Information Systems 101 (2021) 101807

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

The role of local dimensionality measures in benchmarking nearest neighbor search

Martin Aumüller [a], Matteo Ceccarello [b,*]

[a] IT University of Copenhagen, Denmark
[b] Free University of Bozen-Bolzano, Italy

ARTICLE INFO

Article history:
Received 14 March 2020
Received in revised form 4 January 2021
Accepted 17 May 2021
Available online 25 May 2021
Recommended by Gottfried Vossen

Keywords:
Nearest neighbor search
Benchmarking

ABSTRACT

This paper reconsiders common benchmarking approaches to nearest neighbor search. It is shown that the concepts of local intrinsic dimensionality (LID), local relative contrast (RC), and query expansion allow to choose query sets of a wide range of difficulty for real-world datasets. Moreover, the effect of the distribution of these dimensionality measures on the running time performance of implementations is empirically studied. To this end, different visualization concepts are introduced that allow to get a more fine-grained overview of the inner workings of nearest neighbor search principles. Interactive visualizations are available on the companion website.[1] The paper closes with remarks about the diversity of datasets commonly used for nearest neighbor search benchmarking. It is shown that such real-world datasets are not diverse: results on a single dataset predict results on all other datasets well.

1. Introduction

Nearest neighbor (NN) search is a key primitive in many computer science applications, such as data mining, machine learning and image processing. For example, Spring and Shrivastava very recently showed in [1] how nearest neighbor search methods can yield large speed-ups when training neural network models. In this paper, we study the classical k-NN problem in a metric space $(\mathcal{M}, D)$. Given a dataset $S \subseteq \mathcal{M}$, the task is to build an index on S to support the following type of query: For a query point $\mathbf{x} \in \mathcal{M}$, return the k closest points in S under the metric D.

In many practical settings, a dataset consists of points represented as high-dimensional vectors in a vector space $\mathbb{R}^d$. For example, word representations generated by the `glove` algorithm [2] associate with each word in a corpus a d-dimensional real-valued vector. Common choices for d are between 50 and 300 dimensions. Finding the true nearest neighbors in such a high-dimensional space is difficult, a phenomenon often referred to as the "curse of dimensionality" [3]. In practice, it means that finding the true nearest neighbors, in general, cannot be solved much more efficiently than by a linear scan through the dataset (requiring time $O(n)$ for n data points) or in space that is exponential in the dimensionality d, which is impractical for large values of d.

While we cannot avoid these general hardness results [4], most datasets that are used in applications are not *truly* high-dimensional. This means that the dataset can be embedded onto a lower-dimensional space without too much distortion. Intuitively, the intrinsic dimensionality (ID) of the dataset is the minimum number of dimensions that allows for such a representation [5]. There exist many explicit ways of finding good embeddings for a given dataset. For example, the Johnson–Lindenstrauss transformation [6] allows us to embed n data points in $\mathbb{R}^d$ into $\Theta((\log n)/\varepsilon^2)$ dimensions such that all pairwise distances are preserved up to a $(1+\varepsilon)$ factor with high probability. Another classical embedding often employed in practice is given by principal component analysis (PCA), see [7].

In this paper, we put our focus on *local measures of dimensionality*. In particular, we consider "local intrinsic dimensionality" (LID), a measure introduced by Houle in [5], an adapted version of "query expansion", a measure introduced by Ahle et al. in [8], and a local version of the "relative contrast" of the dataset introduced by He et al. in [9]. We defer a detailed discussion of these measures to Section 2. Intuitively, the LID of a data point $\mathbf{x}$ at a distance threshold $r > 0$ measures how difficult it is to distinguish between points at distance r and distance $(1+\varepsilon)r$ in a dataset. The Expansion of a data point $\mathbf{x}$ and a parameter $k > 0$ is the ratio of the distance of its $2k$-th nearest neighbor and its kth nearest neighbor. The relative contrast (RC) of a data point $\mathbf{x}$ is the ratio between the mean distance of $\mathbf{x}$ to the points in the dataset and the distance to its nearest neighbor. The relative contrast of a dataset is then the average RC over all data points.

* Corresponding author.
E-mail addresses: maau@itu.dk (M. Aumüller), matteo.ceccarello@unibz.it (M. Ceccarello).

[1] https://cecca.github.io/role-of-dimensionality/.

https://doi.org/10.1016/j.is.2021.101807

Most importantly, all three measures are *local* measures that can be associated with a single query. It was stated in [10] that the LID might serve as a characterization of the difficulty of k-NN queries. One purpose of this paper is to shed light on this statement, as well as to compare it with the other measures.

A focus of this paper is an empirical study of how these local measures influence the performance of NN algorithms. To be precise, we will benchmark five different implementations [11] which employ different approaches to NN search. Four of them (`HNSW` [12], `IVF` [13], `Annoy` [14]), and `ONNG` [15] stood out as most performant in the empirical study conducted by Aumüller et al. in [16]. Finally, we included the very recent LSH-based approach (`PUFFINN`) from Aumüller et al. [17] that promises to give recall guarantees with an adaptive query algorithm.

Our experiments are based on the `ann-benchmarks` system from [16]. We describe their benchmarking approach and the changes we made to their system in Section 3. We analyze the distribution of local dimensionality measures of real-world datasets in Section 4. For all measures, we will see that there is a substantial difference between these distributions among datasets. We will then conduct two sets of experiments: First, we fix a dataset and choose as query set the set of points with smallest, medium, and largest estimated dimensionality measure, for each one of LID, RC, and query expansion. In addition, we choose a set of "diverse" query points w.r.t. their estimated dimensionality measure. As we will see, there is a clear tendency such that the larger the LID (resp. the smaller the RC and Expansion), the more difficult the query for all implementations. Among the three measures, the LID is the one for which this effect is most pronounced. Next, we will study how the different dimensionality distributions between datasets influence the running time distribution. In a nutshell, it cannot be concluded that any of the three dimensionality measures by itself is a good indicator for the relative performance of a fixed implementation over datasets.

In the first part of our evaluation, we work in the "classical evaluation setting of nearest neighbor search". This means that we relate a performance measure (such as the achieved throughput measured in queries per second) to a quality measure (such as the average fraction of true nearest neighbors found over all queries). While this is the most commonly employed evaluation method, we reason that this way of representing results in fact hides interesting details about the inner workings of an implementation. Using non-traditional visualization techniques provide new insights into their query behavior on real-world datasets. As one example, we see that reporting average recall on the graph-based approaches from [12,15] hides an important detail: For a given query, they either find all true nearest neighbors or not a single one. This behavior is not shared by the three other approaches that we consider; all yield a continuous transition from "finding no nearest neighbors" to "finding all of them".

As a final point, we want, ideally, to benchmark on a collection of "interesting" datasets that show the strengths and weaknesses of individual approaches [18]. We will conclude that there is little diversity among the considered real-world datasets: While the individual performance observations change from dataset to dataset, the relative performance between implementations stays the same.

Our contributions The main contributions of this paper are

- a detailed evaluation of the distribution of local dimensionality measures of many real-world datasets used in benchmarking frameworks,
- an evaluation of the influence of these different dimensionality measures on the performance of NN,
- a systematic way to create query workloads of a wide range of difficulty for nearest neighbor search, search implementations,
- considerations about the result diversity, and
- an exploration of different visualization techniques that shed light on individual properties of certain implementation principles.

We hope that our approach and the tools developed will find use in future benchmarking studies. In particular, the way to choose query workloads with varying difficulties results in interesting testbeds to benchmark implementations.

Related work on benchmarking frameworks for NN. We extend the benchmarking system described in [16] as the starting point for our study. Different approaches to benchmarking nearest neighbor search are described in [19–21]. We refer to [16] for a detailed comparison between the frameworks.

Related work on the meaningfulness of nearest neighbor search. Beyer et al. [22] and Francois et al. [23] showed that under certain randomness assumptions and in the limit $d \to \infty$, nearest neighbor search queries become "meaningless", an effect usually referred to as the "concentration of distances". This means that the nearest and furthest neighbor of a data point become nearly indistinguishable. As mentioned in [9], these observations hold only asymptotically and usually do not occur in real-world datasets.

Related work on application-specific dataset annotations. In many fields such as multimedia image- or video-retrieval systems, both queries and their ground truth are chosen by human annotators. In addition, labels might be involved in judging the quality of a retrieval system. The paper [24] provides an excellent discussion of the challenges of providing good annotations and picking queries of varying difficulty in such a domain. The present work only considers the quality achieved by nearest neighbor search implementations, where the quality is judged by comparing the result to a query to the result of an exhaustive search. It would be interesting if the methods described here could be applied to guide the query selection process or in further automation of the semi-automated steps in [24].

Relation to conference version. This paper is an extended version of the SISAP 2019 paper [25], which focused mainly on LID as a measure of local dimensionality. To have a better understanding of how much our observations generalized, this version includes two other measures (query expansion and relative contrast) and features a new NN implementation based on LSH (`PUFFINN`).

2. Local dimensionality measures

2.1. Local intrinsic dimensionality

We consider a metric space $(\mathcal{M}, D)$ with $D: \mathcal{M} \times \mathcal{M} \to \mathbb{R}$. As described in [26], we consider the distribution of distances within this space with respect to a reference point $\mathbf{x}$. Such a distribution is induced by sampling n points from the space $\mathcal{M}$ under a certain probability distribution. We let $F: \mathbb{R} \to [0, 1]$ be the cumulative distribution function of distances to the reference point $\mathbf{x}$.

Definition 1 (*[5]*)**.** The local continuous intrinsic dimension of F at distance r is given by

$$\mathrm{ID}_F(r) = \lim_{\varepsilon \to 0} \frac{\ln(F((1+\varepsilon)r)/F(r))}{\ln((1+\varepsilon)r/r)},$$

whenever this limit exists.

The measure relates the increase in distance to the increase in probability mass (the fraction of points that are within the ball of radius r and $(1+\varepsilon)r$ around the query point). Intuitively, the larger the LID, the more difficult it is to distinguish true nearest neighbors at distance r from the rest of the dataset. As described in [10], in the context of k-NN search we set r as the distance of the kth nearest neighbor to the reference point $\mathbf{x}$.

Estimating LID We use the Maximum-Likelihood estimator (MLE) described in [27,28] to estimate the LID of $\mathbf{x}$ at distance r. Let $r_1 \leq \cdots \leq r_k$ be the sequence of distances of the k-NN of $\mathbf{x}$. The MLE $\hat{\mathrm{ID}}_\mathbf{x}$ is then

$$\hat{\mathrm{ID}}_\mathbf{x} = -\left(\frac{1}{k}\sum_{i=1}^{k}\ln\frac{r_i}{r_k}\right)^{-1}. \qquad (1)$$

Amsaleg et al. showed in [26] (full version [28]) that MLE estimates the LID well. We remark that in very recent work, Amsaleg et al. proposed in [29] a new MLE-based estimator that provides more accurate LID estimates for smaller values of k compared to (1).

In the following, we denote with LID_k the LID estimate obtained with parameter k. If the parameter is omitted, we refer to LID_{100}.

Local ID representation theorem The *Local ID Representation* Theorem by Houle [30, Theorem 2] shows that Definition 1 of $\mathrm{ID}_F(r)$ is equivalent to the relation $F(x)/F(w) = (x/w)^{\mathrm{ID}_F(r)}$ in an asymptotic sense, see [30] for the details. Here, the local intrinsic dimensionality relates the ratio of the CDF of two distances, which intuitively means the number of neighbors within the ball at these two distances, to the "possible" growth $(x/w)^{\mathrm{ID}_F(r)}$ if the intrinsic dimensionality *were* $\mathrm{ID}_F(r)$.

The representation theorem gives precise conditions under which the *expansion dimension* (ED) introduced by Karger and Ruhl in [31] and the *generalized expansion dimension* (GED) by Houle et al. in [32] are an accurate estimate of the intrinsic dimensionality. In these works, two distance thresholds r_1 and r_2 are fixed, and – with the same reasoning as above – the dimensionality of $\mathbf{x}$ is estimated as $\log(B_{r_2}/B_{r_1})/\log(r_2/r_1)$, where B_{r_i} means the number of points within distance r_i for $i \in \{1, 2\}$.[2]

For our application to k-NN search, it is natural to fix the distances based on the distance of the kth nearest neighbor, relating it to another, further away neighbor. We introduce two specific concepts of setting these distances below.

2.2. Query expansion

The concept of the Expansion around a query point at a distance threshold $r > 0$ was introduced by Ahle et al. in [8]. In their work, the query expansion $c^*_\mathbf{x}$ is the largest $c^*_\mathbf{x} > 0$ such that the number of points within distance $c^*_\mathbf{x} r$ is at most twice the number of points at distance r. They use this concept to show that an LSH approach can adapt to the query expansion. More precisely, the larger the query expansion, the less work is conducted by their adaptive query algorithm in expectation.

For our use case in k-NN search, we adapt the notion of query expansion as follows.

Definition 2. Given a data set S, an integer $k > 0$, and a data point $\mathbf{x}$, the Expansion of $\mathbf{x}$ at k with respect to $k' > k$ is $\mathrm{dist}(\mathbf{x}, \mathbf{x}_{k'})/\mathrm{dist}(\mathbf{x}, \mathbf{x}_k)$, where $\mathbf{x}_i$ is the ith nearest neighbor of $\mathbf{x}$ in S for $1 \leq i \leq |S|$.

Of course, this is a cardinality-based interpretation of the (generalized) expansion dimension in [31,32]. We refer to the value $\frac{\log(k'/k)}{\log(\mathrm{dist}(\mathbf{x},\mathbf{x}_{k'})/\mathrm{dist}(\mathbf{x},\mathbf{x}_k))}$ as the *expansion dimension* of $\mathbf{x}$ at k w.r.t. k'.

In the following, we denote with $\mathrm{Expansion}_{k'|k}$ the expansion dimension of $\mathbf{x}$ at k with respect to k'. If the parameters are omitted, we refer to $\mathrm{Expansion}_{20|10}$.

2.3. Relative contrast

The concept of relative contrast (RC) was introduced by He et al. in [9]. Here, we concentrate on the following local variant.

Definition 3. Given a data set S, an integer $k > 0$, and a data point $\mathbf{x}$, let d_{mean} be the average distance of $\mathbf{x}$ to the points in S. The local relative contrast (LRC) of $\mathbf{x}$ in S is then $d_{\mathrm{mean}}/\mathrm{dist}(\mathbf{x}, \mathbf{x}^*_k)$, where $\mathbf{x}^*_k$ is the kth nearest neighbor of $\mathbf{x}$ in S.

The relative contrast of the dataset S is the average local relative contrast over all points in a query set. It was shown in [9] that – if the relative contrast of the dataset is known – there is a way to choose LSH parameters to adapt to the RC. In the same way as query expansion, higher contrast means faster query times.

LRC has a much more global view on the dataset while Expansion considers distances between close points. As with the Expansion above, the relative contrast can be considered as a special cardinality-based variant of GED. Assuming that the distance distribution is such that the average and median distance are close to other, we refer to the value $\log(|S|/(2k))/\log(d_{\mathrm{mean}}/\mathrm{dist}(\mathbf{x}, \mathbf{x}^*_k))$ as the *local relative contrast dimension* of $\mathbf{x}$ at k, which we denote with RC_k. If k is not specified, we implicitly refer to RC_{100}.

3. Overview over the benchmarking framework

We extend the `ann-benchmarks` system described in [16] to conduct our experimental study. Ann-benchmarks is a framework for benchmarking NN search algorithms. It covers dataset creation, performing the actual experiment, and storing the results of these experiments in a transparent and easy-to-share way. Moreover, results can be explored through various plotting functionalities, e.g., by creating a website containing interactive plots for all experimental runs.

Ann-benchmarks interfaces with a NN search implementation by calling its preprocess (index building) and search (query) methods with certain parameter choices. Implementations are tested on a large set of parameters usually provided by the original authors of an implementation. The answers to queries are recorded as the indices of the points returned. Ann-benchmarks stores these parameters together with further statistics such as individual query times, index size, and auxiliary information provided by the implementation. See [16] for more details.

Compared to the system described in [16], we added tools to estimate the LID based on Eq. (1), to estimate the query Expansion based on Definition 2, to estimate the RC based on Definition 3, pick "challenging query sets" according to the LID, query expansion, and RC of individual points, added new datasets and implementations, and extended the website to interactively explore the results. Moreover, we implemented a mechanism that allows an implementation to provide further query statistics after answering a query. To showcase this feature, all implementations in this study report the number of distance computations performed to answer a query.[3]

4. Algorithms and datasets

4.1. Algorithms

Nearest neighbor search algorithms for high dimensions are usually graph-, tree-, or hashing-based. We refer the reader

[2] We remark that the Amsaleg et al. showed in [28] that (G)ED can be seen as a special case of *regularly varying functions*, which for certain parameter choices other than (G)ED converge to the MLE estimator in (1).

[3] We thank the authors of the implementations for their help and responsiveness in adding this feature to their library.

to [16] for an overview over these principles and available implementations. In this study, we concentrate on the three implementations considered most performant in [16], namely HNSW [12], Annoy [14] and FAISS-IVF [13] (IVF from now on). We consider the very recent graph-based approach ONNG [15], and the recent LSH-based approach PUFFINN [17] in this study as well.

HNSW and ONNG are graph-based approaches. This means that they build a k-NN graph during the preprocessing step. In this graph, each vertex is a data point and a directed edge (u, v) means that the data point associated with v is "close" to the data point associated with u in the dataset. At query time, the graph is traversed to generate candidate points. Algorithms differ in details of the graph construction, how they build a navigation structure on top of the graph, and how the graph is traversed.

Annoy is an implementation of a random projection forest, which is a collection of random projection trees. Each node in a tree is associated with a set of data points. It splits these points into two subsets according to a chosen hyperplane. If the dataset in a node is small enough, it is stored directly and the node is a leaf. Annoy employs a data-dependent splitting mechanism in which a splitting hyperplane is chosen as the one splitting two "average points" by repeatedly sampling dataset points. In the query phase, trees are traversed using a priority queue until a predefined number of points is found.

IVF builds an inverted file based on clustering the dataset around a predefined number of centroids. It splits the dataset based on these centroids by associating each point with its closest centroid. During query it finds the closest centroids and checks points in the dataset associated with those.

PUFFINN uses an adaptive trie-like multi-layer LSH data structure to guide the search. Using the probabilistic nature of LSH, it exploits adaptive termination criteria to give guaranteed recall [17] without the need of parameter tuning as in the other approaches. We note that PUFFINN does not support Euclidean distance and is thus missing in some plots.

We remark that we used both IVF and HNSW implementations from FAISS.[4]

4.2. Datasets

Table 1 presents an overview over the datasets that we consider in this study. We restrict our attention to datasets that are usually employed in connection with Euclidean distance and Angular/Cosine distance: SIFT, MNIST, and GLOVE are among the most-widely used datasets for benchmarking nearest neighbor search algorithms. Fashion-MNIST is considered as a replacement for MNIST, which is usually considered too easy for machine learning tasks [33].

For each dataset, we compute the LID distribution with respect to the 100-NN as discussed in Section 2, in order to get a stable estimate. Furthermore, we compute the $\text{Expansion}_{20|10}$ as discussed in Section 2. The RC_{100} is estimated by computing the distance of each point to a sample of 3000 points. In order to investigate the impact of the choice of parameters on the estimates and the ranking, which will be discussed below, we also estimate LID_{10}, RC_{10}, and $\text{Expansion}_{100|5}$.

Fig. 1 provides a visual representation of the estimated distributions of LID, RC, and Expansion of each dataset. The gray curve reports the distribution of LID_{100}, RC_{100}, and $\text{Expansion}_{20|10}$. The overlayed red curve reports the distribution of LID_{10}, RC_{10}, and $\text{Expansion}_{5|100}$. While the datasets differ widely in their original dimensionality, the median LID ranges from around 13 for MNIST to about 26 for GLOVE-2M. All distributions are asymmetric and show a long tail behavior, differing in their shape and the length of the tail. We observe that changing parameterization of the estimators yields similar distributions. Remarkably, the distribution of Expansion and LID are very similar under both parameterizations of the estimator. This is in line with the observation in [28] that variants of the Expansion can be viewed as an estimator of LID.

Fig. 2 reports the change in ranking of vectors of the datasets when we define the ranking using different parameterizations of the LID estimator, to further investigate the behavior of individual vectors. In particular, we partition the rankings obtained with LID_{100} into 30 groups, arranged from the highest ranked to the lowest ranked from top to bottom on the y axis. For each group, we estimate the probability distribution of the ranking assigned to the vectors in the group according to LID_{10}. We observe that for vectors with extremal LID values the ranking is rather unlikely to change, whereas for other vectors the displacement is much more pronounced. The behavior for RC and Expansion is similar.

These two observations combined suggest that, while the overall distribution of estimates remains similar under different parameterizations, many vectors are likely to trade ranks under LID_{10}. Given the central role that ranking according to these dimensionality measures plays in this work, in the following we use the more accurate LID_{100}, RC_{100}, and $\text{Expansion}_{20|10}$ estimates. In Appendix A.3 we compare how the different parameterizations of the estimators influence the difficulty of the workloads. In a nutshell, using a small value of k for estimation results in query workloads that are not meaningful.

5. Evaluation

This section reports on the results of our experiments. Due to space constraints, we only present some selected results. More results can be explored via interactive plots at https://cecca.github.io/role-of-dimensionality/, which also contains a link to the source code repository. For a fixed implementation, the plots presented here consider the Pareto frontier over all parameter choices [16]. Tested parameter choices and the associated plots are available on the website.

Experimental setup Experiments were run on 2x 14-core Intel Xeon E5-2690v4 (2.60 GHz) with 512GB RAM using Ubuntu 16.10 (kernel 4.4.0). Index building was multi-threaded, queries where answered in a single thread.

Quality and performance metrics As quality metric we measure the individual recall of each query, i.e., the fraction of points reported by the implementation that are among the true k-NN, and the *relative error*, defined as the ratio between the sum of distances from the query of the returned k-NN and the true k-NN. Usually, we report on the *average recall* and *average relative error* by averaging the individual recall/error values over all queries in the query workload, as specified below. We perform all queries as 10-NN queries.[5] As performance metric, we record individual query times and the total number of distance computations needed to answer all queries. We usually report on the throughput, i.e. the average number of queries that can be answered in one second, in the plots denoted as QPS for *queries per second*.

Objectives of the experiments Our experiments are tailored to answer the following questions:

(Q1) How do LID, Expansion, and RC correlate with each other? (Section 5.1)
(Q2) How do the LID, Expansion, and RC of a query set influence performance of an implementation? (Sections 5.2 and 5.3)

[4] https://github.com/facebookresearch/faiss

[5] We report on results with 100-NN in Appendix A.2. While queries take longer to answer, the general observations also hold true for larger neighbor counts.

Table 1
Datasets under consideration with their average and median LID_{100}, their $\text{Expansion}_{20|10}$, and their RC_{100}. .

Dataset	Data points	dim.	LID		RC		Expansion		Metric
			Avg	Median	Avg	Median	Avg	Median	
Fashion-MNIST	65 000	784	15.40	13.75	7.39	6.90	16.86	14.28	Euclidean
GLOVE	1 183 514	100	17.91	17.62	14.27	14.59	17.57	15.87	Cosine
GLOVE-2M	2 196 018	300	25.83	23.21	22.25	23.92	25.04	20.37	Cosine
GNEWS	3 000 000	300	21.02	19.95	15.39	15.23	21.42	18.93	Cosine
MNIST	65 000	784	13.87	13.05	11.68	11.54	14.05	12.56	Euclidean
SIFT	1 000 000	128	19.41	18.98	11.35	11.32	20.20	18.36	Euclidean

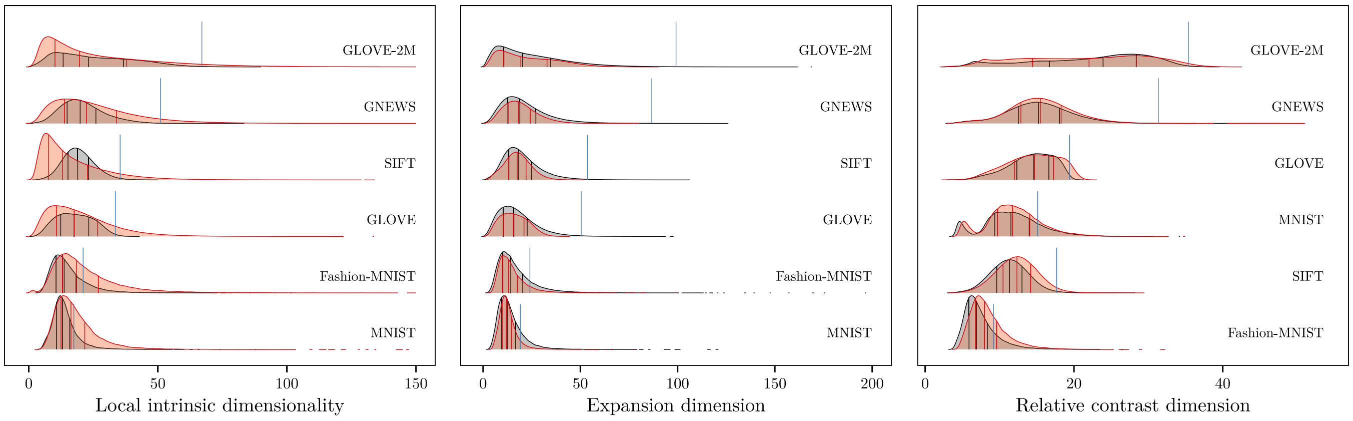

Fig. 1. Dimensionality measures for each dataset. Lines within each distribution curve correspond to the 25, 50 and 75 percentiles. For each dataset, we report two distributions: the gray curve represents the distribution of LID_{100}, RC_{100}, and $\text{Expansion}_{20|10}$; the red curve reports estimates for LID_{10}, RC_{10}, and $\text{Expansion}_{100|5}$, respectively. The blue line marks the 10 000 most difficult queries according to each score. Datasets are sorted by the median value of the measure. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

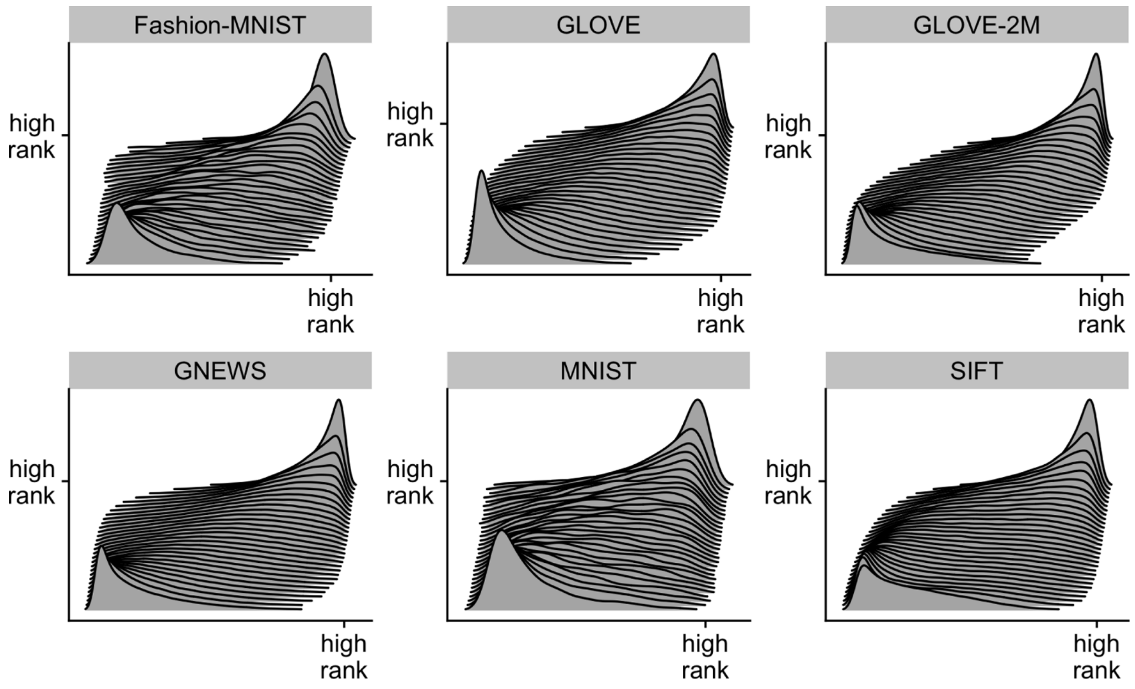

Fig. 2. Change in LID-based ranking as a consequence of different parameterizations. The plot depicts, as a heatmap, the probability distribution of the rank with the LID estimated with $k = 10$ conditioned on the rank estimated with $k = 100$ falling in one of the 30 buckets in which the x axis is split.

(Q3) How well does the number of distance computations reflect the relative running time performance of the tested implementations? (Section 5.4)

(Q4) How diverse are measurements obtained on datasets? Do relative differences between the performance of different implementations stay the same over multiple datasets? (Section 5.4)

(Q5) How concentrated are quality and performance measures around their mean for the tested implementations? (Section 5.5)

Choosing query sets For each dataset, we select eight different query sets:

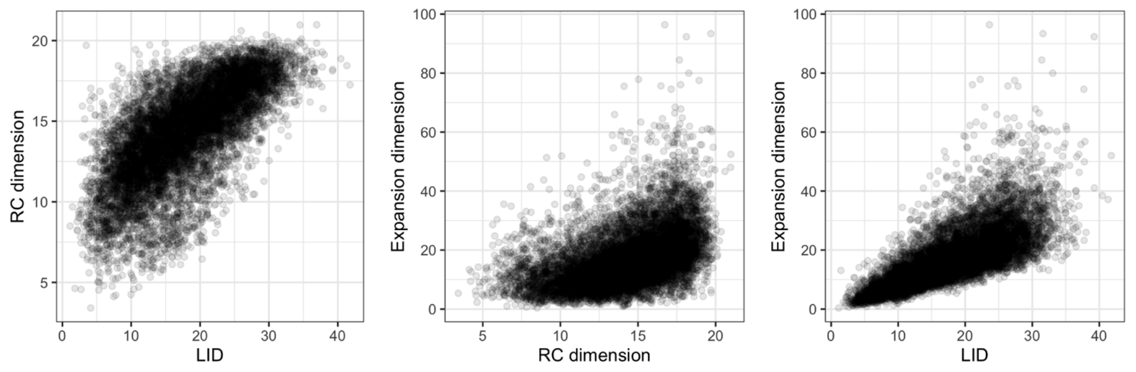

Fig. 3. Scatterplots relating LID_{100}, $\text{Expansion}_{20|10}$, and RC_{100} for the GLOVE dataset for a sample of 10 000 data points. All the scales are linear.

Table 2
Pearson correlation of between the three different measures for each dataset. The correlations, although mild, are all statistically significant.

Dataset	$\text{LID}_{100}/\text{RC}_{100}$	$\text{Expansion}_{20\|10}/\ \text{RC}_{100}$	$\text{LID}_{100}\ /\ \text{Expansion}_{20\|10}$
GLOVE	0.691	0.452	0.729
GLOVE-2M	0.518	0.344	0.814
GNEWS	0.372	0.174	0.732
MNIST	0.666	0.377	0.637
FASHION-MNIST	0.575	0.389	0.753
SIFT	0.533	0.320	0.623

easy the 10 000 points with the lowest estimated LID/expansion/RC

medium the 10 000 points around the data point with median estimated LID/expansion/RC

hard the 10 000 points with the highest estimated LID/expansion/RC

diverse 5 000 points chosen so to span the entire range of LID values (resp. Expansion/RC values). For the LID, we split all data points up into buckets, according to their rank by LID. For each query, we pick a non-empty bucket uniformly at random, and inside the bucket we pick a random point (with repetition). For Expansion and RC, we pick the 1 500 points with smallest and largest values, and add 2 000 points picked uniformly at random from the remaining points (with repetition).

Fig. 1 marks with a red line the LID used as a threshold to build the *hard* queryset.

Main takeaways The following experimental evaluation presents a lot of results, giving the following main insights. First, we can use local dimensionality measures to build benchmark query sets of varying difficulty. Second, among these measures, the Local Intrinsic Dimensionality is the single most effective one at selecting queries of the desired accuracy. Then, the *diverse* query set is a good general benchmark, in that it includes queries of a wide range of difficulties. Finally, average performance measures are convenient but often hide interesting behavior, which is best studied by looking at their distribution.

5.1. How well do the local dimensionality measures correlate?

Fig. 3 visualizes the correlation between the three different local dimensionality measures. As our working hypothesis, a higher LID score is associated with a higher difficulty for a query, as is a higher expansion or RC score. The plot shows some correlation between the scores: points with a high LID also have a high expansion and RC dimension, and vice versa. If we compute the Pearson correlation between the different measures (reported in Table 2) we can see that they are mildly correlated. We observe that both the RC and expansion dimensions correlate more strongly with LID than they do with each other.

5.2. Influence of dimensionality measures on performance

Fig. 4 reports the performance of different configurations of all the algorithms we consider on the GLOVE, GLOVE-2M, and SIFT datasets, drawing queries according to the LID. In these plots, the best performance is attained in the upper right corner: high recall and high throughput.

We observe a clear influence of the LID of the query set on the performance: the more difficult the query set, i.e., the larger the LID, the more down and to the left the graphs move, for all algorithms. This means that for higher LID it is more expensive, in terms of time, to answer queries with good recall.

For all datasets except GLOVE-2M (and GNEWS with the difficult query set), almost all implementations were still able to achieve close to perfect recall with the parameters set. This means that even for queries with large LID there are points in the dataset that can be efficiently separated from the others.

We now turn our focus on the relationship of the three dimensionality measures with the performance of the algorithm. Fig. 5 considers the same setup as before showing the results for `Annoy` for LID, Expansion, and RC. First of all, we observe that *easy*, *middle*, and *hard* query sets show the same behavior we observed in Fig. 4: selecting queries with higher LID/expansion/RC makes them more difficult to solve for the algorithm. The three dimensionality measures yield query sets of comparable difficulty.

We consider an additional quality metric, namely the *relative error*, in order to investigate whether false positives reported in the k-NN are very far away from the query compared to the true positives. This metric has a similar behavior to the recall: high quality solutions (thus with small relative error) entail a lower throughput. Therefore, instead of reporting a plot directly comparing the queries per second to the relative error (which can be nonetheless found in the online interactive supplemental material), in Fig. 6 we compare the relative error with the recall, distinguishing runs by the difficulty of their workload. Unsurprisingly, we find that a high recall corresponds to a low relative error, and vice-versa. Interestingly, the difficulty of the workload influences the relationship between the two metrics: for the same recall easy workloads have a worse relative error than difficult ones. This means that when algorithms are including false positives in the reported k-NN, in terms of distance from the query these false positives are far more off for easy than for hard workloads. This is to be expected: intuitively a query is hard precisely for the reason that it is very hard to distinguish its k-NN from the immediately following ones. Therefore, when an

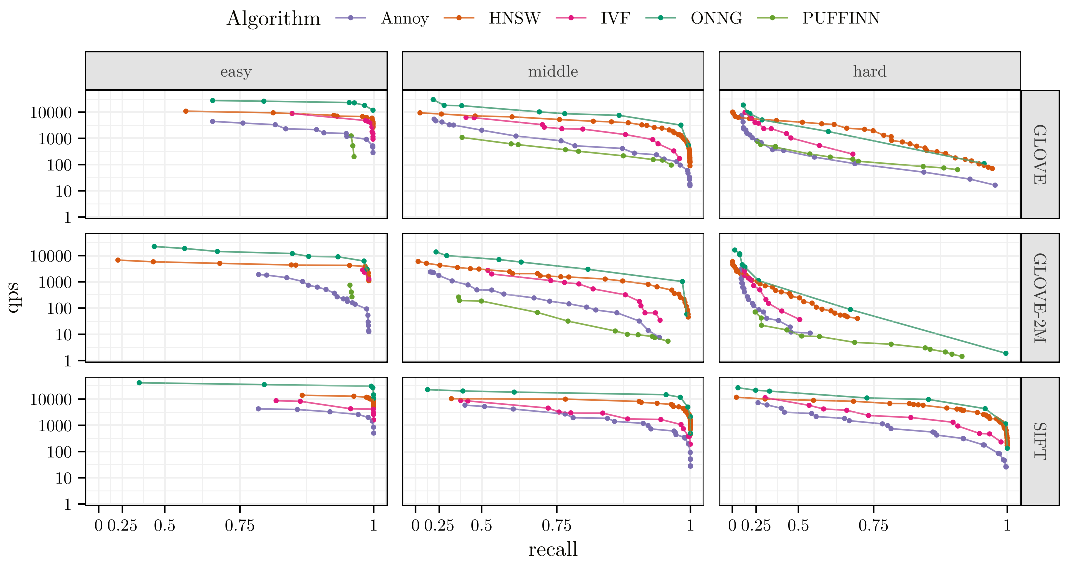

Fig. 4. Recall-QPS (1/s) tradeoff – up and to the right is better – for queries selected according to LID, solved using different algorithms. Three datasets are considered here: GLOVE, GLOVE-2M and SIFT. The scale is logarithmic on the y axis and exponential on the x axis, to take into account the scale of the data.

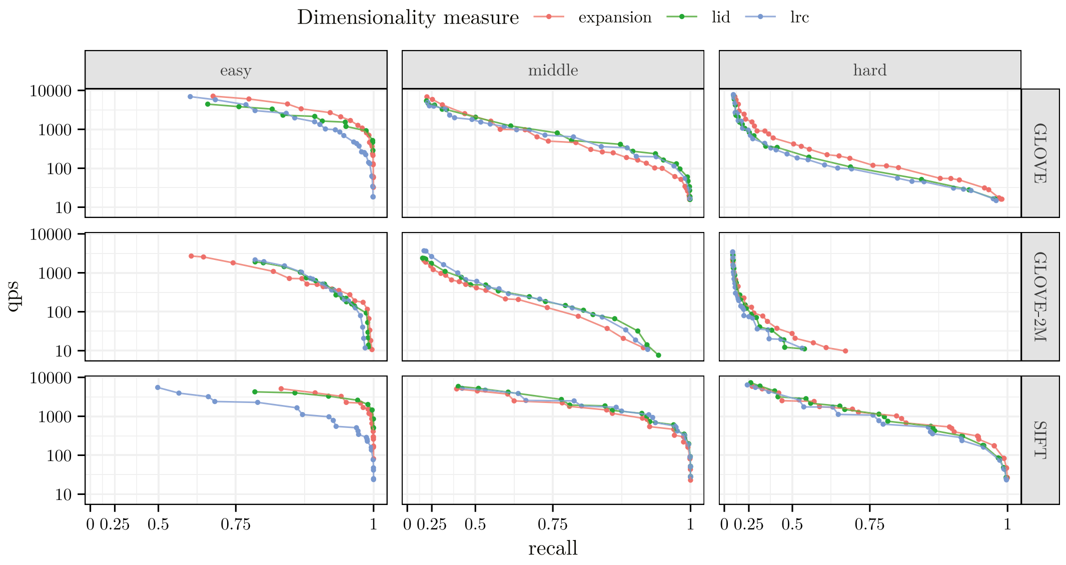

Fig. 5. Recall-QPS (1/s) tradeoff – up and to the right is better – for algorithm Annoy solving queries selected according to LID, RC, and Expansion. Three datasets are considered here: GLOVE, GLOVE-2M and SIFT. The scale is logarithmic on the y axis and exponential on the x axis, to take into account the scale of the data.

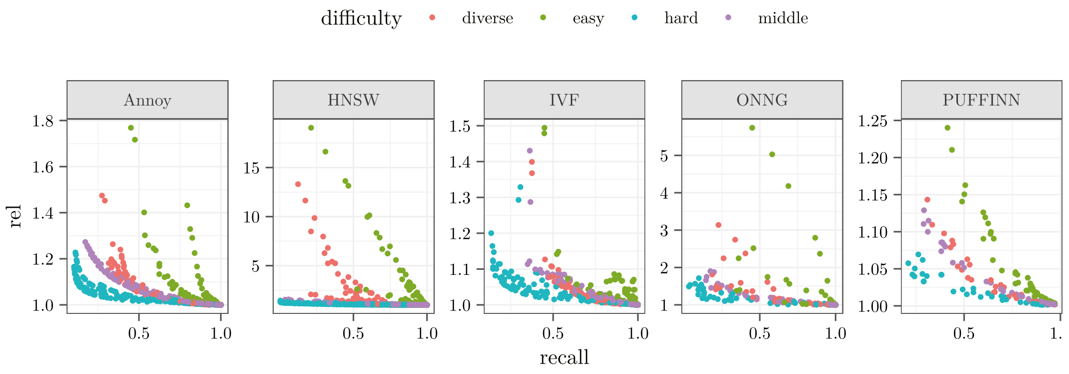

Fig. 6. Comparison of recall and relative error, for all the algorithms considered, with difficulty estimated according to LID. Each dot represents the average score obtained by a configuration of the given algorithm on a dataset. Dots are colored according to the difficulty of the workload.

algorithm misreports a point as being in the k nearest neighbors, most likely it is not much farther away from the query compared to the true k nearest neighbors. Conversely, for easy queries points that are not in the k nearest neighbors are much farther away. Therefore, for similar recall values, easy queries will be affected by worse relative error than difficult queries.

These general observations can be further investigated on a *per-implementation-basis*. Fig. 6 shows that for a fixed average recall value, implementations differ widely in the relative error they attain. In fact, PUFFINN can be seen as the most robust implementation, achieving considerable smaller relative errors than the other four implementations. In particular, *graph-based approaches* such as ONNG and HNSW have large relative error, which means that the false positives are far away from the true nearest neighbors. This phenomenon will be further explored in Section 5.4.

To better investigate the influence that dimensionality measures have for all datasets and implementations, consider Fig. 7, which reports the change in performance of the fastest configuration attaining recall at least 0.9, for each algorithm. Clearly, all measures allow to select query sets which are progressively more difficulty to solve accurately for all algorithms. However, as shown by the labels in each plot, the LID allows to select *easy* and *hard* querysets that have a wider performance gap than the ones selected by Expansion or RC, also for the datasets in which all implementations achieve high recall on the hard query set.

5.3. Predictive quality of dimensionality measures

In the previous two subsections, we found evidence that all dimensionality measures allow to pick query sets of various difficulties. Fixing the implementation and considering all datasets, how well does a dimensionality measure work between two different datasets? Fig. 8 reports the queries per second of Annoy for a certain choice of datasets, with queries chosen from the middle, hard, and diverse query set.

Comparing results to the dimensionality measurements depicted in Fig. 1, we first observe that the estimated median LID, RC and Expansion all give a good estimate on the relative performance of the algorithms on the data sets: recall that in Fig. 1 the datasets are sorted by median score. (The plot is missing lines for GNEWS and GLOVE-2M, which are considerably more challenging according to Fig. 7.) As an exception, SIFT (middle) is much easier than predicted by its LID and Expansion distribution, but the RC measure predicts this, ranking SIFT lower than GLOVE. In particular, the hard SIFT instance (orange solid line) is as challenging as the medium GLOVE version (green dotted line). On the other hand, RC classifies MNIST as rather difficult to index, in particular compared to Fashion-MNIST. The plot on the right in Fig. 8 clearly indicates that this is not true, and instead the two datasets are basically equivalent. From this, we cannot conclude that the considered local dimensionality measures as a single indicator explain performance differences of an implementation across different datasets.

It can also be seen from the plot that the diverse query set is more difficult than the medium query set. In particular, at high recall it generally becomes nearly as difficult as the difficult dataset. For many implementations, the reason for this behavior is that they cannot adapt to the difficulty of a query. They only achieve high average recall when they can solve sufficiently many queries with high LID or Expansion. The parameter settings that allow for such guarantees slow down answering the easy queries by a lot. This manifests in running times that are indistinguishable from those on the hard dataset, while only roughly 30% of the queries are characterized as difficult ones. As we shall see in Section 5.5, some algorithms are indeed able to adapt to the difficulty of the query. We believe that the "diverse" query sets thus allow for challenging benchmarking datasets for adaptive query algorithms.

As a side note, we remark that Fashion-MNIST is as difficult to solve as MNIST for all implementations, and is by far the easiest dataset for all implementations. Thus, while there is a big difference in the difficulty of solving the classification task [33], there is no measurable difference between these two datasets in the context of NN search.

5.4. Diversity of results

Fig. 9 gives an overview over how algorithms compare to each other among all "medium difficulty" querysets, selected according to the LID. Results for Expansion- and RC-based querysets are similar. We consider two metrics, namely the number of queries per second (top plot), and the number of distance computations (bottom plot). For two different average recall thresholds (0.75 and 0.9) we then select, for each algorithm, the best performing parameter configuration that attains at least that recall. For each dataset, the plots report the ratio with the best performing algorithm on that dataset, therefore the best performer is reported with ratio 1. Considering different dataset, we see that there is little variation in the ranking of the algorithms. Only the two graph-based approaches trade ranks, all other rankings are stable. Annoy makes fewer distance computations (hence ranks higher in the figure) but is consistently outperformed by IVF.[6]

Comparing the number of distance computations to running time performance, we see that an increase in the number of distance computations is not reflected in a proportional decrease in the number of queries per second. This means that the candidate set generation is in general more expensive for graph-based approaches, but the resulting candidate set is of much higher quality and fewer distance computations have to be carried out. Generally, both graph-based algorithms are within a factor 2 from each other, whereas the other two need much larger candidate lists to achieve a certain recall. The relative difference usually ranges from 5x to 30x more distance computations for the non-graph based approaches, in particular at high recall. This translates well into the performance differences we see in this setting: consider for instance Fig. 4, where the lines corresponding to HNSW and ONNG upper bound the lines relative to the other algorithms.

5.5. Reporting the distribution of performance

In the previous sections, we made extensive use of recall/ queries per second plots, where each configuration of each algorithm results in a single point, namely the average recall and the inverse of the average query time. As we shall see in this section, concentrating on averages can hide interesting information in the context of k-NN queries. In fact, not all queries are equally difficult to answer. Consider the plots in Fig. 10, which report the performance of the five algorithms on the GLOVE-2M dataset, with medium and diverse difficulty queries selected according to LID. The top 2×5 plots report the recall versus the number of queries per second for middle (top) and diverse (bottom) query sets, and black dots correspond to the averages. Additionally, for each configuration, we report the distribution of the recall scores: the baseline of each recall curve is positioned at the corresponding queries per second performance. Similarly, the bottom plots report on the inverse of the individual query times (the average of these is the QPS in the left plot) against the average recall. In both plots, the best performance is achieved towards the top-right corner.

Plotting the distributions, instead of just reporting the averages, uncovers some interesting behavior that might otherwise go unnoticed, in particular with respect to the recall. The average

[6] We note that IVF counts the initial comparisons to find the closest centroids as distance computations, whereas Annoy did not count the inner product computations during tree traversal.

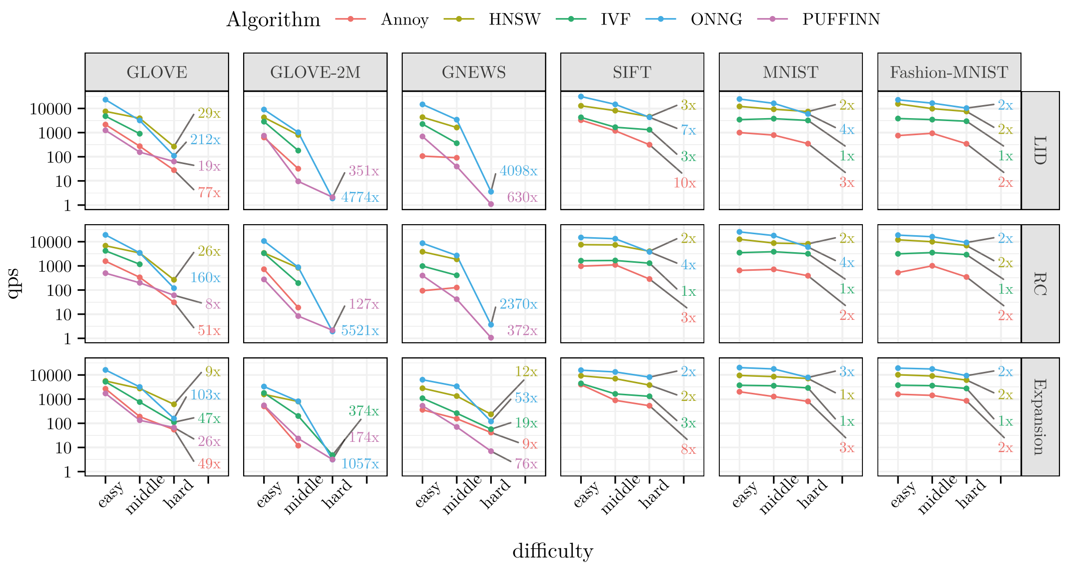

Fig. 7. Change of performance of the fastest configuration achieving at least 0.9 recall as the difficulty of the dataset changes. The colored labels report the slowdown factor of the *hard* queryset compared with the *easy* one.

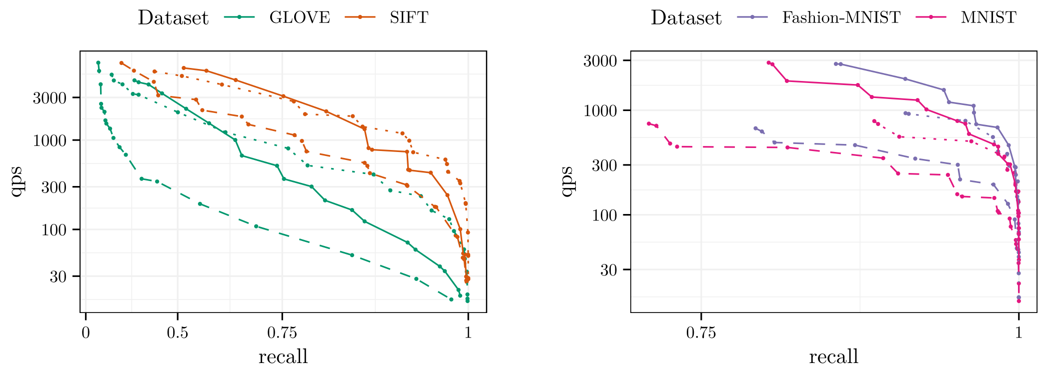

Fig. 8. Recall-QPS (1/s) tradeoff – up and to the right is better – for Annoy on GLOVE, SIFT, Fashion-MNIST, and MNIST with queries selected according to LID. Dashed lines are *hard* query sets, solid lines are *diverse* query sets, dotted lines are *middle* query sets.

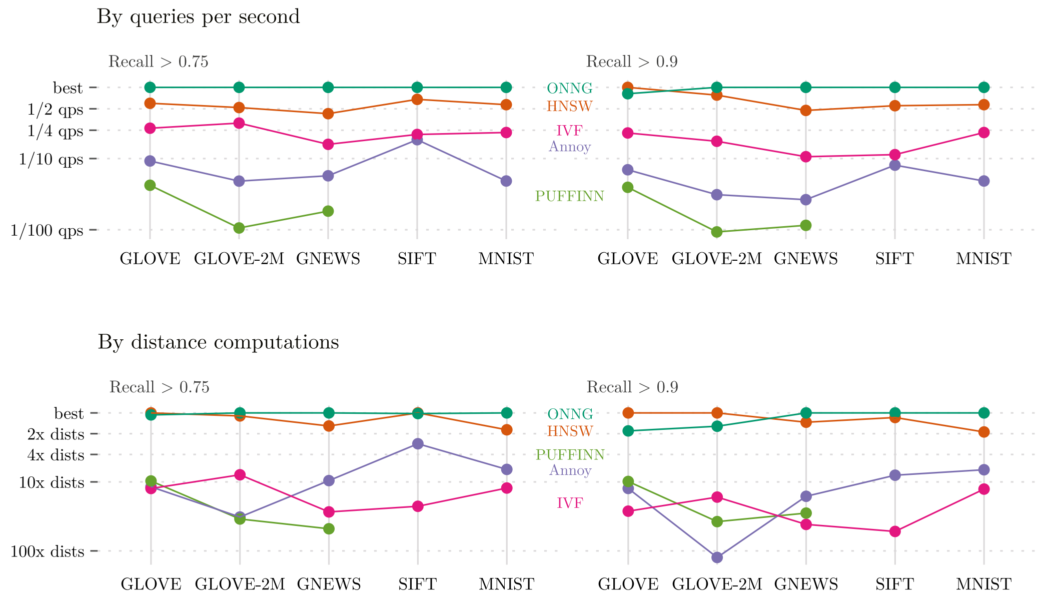

Fig. 9. Ranking of algorithm on five different datasets, according to recall ≥ 0.75 and ≥ 0.9, and according to two different performance measures: number of queries per second (top) and number of distance computations (bottom). Both plots report the ratio with the best performing algorithm on each dataset, higher is better. Note that the scale is logarithmic.

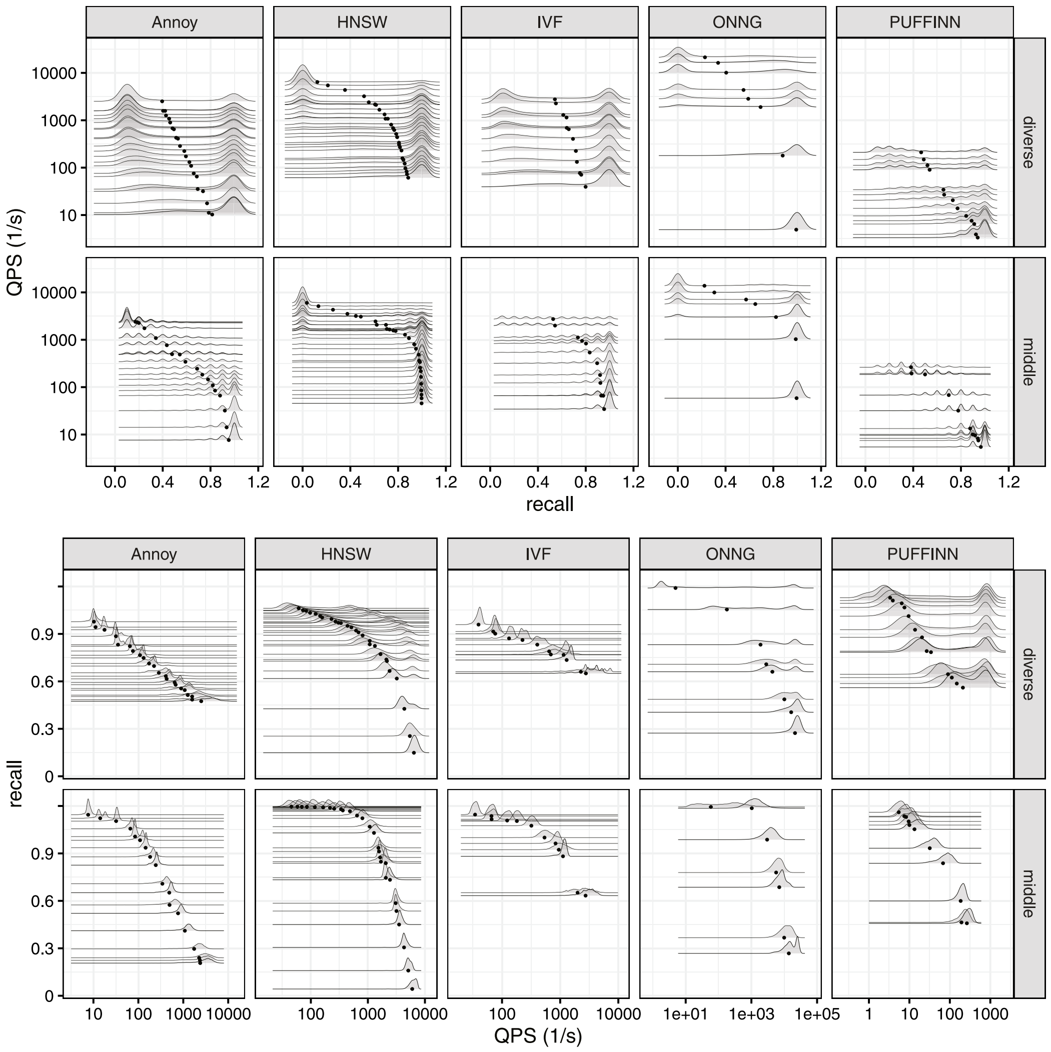

Fig. 10. Distribution of performance for queries on the GLOVE-2M (medium difficulty) dataset. Looking just at the average performance can hide interesting behavior.

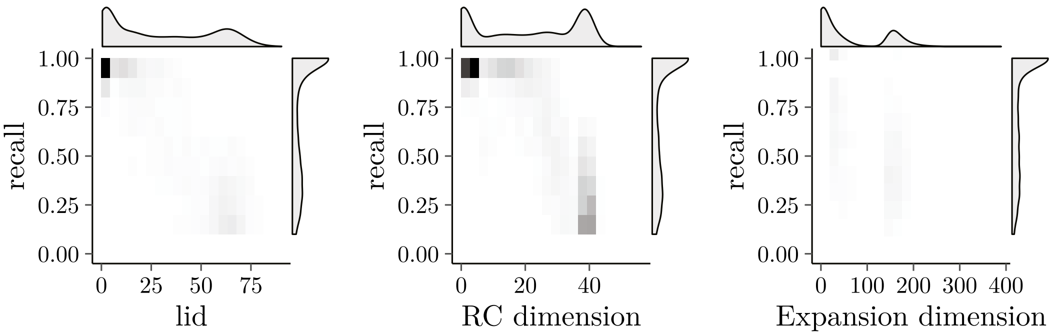

Fig. 11. Distribution of Recall vs. LID, RC dimension, and Expansion dimension plot on the GLOVE-2M dataset, using `Annoy`. Intensity reflects number of queries that achieve a combination of recall vs. LID (or RC or Expansion).

recall gradually shifts towards the right as the effect of more and more queries achieving good recalls. Perhaps surprisingly, for graph-based algorithms this shift is very sudden: most queries go from having recall 0 to having recall 1, taking no intermediate values, even for the query set that have very similar LID values. Taking the average recall as a performance metric is convenient in that it is a single number to compare algorithms with. However, the same average recall can be attained with very different distributions: looking at such distributions can provide more insight.

For the bottom plots and the middle query set, we observe that individual query times of all the algorithms are well concentrated around their mean. For the diverse dataset, algorithms might be able to adapt to the query difficulty. We observe that this is not true for `Annoy` and `IVF`. Both of them have a single peak in their query time, which means that they spend about the same time per query. On the other hand, `PUFFINN`, `HNSW`, and `ONNG` have two peaks in their performance distribution when they approach high recall. This means that they adapt to the presence of easy queries (where both `ONNG` and `PUFFINN` report with the same

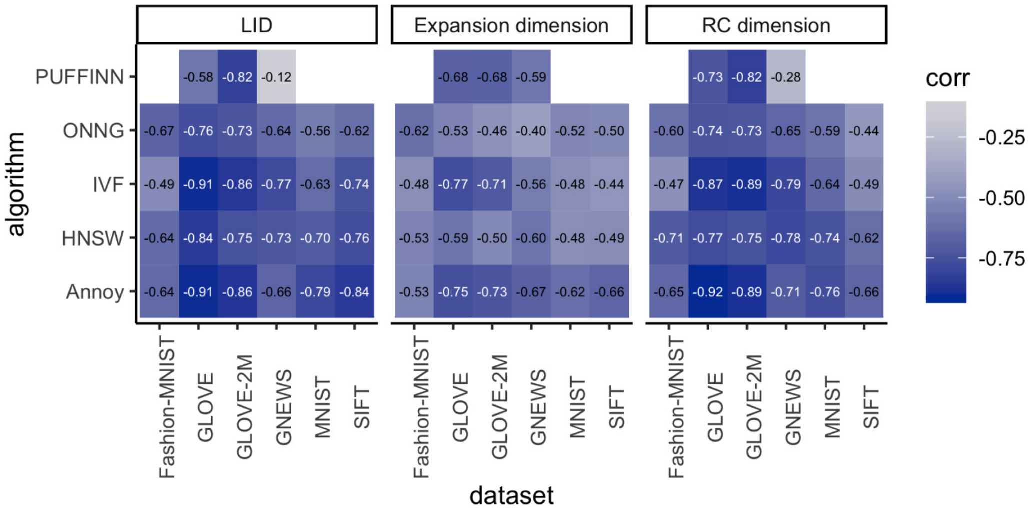

Fig. 12. Correlation between recall and dimensionality measure for dataset/algorithm pairs, for each type of dimensionality measure on workloads of *diverse* difficulty. For each combination of algorithm and dataset, we select the parameter configuration yielding recall at least 0.9 with the highest queries per second. If no parameter configuration is accurate enough, we report on the one achieving the highest recall. Note that some algorithms (Annoy, IVF) are more sensitive than others to the dimensionality of the queries.

Table 3
Number of algorithm/dataset pairs in which a dimensionality measure correlates best with the recall, grouped according to the difficulty of the workload. The top half of the table reports the cases in which the difference between the strongest and second strongest correlation is statistically significant [34], with Bonferroni correction for an overall significance level of 0.01. The bottom half of the table reports the cases in which the difference was not statistically significant.

	easy	*middle*	*diverse*	*hard*
LID	9	13	7	14
RC dim.	15	11	6	5
Expansion dim.	0	0	1	0
LID/RC dim.	3	3	11	7
LID/Exp. dim.	0	0	1	0
RC dim./Exp. dim.	0	0	1	1

performance, and HNSW becomes slower for higher recall). It is surprising to see that all adaptive algorithms have two peaks, while the diverse query set is a mix of three different difficulties.[7]

Fig. 11 gives another distributional view on the achieved result quality. The plots show a run of Annoy on the GLOVE-2M dataset with diverse queries. On the top margin we see the distribution of estimated LID values (left plot), RC values (middle plot), and Expansion values (right plot) for the diverse query set, on the right margin we see the distribution of recall values achieved by the implementation. Each of the queries corresponds to a single data point in the recall/LID plot and data points are summarized through squares, where the color intensity of a square indicates the number of data points falling into this region. The plots show that the higher the LID of a query, there is a clear tendency for the query to achieve lower recall. Expansion and RC, instead, are less predictive in this setting: we can still observe that high Expansion (i.e. difficult) queries have low recall, but the relationship is less marked.

[7] We remark that different nearest neighbor search approaches allow for different approaches to adapt to the difficulty of the query. LSH-based PUFFINN [17] uses a probabilistic stopping criteria that makes sure that the top-k candidates found so far are in fact the true nearest neighbors with a user-specified success probability. For the graph-based approach ONNG [15], one round in the algorithm consists of inspecting the neighbors of a node and adding them to a list of candidates. The algorithm returns that current top-k as soon as all candidates are at distance at least $(1+\varepsilon)r$ from the query, where r is the distance of the current kth nearest neighbor and ε is a user parameter. While this lacks a theoretical guarantee, it is easy to imagine that the search procedure adapts to the local structure.

To further investigate the relationship between the dimensionality measures and the recall, we compute the correlation between each measure and the recall, reporting it in Fig. 12. We observe that, as expected, all three measures are negatively correlated with the recall (i.e. the higher the measure, the harder it is to answer the query accurately).

In Table 3 we report a more comprehensive overview of the correlation between each dimensionality measure and the recall. For each dimensionality measure, we report in the top half of the table the number of combinations of algorithm and dataset in which its correlation with the recall was the strongest, with statistical significance. In the bottom half, we report the other cases, where the difference in correlation of the first two measures with the recall was not statistically significant. We observe the following: for *easy* workloads the RC dimension is the most correlated to the recall in the majority of cases, whereas for *hard* workloads the LID correlates better. For *diverse* workloads there is no clear winner: about half of the times there is no significant difference between LID and RC dimension, and the rest are almost equally divided between LID and RC dimension. As for the Expansion measure, we note that it correlates with the recall significantly better than the other two measures only in one case.

Therefore, if we have to pick queries according to a single local dimensionality measure, the LID is the best predictor for the difficulty in most workloads, with the exception of *easy* workloads, for which the RC dimension is a better predictor. Obviously, our observation that no single dimensionality measure is a perfect predictor for the difficulty of queries still holds.

6. Summary

In this paper we studied the influence of LID, RC, and Expansion to the performance of nearest neighbor search algorithms. We showed that all three measures allow to choose query sets of a wide range of difficulty from a given dataset. We also showed how different LID, RC, and Expansion distributions influence the running time performance of the algorithms. In this respect, we found that LID is a good overall predictor of performance only outperformed by RC dimension on easy workloads. In any case, we could not conclude that any of the three scores alone can predict running time differences well. In particular, SIFT is usually easier than GLOVE for the algorithms: while GLOVE's LID distribution would predict the opposite, the RC distribution correctly

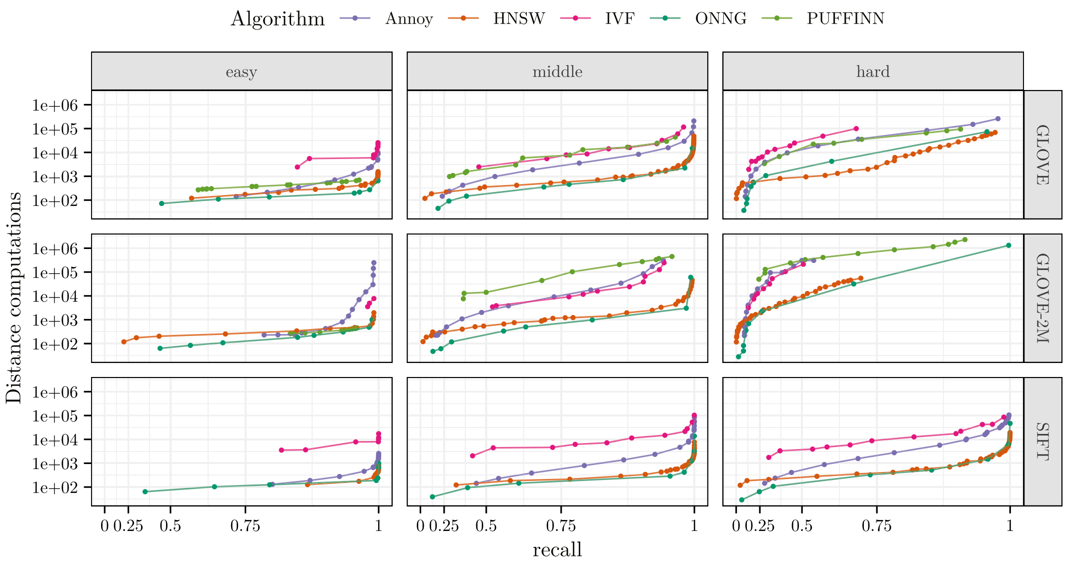

Fig. 13. Recall-Distance computations tradeoff – down and to the right is better – for queries selected according to their LID. Three datasets are considered here: GLOVE, GLOVE-2M and SIFT. The scale is logarithmic on the y axis and exponential on the x axis, to take into account the scale of the data.

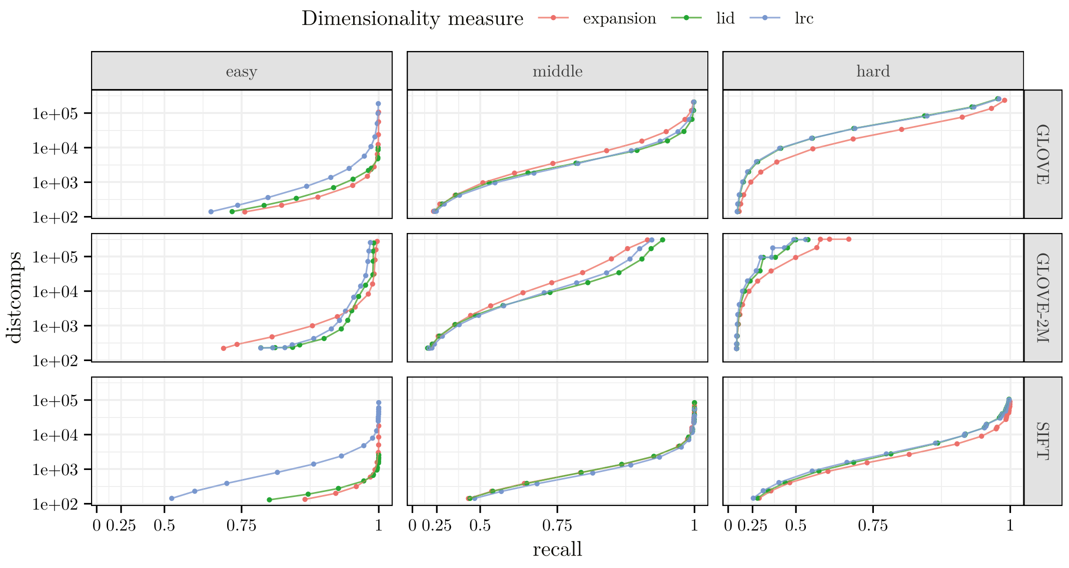

Fig. 14. Recall-Distance computations tradeoff – down and to the right is better – for queries solved with Annoy. Three datasets are considered here: GLOVE, GLOVE-2M and SIFT. The scale is logarithmic on the y axis and exponential on the x axis, to take into account the scale of the data.

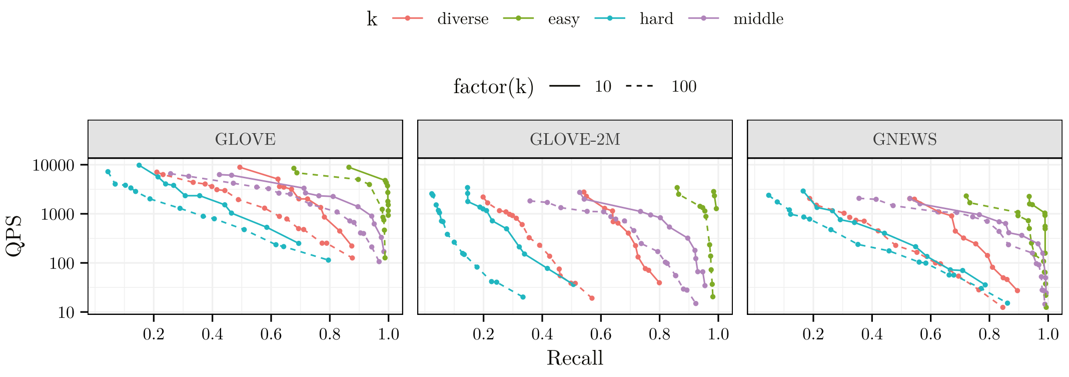

Fig. 15. Recall-QPS tradeoff for queries on three datasets, with queries of difficulty selected according to the LID measure, querying for the $k = 10$ and $k = 100$ nearest neighbors.

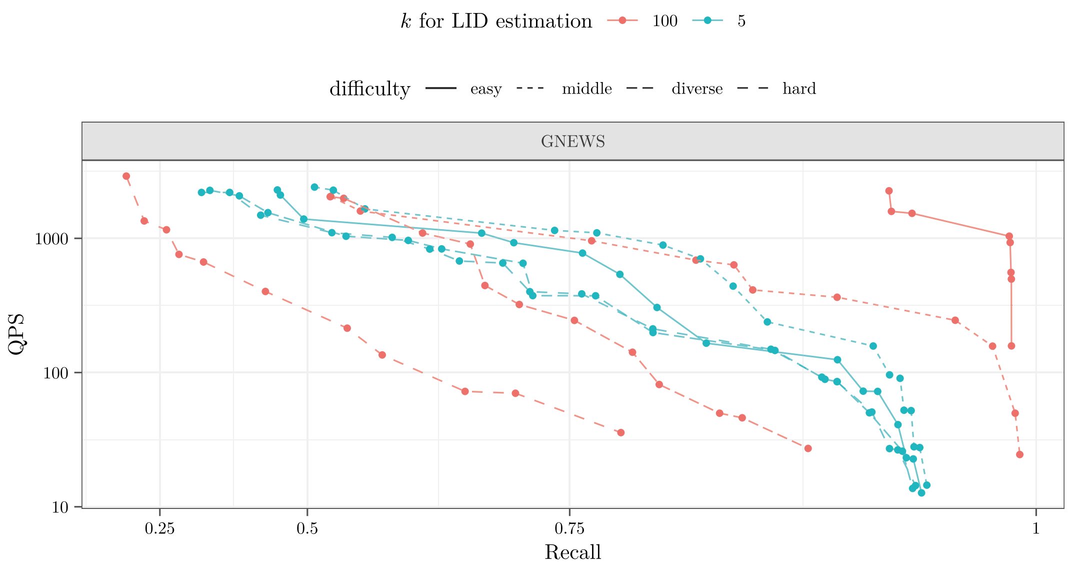

Fig. 16. Recall-QPS tradeoff for queries drawn according to LID_{100} (red curves) and LID_5 (blue curves) on GNEWS with IVF. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

predicts this relationship between the datasets. However, the RC distribution does not predict differences correctly, either.

With regard to challenging query workloads, we described a way to choose diverse query sets. They have the property that for most implementations it is easy to perform well for most of the query points, but they contain many more easy and difficult queries than query workloads chosen randomly from the dataset. We believe this is a very interesting benchmarking workload for approaches that try to adapt to the difficulty of an individual query.

We introduced novel visualization techniques to show the uncertainty within the answer to a set of queries, which made it possible to show a clear difference between the graph-based algorithms and the other approaches. Furthermore, these visualizations allow to see whether a particular algorithm is able to adapt to the difficulty of the queries.

We hope that this study initiates the search for more diverse datasets, or for theoretical reasoning why certain algorithmic principles are generally better suited for nearest neighbor search. On a more practical side, Casanova et al. showed in [35] how dimensionality testing can be used to speed up reverse k-NN queries. We would be interested in seeing whether the LID can be used at other places in the design of NN algorithms to guide the search process or the parameter selection.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the reviewers of the conference version of this paper, who helped to improve the presentation of the paper. We are furthermore thankful to the reviewers of the journal submission. The research leading to these results has received funding from the European Research Council under the European Union's 7th Framework Programme (FP7/2007-2013)/ERC grant agreement no. 614331, and from the Free University of Bozen-Bolzano under project SSJoin and the Open Access Publishing Fund.

Appendix A. Additional experiments

A.1. How well is running time reflected in distance computations

Figs. 13 and 14 present the same setup as in Section 5.2, but this time relating recall to the number of distance computations required to achieve that recall. This cost measure is more robust to implementation details and gives a more general view on how well an approach is able to efficiently index the data set.

Let us consider Fig. 13. For the recall vs. distance computations trade-off, we aim for all curves to be down and to the right, which reflects high recall with a small number of distance computations. In general, the trend observed in the running time study continues for distance computations: the easy, middle, and hard query sets are progressively more difficult to answer. Graph-based approaches compute considerably fewer distances, and there is little difference in these two approaches. With regard to the other approaches, Annoy computes fewest distances, but turns out to be the slowest implementation on most of the data and query sets combinations.

Fig. 14 shows the influence of the three different dimensionality measures for Annoy. First, we notice that there is remarkably little difference between the three different dimensionality measures in terms of distance computations, in particular for SIFT. For the difficult query set, we see that Expansion provides the easier-to-index queries, whereas RC provides considerably more difficult queries than the two others considering the easy queryset. LID provides the best of both worlds.

A.2. Experiments with $k = 100$

Fig. 15 reports the performance, in terms of recall and queries per second, of the IVF algorithm on three datasets for workloads of different difficulty, selected according to the LID measure. In all cases, we can see that the relationship between the performance attained by the algorithm on workloads of varying difficulty is the same for both choices of k: *easy* workloads are indeed the easiest ones while *hard* workloads are the most difficult, the *diverse* workload is more difficult than the middle workload. As expected, we can observe that answering 100-NN queries takes more time than answering 10-NN queries for any given recall level.

A.3. Experiments with queries drawn according to LID_{100} and LID_5

In Fig. 16 we report the performance tradeoff curves for IVF on GNEWS for queries drawn according to LID estimated with different parameterizations, in particular LID_{100} and LID_5. Note how workloads defined in terms of LID_{100} are very well separated on the Recall-QPS plane, whereas LID_5 induces query workloads whose performance profile is very similar. Even more, with LID_5 the *easy* workload is more difficult than the *middle* workload.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.is.2021.101807.

References

[1] R. Spring, A. Shrivastava, Scalable and sustainable deep learning via randomized hashing, in: KDD'17, 2017, pp. 445–454, http://dx.doi.org/10.1145/3097983.3098035.
[2] J. Pennington, R. Socher, C.D. Manning, GloVe: Global vectors for word representation, in: Empirical Methods in Natural Language Processing, EMNLP, 2014, pp. 1532–1543.
[3] E. Chávez, G. Navarro, R. Baeza-Yates, J.L. Marroquín, Searching in metric spaces, ACM Comput. Surv. 33 (3) (2001) 273–321, http://dx.doi.org/10.1145/502807.502808.
[4] J. Alman, R. Williams, Probabilistic polynomials and hamming nearest neighbors, in: FOCS'15, pp. 136–150.
[5] M.E. Houle, Dimensionality, discriminability, density and distance distributions, in: Data Mining Workshops, ICDMW, IEEE, 2013, pp. 468–473.
[6] W.B. Johnson, J. Lindenstrauss, G. Schechtman, Extensions of Lipschitz maps into Banach spaces, Israel J. Math. 54 (2) (1986) 129–138.
[7] I. Jolliffe, Principal Component Analysis, Springer, 2011.
[8] T.D. Ahle, M. Aumüller, R. Pagh, Parameter-free locality sensitive hashing for spherical range reporting, in: SODA, SIAM, 2017, pp. 239–256.
[9] J. He, S. Kumar, S. Chang, On the difficulty of nearest neighbor search, in: ICML, icml.cc/Omnipress, 2012.
[10] M.E. Houle, E. Schubert, A. Zimek, On the correlation between local intrinsic dimensionality and outlierness, in: SISAP'18, 2018, pp. 177–191, http://dx.doi.org/10.1007/978-3-030-02224-2_14.
[11] H. Kriegel, E. Schubert, A. Zimek, The (black) art of runtime evaluation: Are we comparing algorithms or implementations? Knowl. Inf. Syst. 52 (2) (2017) 341–378.
[12] Y.A. Malkov, D.A. Yashunin, Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs, 2016, ArXiv e-prints.
[13] J. Johnson, M. Douze, H. Jégou, Billion-scale similarity search with GPUs, 2017, CoRR abs/1702.08734.
[14] E. Bernhardsson, Annoy, https://github.com/spotify/annoy.
[15] M. Iwasaki, D. Miyazaki, Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data, 2018, ArXiv e-prints.
[16] M. Aumüller, E. Bernhardsson, A.J. Faithfull, ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms, Inf. Syst. 87 (2020) See https://arxiv.org/abs/1807.05614 for an open access version.
[17] M. Aumüller, T. Christiani, R. Pagh, M. Vesterli, PUFFINN: parameterless and universally fast finding of nearest neighbors, in: ESA, in: LIPIcs, vol. 144, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 10:1–10:16.
[18] K. Smith-Miles, D. Baatar, B. Wreford, R. Lewis, Towards objective measures of algorithm performance across instance space, Comput. Oper. Res. 45 (2014) 12–24.
[19] R.R. Curtin, J.R. Cline, N.P. Slagle, W.B. March, P. Ram, N.A. Mehta, A.G. Gray, MLPACK: A scalable C++ machine learning library, J. Mach. Learn. Res. 14 (2013) 801–805.
[20] M. Edel, A. Soni, R.R. Curtin, An automatic benchmarking system, in: NIPS 2014 Workshop on Software Engineering for Machine Learning, 2014.
[21] W. Li, Y. Zhang, Y. Sun, W. Wang, W. Zhang, X. Lin, Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement (v1.0), 2016, CoRR abs/1610.02455.
[22] K.S. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful? in: ICDT, in: Lecture Notes in Computer Science, vol. 1540, Springer, 1999, pp. 217–235.
[23] D. François, V. Wertz, M. Verleysen, The concentration of fractional distances, IEEE Trans. Knowl. Data Eng. 19 (7) (2007) 873–886.
[24] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, O. Chum, Revisiting oxford and Paris: Large-scale image retrieval benchmarking, in: CVPR, 2018.
[25] M. Aumüller, M. Ceccarello, The role of local intrinsic dimensionality in benchmarking nearest neighbor search, in: SISAP, in: Lecture Notes in Computer Science, vol. 11807, Springer, 2019, pp. 113–127.
[26] L. Amsaleg, O. Chelly, T. Furon, S. Girard, M.E. Houle, K.I. Kawarabayashi, M. Nett, Estimating local intrinsic dimensionality, in: KDD'15, ACM, 2015, pp. 29–38.
[27] E. Levina, P.J. Bickel, Maximum likelihood estimation of intrinsic dimension, in: NIPS'15, 2005, pp. 777–784.
[28] L. Amsaleg, O. Chelly, T. Furon, S. Girard, M.E. Houle, K. Kawarabayashi, M. Nett, Extreme-value-theoretic estimation of local intrinsic dimensionality, Data Min. Knowl. Discov. 32 (6) (2018) 1768–1805.
[29] L. Amsaleg, O. Chelly, M.E. Houle, K.i. Kawarabayashi, M. Radovanović, W. Treeratanajaru, Intrinsic dimensionality estimation within tight localities, in: Proceedings of the 2019 SIAM International Conference on Data Mining, SIAM, 2019, pp. 181–189.
[30] M.E. Houle, Local intrinsic dimensionality I: an extreme-value-theoretic foundation for similarity applications, in: SISAP, in: Lecture Notes in Computer Science, vol. 10609, Springer, 2017, pp. 64–79.
[31] D.R. Karger, M. Ruhl, Finding nearest neighbors in growth-restricted metrics, in: STOC, ACM, 2002, pp. 741–750.
[32] M.E. Houle, H. Kashima, M. Nett, Generalized expansion dimension, in: ICDM Workshops, IEEE Computer Society, 2012, pp. 587–594.
[33] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017, CoRR abs/1708.07747.
[34] A.G. Asuero, A. Sayago, A. Gonzalez, The correlation coefficient: An overview, Crit. Rev. Anal. Chem. 36 (1) (2006) 41–59.
[35] G. Casanova, E. Englmeier, M.E. Houle, P. Kröger, M. Nett, E. Schubert, A. Zimek, Dimensional testing for reverse k-nearest neighbor search, PVLDB 10 (7) (2017) 769–780.