Efficient Point-to-Subspace ANNS in Manhattan and L_p Space by LSH Pruning

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

***Abstract*—Point-to-subspace approximate nearest neighbor search in L_p metric (L_p-P2S-ANNS) is a challenging research problem: Its only existing solution, called LDL1, is barely faster than the naïve linear scan, because its pruning (for promising ANNS candidates) metric is P2S distance in L_p, whose computation involves linear or convex programming that is computationally intensive. In this paper, we propose a novel scheme whose pruning metric is P2S distance in L_2 instead, which is computationally cheaper by four orders of magnitude, yet is almost as effective for pruning as LDL1's empirically. We also propose a new framework named LSH pruning, which subsumes and improves all existing dimension reduction schemes, and propose a performance model well-grounded in statistics theory. Our experiments show that these contributions in combination reduce the query time by a factor of 4.8 to 54.**

I. Introduction

Approximate nearest neighbor search (ANNS), a.k.a. similarity search, is a fundamental algorithmic problem arising in many areas of computer science, such as active machine learning [1], [2], computer vision [3], and information retrieval [4]. In these applications, data items are typically represented by vectors (points) in a massive high-dimensional (say in thousands) dataset $\mathcal{D}$; given a query vector $\vec{q}$, the ANNS problem is to find in $\mathcal{D}$ the nearest neighbors (NNs) of $\vec{q}$ according to a certain distance metric. In order to identify the NNs, an ANNS solution usually spends a considerable amount of time calculating query-to-data distances. Hence, for faster query processing, an ANNS solution often needs to reduce the number of such calculations (say using a *selective* index) and/or the cost of each calculation (say using distance approximation techniques such as dimension reduction [5]).

A. L_p-P2S-ANNS: Application and Challenges

In this paper, we study point-to-subspace (P2S)-ANNS, which extends traditional point-to-point (P2P)-ANNS problems as follows: The dataset now contains many high-dimensional linear subspaces, and the goal is to find the nearest subspaces to the query vector $\vec{q}$. P2S-ANNS arises in many research areas such as computer vision [3], [6] and neuroscience [7], wherein a class of similar data items (which can be either *raw features vector* such as images or *embedded feature vectors* such as transformer outputs in [8]), after being processed by dimension reduction techniques such as PCA (principal component analysis), are represented by a linear subspace. P2S-ANNS efficiently identifies similar classes (NN subspaces) to a given query point $\vec{q}$, which can speed up the classification [3] and clustering [9] of high-dimensional data.

P2S-ANNS is a harder computation problem than traditional P2P-ANNS for two reasons. The first reason is quite obvious: The P2S distances in the former problem is inherently more general than P2P distances in the latter (since points are degenerated subspaces) and hence are usually much harder to compute. For example, the time complexity of computing P2P distances (in L_p for $0 < p \leq 2$) is $O(d)$, whereas that of computing P2S distances is $O(d^2)$ in L_2 and $\Omega(d^{3.5})$ in L_p for $1 \leq p < 2$ [10]. The second reason is subtle but intuitive: It is even harder to build a selective index for a distance that is already hard to compute, since indexing is more challenging than distance computation; and as a result, P2S-ANNS indices [3], [6], [1], [11] are less selective than P2P-ANNS indices, which means query-to-data distance computations are needed for more data items.

Since P2S-ANNS is a hard problem, it has been studied mostly for the *easiest* case of P2S-ANNS *in* L_2 (called L_2-P2S-ANNS), and these solutions are restricted to this case. In this work, we study and solve the much harder case of P2S-ANNS in the general L_p metric (for $1 \leq p < 2$), or L_p-P2S-ANNS in short; such hardness is due to the high complexity of computing L_p-P2S distances, which, as mentioned above, is $\Omega(d^{3.5})$ whereas that of L_2-P2S is $O(d^2)$. In practice, the former computation takes roughly four orders of magnitude longer ($800\,\mathrm{ms}$ vs. about $30\,\mu\mathrm{s}$, as we show in Figure 1 for $d = 4096$). However, as a research problem, L_p-P2S-ANNS is at least as worthy as L_2-P2S-ANNS, because NN subspaces identified under the L_1-P2S distance lead to more robust image classification than those under L_2-P2S distance in the presence of errors such as occlusions, shadows, and peculiarities [10].

B. Existing Dimension Reduction Technique

So far, only a single solution has been proposed for L_p-P2S-ANNS. It is called LDL1 (low-dimensional L_1) [10] in the case of $p = 1$ and is based on the following dimension reduction technique (from one L_1 space to another) [12]. LDL1 first applies a *dimension-reducing* random Cauchy projection W (corresponding to a Cauchy random matrix with much fewer rows than columns) to each data item (subspace) $\mathcal{S}$ in $\mathcal{D}$ and the query vector $\vec{q}$; it then selects those $\mathcal{S}$ in $\mathcal{D}$ whose projected (data-to-query L_1-P2S) distance $\mathfrak{D}_1(W\vec{q}, W\mathcal{S})$ (the exact definition of $\mathfrak{D}_1$ will be given in (1)) are among the smallest, as the NN (nearest neighbor) candidates and prunes the rest of the dataset. Thanks to the dimension reduction, it is cheaper to compute $\mathfrak{D}_1(W\vec{q}, W\mathcal{S})$ than to compute $\mathfrak{D}_1(\vec{q}, \mathcal{S})$; and if this pruning (by $\mathfrak{D}_1(W\vec{q}, W\mathcal{S})$) is effective in reducing the number of NN candidates (to be "painstakingly" checked out by computing their $\mathfrak{D}_1(\vec{q}, \mathcal{S})$ with high time complexity), the pruning cost (of computing $\mathfrak{D}_1(W\vec{q}, W\mathcal{S})$ for every $\mathcal{S} \in \mathcal{D}$) can be well compensated for.

We now explain the rationale behind LDL1's pruning-by-$\mathfrak{D}_1(W\vec{q}, W\mathcal{S})$ technique. It was established in the celebrated E2LSH result [13] that the random projection by W is stochastically order-preserving in the following sense: For any two vectors $\vec{x}$ and $\vec{y}$, if $\vec{x}$ is closer to the query $\vec{q}$ than $\vec{y}$ in L_1 distance, then $W\vec{x}$ "tends to be" (by a notion that was made precise in [13]) closer to $W\vec{q}$ than $W\vec{y}$ in L_1. Authors of LDL1 conjectured that this order-preserving property in the P2P case could extend to the P2S case, so that $\mathfrak{D}_1(W\vec{q}, W\mathcal{S})$ can be used to *test* an $\mathcal{S} \in \mathcal{D}$ for whether $\mathcal{S}$ should be pruned; and showed that this extension worked empirically on some high-dimensional datasets, although the *power* of each test (to be made precise later in § IV-B) is much weaker than that in the P2P case.

While the rationale of LDL1 is sensible, its empirical efficacy is limited if the dataset (ambient) dimension is not so high (about 100 to 1000): It is barely faster than the linear scan (see Table II). The main reason is that its pruning process is not cost-efficient: On the *cost* side, reducing the dimension by say 50 times translates into far less than $50^{3.5}\times$ reduction in (P2S distance) computation time than its $\Omega(d^{3.5})$ asymptotic complexity would suggest (due likely to a "sticky" algorithm set-up overhead that does not shrink much with problem size). On the *efficiency* side, however, a $50\times$ reduction in dimension degrades the power of each test (explained in the previous paragraph) by so much that the pruning process is virtually useless for reducing the number of NN candidates.

C. New Feature and Framework

In this work, we propose a novel heuristic solution to L_p-P2S-ANNS that significantly outperforms LDL1 empirically. The proposed solution improves over LDL1 in two aspects. First, it also employs a Cauchy projection (matrix) W, but no dimensionality reduction is involved in the sense that W contains as many rows as columns. Second, whereas LDL1 uses the L_1-P2S distance, i.e., $\mathfrak{D}_1(W\vec{q}, W\mathcal{S})$, to test (prune) each subspace $\mathcal{S} \in \mathcal{D}$, our scheme instead uses the L_2-P2S distance, denoted by $\mathfrak{D}_2(W\vec{q}, W\mathcal{S})$. Using L_2-P2S distance as a novel feature for the pruning tests is the first major contribution of this work.

The pruning process in our solution is much more cost-efficient than that in LDL1. On the *cost* side, each test is much cheaper in our solution: Each L_2-P2S distance $\mathfrak{D}_2(W\vec{q}, W\mathcal{S})$ (instance) is roughly four orders of magnitude faster to compute (as mentioned earlier) than each L_1-P2S distance, despite that the former does not involve dimension reduction whereas the latter does. On the *efficiency* side, the power of each test in our solution is stronger (thanks to no dimension reduction being involved) than that in LDL1, as we will show in § IV-D. Therefore, our solution can afford to perform many more independent tests for each $\mathcal{S} \in \mathcal{D}$ than LDL1, which have much stronger power overall and significantly reduce the number of NN candidates to be "painstakingly" checked out.

The "scientific foundation" of our new feature (L_2-P2S) is just as solid as LDL1's, in the following sense. It has a similar rationale in the P2P case: It was first stated in [14] (and later verified by us) that $W\vec{x}$ "tends to be" (by the same notion as above) closer to $W\vec{q}$ than $W\vec{y}$ *also in L_2 distance*, if $\vec{x}$ is closer to the query $\vec{q}$ than $\vec{y}$ in L_1 distance. Also like in LDL1, our solution is to simply extend this approach to the P2S case, without worrying about a rigorous theoretical justification for the extension.

The second major contribution of this work is to propose a novel, principled framework (in § IV), called *LSH pruning*, that subsumes and improves nearly all existing dimension-reduction-based solutions for ANNS. Our LSH pruning framework is well-grounded in statistics theory and has a precise model for query performance (in terms of recall and query time). This model allows us to parameterize our aforementioned pruning process (e.g., the number of independent tests performed on each $\mathcal{S} \in \mathcal{D}$) for near-optimal query performance. With our new feature and this optimization framework, our proposed scheme achieves up to 54 times shorter query time than LDL1.

II. Background

In this section, we formulate the L_p-P2S-ANNS problem in § II-A and explain how the hardness (time complexity) of computing L_p-P2S distances varies with the value of p. Then in § II-B, we describe LDL1 [10], the dimension reduction scheme for L_1-P2S-ANNS.

A. P2S-ANNS in L_p Metric Space

In P2S-ANNS, we are given a dataset $\mathcal{D}$ consisting of linear subspaces $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ in a d-dimensional *ambient space* $\mathbb{R}^d$. For ease of presentation, we assume each subspace has the same rank τ, which is usually much less than d. Our solutions in this paper can be easily generalized to the case wherein each subspace is an affine subspace (not containing the origin), or wherein subspaces have different ranks.

Given a query point $\vec{q} \in \mathbb{R}^d$, the goal of P2S-ANNS is to find the nearest neighbor (NN) subspaces in $\mathcal{D}$ at a given recall (say 90%) in as little query time as possible. In the sequel (except evaluation), we adopt the so-called (r, cr) definition for NN, by assuming that the L_p-P2S distance from all NN subspaces (NNs in short) to the query is at most r, whereas from all other subspaces (non-NNs) is at least cr (for some constant $c > 1$).

The aforementioned L_p-P2S distance $\mathfrak{D}_p(\vec{q}, \mathcal{S})$ is defined as the minimum L_p distance between $\vec{q}$ and any point $\vec{z}$ on $\mathcal{S}$ as follows (S is a basis matrix of $\mathcal{S}$, i.e., $\mathcal{S} = \{S\vec{x} \mid \vec{x} \in \mathbb{R}^\tau\}$)

$$\mathfrak{D}_p(\vec{q}, \mathcal{S}) \triangleq \min_{\vec{z} \in \mathcal{S}} \|\vec{q} - \vec{z}\|_p = \min_{\vec{x} \in \mathbb{R}^\tau} \|\vec{q} - S\vec{x}\|_p, \qquad (1)$$

wherein the L_p norm $\|\vec{v}\|_p$ of any vector $\vec{v} = (v_1, v_2, \dots, v_d)$ is defined as $(|v_1|^p + |v_2|^p + \cdots + |v_d|^p)^{1/p}$.

As mentioned above, the hardness of calculating L_p-P2S distances is fundamentally different by the value of p:

- The L_2-P2S distance can be computed efficiently in $O(d^2)$ (or about $10\,\mu\text{s}$ in Figure 1) by the following closed-form *least squares* linear regression formula [15].
$$\mathfrak{D}_2(\vec{q}, \mathcal{S}) = \|\vec{q} - S(S^TS)^{-1}S^T\vec{q}\|_2. \qquad (2)$$
- The L_1-P2S distance is usually solved by linear programming (LP) with $\Omega(d^{3.5})$ time complexity [10]. By our measurements on `GUROBI` [16], a commercial LP solver, the calculation time starts from about $10\,\text{ms}$ and skyrockets to more than $800\,\text{ms}$ in high dimensions, as shown in Figure 1.
- The L_p-P2S distance ($1 < p < 2$) can be calculated by convex optimization solvers such as Adam [17], which are a few times faster than LP solvers on high dimension.
- The L_p-P2S distance ($0 < p < 1$) is NP-hard [18] and intractable to compute (so in this work, we only consider the case of $1 \le p < 2$).

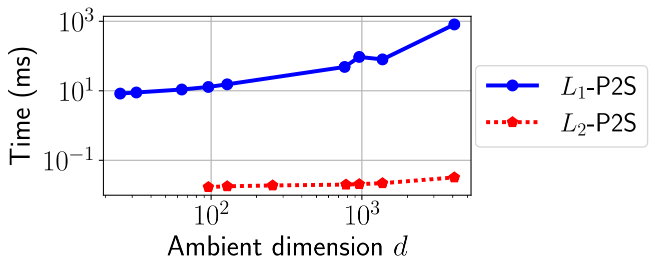

Fig. 1. Average computation times of each L_1-P2S and L_2-P2S instance on our workstation described in § V-A (both axes in log scale). L_2-P2S computation is faster than L_1-P2S computation by two to four orders of magnitude and is more scalable to high ambient dimensions.

B. Low Dimensional L_1 (LDL1)

The LDL1 [10] scheme can be described as follows. In the indexing phase, it repeats the following procedure for M times. In each repetition, it generates and fixes a $\mu \times d$ ($\mu \ll d$) dimension-reducing random Cauchy projection matrix W. Then, it projects every subspace $\mathcal{S} \in \mathcal{D}$ (in the d-dimensional *original space*) into $W\mathcal{S} \triangleq \{W\vec{z} \mid \vec{z} \in \mathcal{S}\}$ in a μ-dimensional *projected space*.

As explained earlier, given a query vector $\vec{q}$, LDL1 finds its NNs in two stages. In the *pruning* stage, for the Cauchy random projection W in each repetition, it projects $\vec{q}$ to $W\vec{q}$ and calculates the L_1-P2S distance in the projected space, namely $\mathfrak{D}_1(W\vec{q}, W\mathcal{S})$ (which we refer to as the *projected distance* in the sequel), for every subspace $\mathcal{S}$ in the dataset. The subspaces with the shortest projected distances in each repetition of W are selected as the *NN candidates*. In the *verification* stage, LDL1 calculates the L_1-P2S distance in the original high-dimensional space, namely $\mathfrak{D}_1(\vec{q}, \mathcal{S})$, (which we refer to as *original distance*) for every NN candidate and returns true NNs whose original distances are at most r.

III. New Feature and Algorithm

In this section, we describe our solution to L_1-P2S-ANNS, which prunes data items using a new feature (L_2-P2S distance) that is "dirt cheap" to compute. After describing our algorithm in § III-A, we intuitively explain the rationale behind our new feature in § III-B, show a worst-case scenario in § III-C, and discuss some extensions of our solution (e.g., from L_1 to general L_p) in § III-D.

A. Our New Algorithm for L_1-P2S-ANNS

As mentioned earlier, the most distinctive difference between our algorithm and LDL1 is the feature used in pruning tests: While LDL1 prunes data items (and selects NN candidates) by the L_1-P2S distances $\mathfrak{D}_1(\cdot,\cdot)$ in the projected space, our algorithm uses the L_2-P2S distances $\mathfrak{D}_2(\cdot,\cdot)$ for this purpose. In fact, our algorithm (shown in Algorithm 1) differs from LDL1 also in the following aspects.

```
1  Indexing Phase: Given database $\mathcal{D}$,
2      for $i = 1$ to $M$ do
3          Generate a $d \times d$ Cauchy random matrix $W_i$;
4          foreach subspace $\mathcal{S} \in \mathcal{D}$ do
5              Compute an orthogonal basis of $W_i\mathcal{S}$;

6  Query Phase: Given query $\vec{q}$,
7      NN candidates $\mathcal{C} = \emptyset$;
8      foreach subspace $\mathcal{S} \in \mathcal{D}$ do
9          for $i = 1$ to $M$ do
10             $R_i(\mathcal{S}) = \mathfrak{D}_2(W_i\vec{q}, W_i\mathcal{S})$;
11         $R^{(T)}(\mathcal{S}) =$ the $T^{th}$ smallest in
               $\{R_i(\mathcal{S}) \mid 1 \le i \le M\}$;
12         Add $\mathcal{S}$ to $\mathcal{C}$ if $R^{(T)}(\mathcal{S}) < \Theta$;
13     Verify each $\mathcal{S} \in \mathcal{C}$ and return NNs with
           $\mathfrak{D}_1(\vec{q}, \mathcal{S}) \le r$;
```
Algorithm 1: Proposed Algorithm.

In the indexing phase, our random Cauchy projections (by $d \times d$ matrices W_i's) do not involve dimension reductions as mentioned earlier, whereas LDL1's reduces the ambient dimension from d to $\mu \ll d$. The rationale of our design is that reducing the ambient dimension has little impact on the computing time of L_2-P2S distance, yet it negatively impacts the accuracy (the locality sensitivity below) of pruning. Furthermore, in our extension in § III-D, for a different purpose of reducing the memory usage, we can further apply a dimension-reducing L_2-to-L_2 mapping (called Johnson-Lindenstrauss transform, or JLT in short) to the projected (L_2) space, which is known [12] to be much more accurate (in preserving the power of each pruning test) than random Cauchy projections.

In addition, in order to speed up the computation of L_2-P2S projected distances in the query phase, we can preprocess the projected subspaces to orthogonalize their basis matrices B (say by QR factorization [19]) so that $B^TB = I$. In this way, we avoid computing the inverse term $(S^TS)^{-1}$ in (2) and reduce the time complexity of P2S distance calculation from $O(\tau^3 + \tau d)$ to $O(\tau d)$, where τ is the subspace rank, or by up to one order of magnitude in practice.

In the query phase, for each subspace $\mathcal{S}$, our algorithm summarizes its projected distances $R_i(\mathcal{S})$ in all repetitions $i = 1, 2, \ldots, M$ into the T^{th} order statistic $R^{(T)}(\mathcal{S})$ (the T^{th} smallest value in order) and use $R^{(T)}(\mathcal{S})$ in our decision rule for selecting NN candidates (Line 12). In contrast, in LDL1, each repetition separately selects some subspaces with the shortest projected distances, and the final NN candidate set is their union. As we will show in our LSH pruning framework (§ IV-C), our new decision rule subsumes LDL1's rule and improves its power of reducing the number of NN candidates.

B. Why Pruning by L_2-P2S Distances Works

As mentioned earlier, a clear advantage of our feature (L_2-P2S distance) over LDL1's is its "dirt cheap" computation cost: Compared to LDL1, using our feature reduces the total time of the pruning stage by two to four orders of magnitude. This reduction alone, however, is not sufficient for "proving" that our feature is more cost-efficient: Our feature (when used for the aforementioned tests) also needs to be accurate in discriminating NNs from non-NNs, in the sense few true NNs are missing from the NN candidate set (thus high recall), and few non-NNs are included (thus small NN candidate set size).

Our feature has good discriminating power, because the random Cauchy projection is a locality sensitive mapping in Definition 1 (similar to locality sensitive hashing [13]) from L_1-P2S distance to L_2-P2S distance in most cases, as is supported by empirical evidence to be presented shortly. As a result, NNs (the first bullet in Definition 1) are more likely to have shorter projected distances than non-NNs (second bullet), and the effectiveness can be measured by the gap between p_1 and p_2 (to be formalized in § IV-C).

Definition 1. *A random projection W is a* locality sensitive *mapping from distance metric $\mathfrak{D}$ to $\mathfrak{D}'$ if there exists a threshold Θ and probability values $p_1 > p_2$ such that for any query vector $\vec{q}$ and subspace $\mathcal{S}$, we have*

- *if $\mathfrak{D}(q, \mathcal{S}) \le r$, then $\Pr_W[\mathfrak{D}'(W\vec{q}, W\mathcal{S}) < \Theta] \ge p_1$,*
- *if $\mathfrak{D}(q, \mathcal{S}) > cr$, then $\Pr_W[\mathfrak{D}'(W\vec{q}, W\mathcal{S}) < \Theta] \le p_2$.*

As our empirical evidence, we make two observations from Figure 2 (the methodology for generating this figure is given in its caption). First, for every projected distance value Θ (x-coordinate), the corresponding CDF (cumulative distribution function) value (y-coordinate) is higher for NNs (subfigure (a)) than for non-NNs (subfigure (b)), so for every Θ value, Definition 1 is satisfied for some $p_1 > p_2$ for both our feature and LDL1's. Second, no matter whether measured in L_1-P2S or L_2-P2S, the empirical distributions of projected distances have almost the same shape, which implies that our feature and LDL1's are roughly equally accurate in discriminating NNs from non-NNs.

However, we observe a worst-case pair of NN and non-NN for which neither our feature nor LDL1's works. This, unfortunately, implies that the effectiveness of our new feature

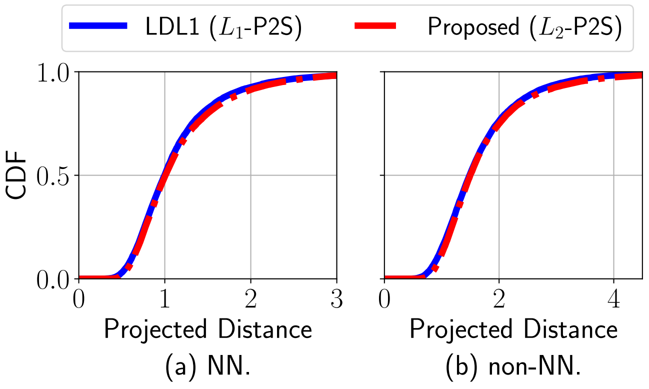

Fig. 2. We use a sampled query from the SIFT dataset (in Table I), and select a non-NN that is $c = 1.5$ times as far as the NN in the original L_1-P2S distance. For both our algorithm and LDL1 ($\mu = 25$), we measure the CDF of projected distances under 10,000 random Cauchy projections. For better comparison, we normalize the projection distances in both schemes (and in Figure 3 also) so that their medians are both 1 for the NN.

can only be established by empirical evaluations rather than theorems. It will be clear, however, that this worst case is so counter-intuitive and contrived that no real-world dataset (that is not generated by an adversarial party) can be anywhere close to such a worst-case scenario.

C. Worst-Case Scenario

In this scenario, we let the ambient dimension $d = 128$, and the query vector be $\vec{q} = (1, \underbrace{0.05, \ldots, 0.05}_{2^{nd} \text{ to } 10^{th}}, \underbrace{0, \ldots, 0}_{11^{th} \text{ to } 128^{th}})^T$. We construct two subspaces with rank $\tau = 9$: one NN and one non-NN. The NN subspace is spanned by $\vec{e}_2, \ldots, \vec{e}_{10}$ (each $\vec{e}_i$, $i = 1, \ldots, 10$, is a standard basis vector with only the i^{th} coordinate being 1 and remaining coordinates being 0), and the non-NN subspace is spanned by $\vec{e}_2 - \vec{e}_1, \ldots, \vec{e}_{10} - \vec{e}_1$. It is not hard to verify that the original distance to the NN is $r = 1$, and to the non-NN is $cr = 1.45$. Figure 3 shows that p_2 (red dashed line) is always greater than p_1 (blue solid line) regardless of the threshold Θ (x-coordinate). Hence, Definition 1 is violated at least by this pair of NN and non-NN.

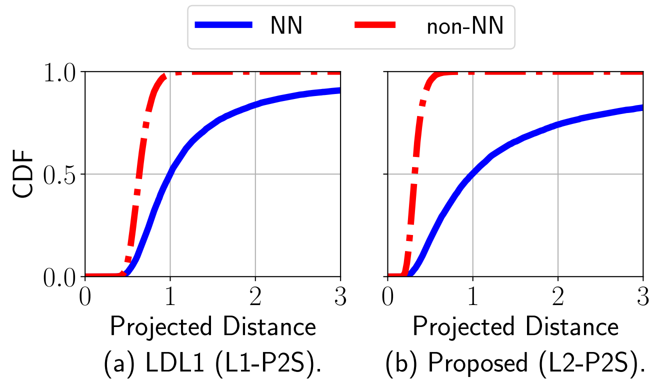

Fig. 3. A worst-case scenario in which Definition 1 is violated for both LDL1 and our algorithm.

D. Extensions

1) JLT Dimension Reduction: Our index stores the basis matrices of all projected subspaces computed in the indexing phase, whose size is $O(nd\tau M)$, where n is the dataset size. In practice, however, this index size can be very large for high dimensional datasets, since d is large, and we use many repetitions M for the pruning (to achieve a small NN candidate set size) as mentioned earlier. To reduce the index size, we can reduce the projection dimension from d to μ by the aforementioned JLT technique [5] as follows. For each repetition of the random $(d \times d)$ Cauchy projection (matrix) W_i, we generate (and fix) a $\mu \times d$ Gaussian random matrix G_i that further projects the d-dimensional projected (by Cauchy) space into a μ-dimensional space. In other words, we replace each W_i in Algorithm 1 with $G_i W_i$. According to our experiments, the JLT dimension reduction results in a 90% smaller index size while impacting the effectiveness of pruning only slightly.

2) General L_p-P2S Distance: Our algorithm can solve the L_p-P2S-ANNS problem for any $1 < p < 2$ in general with just one modification: Replace each Cauchy (1-stable) random projection W_i with a p-stable one as in [13] (wherein the elements are i.i.d. p-stable random variables). The resulting p-stable projection has a similar locality sensitive property in L_p as the Cauchy projection in L_1, which allows it to effectively prune subspaces for promising NN candidates.

IV. LSH Pruning Framework

In this section, we describe our aforementioned second contribution: an LSH pruning framework and performance model. After describing our model in § IV-A, we elaborate the hypothesis testing theory behind our framework in § IV-B, which reveals a fundamental trade-off between the cost and the effectiveness of pruning. Based on this theory, we get an effective decision rule for selecting NN candidates in § IV-C, and show that our new feature is empirically at least as effective as LDL1 in § IV-D using the discriminating power metric derived from the theory above.

A. Performance Model for LSH Pruning

In this section, we propose the following general LSH (locality sensitive hashing) pruning framework that subsumes many ANNS schemes such as LDL1, our proposed algorithm in § III, and Bayesian LSH [20]. An LSH pruning process consists of the following two stages.

Pruning: In this stage, the feature distance $\mathfrak{D}'(h(\vec{q}), h(\mathcal{S}))$ between hash values is computed for each data item $\mathcal{S}$ in the dataset $\mathcal{D}$, possibly repetitively using independent LSH functions $h(\cdot)$'s; and a specific *decision rule* (to be given shortly) is used to select NN candidates $\mathcal{C}$ with short feature distances.

Verification: In this stage, the original distance $\mathfrak{D}(q, \mathcal{S})$ is computed for each $\mathcal{S} \in \mathcal{C}$, and verified NNs with $\mathfrak{D}(q, \mathcal{S}) < r$ are returned.

For example, in our solution the LSH function is the random Cauchy projection $h(\mathcal{S}) = W\mathcal{S}$ (and $h(\vec{q}) = W\vec{q}$), and the feature distance is L_2-P2S. As mentioned earlier, for an LSH pruning scheme to be cost-efficient, it must use a cost-efficient feature distance $\mathfrak{D}'$ that has the following two properties.

1) The computation cost of $\mathfrak{D}'$ itself should be low.
2) $\mathfrak{D}'$ can effectively *discriminate* NNs from non-NNs. In other words, it only selects a small set of NN candidates that need to be verified at a high computation cost.

As we will show shortly, there is a fundamental tradeoff between the cost and the effectiveness, with the former corresponding to the time complexity of the pruning stage and the latter corresponding to that of the verification stage.

As the first step of modeling the performance of LSH pruning, we identify two metrics for the cost and the effectiveness respectively, and how they determine the overall query time, as follows. The *overall speedup* ratio η of an LSH pruning algorithm (Alg) over the linear scan (LS) is equal to

$$\eta \triangleq \frac{T_{LS}}{T_{Alg}} = \frac{T_{LS}}{T_{prune} + T_{verify}} = \frac{1}{M/\gamma + \beta(M)}. \quad (3)$$

This is because the total pruning time $T_{prune} = M/\gamma \cdot T_{LS}$, where M, as a tunable parameter, is the number of repetitions (LSH functions) used, and γ, called the *LSH speedup*, is the ratio between the calculation time of the original distance $\mathfrak{D}$ and the feature distance $\mathfrak{D}'$. Also, the total verification time $T_{verify} = \beta(M) \cdot T_{LS}$, where $\beta(M) \triangleq |\mathcal{C}|/n$, called the *selectivity*, is the fraction of the dataset (say of size n) that are selected into the NN candidate set $\mathcal{C}$.

In LSH pruning schemes, usually the LSH speedup γ is determined by the feature distance metric $\mathfrak{D}'$, and hence is a fixed value (when parameters are given). As a result, the aforementioned trade-off between cost and effectiveness, or pruning and verification times, is controlled only by M: Using a larger M leads to a better discriminating power and a lower selectivity β, but inflates the pruning time proportionally.

Finally, we note that our performance model is different from all existing ones for hash-table-based LSH schemes for ANNS such as QALSH [21]. In these schemes, NN candidates are selected by collisions in hash tables, which does not involve the calculation of feature distances (and their costs). Hence, in these models, the overall query time is almost equivalent to the verification time, which is very different from our trade-off here.

B. Pruning is Hypothesis Test

Now we answer a key question in our performance model:

How is the selectivity β determined by M?

The answer lies in the theory of hypothesis testing, because as we will show shortly, the procedure in the pruning stage is essentially performing the following test between two hypotheses on each data item $\mathcal{S}$ (which we have already mentioned, without providing a precise definition, in the introduction):

Null hypothesis: $\mathcal{S}$ is an NN of $\vec{q}$ with the original distance $\mathfrak{D}(\vec{q}, \mathcal{S}) < r$, so it should be selected as an NN candidate.

Alternative hypothesis: $\mathcal{S}$ is a non-NN with $\mathfrak{D}(\vec{q}, \mathcal{S}) > cr$, so it should be pruned.

This test, on whether $\mathfrak{D}(\vec{q}, \mathcal{S})$ is greater than cr or less than r, belongs to the *one-sided* subtype of hypothesis tests [22]. Hence, we focus on this subtype in the sequel.

In the context of hypothesis testing, if a data item is selected as an NN candidate, we say the null hypothesis is *accepted* by the pruning scheme; otherwise, the null hypothesis is *rejected*, and the alternative hypothesis is accepted instead. Such statistical inference, however, may be erroneous sometimes. A *type-I error* occurs when the null hypothesis is erroneously rejected (on a true NN), and a *type-II error* occurs when it is erroneously accepted (on a non-NN). In the statistics literature, usually the type-I error rate is denoted by α ($1-\alpha$ is called the *confidence level*), and the type-II error rate is denoted by β ($1-\beta$ is called the *power* of the test).

Relating to our question above, the selectivity β of pruning is almost equal to the type-II error rate of the corresponding hypothesis test. (Hence, we can denote both values by the same letter β.) This is because the number of non-NNs in the dataset is much larger than that of NNs, so the number of NN candidates is roughly the number of non-NNs that are erroneously accepted. Therefore, the aforementioned function $\beta(M)$ is in fact given by the result of the following optimization problem: Given M instances of feature distances, design a hypothesis testing scheme so that the type-II error rate β is minimized (this minimum value is the aforementioned $\beta(M)$) given that the confidence level $1-\alpha$ (the percentage of true NNs that are found by the algorithm) is always above the desired recall level. As we will show shortly, this optimization is a classical problem that has been answered by Neyman and Pearson [22], resulting in the following decision rule and parameters.

C. Efficient Decision Rule and Parameters

The aforementioned hypothesis testing scheme [22] decides which hypothesis to accept based on the *likelihood* of the null hypothesis, or the probability of the observed feature distances (of a data item) conditioning on that this data item is actually an NN: The null hypothesis is accepted if this likelihood is greater than $1-\alpha$ and is rejected otherwise. Hence, the specific decision rule is a function of the observed feature distances. Since the distribution of feature (projected) distances, as we have plotted in Figure 3, is complicated, the likelihood function and the resulting decision rule can be hard to describe and analyze. To overcome this issue, following the common practice in statistical inference, we summarize the observed feature distances into a single (scalar value) *statistic* and use a simple decision rule: The null hypothesis is accepted if and only if this statistic is greater than some *critical value*.

A simple and natural choice for such a statistic is the number V of repetitions in which the feature distance $\mathfrak{D}'(h(q), h(S))$ is shorter than a fixed threshold Θ. By Definition 1, the locality sensitivity of $h(\cdot)$ implies that $\mathfrak{D}'(h(q), h(S)) < \Theta$ in each repetition with probability $p \geq p_1$ if a data item is an NN, and with probability $p \leq p_2$ if it is a non-NN. The likelihood function in this decision rule is $1-B(T; M, p_1)$, where $B(\cdot)$ is the CDF of the binomial distribution with M trials and success rate p_1. Hence, the null hypothesis is accepted if $V > T$, for a parameter (critical value) T that we specify shortly, and the corresponding type-II error rate is $\beta = B(T; M, p_2)$.

With this formulation, under a fixed set of parameters M, α, and Θ (and correspondingly p_1 and p_2), we can solve the aforementioned optimization problem for an approximate formula of $\beta(M)$ as follows.

Theorem 2 ([22] p249)**.** *For fixed M, α, and Θ, the optimal T is the greatest integer such that $B(T; M, p_1) < \alpha$. With M large enough so that $M > \frac{5}{p_2(1-p_2)}$, this value is approximately $T^* \approx p_1 M - \Phi^{-1}(1-\alpha)\sqrt{p_1(1-p_1)M}$, where $\Phi(\cdot)$ is the CDF of the standard normal distribution and $\Phi^{-1}(\cdot)$ is its inverse.*

In this case, the optimal β (denoted by β^*), as a function of M, is approximately

$$\beta^*(M) \approx 1 - \Phi\left(\sqrt{\xi M} + \lambda \Phi^{-1}(\alpha)\right), \tag{4}$$

wherein $\xi = \frac{(p_1-p_2)^2}{p_2(1-p_2)}$ and $\lambda = \sqrt{\frac{p_1(1-p_1)}{p_2(1-p_2)}}$.

Moreover, (4) leads to a metric for comparing the effectiveness (discriminating power) per repetition of feature distance calculation. Since in most scenarios, p_1 and p_2 are on the same order and as a result the value of λ is roughly a constant, the optimal test power $1-\beta^*$ is roughly determined by ξM. Therefore, the value ξ, determined by p_1 and p_2 above, can be considered as the *unit discriminating power* per repetition: A larger ξ implies that the LSH pruning scheme is more effective in the sense it has a smaller β at a fixed recall level $1-\alpha$.

The formula (4) also guides the parameter tuning of our LSH pruning framework as follows. First, we find the best Θ such that each repetition is most effective, i.e., having the largest unit discriminating power ξ. This can be done by trying different values of Θ across a small interval. Then, with Θ fixed and $\beta(M)$ replaced by $\beta^*(M)$ in (4), the rightmost side of (3) becomes a well-defined function of M. Hence, the optimal number M of repetitions can be easily found, say via binary search, and the last parameter T can be dynamically adjusted by the algorithm to achieve a target recall level.

Our actual Algorithm 1 uses a different but equivalent decision rule, for the following reason: Dynamically adjusting T is less flexible than adjusting Θ, since T can only be integers, but Θ is a real number. In fact, our top-K (ANNS) procedure in § V-A makes full use of this flexibility. In order to dynamically adjust Θ, our alternative decision rule hinges on a different statistic than the one we already mentioned, which is the number V out of the M feature distances R_i's (for $i = 1, 2, \ldots, M$) in all M repetitions that satisfy $R_i < \Theta$. The new statistic, denoted by $R^{(T)}$, is the T^{th} *order statistic*, or the T^{th} smallest value, among R_i's for $i = 1, 2, \ldots, M$. Hence, the null hypothesis is accepted if $R^{(T)} < \Theta$, and is rejected instead. It is not hard to verify that these two decision rules are equivalent: $R^{(T)} < \Theta$ if and only if $V > T$.

D. Empirical Effectiveness Comparison with LDL1

In this subsection, we show empirically that our new L_2-P2S feature has higher unit discriminating power ξ than LDL1's, so by Theorem 2, each test in our algorithm is at least as effective as that in LDL1 for pruning. Recall that ξ is a function of p_1 and p_2, whose values depend on both the threshold Θ and the relative position of the NN and the non-NN with the query. Since the worst-case scenario in § III-C has ruled out any theoretical guarantee of ξ if the NN and non-NN can be arbitrary (adversarial) subspaces, our comparison has to be performed in an empirical scenario by Monte Carlo simulation, in which the NN and non-NN are drawn (separately) from two collections of subspaces that we construct as follows.

In our empirical scenario, the query $\vec{q}_0$ is fixed, and we create two collections of subspaces, one for NNs and the other for non-NNs. Each collection consists of 10,000 subspaces that have a fixed P2S distance to $\vec{q}_0$: $\mathfrak{D}_1(\vec{q}_0, \mathcal{S}') = r$ for all subspaces $\mathcal{S}'$ in the NN collection, and is equal to cr for all subspaces in the non-NN collection. In this way, we factor out the impact of varying P2S distances on our results.

To generate these two collections, we randomly sample 100 subspaces $\mathcal{S}$ with rank $\tau = 9$ and 100 queries $\vec{q}$ from the SIFT dataset ($d = 128$, see Table I), which are combined into 10,000 $(\vec{q}, \mathcal{S})$ pairs. Each $(\vec{q}, \mathcal{S})$ pair is then normalized to $(\vec{q}_0, \mathcal{S}')$ as follows. First, $(\vec{q}, \mathcal{S})$ is linearly translated to $(\vec{q}_0, \mathcal{S}'')$ with $\mathcal{S}'' = \mathcal{S} + \vec{q}_0 - \vec{q}$. This step centers all pairs to the fixed query $\vec{q}_0$ without changing the projected distances between the pair. Then, $(\vec{q}_0, \mathcal{S}'')$ is linearly scaled (while centered at $\vec{q}_0$) to $(\vec{q}_0, \mathcal{S}')$ so that $\mathfrak{D}_1(\vec{q}_0, \mathcal{S}')$ is the aforementioned constant (r in the NN collection, and cr in the non-NN collection). *This scaling step changes the projected distances proportionally: Suppose the scaling factor, for a $(\vec{q}_0, \mathcal{S}'')$ pair, is z, then the projected distance $\mathfrak{D}_p(W\vec{q}_0, W\mathcal{S}') = z\mathfrak{D}_p(W\vec{q}_0, W\mathcal{S}'')$ for both $p = 1$ (LDL1) and $p = 2$ (our algorithm). This is because the random Cauchy projection W is a linear mapping.*

For each subspace $\mathcal{S}'$ in the NN collection, we tune the threshold Θ and find the maximum unit discriminating power $\xi = \frac{(p_1-p_2)^2}{p_2(1-p_2)}$. Here, p_1 is calculated by simulating $\Pr(\mathfrak{D}'(W\vec{q}_0, W\mathcal{S}') < \Theta)$ using 10,000 random Cauchy projections W as we did for Figure 2 ($\mathfrak{D}'$ being L_2-P2S for us and L_1-P2S for LDL1), and p_2 is the empirical average of the probability $\Pr(\mathfrak{D}'(W\vec{q}_0, W\mathcal{S}_{non-NN}) < \Theta)$ wherein $\mathcal{S}_{non-NN}$ is uniformly sampled from the non-NN collection. The result is 10,000 different ξ values, one for each subspace $\mathcal{S}'$ in the NN collection, which by themselves form a distribution of ξ. Figure 4 plots the CDF of these values, measured under $r = 1$ and two different values of c (1.5 and 2).

Both subfigures of Figure 4 show that each test in our approach (with no dimension reduction as explained in § III-A) has better unit discriminating power than LDL1 (with dimension reduced to $\mu = 25$) in both cases of $c = 1.5$ and $c = 2$, and by a slightly larger margin in the latter case. This supports our previous claim that our algorithm is at least as effective as LDL1, and that is because we avoid the loss of locality sensitivity caused by aggressive dimension reduction in LDL1. Also, our algorithm achieves a positive ξ (which implies $p_1 > p_2$) on almost all NN instances, which shows that our new L_2-P2S feature distance is almost always locality sensitive in practice (despite the worst case shown in Figure 3).

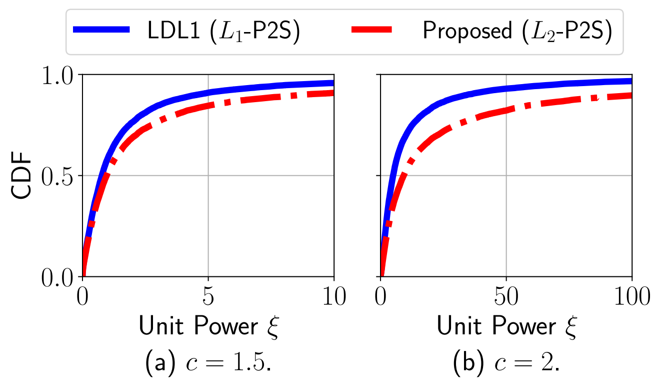

Fig. 4. The CDFs of maximum unit discriminating powers ξ from our simulation. Our algorithm has larger (better) ξ at all CDF quantiles (fixed y-coordinates) in both subfigures.

V. Evaluation

In this section, we conduct extensive study on L_p-P2S-ANNS using large scale datasets containing hundreds of thousands of subspaces. Our results show conclusively that our proposed scheme is faster than LDL1 by a factor of 4.8 to 54 on datasets with different ambient dimensions and subspace ranks, and under different L_p metrics.

A. Evaluation Setup

We use six datasets, shown in Table I, that are commonly used in ANNS literature [23]. Their sizes n' are up to one million and their ambient dimensions d range from 96 to 4096. Each such dataset $\mathcal{D}'$ is converted to a subspace dataset $\mathcal{D}$ containing $n = \lfloor n'/\tau \rfloor$ linear subspaces, each of which has rank τ and is spanned by τ consecutive vectors (data points) in $\mathcal{D}'$. For each such $\mathcal{D}$, the P2S-ANNS query set contains 100 points sampled uniformly at random from the set of query points associated with the corresponding $\mathcal{D}'$.

TABLE I
Summary of Datasets.

Dataset	n'	d	Type
Deep [24]	1.0M	96	Image
SIFT [25]	1.0M	128	Image
MNIST [26]	69.2K	784	Image
GIST [25]	1.0M	960	Image
Enron [27]	94.2K	1369	Text
Trevi [28]	99.1K	4096	Image

In this evaluation, we compare our proposed solution ("Ours") with LDL1 (the only existing scheme) in the following three aspects.

a) Memory usage: The memory usage includes two parts: the original dataset $\mathcal{D}$ (used in the verification stage) and the index (used in the pruning stage) that consists of the M repetitions of random Cauchy projections of each subspace in $\mathcal{D}$ as described earlier. The memory usage is equal to $4(n\tau\mu M + n\tau d)$ bytes when all vectors are encoded in 32-bit floating point format.

b) Top-K Query Recall: We measure the query accuracy by the top-K query recall defined as follows. Denote by $\mathcal{K}^*$ the top-K NNs with shortest L_p-P2S distances to the query $\vec{q}$ and by $\mathcal{K}$ the K approximate NNs in the result. The top-K query recall is equal to $|\mathcal{K}^* \cap \mathcal{K}|/K$. In this experiment, we set K to a small number 10 (since this problem is hard for large K) and report the average recall over all queries.

c) Query Time: We measure query times on a workstation running Ubuntu 18.04 with Intel Core i7–9800X CPU @ 3.80 GHz and 128 GB RAM. The computations of P2S distances are programmed in Python language (version 3.9.12), with all L_1-P2S distance calculations calling the `GUROBI` [16] optimizer (version 9.5.2). In addition to the measured times, we also show the overall speedup ratio η over the linear scan.

Parameter Setup: In all experiments, we tune parameters (M, T) of both our algorithm and LDL1 to achieve near-optimal query time at 0.9 average recall (at $K = 10$), using the technique described in § IV-C. We apply the JLT extension in § III-D to our algorithm in the MNIST, GIST, Enron, and Trevi datasets to reduce the memory usage. The final projection dimension μ is reported below for all experiments.

A last parameter, namely the threshold (critical value) Θ in our decision rule, needs to be specifically adapted to our setting of top-K queries. Recall from the last paragraph of § IV-D that Θ can be dynamically adjusted for the target recall. To this end, Θ needs to be proportional (by a constant slope, say, θ) to the radius r of NNs, which is a constant given by the ANNS problem, since the distribution of projected distances divided by r remains the same when r varies (as explained using italicized text in the third paragraph in § IV-D). As we only take top-K NNs for each query, the radius of the K^{th} nearest neighbor, denoted by r_K, varies from query to query. As a result, in principle, Θ should also vary with the query by the form of θr_K, yet the exact value of r_K is unknown to us (since it comes from the ground truth answer of ANNS).

To overcome this issue, we dynamically adjust Θ for each query by the following scheme similar to the SRS algorithm [14] for L_2-ANNS. We let Θ be $\theta \hat{r}_K$, where $\hat{r}_K$ is a dynamic and conservative (always guaranteeing $\hat{r}_K \geq r_K$) estimate of r_K initialized to a large enough number. The NN candidates are verified in the increasing order of their (T^{th} order statistic of) projected distances, starting from the most promising ones. We use a heap to track the top-K NNs (the set of result candidates) with the shortest original distances that have been calculated so far, and $\hat{r}_K$ is always the largest original distance in this heap. In this design, the projected distance to be pruned always increases, and the threshold always decreases, so we can safely end the verification stage (and return the result candidates) once our decision rule fails for some subspace.

Unfortunately, the constant slope θ in the above scheme can only be tuned assuming the knowledge of ground-truth NNs of all 100 queries. To make sure (the performance of) our scheme does not benefit from such knowledge at all, once this θ is tuned for each experiment, it is fixed for all queries in it; and we furthermore measure the recall and query time under a fresh (generated with new seeds) set of random Cauchy projections obliviously of the ground-truth knowledge (except the parameter θ).

B. Evaluation on Ambient Dimension

In this experiment, we fix the subspace rank at $\tau = 9$ like in the LDL1 paper [10] and evaluate on six datasets with increasing ambient dimensions. Table II shows that the overall speedup ratios η of our solutions are larger than those of LDL1, by a factor of 4.8 (on Enron) to 47 (on Trevi).

On Deep and SIFT, the two datasets with the lowest ambient dimensions, LDL1 can only afford one repetition of random Cauchy projection since its LSH speedup γ is barely larger than 1. As a result, its selectivity β is very large, resulting in $\eta < 1$ (slower than linear scan), on both datasets. In contrast, our solution achieves 24.3× (on Deep) 21.8× (on SIFT) speedups thanks to its much larger γ and much smaller β (by using 12 repetitions). On datasets with higher ambient dimensions, both LDL1 and our solution have larger γ, yet our solution has 500 times larger γ than LDL1. This allows our solution to use more repetitions (up to 93), and hence to have much lower β than LDL1, by a factor of 4.8 (on Enron) to 40 (on Trevi). Our memory usage is higher than LDL1, by a factor of 4.7 (on MNIST) to 10.9 (on SIFT), because we have tuned our parameter for the shortest query time instead of the minimum memory usage. Our algorithm, however, can be configured, say by using fewer repetitions M or a lower projection dimension μ, for less memory usage, whereas LDL1 cannot be configured to run faster than the times reported in Table II.

C. Evaluation on Subspace Rank

In this experiment, we vary subspace rank τ from 4 to 64 (in powers of 2) on SIFT (low ambient dimension) and from 4 to 128 on Trevi (high ambient dimension), with the case $\tau = 8$ replaced by $\tau = 9$ in the last experiment. Table III shows our solution outperforms LDL1, in terms of query time, by a factor of 16.5 to 54.5 on SIFT and 12.4 to 47.2 on Trevi. On both datasets, γ always increases with τ, whereas β decreases with τ on SIFT and increases with τ on Trevi.

D. Evaluation for $L_{1.2}$-P2S-ANNS

In this experiment, we explore the general L_p-P2S-ANNS problem with $p = 1.2$. We set $\tau = 9$ and use the two datasets (SIFT and Trevi) as in our last experiment. We apply the aforementioned p-stable extension to both algorithms and use libstable [29] for the generation of 1.2-stable random variables and compute $L_{1.2}$-$P2S$ distances using the Adam [17] convex optimizer implemented in Python.

Table IV shows that our solution is faster than LDL1, by a factor of 33.9 on SIFT and 20.3 on Trevi. Compared with L_1-P2S distances, $L_{1.2}$-P2S distances are slightly slower to compute under low d but are much faster under high d. As a result, the γ value (and the overall speedup η) in this experiment is larger than that in L_1-P2S-ANNS on SIFT, but is the other way around on Trevi.

TABLE II
EVALUATION RESULTS UNDER DIFFERENT DATASETS. NUMBERS IN BOLDFACE ARE THE BEST IN EACH GROUP.

Dataset	τ	Algorithm	μ	Recall	Memory (B)	T/M	LSH Speedup γ	Selectivity β	Query Time (s)	Overall Speedup η
Deep	9	LDL1	25	0.900	**484M**	$1/1$	1.55	37.2%	1.46K	0.98
		Ours	96	**0.909**	4.99G	$3/12$	**759**	**2.5%**	**58.7**	**24.3**
SIFT	9	LDL1	25	**0.905**	**612M**	$1/1$	1.84	57.7%	1.91K	0.89
		Ours	128	0.903	6.66G	$3/12$	**900**	**3.3%**	**78.2**	**21.8**
MNIST	9	LDL1	64	0.905	**231M**	$1/1$	4.46	35.9%	214	1.72
		Ours	128	**0.908**	1.09G	$2/25$	**2.77K**	**6.3%**	**26.5**	**13.9**
GIST	9	LDL1	64	0.902	**4.10G**	$1/1$	9.14	63.9%	7.84K	1.34
		Ours	128	**0.914**	32.5G	$10/56$	**5.29K**	**2.8%**	**399**	**26.2**
Enron	9	LDL1	64	0.900	**540M**	$1/1$	7.35	52.0%	549	1.52
		Ours	128	**0.906**	4.23G	$8/77$	**4.48K**	**12.0%**	**114**	**7.31**
Trevi	9	LDL1	32	0.907	**1.78G**	$1/12$	90.8	24.0%	3.32K	2.69
		Ours	256	**0.915**	10.9G	$15/91$	**43.2K**	**0.6%**	**70.1**	**127**

TABLE III
EVALUATION RESULTS UNDER DIFFERENT SUBSPACE RANKS.

Dataset	τ	Algorithm	μ	Recall	Memory (B)	T/M	LSH Speedup γ	Selectivity β	Query Time (s)	Overall Speedup η
SIFT	4	LDL1	25	0.901	**612M**	$1/1$	1.75	51.6%	3.76K	0.92
		Ours	128	**0.906**	6.66G	$3/12$	**815**	**5.1%**	**229**	**15.2**
	9	LDL1	25	**0.905**	**612M**	$1/1$	1.84	57.7%	1.91K	0.89
		Ours	128	0.903	6.66G	$3/12$	**900**	**3.3%**	**78.2**	**21.8**
	16	LDL1	25	0.901	**612M**	$1/1$	1.93	59.9%	1.19K	0.90
		Ours	128	**0.906**	5.63G	$2/10$	**966**	**3.4%**	**48.0**	**22.3**
	32	LDL1	48	0.903	**612M**	$1/1$	1.78	44.7%	683	0.99
		Ours	128	**0.904**	7.68G	$3/14$	**1.17K**	**1.3%**	**16.6**	**40.7**
	64	LDL1	80	**0.905**	**612M**	$1/1$	1.41	32.4%	476	0.97
		Ours	128	0.904	6.14G	$4/11$	**1.42K**	**1.1%**	**8.72**	**52.9**
Trevi	4	LDL1	64	0.905	**1.73G**	$1/4$	59.3	21.5%	4.23K	3.54
		Ours	256	**0.918**	10.1G	$16/83$	**33.7K**	**0.5%**	**112**	**133**
	9	LDL1	32	0.907	**1.78G**	$1/12$	90.8	24.0%	3.32K	2.69
		Ours	256	**0.915**	10.9G	$15/91$	**43.2K**	**0.6%**	**70.1**	**127**
	16	LDL1	64	0.902	**1.85G**	$1/9$	89.9	25.7%	2.32K	2.80
		Ours	256	**0.918**	10.1G	$10/83$	**51.7K**	**1.4%**	**103**	**63.1**
	32	LDL1	64	0.909	**1.80G**	$1/7$	122	37.0%	2.31K	2.34
		Ours	256	**0.918**	10.0G	$15/83$	**75.7K**	**1.7%**	**97.6**	**55.2**
	64	LDL1	64	0.909	**2.27G**	$1/17$	150	38.5%	2.90K	2.01
		Ours	256	**0.920**	10.7G	$10/90$	**129K**	**2.4%**	**142**	**41.1**
	128	LDL1	64	0.910	**2.54G**	$1/12$	122	35.8%	3.67K	2.19
		Ours	256	**0.912**	10.4G	$23/86$	**340K**	**3.7%**	**296**	**27.2**

TABLE IV
EVALUATION RESULTS FOR $L_{1.2}$-P2S-ANNS.

Dataset	τ	Algorithm	μ	Recall	Memory (B)	T/M	LSH Speedup γ	Selectivity β	Query Time (s)	Overall Speedup η
SIFT	9	LDL1	25	0.901	**612M**	$1/1$	1.48	40.0%	2.57K	0.93
		Ours	128	**0.908**	5.63G	$3/10$	**1.26K**	**2.4%**	**75.9**	**31.5**
Trevi	9	LDL1	64	0.906	**1.67G**	$1/2$	12.6	20.3%	781	2.77
		Ours	256	**0.915**	5.89G	$8/42$	**10.4K**	**1.4%**	**38.5**	**56.3**

VI. RELATED WORK

In this section, we elaborate on how our LSH pruning framework generalizes and improves existing algorithms in the literature. Many ANNS solutions use a pruning stage to reduce the number of costly calculations of original distances. For example, Bayesian LSH [20] for Jaccard-ANNS uses the number of MinHash collisions as the statistic in its decision rule: A data item is selected if it has more hash collisions with the query than a threshold. The decision rule therein is a special case of ours: their input is binary (hash collision or not), whereas our input is real-valued projected distances. To this end, our framework introduces and tunes a new parameter, namely the threshold Θ, in our decision rule.

The LDL1 scheme [10] in § II-B, which selects NN candidates separately in each repetition of random Cauchy projection, is another special case of our decision rule with $T = 1$. Our decision rule, which uses multiple repetitions collectively, has better query performance, since by Theorem 2, the optimal setting of T is greater than 1.

Moreover, the statistical reasoning on hypothesis testing is also a contribution of our framework. So far, only one paper [30] has applied the hypothesis testing theory to ANNS, but its formulation is much more complex than ours; and none of the existing works had analyzed the query time of LSH pruning as thoroughly as ours using formula (3) and Theorem 2. Hence, we are the first to point out that the decision rule in LSH pruning can be based on any statistics with high discriminating power, not just estimates of the original distances as commonly believed before [20].

VII. Conclusion

In this paper, we propose a new solution approach to P2S-ANNS, in the general L_p-P2S distance, that achieves significant query speedup over LDL1. Our solution prunes subspaces effectively and efficiently by the L_2-P2S distances that can be computed faster than the L_1-P2S distances in LDL1 by two to four orders of magnitude. Moreover, we develop a general LSH pruning framework that is grounded in statistics theory for the first time. Finally, by extensive experiments, we show that our proposed scheme has shorter query times than LDL1 by a factor of 4.8 to 54 on various datasets, subspace ranks, and L_p metrics.

Acknowledgment. This work was supported in part by the National Science Foundation under Grant No. CNS-2007006.

References

[1] Q. Huang, Y. Lei, and A. K. H. Tung, "Point-to-hyperplane nearest neighbor search beyond the unit hypersphere," in *Int. Conf. on Manage. of Data*, ser. SIGMOD '21. New York: ACM, 2021, pp. 777–789.

[2] S. Vijayanarasimhan, P. Jain, and K. Grauman, "Hashing hyperplane queries to near points with applications to large-scale active learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 276–288, 2014.

[3] R. Basri, T. Hassner, and L. Zelnik-Manor, "Approximate nearest subspace search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 2, pp. 266–278, 2011.

[4] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Conf. on Comput. Vis. and Pattern Recognit. Workshops (CVPRW)*. Boston, USA: IEEE, June 2015, pp. 27–35.

[5] N. Ailon and B. Chazelle, "The fast Johnson–Lindenstrauss transform and approximate nearest neighbors," *SIAM J. on Comput.*, vol. 39, no. 1, pp. 302–322, 2009.

[6] J. Meng, H. Wang, J. Xu, and M. Ogihara, "One index for all kernels (oniak): A zero re-indexing lsh solution to anns-alt (after linear transformation)," *Proc. VLDB Endow.*, vol. 15, no. 13, pp. 3937–3949, sep 2022.

[7] J. D. Semedo, A. Zandvakili, C. K. Machens, B. M. Yu, and A. Kohn, "Cortical areas interact through a communication subspace," *Neuron*, vol. 102, no. 1, pp. 249–259.e4, 2019.

[8] W.-C. Chang, F. X. Yu, Y.-W. Chang, Y. Yang, and S. Kumar, "Pre-training tasks for embedding-based large-scale retrieval," in *International Conference on Learning Representations*, 2020.

[9] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, 2013.

[10] J. Sun, Y. Zhang, and J. Wright, "Efficient point-to-subspace query in l_1 with application to robust object instance recognition," *SIAM J. on Imaging Sci.*, vol. 7, no. 4, pp. 2105–2138, 2014.

[11] K. Lu, Y. Ishikawa, and C. Xiao, "Mqh: Locality sensitive hashing on multi-level quantization errors for point-to-hyperplane distances," *Proc. VLDB Endow.*, vol. 16, no. 4, pp. 864–876, dec 2022.

[12] C. Sohler and D. P. Woodruff, "Subspace embeddings for the l1-norm with applications," in *Symp. on Theory of Comput.*, ser. STOC '11. New York, NY, USA: ACM, 2011, pp. 755–764.

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

[14] Y. Sun, W. Wang, J. Qin, Y. Zhang, and X. Lin, "Srs: solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index," *Proc. VLDB Endow.*, vol. 8, no. 1, pp. 1–12, sep 2014.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *Linear Methods for Regression*. New York, NY: Springer New York, 2009, p. 56.

[16] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023, product access under an academic license. [Online]. Available: https://www.gurobi.com

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[18] D. Ge, X. Jiang, and Y. Ye, "A note on the complexity of l p minimization," *Mathematical programming*, vol. 129, pp. 285–299, 2011.

[19] R. T. David Cherney, Tom Denton and A. Waldron, *Linear Algebra*. Davis, CA, USA: UC Davis, 2013.

[20] V. Satuluri and S. Parthasarathy, "Bayesian Locality Sensitive Hashing for Fast Similarity Search," *Proc. VLDB Endow.*, vol. 5, no. 5, p. 430–441, jan 2012.

[21] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng, "Query-aware locality-sensitive hashing for approximate nearest neighbor search," *Proc. VLDB Endow.*, vol. 9, no. 1, pp. 1–12, sep 2015.

[22] B. Rosner, *Fundamentals of Biostatistics*, 7th ed. Boston, MA, USA: Cengage Learning, 2010.

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

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

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

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

[27] W. W. Cohen, "Enron email dataset," http://www.cs.cmu.edu/∼enron/, 2015.

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

[29] J. R. del Val and F. S. Wattenberg, "Libstable: Fast, Parallel and High-Precision Computation of alpha-Stable Distributions in C and MATLAB," 2015. [Online]. Available: https://github.com/o-90/libstable

[30] A. Chakrabarti and S. Parthasarathy, "Sequential hypothesis tests for adaptive locality sensitive hashing," in *Int. Conf. on World Wide Web*, ser. WWW '15. Florence, Italy: WWW Conf., 2015, pp. 162–172.