown [Lia19]. To accurately capture this requirement, we assign each point a *color*, representing a categorical label such as a site ID or a seller [AIK+25], and require the algorithm to return a set S of k points with distinct colors, while keeping their distances to the query q near-optimal. We next provide the formal definition of the k-diverse c-ANN problem.

Definition 1.2 (The k-diverse c-ANN problem)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$, where each point is assigned a color, the goal is to construct a data structure that satisfies the following: for any query q, if the smallest ball around q which contains k points of distinct colors has radius at most r, it returns a set $\widehat{S}$ of k points of distinct colors that satisfies $\max_{p \in \widehat{S}} \mathrm{dist}(p, q) \leq cr$. A randomized data structure for this problems succeeds with probability $1 - \delta$ if, for any fixed q, the preceding condition holds except with probability at most δ.*

To solve the k-diverse c-ANN problem, the early work of [CG98] introduced a two-stage diversity-based re-ranking approach: they first select m closest points to the query, and then post-process the m points to identify k diverse points. Although widely used, this pipeline has a fundamental efficiency drawback. In practice, the algorithm must retrieve $m \gg k$ points (with $m = \Omega(n)$ in the worst case) to ensure that we recover the k diverse points. To address this issue, subsequent work modified classical c-ANN algorithms to return k diverse points, including the algorithm in [AAI+13], which stores k diverse points in each LSH bucket, and the algorithm in [AIK+25], which constructs disk-ANN graphs subject to diversity constraints. These results bypass the dependence on m, demonstrating that diversity-aware search can be achieved without first retrieving all relevant points. However, their query time still scales as k^2, which becomes inefficient when k is large, e.g., in recommending hundreds of diverse products to a user. This raises a natural question:

Is it possible to bypass the quadratic dependence on k in the query time for the k-diverse c-ANN problem?

In this work, we provide algorithms that significantly improve on the k factors, breaking the k^2 barrier and achieving efficient query time. The overheads are sublinear in k in Euclidean spaces and linear in

k in general metric spaces, with space comparable to or better than previous work. Therefore, our approaches support applications to large-scale recommendation and retrieval systems. Importantly, we provide a unified color bucketing framework that black-boxes standard c-ANN algorithms to solve diverse search problems with efficient computational overhead.

1.1 Our Results

In this section, we introduce the guarantees of our k-diverse c-ANN algorithms. We present the space and query time by the computational overhead compared to classical c-ANN algorithms, for instance, the LSH-based c-ANN algorithm in Euclidean spaces uses $\mathcal{O}\left(dn^{\frac{1}{c^2}+o(1)}\right)$ time, while our algorithm for k-diverse c-ANN in Euclidean spaces uses $\mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)}\log k\cdot dn^{\frac{1}{c^2}+o(1)}\right)$ time, then the overhead is $\mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)}\log k\right)$. This notion allows a fair comparison with previous work [AAIM13, AAI+13, AIK+25], since those algorithms rely heavily on the data structures in classical c-ANN algorithms while our algorithms directly apply c-ANN algorithms as black boxes. We first present the result of our randomized algorithm, which achieves a query time sublinear in k.

Theorem 1.3 (Informal statement of Theorem 2.1)**.** *Given a set P of n points in a metric space $(\mathcal{X},\mathrm{dist})$, there exists a randomized algorithm that solves the k-diverse c-ANN problem (see Definition 1.2) with probability at least 0.99. The algorithm achieves the following overheads:*

- $\mathcal{O}\left(k^{1-1/c^2-o(1)}\log k\right)$ *in query time and* $\mathcal{O}\left(\log k\right)$ *in space in Euclidean spaces;*
- $\mathcal{O}\left(k\log k\right)$ *in query time and* $\mathcal{O}\left(\log k\right)$ *in space in general metric spaces.*

For comparison, the algorithm of [AAI+13] incurs an $\mathcal{O}\left(k\log k\right)$ overhead (implicitly) in Euclidean spaces, while our algorithm only has $\mathcal{O}\left(k^{1-1/c^2-o(1)}\log k\right)$, achieving sublinear dependence on k. In general metric spaces, the algorithm of [AIK+25] incurs an $\mathcal{O}\left(k^2\right)$ overhead, while our algorithm only has $\mathcal{O}\left(k\log k\right)$, reducing a k factor and bypassing the quadratic barrier. Furthermore, the $\log k$ factor can be replaced by $\log\frac{1}{\varepsilon}$ for the relaxed (k,ε)-diverse c-ANN problem, where we allow the output to contain only $\lceil(1-\varepsilon)k\rceil$ points with distinct colors. Our results are based on a simple and unified color bucketing framework, which selects a subset of colors in each bucket and implements an independent c-ANN algorithm on the corresponding points.

We also provide a deterministic algorithm via expander graphs in general metric spaces, achieving improved dependence on k for the relaxed (k,ε)-diverse c-ANN problem.

Theorem 1.4 (Informal statement of Theorem 3.3)**.** *Given a set P of n points in a metric space $(\mathcal{X},\mathrm{dist})$, there is a deterministic algorithm that solves the (k,ε)-diverse c-ANN problem with an $\mathcal{O}\left(k^{1+o(1)}\gamma\right)$ overhead in query time and an $\mathcal{O}\left(\gamma\right)$ in space, where $\gamma=\left(\frac{\log k\log N}{\varepsilon}\right)^{\mathcal{O}\left(\sqrt{\log k}\right)}$.*

We remark that the color bucketing scheme in Theorem 1.4 is dataset-oblivious, meaning that the color assignment only depends on the color set and problem parameters, and is oblivious to the spatial configuration of the dataset and the query point. Requiring all k distinct colors imposes a stronger condition on such a scheme. If every bucket containing a color x also contains one of $k-1$ competing colors, we can place those competitors close enough to the query to mask x in every such bucket, even when x is needed for a valid answer. A deterministic, dataset-oblivious scheme must therefore isolate each color from every set of $k-1$ other colors. This is precisely the $(k-1)$-disjunctness property, which gives the following lower bound on the number of buckets.

Theorem 1.5 (Informal statement of Theorem 3.4)**.** *Any dataset-oblivious deterministic bucketing scheme that solves the k-diverse c-ANN problem (see Definition 1.2) with $N=\Omega(k^2)$ colors must evaluate $L=\Omega\left(k^2\frac{\log N}{\log k}\right)$ buckets.*

To circumvent the quadratic barrier, we design a dataset-aware deterministic scheme that queries the buckets adaptively based on the dataset and the query, achieving an $\mathcal{O}\left(k\log N\right)$ overhead.

Theorem 1.6 (Informal statement of Theorem 3.5)**.** *Given a set P of n points in a metric space $(\mathcal{X},\mathrm{dist})$, there is an adaptive deterministic algorithm that solves the k-diverse c-ANN problem (see Definition 1.2) with an $\mathcal{O}\left(k\log N\right)$ overhead in query time and an $\mathcal{O}\left(\log N\right)$ overhead in space in general metric spaces.*

We summarize our main results for the k-diverse problem in Table 1. We improve on the dependence on k in query time while maintaining comparable or better space compared to prior work.

References	Metric	Query time	Space	Remark
[AAI+13]	Euclidean	$\tilde{\mathcal{O}}(k)$	$\tilde{\mathcal{O}}(1)$	Randomized
Theorem 1.3	Euclidean	$\tilde{\mathcal{O}}\left(k^{1-1/c^2-o(1)}\right)$	$\tilde{\mathcal{O}}(1)$	Randomized
[AIK+25]	General	$\mathcal{O}\left(k^2\right)$	$\mathcal{O}(k)$	Deterministic
Theorem 1.3	General	$\tilde{\mathcal{O}}(k)$	$\tilde{\mathcal{O}}(1)$	Randomized
Theorem 1.6	General	$\mathcal{O}(k \log N)$	$\mathcal{O}(\log N)$	Deterministic

Table 1: Comparison with previous work. We use $\tilde{\mathcal{O}}(\cdot)$ to hide the $\log k$ factor.

1.2 Technical Novelties

The primary challenge in diverse nearest neighbor search is satisfying categorical constraints without incurring prohibitive search costs. While early post-processing approaches [CG98] completely decoupled search from diversity, they required an exhaustive scan of up to $\Omega(n)$ points to guarantee k diverse results. Recent methods [AAIM13, AAI+13, AIK+25] instead embed diversity constraints directly into the search index, but this approach forces the query procedure to redundantly traverse dense, heavily overlapping spatial neighborhoods to discover new colors, creating a $\mathcal{O}\left(k^2\right)$ query time bottleneck. To systemically overcome this quadratic bottleneck, we decouple the spatial search from the diversity constraints, thereby establishing novel connections between diverse retrieval, graph theory, group testing, and coding theory.

Color bucketing framework via group testing. We first show that querying for k diverse points can be viewed as a geometric analogue of non-adaptive group testing [Dor43, INR10, DH99], where one seeks to identify a small set of "defective" items through parallel aggregate tests. By treating the k closest uniquely colored points as the defectives and a standard black-box c-ANN oracle as the group test, we introduce a randomized color bucketing scheme. We prove that assigning colors to $\mathcal{O}(k \log k)$ buckets independently with probability $\frac{1}{k}$ isolates target colors from spatially closer but redundant points with high probability, thereby bypassing the previous $\mathcal{O}\left(k^2\right)$ barrier.

Information-theoretic lower bounds for exact search. To understand the fundamental limits of exact deterministic retrieval, we formalize the study of *dataset-oblivious* bucketing schemes, where color assignments are fixed independently of the dataset. We identify a geometric "masking" phenomenon, where target colors are consistently occluded by closer, redundant points across queries. We prove a strict equivalence between avoiding this masking and algebraic coding theory: for any static scheme to guarantee exact recovery, its color-to-bucket incidence matrix must form a *separable matrix*, i.e., a $(k-1)$-disjunct matrix or $(1, k-1)$-cover-free family, meaning no column is contained in the union of any $k-1$ others. By translating spatial diversity into this combinatorial requirement, we leverage bounds on disjunctive codes from extremal combinatorics to establish an $\Omega\left(k^2 \frac{\log N}{\log k}\right)$ lower bound on the index size, thus proving that the quadratic overhead observed in prior deterministic algorithms is an inherent information-theoretic barrier.

Bypassing bounds via expander graphs. Motivated by this impossibility result, we consider the (k, ε)-diverse problem and design a dataset-oblivious scheme based on unbalanced bipartite expander graphs. We show that the vertex expansion properties of these graphs preserve spatial metric neighborhoods sufficiently well to guarantee the isolation of most target colors, without strictly requiring the rigid $(k-1)$-disjunctness property. This relaxation allows us to safely drop the deterministic query overhead to $\mathcal{O}\left(k^{1+o(1)}\right)$.

Adaptive segment trees for deterministic k-diverse search. To achieve *exact* k-diverse recovery without the coding-theoretic penalty, we shift to a dynamic, dataset-aware approach. We construct an adaptive segment tree over the color space that interleaves categorical partitioning with spatial priority search. During a query, the algorithm dynamically routes down the tree, computing mutually disjoint partitions of the remaining active colors conditioned on the geometry of candidates retrieved

on the fly. This adaptive routing sidesteps the worst-case covering bounds of static matrices entirely, achieving exact deterministic recovery with only an $\mathcal{O}(k \log N)$ overhead.

Sub-quadratic thresholding for continuous diversity. We extend our framework to spatial diversity objectives maximizing minimum pairwise distance. Prior LSH-based pipelines [AAIM13] rely on Greedy Maximal Margin selection, incurring $\mathcal{O}(mk)$ time from exhaustive distance checks. We replace this with an LSH-augmented thresholding scheme that selects points via approximate nearest-neighbor queries while maintaining a dynamic index, eliminating explicit pairwise comparisons. This gives a constant-factor approximation in sub-quadratic time, reducing overhead to $\mathcal{O}\left(mk^{1/c^2}\right)$.

1.3 Notations

We introduce the notations used in this paper. We refer the readers to Section A for a preliminary introduction on standard c-ANN algorithms. Let P be a set of n points in a metric space $(\mathcal{X}, \mathrm{dist})$, e.g., the Euclidean space $(\mathbb{R}^d, \ell_2)$ and the Hamming metric space $(\{0,1\}^d, \ell_1)$. Let $\mathrm{dist}(x, S) = \min_{s \in S} \mathrm{dist}(x, s)$ be the distance between a point $x \in P$ and a set $S \subset P$. Let $\mathcal{C}$ be the set of colors with size N, and for a point $p \in P$, let $c(p)$ be its color. Let Δ denote the aspect ratio, which is the ratio between the diameter of P and the minimum pair-wise distance of P, i.e., $\min_{p,p' \in P} \mathrm{dist}(p, p')$. Let $B(p, r) = \{x \in \mathcal{X}, \mathrm{dist}(x, p) < r\}$ be the ball of radius r centered at p. Let $\mathbf{ddim}$ denote the doubling dimension of P, which is the smallest integer such that for any ball of radius r centered at some point $p \in P$, the set $B(p, r) \cap P$ can be covered by at most $2^{\mathbf{ddim}}$ balls of radius $r/2$.

2 Randomized Algorithms for Diverse Nearest Neighbor Search

In this section, we introduce our color sampling approach inspired by group testing. In group testing, the goal is to identify a small set of "defective" items by grouping together subsets of items and testing them collectively, rather than testing each one individually. A positive test guarantees that at least one defective item lies in the group, while a negative test eliminates the entire group. Suppose that there are k defectives, a randomized algorithm for group testing is by sampling an item with probability $\frac{1}{k}$ in each test, and this identifies the k defectives using $\mathcal{O}(k \log k)$ tests.

In our setting, the defectives correspond to the k distinct colors of the target set, i.e., the set of k points with distinct colors and has closest distance to the query q. Based on this observation, in each bucket, we sample each color with probability $\frac{1}{k}$ and implement a separate c-ANN algorithm. The guarantees of the c-ANN algorithm and group testing then ensure that $\mathcal{O}(k \log k)$ randomized buckets collectively select a set of nearest neighbors with distinct colors, satisfying the requirement of k-diverse c-ANN problem. We now formalize this approach in Algorithm 1. We summarize the guarantees of our algorithm in Theorem 2.1 and defer the formal analysis to Section B. We note that our algorithm achieves sublinear overhead in k in Euclidean spaces and Hamming spaces and near-linear overhead in general metric spaces.

Algorithm 1 Diversity search by color sampling

Require: Input dataset $P = \{p_1, \cdots, p_n\}$, each with a color $c \in \mathcal{C}$, an integer $k \leq |\mathcal{C}|$
Ensure: Data structure to solve the k-diverse nearest neighbor problem
1: Let $L \leftarrow \mathcal{O}(k \log k)$
2: **for** $i \in [L]$ **do**
3: Sample each color $c \in \mathcal{C}$ with probability $\frac{1}{k}$
4: Let $\mathcal{C}_i$ be the set of sampled colors, and let P_i be the set of points with colors in $\mathcal{C}_i$
5: Maintain a data structure that solves the c-ANN problem for P_i
6: **end for**
7: **for** each query q **do**
8: Let $\widehat{p_i}$ be the reported c-ANN of bucket i for each $i \in [L]$
9: **Return:** k nearest points to q in $\{\widehat{p_1} \ldots, \widehat{p_L}\}$ of distinct colors
10: **end for**

Theorem 2.1. *Given a set P of n points, there exist algorithms that solve the k-diverse c-ANN problem (see Definition 1.2) with probability 0.99 in the following spaces:*

- *Euclidean spaces* $(\mathbb{R}^d, \ell_2)$*: the algorithm uses* $\mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)} \log k \cdot dn^{\frac{1}{c^2}+o(1)}\right)$ *query time and* $\mathcal{O}\left(\log k \cdot dn^{1+\frac{1}{c^2}+o(1)}\right)$ *space.*
- *Hamming spaces* $(\{0,1\}^d, \ell_1)$*: the algorithm uses* $\mathcal{O}\left(k^{1-\frac{1}{c}-o(1)} \log k \cdot dn^{\frac{1}{c}+o(1)}\right)$ *query time and* $\mathcal{O}\left(\log k \cdot dn^{1+\frac{1}{c}+o(1)}\right)$ *space*
- *General metric spaces* $(\mathcal{X}, \mathrm{dist})$*: the algorithm uses* $\mathcal{O}\left(k \log k \cdot (4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$ *query time and* $\mathcal{O}\left(\log k \cdot n(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ *space, where* $\mathbf{ddim}$ *is the doubling dimension and* Δ *is the aspect ratio.*

We extend our method to solve two variants of the k-diverse c-ANN problem. First, using only $L = \mathcal{O}\left(k \log \frac{1}{\varepsilon}\right)$ buckets in our algorithm solves the relaxed (k, ε)-diverse c-ANN problem (see Definition B.5), improving the $\log k$ factor in the overhead to a $\log \frac{1}{\varepsilon}$ factor (see Section B). Second, sampling each color with probability $\frac{k'}{2k}$ and return k' c-ANNs, instead of one, in each bucket solves the (k, k')-diverse c-ANN problem (see Definition C.1), which incurs an $\mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)}(k')^{\frac{1}{c^2}+o(1)} \log k\right)$ overhead in query time, improving the overheads obtained by [AIK+25] (see Section C). These results demonstrate the adaptivity of our method.

3 Deterministic Algorithms for Diverse Nearest Neighbor Search

In this section, we provide the construction and lower bound for deterministic algorithms for the k-diverse c-ANN problem. To obtain deterministic algorithms, we replace random sampling with deterministic bucketing and querying constructions.

3.1 Graph Expander Construction

First, we provide our dataset-oblivious color bucketing scheme via graph-expander. Expanders are bipartite graphs with strong neighborhood expansion properties: any small subset of left vertices has many right neighbors. We next formalize the definition of a (k, Γ)-expander.

Definition 3.1 (Expander)**.** *Let $G = (A, B, E)$ be a bipartite graph, where A is the set of left vertices with size N and left-degree D, B is the set of right vertices with size M, and $E : [N] \times [D] \to [M]$ is the edge list. Let $k \le |A|$ and Γ be two integers. G is a (k, Γ)-expander if for every set $S \subset A$ such that $|S| \le k$, $|N(S)| \ge \Gamma \cdot |S|$, where $N(S)$ is the neighborhood of S.*

The following theorem shows a deterministic construction of expanders with small degree introduced by [GUV09], which ensures the efficiency of our algorithm.

Theorem 3.2 ([GUV09])**.** *For all positive integers k, N, all $\varepsilon > 0$, and all $\gamma \in (0, \frac{\log x}{\log \log x})$ for $x = \frac{\log k \log N}{\varepsilon}$, there is an explicit $(k, (1-\varepsilon)D)$-expander $G = (A, B, E)$ with $E : [N] \times [D] \to [M]$ such that $D = \mathcal{O}\left(\left(\frac{\log k \log N}{\varepsilon}\right)^{1+\frac{1}{\gamma}}\right)$ and $M \le D^2 k^{1+\gamma}$.*

In our setting, the left vertices in A represent the colors, and the right vertices in B represent the buckets, and each bucket includes the points with colors connected to it in the expander. The expander property in Theorem 3.2 guarantees that for every set of k colors, there exist at least $k(1-\varepsilon)D$ buckets in their neighbor set, ensuring that at least $k(1-\varepsilon)$ colors are found since each color has degree D. In addition, we query at most $M \le D^2 k^{1+\gamma}$ buckets, giving an overhead of roughly $\mathcal{O}\left(k^{1+o(1)}\right)$. We note that the DiskANN algorithm with provable guarantee in [IX23] is deterministic, so combining it with our expander-based color bucketing approach gives a deterministic algorithm. We defer the formal discussion of the above process to Algorithm 5 in Section D.

Theorem 3.3. *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$ with doubling dimension $\mathbf{ddim}$, aspect ratio Δ, and N colors, there is an algorithm that solves the (k, ε)-diverse c-ANN problem using $\mathcal{O}\left(\gamma \cdot n(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ space, where $\gamma = \left(\frac{\log k \log N}{\varepsilon}\right)^{\mathcal{O}\left(\sqrt{\log k}\right)}$, and query time $\mathcal{O}\left(k^{1+o(1)}\gamma \cdot (4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$.*

3.2 Lower Bound for Dataset-Oblivious Bucketing

In this section, we show a lower bound on deterministic dataset-oblivious color bucketing schemes. The key observation is that for dataset-oblivious scheme that solves the k-diverse problem, the color assignment must satisfy k-disjunctness. If not, there exists at least a color $x^* \in \mathcal{C}$ and a subset of colors $S^* \subset \mathcal{C} \setminus \{x^*\}$ of size $k-1$, such that every bucket containing x^* also contains at least one color from S^*. Because the scheme is dataset-oblivious, it must be universally correct for any spatial configuration of the dataset. Then consider dataset P and query q satisfying the following properties:

(1) For every color $\gamma \in \mathcal{C}$, there is exactly one point $p_\gamma \in P$ with color γ.

(2) For all $y \in S^*$, $\operatorname{dist}(q, p_y) = 1$.

(3) $\operatorname{dist}(q, p_{x^*}) = c + \varepsilon$, for an arbitrarily small $\varepsilon > 0$.

(4) Let $K^* = S^* \cup \{x^*\}$. For all $z \in \mathcal{C} \setminus K^*$, $\operatorname{dist}(q, p_z) = c(c+\varepsilon) + \varepsilon$.

In the above construction, any point outside K^* is not a c-ANN, so we must return exactly K^* in our solution. However, for any bucket that contains x^*, another color $y \in S^*$ also exists, then the c-ANN algorithm would not report x^* since it is not a c-ANN in that bucket due to $\operatorname{dist}(q, p_y) = 1$, which leads to a contradiction since x^* must be reported. Then by information-theoretic lower bounds for k-disjunctness, we have the following lower bound. We defer the formal analysis to Section E.

Theorem 3.4. *Assuming $k \geq 3$ and $k^2 \frac{\log N}{\log k} \leq C \cdot N$ for some constant C, any deterministic dataset-oblivious bucketing scheme that correctly solves the k-diverse c-ANN problem for all valid datasets must evaluate $L = \Omega\left(k^2 \frac{\log N}{\log k}\right)$ buckets.*

3.3 Adaptive Segment Tree Bucketing

To circumvent the $\Omega(k^2)$ barrier, we introduce an adaptive deterministic algorithm that selectively queries buckets based on the specific spatial configuration and color assignment of the dataset, using a segment tree with N leaves over the color space. Unlike the relaxed guarantee in Section 3.1, the segment tree returns k points of pairwise distinct colors, while retaining the same c-approximation to the optimal radius. To construct the tree, we assign a different color to each of the N leaves, and for each node u in $\mathcal{T}$, we include all colors associated with the leaves of the subtree rooted at u. Then a separate c-ANN algorithm is ran on each node.

When receiving a query q, the algorithm dynamically maintains a priority queue Q that stores a list of pairs $(\widehat{p_u}, u)$ in the order of their distance to q. Here, $(\widehat{p_u}, u)$ denotes a candidate point $\widehat{p_u}$ and the node u that reports $\widehat{p_u}$. We initialize the queue by querying the root node. Next, in each iteration, we first add the minimum priority element $(\widehat{p_u}, u)$ in Q to the result set, then query a group of descendent nodes of u to add new candidates to Q such that we avoid $c(p_u)$ while ensuring that all other colors in u are considered. To achieve these requirements, we look at the downward path $v_0 = u, v_1, ..., v_m$ in $\mathcal{T}$ from u to the leaf corresponding to $c(\widehat{p_u})$. Note that for each $i \in [m]$, v_i contains $c(\widehat{p_u})$, so we instead query the other child w_{i-1} of v_{i-1} that is not v_i and add its reported point to Q. The key observation is that $\{w_i \mid i \in [m-1]\}$ forms a mutually disjoint partition of the colors in u except for $c(\widehat{p_u})$. Therefore, we maintain an invariance in each iteration that the remaining nodes in Q forms a mutually disjoint partition of $\mathcal{C} \setminus \mathcal{C}_\mathcal{R}$, where $\mathcal{C}_\mathcal{R}$ is the set of colors in the current result set. This implies that a new valid c-ANN with a distinct color is extracted from Q in each iteration, so the algorithm terminates in k iterations. Since the length of the path is at most $\log N$, we have $\mathcal{O}(k \log N)$ overhead in query time in total. We formalize the above procedure in Algorithm 2 and defer the formal analysis to Section F.

Combining our algorithm with diskANN [IX23], we have the following result.

Theorem 3.5 (k-diverse c-ANN with general metrics)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \operatorname{dist})$ with doubling dimension $\mathbf{ddim}$ and aspect ratio Δ, there is a deterministic algorithm that solves the k-diverse c-ANN problem (see Definition 1.2) for $c = \frac{\alpha+1}{\alpha-1} + \varepsilon$ using $\mathcal{O}\left(k \log N \cdot (4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$ query time and $\mathcal{O}\left(\log N \cdot n(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ space.*

Algorithm 2 Adaptive query scheme for k-diverse c-ANN

Require: Query $q \in \mathcal{X}$, target diversity $k \leq N$, segment tree $\mathcal{T}$
Ensure: Solution to k-diverse search

Initialize a priority queue Q, ordering pairs $(\widehat{p_u}, u)$ by $dist(q, \widehat{p_u})$. Query $\mathcal{A}_{\text{root}}$ to obtain a c-ANN $\widehat{p_{\text{root}}}$ and insert $(\widehat{p_{\text{root}}}, \text{root})$ into Q
Initialize result set $\mathcal{R} \leftarrow \emptyset$ and active colors $\mathcal{C}_{\mathcal{R}} \leftarrow \emptyset$
while $|\mathcal{R}| < k$ **do**
  Extract the pair $(\widehat{p_u}, u)$ with minimum distance to q from Q
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\widehat{p_u}\}$, $\mathcal{C}_{\mathcal{R}} \leftarrow \mathcal{C}_{\mathcal{R}} \cup \{c(\widehat{p_u})\}$
  Let $v_0 = u, v_1, ..., v_m$ be the downward path in $\mathcal{T}$ from u to the leaf corresponding to $c(\widehat{p_u})$
  for $i = 0$ to $m - 1$ **do**
    Let w_i be the child of v_i that is not v_{i+1}.
    if $P_{w_i} \neq \emptyset$ **then**
      Query $\mathcal{A}_{w_i}$ to obtain a c-ANN $\widehat{p_{w_i}}$ and insert $(\widehat{p_{w_i}}, w_i)$ into Q
    end if
  end for
end while
return $\mathcal{R}$

4 Experiments

In this section, we present experiments on semi-synthetic and real datasets to demonstrate the validity and practicality of our approach. We performed all of our experiments on a Windows Intel i7 desktop with 32 GB of RAM. Empirical results in [AIK+25] demonstrate that their diverse search algorithms outperform algorithms that run standard search and perform post-processing on reported candidates. Thus, we use their methods as a baseline for comparison. Our algorithms achieve faster query time and more accurate answers on both semi-synthetic and real-world datasets, validating the optimality of our color sampling framework.

Datasets. We run our algorithms on the following datasets:

(1) *ArXiv OpenAI Embeddings* [Sim24]: Like [AIK+25], we consider the public real-world ArXiv dataset containing 1536-dimensional vectors obtained from OpenAI embeddings of paper abstracts. Figure 1 uses the 100K partition and Figure 3 uses the full 2M partition. Since this dataset does not contain color information, we synthetically assign colors to each vector using the probabilistic distribution proposed by [AIK+25]. Specifically, for each vector, its color is selected uniformly at random from a primary set $\{1, 2, 3\}$ with probability 0.9, and is selected uniformly from an expanded set $\{4, \ldots, 1000\}$ with probability 0.1. This assignment is motivated by very skewed real-world scenarios, i.e., a dominant seller in a market, that necessitate diversity in search results.

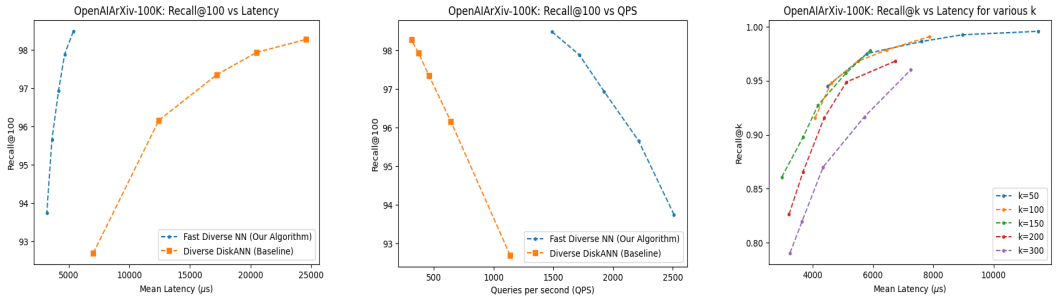
(2) *LAION* [SVB+21]: Second, we consider the real-world LAION dataset containing 4 millions image embeddings of 512 dimensions, where the embeddings are generated using CLIP [RKH+21]. This dataset has previously been used to examine graph-based nearest neighbor search algorithms for hybrid search [PKGZ24]. Each image in the dataset has multiple keywords that describe its content, which are used to define the colors in our experiments. See Figure 2c for the color distribution.

(3) *ArXiv-for-fanns* [IBC+25]: We also consider the real-world ArXiv dataset containing 4096-dimensional vectors obtained by embedding abstracts via a Transformer-based model [Nov25, ZW24]. Figure 4 uses the 100K partition and Figure 5 uses the full 2.8M partition. For feasibility, we use a 100K partition in our experiments. Previously, this dataset has been used to benchmark diverse nearest neighbor search algorithms [IBC+25]. Since this dataset is composed of paper abstract embeddings, each item is associated with a main category, e.g., math, computer science, quantum physics, etc. There are 30 unique categories, which we use as the "colors." See Figure 4c and Figure 5c for the color distribution.

Methods. For the OpenAI Embeddings dataset, we aim to retrieve $k = 100$ nearest neighbors with distinct colors. The LAION dataset only contains 28 unique categories, so we aim to retrieve $k = 5$ nearest neighbors of distinct colors. Recall that, [AIK+25]'s algorithm constructs a diverse

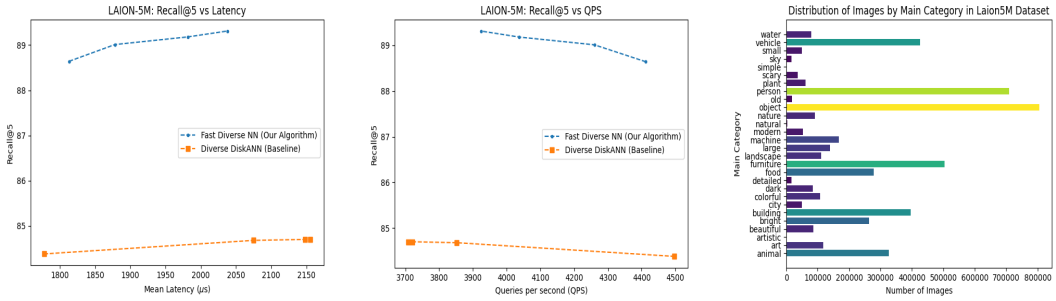
navigation graph based on the graph structure of DiskANN [SDS+19] and achieves state-of-the-art performance in diverse search. In contrast, our approach randomly groups the colors into buckets and uses the standard DiskANN graph in each bucket. As such, we demonstrate our algorithm's performance by using [AIK+25]'s algorithm as a baseline for comparison.

Evaluation. We follow the same procedure as [SDS+19] and [AIK+25] to measure the performance of our algorithm. For some query q, let the true k nearest neighbors of different colors be X and let the inferred neighbors be $\hat{X}$. Then, recall is defined as $\frac{|X \cap \hat{X}|}{k}$. For our algorithm, we vary the recall by changing the bucket count, and, for the baseline, we vary it by changing the search list size. For various recalls, we measure the mean latency in answering 20K queries for the OpenAI Embeddings dataset and 10K queries for the LAION and ArXiv-for-fanns datasets, allowing us to observe the recall-vs-latency tradeoff. We also analyze throughput by measuring queries per second (QPS) for both algorithms.



(a) Recall-latency tradeoff (b) Throughput tradeoff (c) Recall-latency tradeoff

Fig. 1: Comparison on OpenAI Embeddings 100K Dataset



(a) Recall-latency tradeoff (b) Throughput tradeoff (c) Image category distribution

Fig. 2: Comparison on real-world LAION dataset

Parameter setup. We configured the baseline algorithm through multiple tests on different graph degree (R) and list size (L) combinations. For both versions of the OpenAI Embeddings dataset, we used $R = 64$ and $L = 128$ and varied L during search between 100 and 500. For the ArXiv-for-fanns dataset, we used $R = 128$ and $L = 128$ and used $L \in \{3, 9, 27, 81, 243\}$ during search. For the LAION dataset, we utilized $R = 64$, $L = 64$, and used $L \in \{5, 10, 15, 20, 25\}$ during search time. For our algorithm, we used $R = 16$ and $L = 16$ for the OpenAI and Arxiv-for-Fanns Datasets and $R = 64$ and $L = 64$ for the LAION dataset. All evaluations used 8 threads during search time.

Results. Figure 1a and Figure 1b compare the recall-latency and recall-QPS tradeoff of our algorithm to the baseline's on the semi-synthetic OpenAI Embeddings dataset. Our algorithm's recall is in a higher range (93.75 - 98.48) than the baseline's recall (92.70 - 98.20). As latency increases, the gap between the algorithms widens, highlighting that our algorithm provides higher recall for the fixed latency. Figure 1c shows the recall-latency tradeoff for various k on the OpenAI Embeddings dataset. Note that these experiments utilized different parameters. Specifically, we set $R = 64$

and $L = 128$, assigned points from the same skewed distribution but with 1987 unique colors, and set buckets count approximately between $6k$ and $10k$. We notice that across the various values of k, our algorithm achieves generally good performance, with recall ranging from 79.05 to 99.57. Additionally, the latency is comparable to that in Figure 1, even with higher graph degree and list size, so the performance can likely be further optimized by increasing the bucket count.

Figure 2 compares the recall-latency and recall-QPS tradeoff of our algorithm to that of the baseline on the real-word LAION dataset. Our results are consistent with the other datasets, as we have a higher recall range (88.64 - 89.31) than the baseline (80.82 - 83.71). For the same latency and QPS, our algorithm produces higher recall. As latency increases, the gap between the algorithms grows. Figure 2c shows the distribution of main categories or "colors" in the experiments involving the LAION dataset. This distribution is considerably less skewed than the synthetically generated one for the OpenAI Embeddings dataset, but our algorithm maintains its advantage, highlighting that it can be deployed in practical tasks like finding k images of distinct objects.

5 Conclusion

Our color-bucketing framework reduces k-diverse nearest neighbor search to standard c-ANN queries. In general metric spaces, our randomized algorithm improves the query-time overhead from $\mathcal{O}\left(k^2\right)$ to $\mathcal{O}\left(k \log k\right)$, with an $\mathcal{O}\left(\log k\right)$ space overhead. As in prior work, the geometric parameters can still dominate the cost. For example, one general-metric implementation incurs a factor of $(4\alpha)^{\mathrm{ddim}}$ and also depends on the aspect ratio Δ. Since each bucket uses a c-ANN data structure as a black box, improvements in its dependence on either parameter carry over directly.

The main open question is the extent of how much the $\mathcal{O}\left(k \log k\right)$ query-time overhead can be improved for randomized approaches on general metrics. Our deterministic lower bound relies on a spatial configuration in which $k-1$ closer colors can mask a required color. To succeed on every such configuration, a dataset-oblivious bucketing scheme must isolate each color from every set of $k-1$ others. This forces its incidence matrix to be $(k-1)$-disjunct and requires $\Omega(k^2 \log N / \log k)$ buckets when there are $N = \Omega(k^2)$ colors. For each fixed query, however, independent random sampling suffices to recover k distinct colors with probability at least 0.99 using $\mathcal{O}\left(k \log k\right)$ buckets. Neither the deterministic argument nor known lower bounds for non-adaptive group testing establish a matching lower bound for randomized approaches. It also remains open whether the $\mathcal{O}\left(\log k\right)$ space overhead can be reduced while preserving the query-time guarantee.

Acknowledgments

Justin Y. Chen was supported by an NSF Graduate Research Fellowship under Grant No. 17453. Shenghao Xie is supported in part by NSF CCF-2335411, the work was completed when the author was affiliated with Texas A&M University. Samson Zhou is supported in part by NSF CCF-2335411 and gratefully acknowledges funding provided by the Oak Ridge Associated Universities (ORAU) Ralph E. Powe Junior Faculty Enhancement Award.

References

[AAI+13] Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, Sepideh Mahabadi, and Kasturi R. Varadarajan. Diverse near neighbor problem. In Guilherme Dias da Fonseca, Thomas Lewiner, Luis Mariano Peñaranda, Timothy M. Chan, and Rolf Klein, editors, *Symposium on Computational Geometry, SoCG*, pages 207–214. ACM, 2013. 2, 3, 4, 27, 29

[AAIM13] Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, and Sepideh Mahabadi. Real-time recommendation of diverse related articles. In Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo Baeza-Yates, and Sue B. Moon, editors, *22nd International World Wide Web Conference, WWW*, pages 1–12. International World Wide Web Conferences Steering Committee / ACM, 2013. 3, 4, 5

[ABF24] Martin Aumueller, Erik Bernhardsson, and Alec Faitfull. Ann benchmarks. https://ann-benchmarks.com, 2024. 2

[AI08] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008. 2, 13, 28

[AIK+25] Piyush Anand, Piotr Indyk, Ravishankar Krishnaswamy, Sepideh Mahabadi, Vikas C. Raykar, Kirankumar Shiragur, and Haike Xu. Graph-based algorithms for diverse similarity search. In *International Conference on Machine Learning (ICML)*, 2025. 2, 3, 4, 6, 8, 9, 17, 26

[CG98] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336, 1998. 2, 4

[DH99] Ding-Zhu Du and Frank Kwang-ming Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 1999. 4

[Dor43] Robert Dorfman. The detection of defective members of large populations. *The Annals of mathematical statistics*, 14(4):436–440, 1943. 4

[DR82] Arkadii Georgievich D'yachkov and Vladimir Vasil'evich Rykov. Bounds on the length of disjunctive codes. *Problemy Peredachi Informatsii*, 18(3):7–13, 1982. 23

[Gon85] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985. 27

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009. 6

[HJK+23] Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David P Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. *arXiv preprint arXiv:2310.05869*, 2023. 2

[IBC+25] Patrick Iff, Paul Bruegger, Marcin Chrapek, Maciej Besta, and Torsten Hoefler. Benchmarking filtered approximate nearest neighbor search algorithms on transformer-based embedding vectors, 2025. 8

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 604–613. ACM, 1998. 2, 12, 13

[IM18] Masajiro Iwasaki and Daisuke Miyazaki. Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data. arXiv preprint arXiv:1810.07355, 2018. 2, 13

[INR10] Piotr Indyk, Hung Q. Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1126–1142, 2010. 4

[IX23] Piotr Indyk and Haike Xu. Worst-case performance of popular approximate nearest neighbor search implementations: Guarantees and limitations. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2023. 2, 6, 7, 13, 21

[JDSP10] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3304–3311. IEEE, 2010. 2

[KL04] Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In J. Ian Munro, editor, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 798–807. SIAM, 2004. 2, 13

[KLJ+19] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019. 2

[Lia19] Search Liaison. Google announces site diversity change to search results. https://www.searchenginejournal.com/google-site-diversity-change/311557/, 2019. 2

[MY18] Yu A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2018. 2, 13

[Nov25] NovaSearch. Stella_em_400m_v5. https://huggingface.co/NovaSearch/stella_en_400M_v5, 2025. 8

[PKGZ24] Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. ACORN: performant and predicate-agnostic search over vector embeddings and structured data. *Proc. ACM Manag. Data*, 2(3):120, 2024. 8

[RKH+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021. 8

[SDI06] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision*. MIT Press, 2006. 2

[SDS+19] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. Diskann: Fast accurate billion-point nearest neighbor search on a single node. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 2, 9, 21

[Sim24] Harsha Simhadri. Openai embeddings of arxiv abstracts. https://github.com/harsha-simhadri/big-ann-benchmarks/commit/df1e53aa3cd9cd29c6a9daf24ce2e64271fa9ed1, 2024. 8

[SVB+21] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs, 2021. 8

[ZW24] Dun Zhang and Fulong Wang. Jasper and stella: distillation of SOTA embedding models. *CoRR*, abs/2412.19048, 2024. 8

A Algorithms for Standard Nearest Neighbor Search

In this section, we summarize existing approaches for the standard c-ANN problem, which are applied in our diverse algorithms.

Euclidean spaces. For Euclidean spaces, [IM98] introduced the prominent Locality Sensitive Hashing (LSH) technique, which constructs hash functions so that nearby points collide with higher probability than far points. By storing the dataset in multiple hash tables and probing only a small fraction of them, LSH achieves efficient query time in high dimensions. We next state the formal definition of LSH.

Definition A.1 (Locality-Sensitive Hashing)**.** *Given the Euclidean space* $(\mathcal{X}, \mathrm{dist})$*, a hash family* $\mathcal{H} = \{h : \mathcal{X} \to U\}$ *is a* (r_1, r_2, p_1, p_2)*-sensitive hashing if, for any points* $p, q \in \mathcal{X}$*, the following conditions hold:*

- *If* $\mathrm{dist}(p, q) \leq r_1$*, then* $\Pr_{\mathcal{H}}[h(p) = h(q)] \geq p_1$*,*

- *If* $\mathrm{dist}(p,q) \geq r_2$, *then* $\Pr_{\mathcal{H}}[h(p) = h(q)] \leq p_2$.

Intuitively, LSH hashes close pairs of points to the same bucket and far pairs of points to different buckets, so that we only need to search through a small amount of buckets that contains the query point. In the c-ANN algorithm [IM98], given a family of hash functions defined as Definition A.1, we construct L composite hash functions $g_1, g_2, \ldots, g_L$, each associated with a hash table $A_1, A_2, \ldots, A_L$. Each hash function $g_i = \langle h_{i,1}, \ldots, h_{i,K} \rangle$ is formed by selecting K functions $h_{i,j}$ uniformly at random from $\mathcal{H}$ and can be computed in $\mathcal{O}\left(dn^{o(1)}\right)$ time. For a data point $p \in \mathcal{X}$, it is stored in bucket $g_i(p)$ of table A_i for each $i \in \{1, \ldots, L\}$. To process a query point q, the algorithm retrieves points from the union of buckets $A_1(g_1(q)) \cup \cdots \cup A_L(g_L(q))$ and selects the approximated nearest neighbor, i.e., only the hash bucket corresponding to q in each table is examined. We obtain the results with the Hamming metric (see Theorem A.2) using $L = n^{\frac{1}{c}+o(1)}$ hash functions and the results in Euclidean spaces (see Theorem A.3) using $L = n^{\frac{1}{c^2}+o(1)}$ hash functions. The next statement shows the result given by [IM98] with the Hamming metric.

Theorem A.2 (c-ANN with the Hamming metric, [IM98])**.** *Given a set P of n points in the Hamming metric space $(\{0,1\}^d, \ell_1)$, there exists an algorithm that solves the c-ANN problem with probability 0.99, using $\mathcal{O}\left(dn^{\frac{1}{c}+o(1)}\right)$ query time and $\mathcal{O}\left(dn^{1+\frac{1}{c}+o(1)}\right)$ space and preprocessing.*

Based on the previous work, [AI08] designed a family of LSH for the Euclidean metric, achieving efficient algorithms in the Euclidean space as shown below.

Theorem A.3 (c-ANN in Euclidean spaces, [AI08])**.** *Given a set P of n points in the Euclidean space $(\mathbb{R}^d, \ell_2)$, there exists an algorithm that solves the c-ANN problem with probability 0.99, using $\mathcal{O}\left(dn^{\frac{1}{c^2}+o(1)}\right)$ query time and $\mathcal{O}\left(dn^{1+\frac{1}{c^2}+o(1)}\right)$ space and preprocessing.*

General metric spaces. For general metric spaces, LSH does not apply. Instead, [KL04] introduced the navigating nets, which recursively partition the metric space into a hierarchy of nets at geometrically decreasing scales, enabling efficient search by "navigating" through progressively finer scales. The next statement shows the bounds on the space and query time of the c-ANN algorithm by the doubling dimension.

Theorem A.4 (c-ANN with general metrics, [KL04])**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$ with doubling dimension $\mathbf{ddim}$ and aspect ratio Δ, there exists an algorithm that solves the c-ANN problem, using $\mathcal{O}\left(2^{\mathcal{O}(\mathbf{ddim})} \log \Delta + \left(\frac{1}{c-1}\right)^{\mathcal{O}(\mathbf{ddim})}\right)$ query time and $\mathcal{O}\left(n2^{\mathcal{O}(\mathbf{ddim})}\right)$ space.*

Recently, [IM18, MY18], and [IX23] studied graph-based algorithms, which achieve remarkable results in practice. These algorithms build a graph where vertices correspond to data points and edges connect nearby points. At query time, the algorithm performs a greedy search in the edge-list of an entry point based on the distance to the query. The following theorem shows the guarantee of the diskANN algorithm.

Theorem A.5 (c-ANN with general metrics, [IX23])**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$ with doubling dimension $\mathbf{ddim}$ and aspect ratio Δ, there exists an algorithm that solves the c-ANN problem for $c = \frac{\alpha+1}{\alpha-1} + \varepsilon$, using $\mathcal{O}\left((4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$ query time and $\mathcal{O}\left(n(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ space.*

B Missing Proofs in Section 2

In this section, we provide the missing proofs for our random color sampling scheme in Section 2. We restate our Algorithm 1 in Algorithm 3 for completeness.

The following lemma shows the correctness of our randomized sampling procedure.

Lemma B.1. *Algorithm 3 solves the k-diverse c-ANN problem (see Definition 1.2) with probability 0.99.*

Proof. Let $S \subset P$ be the set of k nearest points to q of distinct colors that satisfies $\mathrm{dist}(S, q) \leq r$. Let $\{\widehat{p_1} \ldots, \widehat{p_L}\}$ be the set of reported nearest neighbors in each bucket of Algorithm 3. It suffices to

Algorithm 3 Restatement of Algorithm 1

Require: Input dataset $P = \{p_1, \cdots, p_n\}$, each with a color $c \in \mathcal{C}$, an integer $k \leq |\mathcal{C}|$
Ensure: Data structure to solve the k-diverse nearest neighbor problem
1: Let $L \leftarrow \mathcal{O}(k \log k)$
2: **for** $i \in [L]$ **do**
3: Sample each color $c \in \mathcal{C}$ with probability $\frac{1}{k}$
4: Let $\mathcal{C}_i$ be the set of sampled colors, and let P_i be the set of points with colors in $\mathcal{C}_i$
5: Maintain a data structure that solves the c-ANN problem for P_i {Choose a subroutine from Theorem A.2, Theorem A.3, Theorem A.4, Theorem A.5 based on the dataset}
6: **end for**
7: **for** each query q **do**
8: Let $\widehat{p_i}$ be the reported c-ANN of bucket i for each $i \in [L]$
9: **Return:** k nearest points to q in $\{\widehat{p_1} \ldots, \widehat{p_L}\}$ of distinct colors
10: **end for**

show that there exists a set S' of k points of distinct colors in $\{\widehat{p_1} \ldots, \widehat{p_L}\}$ such that $\mathrm{dist}(S', q) \leq cr$, as the correctness of Algorithm 3 follows. We prove this statement by an inductive argument.

For the initial stage, note that for each bucket, we sample each color $c \in \mathcal{C}(S)$ with probability $\frac{1}{k}$. Here, we use $\mathcal{C}(E)$ to denote the set of colors obtained by the points in a set $E \subset P$. Since $|\mathcal{C}(S)| = k$, a bucket contains a color $c \in \mathcal{C}(S)$ with probability at least

$$1 - \left(1 - \frac{1}{k}\right)^k \geq 1 - \frac{1}{e} = \Omega(1).$$

We suppose that bucket i_1 contains a color in $\mathcal{C}(S)$, and let $T_1 = \{\widehat{p_{i_1}}\}$, where $\widehat{p_{i_1}}$ is the reported nearest neighbor in bucket i_1. Since bucket i_1 has color c, by our algorithm construction, it contains all points of color c, including the point in S. Therefore, the nearest neighbor in bucket i_1 is within distance r to q, and then if the c-ANN algorithm in bucket i_1 succeeds, which is with probability 0.99, we have $\mathrm{dist}(T_1, q) \leq cr$. Then let event $\mathcal{E}_1$ denote the event that a bucket contains a color $c \in \mathcal{C}(S)$ and the c-ANN algorithm succeeds, we have $\mathbf{Pr}[\mathcal{E}_1] = \Omega(1)$. Let X_1 be the number of buckets needed to ensure that such a bucket i_1 exists, we have $\mathbb{E}[X_1] = \mathcal{O}(1)$.

Next, we suppose that there is a set $T_{j-1} \subset \{\widehat{p_1} \ldots, \widehat{p_L}\}$ such that it contains $j - 1 < k$ points of distinct colors and satisfies $\mathrm{dist}(T_{j-1}, q) \leq cr$. Then a bucket contains no color in $\mathcal{C}(T_{j-1})$ with probability at least

$$\left(1 - \frac{1}{k}\right)^{j-1} \geq \left(1 - \frac{1}{k}\right)^k \geq \frac{1}{4} = \Omega(1).$$

In addition, since $|\mathcal{C}(S \backslash T_{j-1})| = k - j + 1$, a bucket contains a color in $\mathcal{C}(S \backslash T_{j-1})$ with probability at least ;

$$1 - \left(1 - \frac{1}{k}\right)^{k-j+1} = 1 - e^{-\frac{k-j+1}{k}} \geq \frac{k-j+1}{2k} = \Omega(1) \cdot \frac{k-j+1}{k}.$$

We suppose that bucket i_j contains a color in $\mathcal{C}(S \backslash T_{j-1})$ and no colors in T_{j-1}, and let $T_j = T_{j-1} \cup \{\widehat{p_{i_j}}\}$, where $\widehat{p_{i_j}}$ is the reported nearest neighbor in bucket i_j. Again, since bucket i_j selects the color c, by our algorithm construction, it contains all points of the color c, including the point in S. Therefore, the nearest neighbor in bucket i_j is within distance r to q, and then if the c-ANN algorithm in bucket i_j succeeds, which is with probability 0.99, we have $\mathrm{dist}(T_j, q) \leq cr$. Moreover, since bucket i_j contains no colors in $\mathcal{C}(T_j)$, $\widehat{p_{i_j}}$ has a new color, and so T_j has j distinct colors. Then let event $\mathcal{E}_j$ denote the event that a bucket contains a color $c \in \mathcal{C}(S \backslash T_j)$ and no colors in T_j, and the c-ANN algorithm succeeds, we have $\mathbf{Pr}[\mathcal{E}_j] = \Omega(\frac{k-j+1}{k})$. Let X_j be the number of buckets needed to ensure that such a bucket i_j exists, we have $\mathbb{E}[X_j] = \mathcal{O}\left(\frac{k}{k-j+1}\right)$.

Conditioned on the event that there exists bucket $i_1, \ldots, i_k$ that satisfy event $\mathcal{E}_1, \ldots, \mathcal{E}_k$ with $T_j := \{\widehat{p_{i_{j'}}}, j' \in [j]\}$ for each $j \in [k]$, then the above induction step shows that $T_k = \{\widehat{p_{i_j}}, j \in [k]\} \subset \{\widehat{p_1} \ldots, \widehat{p_L}\}$ contains k points of distinct colors and satisfies $\mathrm{dist}(T_k, q) \leq cr$, demonstrating the desired statement. Note that for each $j \in [k]$, we require $\mathcal{O}\left(\frac{k}{k-j+1}\right)$ buckets in expectation to ensure

the existence of bucket i_j, which in total gives

$$\sum_{j\in[k]} \mathcal{O}\left(\frac{k}{k-j+1}\right) = \mathcal{O}\left(k\log k\right)$$

buckets in expectation. Then, by Markov's inequality, with probability 0.99, we need $\mathcal{O}\left(k\log k\right)$ buckets in total to guarantee the correctness. Since we set $L = \mathcal{O}\left(k\log k\right)$ in Algorithm 3, its output is correct with probability 0.99. □

We next analyze the space and query time of our algorithm in Euclidean spaces.

Theorem B.2 (k-diverse c-ANN in Euclidean spaces)**.** *Given a set P of n points in the Euclidean space $(\mathbb{R}^d, \ell_2)$, there is an algorithm that solves the k-diverse c-ANN problem (see Definition 1.2) using $\mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)}\log k\cdot dn^{\frac{1}{c^2}+o(1)}\right)$ query time and $\mathcal{O}\left(\log k\cdot dn^{1+\frac{1}{c^2}+o(1)}\right)$ space with probability 0.99.*

Proof. It suffices to bound the query time and space of Algorithm 3 when we use the subroutine in Theorem A.3 for c-ANN. Now, we consider an arbitrary bucket. For each color $c\in\mathcal{C}$, let n_c denote the number of points in P of color c. Let X denote the number of points sampled in the bucket, we have $\mathbb{E}\left[X\right] = \frac{1}{k}\sum_{c\in\mathcal{C}} n_c = \frac{n}{k}$. By Theorem A.3, the c-ANN algorithm in each bucket uses $\mathcal{O}\left(dX^{\frac{1}{c^2}+o(1)}\right)$ query time and $\mathcal{O}\left(dX^{1+\frac{1}{c^2}+o(1)}\right)$ space. Since $n^{\frac{1}{c^2}+o(1)}$ is a concave function, by Jensen's inequality, the expected query time in the bucket is at most

$$\mathbb{E}\left[\mathcal{O}\left(dX^{\frac{1}{c^2}+o(1)}\right)\right] \le \mathcal{O}\left(d\mathbb{E}\left[X\right]^{\frac{1}{c^2}+o(1)}\right) \le \mathcal{O}\left(d\left(\frac{n}{k}\right)^{\frac{1}{c^2}+o(1)}\right).$$

Then, since there are $L = \mathcal{O}\left(k\log k\right)$ buckets, the total query time in expectation is at most

$$L\cdot\mathcal{O}\left(d\left(\frac{n}{k}\right)^{\frac{1}{c^2}+o(1)}\right) = \mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)}\log k\cdot dn^{\frac{1}{c^2}+o(1)}\right).$$

We note that there is an extra $\mathcal{O}\left(k\log k\right)$ factor in the query time due to the selection of the k points of distinct colors from the reported points from all buckets, but it is dominated by the $\mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)}dn^{\frac{1}{c^2}+o(1)}\log k\right)$ factor. In terms of space complexity, we write X as $\sum_{c\in\mathcal{C}} n_c X_c$, where X_c is 1 with probability $\frac{1}{k}$ and 0 otherwise. Note that $\mathbb{E}\left[d\left(\sum_{c\in\mathcal{C}} n_c X_c\right)^{\frac{1}{c^2}+o(1)}\right]$ is Schur-convex in the vector $[n_1\ldots,n_{|\mathcal{C}|}]$, and it reaches the maximum when there is a $n_c = n$ while all others are 0. Then the expected space in each bucket is at most

$$\frac{1}{k}\cdot\mathcal{O}\left(dn^{1+\frac{1}{c^2}+o(1)}\right).$$

Since there are $L = \mathcal{O}\left(k\log k\right)$ buckets, the total space in expectation is at most

$$L\cdot\frac{1}{k}\cdot\mathcal{O}\left(dn^{1+\frac{1}{c^2}+o(1)}\right) = \mathcal{O}\left(\log k\cdot dn^{1+\frac{1}{c^2}+o(1)}\right).$$

By Markov's inequality, we have the desired result. □

An analogous guarantee holds for the Hamming metric, as stated below.

Theorem B.3 (k-diverse c-ANN with the Hamming metric)**.** *Given a set P of n points in the Hamming metric space $(\{0,1\}^d, \ell_1)$, there is an algorithm that solves the k-diverse c-ANN problem (see Definition 1.2) using $\mathcal{O}\left(k^{1-\frac{1}{c}-o(1)}\log k\cdot dn^{\frac{1}{c}+o(1)}\right)$ query time and $\mathcal{O}\left(\log k\cdot dn^{1+\frac{1}{c}+o(1)}\right)$ space with probability 0.99.*

Proof. The analysis follows from the same calculation in Theorem B.2. □

In general metric spaces, combining our sampling approach with graph-based algorithms and navigating nets yields the following guarantees.

Theorem B.4 (k-diverse c-ANN with general metrics)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$ with doubling dimension $\mathbf{ddim}$ and aspect ratio Δ, there is an algorithm that solves the k-diverse c-ANN problem (see Definition 1.2) for $c = \frac{\alpha+1}{\alpha-1} + \varepsilon$ using $\mathcal{O}\left(k \log k \cdot (4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$ query time and $\mathcal{O}\left(\log k \cdot n(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ space with probability 0.99. There is another algorithm that solves the k-diverse c-ANN problem using $\mathcal{O}\left(k \log k \cdot \left(2^{\mathcal{O}(\mathbf{ddim})} \log \Delta + \left(\frac{1}{c-1}\right)^{\mathcal{O}(\mathbf{ddim})}\right)\right)$ query time and $\mathcal{O}\left(\log k \cdot n 2^{\mathcal{O}(\mathbf{ddim})}\right)$ space with probability 0.99.*

Proof. We consider an arbitrary bucket. For each color $c \in \mathcal{C}$, let n_c denote the number of points in P of color c. Let X denote the number of points sampled in the bucket, we have $\mathbb{E}[X] = \frac{1}{k}\sum_{c\in\mathcal{C}} n_c = \frac{n}{k}$. By Theorem A.5, the c-ANN algorithm in each bucket uses $\mathcal{O}\left((4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$ query time and $\mathcal{O}\left(X(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ space. Since there are $L = \mathcal{O}(k \log k)$ buckets, the total query time is $\mathcal{O}\left(k \log k \cdot (4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$. The expected space in each bucket is $\mathcal{O}\left(\mathbb{E}[X](4\alpha)^{\mathbf{ddim}} \log \Delta\right) = \mathcal{O}\left(\frac{n}{k}(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$, and so we in total need $\mathcal{O}\left(\log k \cdot n(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ space in expectation. By Markov's inequality, we have the desired result. The second result follows from a similar proof, where we instead apply the c-ANN algorithm in Theorem A.4 to each bucket. □

A relaxed definition. We consider a relax definition of the k-diverse problem called the (k, ε)-diverse problem, where we allow to have only $(1-\varepsilon)\cdot k$ points of distinct colors in the solution set of k points. This indicates that the recommendation system could return redundant or replicated results within distance cr to the query point, as long as there are $(1-\varepsilon)\cdot k$ diverse points. The formal definition is stated as follows:

Definition B.5 (The (k, ε)-diverse c-ANN problem)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$ and parameter $r > 0$, the goal is to construct a data structure that satisfies the following: given any query q, if the smallest ball around q which contains k points of distinct colors has radius at most r, it returns a set $\widehat{S}$ of $(1-\varepsilon)\cdot k$ points of distinct colors that satisfies $\max_{p\in\widehat{S}} \mathrm{dist}(p, q) \le cr$.*

We show that our color sampling approach still applies to the relaxed setting, and moreover, we replace the $\log k$ dependence by $\log \frac{1}{\varepsilon}$, achieving more efficient query time.

Lemma B.6. *Implementing Algorithm 3 with $L = \mathcal{O}\left(k \log \frac{1}{\varepsilon}\right)$ solves the (k, ε)-diverse c-ANN problem (see Definition B.5) with probability 0.99.*

Proof. Without loss of generality, we assume εk is an integer. Let $S \subset P$ be the set of k nearest points to q of distinct colors that satisfies $\mathrm{dist}(S, q) \le r$. Let $\{\widehat{p_1} \ldots, \widehat{p_L}\}$ be the set of reported nearest neighbors in each bucket of Algorithm 3. It suffices to show that there exists a set S' of $(1-\varepsilon)\cdot k$ points of distinct colors in $\{\widehat{p_1} \ldots, \widehat{p_L}\}$ such that $\mathrm{dist}(S', q) \le cr$.

Note that in the proof of Lemma B.1, we show that for any set $T_{j-1} \subset \{\widehat{p_1} \ldots, \widehat{p_L}\}$ such that it contains $j-1 < k$ points of distinct colors and satisfies $\mathrm{dist}(T_{j-1}, q) \le cr$, Algorithm 3 reports a nearest neighbor with a new color within radius cr using additional $\mathcal{O}\left(\frac{k}{k-j+1}\right)$ buckets in expectation. Therefore, to reach $(1-\varepsilon)\cdot k$ colors we only need

$$\sum_{j\in[(1-\varepsilon)\cdot k]} \mathcal{O}\left(\frac{k}{k-j+1}\right) = \mathcal{O}\left(k \log \frac{1}{\varepsilon}\right)$$

buckets in expectation. Thus, by Markov's inequality, using $L = \mathcal{O}\left(k \log \frac{1}{\varepsilon}\right)$ buckets in Algorithm 3 gives us the desired result with probability 0.99. □

The following theorem states the guarantees of our (k, ε)-diverse c-ANN algorithm in Euclidean spaces. The analysis follows directly from that of the k-diverse c-ANN algorithms (see e.g., Theorem B.2) and hence is omitted.

Theorem B.7 ((k,ε)-diverse c-ANN in Euclidean spaces)**.** *Given a set P of n points in the Euclidean space $(\mathbb{R}^d, \ell_2)$, there is an algorithm that solves the (k,ε)-diverse c-ANN problem (see Definition B.5) using $\mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)} \log \frac{1}{\varepsilon} \cdot dn^{\frac{1}{c^2}+o(1)}\right)$ query time and $\mathcal{O}\left(\log \frac{1}{\varepsilon} \cdot dn^{1+\frac{1}{c^2}+o(1)}\right)$ space with probability 0.99.*

The following theorem states the guarantees of the Hamming metric.

Theorem B.8 ((k,ε)-diverse c-ANN with the Hamming metric)**.** *Given a set P of n points in the Hamming metric space $(\{0,1\}^d, \ell_1)$, there is an algorithm that solves the (k,ε)-diverse c-ANN problem (see Definition B.5) using $\mathcal{O}\left(k^{1-\frac{1}{c}-o(1)} \log \frac{1}{\varepsilon} \cdot dn^{\frac{1}{c}+o(1)}\right)$ query time and $\mathcal{O}\left(\log \frac{1}{\varepsilon} \cdot dn^{1+\frac{1}{c}+o(1)}\right)$ space with probability 0.99.*

The following theorem states the guarantees of general metrics.

Theorem B.9 ((k,ε)-diverse c-ANN with general metrics)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$ with doubling dimension $\mathbf{ddim}$ and aspect ratio Δ, there is an algorithm that solves the (k,ε)-diverse c-ANN problem (see Definition B.5) using $\mathcal{O}\left(k \log \frac{1}{\varepsilon} \cdot (4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$ query time and $\mathcal{O}\left(\log \frac{1}{\varepsilon} \cdot n(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ space with probability 0.99. There is another algorithm that solves the (k,ε)-diverse c-ANN problem using $\mathcal{O}\left(k \log \frac{1}{\varepsilon} \cdot \left(2^{\mathcal{O}(\mathbf{ddim})} \log \Delta + \left(\frac{1}{c-1}\right)^{\mathcal{O}(\mathbf{ddim})}\right)\right)$ query time and $\mathcal{O}\left(\log \frac{1}{\varepsilon} \cdot n 2^{\mathcal{O}(\mathbf{ddim})}\right)$ space with probability 0.99.*

C Algorithms with Redundant Points

In this section, we consider another relaxed notion of the diverse problem studied by [AIK+25] called the (k,k')-diverse problem. In this setting, we need to report k nearest neighbors in total and allow each color in the reported points to have k' redundant points. We now formalize the definition of (k,k')-diverse.

Definition C.1 ((k,k')-diverse)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$, where each point is assigned a color. Let S be a set of points in X, we say a subset $S \subset P$ is (k,k')-diverse if $|S| = k$ and for any point $p \in S$, there are at most k' points in S that have the same color as P. Note that being $(k,1)$-diverse is equivalent to the standard notion of k-diverse.*

The following definition (k,k')-diverse c-ANN problem is then induced by the above notion.

Definition C.2 (The (k,k')-diverse c-ANN problem)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$, where each point is assigned a color, and parameter $r > 0$, the goal is to construct a data structure that satisfies the following: given any query q, if the smallest ball around q which contains a (k,k')-diverse subset of P has radius at most r, it returns a set $\widehat{S}$ that satisfies $\max_{p \in \widehat{S}} \mathrm{dist}(p,q) \le cr$ and is (k,k')-diverse.*

Recall that we sample each color with probability $\frac{1}{k}$ in our color sampling scheme in Algorithm 1. As we allow to report k' points for each color, we instead sample each color proportional to $\frac{k'}{k}$ and report k' nearest neighbors from each bucket. Similarly, we can show that the candidate sets from all buckets collectively recover a set of k nearest neighbors satisfying the (k,k')-diverse requirement, giving us a valid approximate solution to the (k,k')-diverse problem. We formally summarize this process in Algorithm 4. To begin with, we introduce a modified version of the LSH that returns m nearest neighbors of a query, so that we can apply in each bucket.

Theorem C.3. *Given a set P of n points in the Euclidean space $(\mathbb{R}^d, \ell_2)$, a query q, and an integer m, there exists an algorithm that returns a set $\widetilde{S}$ of m points. Let r be the distance between q and its nearest neighbor in $P \backslash \widetilde{S}$, then for each point $p \in \widetilde{S}$, $\mathrm{dist}(p,q) \le cr$ with probability 0.99. The algorithm uses $\mathcal{O}\left(mdn^{\frac{1}{c^2}+o(1)}\right)$ query time and $\mathcal{O}\left(dn^{1+\frac{1}{c^2}+o(1)}\right)$ space and preprocessing.*

Proof. Recall that in the c-ANN algorithm in Theorem C.3, we hash each point to several buckets using LSH. Given an LSH object, we report m nearest neighbors iteratively: Whenever we return a

point by the c-ANN algorithm, we delete the returned point from all the LSH buckets it hashes into and continue to find the next with the new LSH object. Since each point is hashed independently in LSH, in the t-the round of the iterative process, the LSH data structure that we maintain is equivalent to the LSH implemented in the dataset $P\backslash\{\widetilde{p_1}, \ldots, \widetilde{p_t}$, where p_τ is the point we obtained in round $\tau \in [t]$. Let p^* be the nearest neighbor of q in $P\backslash\widetilde{S}$, then p^* always exists in the LSH data structure m rounds. Conditioned on the success of each query of the LSH object, the point we returned in each round is always a c-ANN of p^* that has distance r to q, giving the guarantee in Theorem C.3. Finally, notice that we query m times the LSH data structure with at most n points and the time to delete the point in each round is dominated by the query time, the total time we need is then m times that of Theorem C.3. Note that the iterative procedure does not incur additional space, so the space follows directly from Theorem C.3. □

Algorithm 4 Diversity search allowing redundant points

Require: Input dataset $P = \{p_1, \cdots, p_n\}$, each with a color $c \in \mathcal{C}$, an integer $k \leq |\mathcal{C}|$
Ensure: Data structure to solve the (k, k')-diverse nearest neighbor problem

1: Let $L \leftarrow \mathcal{O}\left(\frac{k \log k}{k'}\right)$
2: **for** $i \in [L]$ **do**
3: Sample each color $c \in \mathcal{C}$ with probability $\frac{k'}{2k}$
4: Let $\mathcal{C}_i$ be the set of sampled colors, and let P_i be the set of points with colors in $\mathcal{C}_i$
5: Maintain a data structure that reports k' c-ANNs of each query for P_i {Choose a subroutine from Theorem A.2, Theorem A.3, Theorem A.4, Theorem A.5 based on the dataset}
6: **end for**
7: **for** each query q **do**
8: Let $\widetilde{S_i}$ be the set of the reported k c-ANNs of bucket i for each $i \in [L]$
9: $\mathcal{L} \leftarrow \emptyset$
10: **for** each point h in $\bigcup_i \widetilde{p_i}$ sorted by the distance to q **do**
11: $\mathcal{L} \leftarrow \mathcal{L} \cup \{h\}$ if $\mathcal{L} \cup \{h\}$ is $(|\mathcal{L}| + 1, k')$-diverse{There are less than k' points in $\mathcal{L}$ that have the same color as h}
12: **end for**
13: **Return:** the list $\mathcal{L}$
14: **end for**

We next show the correctness of Algorithm 4.

Lemma C.4. *Algorithm 4 solves the (k, k')-diverse c-ANN problem (see Definition C.2) with probability* 0.99.

Proof. Let $S \subset P$ be a (k, k')-diverse subset that satisfies $\mathrm{dist}(S, q) \leq r$. Let $\widetilde{S_1} \ldots, \widetilde{S_L}$ be the set of reported nearest neighbors in each bucket of Algorithm 4. It suffices to show that there exists a (k, k')-diverse subset in $\bigcup_i \widetilde{p_i}$ such that $\mathrm{dist}(S', q) \leq cr$, as the correctness of Algorithm 4 follows from the greedy search in the post-processing part. We prove this statement by a similar inductive argument as in Lemma B.1. We construct a candidate solution set S' gradually by adding $\frac{k'}{4}$ points at each step while preserving the following invariants: (1). $\mathrm{dist}(S, q) \leq cr$ and (2). S' is $(|S|, k')$-diverse. We maintain a list of colors $\mathcal{C}_{\mathrm{bad}}$ that contains all colors $c \notin \mathcal{C}(S)$ such that there are $\geq \frac{k'}{4}$ points in S' with color c. Here, we use $\mathcal{C}(E)$ to denote the set of colors obtained by the points in a set $E \subset P$. We also maintain a list of points S_{valid} that contains all points with colors in $\mathcal{S}(S')\backslash\left(\mathcal{C}(S) \cup \mathcal{C}_{\mathrm{bad}}\right)$, i.e., they are selected to the candidate solution set S' but they are neither in the optimal solution set S nor a "bad" color. We consider $k = \beta \cdot \frac{k'}{4} + \gamma$ for some integer β and $0 \leq \gamma \leq \frac{k'}{4}$.

For the initial stage, note that we sample each color $c \in \mathcal{C}(S)$ with probability $\frac{k'}{2k}$ independently, so the expected number of points sampled in the optimal solution S is $\frac{k'}{2}$. Therefore, a bucket contains at least $\frac{k'}{4}$ points in S with probability $\Omega(1)$ by the property of Bernoulli variables. We suppose that bucket i_1 is such a bucket and consider the following cases. In the first case, the $2k'$ c-ANNs of bucket i_1, denoted by $\widetilde{S_1}$, contain all the $\frac{k'}{4}$ points in S, then we add these points to S'. If this

case does not hold, then at least one point in the optimal solution S is not included in $\widetilde{S_1}$, and by Theorem C.3, all the $2k'$ reported points have distance $\leq cr$ to the query q conditioned on the success of the c-ANN algorithms. Next, in the second case, there is a color that has at least $\frac{k'}{2}$ points in the reported points, then we add the closest $\frac{k'}{4}$ points to S. Note that the c-ANN algorithm succeeds with probability 0.99, and so with probability $\Omega(1)$, the closest $\frac{k'}{4}$ points in the $\frac{k'}{2}$ points mentioned in the previous sentence are correct c-ANNs. If neither case one nor case two occurs, we add the closest $\frac{k'}{4}$ points in $\widetilde{S_1}$ to S. In all cases, the invariants are automatically satisfied. Then let event $\mathcal{E}_1$ denote the event that the invariants hold after the first step, we have $\mathbf{Pr}[\mathcal{E}_1] = \Omega(1)$. So, let X_1 be the number of independent buckets needed to ensure that such a bucket i_1 exists, we have $\mathbb{E}[X_1] = \mathcal{O}(1)$.

Next, we suppose that after the $(j-1)$-th step, the invariants hold. Consider the j-th step, at the beginning, there is $\frac{(j-1)k'}{4}$ points in S', and so there are at least $k - \frac{(j-1)k'}{4}$ points in S that have not been added to S'. For each bucket, the expected number of points sampled from $S \backslash S'$ is $\frac{k'}{2k} \cdot \left(k - \frac{(j-1)k'}{4}\right)$, and by the property of Bernoulli variables, we sample $\frac{k'}{4}$ points with $S \backslash S'$ with probability $\Omega\left(\frac{k-jk'}{k}\right)$. Moreover, consider the "bad" color list $\mathcal{C}_{\text{bad}}$, its size is less than $j-1$ since each color inside has at least $\frac{k'}{4}$ points in S' and there are $\frac{(j-1)k'}{4}$ points in S' at the current step. Then a bucket contains no color in $\mathcal{C}_{\text{bad}}$ with probability at least

$$\left(1 - \frac{k'}{2k}\right)^{j-1} \geq \left(1 - \frac{k'}{2k}\right)^{k} \geq \frac{1}{16} = \Omega(1).$$

In addition, consider the "valid" point list S_{valid}, we have $|S_{\text{valid}}| \leq k$ because $S_{\text{valid}} \subset S$. Then the expected number of points in S_{valid} sampled to each bucket is $\frac{k'}{2}$, and so by Markov's inequality, each bucket contains at most $\frac{3k'}{4}$ points in S_{valid} with probability $\Omega(1)$. Now notice that $(\mathcal{S} \backslash \mathcal{S}')$, $\mathcal{C}_{\text{bad}}$, $\mathcal{C}(S_{\text{valid}})$ are disjoint sets, so the three events we discussed above are independent, since we sample each color independently. Thus, let event $\mathcal{E}_j$ denote the event that a bucket contains at least $\frac{k'}{4}$ points from $S \backslash S'$, no colors from $\mathcal{C}_{\text{bad}}$, and at most $\frac{3k'}{4}$ points from S_{valid}, event $\mathcal{E}_j$ is satisfied with probability $\Omega\left(\frac{k-jk'}{k}\right)$. Let X_j be the number of buckets needed to ensure that such a bucket i_j exists, we have $\mathbb{E}[X_j] = \mathcal{O}\left(\frac{k}{k-jk'}\right)$.

We suppose that bucket i_j is a bucket that ensures event $\mathcal{E}_j$. Again, in the first case, the returned set of $2k'$ c-ANNs $\widetilde{S_j}$ by bucket i_j contains at least $\frac{k'}{4}$ points in $S \backslash S'$, then we add these points to S'. Note that the invariants are satisfied since these points are included in the optimal solution set S and have not been added to S' before. If the first case does not happen, conditioned on the correctness of the c-ANN algorithm, all points in $\widetilde{S_{j-1}}$ have distance $\leq cr$ to the query q. Next, in the second case, there is a color c in $\mathcal{C}(\widetilde{p_j})$ that has at least $\frac{3k'}{4}$ points. Since we do not have colors in $\mathcal{C}_{\text{bad}}$ in bucket i_j, there are at most $\frac{k'}{4}$ points in S' with color c, and thus, there are at least $\frac{k'}{2}$ points in $\widetilde{S_{j-1}} \backslash S'$ with color c. Therefore, we add the closest $\frac{k'}{4}$ points to S' without violating the invariants. Similarly, these $\frac{k'}{4}$ points are valid c-ANNs with probability $\Omega(1)$. Finally, if none of the above cases occur, then since $\mathcal{C}(\widetilde{p_j})$ has at most $\frac{k'}{4}$ points in $S \backslash S'$ and at most $\frac{3k'}{4}$ points in S', there are at least k' points in $\widetilde{S_j}$ that are not in both S and S'. Moreover, we can add the closest $\frac{k'}{4}$ points of these k' points to S' without violating the (k, k')-diverse requirement, because none of the attained colors in $\mathcal{C}(\widetilde{p_j})$ has at least $\frac{k'}{2}$ points in $\widetilde{S_j}$ and at least $\frac{k'}{4}$ in S' as there are no "bad" colors. Lastly, since the closest $\frac{k'}{4}$ points mentioned in the previous sentence are within distance cr to q with probability $\Omega(1)$, we do not violate the ANN requirement, and so the invariants hold.

Note that the above argument only holds for $k_{\text{rem}} = \Omega(k')$, where $k_{\text{rem}} := k - \frac{(j-1)k'}{4}$ is defined as the number of points we need in the candidate solution set, as we cannot obtain $\frac{k'}{4}$ in one bucket when k is smaller than $\Theta(k')$. However, for each bucket, the expected number of sampled points is $\frac{k'}{2k} \cdot k_{\text{rem}}$, and so we sample at least $\frac{k_{\text{rem}}}{2}$ points with probability $\Omega\left(\frac{k}{k'}\right)$. Then with the same analysis, let event $\mathcal{E}_j$ denote the event that a bucket contains at least $\frac{k_{\text{rem}}}{2}$ points from $S \backslash S'$, no colors from

$\mathcal{C}_{\text{bad}}$, and at most $\frac{3k'}{4}$ points from S_{valid}, event $\mathcal{E}_j$ is satisfied with probability $\Omega\left(\frac{k'}{k}\right)$. Let X_j be the number of buckets needed to ensure that such a bucket i_j exists, we have $\mathbb{E}[X_j] = \mathcal{O}\left(\frac{k}{k'}\right)$. Now we consider bucket i_j that ensures event $\mathcal{E}_j$. Under the first case, we can add the $\frac{k_{\text{rem}}}{2}$ points in $S \backslash S'$ to S'; and under the second case and the third case, we can add $\frac{k'}{4} = \Omega\left(\frac{k_{\text{rem}}}{2}\right)$ points to S'. Thus, k_{rem} decreases by at least half at each step, and so there are at most $\mathcal{O}(\log k')$ steps before we obtain a candidate set with size k, where we end the iteration.

Since in each step the two invariants hold, the candidate set suffices to give a valid (k, k')-diverse c-ANN solution. We now compute the number of buckets needed. For $k_{\text{rem}} > \Omega(k')$, we need $\mathbb{E}[X_j] = \mathcal{O}\left(\frac{k}{k - jk'}\right)$ buckets in step j to recover a valid bucket; and for $k_{\text{rem}} \leq \Omega(k')$, we need $\mathbb{E}[X_j] = \mathcal{O}\left(\frac{k}{k'}\right)$ buckets each step and there are at most $\mathcal{O}(\log k')$ steps. Therefore, we in total need

$$\sum_{j \in [\beta]} \mathcal{O}\left(\frac{k}{k - jk'}\right) + \mathcal{O}(\log k') \cdot \mathcal{O}\left(\frac{k}{k'}\right) = \mathcal{O}\left(\frac{k \log k}{k'}\right)$$

buckets in expectation, where $\beta = \mathcal{O}\left(\frac{k}{k'}\right)$. By Markov's inequality, with probability 0.99, we need $\mathcal{O}\left(\frac{k \log k}{k'}\right)$ buckets to guarantee the correctness. Since we set $L = \mathcal{O}\left(\frac{k \log k}{k'}\right)$ in Algorithm 4, its output is correct with probability 0.99. □

Combining the framework in Algorithm 4 with the LSH-based c-ANN algorithm gives us the following result:

Theorem C.5 (k-diverse c-ANN in Euclidean spaces)**.** *Given a set P of n points in the Euclidean space $(\mathbb{R}^d, \ell_2)$, there is an algorithm that solves the k-diverse c-ANN problem (see Definition 1.2) using $\mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)}(k')^{\frac{1}{c^2}+o(1)} \log k \cdot dn^{\frac{1}{c^2}+o(1)}\right)$ query time and $\mathcal{O}\left(\log k \cdot dn^{1+\frac{1}{c^2}+o(1)}\right)$ space with probability 0.99.*

Proof. The analysis is similar to Theorem C.5. It suffices to bound the query time and space of Algorithm 4 when we use the subroutine in Theorem C.3 for c-ANN. Now, we consider an arbitrary bucket. For each color $c \in \mathcal{C}$, let n_c denote the number of points in P of color c. Let X denote the number of points sampled in the bucket, we have $\mathbb{E}[X] = \frac{k'}{k} \sum_{c \in \mathcal{C}} n_c = \frac{nk'}{k}$. By Theorem C.3, the c-ANN algorithm in each bucket uses $\mathcal{O}\left(k' X^{\frac{1}{c^2}+o(1)}\right)$ query time and $\mathcal{O}\left(dX^{1+\frac{1}{c^2}+o(1)}\right)$ space since we return k' points in each bucket. Since $n^{\frac{1}{c^2}+o(1)}$ is a concave function, by Jensen's inequality, the expected query time in the bucket is at most

$$\mathbb{E}\left[\mathcal{O}\left(dk' X^{\frac{1}{c^2}+o(1)}\right)\right] \leq \mathcal{O}\left(dk' \mathbb{E}[X]^{\frac{1}{c^2}+o(1)}\right) \leq \mathcal{O}\left(dk' \left(\frac{nk'}{k}\right)^{\frac{1}{c^2}+o(1)}\right).$$

Then, since there are $L = \mathcal{O}\left(\frac{k \log k}{k'}\right)$ buckets, the total query time in expectation is at most

$$L \cdot \mathcal{O}\left(dk' \left(\frac{nk'}{k}\right)^{\frac{1}{c^2}+o(1)}\right) = \mathcal{O}\left(k^{1-\frac{1}{c^2}-o(1)}(k')^{\frac{1}{c^2}+o(1)} \log k \cdot dn^{\frac{1}{c^2}+o(1)}\right).$$

We note that there is an extra $\mathcal{O}(k \log k)$ factor in the query time due to the selection of the (k, k')-diverse subset from the reported points from all buckets, but it is dominated by the other factor. Moreover, by Theorem C.3 we know that the space to implement the c-ANN algorithm in each bucket does not incur additional space, so the space complexity direct follows from Theorem B.2. □

D Missing Proofs in Section 3.1

In this section, we provide the missing proofs for our expander graph-based color bucketing scheme in Section 3.1. We first formalize our algorithm in Algorithm 5.

The next statement shows the correctness of Algorithm 5.

Algorithm 5 Formal statement of the algorithm in Section 3.1

Require: Input dataset $P = \{p_1, \cdots, p_n\}$, each with a color $c \in \mathcal{C}$, an integer $k \le |\mathcal{C}| := N$, error ratio $\varepsilon > 0$

Ensure: Data structure to solve the k-diverse nearest neighbor problem

1: Let $\gamma = \frac{1}{\sqrt{\log k}}$, $D = \mathcal{O}\left(\left(\frac{\log k \log N}{\varepsilon}\right)^{1+\frac{1}{\gamma}}\right)$ and $M = D^2 k^{1+\gamma}$
2: Maintain a $(k, (1-\varepsilon)D)$-expander $G = (\mathcal{C}, B, E)$ with $E : [|\mathcal{C}|] \times [D] \to [M]$ {$B = \{b_1, \ldots, b_{|M|}\}$ denotes the set of buckets}
3: **for** each vertex $b_i \in [B]$ **do**
4: Let $\mathcal{C}_i$ contains the colors of all the right-neighbors of b_i, and let P_i be the set of points with colors in $\mathcal{C}_i$
5: Maintain a data structure that solves the c-ANN problem for P_i {Choose a subroutine from Theorem A.4 or Theorem A.5 based on the dataset}
6: **end for**
7: **for** each query q **do**
8: Let $\widehat{p_i}$ be the reported c-ANN of bucket i for each $i \in [L]$
9: **Return:** k nearest points to q in $\{\widehat{p_1} \ldots, \widehat{p_L}\}$ of distinct colors
10: **end for**

Lemma D.1. *Algorithm 5 solves the (k, ε)-diverse c-ANN problem (see Definition B.5).*

Proof. The analysis follows the framework in Lemma B.1. Let $S \subset P$ be the set of k nearest points to q of distinct colors that satisfies $\mathrm{dist}(S, q) \le r$. Let $\{\widehat{p_1} \ldots, \widehat{p_L}\}$ be the set of reported nearest neighbors in each bucket of Algorithm 5. It suffices to show that there exists a set S' of $(1-\varepsilon) \cdot k$ points of distinct colors in $\{\widehat{p_1} \ldots, \widehat{p_L}\}$ such that $\mathrm{dist}(S', q) \le cr$. Here, without loss of generality, we assume $(1-\varepsilon) \cdot k$ is an integer. We then prove this statement by an inductive argument.

We use $\mathcal{C}(E)$ to denote the set of colors obtained by the points in a set $E \subset P$. Let $G = (\mathcal{C}, B, E)$ be the $(k, (1-\varepsilon)D)$-expander obtained by Algorithm 5. For the initial stage, let bucket i_1 be a bucket that contains a color $c \in \mathcal{C}(S)$, and let $T_1 = \{\widehat{p_{i_1}}\}$, where $\widehat{p_{i_1}}$ is the reported nearest neighbor in bucket i_1. Since bucket i_1 has color c, by our algorithm construction, it contains all points of color c, including the point in S. Therefore, the nearest neighbor in bucket i_1 is within distance r to q, and then the c-ANN algorithm in bucket i_1 gives $\mathrm{dist}(T_1, q) \le cr$.

Next, we suppose that there is a set $T_{j-1} \subset \{\widehat{p_1} \ldots, \widehat{p_L}\}$ such that it contains $j - 1 < (1-\varepsilon) \cdot k$ points of distinct colors and satisfies $\mathrm{dist}(T_{j-1}, q) \le cr$. Let $N(\mathcal{C}(T_{j-1}) \cup S)$ be the neighborhood of $\mathcal{C}(T_{j-1}) \cup S$ in B, it satisfies $|N(\mathcal{C}(T_{j-1}) \cup S)| \ge (1-\varepsilon) \cdot kD$ by Theorem 3.2. Then since each color in $\mathcal{C}(T_{j-1})$ only has D neighbors and $|\mathcal{C}(T_{j-1})| < (1-\varepsilon) \cdot k$, there is at least one bucket in $N(\mathcal{C}(T_{j-1}) \cup S)$ that contains no colors in $\mathcal{C}(T_{j-1})$. In addition, since it is in the neighborhood of $\mathcal{C}(T_{j-1}) \cup S$, it must contain at least one color $c \in S \backslash \mathcal{C}(T_{j-1})$. Let i_j denote this bucket and let $T_j = T_{j-1} \cup \{\widehat{p_{i_j}}\}$, where $\widehat{p_{i_j}}$ is the reported nearest neighbor in bucket i_j. Again, since bucket i_j selects the color c, by our algorithm construction, it contains all points of the color c, including the point in S. Therefore, the nearest neighbor in bucket i_j is within distance r to q, and then the c-ANN algorithm in bucket i_j gives $\mathrm{dist}(T_j, q) \le cr$. Moreover, since bucket i_j contains no colors in $\mathcal{C}(T_j)$, $\widehat{p_{i_j}}$ has a new color, and so T_j has j distinct colors. We iterate this inductive argument until $j = (1-\varepsilon) \cdot k$, where we find at least $(1-\varepsilon) \cdot k$ distinct colors in the reported set. Thus, we prove the desired statement. □

We note that the diskANN algorithm studied by [SDS+19, IX23] is deterministic. Thus, combining with our expander-based color bucketing approach, we obtain deteministic algorithms for the (k, ε)-diverse c-ANN problem.

Theorem D.2 ((k, ε)-diverse c-ANN on general metric spaces)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$ with doubling dimension $\mathbf{ddim}$, aspect ratio Δ, and N colors, there is an algorithm that solves the (k, ε)-diverse c-ANN problem using*

$$\mathcal{O}\left(k^{1+o(1)}\left(\frac{\log k \log N}{\varepsilon}\right)^{\mathcal{O}\left(\sqrt{\log k}\right)} \cdot (4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$$

query time and

$$\mathcal{O}\left(\left(\frac{\log k \log N}{\varepsilon}\right)^{\mathcal{O}(\sqrt{\log k})} \cdot n(4\alpha)^{\mathbf{ddim}} \log \Delta\right).$$

space.

Proof. It suffices to compute the query time and space of Algorithm 5. We consider an arbitrary bucket b. For each color $c \in \mathcal{C}$, let n_c denote the number of points in P of color c. Let X denote the number of points in bucket b, we have $X = \sum_{c \in b} n_c$. By Theorem A.5, the c-ANN algorithm in each bucket uses $\mathcal{O}\left((4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$ query time and $\mathcal{O}\left(X_b(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ space. Recall that our expander $G = (\mathcal{C}, B, E)$ has degree $D = \mathcal{O}\left(\left(\frac{\log k \log N}{\varepsilon}\right)^{1+\frac{1}{\gamma}}\right)$ and number of buckets $M \le D^2 k^{1+\gamma}$ with $\gamma = \frac{1}{\sqrt{\log k}}$. Then the total query time is $\mathcal{O}\left(M(4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$, or equivalently

$$\mathcal{O}\left(k^{1+o(1)} \left(\frac{\log k \log N}{\varepsilon}\right)^{\mathcal{O}(\sqrt{\log k})} \cdot (4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right).$$

Moreover, the total space is $\mathcal{O}\left(\sum_{b \in B} X_b(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$. Then since each color c has degree at most D, we have

$$\sum_{b \in B} X_b = \sum_{b \in B} \sum_{c \in b} n_c = \sum_{c \in \mathcal{C}} D n_c = D \cdot n.$$

Thus, the total space is

$$\mathcal{O}\left(Dn(4\alpha)^{\mathbf{ddim}} \log \Delta\right) = \mathcal{O}\left(\left(\frac{\log k \log N}{\varepsilon}\right)^{\mathcal{O}(\sqrt{\log k})} \cdot n(4\alpha)^{\mathbf{ddim}} \log \Delta\right).$$

□

E Lower Bound for Dataset-Oblivious Bucketing

In this section, we provide the formal discussion and analysis for our lower bound in Section 3.2. Recall that in Section 3.1, we give a deterministic color bucketing method based on expander graphs, which provides a (k, ε)-diverse solution. Then a natural question is whether dataset-oblivious algorithms can give exact solutions. In this section, we provide a lower bound on deterministic dataset-oblivious color bucketing schemes. A bucketing scheme is dataset-oblivious if the subsets of colors contained in each bucket are fixed independently of the query and the spatial coordinates of the dataset. We demonstrate that such non-adaptive schemes require $\Omega\left(k^2\right)$ buckets.

Definition E.1 (Dataset-Oblivious Bucketing Scheme)**.** *A deterministic color bucketing scheme is dataset-oblivious if the family of color buckets $\mathcal{B} = \{B_1, \ldots, B_L\} \subseteq 2^{\mathcal{C}}$ is constructed based solely on the color universe $\mathcal{C}$, strictly prior to processing the spatial coordinates of the dataset P. For each $B_i \in \mathcal{B}$, a c-ANN algorithm $\mathcal{A}_i$ is instantiated over $P_i = \{p \in P \mid \chi(p) \in B_i\}$. Given a query q, the scheme queries all L buckets to obtain a candidate set $V = \{\hat{p}_1, \ldots, \hat{p}_L\}$. A deterministic function then selects k points with distinct colors in V.*

We next state the definition of d-disjunctness in the context of deterministic color bucketing, which requires each color to be "isolated" from all subsets of colors by at least one bucket.

Definition E.2. *A family of color subsets $\mathcal{B} = \{B_1, \ldots, B_L\} \subseteq 2^{\mathcal{C}}$ (represented by its incidence matrix) is defined as d-disjunct (or a $(1, d)$-cover-free family) if it satisfies the following condition:*

$$\forall x^* \in \mathcal{C}, \forall S^* \subset \mathcal{C} \setminus \{x^*\} \text{ such that } |S^*| = d, \exists B_i \in \mathcal{B} \text{ such that } x^* \in B_i \text{ and } B_i \cap S^* = \emptyset.$$

We next show the lower bound for dataset-oblivious color bucketing schemes.

Theorem E.3. *Assuming $k \ge 3$ and $k^2 \frac{\log N}{\log k} \le C \cdot N$ for some constant C, any deterministic dataset-oblivious bucketing scheme that correctly solves the k-diverse c-ANN problem for all valid datasets must evaluate $L = \Omega\left(k^2 \frac{\log N}{\log k}\right)$ buckets.*

Proof. Let $\mathcal{B} = \{B_1, \ldots, B_L\} \subseteq 2^{\mathcal{C}}$ be a fixed dataset-oblivious bucketing scheme. We assume for contradiction that $\mathcal{B}$ does not form a $(1, k-1)$-cover-free family (i.e., its incidence matrix is not $(k-1)$-disjunct). Then there exists at least a color $x^* \in \mathcal{C}$ and a subset of colors $S^* \subset \mathcal{C} \setminus \{x^*\}$ of size $k-1$, such that every bucket $B_i \in \mathcal{B}$ containing x^* also contains at least one color from S^*.

Because the scheme is dataset-oblivious, it must be universally correct for any spatial configuration of the dataset. Then consider dataset P and query q satisfying the following properties:

(1) For every color $\gamma \in \mathcal{C}$, there is exactly one point $p_\gamma \in P$ with color γ.

(2) For all $y \in S^*$, $\operatorname{dist}(q, p_y) = 1$.

(3) $\operatorname{dist}(q, p_{x^*}) = c + \varepsilon$, for an arbitrarily small $\varepsilon > 0$.

(4) Let $K^* = S^* \cup \{x^*\}$. For all $z \in \mathcal{C} \setminus K^*$, $\operatorname{dist}(q, p_z) = c(c+\varepsilon) + \varepsilon$.

By construction, the optimal k-diverse solution set for query q is K^*, which has radius $r^* = c + \varepsilon$. Since all points with colors outside of K^* have distance to q larger than $c(c+\varepsilon)$, to satisfy the definition of c-ANNs, the algorithm is required to return exactly the k points with colors in K^*, including p_{x^*}. Therefore, there is a bucket $B_i \in \mathcal{B}$ that contains x^*, however, by our initial non-disjunctness assumption, B_i must also contain some color $y \in S^*$. Thus, both p_{x^*} and p_y belong to P_i. The minimum distance from q to P_i is bounded above by $\operatorname{dist}(q, p_y) = 1$. Then any point $\widehat{p}$ returned by the c-ANN algorithm in bucket B_i satisfies

$$\operatorname{dist}(q, \widehat{p}) \leq c \cdot \operatorname{dist}(q, P_i) \leq c \cdot 1 = c.$$

Because $c < c + \varepsilon = \operatorname{dist}(q, p_{x^*})$, p_{x^*} is not returned by bucket B_i. Thus, the algorithm fails to retrieve x^* into the candidate set V, violating the k-diverse guarantee. Therefore, for a fixed dataset-oblivious scheme to be universally correct, the columns of the incidence matrix representing $\mathcal{B}$ must be $(k-1)$-disjunct.

By the classical extremal combinatorics theorem of [DR82], the minimum universe size L of a d-disjunct family of size N is bounded below by $\Omega\left(d^2 \frac{\log N}{\log d}\right)$. For $k \geq 3$, substituting $d = k-1$ gives us the $L = \Omega\left(k^2 \frac{\log N}{\log k}\right)$ lower bound. Note that we implicitly assume $k^2 \frac{\log N}{\log k} \leq C \cdot N$ for some constant C to avoid trivial color bucketing by placing one color in each bucket. □

F Adaptive Segment Tree Bucketing

In this section, we provide the formal discussion and analysis for our lower bound for dataset-oblivious bucketing in Section 3.3. To circumvent the $\Omega(k^2)$ barrier shown in Section E, we introduce an adaptive deterministic algorithm that selectively queries buckets based on the specific spatial configuration and color assignment of the dataset, using a segment tree over the color space. We achieve a $\log N$ overhead for space and a $k \log N$ overhead for the query time, where N is the total number of colors. This is slightly worst than the random bucketing scheme as $N \geq k$, but we achieve determinism.

F.1 Data Structure Construction

Assume without loss of generality that $N = 2^D$ (otherwise, pad $\mathcal{C}$ with dummy colors that correspond to empty point sets). We construct a perfectly balanced binary tree $\mathcal{T}$ of depth $D = \lceil \log_2 N \rceil$ as follows: (1) we deterministically assign one distinct color to each of the N leaves; (2) for each node u in $\mathcal{T}$, we include all colors associated with the leaves of the subtree rooted at u, and initialize a c-ANN data structure with the corresponding points. We formalize this procedure in Algorithm 6.

F.2 The Query Algorithm

At query time, the algorithm dynamically maintains a priority queue Q that stores a list of pairs $(\widehat{p_u}, u)$, meaning a candidate point $\widehat{p_u}$ and the node that reports it. We begin by adding the c-ANN reported by the root node. Next, in each iteration, we first add the minimum priority element $(\widehat{p_u}, u)$

Algorithm 6 Deterministic color bucketing construction

Require: Color set $\mathcal{C}$ with size N
Ensure: Binary tree $\mathcal{T}$ for solving the k-diverse nearest neighbor problem
1: Deterministically assign each leave of $\mathcal{T}$ a color such that the N leaves of $\mathcal{T}$ are in a fixed bijection with the colors in $\mathcal{C}$
2: **for** each node $u \in \mathcal{T}$ **do**
3: $C_u \leftarrow$ the subset of colors associated with the leaves of the subtree rooted at u
4: $P_u \leftarrow \{p \in P \mid c(p) \in C_u\}$, where $c(p)$ is the color of p
5: Initialize a standard c-ANN oracle $\mathcal{A}_u$ over P_u
6: **end for**

in Q, i.e., with minimum distance to q, to the result set. Then we query a group of descendent nodes of u to add new candidates to Q with the following requirements:

(1) Each node does not contain $c(p_u)$.

(2) The collection of the nodes cover all colors in u except for $c(p_u)$.

Thus, we avoid the color of p_u while ensuring that all other colors in u are considered. To achieve these requirements, we look at the downward path $v_0 = u, v_1, ..., v_m$ in $\mathcal{T}$ from u to the leaf corresponding to $c(\widehat{p_u})$. Note that each v_i contains $c(\widehat{p_u})$, so we instead query the other child w_{i-1} of v_{i-1} that is not v_i and add its reported point to Q, for each $i \in [m]$. The key observation is that $\{w_i \mid i \in [m-1]\}$ forms a mutually disjoint partition of the colors in u except for $c(\widehat{p_u})$. Then under this construction, we can show that in each iteration, the remaining nodes in Q forms a mutually disjoint partition of $\mathcal{C} \setminus \mathcal{C}_{\mathcal{R}}$, where $\mathcal{C}_{\mathcal{R}}$ is the set of colors used by the previously extracted points from Q. Therefore, in each iteration, we are able to add a new valid c-ANN with a distinct color to the result set, and so the algorithm terminates in k iterations, using $\mathcal{O}(k \log N)$ queries of the c-ANN algorithm in total. We formalize the above procedure in Algorithm 2.

Algorithm 7 Adaptive query scheme for k-diverse c-ANN

Require: Query $q \in \mathcal{X}$, target diversity $k \leq N$, segment tree $\mathcal{T}$
Ensure: Solution to k-diverse search
1: Initialize a priority queue Q, ordering pairs $(\widehat{p_u}, u)$ by $dist(q, \widehat{p_u})$. Query $\mathcal{A}_{\text{root}}$ to obtain a c-ANN $\widehat{p_{\text{root}}}$ and insert $(\widehat{p_{\text{root}}}, \text{root})$ into Q
2: Initialize result set $\mathcal{R} \leftarrow \emptyset$ and active colors $\mathcal{C}_{\mathcal{R}} \leftarrow \emptyset$
3: **while** $|\mathcal{R}| < k$ **do**
4: Extract the pair $(\widehat{p_u}, u)$ with minimum distance to q from Q
5: $\mathcal{R} \leftarrow \mathcal{R} \cup \{\widehat{p_u}\}, \mathcal{C}_{\mathcal{R}} \leftarrow \mathcal{C}_{\mathcal{R}} \cup \{c(\widehat{p_u})\}$
6: Let $v_0 = u, v_1, ..., v_m$ be the downward path in $\mathcal{T}$ from u to the leaf corresponding to $c(\widehat{p_u})$
7: **for** $i = 0$ to $m - 1$ **do**
8: Let w_i be the child of v_i that is not v_{i+1}.
9: **if** $P_{w_i} \neq \emptyset$ **then**
10: Query $\mathcal{A}_{w_i}$ to obtain a c-ANN $\widehat{p_{w_i}}$ and insert $(\widehat{p_{w_i}}, w_i)$ into Q
11: **end if**
12: **end for**
13: **end while**
14: **return** $\mathcal{R}$

F.3 Analysis of Algorithm 2

In this section, we analyze the correctness and complexity overheads of Algorithm 2.

Correctness. We first show the correctness of our algorithm. To begin with, we show that the algorithm maintains an invariant after each extraction from Q: the nodes that have not yet been extracted from Q form a mutually disjoint partition of the remaining color set. This property is later used to show that we can always find a new valid c-ANN in the remaining nodes in Q.

Lemma F.1. *At the beginning of each iteration (Line 3), let $\mathcal{U} = \{w, (\widehat{p_w}, w) \in Q\}$ be the set of nodes that have not yet been extracted from Q and let C_w denote the set of colors contained in node w, the collection of color sets $\{C_w \mid w \in \mathcal{F}\}$ forms a mutually disjoint partition of the remaining colors $\mathcal{C} \setminus \mathcal{C}_\mathcal{R}$, i.e., $\bigcup_{w\in\mathcal{F}} C_w = \mathcal{C} \setminus \mathcal{C}_\mathcal{R}$ and the union is mutually disjoint.*

Proof. We proceed by mathematical induction on the size of $\mathcal{R}$. Initially, $\mathcal{R} = \emptyset$ and $\mathcal{C}_\mathcal{R} = \emptyset$. The only queried node is the root, so $\mathcal{F} = \{\text{root}\}$. The set $C_{\text{root}} = \mathcal{C}$ trivially partitions $\mathcal{C} \setminus \emptyset$.

Next, assume the invariant holds at the beginning of an iteration. We extract $(\widehat{p_u}, u)$ from Q. Since $u \in Q \subseteq \mathcal{F}$, the inductive hypothesis guarantees $C_u \subseteq \mathcal{C} \setminus \mathcal{C}_\mathcal{R}$, and so we have $c(\widehat{p_u}) \in C_u$ and $c(\widehat{p_u}) \notin \mathcal{C}_\mathcal{R}$. Adding $c(\widehat{p_u})$ to $\mathcal{C}_\mathcal{R}$ updates the excluded colors to $\mathcal{C}_\mathcal{R} \cup \{c(\widehat{p_u})\}$. This increases $|\mathcal{R}|$ by 1 and thus we proceed the induction by one step after this iteration.

Next, removing u leaves a coverage gap of C_u, which is covered by the newly added points to Q. Let the path from u to the leaf of $c(\widehat{p_u})$ be $v_0 = u, v_1, ..., v_m$. Because $\mathcal{T}$ is a binary tree, for each $0 \le i \le m-1$, the children of v_i are v_{i+1} and w_i. Thus, $C_{v_i} = C_{v_{i+1}} \cup C_{w_i}$, and $C_{v_{i+1}} \cap C_{w_i} = \emptyset$. Telescoping this identity along the path yields:

$$C_u = C_{v_m} \cup \bigcup_{i=0}^{m-1} C_{w_i} = \{c(\widehat{p_u})\} \cup \bigcup_{i=0}^{m-1} C_{w_i}.$$

Consequently, $\bigcup_{i=0}^{m-1} C_{w_i} = C_u \setminus \{c(\widehat{p_u})\}$, and the sets C_{w_i} are mutually disjoint. Since we the nodes $\{w_0, ..., w_{m-1}\}$ to $\mathcal{F}$ after removing u, the updated set $\mathcal{F}$ accurately partitions $\mathcal{C} \setminus (\mathcal{C}_\mathcal{R} \cup \{c(\widehat{p_u})\})$. □

The next lemma shows that whenever the result set contains less than k points, the minimum priority element in Q is a valid c-ANN with a new color, and thus it can be added to the result set.

Lemma F.2. *Suppose that there exists an optimal set $S^* \subseteq P$ of k distinct colors such that $\max_{p\in S^*} dist(q, p) \le r^*$, then at the beginning of any iteration, if $|\mathcal{R}| < k$, the minimum priority element in Q has distance bounded by $c \cdot r^*$, and moreover, its color is not in $\mathcal{C}_\mathcal{R}$.*

Proof. Since $|S^*| = k$ and $|\mathcal{R}| < k$, there exists at least one optimal color $c^* \in \chi(S^*) \setminus \mathcal{C}_\mathcal{R}$. By Lemma F.1, the sets $\{C_w \mid w \in \mathcal{F}\}$ partition $\mathcal{C} \setminus \mathcal{C}_\mathcal{R}$. Thus, there exists exactly one node $w \in \mathcal{F}$ satisfying $c^* \in C_w$. Therefore, $\text{dist}(q, P_w) \le r^*$ since P_w contains at least one point in the optimal solution set S^*. Then when the oracle $\mathcal{A}_w$ was queried, it returns a point $\widehat{p_w} \in P_w$ satisfying:

$$\text{dist}(q, \widehat{p_w}) \le c \cdot \text{dist}(q, P_w) \le c \cdot r^*.$$

Therefore, Q contains at least one point that has distance bounded by $c \cdot r^*$, then the minimum priority element in Q is also bounded by $c \cdot r^*$. In addition, by Lemma F.1, the colors covered by $\{C_w \mid w \in \mathcal{F}\}$ do not appear in $\mathcal{C}_\mathcal{R}$, so the minimum priority element has a new color. □

As a direct corollary of the above lemma, the next statement shows that Algorithm 2 provides a valid solution.

Lemma F.3. *Algorithm 2 solves the k-diverse c-ANN problem in k iterations.*

Proof. By Lemma F.2, as long as $|\mathcal{R}| < k$, the minimum priority element in Q is a valid c-ANN with a new color. Therefore we can add it to $\mathcal{R}$ and proceed to the next iteration. The algorithm will deterministically execute exactly k iterations and return a k-diverse c-ANN set. □

Space usage and update time. We next analyze the space and time complexities of Algorithm 2. We first upper bound the space usage

Lemma F.4. *Let $S(P)$ denote the space complexity of the underlying c-ANN structure for a dataset P. Assuming $S(P)$ is super-additive, i.e., $S(P_1 \cup P_2) \ge S(P_1) + S(P_2)$ for disjoint P_1 and P_2, Algorithm 2 uses $\mathcal{O}(\log N \cdot S(P))$ space.*

Proof. For any fixed depth $l \in [0, \lceil \log_2 N \rceil]$ in $\mathcal{T}$, the sets C_u for nodes u at depth l form a disjoint partition of $\mathcal{C}$. Consequently, the local datasets P_u in buckets at depth l form a disjoint partition of the global dataset P, meaning $\bigcup_{u \text{ at depth } l} P_u = P$. Since S is super-additive, the total space utilized across depth l satisfies $\sum_u S(|P_u|) \le S(\sum_u |P_u|) = S(n)$. Summing across the $\lceil \log_2 N \rceil + 1$ depth levels, the overall space is bounded by $\mathcal{O}(\log N \cdot S(n))$. □

We next analyze the query time.

Lemma F.5. *Let $T(P)$ denote the query time of the underlying c-ANN structure for a dataset P, then Algorithm 2 uses $\mathcal{O}(k \log N \cdot T(P))$ total query time.*

Proof. Let M be the total number of c-ANN queries. By Lemma F.3, there is exactly k iterations in Algorithm 2, and in each of the k iterations, the traversal path length is bounded by the tree depth $\lceil \log_2 N \rceil$. Thus, the algorithm invokes $M \leq 1 + k\lceil \log_2 N \rceil = \mathcal{O}(k \log N)$ queries, giving a $\mathcal{O}(k \log N \cdot T(P))$ query time in total.

Next, we bound the time to maintain the priority queue Q. To achieve fast query time, we can implement Q as a Fibonacci heap, which supports insertions in amortized $\mathcal{O}(1)$ time and extracting the minimum element in amortized $\mathcal{O}(\log M)$ time. Since there are $M = \mathcal{O}(k \log N)$ c-ANN queries, the total time for insertion is $\mathcal{O}(k \log N)$. In addition, since the algorithm performs at most k extractions, the total extraction time is bounded by $\mathcal{O}(k \log M) = \mathcal{O}(k \log N)$. Therefore, the total time for maintaining Q is $\mathcal{O}(k \log N)$, which is dominated by the query time. □

We note that if $T(P)$ is a concave function upper bounded by $\mathcal{O}(n^\rho)$ for $n = |P|$ and $\rho < 1$, a finer analysis can be applied to show that the overhead is $k(\log N)^{1-\rho}$. However, this assumption is only matched by LSH-based algorithms, which is beyond the scope of this section since we want a deterministic algorithm. Next, we implement the graph-based algorithm in each bucket in the binary tree, achieving the following guarantees for deterministic algorithms.

Theorem F.6 (k-diverse c-ANN with general metrics)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$ with doubling dimension $\mathbf{ddim}$ and aspect ratio Δ, there is a deterministic algorithm that solves the k-diverse c-ANN problem (see Definition 1.2) for $c = \frac{\alpha+1}{\alpha-1} + \varepsilon$ using $\mathcal{O}\left(k \log N \cdot (4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$ query time and $\mathcal{O}\left(\log N \cdot n(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ space.*

Proof. The correctness directly follows from Lemma F.3. Then by Theorem A.5, the c-ANN algorithm in each bucket uses $\mathcal{O}\left((4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right)$ query time and $\mathcal{O}\left(n(4\alpha)^{\mathbf{ddim}} \log \Delta\right)$ space. Note that the space complexity is super-additive, then by Lemma F.4, the total space usage is

$$\mathcal{O}(\log N \cdot S(P)) = \mathcal{O}\left(\log N \cdot n(4\alpha)^{\mathbf{ddim}} \log \Delta\right).$$

In addition, by Lemma F.5, the total query time is

$$\mathcal{O}(k \log N \cdot T(P)) = \mathcal{O}\left(k \log N \cdot (4\alpha)^{\mathbf{ddim}} \log_\alpha \frac{\Delta}{(\alpha-1)\varepsilon} \log \Delta\right).$$

□

G Diversity Search with Euclidean Diversity Metric

In this section, we study the k-diverse c-ANN problem under a general notion of diversity, which measures the diversity of a set S by the minimum pairwise distance defined by a diversity metric ρ. We state the formal definition as follows.

Definition G.1. *Let ρ be a diversity metric, for a given set $S \subset \mathbb{R}^d$, its diversity is defined as the minimum pairwise distance between the points of the set, i.e., $\mathrm{div}(S) = \min_{p,p'} \rho(p, p')$.*

Note that the above definition aligns with Definition A.1 in [AIK+25], which calls a point set S to be C-diverse if $\mathrm{div}(S) \geq C$. Given this diversity measurement, we define the general k-diverse c-ANN problem by a bi-criteria guarantee: let S be the target set of nearest neighbor with optimal diversity, then the reported set should be within c times the distance between S and the query, in addition, it gives a constant-factor approximation to the diversity of S. We formalize this idea in the following definition.

Definition G.2 (The k-diverse c-ANN problem with diversity metric ρ)**.** *Given a set P of n points in a metric space $(\mathcal{X}, \mathrm{dist})$, a parameter $r > 0$, and a diverse metric ρ, the goal is to construct a data structure that satisfies the following: given any query q, if the smallest ball around q which contains a set S of k points with $\mathrm{div}(S) = C$ has radius at most r, it returns a set $\widehat{S}$ of k points with $\mathrm{div}(S) = \Omega(C)$ that satisfies $\max_{p \in \widehat{S}} \mathrm{dist}(p, q) \leq cr$.*

In this work, we restrict ρ to be the Euclidean metric. [AAI+13] introduced an LSH-based algorithm that solves this problem using roughly $k^2 + \log n$ overhead in query time. The algorithm maintains a list of diverse point in each LSH bucket, and then for each query, it processes in two stages: first, it selects $\mathcal{O}(k)$ candidates which contains the target set of diverse answers from the LSH buckets; then, it runs the GMM algorithm [Gon85] to select k points from the candidates that gives a constant-factor approximation to the optimal diversity. We observe that the k^2 bottle-neck is due to the slow processing time of the GMM algorithm, so it is our hope that if we can develop a faster algorithm to select k points from the $\mathcal{O}(k)$ candidates, we could potentially achieve better dependency in k in the update time. Next, in Section G.1, we introduce a thresholding approach to select k points in fast runtime, and in Section G.2, we combine it with the framework in [AAI+13] to achieve a query time of roughly $k^{1+\frac{1}{c^2}}$ overhead.

G.1 Selection of Diverse Points

In this section, we introduce a novel algorithm to select the k diverse points with fast runtime. Our algorithm relies heavily on the c-ANN algorithm using LSH introduced in Section A.

Our thresholding approach. Given a set P of m points and a subset $S \subset P$ with size $|S| = k$ and diversity $\operatorname{div}(S) = C$, our goal is to select k points from P with diversity $\Omega(C)$. This can be obtained by a simple greedy algorithm with thresholding: in each round, we prune a point $p \in P$ according to some fixed order, and then add p to the set of selected points T if the distance between p and T is larger than $\frac{C}{2}$. Note that because we assume that P contains a subset S of k points and diversity C, then by the triangle inequality, there always exists some points that are visited and can be added to T according to the threshold before T contains k points. Thus, the greedy procedure is well-defined and output a valid solution (see detailed proof in Lemma G.3). However, since it takes $\mathcal{O}(k)$ time to compute the distance between p and T in each round, this method requires $\mathcal{O}(mk) = \mathcal{O}\left(k^2\right)$ for $m = \mathcal{O}(k)$ pruning time and is prohibitively large for our goal. To achieve efficient time, instead, we keep an LSH data structure for the selected points in T, then in each round, we query to data structure to obtain a c-ANN of p, which gives a constant-factor approximation to the distance between p and T and suffices for our purpose. Then we only need roughly $\mathcal{O}\left(mk^{\frac{1}{c^2}}\right)$ total time to update the LSH data structure and query the c-ANN, which significantly improves on the polynomially dependency on k. We present the algorithm in Algorithm 8.

Algorithm 8 Select k diverse points

Require: Input dataset $P = \{p_1, \cdots, p_m\}$ with $m > k$, diversity metric ρ being the Euclidean metric, constant C
Ensure: Approximate k most diverse points in P
1: Let $T \leftarrow \emptyset$
2: Initiate an empty LSH data structure with $L = \mathcal{O}\left(k^{\frac{1}{c^2}+o(1)} \log m\right)$ hash functions {See the discussion below Definition A.1}
3: **for** $p \in P$ **do**
4: Let $\widehat{p}$ be a c-ANN of p in T
5: **if** $\rho(p, \widehat{p}) \geq \frac{C}{2}$ or $T = \emptyset$ **then**
6: $T \leftarrow T \cup \{p\}$
7: Hash p by the L hash functions in the LSH data structure
8: **end if**
9: **if** $|T| = k$ **then**
10: **Return:** T
11: **end if**
12: **end for**

The next statement demonstrates that Algorithm 8 gives a constant-factor approximation to the maximum diversity.

Lemma G.3. *Suppose that there is a set $S \subset P$ with $\operatorname{div}(S) = C$, then Algorithm 8 gives a set $T \subset P$ of k points with $\operatorname{div}(T) \geq \frac{C}{2c}$ with high probability.*

Proof. By the guarantee of c-ANN algorithm [AI08], its failure probability is $\frac{1}{\mathrm{poly}(m)}$ by setting $L = \mathcal{O}\left(k^{\frac{1}{c^2}+o(1)} \log m\right)$. Therefore, we condition on the success of all c-ANN algorithms, i.e., all $\widehat{p}$ on Line 4 of Algorithm 8 is a c-ANN of p in T, which happens with high probability by a union bound. Fix an arbitrary $i \in [k-1]$, let T_i be the set of the first i points selected in Algorithm 8, and let P_i be the set of points traversed by the algorithm before T_i is obtained. For a point $p \in P$, let $\rho(p, T_i) := \min_{p' \in T_i} \rho(p, p')$.

Now, consider an arbitrary point $u \in P$, without loss of generality, we suppose that $\rho(u, p) \leq \frac{C}{2}$ for some $p \in S$. Note that S satisfies $\mathrm{div}(S) = C$, i.e., any two points in S have diversity distance at least C, then by triangle inequality, u is at least $\frac{C}{2}$ from all points in $S \backslash \{p\}$. Thus, a $\frac{C}{2}$-radius ball centered at any arbitrary point in P can cover at most one point in S. Then, since $|T_i| = i$, there are at least $k - i$ points in $S \backslash T_i$ that satisfy $\rho(p, T_i) \geq \frac{C}{2}$. Let S' denote the set of such points, we assume that there is a point $s \in S'$ traversed by Algorithm 8 before T_i is obtained. Let T_j be the set of selected points when s is traversed by Algorithm 8 where $j < i$, let $\widehat{p}$ be the selected c-ANN of s in T_j on Line 4, then we have

$$\rho(p, \widehat{p}) \geq \rho(p, T_j) \geq \rho(p, T_i) \geq \frac{C}{2}.$$

Therefore, s satisfies the threshold on Line 5 of Algorithm 8 and is selected to T_{j+1} by our algorithm construction, which contradicts $s \in S \backslash T_i$. Then, for all $s \in S'$, we have $s \notin P_i$ and $\rho(s, T_i) \geq \frac{C}{2}$. Thus, for all $i \in [k-1]$, there exists a $p \notin P_i$ that satisfies $\rho(p, T_i) \geq \frac{C}{2}$, so our algorithm obtains a set T of k points and is well-defined.

In addition, let $p_i \in P \backslash T_i$ be the point added to T_i in round $i+1$ of Algorithm 8, then it satisfies the threshold on Line 5, i.e., $\rho(p_i, \widehat{p_i}) \geq \frac{C}{2}$, where $\widehat{p_i}$ is the selected c-ANN of p_i in T_i on Line 4. Then, by the guarantees of c-ANN, we have

$$\mathrm{div}(T_{i+1}) \geq \min\{\mathrm{div}(T_i), \rho(p_i, T)\} \geq \min\{\mathrm{div}(T_i), \frac{1}{c} \cdot \rho(p_i, \widehat{p_i})\} \geq \min\{\mathrm{div}(T_i), \frac{C}{2c}\}.$$

Thus, by induction we have $\mathrm{div}(T_k) \geq \frac{C}{2c}$ as desired. □

Now, we compute the total runtime of Algorithm 8.

Lemma G.4. *The runtime of Algorithm 8 is* $\mathcal{O}\left(dk^{\frac{1}{c^2}+o(1)} m^{1+o(1)} \log m\right)$.

Proof. Note that in Algorithm 8, there are $L = \mathcal{O}\left(k^{\frac{1}{c^2}+o(1)} \log m\right)$ hash functions, each using $\mathcal{O}\left(dm^{o(1)}\right)$ time to compute. Then in each round, we use $\mathcal{O}\left(dm^{o(1)}L\right)$ time to query the c-ANN and hash the selected point p to the L buckets. Since there are m rounds, we need $\mathcal{O}\left(dk^{\frac{1}{c^2}+o(1)} m^{1+o(1)} \log m\right)$ time in total to maintain the data structure. □

We next present the formal statement of our diverse point selection algorithm.

Theorem G.5. *Given a set P of m points in the Euclidean space $(\mathbb{R}^d, \mathrm{dist})$ such that there is a set $S \subset P$ with $\mathrm{div}(S) = C$ and aspect ratio Δ, there is an algorithm that gives a set $T \subset P$ of k points with $\mathrm{div}(T) \geq \frac{C}{4c}$ with high probability. The algorithm runs in $\mathcal{O}\left(dk^{\frac{1}{c^2}+o(1)} m^{1+o(1)} \log m \log \Delta\right)$ time.*

Proof. Note that we do not know the diversity C in advance, so we take geometrically increasing guesses of C, run Algorithm 8 for each guess, and select the most diverse set of solutions. Then, the correctness for each guess follows from Lemma G.3 and the runtime follows from Lemma G.4. Given an aspect ratio Δ of the input set, we only need to test $\log \Delta$ guesses, which gives our desired runtime. In addition, the high probability guarantee is by union bounding across all tests. □

G.2 Implication to Faster Algorithms

In this section, we combine our diverse point selection algorithm with the framework in [AAI+13] to achieve faster algorithms for the k-diverse c-ANN problem. The core of their approach is the construction of (ℓ, β)-robust coreset, which is formally defined as a subset $S' \subseteq S$ such that for any set of outliers O with $|O| \leq \ell$, the remaining set $S' \setminus O$ is a β-coreset of $S \setminus O$ for the k-center problem. During pre-processing, the algorithm stores a sequence of points in each LSH bucket that forms an (ℓ, β)-robust coreset. At query time, the algorithm identifies a candidate set by incrementally scanning the points in each relevant LSH bucket i until it finds a set T_i where the number of actual outliers (points with distance $> cr$ from the query q) does not exceed l. By eliminating these identified outliers, the algorithm constructs an intermediate candidate set $T \setminus O$, where $T = \bigcup_i T_i$, of nearest neighbors of size $\mathcal{O}\left(kn^{\frac{1}{c}} \log k\right)$ that preserves diversity. The remaining step is to run the GMM algorithm to select the k points that give a constant-factor approximation of the optimal diversity (see Section 5 of [AAI+13]). Note that the bottleneck is surprisingly the GMM algorithm, replacing it with our subroutine in Section G.1 would significantly accelerate the process. We state the formal guarantees of the above procedure as follows:

Theorem G.6 ([AAI+13])**.** *Given a set P of n points in the Euclidean space $(\mathbb{R}^d, \ell_2)$, there is an algorithm that selects a set S with $|S| = \mathcal{O}\left(kn^{\frac{1}{c^2}} \log k\right)$ that contains the optimal solution to the k-diverse c-ANN problem with the Euclidean diversity metric (see Definition G.2). The algorithm uses $\mathcal{O}\left(kn^{\frac{1}{c^2}} \log k \cdot d\right)$ query time and $\mathcal{O}\left(\log k \cdot n^{1+\frac{1}{c^2}}\right)$ space with probability* 0.99.

Proof. This theorem follows from the proof of Lemma 8 in [AAI+13]. We note that they consider the Hamming distance in their paper and so their update time has a linear dependency in $n^{\frac{1}{c}}$ (see Theorem A.2). Applying the LSH construction for the Euclidean distance, we replace the $n^{\frac{1}{c}}$ factor with the $n^{\frac{1}{c^2}}$ factor in the update time (see Theorem A.3). □

Combining the subroutines in Theorem G.5 and Theorem G.6, we achieve the following result for k-diverse c-ANN with Euclidean diversity.

Theorem G.7 (k-diverse c-ANN in Euclidean spaces)**.** *Given a set P of n points in the Euclidean space $(\mathbb{R}^d, \ell_2)$ with aspect ratio Δ, there is an algorithm that solves the k-diverse c-ANN problem with diversity metric ρ (see Definition G.2) using*

$$\mathcal{O}\left(k^{1+\frac{1}{c^2}+o(1)} (\log k)^{1+o(1)} \log n \log \Delta \cdot d n^{\frac{1}{c^2}+o(1)}\right)$$

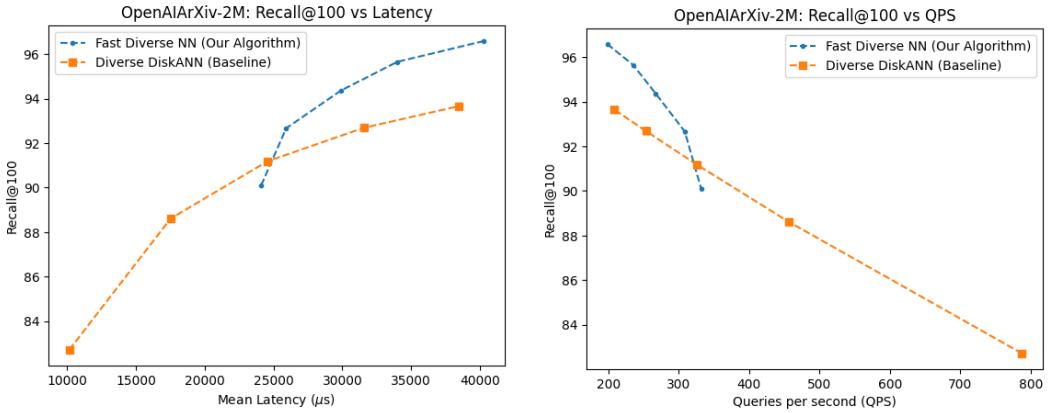
query time and $\mathcal{O}\left(\log k \cdot n^{1+\frac{1}{c^2}}\right)$ space with probability 0.9.

Proof. We first implement the algorithm in Theorem G.6 to select a set S with $|S| = \mathcal{O}\left(kn^{\frac{1}{c^2}} \log k\right)$ that contains the optimal solution. Given this set S, we apply our Algorithm 8 to find a set $T \subset S$ of k points with $\mathrm{div}(T) \geq \frac{C}{2c}$. By Theorem G.5, the algorithm runs in $\mathcal{O}\left(dk^{\frac{1}{c^2}+o(1)} m^{1+o(1)} \log m \log \Delta\right)$ time, where m is the size of S. Since $|S| = \mathcal{O}\left(kn^{\frac{1}{c^2}} \log k\right)$ in our application, the total runtime is $\mathcal{O}\left(k^{1+\frac{1}{c^2}+o(1)} (\log k)^{1+o(1)} \log n \log \Delta \cdot d n^{\frac{1}{c^2}+o(1)}\right)$. In addition, the space to store the data structure follows from Theorem G.6. □

H Additional Experimental Results

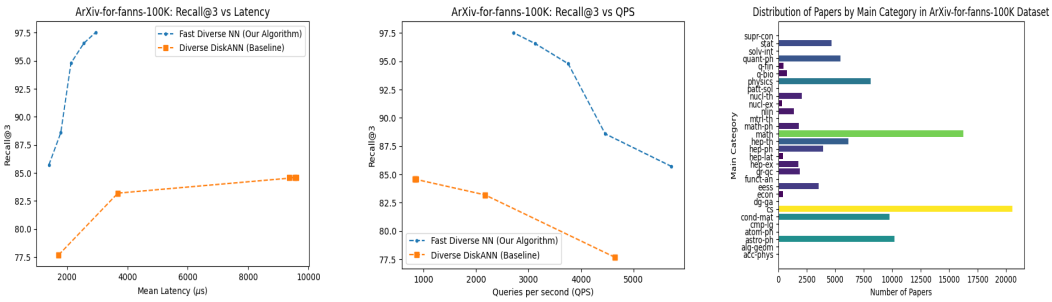
In this section, we provide additional empirical results.

Figure 3a and Figure 3b compare the recall-latency and recall-QPS tradeoff of our algorithm to the baseline's on the semi-synthetic OpenAI Embeddings dataset. Our algorithm's recall is generally in a higher range (90.12 - 96.59) than the baseline's recall (82.71 - 93.66). For latency greater than 25000 μs, our algorithm produces greater recall for fixed latency. As latency increases, the gap between the algorithms widens, highlighting that our algorithm provides increased benefits for applications that require high recall.



(a) Recall-latency tradeoff (b) Throughput tradeoff

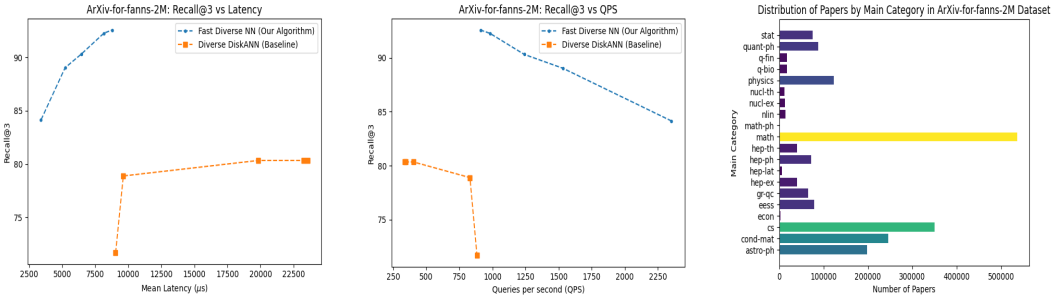
Fig. 3: Comparisons on semi-synthetic OpenAI Embeddings Dataset



(a) Recall-latency tradeoff (b) Throughput tradeoff (c) Paper category distribution

Fig. 4: Comparisons for real-world ArXiv-for-fanns 100K dataset

Figure 4 compares the recall-latency and recall-QPS tradeoff of our algorithm to that of the baseline on the real-word ArXiv-for-fanns dataset. Our results are consistent with the other datasets, as we have a higher recall range (85.70 - 97.52) than the baseline (77.68 - 84.58). For the same latency and QPS, our algorithm produces higher recall. As latency increases, the gap between the algorithms grows. Figure 4c shows the distribution of main categories or "colors" in the experiments involving the ArXiv-for-fanns dataset. This distribution is considerably less skewed than the synthetically generated one for the OpenAI Embeddings dataset, but our algorithm maintains its advantage.



(a) Recall-latency tradeoff (b) Throughput tradeoff (c) Paper category distribution

Fig. 5: Comparisons for real-world ArXiv-for-fanns-2M dataset

Figure 5 compares our algorithm to the baseline on the ArXiv-for-fanns dataset. While the original dataset contains approximately 2.8M total vectors, we sampled 2M vectors to feasibly obtain results. The "colors", i.e paper category, of the first 2M items is shown in Figure 5c. Figure 5a and Figure 5b

Dataset	Baseline	Ours
Arxiv-for-Fanns 100K	15 min.	1 min - 3 min
Arxiv-for-Fanns 2M	5 hr. 10 min.	48 min. - 2 hr. 8 min
OpenAI Embeddings 100k	2 min.	4 min. - 7 min.
OpenAI Embeddings 2M	1hr 6min.	1 hr. 35 min - 3 hr. 3 min.
LAION	36 min.	46 min. - 49 min.

Table 2: Total Index Build Times

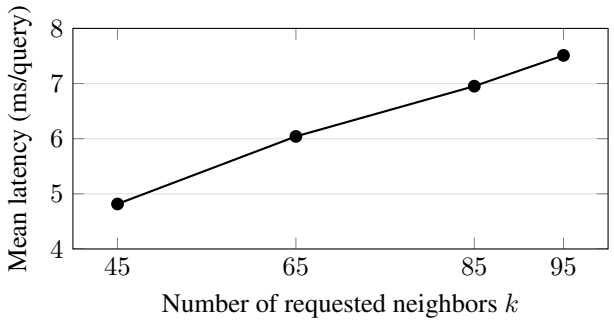


Fig. 6: Mean query latency of randomized color bucketing on OpenAIArXiv-100 as the number of requested neighbors increases.

compare the recall-latency and recall-QPS tradeoff of the two algorithms. Our algorithm has a higher recall range (84.14 - 92.56) than the baseline algorithm (70.71 - 80.17) and obtains higher recall for the same latency.

Table 2 compares the index build times of the baseline to that of our algorithm. Our algorithm offer significantly improved build times for the Arxiv-for-Fanns real-world dataset and competitive performance for the LAION dataset, indicating that it may be more beneficial for real-world color distributions. Conversely, the baseline algorithm is generally faster for the semi-synthetic OpenAI Embeddings dataset.

Query latency as a function of k. Figure 6 shows how mean query latency varies with the number k of requested neighbors on the OpenAIArXiv-100 dataset. Increasing k from 45 to 95 more than doubled the number of requested neighbors (111.1% increase), while mean latency rose from 4.816 to 7.510 milliseconds per query (55.9% increase).

Effect of the number of buckets. We varied the number of index buckets L_{buckets} from 600 to $1{,}000$, keeping the maximum graph degree $R = 16$ and the search list size $L_{\text{search}} = 16$ fixed. Table 3 shows the resulting tradeoff. With 800 buckets, the algorithm attained 94.36% recall at 267.6 queries per second. Increasing the count to $1{,}000$ gained a further 2.23 percentage points of recall, while mean latency rose from 29.89 to 40.30 milliseconds per query.

Buckets	Recall (%)	Build time (s)	Latency (ms/query)	Queries/s
600	90.12	5,727	24.11	331.8
700	92.67	7,015	25.92	308.6
800	94.36	8,154	29.89	267.6
900	95.65	9,454	33.98	235.4
1,000	96.59	11,000	40.30	198.5

Table 3: Effect of the bucket count with $R = 16$ and $L_{\text{search}} = 16$. Build time is the total time to construct all bucket indices.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

(1) **Claims**

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The paper provides algorithms for more efficient diverse nearest neighbors search and corresponding empirical results as claimed in the abstract and introduction.

Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.

- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

(2) **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: The paper presents several assumptions and discusses computational hurdles.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

(3) **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: For all technical statements, e.g. theorems and lemmas, the paper provides full assumptions and proofs, unless the technical statements are direct citations from previous work. We have reviewed the proofs and believe they are correct.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.

- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

(4) **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides technical details regarding the datasets and algorithms used, aiding reproduction.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

(5) **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The paper provides the code in the supplementary materials and details about the synthetic and real-world datasets.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

(6) **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The paper provides details about parameters used for the baseline and developed algorithms and various details about the datasets.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

(7) **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The paper evaluates the tradeoffs over many queries (10k+). As such, the aggregated statistics account for the variance between runs. The performance gains are also substantial, making the results clear without the need for explicit error bars.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

(8) **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The paper provides details about compute devices, including processors and memory.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

(9) **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics and believe we conform to all aspects.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

(10) **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The contribution is mainly theoretical, but the paper touches on applications such as searching for niche sellers in a market heavily skewed by dominant sellers. We do not foresee any negative societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

(11) **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We do not believe our results have risks to be misused.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

(12) **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The datasets and baselines used in the paper are properly cited.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

(13) **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The codes included in the supplementary materials include detailed instructions and documentation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

(14) **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper doesn't involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

(15) **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

(16) **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.